UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA DE RIEGO INTELIGENTE DE CORTO ALCANCE PARA JARDINES A PARTIR DE VISIÓN ARTIFICIAL

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

ÁREA: Comunicaciones

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

AUTOR: Gissela Abigail Jiménez Albán

TUTOR: Ing. Ana Pamela Castro Martin, MSc.

Ambato - Ecuador

febrero -2024

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE RIEGO INTELIGENTE DE CORTO ALCANCE PARA JARDINES A PARTIR DE VISIÓN ARTIFICIAL, desarrollado bajo la modalidad Proyecto de Investigación por la señorita Gissela Abigail Jiménez Albán, estudiante 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 el estudiante ha sido tutorado 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 2024.

Ing. Ana Pamela Castro Martin, MSc.

TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA DE RIEGO INTELIGENTE DE CORTO ALCANCE PARA JARDINES A PARTIR DE VISIÓN ARTIFICIAL 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 2024.

Gissela Abigail Jiménez Albán

C.C. 1804391645

AUTOR

DERECHOS DE AUTOR

Autorizo 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, cedo todos mis 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 2024.

Gissela Abigail Jiménez Albán

C.C. 1804391645

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por la señorita Gissela Abigail Jiménez Albán, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE RIEGO INTELIGENTE DE CORTO ALCANCE PARA JARDINES A PARTIR DE VISIÓN ARTIFICIAL, 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 la señora Presidente del Tribunal.

Ambato, febrero 2024.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Elsa Pilar Urrutia Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL

ELIZABETH PAULINA AYALA BANO

Firmado digitalmente por ELIZABETH PAULINA AYALA BANO
DN: cn=ELIZABETH PAULINA AYALA BANO, c=EC, l=QUITO, o=BANCO CENTRAL DEL ECUADOR, ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE
Motivo: He revisado este documento
Ubicación:
Fecha: 2024-02-08 15:20:05:00

Ing. Elizabeth Paulina Ayala, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
MARLON ANTONIO SANTAMARIA VILLACIS

Ing. Marlon Antonio Santamaría, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

El presente proyecto de investigación va dedicado a mi familia por guiarme y motivarme a seguir adelante.

A mis padres, por sus constantes muestras de apoyo y consejos que me han impulsado a cumplir mis sueños toda mi vida.

A mis hermanos, por estar siempre a mi lado animándome en mi camino.

AGRADECIMIENTO

Agradezco principalmente a mis padres, por ser un pilar fundamental, por creer en mí, por guiarme, y por su aliento constante para superar desafíos.

Agradezco a mis hermanos, por estar siempre presentes ofreciendo palabras de aliento, comprensión y cariño.

Un agradecimiento especial a mi Tutor de Tesis Ing. Ana Pamela Castro Martin, MSc. por su colaboración y asesoramiento constante

También, quiero extender mi gratitud a todas aquellas personas con quienes he compartido tantas situaciones exitosas y desafiantes.

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN EJECUTIVO

El presente proyecto de investigación desarrolla un sistema de riego inteligente de corto alcance para jardines a partir de visión artificial. El objetivo principal es implementar un sistema de riego inteligente como una herramienta de riego eficiente para superficies de césped, que brinde reconocimiento del área y genere un riego acorde utilizando visión artificial. A diferencia de proyectos similares, no controla una red de aspersores o carros robot. El algoritmo de detección, se compone de un entrenamiento en yolov5 para reconocer el césped y utiliza Python para detectar contornos de césped y objetos.

El sistema incorpora componentes electrónicos para lograr un riego eficiente y automatizado. En la adquisición de datos, se optó por el sensor de humedad de suelo FC-28 y la ESP32-CAM como cámara IP. En actuadores, se seleccionó una válvula solenoide normalmente cerrada para controlar el flujo de agua y servomotores TD-8120MG para direccionar el aspersor. La ESP32 controla el sistema, la comunicación, se establece mediante tecnología Wi-Fi con una base de datos local. El monitoreo, se realiza con una aplicación móvil desarrollada en Android Studio por compatibilidad de usuarios y culminar en un sistema integral que demuestra la aplicación práctica de la visión artificial en el riego inteligente para jardines.

Se realizaron pruebas de funcionamiento del sistema, se logró una precisión en la detección del césped del 95,6%. Las pruebas realizadas en un jardín indicaron que el sistema inteligente puede ser beneficioso para el riego eficiente al identificar con precisión contornos de objetos y áreas verdes.

Palabras clave: Visión artificial, riego en jardines, detección de áreas verdes, detección de objetos.

ABSTRACT

The present research project develops an intelligent short-range irrigation system for gardens based on artificial vision. The main objective is to implement an intelligent irrigation system as an efficient irrigation tool for turf surfaces, which provides recognition of the area and generates a corresponding irrigation using artificial vision. Unlike similar projects, it does not control a network of sprinklers or robot carts. The detection algorithm consists of training in yolov5 to recognize the lawn and uses Python to detect lawn contours and objects.

The system incorporates electronic components for efficient and automated irrigation. For data acquisition, the FC-28 soil moisture sensor and the ESP32-CAM as IP camera were chosen. For actuators, a normally closed solenoid valve was selected to control the water flow and TD-8120MG servomotors to direct the sprinkler. The ESP32 controls the system, communication is established through Wi-Fi technology with a local database. Monitoring is done with a mobile application developed in Android Studio for user compatibility and culminating in a comprehensive system that demonstrates the practical application of machine vision in smart garden irrigation.

Performance tests of the system were conducted, and a grass detection accuracy of 95,6% was achieved. Tests conducted in a garden indicated that the intelligent system can be beneficial for efficient irrigation by accurately identifying contours of objects and green areas.

Keywords: Artificial vision, garden irrigation, green area detection, object detection.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA DE RIEGO INTELIGENTE DE CORTO ALCANCE PARA JARDINES A PARTIR DE VISIÓN ARTIFICIAL

1.1.1 Planteamiento del problema

El agua es esencial para la producción agrícola y la seguridad alimentaria. Es el elemento vital de los ecosistemas incluidos los bosques, lagos y humedales, de los que depende nuestra seguridad alimentaria y nutricional presente y futura. Sin embargo, nuestros recursos de agua dulce están disminuyendo a un ritmo alarmante. La creciente escasez de agua es ahora uno de los principales retos para el desarrollo sostenible [1].

A nivel mundial, se concientiza el consumo de agua y la protección de la misma para evitar su desperdicio. A parte de cubrir las necesidades básicas el agua tiene múltiples aplicaciones como lo son el riego en cultivos y jardines. En Estados Unidos más de 19 millones de hectáreas están destinadas a céspedes, lo que supone casi un 5% del total de la tierra destinada a la agricultura en ese país. Esto supone una extensión de terreno considerable el cual para mantener su salud requiere de varios recursos naturales como son el agua y fertilizantes, por lo cual es necesario mantener la sostenibilidad ambiental promoviendo técnicas de conservación y alternativas eficientes para el uso de la tierra [2].

Mediante estudios e investigaciones, se considera a los parámetros de distribución y eficiencia como maneras que ayudan a realizar el riego en jardines eficientemente. A diferencia de la agricultura, se conoce que este tipo de riego, se realiza a mayor escala en la zona urbana, donde los jardines son una parte importante. Aunque el riego en la agricultura ha llamado mayor atención actualmente el riego en jardines ocupa un papel más significativo en el uso sostenible del agua [3].

Además, se debe recordar que el agua es irremplazable y necesaria para vivir, como por ejemplo estudios realizados en América Latina indican que, se prioriza el consumo

de agua en sectores como la agricultura ya que busca mejorar los cultivos en base a sus requerimientos, para el consumo humano ya que, se debe satisfacer las necesidades básicas y en el consumo pecuario tanto para el cuidado, suministro y consumo de los productos que brindan [4].

Es importante conocer que, en el Ecuador, se realiza el aprovechamiento de los cultivos y uso eficiente del agua en sistemas de riego, pero no, se ha realizado a profundidad un estudio para el aprovechamiento del agua en jardín en zonas urbanas donde la mayoría de las viviendas no dan la atención que requiere a el área, no solo para obtener un jardín saludable sino para evitar daño de infraestructura en las edificaciones. Tanto los céspedes de estación cálida como los de estación fría, si son regados a un nivel deficitario pueden reducir del 25% al 30% del agua aplicada con un riego óptimo [2].

Dentro de los problemas de riego del jardín, se puede identificar al más importante como el creciente consumo de agua por sistemas de riego ineficaces, otro problema es la falta de planeación, además de terrenos con riego disparejo, y riego inexacto que origina humedad en estructuras y por consiguiente problemas de salud.

En la actualidad la mayor parte de viviendas o residencias de los ecuatorianos constan con pequeñas áreas de jardín que no solo brindan estética, sino que son consideradas pequeñas áreas de recreación. Las cuales por un riego inadecuado producen filtración y humedad en la estructura del edificio. De acuerdo a estudios realizados, se ha demostrado que la humedad y el moho en los hogares provocan daño a la salud en un 30% a 50% con enfermedades de las vías respiratorias y el asma [5].

Tener un jardín decorativo o con pequeños cultivos implica tener tiempo para constantemente podar plantas, regarlas y cuidarlas. Por lo cual, si, no se conoce el tipo de plantas que, se tiene y las necesidades de las mismas, puede que, se realice demasiado regadío ya sea muy prolongado o muy frecuente produciendo humedad, o que, no se distribuya la cantidad de agua correcta y las plantas, se sequen.

El cuidado de jardines debería incluir un plan de riego para que reciba el agua que en verdad necesita y por lo tanto el riego no sea prolongado o muy frecuente para evitar áreas muy húmedas, o en otro caso áreas secas por la falta de agua.

Existen en el mercado sistemas de riego la mayoría funcionan con principios mecánicos y con escasa o inexistente automatización que por su facilidad de adquisición y al no realizar un estudio previo del área de jardín, provoca que no satisfagan las necesidades reales del área realizando una distribución de agua de manera ineficaz.

1.2 Antecedentes investigativos

Para el presente proyecto, se considera de relevancia los siguientes proyectos de investigación al igual que diversos artículos científicos sobre detección de áreas, monitoreo de cultivos y automatización de sistemas. A continuación, se detalla la documentación seleccionada.

En el año 2015, en México, Rendón, G., Cortes, J., Juárez, D., y Ortega, M. publican el artículo "Sistema de riego inteligente utilizando electroválvulas a partir de sensores de visión", el cual indica el funcionamiento de un sistema inteligente para el riego en una cancha de futbol, para el control del riego, se realiza un escaneo del área mediante una cámara IP, sensores de humedad para determinar el estado del suelo y datos de las condiciones climáticas actuales de la zona. Todos los datos obtenidos son almacenados en un servidor y determinan el área que requiere riego, después, se procede a la activación de electroválvulas que, se encuentran bajo la cancha en un conjunto de tuberías para que los aspersores fijos sobresalgan del suelo y realicen el riego. El escaneo, se realiza mediante una toma fotográfica automática en el transcurso de la mañana. El sistema es capaz de controlar el tiempo de riego, generar un ahorro de agua, y ayudar a la recuperación del campo progresivamente [6].

En el año 2018, en Taiwan, Chang, C., y Lin, K., realizan una publicación llamada "Smart Agricultural Machine with a Computer Vision-Based Weeding and Variable-Rate Irrigation Scheme", en la cual, se diseña un carro robot con una cámara para escanear el terreno y las plantas mientras está en movimiento, mediante control fuzzy-logic, se determina la calidad de la planta, inicia el proceso de deshierbe y obtiene datos de la humedad del suelo a través de sensores. Los resultados experimentales

muestran que la tasa media de deshierbe es del 90%, y que el área media de distribución húmeda del suelo superficial puede mantenerse en el 75%. Aunque el riego requiere un tiempo de operación más largo en esta fase, este método puede ahorrar agua si, se tiene mayor velocidad de procesamiento del hardware [7].

En el año 2019, en Ramin University of Agricultural and Natural Resources, Nadafzadeh, M., y Abdanan, S, publican el artículo "Design and fabrication of an intelligent control system for determination of watering time for turfgrass plant using computer vision system and artificial neural network". El sistema de riego automatizado es capaz de medir y evaluar las condiciones de la planta marchita mediante un escaneo fotográfico, la aplicación de filtros y la evaluación de las condiciones morfológicas de la planta. Esto permite proporcionar el agua necesaria para la planta utilizando redes neuronales con cuatro características extraídas de la imagen de la planta (h, L, H y PDF1). Los resultados de esta investigación son casi los mismos que los obtenidos por el algoritmo Support Vector Machine (SVM) para clasificar las plantas de trigo sanas y con estrés hídrico con una precisión de clasificación igual al 98%; la razón de esta elevada precisión es que, en esta investigación, se detectaron los estados de presencia o ausencia de estrés hídrico en cada medición [8].

En el año 2020, en la Universidad Técnica de Ambato, se realizan el proyecto de investigación, "GESTIÓN DE SISTEMA DE RIEGO INTELIGENTE PARA EL CUIDADO DEL PARQUE "PALOMINO FLORES" DE LA CIUDAD DE BAÑOS DE AGUA SANTA", realizado por Gualpa, T., la investigadora busca automatizar el riego a partir de variables climáticas y del suelo, utiliza el módulo ESP32 para adquirir datos del suelo como son la humedad y temperatura. Además, ha desarrollado una interfaz gráfica para establecer una comunicación entre el usuario y el sistema de riego, con valores almacenados en una base de datos local. La interfaz permite monitorizar, supervisar y controlar el estado del riego [9].

En el año 2022, en la Universidad Politécnica Salesiana de Ecuador, Calapaqui, I., Chiguano, A., realizan el proyecto de investigación "DESARROLLO DE UN SISTEMA INTELIGENTE PARA EL CULTIVO DE CLAVELES", se realiza la captura de imágenes para la visión artificial a través de una ESP32-CAM para la

detección de enfermedades en los claveles. Además, verifica la humedad del campo de cultivo mediante sensores y monitoreo mediante la aplicación BLYNK para realizar el riego y supervisar el estado de los actuadores. Una vez iniciado el riego, los niveles de humedad suben hasta llegar al 70% de humedad para que la bomba y electroválvula, se desactiven automáticamente. El proyecto presenta una precisión del 96,66% de detección por lo que, se comprende que el monitoreo es preciso y de ayuda en el cuidado de claveles [10].

En septiembre del año 2022, en la Universidad Técnica de Ambato, Acosta, A. realiza el proyecto de investigación "SISTEMA HIDROPÓNICO INTELIGENTE APLICADO A LA PRODUCCIÓN DEL FORRAJE VERDE CON ARQUITECTURA IoT", el investigador usa sensores de temperatura, humedad y ph, para a través de un Arduino Mega 2560 y una Raspberry Pi 3b V1.2, disminuir las tareas manuales de riego y mezcla de soluciones nutritivas que demanda la germinación, evolución y crecimiento del cultivo de forraje verde hidropónico. Mediante la combinación de telecontrol y automatización permite al hidrocultor tener mayor productividad en sus tareas diarias por los reportes generados y las alertas. Además de que el tiempo de mantenimiento y cuidado del cultivo, se reduce [11].

En el año 2023, en la Universidad Técnica de Ambato, se realizan el proyecto de investigación, "SISTEMA AUTOMATIZADO DE CONTROL Y MONITOREO BASADO EN TECNOLOGÍA LORAWAN Y MQTT PARA EL CULTIVO DE HORTALIZAS BAJO INVERNADERO", realizado por Pérez, F., implementa un sistema de control y monitoreo para el cultivo de hortalizas bajo invernadero basado en tecnología Lora y MQTT, en un cultivo de pimiento, donde, se obtiene como resultado un incremento de producción del 30,77% equivalente a $96,00 en la producción del invernadero, así como un control manual y automatizado del invernadero en base a la temperatura, humedad, pH e iluminación, dotando al invernadero de una autonomía inteligente, así como de un monitoreo en tiempo real de las condiciones climáticas de su interior. El sistema presenta un consumo de 46,44kW/h [12].

1.3 Fundamentación teórica

1.3.1 Riego

El propósito del riego es satisfacer de manera adecuada las demandas hídricas de las plantas en términos de cantidad y calidad, con el fin de mantener la humedad del suelo en la zona de las raíces en niveles que favorezcan un óptimo desarrollo del cultivo. El riego adquiere una importancia destacada en situaciones donde las precipitaciones naturales resultan insuficientes para mantener el nivel de humedad necesario para el crecimiento saludable de las plantas [9].

El riego es crucial para mantener el césped en un estado constantemente verde. Por lo tanto, es esencial garantizar el acceso a una fuente o reserva de agua confiable y planificar todo el sistema de riego, preferiblemente con componentes como una bomba, un tubo para la aplicación de fertilizantes líquidos, y un controlador automático que incluya tensiómetros y temporizadores. En ausencia de lluvias, se recomienda programar el riego cada tres días [13].

a. Importancia del riego en el crecimiento de las plantas

Una planta depende en gran medida del suministro de agua para mantener su estructura erguida, absorber minerales del suelo y realizar el proceso de fotosíntesis. La implementación de un sistema de riego bien diseñado presenta numerosas ventajas tanto para el propietario del jardín como para el propio espacio verde. No solo contribuye a establecer y mantener un césped atractivo, sino que una instalación de riego profesional también puede generar ahorros significativos en términos de tiempo, energía y costos asociados con el riego manual. Uno de los beneficios más evidentes de un sistema de riego consiste en la protección que brinda a los jardines, las plantas y los árboles frente a la ineficiencia del riego y las condiciones de sequía. Un sistema bien diseñado garantiza que el césped y las plantas reciban la cantidad precisa de agua. En las estaciones más secas o en periodos de precipitación insuficiente, un sistema de riego puede marcar la diferencia entre un césped marchito y plantas moribundas, y un jardín exuberante y saludable. Es esencial comprender que, al igual que proporcionar la cantidad adecuada de agua, evitar el exceso de riego es igualmente importante. El

exceso de agua puede dañar el suelo, asfixiar las plantas e incluso fomentar el crecimiento de hierbas no deseadas. Un sistema de riego eficiente contribuye al control preciso de la cantidad de agua utilizada y garantiza una absorción y drenaje adecuados, aspectos cruciales de un sistema de riego de alta calidad [14].

b. Consumo eficiente de agua

Las plantas necesitan agua tanto para su desarrollo, que implica la creación de materia orgánica o "materia verde", como para la transpiración. Además, el suelo que rodea un cultivo pierde agua debido a la evaporación. La cantidad de agua que, se extrae del suelo cada día debido a estos procesos, se conoce como evapotranspiración. La demanda de agua de un cultivo, se refiere a la cantidad necesaria para compensar el déficit de agua en el suelo durante su período de crecimiento. Para satisfacer esta demanda hídrica, es necesario contar con precipitaciones naturales o, en caso de que estas sean insuficientes, aplicar riego [15].

c. Programación de riegos

La falta de uniformidad en el riego suele deberse al hecho de que las parcelas reciben agua en momentos distintos, incluso si el cultivo es el mismo. Esto es más común en sistemas móviles y semifijos, donde las distintas áreas de riego tienen duraciones variables. Es crucial que todas las divisiones reciban riego durante el mismo período. Cuando el cambio entre divisiones es gestionado manualmente, es necesario planificar los horarios con atención para evitar interferencias con las actividades diarias [16].

Cuando, se trata de determinar la frecuencia de riego, es crucial considerar este aspecto como un factor en el que a menudo, se cometen errores significativos. En particular, en el caso de céspedes, los riegos breves y muy frecuentes pueden dar como resultado un sistema de raíces superficial que no es propicio para el crecimiento saludable de la vegetación. Además, esta práctica puede propiciar la aparición de enfermedades fúngicas, especialmente cuando, se combina con altas temperaturas, ya que la rápida evaporación conlleva a la pérdida de agua y puede dañar la salud del césped. Por lo tanto, en la mayoría de las situaciones, a menos que existan circunstancias excepcionales, se sugiere limitar el riego a una o dos veces al día, como, se mencionó

anteriormente, aplicando la cantidad diaria calculada de agua (entre 5 y 10 litros por metro cuadrado) durante un período total de 30 a 40 minutos [17].

En la jardinería, en parques y en campos deportivos, se deben prestar especial atención a otros aspectos técnicos relacionados con el riego. Estos incluyen la necesidad de garantizar una distribución uniforme del agua y asegurar que los aspersores, se superpongan adecuadamente. Diversos factores pueden influir en estos aspectos. Por ejemplo, es común encontrarse con sistemas de riego que no están bien diseñados o mantenidos, lo que lleva a la combinación de aspersores de diferentes marcas o modelos, o al uso descuidado de boquillas con diferentes flujos y alcances en los aspersores. En tales situaciones, la distribución del agua nunca resulta uniforme [17].

1.3.2 Métodos de riego

a. Riego por gravedad

El proceso, se basa en permitir que el agua fluya de forma natural por la superficie del terreno debido a la inclinación del mismo, aprovechando la fuerza de la gravedad. Este enfoque es viable en áreas donde existe un marcado desnivel topográfico, lo que facilita el movimiento del agua a lo largo de las pendientes [15]. Dentro de esto, se tiene lo siguientes métodos que, se describen en la Tabla 1.

Tabla 1. Métodos de riego por gravedad [15] [18]

Métodos	Descripción
Riego tendido, a flujo libre (wild flooding)	El agua fluye sin restricciones por el terreno, cubriendo de manera uniforme la zona destinada al cultivo.
Riego por surcos	Se crean surcos o canales en el suelo que, provocan que el agua, se dirija a lo largo de estos surcos para llegar a las raíces de las plantas.
Riego por melgas	Se dividen los campos en secciones pequeñas llamadas melgas, donde el agua, se distribuye de manera controlada en cada melga.
Riego por inundación	El agua inunda el campo por completo, creando un ambiente temporalmente inundado que favorece la absorción del agua por parte de las plantas.

Se utiliza principalmente en áreas donde existe una abundante disponibilidad de agua, lo que significa que no es necesario maximizar la eficiencia de la distribución de agua. Esto contrasta con las regiones más áridas o con recursos hídricos limitados, donde cada gota de agua debe utilizarse con la máxima eficiencia posible. En estas áreas, los métodos de riego por gravedad pueden no ser la opción más adecuada debido a su

eficiencia de aplicación relativamente baja, que generalmente no supera el 45%. Sin embargo, en lugares con una fuente de agua abundante, el riego por gravedad sigue siendo una opción viable y económica para proporcionar la humedad necesaria a los cultivos [15].

b. Riego por presurizado

El agua fluye a una presión controlada a través de conductos cerrados, como tuberías, mangueras o cintas de riego. Este método no depende de la topografía del terreno, ya que es la presión del agua y la longitud de los conductos los factores determinantes de hasta dónde, se puede llevar el agua [15]. Dentro de esto, se conocen los siguientes métodos que, se describen en la Tabla 2.

Tabla 2. Métodos de riego por presurizado [15] [18] [19]

Métodos	Descripción
Riego por goteo	Se distribuye el agua en forma de gotas de manera uniforme y constante, lo que resulta en la humedad del suelo necesaria para que las plantas puedan absorber los nutrientes esenciales para su crecimiento. Este enfoque es ideal para riego localizado en cultivos grades de plantas pequeñas, presenta una eficiencia del 75% al 90%.
Riego por aspersión	El agua es distribuida a las plantas o cultivos de manera similar a la lluvia mediante un sistema de aspersión que abarca una mayor extensión de terreno en comparación con el riego por goteo. Ideal para pastos. Independientemente de las características del terreno donde esté instalado, el aspersor puede desempeñar su función con una eficiencia del 65% al 85%.
Riego por micro aspersión	Similar al riego por aspersión, pero con boquillas más pequeñas que dispersan el agua en gotas más finas en un área designada, adecuado para cultivos que requieren una humedad más controlada. Tiene una eficiencia de aplicación del 70% al 90%
Riego por exudación	Proporciona la cantidad necesaria de agua a las plantas mediante un tubo textil poroso que distribuye el agua por los poros de manera uniforme en toda su superficie. Este método presenta versatilidad y numerosas ventajas en contraste con los sistemas de riego tradicionales o el riego por goteo. Permite ahorrar del 50% al 60% de agua.

En el riego por aspersión, el agua viaja mediante tuberías de PVC o mangueras plásticas hasta los aspersores que emularan la lluvia natural. En general, la mayoría de los cultivos pueden adaptarse a este sistema, especialmente los pastos. Su adaptabilidad a terrenos variados lo hace apropiado para zonas con desniveles irregulares. Además, ofrece eficiencia en la distribución del agua, lo que conlleva a ahorros sustanciales en agua y costos, especialmente en áreas con recursos hídricos limitados. Facilita la automatización y el control preciso del riego, aspectos cruciales

en la agricultura moderna para optimizar la producción y minimizar el desperdicio de agua [15].

1.3.3 Sistemas de riego

Un sistema de riego, se refiere a la agrupación de componentes físicos que permite la administración del agua requerida para el cultivo de una zona específica, garantizando las condiciones hídricas adecuadas para las plantas. Hoy en día, se encuentran disponibles diversos procedimientos para el riego de cultivos, los cuales pueden diferir en términos de gasto, eficacia o comodidad de utilización, aunque todos comparten un objetivo común [20].

Los sistemas de riego representan enfoques y estrategias empleados para proveer agua de forma regulada a los cultivos agrícolas, jardines y otros terrenos verdes, se caracterizan por su localización ya que solo humedecen una parte de tal manera que las plantas absorben el agua y los nutrientes necesarios. Se pueden distinguir diversas categorías de sistemas de riego, como el riego manual, el riego automático y el sistema de riego avanzado [21].

a. Sistema de riego manual

Su principal ventaja radica en su sencilla instalación y su atractivo desde un punto de vista económico. Sin embargo, sus desventajas principales, se relacionan con su uso y la dificultad en el control del suministro de agua, lo que suele resultar en un riego desigual, con algunas áreas recibiendo más agua que otras [22]. Las principales características que involucran este sistema de riego manual, se detallan en la Figura 1.

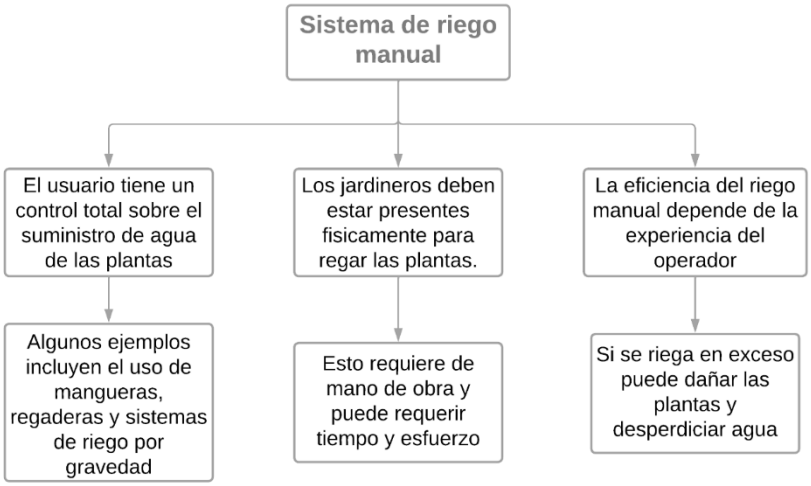


Figura 1. Características de un sistema de riego manual [22] [17]

El sistema de riego manual opera de manera relativamente sencilla, ya que, se basa principalmente en la participación humana para administrar el agua a las plantas de forma regulada. Para el correcto funcionamiento de este sistema, se requiere de los siguientes parámetros detallados en la Tabla 3.

Tabla 3. Funcionamiento de un sistema de riego manual [17]

Parámetros	Descripción
Fuente de agua	Este sistema requiere de un suministro de agua cercano, como un grifo, una bomba de agua, un pozo, etc.
Equipo de riego	Esto incluye mangueras, regaderas, baldes, y aspersores.
Aplicación del agua	El operador del sistema, se mueve por toda el área que quiere regar y aplica el agua de manera directa.
Control de tiempo	Abarca la gestión del período de riego, ya que el operador determina la duración necesaria para regar cada zona o planta particular. Este tiempo puede fluctuar dependiendo de la estación, el clima y los requerimientos de agua de las plantas.
Observación	El encargado debe estar atento a las plantas y al suelo para garantizar que reciban la cantidad apropiada de agua. Debe evitar tanto el riego en exceso, que podría dañar las raíces, como el riego insuficiente.

b. Sistema de riego automático

Este método de riego automatizado administra el suministro de agua a cultivos o áreas verdes de forma independiente, haciendo uso de varios procedimientos como el riego por goteo y la aspersión. Antes de implementar un sistema de riego automatizado, es esencial llevar a cabo un análisis del terreno y de la tipología de las plantas presentes en el área, lo que permitirá determinar la estrategia más eficaz para la distribución del

agua. Este enfoque también, se integra con diversos componentes, como aspersores, difusores, micro aspersores, entre otros [23].

Una de sus principales ventajas radica en la significativa reducción del consumo de agua en comparación con los métodos de riego tradicionales, logrando una disminución de aproximadamente el 30 o 40%. Los elementos necesarios para la implementación de un sistema automatizado de control de riego incluyen sensores o transductores, como sensores de lluvia, humedad y luz solar, actuadores, como electroválvulas, bombas y válvulas motorizadas, unidades de control programables y, por último, sistemas de comunicación [23] [24]. Las principales características de este sistema son las que, se presentan en la Figura 2.

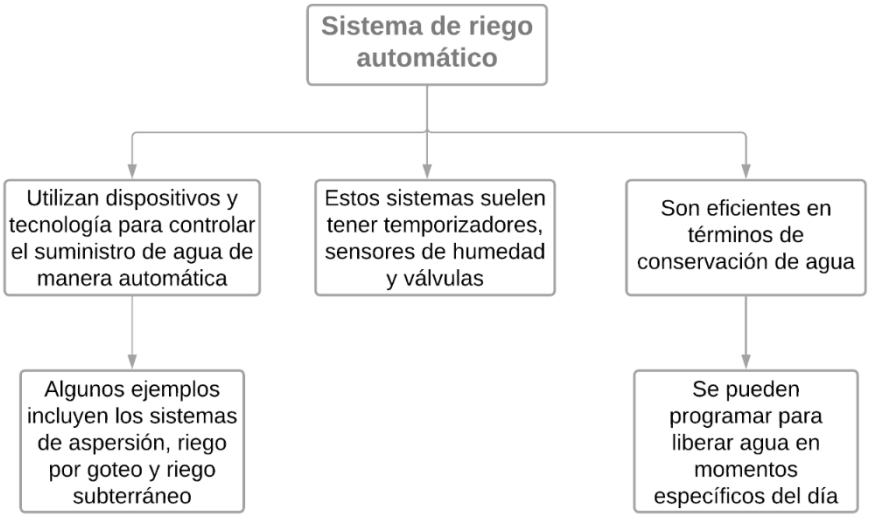


Figura 2. Características de un sistema de riego automático [23].

El funcionamiento de estos sistemas de riego automático, se visualiza en la Figura 3 en donde, se tiene la parte del programador el cual es un dispositivo electrónico compacto, de instalación y uso sencillos, que facilita la gestión del riego al controlar válvulas eléctricas. Además, hay programadores que no dependen de electroválvulas o válvulas solenoides y funcionan simplemente al interrumpir el flujo de agua desde la válvula principal de un sistema de riego. Estos dispositivos no incluyen la capacidad de control automático por sectores, lo que significa que están diseñados para irrigar una única área o sección a la vez. Son una excelente elección para el riego de superficies pequeñas, ya que la gestión por sectores, se puede llevar a cabo de forma manual. Su capacidad de conexión de válvulas puede variar según el modelo, y ofrece

múltiples opciones de programación. Además, en algunos dispositivos, es posible integrar otros sensores útiles para el control del riego. Las electroválvulas o válvulas solenoides son dispositivos que sustituyen a las llaves o válvulas manuales tradicionales empleadas para abrir o cerrar el flujo de agua en una determinada zona de riego. Las conexiones entre el programador y las válvulas solenoides implican que el programador envía una señal eléctrica de baja tensión a las válvulas solenoides, lo que las activa o desactiva [25].

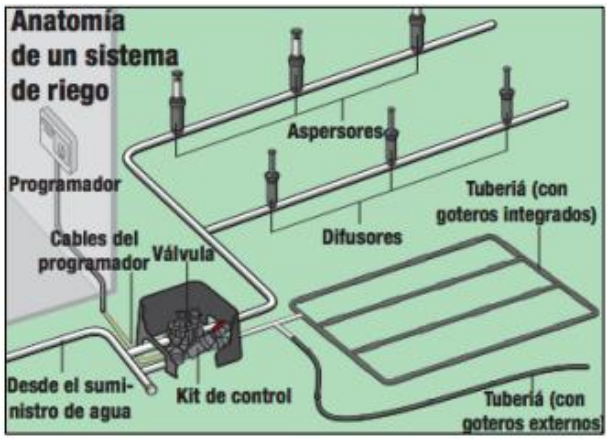


Figura 3. Funcionamiento de un sistema de riego automático [26].

c. Sistema de riego inteligente

El riego inteligente es una técnica avanzada que emplea tecnología para mejorar la eficiencia y la sostenibilidad del riego en la agricultura. Su objetivo es optimizar el uso de agua y energía en la irrigación de cultivos, aumentando la producción de alimentos y reduciendo el impacto ambiental. Este enfoque, se basa en la utilización de sensores que evalúan la humedad del suelo, la temperatura y la evaporación de las plantas, lo que permite determinar con precisión la cantidad necesaria de agua. La información recopilada por los sensores, se transmite al controlador central, el cual realiza ajustes automáticos en la irrigación en el terreno. Asimismo, el riego inteligente puede emplear métodos de teledetección con imágenes satelitales para controlar el desarrollo de los cultivos y adaptar la cantidad de agua proporcionada [27]. Las principales características de estos sistemas inteligentes, se presentan en la Figura 4.

Tabla 4. Funcionamiento general de los sistemas de riego inteligente [28]

Componentes	Descripción
Sensores de Humedad del Suelo	Monitorean la humedad en el suelo en tiempo real y permiten activar el riego cuando la humedad desciende de un umbral predefinido
Sensores Meteorológicos	Registran datos climáticos, como temperatura, humedad, viento y luz solar, ayudan a ajustar el riego según las condiciones climáticas actuales y previsiones.
Conexión a Internet	Los sistemas están conectados a Internet para acceder a datos meteorológicos y permitir a los usuarios controlar y programar el riego por medio de aplicaciones móviles o en línea
Actuadores y Válvulas de Riego	Los actuadores regulan el flujo de agua hacia las áreas de riego, abriendo las válvulas cuando es necesario irrigar
Algoritmos de Control	Programas informáticos analizan los datos de los sensores y toman decisiones sobre el riego, considerando las necesidades de las plantas y las condiciones ambientales.

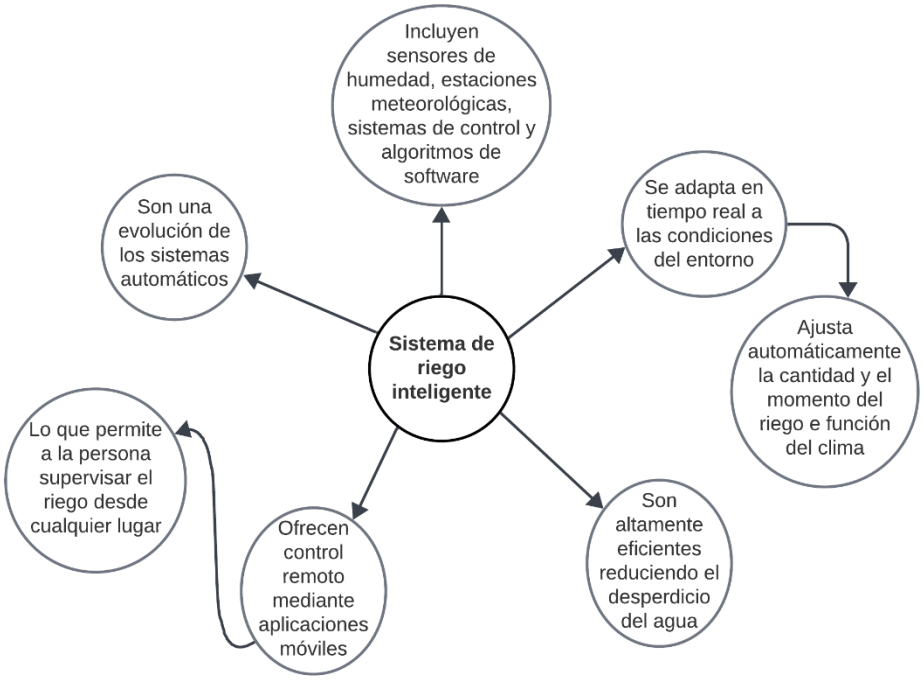


Figura 4. Características de un sistema de riego inteligente [28]

1.3.4 Sistemas de riego para jardines

a. Sistemas de riego tradicionales

Inicialmente los sistemas de riego para jardines eran manuales y consistían en riego por goteo, por aspersores o por manguera. El riego por goteo dirige el agua a las raíces de las plantas, el riego por aspersor simula la lluvia sobre las plantas, y el riego por manguera brinda agua directamente a las plantas [29].

Por lo cual, los sistemas de riego tradicionales, se basan en el uso de métodos de riego por presurizado. De acuerdo a la Tabla 2, los métodos más adecuados son el riego por goteo que, se basa en el enfoque del tipo de cultivo y el riego por aspersión que, se enfoca en pastos. A partir de un análisis cualitativo de los sistemas de riego tradicionales de goteo y aspersión, se presenta en la Tabla 5, el desempeño de los mismos para las categorías de tamaño de área, vegetación, eficiencia hídrica, tipos de sistemas y coste económico.

Tabla 5. Desempeño de los sistemas tradicionales de riego [15] [18] [22] [30]

Categoría	Sistema por goteo	Sistema por aspersión
Área de riego	>2m	>10m
Cultivos de espaciamiento ancho y definido	Apropiado	-
Cultivos de espaciamiento estrecho	-	Apropiado
Áreas de césped/pastos	No en grandes áreas	Apropiado
Adaptabilidad a terrenos irregulares	No	Si
Porcentaje del diámetro de humedecimiento en presencia de viento	No presenta afectación	En distancia de aspersores con marco cuadrado 60% a 6 km/hora 50% a 12 km/hora 40% a 15 km/hora 30% a más de 15 km/hora
Aplicación en condición de escasez de agua	Apropiado	-
Consumo de agua	Bajo	Medio
Eficiencia de riego en relación del agua benéficamente utilizada y el agua total utilizada.	90 - 95%	80 - 85%
Tipos de sistema	“Gota a gota” o Simples “Micro difusión” Autocompensantes Regulables	Oscilantes Circulares Sectoriales Turbina o impacto Multisuperficie
Coste	Medio	Bajo

En base a la comparativa del análisis cualitativo realizado en base al desempeño de los sistemas tradicionales de riego por goteo y aspersión de la Tabla 5, se concluye que ambos sistemas tienen sus méritos y limitaciones, y la selección óptima dependerá de las necesidades y condiciones específicas de cada jardín. Por lo cual, para este proyecto, se considera adecuado profundizar en los sistemas de riego por aspersión debido a su funcionamiento apropiado para áreas donde predomine el césped. Además, de que este sistema tradicional es altamente comercializado y estudiando.

b. Sistemas de riego comerciales

En el mercado, se encuentran sistemas de riego por aspersión y sistemas de riego por aspersión de impacto, ambos ofrecen opciones versátiles para regar áreas extensas de manera uniforme y son adaptables a céspedes, arbustos y flores. Estos sistemas comerciales se detallan en la Tabla 6.

Tabla 6. Sistemas de riego comerciales para jardines [31] [32] [33] [34]

Descripción	Imagen	Material	Costo	Rango	Cobertura	Tipo de sistema
Aspersor Eden 95124 oscilante ajustable de 4 vías		Plástico	29,99	Giro de 360 grados en horizontal y 180 en vertical	Hasta 4,069 pies cuadrados	Manual
Aspersor de impacto		Aleación de Zinc	11,53	Giro de 360 grados en horizontal y arco de agua	Hasta 49.2 pies	Manual
Aspersor oscilante Eden		Plástico	23,99	Giro de 180 grados en vertical	Hasta 3,600 pies cuadrados	Manual
Aspersor rotativo Maxflo		Plástico	17,99	Giro de 360 grados en horizontal	Hasta 33 pies	Manual

Los sistemas comerciales de riego para jardines consisten en uno o varios aspersores que, se conectan a una alimentación de agua. Estos sistemas son de naturaleza manual, lo que significa que requieren la intervención del usuario para ubicarlos en el área deseada, dirigirlos al área de riego ajustando manualmente unas pestañas para determinar la dirección en la que el agua, se dispersará. Para que estos sistemas sean automáticos, se necesita agregar un sistema de control que permita activar las válvulas solo cuando sea necesario. Además, carecen de sensores para evaluar el estado del suelo, por lo cual el riego, se realiza sin considerar las necesidades específicas de cada zona y puede resultar en un desperdicio de agua al extender el riego más allá del área del jardín. No ofrecen un control inteligente de la distribución del agua.

c. Prototipos de Sistemas de riego inteligentes con aspersores

Son los sistemas que utilizan visión artificial y que incorporan tecnología avanzada para optimizar el riego de manera eficiente y precisa. Estos sistemas aprovechan cámaras y algoritmos de procesamiento de imágenes para tomar decisiones informadas sobre cuándo, dónde y cuánto regar. A continuación, se detallan algunos sistemas de riego inteligentes desarrollados en la Tabla 7.

Tabla 7. Sistemas de riego inteligentes para jardines [7] [8] [35] [36]

Sistema	Componentes Físicos	Componentes Electrónicos	Sistema Control	Procesamiento
Carro robot agrícola inteligente	• Estructura de carro. • Mecanismo de deshierbe/ escarda. • Mecanismo de pulverización.	• Cámara. • Sensor de humedad. • Bomba de agua con PWM.	• Control de lógica difusa para accionar la bomba de agua	• Método del umbral adaptativo • Modelo de color HSV • Estimación de umbrales en la segmentación.
Sistema de control inteligente	• Transmisor y receptor inalámbrico • Boscam/SC2000. • Reservorio.	• Válvulas solenoides de 12 V • Cámara Ip Tp-link H.264 1280x1024 (30fps).	• Red neuronal artificial genética (RNA). • Clasificador ANN.	• Matlab R2016a. • Filtro Gaussiano. • Modelo de color HSV • Operador Laplaciano en imágenes segmentadas.
Sistema de riego inteligente con carro robot mecánico	• Estructura de carro • Neumáticos. • Tanque con cañería para spray agua. • Tanque con cañería para spray pesticidas.	• Cámara. • Paneles solares. • Motores. • Arduino Mega 2560. • Puente H L298. • Módulo bluetooth HC-05.	• Sensorización del suelo • Control externo del carro robot • Arduino Mega 2560	• Datos obtenidos de sensores de suelo de ph, nutrientes, y humedad • Aplicación Móvil
Robot de agricultura urbana CityVeg	• Boquilla de riego de 3 grados de libertad. • Marco y rieles de aluminio. • Tanque de agua de 20 litros. • Cámara OV2640 1600x1200.	• RGB Cámara. • Motores a pasos. • Controlador de pasos A4988. • Arduino Mega 2560	• Visión artificial con aprendizaje profundo R-CNN	• Control de micro-stepping. • Microcontrolador Arduino.

Se han desarrollado varios sistemas innovadores para mejorar la agricultura y el riego. Uno utiliza redes neuronales artificiales y visión por computadora para optimizar el crecimiento de plantas en condiciones de escasez de agua. Otro es un sistema de riego inteligente y automatizado con un robot controlado remotamente y alimentado por energía solar. También, se ha creado un sistema robótico económico para la gestión

automática de huertos urbanos, y un sistema de agricultura inteligente que combina visión por computadora y procesos multitarea para desherbar y aplicar riego de tasa variable en campos cultivados. Estos avances buscan aumentar la eficiencia y la sostenibilidad en la agricultura y el cultivo de plantas en la jardinería.

1.3.5 Parámetros para un sistema de riego adecuado en jardines

Un jardín necesita un sistema de riego sobre todo que sea confiable y eficiente que ayude a aumentar el rendimiento y disminuir el uso del agua. Existen diversos parámetros o factores que, se deben considerar para tener un adecuado sistema de riego que, se aplique a los jardines. De acuerdo con la empresa NOVAGRIC los componentes que debe tener un sistema de riego de manera general son equipos de bombeo y fertilización, líneas de conducción y distribución, emisores, válvulas de control y controles electrónicos [30].

Para los sistemas desarrollados con riego por aspersión, se deben considerar los siguientes elementos [37]:

- Grupo de bombeo: Suministra presión y caudal adecuado al sistema.
- Filtración: Es el grado de espesor de filtración relacionado a la calidad del agua, y al aspersor, en específico al tamaño de la boquilla.
- Sistema de abonado
- Red de tuberías
- Aspersores: Factores propios del aspersor para la distribución del agua como son el alcance y el caudal requeridos para la zona, y el tamaño de gota.

Además, se debe tener en consideración varios factores para el diseño de un sistema de riego los cuales, se describen en la Tabla 8.

Tabla 8. Factores generales en el diseño de un sistema de riego [38]

Factores	Descripción
Terreno	Ubicación del suelo, pendientes o desniveles
Pérdidas de carga	Perdida de presión en la tubería
Recursos hídricos	Localización de las fuentes o alimentaciones de agua
Gestión del riego	Gestión manual o automática
Obstáculos	Refiriéndose a obstáculos para la instalación de tuberías y tensión eléctrica

1.3.6 Índice de jardines en el Ecuador

Los jardines domésticos tienen una gran importancia en las sociedades modernas. Al igual que otros espacios verdes urbanos, los jardines privados permiten a los habitantes de las ciudades interactuar con la naturaleza, lo que les aporta considerables beneficios físicos y psicológicos. Ecuador es uno de los 17 países ecológicamente megadiversos del mundo ya que alrededor del 70% de las viviendas de este país tienen espacios con jardines que casi siempre están rodeados de vallas que separan las casas y los jardines de las calles, una práctica que, se sigue por motivos de seguridad [39].

De acuerdo con datos presentados por el INEC del censo de información ambiental económica desarrollado por los Gobiernos Autónomos Descentralizados Municipales, se determinar la cantidad de espacios con jardines en las poblaciones urbanas del Ecuador, teniendo como resultado 13.01 m^2/hab. La OMS recomienda que en una ciudad debe haber por lo menos entre 9 y 15 metros cuadrados por persona de espacios con jardines [40].

En la provincia de Tungurahua, se cumple con esta recomendación de la OMS, ya que se tiene en promedio 10.12 m^2/hab como se muestra en la Figura 5. Al igual que la ciudad de Ambato con un índice de 9.22 metros por persona. Lo que indica que los sistemas de riego pueden tener presencia para el cuidado de las áreas verdes en la ciudad.

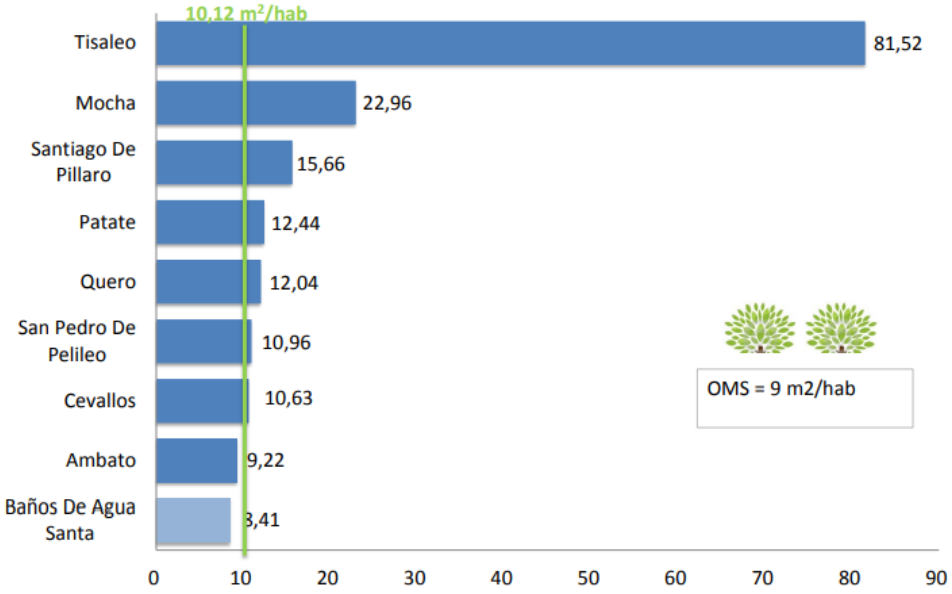


Figura 5. Índice de espacios con jardín en Tungurahua [40]

1.3.7 Dimensiones de un jardín

No existen dimensiones o medidas para un jardín ya que estos pueden variar significativamente según el espacio disponible, las preferencias personales y el propósito del jardín. Sin embargo, en el capítulo IX del Plan de Ordenamiento Territorial de Ambato y la Dirección de Gestión de suelo, se establece los retiros al momento de realizar una construcción, entendiéndose como la distancia entre el lindero y la fachada. Dentro de los retiros frontales establece que, en zonas destinadas para viviendas, estos espacios deben ser ocupados con jardines que favorezcan al ornato. La distancia de estos retiros frontales debe ser por lo menos 5 metros. En lo referente a los retiros laterales y posteriores, se establece que la distancia mínima es de 3 metros [41].

1.3.8 Tipos de césped para jardines

El césped es apreciado globalmente tanto por su estética como por sus múltiples ventajas. Además de su capacidad para embellecer áreas, se utiliza en deportes y recreación, y aporta beneficios medioambientales al reducir la erosión del suelo, mejorar la calidad del aire y proporcionar hábitats para la fauna. Existen una variedad de tipos de césped los cuales, se clasifican de acuerdo al clima.

a. Especies de clima frío

Los tipos de césped que son usados y con adaptación al clima frío son los que, se presentan en la Tabla 9.

Tabla 9. Especies de césped para el clima frío [13]

Césped	Descripción
Pasto azul de Kentucky (Poa pratensis)	Es ampliamente utilizado en jardinería y paisajismo debido a su atractiva apariencia. Este tipo de césped es especialmente popular en campos de golf, jardines ornamentales y es susceptible a enfermedades. Se necesita un suelo bien drenado con un PH de 6.5 a 7 y en temporadas como primavera y otoño la frecuencia de riego debe ser de 1 a 1.5 pulgadas (2.5 a 3.8 cm) por semana. Además, mantener el césped a una altura de corte de 3 pulgadas (7.6 cm) es importante para tener un sistema de raíces más fuerte y resistente.
Raigrases perennes (Lolium sp.)	Conocidas por su capacidad para crecer durante todo el año. Estos pastos son apreciados en la agricultura y la jardinería debido a su resistencia y capacidad para proporcionar una cobertura densa y duradera. No debe cortarse a menos de 18-25 mm caso contario su persistencia es afectada. Además, prefiere suelos bien drenados con un pH cercano a 6.0 a 7.0, la frecuencia de riego debe ser de 1 a 1.5 pulgadas (2.5 a 3.8 cm) por semana.
Festucas	Esta especie es altamente resistente y puede resistir el pisoteo intenso. Se caracteriza por su capacidad de formar grupos densos, prosperar en lugares con poca luz y mantener su rendimiento incluso en zonas deportivas de uso intensivo. Necesitan un PH cercano a 5.5 a 7.0, la altura de corte debe ser de 2 a 3 pulgadas (5.1 a 7.6 cm), la dosis de fertilizante oscila entre 2 y 4 libras (0.9 a 1.8 kg) de nitrógeno por cada 1,000 pies cuadrados (93 metros cuadrados) de césped al año. Pueden crecer en climas fríos con temperaturas de hasta -4°F (-20°C).
Agrostis (Agrostis stoloníferа)	Esta planta es apreciada en la jardinería y en la industria del césped debido a su aspecto atractivo y a su tolerancia al corte frecuente. La dosis de fertilizante está entre 2 a 4 libras (0.9 a 1.8 kg) de nitrógeno, se recomienda regar de 1 a 1.5 pulgadas (2.5 a 3.8 cm) por semana, necesita un PH ligeramente ácido a neutro, en el rango de 6.0 a 7.0, soporta temperaturas de hasta -4°F (-20°C).
Pasto trigo	Es apto para su uso en autopistas, áreas de parques y laderas con suelo de calidad inferior, y es capaz de resistir la sequía y niveles alcalinos del suelo. La tasa de siembra es de 60 a 120 libras (27 a 54 kg) de semillas de pasto trigo por acre (2,471 m^2), se aplican entre 40 y 80 libras (18 a 36 kg) de nitrógeno por acre en función de las condiciones, las semillas, se siembran a una profundidad de 1 a 1.5 pulgadas (2.5 a 3.8 cm) en el suelo, se riega cuando el suelo está seco a una profundidad de aproximadamente 2 a 3 pulgadas (5.1 a 7.6 cm).
Kikuyo (Pennisetum clandestinum)	Este tipo de césped, se establece y restaura mediante el uso de fragmentos cuadrados de césped, conocidos como tepes o chambas. Se siembran de 2 a 3 libras (0.9 a 1.4 kg) de semillas por cada 1,000 pies cuadrados (93 metros cuadrados) o 25 a 30 kilogramos por hectárea, se aplica alrededor de 1 a 2 libras (0.45 a 0.9 kg) de nitrógeno, la humedad del suelo debe estar en el rango de 20% al 60%, es tolerante a la sequía y necesita riego de 1 a 2 veces por semana en promedio.

b. Especies de clima cálido

Las especies de césped más comunes para el clima cálido son las que, se presentan en la Tabla 10.

Tabla 10. Especies de césped para el clima cálido [13]

Césped	Descripción
Pasto Bermuda (Cynodon dactylon)	Adaptable a áreas secas y resistente al pisoteo, esta variedad de césped, se establece en diferentes tipos de suelo. Se reproduce mediante estolones, proporcionando una cobertura eficaz y no requiere mantenimiento especial. Aconsejable sembrar a una densidad de 140 kg/ha, con una vida útil de unos 5-6 años. No es adecuado para lugares sombreados. Se riega cuando el suelo está seco a una profundidad de 2 a 3 pulgadas (5.1 a 7.6 cm), a temperaturas cercanas o por debajo de 32°F (0°C), es probable que el pasto Bermuda sufra daños significativos. Necesita un PH de 5.5 a 8.5.
Césped japonés	Preferible en regiones áridas con suelos bien drenados. Se propaga con mayor éxito vegetativamente. Este césped, que posee hojas diminutas, erguidas y robustas, es ideal para espacios reducidos. Necesita un PH de 6 a 7, la humedad del suelo debe estar en el rango de aproximadamente el 60% al 80%, necesita una frecuencia de riego cada 3 a 5 días durante la temporada de crecimiento activo, su temperatura óptima está en el rango de 75°F a 90°F (24°C a 32°C).
San Agustín (Stenotaphrum secundatum)	Se expande mediante estolones y con reproducción vegetativamente. Sus hojas son amplias y requiere un mayor suministro de microelementos para su desarrollo. Aunque es una opción común para jardines, no se recomienda para campos deportivos que experimenten un uso intensivo. Tolera varios niveles de humedad y en época de crecimiento debe estar entre el 60% y 80%, suele requerir riego cada 3 a 7 días, su temperatura óptima de crecimiento está en el rango de 80°F a 95°F (27°C a 35°C), necesita un PH de 6.5 a 7.5.
Grama misionera (Axonopus compresus)	Tiene una notable tolerancia al pisoteo, la poda, el fuego, la sequía y la saturación de agua. Se adapta de manera versátil a distintas condiciones de suelo. La siembra, se realiza mediante plántulas a una distancia de 60-70 cm entre sí. Necesita un PH de 7 a 8, la humedad del suelo debe estar al 80%, necesita riego de 4 a 7 días en la época de crecimiento.
Maní forrajero (Arachis pintoi)	Pertenece al grupo de las gramíneas perennes y es de naturaleza leguminosa, es ampliamente utilizado en áreas caracterizadas por climas cálidos y húmedos. Destaca por su robustez frente a enfermedades y las plagas comunes del césped, aunque demanda una cantidad significativa de riego para mantener su salud. La humedad del suelo debe estar al 70%, a temperaturas menores a 32 °F (0°C) puede sufrir daños.
Césped filipino	Es habitual hallar este tipo de césped en zonas de climas tropicales y subtropicales. Se caracteriza por su textura suave y fina, además de su crecimiento denso. Sorprendentemente, no demanda un mantenimiento extenso y puede prosperar con un cuidado mínimo. Incluso puede desarrollarse adecuadamente en áreas de sombra parcial. Se recomienda un PH mayor a 6.5 y una temperatura de 30°C.

Específicamente la ciudad de Ambato está ubicada en la provincia de Tungurahua y es una de las 24 provincias que tiene los climas más variados, se encuentra a 2600 metros de altitud. El clima de la ciudad de Ambato es un clima templado cálido que tiene temperaturas que varía desde 9 °C a 25 °C, ya que, se encuentra ubicado en un valle andino [42]. El clima en Ambato, se caracteriza por veranos breves, agradables y con cielos nublados, mientras que los inviernos son también de corta duración, frescos y con algunas nubes en el cielo. Según los datos proporcionados por el Aeropuerto Internacional Cotopaxi, la temperatura registrada en la ciudad de Ambato son los siguientes de la Figura 6.

