



UNIVERSIDAD TÉCNICA DE AMBATO

**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E
INDUSTRIAL
CARRERA DE TELECOMUNICACIONES**

Tema:

**SISTEMA PARA DETECCIÓN DE AUTOMÓVILES ROBADOS
EMPLEANDO VISIÓN ARTIFICIAL EN VEHÍCULO AÉREO NO
TRIPULADO (UAV)**

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la
obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Comunicaciones

LÍNEA DE INVESTIGACIÓN: Tecnología de la información y Sistemas de
control

AUTORES: Jose Luis Chanahuano Azogue

Danny Gabriel Santo Taipicaña

TUTOR: Ing. Franklin Wilfrido Salazar Logroño, Mg.

Ambato - Ecuador

febrero – 2025

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA PARA DETECCIÓN DE AUTOMÓVILES ROBADOS EMPLEANDO VISIÓN ARTIFICIAL EN VEHÍCULO AÉREO NO TRIPULADO (UAV), desarrollado bajo la modalidad Proyecto de Investigación por los señores Jose Luis Chanahuano Azogue y Danny Gabriel Santo Taipicaña, estudiantes de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que los estudiantes han sido tutorados durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, febrero 2025.



Ing. Franklin Wilfrido Salazar Logroño, Mg.
TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA PARA DETECCIÓN DE AUTOMÓVILES ROBADOS EMPLEANDO VISIÓN ARTIFICIAL EN VEHÍCULO AÉREO NO TRIPULADO (UAV) es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, febrero 2025.



Jose Luis Chanahuano Azogue

C.C. 1851014850

AUTOR



Danny Gabriel Santo Taipicaña

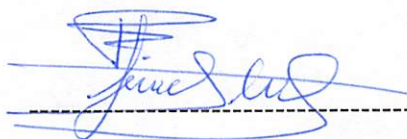
C.C. 0550205322

AUTOR

DERECHOS DE AUTOR

Autorizamos a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedemos todos los derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, febrero 2025.



Jose Luis Chanahuano Azogue

C.C. 1851014850

AUTOR



Danny Gabriel Santo Taipicaña

C.C. 0550205322

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por los señores Jose Luis Chanahuano Azogue y Danny Gabriel Santo Taipicaña, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA PARA DETECCIÓN DE AUTOMÓVILES ROBADOS EMPLEANDO VISIÓN ARTIFICIAL EN VEHÍCULO AÉREO NO TRIPULADO (UAV), nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, febrero 2025.



Firmado electrónicamente por:
**FRANKLIN OSWALDO
MAYORGA MAYORGA**

Ing. Franklin Oswaldo Mayorga Mayorga, Mg.

PRESIDENTE DEL TRIBUNAL



Firmado electrónicamente por:
**VÍCTOR SANTIAGO
MANZANO VILLAFUERTE**

Ing. Víctor Manzano Villafuerte Mg.

PROFESOR CALIFICADOR



Firmado electrónicamente por:
**CARLOS DIEGO GORDON
GALLEGOS**

Ing. Carlos Gordón Gallegos PhD.

PROFESOR CALIFICADOR

DEDICATORIA

Dedico el presente proyecto de investigación a mi padre por ser el pilar fundamental de apoyo en mi camino profesional, por enseñarme los buenos principios y valores para ser una buena persona.

Jose Luis Chanahuano

AGRADECIMIENTO

Agradezco, en primer lugar, a Dios por guiarme y darme fuerzas en cada paso del camino. A mi familia, por estar siempre a mi lado, en las buenas y en las malas, apoyándome incondicionalmente y ayudándome a cumplir cada una de mis metas. También expreso mi gratitud a las personas cercanas que, con su constante motivación, han sido un pilar fundamental tanto en mi vida profesional como personal.

A mi tutor de tesis Ing. Franklin Salazar y a todos los docentes de la carrera de Telecomunicaciones, por ser unos excelentes guías y compartir sus conocimientos para aprender cada día algo nuevo.

A todos mis amigos que de alguna manera han contribuido en este camino, gracias por ser parte de mi vida estudiantil y por inspirarme a alcanzar mis metas.

Al equipo de ingenieros que conforman los Talleres Tecnológicos de la FISEI, por su respaldo y cooperación, proporcionándome el lugar y los recursos requeridos para realizar la investigación.

A ti, mi ángel eterno, que desde el cielo guías mis pasos y llenas mi vida de fortaleza.

Jose Luis Chanahuano

DEDICATORIA

A mis padres Milton y Angelita, por su esfuerzo, sacrificio y apoyo incondicional, convirtiéndose en el modelo a seguir para mi formación personal y profesional. Gracias por cada palabra de aliento, cada abrazo reconfortarle y por creer en mí en todo momento. Cada logro mío siempre será suyo.

A mis hermanas, Eli, Lissita y Camilita, agradecido eternamente por sus consejos, risas, anécdotas y compartir conmigo cada paso que doy en el camino de la vida.

¡Siempre los llevaré en mi corazón!

Danny Gabriel Santo

AGRADECIMIENTO

A Cristiano Ronaldo, cuya magia en el campo de juego fascina a millones, e inspira vidas. Sus goles, su pasión desbordante y su capacidad para brillar en los momentos más difíciles me enseñaron que el talento se convierte en grandeza cuando se mezcla con determinación. Gracias por demostrar que el fútbol, como la vida, es un lugar donde la perseverancia siempre encuentra su recompensa.

A la Universidad Técnica de Ambato por brindarme la oportunidad de formarme como Ingeniero en Telecomunicaciones en la FISEI.

A mis profesores de la carrera que más allá de sus conocimientos en las materias impartidas, compartieron valiosas lecciones de vida.

A Mary y mis amigos de la universidad, por compartir conmigo este camino lleno de aprendizajes, risas y momentos inolvidables. Cada conversación, cada proyecto y cada apoyo marcaron mi vida de manera especial.

A mi tutor de tesis Ing. Franklin Salazar y al equipo de ingenieros que conforman los Talleres Tecnológicos de la FISEI, por su respaldo y cooperación, proporcionándome el lugar y los recursos requeridos para realizar la investigación.

¡Su respaldo incondicional fue mi mayor fortaleza!

Danny Gabriel Santo

ÍNDICE GENERAL DE CONTENIDOS

PORTADA.....	i
APROBACIÓN DEL TUTOR.....	ii
AUTORÍA.....	iii
DERECHOS DE AUTOR.....	iv
APROBACIÓN DEL TRIBUNAL DE GRADO	v
DEDICATORIA.....	vi
AGRADECIMIENTO	vii
DEDICATORIA.....	viii
AGRADECIMIENTO	ix
ÍNDICE GENERAL DE CONTENIDOS.....	x
ÍNDICE DE TABLAS	xv
ÍNDICE DE FIGURAS	xix
ÍNDICE DE ANEXOS.....	xxviii
RESUMEN EJECUTIVO	xxx
ABSTRACT.....	xxxi
CAPÍTULO I. MARCO TEÓRICO	1
1.1 Tema de investigación	1
1.1.1 Planteamiento del problema.....	1
1.2 Antecedentes investigativos.....	2

1.3 Fundamentación teórica	5
1.3.1 Robo de vehículos.....	5
1.3.2 Indicadores de robo de vehículos.....	5
1.3.3 Gestión de seguridad vehicular.....	6
1.3.4 Procedimiento tras el robo de un vehículo en Ecuador	6
1.3.5 Procedimiento operativo para recuperar vehículos robados en Ecuador	7
1.3.6 Tecnologías y herramientas para localización de vehículos.....	8
1.3.7 Identificadores vehiculares únicos.....	11
1.3.8 Sistema de reconocimiento de vehículos robados con visión artificial	13
1.3.9 Permiso para volar UAVs en Ecuador	14
1.3.10 Reglamento de radiocomunicaciones	14
1.3.11 Vehículo aéreo no tripulado (UAV)	15
1.3.12 Software de control de vuelo	19
1.3.13 Emisor y receptor de vuelo	20
1.3.14 GPS para dron FPV.....	21
1.3.15 Fuente de alimentación para dron FPV.....	22
1.3.16 Cámaras RGB	22
1.3.17 Sistemas FPV	23
1.3.18 Inteligencia Artificial.....	26
1.3.19 Deep Learning.....	33

1.3.20 Redes Neuronales Convolucionales.....	34
1.3.21 Red Convolucional YOLO	35
1.3.22 YOLOv8	37
1.3.23 Seguimiento de objetos.....	39
1.3.24 Reconocimiento óptico de caracteres (OCR).....	40
1.3.25 Plataforma para entrenamiento de red neuronal	41
1.3.26 Plataformas de servicios en la nube	42
1.3.27 Gestión con servidores.....	43
1.3.28 Equipo computacional para programación	46
1.4 Objetivos.....	47
1.4.1 Objetivo general.....	47
1.4.2 Objetivos específicos	47
CAPÍTULO II. METODOLOGÍA.....	48
2.1 Materiales.....	48
2.2 Métodos	52
2.2.1 Modalidad de la investigación	52
2.2.2 Recolección de información	52
2.2.3 Procesamiento y análisis de datos.....	53
CAPÍTULO III. RESULTADOS Y DISCUSIÓN	54
3.1 Desarrollo de la propuesta	54

3.1.1 Esquema general del sistema	54
3.1.2 Herramientas utilizadas para la recuperación de vehículos	56
3.1.3 Selección de tecnologías.....	57
3.1.4 Diseño de conexiones electrónicas del sistema de transmisión y captura de video.....	62
3.1.5 Diseño de conexiones electrónicas del módulo GPS.....	63
3.1.6 Implementación del sistema de transmisión, antena y captura de video en el UAV.....	64
3.1.7 Implementación del módulo GPS en el UAV	65
3.1.8 Peso del UAV	66
3.1.9 Consumo de energía y tiempo de vuelo.....	66
3.1.10 Simulación de un modelo cinemático en el software Matlab para ejecución de planes de vuelo.....	68
3.1.11 Configuración del UAV.....	76
3.1.12 Diagrama del sistema.....	95
3.1.13 Entrenamiento de la red neuronal para la detección de placas	97
3.1.14 Codificación de la visualización y detección de placas en automóviles en el ordenador	110
3.1.15 Creación y configuración de la máquina virtual en Azure	139
3.1.16 Creación y configuración de la Base de datos en Azure.....	141
3.1.17 Creación y configuración de la página web en Azure	156
3.1.18 Funcionamiento del registro y consulta en la página web	168

3.1.19 Planteamiento de espacio para desarrollo de pruebas.....	171
3.1.20 Tiempo de vuelo en parqueaderos	173
3.1.21 Detección de placas vehiculares con visión artificial	174
3.1.22 Extracción de caracteres con OCR	179
3.1.23 Evaluación del Sistema de Detección y Reconocimiento OCR	191
3.1.24 Tiempo de procesamiento de información.....	191
3.1.25 Prueba del Sistema de Detección mediante el Vuelo Autónomo del UAV ...	194
3.1.26 Presupuesto	197
CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES	200
4.1 Conclusiones.....	200
4.2 Recomendaciones	201
REFERENCIAS BIBLIOGRÁFICAS.....	203
ANEXOS.....	212

ÍNDICE DE TABLAS

Tabla 1. Indicadores de robo de vehículos en Ecuador.....	6
Tabla 2. Tecnologías de rastreo de vehículos	9
Tabla 3. Sistemas de cámaras de seguridad	11
Tabla 4. Iniciales de placas por provincia.....	13
Tabla 5. Tipos de placas y ejemplos	13
Tabla 6. Bandas de frecuencia asignadas en Ecuador.....	15
Tabla 7. Características técnicas de los UAVs disponibles FISEI.....	16
Tabla 8. Características software controlador de vuelo	19
Tabla 9. Características de los radios controles	20
Tabla 10. Características de los GPS	21
Tabla 11. Baterías LiPo.....	22
Tabla 12. Características de cámaras FPV	24
Tabla 13. Características de transmisores de video	25
Tabla 14. Características de receptores de video	26
Tabla 15. Lenguajes de programación utilizados en IA.....	27
Tabla 16. Comparación de entornos de desarrollo integrado.....	28
Tabla 17. Comparación de Redes Neuronales Convolucionales	35
Tabla 18. Comparación de versiones de YOLO	37
Tabla 19. Modelos de YOLOv8.....	38
Tabla 20. Algoritmos para seguimiento de objetos.....	39

Tabla 21. APIs para reconocimiento óptico de caracteres	41
Tabla 22. Comparación de plataformas basadas en la nube.....	41
Tabla 23. Características de plataformas con servicios en la nube	42
Tabla 24. Características de los servidores web más usados	44
Tabla 25. Comparación de base de datos	45
Tabla 26. Características de ordenadores portátiles	46
Tabla 27. Materiales y dispositivos.....	48
Tabla 28. Softwares y servicios de aplicación	50
Tabla 29. Criterios de recolección de información	53
Tabla 30. Características de la batería LiPo.....	67
Tabla 31. Movimiento del UAV	89
Tabla 32. Componentes del Controlador PID en UAVs	90
Tabla 33. Comandos fundamentales para las rutas en ArduPilot.....	93
Tabla 34. Parámetros de la máquina en Kaggle.....	105
Tabla 35. Funciones de las métricas de modelos entrenados.....	107
Tabla 36. Análisis de resultados de dos redes neuronales entrenadas con diferentes épocas.....	109
Tabla 37. Bibliotecas requeridas para el sistema	111
Tabla 38. Características importantes de la máquina virtual en Azure.....	141
Tabla 39. Bibliotecas requeridas para envío de información a los usuarios	147
Tabla 40. Distancias horizontales para captura de video	172

Tabla 41. Parqueaderos analizados	172
Tabla 42. Tiempos de vuelo en parqueaderos utilizando modo manual	173
Tabla 43. Entornos de iluminación	174
Tabla 44. Errores absolutos y porcentual en detección de placas.	178
Tabla 45. Detección de caracteres de placas en parqueadero 1 con nivel de iluminación bajo.....	179
Tabla 46. Detección de caracteres de placas en parqueadero 2 con nivel de iluminación bajo.....	180
Tabla 47. Detección de caracteres de placas en parqueadero 3 con nivel de iluminación bajo.....	182
Tabla 48. Detección de caracteres de placas en parqueadero 1 con nivel de iluminación medio.....	183
Tabla 49. Detección de caracteres de placas en parqueadero 2 con nivel de iluminación medio.....	184
Tabla 50. Detección de caracteres de placas en parqueadero 3 con nivel de iluminación medio.....	185
Tabla 51. Detección de caracteres de placas en parqueadero 1 con nivel de iluminación alto.....	186
Tabla 52. Detección de caracteres de placas en parqueadero 2 con nivel de iluminación alto.....	187
Tabla 53. Detección de caracteres de placas en parqueadero 3 con nivel de iluminación alto.....	188
Tabla 54. Resumen de parámetros en la detección de caracteres de placas vehiculares.....	189
Tabla 55. Errores absolutos y porcentual en detección de placas	190

Tabla 56. Tiempo de envío de información sobre eventos con reconocimiento y coincidencia de placas reportadas	192
Tabla 57. Tiempo de envío de información de eventos sin reconocimiento de placas.....	192
Tabla 58. Tiempos promedio en el procesamiento de información	193
Tabla 59. Ruta en parqueaderos utilizando modo automático	196
Tabla 60. Presupuesto de los materiales del sistema.....	198

ÍNDICE DE FIGURAS

Figura 1. Proceso para denuncia de vehículo robado.....	7
Figura 2. Proceso para recuperar un vehículo robado.....	8
Figura 3. UAV de ala fija.....	17
Figura 4. Cuadricóptero	17
Figura 5. Configuraciones del rotor más comunes.....	18
Figura 6. Emisora Flysky Fs i6 y Receptor asociado.....	20
Figura 7. Fusión de colores primarios; a) aditivo, b) Sustractivo	23
Figura 8. Cámara FPV.....	24
Figura 9. Sistema de transmisión	25
Figura 10. Redes neuronales multicapa	27
Figura 11. Diagrama básico de visión artificial	29
Figura 12. Etapas de un Sistema de Visión Artificial	30
Figura 13. Detección de objetos.....	32
Figura 14. Regiones con características de CNN.....	32
Figura 15. Intersección sobre unión (IoU)	33
Figura 16. Supresión no máxima (NMS)	33
Figura 17. Estructura organizativa del campo de la Inteligencia Artificial	34
Figura 18. Arquitectura YOLO.....	36
Figura 19. Detección de objetos con YOLO	36
Figura 20. Precisión y Eficiencia entre Versiones de YOLO	38

Figura 21. Esquema general del sistema	55
Figura 22. Diagrama de conexión de la cámara FPV	63
Figura 23. Diagrama de conexión del módulo GPS.....	64
Figura 24. Implementación del sistema de transmisión y captura de video.....	65
Figura 25. Implementación del módulo GPS en el UAV.....	65
Figura 26. Peso total del UAV	66
Figura 27. Marco Inercial de UAV	69
Figura 28. Diseño del cuadricóptero en el software FUSION 360	70
Figura 29. Declaración de variables.....	70
Figura 30. Declaración de la función para el control del movimiento de rotación y traslación	71
Figura 31. Declaración de condiciones iniciales y posiciones deseadas.....	72
Figura 32. Configuración del entorno visual para el monitoreo del movimiento del cuadricóptero.....	73
Figura 33. Matriz jacobiana, matriz diagonal K y ley de control.	74
Figura 34. Actualización del ángulo de orientación, cálculo de nuevas posiciones y verificación del punto deseado.....	75
Figura 35. Simulación de la posición del UAV con tres puntos de referencias y ángulos de inclinación de 0 grados, 90 grados y 180 grados	76
Figura 36. Diagrama de flujo del proceso del funcionamiento del UAV	77
Figura 37. Interfaz principal del software Mission Planner de ArduPilot	78
Figura 38. Selección del tipo de frame del UAV	79
Figura 39. Configuración de parámetros iniciales con respecto al UAV.....	80

Figura 40. Calibración del acelerómetro.....	81
Figura 41. Calibración de la brújula del UAV	82
Figura 42. Configuración del firmware utilizando el Software ExpressLRS Configurator	83
Figura 43. Configuración del Binding y subida del firmware en el RX	84
Figura 44. Configuración del Binding y subida del firmware en el TX	85
Figura 45. Calibración del radio control	86
Figura 46. Asignación de canales: Modos de vuelo; Armado y desarmado del dron.....	86
Figura 47. Modos de vuelo. a) Radio control; b) Mission Planner	87
Figura 48. Calibración del ESC	88
Figura 49. Test de movimiento de los motores del UAV	88
Figura 50. Configuración del giro de los motores.....	89
Figura 51. PIDs para el control de respuesta del UAV en los ejes	90
Figura 52. PIDs para la estabilización y potencia de los motores del UAV	91
Figura 53. Configuración de los PIDs y velocidades para el modo estacionario.....	91
Figura 54. Configuración de la velocidad de desplazamiento y el alcance de radio de cada punto	92
Figura 55. Punto HOME o "H"	92
Figura 56. Trayectorias y alturas de vuelo en el software Mission Planner	94
Figura 57. Configuración del OSD en el software Mission Planner.....	95
Figura 58. Diagrama del proceso primera parte.....	96

Figura 59. Diagrama del proceso segunda parte.	97
Figura 60. Creación de dataset de placas vehiculares	99
Figura 61. Dataset de placas vehiculares en Roboflow.....	100
Figura 62. Imágenes capturadas desde UAV.	100
Figura 63. Asignación de etiquetas.	101
Figura 64. Conformación de dataset en Roboflow.....	101
Figura 65. Parámetros para creación de dataset	102
Figura 66. Arquitectura del dataset.	103
Figura 67. Código de descarga desde Roboflow.....	104
Figura 68. Entorno de ejecución de Kaggle	105
Figura 69. Descarga de librerías ultralytics.....	105
Figura 70. Descarga de dataset desde Roboflow	106
Figura 71. Código para el entrenamiento de la red neuronal.	106
Figura 72. Carpeta de resultados del entrenamiento de la red neuronal	107
Figura 73. Resultados del entrenamiento de la red neuronal con YOLOv8m utilizando 60 épocas.....	108
Figura 74. Resultados del entrenamiento de la red neuronal con YOLOv8m utilizando 80 épocas.....	109
Figura 75. Librerías utilizadas en portada.....	112
Figura 76. Código de parámetros de la interfaz.	113
Figura 77. Interfaz gráfica con la portada.	114
Figura 78. Código de funciones para iniciar y cerrar el programa.....	115

Figura 79. Librerías utilizadas para procesamiento de video y detección de automóviles	117
Figura 80. Lectura del modelo yolov8m.pt.	117
Figura 81. Captura de video para procesamiento.	118
Figura 82. Parámetros Sort para seguimiento de automóviles.	118
Figura 83. Creación de directorio para almacenamiento de imágenes.....	118
Figura 84. Bucle principal de procesamiento de video.	119
Figura 85. Coordenadas de detecciones y nivel de confianza.....	120
Figura 86. Actualización de la detección del automóvil.	120
Figura 87. Procesamiento de objetos rastreados	121
Figura 88. Directorio para almacenamiento de capturas de automóviles	122
Figura 89. Función de enlace al archivo filtrado.py.....	122
Figura 90. Reconocimiento de vehículos en el video	123
Figura 91. Librerías utilizadas para detección de placas.....	123
Figura 92. Función para detectar placas en automóviles	124
Figura 93. Directorios para almacenamiento de imágenes	124
Figura 94. Aplicación del modelo para detectar placas	125
Figura 95. Interacción con placa detectada.	125
Figura 96. Placa detectada en vehículo.	126
Figura 97. Segmentación de placa	126
Figura 98. Directorio para almacenamiento de placas segmentadas.....	127

Figura 99. Segmentación de latitud y longitud de la imagen.....	127
Figura 100. Directorios para almacenamiento de datos de ubicación.....	128
Figura 101. Clasificación de vehículos con placa y sin placa.....	129
Figura 102. Directorio de los vehículos con placa y sin placa.....	129
Figura 103. Obtención de API Key en cloudinary	130
Figura 104. Creación URL desde cloudinary en carpeta "PLACAS"	131
Figura 105. Creación URL desde cloudinary en carpeta "NO_PLACAS"	131
Figura 106. Funciones para extracción de caracteres en caso de detectar placa.....	132
Figura 107. Funciones para extracción de caracteres en caso de no detectar placa.....	133
Figura 108. Cuenta de servicio creada en Cloud Vision API.	134
Figura 109. Generación de clave en el servicio de cuenta	135
Figura 110. Archivo JSON para utilizar servicios de Cloud Vision	135
Figura 111. Función de validación de placa y creación de enlace de ubicación.....	136
Figura 112. Función para creación de enlace de ubicación.	137
Figura 113. Almacenamiento en base de datos cuando la placa es detectada.....	138
Figura 114. Almacenamiento en base de datos cuando la placa no es detectada.....	139
Figura 115. Configuración y creación de la máquina virtual en Azure	140
Figura 116. Instalación de componentes comunes para el desarrollo web mediante el (CLI).....	142
Figura 117. Creación de la base de datos mediante phpMyAdmin.....	142
Figura 118. Creación de las tablas	143

Figura 119. Configuración del Trigger en MariaDB	144
Figura 120. Creación de un Bot utilizando BotFather.	145
Figura 121. Grupos de Telegram para los usuarios y administradores	146
Figura 122. Creación de una contraseña de aplicación para el correo electrónico. .	146
Figura 123. Creación de archivos e instalación de bibliotecas en el (CLI) de la máquina virtual.....	149
Figura 124. Inclusión de las bibliotecas para el envío hacia los usuarios.....	150
Figura 125. Configuración de las credenciales iniciales de Telegram y correo electrónico para los usuarios.	150
Figura 126. Gestión del archivo de última ID procesada para los usuarios.....	151
Figura 127. Funciones para el envío de mensaje hacia Telegram y correo electrónico.	151
Figura 128. Ejecución del script envioATelegram.py.	152
Figura 129. Configuración de las credenciales iniciales de Telegram y correo electrónico para los administradores.....	153
Figura 130. Gestión del archivo de última ID procesada para los administradores.....	153
Figura 131. Ejecución del script envioATelegramAdmin.py.	154
Figura 132. Configuración del servicio para la ejecución automática de los scripts.....	155
Figura 133. Diagrama de flujo del funcionamiento de la página web.	157
Figura 134. Creación de los archivos y otorgación de permisos para el desarrollo de la página web.....	158

Figura 135. Codificación de la etiqueta header del archivo de la página principal.....	159
Figura 136. Codificación de la etiqueta footer del archivo de la página principal...	160
Figura 137. Establecimiento de conexión con la base de datos del archivo de registro.....	161
Figura 138. Estructura de control para solicitudes POST, captura de valores y verificación.....	162
Figura 139. Proceso de transacción de información en la base de datos.	163
Figura 140. Establecimiento de conexión con la base de datos del archivo de consulta.	164
Figura 141. Estructura de control para solicitudes POST y captura de valores.....	164
Figura 142. Proceso de consulta y verificación de existencia del vehículo.	165
Figura 143. Página web sección Inicio.	166
Figura 144. Página web sección Reportes.....	167
Figura 145. Página web sección Consultas.....	167
Figura 146. Página web sección Contactos.....	168
Figura 147. Formulario de registro.	169
Figura 148. Registro exitoso.	169
Figura 149. Registro en la base de datos. a) Datos propietario; b) Datos vehículo.....	170
Figura 150. Formulario de consulta.	170
Figura 151. Información proporcionada si el vehículo ha sido localizado previamente.....	171

Figura 152. Detección de placas con el modelo de red neuronal.....	175
Figura 153. Reflectividad de luz solar en placas vehiculares.	176
Figura 154. Canchas de la Universidad Técnica de Ambato Campus Huachi.....	194
Figura 155. Plan de vuelo del UAV ruta 1.....	195
Figura 156. Plan de vuelo del UAV ruta 2.....	195

ÍNDICE DE ANEXOS

Anexo A. Resolución de las normas generales que regulan la operación de Aeronaves pilotadas a distancia (RPAS).....	212
Anexo B. Descripción de los componentes fundamentales que constituyen el transmisor de video Eachine-TX801.....	213
Anexo C. Radio control Jumper T-Pro ELRS.....	213
Anexo D. Especificaciones técnicas de la tarjeta gráfica GEFORCE GTX 1060 6GB... ..	214
Anexo E. Especificaciones técnicas de la placa de control de vuelo y la placa ESC encargada de regular las velocidades de los motores.....	215
Anexo F. Archivo de codificación "det_vehiculos.py" en ordenador portátil.....	219
Anexo G. Archivo de codificación "filtrado.py" en ordenador portátil.....	222
Anexo H. Archivo de codificación "recortes_placas_ubi_py" en ordenador portátil.... ..	226
Anexo I. Archivo de codificación "recortes_solo_ubi_py" en ordenador portátil..	230
Anexo J. Registro de evento en la base de datos, notificación en el grupo de Telegram de los usuarios y correo electrónico del propietario	232
Anexo K. Codificación en Python del archivo envioATelegram.py	233
Anexo L. Codificación en Python del archivo envioATelegramAdmin.py.....	236
Anexo M. Codificación del archivo de la página principal pagina.php.....	240
Anexo N. Codificación del archivo de registro guardar_datos_registro.php.....	244
Anexo O. Codificación del archivo de consulta datos_consulta.php.....	247
Anexo P. Codificación del archivo de estilos styles.css	251

Anexo Q. Pruebas de vuelo del UAV en los parqueaderos de la Universidad Técnica de Ambato	255
Anexo R. Tabla de abreviaturas	257

RESUMEN EJECUTIVO

La presente investigación se centra en el diseño e implementación de un sistema tecnológico avanzado para la detección de automóviles robados, integrando diversas tecnologías como visión artificial, redes neuronales convolucionales YOLOv8, reconocimiento óptico de caracteres (OCR) y vehículos aéreos no tripulados (UAV). Este sistema innovador captura, procesa y analiza imágenes de vehículos en parqueaderos, permitiendo identificar matrículas y compararlas con bases de datos en la nube gestionadas en Azure. Los resultados obtenidos son notificados a través de una plataforma web y aplicaciones como Telegram y correo electrónico, facilitando así una respuesta ágil y eficiente.

El sistema está compuesto por un UAV equipado con una cámara FPV que transmite video en tiempo real a un receptor conectado a un ordenador. La información procesada se almacena en una base de datos y es verificada automáticamente. La configuración del UAV, permite vuelos estables y seguros, mientras que los modelos de inteligencia artificial utilizados garantizan alta precisión en la detección de matrículas vehiculares, incluso bajo condiciones variables de iluminación. Además, se llevaron a cabo pruebas de campo en diferentes ubicaciones para evaluar el rendimiento del sistema en términos de tiempo de vuelo, consumo energético y procesamiento de datos.

Entre los resultados más destacados se encuentra una tasa de confiabilidad 82.78% en la detección de matrículas vehiculares con tiempos de envío de información promedio de 20.26 segundos por evento detectado y envío de notificaciones respectivas. Este proyecto representa una contribución significativa al fortalecimiento de la seguridad pública, proporcionando una herramienta escalable y tecnológicamente avanzada.

Palabras clave: UAV, detección de automóviles, visión artificial, YOLOv8, OCR, Azure.

ABSTRACT

The present research focuses on the design and implementation of an advanced technological system for the detection of stolen cars, integrating various technologies such as computer vision, YOLOv8 convolutional neural networks, optical character recognition (OCR) and unmanned aerial vehicles (UAV). This innovative system captures, processes and analyzes images of vehicles in parking lots, allowing license plates to be identified and compared with cloud databases managed in Azure. The results obtained are notified through a web platform and applications such as Telegram and email, thus facilitating an agile and efficient response.

The system is composed of a UAV equipped with an FPV camera that transmits real-time video to a receiver connected to a computer. The processed information is stored in a database and automatically verified. The UAV configuration allows stable and safe flights, while the artificial intelligence models used guarantee high accuracy in license plate detection, even under variable lighting conditions. In addition, field tests were carried out in different locations to evaluate the system's performance in terms of flight time, energy consumption and data processing.

Among the most outstanding results is an 82.78% reliability rate in the detection of vehicle license plates with average information sending times of 20.26 seconds per detected event and sending of respective notifications. This project represents a significant contribution to strengthening public security by providing scalable and technologically advanced.

Keywords: UAV, car detection, computer vision, YOLOv8, OCR, Azure.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA PARA DETECCIÓN DE AUTOMÓVILES ROBADOS EMPLEANDO VISION ARTIFICIAL EN VEHÍCULO AÉREO NO TRIPULADO (UAV)

1.1.1 Planteamiento del problema

Para comprender en amplitud el problema del robo de vehículos en América Latina, se debe considerar los hallazgos de diversas fuentes de información correspondientes al caso. Según la Encuesta de las Naciones Unidas sobre Tendencias del Delito y Operaciones de los Sistemas de Justicia Penal (UN-CTS), se observa una tendencia heterogénea en los países de la región. Mientras algunos, como Trinidad y Tobago y Estados Unidos, muestran una disminución del 30% en los índices de robo, otros, como Perú y Colombia, experimentan un aumento notable en este delito. Esta variabilidad en las estadísticas refleja la complejidad del problema y la necesidad de abordarlo de manera integral [1].

Los datos de la Fiscalía General del Estado (FGE) de Ecuador revelan una tendencia alarmante en el país. En los últimos tres años, se ha observado un incremento significativo en los robos de automóviles. En el año 2022, las denuncias alcanzaron la cifra preocupante de 11372, representando un aumento del 150% en comparación con las 4596 denuncias registradas en el año 2020. Este aumento significativo sugiere la necesidad de implementar medidas efectivas o mejorar las existentes para combatir este delito y proteger los bienes de los ciudadanos del país [2].

Los datos específicos de la Fiscalía General del Estado (FGE) sobre el robo de vehículos en la provincia de Tungurahua proporcionan una visión más a detalle del inconveniente. Entre los años 2019 y 2022, se observa un incremento del 78.1% en los casos de robo de vehículos en esta provincia. Estos números subrayan la necesidad de acciones específicas a nivel local para mejorar la seguridad y proteger los activos de los ciudadanos afectados por este delito. En conjunto, estos hallazgos permiten

apreciar la complejidad del problema del robo de vehículos en América Latina y la importancia de abordarlo desde múltiples niveles y enfoques integrales [2].

En la actualidad, la eficacia de las fuerzas de seguridad para la recuperación de vehículos se ve severamente comprometida debido a la asignación insuficiente de presupuestos. Esta falta de recursos financieros limita la capacidad de implementar soluciones innovadoras para abordar el problema. Como resultado, la recuperación de vehículos robados o extraviados se ve obstaculizada, ya que las fuerzas de seguridad carecen de los medios necesarios para adoptar enfoques más avanzados y efectivos.

La tecnología ineficiente para la detección de vehículos robados representa un obstáculo significativo en la lucha contra el crimen automotor. La ausencia de sistemas avanzados de vigilancia y seguimiento dificulta la rápida localización de automóviles sustraídos. Este retraso no solo permite que los delincuentes tengan más tiempo para ocultar o desmantelar los vehículos, sino que también reduce la probabilidad de éxito en las operaciones de recuperación.

La falta de protocolos de colaboración entre agencias de seguridad provoca una descoordinación en las instituciones responsables de combatir el robo de vehículos. Esta ausencia de estrategias conjuntas y comunicación clara resulta en respuestas fragmentadas ante los robos vehiculares, dificultando la identificación y persecución de los delincuentes. Esta descoordinación socava la confianza pública en las instituciones encargadas de garantizar la seguridad y el orden público.

1.2 Antecedentes investigativos

En relación con el tema que se está planteando se revisó antecedentes similares, sirviendo como referencia para el desarrollo de esta investigación.

En la investigación presentada por Navas Núñez, Lourdes Gianella y Obando Reinoso, Christopher Sebastián, de la ESPE en el año 2024, se enfoca en el "Diseño e implementación de un prototipo de sistema de seguridad integral para monitorear el acceso de automóviles, utilizando visión artificial y chatbot para el ingreso al conjunto residencial". Este proyecto emplea visión artificial junto con la red neuronal detectNet para delinear la zona de interés y obtener las coordenadas (x, y) necesarias. La

utilización de filtros y luces infrarrojos garantiza la captura de imágenes claras y de alta calidad, mientras que la Compensación de Luces Altas (HLC) corrige distorsiones causadas por la iluminación. Además, se implementaron bases de datos en un archivo SQLite para gestionar el registro de residentes y visitantes. Todo el desarrollo del software se realizó en Python utilizando librerías externas, con el apoyo de herramientas como OpenCV para el procesamiento de imágenes digitales y PyTorch para el entrenamiento de modelos de redes neuronales. Este enfoque tecnológico permitió la creación de una interfaz gráfica intuitiva que cumple con estándares de usabilidad y seguridad. La red neuronal, tuvo una predicción superior al 80%, se ha observado una ligera diferencia de al menos un 2% entre las etiquetas cargadas y las predichas [3].

En el año 2020, Eduardo Barbecho Bautista y Martín Zhindón Mora llevaron a cabo una investigación titulada "Diseño de un algoritmo de reconocimiento de placas vehiculares ecuatorianas usando redes neuronales convolucionales" en la Universidad Católica de Cuenca. El enfoque de esta investigación fue el desarrollo de un algoritmo de reconocimiento de placas vehiculares ecuatorianas utilizando redes neuronales convolucionales y la arquitectura de Google, COCO Inception V2, como base de entrenamiento. La metodología implementada incluyó el etiquetado manual de imágenes de vehículos para identificar la ubicación de las placas y caracteres, con un enfoque en fotos tomadas por cámaras en carreteras con vehículos en movimiento y alta velocidad. Se separó el set de fotos en grupos de entrenamiento y pruebas, y se realizó un etiquetado manual y recorte de caracteres de la placa para un segundo algoritmo de identificación de caracteres. Los resultados mostraron una precisión del 85.1% en términos del conjunto de datos de fotos de entrenamiento de placas ecuatorianas [4].

En el año 2020, Milton Daniel Torres Carrera realizó la investigación "Reconocimiento automático de la placa de un vehículo de Ecuador" en la Universidad Politécnica Salesiana, Sede Quito. Esta investigación se centró en desarrollar un sistema de reconocimiento de placas vehiculares ecuatorianas utilizando técnicas de visión por computador. Para el preprocesamiento de las imágenes, se realizaron redimensionamiento y binarización, y la segmentación se logró mediante detección de bordes y componentes conectados. La clasificación de los caracteres de las placas se

llevó a cabo mediante dos redes neuronales desarrolladas en MATLAB, una para letras y otra para números, entrenadas con un conjunto de datos de 500 imágenes de letras y 500 de números. Las imágenes fueron capturadas con un dispositivo móvil Samsung S8 a una resolución de 4032 x 3024 píxeles y luego reducidas a 800 x 600 píxeles para el procesamiento. Los resultados mostraron una tasa de efectividad del 97% en condiciones controladas, destacando la robustez del sistema para su posible aplicación en sistemas de monitoreo y vigilancia vehicular [5].

En el año 2018, en la Universidad La Salle Bolivia, Orlando Zambrana Ramírez desarrolló un estudio titulado "Detección de placas vehiculares mediante visión artificial", con el propósito de crear un algoritmo eficiente para la detección automatizada de placas vehiculares. La implementación del algoritmo se llevó a cabo en Matlab, una plataforma común para el procesamiento de imágenes y visión computacional. El proceso incluyó el preprocesamiento de imágenes, que consistió en la conversión a escala de grises, binarización y filtro de ruido. Se aplicó el algoritmo Canny para la detección de bordes, seguido del cálculo y extracción de regiones, que implicó la identificación de regiones conectadas, etiquetado, relleno y finalmente la identificación de cada región como una posible placa vehicular. Los resultados mostraron una eficiencia del 86% en la identificación de placas vehiculares utilizando el algoritmo Canny. Además, se comparó el desempeño con otros enfoques comunes como el Algoritmo Sobel y la Transformada de Hough, donde el Algoritmo Sobel superó ligeramente al algoritmo Canny con un 90% de precisión, mientras que la Transformada de Hough obtuvo un rendimiento del 70% [6].

La investigación realizada por Dennis Lida Rojas Pastrana en su investigación "Desarrollo de un sistema de reconocimiento de placas y su influencia en la detección de vehículos robados en la Municipalidad de San Isidro" en julio del año 2017 se enfocó en implementar un sistema de identificación de placas vehiculares para mejorar la detección de vehículos robados. El estudio se basó en el uso de tecnologías como el reconocimiento óptico de caracteres (OCR) y el software de código abierto "JAVA ANPR". La metodología incluyó el análisis y diseño de un sistema de denuncia de vehículos robados, la configuración de un algoritmo de reconocimiento de patrones para la detección de caracteres de placas vehiculares, y el desarrollo de una interfaz de usuario amigable. Se destacó la importancia de la automatización de los sistemas de

detección de placas, la disminución de costos en horas/hombre, y la mejora en la seguridad en áreas sensibles como bancos. Además, se resaltó la alta fiabilidad en la detección de placas, con un índice superior al 85%, y la maximización del número de vehículos recuperados después de un robo [7].

1.3 Fundamentación teórica

Para llevar a cabo este proyecto, es fundamental contar con información clave para la creación de un sistema para la detección de vehículos robados, empleando visión artificial en un vehículo aéreo no tripulado (UAV). Se abordarán temas como el robo de vehículos, sus indicadores, los procedimientos en Ecuador, y tecnologías avanzadas tales como redes YOLOv8, GPS, cámaras FPV, OCR, inteligencia artificial, plataformas en la nube y equipos computacionales. En las siguientes secciones, se desarrollará de manera detallada cada uno de estos aspectos.

1.3.1 Robo de vehículos

El robo de vehículos implica la apropiación ilegal de un vehículo sin el consentimiento del propietario. Esto involucra una situación de alto riesgo para la vida de las personas. Ante un posible robo, las personas dueños de los vehículos no asegurados se enfrentarán a los delincuentes llegando a poner en riesgo su integridad física; pues saben si su bien es robado sufrirán una doble pérdida: a más de que quedarse sin el vehículo, tendrán que pagar a la institución financiera que les otorgó el crédito para adquirir el carro. El acto delictivo se puede realizar de diferentes formas, como el uso de dispositivos electrónicos, forzamiento de cerraduras, o también la amenaza directa. Las leyes y sanciones asociadas al robo de vehículos varían según la jurisdicción, pero en general, se considera un delito grave que puede conllevar consecuencias legales severas para los perpetradores [8].

1.3.2 Indicadores de robo de vehículos

Los indicadores habituales de robos de vehículos pueden cambiar y estar influenciado por múltiples factores, como las condiciones particulares de seguridad en cada zona, el entorno socioeconómico y la ubicación geográfica [9], [10].

En la Tabla 1 se detallan los indicadores más recurrentes de robos en Ecuador.

Tabla 1. Indicadores de robo de vehículos en Ecuador [10].

Indicadores	Características
Oportunidad y falta de seguridad	La carencia de medidas de protección son causas para que se den muchos de los robos de vehículos. Dejar el vehículo en zonas que no tiene mucha iluminación o sin supervisión, así como no asegurarlo adecuadamente o mantener las ventas abiertas son condiciones favorables para que los delincuentes actúen.
Desarrollo de bandas organizadas	En ciertas ocasiones, los robos de vehículos son cometidos por bandas criminales que operan de forma estructurada y especializada en este tipo de delitos. Estas organizaciones suelen emplear métodos avanzados tanto para sustraer como para comercializar los automóviles o sus componentes.
Altos índices de pobreza y desempleo	Los factores socioeconómicos, el desempleo y la pobreza, pueden desempeñar un papel importante en el incremento de la criminalidad, incluido el hurto de vehículos. Las dificultades económicas pueden empujar a las personas a buscar oportunidades delictivas como una forma de subsistencia.
Tráfico ilícito de vehículos	En la gran mayoría de las ocasiones, los automóviles robados son empleados en actividades ilegales como contrabando de mercancías, tráfico de personas o el transporte de drogas, ya que los criminales seleccionan vehículos adecuados para llevar a cabo sus actividades ilícitas.
Falta de conciencia y educación en seguridad	La ausencia de información acerca de las medidas de seguridad adecuada y la falta de educación pública acerca de la relevancia de proteger los vehículos, pueden llevar a que los propietarios no tomen precauciones necesarias para evitar el robo.

1.3.3 Gestión de seguridad vehicular

Se refiere a un conjunto de métodos y estrategias orientado a garantizar la protección de los vehículos en diferentes ambientes, como estacionamiento, vías públicas y áreas de acceso restringido. Su propósito principal es aplicar medidas preventivas que disminuyan los riesgos de robos, daños o accidentes, al mismo tiempo que se busca mejorar la eficiencia y el uso adecuado de los espacios de estacionamiento [11].

1.3.4 Procedimiento tras el robo de un vehículo en Ecuador

Si un vehículo es robado en Ecuador, es fundamental seguir una serie de pasos para reportar el delito y gestionar los trámites pertinentes. El primer paso es que la persona titular del vehículo presente la denuncia ante la Dirección Nacional de la Policía

Judicial, proporcionando detalles del vehículo y las circunstancias del robo. Posteriormente, es necesario informar a la Agencia Nacional de Tránsito (ANT) y solicitar el bloqueo de la matrícula para prevenir su uso indebido. Además, el propietario debe informar a su aseguradora para activar la cobertura del seguro, y permanecer en comunicación con las autoridades para dar seguimiento al caso, dado resumen se presenta en la Figura 1 [12].

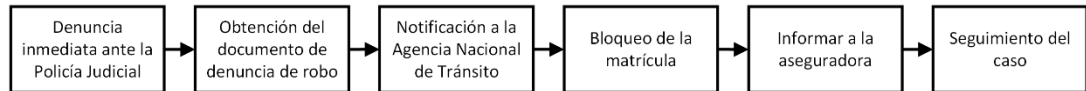


Figura 1. Proceso para denuncia de vehículo robado [12].

1.3.5 Procedimiento operativo para recuperar vehículos robados en Ecuador

Según el Cabo Segundo Luis Guandidango, miembro de la Policía Nacional en una entrevista realizada en diciembre del 2024, tras la denuncia presentada por el dueño del vehículo ante la Policía Judicial que es una entidad que opera las 24 horas del día, al instante se genera una alerta dirigida hacia el personal de la Policía Nacional de la región en donde sucedió el incidente, tanto por radio y por chat, los datos de la denuncia con los detalles respectivos se ingresarán al sistema. Aproximadamente 15 días después se asigna un agente investigador para llevar las pericias del caso, para que el agente logre tener acceso a la información debe realizar un oficio dirigido a la Fiscalía General del Estado solicitando un documento habilitante. Con este documento el agente obtiene acceso a las grabaciones de cámaras privadas y del sistema integrado ECU 911, adicional existen las fuentes humanas que pueden tener la versión de los hechos o pistas que guíen el caso. Terminado el plazo de investigación se debe generar un informe a la Fiscalía detallando todo el proceso de recolección de información y resultados del caso ya sea que se encontró o no el vehículo, adicional si el automotor es localizado es inmediatamente trasladado a los patios de la Policía Nacional para entregarlo a la persona registrada como propietario del vehículo. Con el informe presentado la Fiscalía genera reportes describiendo la situación del caso e inscribiendo la información en sus registros oficiales. En la Figura 2 se muestra el proceso para recuperar un vehículo robado según la Policía Nacional.

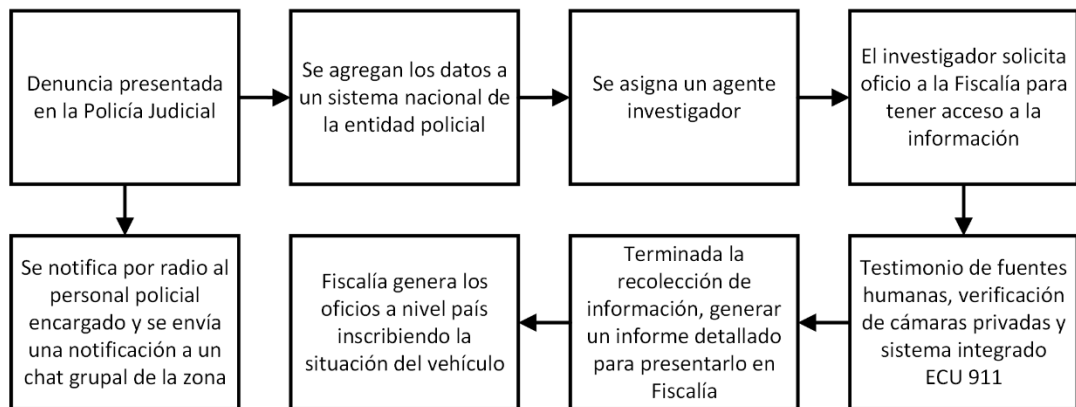


Figura 2. Proceso para recuperar un vehículo robado.

1.3.6 Tecnologías y herramientas para localización de vehículos

Las tecnologías para la localización de vehículos proporcionan herramientas clave para su recuperación. Estas tecnologías incluyen sistemas de rastreo en tiempo real como GPS, RFID, que permiten monitorear un vehículo, obteniendo su ubicación continuamente. Además, con el uso de cámaras especializadas para reconocimiento de placas (LPR, ALPR) y sistemas de video vigilancia CCTV con cámaras convencionales facilitando información de la ruta tomada por un vehículo para darle seguimiento, brindando herramientas de refuerzo para la recuperación. La actualización de estas tecnologías ayuda a gestionar la información de los robos de vehículos, produciendo que la prevención y la localización minimicen los tiempos de respuesta a los eventos delictivos.

a. Tecnologías de rastreo

Las tecnologías de rastreo permiten localizar un vehículo mediante dispositivos que emiten señales continuamente, aprovechando redes de telecomunicaciones y sistemas de posicionamiento global.

- ***GPS (Global Positioning System)***

Un localizador GPS para automóviles es un sistema que opera con la finalidad de ubicar un vehículo a través de una red de satélites que transmite en tiempo real las coordenadas de ubicación del vehículo en el que está instalado el dispositivo de localización [13].

Los satélites se sincronizan en un tiempo corto y la transmisión de envío y recepción de datos con el localizador se realiza en tiempo real, determinando rápida y exactamente la posición del objeto o persona, aplicado en este caso a un auto [13].

- ***RFID (Radio Frequency Identification)***

El seguimiento de un vehículo en grandes distancias se lo puede realizar con el montaje de puntos de control a lo largo de la ruta, la localización del vehículo se puede aproximar en función del tiempo que tarda el automóvil en pasar por los puntos de control. Para las empresas y personas que tienen como objetivo rastrear vehículos en tiempo real los dispositivos GPS para vehículos son una mejor solución [14].

La comparación técnica presentada en la Tabla 2, relacionada con las tecnologías de rastreo de vehículos, es crucial para elegir la opción más adecuada para implementar en el sistema de detección de vehículos robados.

Tabla 2. Tecnologías de rastreo de vehículos [13], [14].

Características	GPS (Global Positioning System)	RFID (Radio Frequency Identification)
Principio de Funcionamiento	Satélites para determinar la ubicación	Etiquetas RFID que interactúan con lectores a distancia
Cobertura	Global, siempre que haya acceso a satélites	Limitada a áreas cercanas a los lectores RFID
Precisión	Alta, con un radio limitado	Menor, dependiendo de la proximidad al lector
Requiere Energía	Constante para los dispositivos de rastreo	No requieren en etiquetas pasivas
Costo	Costoso debido a los dispositivos y tecnología satelital	Económico, en las etiquetas pasivas
Durabilidad	Los dispositivos GPS son duraderos, pero dependen de la señal	Las etiquetas RFID son pequeñas, resistentes y con larga vida útil
Uso Común	Rastreadores vehiculares, navegación	Control de acceso
Instalación	Requiere instalación en el vehículo y visibilidad de satélites	Instalación sencilla de etiquetas y lectores en puntos estratégicos

b. Sistemas de cámaras

Son dispositivos diseñados para capturar videos, procesarlos y almacenarlos con fines como de vigilancia, monitoreo, reconocimiento, o análisis visual. Los sistemas integran cámaras para aplicaciones específicas como el monitoreo de tráfico y seguimiento a vehículos, las nuevas tecnologías incluso han incorporado IA dirigida a reconocimiento de caracteres, rostro y detección de objetos.

- ***ALPR***

El reconocimiento de matrículas automático (ALPR) es un proceso automatizado que captura imágenes de la matrícula y la escanea con la finalidad de identificar el vehículo digitalizando los caracteres de la matrícula. Los lectores automáticos de matrículas son sistemas de cámaras de alta velocidad controlados por un ordenador que integra IA, además suelen utilizar tecnología de reconocimiento óptico de caracteres para identificar las matrículas a una velocidad mayor a la de los operadores humanos o las cámaras convencionales, manejando también bases de datos para aplicaciones específicas [15].

- ***Cámaras de seguridad***

Un CCTV o circuito cerrado de televisión es una instalación de cámaras conectadas permitiendo el acceso a videos de secuencia de imágenes que se procesan en un ordenador en el que se llega almacenar los videos capturados, estas se personalizan para adaptarse a las necesidades personales ya sea orientadas a la seguridad, vigilancia o mejora de servicio [16].

La Tabla 3 expone una comparación técnica de los sistemas de cámaras de seguridad, la cual es clave para identificar la opción óptima que garantice una grabación de video precisa y de alta calidad.

Tabla 3. Sistemas de cámaras de seguridad [15], [16].

Características	ALPR (Automatic License Plate Recognition)	Cámaras CCTV (Circuito Cerrado de Televisión)
Objetivo Principal	Reconocimiento automático de matrículas de vehículos	Vigilancia y grabación de imágenes para monitoreo de seguridad
Tecnología	Cámaras especializadas y software para leer matrículas	Cámaras estándar de video para grabación continua
Procesamiento	Procesamiento en tiempo real con algoritmos de visión artificial	Solo grabación, sin procesamiento automático
Aplicación	Control de acceso vehicular, seguridad pública, peajes	Monitoreo de áreas públicas y privadas
Precisión	Alta precisión en condiciones óptimas (iluminación, distancia).	Depende de la calidad de la cámara y la resolución de imagen.
Tipo de Imagen	Captura imágenes de las matrículas de vehículos	Captura de imágenes generales de área determinada
Costo	Más costoso debido al hardware especializado y software.	Más económico, dependiendo de la calidad y características.
Velocidad de Procesamiento	Alta velocidad para análisis en tiempo real de vehículos.	Sin procesamiento automático, solo grabación.

1.3.7 Identificadores vehiculares únicos

Los identificadores para vehículos actúan como un tipo de documento de identidad para los vehículos. Estas denominaciones, que fusionan letras y números, se graban directamente sobre el armazón y la carrocería del automóvil durante la producción. En términos generales, estas identificaciones son exclusivas para cada vehículo [17]. Se presenta a continuación los tipos de identificación alfanumérica de vehículos:

a. Número de motor

Se refiere a los números impresos en el bloque (motor), los cuales han sido localizados por la empresa de montaje y grabados en un saliente, particularmente en el lado derecho o izquierdo, en la parte anterior o posterior del bloque, dependiendo de la marca del vehículo. Algunas compañías fabricantes suelen utilizar placas metálicas adheridas con remaches como los de Volkswagen y Renault, o soldadura de punto [18].

b. Número de chasis

Es la numeración asignada a la estructura del automóvil, siendo el componente más importante para identificar un vehículo, dado que es la pieza más grande e indispensable de un vehículo y de difícil sustitución. En los vehículos actuales, la numeración del chasis ha sido sustituida por una placa de aluminio o metálica, fijada con tornillos o remaches adheridos a la carrocería o también en el tablero de instrumentos [18].

c. VIN

Es un identificador importante que se incluye en los sistemas de identificación de automóviles, es el número que reconoce y documenta toda la historia de un vehículo. El VIN es relevante ya que es un código asignado al vehículo que facilita su diferenciación de otros con características parecidas. El propósito del número de identificación alfanumérica vehicular es facilitar la identificación indiscutible de cualquier vehículo. Este número se imprime o se remueve en una placa y puede ubicarse en diversas partes del vehículo. La presencia del número VIN en los vehículos tiene como objetivo protegerlos de hurto, manipulación o falsificación, además el registro también es almacenado en la computadora que cada vehículo incorpora, como fuente de respaldo [18].

d. Placa vehicular

Las placas tienen 154 mm de alto y 404 mm de ancho y deben ser reflectivas con el fin de mejorar su visibilidad. Están formadas por tres letras y tres (formato antiguo) o cuatro (formato nuevo) dígitos, partiendo desde 000 al 9999, en formatos de ABC-123 (formato antiguo) y ABC-1234 (formato nuevo), con el nombre del país en mayúsculas en la parte superior, identificando por provincia [19].

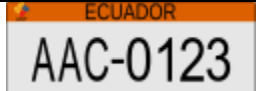
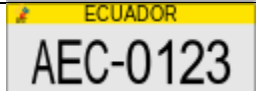
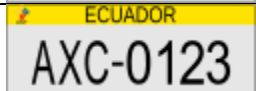
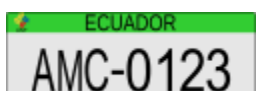
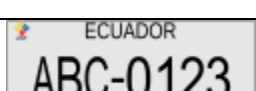
En la Tabla 4 se muestran las iniciales respectivas de las placas vehiculares por provincia.

Tabla 4. Iniciales de placas por provincia [19].

Provincia	1era Letra	Provincia	1era Letra	Provincia	1era Letra
Azuay	A	Galápagos	W	Orellana	Q
Bolívar	B	Guayas	G	Pastaza	S
Cañar	U	Imbabura	I	Pichincha	P
Carchi	C	Loja	L	Santa Elena	Y
Chimborazo	H	Los Ríos	R	Santo Domingo de los Tsáchilas	J
Cotopaxi	X	Manabí	M	Sucumbíos	K
El Oro	O	Morona Santiago	V	Tungurahua	T
Esmeraldas	E	Napo	N	Zamora Chinchipe	Z

Además, el color del fondo y la segunda letra de la matrícula se utiliza para identificar el tipo de vehículo, los ejemplos se muestran en la Tabla 5.

Tabla 5. Tipos de placas y ejemplos [19].

Tipo	2da Letra	Color	Ejemplo	Imagen
Vehículos comerciales (como taxis o autobuses)	A, U, Z	Naranja	AAB-0123	
Vehículos gubernamentales	E	Oro	PEB-0001	
Vehículos de uso oficial	X	Oro	GXA-0100	
Vehículos de los Gobiernos Autónomos Descentralizados (Regiones, Provincias, Cantones, Parroquias)	M	Verde limón	LMA-0010	
Vehículo particular (privado)	Cualquiera menos las anteriores	Blanco - plateado	EBA-0234	

1.3.8 Sistema de reconocimiento de vehículos robados con visión artificial

Un sistema de reconocimiento de vehículos robados con visión artificial es una tecnología que aprovecha algoritmos de procesamiento visual y técnicas de aprendizaje automático para identificar automóviles reportados como robados

utilizando como identificador vehicular el número de placa ya que es el parámetro al que se tiene acceso visualmente. La tecnología se basa en la captura de imágenes en tiempo real y el análisis de características específicas de los vehículos, comparándolas con una base de datos de vehículos robados. En caso de encontrar una coincidencia, se genera una alerta inmediata para las autoridades correspondientes [5].

El sistema se fundamenta en un análisis detallado de imágenes de vehículos obtenido mediante una cámara integrada a un UAV. Con métodos avanzados de transmisión de video y procesamiento, se detectan vehículos y las placas respectivas, después con la información extraída se realiza el manejo respectivo en bases de datos.

1.3.9 Permiso para volar UAVs en Ecuador

Ecuador regula el uso de drones mediante el reglamento de la Dirección General de Aviación Civil (DGAC) presentado en el Anexo A, aplicable a aeronaves no tripuladas de entre 0.25 kg y 150 kg. Es obligatorio registrarlos y contar con seguros de responsabilidad civil por montos de USD 3000 a 12000 según su peso. Los drones deben operar a una altura máxima de 122 m, fuera de zonas restringidas como aeropuertos y áreas de seguridad, y solo en condiciones climáticas adecuadas durante el día. Los operadores deben ser mayores de 18 años, dominar el español, haber completado la enseñanza media y aprobar un curso teórico/práctico dentro de los últimos 24 meses. Para actividades recreativas con drones menores a 25 kg, no se requiere seguro, pero el operador será responsable de los daños causados [20].

1.3.10 Reglamento de radiocomunicaciones

Al ser la ley suprema que rige al país, la Constitución del Ecuador es, por ende, la máxima autoridad que dirige el sector. Basándose en esta, se han establecido algunas leyes complementarias que contribuyen a regular y controlar todo lo relacionado con las Telecomunicaciones en el país, como el Plan Nacional de Frecuencias.

El documento tiene como objetivo promover el uso del espectro radioeléctrico y los servicios de radiocomunicación de manera eficiente, organizada y controlada en todo el país, buscando aprovechar al máximo este recurso [21].

Se aprobará la operación de sistemas de radiocomunicaciones, en las siguientes frecuencias de la Tabla 6.

Tabla 6. Bandas de frecuencia asignadas en Ecuador [21].

Banda [MHz]	Asignación
902 – 928	ICM
2400 – 2483.5	ICM
5150 – 5250	INI
5250 – 5350	INI
5470 – 5725	INI
5725 – 5850	ICM, INI

Considerando que las bandas ICM e INI, se definen como bandas no licenciadas, por lo cual su uso no requiere de una licencia, se representa la asignación de bandas de frecuencia en Ecuador para distintos usos de radiocomunicaciones, indicando los rangos aprobados y su finalidad. Las bandas 902 MHz a 928 MHz, 2400 MHz a 2483.5 MHz, y 5725 MHz a 5850 MHz reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnética en áreas industrial, científica y médica (ICM), como controles remotos y sistemas de monitoreo. Por otro lado, las bandas 5150 MHz a 5250 MHz, 5250 MHz a 5350 MHz, 5470 MHz a 5725 MHz, y parcialmente 5725 MHz a 5850 MHz son asignadas a la Infraestructura Nacional de Información (INI), permitiendo aplicaciones como comunicación entre dispositivos y transmisión de video. En conjunto, se organiza el uso del espectro radioeléctrico para garantizar un funcionamiento eficiente y libre de interferencias entre sistemas, para el proyecto se considera la frecuencia para la comunicación entre el radio control y el UAV, como también la frecuencia a la que se transmite el video capturado.

1.3.11 Vehículo aéreo no tripulado (UAV)

Comúnmente llamados drones, son aeronaves controladas a distancia. Aunque no llevan un piloto a bordo, en algunos casos pueden ser manejados por operadores humanos que los dirigen desde una ubicación remota. No obstante, también existen drones totalmente autónomos, capaces de tomar decisiones por sí mismos y realizar tareas complejas [22].

El UAV es el elemento central para el desarrollo del proyecto ya que es el vehículo encargado de capturar las imágenes para la detección de los vehículos. En la Tabla 7 se presentan las características técnicas de los drones que los Talleres Tecnológicos de la Facultad de Ingeniería en Sistemas Electrónica e Industrial dispone para las investigaciones respectivas del personal docente y estudiantil, en diversos campos como la agricultura de precisión, localización de personas, carga inalámbrica y seguridad.

Tabla 7. Características técnicas de los UAVs disponibles FISEI.

Características	Cuadricóptero MatekF405-STD FPV	Cuadricóptero SpeedyBeeF405 V3 FPV
		
Peso	700 g	600 g
Dimensiones	300x260x100 mm	135x165x40 mm
Autonomía de vuelo	9 min	12 min
Velocidad máxima	20 Km/h	20 Km/h
Altura máxima de vuelo	122 m	120 m
Número de motores	4	4
Motores incorporados	BR2205 2300KV CCW 2S a 4S (7.4 V a 14.8 V)	KAREREA 2306-2650KV 3S a 5S (11.1 V a 18.5 V)
Sistema de navegación	GPS	GPS
Sensores	Acelerómetro Giroscopio Barómetro	Acelerómetro Giroscopio Barómetro
Sistema de batería	2 S a 6 S (7.4 V a 22.2 V)	3 S a 5 S (11.1 V a 18.5 V)
Material del marco	Fibra de carbono PLA	Fibra de carbono PLA
Software compatible	Betaflight /Mission Planner	Betaflight /Mission Planner
Estándar	AFHDS 2A	ExpressLRS

Los UAVs se clasifican en dos grandes categorías según su estructura: UAVs de ala fija y UAVs de ala rotatoria. Además de su estructura, los UAVs también se clasifican

según el método de control, ya sea de manera manual, a través de control remoto, o de forma autónoma, mediante software y algoritmos de navegación [23].

a. UAVs de ala fija

Estos drones por su diseño aerodinámico similar a los de los aviones, se destacan por su capacidad de cubrir largas distancias utilizando solo una batería. Son útiles para tareas como el mapeo de grandes superficies, pero presentan limitaciones: no pueden mantenerse en un punto fijo ni despegar de manera autónoma, lo que requiere intervención manual al inicio del vuelo. Su uso se enfoca principalmente en maximizar la eficiencia aerodinámica y la resistencia [23], [24]. El UAV de ala fija se puede visualizar en la Figura 3.



Figura 3. UAV de ala fija [23].

b. UAV'S de ala rotatoria o multirrotor

Los drones de ala rotatoria son ampliamente utilizados tanto en ocio como en aplicaciones profesionales, destacan por su versatilidad para montar distintos tipos de cámaras. Su capacidad de despegar y aterrizar de manera vertical desde casi cualquier superficie los hace muy prácticos, aunque su principal limitación es la autonomía de vuelo relativamente corta [23]. En la Figura 4 se muestra un ejemplo de un UAV de ala rotatoria.



Figura 4. Cuadricóptero [23].

Los UAV de ala rotatoria se dividen en categorías basadas en la cantidad de rotores que pueden variar entre tres (tricópteros) hasta ocho (octocópteros), como se muestra en la Figura 5, siendo los cuadricópteros, con cuatro rotores, los más comunes. En general un mayor número de motores implica mayor capacidad de elevación, ya que cada motor añade una fuerza de elevación. También existen prototipos de un solo rotor, como el UAV esférico presentado por el Ministerio de Defensa de Japón en 2011, que utiliza superficies de control para compensar la rotación en el eje de guiñada [25].

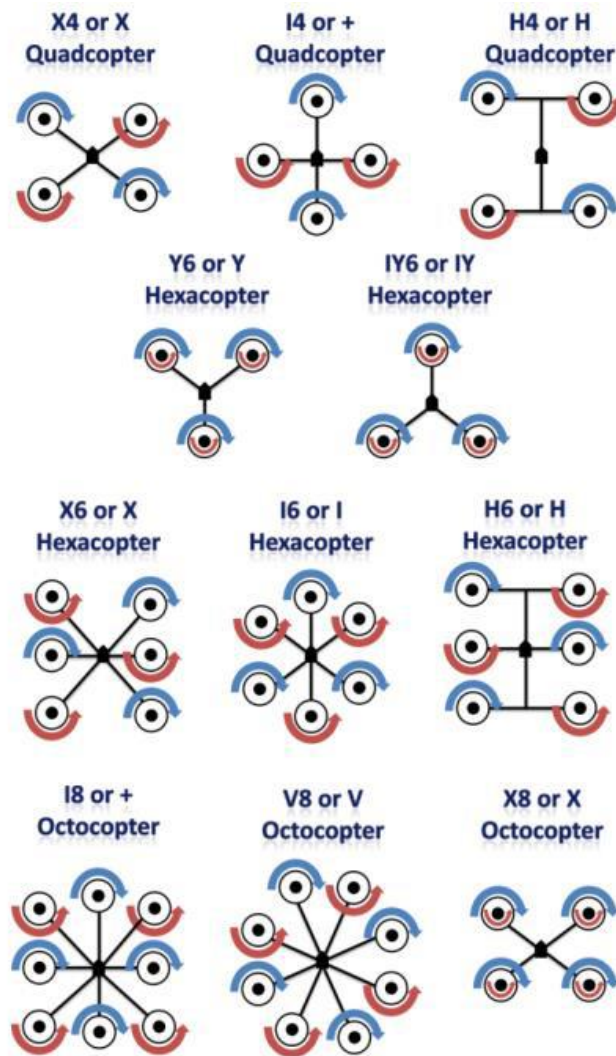


Figura 5. Configuraciones del rotor más comunes [25].

c. UAV'S por el método de control

Según el método de control, podemos clasificar los drones en tres tipos principales. Los drones autónomos que no requieren un piloto, ya que pueden planificar rutas y volar de manera programada, usualmente apoyados en sistemas GPS para su

localización en tiempo real. Por otro lado, los drones controlados remotamente son manejados por un piloto a través de un radiocontrol, siendo esta la opción más utilizada. Finalmente, los drones supervisados que pueden operar parcialmente de manera autónoma, pero requiere de la supervisión continua de un piloto [23].