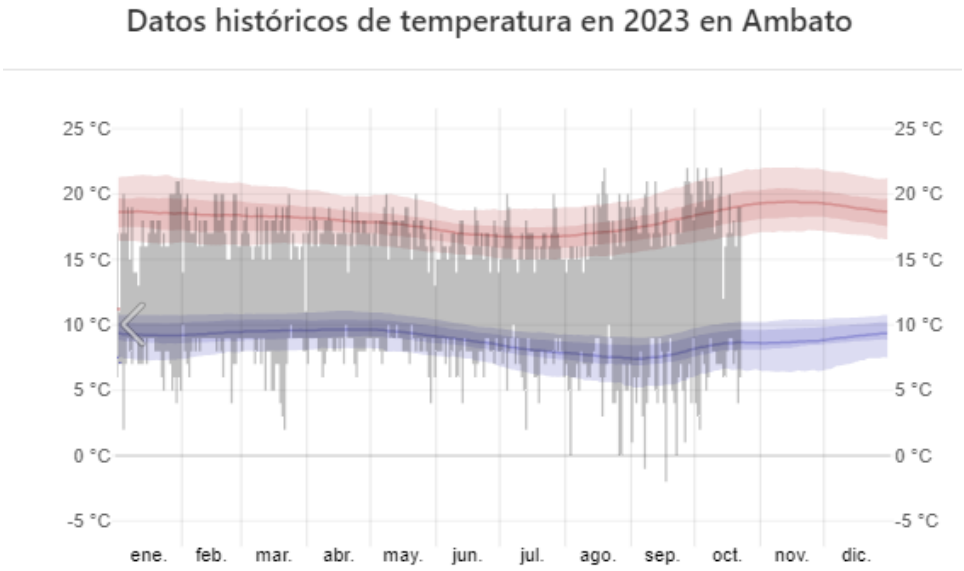


Figura 6. Datos de temperatura de la ciudad de Ambato [43]

Por el clima templado cálido de esta ciudad es común encontrar varios tipos de césped como son los siguientes [13]:

- **Césped Bermuda:** La Bermuda es una elección popular en climas cálidos y puede adaptarse bien a las temperaturas de Ambato. Es resistente y tolerante a la sequía, lo que la hace adecuada para céspedes y campos deportivos.

- **Kikuyo:** El Kikuyo es otra opción que tolera bien el clima subtropical de Ambato. Se utiliza en céspedes y áreas verdes, y es conocido por su capacidad de formar céspedes densos. La profundidad de muestreo de suelo es de 15cm a 20cm por la ubicación de mayor concentración de raíces absorbentes.

- **Césped San Agustín:** El San Agustín es otra opción popular en áreas subtropicales. Es apreciado por su capacidad para formar céspedes densos y su resistencia al calor.

1.3.9 Suelo

El suelo provee varios factores cruciales para mantener y cuidar las plantas, brinda una base estructural, provee nutrientes y agua, además de servir como hábitat para algunas plagas y sus depredadores [13].

a. Textura del suelo

Consiste en el tamaño y la proporción de partículas en el suelo. Los suelos óptimos conocidos como francos, que contienen 20-30% de arcilla, 20% de limo y 50% de arena. Los suelos de textura gruesa, como los arenosos, retienen menos eficientemente agua y nutrientes que los suelos de textura fina, como los arcillosos. La tierra franco equilibrada para el cultivo debe tener entre 6-12% de caliza y un nivel de humus de 4-8%. Los suelos ricos en materia orgánica son fértiles y eficientes en la retención de agua. Es relevante mencionar que algunos tipos de pasto prefieren suelos específicos, ya sea arenosos, arcillosos o francos, mientras que otros no tienen una preferencia clara por la composición física del suelo[13].

b. PH

Se refiere a la evaluación del potencial de hidrógeno, que indica la acidez o alcalinidad, expresado en términos logarítmicos. La acumulación de iones H+ puede deberse tanto a la adición de estos iones como a la pérdida de bases en la solución del suelo. Algunas causas de la acidificación del suelo incluyen el uso de fertilizantes nitrogenados, urea, la extracción de bases (Ca, Mg, Na, K) y el lavado de las bases de intercambio por la lluvia. El pH influye en la disponibilidad de nutrientes y en la actividad de microorganismos; en líneas generales, las gramíneas prosperan en un pH de 5,6 a 6,5 [13].

c. Humedad

Se refiere al contenido de agua, y está mayormente influenciada por sus propiedades físicas, como la textura, la estructura, la porosidad y la densidad aparente. Se utilizan instrumentos como hidrómetros de bloques de yeso y tensiómetros para medir la humedad en el suelo [15]. En la Tabla 11, se detallan los tres estados de humedad del suelo.

Tabla 11. Estados de humedad del suelo [15]

Estados de humedad	Descripción
Saturación	El agua llena todos los poros del suelo, desplazando completamente el aire en estos espacios. Se presenta durante inundaciones permanentes o justo después de lluvias intensas o riegos abundantes, cuando la humedad del suelo alcanza el 100%. El exceso de agua, se drena hacia abajo por gravedad, el suelo tiene un potencial de retención de agua con una presión de 0 atmósferas.
Capacidad de campo (CC)	El suelo retiene la máxima cantidad de agua en los microporos, y recupera espacio de aire en los macroporos. El suelo está completamente húmedo, pero no saturado. Presenta un potencial de retención de 0,3 atmósferas en suelos francos, 0,5 en suelos arcillosos y 0,1 en suelos arenosos. El agua permanece en el suelo después de drenar casi todo el exceso de agua gravitacional.
Marchitez permanente (PM)	El agua retenida es insuficiente para las raíces de las plantas, lo que provoca un marchitamiento irreversible de las hojas. La escasa cantidad de agua remanente en el suelo es retenida en los microporos por una fuerza de succión mayor a la capacidad de absorción de las raíces de la planta. La humedad en el suelo puede alcanzar este extremo cuando el agua, se pierde por evapotranspiración y no se repone mediante riego o lluvia.

- ***Humedad Disponible***

Es el contenido de humedad que el suelo puede retener, situado entre los extremos de "capacidad de campo" y "punto de marchitez". Es crucial considerar que la reposición de agua en el suelo, a través de riego o lluvias, requiere un tiempo específico, para ello la velocidad de infiltración determina la duración y la intensidad del riego, siendo un parámetro esencial en el diseño del sistema [15]. En la Tabla 12, se presentan los valores de algunas de las propiedades físicas del suelo en base a su textura.

Tabla 12. Valores promedios de algunas propiedades físicas de los suelos según la textura y su contenido de humedad aprovechable [44]

Textura	Velocidad de infiltración (mm/h)	Capacidad de campo (2 - %)	Punto de marchitez (3 - %)	Humedad disponible (mm/m)
Arenoso	50 (25 o más)	9 (6 - 14)	4 (2 - 6)	70 - 100
Franco arenoso	25 (13 - 40)	14 (10 - 18)	6 (4 - 8)	90 – 150
Franco	13 (7 - 20)	22 (18 - 26)	10 (8 - 12)	140 – 190
Franco arcilloso	8 (2 - 15)	27 (23 - 31)	13 (11 - 15)	170 – 220
Arcilloso	0.5 (0.1 - 1)	35 (31 - 39)	17 (15 - 19)	200 - 250

d. Sensores de humedad

Se emplean diversas técnicas con sensores de humedad para evaluar el estado de humedad del suelo. Se clasifican en dos tipos según cómo indican el contenido de agua: los que miden la tensión o succión que retiene el agua en el suelo, y los que miden el contenido total de humedad en el suelo, expresado en porcentaje volumétrico [45].

- ***Higrómetro***

Un higrómetro es un dispositivo que mide la humedad en el aire, el suelo y las plantas. Se debe considerar que la humedad representa la cantidad de vapor de agua en el entorno. La saturación de la humedad requiere que la temperatura ambiente sea más baja. En este proceso, el vapor de agua en el aire, se condensa, dando lugar al rocío. Existen diversos tipos de higrómetros según su funcionamiento y características particulares [46].

- ***Equipos TDR***

Dispositivos electrónicos con una unidad de control que almacena las mediciones de humedad. La unidad está conectada a varillas de acero inoxidable insertadas en el suelo. Realiza mediciones del contenido volumétrico del suelo, proporcionando valores a lo largo de las varillas. Fácil de instalar, mediciones instantáneas y mediciones simultáneas en el mismo lugar. Aunque son eficaces, tienen un costo elevado, requieren contacto directo de las varillas con el suelo, no son adecuados para

suelos pedregosos y pueden tener problemas en suelos con altas concentraciones de sales y materia orgánica [45].

- ***Sonda de capacitancia FDR***

Compuesto por una unidad controladora que almacena y transmite datos; una sonda que mide humedad; y un tubo de acceso que facilita la inserción de la sonda en el suelo. Este equipo evalúa la humedad volumétrica en varias profundidades. Durante la medición, la sonda equipada con sensores a varias distancias marca la humedad en cada anillo. La instalación presenta dificultades para mantener un buen contacto con el suelo, y su costo es elevado, con menor precisión en suelos de textura fina [45].

- ***Tensiómetros***

Este dispositivo reacciona ante las variaciones en la tensión de humedad del suelo, operando a través de la fuerza de absorción del suelo. Está compuesto por un medidor de vacío y un tubo sellado que contiene una capa de cerámica con poros, simulando el movimiento del agua en el suelo. A medida que el suelo, se va secando, la lectura del tensiómetro aumenta. La interpretación de estas lecturas depende del cultivo, del tipo de suelo y de la curva de humedad correlacionada. En términos generales, una lectura de 0 a 10 Cb indica saturación del suelo; de 10 a 20 Cb, el suelo, se encuentra en capacidad de campo; y de 30 a 60 Cb, sugiere que el suelo está seco, recomendándose realizar un riego de inmediato [45].

- ***Sensores de resistencia eléctrica (bloque poroso o yeso)***

Este dispositivo usa dos electrodos en paralelo en un bloque de yeso con poros (CaSO4), conectados a la superficie mediante un cable, se mide la resistencia con un óhmetro portátil. La resistencia varía con la cantidad de agua entre los electrodos, indicando la tensión de agua en el suelo. En suelos húmedos, la resistencia es baja. Sin embargo, estos dispositivos comienzan a perder agua alrededor de los 30 Cb debido a las propiedades del bloque [45].

1.3.10 Procesamiento de imágenes

El procesamiento de imágenes, se refiere al conjunto de técnicas y procedimientos empleados para descubrir o resaltar información en una imagen, utilizando principalmente una computadora como herramienta [47].

a. Reconocimiento de patrones

El reconocimiento de patrones involucra el procesamiento de imágenes obtenidas mediante la deconvolución. En esta etapa, se ajustan los niveles de gris para mejorar la legibilidad y descubrir características que puedan tener relevancia física. Este ajuste, se realiza a través de tres subprocesos: realce de imagen, eliminación de ruido, segmentación y detección de bordes. El realce de imagen busca mejorar la calidad visual de la imagen para una mejor percepción [47].

b. Detección de discontinuidades

La información crucial, se encuentra mayormente codificada en las regiones donde, se producen cambios bruscos o discontinuidades. Experiencias que involucran la reconstrucción de imágenes al considerar combinaciones de la fase y la magnitud de diversas fuentes han revelado que, además de la fase que lleva esta información, es viable reconocer el contenido de la imagen solo al examinar estas discontinuidades. Esto fundamenta la existencia de numerosos algoritmos propuestos para detectar estas discontinuidades, tanto los ya establecidos como los más recientes. Estas discontinuidades, se pueden dividir en tres categorías principales: bordes, líneas y esquinas [48].

1.3.11 Visión artificial

La visión artificial, se dedica al procesamiento y análisis de imágenes del mundo real para que puedan ser interpretadas y manipuladas por un ordenador. El componente esencial en la visión artificial es el píxel, que representa cada unidad de una imagen digital. Esta disciplina involucra una serie de operaciones matemáticas aplicadas a los píxeles de la imagen, conocidas como filtros, que varían según el objetivo específico en cada situación, estos filtros, se aplican mediante máscaras [49]. A continuación, se

presenta en la Figura 7 los pasos fundamentales para llevar a cabo una tarea de visión artificial.

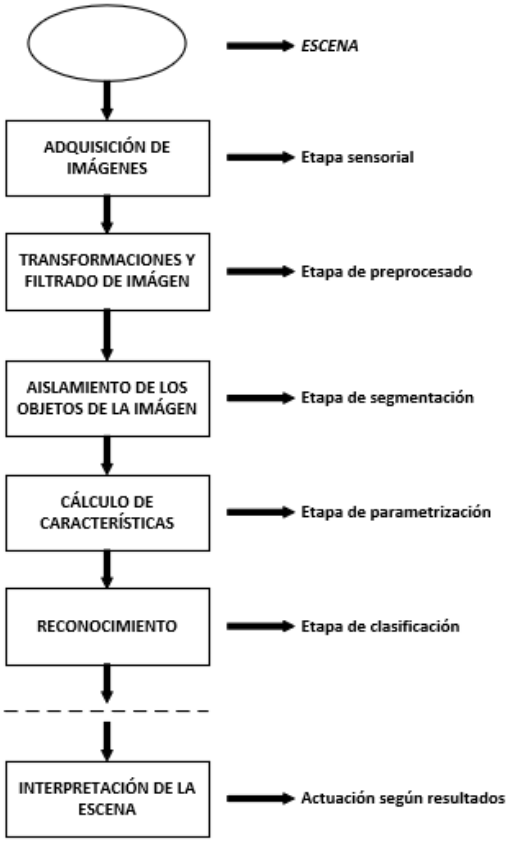


Figura 7. Etapas de un sistema de visión artificial [50].

El proceso inicia al capturar una imagen mediante sensores y convertir su señal en formato digital. Tras obtener la imagen digitalizada, se procede al preprocesamiento, con el fin de mejorar la calidad de la imagen y aumentar las posibilidades de éxito en el objetivo final.

El paso siguiente implica la segmentación, cuyo propósito es dividir la imagen en componentes individuales u objetos que la conforman. La segmentación autónoma plantea un desafío considerable en el procesamiento de imágenes, ya que la precisión en este paso es crucial para el éxito, y errores pueden comprometer el resultado.

A continuación, se lleva a cabo la parametrización o selección de rasgos, se centra en extraer características cuantitativas significativas o elementos esenciales que permiten diferenciar entre diversas clases de objetos.

Finalmente, se efectúan el reconocimiento y la interpretación. El reconocimiento asigna etiquetas a los objetos basándose en los descriptores (clasificación), mientras que la interpretación otorga significado al conjunto de objetos identificados. Estos pasos en conjunto constituyen un proceso integral en el análisis de imágenes digitales [50].

a. Detección de objetos con YOLO

La identificación y detección de objetos representa una tarea esencial en visión por computadora, la cual implica determinar la ubicación y clasificación de ciertos objetos en una imagen. En 2015, surgió el algoritmo YOLO (You Only Look Once), marcando un nuevo enfoque al abordar la detección de objetos como un problema de regresión y ejecutándolo en una sola red neuronal. Este cambio transformó significativamente el campo de la detección de objetos, logrando avances notables en comparación con la última década [51].

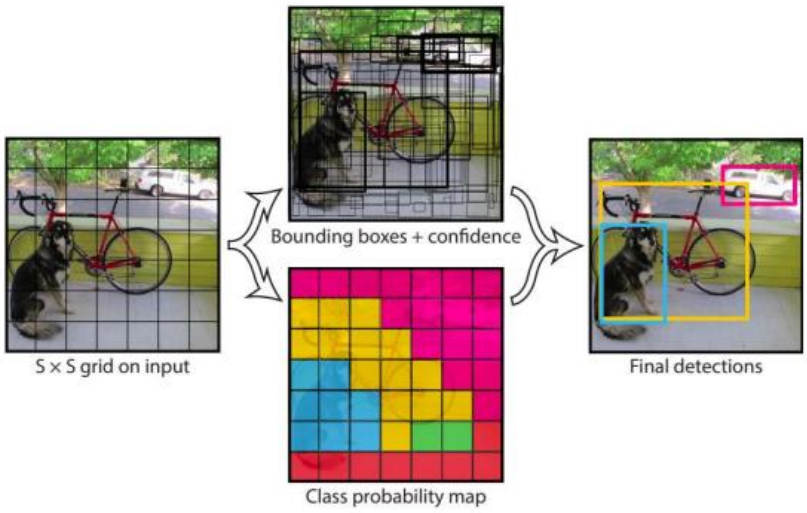


Figura 8. Modelo de funcionamiento inicial de YOLO [51]

La Figura 8 representa el modelo de funcionamiento de YOLO donde, se genera inicialmente cuadros delimitadores de propuestas en la imagen de entrada, posteriormente emplea un clasificador en esos cuadros y finalmente realiza un proceso posterior para eliminar detecciones duplicadas y mejorar la precisión de los cuadros delimitadores [51].

b. YOLOv5

YOLO ha experimentado varias actualizaciones, convirtiéndose en uno de los algoritmos de detección de objetos más destacados, gracias a la combinación de ideas innovadoras provenientes de la comunidad de investigación en visión por computadora. La versión, YOLOv5, aunque no fue desarrollada por el autor original de YOLO, supera a YOLOv4 tanto en precisión como en velocidad. YOLOv5, está programado en lenguaje Python en lugar de lenguaje C que es el lenguaje utilizado en versiones anteriores. Este cambio facilita su instalación e integración en dispositivos de Internet de las cosas [51].

c. Google Collaboratory

Presenta un Entorno de Desarrollo Integrado (IDE) sin costo desarrollado por Google con el propósito de respaldar la investigación y el aprendizaje en el ámbito de la Inteligencia Artificial (IA). Collaboratory proporciona un entorno de código similar a Jupyter Notebook y ofrece la libertad de utilizar Unidades de Procesamiento Gráfico (GPU) y Unidades de Procesamiento Tensorial (TPU). Además, Google Colab incluye bibliotecas preinstaladas, como PyTorch, TensorFlow, Keras y OpenCV, que son altamente reconocidas en el ámbito de la investigación en Aprendizaje Profundo [51].

1.3.12 Visión artificial en la jardinería

La visión artificial aplicada a la jardinería es una tecnología que utiliza sistemas de cámaras y software de procesamiento de imágenes para analizar y optimizar diversos aspectos del cuidado de jardines y áreas verdes. A continuación, en la Tabla 13, se presenta algunas aplicaciones de la visión artificial en la jardinería.

Tabla 13. Aplicaciones de la visión artificial en la jardinería [52]

Aplicación	Descripción
Detección de plagas y enfermedades	Las cámaras de visión artificial pueden vigilar las plantas en busca de signos visuales de plagas o enfermedades y, mediante algoritmos de procesamiento de imágenes, detectar cambios en color, forma o textura en las hojas para alertar a los responsables del jardín sobre problemas fitosanitarios.
Clasificación de plantas y flores	La visión artificial permite reconocer y categorizar plantas y flores en un jardín, lo que facilita el seguimiento de las especies, la identificación de cuidados específicos y la creación de un catálogo digital de las plantas en el jardín.
Riego inteligente	Los sistemas de visión artificial ofrecen mediciones en tiempo real de la humedad del suelo y las condiciones ambientales. Esto permite la adaptación automática del riego para suministrar la cantidad precisa de agua que cada área del jardín requiere, reduciendo el desperdicio de agua.
Mantenimiento de césped	La visión artificial puede analizar la salud del césped y detectar áreas con carencias de hierba o presencia de malas hierbas. Los robots cortacésped equipados con cámaras de visión artificial pueden recorrer el jardín de forma autónoma, cortando el césped solo en las zonas que requieren atención, lo que optimiza el mantenimiento del jardín.
Diseño de jardines	La visión artificial simplifica la planificación y diseño de jardines al proporcionar recursos de realidad aumentada. Estos recursos posibilitan a los diseñadores anticipar cómo lucirán las plantas y las particularidades del jardín antes de su implementación.
Estimación de cosechas	En el caso de jardines que incluyen cultivos, la visión artificial puede estimar el rendimiento de las cosechas y ayudar en la planificación de la cosecha.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema de riego inteligente de corto alcance para jardines partir de visión artificial

1.4.2 Objetivos específicos

- Analizar las características de los sistemas de riego tradicionales para jardines
- Determinar los componentes electrónicos y software necesarios para el sistema de riego inteligente

- Aplicar algoritmo de detección de superficie para el riego mediante visión artificial
- Diseñar una aplicación móvil para el monitoreo del sistema de riego inteligente
- Evaluar el desempeño del sistema de riego inteligente mediante pruebas de funcionamiento

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

El proyecto, se enfocó en el desarrollo de un sistema de riego inteligente con tecnología Wi-Fi el cual, tiene la capacidad de identificar el área de jardín a través del uso de YOLOv5 y algoritmos en los cuales, se detecta el área verde más grande para realizar el riego. Para el riego, se presenta un control de servomotores los cuales, se encargan de direccionar un aspersor hacia el área de jardín detectada. Además, cuenta con la verificación de sensores de humedad y pronostico climático asegurando así un riego eficiente. A continuación, en la Tabla 14, se describen algunos de los materiales que usados.

Tabla 14. Materiales

Materiales	Características	Descripción de uso
ESP32-CAM	Cámara	Cámara que permite la obtención de imagen del jardín
FC-28	Sensor de humedad del suelo	Obtiene datos de humedad del suelo para la toma de decisiones de riego
TD-8120MG	Servomotores de 20 kg	Direccionan el aspersor para cubrir toda el área de verde de jardín
Válvula solenoide	Válvula solenoide de 12V para ½ pulgada	Permite el paso de agua desde la toma hacia el sistema inteligente de riego
Módulo de relé	Módulo de relé de 12V de 1 canal	Puente de enlace y control de la válvula solenoide y el microcontrolador
ESP32 WROOM 32	Microcontrolador	Realiza el control de actuadores de acuerdo con el algoritmo y datos de humedad

2.2 Métodos

2.2.1 Modalidad de la investigación

El presente proyecto de investigación, se contempló como una investigación de tipo aplicada, debido a que, se recurrió a los conocimientos adquiridos en el transcurso de la formación académica para aplicarlos en el diseño e implementación de un sistema inteligente de riego para un jardín.

Para la recolección de información, se empleó la investigación bibliográfica, debido a que, se sustentó mediante artículos científicos, libros, bases de datos de distintas universidades, e información web. Este enfoque permitió adquirir conocimientos sobre los requerimientos y especificaciones necesarios para orientar de manera adecuada el proyecto de investigación.

Se realizó investigación de campo, en un jardín, con el fin de obtener datos reales y sin ningún tipo de manipulación. Con ello, se logró que la investigación tenga mayor valor y que el sistema inteligente propuesto funciones correctamente en condiciones reales.

Además, se llevó a cabo una investigación experimental, en la cual se realizaron diversas pruebas para obtener datos relevantes en el jardín. Estos datos desempeñan un papel crucial en el correcto desarrollo del control de riego y permitieron que el sistema tome decisiones sin intervención del usuario.

2.2.2 Recolección de información

Se procedió mediante un plan de recolección de información, el cual incluyo la selección de criterios los cuales sustentaron los requerimientos del sistema y la utilización de herramientas adecuadas para la recopilación de datos.

Tabla 15. Criterios para la recolección de información

Criterio	Descripción
¿Para qué?	Para determinar los parámetros relacionados al control eficiente de un sistema de riego aplicado a jardines
¿De qué personas u objetos?	De jardines en viviendas
¿Cómo?	Determinando el área de jardín
¿Con qué?	Con un sistema inteligente que usa visión artificial para reconocer el área
¿Cuándo?	Período académico septiembre 2023 – febrero 2024
¿Dónde?	En un jardín de una vivienda ubicada en la ciudad de Ambato

2.2.3 Procesamiento y análisis de datos

Para el procesamiento y análisis de datos, se tomó en cuenta puntos importantes del desarrollo, los cuales fueron:

- Interpretación y optimización de la información
- Análisis de los dispositivos para la adquisición de señales
- Estudio de los sistemas de control y monitoreo previamente realizados
- Presentación de resultados conforme a los objetivos propuestos

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

3.1 Requerimientos del sistema

Un sistema de riego inteligente para jardines representa una solución avanzada que supera la simple aplicación de agua en un horario regular, al añadir un reconocimiento del área verde para su riego eficiente. Para garantizar un uso eficiente del agua y proporcionar un cuidado óptimo a las plantas, es esencial que este sistema incorpore diversas características y cumpla con los siguientes requisitos:

- Portabilidad para transportar fácilmente el sistema de un jardín a otro.
- Inalámbrico para no depender de conexiones físicas en la transmisión de datos
- Adquisición de valores de humedad del suelo
- Detección de límites del jardín
- Delimitación del área de riego para realizar un riego selectivo
- Programación del sistema de riego
- Aplicación móvil para el monitoreo

3.2 Esquema general del sistema

El funcionamiento del sistema de riego inteligente, se describe en cuatro etapas, las cuales están representadas en la Figura 9. La primera etapa es la de adquisición, en la cual, se obtienen los datos externos, como la imagen del jardín y los valores de humedad. En la segunda etapa, se realiza el procesamiento de los datos anteriores, donde el algoritmo determina el área de jardín y la ruta de riego. En la tercera etapa de almacenamiento, se guardan los datos anteriores y genera un registro para el riego. En la etapa final de visualización y monitoreo, se verifica el área de riego y, se genera una gráfica de progreso de la humedad del jardín.

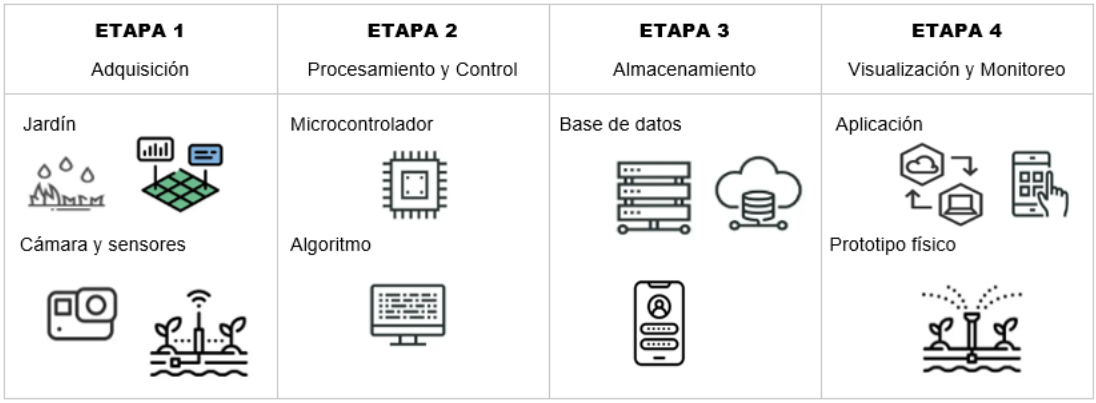


Figura 9. Esquema general del sistema

Etapa 1: El sistema inicia con la adquisición en tiempo real de imágenes mediante una cámara incorporada (ESP32-CAM) para capturar las imágenes del jardín, y utiliza un sensor de humedad para obtener los datos de humedad del suelo, todos estos valores adquiridos son procesados en la siguiente etapa.

Etapa 2: Los datos adquiridos son procesados para la toma de decisiones con el algoritmo, una vez establecidas las acciones, se envía la información al microcontrolador (ESP32) para controlar los actuadores y realizar el riego.

Etapa 3: Para su implementación y aplicabilidad, se desplegó un entorno de servicio de hosting local, que permite guardar y compartir los datos del usuario al igual que los datos obtenidos en las etapas anteriores.

Etapa 4: La aplicación presenta los datos de los sensores y de la cámara para una precisa manipulación de los actuadores del prototipo físico. La interfaz permite al usuario, realizar el monitoreo del riego y controlarlo en un entorno sencillo e intuitivo.

3.2.1 Selección de dispositivos

a. Sensor de humedad del suelo

Se requiere obtener valores de humedad del suelo con precisión por lo cual, se requiere un umbral de mediciones analógicas. En base a la comparativa realizada en la Tabla 16, se seleccionó el sensor FC-28, debido a su disponibilidad en el mercado ecuatoriano. Además, a diferencia del sensor HD-38, el sensor FC-28 tiene un costo

menor y presenta varias de las mismas características técnicas. Revisar el Anexo C para conocer las características técnicas del integrado de acondicionamiento de señal del sensor FC-28.

Tabla 16. Tabla comparativa de sensores de humedad del suelo [53] [54] [55]

Características técnicas	Sensor capacitivo v1.2	HD-38 Higrómetro	FC-28 Higrómetro
Tipo de sensor	Analógico	Analógico/Digital	Analógico/Digital
Voltaje de alimentación	3.3V a 5V en DC	3.3V a 12V en DC	3.3V a 5V en DC
Dimensiones	98x23 mm	36x15 mm	60x20x5 mm
Corriente de operación	5mA	<30mA	35mA
Voltaje de salida	0 a 5V en DC	0 a 5V en AO 3.3V/5V TTL en DO	0 a 5V en AO 3.3V/5V TTL en DO
Disponibilidad	Baja	Baja	Alta
Costo	$4	$10	$3

El sensor opera midiendo la resistencia entre dos electrodos los cuales son insertados directamente en el suelo, siendo esta resistencia dependiente de la humedad del suelo. En condiciones de elevada humedad, la resistencia es mínima, equivalente a un cortocircuito, mientras que, en suelos secos, la resistencia es elevada, equivalente a un circuito abierto. Los resultados de dicho electrodo, se transmiten a una tarjeta de acondicionamiento (YL-38). Es compatible con plataformas como Arduino, PIC, NodeMCU [54].

b. Microcontrolador

En base a la Tabla 17, se selecciona la Nodemcu ESP32 WROOM 32 debido a su capacidad de memoria RAM y velocidad de CPU lo cual permitirá un procesamiento rápido y mejor toma de decisiones en el sistema, además de que su comunicación de WiFi permite interactuar y compartir información con las otras etapas del sistema. Revisar el Anexo A para verificar todas sus características técnicas.

Tabla 17. Tabla comparativa de microcontroladores [56] [57] [58]

Características técnicas	ESP32 WROOM 32	ESP8266	Arduino UNO
Procesador	Xtensa 32bits LX6	Tensilica L106 32-bit	ATMega328P
Dimensiones	51x23 mm	49x26 mm	68x53 mm
Flash	4 MB	4 MB	32 KB
Memoria RAM	520 KB	50 KB	2 KB
Velocidad del CPU	80MHz – 240 MHz	160 MHz	16 MHz
WiFi	802.11 b/g/n	802.11 b/g/n	No
Costo	$14	$14	$16

c. Cámara

La Cámara OV2640 con ESP32–CAM fue seleccionada en base a la comparativa de la Tabla 18 por sus características de resolución de imagen, interfaz de control y fotogramas por segundo. El módulo ESP32-CAM permite compartir y controlar él envió de datos a través de wifi. En el Anexo B, se tienen las características técnicas de la ESP32-CAM

Tabla 18. Tabla comparativa de cámaras [59] [60]

Características técnicas	Cámara OV2640 con ESP32–CAM	Cámara web Genius 1000x V2 black
Fabricante	OmniVision	Genius
Dimensiones	39,80x27 mm	20x22 mm
Resolución de imagen	2MP, rango dentro de 1632x1220 pixeles	1MP, 1280 x 720, 640 x 480 píxeles
Interfaz	ESP32-CAM	USB
Angulo de visión	25 grados no lineales	90 grados de arriba a abajo
Fotogramas por segundo	15 a 60 fps	30 fps
Costo	$13	$18

La placa ESP32-CAM, se puede usar como cámara web, es de menor tamaño a una ESP 32 regular, cuenta con los beneficios de comunicación wifi y bluetooth como las placas de la serie ESP 32, posee 2 CPUs LX6 de 32 bits, además de una cámara integrada. [61]

d. Servomotor

A partir de la Tabla 19, se determina que es necesario emplear dos servomotores TD-8120MG. Esto, se debe a que, se necesita un torque elevado para accionar el mecanismo y resistir la fuerza generada por la presión del agua. Por esta misma razón,

se especifica la necesidad de piñones metálicos en el servomotor. Además, se establece que el ángulo rotacional no debe exceder los 180 grados, ya que, no se requiere una movilidad mayor. En el Anexo D, se tienen más características técnicas de este servomotor.

Tabla 19. Tabla comparativa de servomotores [62] [63] [64]

Características técnicas	TD-8120MG	MG995	MG996R
Tipo de engranajes	Metal	Metal	Metal
Fabricante	Tiankongrc	Tower Pro	Tower Pro
Dimensiones	40,7 × 19,7 × 42,9 mm	40,7 × 19,7 × 42,9 mm	40,6 × 19,8 × 42,9 mm
Voltaje de operación	4,8 a 7,2 V	4,8 a 6 V	4,8 a 6 V
Angulo rotacional	180	360	180
Velocidad de operación	0.18sec / 60degree (4.8v)	0.20sec / 60degree (4.8v)	0.2sec / 60degree (4.8v)
Torque	20 KG / cm	11 KG / cm	13 KG / cm
Costo	$20	$15	$12

e. Válvula solenoide

En la Tabla 20, se realiza la comparación de las válvulas solenoides, generalmente las tomas de agua en las viviendas son de ½ pulgada y ¾ de pulgada, se selecciona la válvula solenoide de ½ pulgada. La válvula de ½ pulgada es una de las más utilizadas y comparte características similares con la válvula de ¾ de pulgada, a excepción de que está otra es menos costosa. El Anexo E contiene más de las características técnicas de esta válvula solenoide.

Tabla 20. Tabla comparativa de válvula solenoide [65] [66]

Características técnicas	VÁLVULA SOLENOIDE ELECTROMAGNÉTICA 12VDC 1/2PLG NC PARA AGUA	VÁLVULA SOLENOIDE ELECTROMAGNÉTICA 12VDC 3/4PLG NC PARA AGUA
Alimentación	12 V en DC	12 V en DC
Modo de operación	Normal cerrado	Normal cerrado
Puerto de entrada/salida	1/2” GB rosca de tornillo	3/4” GB rosca de tornillo
Tiempo de respuesta	Encendido ≤ 0,15 s Apagado ≤ 0,3 s	Encendido ≤ 0,15 s Apagado ≤ 0,3 s
Presión	0.02 a 0.8 MPa	0.02 a 0.8 MPa
Estimado de vida	500.000 ciclos	500.000 ciclos
Costo	$12	$15

f. Módulo Relé

En base a la comparativa de la Tabla 21, se selecciona el módulo relé 5V de 1 canal con optoacoplador, debido a su aislador, la compatibilidad con la señal de entrada de 5V en corriente continua y su disponibilidad de conexión con borneras en entrada y salida que bridan mayor seguridad al evitar cables sueltos. El Anexo F contiene mayor información de las características técnicas de este módulo relé.

Tabla 21. Tabla comparativa de los módulos relé [67] [68]

Características técnicas	MÓDULO RELÉ 5V KY-019 RELÉ 5V 1 CANAL	MÓDULO RELÉ 5V 1 CANAL CON OPTOACOPLADOR HIGH/LOW ALTO/BAJO
Voltaje de alimentación	5 V en DC	5 V en DC
Señal de entrada	3. 5 a 12V en DC	3.3 a 5V en DC
Voltaje de salida	220V AC	250VAC 10A o 30VDC 10A
Aislador	No	Optoacoplador
Indicador	Si	Si
Compatibilidad	Arduino	Arduino, Raspberry Pi, PIC
Conexión	Borneras en entrada	Bornera en entrada y salida
Costo	$2	$2,50

3.2.2 Selección de tecnologías

a. Tecnología inalámbrica

En base a la comparación realizada en la Tabla 22, la comunicación inalámbrica del sistema será con tecnología Wi-Fi, debido a que ofrece mejor velocidad de transmisión, comunicación simultanea entre los diferentes dispositivos y mayor alcance. Esto le permite al sistema una comunicación rápida entre sus dispositivos y una actualización de información más eficiente.

Tabla 22. Tabla comparativa de tecnologías inalámbricas [69] [70]

Características técnicas	Wi-Fi	Bluetooth
Estándar	IEEE802.11 a/b/g	IEEE802 .15. 1
Velocidad de transmisión	hasta 1 Gbps	hasta 2,1 Mbps
Frecuencia	2,4 GHz, 5 GHz	2,4 GHz
Comunicación	Multipunto	Punto a punto
Ancho de banda	22 MHz	1 MHz
Alcance	150 pies	30 pies
Costo	Medio	Alto

b. Base de datos

En la Tabla 23, se tiene la comparación de los sistemas gestores de bases de datos. Se decidió usar MySQL en lugar de MariaDB y PostgreSQL, ya que, se tiene familiaridad con MySQL, afinidad con proyectos pequeños y medianos, y su robusta comunidad de usuarios contiene foros de documentación y ayuda. Además, MySQL ha demostrado consistentemente su rendimiento eficiente y confiable, lo que es esencial para satisfacer las necesidades del proyecto.

Tabla 23. Tabla comparativa de sistemas gestores de bases de datos [71] [72]

Características técnicas	**MySQL**	**MariaDB**	**PostgreSQL**
Tipo	RDBMS	RDBMS	RDBMS
Facilidad de uso	Sencillo	Sencillo	Complejo
Sistemas Operativos	Canonical, Windows, FreeBSD, Solaris, Linux y MacOS	Windows, Linux y MacOS	OpenBSD, FreeBSD, MacOS, Linux y Windows
Velocidad	Rápido	Rápido	Media
Proyectos	Pequeños y medianos	Pequeños	Grandes
Comunidad	Grande	Pequeña	Pequeña
Licencia	Dual: GNU GPL v2 y propietaria	Abierta: GNU GPL v2	Abierta: Licencia PostgreSQL

En base a la comparación de la Tabla 24, se decidió emplear el entorno de desarrollo local XAMPP, por la versatilidad y eficiencia que ofrece XAMPP para configurar un servidor en diversos sistemas operativos. XAMPP proporciona una instalación sencilla, junto con una amplia compatibilidad para varios servicios. La preferencia por XAMPP, se debe a su facilidad de uso, soporte robusto y a la comunidad activa que respalda este entorno, convirtiéndolo en una opción confiable para el desarrollo web en un entorno local.

Tabla 24. Tabla comparativa de entornos de desarrollo local [73]

Características técnicas	**XAMPP**	**MAMP**	**WAMP**
Sistema operativo	Windows, macOS y Linux	MacOS	Windows
Instalación	Sencilla	Sencilla	Sencilla
Servicios	Apache, MySQL, PHP y más	Apache, MySQL y PHP	Apache, MySQL y PHP
Interfaz	Técnica	Intuitiva	Intuitiva
Licencia	Gratuita	Gratuita y de pago	Gratuita

phpMyAdmin es una herramienta esencial para aquellos que desarrollan aplicaciones web con XAMPP y necesitan interactuar con bases de datos MySQL de manera efectiva. Permite ejecutar consultas SQL, importar y exportar datos, y administrar la estructura de la base de datos de manera eficiente. Es esencialmente un panel de control que facilita la interacción con MySQL sin la necesidad de comandos de línea.

c. Interfaz para la aplicación de monitoreo

De acuerdo con la comparativa realizada en la Tabla 25, se selecciona Android Studio por su entorno de desarrollo donde, se puede diseñar APK para diferentes versiones de Android. La plataforma cuenta con diversas plantillas y herramientas que facilitan la creación de las aplicaciones. Además, posee un sistema de sugerencias para prevenir errores y permite generar aplicaciones de menor tamaño en comparación con Flutter Flow y React Native.

Tabla 25. Tabla comparativa de desarrolladores de aplicaciones móviles [74] [75]

Características técnicas	**Flutter Flow**	**Android Studio**	**React Native**
Desarrollador	Google	Google	Facebook
Categoría	Desarrollo móvil multiplataforma	Entorno de desarrollo integrado	Desarrollo móvil multiplataforma
Ejecución	Multiplataforma	Multiplataforma	Multiplataforma
Dispositivos	iOS y Android	Android varias versiones APK	iOS, Android y la web.
Uso	Sencillo	Sencillo	Sencillo
Herramientas	Widgets Personalizables	Plantillas y herramientas de prueba	Módulos personalizados
Comunidad	Grande	Grande	Grande
Rendimiento	Alto	Alto	Alto
Ventaja	Hot Reload: Visualización de aplicación	Sugerencia de corrección de código. Visualización de aplicación	Visualización de aplicación
Tamaño de aplicación desarrollada	Grande	Pequeño	Grande
Lenguaje	Dart	Kotlin	React y JavaScript

3.3 Diagrama general

En el diagrama general del sistema, inicialmente, se tiene dos procesos: el primero, que, se desarrolla en Python y la base de datos, y el segundo proceso que, se desarrolla dentro de la ESP32 para el control de los servomotores.

En la Figura 10, se muestra el flujograma que abarca las etapas de registro e inicio de sesión. Ambas, se desarrollan utilizando datos generales del usuario, como: el nombre de usuario, el correo electrónico y la contraseña. Una vez que el usuario accede a la interfaz, se realiza una solicitud de los parámetros de humedad de suelo y climáticos. Los parámetros climáticos consisten en el tiempo actual, la humedad ambiental, la probabilidad de lluvia y la velocidad del viento. Estos últimos parámetros son fundamentales para el proceso de riego por aspersión. Se aconseja no llevar a cabo el riego si hay una alta probabilidad de lluvia o si la velocidad del viento supera los 12 km/h.

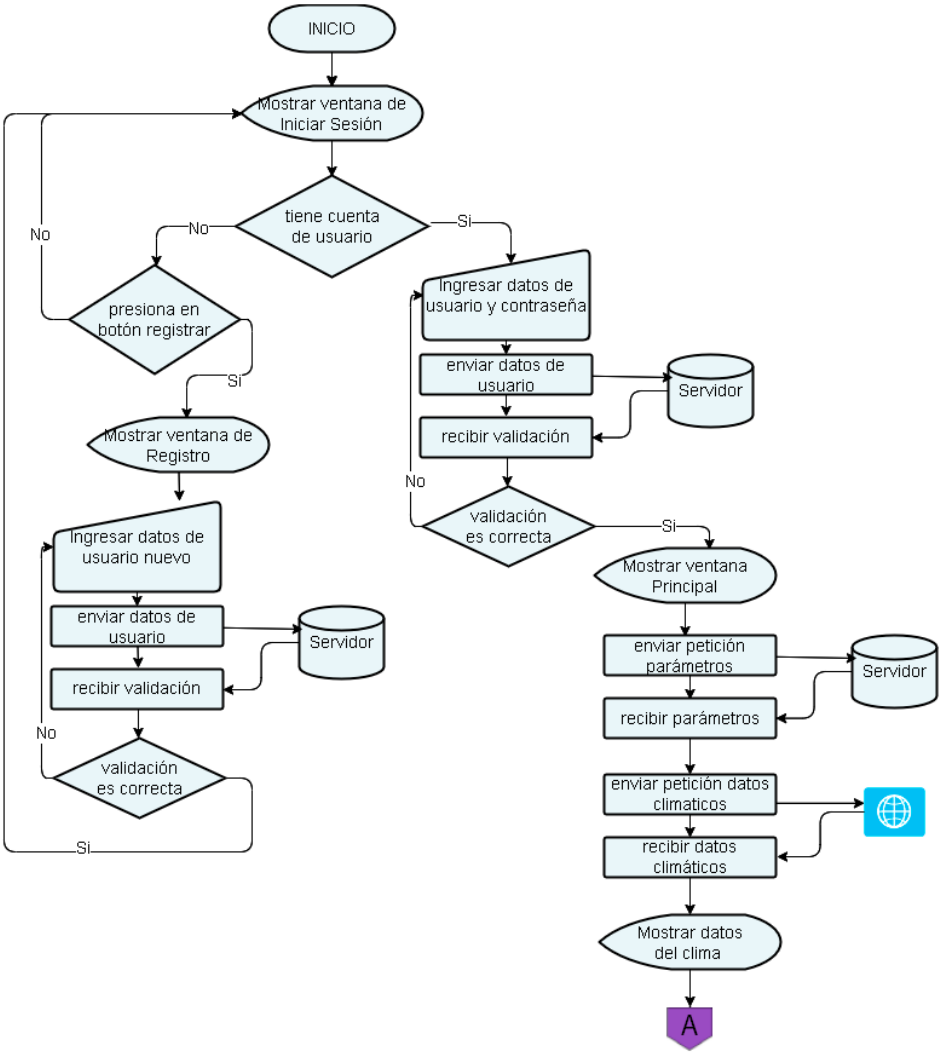


Figura 10. Diagrama de flujo de Registro e ingreso

El diagrama continúa en la Figura 11, donde, se muestra el flujograma que abarca la etapa de inicialización para los indicadores. En esta fase, se realiza una revisión del estado de los indicadores, confirmando la selección del modo automático, donde el riego comienza cuando la humedad del suelo es inferior al 30%, la humedad, se mide con el sensor ubicado verticalmente a 15 cm bajo tierra. En contraste, en el modo manual, el usuario tiene la capacidad de iniciar el riego sin esta consideración. Además, se verifica el estado activado o desactivado la válvula solenoide y el estado del botón de inicio para marcar el inicio del proceso de adquisición y procesamiento de imágenes del jardín.

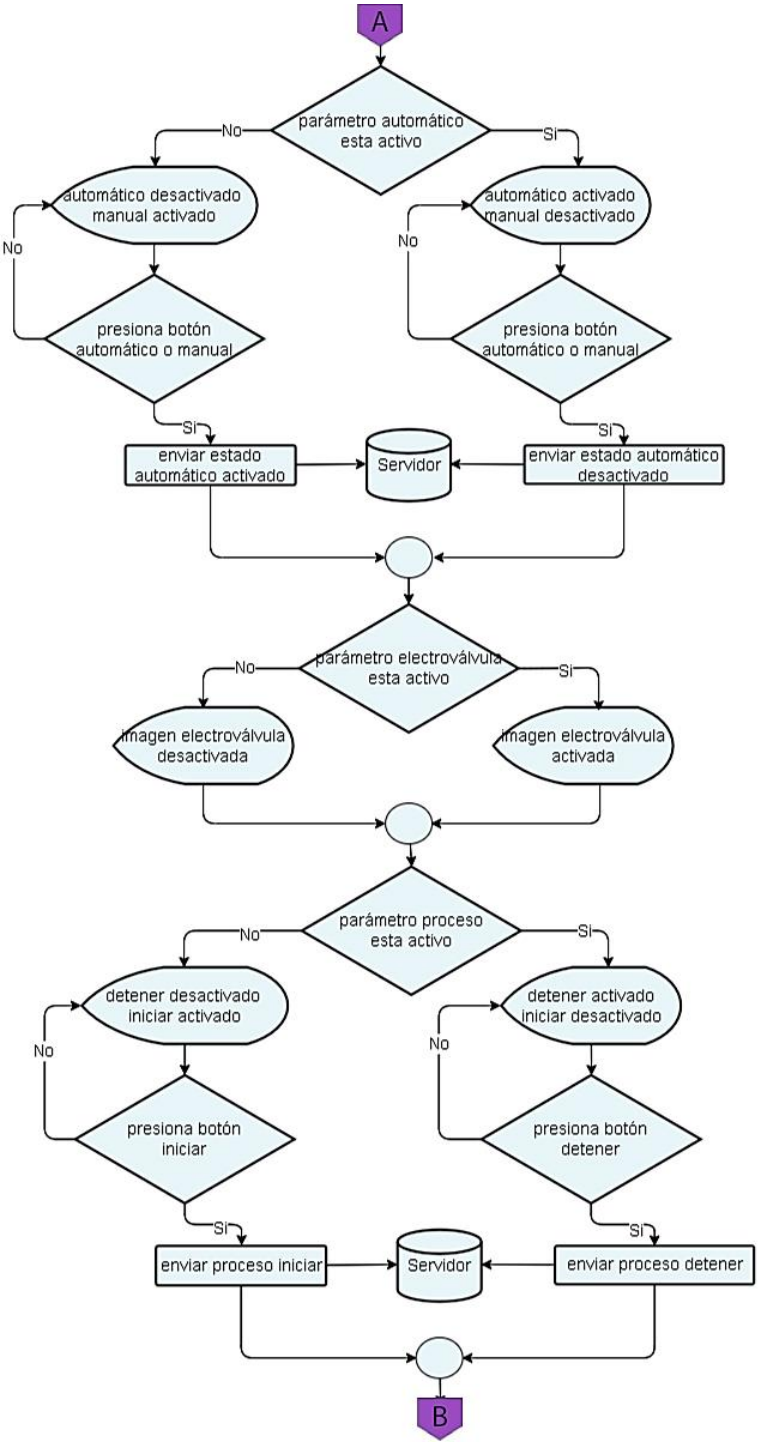


Figura 11. Diagrama de flujo con la inicialización de los indicadores

En la Figura 12, se presenta el flujograma que describe los procesos 1 y 2. En el proceso 1, se adquiere la imagen mediante la ESP32-CAM y, si la captura es exitosa, se envía a la base de datos para el siguiente procedimiento. En el proceso 2, la etapa de procesamiento implica enviar la imagen al modelo preentrenado. Una vez detectado el césped, se procede a la detección del contorno mediante la umbralización de la

imagen en formato HSV con saturación a 5 para corregir la iluminación y basar la detección en los colores. Posteriormente, se filtra la imagen en escala de grises con un umbral más bajo para detectar objetos y dibujar los contornos. En la fase de verificación de la posición, se obtienen los puntos referentes a los ángulos de los servomotores. Una vez que estos valores están guardados en el servidor, se realiza una petición a la ESP32 para iniciar el proceso 3.

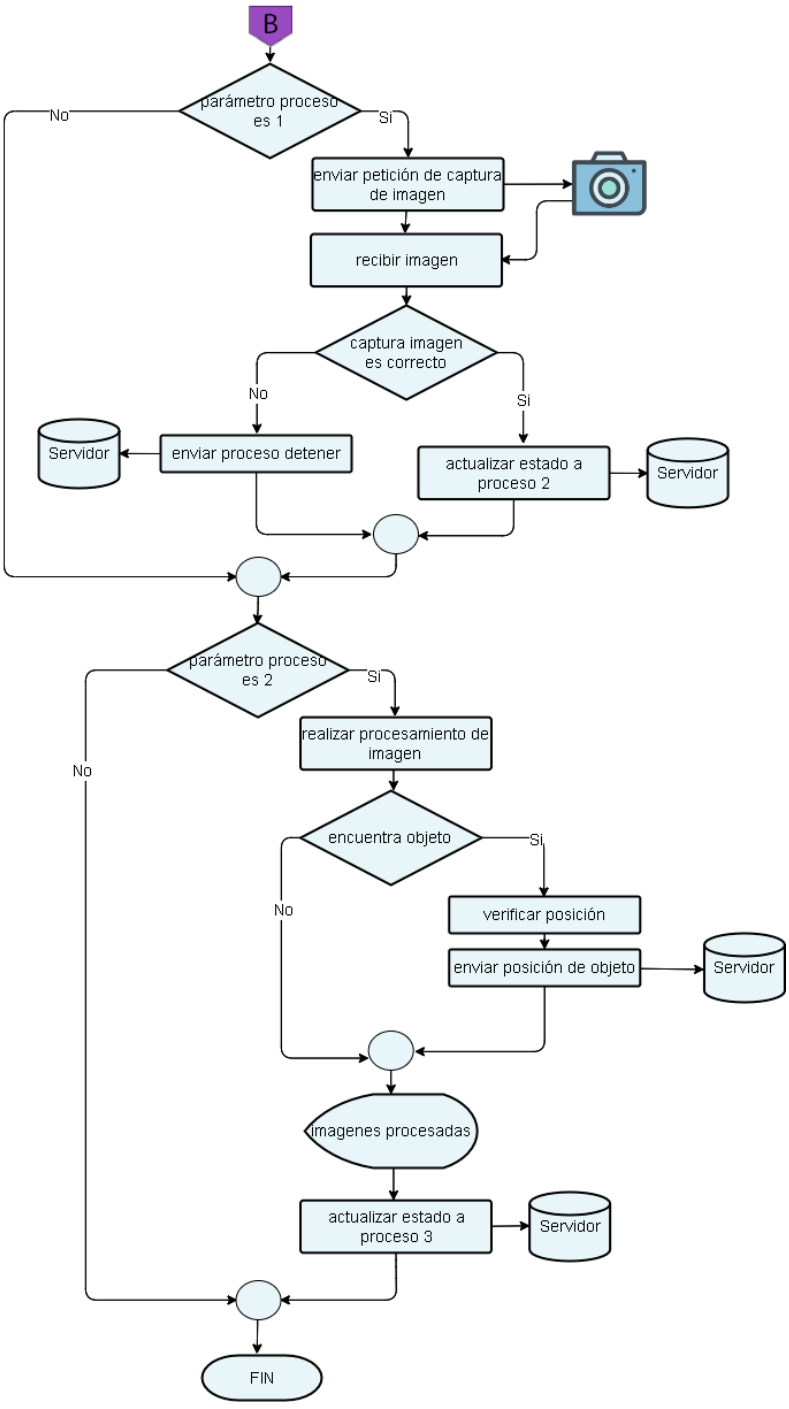


Figura 12. Diagrama de flujo del proceso 1 y del proceso 2

La etapa de control, como indica la Figura 13, comienza con el envío y recepción de parámetros para identificar la fase en la que, se encuentra el sistema, así como la transmisión continua de los valores de humedad del suelo. Una vez recibida la solicitud del proceso tres, se colocan los servomotores en posición cero para iniciar el riego. En caso de detectarse un objeto, se recibe información sobre los puntos de ubicación del objeto para ajustar los ángulos de movimiento. Se activa la válvula solenoide y, se verifica la posición de los servos; si es la correcta, se inicia el incremento del ángulo del servo vertical hasta alcanzar el máximo para después aumenta repetitivamente el ángulo del servo horizontal hasta que también alcance su valor máximo. Una vez que esto sucede, se posicionan los servomotores en su ubicación inicial y, se envía la información al servidor para concluir los procesos del sistema y finalizar el riego.

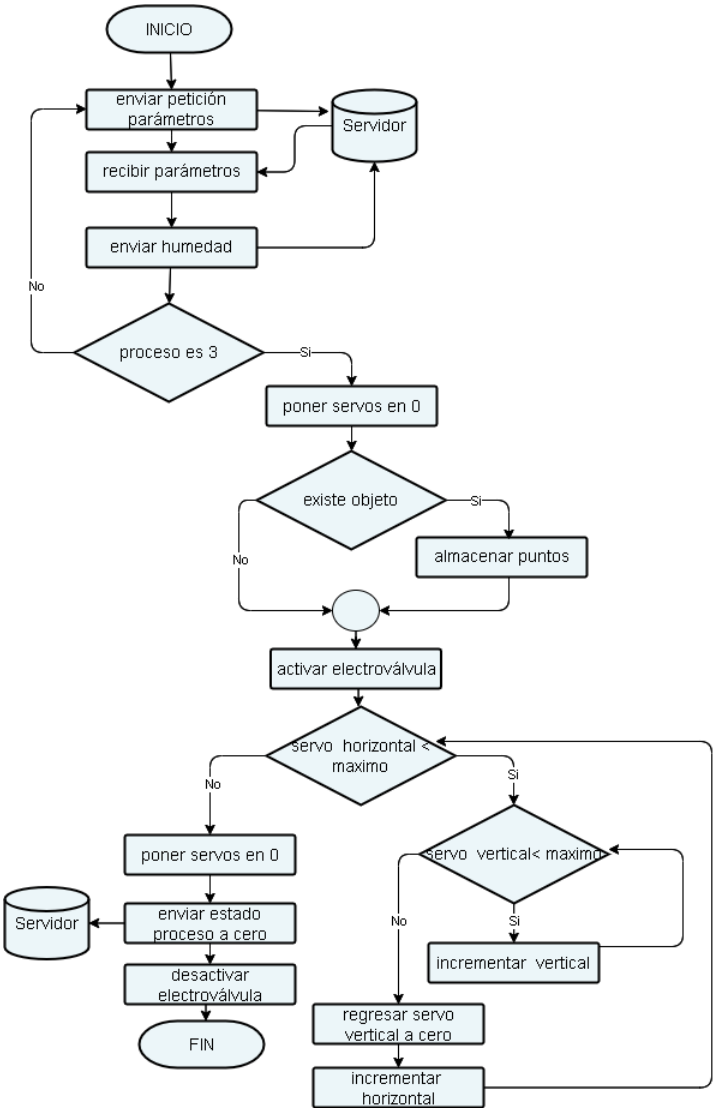


Figura 13. Diagrama de la etapa de control

3.4 Arquitectura del sistema

La arquitectura del sistema mostrada en Figura 14, se organiza en diversas etapas cruciales para su funcionamiento eficiente. La primera etapa, la de adquisición, implica la recopilación de datos provenientes de la ESP32-CAM y del sensor FC-28. Posteriormente, estos datos, se almacenan y someten a un proceso de procesamiento para su análisis. En la etapa de control, se ejecutan las acciones necesarias en la ESP32 para el riego, basadas en la información obtenida, permitiendo la regulación de los servomotores y de la válvula solenoide. Los datos procesados, se almacenan para su posterior referencia en la etapa de almacenamiento. La visualización y el monitoreo, se realizan en las últimas etapas, donde los resultados finales del procesamiento y control, se presentan al usuario, facilitando así una gestión efectiva del sistema.

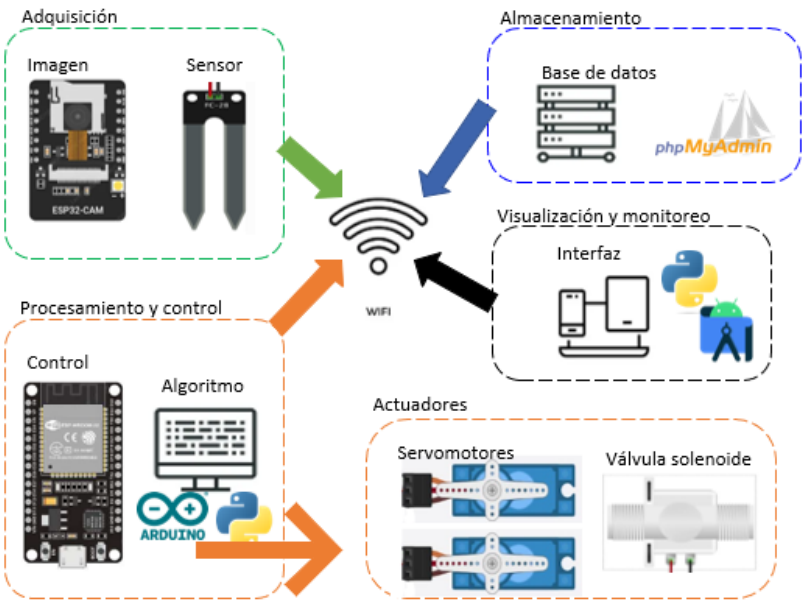


Figura 14. Arquitectura del sistema

3.5 Diseño del circuito

3.5.1 Diagrama del circuito para el módulo ESP32

Debido a que el circuito está conformado en su mayoría de tarjetas y módulos electrónicos, no se considera óptimo realizar una placa para todos los dispositivos, en su lugar, se diseña un circuito de conexión para la ESP32 WROOM 32, como, se

observa en la Figura 15, donde además de añadir canales para los pines de datos, se añade varias entradas para las conexiones de 5V, GND y 3,3V. Se diseña el circuito con varios puertos de conexión para un mismo pin, al igual que borneras para garantizar la sujeción adecuada de cables.

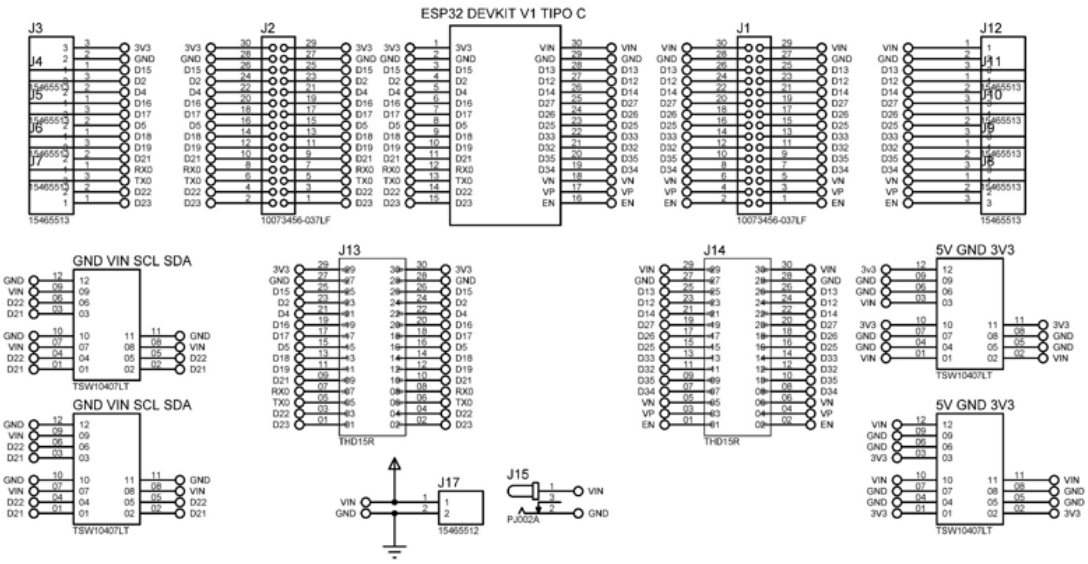


Figura 15. Esquema de conexiones para la ESP32

En la Figura 16, se tiene el diseño de la PCB con el esquema de conexión del circuito, donde, se visualiza la ubicación de borneras y el diseño de las pistas del circuito para la ESP32.

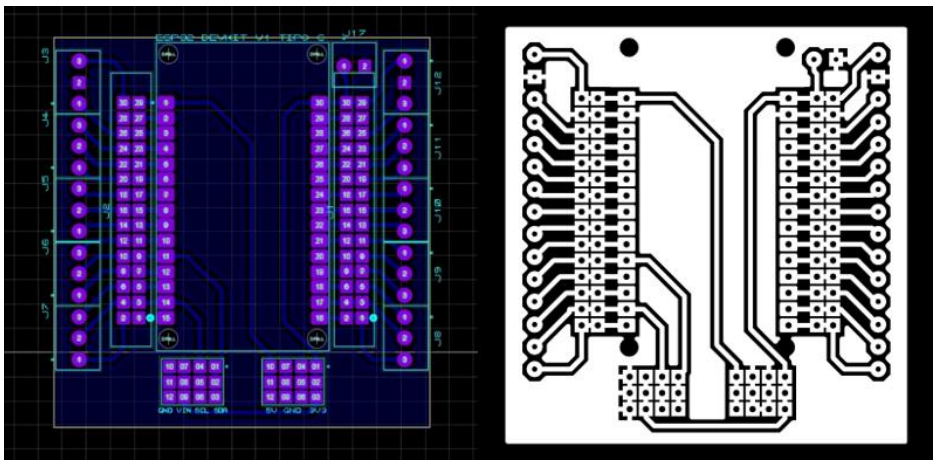


Figura 16. Diseño PCB Layout

La Figura 17, muestra una vista 3D de la simulación del circuito. Aquí, se verifica los respectivos pines de conexión, borneras y demás componentes. La placa tiene dimensiones de 6,4 cm de ancho por 6,6 cm de alto.

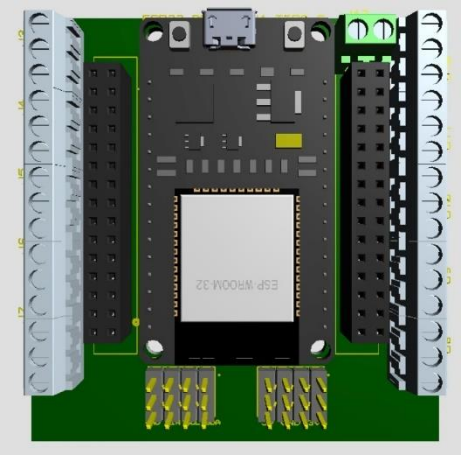


Figura 17. Vista PCB en 3D

Para la PCB, se empleó el método CNC para la definición de las pistas, se optó por baquelita de fibra de vidrío y tinta UV para proteger las pistas y obtener una placa con durabilidad ante la corrosión y la humedad. Todo esto debido a que el sistema manipula agua y en el caso de que, se produzca filtraciones, se requiere minimizar los daños posibles. En la Figura 18, se tiene las diferentes vistas de la placa.

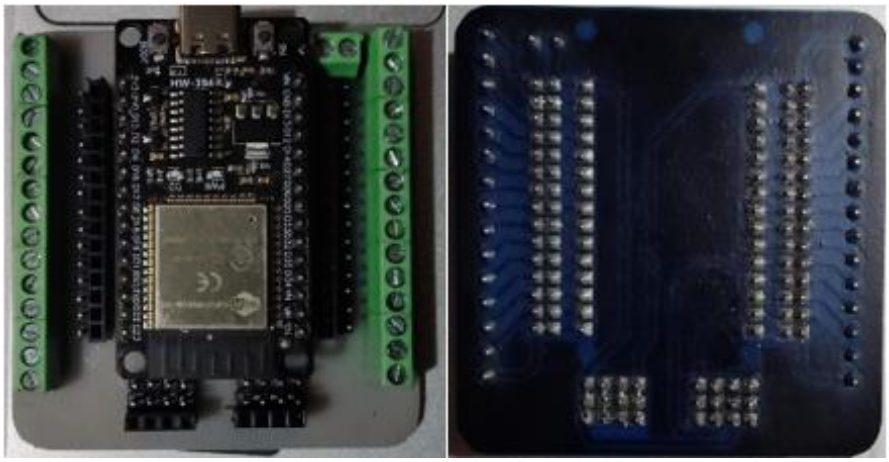


Figura 18. Vista superior e inferior de la placa

3.5.2 Consumo energético

Se requiere conocer el consumo energético de cada uno de los elementos seleccionados para diseñar el circuito del prototipo. En la Tabla 26, se detalla el consumo de corriente y el voltaje de operación de cada uno de los elementos del sistema de acuerdo al fabricante.

Tabla 26. Consumo de potencia del sistema

Elemento	Cantidad	Consumo de corriente [mA]	Voltaje de operación [V]	Potencia [W]
ESP32	1	500	3.3	1,65
ESP32-CAM	1	500	5	2,5
FC-28	1	5	3.3	0,0165
TD-8120MG	2	2000	4.8	19,2
Módulo rele	1	15	5	0,075
Válvula Solenoide	1	600	12	7,2
Total				30,64

Se determina que el consumo total de potencia del sistema es de 30,64 W. La mayoría de los elementos, se pueden energizar desde el módulo de la ESP32 ya que utilizan 3.3V, 5V y corrientes no muy altas, a excepción de la válvula solenoide que requiere de 12V para operar. Por lo cual, se plantea la opción de usar dos fuentes de alimentación para el sistema.

3.5.3 Diagrama del circuito para el prototipo

En base al esquema general del sistema y los dispositivos seleccionados, se diseña el circuito de la Figura 19, respetando la arquitectura del prototipo. En la Figura 19, se presenta el circuito, donde para la etapa de adquisición la ESP32-CAM, se encargará de capturar la imagen del jardín y el sensor FC-28 toma los valores de humedad del suelo. La etapa de procesamiento, se realiza por la ESP32 WROOM 32, para dar instrucciones de control a la válvula solenoide y servomotores. La etapa de almacenamiento, se realiza mediante la transmisión de dato mediante Wi-Fi por parte de la ESP32-CAM y de la ESP32.

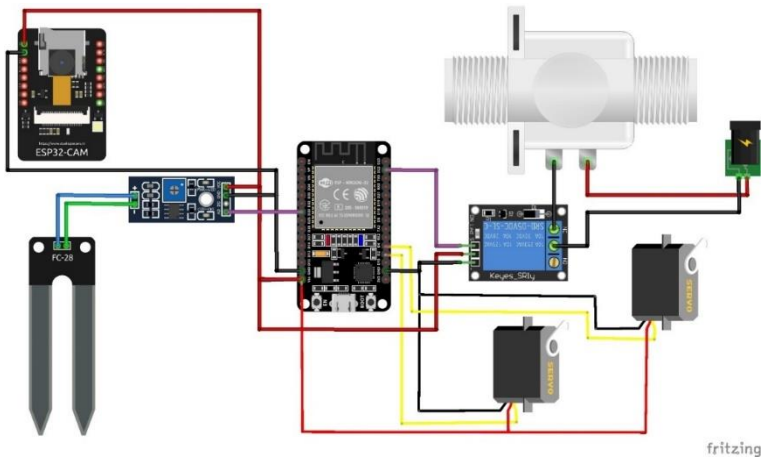


Figura 19. Circuito del prototipo

Además, en la Figura 20, se presenta el diagrama esquemático de conexión del circuito, donde, se observa los pines y puertos de conexión.

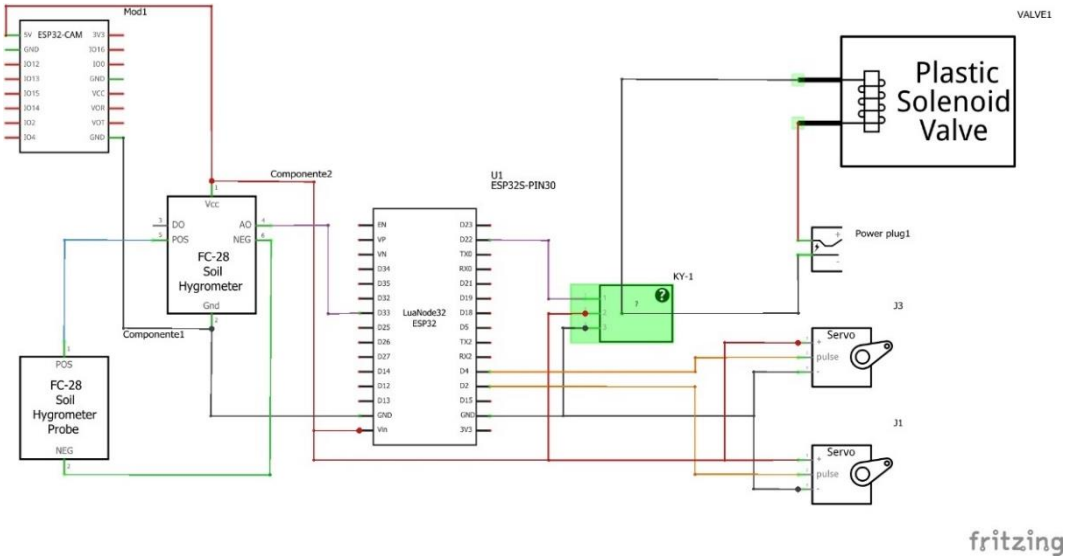


Figura 20. Esquema de conexión del circuito para el prototipo

En la Figura 21, se observa el circuito del prototipo montado con todos los elementos seleccionados previamente y en base al esquema de conexión de la Figura 20.

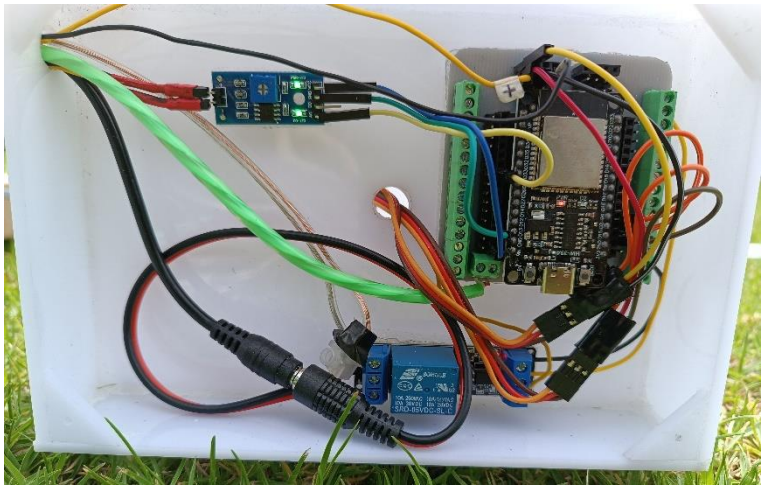


Figura 21. Circuito montado en el prototipo

3.5.4 Alimentación y pines de conexiones

A continuación, en la Tabla 27, se presentan los pines de conexión de los elementos y los valores de voltaje y corriente medidos en la alimentación de cada uno. El circuito funciona con dos fuentes de alimentación: una fuente de alimentación de 5V a 3A para

la alimentación de la ESP32 dando una potencia de 15W y una fuente de alimentación regulable funcionando en 12V a 3A para la válvula solenoide que brinda una potencia de 72 W. El potencial total con ambas fuentes de alimentación es de 87W.

Tabla 27. Pines de conexión y datos de alimentación

Pin de conexión para datos	**Elementos**	**Datos de alimentación**	
		Voltaje [V]	**Corriente [mA]**
-	ESP32-CAM	4,76	23,33
D33	FC-28	3,02	3,32
D4	Servomotor vertical	4,88	6,06
D2	Servomotor horizontal	4,88	6,06
D22	Módulo relé	5,05	1,08

3.6 Diseño del prototipo

El diseño, se centra en la construcción de un sistema de riego basado en aspersión para jardines donde, se requiere el movimiento tanto horizontal como vertical para direccionar el chorro de agua en el área determinada. En la Figura 22, se muestra el modelado completo, para verificar las dimensiones revisar el Anexo G. En este apartado, se presentan los diseños desarrollados tanto para el mecanismo como el case de recubrimiento del mismo.

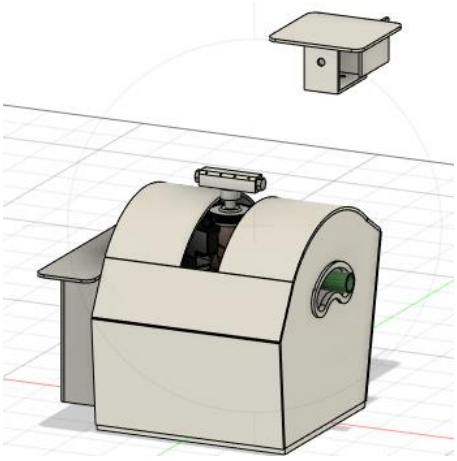


Figura 22. Modelado completo del prototipo

3.6.1 Mecanismo físico

El mecanismo cuenta con cuatro secciones: aspersor, cuerpo superior, cuerpo inferior y base. En la Figura 23, se identifica cada una de las secciones. El cabezal del aspersor, se diseñó en base a aspersores comerciales, específicamente los agujeros, los cuales son cuatro de 2mm de diámetro cada uno para abarcar un área de riego adecuada.

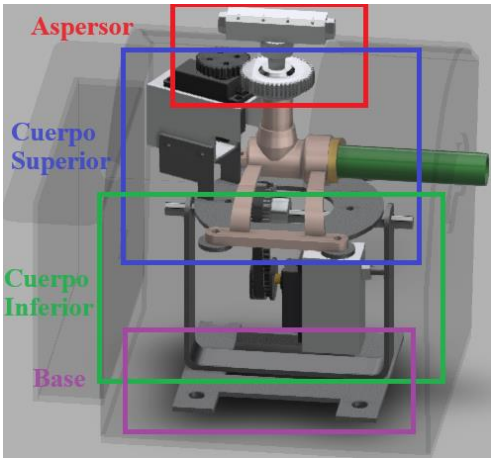


Figura 23. Diseño del mecanismo por secciones

El cuerpo superior está conformado por el servo de movimiento horizontal de 180 grados, un engranaje que permite el movimiento del aspersor, y toma de agua mediante una manguera. En la Figura 24, se ilustra el movimiento, para mantener el flujo de agua del aspersor a la manguera, se utilizó la base del aspersor AQUACRAFT Metal Rotary Sprinkler. El cuerpo inferior consiste en el servo de movimiento vertical con un rango menor a 180 grados, poleas dentadas, y banda dentada que facilite el movimiento del cuerpo superior y aspersor. En la Figura 25, se muestra el movimiento del servomotor. La base consiste en dos placas que sostienen la estructura con orificios para la sujeción al case.

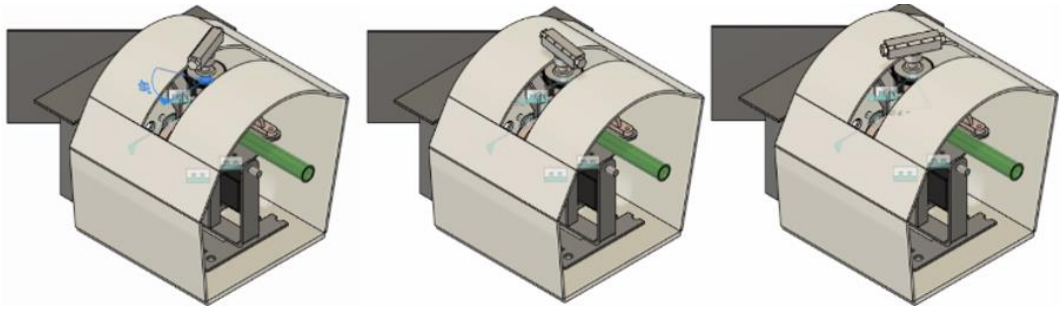


Figura 24. Movimiento servomotor horizontal

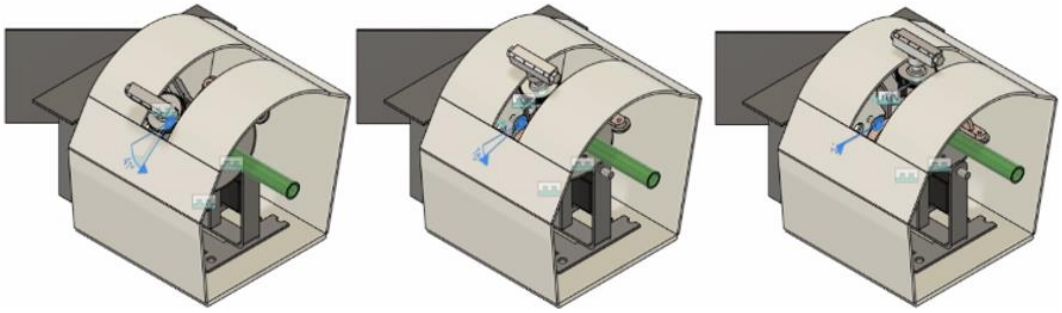


Figura 25. Movimiento servomotor vertical

En la Figura 26, se presenta el mecanismo desarrollado en base al diseño de la Figura 23.

Figura 26. Mecanismo desarrollado

Se realizará pruebas de funcionamiento en una distancia no mayor a 9m, por lo cual, se toma en consideración un ángulo de elevación máximo de 80 grados para el servomotor horizontal.

3.6.2 Case del prototipo

El recubrimiento del mecanismo evita el paso de agua hacia los servomotores, como, se muestra en la Figura 27, cuenta con una salida para la conexión de manguera y un apartado para la conexión de los dispositivos electrónicos. Se diseño con una curvatura en el plano superior para facilitar el movimiento del cabezal aspersor, además de una ranura curveada para el movimiento de la manguera en el cuerpo superior del

mecanismo. Finalmente, se añadió una caja en el lateral derecho para los dispositivos de control.

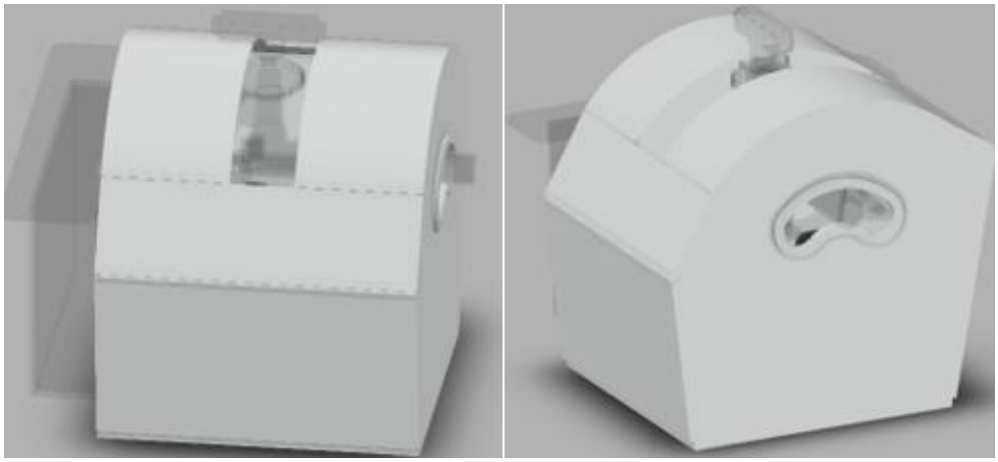


Figura 27. Diseño del case para el mecanismo

En la Figura 28, se presentan las diferentes vistas del case con el mecanismo desarrollado en base al diseño de la Figura 27

Figura 28. Case del mecanismo

En la Figura 29, se tiene una vista interna del case con el mecanismo montado.

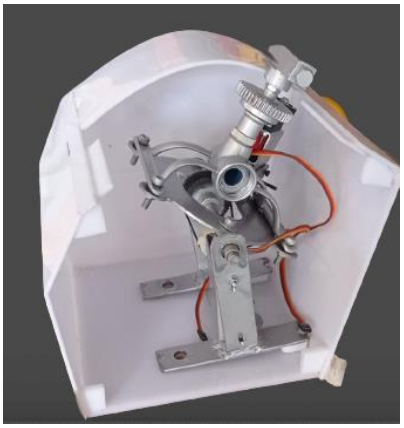


Figura 29. Vista interna del case con el mecanismo

3.6.3 Case de cámara

Se requiere de un recubrimiento para la ESP32-CAM ya que, se encontrará directamente en el ambiente. Se diseño una pequeña caja con orificios para la cámara y salida de cables. Se incorporo una visera que evita el contacto directo de la lluvia con el lente de la cámara OV2640. En la Figura 30, se tiene el diseño con el case desarrollado para la cámara.

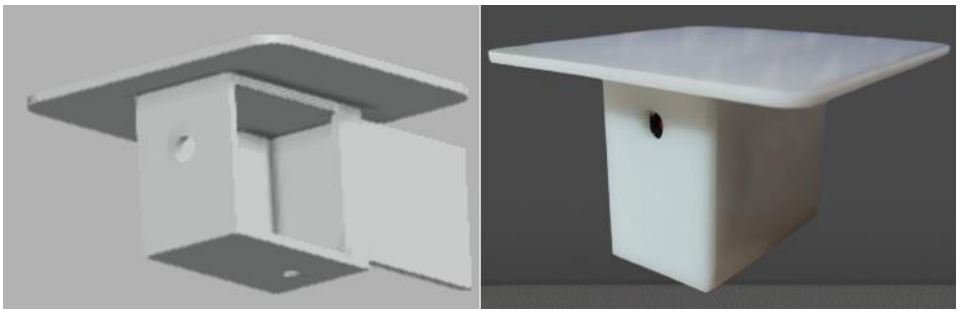


Figura 30. Diseño de case para ESP32-CAM y Case para el cámara desarrollado

3.6.4 Algoritmos

El algoritmo, se desarrolló iniciando con un entrenamiento en YOLOv5 para detectar el césped, con esta detección, se realiza en Python el filtrado para determinar el contorno del césped y el contorno del objeto. El control del riego, se realizó en Arduino IDE, se programó para el control y adquisición de valores del sensor de humedad y los actuadores: la válvula solenoide y los servomotores. Se debe tener en cuenta que los valores obtenidos son compartidos a través de una base de datos local tanto para el

algoritmo en Python, como para el control en la ESP32, de igual manera estos datos son llamados a la interfaz y la aplicación móvil.

a. Entrenamiento

El entrenamiento, se realiza con YOLOv5, para el cual, se tomó 400 imágenes para entrenamiento y 80 imágenes para la validación. Estas imágenes son de patios con jardines verdes en diferentes ubicaciones con diferentes patrones. Inicialmente, se recurre a la plataforma en línea MakeSense.AI que permite marcar el área que, se desea reconocer como un objeto, el área verde etiquetada como césped es marcada manualmente en cada imagen para su procesamiento como, se observa en la Figura 31.

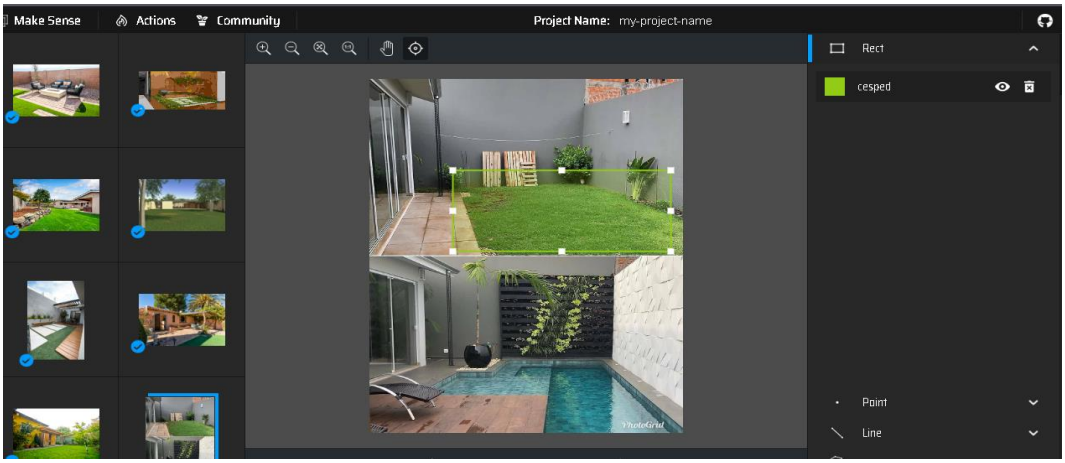


Figura 31. Selección y colocación de la etiqueta césped

Posteriormente, se obtiene un archivo zip que contiene un documento con las etiquetas de césped de cada imagen en formato de texto. Este archivo provee la plataforma MakeSense.AI al finalizar la etiquetación de las imágenes como, se observa en la Figura 32. El archivo, se presenta en un formato compatible con YOLO, por lo cual solo, se requiere organizar las imágenes en carpetas separadas para entrenamiento (train) y validación (val).

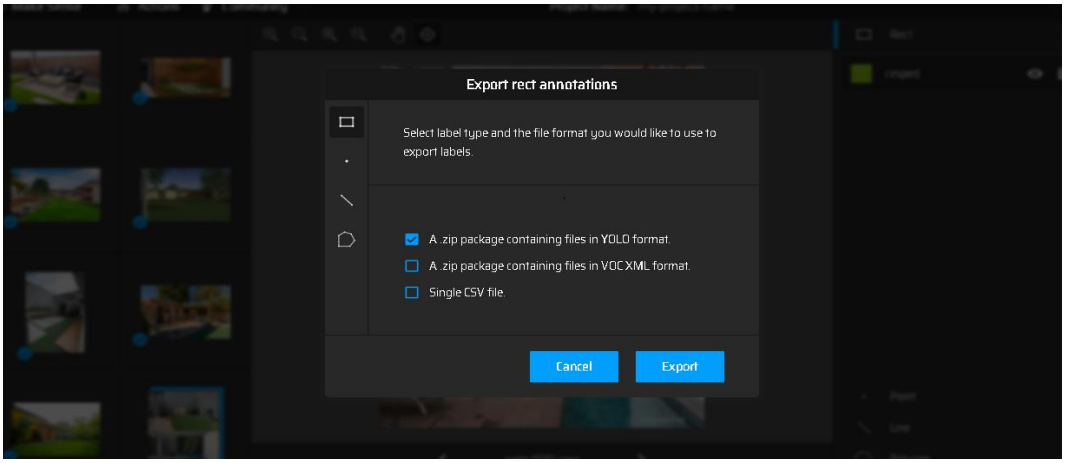


Figura 32. Archivo zip con las etiquetas compatible con YOLO

A continuación, se usa Google Colab Collaboratory, en esta fase lo mejor es cargar la carpeta comprimida de las imágenes y etiquetas para las fases de entrenamiento y validación en Google Drive, una vez cargada la carpeta, se puede iniciar desde Google Colab el entrenamiento con YOLOv5 debido a que provee ejemplos en los cuales, se facilita el entrenamiento ingresando los datos y parámetros requeridos. Se selecciona y descomprime para carpeta data que contiene las imágenes y etiquetas. Además, de cargar los paquetes para el funcionamiento de YOLOv5 y las librerías necesarias. Este proceso, se presenta en la Figura 33, Figura 34, y Figura 35.

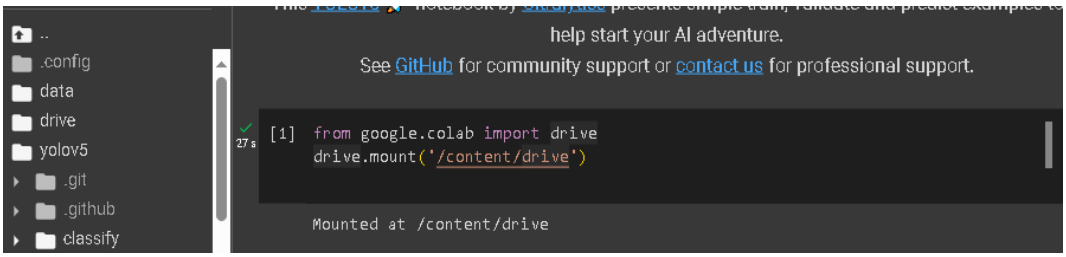


Figura 33. Librería y archivo para cargar desde drive

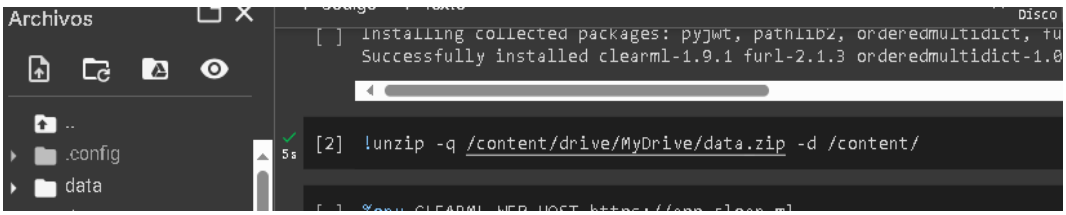


Figura 34. Descomprimir archivo

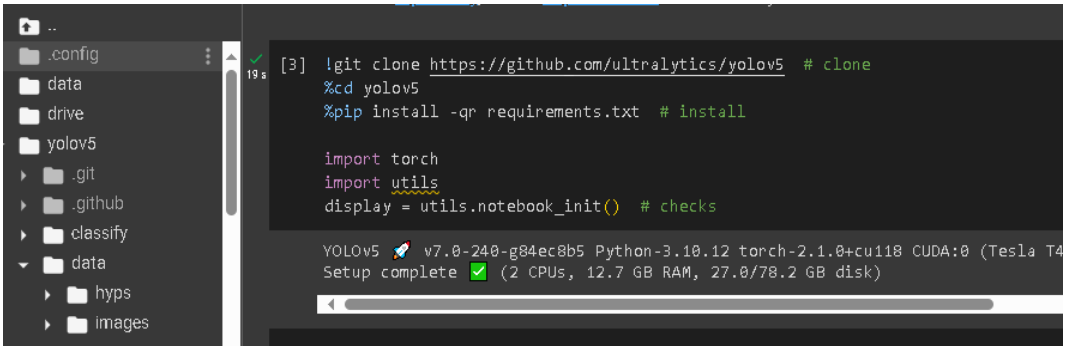


Figura 35. Instalar librerías

En la Figura 36, se define la cantidad de imágenes para el entrenamiento, el lote y la época o iteración. La época o iteración permite definir el número de veces que, se realiza el entrenamiento para obtener el mejor resultado. En este caso, se definió una época (epochs) de 200. El proceso de entrenamiento inicia y puede demorar un poco hasta finalizar. En la Figura 37 y Figura 38, se muestra el inicio y fin de este entrenamiento.

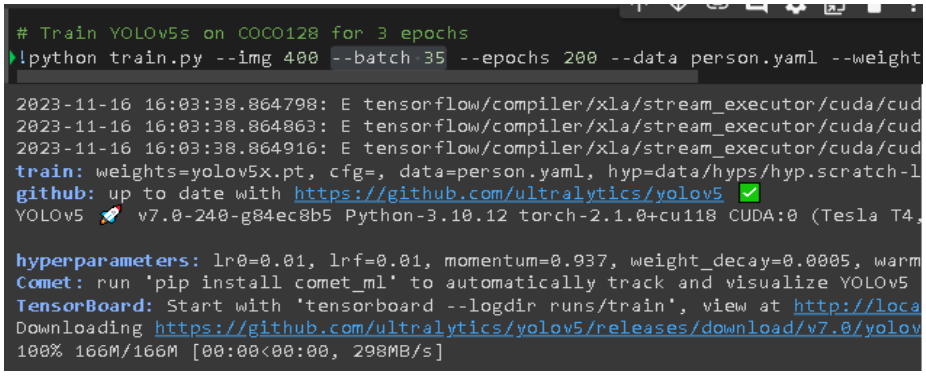


Figura 36. Definición de épocas en el entrenamiento

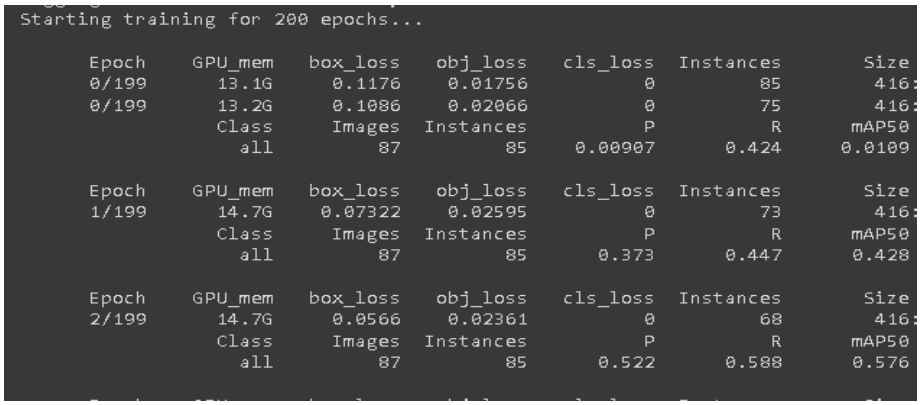


Figura 37. Inicio del entrenamiento

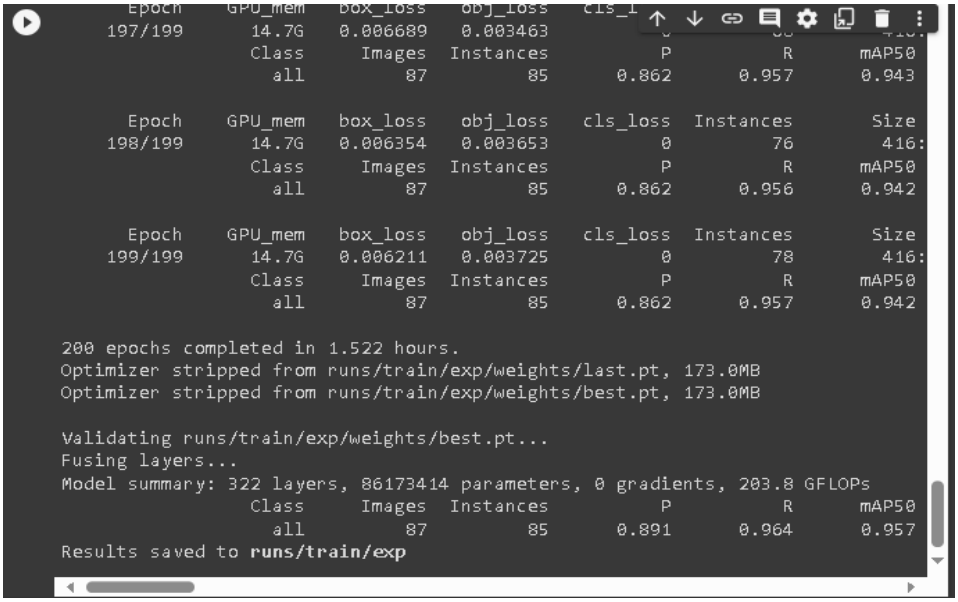


Figura 38. Finalización del entrenamiento

Una vez terminado el entrenamiento, se obtiene un archivo en lenguaje Python en el cual, se encuentran los datos para la detección de césped en cualquier imagen. Este archivo, se lo encuentra con el nombre “best.pt”, como muestra la Figura 38. En la Figura 39, se muestra la ubicación del archivo para descargarlo.

Figura 39. Ubicación del archivo best.pt

b. Programación en Python

Aquí, se tiene una explicación de la programación usada para el algoritmo de detección, se tiene el código completo en el Anexo H. Para el sistema, se usaron las librerías de la Figura 40, las cuales, se describen a continuación

```
import tkinter
import tkinter.font as font
from tkinter import ttk
from tkinter import *
from tkinter import messagebox
import cv2
import time
import threading
import torch
import numpy as np
import json as json
from PIL import Image, ImageTk
from urllib.parse import urlencode
from urllib.request import Request, urlopen
from urllib.parse import quote
import requests
from bs4 import BeautifulSoup
from io import BytesIO
import math
```

Figura 40. Librerías usadas en python

- tkinter: Permite crear el entorno grafico de una aplicación.
- cv2 (OpenCV): Proporciona herramientas para realizar el tratamiento de videos e imágenes en lo que respecta a visión por computadora.
- time: Facilita la interrupción del procedimiento durante un intervalo temporal.
- threading: Permite ejecutar un código en segundo plano, para evitar que la aplicación, se bloquee.
- torch: PyTorch, Permite obtener el resultado de un entrenamiento previamente realizado en YOLO.
- numpy: Permite realizar operaciones matemáticas en arreglos o matrices de datos.
- json: Además de permitir usar datos en formato JSON, ayuda a obtener y enviar datos con url.
- PIL (Pillow): Permite manipular y guardar imágenes en diferentes formatos.
- Urlib: Permite la apertura y lectura de URL, así como el envío de datos a través de ellas.

- base64: Proporciona funciones para codificar y decodificar datos utilizando el esquema de codificación Base64, permite codificar y decodificar una imagen.
- Tkinter.messagebox: Permite crear ventanas de diálogo simples para mostrar mensajes al usuario.
- Io: Permite manipular flujos de datos de manera eficiente y flexible.
- Math: Permite el uso de funciones matemáticas.

El procesamiento, se inicia capturando la imagen de la ESP32-CAM, como, se muestra en la Figura 41, se definió una función en la cual, se trabaja con la espcam en modo por defecto, obteniendo la url y capturando una imagen desde la misma. Una vez, se obtiene la imagen capturada, se la convierte en bytes y, se guarda. Después, se llama a la función cambiarEstado para indicar que el proceso ha finalizado y puede continuar con el siguiente proceso.

```
def capturarImagen():
    global valorEstado
    try:
        actualizarPorcentaje(5)
        tiempo_de_espera = 5
        #se trabaja con la espcam en modo por defecto, obteniendo la url y capturando una imagen
        response = requests.get(f"http://{ipespcam}/capture", timeout=tiempo_de_espera)

        if response.status_code == 200:
            image_data = BytesIO(response.content)
            image = Image.open(image_data)
            image.save('imagenServidor.jpeg', "JPEG")
            #close: libera memoria y la imagen
            image.close()
            cambiarEstado()
            actualizarPorcentaje(10)
        else:
            print(f"Error al capturar la imagen. Código de estado: {response.status_code}")
            valorEstado=0
            actualizarEstado()
            actualizarPorcentaje(0)

    except Exception as e:
        #si existe algun error con la conexión, emitir un mensaje, detener el proceso y reiniciar
        print(f"Error: {e}")
        valorEstado=0
        actualizarEstado()
        actualizarPorcentaje(0)
```

Figura 41. Capturar Imagen

Con la imagen capturada, en la Figura 42, se observa cómo, se define el parámetro de efectividad al mostrar resultados positivos dentro del modelo entrenado, se pide resultados mayores al 90%, se guarda en una variable el resultado del procesamiento de la imagen dentro del modelo, definiendo el tamaño, y redimensionando la imagen.

Después, se obtiene el valor de la detección, y si, se encuentra un objeto, se obtiene las coordenadas que lo delimitan.

```
image_path = "imagenServidor.jpeg"
im1 = Image.open(image_path)
#definimos el parametro de efectivilidad
model.conf = 0.9
detect = model(im1, size=400)
nueva_imagen_Principal = im1.resize((350, 200))
nueva_imgCam1 = ImageTk.PhotoImage(nueva_imagen_Principal)
lblImgCam1.configure(image=nueva_imgCam1)
print("leer imagen")
#devuelve infomacion sobre las coordenadas que delimitan si en la imagen existe objetos detectados
info = detect.pandas().xyxy[0]
arr = np.asarray(info.confidence.array)
detectar = len(arr)
actualizarPorcentaje(18)
if detectar > 0:
    #mensaje o indicador para validar si el procesamiento ha obtenido respuesta desde el modelo
    print("verificando")
    cv2.imwrite("resultado.png", cv2.cvtColor(np.squeeze(detect.render()), cv2.COLOR_BGR2RGB))
    #imagen con la deteccion obtenida y el valor detectado
    cv2.imwrite("C:/xampp/htdocs/sistema/resultado.png", cv2.cvtColor(np.squeeze(detect.render()), cv2.COLO
    nueva_imagen = Image.open('resultado.png').convert('RGB')
    #obtener las cordenadas del objeto detectado
    box_coords = info[['xmin', 'ymin', 'xmax', 'ymax']].to_numpy(dtype=int)[0]
    roi = np.array(nueva_imagen)[box_coords[1]:box_coords[3], box_coords[0]:box_coords[2], :]
    #crear una imagen en la carpeta local donde se mostrará solo la parte que ha detectado
    cv2.imwrite("soloCesped.png", cv2.cvtColor(roi, cv2.COLOR_RGB2BGR))
    #redimencionar la imagen original con la deteccion obtenida
    nueva_imagen = nueva_imagen.resize((350, 200))
    nueva_imgCam2 = ImageTk.PhotoImage(nueva_imagen)
    lblImgCam2.configure(image=nueva_imgCam2)
```

Figura 42. Detección de césped y coordenadas de objetos.

Una vez obtenida la imagen de efectividad del entrenamiento, se recorta la imagen, se convierte a espacio de color HSV, para después multiplicar el valor de saturación de la imagen por el parámetro de saturación definido, y finalmente verifica que los valores estén dentro del rango de 0 a 255 antes de guardarla. A continuación, en la Figura 43, se define valores mínimos y máximos para detectar césped, se toma en consideración que los colores a detectar no son solo verdes, ya que muestra colores grises, amarillos, café y varias tonalidades de verdes, incluyendo valores que asumen un porcentaje de rojo. En la Figura 44, se define una máscara que permite identificar dentro de la imagen los pixeles que contienen los colores, para buscar y almacenar los contornos encontrados en una nueva imagen.

```
imagenSoloCesped = cv2.imread('soloCesped.png')
#cambiar los colores de dicha imagen a HSV
imagen_solocesped_hsv = cv2.cvtColor(imagenSoloCesped, cv2.COLOR_BGR2HSV)
factor_saturacion = 5
#proceso para multiplicar el valor de saturacion de la imagen por el parametro de saturacion definido
imagen_solocesped_hsv[:, :, 1] = np.clip(imagen_solocesped_hsv[:, :, 1] * factor_saturacion, 0, 255).as
#convertir imagen de HSV a BGR
imagen_corregida = cv2.cvtColor(imagen_solocesped_hsv, cv2.COLOR_HSV2BGR)
cv2.imwrite('corregida.png', imagen_corregida)
imagenCorregida1 = cv2.imread('corregida.png')
#cambiar los colores de dicha imagen a HSV
imagen_hsv_corr = cv2.cvtColor(imagenCorregida1, cv2.COLOR_BGR2HSV)
#definir valores minimos y maximos para detectar césped
verde_bajo_imgCor = np.array([0, 0,50], np.uint8)
verde_alto_imgCor = np.array([120, 255, 255], np.uint8)
red_bajo_imgCor=np.array([150, 0, 0], np.uint8)
red_alto_imgCor=np.array([179, 255, 255], np.uint8)
#definir mascara que permite identificar dentro de la imagen pixeles que contienen los colores
mascara_verde_corr = cv2.inRange(imagen_hsv_corr, verde_bajo_imgCor, verde_alto_imgCor)
mask_red_corr = cv2.inRange(imagen_hsv_corr, red_bajo_imgCor, red_alto_imgCor)
#se une las dos máscaras definidas para integrarlas a la imagen
maskCompleta_corr = cv2.add(mascara_verde_corr, mask_red_corr)
pixeles_verdes_completo = cv2.bitwise_and(imagenCorregida1, imagenCorregida1, mask=maskCompleta_corr)
#buscar y almacenar contornos encontrados dentro del rango anterior
contornos, _ = cv2.findContours(mascara_verde_corr, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#permite dibujar los contornos del resultado obtenido
cv2.drawContours(pixeles_verdes_completo, contornos, -1, (0, 255, 0), 2)
#aplicar la mascara obtenido a la imagen
mascara_completa = cv2.bitwise_or(mascara_verde_corr, mask_red_corr)
pixeles_verdes_completo = np.zeros_like(imagenCorregida1)
pixeles_verdes_completo[mascara_completa != 0] = imagenCorregida1[mascara_completa != 0]
#crear una imagen con el resultado de la imagen y los contornos
```

Figura 43. Detección de contornos de césped parte 1

```
cv2.imwrite('verdesCompleto.png', pixeles_verdes_completo)
#Abrir o leer la imagen corregida la saturacion
imagen_VC = cv2.imread('verdesCompleto.png')
#covierte la imagen a escala de grises
imagen_gris_VC = cv2.cvtColor(imagen_VC, cv2.COLOR_BGR2GRAY)
#aplicar un umbral para crear una imagen binaria
imagen_umbral_VC = cv2.adaptiveThreshold(imagen_gris_VC, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BI
#encuentra los contornos de la imagen binaria, y así obtener solo los contornos externos
contornos_VC, _ = cv2.findContours(imagen_umbral_VC, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#definir un umbral para detectar el area de los contornos minimos a mostrar
area_minima_VC = 9000
#filtrar los contornos con los parametros anteriores
contornos_filtradosVC = [cnt for cnt in contornos_VC if cv2.contourArea(cnt) > area_minima_VC]
#dibujar contornos dentro de la imagen
cv2.drawContours(imagen_VC, contornos_filtradosVC, -1, (0, 255, 0), 3)
#crear imagen en la carpeta local no los contornos obtenidos
cv2.imwrite('contornoVerdeCompleto.png', imagen_VC)
#Abrir o leer la imagen corregida la saturacion
cesped_imagen = Image.open('contornoVerdeCompleto.png')
#redimencionar la imagen original con la deteccion obtenida
nueva_imagen2 = cesped_imagen.resize((300, 200))
#permite convertir la imagen al formato adecuado para usarla dentro de python y tkinter
cesped_imgCam1 = ImageTk.PhotoImage(nueva_imagen2)
#se muestra la imagen resultante de los contornos dentro del tercer label
```

Figura 44. Detección de contornos de césped parte 2

Para detectar la posición de los objetos, con la imagen del entrenamiento redimensionada, se realiza una conversión a escala de grises, se crea la imagen binaria y, se detecta los contornos de los objetos con un umbral mínimo definido. Se verifica

los pixeles del objeto, se realiza un filtrado de los contornos y, se crea una nueva imagen con el contorno dibujado del objeto identificado, tal como indica la Figura 45.

```
imagenObjeto = cv2.imread('verdesCompleto.png')
#convertir l imagen a escala de grices
imagen_grisObjeto = cv2.cvtColor(imagenObjeto, cv2.COLOR_BGR2GRAY)
imagen_umbralObjeto = cv2.adaptiveThreshold(imagen_grisObjeto, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THR
#encuentra los contornos de la imagen binaria
contornosObjeto, _ = cv2.findContours(imagen_umbralObjeto, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#definir un umbral para detectar el area de los contornos minimos a mostrar
#se da un parametro para verificar los pixeles donde se buscara el objeto
area_minimaObjeto = 3000
altura_minimaObjeto = 130
altura_maximaObjeto = 260
contornos_filtradosObjeto = [cnt for cnt in contornosObjeto if cv2.contourArea(cnt) > area_minimaObjeto
#crear una copia de la imagen abierta anteriormente para ubicar los objetos y sus contornos
imagen_contornosObjeto = imagenObjeto.copy()
#dibujar contornos
cv2.drawContours(imagen_contornosObjeto, contornos_filtradosObjeto, -1, (0, 0, 255), 3)
#crear imagen en la carpeta local mostrando el objeto detectado y sus contornos
cv2.imwrite('contornoObjeto.png', imagen_contornosObjeto)
#crear imagen en la carpeta del servidor mostrando el objeto detectado y sus contornos
cv2.imwrite("C:/xampp/htdocs/sistema/contorno.png", imagen_contornosObjeto)
#abrir imagen que contiene contornos de objetos
objeto_imagen = Image.open('contornoObjeto.png')
#redimencionar imagen para ubicarla en el label
objeto_imagen2 = objeto_imagen.resize((300, 200))
#permite convertir la imagen al formato adecuado para usarla dentro de python y tkinter
objeto_imgCam1 = ImageTk.PhotoImage(objeto_imagen2)
#se muestra la imagen con contornos en el objeto en el ultimo label
lblImgCam5.configure(image=objeto_imgCam1)
lblImgCam5.image = objeto_imgCam1
```

Figura 45. Detección de contornos de objeto

Se obtiene los puntos de ubicación del objeto detectado, estos puntos se requieren para controlar el aspersor evitando mojar el objeto, por lo cual se detecta en el contorno del objeto los puntos inferiores y los que se ubican en los laterales del mismo. El proceso, se muestra en la Figura 46. Se debe verificar que exista contornos para calcular una nueva escala a partir de las dimensiones originales que permita encontrar los puntos del objeto y almacenarlos.

```python
try:
    #verificar si exiten contornos encontrados
    if contornos_filtradosObjeto:
        #obtiene las dimensiones de la imagen origina
        alto, ancho, _ = imagenObjeto.shape
        nuevo_ancho = 180
        nuevo_alto = 80
        #calcular nueva escala
        factor_escala_ancho = nuevo_ancho / ancho
        factor_escala_alto = nuevo_alto / alto
        #encontrar puntos del objeto encontrado en el contorno filtrado
        contorno_transformado = contornos_filtradosObjeto[0] * [factor_escala_ancho, factor_escala_alto]
        punto_mas_bajo_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 1].argmax()][0]
        punto_mas_izquierda_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 0].argmin(
        punto_mas_derecha_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 0].argmax()]
        #almacenar puntos necesario
        x_bajo, y_bajo = punto_mas_bajo_nuevo
        x_izquierda, y_izquierda = punto_mas_izquierda_nuevo
        x_derecha, y_derecha = punto_mas_derecha_nuevo
    else:
        #si no hay contornos los valores se enviarán en cero
        y_bajo=0
        x_izquierda=0
        x_derecha=0
except Exception as e:
    #si no hay contornos los valores se enviarán en cero
    y_bajo=0
    x_izquierda=0
    x_derecha=0
```

Figura 46. Puntos del objeto detectado

c. Programación en Arduino IDE

En este apartado, se explica la programación usada dentro de la ESP32, en el Anexo I, se tiene todo el código. Para el control de los actuadores y envío de valores del sensor de humedad, se usaron las librerías de la Figura 47, las cuales, se describen a continuación

```
#include <WiFi.h>
#include <ArduinoJson.h>
#include <HTTPClient.h>
#include <base64.h>
#include <Wire.h>
#include <ESP32Servo.h>
```

Figura 47. Librerías usadas en Arduino IDE

- WiFi: Permite a la ESP32 conectarse a una red wifi.
- ArduinoJson: Permite el uso de código JSON
- HTTPClient: Permite la conexión mediante url o solicitudes http
- Base64: Permite manejar funciones de codificación y decodificación.
- Wire: Permite realizar conexiones y comunicaciones.

- ESP32Servo: Permite controles los servomotores en tarjeta ESP.