1.3.12 Software de control de vuelo

a. *Mission Planner*

Este software es una aplicación de estación de control terrestre diseñada para drones que utilizan ArduPilot. Permitiendo configurar y monitorear misiones para vehículos no tripulados (UAVs), así como aviones, helicópteros y rovers [26].

b. *Betaflight*

Este software se centra específicamente en la configuración de drones de forma manual, especialmente en Cuadricóptero y drones de carreras. Su objetivo es proporcionar un control preciso sobre el comportamiento del dron durante el vuelo, facilitando ajustes minuciosos mediante la configuración de parámetros de firmware y sistemas de control PID [27].

La Tabla 8 expone un análisis comparativo de las características más relevantes de los softwares controladores de vuelo, permitiendo seleccionar el que mejor se ajuste a las necesidades del UAV.

Tabla 8. Características software controlador de vuelo [26], [27].

Características	Mission Planner	Betaflight
Enfoque	Drones autónomos y misiones GPS	Drones de carreras y freestyle
Firmware compatible	ArduPilot	Betaflight
Tipo de vuelo	Autónomo y planificado	Manual y acrobático
Interfaz	Completa y detallada (mapas, gráficos)	Sencilla, enfocada en rendimiento
Modos de vuelo	Automático, Loiter, RTL, etc	Acro, Angle, Horizon, etc
Ideal para	Mapeo, rescate, agricultura, largo alcance	Carreras, freestyle, drones acrobáticos
Sistema operativo	Windows (nativo), macOS y Linux con emuladores	Windows, macOS y Linux
Nivel de dificultad	Alto, orientado a proyectos complejos	Moderado, fácil para principiantes
Gratuito	Sí	Sí

1.3.13 Emisor y receptor de vuelo

El emisor es el dispositivo que facilita al usuario el envío de comandos básicos como inclinar, girar o regular la potencia de los motores del UAV. Por otra parte, el receptor, que se encuentra montado en el UAV, capta estas instrucciones a través de señales de radiofrecuencia y las envía a la unidad de control del UAV mediante pulsos a través de cables. Esta comunicación entre el emisor y el receptor permite al usuario ejercer un control total sobre el vuelo del UAV. En la Figura 6 se muestra un ejemplo de un modelo de emisor junto a su receptor correspondiente [28].



Figura 6. Emisora Flysky Fs i6 y Receptor asociado [28].

El radiocontrol representa un componente esencial para establecer la comunicación entre el operador en tierra y la plataforma UAV, permitiendo gestionar y controlar el vuelo a grandes distancias. En la Tabla 9 se presenta un análisis comparativo de los radiocontroles empleados en estas operaciones.

Tabla 9. Características de los radios controles [29], [30], [31].

Características	FlySky FSI6S FS i6s	Fly Sky FS-i6XB	T-pro
			
Rango de RF	2.4055 GHz - 2.475 GHz	2.408 GHz - 2.475 GHz	2,4 GHz - 2,5 GHz
Potencia de RF	<20 dBm	<20 dBm	<20 dBm
Canales	6	8	16
Certificado	CE0678, FCC	CE, FCC	CE, FCC
Estándar	AFHDS 2A	AFHDS 3	ExpressLRS

1.3.14 GPS para dron FPV

El uso del GPS es crucial para lograr una navegación precisa y llevar a cabo tareas autónomas de forma precisa. Es un sistema de navegación satelital proporciona información geográfica altamente precisa, lo que es esencial para el control adecuado del vuelo y orientación del UAV. Además, el GPS facilita la realización de operaciones automáticas en una amplia variedad de aplicaciones con un mayor grado de autonomía y eficiencia. En la Tabla 10 se realiza una comparativa de las características técnicas de tres tipos de GPS disponibles en el mercado.

Tabla 10. Características de los GPS [32], [33], [34].

Características	GPS Matek Systems M10Q05883	GPS GLONASS Beitian Dual BN-880	Readytosky Ublox NEO M8N
			
Dimensión	20 * 20 * 12.4 mm	22 * 20 * 6 mm	35 * 35 * 6 mm
Peso	8 g	5.3 g	10.3 g
Conector	Conector JST-GH-6P (SM06B-GHS-TB) con un paso de 1.27 mm	1,25 mm entre el parche de 6 pines	JST-GH de 6 pines
Conjunto de chips	Módulo GNSS ublox SAM-M10Q-00B y la brújula magnética QMC5883L.	NA	Ublox NEO M8N
Formato de recepción	GPS, GLONASS, Galileo, BeiDou, QZSS y SBAS	GPS, GLONASS, Galileo, BeiDou, QZSS y SBAS	GPS, GLONASS, Galileo, BeiDou, QZSS y SBAS
Canales	72	72	72
Sensibilidad de seguimiento/adquisición	-167 dBm/ -148 dBm a -155 dBm	Traza -167 dBm	-167 dBm/-148 dBm
Sensibilidad de arranque en frío/caliente	-146 dBm. / -157 dBm.	avg25s/avg26s	-148 dBm/-156 dBm
Nivel de datos	TTL	TTL	TTL
Protocolo de datos	UBX y NMEA	NMEA-0183	NMEA, UBX binary, RTCM
Voltaje de operación	4.0 V - 9.0 V	3.6 V - 5.5 V	3.3 V - 5.0 V

1.3.15 Fuente de alimentación para dron FPV

Los drones de carreras que utilizan tarjetas incorporadas compatibles con softwares como Betaflight y Mission Planner necesitan de fuentes externas de alimentación para su operación, para este caso las tarjetas controladoras de vuelo y sus componentes adicionales utilizan batería de polímero de litio (LiPo) utilizadas debido a la compatibilidad y su capacidad, peso, volumen y tensión óptima. Las baterías de LiPo son recargables mediante energía eléctrica utilizando cargadores específicos que balancean su carga y evitan que se sobrecarguen por encima de la tensión nominal. La Tabla 11 presenta un análisis comparativo de las especificaciones técnicas de las baterías LiPo comercializadas en Ecuador, con el propósito de elegir la opción más adecuada para la alimentación del UAV.

Tabla 11. Baterías LiPo [35], [36], [37].

	LIPO TURNIGY NANO-TECH	LIPO TATTU FUNFLY	TATTU R-LINE
Características			
Dimensiones	71x19x35 mm	72x35x36 mm	62x31x36 mm
Peso aproximado	93 g	168 g	116 g
Capacidad mínima	1000 mAh	1550 mAh	650 mAh
Configuración	3 S1P / 11.1 V	4 S1P / 14.8 V	6 S1P / 22.2 V
Tasa de descarga	45 C	100 C	95 C
Velocidad máxima de descarga de ráfaga	90 C	200 C	190 C
Enchufe de carga	JST-XH	JST-XHR	ST-XHR-7P
Enchufe de descarga	XT-60	XT-60	XT-30

1.3.16 Cámaras RGB

Las cámaras RGB son equipos que capturan la luz dentro del espectro visible y la descomponen en tres colores primarios; rojo, verde y azul (RGB). Esta combinación

permite generar una imagen digital, gracias a la mezcla de colores, tanto aditivos como sustractivos, tal como se muestra en la Figura 7 [38].

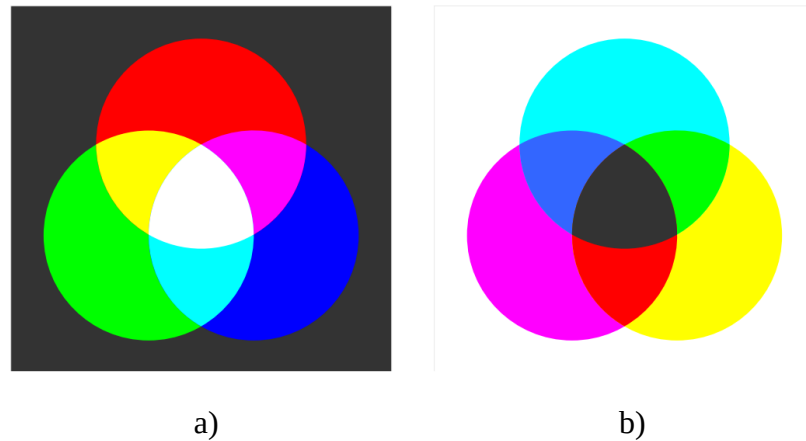


Figura 7. Fusión de colores primarios; a) aditivo, b) Sustractivo [38].

1.3.17 Sistemas FPV

Los sistemas de visión en primera persona (FPV, por sus siglas en inglés) permiten capturar imágenes en tiempo real mediante una cámara instalada en un UAV. Estas imágenes pueden ser visualizadas a través de un monitor o gafas FPV. En la actualidad, la mayoría de los UAV son controlados mediante radiofrecuencia en la banda 2.4 GHz. sin embargo, para integrar un sistema FPV en un UAV comercial, se recomienda utilizar la banda de 5.8 GHz u otras frecuencias que proporcionen un mayor ancho de banda, permitiendo la transmisión de datos a altas velocidades [39].

a. Cámara FPV

Las cámaras FPV están diseñadas para ofrecer una latencia mínima, lo que les permite adaptarse de manera eficiente a las variaciones de la luz en ambientes naturales gracias a su amplio rango dinámico. Tal como se muestra en la Figura 8, estas cámaras, conocidas por su alta eficiencia, suelen ser dispositivos analógicos y por lo general no tiene la capacidad de grabar videos en HD o alta definición [39].



Figura 8. Cámara FPV [39].

La cámara se encarga de capturar las imágenes del video necesarias para la aplicación con parámetros establecidos como los pixeles y velocidad de fotogramas (fps), en la Tabla 12 se presentan las características de las cámaras utilizadas en drones FPV disponibles, tanto la CADDXFPV Ratel y la CMOS Orange cuentan con salida analógica para vuelos FPV.

Tabla 12. Características de cámaras FPV [40], [41].

Características	CADDXFPV Ratel	CMOS Orange
		
Tamaño	19x19x20 mm	25x25x30 mm
Sensor	1/1.8" Starlight HDR CMOS	CMOS
Pixeles	1200 TVL	1200 TVL
Peso	8 g	9.5 g
Sistema de señal	PAL/NTSC	PAL/NTSC
Salida de video	CVBS (analógica)	CVBS (analógica)
Lente	2.1 mm	2.5 mm
Campo de Visión (FOV)	125 grados	120 grados
Voltaje de Funcionamiento	DC 4.5 V-27 V	DC 5 V-12 V

b. Video transmisor y receptor VTX/VRX

Un transmisor de video o VTX es un componente esencial en un sistema FPV, ya que se encarga de transmitir la señal de video capturado por la cámara a un receptor VRX.

Esto permite que las imágenes se visualicen en un monitor o a través de unas gafas FPV, como se muestra en la Figura 9. Los VTX pueden operar en bandas de frecuencia, como 1.2 GHz y 2.4 GHz, pero la banda más comúnmente utilizada es la de 5.8 GHz, ya que permite el uso de antenas más pequeñas [39].

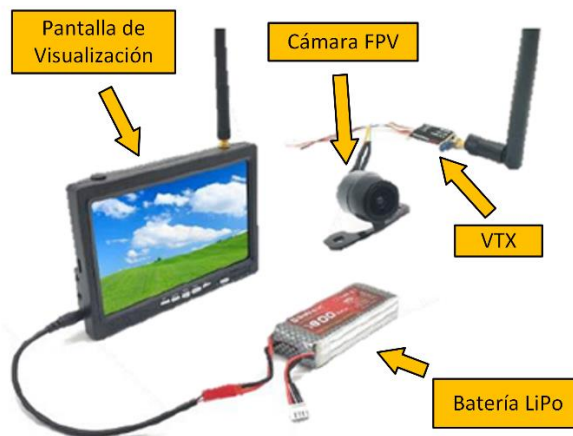


Figura 9. Sistema de transmisión [39].

En la Tabla 13 se detalla las características técnicas de 3 tipos de transmisores de video que se pueden incorporar a un UAV.

Tabla 13. Características de transmisores de video [42], [43], [44].

Características	SpeedyBee - TX 800	Eachine - TX5258	Eachine - TX801
			
Peso	5,6 g	8.25 g	7,3 g
Dimensiones	28 * 28 * 5 mm	31.88 * 22.90 * 5 mm	28 * 20 * 8 mm
Canal de frecuencia	5.8 GHz – 48 canales	5.8 GHz – 72 canales	5.8 GHz – 72 canales
Potencia de salida	25/200/400/800 mW	25/200/500/800 mW	0.01/5/25/50/100 200/400/600 mW
Alcance	2 km	2.5 km	3km
Voltaje de operación	DC 3.7 V- 5 V	DC 7 V – 24 V	DC 7 V- 24 V
Salida de voltaje	5 V	5 V	5 V
Impedancia de video	75 Ω	10 K Ω	10 K Ω
Ancho de banda	18 MHz	18 MHz	18 MHz
Formato de video	NTSC/PAL	NTSC/PAL	NTSC/PAL
Conector de antena	MMCX	BR-SMA	BR-SMA

La comparación entre dos receptores de video disponibles en el mercado se muestra en la Tabla 14, donde se analiza las características técnicas, con el objetivo de seleccionar el más adecuado en términos de calidad de imagen.

Tabla 14. Características de receptores de video [45], [46].

Características	Eachine - ROTG01	Eachine - RC305
		
Peso	28 g	43 g
Dimensiones	61 * 33 * 0.9 mm	61 * 52 * 13 mm
Rango de frecuencias	5645 MHz – 5945 MHz	5705 MHz- 5945 MHz
Canal de frecuencia	5.8 GHz-150 canales	5.8 GHz-8 canales
Sensibilidad	- 90 dBm	- 90 dBm
Potencia	1000 mW	350 mW
Voltaje de operación	200 mA / 5 V	DC 5 V – 12 V
Conector de antena	Rp-SMA Hembra	Rp-SMA Hembra
Conector de salida	Micro USB	Jack 3.5 mm

1.3.18 Inteligencia Artificial

La inteligencia artificial (IA) es un campo de la informática cuyo objetivo es capacitar a las máquinas para realizar tareas complejas que normalmente requieren inteligencia humana, como el aprendizaje y la resolución de problemas. En el ámbito de la IA, un agente inteligente interactúa con su entorno, analiza los datos para tomar decisiones que los dirijan hacia sus objetivos y ejecuta acciones de manera independiente. Además, a través de procesos de aprendizaje, la IA puede mejorar su rendimiento y adaptarse continuamente a nuevas situaciones [47].

Un modelo de la IA que permite a las computadoras procesar datos de manera similar al funcionamiento del cerebro humano son las redes neuronales. Están formadas por nodos interconectados, se los conoce como neuronas artificiales, que cumplen la tarea de procesar datos de entrada para crear una salida. Cada una de las neuronas debe recibir una o más entradas, para transformarlas utilizando una función de activación y luego transmitir el resultado a otras neuronas por las conexiones dadas con

determinados pesos asignados. Cuando se da el entrenamiento de la red, estos pesos se acoplan para minimizar la diferencia entre la salida obtenida y la esperada. El proceso de ajuste utiliza algoritmos de optimización, como el descenso del gradiente. En la Figura 10 se visualiza la representación del funcionamiento general de una red neuronal [48].

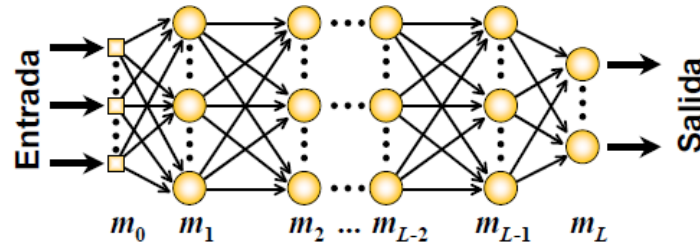


Figura 10. Redes neuronales multicapa [48].

a. *Lenguajes de programación para inteligencia artificial*

Los lenguajes de programación y sus bibliotecas proporcionan una amplia gama de herramientas y opciones para entrenar modelos de inteligencia artificial, además de ofrecer los recursos necesarios para probar y validar el funcionamiento del sistema. Cada lenguaje tiene características particulares que lo hacen más adecuado para distintas aplicaciones, en función del contexto y de los requerimientos específicos del proyecto. Para seleccionar el lenguaje de programación para el reconocimiento de placas vehiculares utilizando UAV y visión artificial, Python, JavaScript, y MATLAB representan tres opciones adecuadas para el caso, cada una con sus características en el marco de inteligencia artificial, como se muestra en la Tabla 15.

Tabla 15. Lenguajes de programación utilizados en IA [49], [50], [51].

Características	Python	JavaScript	MATLAB
Área de especialidad	Versátil en IA y ciencia de datos y	Desarrollo web y aplicaciones	Análisis numérico e ingeniería
Soporte en IA	Muy robusto para librerías como OpenCV, Ultralytics y TensorFlow	Permite desarrollar tareas simples con TensorFlow.js	Adecuado para análisis numérico con toolboxes
Facilidad de aprendizaje	Alta	Media	Alta en aplicaciones de ingeniería
Aplicaciones frecuentes	Redes neuronales y visión artificial	Aplicaciones web en tiempo real	Modelos matemáticos y procesamiento de datos
Compatibilidad con sistemas	Excelente con dispositivos de IA	Limitada a entornos web y Node.js	Buena con sistemas embebidos
Costo	Gratuito	Gratuito	Licencia costosa

b. Entornos de desarrollo integrado (IDE)

Son programas diseñados para facilitar el desarrollo de software, proporcionando una amplia gama de herramientas que optimizan la creación y edición de código. Normalmente, un IDE incluye un editor de código fuente, compiladores, opciones de depuración, y en algunos casos, interfaces gráficas para construir aplicaciones con componentes visuales. En la Tabla 16 se presentan comparaciones de diferentes IDE aplicados a la inteligencia artificial. Entre sus funciones más destacadas están la capacidad de refactorizar el código para hacerlo más claro sin modificar su lógica, insertar tabulaciones de forma automática, ampliar sus capacidades mediante plugins o módulos adicionales, y organizar menús y herramientas para mejorar la eficiencia del desarrollo [52].

Tabla 16. Comparación de entornos de desarrollo integrado [53], [54], [55].

Características	Jupyter Notebook	Google Colab	Visual Studio Code
Lenguajes soportados	Python, R, Julia, entre otros	Principalmente Python, admite otros	Python, JavaScript, HTML, entre otros
Acceso a la nube	No	Sí	No
Gestión de memoria	Automática	Manual	Automática
Acceso a datos externos	A través de bibliotecas y extensiones	Fácil acceso a datos públicos (Google Drive)	Por integración con bibliotecas de Python
Control de versiones	Sí (mediante extensiones Git)	No	Sí (mediante extensiones Git)
Portabilidad	Alta	Media	Media
Interfaz de usuario	Basado en navegador	Basado en navegador	GUI (Interfaz gráfica)
GPUs	Según configuración local	Acceso a GPUs (gratis o pago)	Según configuración local
RAM	Según entorno de ejecución	13 GB	Según entorno de ejecución
Librerías IA	TensorFlow, PyTorch, Keras, etc.	TensorFlow, PyTorch, Keras, etc.	TensorFlow, PyTorch, Keras, etc.

c. Visión artificial

La visión artificial es un campo de la informática dedicado a capturar, procesar y analizar imágenes digitales con el propósito de replicar las capacidades del sistema visual humano. Este campo tiene varias aplicaciones, desde el reconocimiento de objetos hasta sistemas de vigilancia y robótica [56]. Las redes neuronales están usando el reconocimiento de patrones para distinguir objetos diferentes de una imagen, en

lugar de que el programador defina las características del objeto a identificar, la máquina aprende en base a imágenes cargadas [57]. En la Figura 11 se muestra un diagrama básico de visión artificial.

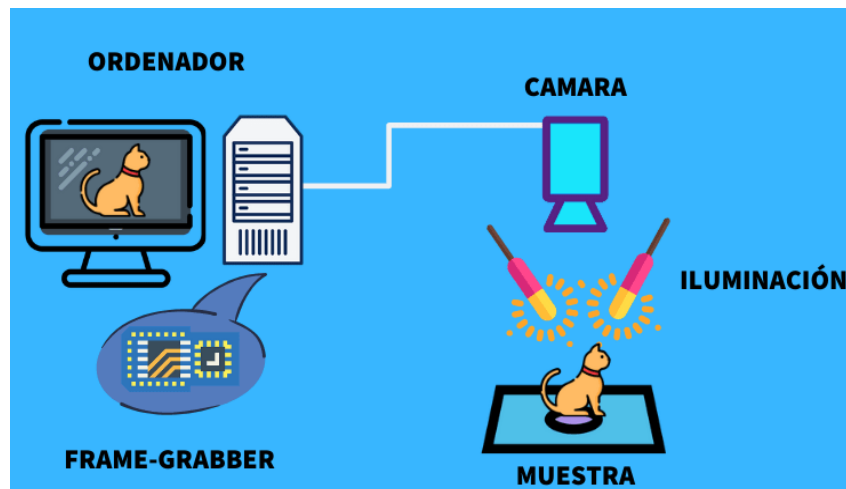


Figura 11. Diagrama básico de visión artificial [57].

d. Componentes de un sistema de visión artificial

Un sistema básico de visión artificial consta de dos elementos: son la recolección de imágenes, y su procesamiento. Para la adquisición, se necesita dispositivos de iluminación y cámaras que captaron las imágenes. Para el procesamiento se incorpora un digitalizador o capturador de imágenes y un software de procesamiento [57].

La configuración del sistema generalmente incluye [57]:

- Un elemento que ilumine el objeto o área a analizarse.
- Una cámara de video para capturar imágenes.
- Un digitalizador (frame-grabber) transforma señales analógicas en datos digitales, creando una imagen con resolución $N \times M$ píxeles.
- Un ordenador o microprocesador, para almacenar las imágenes y la ejecución de los cálculos utilizando software.
- Un monitor, para visualizar imágenes y los resultados del análisis.

e. Etapas de un proceso de visión artificial

Las etapas de un proceso de visión artificial involucran conceptos de software y hardware, además se consideran los avances teóricos en el área. Estas etapas representan los pasos fundamentales para realizar una tarea de visión artificial, como se ilustra en la Figura 12.

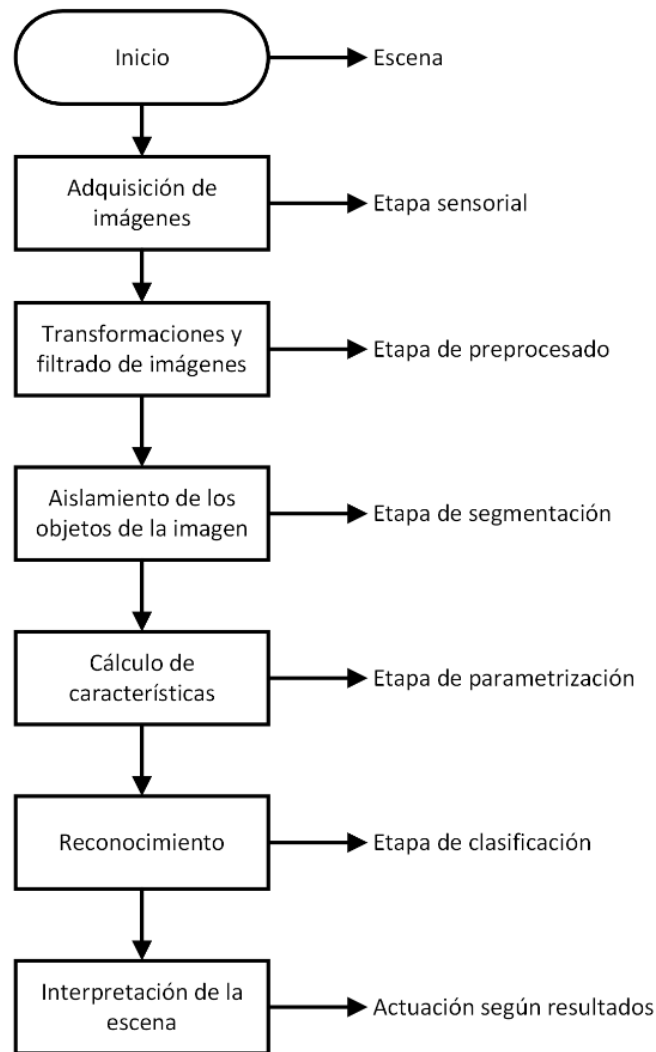


Figura 12. Etapas de un Sistema de Visión Artificial [58].

Las etapas del sistema de visión artificial se describen a continuación [58]:

- **Adquisición de la imagen digital:** Se obtiene una imagen lo más clara posible para asegurar el correcto procesamiento en el reconocimiento.

- **Digitalización de la imagen:** Una vez capturada la imagen digitalizada pasa por un preprocesamiento donde se optimiza y mejora la calidad para posteriores etapas.
- **Segmentación:** Es una de las fases que se encarga de dividir la imagen en secciones o fragmentos que la conforman.
- **Salida del proceso de segmentación:** Se produce una imagen de datos que contiene los puntos o la frontera de la región detectadas. Dependiendo del propósito, se tiene la representación por puntos empleada en examinar las propiedades internas del objeto, como la textura o estructura, y la representación por frontera para analizar características externas, como esquinas, concavidades y convexidades.
- **Parametrización:** Consiste en extraer características clave del objeto, proporcionando información cuantitativa y rasgos distintivos que diferencian a los objetos en la imagen.
- **Reconocimiento e interpretación:** Se asigna una etiqueta al objeto basado en la información obtenida, lo que permite clasificarlo.

f. Detección de objetos

Esta funcionalidad actúa como un "ojo" artificial capaz de localizar y reconocer elementos en imágenes o videos. Forma parte de la visión artificial enfocado en identificar la posición y la existencia de objetos determinados dentro de imágenes digitales. Esta técnica emplea algoritmos y metodologías de aprendizaje para analizar e identificar patrones en las imágenes y detectar objetos.

Los algoritmos hay que entrenarlos con amplios conjuntos de imágenes que contienen los objetos de interés, aprendiendo las características distintivas. Estos algoritmos funcionan como detectives, examinando cada imagen en busca de pistas para identificar los elementos, como se ilustra en la Figura 13.

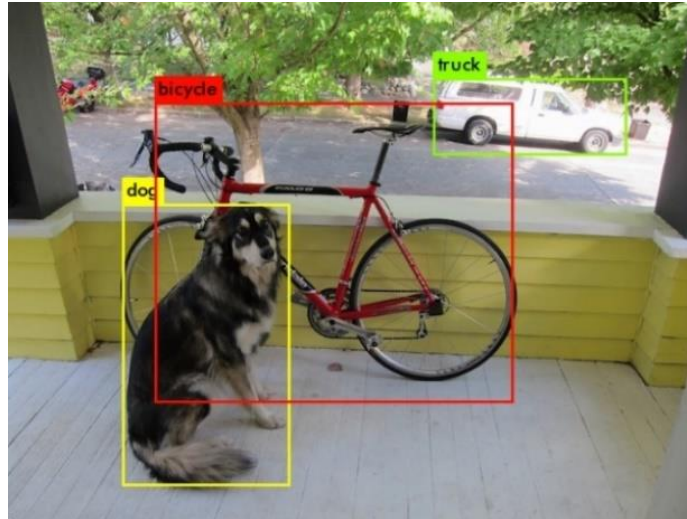


Figura 13. Detección de objetos [56].

El algoritmo busca regiones de interés, por ejemplo, zonas contiguas de tonos similares y bordes, agregándolas a una CNN y determina la clasificación que le corresponde. Con el proceso realizado se determina cuáles de las regiones contiene el objeto específico y su posición, como se muestra en la Figura 14 [59].

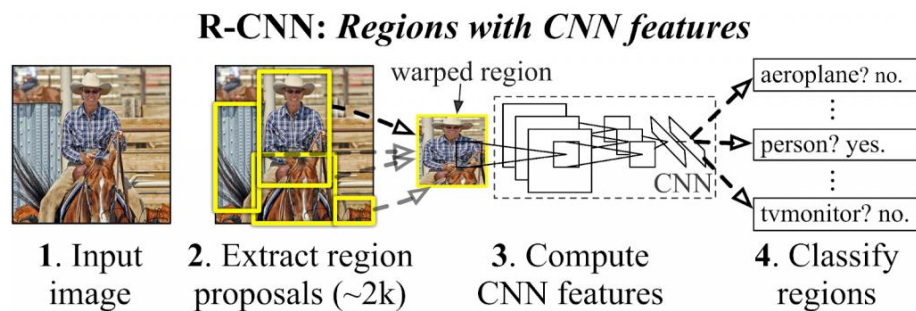


Figura 14. Regiones con características de CNN [59].

Con esta manera de detección de objetos se plantea un inconveniente con las regiones de interés en el caso que distintas áreas solapen al mismo objeto, para solucionar el inconveniente proponen las técnicas de IoU y NMS [59].

La intersección sobre unión (IoU) se encarga de calcular la relación existente entre el área de intersección de dos regiones y el área de unión que tienen, se interesa que mayor llegue a ser este valor, un ejemplo de esta técnica se presenta en la Figura 15

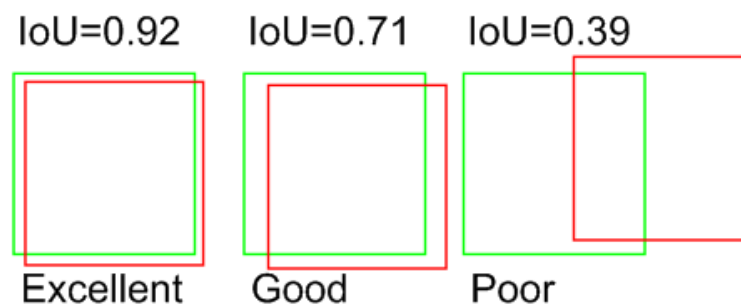


Figura 15. Intersección sobre unión (IoU) [59].

La supresión no máxima (NMS) se encarga de eliminar detecciones que se superponen, manteniendo únicamente la detección que tenga la puntuación más alta, como se muestra en la Figura 16.



Figura 16. Supresión no máxima (NMS) [59].

1.3.19 Deep Learning

El aprendizaje profundo, o Deep Learning (DL), ha adquirido una notable importancia en mundo actual, especialmente en el campo de la Inteligencia Artificial (IA). Por lo tanto, se considera un tema prioritario de investigación, y la gran parte de los avances en IA se enfocan en el Aprendizaje Profundo. Este tipo de aprendizaje es una rama específica del Aprendizaje Automático (Machine Learning) que puede entenderse como una evolución de las redes neuronales artificiales tradicionales [60]. Su estructura organizativa se muestra en la Figura 17.

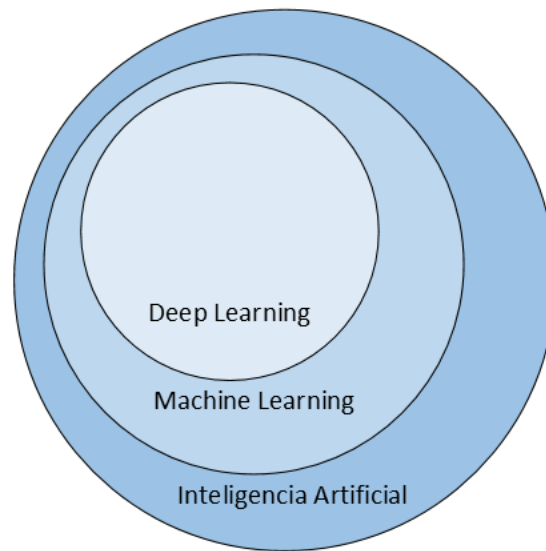


Figura 17. Estructura organizativa del campo de la Inteligencia Artificial [61].

A diferencia de las redes neuronales convencionales, que suelen estar formadas por una capa de entrada, una capa de salida y una capa oculta intermedia, las redes de Aprendizaje Profundo incorporan múltiples capas ocultas. La limitación en las capas de las redes tradicionales se debe al aumento en la complejidad y el esfuerzo computacional requerido. Sin embargo, los avances en algoritmos y el incremento en el poder de procesamiento han permitido crear redes profundas con varias capas ocultas, mejorando así su capacidad de análisis y procesamiento [60].

1.3.20 Redes Neuronales Convolucionales

La red neuronal convolucional (CNN) es una arquitectura diseñada para el procesamiento de datos con estructuras espaciales y temporales como imágenes, video, y series temporales, para la detección de objetos, segmentación y clasificación. En el tiempo se han creado variantes de CNN, como SSD, Faster R-CNN, YOLO, entre otras [48].

Elegir la arquitectura óptima de red neuronal para aplicaciones de visión artificial es fundamental, ya que deben cumplir con estándares de precisión y rapidez en el procesamiento de datos para que el sistema pueda detectar automóviles y placas de

manera eficiente. Las características de las principales redes neuronales convolucionales (CNN) consideradas se detallan en la Tabla 17.

Tabla 17. Comparación de Redes Neuronales Convolucionales [62], [63], [64].

Características	YOLO (You Only Look Once)	SSD (Single Shot MultiBox Detector)	Faster R-CNN
Velocidad	Muy rápida, ideal para aplicaciones en tiempo real	Rápida, casi comparable con YOLO	Lenta, debido a la generación de propuestas de región
Precisión	Alta, con mejoras constantes en últimas versiones	Alta, especialmente en detectar objetos de diferentes tamaños	Muy alta, considerada como un estándar de precisión
Uso de Recursos	Moderado, diseñado para análisis rápido y eficiente	Moderado, más eficiente que Faster R-CNN en tiempo real	Alto, dado a que computa propuestas de regiones
Complejidad	Baja, utiliza un sistema sencillo basado en cuadrículas.	Intermedia, debido al manejo de múltiples cajas y escalas.	Alta, incluye varias etapas para refinar las regiones propuestas.
Detección en Tiempo Real	Excelente debido a su velocidad	Muy buena, con capacidad de uso	No ideal, debido a su velocidad de procesamiento
Aplicaciones Típicas	Vigilancia, vehículos autónomos, robótica	Robótica móvil, sistemas IoT, aplicaciones embebidas.	Investigación académica, aplicaciones de alta precisión
Facilidad de Entrenamiento	Relativamente sencilla comparada con Faster R-CNN.	Requiere ajustar parámetros dependiendo del tamaño de los objetos	Más complicado, necesita ajustes específicos para las regiones propuestas

1.3.21 Red Convolucional YOLO

Es un modelo basado en redes neuronales convolucionales, YOLO (You Only Look Once) se encarga de la detección de objetos en imágenes y videos en tiempo real. En el sistema, la imagen es segmentada a regiones y, para cada una de ellas, el modelo se encarga de emitir las predicciones tanto de la caja delimitadora (bounding boxes) como de la confianza que cada clase de entrenamiento presente en la caja.

Simplemente, el modelo detectará el objeto si se encuentra en el centro de un cuadro de la matriz. La arquitectura YOLO presentada en la Figura 18 consta de 24 capas convolucionales, cuatro capas de agrupamiento máximo y dos capas totalmente conectadas. Para optimizar el proceso, se utilizan capas convolucionales de 1×1 de manera alternada, reduciendo el espacio de características de las capas anteriores [65].

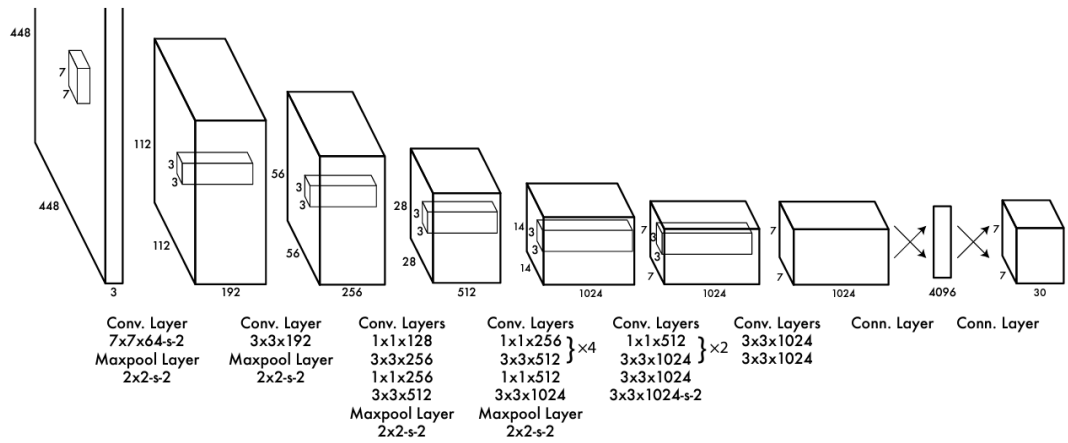


Figura 18. Arquitectura YOLO [65].

En la Figura 19 se presenta como la imagen procesada es dividida por YOLO en una matriz $S \times S$, reutilizando las técnicas anteriormente mencionadas como IoU y NMS, con la diferencia conceptual ya que realiza predicciones utilizando la imagen completa y no por regiones de interés. En el proceso del entrenamiento, se divide las imágenes en una cuadrícula aprendiendo que probabilidad hay de encontrar un objeto en las distintas celdas existentes [59].

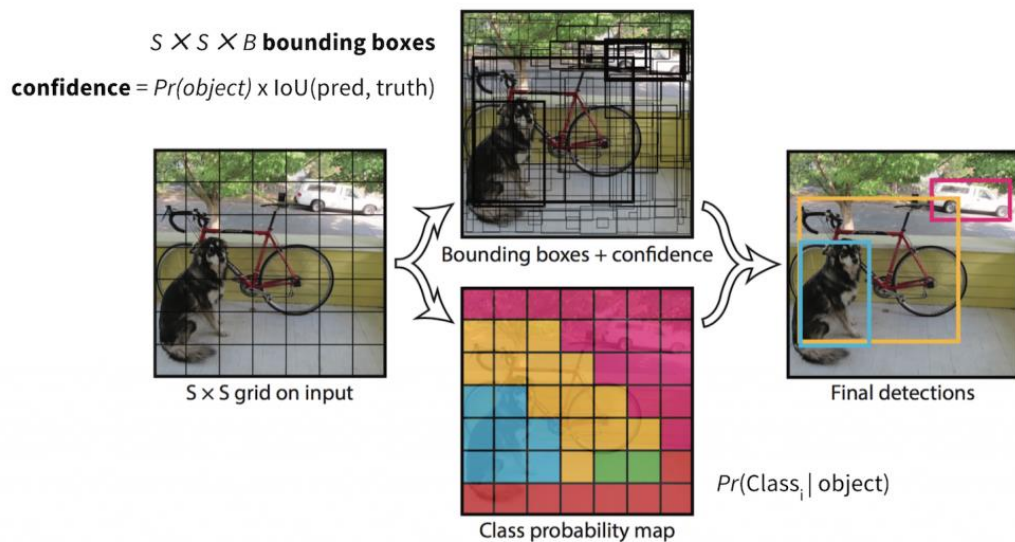


Figura 19. Detección de objetos con YOLO [59].

Las versiones de YOLO han ido variando, logrando importantes avances en la detección de objetos en imágenes y videos en tiempo real. Desde su lanzamiento inicial, YOLO ha tenido actualizaciones en cada versión, con mejoras en parámetros

como precisión, eficiencia y velocidad. También incorpora cambios en la arquitectura del modelo y nuevas técnicas de procesamiento, como se detalla en la Tabla 18.

Tabla 18. Comparación de versiones de YOLO [62].

Versión	Año de lanzamiento	Principales Mejoras	Descripción
YOLO	2015	Detección rápida y precisa	Primer modelo de detección y segmentación de objetos desarrollado por Joseph Redmon y Ali Farhadi en la Universidad de Washington
YOLOv2	2016	Incorporación de anclajes y normalización mejorada	Introdujo anclajes mejorando así la precisión y adoptó normalización por lotes para el entrenamiento
YOLOv3	2018	Manejo eficiente de objetos pequeños	Mejóro la detección de objetos de diversos tamaños y se implementó una arquitectura más robusta
YOLOv4	2020	Nuevos métodos de aprendizaje y preprocesamiento	Aumentó el rendimiento con técnicas como el Mosaic para mezclar imágenes y un mejor manejo de pérdida en el entrenamiento
YOLOv5	2020	Facilidad de uso y compatibilidad	Se enfocó en la usabilidad al permitir un entrenamiento más rápido y la exportación a formatos ampliamente compatibles
YOLOv6	2022	Optimización para sistemas en tiempo real	Adaptado especialmente para aplicaciones prácticas como robots autónomos
YOLOv7	2022	Ampliación de funcionalidades en IA	Extendió las capacidades del modelo al incluir tareas adicionales como la estimación de pose
YOLOv8	2023	Enfoque integral para visión artificial	Última versión de YOLO por Ultralytics, optimizado para una gama completa de tareas de IA, incluyendo detección, segmentación, estimación de pose, seguimiento y clasificación, mejorando la precisión y eficiencia en todas las áreas

1.3.22 YOLOv8

A inicios del año 2023, Ultralytics lanzó YOLOv8 con mejoras, respecto a sus versiones anteriores, convirtiéndolo en un modelo altamente preciso en el conjunto de datos COCO. Agrega nuevas funcionalidades y optimizaciones, transformándolo en una opción ideal para diversas tareas de detección de objetos en una amplia gama de aplicaciones [62].

La Figura 20 en su primera gráfica presenta en sus ejes horizontal y vertical respectivamente el número de parámetros del modelo (M, en millones) que indica el tamaño del modelo vs precisión medida como COCO mAP50–95 en el conjunto de validación, en cambio en la segunda gráfica se detalla en el eje horizontal latencia en milisegundos por imagen medida en una GPU NVIDIA A100 con TensorRT en FP16, haciendo referencia a qué tan rápido procesa el modelo una imagen vs la precisión medida como COCO mAP50–95 en la validación.

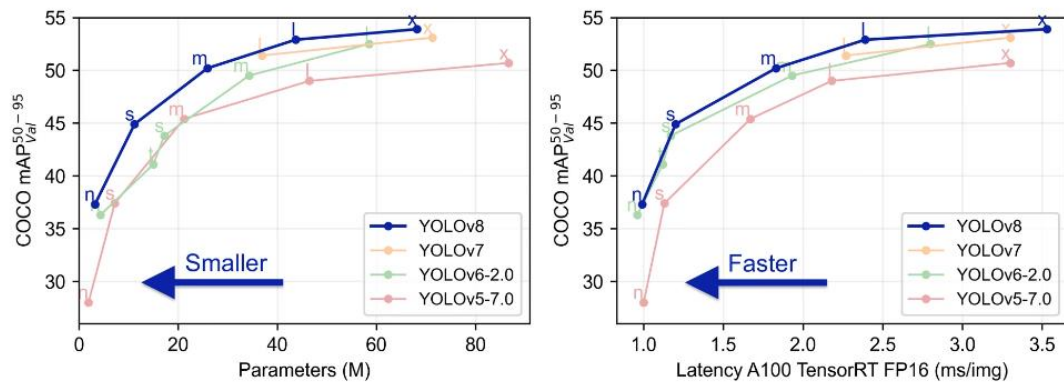


Figura 20. Precisión y Eficiencia entre Versiones de YOLO [62].

La Tabla 19 describe datos característicos de los modelos de YOLOv8 y hace comparación en términos de precisión (mAP), velocidad de inferencia, en CPU y GPU (A100), cantidad de parámetros, y operaciones de punto flotante (FLOP). Estas métricas permiten evaluar el equilibrio entre rendimiento y eficiencia de cada modelo en tareas de detección de objetos.

Tabla 19. Modelos de YOLOv8 [62].

Modelo	Tamaño (pixeles)	mAPval 50-90	Velocidad CPU ONNX (ms)	Velocidad A100 ONNX (ms)	Parámetros (M)	FLOP (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

1.3.23 Seguimiento de objetos

El seguimiento de un objeto permite identificar y rastrear su ubicación específica a lo largo de secuencias de video. Para hacer este proceso más manejable, se aplican simplificaciones que se ajustan a los requisitos de problemas específicos. En el caso de objetos con movimiento, es común asumir ciertas características sobre su dinámica [66].

Esto significa que el sistema de seguimiento debe poder adaptarse a variaciones visuales del objeto, como cambios en la iluminación, oclusiones parciales y movimientos erráticos. Para el proceso respectivo se han creado algoritmos de seguimiento de objetos que son fundamentales en aplicaciones de visión artificial, combinando técnicas de detección y asociación que predicen la posición de un objeto en movimiento. Entre las opciones más utilizadas están SORT, DeepSORT y ByteTrack, destacando por su precisión, velocidad y adaptabilidad a lenguajes de programación como Python, siendo ampliamente aplicados en tareas de vigilancia y monitoreo de tráfico, las características de los algoritmos propuestos se presentan en la Tabla 20.

Tabla 20. Algoritmos para seguimiento de objetos [67].

Característica	SORT	DeepSORT	ByteTrack
Algoritmo base	Filtro de Kalman	SORT + Red neuronal para generación de embeddings visuales	SORT + Algoritmo de asignación de detecciones múltiples y no asignadas
Precisión	Moderada	Alta, especialmente en objetos similares o complejos	Alta, muy precisa en escenas con múltiples objetos
Velocidad	Alta, adecuado para aplicaciones en tiempo real	Alta, pero más lenta que SORT debido a la red neuronal y embeddings	Muy alta, optimizado para mantener velocidad y precisión
Complejidad computacional	Baja	Media-Alta, debido a la carga computacional de redes profundas	Media, requiere manejar múltiples detecciones y asociaciones
Robustez ante oclusiones	Baja	Alta	Alta
Rendimiento en múltiples objetos	Moderado, adecuado para pocos objetos por vez	Alto, optimizado para manejar múltiples objetos similares	Muy alto, optimizado para escenarios con muchos objetos
Memoria requerida	Baja	Alta	Alta
Escalabilidad	Alta, adecuado para implementaciones con alta tasa de fotogramas	Moderada, depende del número de objetos	Alta, adecuado para implementaciones con alta tasa de fotogramas

Característica	SORT	DeepSORT	ByteTrack
Implementación en Python	Fácil, disponible en múltiples librerías y repositorios	Fácil, disponible en múltiples librerías y repositorios	Fácil, disponible en múltiples librerías y repositorios
Uso recomendado	Seguimiento en tiempo real de objetos con trayectorias predecibles	Seguimiento de objetos complejos y similares en entornos densos	Seguimiento en tiempo real de múltiples objetos en escenas densas

1.3.24 Reconocimiento óptico de caracteres (OCR)

El reconocimiento óptico de caracteres (OCR) es una tecnología que permite convertir imágenes que contienen texto, como fotografías de documentos, en un formato de texto legible por las máquinas para que pueda ser procesado y analizado. El OCR funciona en etapas de procesamiento: primero, adquiere la imagen contenedora de texto clasificando las áreas claras y oscuras como fondo y texto, respectivamente. Luego, en un proceso previo, limpia y corrige la imagen para optimizar su calidad, eliminando las manchas, ajustando la alineación y suavizando los bordes. Posteriormente, utiliza algoritmos como la coincidencia de patrones y la extracción de características para reconocimiento de caracteres y así convertirlos en texto digital. Finalmente, la información extraída se guarda en formatos computarizados para que sean procesados [68].

Dentro del ámbito de la programación, se han creado APIs potentes para facilitar la integración del OCR en aplicaciones personalizadas, como las proporcionadas por Google Cloud Vision API, Azure Cognitive Services y OCR Space. Estas herramientas permiten a los desarrolladores implementar soluciones avanzadas de OCR evitando el desarrollo de algoritmos desde cero para el reconocimiento de texto. En la Tabla 21 se presenta una comparación de APIs utilizadas en programación para el reconocimiento de caracteres.

Tabla 21. APIs para reconocimiento óptico de caracteres [47], [48], [49].

Características	Google Cloud Vision API	Azure Cognitive Services OCR	OCR Space
Precisión	~95% en imágenes complejas, incluyendo ruido y rotación.	~90% en documentos bien estructurados y con buena calidad.	~85% en imágenes simples sin ruido ni distorsiones.
Idiomas soportados	Más de 50 idiomas, incluyendo escritura compleja.	Más de 25 idiomas.	Más de 20 idiomas.
Capacidad de análisis	Reconoce texto en imágenes, documentos, señales, y más.	Diseñado principalmente para documentos.	Limitado a imágenes y documentos simples.
Método de Reconocimiento	Utiliza redes neuronales profundas para OCR avanzado, soportando imágenes con ruido o rotación.	Reconocimiento optimizado para texto en imágenes estructuradas (documentos planos).	Basado en algoritmos básicos de OCR sin corrección de ruido avanzada.
Procesamiento de imágenes	Procesa imágenes complejas, incluyendo fondos.	Bueno con imágenes de calidad media-alta.	Menos robusto con imágenes complejas.
Velocidad de procesamiento	Muy rápida con soporte en la nube escalable.	Rápida con servidores dedicados.	Velocidad dependiente de la carga del servidor.
Facilidad de uso	Bien documentada, con soporte para Python.	Bien documentada, con soporte para Python.	Documentación básica.
Costo	Gratuita para uso básico	Gratuita para uso básico	Gratuita para uso básico

1.3.25 Plataforma para entrenamiento de red neuronal

Para entrenar el modelo de inteligencia artificial encargado de la detección de placas de automóviles se requiere una de una plataforma en la nube. Estas herramientas web proporcionan servicios y recursos computacionales a través de internet, evitando dependencia de hardware o una infraestructura local. Entre las opciones disponibles de este tipo de plataformas son Kaggle, Microsoft Azure y Google Cloud Platform, cuyas características se describen en la Tabla 22.

Tabla 22. Comparación de plataformas basadas en la nube [72], [73], [74].

Características	Kaggle	Microsoft Azure	Google Cloud Platform
Entorno de Desarrollo	Kaggle Notebooks	Azure Notebooks, Azure Machine Learning Notebooks	Google Colab, AI Platform Notebooks
Servicios de Machine Learning	Soporte para bibliotecas de IA integradas	Azure Machine Learning	Google Cloud Machine Learning Engine

Características	Kaggle	Microsoft Azure	Google Cloud Platform
Frameworks de Aprendizaje	TensorFlow, PyTorch, Scikit-learn	TensorFlow, PyTorch, ONNX Runtime, entre otros	TensorFlow, PyTorch, Scikit-learn, entre otros
Integración con otras herramientas	Integración limitada a sus propios servicios de datos y notebooks	Integración con servicios de Azure como Azure Blob Storage, Azure Databricks, y Azure Kubernetes Service	Integración con servicios como Google Cloud Storage, BigQuery, y Google Kubernetes Engine
Acceso gratuito a GPU/ Colaboración en Tiempo real	Si	Si	Si
Almacenamiento en la nube	Almacenamiento limitado a datasets propios	Azure Blob Storage	Google Cloud Storage

1.3.26 Plataformas de servicios en la nube

Las plataformas en la nube ofrecen una serie de recursos informáticos que se pueden acceder a través de Internet, permitiendo a los usuarios utilizar aplicaciones, almacenar datos y realizar procesos sin tener que preocuparse por descargar software adicional [75].

En la Tabla 23 se detallan las plataformas de servicios en la nube más destacadas, permitiendo evaluar y contrastar las especificaciones técnicas de cada una.

Tabla 23. Características de plataformas con servicios en la nube [76].

Características	Microsoft Azure	AWS	Google Cloud Platform
Alcance Global	Presencia en múltiples regiones y zonas de disponibilidad en todo el mundo	La red más extensa de centros de datos y zonas de disponibilidad a nivel mundial	Infraestructura en crecimiento con centros de datos en diversas regiones globales
Integración	Excelente con productos y servicios de Microsoft, ideal para entornos Windows	Amplia compatibilidad con diversas tecnologías y sistemas operativos	Fuerte integración con servicios de Google y herramientas de código abierto
Precios	Estructura de precios competitiva, con opciones de pago por uso y descuentos por compromiso	Amplia variedad de opciones de precios, con modelos de pago por uso y reservas a largo plazo	Ofrece precios competitivos, con descuentos por uso prolongado y facturación por segundos
Facilidad de Uso	Interfaz intuitiva y herramientas de gestión basadas en la web	Requiere una inversión significativa para dominar sus herramientas y conocimientos técnicos	Considerado asequible en términos de administración, con una curva de aprendizaje moderada

Características	Microsoft Azure	AWS	Google Cloud Platform
Seguridad y Cumplimiento	Ofrece medidas de sólidas y cumplimiento con múltiples estándares y regulaciones	Amplias certificaciones de seguridad y cumplimiento en diversas industrias	Proporciona seguridad robusta y cumple con numerosos estándares internacionales

1.3.27 Gestión con servidores

a. Servidor Web

Un servidor web es un programa diseñado específicamente para gestionar la transferencia de datos de hipertexto, como páginas web que contienen diversos elementos, incluyendo textos, widgets y banners. Estos servidores funcionan utilizando el protocolo HTTP [77].

Los servidores web están instalados en las computadoras conectadas a Internet y permanecen en espera de recibir peticiones de navegadores web, como cuando un usuario intenta acceder a una página web. Al recibir la solicitud, el servidor responde enviando el código HTML a través de la red. De esta forma, el servidor web procesa las peticiones de los usuarios y envía los archivos requeridos, que pueden ser en formato de texto plano (como HTML o PHP) o en formato binario (como GIF o JPEG) [77].

b. Funcionalidades de un Servidor Web

Una de las funcionalidades esenciales de un servidor web es su capacidad para procesar una gran cantidad de solicitudes HTTP, incluyendo la de realizar múltiples tareas para gestionar varias solicitudes al mismo tiempo. Esto resulta crucial, ya que, si el servidor no puede gestionar adecuadamente las solicitudes más complejas, como las que implican consultas a bases de datos, el servicio podría sobrecargarse y colapsar, lo que afectaría tanto la disponibilidad como el rendimiento del sitio web [77].

Otra función esencial de un servidor web es establecer restricciones para proteger archivos que no deben estar disponibles para el público. Esto abarca gestionar la autenticación de los usuarios y filtrado de las solicitudes según su procedencia, lo que permite un control preciso sobre quién accede a la información y desde qué ubicación.

De esta manera, se protege los datos sensibles y se garantiza que solo personas autorizadas puedan acceder a ellos [77].

Además, la gestión de la información a transmitir en función de su formato es otra característica crucial. El servidor debe identificar el tipo de contenido solicitado y enviar los encabezados correctos para que el navegador sepa cómo gestionarlo. Esto permite que el contenido se interprete y visualice correctamente, garantizando que los distintos tipos de datos, como imágenes, texto y videos, se muestran de manera óptima y eficiente para el usuario final [77].

Finalmente, la gestión de logs es una tarea fundamental para cualquier servidor web. Consiste en guardar todas las solicitudes recibidas, los errores ocurridos y otra información relevante que pueda registrarse. Estos registros son esenciales para su posterior análisis, ya que brindan información relevante para generar estadísticas de acceso, identificar fallos recurrentes y optimizar la seguridad del servidor y la experiencia del usuario [77].

c. Servidores web más usados

La Tabla 24 ofrece una comparación a detalle de las características de los tres servidores web líderes: Apache HTTP Server, Microsoft IIS y Nginx. Son servidores clave de distribución y entrega en la que reside contenido web, aunque con diferencias en sus especificaciones por una lógica de ventajas.

Tabla 24. Características de los servidores web más usados [78].

Características	Apache HTTP Server	Microsoft IIS	Nginx
Licencia	Software libre y de código abierto	Propietario de Microsoft	Software libre y de código abierto
Compatibilidad de Sistemas Operativos	Multiplataforma: Windows, Linux, macOS, entre otros	Exclusivo para Windows	Multiplataforma: Windows, Linux, macOS, entre otros
Arquitectura	Basado en procesos o hilos, lo que puede consumir más recursos en cargas altas	Basado en hilos, optimizado para aplicaciones .NET	Basado en eventos, eficiente en el manejo de múltiples conexiones simultáneas
Manejo de Contenido Estático	Eficiente, pero puede ser superado por Nginx en rendimiento	Eficiente, especialmente en entornos Windows	Muy eficiente, diseñado para servir contenido estático rápidamente
Configuración	Archivos de configuración detallados y flexibles	Interfaz gráfica de usuario (GUI) para configuración	Archivos de configuración simples y directos

Características	Apache HTTP Server	Microsoft IIS	Nginx
Seguridad	Amplia comunidad y actualizaciones regulares	Integración con herramientas de seguridad de Windows	Comunidad activa y actualizaciones frecuentes
Escalabilidad	Escalable, pero puede requerir más recursos en cargas altas	Escalable en entornos Windows	Altamente escalable, diseñado para manejar miles de conexiones simultáneas

d. Base de datos en servidores

Una base de datos en un servidor es un sistema en línea donde la información se almacena y se organiza de una manera estructurada para ser accesible desde cualquier dispositivo conectado a través de una red. Esta base de datos se encuentra en un servidor en que se almacenan todos y cada uno de los datos de forma segura y centralizada. El acceso a la información se concibe a través de aplicaciones o interfaces de usuario, de modo que los diferentes participantes pueden hacer sus consultas y sus actos de inserción o modificación desde la distancia.

La Tabla 25 muestra una comparación de diferentes sistemas de gestión de datos específicamente SQLite, MySQL y MongoDB. La comparación aborda aspectos relevantes como el lenguaje utilizado y la compatibilidad con aplicaciones de Inteligencia Artificial.

Tabla 25. Comparación de base de datos [79], [80], [81].

Características	SQLite	MySQL	MongoDB
Tipo de DBMS	Embebido	Relacional	Relacional
Lenguaje de implementación	C	C, C++	C++
Lenguaje de consulta	SQL	SQL	MQL
Flexibilidad	Alta, no requiere servidor	Moderada, depende de configuración de servidor	Muy alta, permite esquemas flexibles
Escalabilidad	Adecuada para proyectos pequeños	Gestiona grandes volúmenes de datos	Excelente para manejo de big data
Uso de recursos	Bajo	Alto	Muy Alto
Compatibilidad	Funciona con Python y bibliotecas IA/ML	Compatible con herramientas de IA/ML	Se integra bien con frameworks de IA

1.3.28 Equipo computacional para programación

Se refiere al equipo físico utilizado como soporte para el desarrollo del software, el hardware requiere de especificaciones técnicas apropiadas para ejecutar herramientas profesionales, administrar gran cantidad de datos y garantizar la fluidez del desarrollo y depuración de sus aplicaciones, programas o sistemas que se requieran instalar.

Una computadora portátil permite el uso de lenguajes de programación correspondiente para la recepción de video y el procesamiento para la detección de automóviles, reconocimiento de placas y posterior envío de información a la base de datos, además es compatible con software como entornos de programación, configuración de vuelo de dron, grabación de video, etc. La comparación técnica presentada en la Tabla 26 trata de las características de los equipos computacionales propuestos para la incorporación al sistema.

Tabla 26. Características de ordenadores portátiles [82], [83], [84].

Características	Lenovo IdeaPad 330	Dell G5 15 Gaming Laptop	HP Pavilion Gaming 15
			
Procesador (CPU)	Intel Core i5-8250U (8ª Gen, 4 núcleos, 1.60 GHz, Turbo hasta 3.4 GHz)	AMD Ryzen 7 4800H (8 núcleos, 2.9 GHz, Turbo hasta 4.2 GHz)	Intel Core i7-8750H (6 núcleos, 2.2 GHz, Turbo hasta 4.1 GHz)
Gráficos	Intel UHD Graphics 620 (integrado)	AMD Radeon RX 5600M (GPU dedicada)	NVIDIA GeForce GTX 1060 (GPU dedicada, 6 GB GDDR5)
Memoria RAM	8 GB DDR4	8 GB DDR4	8 GB DDR4
Almacenamiento	256 GB SSD	512 GB PCIe SSD	1 TB HDD + 16 GB Intel Optane
Pantalla	15.6" FHD (1920x1080)	15.6" FHD (1920x1080), 144 Hz	15.6" FHD (1920x1080)
Sistema Operativo	Windows 10 Home (64 bits)	Windows 10 Home (64 bits)	Windows 10 Home (64 bits)
Conectividad	Wi-Fi, Bluetooth	Wi-Fi 6, USB-C, HDMI, Bluetooth	Wi-Fi, HDMI, Bluetooth
Peso Aprox.	2.2 kg	2.5 kg	2.4 kg

1.4 Objetivos

1.4.1 Objetivo general

Construir un sistema para detección de automóviles robados empleando visión artificial en vehículo aéreo no tripulado (UAV).

1.4.2 Objetivos específicos

- Determinar las tecnologías y herramientas utilizadas en las técnicas de recuperación de vehículos robados.
- Seleccionar la tecnología, algoritmos y dispositivos que puedan ser aplicados en el sistema de detección inteligente.
- Implementar un sistema inteligente para la localización de vehículos robados utilizando visión artificial.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

A lo largo de este proyecto de investigación, se emplearon diferentes dispositivos y materiales con el objetivo de desarrollar un sistema de localización de vehículos robados específicamente en parqueaderos. En la Tabla 27 se presenta un listado detallado de los materiales utilizados para alcanzar este propósito.

Tabla 27. Materiales y dispositivos.

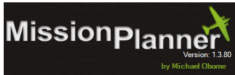
Materiales	Descripción	Imagen
Cámara CMOS Orange	Utilizada para obtener video en tiempo real facilitando la adquisición de imágenes de los vehículos.	
Cuadricóptero SpeedyBee F405 V3	Empleado como soporte de la cámara FPV con el propósito de sobrevolar las trayectorias establecidas.	
Batería LiPo 4S 14.8 V-16.8 V	Alimentación del UAV	
GPS Matek Systems M10Q05883	Proporciona navegación precisa, ubicación y telemetría en UAV para misiones autónomas	
Cargador balanceador imax B6AC	Permite cargar la batería LiPo 4S	

Materiales	Descripción	Imagen
Video transmisor (VTX 801)	Utilizado para transmitir en tiempo real el video emitido por la cámara FPV.	
Video receptor (VRX ROTG01)	Usado para recibir en tiempo real el video emitido por el transmisor.	
Antena Foxxer Lollipop	Permite la comunicación entre transmisor y receptor.	
Computador HP con tarjeta gráfica NVIDIA GEFORCE GTX	Utilizado para el procesamiento de imágenes y envío de información a la base de datos.	
Radio control (Jumper T-pro)	Utilizado para el control del UAV.	

En la Tabla 28 se detallan los softwares y servicios de aplicación utilizados en el proyecto. Estos incluyen herramientas esenciales para el desarrollo de interfaces, el manejo de bases de datos, la creación de páginas web, y la comunicación eficiente, así como plataformas para el entrenamiento neuronal y la gestión de contenido visual. Cada software y servicio ha sido seleccionado por sus características y funcionalidades específicas, que facilitan la implementación y ejecución de tareas clave dentro del proyecto.

Tabla 28. Softwares y servicios de aplicación.

Materiales	Descripción	Imagen
Python	Lenguaje de programación empleado para comunicación con la base de datos, desarrollo de interfaz gráfica, envío de información hacia Telegram y correo electrónico. Para la programación de las funciones se utilizó librerías adicionales.	
Matlab	Empleado para realizar simulaciones de las trayectorias	
OBS	Captura el video y lo graba en tiempo real	
Roboflow	Para etiquetado de imágenes y descarga de dataset	
Kaggle	Plataforma para el entrenamiento de red neuronal	
HTML	Empleado para estructurar y diseñar página web	
JavaScript	Utilizado para crear interactividad y funcionalidad avanzada en el sitio web.	
CSS	Utilizado para dar estilo que define la apariencia visual y diseño de la página web.	

Materiales	Descripción	Imagen
PHP	Utilizado como lenguaje de programación de la página web	
Mission Planner	Empleado para desarrollar vuelos programados mediante trayectorias, utilizando un mapa para ubicar los puntos de altitud y latitud.	
MySQL	Utilizado como sistema de gestión de bases de datos	
Azure	Plataforma en la nube de Microsoft utilizada para montar una máquina virtual capaz de recibir, almacenar y enviar información.	
OCR con Google AI	Mediante una API extrae texto y datos de imágenes	
Cloudinary	Plataforma utilizada para la gestión de imágenes, transformación y entrega de contenido visual.	
Telegram	Aplicación de mensajería instantánea, utilizada para comunicación global entre los administradores y usuarios.	
Gmail	Servicio de correo electrónico personal.	
Visual Studio Code	Editor de código fuente ligero y potente, con soporte para múltiples lenguajes y extensiones personalizables.	

2.2 Métodos

2.2.1 Modalidad de la investigación

Las modalidades presentadas en el proyecto de investigación son las siguientes:

El proyecto aplicó investigación de tipo aplicada, ya que el enfoque principal se centra en establecer un sistema de visión artificial para la detección de placas vehiculares utilizando un UAV, aplicando conocimientos avanzados sobre la electrónica, programación, redes, campos de la Inteligencia Artificial, comunicaciones avanzadas, entre otros.

También, se empleó la modalidad bibliográfica documental, debido a que se fundamentó en investigaciones, tesis, artículos científicos, entre otros, encontrados en repositorios universitarios, bases de datos y revistas de manera libre y gratuita.

La investigación de igual manera fue de campo, dado a que se enfocó en entornos reales, en este caso en parqueaderos de la Universidad Técnica de Ambato campus Huachi, para recopilar información sobre las placas vehiculares y verificar el funcionamiento del sistema.

Por último, se efectuó investigación experimental para determinar la confiabilidad del sistema para el reconocimiento de caracteres de placas de los vehículos, mediante una serie de pruebas pertinentes en el monitoreo y control de los sistemas de detección y localización, evaluando de esta manera la eficiencia del proyecto con los resultados obtenidos.

2.2.2 Recolección de información

La información para la realización del proyecto se obtuvo de tesis, trabajos de investigación, artículos, libros y demás fuentes bibliográficas gratuitas que hay en repositorios de universidades y revistas, que aportaron en la investigación sobre visión artificial para la detección de automóviles, placas vehiculares, reconocimiento de caracteres y utilización de un dron en el sistema.

Además, para el entrenamiento del modelo de reconocimiento de placas, se utilizó un dataset creado en la plataforma Roboflow, y para la detección de vehículos el dataset de Coco definido en YOLOv8, modelos que fueron probados con videos grabados por la cámara incorporada al dron y receptados en la PC. Estos criterios de recolección se presentan en la Tabla 29.

Tabla 29. Criterios de recolección de información.