En la Figura 48, se tiene las credenciales de conexión de red y para conectarse al servidor

```
const char* ssid = "RED_MARTINEZ_Plus";
const char* password = "12041996";
String ip = "http://192.168.214.124/sistema/datos";
```

Figura 48. Credenciales de conexión

El control de los servomotores inicia con la validación del ángulo en el que tiene que iniciar el servo para una distancia predefinida. En la Figura 49, se muestra la fórmula desarrollada, se realiza una regla de tres para obtener el ángulo en que debe iniciar el servo, y luego se realiza un control para validar en que ángulo tiene que finalizar el servo para una distancia predefinida. Después, se define el ángulo horizontal máximo, se realizar regla de tres para obtener el ángulo máximo vertical tomando en consideración un largo de 9m y un ángulo de 80 grados. A continuación, se define la posición final del servo vertical en cada punto horizontal

```
puntoHA=round(90-((90*(distanciaH/2))/4));
puntoHAD=round(180-(90-((90*(distanciaH/2))/4)));
//angulo horizaontal maximo
int puntoHA2=round(90+((90*(distanciaH))/4));
//largo de 9m y un angulo de 80 grados
anguloTotal=round((originalAV*distanciaV)/9);
//posición final del servo vertical en cada punto horizontal
puntoVA=20;
```

Figura 49. Inicialización de los servomotores

Para identificar posiciones del objeto encontrado, como muestra la Figura 50, se encera el servo horizontal, se desarrolla una condición en la que, se busca garantizar que la posición vertical no sea mayor a la posición máxima, en caso de ser mayor, se le da como valor actual el máximo posible.

```
if(0<valorMinX || 0>valorMaxX){
}else{
  //posicion vertical no mayor a posicion máxima
    if(puntoVA>= valorMaxY){
      //valor actual máximo posible
      puntoVA=valorMaxY;
    }
}
```

Figura 50. Condición de seguridad para posición máxima

Para verificar si el tamaño del patio es menor a 8 m de ancho o inicia en un ángulo superior a cero, se usa una condición de seguridad como muestra la Figura 51, se emplea una regla de tres para disminuir o aumentar el ángulo vertical en la posición predeterminada tomando como referencia el valor base del patio.

```
if(puntoHA>=0){
  //regla de tres para disminuir o aumentar el angulo vertical en la posicion predeterminada
  puntoVA=round(((distanciaH/2)*puntoVA)/4);
}
```

Figura 51. Condición de seguridad para patio menor a 8 de ancho

Como, se muestra en la Figura 52, para identificar posiciones del objeto encontrado cuando está en el punto 20 del servo horizontal, se verifica que la posición vertical no sea mayor a la posición máxima, luego, se verifica que el tamaño del patio sea menor a 8 m de ancho o, se inicie en un ángulo superior a 20, y, se disminuye o aumentar el ángulo vertical en la posición predeterminada tomando como referencia el valor base del patio, además de restar el valor para igualar el servo por su posición física.

```
if(20<valorMinX || 20>valorMaxX){
}else{
  //posicion vertical no mayor a la posicion máxima
  if(puntoVA>= valorMaxY){
    //si es mayor se le da como valor actual el máximo posible
    puntoVA=valorMaxY;
  }
}
//verificar tamaño del patio menor a 8 m de ancho o inicia en un angulo superior a 20
if(puntoHA>=20){
  //disminuir o aumentar el angulo vertical en la posicion predeterminada
  //tomando como referencia el valor base del patio
  puntoVA=round(((distanciaH/2)*puntoVA)/4)-10;
}
```

Figura 52. Aumento progresivo de los servomotores

Para iniciar el riego, se verifica que, se encuentre el proceso en el valor 3, en la Figura 53, se envía los valores de la electroválvula, sensor de humedad y, se inicia el riego en ese punto.

```
if(procesoge==3){
  //verificar si el ancho del patio es el adecuado para el angulo correspondiente o es muy pequeñ
  //para dicho angulo horizontal
  if((distanciaH/2)==4){
    //llamar a la funcion para iniciar riego, dando como parámetros posiciones máximos en servo
    //horizontal y vertical respectivamente
    regarPunto(20,puntoVA);
    //enviar porcentaje actual del proceso general
    actualizarPorcentaje(45);
    //establecer en estado bajo el pin de la electrovalvula
    digitalWrite(22, LOW);
    //cambiar estado de electrovalvula
    electro=1;
    //enviar valores de humedad y estado de electrovalvula
    actualizarHumedad();
  }
}
```

Figura 53. Verificar el proceso para iniciar el riego

El código de las Figura 52 y Figura 53, se repite simultáneamente para los demás valores del servo vertical desde 20, 40, 50, 60, 70, 80, 90, 110, 120, 140, 160, hasta 180.

Con la función presentada en la Figura 54, se regresa los ángulos de los servomotores a su posición inicial. Se crearon funciones para los servomotores, de modo que si el ángulo horizontal es mayor que cero, el servo horizontal disminuye, y si el ángulo vertical es mayor que cero, el servo vertical disminuye.

```
void regreCero(){
  //verificar si el angulo horizontal actual es mayor a cero
  if(anguloH>0){
    //llamar a la funcion regresar, que permite disminuir los grados del servo
    regresarH(0);
  }
  //verificar si el angulo vertical actual es mayor a cero
  if(anguloV>0){
    //llamar a la funcion regresar, que permite disminuir los grados del servo
    regresarV();
  }
  //actualizar el valor de los angulos de los servos en cero
  anguloH=0;
  anguloV=0;
}
```

Figura 54. Función para llevar los servomotores a cero

En la Figura 55, la siguiente función permite cambiar el valor de los servos de una posición actual a una preestablecida. Debido a que, se desea regar de forma lenta y continua, el valor del servo vertical, se divide para 10 para moverlo con una posición que aumenta gradualmente evitando movimientos rápidos o bruscos.

```
void regarPunto( int valorHorizontal,int valorVertical){
  //variables que se usará dentro de esta funcion para el moviemiento adecuado de los servos
  //servo vertical division para moverlo evitando un gole brusco
  int proceso=valorVertical/10;
  int punto=0;
  //mostrar en pantalla serial la posicion a la que llegaran los servos
  Serial.println("punto nuevo horizontal: " + String(valorHorizontal) + ", vertical: " + String(val
  //llamar a la funcion subir la que permite aumentar los grados de los servos
  subirH(valorHorizontal);
  subirV(valorVertical);
  //llamar a la funcion regresarV que permite regresar el servo vertical a la posicino inicial o ce
  regresarV();

}
```

Figura 55. Función para cambiar de una posición actual a una preestablecida

Se requiere una función para incrementar el valor de los servomotores, se usó un bucle for que incrementa el valor de la posición del ángulo, al finalizar el incremento del ángulo, se actualiza el valor del ángulo actual al nuevo ángulo final. Esto, se debe realizar para el servomotor horizontal y vertical, como muestra la Figura 56 y Figura 57.

```
//funcion subir permite incrementar el valor del servo horizontal
void subirH(int nuevo){
  //variable que se usara para declarar e inicializar el bucle
  int pos=0;
  //blucle que incrementará el valor de la posición del ángulo
  for(pos = anguloH; pos<=nuevo;pos+=1){
      //enviar el nuevo ángulo a servo
      servoMotorH.write(pos);
      //esperar un tiempo de 180 milisegundos antes de continuar con el proceso
      delay(180);
    }
    //al finalizar el incremento del angulo, se actualiza el valor del angulo actual
  anguloH=nuevo;
}
```

Figura 56. Función para incrementar progresivamente al servomotor horizontal

```
void subirV(int nuevo){
  //variable que se usara para declarar e inicializar el bucle
  int pos=0;
  //blucle que incrementará el valor de la posición del ángulo
  for(pos = 0; pos<=nuevo;pos+=1){
      //enviar el nuevo ángulo a servo
    servoMotorV.write(pos);
      //esperar un tiempo de 180 milisegundos antes de continuar con el proceso
    delay(180);
  }
  //al finalizar el incremento del angulo, se actualiza el valor del angulo actual
  anguloV=nuevo;
}
```

Figura 57. Función para incrementar progresivamente al servomotor vertical

De igual manera, se requiere una función para disminuir el valor de los servomotores, se usó un bucle for que disminuir el valor de la posición del ángulo, al finalizar el decremento del ángulo, se actualiza el valor del ángulo actual al nuevo ángulo final. En la Figura 58 y Figura 59, se muestra que esta función, se debe realizar para el servomotor horizontal y vertical.

```
void regresarH(int valor){
  //variable que se usara para declarar e inicializar el bucle
  int pos=0;
  //blucle que disminuirá el valor de la posición del ángulo
  for(pos = anguloH; pos>=valor;pos-=1){
      //enviar el nuevo ángulo a servo
      servoMotorH.write(pos);
      //esperar un tiempo de 180 milisegundos antes de continuar con el proceso
      delay(180);
    }
  //al finalizar el proceso del angulo, se actualiza el valor del angulo actual
  anguloH=valor;
}
```

Figura 58. Función para incrementar progresivamente al servomotor horizontal

```
void regresarV(){
  //variable que se usara para declarar e inicializar el bucle
  int pos=0;
  //blucle que disminuirá el valor de la posición del ángulo
  for(pos = anguloV; pos>=0;pos-=1){
    //enviar el nuevo ángulo a servo
    servoMotorV.write(pos);
    //esperar un tiempo de 180 milisegundos antes de continuar con el proceso
    delay(180);
  }
  //al finalizar el proceso del angulo, se actualiza el valor del angulo actual
  anguloV=0;
}
```

Figura 59. Función para incrementar progresivamente al servomotor vertical

Además, se incluye la gestión del valor de humedad, en la Figura 60, se presenta una función que permite obtener valores del sensor de humedad y enviarlos al servidor. En esta función de igual manera, se realiza el control del estado para automático, el modo automático inicia el riego cuando el valor de humedad es menor a 30%, por lo cual, se crea una condición que verifique este valor para la activación. La humedad, se mide con el sensor ubicado verticalmente a 15 cm bajo tierra.

```
void actualizarHumedad(){
  //obtener valores del sensor
  valorSensor1 = analogRead(sensorPin1);
  //mapear el valor de la humedad de un rango obtenido a un nuevo rango
  int porcentajeHumedad1 = map(valorSensor1, 0, 4000, 100, 0);
  //se almacena el valor obtenido en la variable global
  humedad=porcentajeHumedad1;
  //llamar a la funcion actualizar humedad para enviar dicho valor
  actualizarHumedadHTTP();
  //verificar si el estado es uno o esta en automático, procede a verific
  if(estado==1){
    //verificar que no haya proceso activo
    if(procesoge==0){
      //verificar el valor de la humedad, si es menor a 30
      //iniciar un nuevo proceso
      //actualizando el estado a uno, para indicar que debe continuar cor
      if (porcentajeHumedad1 < 30) {
        enviarEstado();
      }
    }
  }
}
```

Figura 60. Función para actualizar el valor de humedad en la base de datos

3.6.5 Estructura de la base de datos

La base de datos permite almacenar, organizar y gestionar datos de manera estructurada. En MySQL, la información, se organiza en tablas relacionadas entre sí, y, se accede a través del lenguaje SQL. A continuación, en la Figura 61, se presenta el modelo entidad relación con los datos de información que contiene cada tabla, cabe resaltar que la tabla “prueba”, se encuentra sin relación porque es una tabla auxiliar que, se utiliza para calibrar los servos.

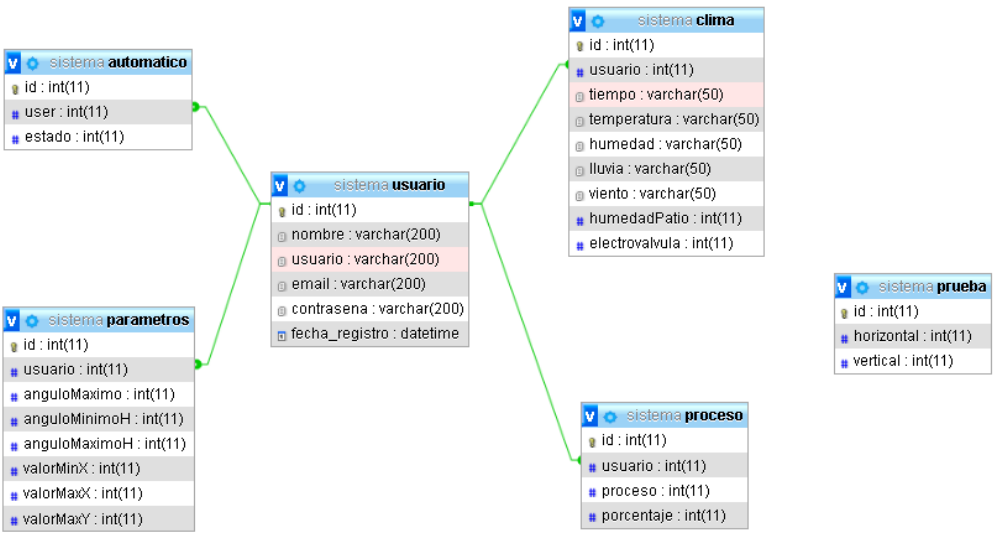
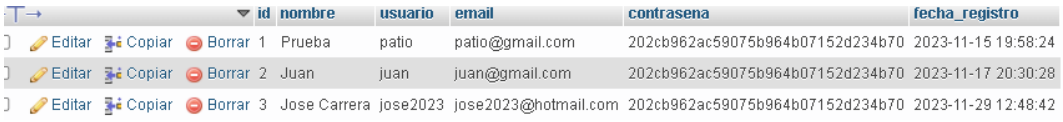


Figura 61. Modelo entidad relación

a. Tabla usuario:

Permite almacenar los datos principales de registro del usuario, los mismos que permite el acceso al sistema como la referencia a cada uno de los procesos. En la Figura 62, se presenta la tabla usuario con los datos almacenados.



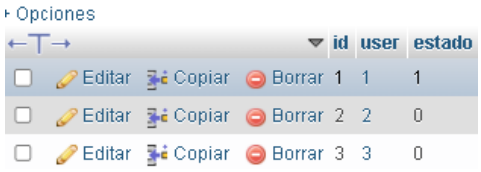
	id	nombre	usuario	email	contrasena	fecha_registro
Editar Copiar Borrar	1	Prueba	patio	patio@gmail.com	202cb962ac59075b964b07152d234b70	2023-11-15 19:58:24
Editar Copiar Borrar	2	Juan	juan	juan@gmail.com	202cb962ac59075b964b07152d234b70	2023-11-17 20:30:28
Editar Copiar Borrar	3	Jose Carrera	jose2023	jose2023@hotmail.com	202cb962ac59075b964b07152d234b70	2023-11-29 12:48:42

Figura 62. Datos de la tabla “usuario”

Esta información, se utiliza para validar el ingreso, verificando los usuarios ingresados actualmente al sistema. La columna “contraseña” almacena los valores de contraseña y los cifra en formato md5 para protegerlos de modificaciones en el envió. Además, tiene una columna llamada “fecha de registro” que, se actualiza automáticamente cuando, se realiza un nuevo registro.

b. Tabla automático:

Permite almacenar el estado actual del sistema si, se encuentra en modo automático o no, en caso de no serlo, se estable el modo manual en la aplicación. Los valores son cero para activado modo manual/desactivado modo automático, y uno para activado modo automático/desactivado modo manual. En la Figura 63, se presenta el estado en el que, se encuentran los usuarios.



Opciones

	id	user	estado
Editar Copiar Borrar	1	1	1
Editar Copiar Borrar	2	2	0
Editar Copiar Borrar	3	3	0

Figura 63. Datos de la tabla “automatico”

c. Tabla parámetros:

Permite almacenar los valores máximos y mínimos de ángulo para controlar el movimiento de los servomotores, además de registrar los valores de la posición del objeto, tal como muestra la Figura 64.

+ Opciones

	id	usuario	anguloMaximo	anguloMinimoH	anguloMaximoH	valorMinX	valorMaxX	valorMaxY
Editar Copiar Borrar	1	1	80	0	180	0	0	0
Editar Copiar Borrar	2	2	70	0	180	0	0	0
Editar Copiar Borrar	3	3	50	0	180	0	0	0

Figura 64. Datos de la tabla “parametros”

Estos valores, se actualizan cuanto el sistema está en funcionamiento. Si hay una detección de objeto, se ingresa la posición en las tres últimas columnas, además los ángulos, se establecen en las pruebas realizadas a diferentes distancias.

d. Tabla clima:

Almacena los valores del clima en tiempo real. Aquí, se almacenan los valores de tiempo actual, temperatura, humedad ambiental, probabilidad de lluvia y velocidad del viento, como, se muestra en la Figura 65. Además, ayuda a compartir la información de humedad del suelo y estado de electroválvula.

· Opciones

	id	usuario	tiempo	temperatura	humedad	lluvia	viento	humedadPatio	electrovalvula
Editar Copiar Borrar	1	1	Nublado	20°C	68%	28%	17	59	1
Editar Copiar Borrar	2	2	Soleado	22°C	68%	15%	7	60	0
Editar Copiar Borrar	3	3	Muy nublado	19°C	37%	27%	7	25	0

Figura 65. Datos de la tabla “clima”

e. Tabla proceso:

Almacena el porcentaje del proceso y el proceso en el que, se encuentra actualmente el sistema. El valor 0 identifica que está detenido el sistema, 1 para indicar que el sistema está capturando una imagen, 2 para indicar que, se está realizando el tratamiento de la imagen, y 3 para indicar que, se está realizando el riego mediante los servomotores. En la Figura 66, se presenta el proceso en el que, se encuentran los usuarios.

+ Opciones

	id	usuario	proceso	porcentaje
Editar Copiar Borrar	1	1	3	50
Editar Copiar Borrar	2	2	0	0
Editar Copiar Borrar	3	3	0	0

Figura 66. Datos de la tabla “proceso”

f. Tabla prueba:

Ayuda a calibrar el sistema en base a la longitud del jardín, permite posicionar los servomotores en ángulos específicos de posición mínima y máxima para los servomotores horizontal y vertical. En la Figura 67, se muestra los valores de id, ángulo de servomotor horizontal y ángulo de servomotor vertical almacenados.

Figura 67. Datos de la tabla “prueba”

A continuación, se detallan los archivos del servidor necesarios para establecer credenciales de conexión, enviar y recibir peticiones de los datos necesarios para el funcionamiento de la aplicación, y realizar el monitoreo.

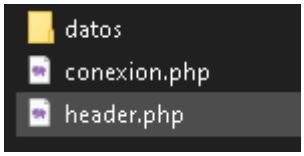


Figura 68. Carpeta principal del sistema

En la Figura 68, se observa que dentro de la carpeta principal, se tiene dos archivos principales de php que permiten obtener los parámetros de conexión a la base de datos. A continuación, se muestra el contenido de cada archivo.

El archivo conexión, de la Figura 69, establece los parámetros de conexión a la base de datos

C: > xampp > htdocs > sistema > conexion.php

```php
<?php
//asignar la direccion del servidor
$servername = "localhost";
//especificar el nombre de la base de datos
$dbname = "sistema";
//establecer el nombre de usuario para conectar a la base de datos
$username = "root";
//establecer la contraseña de usuario para conectar a la base de datos
$password = "";
//crear la conexion a la base de datos enviando los parametros anteriores
$conn = mysqli_connect($servername,$username,$password,$dbname);
//crear nuevo objeto de tipomysqli para la conexion
$mysqli = new mysqli($servername,$username,$password,$dbname);
//establecer configuracino para aceptar caracteres especiales
$mysqli->set_charset("utf8");
//si la conexion es erronea se imprime un mensaje de error
if(!$mysqli){
  //mostrar mensaje de error
  die("Error ".mysqli_connect_error());
}
?>
```

Figura 69. Archivo conexión

El archivo header, presentado en la Figura 70, estable los valores requeridos para permitir el acceso a la base de datos desde diferentes orígenes, también autoriza los métodos para obtener lo datos de la cámara y enviar acciones de control a la ESP32.

C: > xampp > htdocs > sistema > header.php

```php
<?php
    //configura el encabezado http para permitir el acceso desde cualquier origen
    header('Access-Control-Allow-Origin: *');
    //acepta cualquier tipo de encabezados
    header('Access-Control-Allow-Headers: *');
    header("Access-Control-Allow-Headers: Origin, X-Requested-with, Content-type, Accept, Authorization");
    //permite el contenido json
    header("Content-Type: application/json");
    //autoriza los metodos post, get y options
    header("Access-Control-Allow-Methods: POST, GET, OPTIONS");
?>
```

Figura 70. Archivo header

Dentro de la carpeta datos, de la Figura 68, se encuentran los siguientes archivos:

- El archivo registrarUsuario, permite el registro de usuarios, también permite el control del registro en la base a datos, si no son ingresados correctamente genera un mensaje de error. Esto, se observa en la Figura 71.

C: > xampp > htdocs > sistema > datos > registrarUsuario.php

```php
include '../conexion.php';
$json = file_get_contents('php://input');
$obj = json_decode($json,true);
$nombre = $obj['nombre'];
$usuario = $obj['usuario'];
$email = $obj['email'];
$contrasena = md5($obj['contrasena']);
$verificarUsuario = $mysqli->query("SELECT * FROM usuario WHERE usuario='$usuario'");
if ($verificarUsuario->num_rows > 0) {
    echo json_encode(array('ok' => false, 'mensaje' => 'El usuario ya existe en la base de datos'));
} else {
    //si el usuario no existe procede con el registro
    $sql = "INSERT INTO usuario (nombre, usuario, email, contrasena) VALUES (?, ?, ?, ?)";
    $stmt = $mysqli->prepare($sql);
    if ($stmt) {
        //se envian los valores como parametros de tipo string
        $stmt->bind_param("ssss", $nombre, $usuario, $email, $contrasena);
        //ejecutar el codigo sql
        if ($stmt->execute()) {
            //si la ejecución es correcta devuelve un json indicando proceso correcto
            echo json_encode(array('ok' => true, 'mensaje' => ' Si, Datos insertados correctamente en la tab
        } else {
            //si la ejecución es incorrecta devuelve un json indicando el error
            echo json_encode(array('ok' => false, 'mensaje' => 'Error al insertar datos en la tabla "usuario
        }
        $stmt->close(); //cierra la conexion de inserción
    } else {
        echo json_encode(array('ok' => false, 'mensaje' => 'Error al preparar la consulta'));
    }
}
$mysqli->close();
?>
```

Figura 71. Archivo registro usuario

- El archivo iniciarSesion, de la Figura 72, recibe como parámetros el usuario y contraseña, y realiza una consulta a la base de datos para validar si existe el usuario o si los datos están correctos.

C: > xampp > htdocs > sistema > datos > iniciarSesion.php

```php
<?php
include '../header.php';
include '../conexion.php';

$json = file_get_contents('php://input');
$obj = json_decode($json,true);
$usuario = $obj['usuario'];
$contrasena = md5($obj['contrasena']);
$sql = "SELECT * FROM usuario WHERE usuario='$usuario' AND contrasena='$contrasena'";
try {

  if($resultado = $mysqli->query($sql)){

    if(mysqli_num_rows($resultado)>0){

        while($rowData = mysqli_fetch_array($resultado)){
                echo json_encode(array('ok' => true, 'mensaje' => "Si hay Datos",'id'=>$rowData["id"])) ;
        }
    }else{
      echo json_encode(array('estado'=> false, 'mensaje' => "Los datos son incorrectos")) ;
    }
    }else{
      echo json_encode(array('estado'=> false, 'mensaje' => "Los datos son incorrectos")) ;
    }
} catch (\Throwable $th) {

  echo json_encode(array('estado'=> false, 'mensaje' => "Error: ".$th)) ;
}

$mysqli->close();
?>
```

Figura 72. Archivo iniciar sesión

- El archivo proceso permite obtener los valores tanto del estado si el sistema está en automático, en la Figura 73, se muestra como el dato en que el proceso, se encuentra actualmente, y el valor de la humedad enviada desde el sensor

C: > xampp > htdocs > sistema > datos > proceso.php

```php
<?php
    include '../header.php';
    include '../conexion.php';
    $param = json_decode(file_get_contents("php://input"));
    $myArray=[];
    //seleciona todos los valores de la tabla automatico, ademas se realiza una subconsulta
    //para obtener el proceso actual, humedad, porcentaje de proceso y estado de electrovalvula
    $sql = "SELECT estado,
    IFNULL((SELECT proceso FROM proceso ORDER BY id DESC LIMIT 1),0) as proceso,
    IFNULL((SELECT humedadPatio FROM clima ORDER BY id DESC LIMIT 1),0) as humedad,
    IFNULL((SELECT porcentaje FROM proceso ORDER BY id DESC LIMIT 1),0) as porcentaje,
    IFNULL((SELECT electrovalvula FROM clima ORDER BY id DESC LIMIT 1),0) as electro
    FROM automatico";
    //ejecuta el codigo sql
    if ($result = $mysqli->query($sql)){
        //recorrer mientras haya informacion
    while($row = $result->fetch_assoc()) {
        //almacenar cada fila en el array
        $myArray[] = $row;
    }
    //devolver datos almacenados en el array
    echo json_encode($myArray);
    }
    else{
    echo json_encode(array('mens'=>'No hay datos'));
    }
    $mysqli->close();
?>
```

Figura 73. Archivo proceso

- El archivo clima permite obtener los valores de los parámetros antes definidos en la tabla “clima”. En la Figura 74, se muestra los datos a los cuales, se realiza la consulta en la base de datos.

C: > xampp > htdocs > sistema > datos > clima.php

```php
$obj = json_decode($json,true);
//seleciona todos los valors de la tabla clima, ademas se realiza varias subconsultas para utilizar la misma co
//como el estado de automático desde la tabla automático, el proceso del sistema y porcentaje del proceso desde
$sql = "SELECT *, ifnull((SELECT estado FROM automatico),0) as automatico,
ifnull((SELECT proceso FROM proceso ORDER BY id DESC LIMIT 1),0) as proceso,
IFNULL((SELECT humedadPatio FROM clima ORDER BY id DESC LIMIT 1),0) as humedadPatio,
IFNULL((SELECT electrovalvula FROM clima ORDER BY id DESC LIMIT 1),0) as electro,
IFNULL((SELECT porcentaje FROM proceso ORDER BY id DESC LIMIT 1),0) as porcentaje
FROM clima ";
try {
  if($resultado = $mysqli->query($sql)){
    //si la respuesta tiene datos
    if(mysqli_num_rows($resultado)>0){
      //recorrer cada fila
        while($rowData = mysqli_fetch_array($resultado)){
          //devolver un array con los datos obtenidos
                    echo json_encode(
                      array('ok' => true,
                      'mensaje' => "Si hay Datos",
                      'tiempo'=>$rowData["tiempo"],
                      'temperatura'=>$rowData["temperatura"],
                      'humedad'=>$rowData["humedad"],
                      'lluvia'=>$rowData["lluvia"],
                      'automatico'=>$rowData["automatico"],
                      'proceso'=>$rowData["proceso"],
                      'humedadPatio'=>$rowData["humedadPatio"],
                      'electro'=>$rowData["electro"],
                      'viento'=>$rowData["viento"],
                      'porcentaje'=>$rowData["porcentaje"])) ;
      }
    }else{
      echo json_encode(array('estado'=> false, 'mensaje' => "No hay informacion")) ;
```

Figura 74. Archivo clima

- Archivo actualizarClima de la Figura 75 muestra cómo, se recibe los valores del clima en tiempo real y los almacena en la base de datos

C: > xampp > htdocs > sistema > datos > actualizarClima.php

```php
include '../header.php';
include '../conexion.php';
//verfiica si se ha recibido los parametro con metodo get
if (isset($_GET['tiempo']) && isset($_GET['temperatura']) && isset($_GET['humedad']) && isset($_GET['lluvia']))
    //recuperar los valores recibidos
    $tiempo = $_GET['tiempo'];
    $temperatura = $_GET['temperatura'];
    $humedad = $_GET['humedad'];
    $lluvia = $_GET['lluvia'];
    $viento = $_GET['viento'];
    //se construye la consulta sql
    //actualiza el estado en la tabla automático con el valor recibido
    $sql = "UPDATE clima SET tiempo='$tiempo', temperatura='$temperatura', humedad='$humedad', lluvia='$lluvia',
    try {
        //ejecuta el codigo sql
        if ($mysqli->query($sql) === TRUE) {
            //si la ejecución es correcta devuelve un json indicando proceso correcto
            echo json_encode(array('ok' => true, 'mensaje' => "Datos actualizados correctamente en la tabla 'cli
        } else {
            //si la ejecución es incorrecta devuelve un json indicando el error
            echo json_encode(array('ok' => false, 'mensaje' => "Error al actualizar datos en la tabla 'clima'"))
        }
    } catch (\Throwable $th) {
        //devuelve un json indicando el error
        echo json_encode("Error");
    }
    $mysqli->close();
} else {
    //devuelve un json indicando el error
    echo json_encode(array('ok' => false, 'mensaje' => "Faltan parámetros en la URL"));
}
?>
```

Figura 75. Archivo actualizar clima

- El archivo actualizar parámetros de la Figura 76 permite el registro de los parámetros necesarios para el funcionamiento de los servomotores y el riego.

C: > xampp > htdocs > sistema > datos > actualizarParametros.php

```php
//verfiica si se ha recibido los parametros con metodo get
if (isset($_GET['anguloMaximo']) && isset($_GET['anguloMinimoH']) && isset($_GET['anguloMaximoH']) && isset($_GE
    //recuperar los valores recibidos
    $anguloMaximo = $_GET['anguloMaximo'];
    $anguloMinimoH = $_GET['anguloMinimoH'];
    $anguloMaximoH = $_GET['anguloMaximoH'];
    $valorMinX = $_GET['valorMinX'];
    $valorMaxX = $_GET['valorMaxX'];
    $valorMaxY = $_GET['valorMaxY'];
    //se construye la consulta sql
    //actualizalos valores en la tabla oarámetros
    $sql = "UPDATE parametros SET anguloMaximo='$anguloMaximo', anguloMinimoH='$anguloMinimoH', anguloMaximoH='$
    try {
        //ejecuta el codigo sql
        if ($mysqli->query($sql) === TRUE) {
            //si la ejecución es correcta devuelve un json indicando proceso correcto
            echo json_encode(array('ok' => true, 'mensaje' => "Datos actualizados correctamente en la tabla 'par
        } else {
            //si la ejecución es incorrecta devuelve un json indicando el error
            echo json_encode(array('ok' => false, 'mensaje' => "Error al actualizar datos en la tabla 'parametro
        }
    } catch (\Throwable $th) {
        //devuelve un json indicando el error
        echo json_encode(array('ok' => false, 'mensaje' => "Error"));
    }
    //cierra la conexion a la base de datos
    $mysqli->close();
} else {
    //devuelve un json indicando el error
    echo json_encode(array('ok' => false, 'mensaje' => "Faltan parámetros en la URL"));
}
?>
```

Figura 76. Archivo actualizar parámetros

- El archivo actualizarAutomático de la Figura 77 permite actualizar el estado del sistema en modo automático

C: > xampp > htdocs > sistema > datos > actualizarAutomatico.php

```php
<?php
include '../header.php';
include '../conexion.php';
//verfiica si se ha recibido un parametro con metodo get
if (isset($_GET['valor'])) {
    //recuperar el valor recibido
    $valor = $_GET['valor'];
    //declara e inicializa un array vacío donde se almacenaran los datos
    $myArray = [];
    //se construye la consulta sql
    //actualiza el estado en la tabla automático con el valor recibido
    $sql = "UPDATE automatico SET estado='$valor' ";
    try {
        //ejecuta el codigo sql
        if ($mysqli->multi_query($sql) === TRUE) {
            //si la ejecución es correcta devuelve un json indicando proceso correcto
            echo json_encode(array('ok' => true, 'mensaje' => "Proceso Correcto"));
        } else {
            //si la ejecución es incorrecta devuelve un json indicando el error
            echo json_encode(array('ok' => false, 'mensaje' => "Los datos son incorrectos"));
        }
    } catch (\Throwable $th) {
        //devuelve un json indicando el error
        echo json_encode(array('ok' => false, 'mensaje' => "Error"));
    }
    //cierra la conexion a la base de datos
    $mysqli->close();
} else {
    //devuelve un json indicando el error
    echo json_encode(array('ok' => false, 'mensaje' => "'valor' no está presente en la URL"));
}
?>
```

Figura 77. Archivo actualizar automatico

- El archivo actualizar estado permite actualizar el valor del proceso. El código, se muestra en la Figura 78.

C: > xampp > htdocs > sistema > datos > actualizarEstado.php

```php

<?php
include '../header.php';
include '../conexion.php';
//permite leer y decodificar los datos json que ingresan
$param = json_decode(file_get_contents("php://input"));
//extrae y almacena el parametro recibido
$valor=$param->valor;
//actualiza proceso dentro de la tabla proceso
$sql = "UPDATE proceso SET proceso ='$valor' ";
try {
    //ejecuta el codigo sql
    if($mysqli->multi_query($sql) === TRUE){
      //si la ejecución es correcta devuelve un json indicando proceso correcto
          echo json_encode(array('ok' => true, 'mensaje' => "Proceso Correcto")) ;
      }else{
        //si la ejecución es incorrecta devuelve un json indicando el error
        echo json_encode(array('ok'=> false, 'mensaje' => "Los datos son incorrectos")) ;
      }
  } catch (\Throwable $th) {
    //devuelve un json indicando el error
    echo json_encode(array('ok' => false, 'mensaje' => "Error"));
  }
$mysqli->close();
?>
```

Figura 78. Archivo actualizar estado

- El archivo cambiarEstado permite aumentar el proceso en uno, sirve para ir cambiando el estado del proceso cuando termina cada actividad. Se ilustra este archivo en la Figura 79.

C: > xampp > htdocs > sistema > datos > cambiarestado.php

```php
<?php
include '../header.php';
include '../conexion.php';
//permite leer y decodificar los datos jason que ingresan
$param = json_decode(file_get_contents("php://input"), true);
//que actualiza el valor de proceso dentro de la tabla proceso
//lee el valor de proceso y lo incrementa en uno
$stmtIncrementarProceso = $mysqli->prepare('UPDATE proceso SET proceso = proceso + 1');
//ejecuta el codigo
$stmtIncrementarProceso->execute();
//cierra el proceso commit
$stmtIncrementarProceso->close();
//cierra la conexion
$mysqli->close();
?>
```

Figura 79. Archivo cambiar estado

3.7 Desarrollo de la interfaz

La norma ISO 9241-11 establece los parámetros que, se deben considerar en la construcción de una interfaz usable en proyectos de desarrollo de software. Los requisitos ergonómicos para trabajos de oficina con pantallas, se establecen por los usuarios, objetivos y componentes. En base a sus necesidades de visualización de datos, se basa en:

- Eficacia: Metas y objetivos
- Eficiencia: Resultados deseados con menor consumo de recursos.
- Satisfacción: Confort del usuario respecto al sistema.

De acuerdo con el apartado de usabilidad en el contexto de uso, los parámetros que deben tenerse en cuenta al momento de construir una aplicación son:

- Nivel de especificidad: Complejos o partes individuales del sistema
- Especificar objetivos y aspectos de uso
- Usuarios especificados para el uso
- Identificar objetivos especificados para el uso

- Combinaciones posibles de uso

En base a los parámetros de uso especificados por la norma 9241-11, se considera un nivel de especificidad complejo debido a que el sistema requiere la interacción de diferentes elementos de hardware y software. Además de que, se planea obtener un producto para el riego y un servicio de monitoreo, que sean interactivos para con el usuario. Dentro de la interfaz, se ubican en orden de importancia los diferentes elementos para comprensión rápida por parte del usuario, se plantea la idea de usar un color claro en la interfaz que evite elementos distractorios y molestos a la vista.

Se coloca etiquetado en cada área con la información relevante para cada apartado, dentro de las áreas seccionadas, se tiene:

- Clima: Valores relevantes que debe conocer el usuario para realizar un riego adecuado.

- Suelo: Porcentaje de humedad de la tierra actual y semáforo indicador con recomendaciones referentes a la humedad para empezar el riego

- Control: Indicadores y botones de control de modo de funcionamiento del sistema riego.

- Proceso: Porcentaje de proceso realizado por el sistema, útil para conocer si el riego está por iniciar o terminar.

- Secciones de Imágenes: Diferentes apartados para ubicar las imágenes procesadas de detección y contorno para césped y objetos.

3.7.1 Interfaz de computadora

Inicialmente, se genera una ventana de inicio de sesión para los usuarios, como muestra la Figura 80, se deriva a dos opciones: el ingreso para usuarios previamente registrados y el registro para nuevos usuarios.

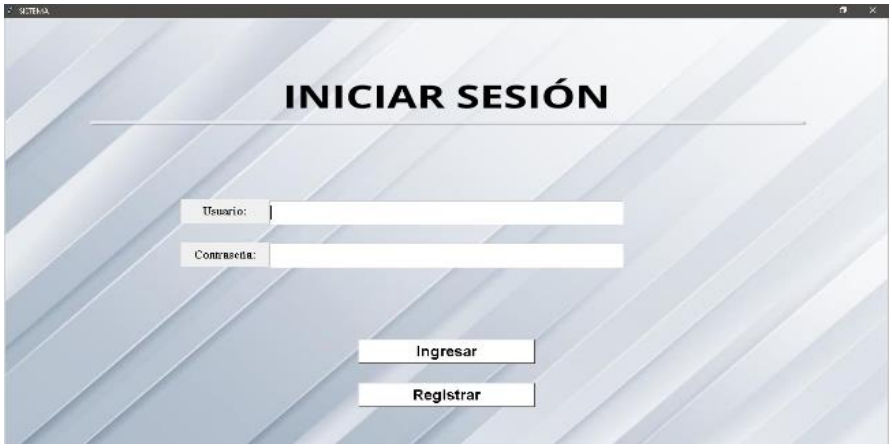


Figura 80. Ventana de iniciar de sesión

En la ventana principal, se resalta el inicio de sesión que cuenta con dos cajas de texto para ingresar el usuario y contraseña, y botones para ingresar y registrar. En la Figura 81, se presenta el código de la creación las ventanas para el sistema, la variable ventana, se usa para la interfaz de iniciar sesión, la variable datos, se usa para la interfaz principal del sistema.

```
ventana = tkinter.Tk()
ventana.state("zoomed")
ventana.title("SISTEMA")
ventana.config(bg="grey")
datos = tkinter.Toplevel(ventana)
datos.state("withdraw")
datos.title("SISTEMA")
datos.config(bg="#d9d9d9")
```

Figura 81. Creación de ventana inicio de sesión y ventana principal.

Para la creación de las cajas de texto, se usa el código de la Figura 82, en donde los parámetros son la ventana, el tipo de fuente y tamaño de letra. Además, se define la posición de ubicación para cada caja de texto requerida.

```
entryInicioUsuario = tkinter.Entry(ventana, font=('Arial', 16))
entryInicioUsuario.place(relx=0.5, rely=0.55, anchor='center', relwidth=0.4, height=40)

entryInicioContrasena = tkinter.Entry(ventana, font=('Arial', 16), show='*')
entryInicioContrasena.place(relx=0.5, rely=0.65, anchor='center', relwidth=0.4, height=40)
```

Figura 82. Ingreso de usuario y contraseña

En la Figura 83, para la creación del botón Ingresar, se usa el siguiente código, en el cual, se envía como parámetros la ventana en la que, se mostrará el botón, además, se

envía el color de fondo y color de texto, como la posición a mostrar, también, se ingresa el nombre de la función o proceso que, se ejecutará al presionar en el botón. En la Figura 84, se muestra como el código es similar para la creación del botón registrar con el botón ingresar.

```
btnContinuar = tkinter.Button(ventana, text="Ingresar", command=abrirPrincipal, fg="#000000", bg="#ffffff")
#configura ubicación dentro de la ventana
btnContinuar.place(relx=0.5, rely=0.8, anchor='s', relwidth=0.2, height=40)
#configurar el estilo del texto
btnContinuar['font'] = font.Font(size=18, weight="bold")
```

Figura 83. Botón Ingresar

```
btnRegistrar = tkinter.Button(ventana, text="Registrar", command=abrirRegistrar, fg="#000000", bg="#ffffff")
#configura ubicación dentro de la ventana
btnRegistrar.place(relx=0.5, rely=0.9, anchor='s', relwidth=0.2, height=40)
#configurar el estilo del texto
btnRegistrar['font'] = font.Font(size=18, weight="bold")
```

Figura 84. Botón Registrar

Todos los datos ingresados en el inicio de sesión son verificados en la base de datos, a continuación, se muestran los diferentes códigos para verificación de ingreso de usuario y para el registro de nuevo usuario que, se realiza directamente en la base de datos.

La función abrirPrincipal de la Figura 85 utiliza variables “global” que permiten hacer uso de las variables que, se encuentran fuera de la función en proceso, también, se procede a obtener el valor que, se encuentra en el usuario y contraseña ingresada por el usuario, después, se llama a la función iniciar sesión. Si el resultado de esta función tiene la palabra “Si” el ingreso es correcto, y, se procede a ejecutar funciones principales como obtener el proceso en el que, se encuentra el sistema, los datos del clima actual, y para finalizar, se minimiza la ventana de iniciar sesión y, se procede a abrir la ventana Principal. En caso que el resultado de la función iniciarSesion no tenga la palabra “Si”, muestra un mensaje indicando al usuario que los datos ingresados son incorrectos

```python
def abrirPrincipal():
    global entryInicioUsuario
    global entryInicioContrasena
    usuario = entryInicioUsuario.get()
    contrasena = entryInicioContrasena.get()
    controlIS = iniciarSesion(usuario, contrasena)
    if 'Si' in controlIS.get('mensaje', ''):
        hilo_proceso = threading.Thread(target=proceso)
        hilo_proceso.start()
        valoresClima = threading.Thread(target=clima)
        valoresClima.start()
        datos.state(newstate="zoomed")
        ventana.state(newstate="withdraw")
    else:
        messagebox.showerror("Error", "Los datos son incorrectos")
```

Figura 85. Función abrir principal para abrir la ventana principal

Función iniciar Sesión, permite declarar la variable url con la dirección del archivo php que, se encuentra en el servidor, el mismo que recibe como parámetros el usuario y contraseña, y devuelve el mensaje si el ingreso es correcto o incorrecto. Este código, se presenta en la Figura 86.

```python
def iniciarSesion(usuario, contrasena):
    #crear una variable que contenga la url a la que se realizara la peticion
    url = "http://localhost/sistema/datos/IniciarSesion.php"
    #definir parametros a enviar
    datos = {'usuario': usuario, 'contrasena': contrasena}
    # Realiza una solicitud a la URL para obtener respuesta, esta es de tipo POST.
    respuesta = requests.post(url, json=datos)
    #retornar resultado en formato JSON
    print(respuesta)
    return respuesta.json()
```

Figura 86. Función para verificar existencia o datos correctos de usuario

En la Figura 87, Figura 88 y Figura 89, se tiene la ventana generada para el registro de nuevos usuarios, en esta ventana, se solicitan los datos de nombre, nombre de usuario, email, contraseña y repetir contraseña para realizar una verificación. Hecha la verificación, el nuevo usuario es almacenado en la base de datos.

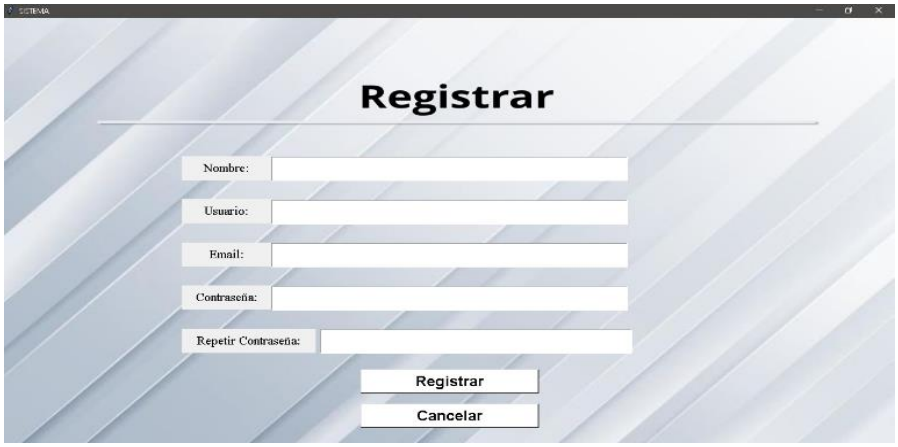


Figura 87. Ventana de registro de usuario

```python
def guardarDatos():
    #global: permite llamar y hacer uso de una variable global. entry: valores que recibe el sistema al regis
    global entryNombre
    global entryUsuario
    global entryContrasena
    global entryRepetirContrasena
    global entryEmail
    #se almacena en las variables los valores a registrar del nuevo usuario
    nombre = entryNombre.get()
    usuario = entryUsuario.get()
    contrasena = entryContrasena.get()
    confirmar = entryRepetirContrasena.get()
    email = entryEmail.get()
    #se valida que todos los datos esten llenos, ya que son obligatorios
    if not nombre or not usuario or not contrasena or not confirmar or not email:
        messagebox.showerror("Error", "Todos los datos son obligatorios")
        #salir de la función y evita continuar con el registro
        return
    #se valida las dos contraseñas sean iguales
    if contrasena != confirmar:
        messagebox.showerror("Error", "Las contraseñas no son iguales")
        return
    #si los controles anterior son correctos se procede a enviar los datos a la funcion registrarUsuario
    controlRegistro = registrarUsuario(nombre,usuario,email,contrasena)
    #si el texto devuelto desde el servidor contiene la palabra si, quiere decir que el registro fue correcto
    if 'Si' in controlRegistro.get('mensaje', ''):
        messagebox.showerror("Información", "Registro Correcto")
        #procede a volver a la funcion carátula que abre la ventana de iniciar sesión
        abrirCaratula()
    else:
        #si el registro no fue correcto se prensenta un mensaje indicando el error
        messagebox.showerror("Error", controlRegistro.get('mensaje', ''))
```

Figura 88. Función guardar datos para registrar nuevo usuario

```python
def registrarUsuario(nombre,usuario,email,contrasena):
    #transformar lo datos a texto para enviarlos
    nombre_encoded = str(nombre)
    usuario_encoded = str(usuario)
    email_encoded = str(email)
    contrasena_encoded = str(contrasena)
    #definir la direccion url
    url = "http://localhost/sistema/datos/registrarUsuario.php"
    #definir parametros a enviar
    datos = {'nombre': nombre_encoded, 'usuario': usuario_encoded,'email': email_encoded, 'contrasena': contras
    # Realiza una solicitud a la URL para obtener respuesta, esta es de tipo POST.
    respuesta = requests.post(url, json=datos)
    #retornar resultado en formato JSON
    return respuesta.json()
```

Figura 89. Función para ingresar nuevos usuarios en la base de datos

Una vez ingresados correctamente los datos del usuario, se despliega la ventana principal donde la interfaz cuenta con los datos requeridos para conocimiento del usuario, y para realizar el proceso de detección y riego. En la Figura 90, se muestra la ventana principal donde, se resaltan las áreas etiquetadas establecidas en el apartado de desarrollo de interfaz.

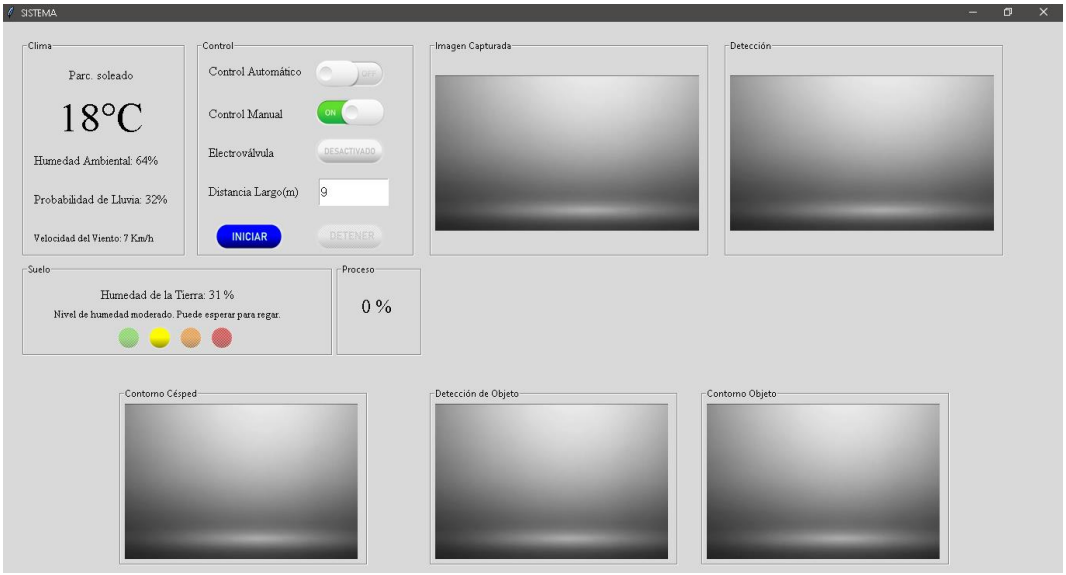


Figura 90. Ventana Principal

A continuación, se detalla el desarrollo de cada área de la ventana principal con sus respectivas funciones para el inicio y control de los diferentes procesos.

La función Proceso, inicialmente permite llamar a la función traerProceso la cual devuelve los valores de proceso, estado, porcentaje, humedad, electroválvula, y humedad de la tierra. En la Figura 91, se muestra esa parte inicial del código.

```
y > proceso
global imgElectro
global desativado_image_tk
global ativado_image_tk
global labelProceso
#proceso que siempre estará en ejecución
while True:
    #funcion que permite traer los datos en tipo Json
    tDatos = traerProceso()
    #transforma de tipo JSON a array para poder recorrerlos
    a = np.array(json.loads(tDatos))
    #recorrer los datos obtenidos
    for x in a:
        #almacenar en variables los datos del servidor obteniendo con el nombre de los mismos
        dato = x['proceso']
        #verifica si el programa esta en modo automatico o manual
        datoA = x['estado']
        #en que porcentaje se encuentra el sistema
        porcentaje = x['porcentaje']
        #humendad que es ingresada desde la esp con el sensor de humedad
        datoHumedad = x['humedad']
        #valor que indicar que la eletro válvula esta activa
        electro = x['electro']
        #ubicar el valor de la humedad en un label para ser visualizado en interfaz
        labelSensorHumedad.config(text=f"Humedad de la Tierra: {datoHumedad} %")
        #obter mensaje a traves de la funcion que recibe la humedad
```

Figura 91. Función proceso llamando a la función traerProceso

Todos esos datos son requeridos para controlar en que etapa del proceso, se encuentra el sistema. Además, en la Figura 92, se verifica el estado de la electroválvula, se obtiene el valor de la humedad obtenida por el sensor, y si la aplicación está en modo automático.

```
labelSensorHumedad.config(text=f"Humedad de la Tierra: {datoHumedad} %")
#obter mensaje a traves de la funcion que recibe la humedad
mensajeHumedad=obtener_mensaje(float(datoHumedad))
#muestra el mensaje en el label correspondiente
labelSensorHumedadDescripcion.config(text=f"{mensajeHumedad}")
#ubica el valor del porcentaje actual del proceso
labelProceso.config(text=f"{porcentaje} %")
#verifica si la electro valvula este inicializada
if (electro=="1"):
    #activa la indicador
    imgElectro.config(image=ativado_image_tk)
else:
    #desactiva el indicador
    imgElectro.config(image=desativado_image_tk)
```

Figura 92. Función proceso obteniendo valor de humedad y estado de electroválvula

Para verificar el estado, cuando, se encuentra en modo automático, se requiere que el valor de estado sea igual a 1, el sistema envía un dato para que el botón se desactive e indique el texto de “Desactivar”. Caso contrario el botón mostrará activar. Ocurre el mismo proceso en inverso para el modo manual cuando el valor es 0, en la Figura 93 se muestra la programación usada en los botones.

```
if (datoA=="1"):
    #almacena en la variable global el estado actual
    valorAutomatico = 1
    #muestra  la imagen de estado automático activado
    botonAutomatico.config(image=on_image_tk)
    #muestra  la imagen de estado manual desactivado
    botonManual.config(image=off_image_tk)
else:
    #almacena en la variable global el estado actual
    valorAutomatico = 0
    #muestra  la imagen de estado automático desactivado
    botonAutomatico.config(image=off_image_tk)
    #muestra  la imagen de estado manual activado
    botonManual.config(image=on_image_tk)
```

Figura 93. Función proceso obteniendo el estado de modo Automático o Manual

En la Figura 94, se genera una condición en la que, si el proceso es mayor o igual a uno, la interfaz desactivará el indicador de “Iniciar” y activará el indicador de “Detener”. Se verifica el dato del proceso. Si está en el proceso 1, se inicia la captura de imagen desde la ESP32-CAM, y si está en el proceso 2, empezara el tratamiento de la imagen para la detección.

```
if int(dato) >= 1:
    #almacena en la variable global el estado actual
    valorEstado = 1
    #muestra  la imagen del proceso para iniciar desactivado
    botonIniciar.config(image=iniciarD_image_tk)
    #muestra  la imagen del proceso para detener activado
    botonDetener.config(image=detenerA_image_tk)
    #desactiva el boton de iniciar
    botonIniciar.configure(state='disabled')
    #activa el boton detener
    botonDetener.configure(state='normal')
    #si el proceso está en estado 1
    if int(dato) == 1:
        #llamar a la funcion cargar imagen que obtiene la imagen desde la es
        capturarImagen()
    #si el proceso está en estado 2
    if int(dato) == 2:
        #debe ininciar le proceso de tratamiento de la imagen obtenida
        ventanaPrincipal()
```

Figura 94. Función proceso para capturar imagen y procesarla

Si el proceso, se encuentra en el valor cero, como muestra la Figura 95, entonces el sistema, se encuentra sin iniciar, aquí, se actualizar el valor de porcentaje de proceso a 0%, se tiene la activación del botón “Iniciar” y, se tiene al “Desactivar” en un estado desactivado. Se define un tiempo de espera de tres segundos para continuar.

```
    else:
        #almacena en la variable global el estado actual
        valorEstado = 0
        #el porcentaje regresaría a cero
        actualizarPorcentaje(0)
        #muestra  la imagen del proceso para iniciar activado
        botonIniciar.config(image=iniciarA_image_tk)
        #muestra  la imagen del proceso para detener desactivado
        botonDetener.config(image=detenerD_image_tk)
        #activa el boton de iniciar
        botonIniciar.configure(state='normal')
        #desactiva el boton de detener
        botonDetener.configure(state='disabled')
#permite que el programa espere 3 segundos antes de continuar
time.sleep(3)
```

Figura 95. Función proceso cuando no ha iniciado

En la Figura 96, Figura 97, Figura 98, y Figura 99, se presenta el desarrollo de la función Clima, la cual, permite obtener el código html de la sección de clima de Microsoft donde, se obtiene los valores del clima actual, una vez que, se tiene el código, se procede a buscar las etiquetas que contiene cada valor, se lo busca por id y por el nombre que, se dio en la clase. Al terminar la función, se definió un tiempo de espera de 4 segundos

```
def clima():
    #global: permite llamar y hacer uso de una variable global o variable que se usa fuera esta funcion
    #los label son los textos donde se visualizaran estos valores
    global labelClima
    global labelTiempo
    global labelHumedadActual
    global labelLluvia
    global labelViento
    #funcion repetitiva que estará activa siempre
    while True:
        #variable donde se almacena la url para obtener los datos del clima
        url_pagina = 'https://www.msn.com/es-xl/el-tiempo/pronostico/in-Ambato,Tungurahua?loc=eyJsIjoiQW1iYXRv
        #obtener el codigo html de la página
        codigo_html = obtener_codigo_html(url_pagina)
        #inicializar los datos con valores por defecto
        tiempo="_"
        temperatura="0"
        humedad="0"
        lluvia="0"
        viento="0"
        #condición que evalua si hay codigo html
        if codigo_html:
            #analiza el codigo html y permite la extración de datos
            soup = BeautifulSoup(codigo_html, 'html.parser')
            #busca en el código el elemento con el id definido
            elemento = soup.find('div', id='CurrentWeatherFeedback')
            #condición que valida si encuentra el elemento
```

Figura 96. Función clima con adquisición de valores

```
clima
if codigo_html:
    #analiza el codigo html y permite la extración de datos
    soup = BeautifulSoup(codigo_html, 'html.parser')
    #busca en el código el elemento con el id definido
    elemento = soup.find('div', id='CurrentWeatherFeedback')
    #condición que valida si encuentra el elemento
    if elemento:
        #obteniene el valor del elemento
        valor_numerico = elemento.get_text().split(' ')[0]
        #lo almacena en la variable declarada e inicializada anteriormente
        temperatura=valor_numerico
        #atualiza el valor en el label que se mostrará en la interfaz
        labelClima.config(text=f"{valor_numerico}")
    #busca en el código el elemento con el id definido
    elemento = soup.find('div', id='OverviewCurrentTemperature')
    #condición que valida si encuentra el elemento
    if elemento:
        #busca en el código el elemento con el id definido
        elemento = soup.find('div', class_='u1SummaryCaptionCompact-E1_1')
        #condición que valida si encuentra el elemento
        if elemento:
            #obteniene el valor del elemento
            descripcion = elemento.get_text(strip=True)
            #lo almacena en la variable declarada e inicializada anteriormente
            tiempo=descripcion
            #atualiza el valor en el label que se mostrará en la interfaz
            labelTiempo.config(text=f"{descripcion}")
        #caso contrario al no encontrar el elemento
        else:
            #emite un mensaje indicando una lectura nula
            print("Clima: N/A")
    #caso contrario al no encontrar el elemento
```

Figura 97. Función clima con petición para la variable de clima

```
clima
            print("Clima: N/A")
    #caso contrario al no encontrar el elemento
    else:
        #emite un mensaje indicando una lectura nula
        print("Precipitaciones: N/A")
    #busca en el código el elemento con el id definido
    elemento = soup.find('div', id='CurrentDetailLineHumidityValue')
    #condición que valida si encuentra el elemento
    if elemento:
        #obteniene el valor del elemento
        valHumedad = elemento.get_text().split(' ')[0]
        #lo almacena en la variable declarada e inicializada anteriormente
        humedad=valHumedad
        #atualiza el valor en el label que se mostrará en la interfaz
        labelHumedadActual.config(text=f"Humedad Ambiental: {valHumedad}")
    #busca en el código el elemento con el id definido
    elemento = soup.find('div', id='ForecastDays')
    #condición que valida si encuentra el elemento
    if elemento:
        #busca en el código el elemento con el id definido
        elemento = soup.find('div', class_='precipitationV3-E1_1')
        #condición que valida si encuentra el elemento
        if elemento:
            #obteniene el valor del elemento
            valLluvia = elemento.get_text(strip=True)
            #lo almacena en la variable declarada e inicializada anteriormente
            lluvia=valLluvia
            #atualiza el valor en el label que se mostrará en la interfaz
            labelLluvia.config(text=f"Probabilidad de Lluvia: {valLluvia}")
        else:
            #caso contrario al no encontrar el elemento
            print("Lluvia: N/A")
```

Figura 98. Función clima con petición para la variable de humedad y probabilidad d lluvia

```
clima
            print("Lluvia: N/A")
        #busca en el código el elemento con el id definido
        elemento = soup.find('div', id='CurrentDetailLineWindValue')
        #condición que valida si encuentra el elemento
        if elemento:
            #busca en el código el elemento con el id definido
            valViento = elemento.get_text().split(' ')[0]
            #lo almacena en la variable declarada e inicializada anteriormente
            viento=valViento
            #atualiza el valor en el label que se mostrará en la interfaz
            labelViento.config(text=f"Velocidad del Viento: {valViento} Km/h")
        #envia los parametros del clima a la funcion que envia los valores a la base
        actualizarClima(tiempo,temperatura,humedad,lluvia,viento)
    #permite que el programa espere 4 segundos antes de continuar
    time.sleep(4)
```

Figura 99. Función clima con petición para la variable velocidad de viento

La función para controlar el semáforo indicador de humedad de la tierra, se presenta en la Figura 100 y Figura 101. Se crea la función obtener mensaje para el sensor de humedad.

El semáforo usa un rango de humedad determinado que, indica el valor de humedad y una descripción a modo de sugerencia para que el usuario tenga en consideración al momento de realizar el riego en modo manual. Los rangos que usa el semáforo para activar los colores y las sugerencias son: 0 – 10 para color rojo, 11 – 30 para color naranja, 31- 40 para color amarillo, y 41 – 100 para color verde.

```
def obtener_mensaje(sensor_humedad):
    #global: permite llamar y hacer uso de una variable global o variable que se us
    global imgVerde
    global imgAmarillo
    global imgNaranja
    global imgRojo
    #si el sensor es mayor a 40, 41-100
    if sensor_humedad > float(40):
        #Se activa el indicador de verde y se desactiva el resto de indicadores
        imgVerde.configure(state='normal')
        imgAmarillo.configure(state='disabled')
        imgNaranja.configure(state='disabled')
        imgRojo.configure(state='disabled')
        #retorna el mensaje correspondiente a este nivel
        return "Nivel de humedad alto. No es necesario regar."
    #si el sensor es mayor a 30, 31-40
    elif sensor_humedad > float(30):
        #Se activa el indicador de amarillo y se desactiva el resto de indicadores
        imgVerde.configure(state='disabled')
        imgAmarillo.configure(state='normal')
        imgNaranja.configure(state='disabled')
        imgRojo.configure(state='disabled')
        #retorna el mensaje correspondiente a este nivel
        return "Nivel de humedad moderado. Puede esperar para regar."
    #si el sensor es mayor a 10, 11-30
    elif sensor_humedad > float(10):
        #Se activa el indicador de naranja y se desactiva el resto de indicadores
        imgVerde.configure(state='disabled')
        imgAmarillo.configure(state='disabled')
        imgNaranja.configure(state='normal')
        imgRojo.configure(state='disabled')
        #retorna el mensaje correspondiente a este nivel
```

Figura 100. Función obtener mensaje para el sensor de humedad de 10 a 40

```
by > obtener_mensaje
elif sensor_humedad > float(10):
    #Se activa el indicador de naranja y se desactiva el resto de indicadores
    imgVerde.configure(state='disabled')
    imgAmarillo.configure(state='disabled')
    imgNaranja.configure(state='normal')
    imgRojo.configure(state='disabled')
    #retorna el mensaje correspondiente a este nivel
    return "Nivel de humedad bajo. Se considera regar."
#si el sensor es mayor o igual a 0, 0-10
elif sensor_humedad >= float(0):
    #Se activa el indicador de rojo y se desactiva el resto de indicadores
    imgVerde.configure(state='disabled')
    imgAmarillo.configure(state='disabled')
    imgNaranja.configure(state='disabled')
    imgRojo.configure(state='normal')
    #retorna el mensaje correspondiente a este nivel
    return "Hora de regar. El nivel de humedad es muy bajo."
else:
    #retorna un mensaje de error al no encontrar el valor de la humedad
    return "Error: El valor del sensor de humedad no es válido."
```

Figura 101. Función obtener mensaje para el sensor de humedad de 0 a 10

La función actualizar porcentaje, permite actualizar los datos de porcentaje en cada proceso, esta información también, se guarda en la base de datos. En la Figura 102, se tiene el código para recibir como parámetro el número de porcentaje y, se crea una variable url para realizar las peticiones.

```
def actualizarPorcentaje(valor):
    #quote se codifica el dato a enviar para evitar problemas de caracteres especiales
    valor_encoded = quote(str(valor))
    #crear una variable que contenga la url a la que se realizara la peticion
    url = f"http://localhost/sistema/datos/actualizarPorcentaje.php?valor={valor_encoded}"
    # Realiza una solicitud a la URL para obtener respuesta.
    request = Request(url)
    #transformar la respuesta a formato JSON.
    json = urlopen(request).read().decode()
    #retornar resultado en formato JSON
    return json
```

Figura 102. Función actualizar porcentaje

En la función ventana Principal, de la Figura 103, es donde, se encuentra la captura de la imagen y todo lo referente al procesamiento realizado para las detecciones. Aquí, se define la ubicación para las imágenes obtenidas de imagen capturas, detección, contorno de césped y detección de contornos. La función inicia con la declaración de las variables globales que, se usara, se realiza el proceso principal para enviar la imagen al modelo pre entrenado para devolver el resultado y obtener solo la parte del césped.

```python
def ventanaPrincipal():
    global lblImgCam1
    global lblImgCam2
    global lblImgCam3
    global lblImgCam3
    global lblImgCam5
    global nueva_imgCam1
    global nueva_imgCam2
    global entryDistancia
    actualizarPorcentaje(15)
    #definimos en una variable el nombre de la imagen capturada desde la espcam
    image_path = "imagenServidor.jpeg"
    im1 = Image.open(image_path)
    #definimos el parametro de efectivilidad
    model.conf = 0.9
    detect = model(im1, size=400)
    nueva_imagen_Principal = im1.resize((350, 200))
    nueva_imgCam1 = ImageTk.PhotoImage(nueva_imagen_Principal)
    lblImgCam1.configure(image=nueva_imgCam1)
    print("leer imagen")
    #devuelve infomacion sobre las coordenadas que delimitan si en la imagen existe objetos detectados
    info = detect.pandas().xyxy[0]
    arr = np.asarray(info.confidence.array)
    detectar = len(arr)
    actualizarPorcentaje(18)
    if detectar > 0:
        #mensaje o indicador para validar si el procesamiento ha obtenido respuesta desde el modelo
        print("verificando")
        cv2.imwrite("resultado.png", cv2.cvtColor(np.squeeze(detect.render()), cv2.COLOR_BGR2RGB))
        #imagen con la deteccion obtenida y el valor detectado
        cv2.imwrite("C:/xampp/htdocs/sistema/resultado.png", cv2.cvtColor(np.squeeze(detect.render()), cv2.COLO
        nueva_imagen = Image.open('resultado.png').convert('RGB')
```

Figura 103. Función ventana Principal

Después, en la Figura 104, Figura 105 y Figura 106, se muestra cómo, se obtiene el área del jardín a regar. Inicia con la búsqueda del área de césped, la recorta y mejora en base a los formatos de color HSV y BGR para después generar los contornos de césped y objetos. Para garantizar que cada proceso sea correcto, se procede a crear una imagen en cada parte del proceso, también, se procede a enviar como parámetros los valores para las tonalidades umbral de verde bajo y alto.

Una vez que, se separa el césped y cualquier objeto encontrado, se procede a verificar, en caso de tener un objeto, sus puntos de posición. Estos, se usan como referencia para calcular los ángulos de los movimientos requeridos para los servomotores y así evitar que, se moje el objeto, como, se muestra en la Figura 107. Es importante obtener el valor total en ancho y alto de la imagen para generar la nueva escala.

```
ventanaPrincipal
nueva_imagen = Image.open('resultado.png').convert('RGB')
#obtener las cordenadas del objeto detectado
box_coords = info[['xmin', 'ymin', 'xmax', 'ymax']].to_numpy(dtype=int)[0]
roi = np.array(nueva_imagen)[box_coords[1]:box_coords[3], box_coords[0]:box_coords[2], :]
#crear una imagen en la carpeta local donde se mostrará solo la parte que ha detectado
cv2.imwrite("soloCesped.png", cv2.cvtColor(roi, cv2.COLOR_RGB2BGR))
#redimencionar la imagen original con la deteccion obtenida
nueva_imagen = nueva_imagen.resize((350, 200))
nueva_imgCam2 = ImageTk.PhotoImage(nueva_imagen)
lblImgCam2.configure(image=nueva_imgCam2)
imagenSoloCesped = cv2.imread('soloCesped.png')
#cambiar los colores de dicha imagen a HSV
imagen_solocesped_hsv = cv2.cvtColor(imagenSoloCesped, cv2.COLOR_BGR2HSV)
factor_saturacion = 5
#proceso para multiplicar el valor de saturacion de la imagen por el parametro de saturacion definido
imagen_solocesped_hsv[:, :, 1] = np.clip(imagen_solocesped_hsv[:, :, 1] * factor_saturacion, 0, 255).as
#convertir imagen de HSV a BGR
imagen_corregida = cv2.cvtColor(imagen_solocesped_hsv, cv2.COLOR_HSV2BGR)
cv2.imwrite('corregida.png', imagen_corregida)
imagenCorregida1 = cv2.imread('corregida.png')
#cambiar los colores de dicha imagen a HSV
imagen_hsv_corr = cv2.cvtColor(imagenCorregida1, cv2.COLOR_BGR2HSV)
#definir valores minimos y maximos para detectar césped
verde_bajo_imgCor = np.array([0, 0,50], np.uint8)
verde_alto_imgCor = np.array([120, 255, 255], np.uint8)
red_bajo_imgCor=np.array([150, 0, 0], np.uint8)
red_alto_imgCor=np.array([179, 255, 255], np.uint8)
#definir mascara que permite identificar dentro de la imagen pixeles que contienen los colores
mascara_verde_corr = cv2.inRange(imagen_hsv_corr, verde_bajo_imgCor, verde_alto_imgCor)
mask_red_corr = cv2.inRange(imagen_hsv_corr, red_bajo_imgCor, red_alto_imgCor)
#se une las dos máscaras definidas para integrarlas a la imagen
maskCompleta_corr = cv2.add(mascara_verde_corr, mask_red_corr)
```

Figura 104. Función ventana Principal con parámetros de detección

```
ventanaPrincipal
#se une las dos máscaras definidas para integrarlas a la imagen
maskCompleta_corr = cv2.add(mascara_verde_corr, mask_red_corr)
pixeles_verdes_completo = cv2.bitwise_and(imagenCorregida1, imagenCorregida1, mask=maskCompleta_corr)
#buscar y almacenar contornos encontrados dentro del rango anterior
contornos, _ = cv2.findContours(mascara_verde_corr, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#permite dibujar los contornos del resultado obtenido
cv2.drawContours(pixeles_verdes_completo, contornos, -1, (0, 255, 0), 2)
#aplicar la mascara obtenido a la imagen
mascara_completa = cv2.bitwise_or(mascara_verde_corr, mask_red_corr)
pixeles_verdes_completo = np.zeros_like(imagenCorregida1)
pixeles_verdes_completo[mascara_completa != 0] = imagenCorregida1[mascara_completa != 0]
#crear una imagen con el resultado de la imagen y los contornos
cv2.imwrite('verdesCompleto.png', pixeles_verdes_completo)
#Abrir o leer la imagen corregida la saturacion
imagen_VC = cv2.imread('verdesCompleto.png')
#covierte la imagen a escala de grises
imagen_gris_VC = cv2.cvtColor(imagen_VC, cv2.COLOR_BGR2GRAY)
#aplicar un umbral para crear una imagen binaria
imagen_umbral_VC = cv2.adaptiveThreshold(imagen_gris_VC, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BI
#encuentra los contornos de la imagen binaria, y así obtener solo los contornos externos
contornos_VC, _ = cv2.findContours(imagen_umbral_VC, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#definir un umbral para detectar el area de los contornos minimos a mostrar
area_minima_VC = 9000
#filtrar los contornos con los parametros anteriores
contornos_filtradosVC = [cnt for cnt in contornos_VC if cv2.contourArea(cnt) > area_minima_VC]
#dibujar contornos dentro de la imagen
cv2.drawContours(imagen_VC, contornos_filtradosVC, -1, (0, 255, 0), 3)
#crear imagen en la carpeta local no los contornos obtenidos
cv2.imwrite('contornoVerdeCompleto.png', imagen_VC)
#Abrir o leer la imagen corregida la saturacion
cesped_imagen = Image.open('contornoVerdeCompleto.png')
#redimencionar la imagen original con la deteccion obtenida
```

Figura 105. Función ventana Principal con contornos dibujados

ventanaPrincipal

```
#redimencionar la imagen original con la deteccion obtenida
nueva_imagen2 = cesped_imagen.resize((300, 200))
#permite convertir la imagen al formato adecuado para usarla dentro de python y tkinter
cesped_imgCam1 = ImageTk.PhotoImage(nueva_imagen2)
#se muestra la imagen resultante de los contornos dentro del tercer label
#para esta caso despues de la actualización del diseño de la interfaz, el label 4 quedó como
#tercera imagen
lblImgCam4.configure(image=cesped_imgCam1)
lblImgCam4.image = cesped_imgCam1
#proceso para detectar posicion de objetos
#crear imagen en la carpeta local no los contornos obtenidos
imagenObjeto = cv2.imread('verdesCompleto.png')
#convertir l imagen a escala de grices
imagen_grisObjeto = cv2.cvtColor(imagenObjeto, cv2.COLOR_BGR2GRAY)
imagen_umbralObjeto = cv2.adaptiveThreshold(imagen_grisObjeto, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THR
#encuentra los contornos de la imagen binaria
contornosObjeto, _ = cv2.findContours(imagen_umbralObjeto, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#definir un umbral para detectar el area de los contornos minimos a mostrar
#se da un parametro para verificar los pixeles donde se buscara el objeto
area_minimaObjeto = 3000
altura_minimaObjeto = 130
altura_maximaObjeto = 260
contornos_filtradosObjeto = [cnt for cnt in contornosObjeto if cv2.contourArea(cnt) > area_minimaObjeto
#crear una copia de la imagen abierta anteriormente para ubicar los objetos y sus contornos
imagen_contornosObjeto = imagenObjeto.copy()
#dibujar contornos
cv2.drawContours(imagen_contornosObjeto, contornos_filtradosObjeto, -1, (0, 0, 255), 3)
#crear imagen en la carpeta local mostrando el objeto detectado y sus contornos
cv2.imwrite('contornoObjeto.png', imagen_contornosObjeto)
#crear imagen en la carpeta del servidor mostrando el objeto detectado y sus contornos
cv2.imwrite("C:/xampp/htdocs/sistema/contorno.png", imagen_contornosObjeto)
#abrir imagen que contiene contornos de objetos
```

Figura 106. Función ventana Principal con imágenes de contornos almacenados

```
try:
    #verificar si exiten contornos encontrados
    if contornos_filtradosObjeto:
        #obtiene las dimensiones de la imagen origina
        alto, ancho, _ = imagenObjeto.shape
        nuevo_ancho = 180
        nuevo_alto = 80
        #calcular nueva escala
        factor_escala_ancho = nuevo_ancho / ancho
        factor_escala_alto = nuevo_alto / alto
        #encontrar puntos del objeto encontrado en el contorno filtrado
        contorno_transformado = contornos_filtradosObjeto[0] * [factor_escala_ancho, factor_escala_alto]
        punto_mas_bajo_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 1].argmax()][0
        punto_mas_izquierda_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 0].argmin
        punto_mas_derecha_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 0].argmax()
        #almacenar puntos necesario
        x_bajo, y_bajo = punto_mas_bajo_nuevo
        x_izquierda, y_izquierda = punto_mas_izquierda_nuevo
        x_derecha, y_derecha = punto_mas_derecha_nuevo
    else:
        #si no hay contornos los valores se enviarán en cero
        y_bajo=0
        x_izquierda=0
        x_derecha=0
except Exception as e:
    #si no hay contornos los valores se enviarán en cero
    y_bajo=0
    x_izquierda=0
    x_derecha=0
```

Figura 107. Función ventana Principal con puntos del objeto detectado

A continuación, se describen las funciones requeridas para compartir la información con la base de datos, de igual manera para realizar peticiones de consulta y actualización de datos.

La función traer proceso de la Figura 108, permite obtener la información sobre el proceso actual del sistema y realiza una consulta a la información que, se encuentra en el archivo proceso.php

```
def traerProceso():
    request = Request("http://localhost/sistema/datos/proceso.php")
    json = urlopen(request).read().decode()
    return json
```

Figura 108. Función traer proceso

La función actualizar clima, como muestra la Figura 109, actualiza los datos del clima almacenados en la base de datos. Crear una variable que contenga la url a la que, se realizara la petición y realiza una solicitud.

```
def actualizarClima(tiempo,temperatura,humedad,lluvia,viento):
    #quote se codifica los datos a enviar para evitar problemas de caracteres especiales
    tiempo_encoded = quote(str(tiempo))
    temperatura_encoded = quote(str(temperatura))
    humedad_encoded = quote(str(humedad))
    lluvia_encoded = quote(str(lluvia))
    viento_encoded = quote(str(viento))
    #crear una variable que contenga la url a la que se realizara la peticion
    url = f"http://localhost/sistema/datos/actualizarClima.php?tiempo={tiempo_encoded}&temperatura={temperatura
    # Realiza una solicitud a la URL para obtener respuesta.
    request = Request(url)
    #transformar la respuesta a formato JSON.
    json = urlopen(request).read().decode()
    #retornar resultado en formato JSON
    return json
```

Figura 109. Función actualiza clima

La función actualizar parámetros, actualizar los datos de los parámetros para los servomotores y, se guardan en la base de datos. En la Figura 110, la función recibe como parámetro de entrada los datos para sincronizar los servomotores. Se crea una variable que contenga la url a la que, se realizará la petición.

```python
def actualizarParametros(anguloMaximo,anguloMinimoH,anguloMaximoH,valorMinX,valorMaxX,valorMaxY):
    #quote se codifica los datos a enviar para evitar problemas de caracteres especiales
    anguloMaximo_encoded = quote(str(anguloMaximo))
    anguloMinimoH_encoded = quote(str(anguloMinimoH))
    anguloMaximoH_encoded = quote(str(anguloMaximoH))
    valorMinX_encoded = quote(str(valorMinX))
    valorMaxX_encoded = quote(str(valorMaxX))
    valorMaxY_encoded = quote(str(valorMaxY))
    #crear una variable que contenga la url a la que se realizara la peticion
    url = f"http://localhost/sistema/datos/actualizarParametros.php?anguloMaximo={anguloMaximo_encoded}&anguloM
    # Realiza una solicitud a la URL para obtener respuesta.
    request = Request(url)
    #transformar la respuesta a formato JSON.
    json = urlopen(request).read().decode()
    #retornar resultado en formato JSON
    return json
```

Figura 110. Función actualizar parámetros

Además, se creó la función cambiar automático, permite actualizar y guardar en la base de datos los valores de porcentaje en cada proceso. Se utiliza una variable interna que almacena el estado actual del modo, ya sea automático o manual. Tal como muestra la Figura 111, se inicializa la variable en cero, cambiara solo si el valor actual de cero, se convierte en uno, caso contrario si es uno lo mantiene en cero. Para así cambiar de mono manual a automático y viceversa. Además, cuenta con una condición que valida si el estado actual de la variable automático es 0 para enviar la solicitud de petición a la base de datos.

```python
def cambiarAutomatico():
    global valorAutomatico
    #0: Manual
    #1: Automático
    valEnviar=0
    #condicion que valida si el estado actual del automatico es 0, o manual activado para cambiarlo
    if valorAutomatico == 0:
        valEnviar=1
    # Realiza una solicitud a la URL para obtener respuesta.
    request = Request("http://localhost/sistema/datos/actualizarAutomatico.php?valor="+str(valEnviar)+"")
    #transformar la respuesta a formato JSON.
    json = urlopen(request).read().decode()
    #retornar resultado en formato JSON
    return json
```

Figura 111. Función cambiar automático

En la Figura 112, la función actualizar estado, permite actualizar el estado actual del sistema. Se definió que con el valor cero el estado es detenido y con el valor 1 el estado activado o en proceso. De igual manera realiza la petición a la base de datos

```python
def actualizarEstado():
    #global: permite llamar y hacer uso de una variable global o variaable que se usa fuera esta funcion
    global valorEstado
    # si es uno lo mantienen en cero, así cambiando de en proceso a detenido y viceversa.
    valEnviar=0
    if valorEstado == 0:
        valEnviar=1
    # Realiza una solicitud a la URL para obtener respuesta.
    request = Request("http://localhost/sistema/datos/actualizarEstado.php?valor="+str(valEnviar)+"")
    #transformar la respuesta a formato JSON.
    json = urlopen(request).read().decode()
    #actualiza el valor del porcentaje subiendo en uno desde el archivo php
    actualizarPorcentaje(0)
    #retornar resultado en formato JSON
    return json
```

Figura 112. Función actualizar estado

La función cambiar estado, de la Figura 113, permite actualizar o aumentar el proceso de estado actual del sistema. El estado actual del sistema, se definió con los valores de: 0 para detenido, 1 para lectura de imagen, 2 para procesamiento de imagen, y 3 para el riego.

```python
def cambiarEstado():
    # Realiza una solicitud a la URL para obtener respuesta.
    request = Request("http://localhost/sistema/datos/cambiarEstado.php")
    #transformar la respuesta a formato JSON.
    json = urlopen(request).read().decode()
    #actualiza el valor del porcentaje subiendo en uno desde el archivo php
    actualizarPorcentaje(0)
    #retornar resultado en formato JSON
    return json
```

Figura 113. Función cambiar estado

La función capturar imagen de la Figura 114, permite obtener la imagen desde la esp32-cam usándola como cámara ip, y almacenando esa imagen en la computadora local, así mismo si obtiene la imagen correctamente, se procede a cambiar el valor del proceso, el cual indica al sistema que el proceso de captura está terminado y debe proceder con el siguiente proceso que es el tratamiento de dicha imagen con el modelo entrenado previamente.

```python
def capturarImagen():
    global valorEstado
    try:
        actualizarPorcentaje(5)
        tiempo_de_espera = 5
        #se trabaja con la espcam en modo por defecto, obteniendo la url y capturando una imagen desde la misma
        response = requests.get(f"http://{ipespcam}/capture", timeout=tiempo_de_espera)
        #condición para validar sí la la petición fue correta
        if response.status_code == 200:
            #se obtiene la imagen capturada y se la convierte en bytes
            image_data = BytesIO(response.content)
            #se abre la imagen desde la variable donde se la almacento en tipo byte
            image = Image.open(image_data)
            #guardar dicha imagen en la carpeta local
            image.save('imagenServidor.jpeg', "JPEG")
            #close: libera memoria y la imagen
            image.close()
            #se llama a la funcion cambiarEstado para indicar que el proceso a finalizado y puede continuar con
            #el siguiente proceso
            cambiarEstado()
            #se llama a la funcion para actualizar el porcentaje a 10
            actualizarPorcentaje(10)
        else:
            print(f"Error al capturar la imagen. Código de estado: {response.status_code}")
            valorEstado=0
            actualizarEstado()
            actualizarPorcentaje(0)
    except Exception as e:
        #si existe algun error con la conexión, proceso reiniciar al estado a cero.
        print(f"Error: {e}")
        valorEstado=0
        actualizarEstado()
```

Figura 114. Función capturar imagen

La función abrir caratula permite abrir la ventana inicial, la ventana de inicio de sesión. Su código, se muestra en la Figura 115.

```python
def abrirCaratula():
    #maximizar ventana inicio de sesión
    ventana.state(newstate="zoomed")
    #minimizar ventana principal o de proceso
    datos.state(newstate="withdraw")
    #minimizar ventana de registro
    registrar.state(newstate="withdraw")
```

Figura 115. Función abrir caratula

Función obtener código html, permite obtener la imagen realizando una solicitud a la url de la espcam. En la Figura 116, se trabaja con la espcam en modo por defecto, obteniendo la url y capturando una imagen desde la misma.

```python
def obtener_codigo_html(url):
    try:
        response = requests.get(url)
        #condición para validar si la la petición fue correta
        if response.status_code == 200:
            #retornar el codigo como texto
            return response.text
        else:
            print(f"Error al obtener el código HTML. Código de estado: {response.status_code}")
            return None
    #except finaliza el control de error, capturando un erro de tipo conexión y respuesta y envía un mensaje de
    except requests.exceptions.RequestException as e:
    #mensaje que indica cual fue el error.
        print(f"Error al hacer la solicitud HTTP: {e}")
        return None
```

Figura 116. Función obtener código html

3.7.2 Aplicación móvil

La aplicación móvil, se desarrolló con Android Studio, se inició con los archivos principales donde, se encuentran todos los procesos principales de la aplicación, como muestra la Figura 117.

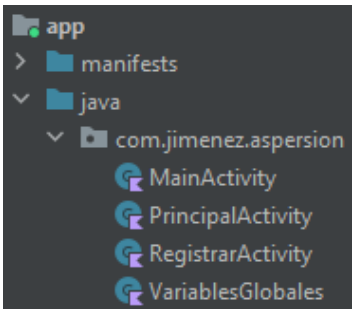


Figura 117. Archivos principales de Android Studio

En la Figura 118, se encuentran todos los Layout principales del sistema. Los layout definen la interfaz de usuario de una aplicación, aquí es donde, se almacena la ubicación, formas, colores y estilos para la aplicación.

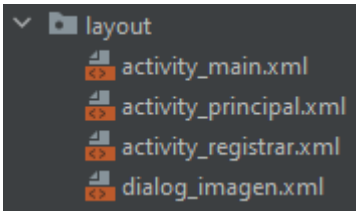


Figura 118. Archivos Layout principales

La ventana principal de la Figura 119, establece el inicio de sesión con el nombre de usuario y contraseña, también, cuenta con los botones de ingreso y registro. Esta ventana, se desarrolla dentro del archivo “MainActivity” y el layout “activiy main”.

Figura 119. Ventana de Inicio de Sesión

En el archivo Main Activity de la Figura 120, se encuentran los procesos que, se ejecutará para el funcionamiento del ingreso de usuario y registro de nuevos usuarios. Inicialmente, se declara las variables de texto para nombre de usuario y contraseña, y los botones ingresar y registrar.

```
package com.jimenez.aspersion

import ...

class MainActivity : AppCompatActivity() {

    //Declarar variable de tipo textview donde se mostrara el titulo del input para usuario en inicio de sesion
    private lateinit var tituloUsuario: TextView
    //Declarar variable de tipo textview donde se mostrara el titulo del input para contraseña en inicio de sesio
    private lateinit var tituloContrasena: TextView
    //Declarar variable de tipo textview donde se mostrara el titulo del input para error en inicio de sesion
    private lateinit var textError: TextView
    //Declarar variable de tipo EdiText donde el usuario ingresar el dato de usuario en inicio de sesion
    private lateinit var editUsuario: EditText
    //Declarar variable de tipo EdiText donde el usuario ingresar el dato de contraseña en inicio de sesion
    private lateinit var editContrasena: EditText
    //Declarar variable de tipo Button que permitirá ingresar al sistema
    private lateinit var botonIngresar: Button
    //Declarar variable de tipo Button que permitirá ingresar al registro de usuario
    private lateinit var botonRegistro: Button
    //Declarar variable de tipo ProgressDialog que permite mostrar la imagen de cargando o en proceso
    private lateinit var progressDialog: ProgressDialog


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

Figura 120. Archivo Main Activity con variables declaradas

En la Figura 121, se inicializa las variables y, se crea el objeto para el mensaje de ingreso cuando los datos son correctos.

```
override fun onCreate(savedInstanceState: Bundle?) {                    ⚠ 27
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    //inciar el edit usuario con el nombre definido en el layout
    editUsuario = findViewById<EditText>(R.id.edit_usuario_main)
    //inciar el edit contrasena con el nombre definido en el layout
    editContrasena = findViewById<EditText>(R.id.edit_contrasena_main)
    //inciar el boton ingresar con el nombre definido en el layout
    botonIngresar = findViewById<Button>(R.id.boton_ingresar_main)
    //inciar el boton registro con el nombre definido en el layout
    botonRegistro = findViewById<Button>(R.id.boton_registrar_main)
    //inciar el textview titulo usuario con el nombre definido en el layout
    tituloUsuario = findViewById<TextView>(R.id.titulo_usuario_main)
    //inciar el textview titulo contrasena con el nombre definido en el layout
    tituloContrasena = findViewById<TextView>(R.id.titulo_contrasena_main)
    //inciar el textview error con el nombre definido en el layout
    textError = findViewById<TextView>(R.id.error_ingreso_main)
    //crear objeto de tipo ProgressDialog
    progressDialog = ProgressDialog( context: this)
    //Asignar mensaje a mostrar
    progressDialog.setMessage("Ingresando...")
    //Configurar para que no se cierre cuando el usuario precione fuera del cuadro del mensaje
    progressDialog.setCancelable(false)
```

Figura 121. Archivo Main Activity con inicialización de variables

El botón de ingreso de la Figura 122 almacena los datos ingresados en los apartados de usuario y contraseña. Una vez almacenados llama a la función iniciar sesión.

```
botonIngresar.setOnClickListener(View.OnClickListener { it: View!
    //almacenar en una variable el dato de usuario ingresado por el usuario en el edittext
    val usuario = editUsuario.text.toString()
    //almacenar en una variable el dato de contrasena ingresado por el usuario en el edittext
    val password = editContrasena.text.toString()
    //llamar a la funcion iniciarsesion la que recibe como parametro usuario y password(contrasena)
    //esta permite validar los datos ingresados y permitir el paso a la ventana principal
    iniciarSesion(usuario, password)
})
```

Figura 122. Botón Ingresar

La función iniciar sesión, se encarga de verificar los datos ingresados en usuario y contraseñas, en la Figura 123, se genera una condición para que cuando el dato sea menor que cero, se genere un mensaje de error de que no hay datos ingresados.

```
private fun iniciarSesion(usuario: String, contrasena: String) {
    //declara e inicializa una variable de tipo text que hace referencia al text
    //dentro de layout
    val errorIngreso = findViewById<TextView>(R.id.error_ingreso_main)
    //verificar si el dato ingresado en usuario quitando los espacios en blanco
    //es cero o menor que cero
    if(usuario.trim().count()<=0){
        //se visualiza el texto de error
        errorIngreso.visibility = View.VISIBLE
        //muestra un mensaje de error indicando que los datos son obligatorios
        errorIngreso.text = "Todos los datos son obligatorios"
    } else {
        //se oculta el texto de error
        errorIngreso.visibility = View.GONE
        //verificar si el dato ingresado en contrasena quitando los espacios en blanco
        //es cero o menor que cero
        if(contrasena.trim().count()<=0){
            //se visualiza el texto de error
            errorIngreso.visibility = View.VISIBLE
            //muestra un mensaje de error indicando que los datos son obligatorios
            errorIngreso.text = "Todos los datos son obligatorios"
        } else {
            //se oculta el texto de error
            errorIngreso.visibility = View.GONE
        }
    }
```

Figura 123. Función iniciar sesión con datos obligatorios

La siguiente condición de la Figura 124 verifica que los datos ingresados no sean nulos o estén en blanco. Además, realiza una petición a la base de datos para verificar los valores de usuario y contraseña.

```
if(usuario.trim().count()>0 && contrasena.trim().count()>0) {
    //se oculta el mensaje de error
    errorIngreso.visibility=View.GONE
    //se inicializa un proceso en segundo plano o en hilo diferente al principal
    //para evitar que el sistema se bloquee
    val task = object : AsyncTask<String, Void, String>() {
        override fun doInBackground(vararg params: String): String? {
            //obtener los valores de los datos ingresado por el usuario
            val ususario = params[0]
            val contrasena = params[1]
            try {
                //declarar un objeto de tipo JSON
                val jsonObject = JSONObject()
                //ingresar parametros en formato JSON
                //enviamos los valores de usuario y contrasena
                jsonObject.put( name: "usuario", ususario)
                jsonObject.put( name: "contrasena", contrasena)
                //declarar e inciar la variable de tipo URL con la dirección url a enviar los datos
                val url =
                    URL( spec: VariablesGlobales.ip+"datos/iniciarsesion.php")
                //abrir la conexión http
                val httpURLConnection = url.openConnection() as HttpURLConnection
                //establecer el tipo de conexión en este caso POST
                httpURLConnection.requestMethod = "POST"
                //configurar el tipo de respuesta en JSON
                httpURLConnection.setRequestProperty("Content-Type", "application/json")
                //habilitar la salida de datos para la conexion http
```

Figura 124. Función iniciar sesión con acciones de ingreso

Las siguientes funciones de la Figura 125 son para realizar actualizaciones en la interfaz de usuario después de completar la operación y para realizar el proceso de consulta a la base de datos.

```
override fun onPostExecute(result: String?) {
    // Actualizar la interfaz de usuario después de completar la operación en segundo plano
    progressDialog.dismiss()
}

override fun onPreExecute() {
    super.onPreExecute()
    //activar el progress antes de realizar el proceso de consulta a la base de datos
    progressDialog.show()
```

Figura 125. Funciones de actualización de interfaz y proceso de consulta

En la Figura 126, el botón de registro despliega una nueva ventana donde, se guardarán los datos de los nuevos usuarios

```
//evento cuando se da click o presiona sobre el boton registro
botonRegistro.setOnClickListener { it: View!
    //declarar e incializar la ventana a abrir
    val intent = Intent(
        packageContext: this@MainActivity,
        RegistrarActivity::class.java
    )
    //abrir ventana
    startActivity(intent)
}
```

Figura 126. Botón Registrar

En la ventana de registro de la Figura 127, se ingresan los datos para los nuevos usuarios, también, cuenta con los botones de registrar y cancelar. Esta ventana, se desarrolla dentro del archivo “RegistrarActivity” y el layout “activiy registrar”.

Figura 127. Ventana de registro

En el archivo Registrar Activity, de la Figura 128 y Figura 129, se encuentran los procesos que, se ejecutará para el funcionamiento del registro de usuarios. Inicialmente, se declara las variables de texto para nombre, nombre de usuario, email, y contraseña, y los botones registrar y cancelar.

```
package com.jimenez.aspersion

import ...

class RegistrarActivity : AppCompatActivity() {
    //Declarar variable de tipo textview donde se mostrara el titulo del input para nombre en registrar
    private lateinit var tituloNombre: TextView
    //Declarar variable de tipo textview donde se mostrara el titulo del input para usuario en registrar
    private lateinit var tituloUsuario: TextView
    //Declarar variable de tipo textview donde se mostrara el titulo del input para email en registrar
    private lateinit var tituloEmail: TextView
    //Declarar variable de tipo textview donde se mostrara el titulo del input para contraseña en registrar
    private lateinit var tituloContrasena: TextView
    //Declarar variable de tipo textview donde se mostrara el titulo del input para error en registrar
    private lateinit var textError: TextView
    //Declarar variable de tipo EdiText donde el usuario ingresar el dato de nombre en registrar
    private lateinit var editNombre: EditText
    //Declarar variable de tipo EdiText donde el usuario ingresar el dato de usuario en registrar
    private lateinit var editUsuario: EditText
    //Declarar variable de tipo EdiText donde el usuario ingresar el dato de email en registrar
    private lateinit var editEmail: EditText
    //Declarar variable de tipo EdiText donde el usuario ingresar el dato de contraseña en registrar
    private lateinit var editContrasena: EditText
    //Declarar variable de tipo Button que permitirá cancelar el registro
    private lateinit var botonCancelar: Button
    //Declarar variable de tipo Button que permitirá guardar el registro
    private lateinit var botonRegistro: Button
```

Figura 128. Archivo Registrar Activity con variables declaradas

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_registrar)
    //iniciar el edit nombre con el nombre definido en el layout
    editNombre = findViewById<EditText>(R.id.edit_nombre_reg)
    //iniciar el edit usuario con el nombre definido en el layout
    editUsuario = findViewById<EditText>(R.id.edit_usuario_reg)
    //iniciar el edit email con el nombre definido en el layout
    editEmail = findViewById<EditText>(R.id.edit_email_reg)
    //iniciar el edit contrasena con el nombre definido en el layout
    editContrasena = findViewById<EditText>(R.id.edit_contrasena_reg)
    //inciar el boton registro con el nombre definido en el layout
    botonRegistro = findViewById<Button>(R.id.boton_registrar_reg)
    //inciar el boton cancelar con el nombre definido en el layout
    botonCancelar = findViewById<Button>(R.id.boton_cancelar_reg)
    //inciar el textview titulo nombre con el nombre definido en el layout
    tituloNombre = findViewById<TextView>(R.id.titulo_nombre_reg)
    //inciar el textview titulo usuario con el nombre definido en el layout
    tituloUsuario = findViewById<TextView>(R.id.titulo_usuario_reg)
    //inciar el textview titulo email con el nombre definido en el layout
    tituloEmail = findViewById<TextView>(R.id.titulo_email_reg)
    //inciar el textview titulo contrasena con el nombre definido en el layout
    tituloContrasena = findViewById<TextView>(R.id.titulo_contrasena_reg)
    //inciar el textview titulo error con el nombre definido en el layout
    textError = findViewById<TextView>(R.id.error_registro_reg)
    //crear objeto de tipo ProgressDialog
    progressDialog = ProgressDialog( context: this)
```

Figura 129. Archivo Registrar Activity con inicialización de variables

En la Figura 130, el botón de registro almacena los datos ingresados en los apartados definidos anteriormente. Una vez almacenados llama a la función iniciar registrar.

```
botonRegistro.setOnClickListener(View.OnClickListener { it: View!
    //almacenar en una variable el dato de usuario ingresado por el usuario en el edittext
    val usuario = editUsuario.text.toString()
    //almacenar en una variable el dato de nombre ingresado por el usuario en el edittext
    val nombre = editNombre.text.toString()
    //almacenar en una variable el dato de email ingresado por el usuario en el edittext
    val email = editEmail.text.toString()
    //almacenar en una variable el dato de contrasena ingresado por el usuario en el edittext
    val password = editContrasena.text.toString()
    //llmar a la funccion registrar enviando como parametros los valores anteriores
    registrar(nombre,usuario,email,password)
})
```

Figura 130. Botón registrar

El botón de cancelar de la Figura 131, despliega la ventana de inicio de sesión donde, se guardarán los datos del usuario.

```
botonCancelar.setOnClickListener { it: View!
    //se abre la ventana de iniciar sesion
    val intent = Intent(
        packageContext: this@RegistrarActivity,
        MainActivity::class.java
    )
    startActivity(intent)
}
```

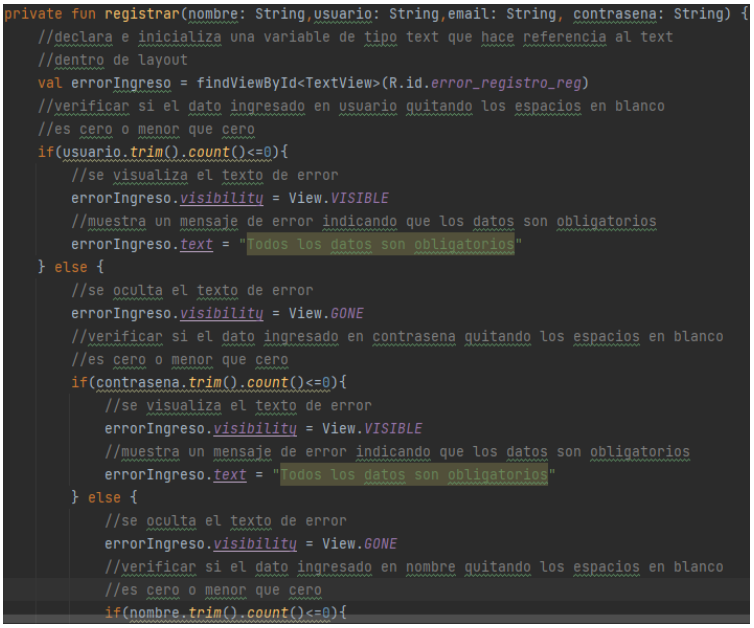
Figura 131. Botón cancelar

La siguiente configuración de la Figura 132, se utiliza para los textos ingresados en todas las variables de la venta registro. Permite la verificación cuando, no se ha ingresado ningún dato en los casilleros.

```
editNombre.addTextChangedListener(object : TextWatcher {
    override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {
        // No se necesita implementar
    }
    override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
        // Verificar si el texto está vacío
        if (s.isNullOrEmpty()) {
            //si no hay texto en el edit el titulo se oculta
            tituloNombre.visibility = View.GONE
        } else {
            //si existe texto en el edit el titulo es visible
            tituloNombre.visibility = View.VISIBLE
        }
    }
    override fun afterTextChanged(s: Editable?) {
        // No se necesita implementar
    }
})
```

Figura 132. Control de variables del registro

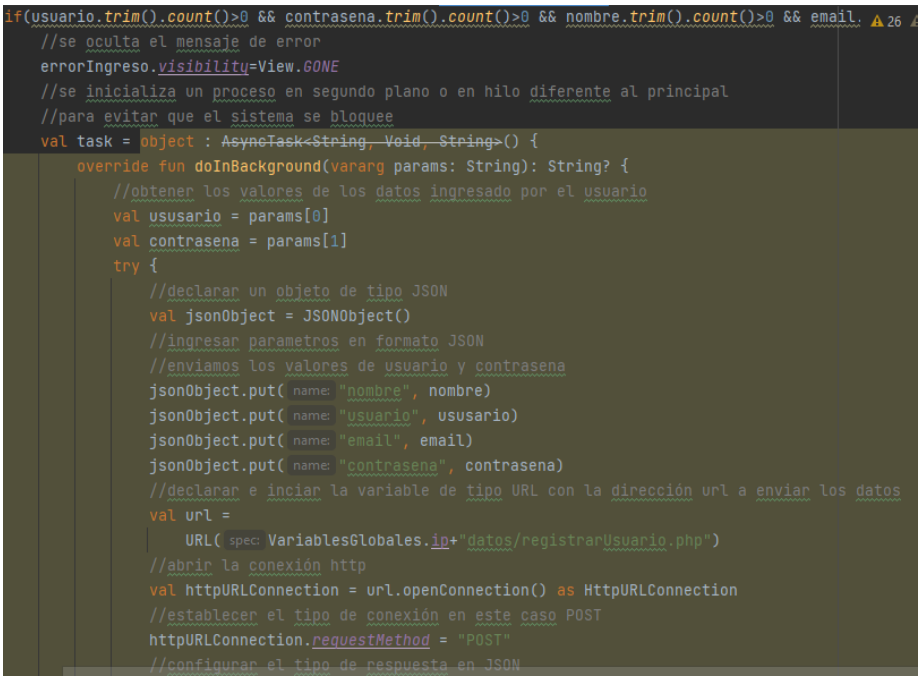
En la Figura 133, la función registro inicia verificando todos los datos ingresados, se genera una condición para que cuando el dato sea menor que cero, se genere un mensaje de error de que no hay datos ingresados y son obligatorios.



```
private fun registrar(nombre: String,usuario: String,email: String, contrasena: String) {
    //declara e inicializa una variable de tipo text que hace referencia al text
    //dentro de layout
    val errorIngreso = findViewById<TextView>(R.id.error_registro_reg)
    //verificar si el dato ingresado en usuario quitando los espacios en blanco
    //es cero o menor que cero
    if(usuario.trim().count()<=0){
        //se visualiza el texto de error
        errorIngreso.visibility = View.VISIBLE
        //muestra un mensaje de error indicando que los datos son obligatorios
        errorIngreso.text = "Todos los datos son obligatorios"
    } else {
        //se oculta el texto de error
        errorIngreso.visibility = View.GONE
        //verificar si el dato ingresado en contrasena quitando los espacios en blanco
        //es cero o menor que cero
        if(contrasena.trim().count()<=0){
            //se visualiza el texto de error
            errorIngreso.visibility = View.VISIBLE
            //muestra un mensaje de error indicando que los datos son obligatorios
            errorIngreso.text = "Todos los datos son obligatorios"
        } else {
            //se oculta el texto de error
            errorIngreso.visibility = View.GONE
            //verificar si el dato ingresado en nombre quitando los espacios en blanco
            //es cero o menor que cero
            if(nombre.trim().count()<=0){
```

Figura 133. Función registro con datos obligatorios

La siguiente condición de la Figura 134 realiza una petición a la base de datos para verificar los valores ingresados.



```
if(usuario.trim().count()>0 && contrasena.trim().count()>0 && nombre.trim().count()>0 && email. ⚠ 26
    //se oculta el mensaje de error
    errorIngreso.visibility=View.GONE
    //se inicializa un proceso en segundo plano o en hilo diferente al principal
    //para evitar que el sistema se bloquee
    val task = object : AsyncTask<String, Void, String>() {
        override fun doInBackground(vararg params: String): String? {
            //obtener los valores de los datos ingresado por el usuario
            val ususario = params[0]
            val contrasena = params[1]
            try {
                //declarar un objeto de tipo JSON
                val jsonObject = JSONObject()
                //ingresar parametros en formato JSON
                //enviamos los valores de usuario y contrasena
                jsonObject.put( name: "nombre", nombre)
                jsonObject.put( name: "usuario", ususario)
                jsonObject.put( name: "email", email)
                jsonObject.put( name: "contrasena", contrasena)
                //declarar e inciar la variable de tipo URL con la dirección url a enviar los datos
                val url =
                    URL( spec: VariablesGlobales.ip+"datos/registrarUsuario.php")
                //abrir la conexión http
                val httpURLConnection = url.openConnection() as HttpURLConnection
                //establecer el tipo de conexión en este caso POST
                httpURLConnection.requestMethod = "POST"
                //configurar el tipo de respuesta en JSON
```

Figura 134. Función registro con ingreso a base de datos

En el archivo de variables globales, se encuentra todas las variables que, se requieren en los demás archivos para una buena interacción de los procesos. En la Figura 135, se presenta el código empleado.

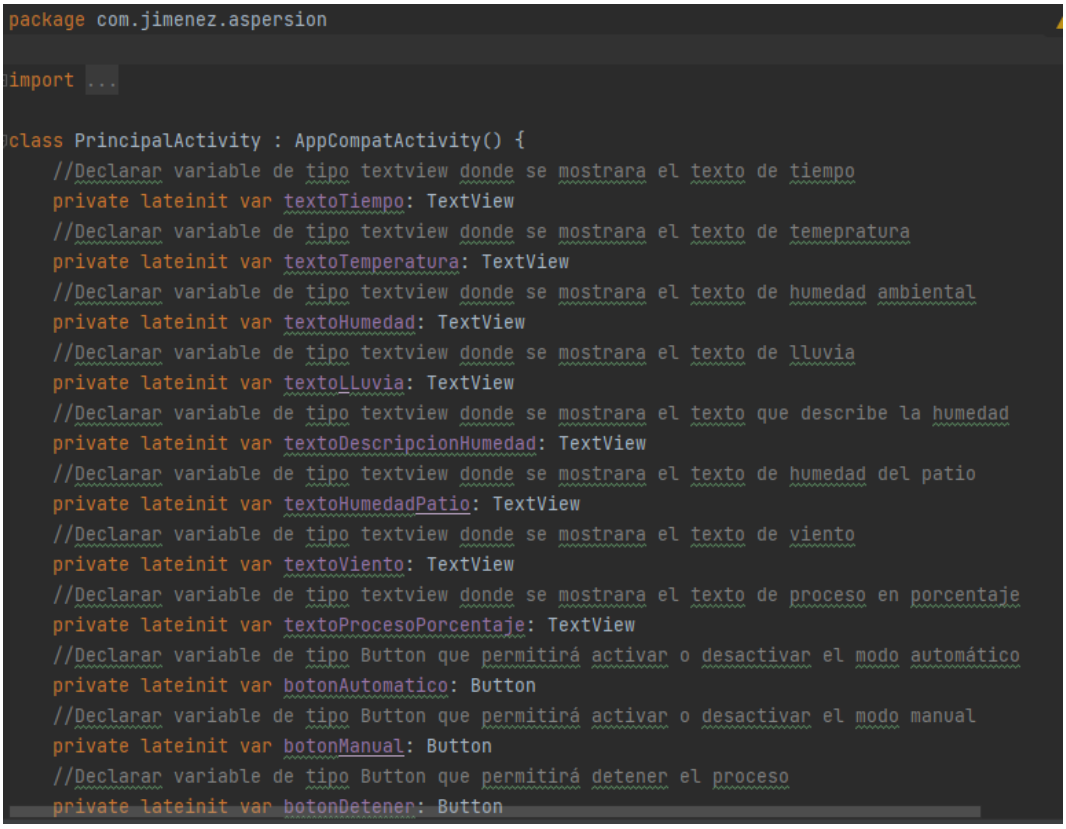
```
package com.jimenez.aspersion

import androidx.appcompat.app.AppCompatActivity

class VariablesGlobales  : AppCompatActivity() {
    companion object {
        //declarar variables globales que se usara en el sistema
        var idUsuario="0"
        var proceso=0
        var automatico=0
        var porcentaje=""
        var ip="http://192.168.100.28/sistema/"
    }
}
```

Figura 135. Archivo variables globales

Una vez ingresados correctamente los datos del usuario, se despliega la ventana principal donde la aplicación móvil cuenta con los datos requeridos para conocimiento del usuario, y para realizar el proceso de detección y riego. A continuación, se muestra la Figura 136 con la ventana principal donde, se resaltan las áreas etiquetadas establecidas en el apartado de desarrollo de interfaz.

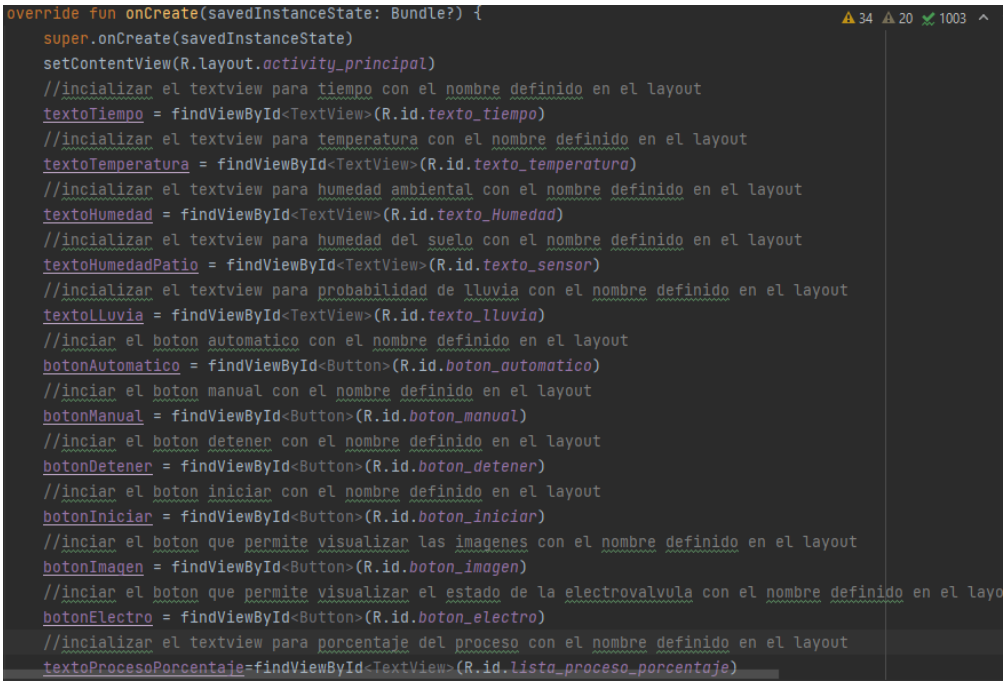
Figura 136. Ventana principal de la aplicación móvil

En el archivo Principal Activity de la Figura 137 y Figura 138, se encuentran los procesos que, se ejecutará para el funcionamiento de la ventana principal. Inicialmente, se declara las variables de texto para los botones de control, indicador de humedad de suelo e indicadores de clima.



```
package com.jimenez.aspersion

import ...

class PrincipalActivity : AppCompatActivity() {
    //Declarar variable de tipo textview donde se mostrara el texto de tiempo
    private lateinit var textoTiempo: TextView
    //Declarar variable de tipo textview donde se mostrara el texto de temepratura
    private lateinit var textoTemperatura: TextView
    //Declarar variable de tipo textview donde se mostrara el texto de humedad ambiental
    private lateinit var textoHumedad: TextView
    //Declarar variable de tipo textview donde se mostrara el texto de lluvia
    private lateinit var textoLLuvia: TextView
    //Declarar variable de tipo textview donde se mostrara el texto que describe la humedad
    private lateinit var textoDescripcionHumedad: TextView
    //Declarar variable de tipo textview donde se mostrara el texto de humedad del patio
    private lateinit var textoHumedadPatio: TextView
    //Declarar variable de tipo textview donde se mostrara el texto de viento
    private lateinit var textoViento: TextView
    //Declarar variable de tipo textview donde se mostrara el texto de proceso en porcentaje
    private lateinit var textoProcesoPorcentaje: TextView
    //Declarar variable de tipo Button que permitirá activar o desactivar el modo automático
    private lateinit var botonAutomatico: Button
    //Declarar variable de tipo Button que permitirá activar o desactivar el modo manual
    private lateinit var botonManual: Button
    //Declarar variable de tipo Button que permitirá detener el proceso
    private lateinit var botonDetener: Button
```

Figura 137. Archivo Principal Activity con declaración de variables



```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_principal)
    //incializar el textview para tiempo con el nombre definido en el layout
    textoTiempo = findViewById<TextView>(R.id.texto_tiempo)
    //incializar el textview para temperatura con el nombre definido en el layout
    textoTemperatura = findViewById<TextView>(R.id.texto_temperatura)
    //incializar el textview para humedad ambiental con el nombre definido en el layout
    textoHumedad = findViewById<TextView>(R.id.texto_Humedad)
    //incializar el textview para humedad del suelo con el nombre definido en el layout
    textoHumedadPatio = findViewById<TextView>(R.id.texto_sensor)
    //incializar el textview para probabilidad de lluvia con el nombre definido en el layout
    textoLLuvia = findViewById<TextView>(R.id.texto_lluvia)
    //inciar el boton automatico con el nombre definido en el layout
    botonAutomatico = findViewById<Button>(R.id.boton_automatico)
    //inciar el boton manual con el nombre definido en el layout
    botonManual = findViewById<Button>(R.id.boton_manual)
    //inciar el boton detener con el nombre definido en el layout
    botonDetener = findViewById<Button>(R.id.boton_detener)
    //inciar el boton iniciar con el nombre definido en el layout
    botonIniciar = findViewById<Button>(R.id.boton_iniciar)
    //inciar el boton que permite visualizar las imagenes con el nombre definido en el layout
    botonImagen = findViewById<Button>(R.id.boton_imagen)
    //inciar el boton que permite visualizar el estado de la electrovalvula con el nombre definido en el layo
    botonElectro = findViewById<Button>(R.id.boton_electro)
    //incializar el textview para porcentaje del proceso con el nombre definido en el layout
    textoProcesoPorcentaje=findViewById<TextView>(R.id.lista_proceso_porcentaje)
```

Figura 138. Archivo Principal Activity con inicialización de variables

Los botones de iniciar de la Figura 139 y detener de la Figura 140 almacenan respectivamente el dato ingresado al interactuar con los mismos. Una vez almacenado el dato, se llama a la función ingresar proceso.

```
botonIniciar.setOnClickListener { it: View!
    //al igual que en python se verifica si el estado actual del proceso es cero
    //indica que el proceso esta detenido por lo que puedo inciarlo
    //cambiandolo de cero a uno, indicando que proceda con el proceso que continua(capturar camara)
    //si el valor ees diferente de cero puede detener el proceso
    val enviar = if (VariablesGlobales.proceso == 0) "1" else "0"
    //llamar a la funcion ingresarProceso la misma que permite enviar al servidor
    //el valor del proceso
    ingresarProceso(enviar.toString(),VariablesGlobales.idUsuario).start()

}
```

Figura 139. Botón iniciar

```
botonDetener.setOnClickListener { it: View!
    //al igual que en python se verifica si el estado actual del proceso es cero
    //indica que el proceso esta detenido por lo que puedo inciarlo
    //cambiandolo de cero a uno, indicando que proceda con el proceso que continua(capturar camara)
    //si el valor ees diferente de cero puede detener el proceso
    val enviar = if (VariablesGlobales.proceso == 0) "1" else "0"
    //llamar a la funcion ingresarProceso la misma que permite enviar al servidor
    //el valor del proceso
    ingresarProceso(enviar.toString(),VariablesGlobales.idUsuario).start()

}
```

Figura 140. Botón detener

En la Figura 141 y Figura 142, los botones de automático y manual almacenan respectivamente el dato ingresado al interactuar con los mismos. Una vez almacenado el dato, se llama a la función actualizar automático.

```
botonAutomatico.setOnClickListener { it: View!
    //al igual que en python se verifica si el estado actual es automático o manual
    //verificando si está en uno esta en automático y lo puede pasar a manual en valor cero
    //verificando si está en cero esta en manual y lo puede pasar a automático en valor uno
    val enviar = if (VariablesGlobales.automatico == 0) "1" else "0"
    //llamar a la funcion actualizarautomático la misma que permite enviar al servidor
    //el valor del estado
    ActualizarAutomaticoTask(enviar).start()
```

Figura 141. Botón automático

```
botonManual.setOnClickListener { it: View!
    //al igual que en python se verifica si el estado actual es automático o manual
    //verificando si está en uno esta en automático y lo puede pasar a manual en valor cero
    //verificando si está en cero esta en manual y lo puede pasar a automático en valor uno
    val enviar = if (VariablesGlobales.automatico == 0) "1" else "0"
    //llamar a la funcion actualizarautomático la misma que permite enviar al servidor
    //el valor del estado
    ActualizarAutomaticoTask(enviar).start()
}
```

Figura 142. Botón manual

La siguiente función de la Figura 143, permite obtener el valor de porcentaje del proceso actual, este valor, se obtiene a través de la base de datos.

```
private fun ingresarPro(valor: String,usuario: String) {                    ⚠ 34 ⚠ 20 ✓ 1004
    try {
        //se estable la direccion url a la cual se enviará los datos
        val url = URL( spec: VariablesGlobales.ip+"datos/actualizarEstadoV1.php?valor=${URLEncoder.encode(valor,
        //se abre una conexion http a la url anterior
        val httpURLConnection = url.openConnection() as HttpURLConnection
        //configurar el metodo de conexión GET
        httpURLConnection.requestMethod = "GET"
        //se obtiene la respueta http
        val responseCode = httpURLConnection.responseCode
        //verificar si la respuesta es correcta
        if (responseCode == HttpURLConnection.HTTP_OK) {
            //obtener el flujo de entrada de la respuesta
            val inputStream = httpURLConnection.inputStream
            //crea un lector de tipo buffer para procesar la respuesta anterior
            val reader = BufferedReader(InputStreamReader(inputStream))
            //permite leer cada dato
            val response = StringBuilder()
            //daclarar la variable linea como string
            var line: String?
            //recorrer linea por linea
            while (reader.readLine().also { line = it } != null) {
                response.append(line)
            }
            //verificar si la respuesta es un JSON válido
            //el proceso siguiente es para validar si hay una respuesta correcta desde el servidor
            if (response.startsWith( prefix: "{") && response.endsWith( suffix: "}")) {
```

Figura 143. Función ingresar proceso

En la Figura 144, la función iniciar tarea, ejecuta la función ejecutar tarea y espera dos segundos para después volver a iniciar la función ejecutar tarea y continuar reiniciando el temporizador.

```
private fun iniciarTarea() {
    //ejecutar por primera vez la funcion ejecutar tarea
    ejecutarTarea()
    //crear y ejecutar proceso que se realizará cada 2 segundos
    //es un temporizados que espera dos segundos
    object : CountDownTimer( millisInFuture: 2000,  countDownInterval: 2000) {
        override fun onTick(millisUntilFinished: Long) {
            // No se utiliza en este caso
        }
        override fun onFinish() {
            // Cuando el temporizador llega a cero, ejecuta la tarea nuevamente
            ejecutarTarea()
            // Reinicia el temporizador
            start()
        }
    }.start()
}
```

Figura 144. Función iniciar tarea

La función ejecutar tarea de la Figura 145 crea la instancia de la Figura 146 en la cual, se desarrolla el proceso para obtención de los valores para los indicadores de clima.