CRITERIO	DESCRIPCIÓN
¿Para qué?	Optimización del tiempo de respuesta para la detección de vehículos robados
¿De qué personas u objetos?	De los vehículos que se encuentran estacionados en un parqueadero.
¿Cómo?	Mediante un sistema de visión artificial para identificar automóviles con placas vehiculares
¿Con qué?	Con una cámara y transmisor de video incorporados en un dron para monitoreo.
¿Cuándo?	Durante el intervalo de septiembre – diciembre 2024
¿Dónde?	En los parqueaderos de la Universidad Técnica de Ambato, campus Huachi

2.2.3 Procesamiento y análisis de datos

Para llevar a cabo esta investigación con el procesamiento y análisis de datos, se consideró las siguientes pautas:

- Recolección de información detallada.
- Revisión de información obtenida.
- Análisis de los fundamentos teóricos relacionados con las variables del estudio.
- Selección de la información óptima para el desarrollo de la investigación.
- Pruebas para verificar el funcionamiento del sistema.
- Análisis de los datos obtenidos durante las pruebas.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

3.1 Desarrollo de la propuesta

El presente proyecto se centra en la utilización de drones para crear una solución innovadora de un sistema que utiliza visión artificial para la identificación de automóviles robados. Un escenario que utiliza tecnología moderna para contrarrestar las ineficiencias en la detección de placas vehiculares para la localización de vehículos, facilitando la recuperación de estos. Mejorando el sistema para detección de vehículos y optimizando el tiempo para que el proceso sea más rápido y efectivo. A continuación, se detalla cómo se llevó a cabo el desarrollo de este sistema inteligente, explicando cada paso de la metodología empleada.

3.1.1 Esquema general del sistema

El esquema general del sistema de reconocimiento de placas vehiculares y detección de automóviles se compone de tres etapas fundamentales, en donde se logra capturar, receptar, visualizar y procesar las imágenes que se obtiene a través de la cámara FPV del dron en un ordenador determinado, en la Figura 21 se presenta cada una de las etapas.

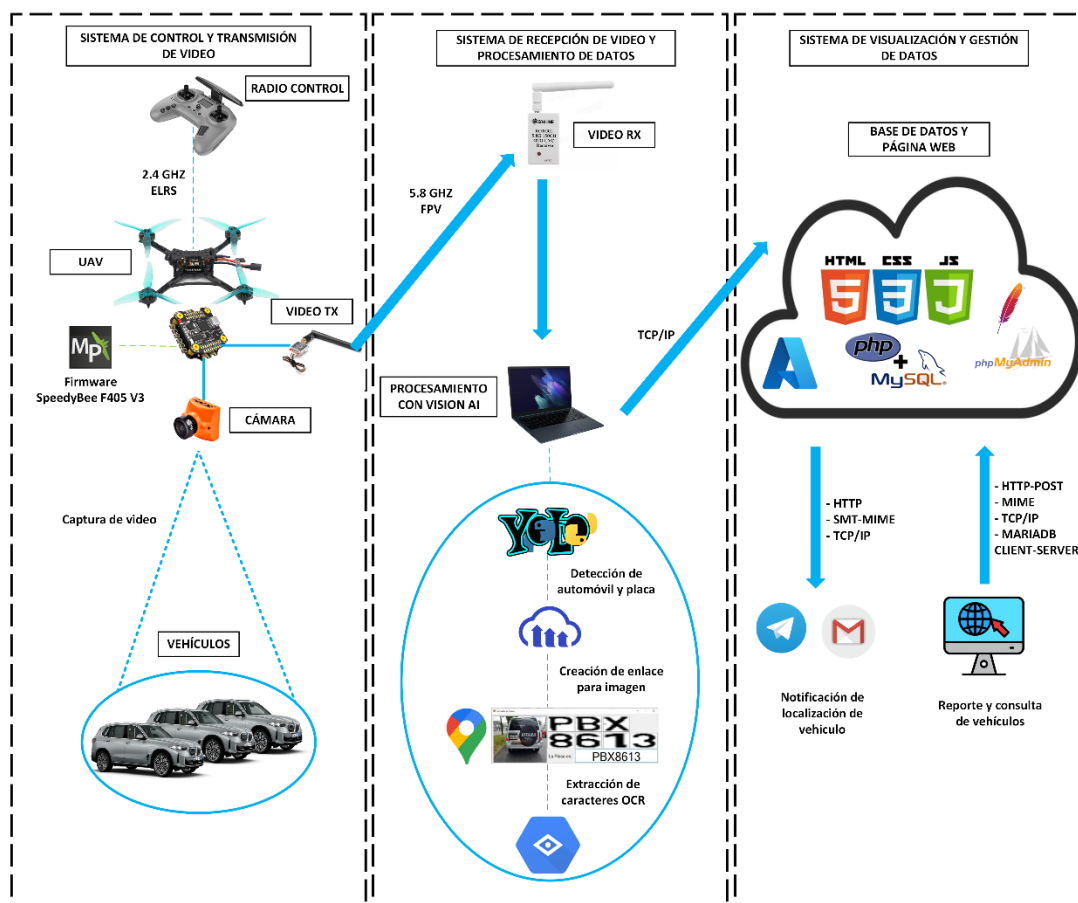


Figura 21. Esquema general del sistema.

En la etapa uno se detalla la captura del video mediante la cámara FPV desde la fuente de transmisión de video conectada a la placa de control, esta señal de video es capturada y enviada por una señal analógica dirigida hacia el transmisor de video, para después ser transmitida mediante radiofrecuencia a 5.8 GHz por medio de un módulo de transmisión de video, frecuencia a la cual trabaja este tipo de tecnología, esta será captada en una estación terrestre. Además, se presenta que el dron contiene una tarjeta, con el firmware de Mission Planner instalado y es operado con el radio control a través de la frecuencia 2.4 GHz para maniobras de vuelo.

La segunda etapa comprende la recepción y digitalización del video mediante un receptor inalámbrico tipo ROTG01, que transforma la señal analógica proveniente del transmisor en una señal digital apta para su visualización y procesamiento en el computador, aplicando visión artificial para la detección de vehículos y placas en las imágenes recibidas. Con la tecnología OCR se extraen caracteres obteniendo los datos de placas y ubicación del vehículo, adicional se genera un enlace que contiene la

imagen capturada para que la información sea enviada a la base de datos donde se almacena.

En la tercera etapa, los eventos relacionados con el reconocimiento de vehículos en los parqueaderos se almacenan en la base de datos, verificando si algún vehículo coincide con los registros de vehículos robados proporcionados por los usuarios. En caso de encontrar una coincidencia, se genera una alerta que se envía tanto a un grupo de Telegram de usuarios como al correo electrónico del propietario del vehículo identificado. Si no se encuentra ninguna coincidencia, la información se guarda únicamente en la base de datos.

Además, a través de una página web, los usuarios pueden registrar la información de sus vehículos en caso de pérdida. Estos datos serán almacenados en la base de datos para facilitar su posterior localización. La plataforma también ofrece la posibilidad de realizar consultas para verificar si un vehículo ha sido localizado. En caso de que el vehículo esté identificado, el sistema mostrará en el sitio web la placa, una fotografía y la ubicación en donde fue detectado, siempre que dicho vehículo haya sido previamente registrado en la base de datos.

3.1.2 Herramientas utilizadas para la recuperación de vehículos

Para cumplir con el primer objetivo específico de determinar las tecnologías y herramientas utilizadas en las técnicas de recuperación de vehículos robados, se realizó un análisis detallado de las principales soluciones disponibles. Se identificaron dos enfoques clave: las tecnologías de rastreo y los sistemas de cámaras. En la Tabla 2 en la primera categoría, se evaluaron dispositivos GPS, que permiten la localización en tiempo real mediante satélites, y la tecnología RFID, utilizada para el control de acceso y seguimiento en puntos estratégicos. En cuanto a los sistemas de cámaras presentados en la Tabla 3, se exploró el uso de ALPR, que automatiza el reconocimiento de matrículas mediante inteligencia artificial, y las cámaras CCTV, que permiten la vigilancia continua. A través de la comparación técnica de estas herramientas, se establecieron sus ventajas, limitaciones y aplicaciones, proporcionando una base sólida para la localización y seguimiento de vehículos robados para posterior recuperación según la información obtenida de las herramientas determinadas.

3.1.3 Selección de tecnologías

Mediante un análisis se seleccionaron elementos óptimos en características técnicas para la aplicación del sistema de detección inteligente, mediante el uso de tablas comparativas se evalúa cada dispositivo, dando así cumplimiento al segundo objetivo específico planteado en el proyecto.

a. Vehículo aéreo no tripulado (UAV)

Para el sistema de detección de automóviles y placas se ha seleccionado el dron Cuadricóptero SpeedyBee FPV con el estándar ExpressLRS, sus características técnicas se presentan en la Tabla 7 considerando su programabilidad y ligereza, siendo la aeronave de menor peso y por ende posee una mayor autonomía.

b. Batería LiPo

La batería seleccionada es la LiPo Tattu Funfly presentada en la Tabla 11, optimizando una combinación equilibrada ya que tiene la mayor capacidad (1550 mAh), con un voltaje (14.8 V, 4 celdas), siendo adecuada para drones que requieren potencia sostenida ya que el tipo de dron seleccionado requieren picos de potencia frecuentes para maniobras rápidas. Una batería de alta descarga (100 C) y descarga máxima de ráfaga (200 C) garantiza que los motores tengan la energía necesaria para no comprometer el rendimiento. Aunque es pesada (168 g), su rendimiento superior compensa este factor, haciéndola ideal para drones de carrera. También se considera que los motores del dron seleccionado Cuadricóptero SpeedyBee FPV soportan baterías LiPo de 3 celdas a 5 celdas como se detalla en la Tabla 7.

c. Dispositivo de captura de video

La cámara CMOS Orange presentada en la Tabla 12 es una buena opción para aplicaciones FPV, debido a que cuenta con un sensor que ofrece 1200 TVL, esto garantiza los detalles en las imágenes que la cámara capture. Su diseño robusto garantiza la durabilidad y compatibilidad con sistemas de transmisión, siendo un dispositivo de confiabilidad y rendimiento en diversas condiciones de luz.

d. GPS

El GPS Matek Systems M10Q05883 presentado en Tabla 10, es una opción optima debido a su diseño compacto que facilita su integración al UAV, tanto en espacio y peso. Ofreciendo compatibilidad con múltiples sistemas de navegación satelital con una alta sensibilidad de seguimiento, además su protocolo de datos UBX y NMEA lo hace ideal para aplicaciones que requieren precisión y confiabilidad en la navegación autónoma.

e. Transmisor de video

Los sistemas de transmisión y recepción que se utilizan en los UAV's requieren del uso de un video transmisor (VTX) que se encuentra montado en la aeronave y su función es enviar la señal de video analógica que captura la cámara a una estación con un receptor de video (VRX). [55] El transmisor Eachine-TX801 de la Tabla 13 es la mejor opción entre los presentados por el tipo de conector que utiliza permitiendo la adaptabilidad a la tarjeta controladora, además el módulo transmisor es comercializado en Ecuador. En el Anexo B se describen los componentes fundamentales que constituyen el transmisor de video Eachine-TX801, con un detalle específico de cada uno de ellos.

f. Radio control

El radio control T-Pro ELRS de la Tabla 9 se eligió por su compatibilidad con el estándar ExpressLRS (ELRS), lo que garantiza un rendimiento destacado en cuanto a alcance y baja latencia. Su configuración es sencilla y el diseño está pensado para ser accesible tanto para técnicos como pilotos. El estándar ELRS se destaca por su capacidad de mantener un rango de comunicación de hasta 40 Km, dependiendo del módulo utilizado y del estado de las baterías del radio y del receptor en el UAV. Además, su diseño está orientado a maximizar la eficiencia y compatibilidad, convirtiéndolo en una herramienta versátil para manejar diferentes tipos de UAV y sus componentes. El Anexo C proporciona una descripción detallada sobre la utilidad de cada parte del radio control T-Pro ELRS.

g. Receptor de video

Para ver el video transmitido por el VTX capturado con la cámara incorporada al dron, es necesario tener un receptor de video (VRX) para conectarlo a la computadora y procesar la señal de video. En este caso, se emplea la tecnología OTG de los presentados en la Tabla 14, el receptor Eachine ROTG01 ofrece características superiores, como un mejor rango de frecuencias para la utilización de canales y mayor potencia, lo cual es fundamental para lugares en el que es difícil acceder, además la salida de video es tipo micro USB en comparación al modelo RC305 que tiene incorporado a la salida un conector jack de 3.5 mm, que necesita una capturadora de video para conectarla a la computadora y obtener el video transmitido, por tal motivo se selecciona el modelo ROTG01 para esta aplicación.

h. Ordenador portátil

La portátil HP Pavilion Gaming presentada en la Tabla 26, es la más adecuado para las aplicaciones que utilizan visión artificial, en este caso para la detección de vehículos y placas. Debido a que cuenta con una tarjeta gráfica NVIDIA GeForce GTX 1060, aprovechando las tecnologías CUDA para acelerar el proceso con modelos de inteligencia artificial. Por otro lado, la GPU NVIDIA permite realizar procesos de visión artificial eficientemente en comparación con las tarjetas gráficas AMD que no son óptimas para este tipo de aplicaciones, ya que NVIDIA es ampliamente reconocida ofreciendo una mejor integración con herramientas de desarrollo como YOLO y OpenCV. En el Anexo D se presentan las especificaciones técnicas de la tarjeta gráfica NVIDIA GeForce GTX 1060.

i. Software controlador de vuelo

Mission Planner descrito en la Tabla 8 es una herramienta que adicional al vuelo manual permite la planificación de una misión autónoma avanzada, vuelos con GPS, mapeo y misiones a largo plazo; se adecua a una completa interfaz y a la compatibilidad con los sensores externos, de utilidad para drones empleados en rescate, agricultura, vigilancia o investigación.

j. Entorno de desarrollo integrado

Visual Studio Code en la Tabla 16 destaca como la mejor alternativa entre los entornos de desarrollo, por la versatilidad, soporte para múltiples lenguajes de programación; Python, JavaScript y HTML; su integración con herramientas de control de versiones tal es el caso de Git, la interfaz gráfica intuitiva y ampliación de funciones mediante extensiones, lo hacen apto para trabajar proyectos de inteligencia artificial.

k. Software de programación

A partir de la Tabla 15, Python fue seleccionado como el lenguaje de programación para el sistema, aprovechando su robustez en el manejo de visión artificial y aprendizaje automático. Con librerías como OpenCV y Ultralytics, Python facilita las tareas de procesar imágenes tomadas por el dron y la implementación de modelos de redes neuronales. Además, incorpora herramientas que posibilitan manejar grandes cantidades de datos, lo cual es clave para el procesamiento en tiempo real.

l. Algoritmo de seguimiento de objetos

SORT es una elección óptima con sus características de la Tabla 20, empleado en aplicaciones de mayor velocidad y eficiencia computacional. Su simplicidad y bajo consumo son ideales para sistemas en tiempo real, comparado con DeepSORT o ByteTrack no es robusto en condiciones complejas, pero en el proyecto desarrollado su alto rendimiento y bajo costo computacional lo hacen ideal para tareas con pocos objetos o en escenarios donde la rapidez es un parámetro por considerar.

m. API para integrar OCR

Google Cloud Vision API es la opción adecuada para tareas de OCR debido a su alta precisión en escenarios complejos, como imágenes con ruido, rotación o baja calidad. La incorporación de redes neuronales profundas garantiza un reconocimiento avanzado y confiable, las características de la herramienta API se presentan en la Tabla 21.

n. Plataforma basada en la nube para entrenamiento de red neuronal

Se ha elegido Kaggle en base a las características presentadas en Tabla 22, debido a sus características computacionales, además ofrece acceso gratuito de 30 horas semanales a sus recursos de alto rendimiento, como GPUs y TPUs, esenciales para entrenar e implementar modelos de visión artificial y aprendizaje profundo de manera eficiente.

o. Arquitectura de visión artificial

Las arquitecturas como YOLO (You Only Look Once) y SSD (Single Shot MultiBox Detector) de la Tabla 17 permiten realizar aplicaciones de detección de objetos en tiempo real, dado a su capacidad de procesar imágenes de forma rápida y con precisión. Se seleccionó YOLO, que aborda la detección de objetos como un problema de regresión, pasando de la imagen completa a las coordenadas de los cuadros delimitadores y las probabilidades de las clases, permitiendo una detección rápida para usos como la vigilancia por video.

YOLOv8 presentado en la Tabla 18 es la opción ideal para un sistema de detección de placas vehiculares, ya que ofrece un enfoque integral para visión artificial, funciona de manera eficiente en tiempo real y es eficaz para identificar detalles en imágenes.

p. Plataforma en la nube

Microsoft Azure, como se describe en la Tabla 23, fue seleccionado debido a su extensa variedad de servicios, entre los cuales destacan las soluciones híbridas que permiten soportar escenarios locales. Esto es crucial para proyectos que necesitan mucho trabajo integrado entre entornos en nube y locales físicos. Asimismo, su modelo de precios competitivo, pagando solo por lo que se utilizó, ofrece ahorros económicos importantes mejorando la administración de recursos. En este sentido, estas capacidades aliadas a la accesibilidad a una interfaz amigable y acceso a las herramientas de Administración desde la web hacen que la gestión de todo el sistema sea adecuadamente simplificada. Por último, proporciona medidas de seguridad y cumplimiento, alineadas normativas internacionales, garantizando la protección confiable de los datos.

q. Servidor web

El servidor web Apache descrita en la Tabla 24 se eligió debido a en su capacidad multiplataforma, que permite una implementación sencilla en sistemas operativos como Windows, Linux y macOS, brindando una gran versatilidad y adaptabilidad. Gracias a su arquitectura basada en procesos o hilos, resulta particularmente robusto para manejar cargas de trabajo moderadas, como la configuración de bases de datos mediante herramientas como phpMyAdmin. Además, Apache sobresale por su compatibilidad con múltiples lenguajes de programación a través de módulos, una característica clave para integrar y gestionar de manera eficiente aplicaciones web conectadas a bases de datos. Por último, su flexibilidad en la configuración permite ajustar y optimizar el entorno según los requisitos específicos del proyecto, consolidándose como la mejor alternativa frente a servidores como Microsoft IIS y Nginx para este caso en particular.

r. Base de datos

MySQL presentada en la Tabla 25, es una excelente opción para gestionar los datos en un sistema de detección basado en visión artificial, debido a la adaptabilidad de montar un servidor externo en una máquina virtual en Azure. Esto es ideal en escenarios con hardware limitado, donde se considera la eficiencia y facilidad de uso. Además, MySQL se integra de manera eficiente con Python y sus bibliotecas. El consumo bajo de recursos y la adaptabilidad con otras aplicaciones asegura su desempeño.

3.1.4 Diseño de conexiones electrónicas del sistema de transmisión y captura de video

La SpedyBee F405 V3 es una placa fundamental para configurar correctamente el sistema eléctrico que conecta la cámara FPV y el transmisor de video (VTX). En el Anexo E, se presenta tanto la placa controladora como la placa ESC, siendo la primera responsable del manejo y supervisión de todo el sistema, mientras que la placa ESC se encarga específicamente de regular las velocidades de los motores. En el caso de la cámara CMOS Orange 1200, se utilizan tres cables principales: uno amarillo para transmitir la señal de video conectado al pin "CAM", uno rojo que proporciona la

alimentación desde el pin "5V", y uno negro destinado a la conexión a tierra, identificado como "G".

El VTX Eachine TX801, por su parte, también utiliza tres cables: dos destinados a la alimentación y uno amarillo encargado de la señal de video. Estas conexiones deben realizarse directamente a través de la SpedyBee F405 V3, lo que permite no solo una integración física eficiente, sino también el acceso a información relevante a través del OSD (On Screen Display). Este sistema facilita la adquisición de video, datos geográficos como la latitud y longitud, para la correcta localización del vehículo. El esquema correspondiente se detalla en la Figura 22.

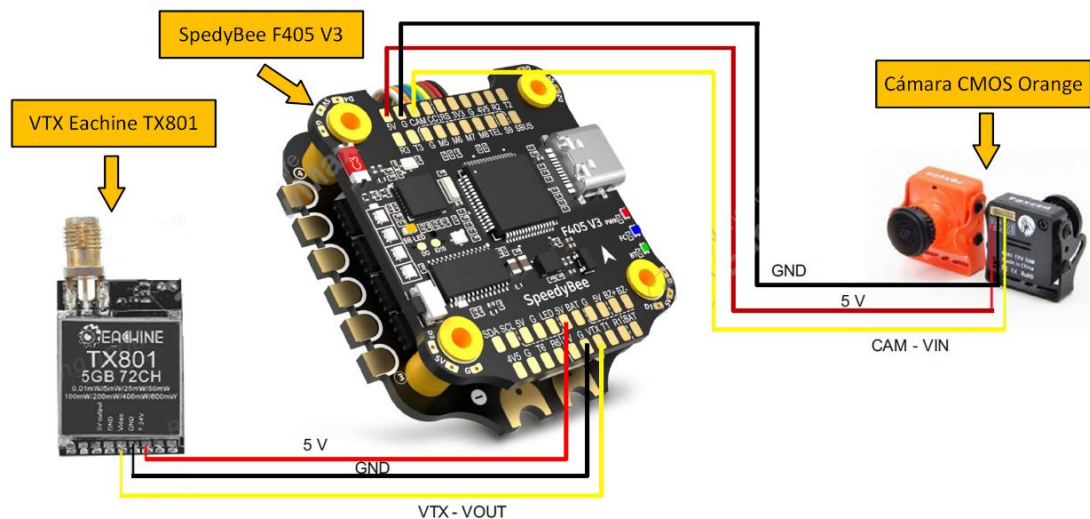


Figura 22. Diagrama de conexión de la cámara FPV.

3.1.5 Diseño de conexiones electrónicas del módulo GPS

Para establecer la conexión entre la placa SpedyBee F405 V3 y el módulo GPS, se identificaron los seis pines del módulo: el pin "5V", conectado mediante un cable rojo al pin "4V5" que proporciona alimentación; el pin "G", conectado con un cable negro a tierra; el pin "RX", enlazado con un cable amarillo al pin "T6", que funciona como entrada de datos serie; el pin "TX", conectado mediante un cable morado al pin "R6", destinado a la salida de datos serie; el pin "CL", enlazado con un cable verde al pin "SCL" para la línea de reloj en comunicación I2C; y finalmente, el pin "DA", conectado con un cable blanco al pin "SDA", encargado de la línea de datos para la

comunicación I2C. Este esquema de conexión, mostrado en la Figura 23, facilita tanto la navegación precisa como el posicionamiento automatizado del dron.

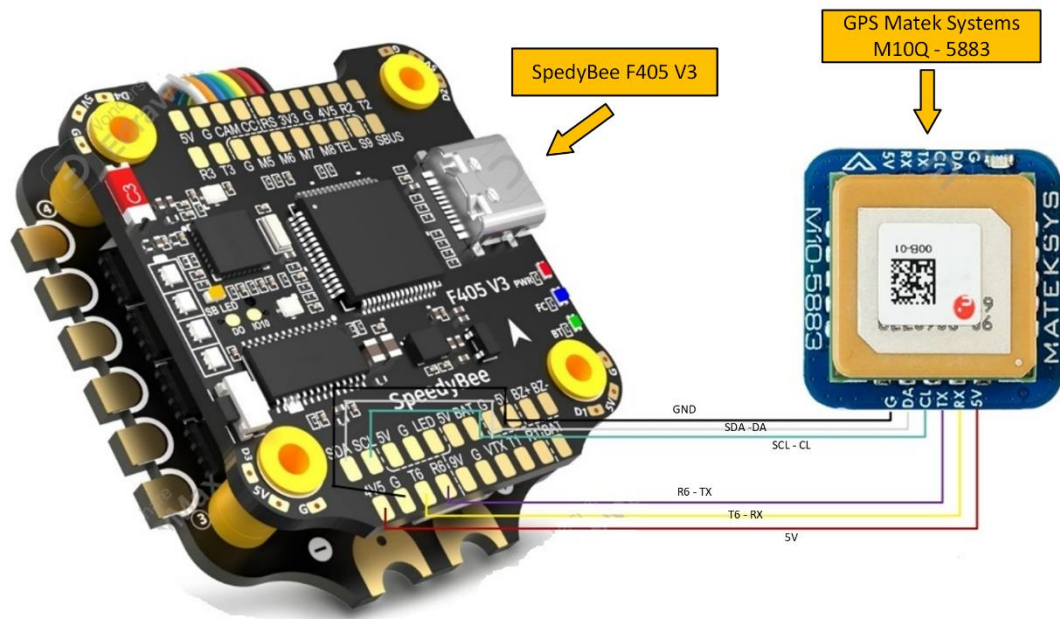


Figura 23. Diagrama de conexión del módulo GPS.

3.1.6 Implementación del sistema de transmisión, antena y captura de video en el UAV

Para la implementación de las partes se requiere de las conexiones respectivas entre el VTX Eachine TX801, antena Foxeer Lollipop y la placa SpedyBee F405 V3, además las conexiones entre la cámara CMOS Orange y la placa SpedyBee F405 V3, en la Figura 24 se visualiza el esquema eléctrico implementado en el UAV.

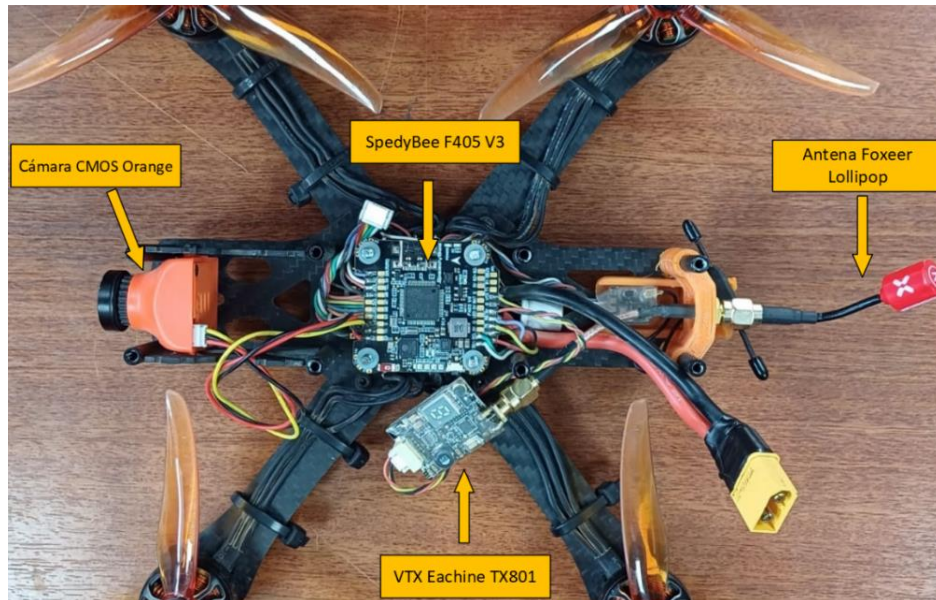


Figura 24. Implementación del sistema de transmisión y captura de video.

3.1.7 Implementación del módulo GPS en el UAV

Para la implementación del módulo se identificó correctamente cada uno de los pines y el funcionamiento de cada uno, para luego realizar las conexiones respectivas entre el GPS Matek Systems M10Q-5883 y la placa SpeedyBee F405 V3. En la Figura 25 se visualiza el esquema electrónico implementado en el UAV para la integración del módulo GPS.

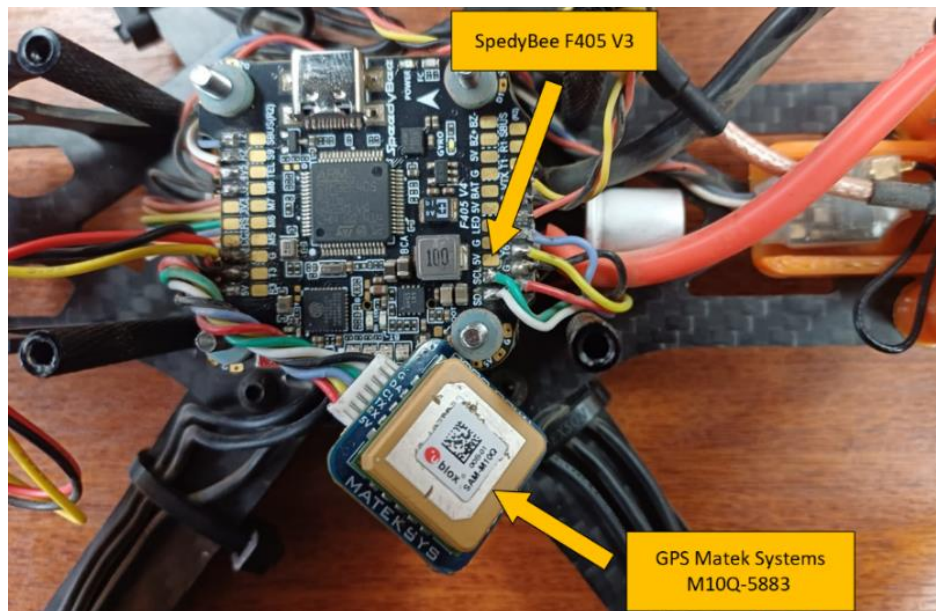


Figura 25. Implementación del módulo GPS en el UAV.

3.1.8 Peso del UAV

En esencial determinar con precisión el peso total del UAV, especialmente considerando las restricciones aplicables a los drones de carrera, cuyo límite máximo permitido es de 2 Kg. Para este propósito, se utilizó una balanza análoga, como se muestra en la Figura 26, registrando un peso total de 600 g.



Figura 26. Peso total del UAV.

3.1.9 Consumo de energía y tiempo de vuelo

El consumo de energía de los elementos del UAV se basó en los parámetros técnicos de cada componente implementado. Este estudio permitió seleccionar una batería LiPo acorde con las recomendaciones técnicas establecidas. En la Tabla 30 se presentan detalladamente las características de la batería para facilitar su evaluación.

Tabla 30. Características de la batería LiPo.

Características	Valor
Número de celdas	4 S
Voltaje por celda	3.7 V – 4.2 V
Voltaje total	14.8 V – 16.8 V
Potencia	22.94 Wh
Corriente	1550 mAh
Tasa de descarga	100 C

La Ecuación (1) representa la capacidad de la batería LiPo, siendo (Q) la capacidad nominal, E la energía de almacenamiento watts-hora y V el Voltaje nominal.

$$Q = \frac{E}{V} \quad (1)$$

$$Q = \frac{22.94}{14.8}$$

$$Q = 1.55 \text{ Ah}$$

La corriente máxima de descarga continua (I_{max-DC}) puede determinarse considerando la capacidad nominal de la batería, que en este caso es de 1550 mAh, y su tasa de descarga (C) de 100 C. mediante la Ecuación (2).

$$C = \frac{I_{max-DC}}{Q} \quad (2)$$

$$100 = \frac{I_{max-DC}}{1.55}$$

$$I_{max-DC} = 100 * 1.55$$

$$I_{max-DC} = 155 \text{ A}$$

La capacidad de esta batería permite suministrar una corriente continua máxima de 155 A cuando se utiliza con una tasa de descarga de 100 C. Este valor corresponde a una batería ideal para aplicaciones que requieren altas tasas de descarga, como drones, vehículos eléctricos y sistemas energéticos de alta eficiencia.

Considerando el peso del UAV, que es de 600 g, y las especificaciones de la batería LiPo, el dron debe generar un empuje equivalente ($600 \text{ g} \approx 5.9 \text{ N}$). Este empuje se distribuye uniformemente entre los cuatro motores, requiriendo aproximadamente 1.475 N por motor. Para estimar el tiempo de vuelo, se emplea la Ecuación (3), donde

(Q) es la capacidad nominal de la batería y (I_A) la corriente consumida por cada motor. En esta configuración, con hélices de 5 pulgadas y motores de 2650 KV, la corriente consumida por cada motor varía entre 2 A y 5 A. En condiciones de vuelo estacionario, donde no se realizan maniobras agresivas, el consumo por motor se reduce a un rango de 2 A y 3 A.

$$t = \frac{Q}{I_A * 4} \quad (3)$$

Con $I_A = 2$ A

$$t = \frac{1.55 \text{ Ah}}{(2 * 4) \text{ A}} = 0.193 \text{ h} = 11.62 \text{ minutos}$$

Con $I_A = 3$ A

$$t = \frac{1.55 \text{ Ah}}{(3 * 4) \text{ A}} = 0.129 \text{ h} = 7.74 \text{ minutos}$$

El tiempo de vuelo estimado puede calcularse aproximando el consumo total de corriente de los motores, el cual está estrechamente relacionado con el peso del UAV y las condiciones climáticas con un intervalo de tiempo entre 7.74 minutos a 11.62 minutos. En escenarios donde se presentan ráfagas de viento, los motores requieren mayor energía para compensar el impacto del entorno, lo que resulta en una reducción significativa del tiempo de vuelo.

3.1.10 Simulación de un modelo cinemático en el software Matlab para ejecución de planes de vuelo

El modelo cinemático tiene como propósito determinar tanto la velocidad lineal como angular a la que se mueve el robot, además de identificar su posición y orientación en el instante inicial del UAV. En la simulación, el dron se representa como un punto ubicado en el plano cartesiano R. La Figura 27 muestra el marco inicial del UAV utilizando las coordenadas X_q , Y_q y Z_q .

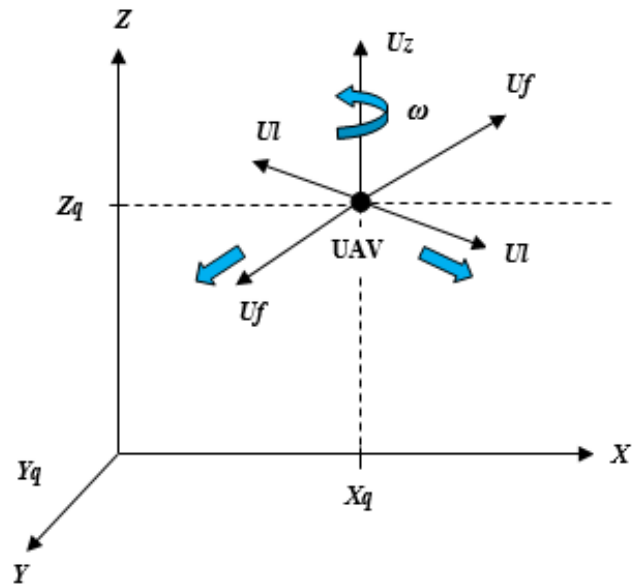


Figura 27. Marco Inercial de UAV [86].

Descripción:

X_q = Posición inicial respecto al eje X

Y_q = Posición inicial respecto al eje Y

Z_q = Posición inicial respecto al eje Z

U_f = Velocidad hacia al frente X

U_l = Velocidad lateral

U_z = Velocidad vertical

ω = Velocidad angular

Una vez identificadas la ubicación y el desplazamiento del UAV en los planos cartesianos, se inicia la simulación de los planes de vuelo empleando las coordenadas correspondientes. El proceso comienza diseñando un cuadricóptero en Autodesk Fusion como se observa en la Figura 28, ajustado a las propiedades y dimensiones del dispositivo real. Este diseño se emplea posteriormente en Matlab para simular el comportamiento del dron.

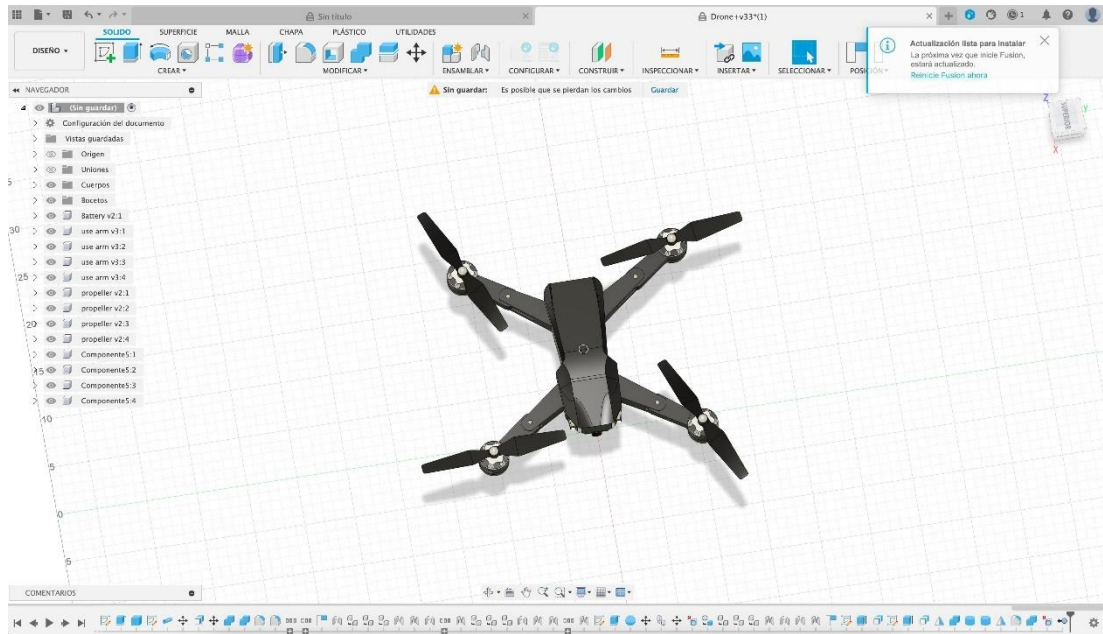


Figura 28. Diseño del cuadricóptero en el software FUSION 360.

Se exportó el diseño del UAV en formato STL y, a través del software Matlab, se creó una función específica, mostrada en la Figura 29. Esta función permite importar el modelo en formato STL y guardarlo en una variable global para su posterior simulación.

```
function uavRobot
    global uav
    partel = stlread('DronPlacas.stl');
    % Obtener los vertices y caras
    uav.partelVertices = partel.Points';
    uav.partelFaces = partel.ConnectivityList;
end
```

Figura 29. Declaración de variables.

Se implementó una función complementaria para controlar los movimientos de rotación y traslación de la aeronave en los ejes x, y, z . La Figura 30 presenta una representación detallada de esta función.

```

function uavGraph uavPlot(x1, y1, z1, phi, theta, psi, scale)
    global uav

    % Matrices de rotación
    Rz = [cos(psi), sin(psi), 0; sin(psi), cos(psi), 0; 0, 0, 1];
    Ry = [cos(theta), 0, sin(theta); 0, 1, 0; sin(theta), 0, cos(theta)];
    Rx = [1, 0, 0; 0, cos(phi), sin(phi); 0, sin(phi), cos(phi)];

    % Transformar los vértices
    robotPatch = (Rz * Ry * Rx * uav.partelVertices)';

    % Escalar y trasladar
    robotPatch(:, 1) = robotPatch(:, 1) * scale + x1;
    robotPatch(:, 2) = robotPatch(:, 2) * scale + y1;
    robotPatch(:, 3) = robotPatch(:, 3) * scale + z1;

    % Crear el gráfico del dron
    uavGraph(1) patch('Faces', uav.partelFaces, 'Vertices', robotPatch, ...
        'FaceColor', 'b', 'EdgeColor', 'none');
end

```

Figura 30. Declaración de la función para el control del movimiento de rotación y traslación.

Para implementar las funciones, se desarrolló un script adicional, como se muestra en la Figura 31, el cual facilita el control y seguimiento de las trayectorias planificadas. Este enfoque aplica una ley de control que posiciona el UAV en coordenadas específicas x, y, z y con una orientación deseada.

El UAV, al poseer capacidades omnidireccionales, puede moverse en cualquier dirección y ajustar sus ángulos de inclinación: roll, pitch y yaw. Se establece un tiempo de muestreo típico de $t_s = 0.1$ segundos, común en robótica, y un vector de tiempo que abarca desde 0 hasta un máximo de 10 segundos, con incrementos de 0.1 segundos en relación con el eje central del dron. Inicialmente, tanto la posición en x, y, z como la orientación del UAV se configura en cero.

```

clc
clear
close all

% 1 tiempo
tf = 60;
ts = 0.1;
t = 0:ts:tf;
N = length(t);

% 2 CONDICIONES INICIALES
% Centro de masa
%->Variables de posicion y orientacion:
hx = zeros(1,N+1);
hy = zeros(1,N+1);
hz = zeros(1,N+1);
yaw = zeros(1,N+1);

%->Asignacion de condiciones iniciales:
hx(1) = 1.5;
hy(1) = 0;
hz(1) = 0;
yaw(1) = 0; % Orientacion inicial

% 3 POSICIONES DESEADAS (lista de puntos
desired_positions = [
    4, 3, 4, 0 * (pi/180);
    4, 6, 4, 90 * (pi/180);
    0, 6, 1.5, 180 * (pi/180)
];

```

Figura 31. Declaración de condiciones iniciales y posiciones deseadas.

Se especifican las referencias deseadas para que el UAV se desplace desde su punto inicial hacia objetivos establecidos en x, y, z . El propósito del control es calcular las posiciones y velocidades necesarias para que el UAV alcance los puntos deseados. Así, al establecer una posición objetivo en x, y, z el algoritmo de control genera las velocidades precisas para que el robot logre llegar a dichas coordenadas.

En la Figura 32 se establece la configuración del entorno visual para monitorear el movimiento del cuadricóptero. Este entorno permite personalizar diversos aspectos, como el color y tamaño de la aeronave, la velocidad de visualización, las dimensiones de los ejes y la orientación de la figura. Además, se incluye la representación del contorno de los ejes mediante cuadros y se generan etiquetas para identificarlos con claridad.

```

uf = zeros(1,N);
ul = zeros(1,N);
uz = zeros(1,N);
w = zeros(1,N);

hxe = zeros(1,N);
hye = zeros(1,N);
hze = zeros(1,N);
yawe = zeros(1,N);

% 6 SIMULACION
scene = figure;
tam = get(0,'ScreenSize');
set(scene, 'position', tam);
set(scene, 'Color', 'white');
set(gca,'FontWeight','bold');
axis equal;
axis([-10 10 -10 10 0 8])
view([-125 30]);
grid on;
box on;

xlabel('x (m)');
ylabel('y (m)');
zlabel('z (m)');

camlight('headlight');
scale = 0.005;

uavRobot;
M1 = uavPlot(hx(1), hy(1), hz(1), 0, 0, yaw(1),
scale);
hold on
M2 = plot3(hx(1), hy(1), hz(1), 'r', 'LineWidth', 2);

```

Figura 32. Configuración del entorno visual para el monitoreo del movimiento del cuadricóptero.

El bucle principal se encarga de seguir los puntos deseados, ajustando continuamente las posiciones del dron. Por otro lado, la matriz jacobiana establece la relación entre las velocidades de entrada u_f, u_l, u_z, ω y las variaciones en la posición y orientación del UAV. La matriz diagonal K , que contiene las ganancias proporcionales de velocidad, se define como la inversa de la matriz jacobiana. Esta ley de control es aplicable a cualquier UAV y permite al dron alcanzar una posición final específica, en la Figura 34 se muestra la codificación mencionada.

```

% Marcar los puntos deseados con color azul
for i = 1:size(desired_positions, 1)
    plot3(desired_positions(i, 1), desired_positions(i, 2),
        desired_positions(i, 3), '0b', 'LineWidth', 2);
end

for i = 1:size(desired_positions, 1)
    hxd = desired_positions(i, 1);
    hyd = desired_positions(i, 2);
    hzd = desired_positions(i, 3);
    yawd = desired_positions(i, 4);

    for k = 1:N
        % Error (diferencias entre las posiciones/orientaciones
        % deseadas y las actuales.)
        hxe(k) = hxd - hx(k);
        hye(k) = hyd - hy(k);
        hze(k) = hzd - hz(k);
        yawe(k) = yawd - yaw(k);

        he = [hxe(k); hye(k); hze(k); yawe(k)];

        % Jacobiano (Relaciona velocidades de control (uf,ul,uz,w) con
        % las velocidades del robot en el sistema de coordenadas global.)
        J = [cos(yaw(k)), -sin(yaw(k)), 0, 0; ...
            sin(yaw(k)), cos(yaw(k)), 0, 0; ...
            0, 0, 1, 0; ...
            0, 0, 0, 1];

        % Parametros de control (Matriz de ganancia proporcional. Aumentar
        % valores acelera el movimiento hacia el objetivo.)
        K = eye(4);

        % Ley de control (Calcula las velocidades de control)
        qpRef = pinv(J) * (K * he); % pinv(J) Inversa pseudo de la matriz Jacobiana.

        % Aplicar al robot
        uf(k) = qpRef(1);
        ul(k) = qpRef(2);
        uz(k) = qpRef(3);
        w(k) = qpRef(4);
    end
end

```

Figura 33. Matriz jacobiana, matriz diagonal K y ley de control.

Además, se implementa la lógica de actualización de la orientación y posición de un UAV utilizando un modelo cinemático diferencial centrado en su centro de masa. La orientación, representada por el ángulo de guiñada (yaw), se ajusta iterativamente en función de la velocidad angular y el tiempo de muestreo (ts). A partir de esta orientación, se calculan las nuevas posiciones en los ejes x, y y z mediante proyecciones trigonométricas basadas en las velocidades controladas. Además, el algoritmo incluye una verificación en cada iteración para determinar si el UAV ha alcanzado el objetivo deseado, considerando una tolerancia definida. En caso de

cumplimiento, se actualizan las condiciones iniciales para continuar hacia el siguiente objetivo. La Figura 34 muestra detalladamente el código implementado.

```

        yaw(k+1) = yaw(k) + ts * w(k); %Actualiza el angulo de orientacion
                                   %usando la velocidad angular.

        %->Calcula las nuevas posiciones usando las velocidades controladas.
        %Modelo cinematico diferencial en el centro de masa
        xlp = uf(k) * cos(yaw(k+1)) - ul(k) * sin(yaw(k+1));
        ylp = uf(k) * sin(yaw(k+1)) + ul(k) * cos(yaw(k+1));
        zlp = uz(k);

        %Posiciones en el centro de masa
        hx(k+1) = hx(k) + ts * xlp;
        hy(k+1) = hy(k) + ts * ylp;
        hz(k+1) = hz(k) + ts * zlp;

        % Actualizar la simulacion
        delete(M1)
        M1 = uavPlot(hx(k), hy(k), hz(k), 0, 0, yaw(k), scale);
        plot3(hx(1:k), hy(1:k), hz(1:k), 'r', 'LineWidth', 2);
        pause(ts)

        % Verificar si el UAV ha alcanzado el punto deseado
        if norm([hxe(k), hye(k), hze(k)]) < 0.1 && abs(yawe(k)) < 0.1
            % Actualizar las condiciones iniciales para el siguiente punto
            hx(1) = hx(k);
            hy(1) = hy(k);
            hz(1) = hz(k);
            yaw(1) = yaw(k);
            break;
        end
    end
end
disp('El UAV ha alcanzado todos los puntos deseados.');
```

Figura 34. Actualización del ángulo de orientación, cálculo de nuevas posiciones y verificación del punto deseado.

Utilizando las referencias ingresadas junto con los ángulos respectivos asociados a cada una, en la Figura 35 se genera la simulación correspondiente, mostrando las posiciones deseadas y la trayectoria recorrida por el UAV.

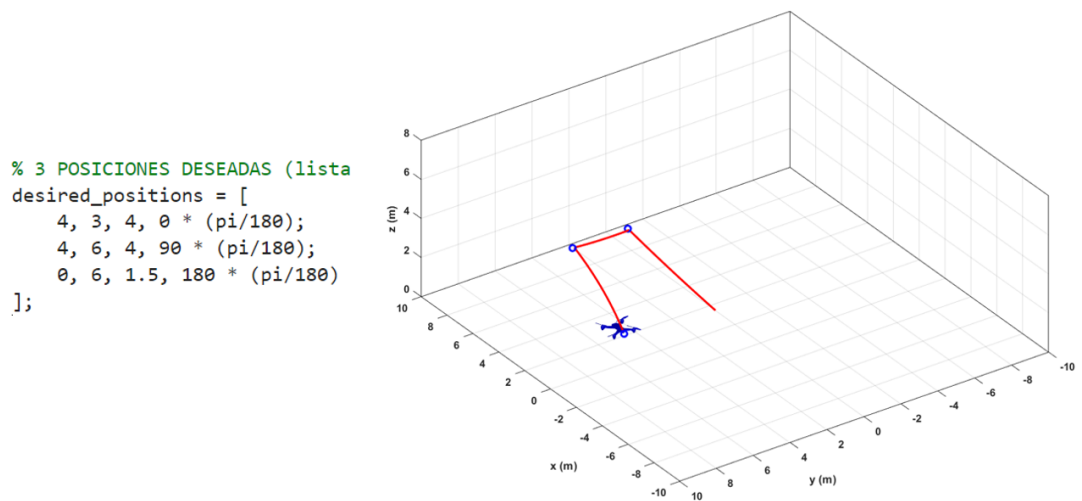


Figura 35. Simulación de la posición del UAV con tres puntos de referencias y ángulos de inclinación de 0 grados, 90 grados y 180 grados.

3.1.11 Configuración del UAV

La elección de ArduPilot para la configuración de vuelo se debe a su accesibilidad y a la facilidad con la que permite realizar ajustes específicos para aplicaciones autónomas y personalizadas. Al ser un software gratuito, representa una ventaja significativa para quienes buscan una herramienta eficiente sin incurrir en costos adicionales.

En la Figura 36 se presenta el diagrama de flujo que explica el proceso del funcionamiento del UAV. En las siguientes secciones se describen detalladamente las configuraciones para el vuelo manual y autónomo.



Figura 36. Diagrama de flujo del proceso del funcionamiento del UAV.

a. *Firmware de controladora de vuelo y parámetros iniciales*

Controlar y coordinar los elementos electrónicos del UAV depende directamente del firmware, que actúa como el cerebro del sistema. Se encarga de recopilar los datos proporcionados por los sensores, procesarlos y distribuir las instrucciones necesarias

para que los componentes electrónicos operen eficientemente. Además, garantiza la estabilidad del UAV durante el vuelo, permitiendo realizar las maniobras requeridas de manera efectiva.

El UAV ocupa una placa SpeedyBee F405 V3, su firmware se encuentra en la página oficial de Ardupilot, donde se descarga el archivo .hex para cargar en la controladora de vuelo. En este caso, para hacer el proceso de flasheo se utilizó el software INAV el cual tiene funciones similares a la de Mission Planner, pero con una con un enfoque menos avanzado en sus configuraciones.

El proceso de configuración comenzó con la descarga e instalación del software Mission Planner 1.3.83 desde el sitio oficial de Ardupilot, como se ilustra en la Figura 37.



Figura 37. Interfaz principal del software Mission Planner de ArduPilot.

Una vez ejecutado el software, se procedió a vincular el UAV con la computadora. En el apartado de SETUP, hay que seleccionar el tipo de frame que tiene el UAV, en este caso es de 4 rotores y tipo X como se muestra en la Figura 38.

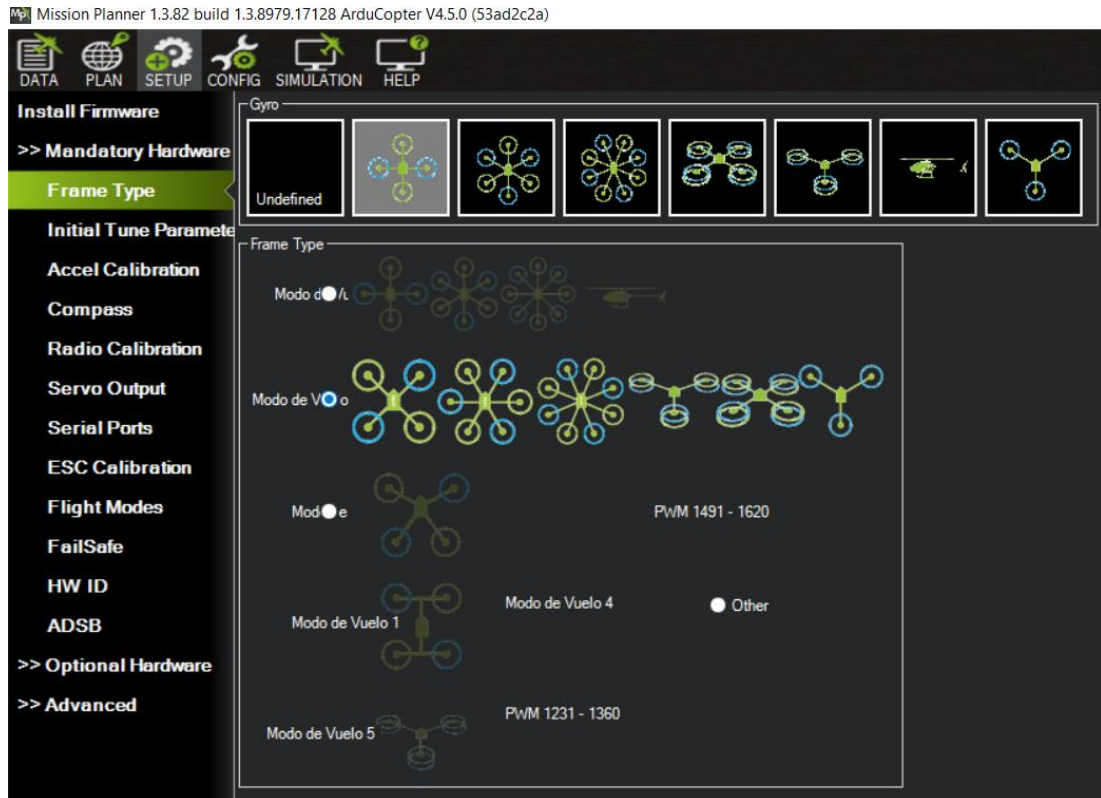


Figura 38. Selección del tipo de frame del UAV.

En los parámetros de tiempo inicial se completa los datos de acuerdo con la información del UAV y elementos seleccionados, en este caso el tamaño de las hélices es de 5 pulgadas, la batería LiPo que tiene 4 celdas con el voltaje máximo de 4.20V y voltaje mínimo de 3.30V. Calculando así los parámetros iniciales para optimizar el vuelo del UAV presentados en la Figura 39.

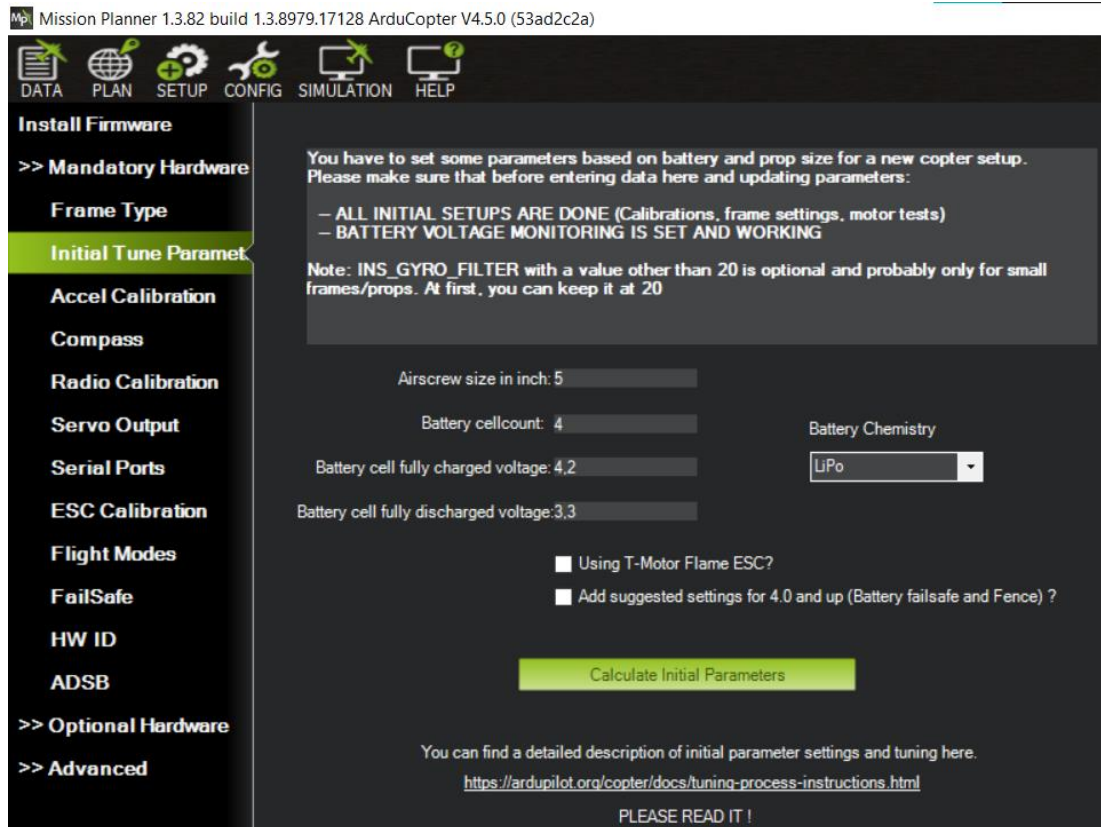


Figura 39. Configuración de parámetros iniciales con respecto al UAV.

b. Calibración del acelerómetro

La calibración del acelerómetro resulta esencial para obtener mediciones precisas de la aceleración lineal del UAV. Esto permite que el sistema de control opere de manera eficiente, mejorando tanto la estabilidad como la precisión de las maniobras de vuelo.

Para realizar el proceso de calibración, se colocó el UAV en una superficie plana y nivelada, garantizando que no existieran vibraciones ni movimientos que interfirieran en la precisión del ajuste. A continuación, el UAV se posicionó en diferentes direcciones: horizontal, hacia la izquierda, hacia la derecha, con la nariz apuntando hacia abajo, hacia arriba y en posición invertida, como se muestra en la Figura 40.

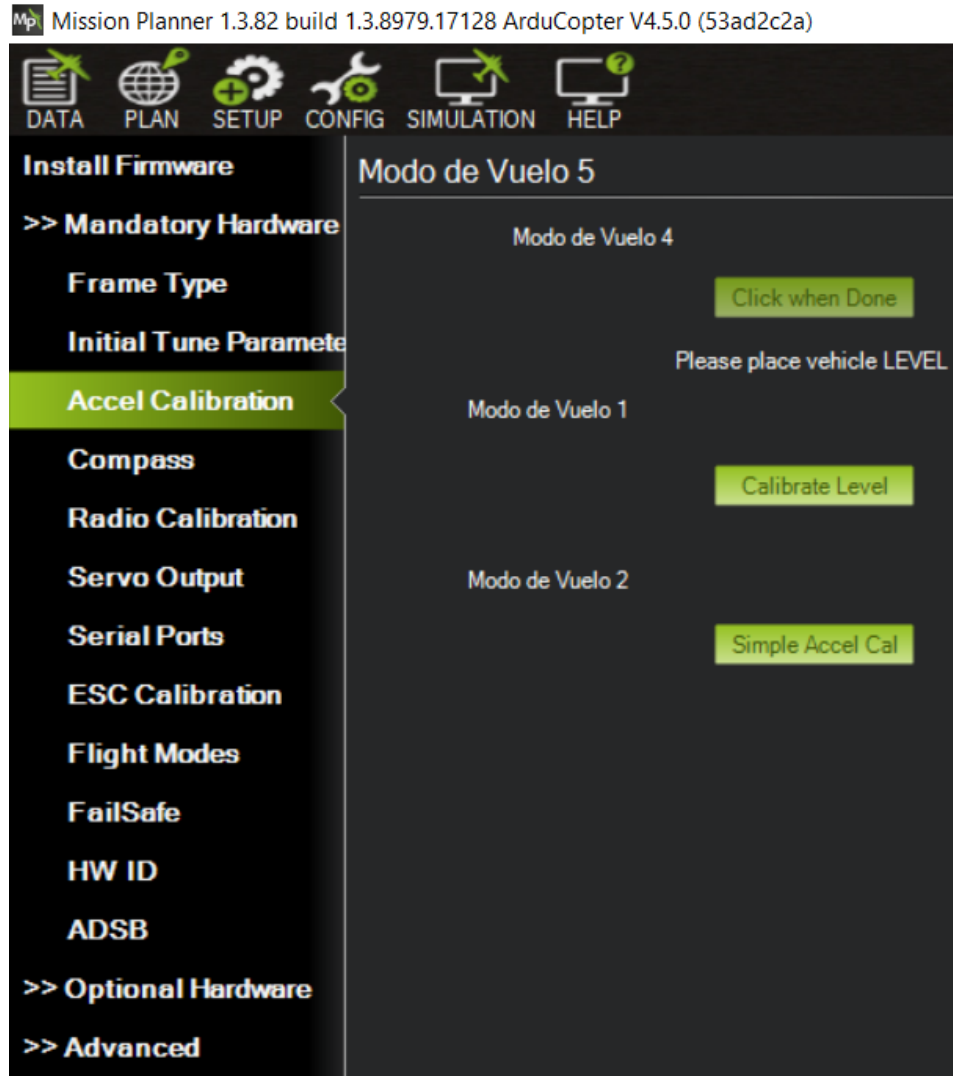


Figura 40. Calibración del acelerómetro.

c. Calibración de la brújula

La calibración correcta de la brújula asegura obtener datos precisos sobre la orientación y el rumbo del UAV, permitiendo que mantenga una trayectoria estable y confiable durante el vuelo.

Para su configuración es necesario estar en un área libre de interferencias magnéticas. En la sección Compass, se lleva a cabo el proceso de calibración, el cual requiere que el UAV sea movido de manera suave y continua, tal como se muestra en la Figura 41.

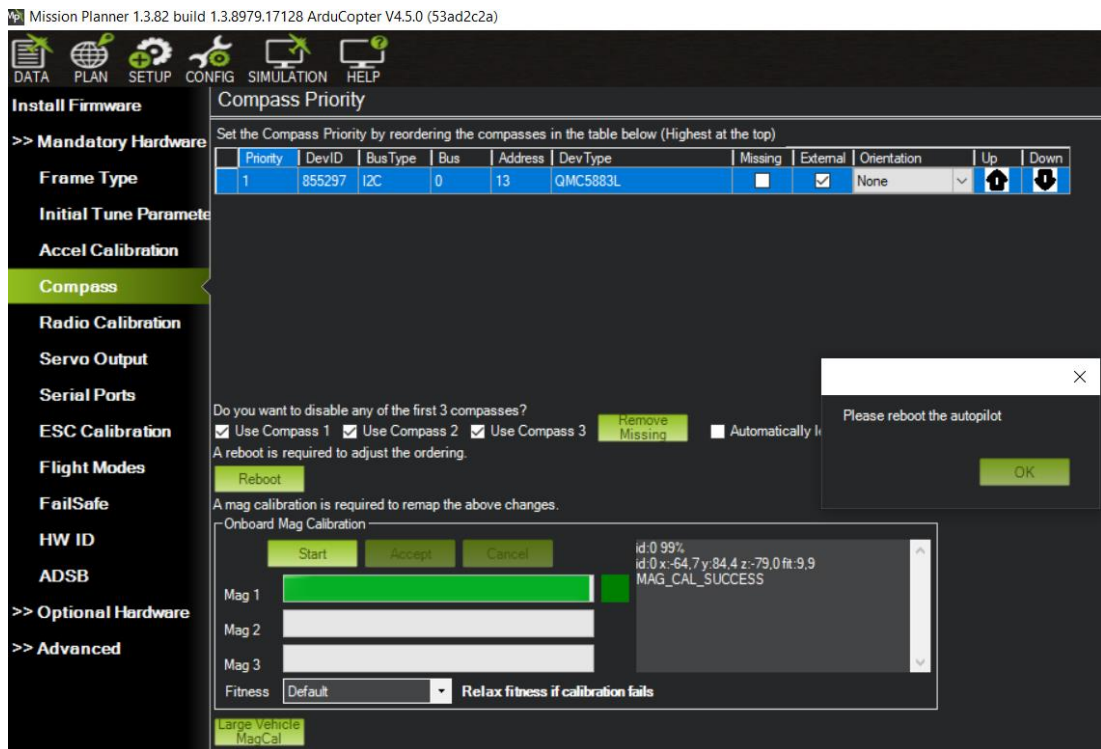


Figura 41. Calibración de la brújula del UAV.

Es importante realizar la recalibración regular de la brújula, sobre todo si surgen dificultades en la orientación durante el vuelo, asegurando así un control óptimo y una trayectoria más fiable del UAV.

d. Configuración y calibración del radio control

Para el proceso de configuración y calibración del radio control y el receptor, se utilizó el Software ExpressLRS Configurator versión 1.5.10. como se muestra en la Figura 42. Este procedimiento requirió determinar la versión del firmware, especificar el tipo de tarjeta, en este caso, una BetaFPV Nano 2.4GHz RX, emplear el método de flasheo por WiFi y establecer una frase única (binding) que permita la vinculación entre el radio control y el receptor.

The screenshot displays the ExpressLRS Configurator interface. At the top, under the 'Target' section, the 'Device category' is set to 'BETA FPV 2.4 GHz' and the 'Device' is 'BETA FPV Nano 2400 RX'. Below this, the 'Flashing Method' is set to 'WIFI'. The 'Device options' section includes 'Standard mode' selected and a 'RESET' button. The 'Regulatory domains' section has 'REGULATORY_DOMAIN_ISM_2400' checked. The 'Performance options' section has 'LOCK_ON_FIRST_CONNECTION' checked. The 'Network' section has 'AUTO_WIFI_ON_INTERVAL' checked with a value of 60. The 'Binding phrase setup' section has 'BINDING_PHRASE' checked, and the 'Custom binding phrase' is set to 'fiseitesis'.

Figura 42. Configuración del firmware utilizando el Software ExpressLRS Configurator.

Una vez descargado el firmware para el receptor (RX), se enciende el dron, lo que permite activar la tarjeta del RX, la cual cuenta con un LED indicador. Al principio, el parpadeo lento del LED sugiere que el sistema está a la espera de conectarse con el radio control. Transcurridos unos minutos, el receptor habilita el modo WiFi, permitiendo que, desde un dispositivo como un celular o computadora, se localice la red "ExpressLRS RX" e ingresar la contraseña predeterminada `espresslrs`. Luego, en un navegador web como en la Figura 43, se accede a la dirección IP 10.0.0.1, donde se ingresa el Binding configurado anteriormente en el software Mission Planner. A partir de esta frase, se genera el UID Flashed. Finalmente, se procede a cargar el firmware para luego guardar y reiniciar los cambios.

ExpressLRS

HappyModel EP1/2 2.4GHz RX
Firmware Rev. 3.2.1 (8dbbdb) ISM2G4

OPTIONS

WIFI

MODEL

UPDATE

Runtime Options

Binding Phrase

Binding Phrase

UID Flashed (Auto updated by changing the bind-phrase above)

38,6,128,133,102,97

WiFi "auto on" interval (s)

60

UART baud

420000

☐ Invert TX pin

☒ Lock on first connection

SAVE & REBOOT

Import/Export

DOWNLOAD MODEL CONFIGURATION FILE

Select a model configuration file to upload

Seleccionar archivo

Ningún archivo seleccionado

Figura 43. Configuración del Binding y subida del firmware en el RX.

El procedimiento para configurar el radio control sigue una secuencia similar a la utilizada en el receptor (RX). En primer lugar, se enciende el radio control y se descarga el firmware correspondiente al modelo Jumper T-Pro. A continuación, se establece la conexión a la red identificada como "ExpressLRS TX", utilizando la misma contraseña empleada para el receptor. Desde un navegador web, se accede a la dirección IP configurada, donde se introduce la frase de vinculación (Binding), la cual debe ser idéntica tanto para el transmisor (TX) como para el receptor (RX). Para concluir el proceso, se sube el firmware, se guardan las configuraciones y se reinicia el sistema, tal como se observa en la Figura 44.