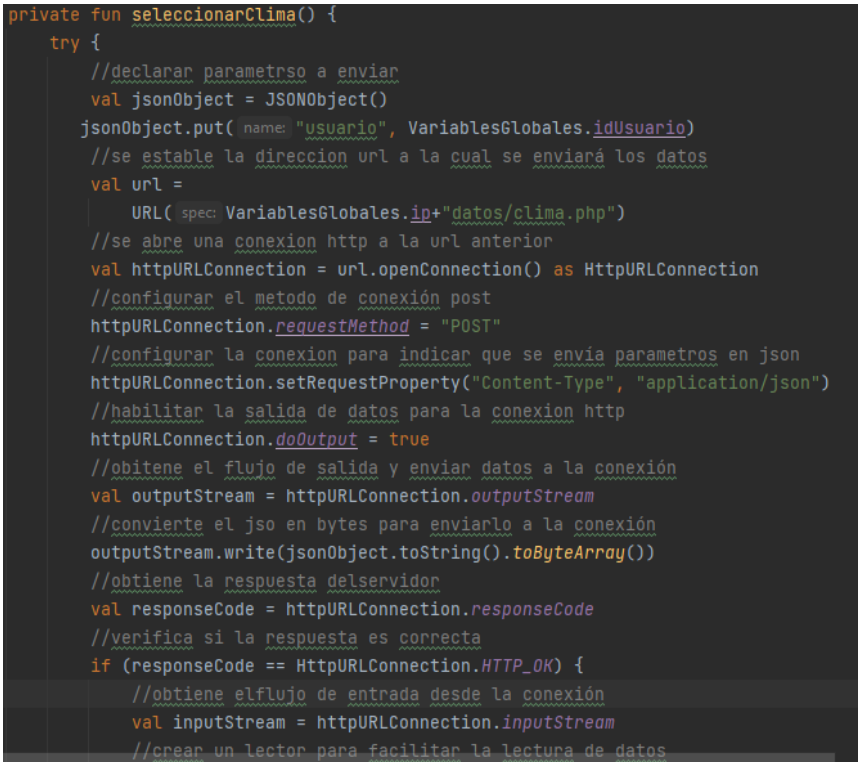
```
private fun ejecutarTarea() {
    // Crea una nueva instancia de la tarea y ejecútala
    val syncDataTask2 = DownloadBackupTask1()
    syncDataTask2.execute()
}
```

Figura 145. Función ejecutar tarea

```
private inner class DownloadBackupTask1() : AsyncTask<Void, Void, Boolean>() {
    @SuppressLint("WrongThread")
    override fun doInBackground(vararg params: Void?): Boolean {
        //llamar a la funcion seleccionaclima
        seleccionarClima()
        return false
    }
```

Figura 146. Función en segundo plano de la función ejecutar tarea

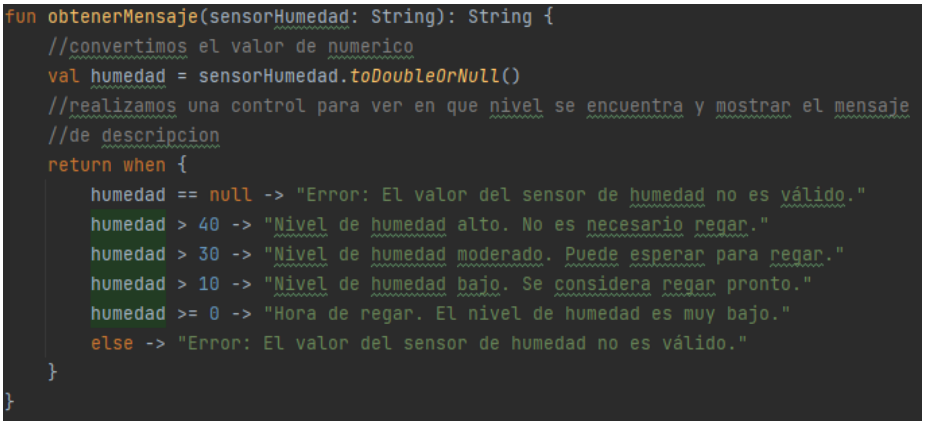
En la Figura 147, la función seleccionar clima permite obtener los valores de los indicadores de clima, inicia declarando las variables, las inicializa y realiza la petición a la base de datos para obtener todos los valores requeridos.



```
private fun seleccionarClima() {
    try {
        //declarar parametrso a enviar
        val jsonObject = JSONObject()
       jsonObject.put( name: "usuario", VariablesGlobales.idUsuario)
        //se estable la direccion url a la cual se enviará los datos
        val url =
            URL( spec: VariablesGlobales.ip+"datos/clima.php")
        //se abre una conexion http a la url anterior
        val httpURLConnection = url.openConnection() as HttpURLConnection
        //configurar el metodo de conexión post
        httpURLConnection.requestMethod = "POST"
        //configurar la conexion para indicar que se envía parametros en json
        httpURLConnection.setRequestProperty("Content-Type", "application/json")
        //habilitar la salida de datos para la conexion http
        httpURLConnection.doOutput = true
        //obitene el flujo de salida y enviar datos a la conexión
        val outputStream = httpURLConnection.outputStream
        //convierte el jso en bytes para enviarlo a la conexión
        outputStream.write(jsonObject.toString().toByteArray())
        //obtiene la respuesta delservidor
        val responseCode = httpURLConnection.responseCode
        //verifica si la respuesta es correcta
        if (responseCode == HttpURLConnection.HTTP_OK) {
            //obtiene elflujo de entrada desde la conexión
            val inputStream = httpURLConnection.inputStream
            //crear un lector para facilitar la lectura de datos
```

Figura 147. Función seleccionar clima

La función obtener mensaje al igual que en la interfaz anterior, se encarga de realizar las peticiones a la base de datos para obtener los valores del sensor de humedad y representarlos en el indicado de la aplicación móvil. De igual manera muestra sugerencias de riego en base al nivel de humedad del suelo, esto, se muestra en la Figura 148.

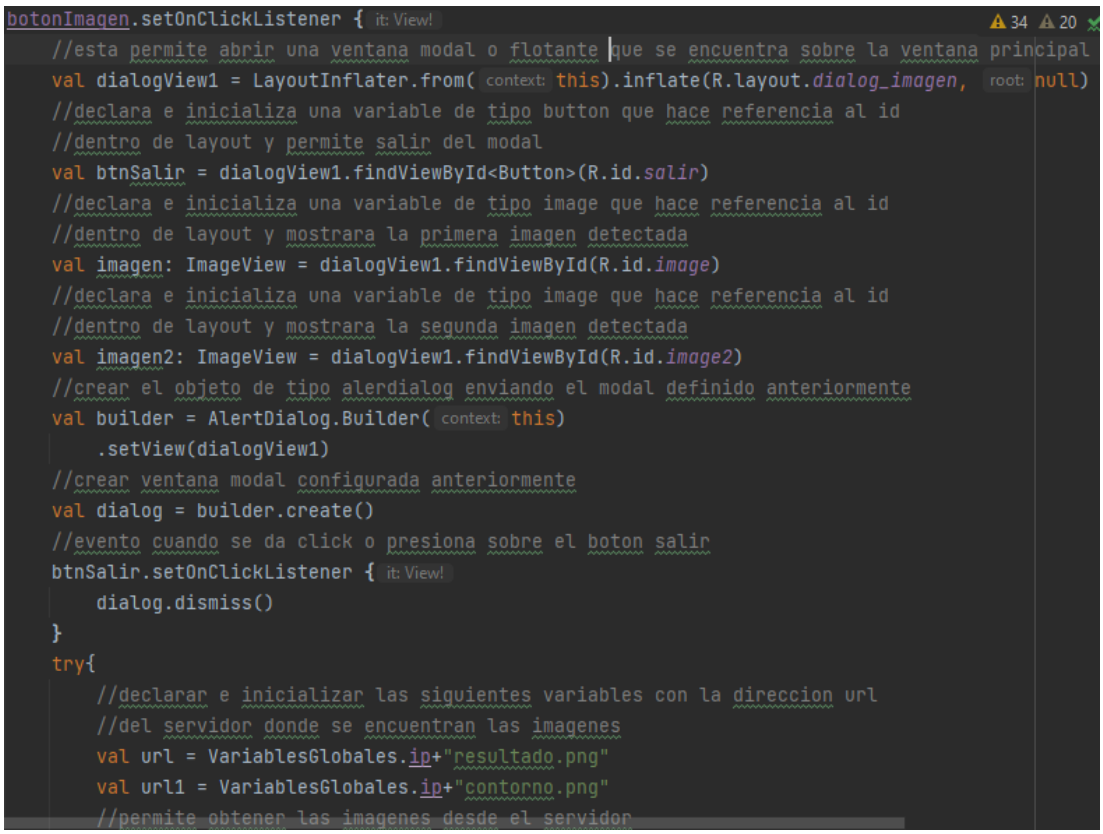


```
fun obtenerMensaje(sensorHumedad: String): String {
    //convertimos el valor de numerico
    val humedad = sensorHumedad.toDoubleOrNull()
    //realizamos una control para ver en que nivel se encuentra y mostrar el mensaje
    //de descripcion
    return when {
        humedad == null -> "Error: El valor del sensor de humedad no es válido."
        humedad > 40 -> "Nivel de humedad alto. No es necesario regar."
        humedad > 30 -> "Nivel de humedad moderado. Puede esperar para regar."
        humedad > 10 -> "Nivel de humedad bajo. Se considera regar pronto."
        humedad >= 0 -> "Hora de regar. El nivel de humedad es muy bajo."
        else -> "Error: El valor del sensor de humedad no es válido."
    }
}
```

Figura 148. Función obtener mensaje

El botón de imagen, de la Figura 149, es el botón de visualizar de la aplicación móvil, permite generar una nueva ventana en la cual, se puedan observar las imágenes de la

detección realizada, inicia declarando las variables e inicializando para realizar las peticiones a la base de datos.



```
botonImagen.setOnClickListener { it: View!                                    ⚠ 34 ⚠ 20
    //esta permite abrir una ventana modal o flotante que se encuentra sobre la ventana principal
    val dialogView1 = LayoutInflater.from( context: this).inflate(R.layout.dialog_imagen,  root: null)
    //declara e inicializa una variable de tipo button que hace referencia al id
    //dentro de layout y permite salir del modal
    val btnSalir = dialogView1.findViewById<Button>(R.id.salir)
    //declara e inicializa una variable de tipo image que hace referencia al id
    //dentro de layout y mostrara la primera imagen detectada
    val imagen: ImageView = dialogView1.findViewById(R.id.image)
    //declara e inicializa una variable de tipo image que hace referencia al id
    //dentro de layout y mostrara la segunda imagen detectada
    val imagen2: ImageView = dialogView1.findViewById(R.id.image2)
    //crear el objeto de tipo alerdialog enviando el modal definido anteriormente
    val builder = AlertDialog.Builder( context: this)
        .setView(dialogView1)
    //crear ventana modal configurada anteriormente
    val dialog = builder.create()
    //evento cuando se da click o presiona sobre el boton salir
    btnSalir.setOnClickListener { it: View!
        dialog.dismiss()
    }
    try{
        //declarar e inicializar las siguientes variables con la direccion url
        //del servidor donde se encuentran las imagenes
        val url = VariablesGlobales.ip+"resultado.png"
        val url1 = VariablesGlobales.ip+"contorno.png"
        //permite obtener las imagenes desde el servidor
```

Figura 149. Botón imagen con la declaración de variables

Una vez obtenidas las imágenes, se requiere la ubicación de estas en la nueva ventana y también, se debe limpiar el cache para prepararlo para almacenar nuevas imágenes en la próxima detección, como muestra la Figura 150. Una vez almacenados los datos, se llama a la función iniciar tarea.

```
        val url1 = VariablesGlobales.ip+"contorno.png"
        //permite obtener las imagenes desde el servidor
        //y ubicarlas dentro de los image correspondientes
        //se limpia la memoria cache para que no quede almacenado la imagen anterior
        Glide.with( activity: this) RequestManager
            .load(url) RequestBuilder<Drawable!>
            .skipMemoryCache( skip: true)
            .diskCacheStrategy(DiskCacheStrategy.NONE)
            .into(imagen)
        //permite obtener las imagenes desde el servidor
        //y ubicarlas dentro de los image correspondientes
        //se limpia la memoria cache para que no quede almacenado la imagen anterior
        Glide.with( activity: this) RequestManager
            .load(url1) RequestBuilder<Drawable!>
            .skipMemoryCache( skip: true)
            .diskCacheStrategy(DiskCacheStrategy.NONE)
            .into(imagen2)
    }catch (e: Exception){
        //si existe un error se muestra en consola e tiempo de desarrollo el error encontrado
        Log.d( tag: "Imagen",  msg: "Error: ${e.message.toString()}")
    }
    //proceso para abrir la ventana modal
    dialog.show()
}
//se llama a la funcion iniciar tarea, que se encargará de traer lo datos del servidor
iniciarTarea()
```

Figura 150. Botón imagen con los datos obtenidos

En la Figura 151, se presenta la ventana generada del botón imagen, en la aplicación móvil es el botón azul de visualizar, aquí, se ubica la imagen capturada del jardín y la detección, ambas obtenidas de la base de datos.

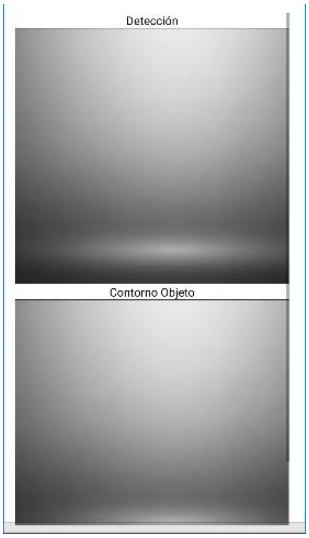


Figura 151. Ventana de visualizar

En la Figura 152, se muestra el archivo layout de los colores disponibles para el uso en la aplicación

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="error">#FF1744</color>
    <color name="dash1">#5eb05a</color>
    <color name="dash2">#4c5dbc</color>
    <color name="dash3">#ea523b</color>
    <color name="dash4">#4ba491</color>
    <color name="dash5">#4cb4f8</color>
    <color name="dash6">#A66C20</color>
    <color name="dash7">#A8EA70</color>
    <color name="dash8">#656763</color>
    <color name="transparent">#00FFFFFF</color>
</resources>
```

Figura 152. Layout Colores

A continuación, se presenta en la Figura 153 las librerías usadas, en la Figura 154 los premisos requeridos para el uso de la aplicación y en la Figura 155 las versiones necesarias para usar la aplicación.

```
dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.9.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    implementation 'com.github.bumptech.glide:glide:4.12.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
}
```

Figura 153. Librerías de la aplicación móvil

```xml
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Figura 154. Permisos que usa la aplicación

```
android {
    namespace 'com.jimenez.aspersion'
    compileSdk 34

    defaultConfig {
        applicationId "com.jimenez.aspersion"
        minSdk 23
        targetSdk 34
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
```

Figura 155. Versiones necesarias para usar la aplicación móvil

En la Figura 156 y Figura 157, se presenta el layout “Activity principal” el cual corresponde a la ventana principal, se detalla los estilos usados, y la configuración realizada para los botones y cada uno de los indicadores.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".PrincipalActivity">
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:orientation="vertical"
        >
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="0dp"
            android:orientation="vertical"
            android:background="@drawable/bordes"
```

Figura 156. Configuración de límites de la ventana y ventana deslizante

```
<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:cardCornerRadius="10dp"
    app:cardElevation="10dp"
    android:layout_margin="2dp"

    android:layout_weight="1"
    android:backgroundTint="@color/dash8">
    <LinearLayout
        android:id="@+id/seccion_automatico"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        android:layout_weight="1"
        >
        <TextView
            android:id="@+id/lista_cuenS4Dta_nuojmbre13"
            android:layout_width="220dp"
            android:layout_height="wrap_content"
            android:text="Automático"
            android:layout_marginBottom="5dp"
            android:layout_marginTop="8dp"
            android:textAlignment="center"
            android:textColor="@color/white"
```

Figura 157. Código de diseño de la pantalla principal

3.8 Pruebas de funcionamiento

3.8.1 Resultados entrenamiento

La documentación de Ultralytics, recomienda utilizar más de 1500 imágenes por clase para los dataset, empezar el entrenamiento con 300 épocas, y verificar el sobreajuste. Cuando hay un sobreajuste rápido se debe reducir las épocas para evitar errores de detección, caso contrario se puede aumentar las épocas hasta obtener el entrenamiento correcto [76]. Aun así, el modelo de entrenamiento “YOLOv5 tutorial” que brinda Google Colaboratory con Ultralytics usa 128 imágenes dando resultados en la detección del 96%.

En cambio, el Estudio de S. Susan y Kumar A, sobre el muestreo optimizado de conjuntos de datos desequilibrados de clases, determina que un límite de clases bien definido produce un rendimiento alto, aquí intervienen tanto las clases mayoritarias y minoritarias desde 42 hasta 1240 muestras [77].

Teniendo en consideración que se puede obtener buenos resultados en el entrenamiento con 128 imágenes, se delimito un dataset de 480 imágenes de jardines predominantes de césped. Se inicio el entrenamiento con 300 épocas como recomienda Ultralytics y se verifico que había un sobreajuste por lo cual se disminuyó a 200 épocas donde se obtuvo una mejor detección. Se uso la aplicación "ClearML", para interpretar los resultados del modelado de entrenamiento.

Inicialmente, se realiza un análisis de la evolución de las métricas, se realiza una comparativa con los valores de las métricas obtenidas donde, se verifico el mejor entrenamiento en base a las 200 iteraciones.

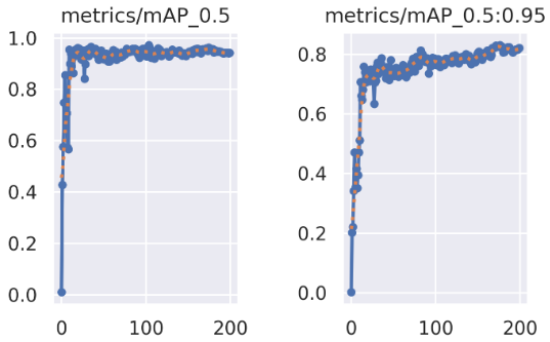


Figura 158. Métricas mAP_0.5 y mAP_0.5:0.95

En la Figura 158, se tiene las gráficas de evolución de las métricas para 200 iteraciones con los valores de mAP 0.5 y mAP 0,5:0,95. En la primera gráfica, se observa que las predicciones evaluadas como "césped detectado" en una intersección sobre unión (IoU) superior 50%, se acercan a la proyección del 100%. En la segunda gráfica, se observa las predicciones evaluadas en una IoU superior al 50% y menor o igual al 0,95% aproximándose con una mejor curva al 90%.

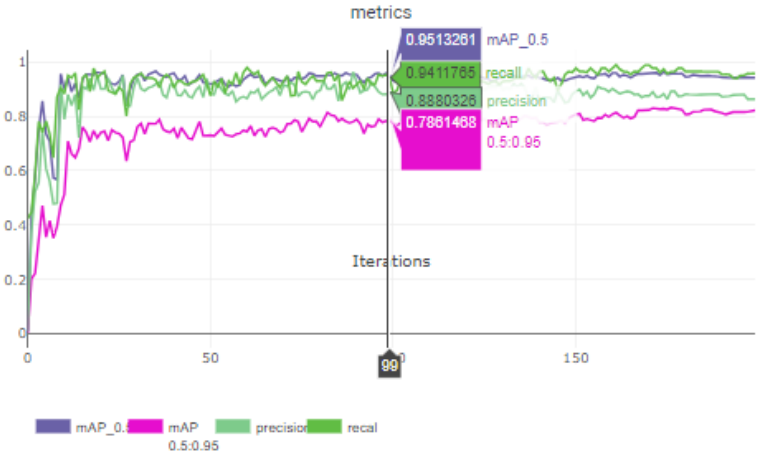


Figura 159. Métricas obtenidas en 100 iteraciones

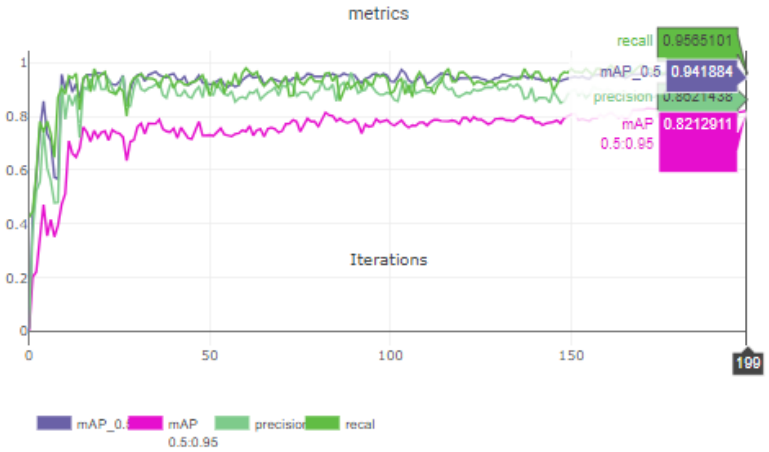


Figura 160. Métricas obtenidas en 200 iteraciones

En base a la Figura 159 y a la Figura 160, se desarrolla una tabla comparativa entre los valores obtenidos en las métricas. En la Tabla 28, se observa un valor más alto para 100 iteraciones en la precisión de predicciones y en mAP 0,5, mientras que, en 200 iteraciones, se obtiene valores más altos para: recall que es el porcentaje de casos positivos en las detecciones positivas; y mAP 0,5:0,95.

Tabla 28. Precisión y sensibilidad obtenida en base a iteraciones

Épocas	Imágenes	Precisión	Recall (Sensibilidad)	Precisión media mAP	mAP 0,5:0,95
100	480	0,88	0,94	0,95	0,78
200	480	0,86	0,96	0,94	0,82

Por consiguiente, los resultados obtenidos de esta tabla sugieren que, al realizarse 200 iteraciones, se tiene 86% de precisión y 96% de sensibilidad. Al tener menor precisión

y una alta sensibilidad, el modelo detecta bien la clase, obteniendo así un entrenamiento bastante confiable.

A continuación, se presenta en la Figura 161 la matriz de confusión con los siguientes valores: verdadero positivo de TP=0,95, verdadero negativo de TN=0, falso negativo de FN=0,05, falso positivo de FP=1. Estos valores, se obtienen porque es una detección de clase única con fondo, el valor TN=0, significa que no hubo muestras negativas verdaderas para la clase de fondo en el conjunto de validación. De manera similar, el valor FP=1, podría significar que hubo un falso positivo detectado como fondo. Por lo tanto, no implica que el modelo sea incorrecto.

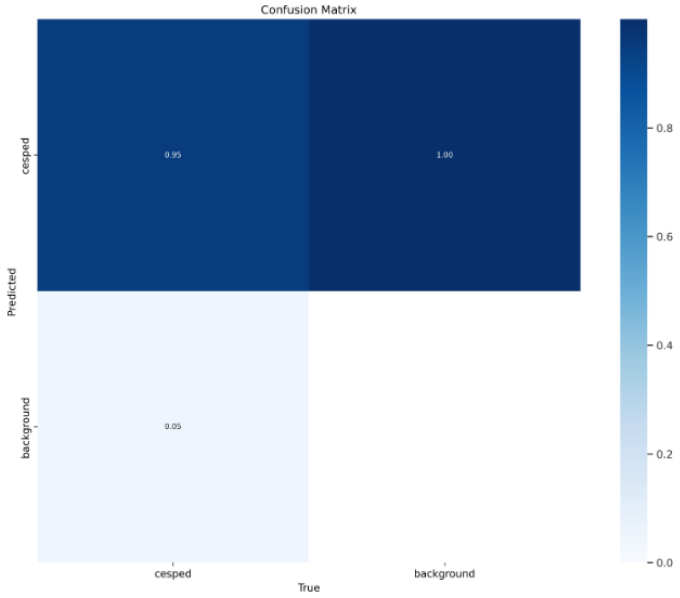


Figura 161. Matriz de confusión

3.8.2 Pruebas en la interfaz

Se realiza pruebas para la verificación de almacenamiento de datos y activación de alertas dentro del registro de usuarios y en la ventana de inicio de sesión.

Dentro de la ventana para el registro de usuario de la Figura 162, se requiere el ingreso de datos correctos para proceder con el registro en la base de datos, si los datos están en blanco, se genera una alerta de datos obligatorios.

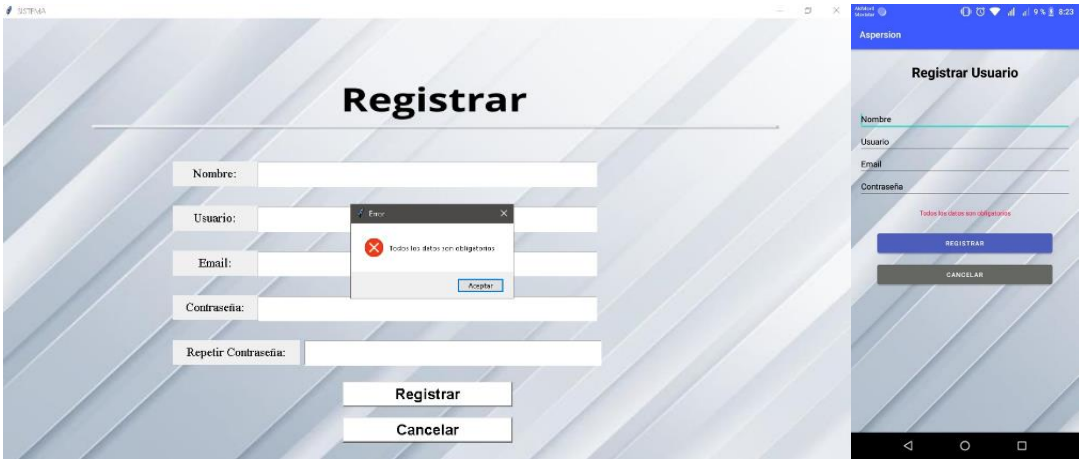


Figura 162. Alerta en registro de usuario

En la Figura 163 y Figura 164, al registrar un nuevo usuario con todos los campos llenados correctamente, se genera un mensaje informativo que indica que el registro es correcto.

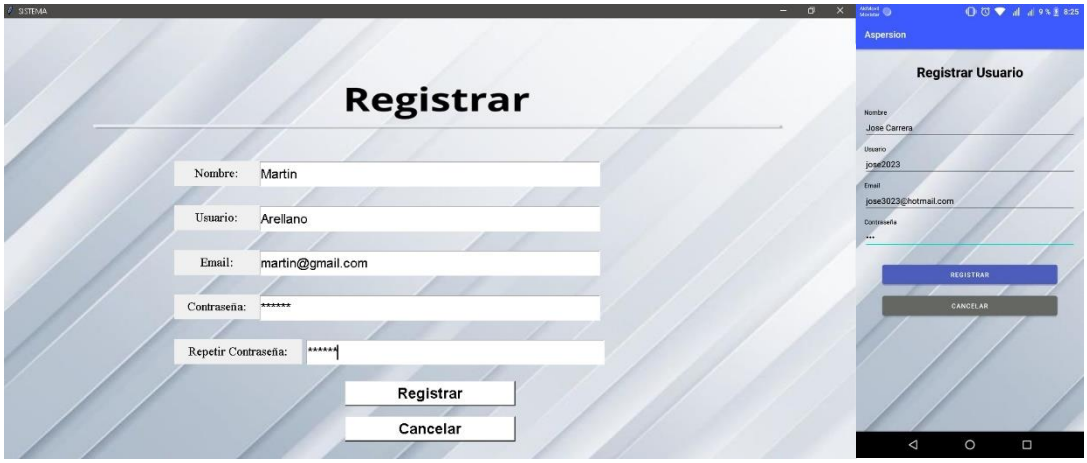


Figura 163. Ingreso de nuevo usuario

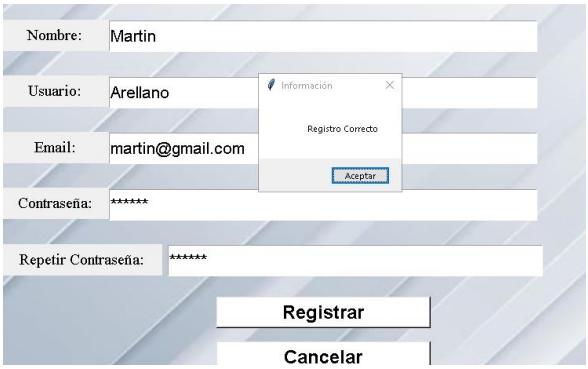


Figura 164. Registro correcto de usuario

Una vez realizado el registro de nuevo usuario, el sistema regresará automáticamente a la ventana de iniciar sesión, y el registro, se almacenará en la base de datos, como muestra la Figura 165.

	id	nombre	usuario	email	contrasena	fecha_registro
Editar Copiar Borrar	1	Prueba	patio	patio@gmail.com	202cb962ac59075b964b07152d234b70	2023-11-15 19:58:24
Editar Copiar Borrar	2	Juan	juan	juan@gmail.com	202cb962ac59075b964b07152d234b70	2023-11-17 20:30:28
Editar Copiar Borrar	3	Jose Carrera	jose2023	jose2023@hotmail.com	202cb962ac59075b964b07152d234b70	2023-11-29 12:48:42
Editar Copiar Borrar	4	Martin	Arellano	martin@gmail.com	925d7518fc597af0e43f5606f9a51512	2024-01-04 11:20:53

Figura 165. Almacenamiento de nuevo usuario en la base de datos

Dentro de la ventana para el inicio de sesión, se requiere el ingreso de datos correctos para proceder con la verificación en la base de datos, si los datos son incorrectos en el campo de usuario o contraseña o ambos, se genera una alerta de datos incorrectos como en la Figura 166.

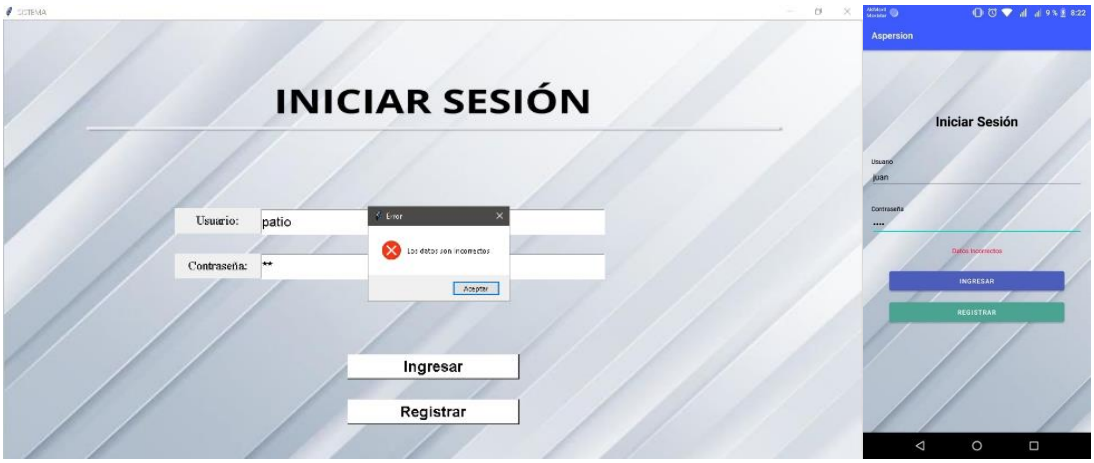


Figura 166. Alerta de datos incorrectos por llenar campos incorrectamente

De igual forma, en la Figura 167, se muestra un ejemplo en el que el usuario intenta ingresar y los datos están en blanco, entonces genera una alerta de datos incorrectos.

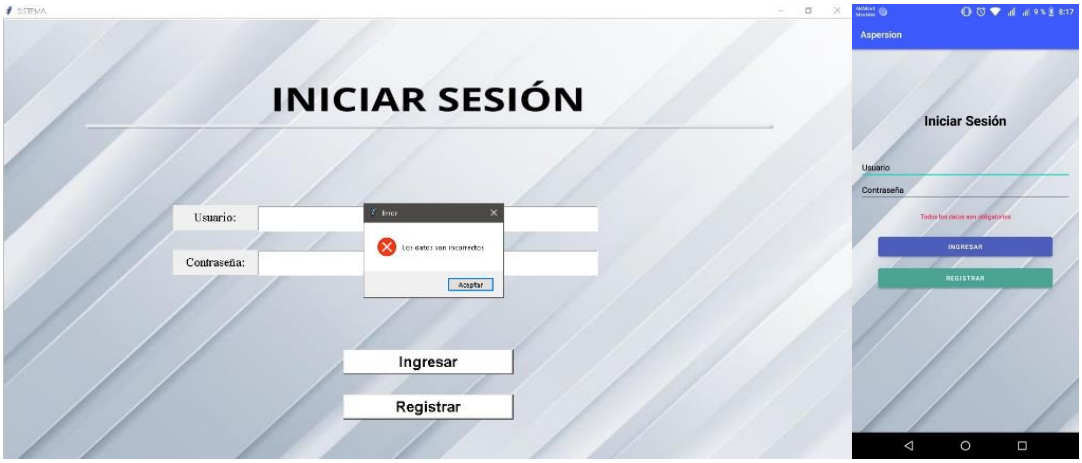


Figura 167. Alerta de datos incorrectos por campos en blanco

3.8.3 Área de pruebas para el riego

Todas las pruebas, se realizaron en un jardín extenso con dimensiones de 12m de largo por 8m de ancho, el jardín cuenta con césped kikuyo, el cual es denso y le recomiendan un valor de humedad de 20-60%. Por su magnitud, se decidió realizar pruebas en 9m de largo. A continuación, se presenta la Figura 168 con las distancias usadas en el desarrollo de las pruebas y la Figura 169 con una vista real del área con el sistema.

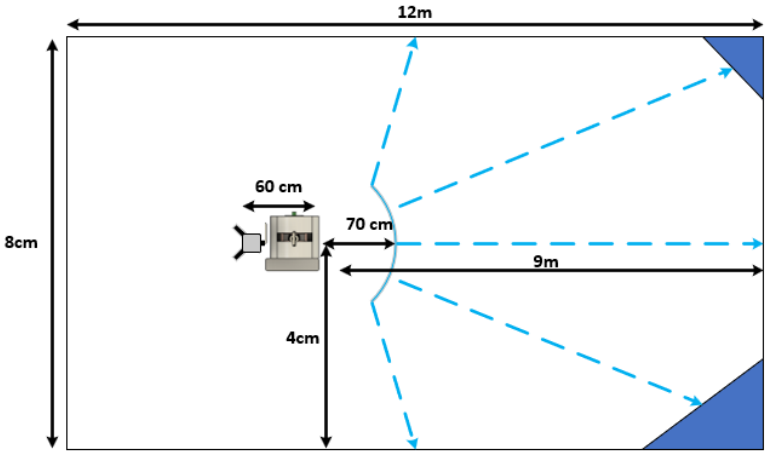


Figura 168. Distancias de ubicación del sistema

Figura 169. Ubicación del sistema en el jardín

3.8.4 Prueba 1: Evaluación de detección y riego general sin obstáculos

Esta prueba, se enfoca en la evaluación de la detección y del riego general sin obstáculos, se llevó a cabo la detección exhaustiva de toda el área del jardín, con un reconocimiento de césped del 96%, sin la presencia de objetos adicionales. Durante esta prueba, se logró una detección precisa de la extensión total del césped, demostrando la eficacia del sistema para identificar y delimitar la zona de riego. En esta fase, se evaluó la capacidad del sistema para realizar un riego completo, asegurándose de respetar los bordes del jardín. Los resultados obtenidos indican un desempeño satisfactorio en la identificación del área de interés y en la aplicación efectiva del riego, sentando así las bases para pruebas posteriores con condiciones más desafiantes.

En la Figura 170, se identifican los resultados obtenidos con activación del modo automático. Se observa que la detección del césped tiene una precisión del 96% y la detección de los contornos es aceptable.

En la Figura 171, se identifican los resultados obtenidos con activación del modo manual. De igual manera presenta una detección de césped del 96%. En los contornos, se aprecia que el contorno de objeto detecta una parte del césped, se considera como un error de detección leve debido a la iluminación.

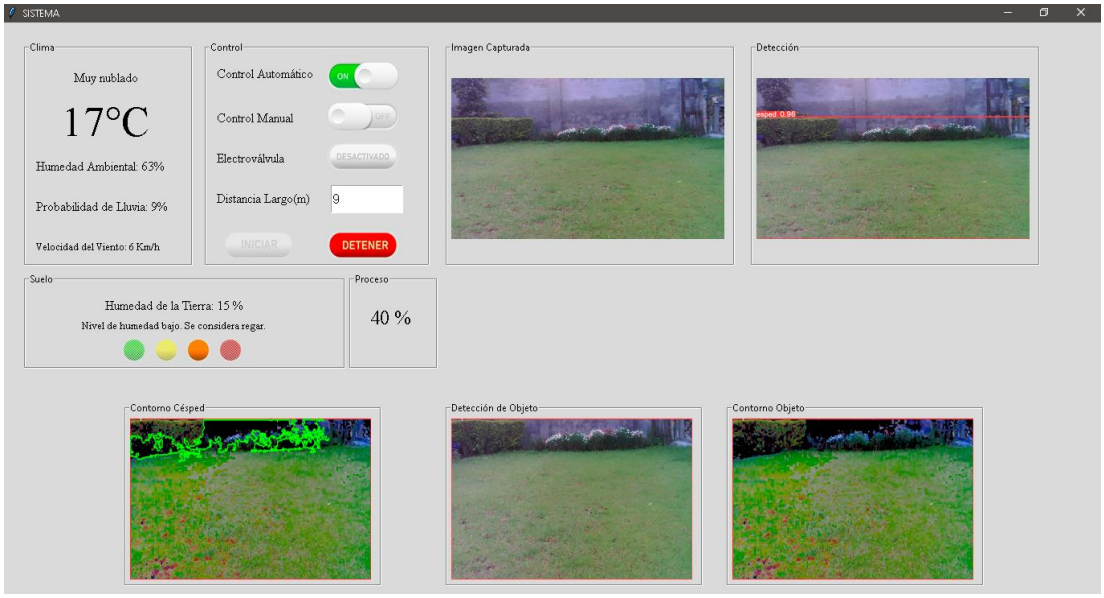


Figura 170. Prueba 1 en modo automático

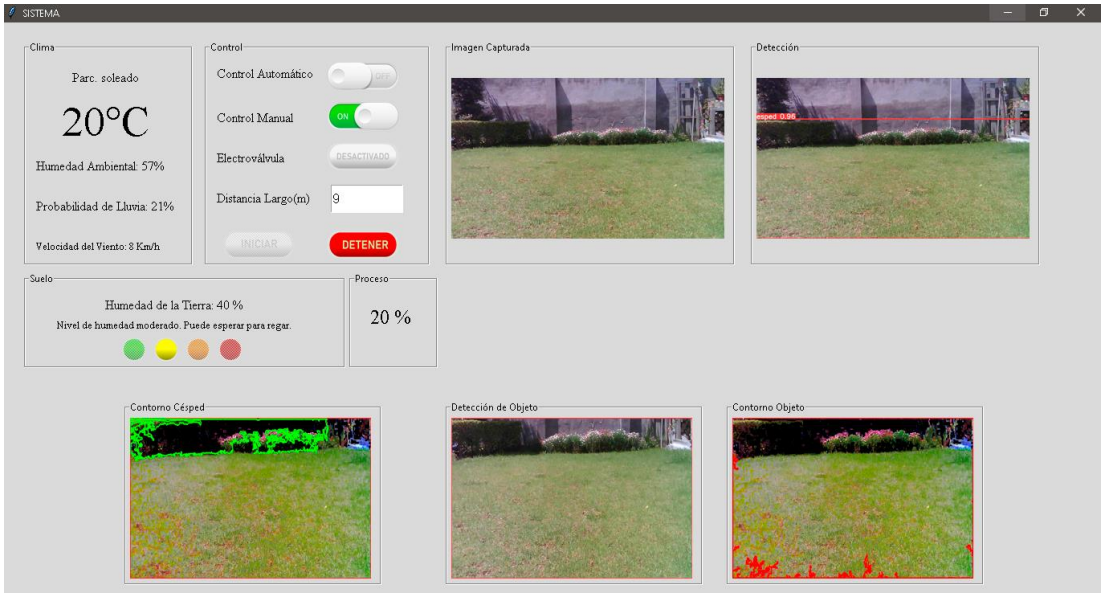


Figura 171. Prueba 1 en modo manual

3.8.5 Prueba 2: Evaluación de detección y riego general con obstáculos

En esta segunda prueba, se llevó a cabo la evaluación de detección y riego general con obstáculos, incorporando un obstáculo grande representado por una caja de frutas y un obstáculo pequeño representado por una botella. Durante esta prueba, el sistema demostró su capacidad para identificar obstáculos en el área de riego. Se observó que el sistema detectó únicamente la caja de frutas. Se concluye que objetos muy pequeños no son reconocidos por el sistema debido al procesamiento con escala de grises, a pesar

de ello, el sistema procedió con el riego, mostrando eficacia al evitar mojar la caja de frutas mientras irrigaba alrededor de la botella. Este resultado indica una respuesta adecuada del sistema ante obstáculos específicos y sienta las bases para la mejora de la detección y respuesta en presencia de múltiples y variados obstáculos.

En la Figura 172, se identifican los resultados obtenidos con activación del modo automático. Se puede apreciar, la precisión en la detección del césped alcanza un 96%, y la identificación de los contornos es satisfactoria. No se reconoce el segundo objeto debido a su magnitud, el sistema solo reconoce objetos medianos y grandes dentro del área.

En la Figura 173, se identifican los resultados obtenidos con activación del modo manual. De igual manera presenta una detección de césped del 96% y un reconocimiento del objeto mediano, representado aquí por una caja de madera para frutas.

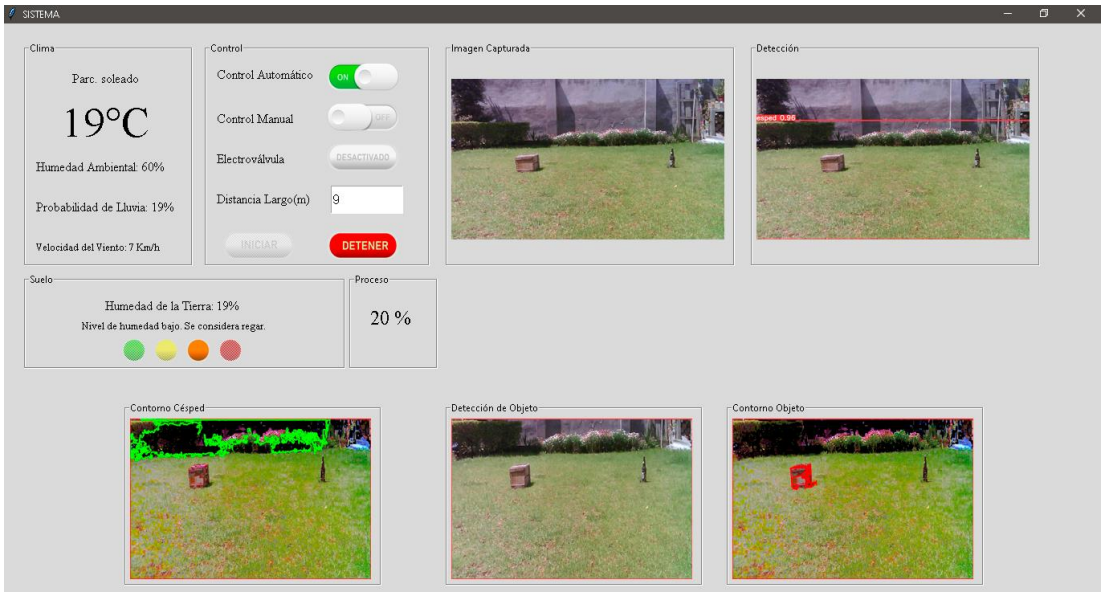


Figura 172. Prueba 2 en modo automático

Figura 173. Prueba 2 en modo manual

3.8.6 Prueba 3: Evaluación de detección y riego en área delimitada

En esta última prueba, se llevó a cabo la evaluación de detección y riego en un área delimitada, donde, se redujo el tamaño del patio a 2,50 metros de ancho mediante el uso de lonas negras que marcaban claramente los límites del jardín. Durante esta prueba, el sistema demostró una capacidad efectiva de reconocer y respetar la delimitación del área de jardín más pequeña. Las imágenes de detección reflejaron correctamente el reconocimiento de los contornos del césped, mientras que los contornos de los objetos, se marcaron como áreas oscuras. El sistema, en consecuencia, ejecutó un riego adecuado dentro de los límites establecidos, destacando la capacidad del sistema para adaptarse a diferentes tamaños de áreas de jardín mediante la identificación precisa de los contornos definidos.

En la Figura 174, se identifican los resultados obtenidos con activación del modo automático. La precisión en la detección del césped es del 95%, y la identificación de los contornos es satisfactoria.

De igual manera en el modo manual para la misma prueba presentada en la Figura 175, se tiene una precisión en la detección del césped del 95%, y detección de los contornos satisfactoria.

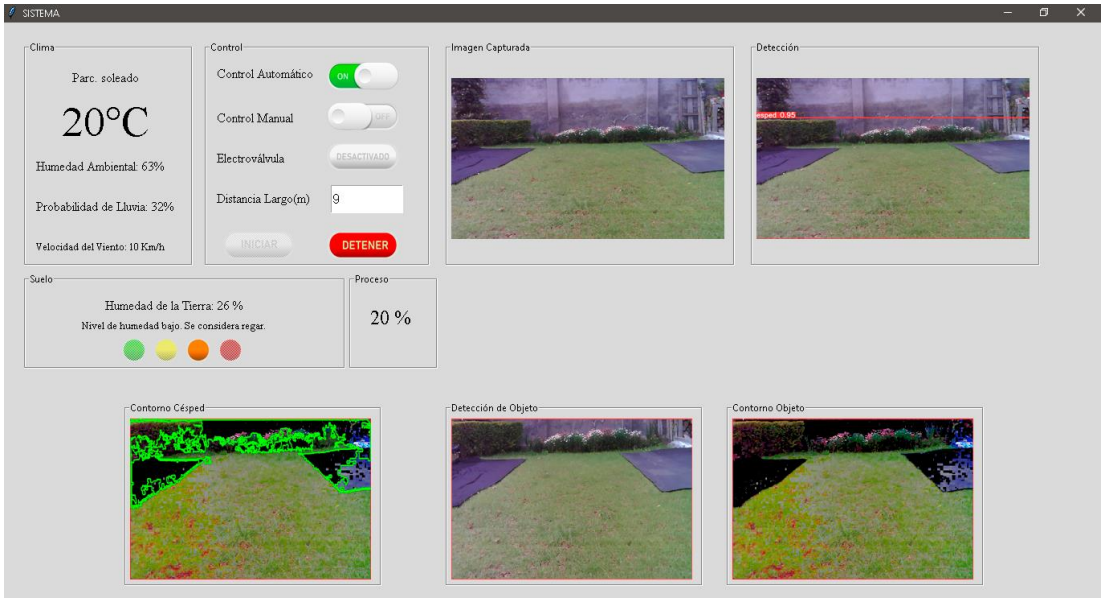


Figura 174. Prueba 3 en modo automático

Figura 175. Prueba 3 en modo manual

3.8.7 Resultados de las pruebas

Durante las pruebas realizadas, se evidencio el uso eficiente de la aplicación móvil para el monitoreo, se verifico la adquisición de datos y su visualización adecuada como muestra la Figura 176.

Figura 176. Monitoreo a través de la aplicación móvil

Es importante resaltar que duran la ejecución de las pruebas de funcionamiento, la aplicación móvil también registro los mismos datos, en la Figura 177, se presentan los resultados obtenidos de la imagen capturada y la detección de contorno realizada para cada una de las pruebas.

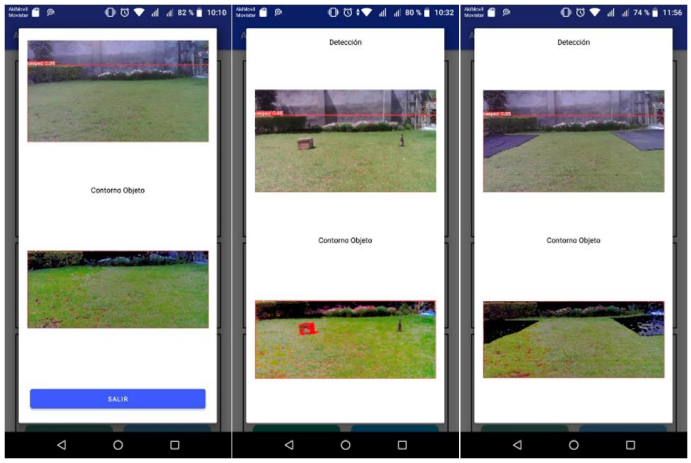


Figura 177. Imágenes de la detección obtenidas de las pruebas 1, 2 y 3

Se realiza una tabla de resumen de las pruebas realizadas para evaluar la calidad del sistema. En la Tabla 29, se compara las diferentes pruebas en base a los aspectos de reconocimiento de césped, calidad de imagen capturada, contorno de césped generado, contorno de objeto y riego.

Tabla 29. Tabla de calidad

	Reconocimiento Césped	Calidad de imagen	Contorno Césped	Contorno Objeto	Riego
Prueba 1: Modo Automático	96%	Excelente	Generado correctamente	No hay objeto	Completo
Prueba 1: Modo Manual	96%	Aceptable	Generado correctamente	Contorno césped áreas oscuras	Completo
Prueba 2: Modo Automático	96%	Excelente	Generado correctamente	Contorno caja	Evita caja
Prueba 2: Modo Manual	96%	Excelente	Generado correctamente	Contorno caja	Evita caja
Prueba 3: Modo Automático	95%	Aceptable	Generado correctamente	No hay objeto	Evita área oscura
Prueba 3: Modo Manual	95%	Aceptable	Generado correctamente	No hay objeto	Evita área oscura

En promedio el reconocimiento del césped es del 95,6%, en todas las pruebas, se obtuvo una calidad de imagen excelente y un reconocimiento del contorno correcto. En las diferentes pruebas, se obtuvo el contorno de objeto esperado y el riego adecuado dentro de lo esperado para cada prueba.

En la Tabla 30, se presenta el tiempo que tarda en completarse cada fase seleccionada. Se verifica el tiempo que demora la interfaz del sistema en responder desde el momento en que, se oprime el botón "Ingresar". Para el inicio del modo, se verifica el tiempo de espera necesario para que inicie, ya sea en el modo automático o manual. Dentro de la imagen capturada, se evalúa el tiempo que demora en visualizarse dentro de la interfaz. Una vez obtenida la imagen, se determina el tiempo que tarda en mostrarse la imagen del proceso de detección. La sección de interfaz abarca el tiempo total necesario para llevar a cabo todo el proceso requerido en la interfaz. El inicio del riego, se delimita desde que inicia el modo seleccionado. La duración del riego consiste en el tiempo necesario para completar todo el ciclo de riego.

Tabla 30. Tiempos de respuesta de interfaz y riego

	Ingreso	Inicio de modo	Imagen capturada	Detección	Duración Interfaz	Inicio de riego	Duración de riego
Prueba 1: Modo Automático	4,1s	33,67s	25,71s	4,39s	31,1s	59s	6min
Prueba 1: Modo Manual	4,5s	12,80s	25,94s	4,48s	31,42s	1min 01s	5min 58s
Prueba 2: Modo Automático	4,2s	29,3s	24,96s	4,27s	30,23s	9,2s	4min 49s
Prueba 2: Modo Manual	4,84s	15,36s	24,74s	4,32s	30,06s	9,1s	4min 51s
Prueba 3: Modo Automático	3,98s	32,52s	29,3s	4,46s	34,76s	1min 11s	2min 50s
Prueba 3: Modo Manual	4,62s	8,64s	31,13s	4,18s	36,31s	1min 08s	2min 51s
Promedio	4,37s	22,04s	26,96s	4,35s	32,31s	-	-

El tiempo promedio de ingreso para cada prueba es de 4,37 segundos, lo cual es bastante aceptable. Se observa que el modo automático tarda más en iniciar debido a la verificación del sensor de humedad y la identificación de la acción correspondiente. Por otro lado, en el modo manual, inicia al presionar el botón "Iniciar". La captura de imagen depende en gran medida de la conexión establecida con el ESP32-CAM, con un tiempo promedio de 26,96 segundos. El proceso de detección tiene una demora de 4,35 segundos antes de que aparezca la imagen. En resumen, el proceso completo realizado en la interfaz tiene un promedio de 32,31 segundos, y la duración del ciclo de riego depende del área reconocida.

Para determinar el consumo hídrico del sistema en el Anexo J, se desarrollaron una serie de cálculos para obtener el valor de consumo de agua teórico, el cual es de 40,15 litros en los 6 minutos que demora el riego completo en el jardín durante el escenario de la prueba 1. En la Tabla 31, se tiene mediciones tomadas de la cantidad de agua que usa el sistema de riego en 5 ocasiones. Se requiere las fórmulas de error absoluto y relativo presentadas a continuación:

$$ErrorAbsoluto = |ValorTeorico - ValorMedido| \quad (1)$$

$$Error\ Relativo = \left|\frac{ValorTeorico - ValorMedido}{ValorTeorico}\right| \times 100 \quad (2)$$

Tabla 31. Mediciones del consumo de agua

Mediciones	**Valor en litros**	**Error absoluto**	**Error Relativo %**
Medición 1	46,00	5,85	14,57
Medición 2	45,75	5,6	13,94
Medición 3	46,25	6,1	15,19
Medición 4	46,36	6,21	15,46
Medición 5	45,80	5,65	14,07
Promedio	46,03	5,88	14,64

El error relativo obtenido del 14,64% indica que la medición se encuentra en un rango aceptable, lo que también determino que el consumo de agua es adecuado en un 85,36%.

Finalmente, se realizan pruebas de comunicación y tráfico de paquetes. Se verifica la comunicación del servidor tanto con la ESP32-CAM como con el módulo ESP32.

En la Figura 178, se muestra el total de paquetes capturados, así como el tiempo transcurrido en cada paquete. En la Figura 179, se tomó el último paquete para verificar los tiempos, este paquete fue capturado en 0,0534 segundos. A continuación, en la Tabla 32, se muestra un resumen del análisis de paquetes entre el servidor y la ESP32-CAM.

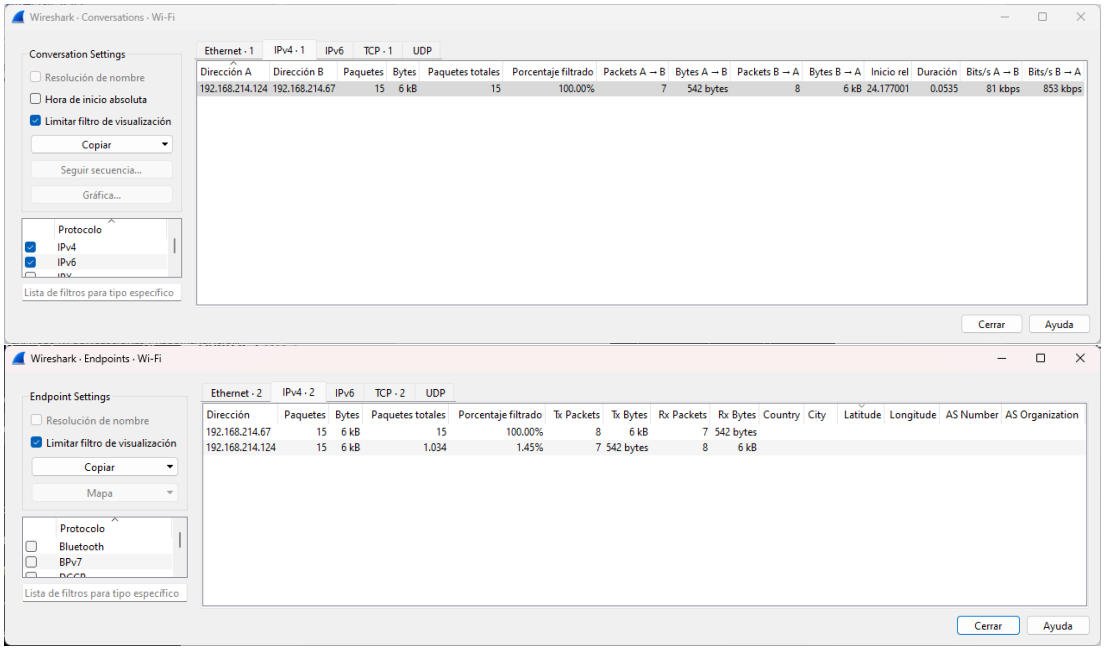


Figura 178. Paquetes enviados a la ESP32-CAM y al servidor

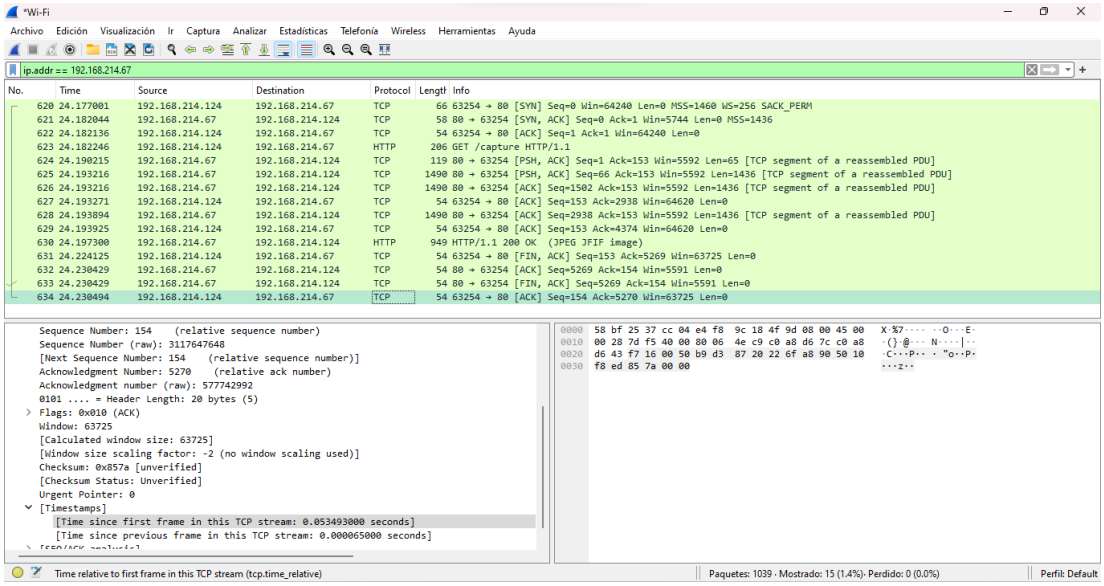


Figura 179. Paquetes capturados en Wireshark de la ESP32-CAM

Tabla 32. Resumen de los paquetes de comunicación con la ESP32-CAM

Parámetros	Servidor	ESP32-CAM
Dirección IP	192.168.214.124	192.168.214.67
Tamaño del paquete	6 kB	6 kB
Paquetes de comunicación	1034	15
Paquetes totales	15	15
Paquetes enviados	7	8
Paquetes recibidos	8	7

En la Figura 180, se muestra el total de paquetes capturados, así como el tiempo transcurrido en cada paquete. En la Figura 181, se tomó el último paquete para verificar los tiempos, este paquete fue capturado en 0,12 segundos. A continuación, en la

Tabla 33, se muestra un resumen del análisis de paquetes entre el servidor y la ESP32.

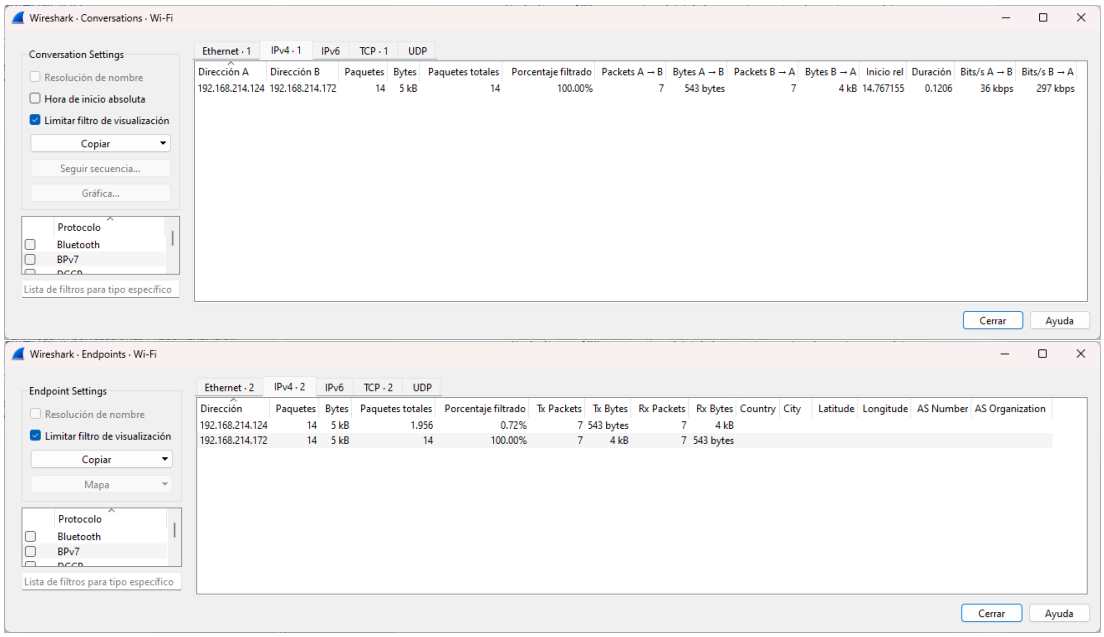


Figura 180. Paquetes enviados a la ESP32 y al servidor

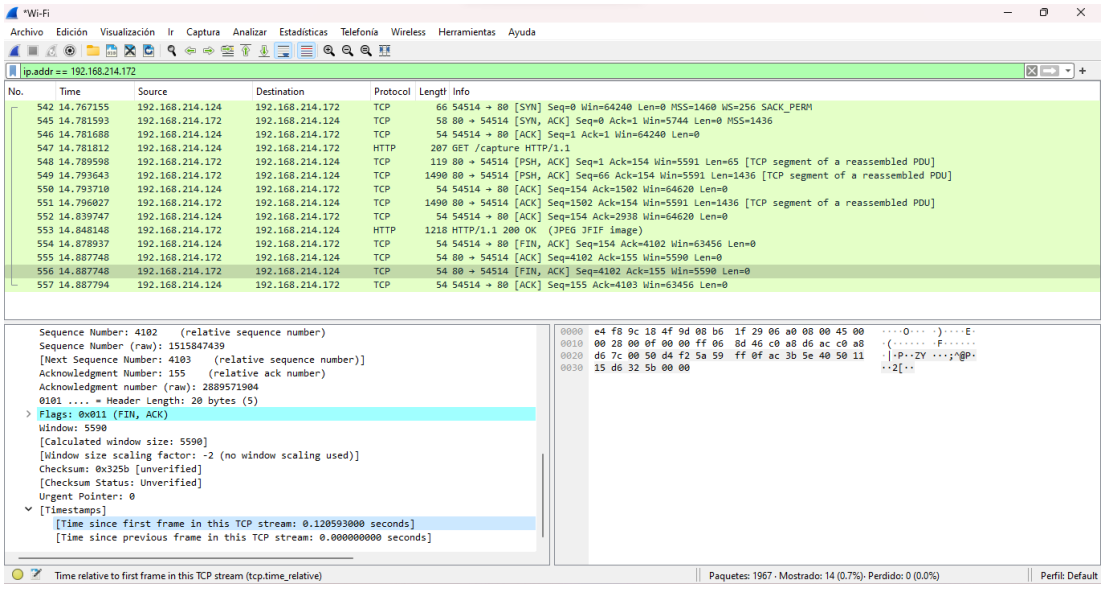


Figura 181. Paquetes capturados en Wireshark de la ESP32

Tabla 33. Resumen de los paquetes de comunicación con la ESP32

Parámetros	Servidor	ESP32
Dirección IP	192.168.214.124	192.168.214.172
Tamaño del paquete	5 kB	5 kB
Paquetes de comunicación	1952	14
Paquetes totales	14	14
Paquetes enviados	7	7
Paquetes recibidos	7	7

3.9 Resultados Generales

De acuerdo a la norma ISO 9241-11, se presenta una serie de pruebas para evaluar la eficacia y eficiencia con respecto a la usabilidad de la interfaz, además de otros aspectos.

Con respecto a la eficacia, se evaluaron los siguientes aspectos:

- *Concordancia entre los resultados reales y los esperados, resultados que pueden causar daños por uso.* Al incluirse secciones etiquetadas e identificadas en la interfaz, el usuario está en libertad de desplazarse por cada una, y tomar decisiones de acuerdo a su conveniencia.

- *Errores o dificultades de uso.* Al contar con un modo automático, el usuario puede controlar el sistema sin complicaciones. Además, al usar el modo manual, el usuario puede decidir, en base al semáforo indicador de humedad, cuando es el mejor momento para efectuar el riego.

- *Elementos innecesarios que interfieren con la tarea del usuario.* La interfaz permite la visualización de datos relevantes de forma clara. Además, cuenta con dos únicos botones, los cuales, son usados para iniciar o detener el proceso del sistema.

- *Elementos de salida inexactos o incompletos.* Durante las pruebas realizadas, no se encontró ninguna falla, pero, no se descarta en otros entornos donde exista problemas de conexión por poca señal Wi-Fi, debido a que el sistema actual depende de esta tecnología para funcionar correctamente.

- *Usuarios del sistema capaces de obtener todos los resultados esperados.* Los resultados esperados fueron un riego por aspersión centrado en el área verde, sin mojar por fuera de ella.

- *Resultados positivos inesperados.* No se esperaba poder detectar áreas de jardín más pequeñas con la detección de contornos, pero ahora es un beneficio del sistema.

- *Resultados individuales precisos sin completar todos los resultados.* Es posible capturar la imagen y tener un buen reconocimiento del área de césped, pero si esto, no se realiza, no podrá iniciar el riego.

- *Resultados completos con malinterpretación de campos.* Al utilizar el sistema en modo manual, se puede realizar el riego aun cuando no sea recomendable.

Para la eficiencia, se evaluaron los siguientes aspectos:

- *Producto o servicio del sistema eficiente con pocos recursos y eficaz en el logro de resultados*. En base a los recursos y elementos usados el sistema presenta un reconocimiento preciso y un funcionamiento eficiente.

- *Satisfecho independientemente de su eficiencia.* Al realizar todos los procesos correctamente y cumplir con el objetivo de solo regar el área verde, el sistema es eficiente aun a pesar del tiempo que demoren sus procesos.

- *Tiempo de uso, dedicado a alcanzar una meta*. En la Tabla 30, se encuentra el tiempo de los procesos en las diferentes pruebas, donde, se concluye que el tiempo de espera es adecuado.

- *Esfuerzo humano suministrado*. El esfuerzo físico para realizar el riego en un jardín de 8m x 9m para un adulto puede no ser mucho al no ser una demanda

excesiva, pero si demora aproximadamente 1 hora. Al realizarse por el sistema toma alrededor de 6 min, pero el usuario solo gastara 32,31 segundos en activar el sistema.

- *Materiales usados.* El mecanismo está fabricado en metal para garantizar una mayor durabilidad y resistencia. Además, para ambos servomotores, se utilizaron acoples de hierro. El recubrimiento, por su parte, fue elaborado en acrílico de 3 mm para asegurar su resistencia a las condiciones ambientales, evitando daños causados por la exposición al sol y la lluvia.

La evaluación de la usabilidad del sistema a través de la interfaz móvil, se realiza mediante el empleo del método SUS (System Usability Scale), de John Brooke. Este método, se compone de 10 declaraciones, diseñadas para evaluar la usabilidad de cualquier sistema. Cada declaración tiene respuestas que siguen la escala de Likert, de cinco opciones para cada pregunta. El Anexo K detalla la encuesta aplicada.

En la Figura 182, se presentan los resultados obtenidos de la encuesta aplicada a 16 personas que accedieron a evaluar la interfaz móvil.

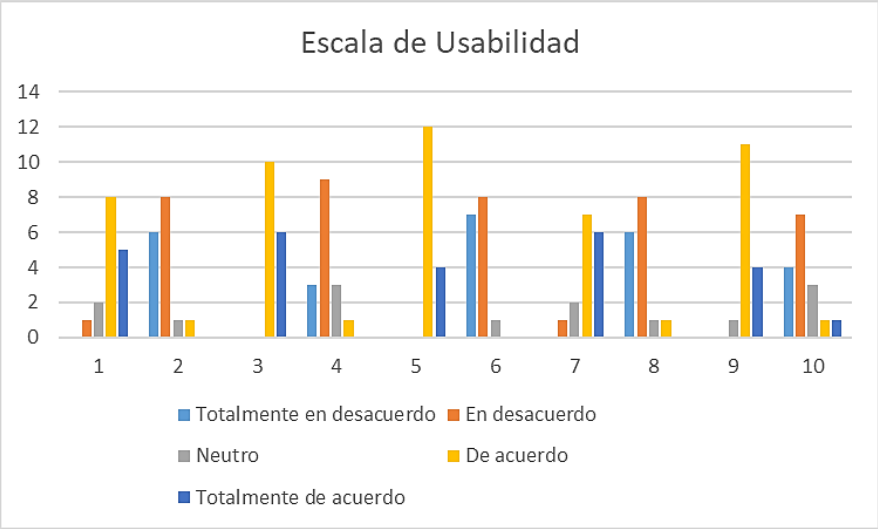


Figura 182. Diagrama de resultados de la encuesta de usabilidad del sistema

Con la encuesta realizada, se determina la puntuación individual para cada pregunta utilizando los datos recopilados. Es importante destacar que las preguntas 1, 3, 5, 7 y 9 son positivas, mientras que las preguntas 2, 4, 6, 8 y 10 son consideradas negativas.

Este enfoque, se sigue en el método SUS para prevenir posibles distorsiones en las respuestas. Los resultados de la encuesta de usabilidad, se presentan en la Tabla 34.

Tabla 34. Resultados de la encuesta de usabilidad del sistema

#	Escala de usabilidad	Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo
1	¿Usaría el sistema con frecuencia?	0	1	2	8	5
2	¿Encuentra el sistema innecesariamente complejo?	6	8	1	1	0
3	¿La interfaz es fácil de usar?	0	0	0	10	6
4	¿Necesitaría el apoyo de otra persona para utilizar la interfaz?	3	9	3	1	0
5	¿Las funciones de la interfaz estaban bien integradas?	0	0	0	12	4
6	¿Considera que hay inconsistencias en esta interfaz?	7	8	1	0	0
7	¿Encuentra útiles los gráficos proporcionados?	0	1	2	7	6
8	¿Encuentra innecesaria la información de la interfaz?	6	8	1	1	0
9	¿Aprendería a utilizar esta interfaz con rapidez?	0	0	1	11	4
10	¿Considera necesario conocer otros aspectos para utilizar la interfaz?	4	7	3	1	1

La evaluación de las respuestas a cada pregunta, se realiza utilizando la escala de Likert, que abarca valores del 1 al 5. El valor más alto en esta escala, que es 5, se asigna como el de mayor peso. En la Tabla 35, se presenta los resultados de la encuesta de usabilidad con la escala de Likert.

Tabla 35. Resultados de la encuesta de usabilidad del sistema con la escala de Likert

#	Escala de usabilidad	Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo
1	¿Usaría el sistema con frecuencia?	1	2	3	4	5
2	¿Encuentra el sistema innecesariamente complejo?	1	2	3	4	5
3	¿La interfaz es fácil de usar?	1	2	3	4	5
4	¿Necesitaría el apoyo de otra persona para utilizar la interfaz?	1	2	3	4	5
5	¿Las funciones de la interfaz estaban bien integradas?	1	2	3	4	5
6	¿Considera que hay inconsistencias en esta interfaz?	1	2	3	4	5
7	¿Encuentra útiles los gráficos proporcionados?	1	2	3	4	5
8	¿Encuentra innecesaria la información de la interfaz?	1	2	3	4	5

9	¿Aprendería a utilizar esta interfaz con rapidez?	1	2	3	4	5
10	¿Considera necesario conocer otros aspectos para utilizar la interfaz?	1	2	3	4	5

De acuerdo a la escala de Likert, se deben de sumar los valores para obtener el puntaje de usabilidad como, se muestra a continuación [78]:

- Sumar las respuestas de los enunciados impares y restar 5

$$(4 + 4 + 4 + 4 + 4) = 20 - 5 = 15$$

- Sumar las respuestas de los enunciados pares y restar el total a 25

$$(2 + 2 + 2 + 2 + 2) = 25 - 10 = 15$$

- Sumar ambos resultados y multiplicar por 2,5

$$(15 + 15) * 2.5 = 75$$

Al obtener un puntaje de 75 en la escala de SUS, de acuerdo con la Figura 183, se concluye que la usabilidad del sistema, se sitúa en la categoría B, calificada como Buena, Aceptable, y la puntuación neta del promotor (NPS) es Pasiva. Este indicador refleja la lealtad de los usuarios hacia el sistema, indicando que están satisfechos con la experiencia de realidad virtual, aunque sugiere la posibilidad de realizar mejoras.

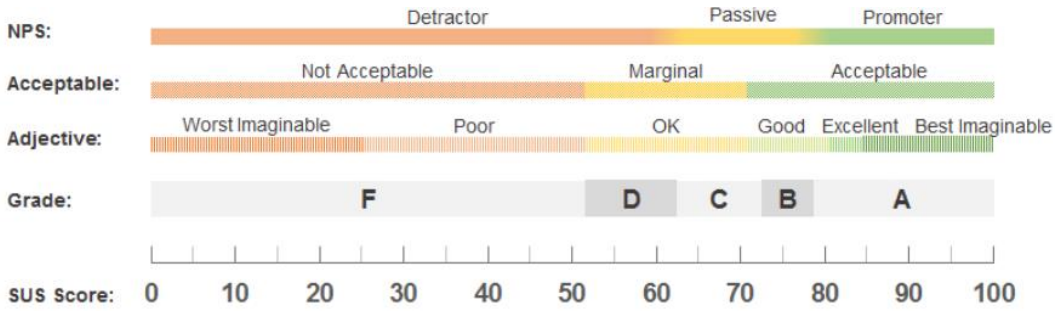


Figura 183. Escala de Usabilidad de un Sistema [78]

3.10 Presupuesto

El presupuesto total del proyecto denominado sistema de riego inteligente de corto alcance para jardines a partir de visión artificial, consta de dos partes: presupuesto de diseño y presupuesto de elaboración. En el presupuesto de diseño, se consideró las

horas de fabricación y el sueldo de un Ingeniero en Telecomunicaciones, de acuerdo a lo establecido en el Ministerio de Trabajo del Ecuador.

El salario diario, considerando 5 días laborables por semana en un total de 21 días laborales por mes; mediante estos datos, al aplicar la ecuación, se obtiene:

$$Salario_d = \frac{Salario_m}{Dias_{lab}} \qquad (3)$$

$$Salario_d = \frac{858}{21}[dólares]$$

$$Salario_d = 40{,}86[dólares]$$

En una jornada laboral son 8 horas diarias, aplicando la ecuación, se puede calcular la remuneración por hora.

$$Salario_h = \frac{Salario_d}{Horas_{lab}} \qquad (4)$$

$$Salario_h = \frac{40{,}86}{8}[dólares]$$

$$Salario_h = 5{,}11[dólares]$$

El costo total para el diseño del proyecto abarca 2 horas de trabajo diario durante 21 días por 4 meses, resultando en 168 horas. Aplicando la ecuación, se tiene:

$$Presupuesto_{diseño} = Horas_t * Salario_h \qquad (5)$$

$$Presupuesto_{diseño} = 168 * 5{,}11\ [dólares]$$

$$Presupuesto_{diseño} = 858{,}48\ [dólares]$$

En el presupuesto de elaboración detallado en la Tabla 36, se enlistan todos los elementos electrónicos y de conexión requeridos para el funcionamiento del sistema.

Tabla 36. Presupuesto del proyecto de investigación

Ítem	Detalle	Cantidad	Valor Unitario	Valor Total
1	ESP32 WROOM 32	1	$14	$14
2	ESP 32 CAM	1	$13	$13
3	Sensor de humedad de suelo fc-28	1	$3	$3
4	Servomotor TD-8120MG	2	$20	$20
5	Válvula solenoide	1	$12	$12
6	Módulo relé	1	$2,50	$2,50
7	Mecanismo metálico	1	$40	$40
8	Impresión PCB CNC	1	$10	$10
9	Case Mecanismo Acrílico	1	$20	$20
10	Case Cámara Acrílico	1	$8	$8
11	Fuente 5V 2ª	1	$10	$10
12	Fuente 12V 3ª	1	$16	$16
13	Manguera	1	$8	$8
14	Trípode	1	$6	$6
15	Internet	6	$30	$180
16	Metros de cables varios	12	$0,35	$4,20
Subtotal				$366,70
IVA 12%				$44
Total				$410,70

Con ambos presupuestos calculados, a continuación, se obtiene el presupuesto total del proyecto de investigación con un total de $1269,18.

$$Presupuesto = Presupuesto_{diseño} + Presupuesto_{elaboración} \quad (6)$$

$$Presupuesto = 858{,}48\ [dólares] + 410{,}70\ [dólares]$$

$$Presupuesto = 1269{,}18\ [dólares]$$

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Los sistemas de riego tradicionales para jardines usan métodos de riego por presurizado, aprovechan la presión del agua para efectuar el riego en zonas específicas, y son sistemas predominantemente manuales. Demandan la intervención del usuario para su posicionamiento, dirección y control. El sistema por goteo tiene una eficiencia de riego del 90% al 95% y adaptabilidad a terrenos irregulares, mientras que el sistema por aspersión es más adecuado para áreas extensas de césped, ofrece un costo inicial más bajo y una eficiencia de riego del 80% al 85%.

- Tras identificar los requisitos y funcionalidades necesarios para implementar un sistema de riego inteligente, se determinó los componentes electrónicos y software esenciales para su desarrollo, se consideró factores clave como la durabilidad, comunicación e integración con el sistema. En el proceso de adquisición de datos, para conocer el porcentaje de humedad del suelo, se optó por el sensor FC-28 a 15 cm de profundidad y la ESP32-CAM en modo cámara IP que captura el área de riego. En cuanto a los actuadores, se seleccionó una válvula solenoide normalmente cerrada y dos servomotores TD-8120MG con engranajes metálicos y torque de 20 kg para soportar la estructura metálica. El control centralizado, se confió a la ESP32, y una base de datos local. El sistema usa comunicación Wi-Fi para compartir, almacenar y gestionar la información.

- La aplicación del algoritmo de detección de superficie representa un avance significativo en el sistema de riego. El entrenamiento del modelo en YOLOv5 realiza una identificación precisa del 95,6% del área de césped, usando Python para la detección de contornos tanto del césped como de objetos circundantes, añaden un enfoque diferente a los sistemas inteligentes donde solo, se requería de sensores. El uso de imágenes para determinar el área de riego, mejora la distribución del riego al dirigirse específicamente a la superficie deseada, y refleja la importancia de la visión artificial para la gestión sostenible del agua.

- Se diseño una aplicación móvil en Android Studio para monitorear el sistema de riego inteligente, a través de una interfaz intuitiva y funcional. La integración con la base de datos permite el control y monitoreo del sistema de riego con un promedio de 32,31 segundos en respuesta de la interfaz, y en 1 minuto de espera aproximadamente los usuarios pueden visualizar el inicio del riego. Este enfoque en conjunto con los requerimientos establecidos por la norma ISO 9241-11 muestran una integración fluida entre la interfaz y el riego. La aplicación de la escala de usabilidad SUS, arrojo como resultado un valor de 75, de acuerdo a este método la usabilidad está dentro de un rango aceptable.

- En conclusión, la evaluación del desempeño del sistema de riego inteligente a través de pruebas de funcionamiento ha sido crucial para entender la utilidad del sistema en condiciones reales, demostró que detecta adecuadamente el área de césped y objetos, y de igual manera realiza un riego adecuado respetando los límites establecidos por los contornos generados. Además, se demostró un consumo de agua adecuado del sistema del 85,36%. Estas evaluaciones no solo validan la funcionalidad del sistema, sino que también sientan las bases para futuras mejoras y desarrollos en aplicaciones prácticas de riego.

4.2 Recomendaciones

- Al iniciar el riego no usar el jardín, en el aspecto de que no haya personas o animales cruzando, esto podría provocar una mala detección y un riego inadecuado.

- Se recomienda el uso de servomotores con un torque de 20 kg o superior, especialmente para el cuerpo inferior, ya que debe tener la capacidad de mover la parte superior, y de tolerar la fuerza generada por el flujo de agua proveniente de la manguera.

- Se sugiere utilizar el sistema en la mañana, a las 8 a. m., para regar sin presencia de sol fuerte, obtener una captura de imagen óptima, evitar el exceso de brillo y sombras que podrían dificultar la detección del césped.

- Evite regar cuando la velocidad del viento sea superior a los 10 km/h, ya que las diminutas gotas del aspersor podrían desviarse en una dirección no deseada, resultando en un riego ineficiente en el 50%.

- Se recomienda identificar los elementos dentro de la interfaz y organizar por prioridad las zonas y los indicadores que requieren mayor atención para los usuarios para lograr un riego adecuado.

- Para un monitoreo a distancia, se recomienda utilizar servidores web como Azure. Esto permitirá acceder al sistema y llevar a cabo el riego incluso cuando, se esté lejos del jardín.

- El sistema desarrollado sirve como punto de partida para futuras investigaciones, se recomienda optimizar el sistema con mejores sensores para el estado del suelo, una cámara de mayor resolución para mejorar la detección de obstáculos, especialmente en situaciones más complejas como en condiciones nocturnas. Además, se sugiere explorar opciones para la detección de contornos o similares, lo que ayuda al sistema a adaptarse a diferentes situaciones en las que, se requiera detectar el jardín como con movimiento de mascotas o personas. De igual forma implementar bombas de agua para brindar mayor control del alcance del riego. Estas mejoras podrían permitir aumentar el desempeño del sistema a otras áreas como la agricultura.

REFERENCIAS BIBLIOGRÁFICAS

[1] Food and Agriculture Organization of the United Nations, "Escasez de agua: Uno de los mayores retos de nuestro tiempo." Accessed: Sep. 26, 2023. [Online]. Available: https://www.fao.org/fao-stories/article/es/c/1185408/

[2] R. J. Monje Jiménez, *Jardinería y Floricultura: Manejo de céspedes con bajo consumo de agua*, Segunda. Sevilla: Viceconsejería. Servicio de Publicaciones y Divulgación, 2006.

[3] M. A. Martínez Gimeno, "ESTRATEGIA PARA LA EVALUACIÓN AGRONÓMICA, HIDRÁULICA Y ENERGÉTICA EN JARDINERÍA. APLICACIÓN A LA GESTIÓN DE JARDINES PÚBLICOS," Universidad Politécnica de Valencia, Valencia, 2014. Accessed: Sep. 26, 2023. [Online]. Available: http://hdl.handle.net/10251/53389

[4] D. S. Fernández Reynoso, M. R. Martínez Menes, C. A. Tavarez Espinosa, R. Castillo Vega, and R. Salas Martínez, *ESTIMACIÓN DE LAS DEMANDAS DE CONSUMO DE AGUA*. México: Colegio de Postgraduados, 2017. Accessed: Sep. 26, 2023. [Online]. Available: https://www.academia.edu/32110228/

[5] W. J. Fisk, Q. Lei-Gomez, and M. J. Mendell, "Meta-analyses of the associations of respiratory health effects with dampness and mold in homes," *Indoor Air*, vol. 17, no. 4, pp. 284–296, 2007, doi: 10.1111/j.1600-0668.2007.00475.x.

[6] G. D. C. Rendón Sustaita, J. G. Cortes Torres, D. I. Juárez Pedraza, and M. J. Ortega Ibarra, "Sistema de riego inteligente utilizando electroválvulas a partir de sensores de visión," *Pistas Educativas*, no. 113, pp. 293–307, Oct. 2015, Accessed: Sep. 26, 2023. [Online]. Available: https://core.ac.uk/download/pdf/229040668.pdf

[7] C. L. Chang and K. M. Lin, "Smart agricultural machine with a computer vision-based weeding and variable-rate irrigation scheme," *Robotics*, vol. 7, no. 3, Jul. 2018, doi: 10.3390/robotics7030038.

[8] M. Nadafzadeh and S. Abdanan Mehdizadeh, "Design and fabrication of an intelligent control system for determination of watering time for turfgrass plant using computer vision system and artificial neural network," *Precis Agric*, vol. 20, no. 5, pp. 857–879, Oct. 2019, doi: 10.1007/s11119-018-9618-x.

[9] T. E. Gualpa Núñez, "Gestión de sistema de riego inteligente para el cuidado del parque Palomino Flores de la Ciudad de Baños de Agua Santa," Universidad Técnica de Ambato, Ambato, 2020. Accessed: Sep. 26, 2023. [Online]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/31984

[10] I. A. Calapaqui Guamaní and A. P. Chiguano Rojas, "Desarrollo de un sistema inteligente para el cultivo de claveles," Universidad Politécnica Salesiana, Quito, 2022. Accessed: Sep. 26, 2023. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/22941

[11] A. F. Acosta Tenelema, "Sistema hidropónico inteligente aplicado a la producción del forraje verde con arquitectura IoT," Universidad Técnica de Ambato, Ambato, 2022. Accessed: Sep. 26, 2023. [Online]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/36511

[12] F. J. Pérez Reyes, “Sistema automatizado de control y monitoreo basado en tecnología Lorawan y MQTT para el cultivo de hortalizas bajo invernadero,” Universidad Técnica de Ambato, Ambato, 2023. Accessed: Sep. 26, 2023. [Online]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/37738

[13] R. León, N. Bonifaz, and F. Gutiérrez, *Pastos y forrajes del Ecuador: siembra y producción de pasturas*, 1st ed. Quito: Editorial Universitaria Abya-Yala, Universidad Politécnica Salesiana, 2018.

[14] M. Lenz, “LA IMPORTANCIA DEL SISTEMA DE RIEGO,” Marcia Lenz Paisajismo. Accessed: Sep. 26, 2023. [Online]. Available: https://www.marcialenz.com/blog/la-importancia-del-sistema-de-riego

[15] A. Floríndez Díaz, Gobierno Regional de Cajamarca, Instituto Cuencas, and Programa Desarrollo Rural Sostenible – PDRS, *Sistemas de riego predial regulados por microrreservorios: cosecha de agua y producción segura. Manual técnico*, 1st ed. Lima: Giacomotti Comunicación Gráfica SAC, 2011. Accessed: Sep. 26, 2023. [Online]. Available: https://idmaperu.org/wp-content/uploads/2023/03/MANUAL-RESERVORIOS-CUENCAS-RURANDES.pdf

[16] R. Fernández Gómez, R. Ávila Alabarces, M. López Rodríguez, P. Gavilán Zafra, and N. A. Oyonarte Gutiérrez, *Manual de riego para agricultores: módulo 1. Fundamentos del riego: manual y ejercicios*, vol. 1. Sevilla: Secretaría General Técnica, Servicio de Publicaciones y Divulgación, 2010.

[17] F. Gil-Albert Velarde, *Manual Técnico de Jardinería. Establecimiento y Mantenimiento*. España: Ediciones Mundi-Prensa, 2019.

[18] Oficina Regional de la FAO para América Latina y el Caribe, *EL DESARROLLO DEL MICRORRIEGO EN AMÉRICA CENTRAL. Oportunidades, Limitaciones y Desafíos*. 2008. Accessed: Sep. 26, 2023. [Online]. Available: https://agua.org.mx/wp-content/uploads/2017/10/El-desarrollo-del-microrriego-en-América-Central-Oportunidades-limitaciones-y-desafíos.pdf

[19] S. Delgado Martorell, “Las claves del riego por exudación que debes conocer,” PRISMAB. Accessed: Sep. 26, 2023. [Online]. Available: https://prismab.com/blog/las-claves-del-riego-por-exudacion-que-debes-conocer/

[20] J. A. Laverde Mena, “Sistema automatizado de riesgo por aspersión para el jardín unicado en la parte lateral del bloque de aulas #2 de Uniandes Quevedo,” Universidad Regional Autónoma de los Andes, Quevedo, 2016. Accessed: Sep. 26, 2023. [Online]. Available: https://dspace.uniandes.edu.ec/handle/123456789/4642

[21] D. A. Saltos Salazar, “El agua de riego y su incidencia en la producción agrícola de un terreno en la parroquia Santa Rosa de la ciudad de Ambato, provincia de Tungurahua,” Universidad Técnica de Ambato, Ambato, 2011. Accessed: Sep. 26, 2023. [Online]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/1595

[22] J. L. Velasco, “El riego correcto,” Interempresas, Mundo Jardín, Sistemas de riego para nuestro jardín. Accessed: Sep. 26, 2023. [Online]. Available: https://www.interempresas.net/Ferreteria/Articulos/185220-El-riego-correcto.html

[23] G. D. Yásig Cuichán, “Implementación de un sistema automatizado de riego por aspersión utilizando Arduino para las áreas verdes y jardines del barrio ‘Valle

Hermoso' ubicado en el cantón Mejía," Universidad de las Fuerzas Armadas ESPE, Latacunga, 2021. Accessed: Sep. 26, 2023. [Online]. Available: http://repositorio.espe.edu.ec/handle/21000/25989

[24] MAHER Electrónica, "Sistema de Riego Automático: en qué consiste y cuáles son sus ventajas," MAHER Smart Agrocontrollers. Accessed: Sep. 26, 2023. [Online]. Available: https://www.maherelectronica.com/sistema-riego-automatico/

[25] H. Barría, "Elementos de un sistema de riego automático básico," *Osorno: Informativo INIA Remehue*, no. 240, 2020.

[26] I. A. Calapaqui Guamaní and A. P. Chiguano Rojas, "Desarrollo de un sistema inteligente para el cultivo de claveles," Universidad Politécnica Salesiana, Quito, 2022. Accessed: Sep. 26, 2023. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/22941

[27] Agroingenia Canarias, "¿Qué es el riego inteligente?," Agroingenia Canarias. Accessed: Sep. 26, 2023. [Online]. Available: https://agroingeniacanarias.com/que-es-el-riego-inteligente/

[28] BASCOMEX and A. Elizondo, "¿Qué es un sistema de riego inteligente?," BASCOMEX. Accessed: Sep. 26, 2023. [Online]. Available: https://bascomex.com/blogs/news/que-es-un-sistema-de-riego-inteligente

[29] MASACA, "INSTALACIÓN Y MANTENIMIENTO DE SISTEMAS DE RIEGO PARA JARDINES Y HUERTOS - TODO LO QUE NECESITAS SABER," Masaca.cr. Accessed: Jan. 03, 2024. [Online]. Available: https://masaca.cr/instalacion-y-mantenimiento-de-sistemas-de-riego-para-jardines/

[30] Novagric, "Riego en Areas Verdes," Servicios de riego. Accessed: Oct. 04, 2023. [Online]. Available: https://www.novagric.com/es/riego/servicios/riego-areas-verdes

[31] Eden, "Eden 95124 Mini aspersor oscilante turbo ajustable de 4 vías con juego de arranque de conexión rápida, cubre hasta 4,069 pies cuadrados base de peso pesado," Amazon. Patio, Césped y Jardín. Accessed: Sep. 30, 2023. [Online]. Available: https://www.amazon.com/95124-oscilante-ajustable-iniciación-cuadrados/dp/B09NP74D4V

[32] CHICIRIS, "CHICIRIS Aspersor de impacto, cabezal de rociador ajustable G1/2, aleación de zinc resistente, rociador de riego de 360 grados con rango de pulverización de 49 pies," Amazon. Patio, Césped y Jardín. Accessed: Sep. 30, 2023. [Online]. Available: https://www.amazon.com/-/es/dp/B0BC2CXQLP/

[33] Melnor Inc, "Eden - Juego de aspersor oscilante de metal," Amazon. Patio, Césped y Jardín. Accessed: Sep. 30, 2023. [Online]. Available: https://www.amazon.com/-/es/dp/B0883S51HR/

[34] Honest Brands, "Rociadores para patio, paquete de 2 | Rociadores de agua resistentes para césped | Aspersor de jardín para patio Sistema de riego de rociadores rotativos de 360 grados | Gran área de cobertura | Conexión rápida a prueba de fugas," Amazon. Patio, Césped y Jardín. Accessed: Sep. 30, 2023. [Online]. Available: https://www.amazon.com/-/es/dp/B08GSSSWYP/

[35] A. Hassan *et al.*, “A wirelessly controlled robot-based smart irrigation system by exploiting arduino,” *Journal of Robotics and Control (JRC)*, vol. 2, no. 1, pp. 29–34, Jan. 2021, doi: 10.18196/jrc.2148.

[36] M. Moraitis, K. Vaiopoulos, and A. T. Balafoutis, “Design and Implementation of an Urban Farming Robot,” *Micromachines (Basel)*, vol. 13, no. 2, Feb. 2022, doi: 10.3390/mi13020250.

[37] Novagric, “Riego por Aspersión,” Sistemas de riego. Accessed: Oct. 04, 2023. [Online]. Available: https://www.novagric.com/es/riego/sistemas-de-riego/riego-por-aspersion

[38] Novagric, “Diseño de Riego,” Servicios de riego. Accessed: Oct. 04, 2023. [Online]. Available: https://www.novagric.com/es/riego/servicios/diseno-de-riego

[39] J. Cruz Cárdenas and N. H. Oleas, “Private Urban Garden Satisfaction and Its Determinants in Quito, Ecuador,” *Sage Open*, vol. 8, no. 1, Mar. 2018, doi: 10.1177/2158244018767242.

[40] INEC and S. Calderón, “Índice Verde Urbano 2012,” 2012. Accessed: Oct. 04, 2023. [Online]. Available: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Encuestas_Ambientales/Verde_Urbano/Presentacion_Indice%20Verde%20Urbano%20-%202012.pdf

[41] G. A. Aguirre Villegas, “LAS CONSTRUCCIONES REALIZADAS EN EL CANTÓN AMBATO Y LAS SANCIONES ADMINISTRATIVAS IMPUESTAS POR EL GOBIERNO AUTÓNOMO DESCENTRALIZADO MUNICIPAL DE AMBATO,” Universidad Técnica de Ambato, Ambato, 2014. Accessed: Oct. 04, 2023. [Online]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/8288

[42] J. M. Guamán Pozo, “Índices de cambio climático y su afectación a la agricultura, caso de estudio cantón Ambato,” Universidad Politécnica Salesiana, Quito, 2020. Accessed: Oct. 04, 2023. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/19187

[43] Weather Spark, “Datos históricos meteorológicos de 2023 en Ambato,” Weatherspark.com. Accessed: Oct. 04, 2023. [Online]. Available: https://es.weatherspark.com/h/y/20027/2023/Datos-históricos-meteorológicos-de-2023-en-Ambato-Ecuador#Figures-Summary

[44] CONGOPE and V. H. Cadena Navarro, *Hablemos de riego*, 1st ed. Ibarra: El Telégrafo EP, 2014.

[45] INTAGRI S. C., “Uso de Sensores de Humedad para Definir Riego,” Dec. 2015. Accessed: Oct. 04, 2023. [Online]. Available: https://www.intagri.com/articulos/agua-riego/uso-de-sensores-de-humedad-para-definir-riego?p=registro

[46] M. Renom, “HIGROMETRIA,” Uruguay, Oct. 2011. Accessed: Oct. 04, 2023. [Online]. Available: http://www.meteorologia.edu.uy/wp-content/uploads/2019/principios_bas_medicones_atms/Bolilla4.pdf

[47] A. Dominguez Torres, *Procesamiento digital de imagenes*. Red Perfiles Educativos, 2006. Accessed: Oct. 04, 2023. [Online]. Available: https://elibro.net/es/lc/uta/titulos/6882

[48] J. P. Alvarado Moya, “Procesamiento y Análisis de Imágenes Digitales,” May 2012. Accessed: Oct. 04, 2023. [Online]. Available: https://www.tec.ac.cr/sites/default/files/media/doc/paid.pdf

[49] B. Borrella Petisco, “Introducción a la visión artificial: procesos y aplicaciones,” Universidad Complutense de Madrid, Madrid, 2022. Accessed: Oct. 04, 2023. [Online]. Available: https://docta.ucm.es/entities/publication/072ca3fb-540f-4dc9-8a16-0241cb5cd986

[50] A. Gonzáles Marcos *et al.*, *Técnicas y algoritmos básicos de visión artificial*, 1st ed. Logroño: Universidad de la Rioja, 2006. Accessed: Oct. 04, 2023. [Online]. Available: https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf.

[51] T. Do, “Evolution of YOLO Algorithm and YOLOv5: The State-of-the-art Object Detection Algorithm,” Oulu University of Applied Sciences, 2021. Accessed: Oct. 04, 2023. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/452552/Do_Thuan.pdf

[52] E. Mavridou, E. Vrochidou, G. A. Papakostas, T. Pachidis, and V. G. Kaburlasos, “Machine vision systems in precision agriculture for crop farming,” *J Imaging*, vol. 5, no. 12, 2019, doi: 10.3390/jimaging5120089.

[53] Naylamp Mechatronics, “Sensor de Humedad de Suelo Capacitivo v1.2,” Naylamp Mechatronics.com. Accessed: Oct. 14, 2023. [Online]. Available: https://naylampmechatronics.com/sensores-temperatura-y-humedad/538-sensor-de-humedad-de-suelo-capacitivo-v1.html

[54] Grupoelectrostore, “MÓDULO SENSOR DE HUMEDAD Y TEMPERATURA DE SUELO FC-28 HIGRÓMETRO,” Grupoelectrostore.com. Accessed: Oct. 14, 2023. [Online]. Available: https://grupoelectrostore.com/shop/sensores/temperatura/modulo-sensor-de-humedad-y-temperatura-de-suelo-fc-28-higrometro/

[55] Grupoelectrostore, “MÓDULO SENSOR DE HUMEDAD DE SUELO HD-38 HIGRÓMETRO ANTICORROSIVO,” Grupoelectrostore.com. Accessed: Oct. 14, 2023. [Online]. Available: https://grupoelectrostore.com/shop/sensores/temperatura/modulo-sensor-de-humedad-de-suelo-hd-38-higrometro-anticorrosivo/

[56] Espressif Systems, “ESP32WROOM32,” Feb. 2023. Accessed: Oct. 14, 2023. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[57] Espressif, “ESP8266EX,” Jun. 2023. Accessed: Oct. 14, 2023. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[58] Arduino, “Arduino UNO R3,” Dec. 2023. Accessed: Dec. 24, 2023. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[59] OmniVision, “OV2640 Color CMOS UXGA (2 MegaPixel) CAMERA CHIP,” Jul. 2006. Accessed: Oct. 14, 2023. [Online]. Available: https://www.uctronics.com/download/cam_module/OV2640DS.pdf

[60] Systemarket, “CAMARA WEB GENIUS FACECAM 1000X V2 BLACK,” systemarket.com.ec. Accessed: Oct. 14, 2023. [Online]. Available: https://systemarket.com.ec/tienda/accesorios/varios/camara-web-genius-1000x-hd-720p-usb-black/

[61] Digikey, “ESP32-CAM Development Board,” Aug. 2019. Accessed: Oct. 14, 2023. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf

[62] Amazon, “TD-8120MG - Engranaje de metal de gran par de torsión digital resistente al agua, 44.1 lbs (180 grados),” Amazon.com. Accessed: Oct. 14, 2023. [Online]. Available: https://www.amazon.com/-/es/TD-8120MG-Engranaje-torsión-resistente-44-1-lbs/dp/B09NKT7FZQ

[63] Grupoelectrostore, “SERVOMOTOR TOWER PRO MG995 11 KG.CM STANDARD 360°,” Grupoelectrostore.com. Accessed: Oct. 14, 2023. [Online]. Available: https://grupoelectrostore.com/shop/motores/servomotores/servomotor-tower-pro-mg995-11-kg-cm-standard-360/

[64] Grupoelectrostore, “SERVOMOTOR TOWER PRO MG996R 13 KG.CM STANDARD 180°,” Grupoelectrostore.com. Accessed: Oct. 14, 2023. [Online]. Available: https://grupoelectrostore.com/shop/motores/servomotores/servomotor-tower-pro-mg996r-13-kg-cm-standard-180/

[65] Grupoelectrostore, “VÁLVULA SOLENOIDE ELECTROMAGNÉTICA 12VDC 1/2PLG NC PARA AGUA,” Grupoelectrostore.com. Accessed: Oct. 14, 2023. [Online]. Available: https://grupoelectrostore.com/shop/valvulas/valvula-solenoide-electromagnetica-12vdc-1-2plg-nc-para-agua/

[66] Grupoelectrostore, “VÁLVULA SOLENOIDE ELECTROMAGNÉTICA 12VDC 3/4PLG NC PARA AGUA,” Grupoelectrostore.com. Accessed: Oct. 14, 2023. [Online]. Available: https://grupoelectrostore.com/shop/valvulas/valvula-solenoide-electromagnetica-12vdc-3-4plg-nc-para-agua/

[67] Grupoelectrostore, “MÓDULO RELÉ 5V KY-019 RELÉ 5V 1 CANAL,” Grupoelectrostore.com. Accessed: Oct. 14, 2023. [Online]. Available: https://grupoelectrostore.com/shop/modulos-y-shields/modulos-rele/modulo-rele-5v-ky-019-rele-5v-1-canal/

[68] Grupoelectrostore, “MÓDULO RELÉ 5V 1 CANAL CON OPTOACOPLADOR HIGH/LOW ALTO/BAJO,” Grupoelectrostore.com. Accessed: Oct. 14, 2023. [Online]. Available: https://grupoelectrostore.com/shop/modulos-y-shields/modulos-rele/modulo-rele-5v-1-canal-con-optoacoplador-high-low-alto-bajo/

[69] Vegavid Technology, “Bluetooth vs. Wi-Fi for IoT: Which is better?,” Vegavid.com. Accessed: Oct. 17, 2023. [Online]. Available: https://vegavid.com/blog/bluetooth-vs-wifi-which-is-better-for-iot/#

[70] A. Khan and A. Kabir, “Comparison among Bluetooth,WI-Fi and Zig bee,” Dec. 2010. Accessed: Oct. 17, 2023. [Online]. Available: https://www.academia.edu/5889265/Comparison_among_Bluetooth_Wi-Fi_and_Zig_Bee

[71] L. Carrero, “10 sistemas de administración de bases de datos populares (DBMS),” Stackscale. Accessed: Oct. 17, 2023. [Online]. Available: https://www.stackscale.com/es/blog/sistemas-administracion-bases-datos-populares/

[72] F. García de Zúñiga, “Los sistemas gestores de bases de datos (SGBD) de código abierto más utilizados,” arsys.es. Accessed: Oct. 17, 2023. [Online]. Available: https://www.arsys.es/blog/comparativa-motores-basesdatos

[73] Blushbreak, “Los entornos de desarrollo local más usados en 2023【 XAMPP, MAMP, WAMP y Local by Flywheel】,” Blushbreak. Accessed: Oct. 17, 2023. [Online]. Available: https://blushbreak.com/entornos-de-desarrollo-local/

[74] S. Das, “Flutter vs Android Studio: Core Differences,” BrowserStack. Accessed: Oct. 17, 2023. [Online]. Available: https://www.browserstack.com/guide/flutter-vs-android-studio

[75] O. Goncharenko, “Flutter vs. React Native - detailed framework comparison,” Brocoders. Accessed: Oct. 17, 2023. [Online]. Available: https://brocoders.com/blog/flutter-vs-react-native/

[76] Ultralytics, “Consejos para obtener los mejores resultados de entrenamiento,” Ultralytics.com. Accessed: Nov. 14, 2023. [Online]. Available: https://docs.ultralytics.com/es/yolov5/tutorials/tips_for_best_training_results/

[77] S. Susan and A. Kumar, “The balancing trick: Optimized sampling of imbalanced datasets—A brief survey of the recent State of the Art,” *Engineering Reports*, vol. 3, no. 4. John Wiley and Sons Inc, Apr. 01, 2021. doi: 10.1002/eng2.12298.

[78] J. Brooke, “A quick and dirty usability scale,” 1986.

ANEXOS

Anexo A. Datasheet de la ESP32

En la Figura A1, Figura A2, y Figura A3, se muestra el datasheet de la ESP 32

Features

Wi-Fi

- 802.11b/g/n
- 802.11n (2.4 GHz), up to 150 Mbps
- WMM
- TX/RX A-MPDU, RX A-MSDU
- Immediate Block ACK
- Defragmentation
- Automatic Beacon monitoring (hardware TSF)
- 4 × virtual Wi-Fi interfaces
- Simultaneous support for Infrastructure Station, SoftAP, and Promiscuous modes
  Note that when ESP32 is in Station mode, performing a scan, the SoftAP channel will be changed.
- Antenna diversity

Bluetooth®

- Compliant with Bluetooth v4.2 BR/EDR and Bluetooth LE specifications
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control
- +9 dBm transmitting power
- NZIF receiver with –94 dBm Bluetooth LE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR and Bluetooth LE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic Bluetooth and Bluetooth LE
- Simultaneous advertising and scanning

CPU and Memory

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s)
- CoreMark® score:
  - 1 core at 240 MHz: 504.85 CoreMark; 2.10 CoreMark/MHz

Figura A1. Parte uno de características técnicas de la ESP 32

  - 2 cores at 240 MHz: 994.26 CoreMark; 4.14 CoreMark/MHz
- 448 KB ROM
- 520 KB SRAM
- 16 KB SRAM in RTC
- QSPI supports multiple flash/SRAM chips

Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/Bluetooth functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
- One RTC timer
- RTC watchdog

Advanced Peripheral Interfaces

- 34 × programmable GPIOs
  - 5 strapping GPIOs
  - 6 input-only GPIOs
  - 6 GPIOs needed for in-package flash/PSRAM (ESP32-D0WDR2-V3, ESP32-U4WDH)
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I2S
- 2 × I2C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- TWAI®, compatible with ISO 11898-1 (CAN Specification 2.0)
- RMT (TX/RX)
- Motor PWM
- LED PWM up to 16 channels

Figura A2. Parte dos de características técnicas de la ESP 32

Power Management

- Fine-resolution power control through a selection of clock frequency, duty cycle, Wi-Fi operating modes, and individual power control of internal components
- Five power modes designed for typical scenarios: Active, Modem-sleep, Light-sleep, Deep-sleep, Hibernation
- Power consumption in Deep-sleep mode is 10 μA
- Ultra-Low-Power (ULP) coprocessors
- RTC memory remains powered on in Deep-sleep mode

Security

- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration:
  - AES
  - Hash (SHA-2)
  - RSA
  - ECC
  - Random Number Generator (RNG)

Applications

With low power consumption, ESP32 is an ideal choice for IoT devices in the following areas:

- Smart Home
- Industrial Automation
- Health Care
- Consumer Electronics
- Smart Agriculture
- POS machines
- Service robot
- Audio Devices
- Generic Low-power IoT Sensor Hubs
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- Speech Recognition
- Image Recognition
- SDIO Wi-Fi + Bluetooth Networking Card
- Touch and Proximity Sensing

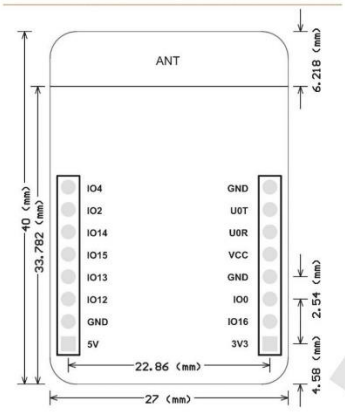
Figura A3. Parte tres de características técnicas de la ESP 32

En la Figura B1, Figura B2, Figura B3, y Figura B4, se muestra el datasheet de la ESP32-CAM

安信可科技 Ai-Thinker

ESP32-CAM Wi-Fi+BT SoC Module V1.0

ESP32-CAM Module



Features

- The smallest 802.11b/g/n Wi-Fi BT SoC Module
- Low power 32-bit CPU,can also serve the application processor
- Up to 160MHz clock speed，Summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras,Built-in Flash lamp.
- Support image WiFI upload
- Support TF card
- Supports multiple sleep modes.
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

Overview

The ESP32-CAM has a very competitive small-size camera module that can operate independently as a minimum system with a footprint of only 27*40.5*4.5mm and a deep sleep current of up to 6mA.

ESP-32CAM can be widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals and other IoT applications. It is an ideal solution for IoT applications.

ESP-32CAM adopts DIP package and can be directly inserted into the backplane to realize rapid production of products, providing customers with high-reliability connection mode, which is convenient for application in various IoT hardware terminals.

Figura B1. Parte uno de características técnicas de la ESP32-CAM

Product Specifications

Module Model	ESP32-CAM
Package	DIP-16
Size	27*40.5*4.5（±0.2）mm
SPI Flash	Default 32Mbit
RAM	520KB SRAM +4M PSRAM
Bluetooth	Bluetooth 4.2 BR/EDR and BLE standards
Wi-Fi	802.11 b/g/n/
Support interface	UART、SPI、I2C、PWM
Support TF card	Maximum support 4G
IO port	9
UART Baudrate	Default 115200 bps
Image Output Format	JPEG(OV2640 support only),BMP,GRAYSCALE
Spectrum Range	2412 ~2484MHz
Antenna	Onboard PCB antenna, gain 2dBi
Transmit Power	802.11b: 17±2 dBm (@11Mbps) 802.11g: 14±2 dBm (@54Mbps) 802.11n: 13±2 dBm (@MCS7)
Receiving Sensitivity	CCK, 1 Mbps : -90dBm CCK, 11 Mbps: -85dBm 6 Mbps (1/2 BPSK): -88dBm 54 Mbps (3/4 64-QAM): -70dBm MCS7 (65 Mbps, 72.2 Mbps): -67dBm
Power Dissipation	Turn off the flash lamp:180mA@5V Turn on the flash lamp and turn on the brightness to the maximum:310mA@5V Deep-sleep: Minimum power consumption can be achieved 6mA@5V Moderm-sleep: Minimum up to 20mA@5V Light-sleep: Minimum up to 6.7mA@5V
Security	WPA/WPA2/WPA2-Enterprise/WPS
Power Supply Range	5V
Operating Temperature	-20 ℃ ~ 85 ℃
Storage Environment	-40 ℃ ~ 90 ℃ , < 90%RH

Figura B2. Parte dos de características técnicas de la ESP32-CAM

Weight	10g

ESP32-CAM module picture output format rate

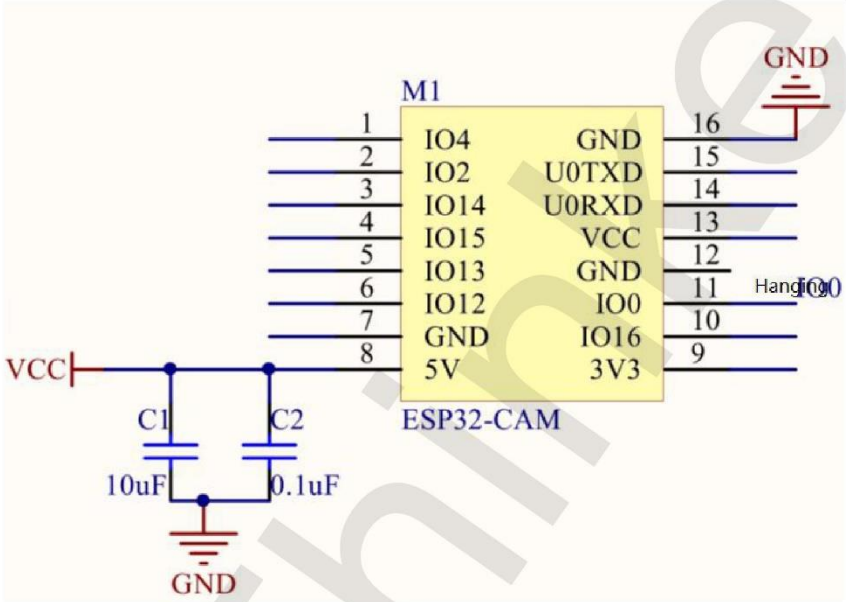
Format Size	QQVGA	QVGA	VGA	SVGA
JPEG	6	7	7	8
BMP	9	9	-	-
GRAYSCALE	9	8	-	-

Internal Pin Connect

CAM	ESP32	SD	ESP32
D0	PIN5	CLK	PIN14
D1	PIN18	CMD	PIN15
D2	PIN19	DATA0	PIN2
D3	PIN21	DATA1/Flash lamp	PIN4
D4	PIN36	DATA2	PIN12
D5	PIN39	DATA3	PIN13
D6	PIN34		
D7	PIN35		
XCLK	PIN0		
PCLK	PIN22		
VSYNC	PIN25		
HREF	PIN23		
SDA	PIN26		
SCL	PIN27		
POWER PIN	PIN32		

Figura B3. Parte tres de características técnicas de la ESP32-CAM

Minimum system diagram



Contact US

Shenzhen Ai-Thinker Technology Co., Ltd

Address: 7/F, Fengze Building B, Huafeng Industrial Park 2th, Hangkong street,Xixiang Raod, Baoan, Shenzhen China

Website:www.ai-thinker.com **Tel：0755-29162996** **E-mail:support@aithinker.com**

Figura B4. Parte cuatro de características técnicas de la ESP32-CAM

El sensor de humedad, se compone del FC-28 que son los electrodos y el circuito acondicionador YL-38 formado por el circuito integrado LM393. En la Figura C1, se muestra las características del circuito integrado LM393. En la Figura C2, se muestra el esquema de conexión del circuito YL-38.

TEXAS INSTRUMENTS

LM193-N, LM2903-N, LM293-N, LM393-N

SNOSBJ6G – OCTOBER 1999 – REVISED OCTOBER 2018

LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators

1 Features

- Wide Supply
  - Voltage Range: 2.0 V to 36 V
  - Single or Dual Supplies: ±1.0 V to ±18 V
- Very Low Supply Current Drain (0.4 mA) — Independent of Supply Voltage
- Low Input Biasing Current: 25 nA
- Low Input Offset Current: ±5 nA
- Maximum Offset voltage: ±3 mV
- Input Common-Mode Voltage Range Includes Ground
- Differential Input Voltage Range Equal to the Power Supply Voltage
- Low Output Saturation Voltage: 250 mV at 4 mA
- Output Voltage Compatible with TTL, DTL, ECL, MOS and CMOS logic systems
- Available in the 8-Bump (12 mil) DSBGA Package
- See AN-1112 (SNVA009) for DSBGA Considerations
- Advantages
  - High Precision Comparators
  - Reduced V_{OS} Drift Over Temperature
  - Eliminates Need for Dual Supplies
  - Allows Sensing Near Ground
  - Compatible with All Forms of Logic
  - Power Drain Suitable for Battery Operation

2 Applications

- Battery Powered Applications
- Industrial Applications

3 Description

The LM193-N series consists of two independent precision voltage comparators with an offset voltage specification as low as 2.0 mV max for two comparators which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage. These comparators also have a unique characteristic in that the input common-mode voltage range includes ground, even though operated from a single power supply voltage.

Application areas include limit comparators, simple analog to digital converters; pulse, squarewave and time delay generators; wide range VCO; MOS clock timers; multivibrators and high voltage digital logic gates. The LM193-N series was designed to directly interface with TTL and CMOS. When operated from both plus and minus power supplies, the LM19-N series will directly interface with MOS logic where their low power drain is a distinct advantage over standard comparators.

The LM393 and LM2903 parts are available in TI's innovative thin DSBGA package with 8 (12 mil) large bumps.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM193-N	TO-99 (8)	9.08 mm x 9.08 mm
LM293-N		
LM393-N	SOIC (8)	4.90 mm x 3.91 mm
	DSBGA (8)	1.54 mm x 1.54 mm
LM2903-N	SOIC (8)	4.90 mm x 3.91 mm
	DSBGA (8)	1.54 mm x 1.54 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

Simplified Schematic

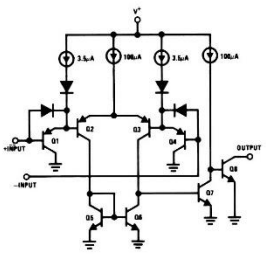


Figura C1. Características técnicas del circuito LM393

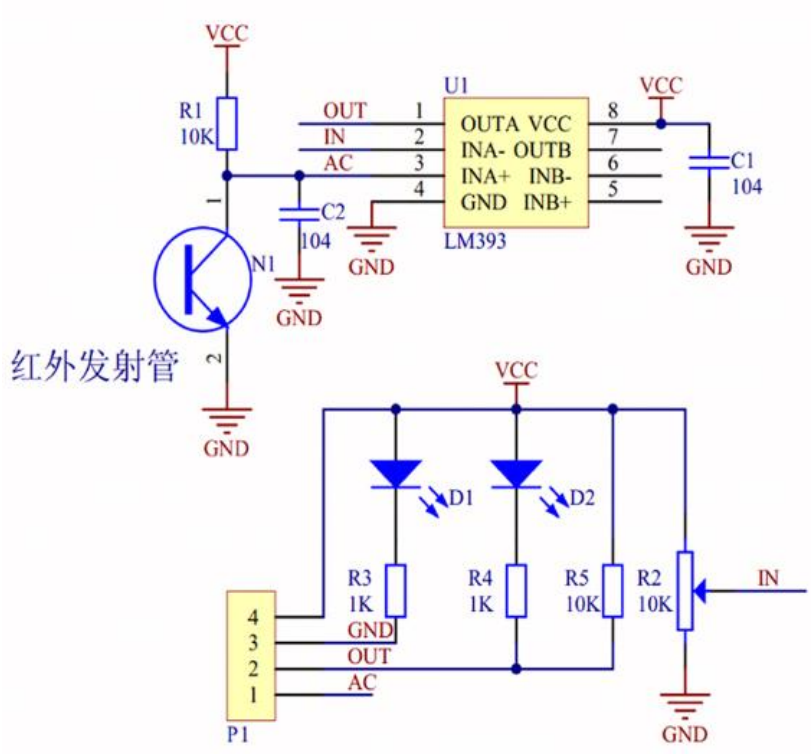


Figura C2. Esquema de conexión del circuito YL-38

Anexo D. Datasheet del servomotor TD-8120MG

En la Figura D1, se muestran las características técnicas del servomotor TD-8120MG

SHENZHEN SKY STAR TECHNOLOGY CO., LTD			
Data Sheet of TD-8120MG Digital Servo Motor			
Products Specifcation			
Products Name:TD-8120MG Digital servo		**Products No.**	
1.Apply Environmental Conditon			
No.	Item	Specifcation	
1.1	Storage Temperature Range	-30°C～80°C	
1.2	Operating Temperature Range	-25°C～70°C	
2. Standard Test Environment			
No.	Item	Specifcation	
2.1	Temperature Range	-25°C～70°C	
2.2	Humidity Range	65%±10%	
3. Mechanical Specification			
No.	Item	Specifcation	
3.1	Size	40.0*20.5*40.5mm	
3.2	Weight	60g±5%	
3.3	Gear type	5 Metal	
3.4	Limit angle	360°	
3.5	Bearing	2BB	
3.6	Horn gear spline	25T Diameter:6.0mm Height :4.0mm	
3.7	Horn type	Plastic, POM	
3.8	Case	Engineering plastics(PBT)+Aluminum alloy	
3.9	Connector wire	320mm±5mm	
3.10	Motor	Carbon brush motor	
3.11	Splash water resistance	yes	
4. Electrical Specification			
No.	Operation voltage	4.8V	8.4V
4.1	Idle current	mA	mA
4.2	No load speed	0.18sec/60°	0.14sec/60°
4.3	Runnig current	210mA	260mA
4.4	Peak stall torque	20.5kg.cm	22.8kg.cm
4.5	Stall current	2100mA±10%	2700mA±10%
4.6	Working voltage range	4.8-8.4V	
5. Control Specification			
No.	Item	Specifcation	
5.1	Command signal	Pulse width modification	
5.2	Amplifier type	Digital controller	
5.3	Pulse width range	500～2500usec	
5.4	Neutral position	1500usec	
5.5	Running degree	180°±3°(when 500～2500usec)	
5.6	Dead band width	3 usec	
5.7	Rotating direction	Counterclockwise (when 1000～2000usec)	
6. Drawings:		Wire And Plug:	

Figura D1. Características técnicas del servomotor TD-8120MG

Anexo E. Datasheet de la válvula solenoide

En la Figura E1, se muestra las características técnicas de válvula solenoide

ZE-4F180
12V Water Solenoid Valve
Technical Manual Rev 1r0

e-Gizmo
MECHATRONIX CENTRAL

The ZE-4F180 12V Water Solenoid Valve is normally closed water solenoid valve, so if you put pressurized water, the water will be blocked. Then, if you power the magnet with the expected current/voltage, the valve will open and the water will flow.

FEATURES:

- *Normally Closed*
- *For water or low viscosity fluids control*

GENERAL SPECIFICATIONS:

- **Input Supply: *+12VDC***
- **Rated Power: 5W**
- **Material: *Plastic***
- **Flow Characteristics: *1.5L/min, 20L/min***
- **Water Pressure: *0.02 to 0.8MPa***
- **Port size: G1/2 inches**
- **Fluid Temperature: 0 ~ 100 deg Cm**
- **PCB Dimensions: 8.5cmx4.5cm**
- **Weight: 100g**

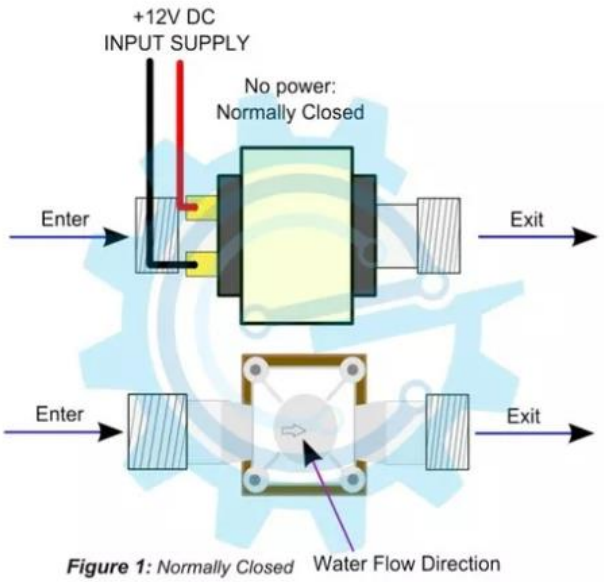


Figure 1: *Normally Closed*

Figura E1. Características técnicas de válvula solenoide

En la Figura F1 y en la Figura F2, se muestra el datasheet del relé SRD-05VDC-SL-C

SONGLE RELAY

松乐继电器 ® SONGLE RELAY	RELAY ISO9002	SRD

1. MAIN FEATURES

- Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- UL,CUL,TUV recognized.
- Selection of plastic material for high temperature and better chemical solution performance.
- Sealed types available.
- Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

- Domestic appliance, office machine, audio, equipment, automobile, etc.
  (Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SRD	03、05、06、09、12、24、48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

4. RATING

CCC	FILE NUMBER:CH0052885-2000	7A/240VDC
CCC	FILE NUMBER:CH0036746-99	10A/250VDC
UL /CUL	FILE NUMBER: E167996	10A/125VAC 28VDC
TUV	FILE NUMBER: R9933789	10A/240VAC 28VDC

5. DIMENSION(unit:mm) **DRILLING**(unit:mm) **WIRING DIAGRAM**

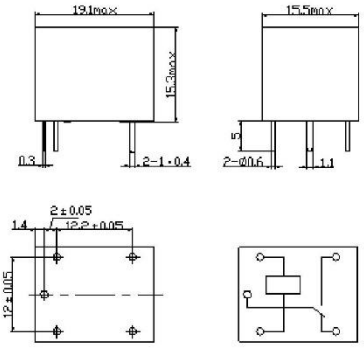


Figura F1. Parte uno de características técnicas del relé SRD-05VDC-SL-C

6. COIL DATA CHART (AT20°C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ±10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

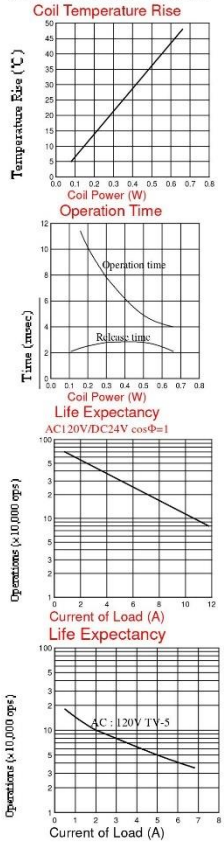
7. CONTACT RATING

Item \ Type	SRD	
	FORM C	FORM A
Contact Capacity Resistive Load (cosΦ=1)	7A 28VDC 10A 125VAC 7A 240VAC	10A 28VDC 10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at initial value)

Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-25°C to +70°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA



2

Figura F2. Parte dos de características técnicas del relé SRD-05VDC-SL-C

Anexo G. Planos del sistema de riego

En la Figura G1, Figura G2, Figura G3, Figura G4, Figura G5, Figura G6, Figura G7 y Figura G8, se muestran los planos del sistema de riego

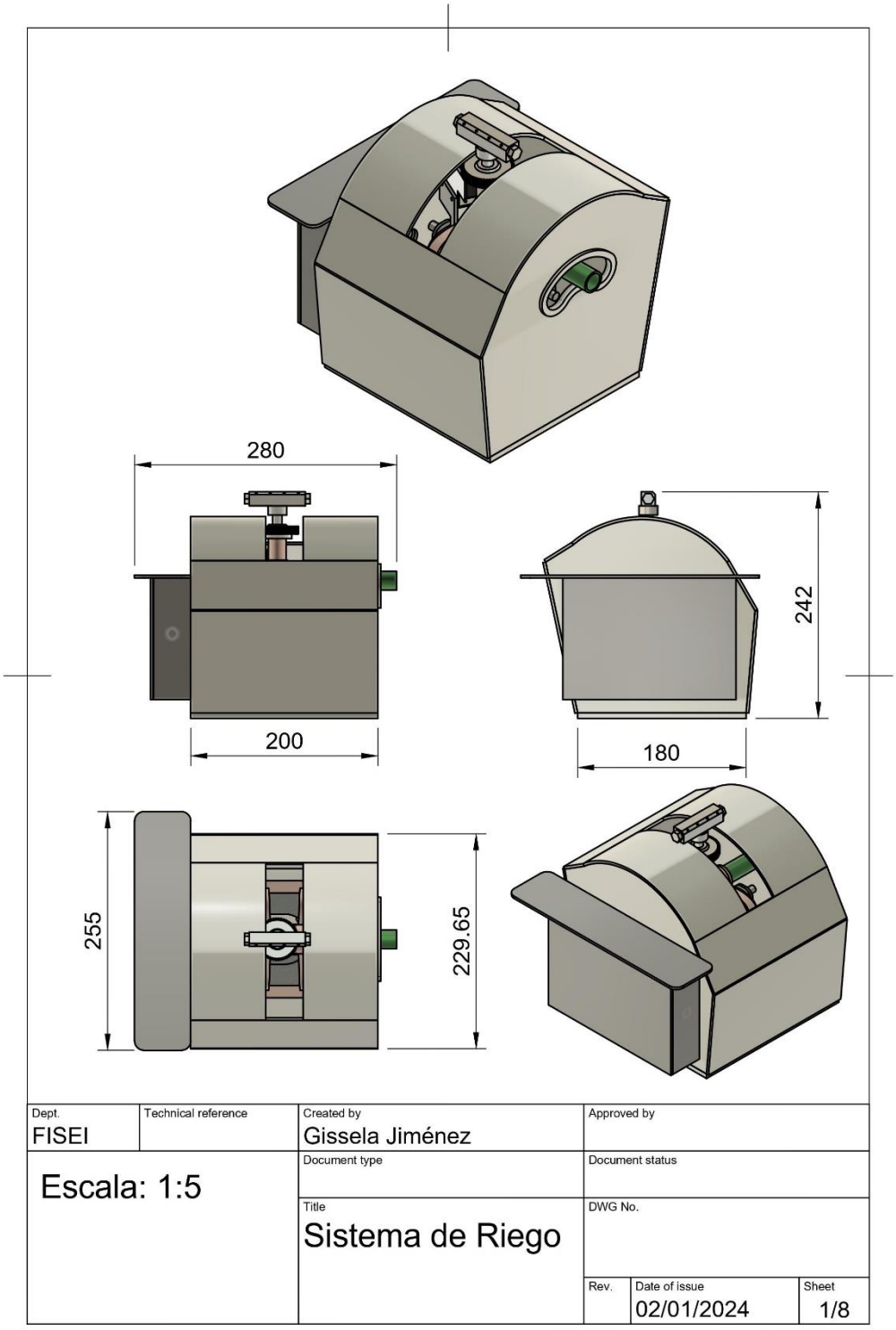
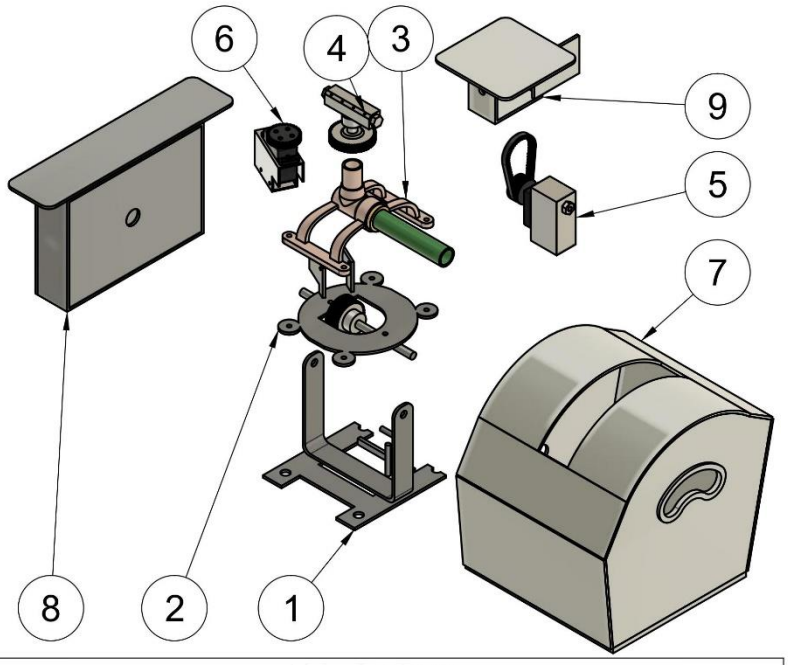


Figura G1. Plano general del sistema de riego



Lista de piezas				
Elemento	Ctd	Nombre de pieza	Descripción	Material
1	1	Base	Estructura de platinas soldadas	
2	1	Eje	Ø 8mm	
3	1	Acople de manguera	Pieza Fundida	
4	1	Aspersor		Acero
5	1	TD-8120MG 01 v1	Servo 20 Kg	
6	1	TD-8120MG 02 v1	Servo 20 Kg	
7	1	Case Mecanismo	Paneles de 3mm	Acrílico
8	1	Case Dispositivos	Paneles de 3mm	Acrílico
9	1	Case Camara	Paneles de 3mm	Acrílico

Dept.	Technical reference	Created by	Approved by	
FISEI		Gissela Jiménez		
Escala: 1:5		Document type	Document status	
		Title: Sistema de Riego Explosionado	DWG No.	
			Rev.	Date of issue: 02/01/2024 — Sheet: 2/8

Figura G2. Elementos del sistema

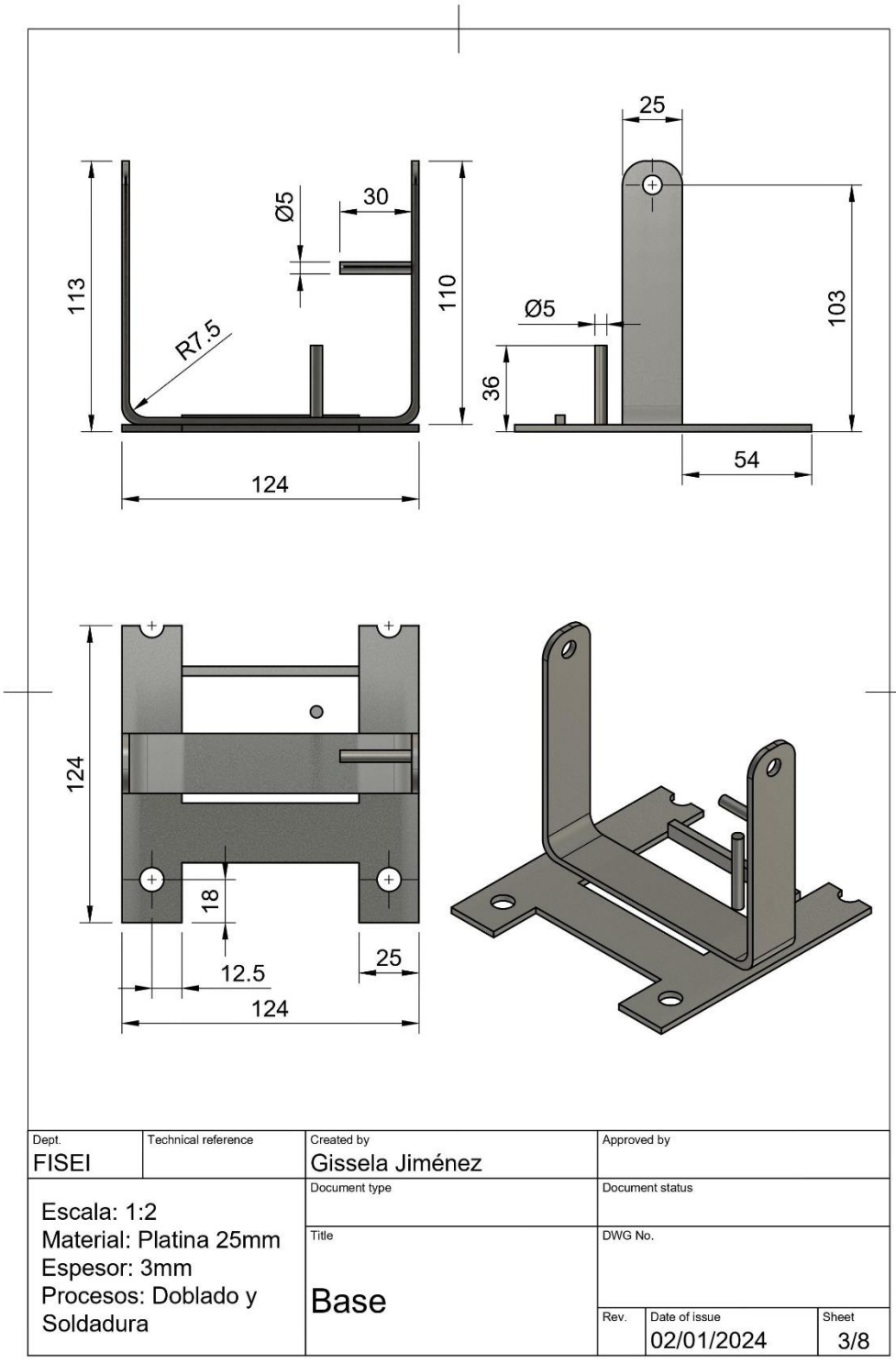


Figura G3. Planos de la base

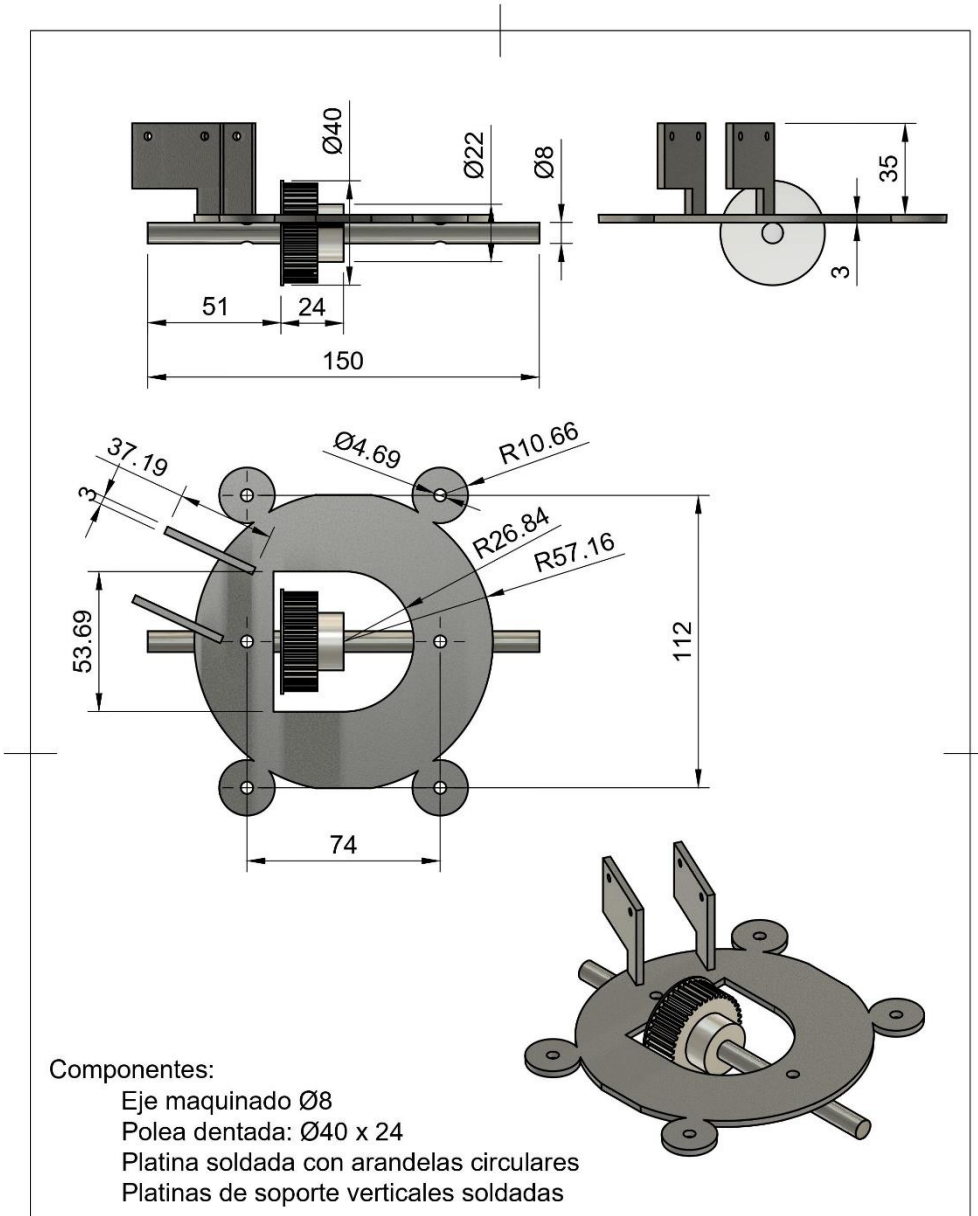


Figura G4. Planos del eje

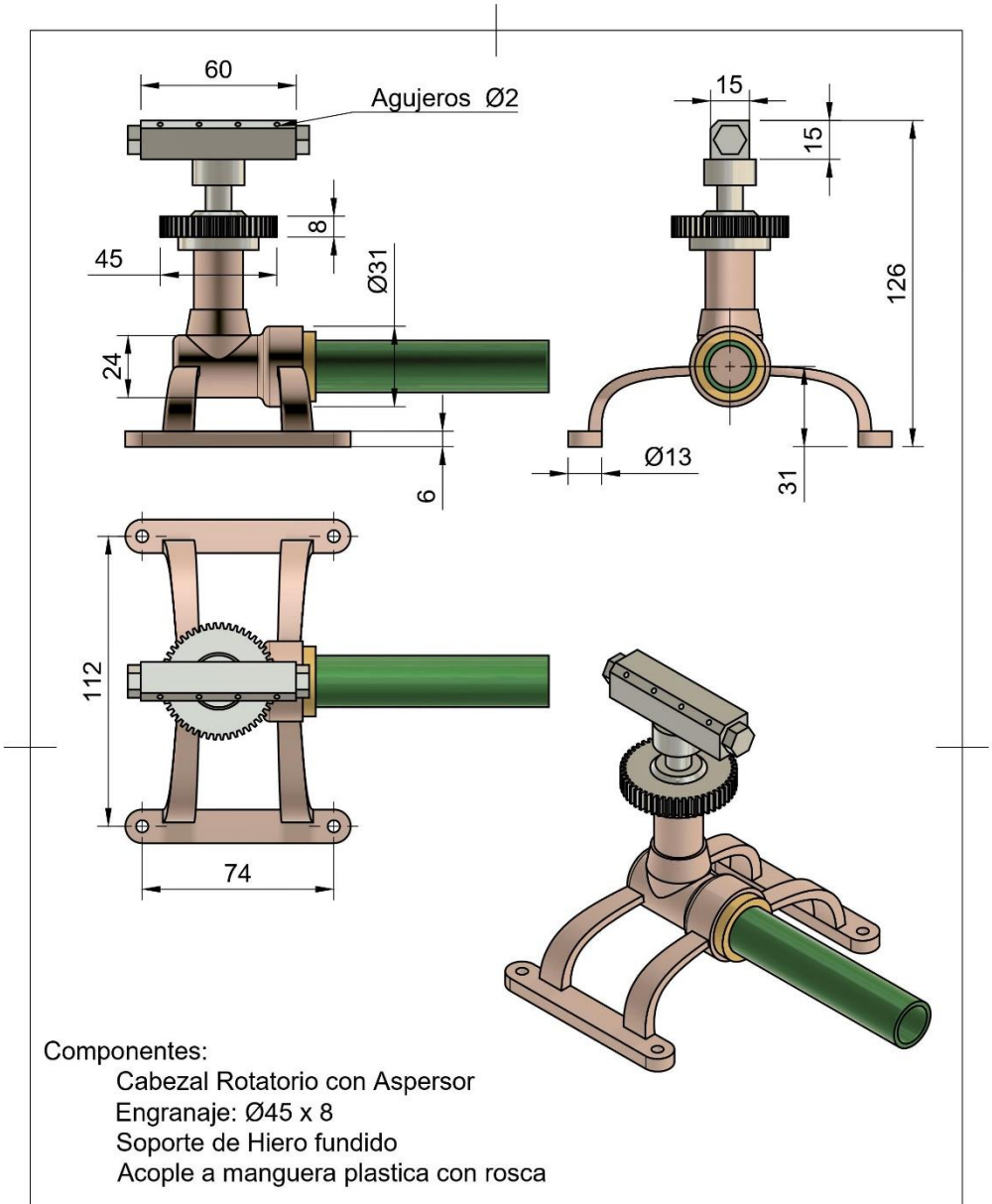
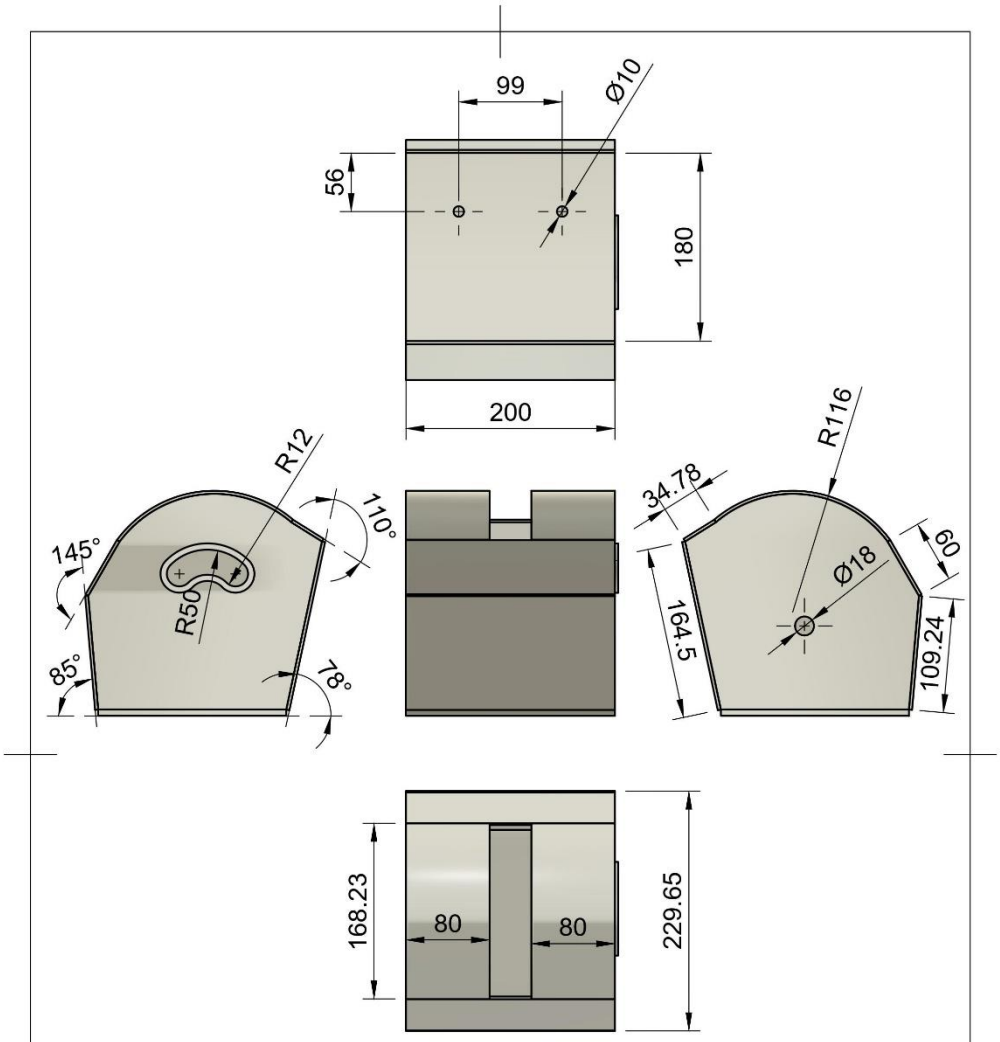


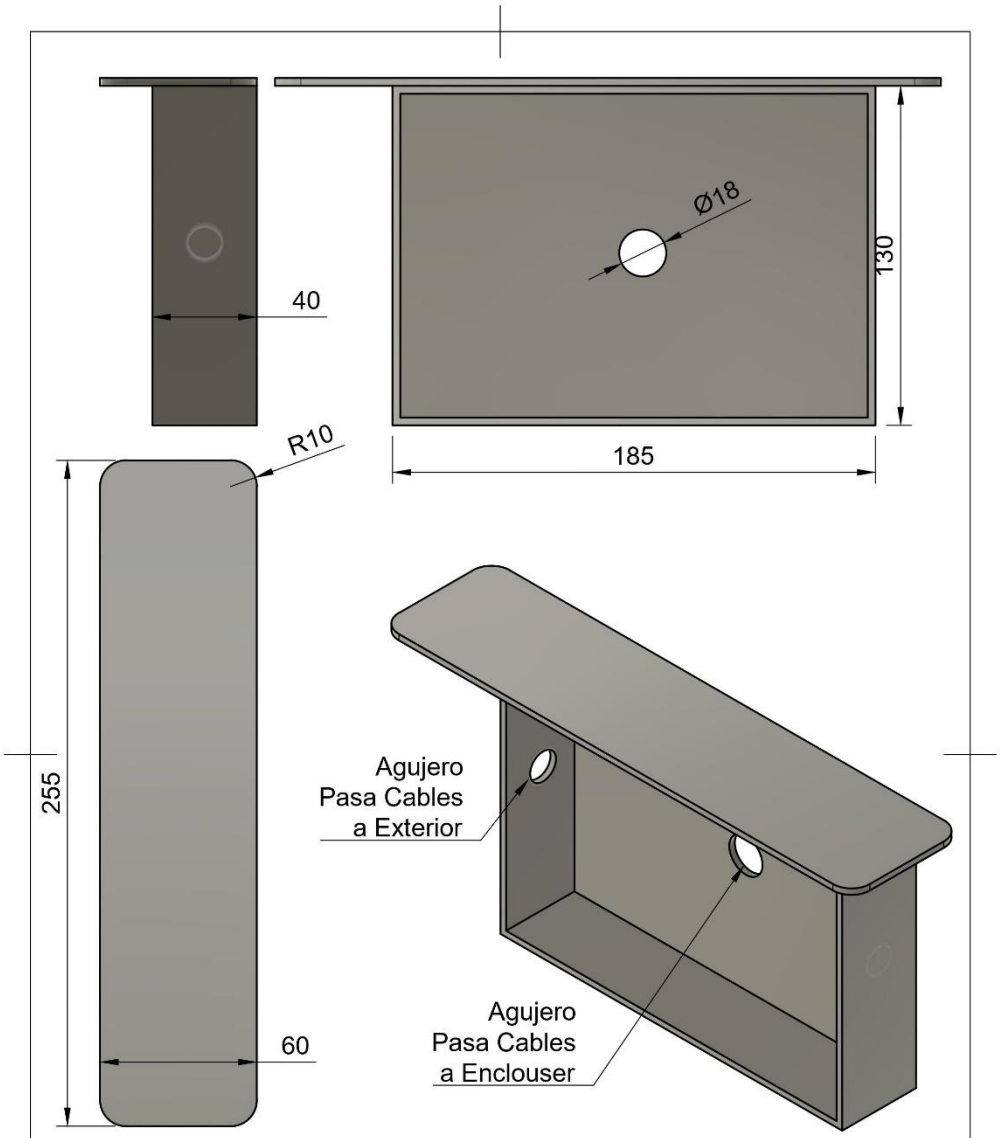
Figura G5. Planos del aspersor



Caja Protectora de Componentes Mecanicos y Servos
Localizacion: Rodeando la Estructura del Sistema de Riego
Especificaciones:
Placha de base. Espesor 6mm
Resto de superficies. Espesor 3mm

Dept.	Technical reference	Created by	Approved by	
FISEI		Gissela Jiménez		
Escala: 1:5 Material: Acrílico Procesos: Corte de Acrílico y Soldadura Química		Document type	Document status	
		Title Case Mecanismo	DWG No.	
			Rev.	Date of issue: 02/01/2024 / Sheet: 6/8

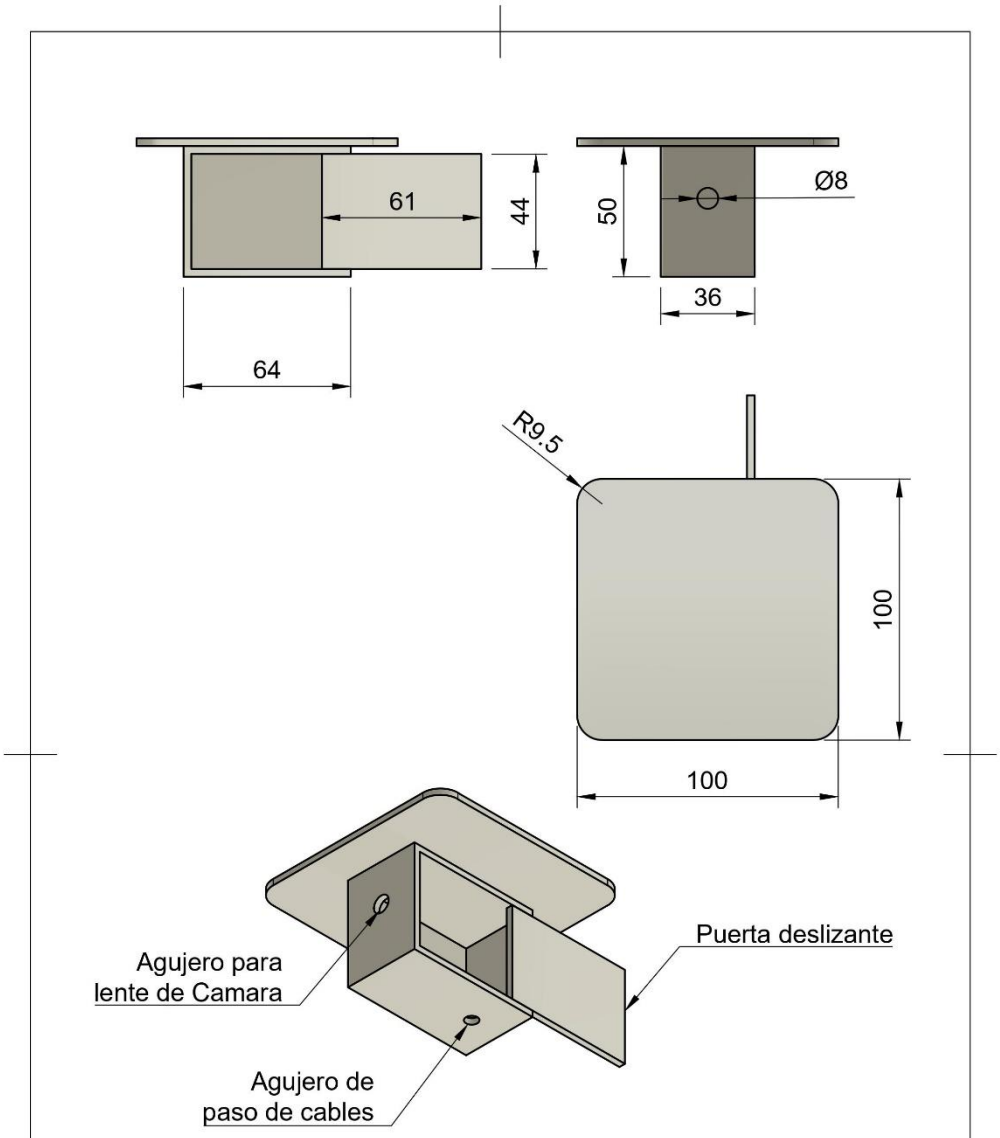
Figura G6. Planos del case del mecanismo



Caja Protectora de Componentes Electronicos del sistema
Localizacion: Adyacente al Enclouser

Dept.	Technical reference	Created by	Approved by	
FISEI		Gissela Jiménez		
Escala: 1:2 Material: Acrílico de 3mm Procesos: Corte de Acrílico y Soldadura Química		Document type	Document status	
		Title Case Dispositivos	DWG No.	
			Rev.	Date of issue: 02/01/2024 / Sheet: 7/8

Figura G7. Planos del case de dispositivos



Caja Protectora de Camara para Observacion Remota
Localizacion: Aparte del sistema de riego, en posicion elevada

Dept.	Technical reference	Created by	Approved by	
FISEI		Gissela Jiménez		
Escala: 1:2 Material: Acrílico de 3mm Procesos: Corte de Acrílico y Soldadura Química		Document type	Document status	
		Title Case Camara	DWG No.	
			Rev.	Date of issue: 02/01/2024 — Sheet: 8/8

Figura G8. Planos del case de la cámara