ExpressLRS

Jumper AION T-Pro 2.4GHz TX
Firmware Rev. 3.2.1 (8dbbdb) ISM2G4

OPTIONS

WIFI

UPDATE

Runtime Options

Binding Phrase

Binding Phrase

UID Flashed (Auto updated by changing the bind-phrase above)

38,6,128,133,102,97

WiFi "auto on" interval (s)

60

TLM report interval (ms)

240

☐ UART inverted

Fan runtime (s)

30

SAVE & REBOOT

RESET RUNTIME OPTIONS TO DEFAULTS

Import/Export

DOWNLOAD MODEL CONFIGURATION FILE

Select a model configuration file to upload

Seleccionar archivo

Ningún archivo seleccionado

Figura 44. Configuración del Binding y subida del firmware en el TX.

El proceso final de calibración se lleva a cabo utilizando el software Mission Planner. Para ello, se conecta la placa controladora de vuelo a la computadora y se enciende el radio control. En el software, se selecciona la opción "Radio Calibration", donde se verifica que cada canal del radio control pueda moverse con libertad, alcanzando sus rangos máximos y mínimos, tal como se aprecia en la Figura 45. Al iniciar la calibración con la opción "Calibrar radio", se debe mover cada stick del radio control en todas las direcciones posibles para capturar con precisión los rangos extremos de funcionamiento del sistema.

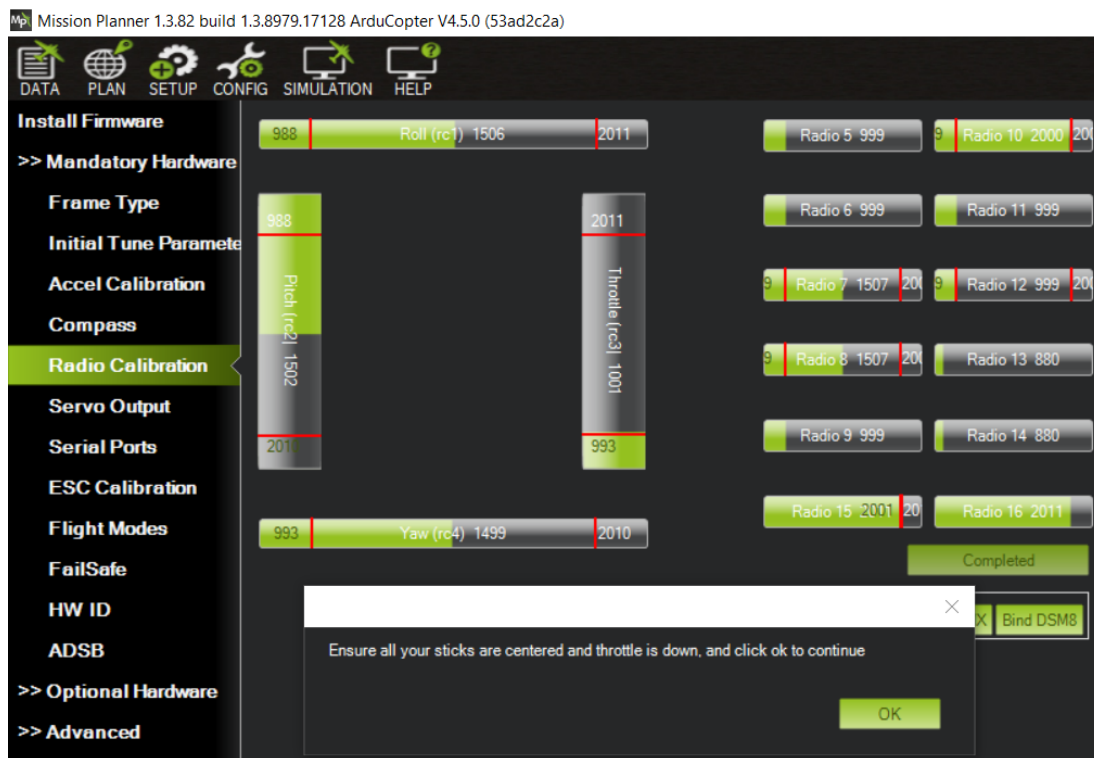


Figura 45. Calibración del radio control.

Tras completar exitosamente la calibración, se procedió a la configuración de los modos de vuelo. Este proceso comenzó en la sección de configuración (CONFIG), específicamente en la "Full Parameter List", donde se asignaron los canales correspondientes. El parámetro "FLTMODE_CH" fue asignado al canal 7 para gestionar los cambios de modo de vuelo, mientras que el parámetro "RC10_OPTION" correspondiente al canal 10 se configuró para controlar el armado y desarmado del dron. Esta configuración puede observarse detalladamente en la Figura 46.

Name	Value	Default	Units	Options	Desc
FLTMODE_CH	7	5		Channel7	RC Channel to use for flight mode contr
Name	Value	Default	Units	Options	Desc
RC10_OPTION	153	0		ArmDisarm (4.2 and higher)	Function assigned to this RC channel

Figura 46. Asignación de canales: Modos de vuelo; Armado y desarmado del dron.

En el apartado de SETUP, específicamente en "Flight Modes", se configura el interruptor asignado al canal 10, el cual tiene tres posiciones. Estas posiciones permiten definir los siguientes modos: el primero para vuelos estabilizados, el segundo

para vuelos de tipo fotografía y el tercero para vuelos automáticos. La representación de esta configuración puede observarse en la Figura 47.

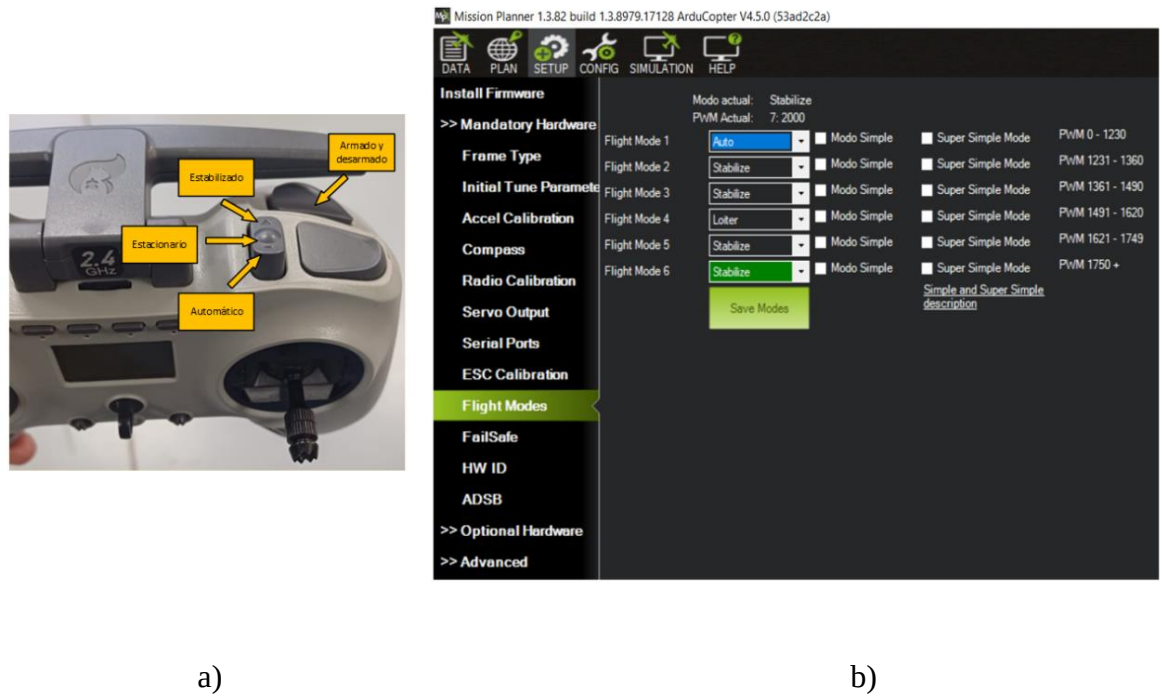


Figura 47. Modos de vuelo. a) Radio control; b) Mission Planner.

e. Calibración del ESC

El proceso de calibración del ESC es esencial para controlar la velocidad y el sentido de giro de los motores del UAV. Para iniciar, se ingresa a la opción "ESC Calibration", como se observa en la Figura 48. Es importante comprobar que la batería LiPo esté desconectada antes de hacer clic en "Calibrate ESCs". Luego de este paso, se desconecta el puerto USB de la computadora y se conecta nuevamente la batería LiPo. En ese momento, los LEDs se iluminan y se escucha un sonido característico, el cual confirma que la calibración del ESC se ha completado exitosamente. Finalmente, para hacer las pruebas se debe reiniciar desconectando la batería para que salga del modo de calibración.

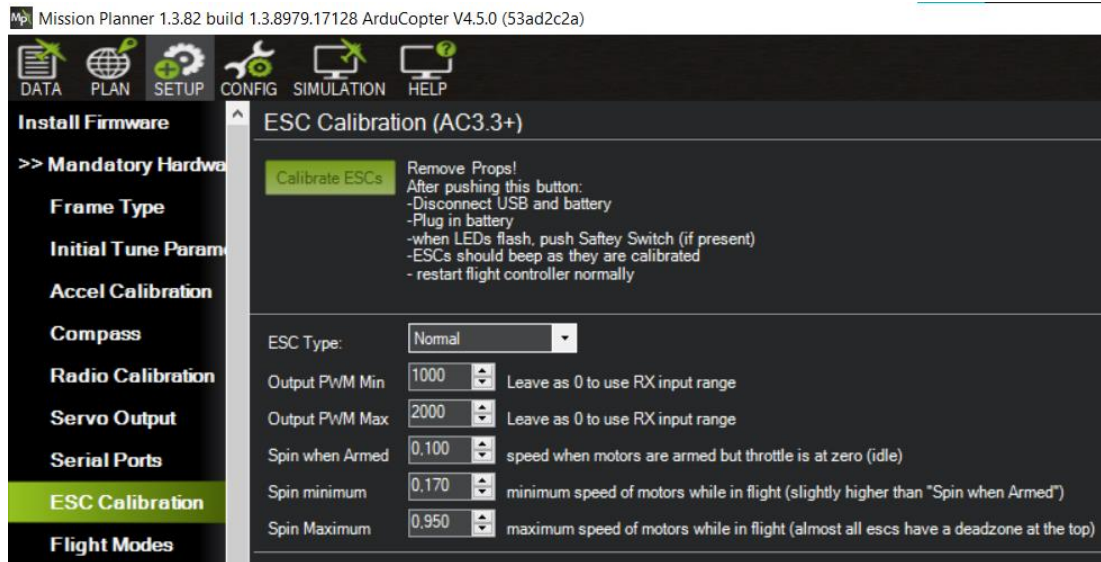


Figura 48. Calibración del ESC.

f. Test de los motores

Dentro de la opción "Motor Test", ubicada dentro del apartado "SETUP" y específicamente en la sección "Optional Hardware", como se muestra en la Figura 49. Se realizó las pruebas de motores tanto de manera individual y conjunta, verificando así su dirección de rotación.

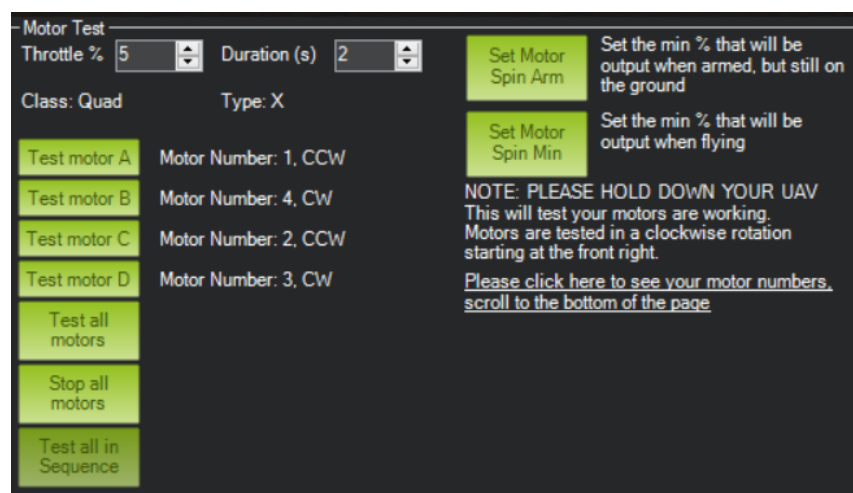


Figura 49. Test de movimiento de los motores del UAV.

En caso de que se observe alguna inconsistencia en el sentido de giro, hay que dirigirse a la opción "Servo Output". Desde esta opción, es posible realizar los ajustes correspondientes y configurar la dirección deseada para el funcionamiento de los

motores. La Figura 50 muestra de manera detallada el procedimiento necesario para efectuar dichos cambios de manera precisa.

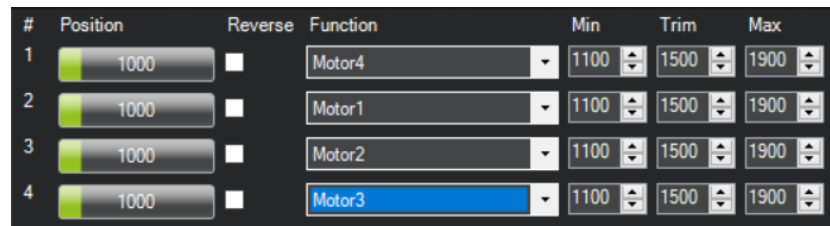


Figura 50. Configuración del giro de los motores.

g. Movimientos del UAV

En la Tabla 31 se describen los movimientos realizados por el UAV para desplazarse en respuesta a las órdenes enviadas por el piloto a través del radio control, siendo estos denominados como Pitch, Yaw y Roll.

Tabla 31. Movimiento del UAV [87].

Movimiento	Descripción
Pitch	Este movimiento se relaciona con la inclinación del UAV hacia adelante o hacia atrás alrededor del eje lateral (eje X). En este tipo de maniobra, la nariz del dron apunta hacia abajo cuando se desplaza hacia adelante, y hacia arriba al moverse en sentido contrario.
Roll	Este movimiento hace referencia a la inclinación lateral del UAV, es decir, el desplazamiento hacia la derecha o izquierda alrededor del eje longitudinal (eje Y). En este tipo de maniobra, el dron cambia de dirección lateralmente mientras mantiene su estabilidad.
Yaw	Se trata de un movimiento de rotación alrededor del eje vertical (eje Z), el cual permite al UAV realizar giros en su propio eje. Este movimiento ajusta la dirección del dron en el plano horizontal, permitiendo giros tanto en sentido horario como antihorario.

h. Configuración de los PIDs

El control PID es fundamental en los drones, ya que permite mantener la estabilidad y exactitud durante el vuelo. En la Tabla 32 se describen las funciones correspondientes a los tres componentes del controlador PID que se utilizan en un UAV.

Tabla 32. Componentes del Controlador PID en UAVs [26].

Componente (P, I, D)	Descripción
P (Ganancia Proporcional)	Determina la intensidad de la respuesta inmediata del dron al error actual entre la posición deseada y la posición real. Un valor más alto de P permite una corrección más rápida del error, pero si es excesivo, puede causar oscilaciones e inestabilidad. Este término es clave para ajustes inmediatos, como corregir una inclinación no deseada en el eje de cabeceo (Pitch) al detectar desviaciones.
I (Ganancia integral)	Se encarga de corregir los errores acumulados a lo largo del tiempo que el término proporcional no puede eliminar, como inclinaciones persistentes que permanecen sin corregirse. Este término asegura que el dron alcance con precisión la posición deseada, pero valores muy altos pueden provocar oscilaciones lentas y excesivas. Es ideal para ajustar errores estacionarios y mantener estabilidad a largo plazo.
D (Ganancia derivativa)	Controla la velocidad de cambio del error y actúa como un amortiguador que suaviza las correcciones realizadas por los términos proporcional e integral. Esto ayuda a evitar movimientos bruscos y oscilaciones, permitiendo transiciones más estables y precisas. Un valor de D demasiado alto puede hacer que el dron responda con lentitud, pero un ajuste equilibrado mejora significativamente la estabilidad durante maniobras rápidas.

Los valores de los PIDs varían según el UAV y las características de vuelo deseadas. En este caso, dado que se necesitaba un vuelo estabilizado, estacionario y automático, se ajustaron los valores conforme a las recomendaciones estándar para un dron de carrera de cuatro rotores.

Para llevar a cabo la configuración, se ingresó al apartado "CONFIG" y se eligió la opción "Extended Tuning". Se comenzaron a ajustar los parámetros generales, los cuales juegan un papel clave en el control de la respuesta del dron sobre los ejes. En la Figura 51 se detallan los valores asignados a las ganancias proporcional, integral y derivativa correspondientes a cada Rate en Roll, Pitch y Yaw.

Rate Roll		Rate Pitch		Rate Yaw	
P	0,0325	P	0,033	P	0,180
I	0,033	I	0,033	I	0,018
D	0,0005	D	0,0005	D	0,000

Figura 51. PIDs para el control de respuesta del UAV en los ejes.

Para lograr la estabilización del UAV y mantener los ángulos de vuelo deseados, se asignó un valor de 4.5 a cada uno de los movimientos del dron (Roll, Pitch y Yaw).

Además, para optimizar la potencia de los motores, se ajustaron los valores del controlador PID, tal como se detalla en la Figura 52.

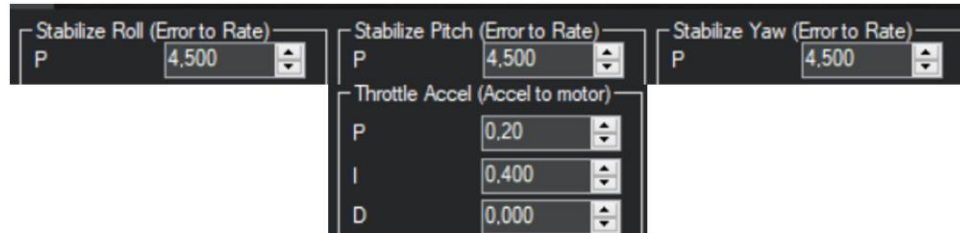


Figura 52. PIDs para la estabilización y potencia de los motores del UAV.

Para configurar el modo estacionario, en el cual el dron mantiene automáticamente su posición, se ajustó el parámetro proporcional a 1.0 lo que favorece un rápido ajuste de su posición horizontal. Además, fue necesario configurar el PID de la velocidad en el plano XY, como se muestra en la Figura 53, para lograr movimientos laterales más suaves. En el apartado de "WPNav", se asignaron velocidades reducidas para asegurar vuelos más estables, incluyendo una velocidad horizontal de 1250 cm/s, una velocidad de ascenso (Speed Up) de 250 cm/s y una velocidad de descenso de 150 cm/s

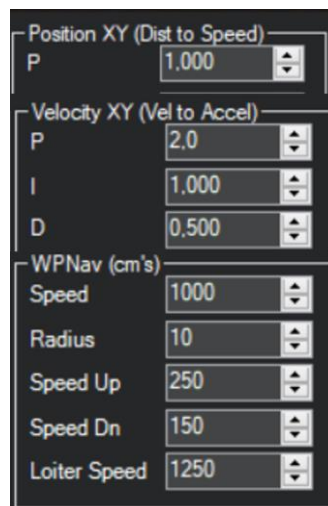


Figura 53. Configuración de los PIDs y velocidades para el modo estacionario.

Para el modo automático, al igual que en el modo estacionario, se configuraron en el apartado de "WPNav" la velocidad entre puntos a 1000 cm/s y un radio de alcance de 10 cm para cada punto de la trayectoria, tal como se muestra en la Figura 54. Estos valores fueron seleccionados porque resultan ser los más adecuados para aplicaciones

de captura y transmisión de video hacia una estación base, asegurando un rendimiento estable y eficiente en el proceso.

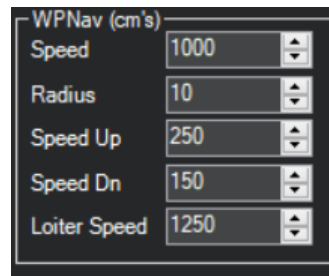


Figura 54. Configuración de la velocidad de desplazamiento y el alcance de radio de cada punto.

i. Planificación de vuelo.

Para llevar a cabo misiones de vuelo automáticas con el UAV, se accede a la sección PLAN. En este apartado, se define en el mapa el lugar exacto donde se ejecutarán las misiones. Para este caso, se utilizó la cancha de la Universidad Técnica de Ambato como punto de referencia, como se muestra en la Figura 55.

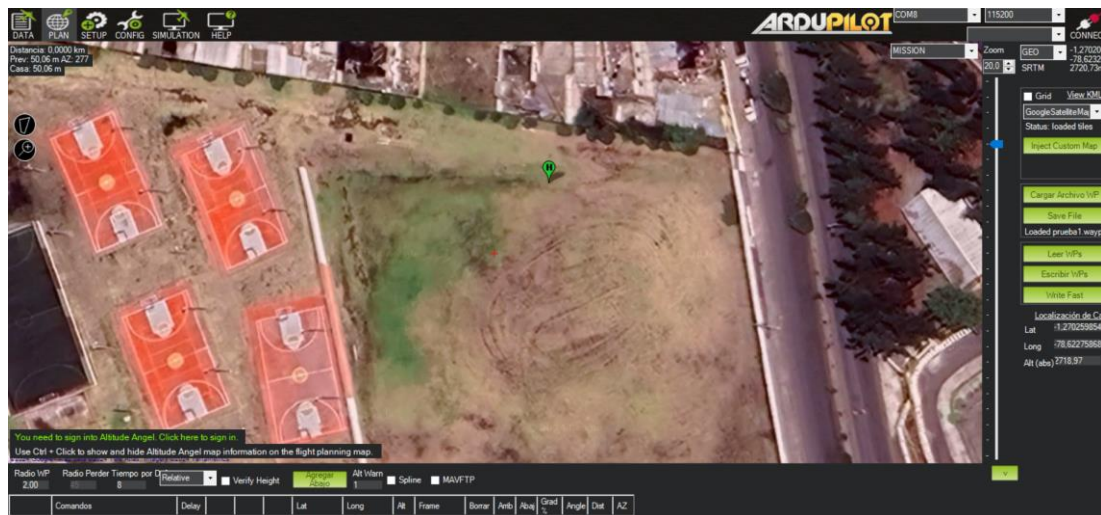


Figura 55. Punto HOME o "H".

El sistema de vuelo automático se basa en dos categorías de comandos: los llamados "DO", utilizados para establecer condiciones específicas, y los comandos que permiten trazar rutas automatizadas. En Tabla 33 presenta una descripción detallada de los comandos esenciales que garantizan el éxito de la misión de vuelo [88].

Tabla 33. Comandos fundamentales para las rutas en ArduPilot [88].

Comando	Descripción
TAKEOFF	Desde su ubicación actual, el vehículo se elevará directamente hasta la altura especificada en metros. Este paso es esencial y debe ejecutarse como el primer comando en cualquier misión planificada.
WAYPOINT	El trayecto del vehículo será una línea recta que lo conducirá directamente hasta el destino establecido, el cual está determinado por las coordenadas de latitud, longitud y altura en metros.
SPLINE_WAYPOINT	Al igual que los puntos de ruta convencionales, los "spline" requieren parámetros como latitud, longitud, altitud y tiempo de espera. La diferencia radica en que, al ejecutarse, trazan trayectorias suaves y curvas, garantizando que el vehículo siga las coordenadas predefinidas con precisión.
DO_SET_ROI	Apunta la cámara hacia una región de interés específica (posición, incluida la altitud)
LOITER_TURNS	El vehículo volará en círculo manteniéndose alrededor de las coordenadas de latitud, longitud y altitud previamente definidas en metros.
RETURN_TO_LAUNCH	El vehículo cuenta con una función que le permite retornar automáticamente al sitio donde fue montado por última vez, asegurando así un desplazamiento controlado hacia su origen.
LAND	El sistema se emplea para facilitar aterrizajes exactos, basándose en las coordenadas de latitud y longitud indicadas como referencia.
DELAY	El sistema del vehículo está programado para iniciar el despegue de manera automática una vez que se alcance el tiempo establecido.

El uso de los comandos fundamentales está condicionado por la habilidad del piloto para aplicarlos correctamente. Para optimizar el desempeño, es aconsejable emplear WAYPOINTS que permitan programar con precisión los puntos de misión, reservando el control manual para las maniobras de ascenso y descenso, donde se requiere mayor exactitud. La Figura 56 se puede observar una misión en la que el UAV toma como referencia el punto H que corresponde al lugar de inicio. Posteriormente, se dirige al punto 1 (P1) a una altura de 1 metro, permaneciendo 5 segundos en cada punto. Luego avanza al punto 2 (P2) a 1.5 metros, al punto 3 (P3) a 2 metros, al punto 4 (P4) a 1.5 metros y, finalmente, al punto 5 (P5) a 1 metro. En este último, el control cambia a modo manual para efectuar un descenso más preciso y minimizar posibles errores.

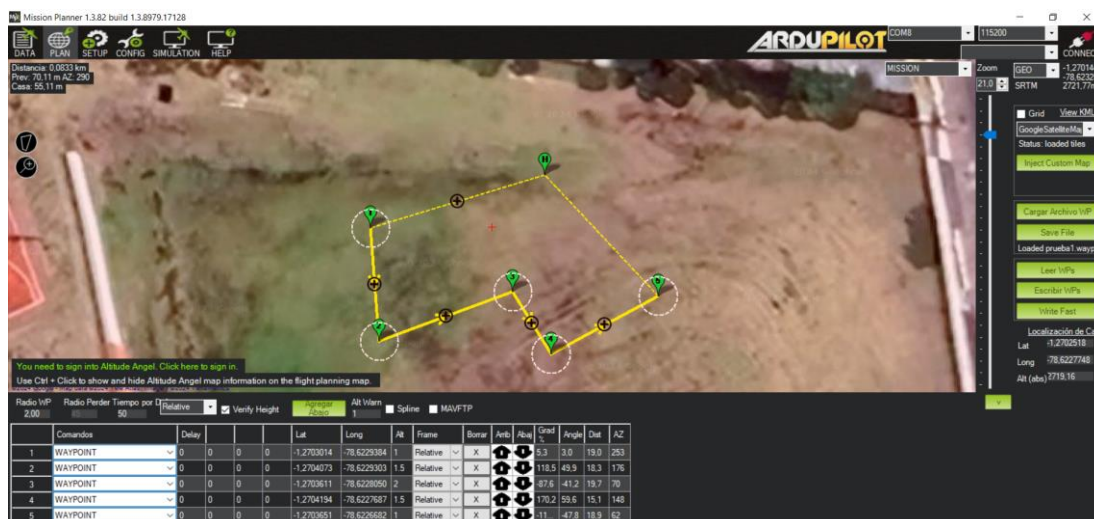


Figura 56. Trayectorias y alturas de vuelo en el software Mission Planner.

A continuación, la trayectoria se transfiere al UAV utilizando la función "Write", siempre que este esté vinculado al computador. Para guardar la ruta y facilitar su uso en misiones posteriores, es necesario seleccionar la opción "Save File" y guardarla como un archivo reutilizable.

j. Configuración del OSD

La configuración del OSD requiere primero conectar el UAV al sistema. Luego, en la sección "CONFIG", específicamente en el apartado de "Onboard OSD", se presentan diversas opciones que pueden ajustarse según las necesidades específicas del usuario. En la Figura 57 se detalla cómo se configuró la interfaz para priorizar los datos clave que permiten una adquisición precisa de imágenes de los vehículos. Cabe destacar que la cámara FPV integrada en el UAV emite señales en formato NTSC, por lo que toda la información debe mantenerse dentro de los límites de la pantalla.



Figura 57. Configuración del OSD en el software Mission Planner.

3.1.12 Diagrama del sistema

El diagrama de procesos con los bloques que conforman el sistema de recepción de video y procesamiento para detección de automóviles y placas, con la extracción de los caracteres de placa y ubicación para el envío a la base de datos, se presenta en la Figura 58.

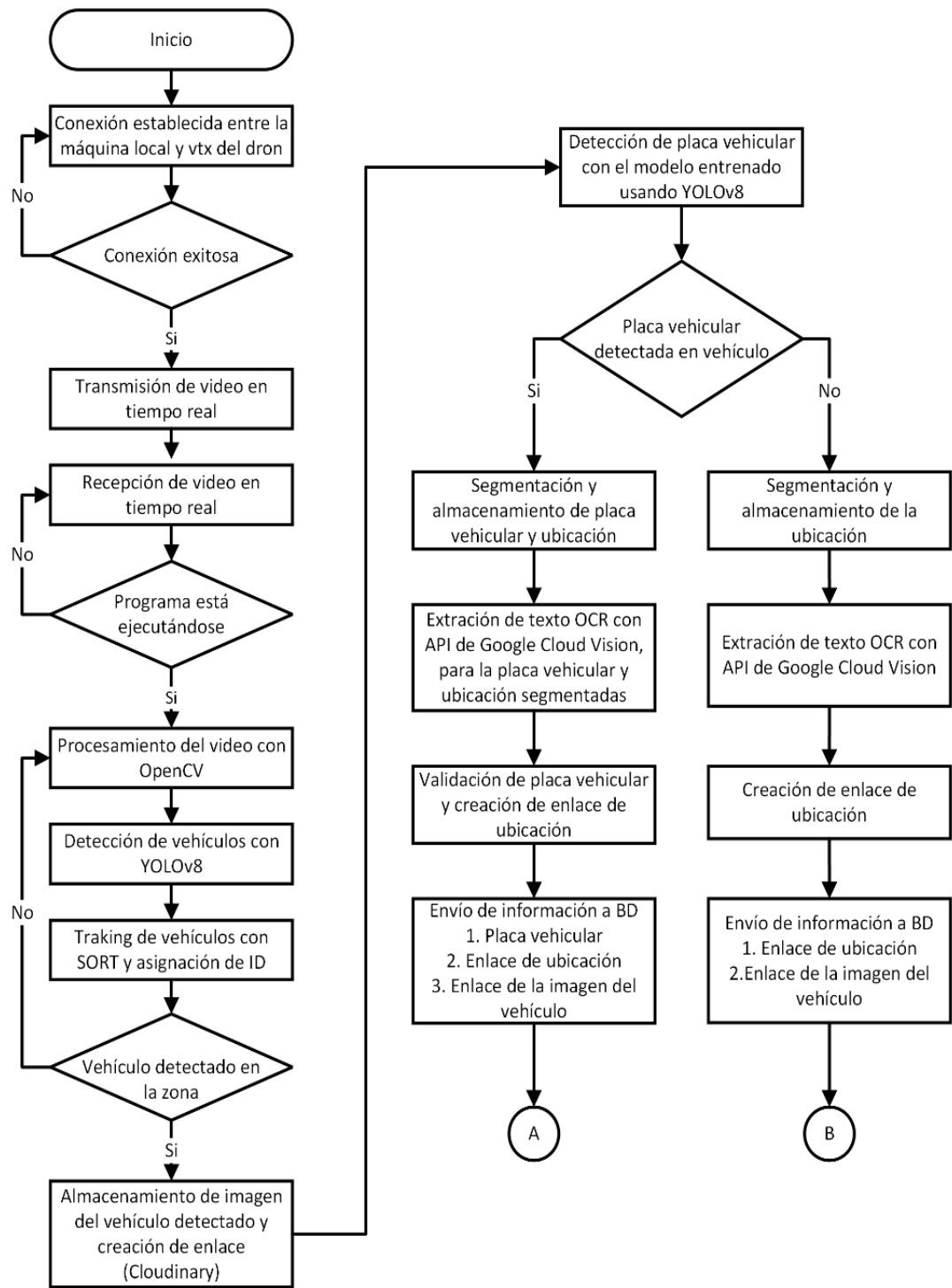


Figura 58. Diagrama del proceso primera parte.

El diagrama de la Figura 59, presenta bloques de como la información enviada desde el ordenador ahora se procesa en la base de datos según sea el caso de vehículo reportado o automóvil sin placa, almacenando la información en las tablas y realizando el proceso respectivo, para el envío de notificaciones.

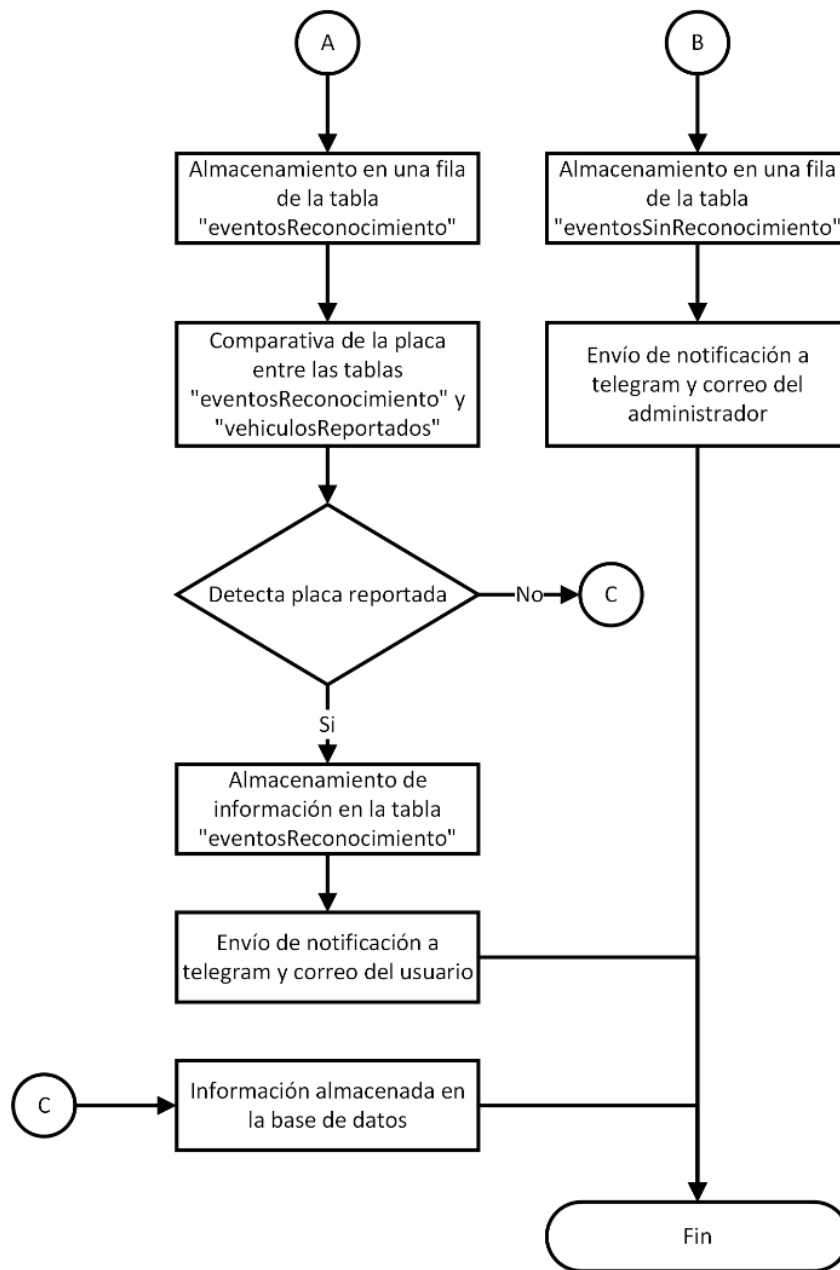


Figura 59. Diagrama del proceso segunda parte.

3.1.13 Entrenamiento de la red neuronal para la detección de placas

Para cumplir con el tercer objetivo específico de implementar un sistema inteligente para la localización de vehículos robados utilizando visión artificial, se desarrolló un modelo basado en redes neuronales entrenadas con YOLOv8m para la detección de matrículas vehiculares. El proceso incluyó la recopilación y anotación de un conjunto de datos específico, seguido del entrenamiento del modelo para optimizar la precisión

en la identificación de placas en diversas condiciones. Posteriormente, se integró un sistema de reconocimiento óptico de caracteres (OCR) para extraer el texto de las matrículas detectadas y enviarlo a una base de datos. En el almacenamiento se realizó la comparación automática con registros de vehículos reportados como robados, facilitando la detección en el menor tiempo posible dentro de un parqueadero.

a. Dataset COCO para detección de vehículos

El dataset COCO (Common Objects in Context), es uno de los conjuntos de datos más utilizados en los proyectos de visión por computadora. Creado por Microsoft y usado en tareas de detección de objetos, segmentación de instancias, y puntos de predicción. COCO tiene más de 328000 imágenes, y más de 80 clases de entidades de anotación, una sola anotación de COCO, de promedio, tiene varios objetos en una imagen, en la práctica se tendrían casos variados, para escenas de la vida real que pertenecen a los objetos anotados [89].

- ***Estructura y etiqueta***

El dataset COCO tiene un esquema estructural general previo con [89]:

Imágenes: Cuenta con 2 500 000 instancias etiquetadas en 328 000 imágenes.

Clases: Conjunto de datos con 80 tipos de objetos, desde personas, animales, vehículos hasta elementos de hogar.

Etiquetas: Permite anotar varios objetos en una única imagen con limitaciones, segmentaciones y puntos clave.

- ***Clases de vehículos***

En COCO, la categoría de "vehículos" tiene subcategorías como: automóviles, autobuses, camiones, motocicletas y bicicletas etiquetadas para cada caso. Dentro de la clase vehículos contiene aproximadamente 60000 imágenes distribuidos de la siguiente manera [89]:

Automóvil (car): aproximadamente 44 000

Motocicleta (motorcycle): aproximadamente 4 000

Autobús (bus): aproximadamente 5 000

Camión (truck): aproximadamente 3 000

Bicicleta (bicycle): aproximadamente 4 000

Para la aplicación en el sistema se utiliza la categoría automóvil (car), identificado dentro del dataset con el número de clase "2" para programar la identificación de automóviles ya que el proyecto de investigación está centrado en esta categoría.

b. Dataset de placas vehiculares

En la Figura 60 se presenta el diagrama de procesos para la creación del dataset para la detección de placas vehiculares.

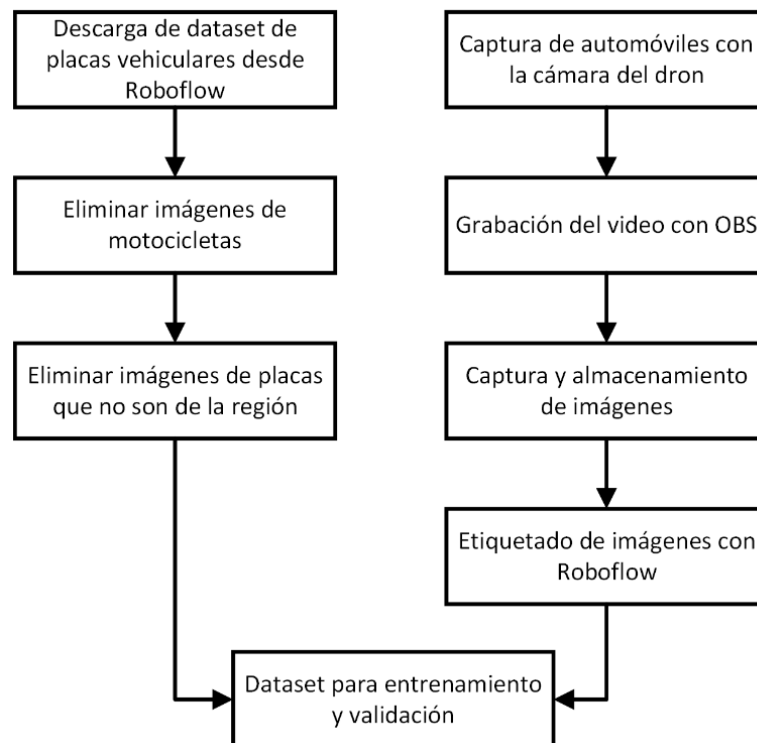


Figura 60. Creación de dataset de placas vehiculares.

Para configurar el sistema de captura y detección de placas vehiculares, lo primero es obtener un dataset para el entrenamiento con las imágenes más representativas del objeto a detectar. Para esto se seleccionó un dataset cargado en la página de Roboflow, que contine una gran base de información en el área de visión por computadora para

diferentes aplicaciones, en este caso se utilizó un dataset está conformado por la parte de entrenamiento, validación y test, con 24000 imágenes aproximadamente, como se muestra en la Figura 61 y en la URL <https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/4>.

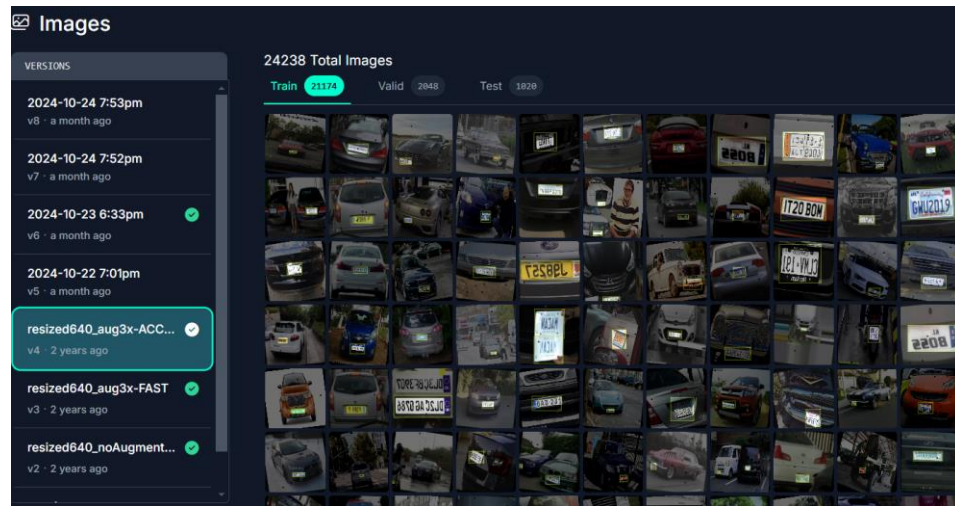


Figura 61. Dataset de placas vehiculares en Roboflow.

Debido a que el dataset encontrado en Roboflow contiene imágenes de motocicletas y placas de vehículos que no pertenecen a la región, para eso se realiza un proceso de filtrado, eliminando esas imágenes y además se adicionaron capturas de vehículos realizadas con la cámara incorporada al dron desde el computador portátil. Los videos se capturaron con el software OBS para almacenarlos, posteriormente, se extrajeron 1000 imágenes para agregarlas al dataset como se puede visualizar en la Figura 62

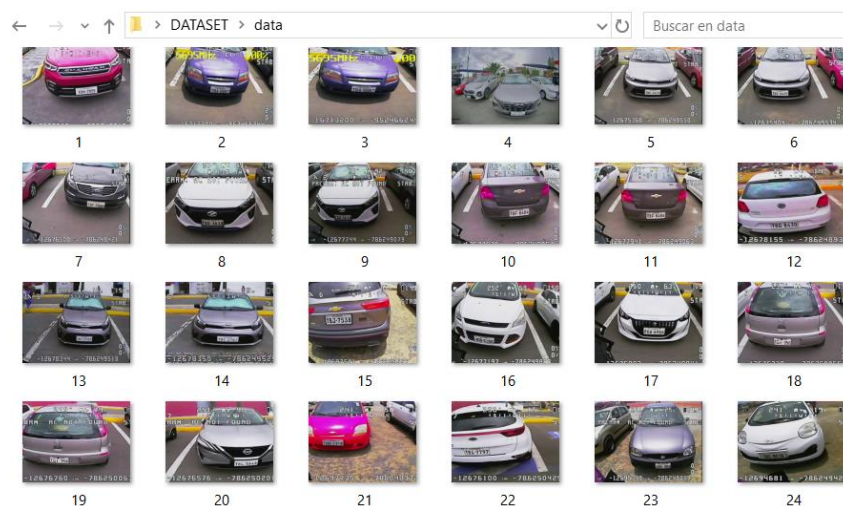


Figura 62. Imágenes capturadas desde UAV.

- **Estructura y etiqueta**

Para el etiquetado se utilizó la plataforma roboflow donde se subieron las capturas y se creó la etiqueta "PLACA", posteriormente se busca la posición de la placa del automóvil y se lo selecciona con un cuadro delimitador, este proceso se lo realiza con todas las imágenes como se muestra en la Figura 63.

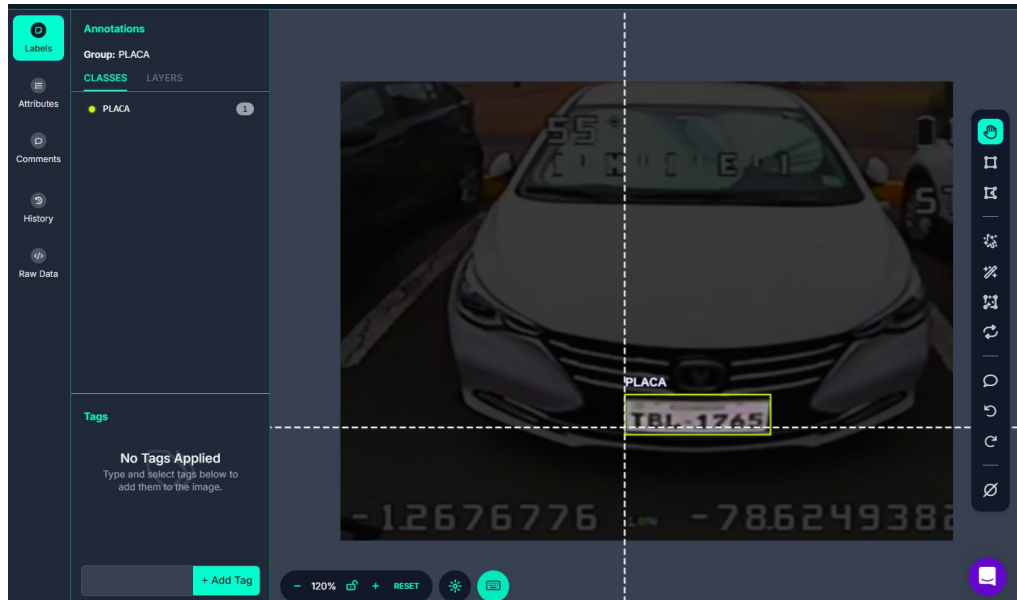


Figura 63. Asignación de etiquetas.

Con las etiquetas terminadas se añade las imágenes descargadas desde Roboflow respectivamente filtradas, para crear un dataset óptimo para la detección de placas, con la estructura que se muestra en la Figura 64.

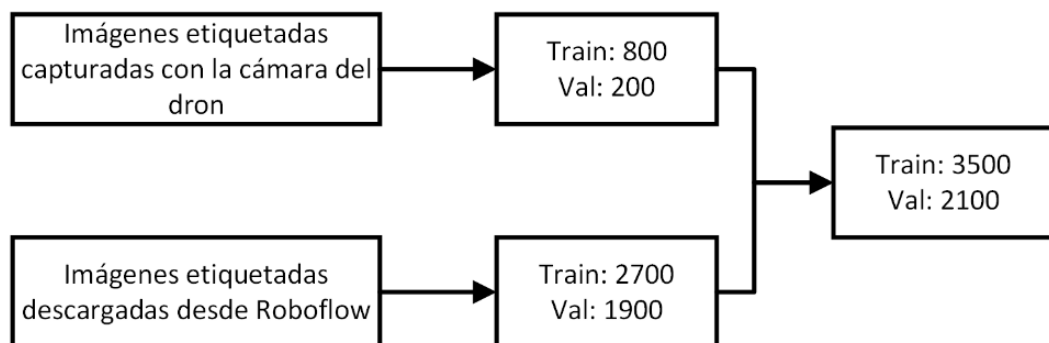


Figura 64. Conformación de dataset en Roboflow.

Con el dataset cargado en Roboflow, se debe crear una nueva versión donde las imágenes atraviesan por etapas de preprocesamiento y redimensión, adicional para la generación de imágenes adicionales aplican 3 generadores de imagen a partir de la original como voltear, rotación y escala de grises, para la creación de un dataset extenso que permite pasar de 3500 imágenes de entrenamiento a 10500, estas características se presentan en la Figura 65.

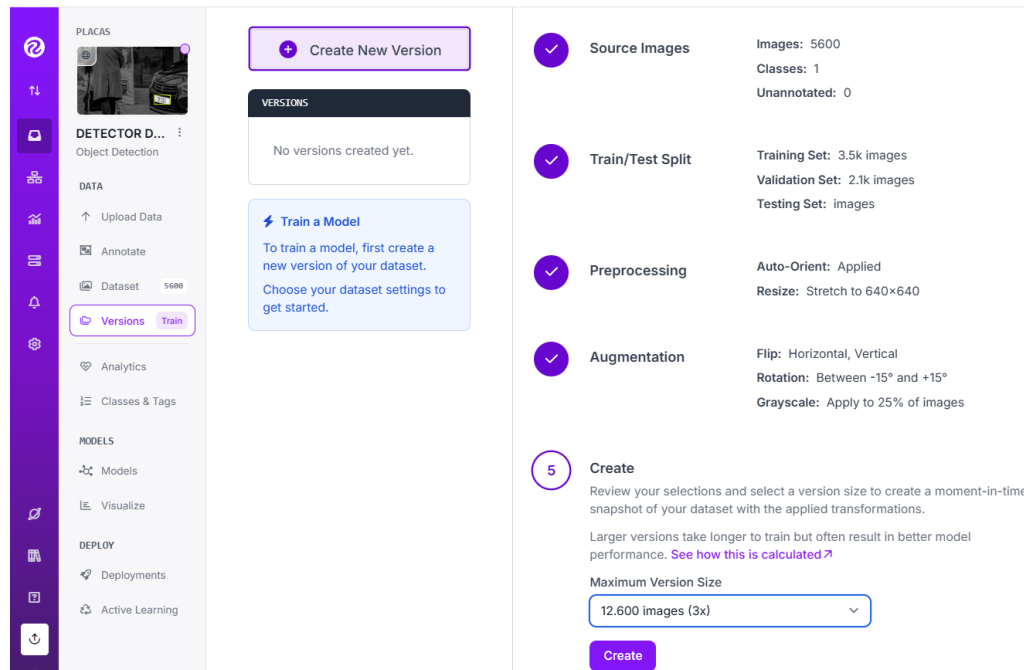


Figura 65. Parámetros para creación de dataset.

La arquitectura del dataset está compuesta por dos directorios, el primer componente "train" se compone por 2 subdirectorios "images" y "labels", en la carpeta imágenes se almacena los elementos que se requieren analizar y se usan como entrada para analizar el modelo. Por otra parte, la carpeta "labels" contiene detalles de las anotaciones, como las coordenadas de las cajas delimitadoras.

El segundo componente es el directorio "val", que mantiene la estructura de la carpeta "train", con las subcarpetas "images" y "labels". Se diferencia de "train", en que las imágenes de "val" son usadas únicamente para validación y no participan en el entrenamiento del modelo. Con estas imágenes se verifica cómo el modelo puede adaptarse a nuevos datos.

El dataset queda estructurado con un total de 12600 imágenes entre las carpetas "train" con un total de 10500 imágenes (80%) y para validación "val" 2100 imágenes (20%), utilizando una división en proporción según las recomendaciones para YOLOv8 [90]. Adicional el dataset debe contener el archivo data.yaml que direcciona a las carpetas correspondientes donde se almacenan las imágenes para el proceso respectivo, esta arquitectura se presenta en la Figura 66.

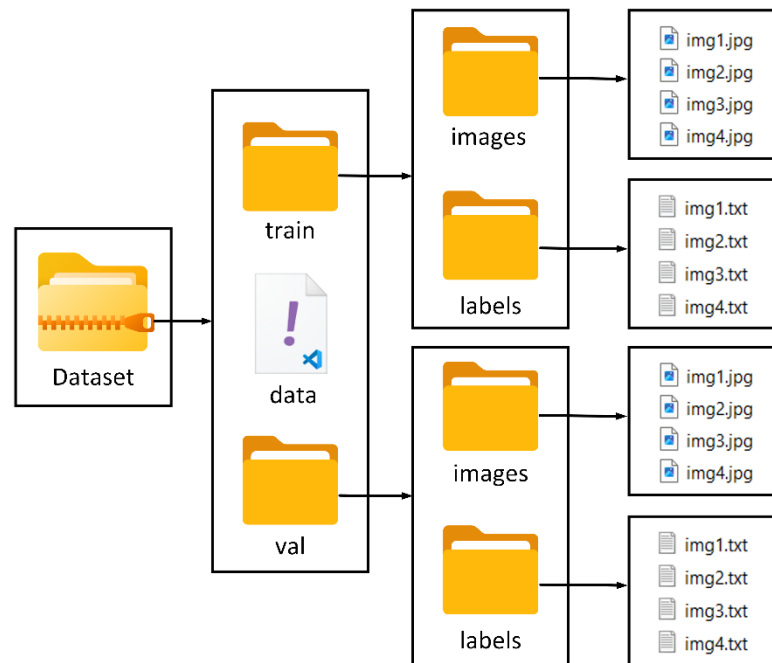


Figura 66. Arquitectura del dataset.

c. Entrenamiento de la red neuronal

Para el entrenamiento de la red neuronal se utilizó la plataforma en línea Kaggle que es una plataforma que permite la ejecución de códigos en Python desde un navegador web con conexión a internet, además opera con YOLOv8 y sus respectivos modelos. La versión de YOLOv8x tiene un grado de precisión más alto y de igual manera su latencia al realizar las detecciones es de 479.1 milisegundos (ms).

En este proyecto se utiliza YOLOv8m optimizando la velocidad de procesamiento a un tiempo de 234.7 milisegundos (ms). Además, en la Tabla 19 se representa que el modelo mediano logra una precisión de 50,2 de mAP aproximando a sus variantes superiores como YOLOv8l con 52.9 mAP y YOLOv8x con 53.9 mAP. Para esta

aplicación que se basa en la detección de objetos no requiere de mayores recursos, equilibrando así la detección y tiempo de respuesta.

- **Descarga de Dataset**

Para iniciar el entrenamiento de la red neuronal se debe descargar el dataset creado en Roboflow, en este caso para no consumir recursos de almacenamiento se enlaza por medio de un SDK y una clave API para que se descargue directamente en la máquina que proporciona Kaggle y se selecciona el parámetro Jupiter para elegir el formato, obteniendo el código como se muestra en la Figura 67.

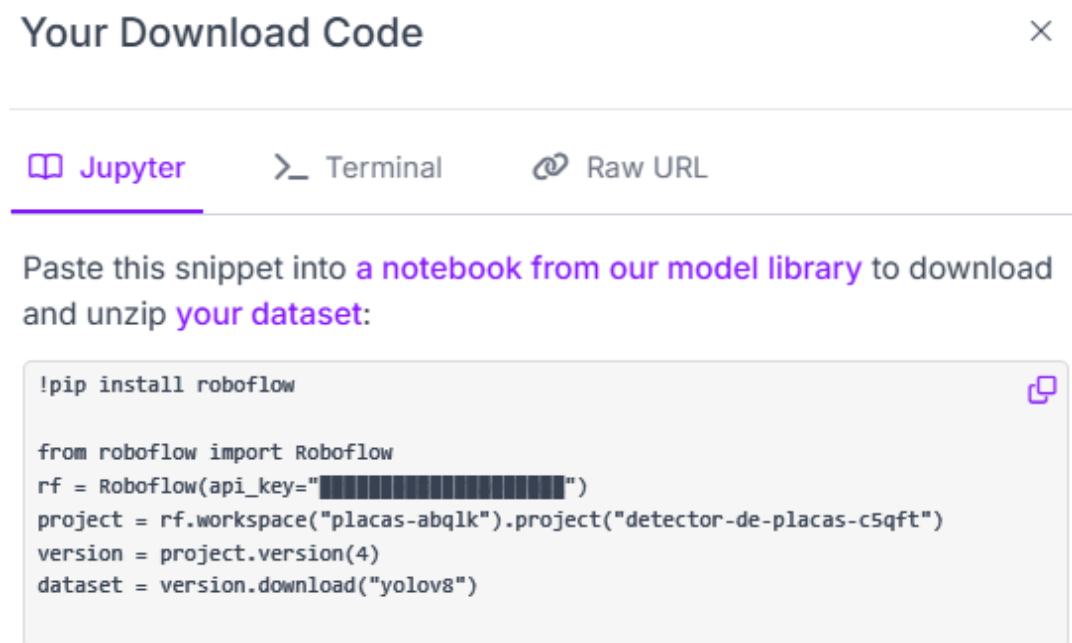


Figura 67. Código de descarga desde Roboflow.

- **Características de la máquina utilizada en el entrenamiento.**

Para conseguir un mejor rendimiento del entrenamiento y optimizar el tiempo se usó la plataforma Kaggle en su versión gratuita de 30 horas de uso a la semana, proporciona un entorno de notebook de Jupyter y cuenta con una RAM de 31 GB y se habilita la GPU Tesla T4. En la Tabla 34 se detallan las características de la máquina utilizada.

Tabla 34. Parámetros de la máquina en Kaggle.

Parámetro	Especificación
CPU	Intel Xeon @ 2.00 GHz
RAM	31 GB
GPU	Tesla T4
Memoria de GPU	15 GB
CUDA Version	12.6
Sistema Operativo	Linux (Ubuntu 22.04.4 LTS)
Almacenamiento	7.9 TB

- **Código para el entrenamiento**

En Kaggle se configura el uso de una tarjeta gráfica para un consumo de recursos más apropiado, para esto se debe crear un nuevo notebook y ajustar los parámetros de ejecución y seleccionar el uso de GPU, como se muestra en la Figura 68.

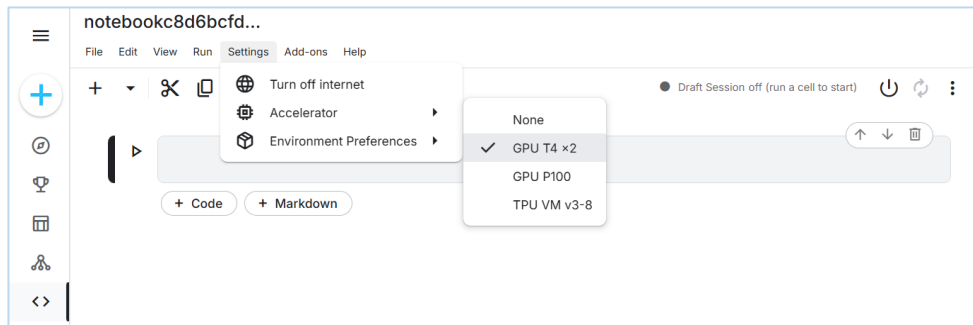


Figura 68. Entorno de ejecución de Kaggle.

Para empezar con el entrenamiento se debe instalar las librerías de ultralytics que se basa en el desarrollo de software de código abierto para el uso de inteligencia artificial, con el código de la Figura 69 en una celda del notebook.

```
# Instala la biblioteca Ultralytics (YOLOv8).
!pip install ultralytics

# Limpia la salida de la celda para mantener el notebook ordenado.
from IPython import import display
display.clear_output()

# Importa Ultralytics y verifica el entorno.
import ultralytics
ultralytics.checks()
```

Figura 69. Descarga de librerías ultralytics.

Posteriormente se crea un directorio para almacenar la descarga del dataset desde Roboflow en el entorno de Kaggle en la siguiente celda, Figura 70.

```
# Crea y accede al directorio 'datasets'.
!mkdir {HOME}/datasets
%cd {HOME}/datasets

# Instala Roboflow.
!pip install roboflow

# Importa Roboflow y autentica con la API.
from roboflow import Roboflow
rf = Roboflow(api_key="api_key")

# Obtiene el dataset de detección de placas y lo descarga en formato YOLOv8.
project = rf.workspace("placas-abqlk").project("detector-de-placas-c5qft")
dataset = project.version(4).download("yolov8")
```

Figura 70. Descarga de dataset desde Roboflow.

En la siguiente celda se ejecuta el comando para el crear el modelo de entrenamiento con 80 épocas como se muestra en la Figura 71.

```
%cd {HOME}
!yolo task=detect mode=train model=yolov8m.pt data={dataset.location}/data.yaml epochs=80 plots=True

/kaggle/working
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8m.pt to 'yolov8m.pt'...
100%|██████████████████████████████| 49.7M/49.7M [00:00<00:00, 270MB/s]
Ultralytics 8.3.57 Python-3.10.14 torch-2.4.0 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: task=detect, mode=train, model=yolov8m.pt, data=/kaggle/working/datasets/DETECTOR-DE-PLACAS-4/dat
a.yaml, epochs=100, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=N
terministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=
1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=
val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, sourc
e=None, vid_stride=1, stream_buffer=False, visualize=False, augment=False, agnostic_nms=False, classes=None, retin
a_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_conf=False, save_crop=False, show_l
abels=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False, optimize=False, int
8=False, dynamic=False, simplify=True, opset=None, workspace=None, nms=False, lr0=0.01, lrf=0.01, momentum=0.937,
weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, df1=1.5, pose=1
2.0, kobj=1.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspe
ctive=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, copy_paste_mode=fliplr, auto_augm
ent=pandaugment, erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs/detect/train
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100%|██████████████████████████████| 755k/755k [00:00<00:00, 17.1MB/s]
Overriding model.yaml nc=80 with nc=1
```

	from	n	params	module	arguments
0	-1	1	1392	ultralytics.nn.modules.conv.Conv	[3, 48, 3, 2]
1	-1	1	41664	ultralytics.nn.modules.conv.Conv	[48, 96, 3, 2]
2	-1	2	111360	ultralytics.nn.modules.block.C2f	[96, 96, 2, True]
3	-1	1	166272	ultralytics.nn.modules.conv.Conv	[96, 192, 3, 2]

Figura 71. Código para el entrenamiento de la red neuronal.

- **Resultado del modelo entrenado**

Con un tiempo aproximado de 10 horas de entrenamiento se genera el archivo del modelo entrenado "best.pt" que se encuentra en la carpeta weights en la dirección /kaggle/working/runs/detect/train, como se puede visualizar en la Figura 72.

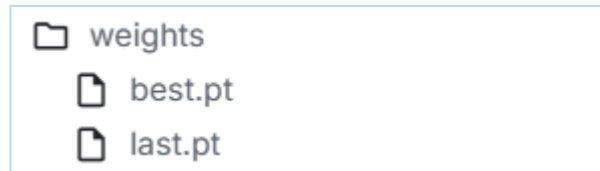


Figura 72. Carpeta de resultados del entrenamiento de la red neuronal

La Figura 73 y la Figura 74 presentan las gráficas de los resultados que se obtuvieron al entrenar la red neuronal con diferentes valores de épocas en YOLOv8m, donde se visualiza que tan óptimo fue el entrenamiento realizado en Kaggle.

Con los parámetros de entrenamiento se evalúa el comportamiento de la red neuronal en la detección de objetos y su segmentación, en la Tabla 35 se especifica cada una de las métricas en los resultados.

Tabla 35. Funciones de las métricas de modelos entrenados.

Métricas	Descripción
Train/box_loss	Representa el error en las coordenadas de la caja delimitadora o etiqueta vs coordenadas reales.
Trains/cls_loss	Mide el error entre las probabilidades de clase predichas y las etiquetas de clase reales para los objetos presentes en la imagen.
Train/dfl_loss	Mide la efectividad del modelo en detectar objetos.
Precision	Medición de la exactitud a las predicciones positivas frente a falsos positivos por la ecuación mencionada: (verdades positivas / (verdades positivas + falsos positivos)).
Recall	Evalúa las observaciones positivas predichas correctamente referente a los positivos reales.
MpA50	Mide la precisión promedio a un umbral del 50%, evaluado en detecciones correctas.
Mpa50-95	Esta métrica considera la precisión media a través de diferentes umbrales de superposición, desde el 50% hasta el 95%.

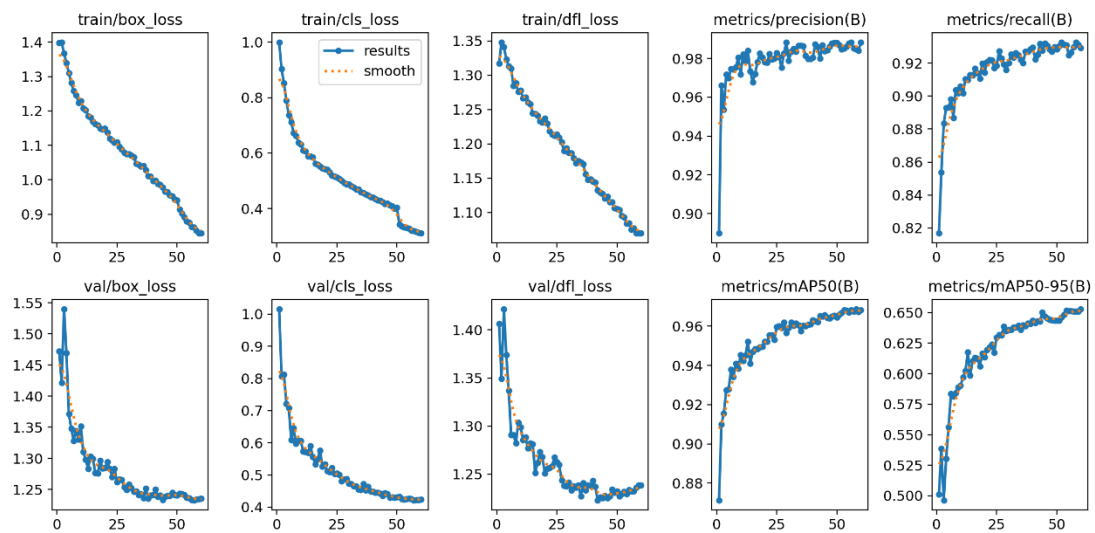


Figura 73. Resultados del entrenamiento de la red neuronal con YOLOv8m utilizando 60 épocas.

El primer entrenamiento con 60 épocas de YOLOv8m presenta disminuciones en las pérdidas de caja (box_loss), clasificación (cls_loss) y DFL (df_l_loss), tanto en entrenamiento como en validación. Las métricas de precisión y recall alcanzan valores elevados, con la precisión manteniéndose por encima de 0.96 y el recall aumentando progresivamente. Además, el rendimiento en términos de mAP50 y mAP50-95 muestra una mejora constante, alcanzando más de 0.96 y 0.65, respectivamente, lo que indica un desempeño robusto en la detección de objetos y buena generalización del modelo en el conjunto de validación.

Con la finalidad de mejorar los parámetros del entrenamiento de la red neuronal, se aumentó de 60 épocas a 80 épocas en el entorno de Kaggle.

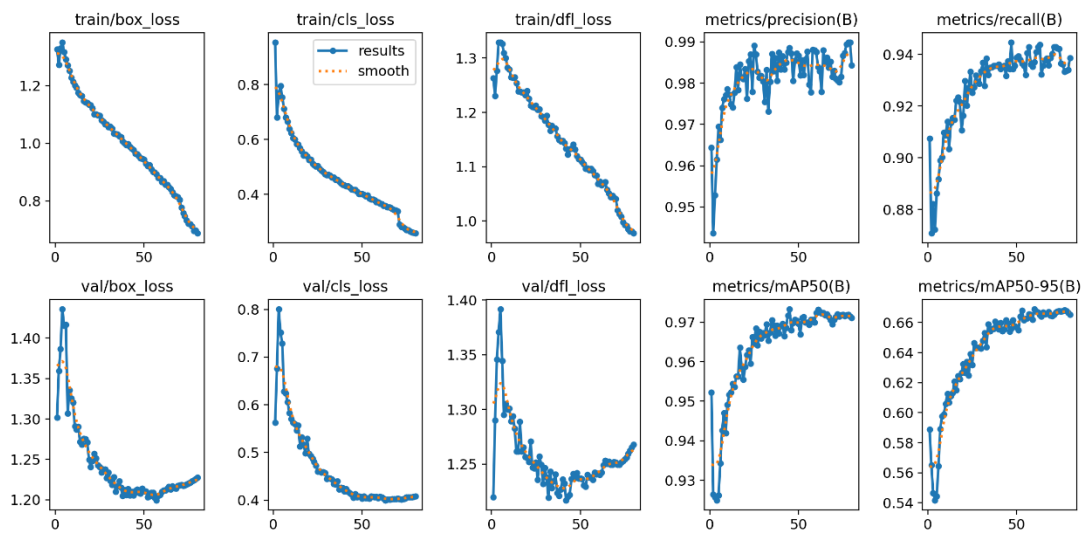


Figura 74. Resultados del entrenamiento de la red neuronal con YOLOv8m utilizando 80 épocas.

El segundo entrenamiento con 80 épocas de YOLOv8m muestra una disminución sostenida en las pérdidas de caja (box_loss), clasificación (cls_loss) y DFL (df_l_loss), aunque las fluctuaciones son más notorias en el conjunto de validación. Las métricas de precisión son altas y estables superiores a 0.97, el recall presenta más variabilidad aumentando progresivamente. En términos de mAP, los valores alcanzan 0.97 para mAP50 y 0.66 para mAP50-95, siendo ligeramente superiores al primer entrenamiento, presentando un desempeño adecuado para la detección.

En la Tabla 36 presenta un análisis numérico de las redes neuronales entrenadas con 60 épocas y 80 épocas respectivamente. Los resultados revelan una notable similitud entre ambas configuraciones en la mayoría de las métricas, indicando que la red neuronal presentó variaciones mínimas en las métricas evaluadas.

Tabla 36. Análisis de resultados de dos redes neuronales entrenadas con diferentes épocas.

Métricas	YOLOv8m-60 ep	YOLOv8m-80 ep
Train/box_loss	0.8451	0.68772
Trains/cls_loss	0.31111	0.25821
Train/df_l_loss	1.06997	0.97732
Precisión	0.98826	0.98431
Recall	0.92928	0.93862
mPA50	0.9683	0.97119
mpa50-95	0.65288	0.66532

Train/box_loss, Train/cls_loss, Train/dfl_loss: Estas medidas reflejan las pérdidas durante el entrenamiento del modelo. En ambos modelos, se nota una reducción en estas pérdidas conforme se incrementa la cantidad de épocas, lo que señala un mejoramiento en la adaptación del modelo a los datos.

Precision, Recall: Se relacionan con la habilidad de detectar objetos correctamente. En esta situación, ambos modelos poseen una precisión elevada (0.98) y un recall moderadamente elevado (entre 0.92 y 0.93). Por lo cual los modelos tienen la habilidad de reconocer adecuadamente los objetos pertinentes, aunque algunos pueden pasar por alto.

mPA50, mPA50-95: Estas medidas calculan el promedio de exactitud para distintos límites de IoU. En el modelo YOLOv8m-60, un mPA50 de 0.96 señala una precisión elevada cuando se admite una superposición del 50% entre la predicción y la verdad esencial. No obstante, existe un leve incremento en las etapas del modelo YOLOv8m-80. En cambio, mPA50-95 es un promedio de precisión más riguroso, tomando en cuenta un IoU del 50% al 95%. En este caso, ambos modelos exhiben una mejora conforme se amplían las temporadas de entrenamiento. En conclusión, las dos versiones del modelo YOLOv8m exhiben un desempeño muy satisfactorio en la identificación de objetos, mostrando avances mínimos en ciertas métricas conforme se incrementa el número de periodos de entrenamiento por lo cual se utilizó en el proyecto el modelo entrenado con 80 épocas.

3.1.14 Codificación de la visualización y detección de placas en automóviles en el ordenador

Se creó un código en Python que permite visualizar las imágenes de una cámara FPV como los datos de la placa vehicular y ubicación enviados desde el UAV. Para esto, se usó la biblioteca tkinter, que facilita el diseño de una interfaz gráfica. El programa se complementa con funciones para detectar automóviles con placas vehiculares en el video, obtener los caracteres con OCR y envió de información a la base de datos.

a. Bibliotecas requeridas

Las bibliotecas utilizadas para la implementación del sistema se detallan en la Tabla 37, presentada a continuación:

Tabla 37. Bibliotecas requeridas para el sistema

Biblioteca	Funciones Utilizadas	Características
Tkinter	Tk, Label, Canvas, Button, messagebox	Permite generar ventanas, mostrar imágenes y cuadros de diálogo.
PIL	Image.open, resize, ImageTk.PhotoImage	Permite trabajar con imágenes: abrir, redimensionar y convertirlas para usarlas en tkinter.
Subprocess	Popen, terminate, kill, poll	Permite ejecutar y controlar procesos externos como scripts Python desde el código.
Time	sleep, time	Proporciona funciones para pausas, medir intervalos o gestionar fechas y horas.
Cv2 (OpenCV)	VideoCapture, resize, rectangle, putText, imshow, waitKey, imwrite, imread, destroyAllWindows, release	Ofrece herramientas para capturar video, aplicar filtros y detectar objetos.
NumPy	np.array	Eficiente para manejar matrices y realizar operaciones matemáticas complejas.
Ultralytics	YOLO	Detecta objetos en imágenes o videos en tiempo real, utilizando modelos YOLO.
Sort	Sort	Implementa el algoritmo SORT para rastrear objetos en tiempo real.
Os	path.basename, path.join, makedirs, exists, environ, path.isfile	Maneja archivos, directorios y procesos del sistema operativo.
Datetime	datetime.now, strftime	Trabaja con fechas y horas, útil para generar marcas de tiempo.
Threading	Thread	Permite manejar hilos para realizar varias acciones simultáneamente.
Cloudinary	config, upload	Facilita subir, descargar y manipular archivos en la plataforma Cloudinary.
Google.cloud	vision.ImageAnnotatorClient, vision.Image	Utiliza la API de Google Vision para OCR y detección de objetos.
Re	match, sub	Trabaja con expresiones regulares para encontrar o reemplazar patrones específicos.

Biblioteca	Funciones Utilizadas	Características
Mysql.connector	Connect, cursor, execute, commit, close	Permite ejecutar consultas y administrar datos desde bases de datos MySQL.

b. Codificación de la interfaz gráfica para la portada

- ***Importación de librerías***

Dentro del script de programación para la portada se inicia importando las librerías requeridas para crear la interfaz, pestañas auxiliares para mostrar mensajes, creación de labels para texto e imágenes, inicio del proceso de captura de video y su procesamiento. Las librerías utilizadas en este apartado se presentan a continuación en la Figura 75.

```
# Importa Tkinter para crear interfaces gráficas.
import tkinter as tk
from tkinter import messagebox

# Importa PIL para manejar imágenes.
from PIL import Image, ImageTk

# Importa subprocess para ejecutar comandos
externos y time para manejar pausas.
import subprocess
import time
```

Figura 75. Librerías utilizadas en portada.

- ***Parámetros para creación de la interfaz gráfica***

La interfaz gráfica está conformada por elementos como labels de texto e imágenes y botones para llamados a la acción dentro de la ventana con las configuraciones respectivas, el código de como formar los elementos principales se detalla en la Figura 76.

```

# Crear la ventana principal con título, tamaño fijo y fondo de color.
ventana = tk.Tk()
ventana.title("Portada de la Aplicación")
ventana.geometry("640x480")
ventana.resizable(False, False)
ventana.configure(bg="lightblue")

# Agregar el título de la aplicación.
titulo = tk.Label(ventana, text="Universidad Técnica de Ambato",
                  font=("Times New Roman", 25, "bold"), bg="lightblue")
titulo.place(x=50, y=10)

# Cargar y mostrar la imagen del logo.
imagen1 = Image.open("UTA.webp").resize((120, 120), Image.LANCZOS)
imagenTk = ImageTk.PhotoImage(imagen1)
label_imagen = tk.Label(ventana, image=imagenTk, bg="lightblue")
label_imagen.image = imagenTk
label_imagen.place(x=510, y=12)

# Dibujar una línea horizontal decorativa.
canvas = tk.Canvas(ventana, width=640, height=2, bg="lightblue", highlightthickness=0)
canvas.place(x=0, y=170)
canvas.create_line(20, 1, 620, 1, fill="black")

# Agregar un botón para iniciar el programa.
boton_iniciar = tk.Button(ventana, text="Iniciar", font=("Times New Roman", 14),
                          bg="green", fg="white", command=iniciar_programa)
boton_iniciar.place(x=400, y=420)

```

Figura 76. Código de parámetros de la interfaz.

tk.Tk(): Se crea la ventana principal para la portada.

ventana.geometry("640x480"): Define las dimensiones de la ventana en 640 píxeles de ancho y 480 píxeles de alto.

ventana.configure(bg="lightblue"): Se define el color de fondo de la ventana en azul claro.

tk.Label(ventana, text="texto", font=("Times New Roman", 25, "bold"), bg="lightblue"): Crea un label en la ventana que contiene texto con fuente "Times New Roman", tamaño 25, fondo azul claro.

Image.open("imagen.png"): Se carga una imagen determinada utilizando la biblioteca PIL.

ImageTk.PhotoImage(imagen): Convierte la imagen redimensionada a un objeto para ser utilizado en tkinter.

tk.Label(ventana, image=imagen_tk, bg="lightblue"): Crea un label en la ventana para mostrar la imagen determinada en la ventana.

tk.Canvas(ventana,width=640,height=2,bg="lightblue",highlightthickness=0):
Crea un widget Canvas de 640 píxeles de ancho y 2 píxeles de alto, sin bordes destacados.

place(): Define la posición en la que se ubica el elemento dentro de la ventana.

tk.Button(ventana, text = "Iniciar", Font = ("Times New Roman", 14), bg="green", fg="white", command=iniciar_programa): Crea un botón sobre la ventana con texto "Iniciar" en la fuente "Times New Roman", tamaño 14, fondo verde, texto blanco; además está vinculado con la función iniciar_programa para que se active al hacer clic.

ventana.mainloop(): Da inicio al bucle principal de tkinter, manteniendo la ventana abierta y a la espera a que ocurran eventos (como clics de botón) hasta que se cierre la aplicación, esta interfaz se muestra en la Figura 77.



Figura 77. Interfaz gráfica con la portada.

- **Funciones para iniciar y cerrar el programa**

En el código presentado en la Figura 78 hay dos funciones que se encargan de la ejecución de un script con nombre de archivo `det_vehiculos.py`, dentro de la interfaz gráfica.

```
# Función para iniciar el programa
def iniciar_programa():
    global proceso
    if proceso is None:
        try:
            # Ejecuta "det_vehiculos.py" como un subprocess.
            proceso = subprocess.Popen(["python", "det_vehiculos.py"],
                                       stdout=subprocess.PIPE, stderr=subprocess.PIPE)
            messagebox.showinfo("Estado", "El programa se está ejecutando.")
        except Exception as e:
            messagebox.showerror("Error", f"No se pudo iniciar el programa: {e}")
    else:
        messagebox.showwarning("Advertencia", "El programa ya está en ejecución.")

# Función para cerrar la aplicación y detener el proceso
def cerrar_programa():
    global proceso
    if proceso is not None:
        respuesta = messagebox.askyesno("Salir", "¿Deseas detener el programa y cerrar?")

        if respuesta:
            try:
                proceso.terminate() # Intenta finalizar el proceso.
                time.sleep(1) # Espera para asegurar su detención.
                if proceso.poll() is None: # Si sigue activo, lo fuerza a cerrarse.
                    proceso.kill()
                proceso = None
                messagebox.showinfo("Estado", "El programa se ha detenido.")
            except Exception as e:
                messagebox.showerror("Error", f"No se pudo detener el programa: {e}")
        else:
            messagebox.showinfo("Estado", "El programa sigue en ejecución.")
    else:
        messagebox.showinfo("Estado", "No hay un programa en ejecución.")

# Cierra la ventana principal.
ventana.quit()
```

Figura 78. Código de funciones para iniciar y cerrar el programa.

iniciar_programa(): Comprueba si existe un "proceso" en ejecución; si no lo está, intentará enlazarse para ejecutar el archivo.

- El comando `subprocess.Popen` se encarga de ejecutar el archivo `det_vehiculos.py` si no está siendo ejecutado.

- Dentro de esta función se utiliza la herramienta messagebox para habilitar cuadros de diálogo con mensajes de advertencia.

cerrar_programa(): Permite detener el proceso activo mediante una condicional y una solicitud de entrada antes de cerrar el script, si el proceso no se detiene, se forzará la terminación.

- Dentro de esta función también se utiliza la herramienta messagebox para habilitar mensajes informativos.
- La línea de código proceso.terminate(), intenta cerrar el proceso de manera ordenada.
- time.sleep(1) detiene la ejecución durante 1 segundo para dar tiempo a que el proceso se cierre.
- Si el proceso sigue activo proceso.kill() forzara su terminación.
- Finalmente, se cierra la aplicación desde la ventana principal ventana.quit().

c. Codificación para recepción de video y detección de automóviles

- ***Importación de librerías***

Las librerías de la Figura 79 cv2 (OpenCV) se utilizan para capturar el video, procesarlo y mostrarlo en pantalla. numpy realiza operaciones matemáticas y manejo eficiente de matrices. YOLO (de la librería ultralytics) es utilizado para la detección de automóviles, ideal para identificar objetos en tiempo real. Sort es un algoritmo utilizado para seguimiento de los automóviles en el video capturado. Os y datetime gestionan los directorios y archivos que se generen en el proceso, time y threading permiten medir tiempos y ejecutar procesos simultáneos, que permiten trabajar en conjunto para empezar a crear el sistema en el archivo actual det_vehiculos.py.

```

# Importa OpenCV para procesamiento de imágenes.
import cv2

# Importa NumPy para operaciones con matrices.
import numpy as np

# Importa YOLO de Ultralytics para detección de objetos.
from ultralytics import YOLO

# Importa SORT para el seguimiento de objetos.
from sort.sort import Sort

# Importa módulos del sistema operativo y gestión de tiempo.
import os
import datetime
import time

# Permite ejecutar tareas en paralelo.
import threading

```

Figura 79. Librerías utilizadas para procesamiento de video y detección de automóviles.

- ***Importación de YOLO para detección de automóviles***

En la Figura 80 se desarrolla la lectura del modelo para la detección de objetos, en este caso para detectar automóviles, se crea una instancia de clase YOLO con su archivo por defecto "yolov8m.pt".

```

vehicle_model = YOLO('yolov8m.pt')

```

Figura 80. Lectura del modelo yolov8m.pt.

- ***Programación para detección de automóviles***

En la Figura 81 se implementan las líneas de comando para capturar el video por medio de conexión USB con el video receptor Eachine ROTG01, en caso de no encontrar conectado el dispositivo se imprime una advertencia.

```

# Inicia la captura de video desde la cámara (índice 1).
cap = cv2.VideoCapture(1)

# Reduce el tamaño del buffer para minimizar el retraso en la transmisión.

cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

# Verifica si la captura de video se abrió correctamente.
if not cap.isOpened():
    print("Error: No se puede abrir el video.")
    exit()

```

Figura 81. Captura de video para procesamiento.

En la Figura 82 se proponen los parámetros para configuración de Sort en el seguimiento de vehículos, `max_age` representa el número máximo de frames para mantener un objeto sin detección, `min_hits` es el número mínimo de detecciones antes de asignar un ID y `iou_threshold` como el umbral de IOU para la asociación de objetos

```

# Inicializa el rastreador SORT con parámetros personalizados.
max_age = 30 # Máximo número de frames sin detección antes de eliminar el objeto.
min_hits = 1 # Mínimo de detecciones requeridas para asignar un ID a un objeto.
iou_threshold = 0.7 # Umbral de IOU para asociar objetos entre frames.

# Crea una instancia del rastreador SORT con los parámetros especificados
tracker = Sort(max_age=max_age, min_hits=min_hits, iou_threshold=iou_threshold)

```

Figura 82. Parámetros Sort para seguimiento de automóviles.

Se debe almacenar la captura del vehículo detectado para su posterior procesamiento, por lo cual se verifica que exista un directorio identificado como "capturas", en caso de no existir lo va a crear esto se muestra en la Figura 83.

```

# Crea una carpeta llamada 'capturas' para guardar las imágenes capturadas.
output_folder = "capturas"
# No genera error si la carpeta ya existe.
os.makedirs(output_folder, exist_ok=True)

```

Figura 83. Creación de directorio para almacenamiento de imágenes.

Se crea un bucle infinito presentado en la Figura 84 que se ejecutará hasta que se interrumpa con un `break`, capturando el tiempo inicial para medir el intervalo que toma

procesar cada frame, si la imagen es detectada continúa el proceso caso contrario se imprime una advertencia rompiendo el bucle. Verifica cada cuanto se debe procesar el frame y lo redimensiona para llamar al modelo de detección de vehículos.

```
# Coordenadas y dimensiones de la zona de interseccion.
roi_x, roi_y, roi_w, roi_h = 300, 50, 20, 250
# Coordenadas para la zona de procesamiento.
x, y, w, h = 60, 0, 520, 430

# Bucle para procesar el video frame por frame.
while True:
    start_frame_time = time.time() # Marca el inicio del procesamiento del frame.
    ret, frame = cap.read() # Captura el siguiente frame.
    if not ret:
        print("Fin del video o error al recibir el frame.")
        break

    frame_count += 1 # Incrementa el contador de frames.
    if frame_count % frame_skip != 0: # Omite frames según la configuración.
        continue

    frame = cv2.resize(frame, (640, 480)) # Redimensiona el frame a 640x480.

    roi_frame = frame[y:y + h, x:x + w] # Extrae la región de interés del frame.

    # Realiza la detección de vehículos en la región de interés.
    vehicle_results = vehicle_model(roi_frame)
```

Figura 84. Bucle principal de procesamiento de video.

Para el procesamiento de resultados de detección se crea una lista para almacenar las detecciones de los automóviles y otro bucle para recorrer cada resultado obtenido y extraer las cajas delimitadoras (bounding boxes) con el nivel de confiabilidad, si este nivel supera el umbral mínimo definido y detecta que existe un automóvil identificado con el número 2 en YOLO, extraerá las coordenadas y se convertirá en un array de numpy, esta información más el valor de confianza se agregan a lista de detecciones, el código se visualiza en la Figura 85.

```

# Lista para almacenar las detecciones
detections = []

# Iterar sobre los resultados de la detección de vehículos
for result in vehicle_results:
    boxes = result.boxes # Obtener las cajas delimitadoras de cada detección.

    # Iterar sobre cada caja para extraer detalles
    for box in boxes:
        class_id = int(box.cls) # Obtener el ID de la clase del objeto detectado.
        confidence = box.conf[0] # Obtener la confianza de la detección.

        # Filtrar solo vehículos según sus IDs en el dataset COCO (clase 2) y el umbral de confianza
        if class_id in [2] and confidence > confidence_threshold:
            # Convertir coordenadas de la caja a formato NumPy y ajustarlas a la posición de la ROI
            x1, y1, x2, y2 = box.xyxy[0].cpu().numpy() # Coordenadas de la caja.
            x1 += x
            y1 += y
            x2 += x
            y2 += y

            # Añadir la detección (coordenadas ajustadas y confianza) a la lista de detecciones.
            detections.append([x1, y1, x2, y2, confidence.cpu().numpy()])

```

Figura 85. Coordenadas de detecciones y nivel de confianza.

Para la actualización de automóviles rastreados en la Figura 86 se verifica si hay una detección en la lista para llamar al método update del objeto tracker y actualiza la lista de objetos rastreados con las nuevas detecciones, caso contrario inicializa tracked_objects como una lista vacía.

```

# Si hay detecciones, actualiza las posiciones de los vehículos
if len(detections) > 0:
    # Actualiza el rastreador con las nuevas detecciones.
    tracked_objects = tracker.update(np.array(detections))
else:
    # Si no hay detecciones, no hay objetos a rastrear.
    tracked_objects = []

```

Figura 86. Actualización de la detección del automóvil.

En la Figura 87 se codifica el procesamiento de objetos que recorre cada automóvil rastreado extrayendo las coordenadas y el ID, y de ser necesario se asignará uno nuevo para obtener un ID estable. Se dibuja un rectángulo verde alrededor del automóvil detectado en el frame con una etiqueta que presenta el ID y el nivel de confianza de que sea un vehículo en determinada posición. Para que la captura sea tomada se verifica que el ID del objeto no está en el diccionario de IDs capturados y que el automóvil esté dentro de la zona de interés definida durante al menos 3 segundos. Las

imágenes se almacenan en el directorio con un nombre específico obteniendo la información del ordenador. Posteriormente se llama a la función creada dentro del mismo archivo `detectar_placas_async()`, encargada de la conexión al archivo que se encargará de la detección de placas vehiculares, esto es posible ya que se utiliza programación por hilos para ejecutar múltiples tareas de manera simultánea.

```
# Iterar sobre los objetos rastreados
for obj in tracked_objects:
    # Obtener las coordenadas de la caja y el ID del objeto
    x1, y1, x2, y2, obj_id = obj
    p1 = (int(x1), int(y1))
    p2 = (int(x2), int(y2))

    # Asignar un nuevo ID si no existe en el mapa
    if obj_id not in id_mapping:
        id_mapping[obj_id] = next_id
        next_id += 1

    stable_id = id_mapping[obj_id] # Obtener el ID estable del objeto

    # Dibujar la caja delimitadora y el ID en el fotograma
    cv2.rectangle(frame, p1, p2, (0, 255, 0), 1, 1)
    label_with_confidence = f"ID {stable_id}: {confidence:.2f}"

    # Posicionar el texto debajo de la caja delimitadora
    text_position = (p1[0], p2[1] - 250) # Ajuste de posición

    # Colocar el texto con el ID y la confianza en el fotograma
    cv2.putText(frame, label_with_confidence, text_position,
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)

    # Verificar si el vehículo está dentro de la ROI
    if (p1[0] < roi_x + roi_w and p2[0] > roi_x and
        p1[1] < roi_y + roi_h and p2[1] > roi_y):

        # Obtener el tiempo actual y comprobar si se capturó previamente
        current_time = time.time()
        if stable_id not in captured_ids:
            captured_ids[stable_id] = {"time": current_time, "captured": False}

        # Verificar si han pasado 3 segundos desde la última captura
        elapsed_time = time.time() - captured_ids[stable_id]["time"]
        if elapsed_time >= 3 and not captured_ids[stable_id]["captured"]:
            # Guardar la imagen capturada con un timestamp
            timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
            capture_filename = os.path.join(output_folder, f"vehiculo_{timestamp}.jpg")
            cv2.imwrite(capture_filename, frame) # Guardar la captura
            print(f"Captura guardada: {capture_filename}")

            # Iniciar la detección de placas en un hilo separado
            threading.Thread(target=detectar_placas_async, args=(capture_filename,)).start()
            captured_ids[stable_id]["captured"] = True # Marcar que ya se capturó el vehículo
```

Figura 87. Procesamiento de objetos rastreados.

Las imágenes se almacenan en el directorio creado "capturas" con un nombre único y la imagen en formato jpg está identificada con la fecha y hora en que se realiza la captura como se muestra en la Figura 88.

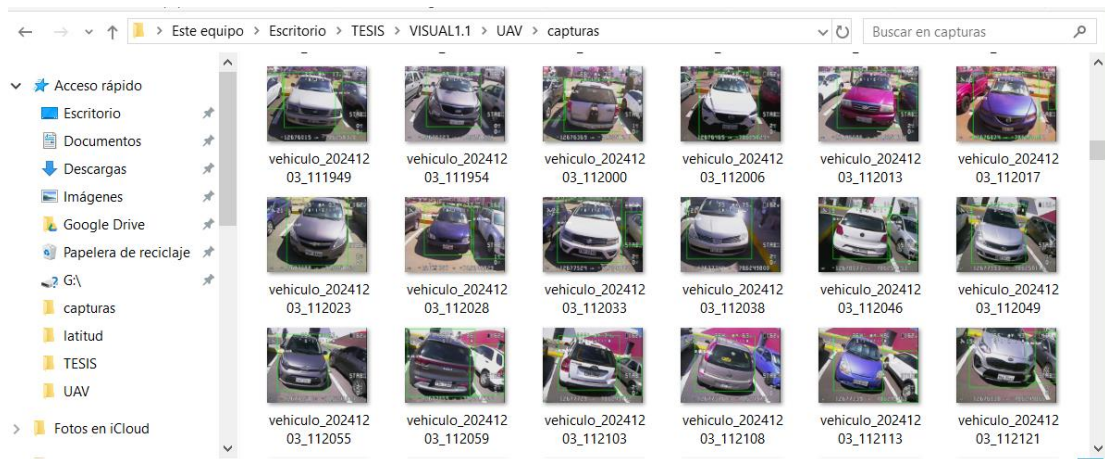


Figura 88. Directorio para almacenamiento de capturas de automóviles.

Para llamar a una función que se encuentra en otro archivo, se considera que los archivos `det_vehiculos.py` y `filtrado.py` estén dentro del mismo directorio, dado lo anterior el código de la función y los requerimientos se presentan en la Figura 89.

```
# Importar el módulo 'filtrado', que contiene la
función de detección de placas
import filtrado

# Función para detectar placas de manera asíncrona
def detectar_placas_async(capture_filename):
    filtrado.detectar_placas(capture_filename)
```

Figura 89. Función de enlace al archivo `filtrado.py`.

El proceso del video capturado con la ejecución del archivo `det_vehiculos.py` para detección de automóviles se muestra en la Figura 90, además el código compacto del archivo `det_vehiculos.py` se encuentra en el Anexo F.



Figura 90. Reconocimiento de vehículos en el video.

d. Codificación para detección de placas vehiculares y enlace a cloudinary

- **Importación de librerías**

En el archivo filtrado.py es necesario importar las librerías de la Figura 91 como cv2 que se utiliza para capturar la imagen y procesarla, YOLO es utilizado para la detección de placas en automóviles incorporando el modelo creado. Os gestiona los directorios y archivos que se generen en el proceso.

```
# Importa OpenCV para manipulación de imágenes y video
import cv2

# Importa el modelo YOLO para la detección de objetos
from ultralytics import YOLO

# Importa el módulo OS para manejar rutas y archivos del sistema operativo
import os
```

Figura 91. Librerías utilizadas para detección de placas.

- **Función para detección de placas vehiculares**

La función detectar_placas() se encarga de la detección y segmentación de la placa vehicular en la imagen del automóvil capturada, utilizando YOLO('best80.pt') donde

se aplica el modelo entrenado para la detección de placas, este comando se muestra en la Figura 92.

```
# Función para detectar placas
def detectar_placas(image_path):

    # Cargar el modelo YOLOv8
    model = YOLO('best80.pt')
```

Figura 92. Función para detectar placas en automóviles.

Ahora se crean las variables en la Figura 93 para la asignación de nombre a las carpetas para almacenar las imágenes, placas (automóviles con placas), no_placas (automóviles sin placa), recorte_placas (segmentación de la placa encontrada), latitud (latitud del vehículo encontrado), longitud (longitud del vehículo encontrado).

```
# Definir carpetas de salida para almacenar los resultados
de la detección de placas
output_folder_placas = "placas"
output_folder_no_placas = "no_placas"
output_folder_recortes = "recorte_placas"
output_folder_latitud = "latitud"
output_folder_longitud = "longitud"
```

Figura 93. Directorios para almacenamiento de imágenes.

En la Figura 94 con `cv2.imread(image_path)` se lee la imagen del automóvil capturada desde su ruta específica, para la aplicación del modelo con un umbral mínimo para validar la detección. Con el comando `os.path.basename(image_path)` se obtiene el nombre de la imagen original.

```

# Leer la imagen
image = cv2.imread(image_path)

# Realizar la detección
results = model(image)

# Umbral mínimo de confianza
confidence_threshold = 0.7

# Bandera para verificar si se detectó alguna placa
placa_detectada = False

# Obtener el nombre de la imagen original
image_name = os.path.basename(image_path)

```

Figura 94. Aplicación del modelo para detectar placas.

En el bucle de la Figura 95 se recorren los resultados obtenidos de la detección, verificando que el nivel de confianza sea mayor que el umbral mínimo establecido en la Figura 94, obteniendo así la información de las coordenadas de la caja limitadora con `map(int,box.xyxy[0])` y el ID de la clase con `box.cls[0]`.

```

# Mostrar los resultados de la detección
for result in results: # Iterar sobre cada resultado obtenido del modelo YOLO
    boxes = result.boxes # Extraer las cajas delimitadoras detectadas en la imagen
    for box in boxes: # Iterar sobre cada caja detectada
        confidence = box.conf[0] # Obtener el nivel de confianza de la detección
        if confidence >= confidence_threshold: # Filtrar detecciones con baja confianza
            placa_detectada = True # Indicar que se ha detectado al menos una placa
            x1, y1, x2, y2 = map(int, box.xyxy[0]) # Obtener coordenadas de la caja delimitadora
            class_id = box.cls[0] # Obtener la clase del objeto detectado

            # Dibujar el rectángulo de la detección en la imagen
            cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)

            # Crear la etiqueta con la clase y la confianza de la detección
            label = f"Class: {int(class_id)}, Conf: {confidence:.2f}"

            # Mostrar la etiqueta en la imagen, posicionándola sobre la detección
            cv2.putText(image, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
                        0.5, (0, 255, 0), 2)

```

Figura 95. Interacción con placa detectada.

Con la información adquirida se dibuja un rectángulo alrededor de la detección, colocando una etiqueta con la clase y el nivel de confianza de la detección, esto se observa en la Figura 96.



Figura 96. Placa detectada en vehículo.

En la Figura 97 se detalla como los datos de las coordenadas de la caja limitadora se fijan para recortar la sección, verificando si el directorio para almacenamiento existe, caso contrario lo crea.

```
# Recortar la zona detectada
recorte = image[(y1+15):y2, x1:x2]

# Crear la carpeta de salida para recortes si no existe
if not os.path.exists(output_folder_recortes):
    os.makedirs(output_folder_recortes)

# Ruta de salida para el recorte
recorte_path = os.path.join(output_folder_recortes, image_name)

# Guardar el recorte
cv2.imwrite(recorte_path, recorte)
```

Figura 97. Segmentación de placa.

En caso de detectar una placa las imágenes se almacenan en el directorio creado "recorte_placas" con el nombre de la imagen original, la carpeta de las segmentaciones recortadas se destaca en la Figura 98.

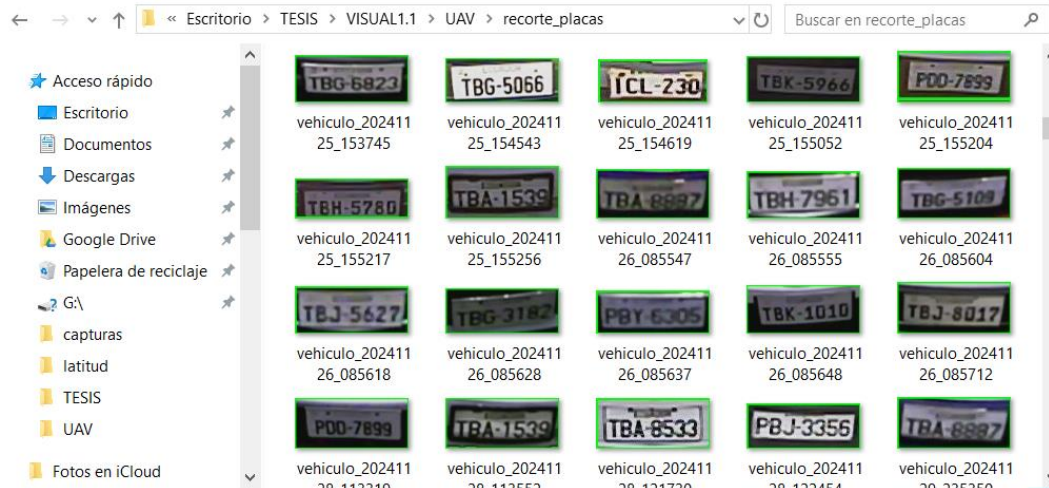


Figura 98. Directorio para almacenamiento de placas segmentadas.

Dado que las coordenadas de latitud y longitud en la imagen recibida desde la cámara del dron se encuentran en una posición fija, se pueden segmentar para procesar esa información y obtener la ubicación del vehículo detectado, para esto se recorta esta parte y se guarda en carpetas, la codificación es presentada en la Figura 99.

```
# Crear las carpetas de salida si no existen
if not os.path.exists(output_folder_latitud):
    os.makedirs(output_folder_latitud)
if not os.path.exists(output_folder_longitud):
    os.makedirs(output_folder_longitud)

# Definir las regiones de interés (ROI) donde se encuentran los datos de latitud y longitud
zone_x, zone_y, zone_w, zone_h = 80, 435, 220, 40 # Coordenadas para latitud
zone_x2, zone_y2, zone_w2, zone_h2 = 330, 435, 250, 40 # Coordenadas para longitud

# Extraer y guardar los recortes de latitud y longitud
roi_lat = image[zone_y:zone_y + zone_h, zone_x:zone_x + zone_w]
roi_lon = image[zone_y2:zone_y2 + zone_h2, zone_x2:zone_x2 + zone_w2]

# Guardar los recortes en sus respectivas carpetas
cv2.imwrite(os.path.join(output_folder_latitud, image_name), roi_lat)
cv2.imwrite(os.path.join(output_folder_longitud, image_name), roi_lon)
```

Figura 99. Segmentación de latitud y longitud de la imagen.

De cada frame capturado se extrae la información de latitud y longitud del GPS incorporado al dron, las capturas se almacenan en los directorios "latitud" y "longitud" respectivamente como se muestra en la Figura 100.

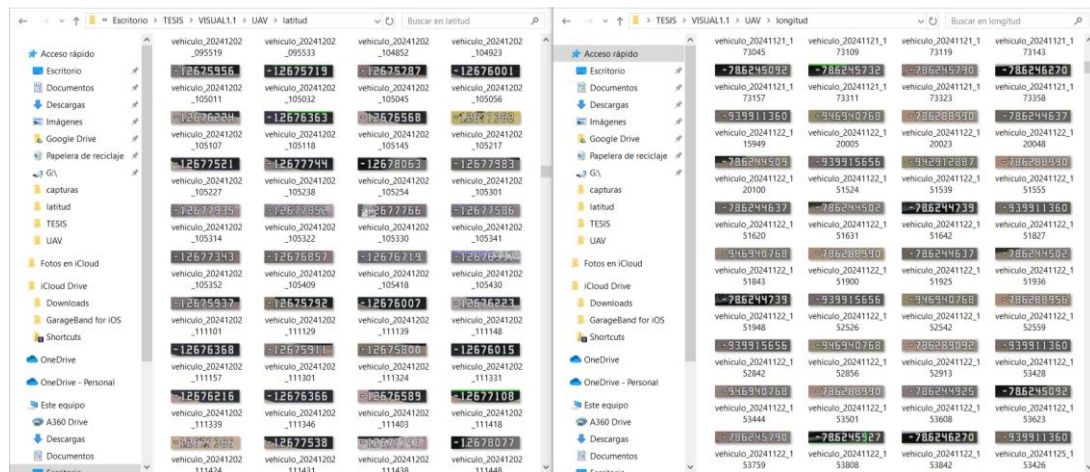


Figura 100. Directorios para almacenamiento de datos de ubicación.

Estas imágenes procesadas se guardan y se envían a la plataforma web clouidinary que actúa como un respaldo de archivos online, con la factibilidad que también convierte la imagen en una URL, que permite acceder a la captura desde un navegador web. Se consideran dos casos posibles, el primero es que el vehículo detectado contenga una placa y el segundo es que el vehículo detectado no contenga placa, en la Figura 101 también se muestra como conectarse a los scripts recortes_placas_ubi.py y recortes_solo_ubi.py.

```

import recortes_placas_ubi
import recortes_solo_ubi

# Crear las carpetas de salida para clasificar imágenes con y sin placas detectadas
if not os.path.exists(output_folder_placas):
    os.makedirs(output_folder_placas)
if not os.path.exists(output_folder_no_placas):
    os.makedirs(output_folder_no_placas)

# Si se detectó una placa, guarda la imagen en la carpeta correspondiente
if placa_detectada:
    output_path = os.path.join(output_folder_placas, image_name)

    # Subir la imagen a Cloudinary y obtener la URL
    url = str(upload_image_to_cloudinary(image_path))

    # Llamar al módulo que procesa los recortes de la placa junto con la ubicación
    images_info = [recorte_path, recorte_lat, recorte_lon]
    recortes_placas_ubi.main(images_info, url)

# Si no se detectó una placa, clasifica la imagen en la carpeta de "no placas"
else:
    output_path = os.path.join(output_folder_no_placas, image_name)

    # Subir la imagen a Cloudinary y obtener la URL
    url = str(upload_image_to_cloudinarynp(image_path))

    # Llamar al módulo que solo procesa la ubicación sin placa detectada
    images_info = [recorte_lat, recorte_lon]
    recortes_solo_ubi.main(images_info, url)

# Guardar la imagen procesada en la carpeta correspondiente
cv2.imwrite(output_path, image)

# Mensaje de advertencia si no se detectó ninguna placa en la imagen
if not placa_detectada:
    print(f"No se detectó ninguna placa en la imagen: {image_path}")

```

Figura 101. Clasificación de vehículos con placa y sin placa.

Estos frames procesados se almacenan en las carpetas "placas" y "no_placas" según sea el caso, como se muestra en la Figura 102.

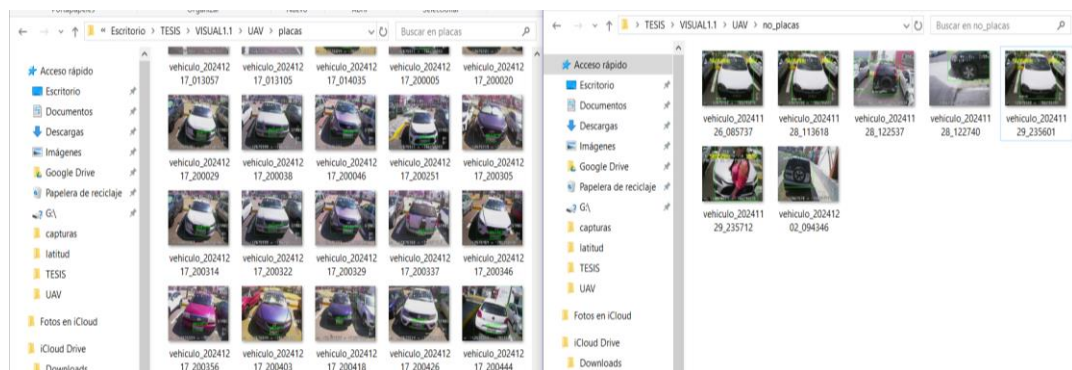


Figura 102. Directorio de los vehículos con placa y sin placa.

- ***Función para obtener enlace de clouinary***

Para conectarse a clouinary desde una función creada en Python, primero hay que crear una cuenta en la plataforma para conseguir credenciales de acceso como cloud name, API key, API secret. Para conseguir los parámetros dentro de la plataforma hay que dirigirse al apartado "Programmable Media" y en "Dashboard" generar una API key que se presenta en la Figura 103.

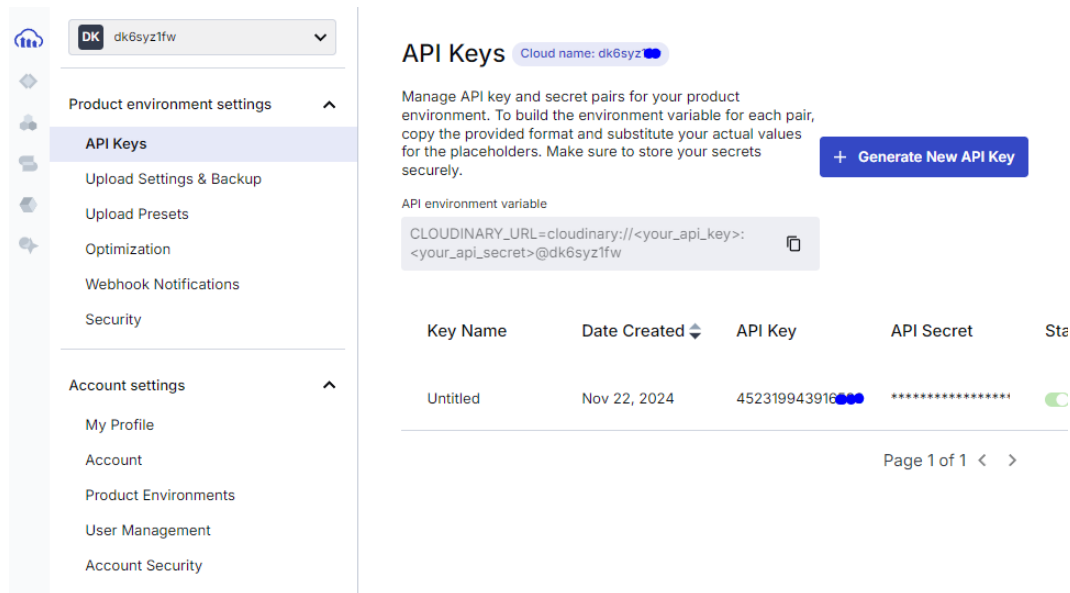


Figura 103. Obtención de API Key en clouinary.

Con las claves de acceso generadas se crea la función `upload_image_to_clouinary()`, para enlazarse desde el ordenador a clouinary en la Figura 104, desde la línea de comando `cloudinary.uploader.upload(image_path,folder="PLACAS")` se carga la imagen en la carpeta "PLACAS" de la biblioteca multimedia de clouinary cuando se realiza la detección.

```

import cloudinary
import cloudinary.uploader
import cloudinary.api

def upload_image_to_cloudinary(image_path):

    # Configuración de Cloudinary
    cloudinary.config(
        cloud_name="cloud_name",
        api_key="api_key",
        api_secret="api_secret"
    )

    try:
        # Subir la imagen
        response = cloudinary.uploader.upload(image_path, folder="PLACAS")
        # Retornar la URL de la imagen
        return response['secure_url']

    except Exception as e:
        return f"Error al subir la imagen: {e}"

```

Figura 104. Creación URL desde cloudinary en carpeta "PLACAS".

Caso contrario se carga en la carpeta "NO_PLACAS" con la función `upload_image_to_cloudinarynp()` presentada en la Figura 105, que realiza un proceso similar, ya con el archivo en cloudinary a partir de las dos funciones se genera un enlace de acceso a la imagen. El código del archivo `filtrado.py` completo se expone en el Anexo G.

```

def upload_image_to_cloudinarynp(image_path):

    # Configuración de Cloudinary
    cloudinary.config(
        cloud_name="cloud_name",
        api_key="api_key",
        api_secret="api_secret"
    )

    try:
        # Subir la imagen
        response = cloudinary.uploader.upload(image_path, folder="NO_PLACAS")
        # Retornar la URL de la imagen
        return response['secure_url']

    except Exception as e:
        return f"Error al subir la imagen: {e}"

```

Figura 105. Creación URL desde cloudinary en carpeta "NO_PLACAS".

- **Funciones para detección de caracteres con API de Cloud Vision**

Si en caso se detecta una placa se llama a la función `recortes_placas_ubi.main(images_info,url)` desde la Figura 101 para la conexión al archivo `recortes_placas_ubi.py` presentado en el Anexo H, para la posterior extracción de caracteres de placa vehicular e información de latitud y longitud, como se visualiza en el código de la Figura 106.

```
import os
from google.cloud import vision

# Inicialización de variables para almacenar los textos detectados
text1, text2, text3 = "", "", ""

# Establecer la ubicación del archivo de credenciales de Google Cloud
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'Ubicacion del archivo vision_key.json'

def detect_text(path):
    client = vision.ImageAnnotatorClient() # Crear cliente para la API de Google Vision
    with open(path, "rb") as image_file: # Leer la imagen desde el archivo
        content = image_file.read()

    # Convertir la imagen a un objeto compatible con la API
    image = vision.Image(content=content)

    # Llamar a la API para realizar la detección de texto
    response = client.text_detection(image=image)

    texts = response.text_annotations # Obtener los resultados de la detección de texto

    # Si ocurre un error, se lanza una excepción con el mensaje
    if response.error.message:
        raise Exception(
            "{}\nFor more info on error messages, check: "
            "https://cloud.google.com/apis/design/errors".format(response.error.message)
        )

    # Si se encontró texto, devolver el texto de la primera anotación
    if texts:
        return texts[0].description
    else:
        return ""

def main(image_paths, result):
    # Iterar sobre las rutas de las imágenes
    for i, image_path in enumerate(image_paths):
        # Verificar si la ruta corresponde a un archivo válido
        if os.path.isfile(image_path):
            # Detectar el texto en la imagen
            text = detect_text(image_path)

            # Crear una variable con nombre dinámico para almacenar el texto
            var_name = f"text{i+1}"
            globals()[var_name] = text
        else:
            print(f"File {image_path} does not exist.") # Mensaje si el archivo no existe

    # Llamar a la función para procesar el texto y crear un enlace
    procesar_placa_y_crear_enlace(text1, text2, text3, result)
```

Figura 106. Funciones para extracción de caracteres en caso de detectar placa.

El otro caso es cuando no se detecta una placa, en cambio se llama a la función `recortes_solo_ubi.main(images_info,url)` desde la Figura 101 para la conexión al archivo `recortes_solo_ubi.py` presentado en el Anexo I , para la extracción de caracteres de la información de latitud y longitud, como se visualiza en el código de la Figura 107.

```
import os
from google.cloud import vision

# Variables para almacenar el texto detectado en las imágenes
text1, text2 = "", ""

# Configuración de la autenticación con Google Cloud Vision API
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'Ubicacion del archivo vision_key.json'

def detect_text(path):
    client = vision.ImageAnnotatorClient() # Crear un cliente para la API de Google Vision

    with open(path, "rb") as image_file: # Leer el archivo de la imagen
        content = image_file.read()

    # Convertir el contenido a una imagen que la API puede procesar
    image = vision.Image(content=content)

    # Realizar la detección de texto
    response = client.text_detection(image=image)

    texts = response.text_annotations # Obtener las anotaciones de texto detectado

    # Si ocurre un error, se lanza una excepción con el mensaje
    if response.error.message:
        raise Exception(
            "{}\nFor more info on error messages, check: "
            "https://cloud.google.com/apis/design/errors".format(response.error.message)
        )

    # Si se encontró texto, devolver el texto de la primera anotación
    if texts:
        return texts[0].description
    else:
        return "" # Si no se encontró texto, devolver cadena vacía

def main(image_paths, result):
    # Iterar sobre cada ruta de imagen
    for i, image_path in enumerate(image_paths):
        # Verificar si la imagen existe
        if os.path.isfile(image_path):
            # Detectar el texto en la imagen
            text = detect_text(image_path)

            # Crear una variable con nombre dinámico para almacenar el texto
            var_name = f"text{i+1}"
            globals()[var_name] = text

        else:
            print(f"File {image_path} does not exist.") # Mensaje si la imagen no existe

    # Llamar a la función de procesamiento con los textos detectados
    procesar_ubi_y_crear_enlace(text1, text2, result)
```

Figura 107. Funciones para extracción de caracteres en caso de no detectar placa.

- **Función para corrección de texto extraído y creación de enlace de ubicación**

Previamente se debe tener una cuenta en Google cloud para posterior generación de una clave API key para hacer uso de la herramienta de OCR, en los productos se debe buscar la opción "Cloud Vision API" para administrarla. Dentro de administración se selecciona "Credenciales" para la creación de credencial como cuenta de servicio.

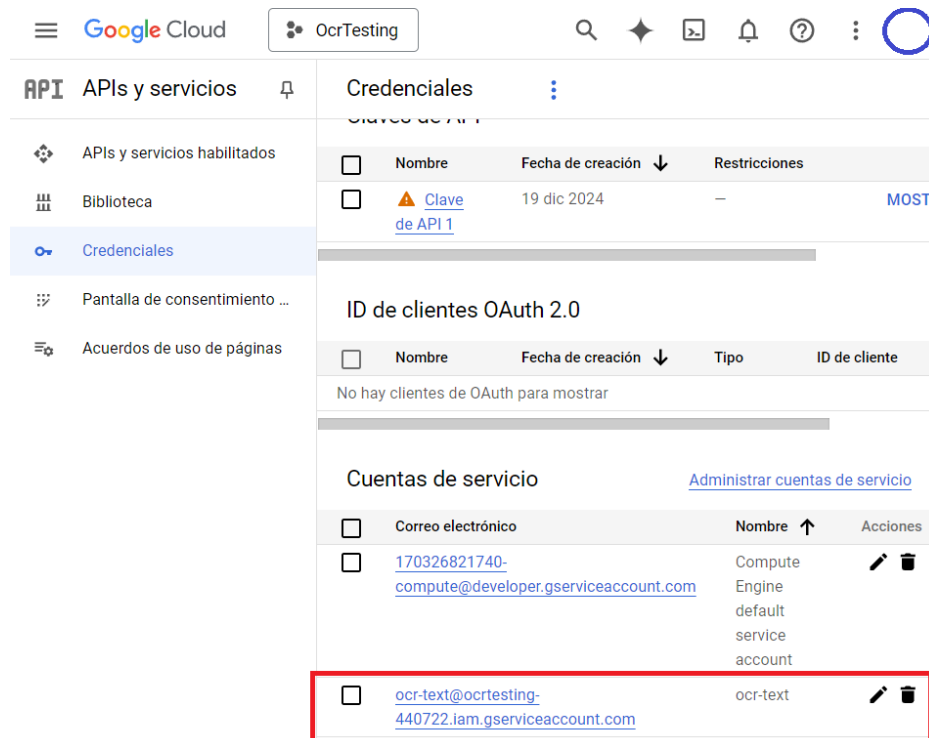


Figura 108. Cuenta de servicio creada en Cloud Vision API.

Dentro de la cuenta de servicio creada en el apartado de claves se debe agregar una nueva para utilizarla en el proyecto como se ve en la Figura 109.



Las claves de cuenta de servicio podrían poner en riesgo la seguridad si se ven comprometidas. Te recomendamos que no descargues claves de cuenta de servicio y que, en su lugar, uses la [Federación de identidades para cargas de trabajo](#). [Obtén más información sobre la mejor forma de autenticar las cuentas de servicio en Google Cloud](#).



Google inhabilita automáticamente las claves de cuentas de servicio que se detecten en repositorios públicos. Puedes personalizar este comportamiento con la política de la organización "iam.serviceAccountKeyExposureResponse". [Más información](#)

Agrega un nuevo par de claves o sube un certificado de clave pública de un par de claves existente.

Impide la creación de claves de cuentas de servicio con las [políticas de la organización](#).

[Más información para configurar políticas de la organización en cuentas de servicio](#)

AGREGAR CLAVE ▾

Tipo	Estado	Clave	Fecha de creación	Fecha de vencimiento
	✓ Activa	a1b968de36cd58d9c1	4 nov 2024	31 dic 9999

Figura 109. Generación de clave en el servicio de cuenta.

Con los datos obtenidos se crea un archivo JSON con el formato presentado en la Figura 110, donde se debe cambiar la información correspondiente en los parámetros "private_key_id" y "client_id".

```
1  {
2    "type": "service_account",
3    "project_id": "name of your project",
4    "private_key_id": "value of private key id",
5    "private_key": "-----BEGIN PRIVATE KEY-----\n your key value \n-----END PRIVATE KEY-----\n",
6    "client_email": "emailname.gserviceaccount.com",
7    "client_id": "client id value",
8    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
9    "token_uri": "https://oauth2.googleapis.com/token",
10   "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
11   "client_x509_cert_url": "your client url",
12   "universe_domain": "googleapis.com"
13 }
```

Figura 110. Archivo JSON para utilizar servicios de Cloud Vision.

Con la extracción de texto de la Figura 106 mediante el comando `procesar_placa_y_crear_enlace(text1, text2, text3, result)` continua a una etapa de validación para que se cumpla con algunas características en el texto de placa como letras en mayúsculas, no mezclar números con letras y correcciones en caso de que

exista, eliminar caracteres inválidos y espacios, y corroborar que la longitud de caracteres se la adecuada, para lo que refiere a latitud y longitud en cambio se hace control del espaciado y caracteres especiales para la generación del enlace a Google maps con la ubicación de donde fue capturado el vehículo, la función se muestra en la Figura 111.

```
import re

def procesar_placa_y_crear_enlace(placa, latitud, longitud, result):

    # Eliminar caracteres inválidos y convertir a mayúsculas
    placa = re.sub(r'^[A-Za-z0-9]', '', placa).upper()
    # Validar que tiene al menos 3 letras y números combinados
    if len(placa) < 3:
        return "Inválida: la placa debe tener al menos 3 letras seguidas de números."

    # Diccionarios para conversión entre letras y números
    numeros_a_letras = {'0': '0', '1': 'I', '2': 'Z', '3': 'J', '4': 'A',
                        '5': 'S', '6': 'G', '7': 'T', '8': 'B', '9': 'P'}

    # Validar y ajustar los tres primeros caracteres como letras
    letras = placa[:3]
    # Corrige números mal ubicados
    letras = ''.join([numeros_a_letras.get(char, char) if char in numeros_a_letras
                      else char for char in letras])
    if not re.match(r"^[A-Z]{3}$", letras):
        return "Inválida: los primeros tres caracteres deben ser letras."

    # Diccionarios para conversión entre letras y números
    letras_a_numeros = {'0': '0', 'I': '1', 'Z': '2', 'E': '3', 'A': '4',
                        'S': '5', 'G': '6', 'T': '7', 'B': '8', 'P': '9',
                        'Q': '0', 'L': '1', 'J': '1', 'U': '0', 'D': '0'}

    # Validar y ajustar los caracteres restantes como números
    numeros = placa[3:]
    # Corrige letras mal ubicadas
    numeros = ''.join([letras_a_numeros.get(char, char) if char in letras_a_numeros
                      else char for char in numeros])
    # Ajustar los números a 4 dígitos
    numeros = numeros.zfill(4)
    # Formar la placa final
    placa_formateada = f"{letras}{numeros}"

    # Corregir formato de latitud
    latitud = re.sub(r'^[0-9.-]', '', latitud)
    if latitud.count('-') > 1 or (latitud.count('-') == 1 and latitud.index('-') != 0):
        # Mueve el signo negativo al principio si existe
        latitud = '-' + latitud.replace('-', '')
    latitud = float(latitud)

    # Corregir formato de longitud
    longitud = re.sub(r'^[0-9.-]', '', longitud)
    if longitud.count('-') > 1 or (longitud.count('-') == 1 and longitud.index('-') != 0):
        # Mueve el signo negativo al principio si existe
        longitud = '-' + longitud.replace('-', '')
    longitud = float(longitud)

    # Crear el enlace de Google Maps
    url = f"https://www.google.com/maps/search/?api=1&query={latitud},{longitud}"

    insertar_datos(str(placa_formateada), str(url), result)
```

Figura 111. Función de validación de placa y creación de enlace de ubicación.

Ahora con la extracción de texto de la Figura 107 mediante el comando `procesar_ubi_y_crear_enlace(text1, text2, result)`, se realiza un proceso de tratamiento de información para los datos de ubicación y generar el enlace de ubicación para abrirlo en Google maps, esta función se presenta en la Figura 112.

```
import re

def procesar_ubi_y_crear_enlace(latitud, longitud,result):

    # Corregir formato de latitud
    latitud = re.sub(r'^0-9.-', '', latitud)
    if latitud.count('-') > 1 or (latitud.count('-') == 1 and latitud.index('-') != 0):
        latitud = '-' + latitud.replace('-', '')

    latitud = float(latitud)

    # Corregir formato de longitud
    longitud = re.sub(r'^0-9.-', '', longitud)
    if longitud.count('-') > 1 or (longitud.count('-') == 1 and longitud.index('-') != 0):
        longitud = '-' + longitud.replace('-', '')

    longitud = float(longitud)

    # Crear el enlace de Google Maps
    url = f"https://www.google.com/maps/search/?api=1&query={latitud},{longitud}"

    insertar_datos( str(url),result)
```

Figura 112. Función para creación de enlace de ubicación.

Desde la Figura 111 a través del comando `insertar_datos(str(placa_formateada), str(url), result)` se conecta a la función de la Figura 113 para almacenamiento de la información en la base de datos con el texto de la placa extraída, URL de ubicación y URL de captura de imagen generado en cloudinary.

```

import mysql.connector

def insertar_datos(placa, ubicacion,result):

    # Configuración de la conexión a la base de datos
    config = {
        'user': 'user',
        'password': 'password',
        'host': 'host',
        'database': 'vehiculos',
        'port': 3306 # Puerto por defecto de MariaDB
    }

    try:
        # Establecer la conexión
        conn = mysql.connector.connect(**config)
        cursor = conn.cursor()

        # Consulta SQL para insertar los datos
        query = "INSERT INTO eventosReconocimiento (placa, ubicacion, foto) VALUES (%s, %s, %s)"
        values = (placa, ubicacion,result)

        # Ejecutar la consulta
        cursor.execute(query, values)

        # Confirmar la transacción
        conn.commit()

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        # Cerrar la conexión
        if conn.is_connected():
            cursor.close()
            conn.close()
            print("Conexión cerrada.")

```

Figura 113. Almacenamiento en base de datos cuando la placa es detectada.

Desde la Figura 112 a través del comando `insertar_datos(str(url),result)` se enlaza a la función de la Figura 114 para guardar la información en la base de datos de la URL de ubicación y URL de la captura generado en cloudinary.

```

import mysql.connector

def insertar_datos(ubicacion,result):

    # Configuración de la conexión a la base de datos
    config = {
        'user': 'user',
        'password': 'password',
        'host': 'host',
        'database': 'vehiculos',
        'port': 3306 # Puerto por defecto de MariaDB
    }

    try:
        # Establecer la conexión
        conn = mysql.connector.connect(**config)
        cursor = conn.cursor()

        # Consulta SQL para insertar los datos
        query = "INSERT INTO eventosSinReconocimiento (ubicacion,foto) VALUES (%s, %s)"
        values = (ubicacion,result)

        # Ejecutar la consulta
        cursor.execute(query, values)

        # Confirmar la transacción
        conn.commit()

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        # Cerrar la conexión
        if conn.is_connected():
            cursor.close()
            conn.close()
            print("Conexión cerrada.")

```

Figura 114. Almacenamiento en base de datos cuando la placa no es detectada.

Con los datos insertados en las tablas de la base de datos respectivamente según sea el caso, continúa el proceso donde se va a realizar comparaciones con la información almacenada y la nueva información que ingresa en las tablas, a continuación, se describe el proceso realizado en Azure.

3.1.15 Creación y configuración de la máquina virtual en Azure

Para iniciar el proceso de creación de una máquina virtual en Azure, es necesario crear una cuenta y acceder a la plataforma. A continuación, se debe navegar a la sección dedicada a la creación de máquinas virtuales. En este proyecto, se ha optado por utilizar Azure debido a la necesidad de contar con una base de datos alojada en la nube,

lo que permite un acceso remoto desde cualquier ubicación a través de internet, en la Figura 115 se muestra la creación de la máquina con sus respectivos parámetros.

The screenshot shows the 'Crear una máquina virtual' (Create a virtual machine) page in the Microsoft Azure portal. The page is in Spanish and displays the following configuration options:

- Suscripción ***: Azure for Students
- Grupo de recursos ***: Server_group
- Detalles de instancia**:
 - Nombre de máquina virtual ***: Server1
 - Región ***: (Asia Pacific) Korea Central
 - Opciones de disponibilidad**: Zona de disponibilidad
 - Opciones de zona**:
 - ☒ Zona autoseleccionada (Elija hasta 3 zonas de disponibilidad, una máquina virtual por zona)
 - ☐ Zona seleccionada por Azure (versión preliminar) (Permitir que Azure asigne la mejor zona para sus necesidades)
 - Zona de disponibilidad ***: Zona 1
- Tipo de seguridad**: Máquinas virtuales de inicio seguro
- Imagen ***: Ubuntu Server 20.04 LTS - x64 gen. 2
- Arquitectura de VM**:
 - ☐ Arm64
 - ☒ x64
- Ejecución de Azure Spot con descuento**: ☐

At the bottom, there are navigation buttons: '< Anterior', 'Siguiente: Discos >', and 'Revisar y crear'.

Figura 115. Configuración y creación de la máquina virtual en Azure.

Dado que el proyecto no requiere alto consumo de recursos ni se espera un alto volumen de datos, se seleccionó una máquina virtual con las características detalladas en la Tabla 38. Esta configuración resulta adecuada para garantizar un rendimiento eficiente sin sobrepasar los requerimientos necesarios.

Tabla 38. Características importantes de la máquina virtual en Azure.

Característica	Valor
Nombre de la Máquina Virtual	Server
Sistema operativo	Linux
Tamaño de la VM	Standard B2s
Memoria RAM	4 GB
Procesadores Virtuales (vCPU)	2
Disco del Sistema Operativo	30 GB SSD
Discos Adicionales	NA
Región	Korea Central (Zona 1)
Red virtual/subred	Server-vnet/default
Conexión SSH/RDP	Habilitada
Grupo de Recursos	Server_group

3.1.16 Creación y configuración de la Base de datos en Azure

Mediante el uso de la herramienta PuTTY y el protocolo SSH, iniciamos sesión en la Interfaz de Línea de Comandos (CLI) de la máquina virtual utilizando las credenciales asignadas. A continuación, se presenta un análisis detallado de los comandos aplicados, los recursos que se instalaron y los archivos de programación que fueron desarrollados para garantizar el funcionamiento eficiente de la base de datos, así como la correcta transmisión de datos hacia el correo electrónico y los grupos de Telegram.

a. Instalación de los componentes comunes para el desarrollo web

Comenzamos con la actualización del sistema para garantizar que todos los recursos necesarios se instalen adecuadamente. Posteriormente, procedemos a instalar los componentes comunes en el desarrollo web, agrupados bajo el término LAMP, En la Figura 116 se observa los comandos ingresados en el CLI. LAMP está compuesto por Apache, MariaDB/MySQL y PHP. Además, se instaló phpMyAdmin, una interfaz gráfica que facilita la administración de bases de datos, proporcionando una interacción visual intuitiva para la creación y organización de tablas dentro de las bases de datos.

```

# Actualiza la lista de paquetes
# disponibles en los repositorios.
sudo apt update
# Instala las actualizaciones disponibles
# para los paquetes ya instalados.
sudo apt upgrade
# Instala el servidor web Apache.
sudo apt install apache2
# Instala el servidor de bases de datos MariaDB.
sudo apt install mariadb-server
# Instala PHP, el lenguaje de programación para
# páginas web dinámicas.
sudo apt install php
# Instala phpMyAdmin, una interfaz web para
# gestionar bases de datos MariaDB/MySQL.
sudo apt install phpmyadmin

```

Figura 116. Instalación de componentes comunes para el desarrollo web mediante el (CLI).

b. Creación y configuración de la base de datos

La creación de la base de datos se realizó accediendo a phpMyAdmin desde el navegador. En la opción para crear una nueva base de datos, se le dio el nombre "vehículos", tal como se ilustra en la Figura 117. Es importante recordar que, al configurar la base de datos, se deben asignar las credenciales correspondientes para su correcta gestión y seguridad.

Bases de datos

 Crear base de datos 

Crear

Figura 117. Creación de la base de datos mediante phpMyAdmin.

Se procedió también a la creación de diversas tablas que facilitan la organización eficiente de la información relacionada con el sistema de vehículos robados. En la

Figura 118 se presentan las tablas que fueron configuradas para asegurar el adecuado funcionamiento del sistema.

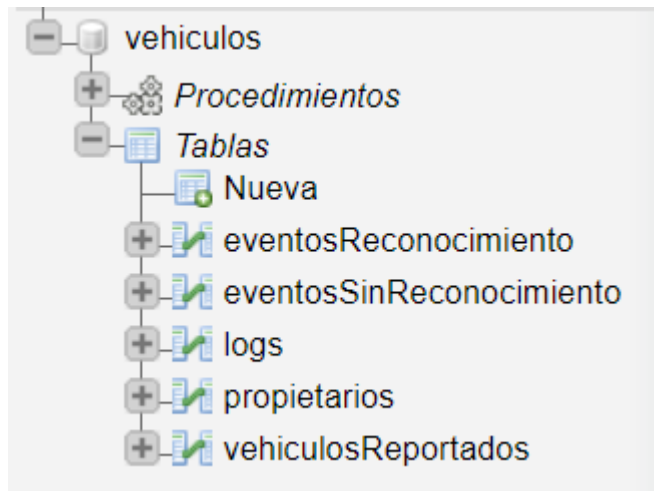


Figura 118. Creación de las tablas.

La tabla "eventosReconocimiento" tiene la función de almacenar la información de los vehículos cuya placa fue detectada correctamente. En esta tabla se registra la placa procesada, además de incluir los enlaces correspondientes a su ubicación y a la fotografía del vehículo alojada en Clouinary, en el Anexo J se muestra el registro de información de un evento de reconocimiento en la base de datos. Por otro lado, la tabla "eventosSinReconocimiento" almacena información de los vehículos cuya placa no fue detectada correctamente o que carecen de ella. En este caso, únicamente se guarda el enlace de ubicación y la imagen del vehículo.

En cuanto a la tabla "propietarios", su función es registrar los datos de las personas que han reportado un vehículo robado, almacenando información como la placa, el nombre, el teléfono y el correo electrónico del denunciante. A su vez, la tabla "vehiculosReportados" organiza la información de los vehículos reportados, especificando su placa, marca, modelo y año. Finalmente, la tabla "logs" guarda los datos necesarios que se enviarán automáticamente al grupo de Telegram y a los correos electrónicos de los usuarios correspondientes.

c. Comparación de placas usando Trigger

Para comprobar si un evento de reconocimiento corresponde a un vehículo previamente reportado, se implementó un Trigger. Este elemento se encarga de automatizar los procesos relacionados con las tablas de la base de datos. La Figura 119 presenta la estructura del Trigger diseñado para comparar las placas registradas en la tabla "eventosReconocimiento" con aquellas presentes en la tabla "vehiculosReportados".

Cuando se detecta una coincidencia entre las placas, el Trigger ejecuta automáticamente un proceso que transfiere la información pertinente, incluyendo la placa, el enlace de ubicación y la imagen del vehículo, hacia la tabla "logs", garantizando un flujo eficiente de los datos.

```
-- Cambia el delimitador de comandos SQL a "/" para permitir
-- la ejecución de múltiples instrucciones en el trigger.
DELIMITER //

-- Crea un trigger llamado "verificar_placa".
CREATE TRIGGER verificar_placa
-- Ejecuta el trigger después de insertar un nuevo
-- registro en la tabla "eventosReconocimiento".
AFTER INSERT ON eventosReconocimiento
-- Indica que el trigger se ejecutará para cada fila insertada.
FOR EACH ROW
BEGIN
-- Declara una variable llamada "propietarioId" de tipo entero
-- para almacenar el ID del propietario si hay coincidencia.
DECLARE propietarioId INT;

-- Compara la placa recién insertada con la tabla de vehículos reportados
SELECT id INTO propietarioId
FROM vehiculosReportados
-- Busca si la placa recién insertada existe en la tabla "vehiculosReportados".
WHERE placa = NEW.placa;

-- Si hay coincidencia
IF propietarioId IS NOT NULL THEN
-- Insertar un mensaje en la tabla de logs con la ubicación y la foto
INSERT INTO logs (mensaje, ubicacion, foto)
VALUES (NEW.placa, NEW.ubicacion, NEW.foto);
END IF;
END // -- Finaliza el bloque del trigger|.

DELIMITER ; -- Restaura el delimitador por defecto ";".
```

Figura 119. Configuración del Trigger en MariaDB.

d. Configuración de Telegram

Para enviar información hacia Telegram, se empleó BotFather, una herramienta que, mediante una serie de comandos, facilita la creación de un Bot personalizado. Este Bot utiliza un "TOKEN" asignado al momento de su creación y el "CHAT ID" del grupo de Telegram para recibir y procesar la información enviada desde la base de datos. En la Figura 120 se describen los pasos necesarios para llevar a cabo la creación del Bot utilizando BotFather en Telegram.

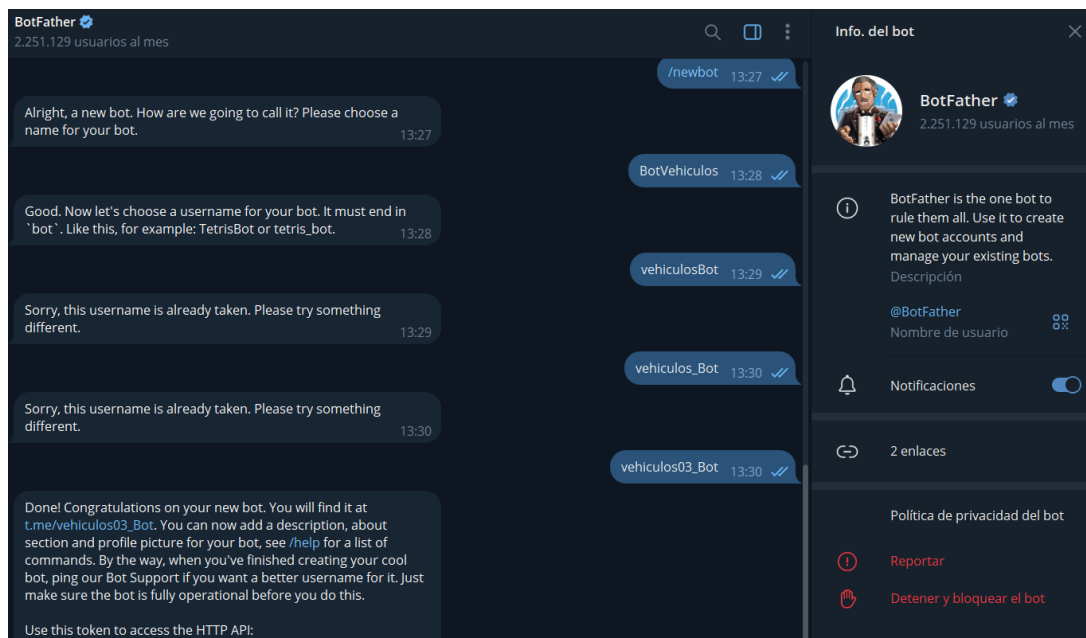


Figura 120. Creación de un Bot utilizando BotFather.

Para la gestión de la información en Telegram, se crearon dos grupos con propósitos diferenciados. El grupo "VEHÍCULOS REPORTADOS" está destinado a notificar a los usuarios que hayan registrado el reporte de su vehículo, informándoles cuando alguno de estos vehículos sea localizado en una ubicación específica.

El segundo grupo, denominado "VEHÍCULOS REPORTADOS ADMIN", está dirigido a administradores, como la policía u otras entidades, y se usará para compartir información sobre vehículos que sean sospechosos o carezcan de placas. La Figura 121 muestra la creación de estos dos grupos en Telegram.

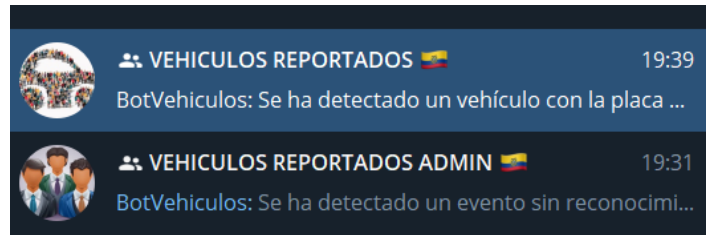


Figura 121. Grupos de Telegram para los usuarios y administradores.

e. Contraseña de aplicación del correo electrónico

SMTP es un protocolo que sirve para facilitar el envío de correos electrónicos desde un cliente, como una aplicación o script, hacia un servidor de correo, el cual posteriormente los entrega al destinatario final. Como primer paso, se ingresa al correo electrónico que funcionará como emisor hacia las cuentas de los usuarios correspondientes.

En el apartado de configuraciones del correo, debemos acceder a la opción de contraseñas de aplicación y generar una nueva, en caso de no contar con una previamente. Se configuró una contraseña de aplicación denominada "appvehiculos", con el código generado, como se muestra en la Figura 122. Finalmente, para que el sistema funcione correctamente, se requiere instalar el módulo smtpplib en la CLI de la máquina virtual.

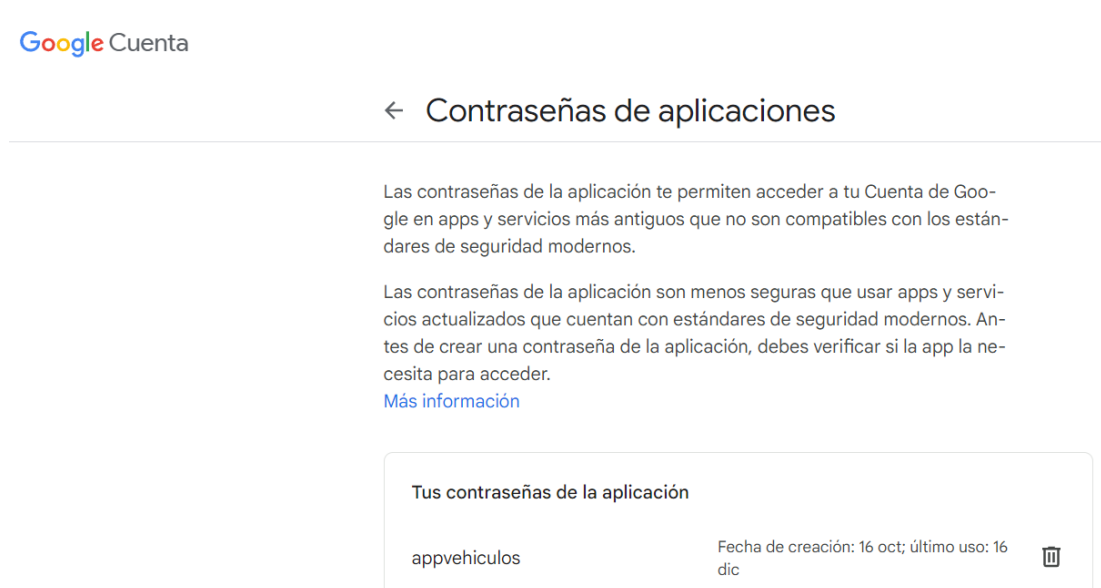


Figura 122. Creación de una contraseña de aplicación para el correo electrónico.

f. Codificación para el envío de información hacia Telegram y correo electrónico de los usuarios y administradores

Como parte del desarrollo, se empleó Python para implementar el envío de información. Se crearon dos scripts junto con dos archivos de texto que trabajan en conjunto con un servicio diseñado exclusivamente para ejecutar de manera ininterrumpida los archivos de Python, asegurando la disponibilidad constante del sistema. A continuación, se detalla cada una de las acciones implementadas para garantizar el correcto funcionamiento del sistema.

- ***Bibliotecas requeridas***

Las bibliotecas utilizadas para la codificación se detallan en la Tabla 39, presentada a continuación:

Tabla 39. Bibliotecas requeridas para envío de información a los usuarios.

Biblioteca	Funciones Utilizadas	Características
mariadb	mariadb.connect(), cursor.execute(), cursor.fetchall(), conexion.cursor(), cursor.close(), conexion.close()	Permite la interacción con bases de datos MariaDB. Proporciona métodos para conectarse a bases de datos, ejecutar consultas SQL, recuperar resultados y gestionar la conexión de manera eficiente.
asyncio	asyncio.run(), await bot.send_message()	Permite la programación asíncrona y concurrente en Python. Usado para ejecutar tareas de manera no bloqueante, lo que permite hacer múltiples tareas al mismo tiempo (como enviar mensajes de Telegram).
smtplib	smtplib.SMTP(), server.starttls(), server.login(), server.sendmail()	Permite enviar correos electrónicos usando el protocolo SMTP. Es una herramienta estándar de Python para interactuar con servidores de correo, autenticarse y enviar mensajes.
telegram	from telegram import Bot	Proporciona la interfaz para crear y gestionar bots en Telegram. Permite enviar mensajes a los usuarios a través de Telegram de forma programática.
email.mime.text	MIMEText()	Permite crear contenido de texto para correos electrónicos. Se utiliza para generar cuerpos de mensajes en formato MIME, que puede ser enviado a través de SMTP.
email.mime.multipart	MIMEMultipart()	Permite crear correos electrónicos con varias partes, como texto y archivos adjuntos. Se utiliza para la creación de mensajes más complejos, como los correos con adjuntos.

Biblioteca	Funciones Utilizadas	Características
OS	os.path.exists(), open(), os.write()	Permite interactuar con el sistema operativo. Usado para trabajar con archivos y directorios, verificar la existencia de archivos y manipular el sistema de archivos.

- ***Creación de archivos e instalación de Bibliotecas***

El proceso inició con la instalación del conector de MariaDB. Para ello, se utiliza el comando `wget` junto con el enlace oficial para descargar la versión "mariadb-connector-c-3.3.1". Posteriormente, se descomprime el archivo descargado mediante el comando `tar` y se accede a la carpeta resultante. Dentro del directorio, se creó un directorio llamado "build" para realizar la compilación.

En el directorio "build", se ejecuta el comando `make` para compilar el código fuente del conector. Finalmente, se verifica la instalación en el sistema utilizando `sudo make install`, que integra el conector compilado de manera correcta. Además, se procedió a crear un directorio denominado "scripts" al que se le configuraron los permisos correspondientes. Este directorio "scripts" se empleó para guardar los archivos de codificación y los documentos de texto requeridos para gestionar el envío eficiente de información, tanto a Telegram como hacia los correos electrónicos. En la Figura 123 se describen detalladamente los comandos y pasos realizados desde la interfaz de línea de comandos (CLI).

```

# Descarga el código fuente del conector C de MariaDB versión 3.3.1.
wget https://downloads.mariadb.com/MariaDB/mariadb-connector-c-3.3.1-src.tar.gz
# Extrae el contenido del archivo descargado.
tar -xvzf mariadb-connector-c-3.3.1-src.tar.gz
# Entra en el directorio del código fuente extraído.
cd mariadb-connector-c-3.3.1-src
# Crea un directorio "build" para la compilación.
mkdir build
# Entra en el directorio "build".
cd build/
# Compila el código fuente del conector.
make
# Instala el conector C de MariaDB en el sistema.
sudo make install
# Instala asyncio, una librería para manejo de operaciones asíncronas en Python.
pip install asyncio
# Instala la librería python-telegram-bot para interactuar con la API de Telegram.
pip install python-telegram-bot
# Crea un directorio llamado "scripts" para almacenar archivos y scripts.
mkdir scripts
# Concede permisos de lectura, escritura y ejecución a todos los usuarios en
# la carpeta "scripts".
chmod -R 777 scripts/
# Entra en la carpeta "scripts".
cd scripts
# Crea archivos vacíos para los scripts y archivos de almacenamiento de IDs.
touch envioATelegram.py envioATelegramAdmin.py ultima_id.txt ultima_id_f.txt

```

Figura 123. Creación de archivos e instalación de bibliotecas en el (CLI) de la máquina virtual.

- ***Desarrollo de la codificación***

En el archivo envioATelegram.py se realizó la codificación correspondiente para el grupo de Telegram y correo electrónico de los usuarios que hicieron el correspondiente reporte de su vehículo en el dado caso de ser extraviado. En el Anexo K se ilustra a detalle la codificación del archivo envioATelegram.py.

Se empieza incluyendo las bibliotecas necesarias para la programación como se muestra en la Figura 124.

```

# Importar el conector de MariaDB para interactuar con la base de datos
import mariadb
# Importar asyncio para manejar operaciones asíncronas
import asyncio
# Importar smtplib para el envío de correos electrónicos
import smtplib
# Importar la clase Bot de telegram para enviar mensajes
from telegram import Bot
# Importar clases para manejar correos electrónicos con formato MIME
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
# Importar os para manejar archivos del sistema
import os

```

Figura 124. Inclusión de las bibliotecas para el envío hacia los usuarios.

En la Figura 125 se detalla la configuración inicial realizada. Para la integración con Telegram, se configuro el TOKEN, el CHAT ID y la instancia del Bot que hará uso de la API de esta plataforma. En el caso del correo electrónico, se especificaron las credenciales necesarias del remitente y se ajustaron los valores para el servidor SMTP de Gmail junto con su puerto correspondiente.

```

# Configuración del bot de Telegram y la base de datos
TELEGRAM_TOKEN = 'TOKEN DEL BOT DE TELEGRAM'
CHAT_ID = 'ID DEL CHAT DE TELEGRAM'
bot = Bot(token=TELEGRAM_TOKEN)

# Configuración del servidor de correo
EMAIL_USER = 'USUARIO DE CORREO'
EMAIL_PASS = 'contraseña generada por la app'
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = # Define el puerto SMTP del servidor de correo

```

Figura 125. Configuración de las credenciales iniciales de Telegram y correo electrónico para los usuarios.

En la Figura 126 se muestran dos funciones diseñadas para trabajar con el archivo ultimo_id.txt. Dicho archivo cumple el propósito de registrar y consultar el último identificador asociado a una fila de la tabla logs en la base de datos, permitiendo gestionar de forma eficiente los datos y prevenir envíos redundantes de información.

```

# Definiendo el nombre donde se guarda la última ID procesada
ID_FILE = 'ultima_id.txt'

# Función para leer la última ID procesada desde un archivo
def leer_ultima_id():
    if os.path.exists(ID_FILE):
        with open(ID_FILE, 'r') as f:
            return int(f.read().strip())
    return 0

# Función para guardar la última ID procesada en un archivo
def guardar_ultima_id(id):
    with open(ID_FILE, 'w') as f:
        f.write(str(id))

```

Figura 126. Gestión del archivo de última ID procesada para los usuarios.

Se implementaron las funciones necesarias para el envío de mensajes. En la Figura 127 se detalla la función `enviar_mensaje`, que permite enviar mensajes al chat de Telegram de forma asíncrona a través del Bot configurado. Adicionalmente, la función `enviar_correo` tiene la tarea de enviar un correo electrónico al destinatario, incluyendo un mensaje de alerta como parte de la notificación.

```

# Función asíncrona para enviar un mensaje a Telegram
async def enviar_mensaje(mensaje):
    await bot.send_message(chat_id=CHAT_ID, text=mensaje)

# Función para enviar un correo electrónico
def enviar_correo(destinatario, mensaje):
    try:
        print(f"Enviando correo a {destinatario}...")
        msg = MIMEText(mensaje) # Crear un mensaje MIME
        msg['From'] = EMAIL_USER # Remitente
        msg['To'] = destinatario # Destinatario
        msg['Subject'] = 'Alerta de Vehículo Detectado' # Asunto del correo

        # Adjuntar el cuerpo del mensaje
        msg.attach(MIMEText(mensaje, 'plain'))

        # Establecer conexión con el servidor SMTP y enviar el correo
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls() # Iniciar la comunicación segura
            server.login(EMAIL_USER, EMAIL_PASS) # Autenticación
            server.sendmail(EMAIL_USER, destinatario, msg.as_string()) # Enviar correo
        print("Correo enviado.")
    except Exception as e:
        print(f"Error enviando el correo: {e}")

```

Figura 127. Funciones para el envío de mensaje hacia Telegram y correo electrónico.

Luego tenemos la función encargada de verificar nuevos eventos en la base de datos. El proceso comienza utilizando la función que lee el último ID almacenado en el archivo ultimo_id.txt. Posteriormente, se establece la conexión con la base de datos "vehiculos" mediante las credenciales correspondientes. Utilizando el comando cursor.execute, se realiza una consulta en la tabla logs para identificar registros cuyo ID sea mayor al último registrado. A continuación, mediante un bucle for, se procesan los registros obtenidos, extrayendo información como la placa detectada, la fecha, la ubicación y la posible ruta de una foto. Adicionalmente, otra consulta con cursor.execute verifica si la placa detectada tiene un propietario registrado en la tabla "propietarios". Finalmente, se emplea una estructura condicional if para enviar las notificaciones necesarias y actualizar el ID correspondiente.

La Figura 128 muestra cómo se lleva a cabo la ejecución del script. Este proceso marca el inicio de la verificación de los logs y garantiza el envío efectivo de las notificaciones derivadas de los eventos detectados en la base de datos.

```
# Punto de entrada del script
if __name__ == '__main__':
    print("Iniciando script...")
    # Ejecutar la función de verificación de logs
    asyncio.run(verificar_logs())
    print("Script finalizado.")
```

Figura 128. Ejecución del script envioATelegram.py.

Del mismo modo, se realiza la codificación en el archivo envioATelegramAdmin.py, en el Anexo L se muestra la codificación completa a detalle para enviar información al grupo de administradores, siguiendo una estructura similar a la utilizada para los usuarios. En este caso, se emplean las mismas bibliotecas para gestionar el envío de mensajes, con las configuraciones iniciales modificadas, ajustando el CHAT ID y asignando un correo electrónico fijo que pertenezca al administrador responsable, según lo presentado en la Figura 129.

```

# Configuración del bot de Telegram y la base de datos
TELEGRAM_TOKEN = 'TOKEN DEL BOT DE TELEGRAM'
CHAT_ID = 'ID DEL CHAT DE TELEGRAM'
bot = Bot(token=TELEGRAM_TOKEN)

# Configuración del servidor de correo
EMAIL_USER = 'USUARIO DE CORREO'
EMAIL_PASS = 'contraseña generada por la app'
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = # Define el puerto SMTP del servidor de correo
EMAIL_FIJO = 'Poner el correo fijo' # Correo fijo al que se enviarán las alertas

```

Figura 129. Configuración de las credenciales iniciales de Telegram y correo electrónico para los administradores.

Por otro lado, las funciones destinadas a la gestión del archivo que almacena el último ID presentan una estructura similar a las ya implementadas, con la única diferencia de que se modifica el nombre tanto de la función como del archivo, que pasa a llamarse ultimo_id_f.txt. En la Figura 130 se detallan los ajustes realizados.

```

# Definiendo el nombre donde se guarda la última ID procesada
ID_FILE_F = 'ultima_id_f.txt'

# Función para leer la última ID procesada
def leer_ultima_id_f():
    if os.path.exists(ID_FILE_F):
        with open(ID_FILE_F, 'r') as f:
            return int(f.read().strip())
    return 0

# Función para guardar la última ID procesada
def guardar_ultima_id_f(id):
    with open(ID_FILE_F, 'w') as f:
        f.write(str(id))

```

Figura 130. Gestión del archivo de última ID procesada para los administradores.

Las funciones programadas para enviar mensajes a Telegram y correo electrónico mantienen una estructura idéntica a la utilizada en el archivo envioTelegram.py. Sin embargo, en la función de verificación de eventos se introducen algunas modificaciones, como el cambio de nombre a verificar_eventos_sin_reconocimiento. Además, al utilizar el comando cursor.execute, se realiza una consulta específica en la tabla eventosSinReconocimiento para identificar el último ID registrado en dicha tabla. Por otro lado, a través de un bucle for, se procesan los registros obtenidos, extrayendo información como la fecha, la ubicación y el enlace de la foto capturada.

Finalmente, se emplea una estructura condicional `if` para gestionar el envío de notificaciones al grupo de Telegram y al correo electrónico configurado, así como para actualizar el ID correspondiente.

La Figura 131 presenta el procedimiento de ejecución del archivo `envioATelegramAdmin.py`. Este script está diseñado específicamente para gestionar el envío de información hacia el grupo de Telegram y la dirección de correo electrónico previamente configurados para los administradores, asegurando la entrega de los datos correspondientes.

```
# Punto de entrada del script
if __name__ == '__main__':
    print("Iniciando script para eventos sin reconocimiento...")
    # Ejecuta la función principal
    asyncio.run(verificar_eventos_sin_reconocimiento())
    print("Script finalizado.") # Mensaje de finalización
```

Figura 131. Ejecución del script `envioATelegramAdmin.py`.

- ***Configuración del servicio para la ejecución automática de los scripts***

Para garantizar la ejecución continua de los archivos de Python en segundo plano, se configuró un servicio llamado `envio_telegram.service`. Este servicio permite realizar consultas constantes y enviar la información requerida tanto a usuarios como a administradores. El archivo de configuración del servicio se creó en el directorio `/etc/systemd/system/` bajo el nombre `envio_telegram.service`. En la Figura 132 se detalla el código completo de esta configuración.

El encabezado del servicio incluye el bloque `[Unit]`, que define aspectos básicos como la descripción del servicio y la directiva `After=network.target`, indicando que el servicio se iniciará únicamente después de que la red esté disponible.

En la configuración del servicio se incluyó el bloque `[Service]`, donde se especifica el comando que inicia el servicio mediante la directiva `ExecStart`. Este comando utiliza `/bin/bash -c` para ejecutar las instrucciones en una instancia de Bash, y el ejecutable de Python, ubicado en `/usr/bin/python3`, para ejecutar en segundo plano los scripts `envioATelegram.py` y `envioATelegramAdmin.py`. La directiva `WorkingDirectory` establece el directorio donde se ejecutarán estos scripts. Asimismo, se configuró una

política de reinicio automático en caso de fallos, definida por Restart y RestartSec, que indican el comportamiento del reinicio y el tiempo de espera antes de reintentar.

Por otra parte, se definieron los permisos de ejecución del servicio mediante las directivas User y Group, asignando el usuario y grupo correspondientes. Adicionalmente, se utilizaron StandardOutput y StandardError para determinar las ubicaciones donde se registrarán tanto la salida estándar como los errores del servicio, garantizando un seguimiento adecuado del funcionamiento.

Por último, se incluyó el bloque [Install], que establece las condiciones en las cuales el servicio será habilitado. Mediante la directiva WantedBy=multi-user.target, se indica que el servicio se iniciará automáticamente al alcanzar el modo multiusuario. Este nivel operativo es habitual en servidores sin entorno gráfico, asegurando que el servicio esté disponible de manera inmediata al reiniciar o encender el sistema.

```
# Archivo de servicio systemd para ejecutar los scripts de envío de mensajes
[Unit]
# Descripción del servicio
Description=Script para enviar mensajes a Telegram y correo electrónico
# Asegura que el servicio se inicie después de que la red esté disponible
After=network.target

[Service]
# Comando que ejecuta los scripts en segundo plano
ExecStart=/bin/bash -c "
    /usr/bin/python3 /root/mariadb-connector-c-3.3.1-src/build/scripts/envioATelegram.py &
    /usr/bin/python3 /root/mariadb-connector-c-3.3.1-src/build/scripts/envioATelegramAdmin.py &
    "
# Directorio de trabajo del servicio
WorkingDirectory=/root/mariadb-connector-c-3.3.1-src/build/scripts
# Reinicia el servicio automáticamente en caso de fallo
Restart=always
# Tiempo de espera antes de reiniciar el servicio tras un fallo
RestartSec=10
# Usuario que ejecuta el servicio
User=root
# Grupo que ejecuta el servicio
Group=root
# Redirige la salida estándar a un archivo de log
StandardOutput=file:/var/log/envio_telegram.log
# Redirige la salida de errores a un archivo de log
StandardError=file:/var/log/envio_telegram_error.log

[Install]
# Asegura que el servicio se inicie en el arranque del sistema
WantedBy=multi-user.target
```

Figura 132. Configuración del servicio para la ejecución automática de los scripts.

Tras la implementación y activación del servicio, el sistema de notificaciones al grupo de Telegram y al correo electrónico personal quedó operando de manera efectiva. En el Anexo J se evidenció la llegada de un mensaje al grupo de Telegram de los usuarios y una alerta en el correo del dueño del vehículo, validando la funcionalidad del sistema.

3.1.17 Creación y configuración de la página web en Azure

El desarrollo de la página web se llevó a cabo utilizando PHP junto con HTML, JavaScript y CSS. Se optó por PHP debido a que es un lenguaje de scripting del lado del servidor, lo que lo convierte en una herramienta adecuada para la creación de aplicaciones web dinámicas. En la Figura 133 se presenta el diagrama de flujo que ilustra el funcionamiento de la página.

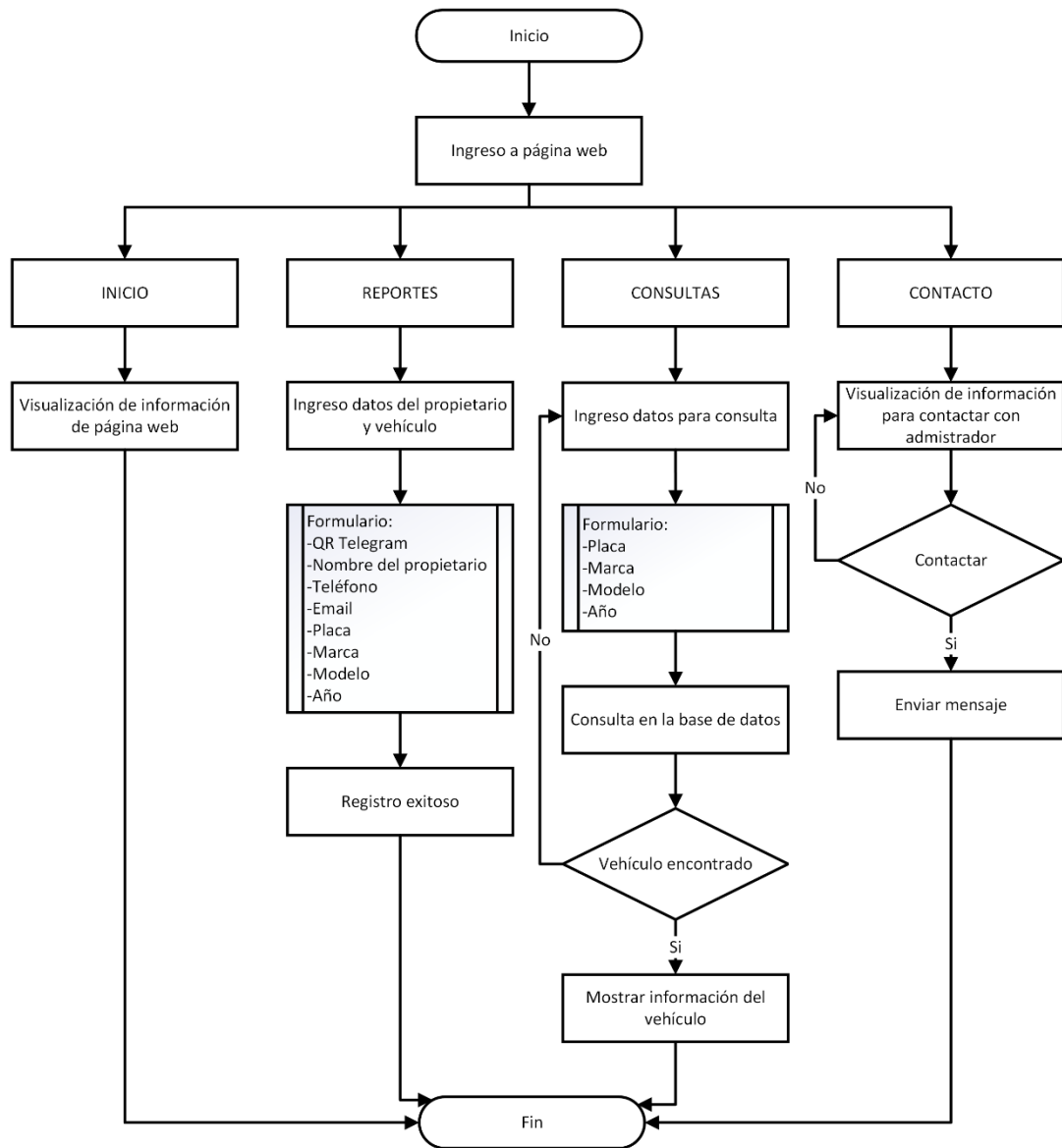


Figura 133. Diagrama de flujo del funcionamiento de la página web.

En las siguientes secciones se describe a detalle los procesos de creación e implementación realizada para garantizar el óptimo rendimiento del sitio web.

a. Creación de los archivos para la página web

Hay que dirigirse a /var/www/html/, donde se establece una carpeta denominada "paginaWeb". Dentro del directorio, se creó un subdirectorío llamado "img", destinado a almacenar las imágenes que se utilizarán en el sitio web. También se generaron los archivos pagina.php, guardar_datos_registro.php, datos_consulta.php y styles.css, diseñados para garantizar el funcionamiento y la presentación adecuada de la página

web. En la Figura 134 se detalla el procedimiento completo, incluyendo los comandos utilizados para crear los archivos y asignarles los permisos correspondientes.

```
# Cambia al directorio /var/www/html/  
cd /var/www/html/  
# Crea un nuevo directorio llamado 'paginaWeb'  
mkdir paginaWeb  
# Da permisos de lectura, escritura y ejecución a todos los  
# usuarios en el directorio 'paginaWeb'  
sudo chmod -R 777 paginaWeb  
# Entra al directorio 'paginaWeb'  
cd paginaWeb  
# Crea un directorio 'img' dentro de 'paginaWeb'  
mkdir img  
# Da permisos de lectura, escritura y ejecución a todos los  
# usuarios en el directorio 'img'  
sudo chmod -R 777 img  
# Crea los archivos para el funcionamiento de la pagina web  
touch datos_consulta.php datos_registro.php pagina.php styles.css
```

Figura 134. Creación de los archivos y otorgación de permisos para el desarrollo de la página web.

b. Codificación del archivo de la página principal

La programación de la página principal se lleva a cabo en el archivo pagina.php, cuyo contenido completo se detalla en el Anexo M. Este archivo está organizado en un formato basado en HTML e incluye fragmentos de código en JavaScript y PHP, diseñados para gestionar de manera eficiente la lógica y el funcionamiento del sitio web.

La codificación de la sección head incluye elementos esenciales como la definición del conjunto de caracteres, la configuración de la vista adaptable y el título de la página. El diseño visual se gestiona mediante el archivo styles.css y una hoja de estilos específica para los íconos. En cuanto a las funcionalidades, se utilizó JavaScript, incluyendo la creación de la función showSection(sectionId), que tiene la tarea de ocultar todas las secciones, mostrar únicamente la sección especificada y reiniciar los formularios. Por otro lado, la función resetForms() se encarga de restablecer los formularios y limpiar cualquier mensaje de éxito. Finalmente, el evento DOMContentLoaded asegura que, al cargar la página, se visualice la sección correspondiente según los parámetros del URL.

La sección head o cuerpo destaca por ser la más extensa dentro del código, ya que en ella se define gran parte de la estructura fundamental de la página web. Esta área incluye no solo la organización de los elementos principales, sino también la integración de los botones de navegación, el contenido informativo que se presentará al usuario y las imágenes representativas que complementan el diseño.

En la etiqueta header establecemos el logotipo personalizado que identifica la página web, junto con la incorporación de cuatro botones de navegación denominados Inicio, Reportes, Consultas y Contactos. Estos botones permiten acceder de manera intuitiva a las distintas secciones del sitio web. En la Figura 135 se detalla el código implementado para desarrollo de la sección.

```
<!-- Cuerpo de la página con una clase inicial definida -->
<body class="inicio-contactos">
  <!-- Cabecera con el logo y la navegación principal -->
  <header>
    <div class="logo">
      <!-- Imagen del logo -->
      
    </div>
    <!-- Menú de navegación -->
    <nav>
      <ul>
        <!-- Enlaces de navegación que muestran diferentes secciones -->
        <li><a href="javascript:void(0)" onclick="showSection('inicio'); d
        <li><a href="javascript:void(0)" onclick="showSection('reportes');
        <li><a href="javascript:void(0)" onclick="showSection('consultas')
        <li><a href="javascript:void(0)" onclick="showSection('contactos')
      </ul>
    </nav>
  </header>
```

Figura 135. Codificación de la etiqueta header del archivo de la página principal.

La etiqueta main se utilizó para estructurar y programar los apartados que se muestran al seleccionar las diversas secciones a través de los botones de navegación. En la sección Inicio se definió un título, un texto descriptivo y una imagen representativa del sitio web.

En la sección Reportes se agregó un título, acompañado de un código QR que permite a los usuarios unirse al grupo de Telegram. También se incluyó un formulario para reportar vehículos robados, con campos como nombre, teléfono, correo electrónico, placa, marca, modelo y año, necesarios para registrar la información en la base de datos

y permitir el monitoreo a través del sistema desarrollado. Como complemento, se incluyó un botón de registro y un mensaje indicador para confirmar el éxito del proceso o notificar errores. Como complemento, se incluyó un botón de registro y un mensaje indicador para confirmar el éxito del proceso o notificar errores.

La sección Consultas cuenta con un título acompañado de un formulario diseñado para completar los datos principales de un vehículo, como placa, marca, modelo y año. Al ingresar esta información y presionar el botón de consulta, el sistema permitirá verificar si el vehículo reportado se ha detectado en algún lugar específico.

En la última sección llamada Contactos se incluyó un título junto con información clave, como el número telefónico, el correo electrónico y la dirección del personal administrativo. Esta información fue proporcionada para facilitar cualquier consulta vinculada al reporte de vehículos robados. La codificación de las distintas secciones de la página web se desarrolló dentro de la etiqueta main, utilizando una variedad de etiquetas para estructurar e incorporar el contenido textual.

La codificación del archivo pagina.php concluyó con la implementación de la etiqueta footer, que funciona como pie de página. En este apartado, se incluyeron los íconos correspondientes a las redes sociales del personal administrativo. En la Figura 136 se puede observar la codificación desarrollada para esta parte de la página web.

```
<!-- Pie de página con redes sociales -->
<footer>
  <h3>REDES SOCIALES</h3>
  <div class="social-icons">
    <!-- Enlaces a redes sociales -->
    <a href="#"><i class="fab fa-facebook-f"></i></a>
    <a href="#"><i class="fab fa-twitter"></i></a>
    <a href="#"><i class="fab fa-whatsapp"></i></a>
    <a href="#"><i class="fab fa-instagram"></i></a>
  </div>
</footer>
</body>
</html>
```

Figura 136. Codificación de la etiqueta footer del archivo de la página principal.

c. Codificación del archivo de registro

El archivo `guardar_datos_registro.php` tiene como objetivo principal gestionar el registro de vehículos robados en la base de datos, cuya codificación completa se presenta en el Anexo N. Este proceso se lleva a cabo mediante un formulario, en el que se verifica que los datos ingresados por el usuario cumplan con los formatos establecidos. Además, asegura una experiencia estética fluida, haciendo que el registro se perciba como una acción integrada en la misma página principal, sin necesidad de redirecciones.

Se comenzó por establecer la conexión con la base de datos, un paso clave que se ilustra en la Figura 137.

```
<?php
// Datos de conexión a la base de datos
$servername = "ip";
$username = "usuario";
$password = "contraseña";
$dbname = "nombre de la BD";

// Crear conexión a la base de datos
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar si la conexión es exitosa
if ($conn->connect_error) {
    // Si hay un error de conexión, muestra un mensaje y detiene la ejecución
    die("Conexión fallida: " . $conn->connect_error);
}
```

Figura 137. Establecimiento de conexión con la base de datos del archivo de registro.

A continuación, se verifica si el formulario se envió de manera correcta utilizando el método POST. Se capturan los valores que fueron enviados desde el formulario completado en la sección de Registros y se realiza el proceso de validación de cada campo. La Figura 138 muestra el código correspondiente a este proceso.

```

$success = false; // Variable para determinar si la inserción fue exitosa
$message = ""; // Variable para almacenar el mensaje que se mostrará al usuario

// Verificar si se ha enviado un formulario por POST
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Obtener los valores del formulario que el usuario ha enviado
    $nombre = $_POST['nombre'];
    $telefono = $_POST['telefono'];
    $email = $_POST['email'];
    $placa = $_POST['placa'];
    $marca = $_POST['marca'];
    $modelo = $_POST['modelo'];
    $anio = $_POST['anio'];

    // Validación de los datos del formulario
    if (!preg_match("/^[A-Z\s]+$/", $nombre)) {
        // Validación del nombre: solo letras mayúsculas
        $message = "Nombre del propietario inválido. Solo se permiten letras mayúsculas.";
    } elseif (!preg_match("/^[0-9]{10}$/", $telefono)) {
        // Validación del teléfono: debe tener 10 dígitos
        $message = "Teléfono inválido. Deben ser 10 dígitos.";
    } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL) || !preg_match("/^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$/", $email)) {
        // Validación del email: formato válido
        $message = "Email inválido.";
    } elseif (!preg_match("/^[A-Z]{3}[0-9]{4}$/", $placa)) {
        // Validación de la placa: formato de 3 letras y 4 números
        $message = "Placa del vehículo inválida.";
    } elseif (!preg_match("/^[A-Z]+$/", $marca)) {
        // Validación de la marca: solo letras mayúsculas
        $message = "Marca del vehículo inválida.";
    } elseif (!preg_match("/^[A-Z\s]+$/", $modelo)) {
        // Validación del modelo: solo letras mayúsculas
        $message = "Modelo del vehículo inválido.";
    } elseif ($anio < 1886 || $anio > 2024) {
        // Validación del año: debe estar entre 1886 y el año actual
        $message = "Año del vehículo inválido.";
    }
}

```

Figura 138. Estructura de control para solicitudes POST, captura de valores y verificación.

Se procede con el inicio del proceso de transacción, cuya codificación detallada se encuentra en la Figura 139. El comando `$conn->begin_transaction();` es utilizado para garantizar que las operaciones sean atómicas; es decir, si una inserción no se ejecuta correctamente, ninguna de las otras se guarda. Los comandos SQL permiten insertar datos en las tablas propietarios y vehiculosReportados. Para confirmar que la transacción se complete correctamente, se utiliza `$conn->commit();` y en caso de detectar errores, se ejecuta `$conn->rollback();`. Finalmente, se agregan mensajes de notificación para informar al usuario sobre el estado del registro.

```

    } else {
        // Si todas las validaciones pasan, iniciar la transacción en la base de datos
        $conn->begin_transaction();

        try {
            // Insertar los datos del propietario en la tabla 'propietarios'
            $sql_propietarios = "INSERT INTO propietarios (placa, propietario, telefono, email) VALUES (" . $placa . ", " . $propietario . ", " . $telefono . ", " . $email . ")";
            if ($conn->query($sql_propietarios) !== TRUE) {
                // Si la inserción falla, lanza una excepción y la transacción se revierte
                throw new Exception("Error: " . $sql_propietarios . "<br>" . $conn->error);
            }

            // Insertar los datos del vehículo en la tabla 'vehiculosReportados'
            $sql_vehiculos = "INSERT INTO vehiculosReportados (placa, marca, modelo, año) VALUES (" . $placa . ", " . $marca . ", " . $modelo . ", " . $año . ")";
            if ($conn->query($sql_vehiculos) !== TRUE) {
                // Si la inserción falla, lanza una excepción y la transacción se revierte
                throw new Exception("Error: " . $sql_vehiculos . "<br>" . $conn->error);
            }

            // Si no hubo errores, confirmar la transacción
            $conn->commit();
            $message = "Nuevo registro creado exitosamente.";
            $success = true;
        } catch (Exception $e) {
            // Si ocurre un error, revertir los cambios realizados en la base de datos
            $conn->rollback();
            $message = "No se ha realizado el registro, intentelo nuevamente.";
        }
    }

    // Cerrar la conexión a la base de datos
    $conn->close();
}
?>

```

Figura 139. Proceso de transacción de información en la base de datos.

Con el objetivo de mantener la coherencia en el diseño, se ha creado una estructura HTML similar a la del archivo pagina.php, específicamente para la sección de Reportes. En este caso, el formulario se omite, pero se conserva el encabezado, los botones de navegación para dirigir al archivo pagina.php, así como el pie de página. En el cuerpo de la página, únicamente se muestra el título asignado, un código QR para unirse al grupo de Telegram de los usuarios y un mensaje que informa al usuario si el registro se ha realizado correctamente o no.

d. Codificación del archivo de consulta

El propósito principal del archivo datos_consulta.php es consultar la base de datos en relación con un vehículo que haya sido reportado, tal como se ilustra en el Anexo O. En caso de que el vehículo se haya localizado en algún lugar, esa información se almacenará en la base de datos. Al realizar la consulta, la página web mostrará un

mensaje con los detalles del vehículo, incluyendo la placa, la fecha de detección, el enlace a la ubicación donde fue encontrado y el URL de la foto del vehículo.

De manera similar al archivo de registro, se inició el proceso conectando el sistema con la base de datos, un paso fundamental que se detalla en la Figura 140.

```
<?php
// Datos de conexión a la base de datos
$servername = "ip";
$username = "usuario";
$password = "contraseña";
$dbname = "nombre de la BD";

// Crear conexión a la base de datos
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar si la conexión fue exitosa
if ($conn->connect_error) {
    // Si hay un error de conexión, muestra un mensaje y detiene la ejecución
    die("Conexión fallida: " . $conn->connect_error);
}
```

Figura 140. Establecimiento de conexión con la base de datos del archivo de consulta.

Se llevó a cabo la verificación del envío correcto del formulario utilizando el método POST, capturando los valores enviados desde el formulario completado en la sección de Consultas. Para garantizar la seguridad, se empleó la función `real_escape_string` con el fin de limpiar los datos recibidos. El código detallado de este proceso se presenta en la Figura 141.

```
// Verificar si se ha enviado un formulario por POST
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Obtener y escapar los datos enviados desde el formulario para evitar inyecciones SQL
    $placa = $conn->real_escape_string($_POST['placa-consulta']);
    $marca = $conn->real_escape_string($_POST['marca-consulta']);
    $modelo = $conn->real_escape_string($_POST['modelo-consulta']);
    $anio = $conn->real_escape_string($_POST['anio-consulta']);
}
```

Figura 141. Estructura de control para solicitudes POST y captura de valores.

Para realizar la búsqueda de un vehículo, se elaboró una consulta SQL que utiliza los datos ingresados en la tabla `vehiculosReportados` basándose en los datos ingresados. Para ejecutar la consulta, se utilizó el comando `$conn->query`. A continuación, se verificó si la consulta encontró coincidencias en los campos de placa, marca, modelo y año del vehículo. En caso afirmativo, se realizó una segunda búsqueda en la tabla `logs`, utilizando la placa como criterio, para determinar si ya existía algún registro

relacionado. Si se encontraban coincidencias, se concatena información como el mensaje, la fecha, la ubicación y la foto al texto, de lo contrario, se actualiza el mensaje con un error. El código detallado de este proceso se muestra en la Figura 142.

```
// Consulta a la tabla 'vehiculosReportados' buscando coincidencias exactas
// con los datos proporcionados
$sql_vehiculos = "SELECT * FROM vehiculosReportados WHERE placa = '$placa' AND";
$result_vehiculos = $conn->query($sql_vehiculos);

// Si se encuentra el vehículo en la tabla 'vehiculosReportados'
if ($result_vehiculos->num_rows > 0) {
    // Consulta a la tabla 'logs' buscando registros que contengan
    // la placa del vehículo
    $sql_logs = "SELECT * FROM logs WHERE mensaje LIKE '%$placa%'";
    $result_logs = $conn->query($sql_logs);

    // Si se encuentran registros en 'logs' relacionados con la placa
    // del vehículo
    if ($result_logs->num_rows > 0) {
        // Iterar sobre los registros encontrados en 'logs' y mostrar
        // la información
        while($row_logs = $result_logs->fetch_assoc()) {
            $message .= "Vehículo encontrado con placa: " . $row_logs["mensaje"] . "  
";
            $message .= "Fecha: " . $row_logs["fecha"] . "<br>";
            $message .= "Ubicación: <a href='" . $row_logs["ubicacion"] . "'>";
            $message .= "<img src='" . $row_logs["foto"] . "' alt='Foto del veh";
        }
    } else {
        // Si no se encuentran registros en 'logs', se muestra
        // un mensaje indicando que no se encontraron coincidencias
        $message = "No se ha encontrado el vehículo con los datos ingresados.";
    }
} else {
    // Si no se encuentra el vehículo en 'vehiculosReportados', se
    // muestra un mensaje indicando que no se encontró
    $message = "No se ha encontrado el vehículo con los datos ingresados."; //
}

// Cerrar la conexión a la base de datos
$conn->close();
```

Figura 142. Proceso de consulta y verificación de existencia del vehículo.

Para concluir, se diseñó una estructura HTML similar a la implementada en el archivo pagina.php, específicamente adaptada para la sección de Consultas. Aunque el formulario fue omitido, permanecen componentes esenciales como el encabezado, los botones de navegación hacia el archivo pagina.php y el pie de página. En el cuerpo del diseño, en caso de encontrar el vehículo solicitado, se muestra información clave, incluyendo la placa, la fecha registrada, la ubicación exacta y una imagen del vehículo.

- **Codificación del archivo de estilos**

El diseño estético del sistema web fue desarrollado a través del archivo `styleless.css`, cuyo propósito principal fue optimizar y estructurar la codificación de manera más ordenada. Los estilos fueron aplicados directamente a etiquetas, clases asignadas e identificadores específicos (IDs). En el Anexo P se presenta el código completo del archivo de estilos, donde se incluyen configuraciones aplicadas a la etiqueta `body`, a la clase `inicio-contactos`, entre otras.

- **Interfaz de usuario**

Tras completar la creación y codificación de los archivos, es posible acceder a la página web mediante un navegador, donde se despliega la interfaz principal, como se ilustra en la Figura 143. El sitio está dividido en cuatro apartados principales: Inicio, Reportes, Consultas y Contactos, accesibles mediante un menú de navegación. En la sección de Inicio se encuentra la página principal, que ofrece una introducción resumida sobre el contenido y propósito de la plataforma.



Figura 143. Página web sección Inicio.

En la sección de reportes, representada en la Figura 144, los usuarios tienen la opción de unirse al grupo de Telegram mediante un código QR establecido. Este grupo permite recibir notificaciones al grupo, además de enviar alertas al correo electrónico en caso de detectar un vehículo robado. Para realizar un reporte, es necesario completar un formulario que incluye datos del propietario, como nombre, teléfono y correo electrónico, así como datos del vehículo, incluyendo placa, marca, modelo y año. Si

los datos son ingresados correctamente, el sistema mostrará un mensaje confirmando el registro exitoso; de lo contrario, se generará un aviso de error en el procedimiento.



Figura 144. Página web sección Reportes.

En la Figura 145 se tiene la sección de consultas, en este apartado se podrá consultar si tu vehículo fue detectado en algún lugar, claro esta es otra forma de saber dicha información si no tiene acceso al grupo de Telegram o correo electrónico. La consulta de un vehículo robado se deberá llenar el formulario que tiene campos del vehículo como placa, marca, modelo y año. Una vez llenado el formulario se realiza una consulta, en el caso de haber ingresado los datos correctamente se mostrará la información del vehículo como la placa, la fecha, el enlace de la ubicación donde se ha detectado y la foto del vehículo. Y si no se ha encontrado se mostrará un mensaje informando que no se ha detectado ese vehículo aún.

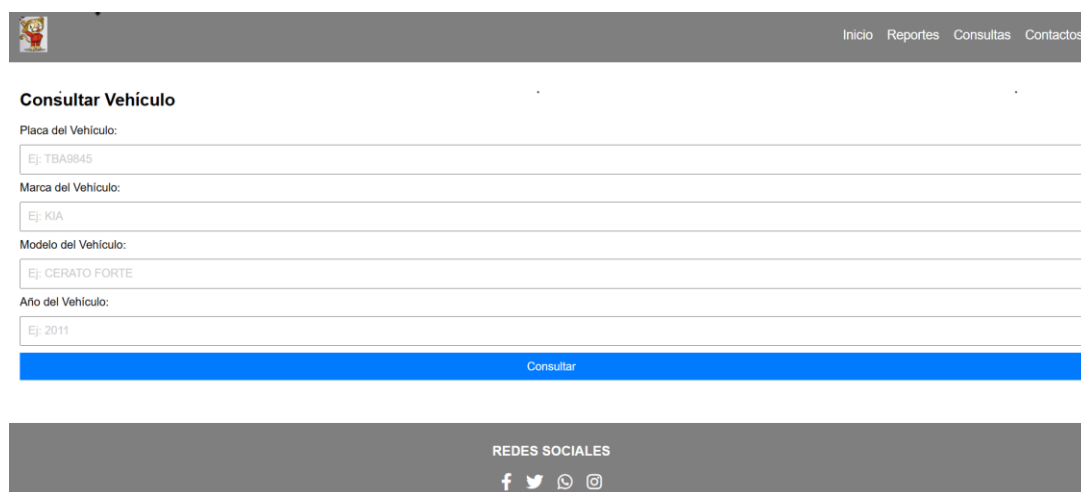


Figura 145. Página web sección Consultas.

En el apartado de contactos se presenta información clave sobre el personal administrativo, diseñada para que los usuarios puedan gestionar consultas de forma directa cuando lo requieran. En la Figura 146 se observa esta sección, que se destaca por ofrecer una solución práctica y accesible para resolver dudas o realizar gestiones específicas dentro del sistema.

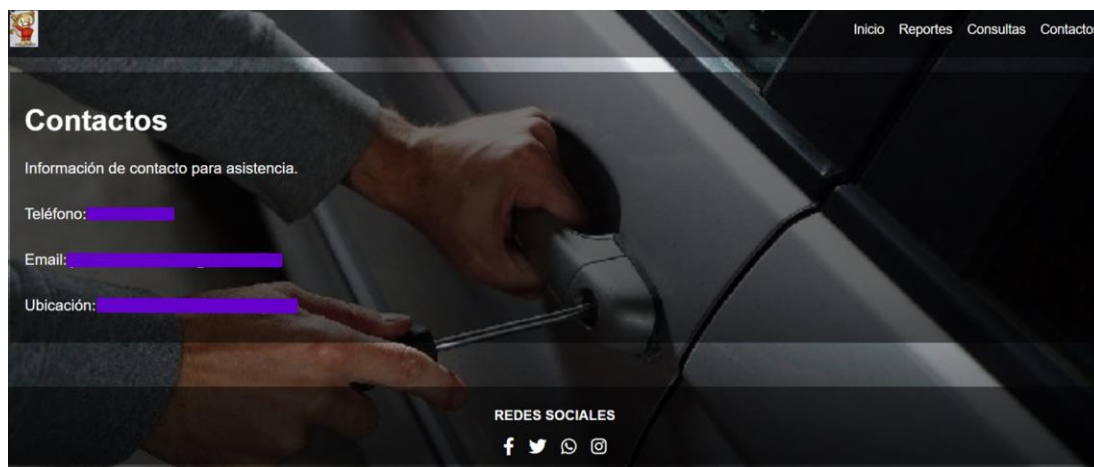


Figura 146. Página web sección Contactos.

3.1.18 Funcionamiento del registro y consulta en la página web

Al acceder a la página web, se indicó que, en caso de ser necesario realizar un registro, es preciso dirigirse a la sección denominada "reportes". En esta sección, se completa el formulario con la información requerida del propietario y del vehículo, como se ilustra en la Figura 147.

Nombre del Propietario:

VIVIANA

Teléfono:

0983896753

Email:

viviana23@gmail.com

Placa del Vehículo:

PBY6305

Marca del Vehículo:

HYUNDAI

Modelo del Vehículo:

I10 5P 12 AC

Año del Vehículo:

2012

Registrar

Figura 147. Formulario de registro.

Cuando los datos ingresados son correctos, el sistema mostrará un mensaje de confirmación indicando que el registro ha sido creado exitosamente, tal como se evidencia en la Figura 148. Esto permite al usuario verificar que la información ha sido registrada de manera adecuada. En caso de que se despliegue un mensaje distinto, se sugiere regresar a la sección de reportes y volver a ingresar los datos.



Figura 148. Registro exitoso.

Tras completar el registro, los datos ingresados serán almacenados en la base de datos de manera estructurada. La información correspondiente al usuario se guarda en una

tabla denominada "propietarios", mientras que los datos relacionados con el vehículo en la tabla "vehiculosReportados", como se puede observar en la Figura 149.

	id	placa	propietario	telefono	email
☐ Editar ☐ Copiar ☐ Borrar	66	PBY6305	VIVIANA	0983896853	viviana23@gmail.com

a)

	id	fecha	placa	marca	modelo	año
☐ Editar ☐ Copiar ☐ Borrar	65	2025-01-18 18:23:22	PBY6305	KIA	SPORTAGE GT	2012

b)

Figura 149. Registro en la base de datos. a) Datos propietario; b) Datos vehículo.

En caso de necesitar realizar una consulta, se debe acceder a la sección denominada "consultas" y completar el formulario correspondiente con los datos específicos del vehículo. Este procedimiento, ilustrado en la Figura 150, permite que el sistema procese la información de manera eficiente para obtener los resultados requeridos.

Consultar Vehículo

Placa del Vehículo:

TBF8484

Marca del Vehículo:

CHEVROLET

Modelo del Vehículo:

SAIL STD 1.4 4P 4X2 TM

Año del Vehículo:

2017

Consultar

Figura 150. Formulario de consulta.

Cuando los datos se ingresan de manera correcta y siguiendo el formato requerido, se procede a realizar la consulta correspondiente. En caso de que el vehículo ha sido localizado, el sistema desplegará en la página web información relevante, como la

placa, la ubicación y una fotografía del vehículo, tal como se aprecia en la Figura 151. Si el vehículo no ha sido encontrado, se mostrará un mensaje indicando que no se ha obtenido resultados.



Consultar Vehículo



Figura 151. Información proporcionada si el vehículo ha sido localizado previamente.

3.1.19 Planteamiento de espacio para desarrollo de pruebas

Para el planteamiento de resultados se consideró mantener una distancia horizontal constante planteando mantener una visión del vehículo adecuada en la Tabla 40 se visualiza distancias de referencias para captura de video desde el UAV, siendo lo óptimo mantener una distancia horizontal promedio de 2.0 m.

Tabla 40. Distancias horizontales para captura de video.

Distancia Horizontal [m]	1	2	3
Imagen Referencia			

La velocidad del viento promedio en Ambato es de 5 Km/h, para este estudio se ha fijado el parámetro para calcular el tiempo de vuelo en base a pruebas continuas. Para determinar el tiempo de vuelo en los parqueaderos presentados en la Tabla 41, se empleó una estrategia de vuelo continuo, es decir la aeronave se mantuvo volando en modo manual mientras la capacidad de batería lo permita, contabilizando las capturas de vehículos que se obtiene durante el intervalo de tiempo medido con visión artificial. En el Anexo Q se muestran las pruebas de vuelo del UAV en cada uno de los parqueaderos.

Tabla 41. Parqueaderos analizados.

Parqueadero	Imagen Referencia	Características
1		- Parqueadero abierto - 26 espacios designados para estacionamiento

Parqueadero	Imagen Referencia	Características
2		- Parqueadero abierto - 20 espacios designados para estacionamiento
3		- Parqueadero cubierto - 28 espacios designados para estacionamiento

3.1.20 Tiempo de vuelo en parqueaderos

Utilizando una LiPo de 4 celdas de 1550 mAh con un rango de voltaje de operación entre 14.8 V-16.8 V, se salvaguardo la vida útil de la batería utilizando un 80% de su capacidad, la batería no se descargó a un voltaje menor a 14 V. Durante el vuelo del dron se mantuvo una altura determinada para cada caso, la cual fue variando en intervalos. Los tiempos medidos para cada parqueadero en modo de vuelo manual se detallan en la Tabla 42, especificando el lugar, el número de automóviles capturados y la velocidad a la que se movía el dron en cada caso.

Tabla 42. Tiempos de vuelo en parqueaderos utilizando modo manual.

Parqueadero	Número de Automóviles	Altura [m]	Velocidad del Dron [m/s]	Tiempo de vuelo [min]
1	44	1.0	0.3	9.95
	41	1.5	0.3	9.22
	37	2.0	0.4	8.40

Parqueadero	Número de Automóviles	Altura [m]	Velocidad del Dron [m/s]	Tiempo de vuelo [min]
2	43	1.0	0.3	9.78
	39	1.5	0.3	8.88
	36	2.0	0.4	8.23
3	50	1.0	0.3	11.60
	48	1.5	0.3	11.05
	45	2.0	0.4	10.38
Promedio	42	1.5	0.3	9.72

Se obtuvo un tiempo promedio de vuelo en parqueaderos utilizando el modo manual de 9.72 minutos, capturando 42 automóviles en promedio. Considerando que la velocidad del viento en el parqueadero 3 es menor ya que es un entorno cerrado, por lo cual el tiempo de vuelo en los 3 casos de altitud fue mayor que en los parqueaderos 1 y 2, además para una altura mayor el tiempo de vuelo fue menor ya que requiere de mayor potencia en los motores para incrementar la altura.

3.1.21 Detección de placas vehiculares con visión artificial

Para calificar el rendimiento en la detección de placas utilizando la red neuronal entrenada con YOLOv8m se realizaron pruebas en diferentes entornos de iluminación presentados en la Tabla 43.

Tabla 43. Entornos de iluminación.

Nivel	Característica
Bajo	Anochecer o día sombrío
Medio	Día parcialmente nublado o soleado bajo
Alto	Día soleado alto

Para la detección de placas de automóviles en los 3 tipos de parqueaderos propuestos se utilizó una altura promedio de vuelo de 1.5 m calculada previamente en la Tabla 42, y se varió el entorno de iluminación, con el propósito de medir el grado de efectividad de la red neuronal entrenada. La recolección de los datos se presenta en la Figura 152 con un total de 50 automóviles analizados en cada parqueadero variando su entorno.

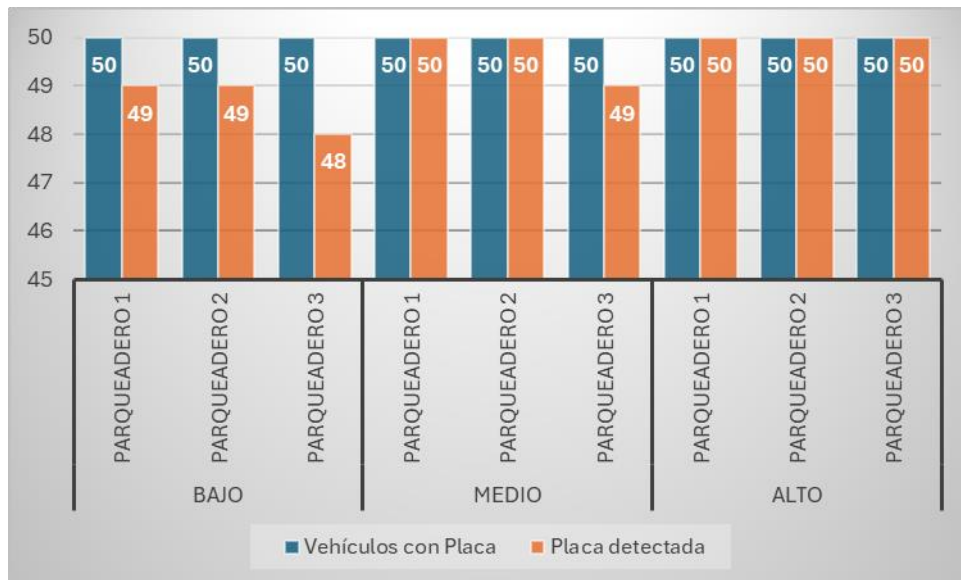


Figura 152. Detección de placas con el modelo de red neuronal.

En el caso de iluminación baja, aunque el modelo mostró un alto desempeño general, las detecciones fueron ligeramente inferiores. En el parqueadero 3, el modelo presentó su menor rendimiento con 48 placas detectadas, mientras que en los parqueaderos 1 y 2 se detectaron 49 placas. En condiciones de iluminación media, el rendimiento mejoró, alcanzando detecciones constantes de 50 placas en los parqueaderos 1 y 2, mientras en el parqueadero 3 se detectaron 49 placas. Finalmente, en condiciones de iluminación alta, el modelo mantuvo un desempeño óptimo, detectando la totalidad de las placas vehiculares (50 de 50) en todos los casos.

Estos resultados reflejan la capacidad de la red neuronal para adaptarse a diferentes condiciones lumínicas, destacándose su eficiencia en entornos con iluminación media y alta. Sin embargo, también sugieren que en condiciones de iluminación baja podría ser necesario optimizar el modelo o ajustar parámetros, como la calidad de la imagen capturada o el preprocesamiento de datos, para garantizar una mayor precisión en la detección de placas vehiculares.

En el estudio realizado, no se consideraron las placas vehiculares que presentan un alto nivel de reflectividad debido a la incidencia directa de la luz solar. Este fenómeno ocurre porque las placas están diseñadas para reflejar la luz y ser visibles en condiciones nocturnas, pero bajo reflexión de la luz solar, la imagen capturada reduce significativamente el contraste. Estas condiciones adversas dificultan el

reconocimiento óptico y la detección precisa de objetos, ya que los algoritmos pierden detalles esenciales de la placa vehicular. Por esta razón, se excluyeron de los análisis los casos cuando la luz solar directa provoca un efecto de alta reflectividad. En la Figura 153 se muestran ejemplos de cómo la luz provoca reflexión en la placa, lo que afecta la visualización.



Figura 153. Reflectividad de luz solar en placas vehiculares.

A continuación, se presentan los resultados del análisis de detección de placas vehiculares bajo tres niveles de iluminación: baja, media y alta. El estudio considera escenarios con reflexión nula, excluyendo casos donde la luz directa provoca alta reflectividad, para garantizar condiciones óptimas de evaluación.

a. Resultados con nivel de iluminación bajo

Con la aplicación de la Ecuación (4) para calcular el error absoluto y la Ecuación (5) para calcular el error porcentual se obtienen los parámetros base para medir la confiabilidad del sistema.

- **Error absoluto**

$$\text{Error Absoluto} = |\text{Valor real} - \text{Valor obtenido}| \quad (4)$$

$$\text{Error Absoluto} = |150 - 146|$$

$$\text{Error Absoluto} = 4$$

- **Error porcentual**

$$\text{Error Porcentual} = \frac{\text{Error Absoluto}}{\text{Valor Real}} * 100 \quad (5)$$

$$\text{Error Porcentual} = \frac{4}{150} * 100$$

$$\text{Error Porcentual} = 2.66\%$$

- **Confiabilidad**

El error porcentual es de 2.66% para un nivel de luminosidad bajo de un número total de 150 automóviles en diferentes parqueaderos, con este valor se determina que el sistema tiene un valor de confiabilidad del 97.34% en detecciones de placas.

b. Resultados con nivel de iluminación medio

- **Error absoluto**

$$\text{Error Absoluto} = |\text{Valor real} - \text{Valor obtenido}|$$

$$\text{Error Absoluto} = |150 - 149|$$

$$\text{Error Absoluto} = 1$$

- **Error porcentual**

$$\text{Error Porcentual} = \frac{\text{Error Absoluto}}{\text{Valor Real}} * 100$$

$$\text{Error Porcentual} = \frac{1}{150} * 100$$

$$\text{Error Porcentual} = 0.66\%$$

- **Confiabilidad**

El error porcentual es de 0.66% para un nivel de luminosidad medio de un número total de 150 automóviles en los 3 parqueaderos, con este valor se determina que el sistema tiene un valor de confiabilidad del 99.34% en detecciones de placas para este caso.

c. **Resultados con nivel de iluminación alto**

- **Error absoluto**

$$\text{Error Absoluto} = |\text{Valor real} - \text{Valor obtenido}|$$

$$\text{Error Absoluto} = |150 - 150|$$

$$\text{Error Absoluto} = 0$$

- **Error porcentual**

$$\text{Error Porcentual} = \frac{\text{Error Absoluto}}{\text{Valor Real}} * 100$$

$$\text{Error Porcentual} = \frac{0}{150} * 100$$

$$\text{Error Porcentual} = 0\%$$

- **Confiabilidad**

El error porcentual es de 0% para un nivel de luminosidad medio de un número total de 150 vehículos en los parqueaderos propuestos, con este valor se determina que el sistema tiene un valor de confiabilidad del 100% en detecciones de placas en condiciones de iluminación alta.

Se obtienen los valores de detección de placas de automóviles en diferentes entornos de iluminación, con los cuales se calcula el error del comportamiento del modelo entrenado, mediante los valores que se muestran en la Tabla 44. Se obtiene un promedio con los valores para medir la confiabilidad.

Tabla 44. Errores absolutos y porcentual en detección de placas.

Nivel Iluminación	Error Absoluto	Error Porcentual
Bajo	4	2.66%
Medio	1	0.66%
Alto	0	0%
Promedio	1.66	1.10%

El nivel de iluminación donde mayor error porcentual se tiene es el bajo con 2.66%, por lo cual una iluminación estable es un parámetro para considerar al momento de realizar las detecciones. Se tiene un error porcentual 1.10% por lo tanto el nivel de confiabilidad es 98.90% para la detección de placas.

3.1.22 Extracción de caracteres con OCR

Con el reconocimiento de placas vehiculares para la extracción de caracteres, se tomaron las capturas de placas detectadas y recortadas, en los diferentes entornos con variación de alturas, para calcular el error se presenta los caracteres de placa detectados con la visión humana y los caracteres utilizando OCR.

Considerando el apartado del tiempo de procesamiento, que corresponde al intervalo que existe desde que la imagen es capturada en el ordenador cuando detecta un vehículo, siendo procesada para la detección de placas de automóviles y extracción de caracteres con OCR hasta el envío de información a la base de datos MySQL.

Los datos de los automóviles del parqueadero 1 con un nivel de iluminación bajo a diferentes alturas se presentan en la Tabla 45. Presentando la información de las placas observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 45. Detección de caracteres de placas en parqueadero 1 con nivel de iluminación bajo.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Bajo	1	TBH3936	TBH3936	7.55	1
		PVW0921	PVK9211	8.21	0
		PDN3390	0000000	8.26	0
		TBF8484	TBF8484	9.12	1
		TBL1765	TBL1765	8.31	1
		TBG8659	TBG8659	7.75	1
		TDI0356	TDI0356	7.77	1
		TBI3021	TBI3021	7.10	1
		POT0250	PAT0260	7.24	0
		TBG3618	TBG3618	7.57	1

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
	1.5	TBH3936	TBH3906	7.86	1
		PVW0921	PVH0921	7.62	0
		PDN3390	PDN3390	7.76	1
		TBF8484	0000000	8.24	0
		TBL1765	TBL1765	7.69	1
		TBG8659	TBG8659	8.07	1
		TDI0356	TDI0356	8.06	1
		TBI3021	TBI3021	7.78	1
		POT0250	POT0250	8.56	1
		TBG3618	TBG3618	7.27	1
	2	TBH3936	TBH3936	8.11	1
		PVW0921	PVW0921	7.50	1
		PDN3390	PON3390	8.71	0
		TBF8484	TBF8484	7.32	1
		TBL1765	TBL1765	7.24	1
		TBG8659	TBG8659	6.79	1
		TDI0356	TDI0356	7.69	1
		TBI3021	TBI3021	7.59	1
		POT0250	POT0250	7.25	1
		TBG3618	TBG3618	7.36	1
Promedio				7.77	24/30

Los datos de los automóviles del parqueadero 2 con un nivel de iluminación bajo a diferentes alturas se presentan en la Tabla 46. Presentando la información de las placas observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 46. Detección de caracteres de placas en parqueadero 2 con nivel de iluminación bajo.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Bajo	1	TBG5879	TBG5879	7.66	1
		TBH4238	TBH4238	6.99	1
		TDF0891	TOF0891	7.81	0
		TBL5329	TBL5329	7.82	1
		PCL8698	PCL8689	7.26	1
		TBF5088	TBF5088	7.31	1

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
		TBG5109	TBG5109	6.92	1
		TBI6688	TBI6688	7.51	1
		TBK6360	TBK6360	7.39	1
		ICN0634	ICN0634	7.58	1
	1.5	TBG5879	00000000	7.40	0
		TBH4238	TBH4288	7.44	1
		TDF0891	TDF0891	8.20	1
		TBL5329	TBL5329	7.34	1
		PCL8698	PCI8689	8.25	0
		TBF5088	TBF5088	6.72	1
		TBG5109	TBG5109	7.21	1
		TBI6688	TBI6688	7.18	1
		TBK6360	TBK6360	6.78	1
		ICN0634	ICN0634	6.72	1
	2	TBG5879	TBC5879	7.03	0
		TBH4238	TBH4238	7.17	1
		TDF0891	TDF0891	6.86	1
		TBL5329	TBL5329	7.36	1
		PCL8698	PCL8695	7.58	0
		TBF5088	TBF5088	7.19	1
		TBG5109	TBG6109	6.83	0
		TBI6688	TBI6688	7.24	1
		TBK6360	TBK6860	7.04	0
		ICN0634	ICN0634	6.86	1
Promedio				7.28	23/30

Los datos de los automóviles del parqueadero 3 con un nivel de iluminación bajo a diferentes alturas se presentan en la Tabla 47. Presentando la información de las placas observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 47. Detección de caracteres de placas en parqueadero 3 con nivel de iluminación bajo.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Bajo	1	TEI1229	TEI1229	7.46	1
		TEI1140	TEI1140	7.06	1
		TEA0739	TEA0739	7.52	1
		TEI1055	TEI1055	7.68	1
		TBL6197	TBL6197	6.93	1
		TBJ3549	TBJ3540	7.52	0
		HBC2844	HBC2844	7.68	1
		TBF3236	TBF3230	7.76	0
		TBL4935	TBL4935	6.94	1
		TBG4686	TBG4686	7.59	0
	1.5	TEI1229	TEI1229	7.89	1
		TEI1140	TEI1140	7.92	1
		TEA0739	TEA0739	7.73	1
		TEI1055	TEI1055	7.86	1
		TBL6197	TBL6197	7.91	1
		TBJ3549	TBJ3549	7.67	1
		HBC2844	HBC2844	7.83	1
		TBF3236	TBF3230	7.95	0
		TBL4935	TBL4935	8.27	1
		TBG4686	TBG4686	7.82	1
	2	TEI1229	TEI1209	7.62	0
		TEI1140	TEI1140	7.58	1
		TEA0739	TEA0739	7.49	1
		TEI1055	TEI1055	7.73	1
		TBL6197	TBL6197	8.18	1
		TBJ3549	TBJ3549	7.58	1
		HBC2844	HBC2844	7.16	1
		TBF3236	TBF3230	7.79	0
		TBL4935	TBL4935	8.19	1
		TBG4686	TBC4680	7.63	0
Promedio				7.66	24/30

Los datos de los automóviles del parqueadero 1 con un nivel de iluminación medio a diferentes alturas se presentan en la Tabla 48. Presentando la información de las placas observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o

fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 48. Detección de caracteres de placas en parqueadero 1 con nivel de iluminación medio.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Medio	1	TBF8484	TBF8484	7.97	1
		TBA2588	TBA2588	7.69	1
		POT0250	POT0250	7.56	1
		TBG3618	TBG3618	8.46	1
		TBH6491	TBH6491	7.72	1
		TBI2767	TBI2767	7.49	1
		TBL2580	TBL2580	7.68	1
		PCK4969	PCK4909	7.81	0
		TBA7054	TBA7054	7.74	1
		TBG3885	TBC3385	8.16	0
	1.5	TBF8484	TBF8484	7.67	1
		TBA2588	TBA2588	7.80	1
		POT0250	POT0250	8.53	1
		TBG3618	TBG3618	7.65	1
		TBH6491	TBH6491	7.71	1
		TBI2767	TBI2767	7.73	1
		TBL2580	TBL2580	8.37	1
		PCK4969	PCK4969	7.88	1
		TBA7054	TBA7054	7.91	1
		TBG3385	TBG3385	7.62	1
	2	TBF8484	TBF8404	7.13	0
		TBA2588	TBA2588	7.71	1
		POT0250	POT0260	7.98	0
		TBG3618	TBG3618	7.61	1
		TBH6491	TBH6491	7.82	1
		TBI2767	TBI2767	8.59	1
		TBL2580	TBL2580	7.68	1
		PCK4969	PCK4969	7.81	1
		TBA7054	TBA7054	8.29	1
		TBG3385	TBG3385	6.73	1
Promedio				7.81	26/30

Los datos de los automóviles del parqueadero 2 con un nivel de iluminación medio a diferentes alturas se presentan en la Tabla 49. Presentando la información de las placas

observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 49. Detección de caracteres de placas en parqueadero 2 con nivel de iluminación medio.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Medio	1	TBE7340	TBE7340	7.65	1
		PTK0306	PTK0306	7.58	1
		TBE1792	TBE1792	9.17	1
		TBH1974	TBH1974	7.68	1
		TBG9194	TBG9194	7.94	1
		PDF5060	PDF5060	7.62	1
		TBJ6414	TBJ6414	8.19	1
		HCJ0309	HCJ0309	7.81	1
		TBI4012	TBI4012	7.23	1
		TBL6042	TBL6042	7.29	1
	1.5	TBE7340	TBE7340	6.86	1
		PTK0306	DTK0300	7.59	0
		TBE1792	TBE1792	7.61	1
		TBH1974	TBH1974	7.80	1
		TBG9194	TBG9194	7.72	1
		PDF5060	PDF5060	7.14	1
		TBJ6414	TBJ6414	7.35	1
		HCJ0309	HCJ0309	7.68	1
		TBI4012	TBI4012	7.53	1
		TBL6042	TBL6042	7.92	1
	2	TBE7340	TBE730	6.57	0
		PTK0306	PTK0306	8.69	1
		TBE1792	TBE1792	7.79	1
		TBH1974	TBH1974	7.71	1
		TBG9194	TBC9194	8.73	0
		PDF5060	PDF5000	7.62	0
		TBJ6414	TBJ6414	7.67	1
		HCJ0309	HCL0300	6.74	0
		TBI4012	TBI4012	8.68	1
		TBL6042	TBL6042	7.63	1
Promedio				7.70	25/30

Los datos de los automóviles del parqueadero 3 con un nivel de iluminación medio a diferentes alturas se presentan en la Tabla 50. Presentando la información de las placas observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 50. Detección de caracteres de placas en parqueadero 3 con nivel de iluminación medio.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Medio	1	TBI7428	TBI7428	6.59	1
		PBI6650	PBI6650	7.57	1
		PIU0783	PIU783	6.82	1
		TBL9068	TBL9068	7.45	1
		PBN3510	PBN3510	6.91	1
		TBH8429	TBH8429	7.35	1
		TBM1621	TBH1621	7.96	0
		TBJ8108	TBJ8108	7.68	1
		POE0149	POE0149	7.29	1
		TBJ6517	TBJ6517	9.73	1
	1.5	TBI7428	TBI7428	6.51	1
		PBI6650	PBI6650	7.91	1
		PIU0783	PIU783	7.48	1
		TBL9068	TBL9068	7.52	1
		PBN3510	PBN3510	6.83	1
		TBH8429	TBH8429	7.36	1
		TBM1621	TBM1621	7.53	1
		TBJ8108	TBJ8108	6.97	1
		POE0149	POE0149	6.73	1
		TBJ6517	TBJ6517	7.29	1
	2	TBI7428	TBI7428	7.83	1
		PBI6650	PBI6050	9.61	0
		PIU0783	PIU783	7.48	1
		TBL9068	TBL9068	7.37	1
		PBN3510	PBN3510	7.84	1
		TBH8429	TBH8429	7.62	1
		TBM1621	TBH1621	6.85	0
		TBJ8108	TBI8103	7.19	0
		POE0149	POE0149	6.93	1
		TBJ6517	TBJ6517	9.31	1
Promedio				7.51	26/30

Los datos de los automóviles del parqueadero 1 con un nivel de iluminación alto a diferentes alturas se presentan en la Tabla 51. Presentando la información de las placas observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 51. Detección de caracteres de placas en parqueadero 1 con nivel de iluminación alto.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Alto	1	TBG8659	TBG8659	6.23	1
		TBG7943	TBG7943	7.14	1
		PDD8268	PDD8268	6.67	1
		TBJ4413	TBJ4413	7.39	1
		TBA7054	TBA7054	6.91	1
		TDI0356	TDI0356	7.11	1
		TBL2580	TBL1680	7.48	0
		TBF8107	TBF8107	7.01	1
		PDE1902	PDE1902	7.26	1
		TBD3623	TBD3023	6.79	1
	1.5	TBG8659	TBG8699	7.25	0
		TBG7943	TBG7943	9.18	1
		PDD8268	PDD8268	6.54	1
		TBJ4413	TBJ4413	6.92	1
		TBA7054	TBA7054	7.13	1
		TDI0356	TDI0356	7.19	1
		TBL2580	TBL2580	6.63	1
		TBF8107	TBF8107	7.08	1
		PDE1902	PDE1902	7.13	1
		TBD3623	TBD3623	6.78	1
	2	TBG8659	TBG8659	7.28	1
		TBG7943	TBG7943	6.86	1
		PDD8268	PDD8218	6.73	0
		TBJ4413	TBJ4413	7.39	1
		TBA7054	TBA7054	6.72	1
		TDI0356	TDI0856	6.85	0
		TBL2580	TBL2580	7.17	1
		TBF8107	TBF8107	6.93	1
		PDE1902	PDE1902	7.14	1
		TBD3623	TBD3623	7.29	1
Promedio				7.07	26/30

Los datos de los automóviles del parqueadero 2 con un nivel de iluminación alto a diferentes alturas se presentan en la Tabla 52. Presentando la información de las placas observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 52. Detección de caracteres de placas en parqueadero 2 con nivel de iluminación alto.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Alto	1	TBD6640	TBD6640	7.13	1
		TBF5639	TBF5639	6.91	1
		TBL5329	TBL5329	7.04	1
		PDK2090	PDK2090	7.37	1
		TBL6042	TBL6042	6.78	1
		TBJ6414	TBI6414	6.61	0
		TBI4012	TBI4012	6.96	1
		PDF5060	PDF5060	7.15	1
		PBA7796	PBA7796	6.85	1
		TBG9194	TBG9194	7.28	1
	1.5	TBD6640	TBD6548	9.18	0
		TBF5639	TBF5639	7.25	1
		TBL5329	TBL5329	7.86	1
		PDK2090	PDK2090	7.09	1
		TBL6042	TBL6042	6.94	1
		TBJ6414	TBJ6414	7.27	1
		TBI4012	TBI4012	7.19	1
		PDF5060	PDF5060	6.84	1
		PBA7796	PBA7796	6.92	1
		TBG9194	TBG9194	6.72	1
	2	TBD6640	TBD6640	6.97	1
		TBF5639	TBF5689	8.29	0
		TBL5329	TBL5329	7.08	1
		PDK2090	PDK2090	7.73	1
		TBL6042	TBL6042	7.11	1
		TBJ6414	TBJ6414	6.69	1
		TBI4012	TBI4012	7.37	1
		PDF5060	PDF5060	6.79	1
		PBA7796	PBA7796	6.91	1
		TBG9194	TBG9194	7.63	1
Promedio				7.19	27/30

Los datos de los automóviles del parqueadero 3 con un nivel de iluminación alto a diferentes alturas se presentan en la Tabla 53. Presentando la información de las placas observadas visualmente y las detectadas por el sistema OCR, el tiempo requerido para procesar cada placa y el resultado [0/1], que indica si la detección fue exitosa (1) o fallida (0). Al final, se muestra el promedio del tiempo de procesamiento y la relación entre las placas correctamente detectadas y el total analizado.

Tabla 53. Detección de caracteres de placas en parqueadero 3 con nivel de iluminación alto.

Nivel Iluminación	Altura [m]	Visión Humana	Detección OCR	Tiempo [s]	Resultado [0/1]
Alto	1	XBA7914	XBA7914	6.51	1
		TBA1432	TBA1432	7.28	1
		PDI8670	PDI8670	7.19	1
		TBF9022	TBF9022	9.83	1
		HBE1201	HBE1201	7.03	1
		TBB6419	TBB6419	6.15	1
		TBL4296	TBL4296	7.06	1
		TBI8898	TBI8898	7.21	1
		TBJ6240	TBJ6240	7.37	1
		TBA2565	TBA2565	6.89	1
	1.5	XBA7914	XBA1914	8.18	0
		TBA1432	TBA1432	6.68	1
		PDI8670	PDI8670	7.02	1
		TBF9022	TBF9022	6.97	1
		HBE1201	HBE1201	7.17	1
		TBB6419	TBB6419	7.08	1
		TBL4296	TBL4296	6.89	1
		TBI8898	TBI8898	8.91	1
		TBJ6240	TBJ6240	7.36	1
		TBA2565	TBA2666	8.29	0
	2	XBA7914	XBA7914	7.29	1
		TBA1432	TBA7982	7.01	0
		PDI8670	PDI8670	6.73	1
		TBF9022	TBF9022	7.17	1
		HBE1201	HBE1201	7.04	1
		TBB6419	TBB6419	6.85	1
		TBL4296	TBJ4296	6.93	0
		TBI8898	TBI8898	7.26	1
		TBJ6240	TBJ6240	6.81	1
		TBA2565	TBA2509	9.33	0
Promedio				7.31	25/30

Recolectada la información de extracción de caracteres en los entornos propuestos se presenta el resumen compacto en la Tabla 54 de los datos de tiempo promedio de procesamiento y la relación entre las placas correctamente detectadas y el total analizado en cada escenario analizado.

Tabla 54. Resumen de parámetros en la detección de caracteres de placas vehiculares.

Nivel Iluminación	Parqueadero	Tiempo [s]	Resultado
Bajo	1	7.77	24/30
	2	7.28	23/30
	3	7.66	24/30
Medio	1	7.81	26/30
	2	7.70	25/30
	3	7.51	26/30
Alto	1	7.07	26/30
	2	7.19	27/30
	3	7.31	25/30
Promedio		7.47	226/270

Se obtienen los valores de la extracción de caracteres de placas de automóviles en diferentes entornos de iluminación, con los cuales se calcula el error del reconocimiento por OCR, utilizando la Ecuación (4) y Ecuación (5) respectivamente. Los valores que se detallan en la Tabla 55, obteniendo un promedio con los valores para medir la confiabilidad.

- **Error absoluto nivel de iluminación bajo**

$$\text{Error Absoluto} = |90 - 71|$$

$$\text{Error Absoluto} = 19$$

- **Error porcentual nivel de iluminación bajo**

$$\text{Error Porcentual} = \frac{19}{90} * 100$$

$$\text{Error Porcentual} = 21.11\%$$

- **Error absoluto nivel de iluminación medio**

$$\text{Error Absoluto} = |90 - 77|$$

$$\text{Error Absoluto} = 13$$

- **Error porcentual nivel de iluminación medio**

$$\text{Error Porcentual} = \frac{13}{90} * 100$$

$$\text{Error Porcentual} = 14.44\%$$

- **Error absoluto nivel de iluminación alto**

$$\text{Error Absoluto} = |90 - 78|$$

$$\text{Error Absoluto} = 12$$

- **Error porcentual nivel de iluminación alto**

$$\text{Error Porcentual} = \frac{12}{90} * 100$$

$$\text{Error Porcentual} = 13.33\%$$

Tabla 55. Errores absolutos y porcentual en detección de placas.

Nivel Iluminación	Error Absoluto	Error Porcentual
Bajo	19	21.11%
Medio	13	14.44%
Alto	12	13.33%
Promedio	14.66	16.29%

El nivel de iluminación donde mayor error porcentual se tiene es el bajo con 21.11%, por lo cual la iluminación debe considerar para el uso de OCR. Se tiene un error porcentual 16.29% por lo tanto el nivel de confiabilidad porcentual es 83.71% para la extracción de caracteres.

3.1.23 Evaluación del Sistema de Detección y Reconocimiento OCR

Con los valores obtenidos de la confiabilidad de la detección de placas (0.9890) y del reconocimiento de caracteres por OCR (0.8371) respectivamente, se planteó realizar la evaluación del sistema que opera en una secuencia, es decir, primero detecta la placa y luego extrae los caracteres, el nivel de confiabilidad total calculado con la Ecuación (6) es importante, representando la probabilidad conjunta de éxito de ambos procesos,

$$\text{Confiabilidad Total} = \text{Detección de Placas} * \text{Reconocimiento OCR} * 100\% \quad (6)$$

$$\text{Confiabilidad Total} = 0.9890 * 0.8371 * 100\%$$

$$\text{Confiabilidad Total} = 82.78\%$$

El nivel de confiabilidad total del sistema completo es 82.78%, siendo un indicador que, cuando ambos subsistemas trabajan en conjunto, el sistema tiene esa probabilidad de éxito en todo el proceso. Este análisis muestra cómo la confiabilidad individual de los subsistemas impacta el rendimiento compacto del sistema.

3.1.24 Tiempo de procesamiento de información

En la Tabla 56 se detalla el tiempo promedio en segundos requerido para el envío de información en eventos donde se ha realizado el reconocimiento de placas vehiculares y existe coincidencia con placas previamente reportadas. Esta tabla permite analizar la eficiencia del sistema en la detección y transmisión de datos, proporcionando una referencia clave para evaluar el desempeño del proceso de identificación y notificación.

Tabla 56. Tiempo de envío de información sobre eventos con reconocimiento y coincidencia de placas reportadas.

Placa	Tiempo [s]
TBA9845	12.52
HCM0947	15.23
TBI7514	14.26
TBG8430	12.55
TBI3021	15.58
TBL6310	9.94
TBF8484	13.77
PBM1668	7.50
TBB9845	11.05
TBM7575	15.53
Promedio	12.79

Hay situaciones en las que no se identifica la placa vehicular. En la Tabla 57 se especifica el tiempo promedio en segundos requerido para enviar una notificación desde la base de datos al grupo de Telegram y al correo electrónico de un administrador.

Tabla 57. Tiempo de envío de información de eventos sin reconocimiento de placas.

Vehículo sin placa	Tiempo [s]
1	12.22
2	8.52
3	9.04
4	11.24
5	11.61
6	9.94
7	12.77
8	10.50
9	11.05
10	10.53
Promedio	10.74

Para tener el tiempo promedio del procesamiento de información desde que el vehículo es capturado con su respectiva detección de placa o no detección y el envío de información a la base de datos para su inscripción y verificación respectiva.

Se considera el tiempo promedio medido en el ordenador desde la detección de placa y reconocimiento por OCR (T1) de la Tabla 54, adicionando el tiempo promedio de la Tabla 56 del envío de información en vehículo reportado (T2) y el tiempo de la Tabla 57 de envío de información de un vehículo sin placa (T3), el resumen de los tiempos se presenta en la Tabla 58.

Tabla 58. Tiempos promedio en el procesamiento de información.

Tiempo de detección de placa y reconocimiento OCR [s]	Tiempo de envío de información en vehículo reportado [s]	Tiempo de envío de información en vehículo sin placa [s]
7.47	12.79	10.74

a. Tiempo total en el procesamiento de información para vehículo reportado

Para calcular el tiempo de procesamiento en la detección de vehículos reportados se debe adicionar el tiempo de detección de placa y reconocimiento OCR más el tiempo de envío de información en vehículo reportado, con la Ecuación (7).

$$\text{Tiempo de procesamiento 1} = T1 + T2 \quad (7)$$

$$\text{Tiempo de procesamiento 1} = 7.47 + 12.79$$

$$\text{Tiempo de procesamiento 1} = 20.26 \text{ segundos}$$

b. Tiempo total en el procesamiento de información para vehículo sin placa

Para calcular el tiempo de procesamiento para un vehículo sin placa se debe adicionar el tiempo de detección de placa y reconocimiento OCR más el tiempo de envío de información en vehículo sin placa, con la Ecuación (8).

$$\text{Tiempo de procesamiento 2} = T1 + T3 \quad (8)$$

$$\text{Tiempo de procesamiento 2} = 7.47 + 10.74$$

$$\text{Tiempo de procesamiento 2} = 18.21 \text{ segundos}$$

Con los tiempos respectivos calculados para cada caso se obtuvo que el tiempo promedio de procesamiento cuando existe un vehículo reportado es de 20.26 segundos y el tiempo cuando existe un vehículo sin reconocimiento de placa es de 18.21 segundos.

3.1.25 Prueba del Sistema de Detección mediante el Vuelo Autónomo del UAV

El vuelo de un UAV en un estacionamiento de vehículos presenta desafíos significativos debido a las características del entorno. La presencia de personas y la proximidad a los vehículos incrementa el riesgo de posibles accidentes. Además, en entornos como los estacionamientos, se generan mayores interferencias electromagnéticas, especialmente cuando el UAV vuela cerca del suelo y de superficies metálicas, como las carrocerías de los vehículos. Estas condiciones pueden afectar la precisión de los sistemas de posicionamiento global (GPS), lo que resulta en una identificación incorrecta de las coordenadas de latitud y longitud. Esto, a su vez, puede provocar desviaciones en las rutas programadas, comprometiendo tanto la eficacia del envío de video en tiempo real a la estación terrestre como la seguridad operativa. Por estas razones, se ha decidido realizar las pruebas del sistema en un entorno más seguro, como las canchas de la Universidad Técnica de Ambato como se muestra en la Figura 154. Este espacio controlado permitirá simular la operación del UAV con un vehículo, garantizando así la evaluación del sistema en condiciones más seguras y adecuadas.



Figura 154. Canchas de la Universidad Técnica de Ambato Campus Huachi.

Se estableció el punto HOME, como el punto de partida, posteriormente, se definieron dos puntos estratégicos para el plan de vuelo, considerando que el punto 1 corresponde a la ubicación del vehículo a evaluar, mientras que el 2 representan la ubicación donde desciende el dron, En la Figura 155 se ilustra el plan de vuelo correspondiente a la ruta 1, mientras que en la Figura 156 se detalla el plan de vuelo para la ruta 2.

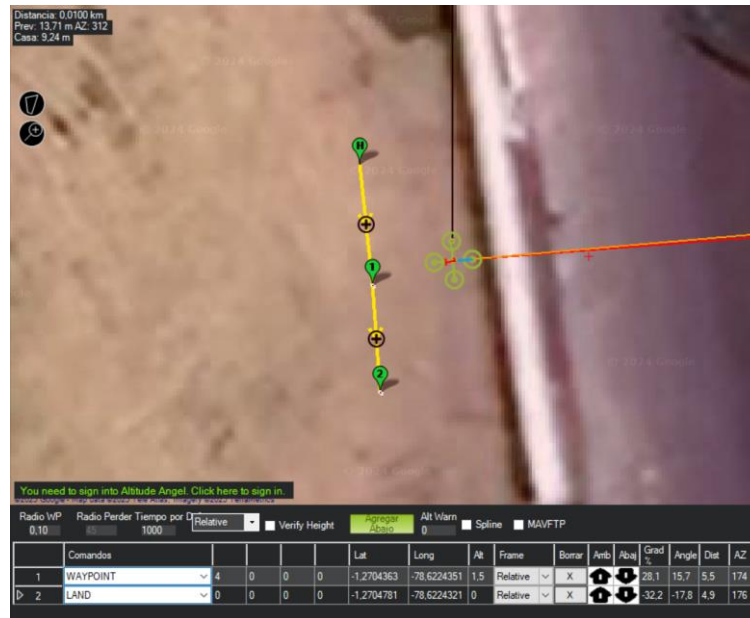


Figura 155. Plan de vuelo del UAV ruta 1.

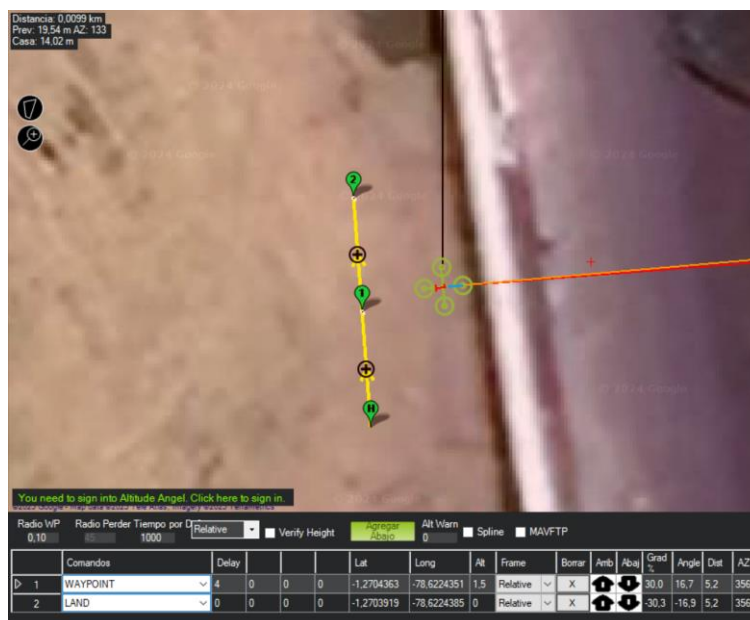


Figura 156. Plan de vuelo del UAV ruta 2.

Previamente planificada la ruta de la trayectoria del dron con los parámetros respectivos, se considera realizar algunas intervenciones variando sus configuraciones para calcular la confiabilidad del sistema de detección con la aplicación del UAV ejecutando rutas, en la Tabla 59 se presentan los datos de los vuelos por ruta.

De acuerdo con las pruebas manuales realizadas con el UAV, la trayectoria se desplazará a una velocidad de 0.3 m/s, velocidad promedio obtenida en las pruebas de

vuelo manual. Se consideró un tiempo de espera a partir de 3 segundos ya que es el tiempo programado para capturar el vehículo detectado utilizando YOLOv8m, suficiente para que el dron se estabilice correctamente. Además, este intervalo permite al modelo realizar la captura de imágenes una vez que un vehículo entra en el campo de visión de la cámara. Considerando también las condiciones de iluminación adecuadas determinadas en las pruebas de vuelo manual para una correcta detección de caracteres.

Tabla 59. Ruta en parqueaderos utilizando modo automático.

Rutas	Altura de posicionamiento [m]	Tiempo de posicionamiento [s]	Visión Humana	Detección OCR	Resultado [0/1]
1	1	3	PWH468	PWH468	1
		4	PWH468	PWH468	1
		5	PWH468	PVH463	0
	1.5	3	PWH468	PWH468	1
		4	PWH468	PWH468	1
		5	PWH468	PMH463	0
	2	3	PWH468	PWH468	1
		4	PVH468	PWH468	0
		5	PWH468	0000000	0
2	1	3	PWH468	PWH468	1
		4	PWH468	PWH468	1
		5	PWH468	PWH468	1
	1.5	3	PWH468	PWH468	1
		4	PWH468	PWH468	1
		5	PWH468	PWH468	1
	2	3	PWH468	PWH463	0
		4	PWH468	PWH468	1
		5	PWH468	PVH463	0
Promedio					12/18

Se obtienen los valores de la extracción de caracteres de la placa vehicular en diferentes entornos de altura de posicionamiento, considerando que existió un rango de variación con los parámetros configurados para altura y radio de posicionamiento al momento de hacer las pruebas. Con la información extraída se calcula el error del reconocimiento por OCR absoluto y porcentual, utilizando la Ecuación (4) y Ecuación (5) respectivamente.

- **Error absoluto**

$$\text{Error Absoluto} = |18 - 12|$$

$$\text{Error Absoluto} = 6$$

- **Error porcentual**

$$\text{Error Porcentual} = \frac{6}{18} * 100$$

$$\text{Error Porcentual} = 33.33\%$$

Para el uso de rutas dentro del sistema acoplando el reconocimiento OCR. Se tiene un error porcentual 33.33% por lo tanto el nivel de confiabilidad porcentual es 66.64% para la detección de placas, debido a que en las pruebas realizadas la ubicación por GPS jugo un papel fundamental para el posicionamiento y se notaron variaciones desde los parámetros configurados desde el software Mission Planner.

3.1.26 Presupuesto

Para el cálculo del presupuesto total del sistema de detección de vehículos robados, se tomó en cuenta el presupuesto del diseño y materiales utilizados en la construcción del sistema. La estimación del presupuesto del diseño se basa en el tiempo empleado en su diseño y el sueldo promedio de un ingeniero en telecomunicaciones en Ecuador. Según el informe de salarios mínimos sectoriales del año 2024 emitido por el Ministerio de Trabajo, un especialista en telecomunicaciones gana \$494.51 mensuales. Para determinar el salario diario, se considera un promedio de 20 días laborales al mes. Aplicando estos datos a la Ecuación (9), se calcula el salario diario de un especialista en Telecomunicaciones:

$$\text{Salario}_{diario} = \frac{\text{Salario}_{mensual}}{\text{Días}_{laborales}} \quad (9)$$

$$\text{Salario}_{diario} = \frac{494.51}{20}$$

$$\text{Salario}_{diario} = \$24.72$$

Para determinar el salario correspondiente por hora, se utiliza la Ecuación (10), considerando que la duración de la jornada laboral es de 8 horas al día.

$$\text{Salario}_{\text{hora}} = \frac{\text{Salario}_{\text{diario}}}{\text{Horas}_{\text{laborales}}} \quad (10)$$

$$\text{Salario}_{\text{hora}} = \frac{24.72}{8}$$

$$\text{Salario}_{\text{hora}} = \$3.09$$

Para determinar el presupuesto destinado al diseño, se empleó la Ecuación (11), considerando que la dedicación diaria al proyecto fue de 4 horas, distribuidas a lo largo de un periodo de 4 meses. En total, se acumularon 336 horas laborables.

$$\text{Presupuesto}_{\text{diseño}} = \text{Salario}_{\text{hora}} * \text{Horas}_{\text{laborables}} \quad (11)$$

$$\text{Presupuesto}_{\text{diseño}} = 2.94 * 336$$

$$\text{Presupuesto}_{\text{diseño}} = \$987.84$$

En la Tabla 60 proporciona una descripción detallada del presupuesto de los materiales.

Tabla 60. Presupuesto de los materiales del sistema.

Ítem	Recursos	Cantidad	Precio Unitario	Precio Total
1	Cámara FPV CMOS Orange 1200	1	\$25.00	\$25.00
2	Cuadricóptero SpeedyBee F405 V3	1	\$541.00	\$541.00
3	Batería LiPo 4S 14.8v ~16.8V	1	\$37.38	\$37.38
4	GPS Matek Systems M10Q05883	1	\$42.00	\$42.00
5	Cargador balanceador imax B6AC	1	\$28.00	\$28.00
6	Video transmisor (VTX 801)	1	\$25.00	\$25.00
7	Video receptor (VRX ROTG01)	1	\$45.00	\$45.00
8	Antena Foxxer Lollipop	1	\$18.00	\$18.00
9	Computador HP con tarjeta gráfica NVIDIA GEFORCE GTX	1	\$744.99	\$744.99
10	Radio control (Jumper T-pro)	1	\$110.00	\$110.00
Subtotal				\$1615.99
Imprevistos (10%)				\$161.59
Total				\$1777.58

Se obtiene el presupuesto total del sistema usando la Ecuación (12), sumando el presupuesto del diseño y construcción del sistema de la siguiente manera.

$$\mathbf{Presupuesto_{total} = Presupuesto_{dise\tilde{n}o} + Presupuesto_{materiales}} \quad (12)$$

$$Presupuesto_{total} = 987.84 + 1777.58$$

$$Presupuesto_{total} = \$2765.42$$

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La investigación permitió identificar que las tecnologías más efectivas para la recuperación de vehículos robados incluyen sistemas de cámaras de vigilancia algunos combinados con técnicas avanzadas de inteligencia artificial, como algoritmos de aprendizaje profundo para reconocimiento de vehículos de placas. Estas herramientas ofrecen un gran grado de adaptabilidad en diversos entornos y facilitan el manejo de grandes volúmenes de datos en tiempo real, lo cual resulta adecuado en operaciones de monitoreo y recuperación.
- Se desarrolló un sistema tecnológico avanzado que integra herramientas como UAVs, cámaras FPV y algoritmos de detección como YOLOv8 para la detección de automóviles robados. Este sistema no solo optimiza los procesos tradicionales de identificación vehicular, sino que también demuestra una alta capacidad de adaptación a condiciones variables, ofreciendo una solución robusta y eficiente para combatir el robo vehicular. El uso del modelo YOLOv8 permitió alcanzar una tasa de precisión del 93% en la detección y reconocimiento de matrículas vehiculares. Este resultado fue validado en escenarios reales, demostrando que el sistema puede operar bajo diversas condiciones de iluminación y entornos complejos, garantizando un rendimiento constante.
- La gestión de datos en la nube a través de Azure facilitó la escalabilidad y la seguridad de la información. Las pruebas realizadas en distintos parqueaderos evidenciaron que el UAV tiene una autonomía de vuelo adecuada. Asimismo, el sistema demostró ser capaz de operar eficientemente en parqueaderos con alta densidad vehicular, validando su aplicabilidad en escenarios reales.
- El vuelo manual del UAV demostró una mayor precisión en la navegación dentro de los parqueaderos, ya que el control directo permitía maniobras más detalladas y ajustadas al entorno. Por su parte, la integración del sistema GPS facilitó la implementación de navegación autónoma, favoreciendo la

automatización de las trayectorias. Sin embargo, este método presentó limitaciones en su precisión debido a errores en el posicionamiento y a la interferencia electromagnética provocada por la cercanía a estructuras metálicas, como las carrocerías de los automóviles.

- La implementación del sistema representa una solución rentable que contribuye significativamente a la seguridad pública. Además, fomenta la confianza ciudadana en el uso de tecnologías innovadoras como los UAV y la incorporación de visión artificial, para optimizar los tiempos de recuperación de vehículos. El diseño modular del sistema permite su adaptación e integración con tecnologías futuras, garantizando su relevancia a largo plazo. Además, su infraestructura en la nube asegura que pueda ser escalado para abarcar mayores áreas geográficas sin comprometer su eficiencia.

4.2 Recomendaciones

- Extender el sistema a regiones urbanas y rurales con alta incidencia de robos vehiculares, incluyendo la posibilidad de integración con sistemas de vigilancia existentes como CCTV y plataformas de denuncia ciudadana. Esta ampliación podría ser respaldada mediante convenios con autoridades locales y empresas privadas.
- Incorporar mejoras en el diseño del UAV, como el uso de materiales livianos, sistemas de transmisión de video digitales para mejor calidad de imagen y baterías de alta capacidad para aumentar la autonomía de vuelo. También se recomienda investigar el uso de energía solar como fuente complementaria para operaciones prolongadas.
- Implementar programas de formación para los operadores del sistema, asegurando su correcta utilización. Asimismo, capacitar a las fuerzas de seguridad pública para maximizar la efectividad en la gestión de los datos generados.

- Establecer un plan de mantenimiento periódico que incluya actualizaciones del software de inteligencia artificial y revisiones técnicas del hardware del UAV, asegurando el funcionamiento óptimo del sistema.
- Realizar evaluaciones periódicas del impacto social, incluyendo indicadores como, tiempo de recuperación de vehículos y satisfacción del usuario. Utilizar esta información para ajustar y mejorar las estrategias de implementación.

REFERENCIAS BIBLIOGRÁFICAS

- [1] Observatorio Interamericano de Seguridad, “Encuesta de las Naciones Unidas sobre Tendencias del Delito y Operaciones de los Sistemas de Justicia Penal.” Accessed: Jan. 09, 2025. [Online]. Available: <https://www.oas.org/IOS/countries.aspx?lang=En>
- [2] Fiscalía General del Estado, “Analítica Cifras de Robo.” [Online]. Available: <https://www.fiscalia.gob.ec/analitica-cifras-de-robo/>.
- [3] L. Navas and C. Obando, “Diseño e implementación de un prototipo de sistema de seguridad integral con el fin monitorear el acceso de automóviles, utilizando visión artificial y chatbot para el ingreso al conjunto residencial,” Universidad de las Fuerzas Armadas, Latacunga, 2021.
- [4] E. Barbecho and M. Zhindón, “Diseño de un algoritmo de reconocimiento de placas vehiculares ecuatorianas usando redes neuronales convolucionales,” Cuenca, 2020.
- [5] M. Torres Cabrera, “Reconocimiento automático de la placa de un vehículo de Ecuador,” 2020.
- [6] O. Zambrana, “Detección de placas vehiculares, mediante vision artificial,” *Revistas Bolivarianas*, vol. 10, pp. 17–30, Nov. 2018.
- [7] D. Rojas, “Desarrollo de un sistema de reconocimiento de placas y su influencia en la detección de vehículos robados en la Municipalidad de San Isidro,” Lima, 2017.
- [8] J. C. Larrea, “Aseguramiento vehicular: protección al patrimonio de las personas,” Quito, 2009.
- [9] J. E. Oñate Núñez, *El incremento de las tasas de criminalidad en la ciudad de Ambato, Ecuador durante el año 2022*. Ms thesis, 2023.
- [10] K. R. Encarnación Encarnación and J. A. Guachamin Pancho, *Formulación de indicadores para el análisis de la seguridad vial en la ciudad de Cuenca*. Cuenca, 2015.

- [11] J. Sanchez and A. Enriquez, “Implantación de Sistemas de Gestión de la Seguridad Vial la Norma iso 39001,” Bogotá, 2015.
- [12] Ecuador Online, “Guía Esencial: Pasos a Seguir si te Roban el Vehículo en Ecuador.” Accessed: Jan. 09, 2025. [Online]. Available: <https://ecuadoronline.org/que-hacer-si-te-roban-el-vehiculo-en-ecuador/>
- [13] Monnet, “Localizador GPS para autos: ¿cómo funciona?” Accessed: Jan. 05, 2025. [Online]. Available: <https://monnet.com.bo/localizador-gps-para-autos-como-funciona/>
- [14] Xinyetong, “Cómo usar RFID para rastrear el automóvil.” Accessed: Jan. 05, 2025. [Online]. Available: <https://www.asiarfid.com/es/how-to-use-rfid-to-tracking-car.html>
- [15] Senstar, “Lectores automáticos de matrículas (ALPR).” Accessed: Jan. 05, 2025. [Online]. Available: <https://senstar.com/es/senstarpedia/lectores-automaticos-de-matriculas/>
- [16] Imasel Seguridad, “¿Qué es CCTV? ¿Cuáles son sus funciones y objetivos?” Accessed: Jan. 05, 2025. [Online]. Available: <https://www.imsel.com/que-es-cctv-y-cual-es-su-funcion/>
- [17] L. Bastidas, “Tipificación como delito a la alteración de los sistemas de identificación vehicular en la legislación penal ecuatoriana.” Accessed: Jan. 04, 2025. [Online]. Available: <https://www.dspace.uce.edu.ec/entities/publication/da348374-9a22-4696-a8f4-5d36167b7bec>
- [18] C. Colcha, “Identificación de grabados y marcas seriales en el robo de vehículos en la ciudad de Riobamba,” Jul. 2023, Accessed: Jan. 04, 2025. [Online]. Available: <http://dspace.unach.edu.ec/handle/51000/11299>
- [19] LEXIS FINDER, “LEY ORGÁNICA DE TRANSPORTE TERRESTRE TRÁNSITO Y SEGURIDAD VIAL,” Quito, 2008.
- [20] Dirección General de aviación Civil, “Normas Generales que Regulan la Operación de Aeronaves Pilotadas a Distancia (RPAS),” Quito, Mar. 2024.

[21] Agencia de Regulación y Control de las Telecomunicaciones, “PLAN NACIONAL DE FRECUENCIAS,” 2021.

[22] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, “Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey,” *Networks*, vol. 72, no. 4, pp. 411–458, 2018, doi: <https://doi.org/10.1002/net.21818>.

[23] C. Valero, “Todos los tipos de drones según el uso, diseño o control,” Alicante, May 2024.

[24] C. Sánchez, “Vehículos aéreos no tripulados: descripciones generales y aplicaciones,” *ResearchGate*, 2017.

[25] C. Cuerno-Rejado, L. García-Hernández, A. Sanchez-Carmona, A. Carrio, J. L. Sánchez-López, and P. Company, “EVOLUTION OF THE UNMANNED AERIAL VEHICLES UNTIL PRESENT,” *Dyna(spain)*, vol. 91, no. 3, p. 332, 2016.

[26] ARDUPILOT, “ArduPilot Documentation.” Accessed: Nov. 15, 2024. [Online]. Available: <https://ardupilot.org/planner/docs/mission-planner-overview.html>

[27] MicratheneFPV, “Betaflight vs INAV: una comparación entre los dos softwares líderes en el control de drones.” Accessed: Dec. 12, 2024. [Online]. Available: <https://www.micrathenefpv.com/t-es-es/blog/betaflight-vs-inav-comparing-the-two-leading-drone-control-software>

[28] M. Hinojosa Pérez, “Diseño y construcción de drone autónomo para medición de radiación solar,” Universidad de Sevilla, Sevilla, 2019.

[29] Amazon, “Flysky FS-i6s 2.4G 10CH AFHDS 2A Transmisor RC con receptor FS-iA6B para RC Drone.” Accessed: Dec. 29, 2024. [Online]. Available: <https://tinyurl.com/6p3zk5mf>

[30] Amazon, “DTXMX Flysky FS-i6 6 canales 2.4G RC Transmisor y receptor FS-iA6B.” Accessed: Dec. 29, 2024. [Online]. Available: <https://tinyurl.com/yjv2vdex>

[31] Amazon, “Jumper T Pro OpenTX Hall Sensor Gimbal Control remoto con JP4IN1 Módulo Multi-Protocolo Radio.” Accessed: Dec. 29, 2024. [Online]. Available:

<https://www.amazon.com/-/es/Jumper-OpenTX-Sensor-Control-Multi-Protocolo/dp/B09Q378HGR>

[32] Amazon, “Matek Systems MATEKSYS M10Q-5883 GNSS.” Accessed: Dec. 29, 2024. [Online]. Available: <https://www.amazon.com/-/es/Matek-Systems-MATEKSYS-M10Q-5883-Br%C3%BAjula/dp/B0C4V5Y61T>

[33] Amazon, “Módulo GPS HMC5883.” Accessed: Dec. 29, 2024. [Online]. Available: <https://tinyurl.com/2ue72ya4>

[34] Amazon, “Readytosky Módulo GPS M8N.” Accessed: Dec. 29, 2024. [Online]. Available: <https://www.amazon.com/-/es/Readytosky-Br%C3%BAjula-incorporada-controlador-est%C3%A1ndar/dp/B01KK9A8QG>

[35] Grupo Electrostore, “BATERÍA LIPO 3S 11,1V TURNIGY NANO-TECH.” Accessed: Jan. 05, 2025. [Online]. Available: <https://grupoelectrostore.com/shop/baterias/baterias-lipo/turnigy-nano-tech/bateria-lipo-3s-111v-1000mah-turnigy-nano-tech-45-90c/>

[36] Grupo Electrostore, “BATERÍA LIPO 4S 14,8V TATTU FUNFLY 100C.” Accessed: Jan. 14, 2025. [Online]. Available: <https://grupoelectrostore.com/shop/baterias/baterias-lipo/tattu-funfly/bateria-lipo-4s-148v-1550mah-tattu-funfly-100c/>

[37] Grupo Electrostore, “BATERÍA LIPO 6S 22,2V TATTU.” Accessed: Jan. 05, 2025. [Online]. Available: <https://grupoelectrostore.com/shop/baterias/baterias-lipo/tattu/bateria-lipo-6s-222v-650mah-tattu-r-line-95c-xt30/>

[38] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[39] A. Usama, M. Dora, M. Elhadidy, H. Khater, and O. Alkelany, “First Person View Drone - FPV,” *IUGRC*, pp. 437–440, 2021.

[40] Amazon, “CADDXFPV Ratel 2 1200TVL FPV Micro cámara.” Accessed: Dec. 29, 2024. [Online]. Available: <https://www.amazon.com/-/es/CADDXFPV-Ratel-1200TVL-c%C3%A1mara-Starlight/dp/B095YG62FT>

- [41] ARIES, “1200TVL CMOS 2.5mm/2.8mm 130/120 Degree 16:9 Mini FPV Camera 5V-12V For RC Drone.” Accessed: Jan. 04, 2025. [Online]. Available: <https://tinyurl.com/59emtajw>
- [42] SpeedyBee, “SpeedyBee TX800.” Accessed: Dec. 29, 2024. [Online]. Available: <https://www.speedybee.com/speedybee-tx800/>
- [43] Eachine, “Eachine TX5258 5.8G 72CH.” Accessed: Dec. 29, 2024. [Online]. Available: <https://www.eachine.com/Eachine-TX5258-5.8G-72CH-25-200-500-800mW-Switchable-FPV-Transmitter-Support-OSD-Configuring-p-1069.html>
- [44] Amazon, “Quickbuying TX801 5.8G 72CH.” Accessed: Dec. 29, 2024. [Online]. Available: <https://www.amazon.com/-/es/Quickbuying-TX801-0-01mW-Transmisior-conmutado/dp/B072MCX86P>
- [45] Eachine, “Eachine ROTG01 Pro UVC OTG 5.8G 150CH Full Channel FPV Receiver W/Audio.” Accessed: Dec. 29, 2024. [Online]. Available: <https://tinyurl.com/364pmav7>
- [46] Racer, “Boscam FPV 5.8G 8CH 200mW AV Wireless Receiver RC305.” Accessed: Dec. 29, 2024. [Online]. Available: <https://www.racer.lt/item/boscam-fpv-5-8g-8ch-200mw-av-wireless-receiver-rc305-6728.html>
- [47] C. Justo de Frías, “Visión artificial aplicada en la identificación de objetos y su parametrización geométrica,” 2019, *Leganés*.
- [48] W. Ramírez and C. Ramírez, *Programación de Inteligencia Artificial*. Madrid: Rama Editorial, 2023.
- [49] Python Software Foundation, “3.13.0 Documentation.” Accessed: Oct. 26, 2024. [Online]. Available: <https://docs.python.org/3/>
- [50] MDN contributors, “JavaScript documentation — DevDocs.” Accessed: Oct. 26, 2024. [Online]. Available: <https://devdocs.io/javascript/>
- [51] The MathWorks, “MATLAB Documentation - MathWorks América Latina.” Accessed: Oct. 26, 2024. [Online]. Available: <https://la.mathworks.com/help/matlab/>

- [52] M. Ramos, *Entornos de desarrollo*. Madrid: Editorial Garceta, 2014.
- [53] Mactic, *Uso de los materiales didácticos mediante Jupyter*. Lima, 2019.
- [54] Macti, *Uso de los materiales didácticos mediante Google Colab*. Lima, 2018.
- [55] Visual Studio Code, “Visual Studio Code - Code Editing. Redefined.” Accessed: Nov. 30, 2024. [Online]. Available: <https://n9.cl/fma48>
- [56] Á. Nolla de Celis, *Geometría Proyectiva y Visión artificial*. Madrid: RA-MA Editorial, 2021.
- [57] Centro de formación técnica para la industria, “Visión Artificial: todo lo que necesitas saber.” Accessed: Jan. 02, 2025. [Online]. Available: <https://www.cursosaula21.com/que-es-la-vision-artificial/>
- [58] J. Ordieres-Meré *et al.*, *Técnicas y algoritmos básicos de visión artificial Recurso electrónico - En línea*. 2006.
- [59] Data Machine Learning Visualization, “Reconocimiento de objetos con Deep Learning - Damavis Blog.” Accessed: Jan. 03, 2025. [Online]. Available: <https://blog.damavis.com/reconocimiento-de-objetos-con-deep-learning/>
- [60] W. Ramírez and C. Ramírez, “Programación de Inteligencia Artificial. Curso Práctico,” 2023, *Ra-Ma*.
- [61] J. Giménez, “APLICACIONES DE GPU_s EN VISIÓN E INTELIGENCIA ARTIFICIAL PARA EL RECONOCIMIENTO DE FORMAS.” Accessed: Jan. 15, 2025. [Online]. Available: <https://riunet.upv.es/handle/10251/174441>
- [62] Ultralytics, “YOLOv8 - Ultralytics YOLO Documentos.” Accessed: Oct. 26, 2024. [Online]. Available: <https://docs.ultralytics.com/es/models/yolov8/>
- [63] Roboflow, “MobileNet SSD v2 Object Detection Model: What is, How to Use.” Accessed: Oct. 26, 2024. [Online]. Available: <https://roboflow.com/model/mobilenet-ssd-v2>

- [64] R. Girshick, “rbgirshick (Ross Girshick) · GitHub.” Accessed: Oct. 26, 2024. [Online]. Available: <https://github.com/rbgirshick/>
- [65] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” Jun. 2015, [Online]. Available: <http://arxiv.org/abs/1506.02640>
- [66] F. Florez, *Seguimiento de objetos y análisis de movimiento*. RUA, 2016.
- [67] Y. Zhang *et al.*, “ByteTrack: Multi-object Tracking by Associating Every Detection Box,” 2022, pp. 1–21. doi: 10.1007/978-3-031-20047-2_1.
- [68] AWS, “¿Qué es el reconocimiento óptico de caracteres (OCR)?” Accessed: Dec. 31, 2024. [Online]. Available: <https://aws.amazon.com/es/what-is/ocr/>
- [69] Google Cloud, “OCR con Google AI | Google Cloud.” Accessed: Dec. 31, 2024. [Online]. Available: <https://cloud.google.com/use-cases/ocr?hl=es-419>
- [70] Azure AI, “Extract text from images.” Accessed: Dec. 31, 2024. [Online]. Available: <https://portal.vision.cognitive.azure.com/demo/extract-text-from-images>
- [71] OCR space, “Free OCR API.” Accessed: Dec. 31, 2024. [Online]. Available: <https://ocr.space/OCRAPI>
- [72] Kaggle, “Your Home for Data Science.” Accessed: Dec. 28, 2024. [Online]. Available: <https://www.kaggle.com/>
- [73] Microsoft Azure, “Servicios de informática en la nube | Microsoft Azure.” Accessed: Oct. 26, 2024. [Online]. Available: <https://azure.microsoft.com/es-es>
- [74] Google Cloud, “Servicios de computación en la nube | Google Cloud | Google Cloud.” Accessed: Oct. 26, 2024. [Online]. Available: https://cloud.google.com/?hl=es_419
- [75] Red Hat, “¿Qué son los servicios de nube?” Accessed: Jan. 01, 2025. [Online]. Available: <https://www.redhat.com/es/topics/cloud-computing/what-are-cloud-services>

- [76] L. Sanchez, “AWS vs Azure vs Google Cloud: el dilema de elegir la mejor plataforma cloud para tu empresa.” Accessed: Jan. 01, 2025. [Online]. Available: https://www.initiumsoft.com/blog_initium/aws-azure-google-cloud/
- [77] B. Chavarria and E. Gudiño, *Implementación de un servidor web*. UPS, 2017.
- [78] Linuxcareers, “Apache vs IIS vs Nginx: An In-depth Comparison of Web Servers.” Accessed: Jan. 01, 2025. [Online]. Available: <https://www.linuxcareers.com/resources/blog/2023/07/apache-vs-iis-vs-nginx-an-in-depth-comparison-of-web-servers/>
- [79] SQLite, “SQLite Documentation.” Accessed: Oct. 27, 2024. [Online]. Available: <https://www.sqlite.org/docs.html>
- [80] MySQL, “MySQL :: MySQL 8.4 Reference Manual.” Accessed: Oct. 27, 2024. [Online]. Available: <https://dev.mysql.com/doc/refman/8.4/en/>
- [81] MongoDB, “MongoDB Atlas: Cloud Document Database | MongoDB.” Accessed: Oct. 27, 2024. [Online]. Available: https://www.mongodb.com/es/lp/cloud/atlas/try4?utm_source=google&utm_campaign
- [82] Amazon, “HP Pavilion 15.6.” Accessed: Dec. 13, 2024. [Online]. Available: <https://www.amazon.com/-/es/HP-Pavilion-i7-8750-GeForce-memoria/dp/B07FN9BLM7>
- [83] Amazon, “Dell G5 15.6.” Accessed: Dec. 13, 2024. [Online]. Available: <https://www.amazon.com/-/es/Dell-G5-i7-10750H-Backlit-Nahimic/dp/B08JGZHDJC>
- [84] Amazon, “Lenovo Laptop IdeaPad 330 81DE00L0US.” Accessed: Dec. 13, 2024. [Online]. Available: <https://www.amazon.com/-/es/Lenovo-IdeaPad-81DE00L0US-generaci%C3%B3n-Windows/dp/B07FN595NS>
- [85] Prometec, “Guía FPV para multirrotores – Prometec.” Accessed: Oct. 22, 2024. [Online]. Available: <https://www.prometec.net/guia-fpv-para-multirrotores/>
- [86] D. Laura, “SISTEMA DE TELEDETECCIÓN AÉREA PARA CONTROL DE AGENTES PATÓGENOS Y ENFERMEDADES EN CULTIVOS DE BRÓCOLI CON

EL USO DE VISIÓN ARTIFICIAL,” UNIVERSIDAD TÉCNICA DE AMBATO, Ambato, 2023.

[87] J. Culcay and D. Garzón, “CONSTRUCCIÓN DE UN VEHÍCULO AÉREO NO TRIPULADO (UAV) ROTATIVO UTILIZANDO SOFTWARE LIBRE PARA APLICACIONES DE MONITORIZACIÓN DE LA UNIVERSIDAD TÉCNICA DE AMBATO,” Ambato, 2023.

[88] J. Hirvinen, “ArduPilot Documentation.” Accessed: Nov. 15, 2024. [Online]. Available: <https://ardupilot.org/ardupilot/index.html#>

[89] T.-Y. Lin *et al.*, “LNCS 8693 - Microsoft COCO: Common Objects in Context,” 2014.

[90] R. P. Abao, M. R. J. E. Estuar, and P. A. R. Abu, “Physical Distancing and Mask Wearing Behavior Dataset Generator from CCTV Footages Using YOLOv8,” 2023, pp. 295–304. doi: 10.1007/978-3-031-43129-6_29.

[91] Dirección General de aviación Civil, “Normas Generales que Regulan la Operación de Aeronaves Pilotadas a Distancia (RPAS),” Quito, Mar. 2024.

[92] NVIDIA, “LA POTENCIA DE GEFORCE GTX 1060,” 2024. Accessed: Jan. 04, 2025. [Online]. Available: <https://www.nvidia.com/es-la/geforce/products/10series/geforce-gtx-1060/>

[93] SpeedyBee, “SpeedyBee F405 V3 User Manual V1.0,” 2023.

ANEXOS

Anexo A. Resolución de las normas generales que regulan la operación de Aeronaves pilotadas a distancia (RPAS) [91].



Resolución Nro. DGAC-DGAC-2024-0019-R

Quito, D.M., 06 de marzo de 2024

detalla los inconvenientes de la aplicación de la RDAC 102 y se resuelve recomendar a la máxima autoridad de la Dirección General de Aviación Civil **DEROGAR** la RDAC 102 Sistemas de Aeronaves No Tripuladas (UAS) y emitir normas generales que regulen la operación de aeronaves pilotadas a distancia (RPAS);

De acuerdo al marco normativo expuesto en los considerandos precedentes en uso de las facultades legales, en especial, el artículo 6, numeral 3, literal a) de la Codificación de la Ley de Aviación Civil,

RESUELVE:

Artículo 1.- APROBAR las "Normas Generales que Regulan la Operación de Aeronaves Pilotadas a Distancia (RPAS)", conforme se detallan a continuación:

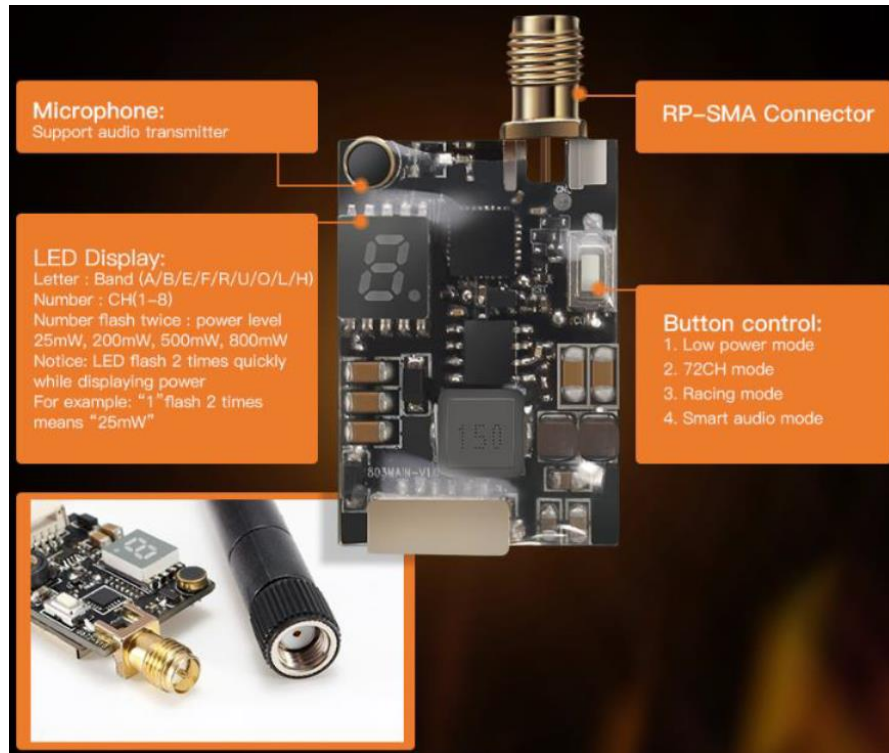
CAPÍTULO A) REGLAS DE VUELO Y OPERACIÓN GENERAL

Artículo 2.- Aplicabilidad. Este capítulo establece las reglas de vuelo para la operación de aeronaves pilotadas a distancia (RPAs), la operación de las RPAs se ajustará a lo establecido en el presente capítulo, excepto que la Dirección General de Aviación Civil (DGAC) autorice expresamente de otra forma conforme al capítulo B.

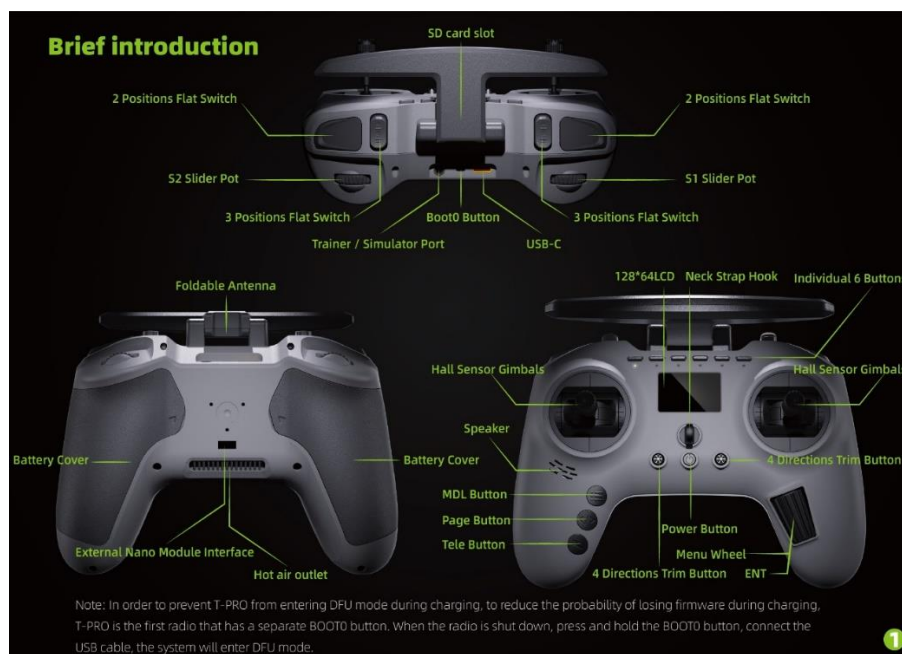
Artículo 3.- Operación negligente o temeraria de aeronaves.

1. La operación de las RPAs debe realizarse de tal forma que no ponga en peligro la seguridad de las operaciones aéreas, de las personas en la superficie, de sus bienes y de la fauna silvestre.
2. El operador de una RPA suspenderá inmediatamente el vuelo, en cualquier momento en que la seguridad de las personas, de los bienes y de la fauna esté en peligro como resultado de esta operación, o cuando no pueda cumplir con todos los requisitos del presente capítulo.
3. Las operaciones interrumpidas según el literal (b) no se reanudarán en tanto las condiciones que generan el peligro estén presentes.

Anexo B. Descripción de los componentes fundamentales que constituyen el transmisor de video Eachine-TX801 [44].



Anexo C. Radio control Jumper T-Pro ELRS [31].



Anexo D. Especificaciones técnicas de la tarjeta gráfica GEFORCE GTX 1060 6GB [92].

Especificaciones del motor de la GPU:		Compatibilidad de visualización:	
	6 GB	Resolución digital máxima2	7680x4320@60Hz
NVIDIA CUDA® Cores	1280	Conectores de visualización estándares	DP 1.4, HDMI 2.0b, Dual Link-DVI
Base Clock (MHz)	1506	Varios monitores	Si
Boost Clock (MHz)	1708	HDCP	2.2
Memory Specs:		Medidas de la tarjeta de gráficos:	
Memory Speed	8 Gbps	Altura	4.378"
Standard Memory Config	6 GB GDDR5	Largo	9.823"
Memory Interface Width	192-bit	Ancho	2-Slot
Memory Bandwidth (GB/sec)	192	Especificaciones de energía y de temperatura:	
Compatibilidad de tecnología:		Temperatura de la GPU máxima (en °C)	94
Proyección múltiple	Si	Energía de la tarjeta de gráficos (W)	120 W
<u>Compatible con la realidad virtual</u>	Si	Energía para el sistema recomendada4 (W)	400 W
Proyecto Ansel	Si	Conectores de energía auxiliares	6-Pin
Compatibilidad con NVIDIA SLI	No		
<u>Compatibilidad con NVIDIA G-Sync™</u>	Si		
Compatible con NVIDIA GameStream™-Ready	Si		
NVIDIA GPU Boost	3.0		
Microsoft DirectX	12 API with feature level 12_1		

Anexo E. Especificaciones técnicas de la placa de control de vuelo y la placa ESC encargada de regular las velocidades de los motores.

En la Figura E1 se indica las partes de la placa controladora SpeedyBee F405 V3.

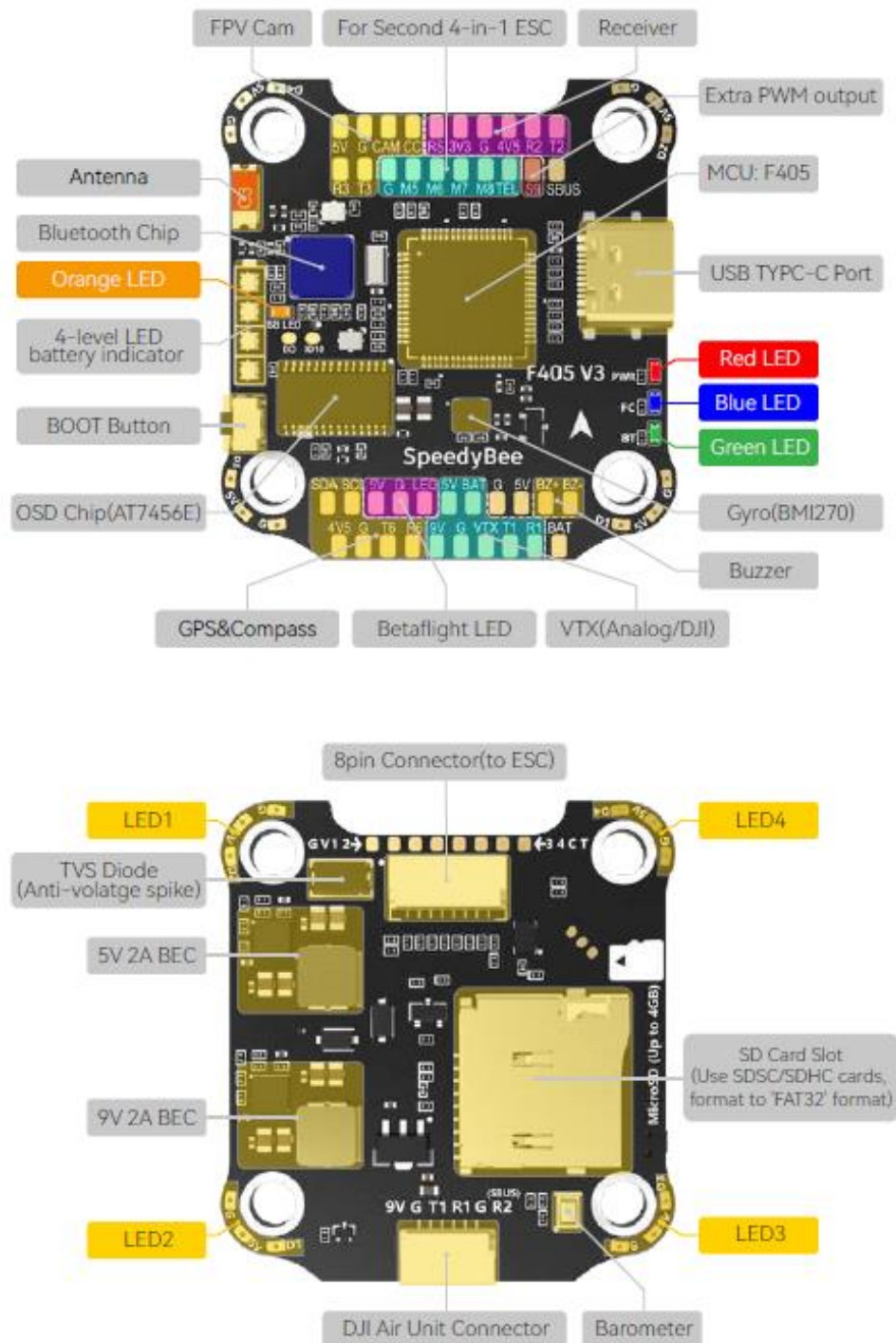


Figura E1. Placa controladora SpeedyBee F405 V3 [93].

En la Figura E2 se indica las especificaciones técnicas de la placa controladora SpeedyBee F405 V3.

MCU	STM32F405
IMU(Gyro)	BMI270
USB Port Type	Type-C
Barometer	Built-in
OSD Chip	AT7456E chip
BLE Bluetooth	Supported. Used to connect with the SpeedyBee App for flight controller and ESC parameter configuration. Please make sure the MSP switch on UART 4 is turned on and set to a baud rate of 115200, otherwise Bluetooth functionality will not be available.
WiFi	Not supported
DJI Air Unit Connection Way	Two ways supported: 6-pin connector or direct soldering.
6-pin DJI Air Unit Plug	Supported. Completely compatible with DJI O3/RunCam Link/Caddx Vista/DJI Air Unit V1, no wire is needed to be changed.
Blackbox MicroSD Card Slot	*Betaflight firmware requires the type of the microSD card to be either Standard (SDSC) or High capacity (SDHC), so extended capacity cards (SDXC) are not supported (Many high-speed U3 cards are SDXC). Also the card MUST be formatted with the FAT16 or FAT32 (recommended) filesystems. So, you could use any SD card less than 32GB, but the Betaflight can only recognize 4GB maximum. We suggest you use this 3rd party formatting tool and choose 'Overwrite format' then format your card. Also check out here for the recommended SD cards or buy the tested cards from our store.
Current Sensor Input	Supported. For SpeedyBee BLS 50A ESC, please set scale = 386 and Offset = 0.
Power Input	3-8S LiPo. The flight controller is powered through the G, V wires of the 8pin cable or G, V pads from the bottom side of the flight controller.
5V Output	9 groups of 5V output, four +5V pads and 1 BZ+ pad(used for Buzzer) on front side, and 4x LED 5V pads. The total current load is 2A.
9V Output	2 groups of 9V output, one +9V pad on front side and other included in a connector on bottom side. The total current load is 2A.
3.3V Output	Supported. Designed for 3.3V-input receivers. Up to 500mA current load.
4.5V Output	Supported. Designed for receiver and GPS module even when the FC is powered through the USB port. Up to 1A current load.
ESC Signal	M1 - M4 on bottom side and M5-M8 on front side.
UART	6 sets(UART1, UART2, UART3, UART4(Dedicated for Bluetooth connection)), UART5(Dedicated for ESC telemetry),UART6
ESC Telemetry	UART R5(UART5)
I2C	Supported. SDA & SCL pads on front side. Used for magnetometer, sonar, etc.
Traditional Betaflight LED Pad	Supported. 5V, G and LED pads on bottom of the front side. Used for WS2812 LED controlled by Betaflight firmware.
Buzzer	BZ+ and BZ- pad used for 5V Buzzer
BOOT Button	Supported. [A]. Press and hold BOOT button and power the FC on at the same time will force the FC to enter DFU mode, this is for firmware flashing when the FC gets bricked. [B]. When the FC is powered on and in standby mode, the BOOT button can be used to controller the LED strips connected to LED1-LED4 connectors on the bottom side. By default, short-press the BOOT button to cycle the LED displaying mode. Long-press the BOOT button to switch between SpeedyBee-LED mode and BF-LED mode. Under BF-LED mode, all the LED1-LED4 strips will be controlled by Betaflight firmware.
RSSI Input	Supported. Named as RS on the front side.
Smart Port / F.Port	Not supported
Supported Flight Controller Firmware	BetaFlight(Default), INAV (INAV firmware can only use Multishot (recommended) and OneShot125. Please note that DShot is not supported.)
Firmware Target Name	SPEEDYBEEF405V3
Mounting	30.5 x 30.5mm(4mm hole diameter)
Dimension	41.6(L) x 39.4(W) x 7.8(H)mm
Weight	9.6g

Figura E2. Especificaciones técnicas de la placa controladora SpeedyBee F405 V3 [93].

En la Figura E3 se indica las partes de la placa ESC encargada de regular las velocidades de los motores.

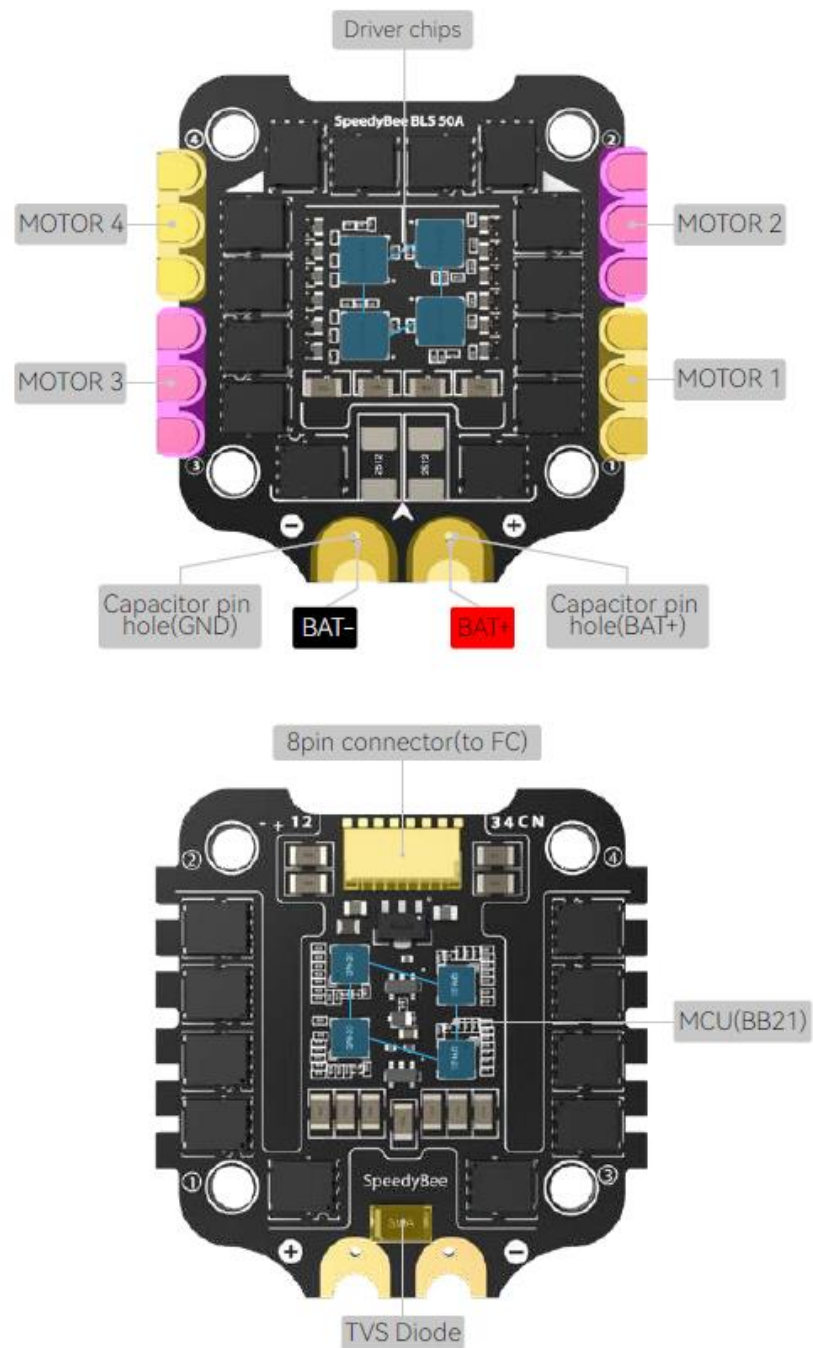


Figura E3. Placa ESC encargada de regular las velocidades de los motores [93].

En la Figura E4 se indica las especificaciones técnicas de la placa ESC encargada de regular las velocidades de los motores.

Firmware	BLHeli_S JH50
Wireless Configuration	Full Configuration Supported in the SpeedyBee app
PC Configurator Download Link	https://esc-configurator.com/
Continuous Current	50A * 4
Burst Current	55A(5S)
TVS Protective diode	Yes
External Capacitor	1000uF Low ESR Capacitor(In the package)
ESC Protocol	DSHOT300/600
Power Input	3-6S LiPo
Power Output	VBAT
Current Sensor	Support (Scale=386 Offset=0)
Mounting	30.5 x 30.5mm(4mm hole diameter)
Dimension	45.6(L) * 44(W) * 6.1mm(H)
Weight	13.8g

Figura E4. Especificaciones técnicas de la placa ESC encargada de regular las velocidades de los motores [93].

Anexo F. Archivo de codificación "det_vehiculos.py" en ordenador portátil.

```
import cv2
import numpy as np
from ultralytics import YOLO
from sort.sort import Sort
import os
import datetime
import time
import threading
import filtrado

import logging
# Opción para mostrar solo errores
logging.getLogger('ultralytics').setLevel(logging.ERROR)

# Cargar el modelo YOLOv8m para la detección de vehículos
vehicle_model = YOLO('yolov8m.pt')

# Abrir la captura de video desde el modulo
cap = cv2.VideoCapture(1)
# Ajustar el tamaño del buffer para reducir el retraso
cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

if not cap.isOpened():
    print("Error: No se puede abrir el video.")
    exit()

# Obtener el FPS del video
fps = cap.get(cv2.CAP_PROP_FPS) or 30 # Valor por defecto

frame_skip = 1 # Procesar cada frame
frame_count = 0
confidence_threshold = 0.7 # Umbral de confianza mínima para detectar
automovil

# Inicializar el rastreador SORT con parámetros ajustados
max_age = 30 # Número máximo de frames para mantener un objeto sin
detección
min_hits = 1 # Número mínimo de detecciones antes de asignar un ID
iou_threshold = 0.7 # Umbral de IOU para la asociación de objetos

tracker = Sort(max_age=max_age, min_hits=min_hits,
iou_threshold=iou_threshold)

# Mapeo de IDs para asegurar secuencia
id_mapping = {}
```

```

next_id = 1

# Crear una carpeta para guardar las capturas
output_folder = "capturas"
os.makedirs(output_folder, exist_ok=True)

# Conjunto para rastrear los IDs de los vehículos que ya han sido
capturados
captured_ids = {}

# Definir la región de interés (ROI) para la detección del vehículo
roi_x, roi_y, roi_w, roi_h = 300, 50, 20, 250
x, y, w, h = 60, 0, 520, 430

def detectar_placas_async(capture_filename):
    filtrado.detectar_placas(capture_filename)

while True:
    start_frame_time = time.time() # Marcar el tiempo de inicio
    ret, frame = cap.read()
    if not ret:
        print("Fin del video o error al recibir el frame.")
        break

    frame_count += 1
    if frame_count % frame_skip != 0:
        continue

    frame = cv2.resize(frame, (640, 480)) # Redimensionar

    roi_frame = frame[y:y + h, x:x + w]

    # Realizar la detección de vehículos en todo el fotograma
    vehicle_results = vehicle_model(roi_frame)

    detections = []
    for result in vehicle_results:
        boxes = result.bboxes
        for box in boxes:
            class_id = int(box.cls)
            confidence = box.conf[0]
            # IDs de vehículos en COCO dataset y umbral de confianza
            if class_id in [2, 5, 7] and confidence >
confidence_threshold:
                # Mover a CPU y convertir a NumPy
                x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()

```

```

        x1 += x
        y1 += y
        x2 += x
        y2 += y
        detections.append([x1, y1, x2, y2,
confidence.cpu().numpy()])

    # Actualizar las posiciones de los vehículos solo si hay
detecciones
    if len(detections) > 0:
        tracked_objects = tracker.update(np.array(detections))
    else:
        tracked_objects = []

for obj in tracked_objects:
    x1, y1, x2, y2, obj_id = obj
    p1 = (int(x1), int(y1))
    p2 = (int(x2), int(y2))

    # Asignar un nuevo ID secuencial si es necesario
    if obj_id not in id_mapping:
        id_mapping[obj_id] = next_id
        next_id += 1

    stable_id = id_mapping[obj_id]

    # Dibujar la caja delimitadora y el ID en el fotograma
    cv2.rectangle(frame, p1, p2, (0, 255, 0), 1, 1)
    label_with_confidence = f"ID {stable_id}: {confidence:.2f}"

    # Calcular la posición para el texto en la parte inferior
izquierda
    text_position = (p1[0], p2[1] - 250) # Ajusta el valor -15
según sea necesario

    cv2.putText(frame, label_with_confidence, text_position,
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)

    # Verificar si el vehículo está intersectado en la ROI
    if (p1[0] < roi_x + roi_w and p2[0] > roi_x and
p1[1] < roi_y + roi_h and p2[1] > roi_y):
        current_time = time.time()
        if stable_id not in captured_ids:
            captured_ids[stable_id] = {"time": current_time,
"captured": False}

    # Verificar si han pasado al menos 3 segundos desde la
detección inicial

```

```

        elapsed_time = time.time() -
captured_ids[stable_id]["time"]
        if elapsed_time >= 3 and not
captured_ids[stable_id]["captured"]:
            # Guardar la captura solo si aún no ha sido capturada
            timestamp =
datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
            capture_filename = os.path.join(output_folder,
f"vehiculo_{timestamp}.jpg")
            cv2.imwrite(capture_filename, frame)
            print(f"Captura guardada: {capture_filename}")

            # Ejecutar la detección de placas en un hilo separado
            threading.Thread(target=detectar_placas_async,
args=(capture_filename,)).start()
            # Marcar que la captura ya se realizó para este
vehículo

            captured_ids[stable_id]["captured"] = True

# Dibujar la región de interés (ROI) en el fotograma
color = (255, 255, 255)
thickness = 1
cv2.rectangle(frame, (x, y), (x + w, y + h), color, thickness)

processing_time = time.time() - start_frame_time
wait_time = max(1, int((1 / fps - processing_time) * 1000))
cv2.imshow('Deteccion de Vehiculos', frame)

if cv2.waitKey(wait_time) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

```

Anexo G. Archivo de codificación "filtrado.py" en ordenador portátil.

```

import cv2
from ultralytics import YOLO
import os
import recortes_placas_ubi
import recortes_solo_ubi
import cloudinary
import cloudinary.uploader
import cloudinary.api

```

```

from my_timer import my_timer

def upload_image_to_cloudinary(image_path):

    # Configuración de Cloudinary
    cloudinary.config(
        cloud_name="cloud_name", # Reemplaza con cloud name
        api_key="api_key",        # Reemplaza con API key
        api_secret="api_secret"   # Reemplaza con API secret
    )

    try:
        # Subir la imagen
        response =
cloudinary.uploader.upload(image_path, folder="PLACAS")
        # Retornar la URL de la imagen
        return response['secure_url']
    except Exception as e:
        return f"Error al subir la imagen: {e}"

def upload_image_to_cloudinarynp(image_path):

    # Configuración de Cloudinary
    cloudinary.config(
        cloud_name="cloud_name", # Reemplaza con cloud name
        api_key="api_key",        # Reemplaza con API key
        api_secret="api_secret"   # Reemplaza con API secret
    )

    try:
        # Subir la imagen
        response =
cloudinary.uploader.upload(image_path, folder="NO_PLACAS")
        # Retornar la URL de la imagen
        return response['secure_url']
    except Exception as e:
        return f"Error al subir la imagen: {e}"

@my_timer
def detectar_placas(image_path):

    # Cargar el modelo YOLOv8
    model = YOLO('best80.pt')

    # Crear las carpetas de salida

```

```

output_folder_placas = "placas"
output_folder_no_placas = "no_placas"
output_folder_recortes = "recorte_placas"
output_folder_latitud = "latitud"
output_folder_longitud = "longitud"

# Leer la imagen
image = cv2.imread(image_path)
# Realizar la detección
results = model(image)
# Umbral mínimo de confianza
confidence_threshold = 0.7
# Bandera para verificar si se detectó alguna placa
placa_detectada = False
# Obtener el nombre de la imagen original
image_name = os.path.basename(image_path)

# Mostrar los resultados
for result in results:
    boxes = result.bboxes
    for box in boxes:
        confidence = box.conf[0]
        if confidence >= confidence_threshold:
            placa_detectada = True
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            class_id = box.cls[0]

            # Dibujar el rectángulo y la etiqueta en la imagen
            cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0),
2)

            label = f"Class: {int(class_id)}, Conf:
{confidence:.2f}"
            cv2.putText(image, label, (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

            # Recortar la zona detectada
            recorte = image[(y1+15):y2, x1:x2]
            # Crear la carpeta de salida para recortes si no existe
            if not os.path.exists(output_folder_recortes):
                os.makedirs(output_folder_recortes)

            # Ruta de salida para el recorte
            recorte_path = os.path.join(output_folder_recortes,
image_name)

            # Guardar el recorte
            cv2.imwrite(recorte_path, recorte)

```

```

if not os.path.exists(output_folder_latitud):
    os.makedirs(output_folder_latitud)

zone_x, zone_y, zone_w, zone_h = 80, 435, 220, 40

# Guardar la zona específica del ROI para LATITUD
roi_lat = image[zone_y:zone_y + zone_h, zone_x:zone_x + zone_w]

# Ruta de salida para el recorte
recorte_lat = os.path.join(output_folder_latitud, image_name)

# Guardar el recorte
cv2.imwrite(recorte_lat, roi_lat)
#print(f"Recorte latitud guardado en: {recorte_lat}")

if not os.path.exists(output_folder_longitud):
    os.makedirs(output_folder_longitud)

zone_x2, zone_y2, zone_w2, zone_h2 = 330, 435, 250, 40
# Guardar la zona específica del ROI para LONGITUD
roi_lon = image[zone_y2:zone_y2 + zone_h2, zone_x2:zone_x2 +
zone_w2]

# Ruta de salida para el recorte
recorte_lon = os.path.join(output_folder_longitud, image_name)

# Guardar el recorte
cv2.imwrite(recorte_lon, roi_lon)

# Crear las carpetas de salida si no existen
if not os.path.exists(output_folder_placas):
    os.makedirs(output_folder_placas)
if not os.path.exists(output_folder_no_placas):
    os.makedirs(output_folder_no_placas)

# Ruta de salida para la imagen con detecciones
if placa_detectada:
    output_path = os.path.join(output_folder_placas, image_name)

    # Llamar a la función y mostrar el resultado
    url = upload_image_to_cloudinary(image_path)
    url= str(url)
    # Cambia estas rutas a las imágenes específicas
    images_info = [recorte_path, recorte_lat,
recorte_lon]
    recortes_placas_ubi.main(images_info,url)

```

```

else:
    output_path = os.path.join(output_folder_no_placas, image_name)

    # Llamar a la función y mostrar el resultado
    url = upload_image_to_cloudinarynp(image_path)
    url= str(url)
    images_info = [recorte_lat, recorte_lon]
    # Cambia estas rutas a las imágenes específicas
    recortes_solo_ubi.main(images_info,url)

# Guardar la imagen con las detecciones
cv2.imwrite(output_path, image)

# Mensaje adicional si no se detectó ninguna placa
if not placa_detectada:
    print(f"No se detectó ninguna placa en la imagen:
{image_path}")

```

Anexo H. Archivo de codificación "recortes_placas_ubi_py" en ordenador portátil

```

import os
from google.cloud import vision
import re
import mysql.connector

text1,text2,text3="", "", ""
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'Ubicacion del archivo
vision_key.json'

def detect_text(path):
    client = vision.ImageAnnotatorClient()
    with open(path, "rb") as image_file:
        content = image_file.read()
    image = vision.Image(content=content)
    response = client.text_detection(image=image)
    texts = response.text_annotations

    if response.error.message:
        raise Exception(
            "{}\nFor more info on error messages, check: "

```

```

        "https://cloud.google.com/apis/design/errors".format(response.error.message)
    )

    if texts:
        return texts[0].description
    else:
        return ""

def main(image_paths,result):
    for i, image_path in enumerate(image_paths):
        if os.path.isfile(image_path):
            text = detect_text(image_path)
            var_name = f"text{i+1}"
            globals()[var_name] = text
        else:
            print(f"File {image_path} does not exist.")

    # Imprimir las variables creadas
    if 'text1' in globals():
        print(str(text1))
    if 'text2' in globals():
        print(str(text2))
    if 'text3' in globals():
        print(str(text3))

    procesar_placa_y_crear_enlace(text1, text2, text3, result)

def procesar_placa_y_crear_enlace(placa, latitud, longitud, result):

    # Eliminar caracteres inválidos y convertir a mayúsculas
    placa = re.sub(r'^A-Za-z0-9', '', placa).upper()

    # Validar que tiene al menos 3 letras y números combinados
    if len(placa) < 3:
        return "Inválida: la placa debe tener al menos 3 letras seguidas de números."

    # Diccionarios para conversión entre letras y números
    numeros_a_letras = {'0': 'O', '1': 'I', '2': 'Z', '3': 'J', '4': 'A',
                        '5': 'S', '6': 'G', '7': 'T', '8': 'B', '9': 'P'}

    # Validar y ajustar los tres primeros caracteres como letras
    letras = placa[:3]

```

```

    # Corrige números mal ubicados
    letras = ''.join([numeros_a_letras.get(char, char) if char in
numeros_a_letras else char for char in letras])
    if not re.match(r"^[A-Z]{3}$", letras):
        return "Inválida: los primeros tres caracteres deben ser
letras."

    # Diccionarios para conversión entre letras y números
    letras_a_numeros = {'O': '0', 'I': '1', 'Z': '2', 'E': '3', 'A':
'4',
                        'S': '5', 'G': '6', 'T': '7', 'B': '8', 'P':
'9',
                        'Q': '0', 'L': '1', 'J': '1', 'U': '0', 'N':
'2',
                        'D': '0'}

    # Validar y ajustar los caracteres restantes como números
    numeros = placa[3:]
    # Corrige letras mal ubicadas
    numeros = ''.join([letras_a_numeros.get(char, char) if char in
letras_a_numeros else char for char in numeros])

    # Ajustar los números a 4 dígitos
    numeros = numeros.zfill(4)

    # Formar la placa final
    placa_formateada = f"{letras}{numeros}"

    # Corregir formato de latitud
    latitud = re.sub(r'^0-9.-', '', latitud)
    if latitud.count('-') > 1 or (latitud.count('-') == 1 and
latitud.index('-') != 0):
        # Mueve el signo negativo al principio si existe
        latitud = '-' + latitud.replace('-', '')

    latitud = float(latitud)

    # Corregir formato de longitud
    longitud = re.sub(r'^0-9.-', '', longitud)
    if longitud.count('-') > 1 or (longitud.count('-') == 1 and
longitud.index('-') != 0):
        # Mueve el signo negativo al principio si existe
        longitud = '-' + longitud.replace('-', '')

    longitud = float(longitud)

    # Crear el enlace de Google Maps
    url =
f"https://www.google.com/maps/search/?api=1&query={latitud},{longitud}"

```

```

    insertar_datos(str(placa_formateada), str(url), result)

    return f"Placa válida: {placa_formateada}, Enlace de Google Maps: {url}"

def insertar_datos(placa, ubicacion,result):
    # Configuración de la conexión a la base de datos
    config = {
        'user': 'user', # Reemplaza con usuario
        'password': 'password', # Reemplaza con contraseña
        'host': 'host', # Reemplaza con host
        'database': 'vehiculos', # Reemplaza con nombre de la base de
datos
        'port': 3306 # Puerto por defecto de MariaDB
    }

    try:
        # Establecer la conexión
        conn = mysql.connector.connect(**config)
        cursor = conn.cursor()

        # Consulta SQL para insertar los datos
        query = "INSERT INTO eventosReconocimiento (placa, ubicacion,
foto) VALUES (%s, %s, %s)"
        values = (placa, ubicacion,result)

        # Ejecutar la consulta
        cursor.execute(query, values)

        # Confirmar la transacción
        conn.commit()

        print("Datos insertados correctamente.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        # Cerrar la conexión
        if conn.is_connected():
            cursor.close()
            conn.close()
            print("Conexión cerrada.")

```

Anexo I. Archivo de codificación "recortes_solo_ubi_py" en ordenador portátil.

```
import os
from google.cloud import vision
import re
import mysql.connector

text1,text2="", ""

os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'Ubicacion del archivo
vision_key.json'

def detect_text(path):
    client = vision.ImageAnnotatorClient()
    with open(path, "rb") as image_file:
        content = image_file.read()
    image = vision.Image(content=content)
    response = client.text_detection(image=image)
    texts = response.text_annotations

    if response.error.message:
        raise Exception(
            "{}\nFor more info on error messages, check: "
            "https://cloud.google.com/apis/design/errors".format(respon
se.error.message)
        )

    if texts:
        return texts[0].description
    else:
        return ""

def main(image_paths,result):
    for i, image_path in enumerate(image_paths):
        if os.path.isfile(image_path):
            text = detect_text(image_path)
            var_name = f"text{i+1}"
            globals()[var_name] = text
            print(f"Processing {image_path}")
        else:
            print(f"File {image_path} does not exist.")

    # Imprimir las variables creadas
    if 'text1' in globals():
        print(str(text1))
    if 'text2' in globals():
        print(str(text2))
```

```

    procesar_ubi_y_crear_enlace(text1, text2, result)

def procesar_ubi_y_crear_enlace(latitud, longitud,result):

    # Corregir formato de latitud
    latitud = re.sub(r'^0-9.-', '', latitud)
    if latitud.count('-') > 1 or (latitud.count('-') == 1 and
latitud.index('-') != 0):
        # Mueve el signo negativo al principio si existe
        latitud = '-' + latitud.replace('-', '')

    latitud = float(latitud)

    # Corregir formato de longitud
    longitud = re.sub(r'^0-9.-', '', longitud)
    if longitud.count('-') > 1 or (longitud.count('-') == 1 and
longitud.index('-') != 0):
        # Mueve el signo negativo al principio si existe
        longitud = '-' + longitud.replace('-', '')

    longitud = float(longitud)

    # Crear el enlace de Google Maps
    url =
f"https://www.google.com/maps/search/?api=1&query={latitud},{longitud}"

    # Imprimir el enlace generado
    insertar_datos( str(url),result)

def insertar_datos(ubicacion,result):
    # Configuración de la conexión a la base de datos
    config = {
        'user': 'user', # Reemplaza con usuario
        'password': 'password', # Reemplaza con contraseña
        'host': 'host', # Reemplaza con host
        'database': 'vehiculos', # Reemplaza con nombre de la base de
datos
        'port': 3306 # Puerto por defecto de MariaDB
    }

    try:
        # Establecer la conexión
        conn = mysql.connector.connect(**config)
        cursor = conn.cursor()

```

```

        # Consulta SQL para insertar los datos
        query = "INSERT INTO eventosSinReconocimiento (ubicacion,foto)
VALUES (%s, %s)"
        values = (ubicacion,result)

        # Ejecutar la consulta
        cursor.execute(query, values)

        # Confirmar la transacción
        conn.commit()

        print("Datos insertados correctamente.")

except mysql.connector.Error as err:
    print(f"Error: {err}")

finally:
    # Cerrar la conexión
    if conn.is_connected():
        cursor.close()
        conn.close()
        print("Conexión cerrada.")

```

Anexo J. Registro de evento en la base de datos, notificación en el grupo de Telegram de los usuarios y correo electrónico del propietario.

La Figura J1 muestra el registro de un evento de reconocimiento almacenado en la base de datos.

	id	fecha	placa	ubicacion	foto
☐ Editar ☐ Copiar ☐ Borrar	1579	2025-01-18 17:28:20	TBF8484	https://www.google.com api=1&query=-1...	https://res.cloudinary.com/dk6

Figura J1. Registro de un evento de reconocimiento en la base de datos.

La Figura J2 presenta la recepción del mensaje en el grupo de Telegram destinado a los usuarios, acompañado de la notificación enviada al correo electrónico personal del propietario.

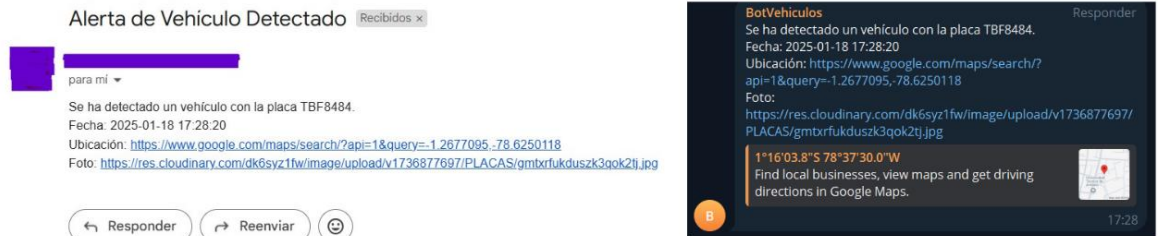


Figura J2. Recepción del mensaje en el grupo de Telegram y notificación enviada al correo electrónico personal del propietario.

Anexo K. Codificación en Python del archivo envioATelegram.py.

```
# Importar el conector de MariaDB para interactuar con la base de datos
import mariadb
# Importar asyncio para manejar operaciones asíncronas
import asyncio
# Importar smtplib para el envío de correos electrónicos
import smtplib
# Importar la clase Bot de telegram para enviar mensajes
from telegram import Bot
# Importar clases para manejar correos electrónicos con formato MIME
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
# Importar os para manejar archivos del sistema
import os

# Configuración del bot de Telegram y la base de datos
TELEGRAM_TOKEN = 'TOKEN DEL BOT DE TELEGRAM'
CHAT_ID = 'ID DEL CHAT DE TELEGRAM'
bot = Bot(token=TELEGRAM_TOKEN)

# Configuración del servidor de correo
EMAIL_USER = 'USUARIO DE CORREO'
EMAIL_PASS = 'contraseña generada por la app'
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = # Define el puerto SMTP del servidor de correo
```

```

# Definiendo el nombre donde se guarda la última ID procesada
ID_FILE = 'ultima_id.txt'

# Función para leer la última ID procesada desde un archivo

def leer_ultima_id():
    if os.path.exists(ID_FILE):
        with open(ID_FILE, 'r') as f:
            return int(f.read().strip())
    return 0

# Función para guardar la última ID procesada en un archivo

def guardar_ultima_id(id):
    with open(ID_FILE, 'w') as f:
        f.write(str(id))

# Función asíncrona para enviar un mensaje a Telegram
async def enviar_mensaje(mensaje):
    await bot.send_message(chat_id=CHAT_ID, text=mensaje)

# Función para enviar un correo electrónico

def enviar_correo(destinatario, mensaje):
    try:
        print(f"Enviando correo a {destinatario}...")
        msg = MIMEMultipart() # Crear un mensaje MIME
        msg['From'] = EMAIL_USER # Remitente
        msg['To'] = destinatario # Destinatario
        msg['Subject'] = 'Alerta de Vehículo Detectado' # Asunto del correo

        # Adjuntar el cuerpo del mensaje
        msg.attach(MIMEText(mensaje, 'plain'))

        # Establecer conexión con el servidor SMTP y enviar el correo
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls() # Iniciar la comunicación segura
            server.login(EMAIL_USER, EMAIL_PASS) # Autenticación
            server.sendmail(EMAIL_USER, destinatario,
                            msg.as_string()) # Enviar correo
        print("Correo enviado.")
    except Exception as e:
        print(f"Error enviando el correo: {e}")

```

```

# Función asíncrona para verificar la tabla de logs en la base de datos
y enviar alertas
async def verificar_logs():
    print("Conectando a la base de datos...")
    ultima_id = leer_ultima_id() # Leer la última ID procesada

    try:
        # Establecer conexión con la base de datos
        conexion = mariadb.connect(
            user='Usuario de la BD',
            password='contra de la BD',
            host='ID del servidor',
            database='nombre de la BD'
        )
        cursor = conexion.cursor()
        print("Conexión exitosa. Consultando la tabla logs...")

        # Ejecutar consulta para obtener registros nuevos
        cursor.execute(
            "SELECT id, mensaje, fecha, ubicacion, foto FROM logs WHERE
id > ? ORDER BY id ASC", (ultima_id,))
        resultados = cursor.fetchall()

        # Recorrer los resultados obtenidos
        for resultado in resultados:
            id_log, mensaje, fecha, ubicacion, foto = resultado
            print(
                f"Mensaje encontrado: {mensaje}, Fecha: {fecha},
Ubicación: {ubicacion}, Foto: {foto}")

            # Formatear el mensaje para el envío
            mensaje_formateado = (
                f"Se ha detectado un vehículo con la placa
{mensaje}.\n"
                f"Fecha: {fecha}\n"
                f"Ubicación: {ubicacion}\n"
                f"Foto: {foto}"
            )

            # Verificar si la placa detectada existe en la tabla
propietarios
            cursor.execute(
                "SELECT email FROM propietarios WHERE placa = %s",
(mensaje,))
            propietario = cursor.fetchone()

```

```

        if propietario:
            email_destino = propietario[0]
            # Enviar mensaje por Telegram
            await enviar_mensaje(mensaje_formateado)
            # Enviar correo electrónico
            enviar_correo(email_destino, mensaje_formateado)
        else:
            print(
                "No se encontró un propietario asociado a la placa
en la tabla propietarios.")

            # Actualizar la última ID procesada
            ultima_id = id_log
            guardar_ultima_id(ultima_id)

            if not resultados:
                print("No se encontraron registros nuevos en la tabla
logs.")

            cursor.close()
            conexion.close()
        except mariadb.Error as e:
            print(f"Error conectando a la base de datos: {e}")

# Punto de entrada del script
if __name__ == '__main__':
    print("Iniciando script...")
    # Ejecutar la función de verificación de logs
    asyncio.run(verificar_logs())
    print("Script finalizado.")

```

Anexo L. Codificación en Python del archivo envioATelegramAdmin.py.

```

# Importa la librería para conectar con MariaDB
import mariadb
# Importa la librería asyncio para operaciones asíncronas
import asyncio
# Importa la librería para enviar correos electrónicos
import smtplib
# Importa la clase Bot de la librería telegram para
# interactuar con la API de Telegram
from telegram import Bot
# Importa las clases necesarias para el manejo de correos

```

```

# electrónicos
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
# Importa la librería os para manejar archivos en el sistema
import os

# Configuración del bot de Telegram y la base de datos
TELEGRAM_TOKEN = 'TOKEN DEL BOT DE TELEGRAM'
CHAT_ID = 'ID DEL CHAT DE TELEGRAM'
bot = Bot(token=TELEGRAM_TOKEN)

# Configuración del servidor de correo
EMAIL_USER = 'USUARIO DE CORREO'
EMAIL_PASS = 'contraseña generada por la app'
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = # Define el puerto SMTP del servidor de correo
EMAIL_FIJO = 'Poner el correo fijo' # Correo fijo al que se enviarán
las alertas

# Definiendo el nombre donde se guarda la última ID procesada
ID_FILE_F = 'ultima_id_f.txt'

# Función para leer la última ID procesada

def leer_ultima_id_f():
    if os.path.exists(ID_FILE_F):
        with open(ID_FILE_F, 'r') as f:
            return int(f.read().strip())
    return 0

# Función para guardar la última ID procesada

def guardar_ultima_id_f(id):
    with open(ID_FILE_F, 'w') as f:
        f.write(str(id))

# Función asíncrona para enviar un mensaje a Telegram

async def enviar_mensaje(mensaje):
    await bot.send_message(chat_id=CHAT_ID, text=mensaje)

# Función para enviar un correo electrónico

def enviar_correo(destinatario, mensaje):

```

```

try:
    print(f"Enviando correo a {destinatario}...") # Mensaje de
    depuración
    msg = MIMEMultipart() # Crea un objeto de correo multipart
    msg['From'] = EMAIL_USER # Establece el remitente
    msg['To'] = destinatario # Establece el destinatario
    # Establece el asunto
    msg['Subject'] = 'Alerta de Evento sin Reconocimiento'

    # Agrega el mensaje en formato de texto plano
    msg.attach(MIMEText(mensaje, 'plain'))

    with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server: # Conecta
    con el servidor SMTP
        server.starttls() # Inicia la conexión segura
        # Inicia sesión en el servidor SMTP
        server.login(EMAIL_USER, EMAIL_PASS)
        server.sendmail(EMAIL_USER, destinatario,
                        msg.as_string()) # Envía el correo
        print("Correo enviado.") # Mensaje de confirmación
except Exception as e:
    print(f"Error enviando el correo: {e}") # Manejo de errores

# Función asíncrona para verificar la tabla eventosSinReconocimiento y
enviar alertas

async def verificar_eventos_sin_reconocimiento():
    print("Conectando a la base de datos...") # Mensaje de depuración
    ultima_id = leer_ultima_id_f() # Lee la última ID procesada

    try:
        # Conexión a la base de datos MariaDB
        conexion = mariadb.connect(
            user='Usuario de la BD', # Nombre de usuario de la base de
datos
            password='contra de la BD', # Contraseña del usuario de la
base de datos
            host='ID del servidor', # Dirección del servidor de la
base de datos
            database='nombre de la BD' # Nombre de la base de datos
        )
        cursor = conexion.cursor() # Crea un cursor para ejecutar
consultas
        # Mensaje de depuración
        print("Conexión exitosa. Consultando la tabla
eventosSinReconocimiento...")

```

```

        # Consulta para obtener registros nuevos en la tabla
eventosSinReconocimiento
        cursor.execute(
            "SELECT id, fecha, ubicacion, foto FROM
eventosSinReconocimiento WHERE id > ? ORDER BY id ASC", (ultima_id,))
        resultados = cursor.fetchall() # Obtiene los resultados de la
consulta

        # Procesa cada evento nuevo encontrado
        for resultado in resultados:
            id_evento, fecha, ubicacion, foto = resultado # Extrae los
datos del evento
            print(
                f"Evento encontrado: Fecha: {fecha}, Ubicación:
{ubicacion}, Foto: {foto}") # Mensaje de depuración

            # Formatear el mensaje para correo y Telegram
            mensaje_formateado = (
                f"Se ha detectado un evento sin reconocimiento:\n"
                f"Fecha: {fecha}\n"
                f"Ubicación: {ubicacion}\n"
                f"Foto: {foto}"
            )

            # Enviar mensaje a Telegram
            await enviar_mensaje(mensaje_formateado)

            # Enviar correo electrónico al correo fijo
            enviar_correo(EMAIL_FIJO, mensaje_formateado)

            # Actualizar la última ID procesada
            ultima_id = id_evento
            guardar_ultima_id_f(ultima_id)

        if not resultados:
            # Mensaje de depuración
            print(
                "No se encontraron registros nuevos en la tabla
eventosSinReconocimiento.")

        cursor.close() # Cierra el cursor
        conexion.close() # Cierra la conexión a la base de datos
    except mariadb.Error as e:
        print(f"Error conectando a la base de datos: {e}") # Manejo de
errores

# Punto de entrada del script
if __name__ == '__main__':

```

```

print("Iniciando script para eventos sin reconocimiento...")
# Ejecuta la función principal
asyncio.run(verificar_eventos_sin_reconocimiento())
print("Script finalizado.") # Mensaje de finalización

```

Anexo M. Codificación del archivo de la página principal pagina.php.

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Registro de Vehículos Robados</title>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
  <script>
    function showSection(sectionId) {
      const sections = document.querySelectorAll('main section');
      sections.forEach(section => {
        section.style.display = 'none';
      });
      document.getElementById(sectionId).style.display = 'block';
      resetForms();
    }

    function resetForms() {
      const forms = document.querySelectorAll('form');
      forms.forEach(form => {
        form.reset();
      });
      const successMessage = document.getElementById('success-
message');
      if (successMessage) {
        successMessage.innerHTML = '';
      }
    }

    document.addEventListener('DOMContentLoaded', () => {

```

```

        const urlParams = new
URLSearchParams(window.location.search);
        const section = urlParams.get('section');
        if (section) {
            showSection(section);
            document.body.className = section === 'inicio' ||
section === 'contactos' ? 'inicio-contactos' : 'otras-secciones';
        } else {
            showSection('inicio');
        }
    });
</script>
</head>
<!-- Cuerpo de la página con una clase inicial definida -->
<body class="inicio-contactos">
    <!-- Cabecera con el logo y la navegación principal -->
    <header>
        <div class="logo">
            <!-- Imagen del logo -->
            
        </div>
        <!-- Menú de navegación -->
        <nav>
            <ul>
                <!-- Enlaces de navegación que muestran diferentes
secciones -->
                <li><a href="javascript:void(0)"
onclick="showSection('inicio'); document.body.className='inicio-
contactos';">Inicio</a></li>
                <li><a href="javascript:void(0)"
onclick="showSection('reportes'); document.body.className='otras-
secciones';">Reportes</a></li>
                <li><a href="javascript:void(0)"
onclick="showSection('consultas'); document.body.className='otras-
secciones';">Consultas</a></li>
                <li><a href="javascript:void(0)"
onclick="showSection('contactos'); document.body.className='inicio-
contactos';">Contactos</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section id="inicio" class="text-inicio">
            <h1>¡Recupera Tu Tranquilidad! Encuentra o Reporta Tu
Vehículo Rápidamente</h1>
            <div style="display: flex; align-items: center;">
                <p>Bienvenido a AutoTracker, la plataforma diseñada
para ayudarte en los momentos más críticos. Si tu vehículo ha sido

```

robado, aquí puedes reportarlo de manera rápida y efectiva, ayudando a crear una red que facilite su recuperación. Además, si deseas consultar si tu vehículo ha sido localizado, nuestra herramienta te permitirá verificarlo al instante. Porque sabemos que tu seguridad y tranquilidad son lo más importante, estamos aquí para apoyarte en cada paso del proceso.</p>

```

</div>
</section>
<section id="reportes" class="otras-secciones">
  <h2>Reportar Vehículo Robado</h2>
  <div id="qr-code-section" class="qr-code-section">
    <h1>Únete aquí al grupo de Telegram mediante el código
QR:</h1>
    
  </div>
  <form id="report-form" method="POST"
action="guardar_datos_registro.php">
    <label for="nombre">Nombre del Propietario:</label>
    <input type="text" id="nombre" name="nombre"
placeholder="Ej: ANDRES" required pattern="[A-Z\s]+">
    <label for="telefono">Teléfono:</label>
    <input type="tel" id="telefono" name="telefono"
placeholder="Ej: 0983896753" required pattern="[0-9]{10}">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email"
placeholder="Ej: joseluisch@gmail.com" required pattern="[a-z0-9._%+-
]+@[a-z0-9.-]+\.[a-z]{2,4}">
    <label for="placa">Placa del Vehículo:</label>
    <input type="text" id="placa" name="placa"
placeholder="Ej: TBA9845" required pattern="[A-Z]{3}[0-9]{4}">
    <label for="marca">Marca del Vehículo:</label>
    <input type="text" id="marca" name="marca"
placeholder="Ej: KIA" required pattern="[A-Z]+">
    <label for="modelo">Modelo del Vehículo:</label>
    <input type="text" id="modelo" name="modelo"
placeholder="Ej: CERATO FORTE" required pattern="[A-Z0-9\s]+">
    <label for="anio">Año del Vehículo:</label>
    <input type="number" id="anio" name="anio"
placeholder="Ej: 2011" required min="1886" max="2024">
    <button type="submit">Registrar</button>
    <label id="success-message" class="success-
message"></label>
  </form>
</section>
<section id="consultas" class="otras-secciones">
```

```

        <h2>Consultar Vehículo</h2>
        <form method="POST" action="datos_consulta.php">
            <label for="placa-consulta">Placa del Vehículo:</label>
            <input type="text" id="placa-consulta" name="placa-
consulta" placeholder="Ej: TBA9845" required pattern="[A-Z]{3}[0-
9]{4}">
            <label for="marca-consulta">Marca del Vehículo:</label>
            <input type="text" id="marca-consulta" name="marca-
consulta" placeholder="Ej: KIA" required pattern="[A-Z]+">
            <label for="modelo-consulta">Modelo del
Vehículo:</label>
            <input type="text" id="modelo-consulta" name="modelo-
consulta" placeholder="Ej: CERATO FORTE" required pattern="[A-Z0-
9\s.]+>
            <label for="anio-consulta">Año del Vehículo:</label>
            <input type="number" id="anio-consulta" name="anio-
consulta" placeholder="Ej: 2011" required min="1886" max="2024">
            <button type="submit">Consultar</button>
        </form>
    </section>
    <section id="contactos" class="text-contactos">
        <h2>Contactos</h2>
        <p>Información de contacto para asistencia.</p>
        <p>Teléfono: 0328-24804</p>
        <p>Email: jchanahuano4850@uta.edu.ec</p>
        <p>Ubicación: FISEI - Talleres Tecnológicos</p>
    </section>
</main>
<!-- Pie de página con redes sociales -->
<footer>
    <h3>REDES SOCIALES</h3>
    <div class="social-icons">
        <!-- Enlaces a redes sociales -->
        <a href="#"><i class="fab fa-facebook-f"></i></a>
        <a href="#"><i class="fab fa-twitter"></i></a>
        <a href="#"><i class="fab fa-whatsapp"></i></a>
        <a href="#"><i class="fab fa-instagram"></i></a>
    </div>
</footer>
</body>
</html>

```

Anexo N. Codificación del archivo de registro guardar_datos_registro.php.

```
<?php
// Datos de conexión a la base de datos
$servername = "ip";
$username = "usuario";
$password = "contraseña";
$dbname = "nombre de la BD";

// Crear conexión a la base de datos
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar si la conexión es exitosa
if ($conn->connect_error) {
    // Si hay un error de conexión, muestra un mensaje y detiene la
    // ejecución
    die("Conexión fallida: " . $conn->connect_error);
}

$success = false; // Variable para determinar si la inserción fue
// exitosa
$message = ""; // Variable para almacenar el mensaje que se mostrará al
// usuario

// Verificar si se ha enviado un formulario por POST
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Obtener los valores del formulario que el usuario ha enviado
    $nombre = $_POST['nombre'];
    $telefono = $_POST['telefono'];
    $email = $_POST['email'];
    $placa = $_POST['placa'];
    $marca = $_POST['marca'];
    $modelo = $_POST['modelo'];
    $anio = $_POST['anio'];

    // Validación de los datos del formulario
    if (!preg_match("/^[A-Z\s]+$/", $nombre)) {
        // Validación del nombre: solo letras mayúsculas
        $message = "Nombre del propietario inválido. Solo se permiten
letras mayúsculas.";
    } elseif (!preg_match("/^[0-9]{10}$/", $telefono)) {
        // Validación del teléfono: debe tener 10 dígitos
        $message = "Teléfono inválido. Deben ser 10 dígitos.";
    } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL) ||
!preg_match("/^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$/", $email)) {
        // Validación del email: formato válido
        $message = "Email inválido.";
```

```

    } elseif (!preg_match("/^[A-Z]{3}[0-9]{4}$/", $placa)) {
        // Validación de la placa: formato de 3 letras y 4 números
        $message = "Placa del vehículo inválida.";
    } elseif (!preg_match("/^[A-Z]+$/", $marca)) {
        // Validación de la marca: solo letras mayúsculas
        $message = "Marca del vehículo inválida.";
    } elseif (!preg_match("/^[A-Z\s]+$/", $modelo)) {
        // Validación del modelo: solo letras mayúsculas
        $message = "Modelo del vehículo inválido.";
    } elseif ($anio < 1886 || $anio > 2024) {
        // Validación del año: debe estar entre 1886 y el año actual
        $message = "Año del vehículo inválido.";
    } else {
        // Si todas las validaciones pasan, iniciar la transacción en
la base de datos
        $conn->begin_transaction();

        try {
            // Insertar los datos del propietario en la tabla
'propietarios'
            $sql_propietarios = "INSERT INTO propietarios (placa,
propietario, telefono, email) VALUES ('$placa', '$nombre', '$telefono',
'$email')";
            if ($conn->query($sql_propietarios) !== TRUE) {
                // Si la inserción falla, lanza una excepción y la
transacción se revierte
                throw new Exception("Error: " . $sql_propietarios .
"<br>" . $conn->error);
            }

            // Insertar los datos del vehículo en la tabla
'vehiculosReportados'
            $sql_vehiculos = "INSERT INTO vehiculosReportados (placa,
marca, modelo, año) VALUES ('$placa', '$marca', '$modelo', '$anio')";
            if ($conn->query($sql_vehiculos) !== TRUE) {
                // Si la inserción falla, lanza una excepción y la
transacción se revierte
                throw new Exception("Error: " . $sql_vehiculos . "<br>"
. $conn->error);
            }

            // Si no hubo errores, confirmar la transacción
            $conn->commit();
            $message = "Nuevo registro creado exitosamente.";
            $success = true;
        } catch (Exception $e) {
            // Si ocurre un error, revertir los cambios realizados en
la base de datos

```

```

        $conn->rollback();
        $message = "No se ha realizado el registro, intentelo
nuevamente.";
    }
}

// Cerrar la conexión a la base de datos
$conn->close();
}
?>

<!DOCTYPE html>
<html lang="es">
<head>
    <!-- Definición del juego de caracteres para la página -->
    <meta charset="UTF-8">
    <!-- Definición de la vista para dispositivos móviles -->
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <!-- Título de la página -->
    <title>Registro de Vehículos Robados</title>
    <!-- Enlaces a hojas de estilo externas -->
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
</head>
<body class="otras-secciones">
    <!-- Cabecera con el logo y los enlaces de navegación -->
    <header>
        <div class="logo">
            
        </div>
        <nav>
            <ul>
                <li><a href="pagina.php?section=inicio">Inicio</a></li>
                <li><a
href="pagina.php?section=reportes">Reportes</a></li>
                <li><a
href="pagina.php?section=consultas">Consultas</a></li>
                <li><a
href="pagina.php?section=contactos">Contactos</a></li>
            </ul>
        </nav>
    </header>

    <!-- Sección principal donde se muestra el formulario y el mensaje
de éxito/error -->

```

```

    <main>
        <section id="reportes" class="otras-secciones">
            <h2>Reportar Vehículo Robado</h2>
            <!-- Sección con el código QR para unirse a un grupo de
Telegram -->
            <div id="qr-code-section" class="qr-code-section">
                <h1>Únete aquí al grupo de Telegram mediante el código
QR:</h1>
                
            </div>
            <!-- Sección que muestra el mensaje de éxito o error
después de procesar el formulario -->
            <div id="success-message" class="success-message">
                <p><?php echo $message; ?></p>
            </div>
        </section>
    </main>

    <!-- Pie de página con enlaces a redes sociales -->
    <footer>
        <h3>REDES SOCIALES</h3>
        <div class="social-icons">
            <a href="#"><i class="fab fa-facebook-f"></i></a>
            <a href="#"><i class="fab fa-twitter"></i></a>
            <a href="#"><i class="fab fa-whatsapp"></i></a>
            <a href="#"><i class="fab fa-instagram"></i></a>
        </div>
    </footer>
</body>
</html>

```

Anexo O. Codificación del archivo de consulta datos_consulta.php.

```

<?php
// Datos de conexión a la base de datos
$servername = "ip";
$username = "usuario";
$password = "contraseña";
$dbname = "nombre de la BD";

// Crear conexión a la base de datos
$conn = new mysqli($servername, $username, $password, $dbname);

```

```

// Verificar si la conexión fue exitosa
if ($conn->connect_error) {
    // Si hay un error de conexión, muestra un mensaje y detiene la
    ejecución
    die("Conexión fallida: " . $conn->connect_error);
}

$message = ""; // Variable para almacenar el mensaje que se mostrará al
usuario

// Verificar si se ha enviado un formulario por POST
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Obtener y escapar los datos enviados desde el formulario para
    evitar inyecciones SQL
    $placa = $conn->real_escape_string($_POST['placa-consulta']);
    $marca = $conn->real_escape_string($_POST['marca-consulta']);
    $modelo = $conn->real_escape_string($_POST['modelo-consulta']);
    $anio = $conn->real_escape_string($_POST['anio-consulta']);

    // Consulta a la tabla 'vehiculosReportados' buscando coincidencias
    exactas
    // con los datos proporcionados
    $sql_vehiculos = "SELECT * FROM vehiculosReportados WHERE placa =
'$placa' AND marca = '$marca' AND modelo = '$modelo' AND año =
'$anio'";
    $result_vehiculos = $conn->query($sql_vehiculos);

    // Si se encuentra el vehículo en la tabla 'vehiculosReportados'
    if ($result_vehiculos->num_rows > 0) {
        // Consulta a la tabla 'logs' buscando registros que contengan
        // la placa del vehículo
        $sql_logs = "SELECT * FROM logs WHERE mensaje LIKE '%$placa%'";
        $result_logs = $conn->query($sql_logs);

        // Si se encuentran registros en 'logs' relacionados con la
        placa
        // del vehículo
        if ($result_logs->num_rows > 0) {
            // Iterar sobre los registros encontrados en 'logs' y
            mostrar
            // la información
            while($row_logs = $result_logs->fetch_assoc()) {
                $message .= "Vehículo encontrado con placa: " .
                $row_logs["mensaje"] . "<br>";
                $message .= "Fecha: " . $row_logs["fecha"] . "<br>";
                $message .= "Ubicación: <a href='" .
                $row_logs["ubicacion"] . "'>" . $row_logs["ubicacion"] . "</a><br>";
            }
        }
    }
}

```

```

        $message .= "<img src='" . $row_logs["foto"] . "'
alt='Foto del vehículo'><br>";
    }
} else {
    // Si no se encuentran registros en 'logs', se muestra
    // un mensaje indicando que no se encontraron coincidencias
    $message = "No se ha encontrado el vehículo con los datos
ingresados."; // En la tabla 'logs'
}
} else {
    // Si no se encuentra el vehículo en 'vehiculosReportados', se
    // muestra un mensaje indicando que no se encontró
    $message = "No se ha encontrado el vehículo con los datos
ingresados."; // En la tabla 'vehiculosReportados'
}

// Cerrar la conexión a la base de datos
$conn->close();
}
?>
<!DOCTYPE html>
<html lang="es">
<head>
    <!-- Definición del conjunto de caracteres para la página -->
    <meta charset="UTF-8">
    <!-- Definición de la vista para dispositivos móviles -->
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <!-- Título de la página -->
    <title>Consulta de Vehículos</title>
    <!-- Enlace a hojas de estilo externas -->
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
</head>
<body class="otras-secciones">
    <!-- Cabecera con el logo y los enlaces de navegación -->
    <header>
        <div class="logo">
            
        </div>
        <nav>
            <ul>
                <li><a href="pagina.php?section=inicio">Inicio</a></li>
                <li><a
href="pagina.php?section=reportes">Reportes</a></li>

```

```

        <li><a
href="pagina.php?section=consultas">Consultas</a></li>
        <li><a
href="pagina.php?section=contactos">Contactos</a></li>
    </ul>
</nav>
</header>

<!-- Contenido principal de la página donde se muestra el resultado
de la consulta -->
<main>
    <section id="consultas" class="otras-secciones">
        <h2>Consultar Vehículo</h2>
        <!-- Mostrar el mensaje de resultado de la consulta (éxito
o error) -->
        <div id="success-message" class="success-message">
            <p><?php echo $message; ?></p>
        </div>
    </section>
</main>

<!-- Pie de página con enlaces a redes sociales -->
<footer>
    <h3>REDES SOCIALES</h3>
    <div class="social-icons">
        <a href="#"><i class="fab fa-facebook-f"></i></a>
        <a href="#"><i class="fab fa-twitter"></i></a>
        <a href="#"><i class="fab fa-whatsapp"></i></a>
        <a href="#"><i class="fab fa-instagram"></i></a>
    </div>
</footer>
</body>
</html>

```

Anexo P. Codificación del archivo de estilos styles.css.

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-size: cover;
  background-repeat: no-repeat;
  background-attachment: fixed;
}

.inicio-contactos {
  background-image: url('img/inicio.png');
  color: white;
}

.otras-secciones {
  background-image: url('img/blanco.png');
  color: black;
}

.text-inicio {
  display: flex;
  justify-content: flex-start; /* Alinea los elementos al inicio */
  align-items: center;
  background-color: rgba(0, 0, 0, 0.5);
  padding: 10px 20px;
}

.text-inicio h1 {
  font-size: 2.5em; /* Ajusta el tamaño del título */
  margin-bottom: 10px; /* Espacio entre el título y el párrafo */
}

.text-inicio p {
  text-align: justify;
  width: 50%;
  margin: 0; /* Elimina el auto margin para alinear a la izquierda */
  line-height: 2; /* Ajusta el espacio entre líneas */
  font-size: 1.3em; /* Ajusta el tamaño del párrafo */
}

.responsive-img {
  margin-left: 100px;
  width: 400px; /* Ajusta el ancho según tus necesidades */
  height: auto;
```

```

    object-fit: contain;
}

.text-contactos {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: rgba(0, 0, 0, 0.5);
    padding: 10px 20px;
}

.text-contactos h2 {
    font-size: 2.5em; /* Ajusta el tamaño del título */
    margin-bottom: 10px; /* Espacio entre el título y el párrafo */
}

.text-contactos p {
    text-align: justify;
    line-height: 2; /* Ajusta el espacio entre líneas */
    font-size: 1.3em; /* Ajusta el tamaño del párrafo */
}

header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: rgba(0, 0, 0, 0.5);
    padding: 10px 20px;
}

.logo img {
    height: 50px;
}

nav ul {
    list-style: none;
    display: flex;
    gap: 20px;
}

nav ul li {
    display: inline;
}

nav ul li a {
    color: white;
    text-decoration: none;
}

```

```

    font-size: 18px;
    cursor: pointer;
}

main {
    padding: 20px;
}

section {
    margin-bottom: 40px;
}

.qr-code-section {
    text-align: center;
    margin-top: 20px;
}

.success-message {
    display: block;
    text-align: center;
    margin-top: 20px;
    color: green;
}

h1, h2 {
    color: inherit;
}

form {
    display: flex;
    flex-direction: column;
    gap: 10px;
}

label {
    color: inherit;
}

input, button {
    padding: 10px;
    font-size: 16px;
}

input::placeholder {
    color: #ccc;
}

button {

```

```
background-color: #007BFF;
color: white;
border: none;
cursor: pointer;
}

button:hover {
background-color: #0056b3;
}

footer {
background-color: rgba(0, 0, 0, 0.5);
color: white;
text-align: center;
padding: 10px 0;
}

footer .social-icons {
display: flex;
justify-content: center;
gap: 20px;
}

footer .social-icons a {
color: white;
font-size: 24px;
text-decoration: none;
}
```

Anexo Q. Pruebas de vuelo del UAV en los parqueaderos de la Universidad Técnica de Ambato.

En la Figura Q1 se muestra el control de vuelo del UAV de forma manual.



Figura Q1. Control de vuelo del UAV de forma manual.

En la Figura Q2, Q3, Q4 se observan las pruebas de vuelo del UAV en el parqueadero 1, 2 y 3 de la Universidad Técnica de Ambato.



Figura Q2. Prueba de vuelo del UAV en el parqueadero 1 de forma manual.



Figura Q3. Prueba de vuelo del UAV en el parqueadero 2 de forma manual.

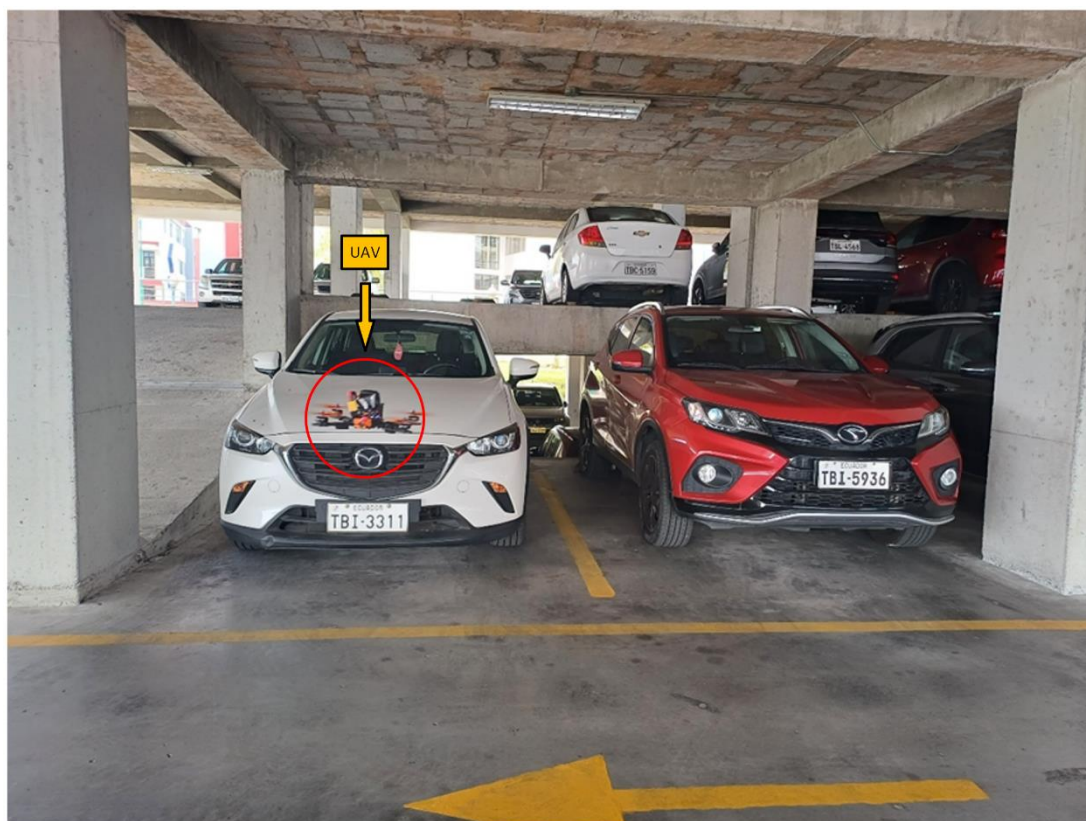


Figura Q4. Prueba de vuelo del UAV en el parqueadero 3 de forma manual.

Anexo R. Tabla de abreviaturas.

Abreviaturas	Definición
ALPR - ANPR	Reconocimiento Automático de Matrículas de Vehículos.
CCTV	Televisión de Circuito Cerrado (sistema de cámaras de video que transmiten imágenes a un número limitado de monitores en un circuito cerrado).
CNN	Red Neuronal Convolutacional.
COCO	Conjunto de datos ampliamente utilizado en el campo de la visión por computadora.
CPU	Unidad Central de Procesamiento
CPU ONNX	Unidad Central de Procesamiento compatible con ONNX
DeepSORT	SORT Basado en Aprendizaje Profundo (extensión de SORT que utiliza características de aprendizaje profundo para mejorar la asignación de objetos y el seguimiento).
DGAC	Dirección General de Aviación Civil.
DL	Deep Learning (Aprendizaje Profundo).
Faster R-CNN	Red Neuronal Convolutacional Basada en Regiones Más Rápida.
FPV	Vista en Primera Persona.

Abreviaturas	Definición
GPS	Sistema de Posicionamiento Global.
HTML	Lenguaje de Marcado de Hipertexto.
HTTP	Protocolo de Transferencia de Hipertexto.
HLC	Compensación de luces altas.
IA	Inteligencia artificial.
IoU	Intersección sobre Unión (es una métrica para evaluar la precisión de un modelo al predecir la localización de un objeto dentro de una imagen).
IP	Protocolo de Internet.
LiPo	Polímero de Litio.
LPR	Reconocimiento de Matrículas de Vehículos.
MIME	Extensiones de Correo de Internet para Propósitos Múltiples.
NMS	Supresión de No Máximos (algoritmo utilizado en visión por computadora, para eliminar predicciones redundantes de objetos y mantener solo las más precisas).
OCR	Reconocimiento óptico de caracteres.
OSD	Pantalla de Visualización en Pantalla.
OBS	Software de Transmisión Abierta.
PHD	Preprocesador de Hipertexto.
RGB	Red, Green, Blue (colores primarios de la luz utilizados en la creación de otros colores en pantallas electrónicas como monitores, televisores y cámaras digitales).
RFID	Identificación por Radiofrecuencia.
SORT	Seguimiento en Línea y en Tiempo Real Sencillo (algoritmo simple y rápido para el seguimiento de objetos en tiempo real.)
SMT	Protocolo Simple de Transferencia de Correo.
SQL	Sistema de gestión de bases de datos.
SSD	Detección de Múltiples Cajas en una Sola Toma.
ByteTrack	Algoritmo de Seguimiento Robusto Usando Codificación de Pares de Bytes para la Asociación de Datos.
TCP	Protocolo de Control de Transmisión.
UAV	Vehículo aéreo no tripulado.
VTX	Transmisor de Video.
VRX	Receptor de Video.
vCPU	CPU Virtual
YOLO	You Only Look Once (realiza toda la detección de objetos en una sola pasada).