```
import tkinter
import tkinter.font as font
from tkinter import ttk
from tkinter import *
from tkinter import messagebox
import cv2
import time
import threading
import torch
import numpy as np
import json as json
from PIL import Image, ImageTk
from urllib.parse import urlencode
from urllib.request import Request, urlopen
from urllib.parse import quote
import requests
from bs4 import BeautifulSoup
from io import BytesIO
import math

nueva_imgCam1 = None
nueva_imgCam2 = None
valorAutomatico = 0
valorEstado=0
ipespcam = "192.168.214.67"
try:
    time.sleep(1)
    model = torch.hub.load('ultralytics/yolov5', 'custom',
                    path='C:/Users/PC/Documents/Proyecto/Sistema Final Python/Sistema Final/modelo4.pt')
except Exception as e:
    print(f"Error al conectarse a Yolo: {e}")
ventana = tkinter.Tk()
ventana.state("zoomed")
ventana.title("SISTEMA")
ventana.config(bg="grey")
datos = tkinter.Toplevel(ventana)
datos.state("withdraw")
datos.title("SISTEMA")
datos.config(bg="#d9d9d9")
registrar = tkinter.Toplevel(ventana)
registrar.state("withdraw")
registrar.title("SISTEMA")
registrar.config(bg="#d9d9d9")
def traerProceso():
    request = Request("http://localhost/sistema/datos/proceso.php")
    json = urlopen(request).read().decode()
    return json
def actualizarClima(tiempo,temperatura,humedad,lluvia,viento):
    tiempo_encoded = quote(str(tiempo))
    temperatura_encoded = quote(str(temperatura))
    humedad_encoded = quote(str(humedad))
    lluvia_encoded = quote(str(lluvia))
    viento_encoded = quote(str(viento))
    url =
f"http://localhost/sistema/datos/actualizarClima.php?tiempo={tiempo_encoded}&temperatura={temperatura_enc
oded}&humedad={humedad_encoded}&lluvia={lluvia_encoded}&viento={viento_encoded}"
    request = Request(url)
    json = urlopen(request).read().decode()
    return json
def registrarUsuario(nombre,usuario,email,contrasena):
    nombre_encoded = str(nombre)
```

```
    usuario_encoded = str(usuario)
    email_encoded = str(email)
    contrasena_encoded = str(contrasena)
    url = "http://localhost/sistema/datos/registrarUsuario.php"
    datos = {'nombre': nombre_encoded, 'usuario': usuario_encoded,'email': email_encoded, 'contrasena': contrasena_encoded}
    respuesta = requests.post(url, json=datos)
    return respuesta.json()
def actualizarParametros(anguloMaximo,anguloMinimoH,anguloMaximoH,valorMinX,valorMaxX,valorMaxY):
    anguloMaximo_encoded = quote(str(anguloMaximo))
    anguloMinimoH_encoded = quote(str(anguloMinimoH))
    anguloMaximoH_encoded = quote(str(anguloMaximoH))
    valorMinX_encoded = quote(str(valorMinX))
    valorMaxX_encoded = quote(str(valorMaxX))
    valorMaxY_encoded = quote(str(valorMaxY))
    url = f"http://localhost/sistema/datos/actualizarParametros.php?anguloMaximo={anguloMaximo_encoded}&anguloMinimoH={anguloMinimoH_encoded}&anguloMaximoH={anguloMaximoH_encoded}&valorMinX={valorMinX_encoded}&valorMaxX={valorMaxX_encoded}&valorMaxY={valorMaxY_encoded}"
    request = Request(url)
    json = urlopen(request).read().decode()
    return json
def actualizarPorcentaje(valor):
    valor_encoded = quote(str(valor))
    url = f"http://localhost/sistema/datos/actualizarPorcentaje.php?valor={valor_encoded}"
    request = Request(url)
    json = urlopen(request).read().decode()
    return json
def cambiarAutomatico():
    global valorAutomatico
    valEnviar=0
    if valorAutomatico == 0:
        valEnviar=1
    request = Request("http://localhost/sistema/datos/actualizarAutomatico.php?valor="+str(valEnviar)+"")
    json = urlopen(request).read().decode()
    return json
def iniciarSesion(usuario, contrasena):
    url = "http://localhost/sistema/datos/IniciarSesion.php"
    datos = {'usuario': usuario, 'contrasena': contrasena}
    respuesta = requests.post(url, json=datos)
    print(respuesta)
    return respuesta.json()
def actualizarEstado():
    global valorEstado
    valEnviar=0
    if valorEstado == 0:
        valEnviar=1
    request = Request("http://localhost/sistema/datos/actualizarEstado.php?valor="+str(valEnviar)+"")
    json = urlopen(request).read().decode()
    actualizarPorcentaje(0)
    return json
def cambiarEstado():
    request = Request("http://localhost/sistema/datos/cambiarEstado.php")
    json = urlopen(request).read().decode()
    actualizarPorcentaje(0)
    return json
def capturarImagen():
    global valorEstado
    try:
        actualizarPorcentaje(5)
        tiempo_de_espera = 5
        response = requests.get(f"http://{ipespcam}/capture", timeout=tiempo_de_espera)
        if response.status_code == 200:
            image_data = BytesIO(response.content)
            image = Image.open(image_data)
            image.save('imagenServidor.jpeg', "JPEG")
```

```
            image.close()
            cambiarEstado()
            actualizarPorcentaje(10)
        else:
            print(f"Error al capturar la imagen. Código de estado: {response.status_code}")
            valorEstado=0
            actualizarEstado()
            actualizarPorcentaje(0)
    except Exception as e:
        print(f"Error: {e}")
        valorEstado=0
        actualizarEstado()
        actualizarPorcentaje(0)
def abrirCaratula():
    ventana.state(newstate="zoomed")
    datos.state(newstate="withdraw")
    registrar.state(newstate="withdraw")
def ventanaPrincipal():
    global lblImgCam1
    global lblImgCam2
    global lblImgCam3
    global lblImgCam3
    global lblImgCam5
    global nueva_imgCam1
    global nueva_imgCam2
    global entryDistancia
    actualizarPorcentaje(15)
    image_path = "imagenServidor.jpeg"
    im1 = Image.open(image_path)
    model.conf = 0.9
    detect = model(im1, size=400)
    nueva_imagen_Principal = im1.resize((350, 200))
    nueva_imgCam1 = ImageTk.PhotoImage(nueva_imagen_Principal)
    lblImgCam1.configure(image=nueva_imgCam1)
    print("leer imagen")
    info = detect.pandas().xyxy[0]
    arr = np.asarray(info.confidence.array)
    detectar = len(arr)
    actualizarPorcentaje(18)
    if detectar > 0:
        print("verificando")
        cv2.imwrite("resultado.png", cv2.cvtColor(np.squeeze(detect.render()), cv2.COLOR_BGR2RGB))
        cv2.imwrite("C:/xampp/htdocs/sistema/resultado.png", cv2.cvtColor(np.squeeze(detect.render()),
cv2.COLOR_BGR2RGB))
        nueva_imagen = Image.open('resultado.png').convert('RGB')
        box_coords = info[['xmin', 'ymin', 'xmax', 'ymax']].to_numpy(dtype=int)[0]
        roi = np.array(nueva_imagen)[box_coords[1]:box_coords[3], box_coords[0]:box_coords[2], :]
        cv2.imwrite("soloCesped.png", cv2.cvtColor(roi, cv2.COLOR_RGB2BGR))
        nueva_imagen = nueva_imagen.resize((350, 200))
        nueva_imgCam2 = ImageTk.PhotoImage(nueva_imagen)
        lblImgCam2.configure(image=nueva_imgCam2)
        imagenSoloCesped = cv2.imread('soloCesped.png')
        imagen_solocesped_hsv = cv2.cvtColor(imagenSoloCesped, cv2.COLOR_BGR2HSV)
        factor_saturacion = 5
        imagen_solocesped_hsv[:, :, 1] = np.clip(imagen_solocesped_hsv[:, :, 1] * factor_saturacion, 0,
255).astype(np.uint8)
        imagen_corregida = cv2.cvtColor(imagen_solocesped_hsv, cv2.COLOR_HSV2BGR)
        cv2.imwrite('corregida.png', imagen_corregida)
        imagenCorregida1 = cv2.imread('corregida.png')
        imagen_hsv_corr = cv2.cvtColor(imagenCorregida1, cv2.COLOR_BGR2HSV)
        verde_bajo_imgCor = np.array([0, 0,50], np.uint8)
        verde_alto_imgCor = np.array([120, 255, 255], np.uint8)
        red_bajo_imgCor=np.array([150, 0, 0], np.uint8)
        red_alto_imgCor=np.array([179, 255, 255], np.uint8)
        mascara_verde_corr = cv2.inRange(imagen_hsv_corr, verde_bajo_imgCor, verde_alto_imgCor)
        mask_red_corr = cv2.inRange(imagen_hsv_corr, red_bajo_imgCor, red_alto_imgCor)
```

```
        maskCompleta_corr = cv2.add(mascara_verde_corr, mask_red_corr)
        pixeles_verdes_completo = cv2.bitwise_and(imagenCorregida1, imagenCorregida1,
mask=maskCompleta_corr)
        contornos, _ = cv2.findContours(mascara_verde_corr, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cv2.drawContours(pixeles_verdes_completo, contornos, -1, (0, 255, 0), 2)
        mascara_completa = cv2.bitwise_or(mascara_verde_corr, mask_red_corr)
        pixeles_verdes_completo = np.zeros_like(imagenCorregida1)
        pixeles_verdes_completo[mascara_completa != 0] = imagenCorregida1[mascara_completa != 0]
        cv2.imwrite('verdesCompleto.png', pixeles_verdes_completo)
        imagen_VC = cv2.imread('verdesCompleto.png')
        imagen_gris_VC = cv2.cvtColor(imagen_VC, cv2.COLOR_BGR2GRAY)
        imagen_umbral_VC = cv2.adaptiveThreshold(imagen_gris_VC, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 11, 4)
        contornos_VC, _ = cv2.findContours(imagen_umbral_VC, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        area_minima_VC = 9000
        contornos_filtradosVC = [cnt for cnt in contornos_VC if cv2.contourArea(cnt) > area_minima_VC]
        cv2.drawContours(imagen_VC, contornos_filtradosVC, -1, (0, 255, 0), 3)
        cv2.imwrite('contornoVerdeCompleto.png', imagen_VC)
        cesped_imagen = Image.open('contornoVerdeCompleto.png')
        nueva_imagen2 = cesped_imagen.resize((300, 200))
        cesped_imgCam1 = ImageTk.PhotoImage(nueva_imagen2)
        lblImgCam4.configure(image=cesped_imgCam1)
        lblImgCam4.image = cesped_imgCam1
        imagenObjeto = cv2.imread('verdesCompleto.png')
        imagen_grisObjeto = cv2.cvtColor(imagenObjeto, cv2.COLOR_BGR2GRAY)
        imagen_umbralObjeto = cv2.adaptiveThreshold(imagen_grisObjeto, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 11, 4)
        contornosObjeto, _ = cv2.findContours(imagen_umbralObjeto, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        area_minimaObjeto = 3000
        altura_minimaObjeto = 130
        altura_maximaObjeto = 260
        contornos_filtradosObjeto = [cnt for cnt in contornosObjeto if cv2.contourArea(cnt) > area_minimaObjeto
and altura_minimaObjeto < cnt[0][0][1] < altura_maximaObjeto]
        imagen_contornosObjeto = imagenObjeto.copy()
        cv2.drawContours(imagen_contornosObjeto, contornos_filtradosObjeto, -1, (0, 0, 255), 3)
        cv2.imwrite('contornoObjeto.png', imagen_contornosObjeto)
        cv2.imwrite("C:/xampp/htdocs/sistema/contorno.png", imagen_contornosObjeto)
        objeto_imagen = Image.open('contornoObjeto.png')
        objeto_imagen2 = objeto_imagen.resize((300, 200))
        objeto_imgCam1 = ImageTk.PhotoImage(objeto_imagen2)
        lblImgCam5.configure(image=objeto_imgCam1)
        lblImgCam5.image = objeto_imgCam1
        objeto_imagen_so = Image.open('soloCesped.png')
        objeto_imagen2_so = objeto_imagen_so.resize((300, 200))
        objeto_imgCam1_so = ImageTk.PhotoImage(objeto_imagen2_so)
        lblImgCam3.configure(image=objeto_imgCam1_so)
        lblImgCam3.image = objeto_imgCam1_so
        try:
            if contornos_filtradosObjeto:
                alto, ancho, _ = imagenObjeto.shape
                nuevo_ancho = 180
                nuevo_alto = 80
                factor_escala_ancho = nuevo_ancho / ancho
                factor_escala_alto = nuevo_alto / alto
                contorno_transformado = contornos_filtradosObjeto[0] * [factor_escala_ancho, factor_escala_alto]
                punto_mas_bajo_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :,
1].argmax()][0]))
                punto_mas_izquierda_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :,
0].argmin()][0]))
                punto_mas_derecha_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :,
0].argmax()][0]))
                x_bajo, y_bajo = punto_mas_bajo_nuevo
                x_izquierda, y_izquierda = punto_mas_izquierda_nuevo
```

```
            x_derecha, y_derecha = punto_mas_derecha_nuevo
        else:
            y_bajo=0
            x_izquierda=0
            x_derecha=0
    except Exception as e:
        y_bajo=0
        x_izquierda=0
        x_derecha=0
        y_bajo=(math.floor(y_bajo / 10) * 10)-8
    x_izquierda=(math.floor(x_izquierda / 10) * 10 )
    x_derecha=(math.ceil(x_derecha / 10) * 10)
    actualizarParametros(70,0,180,x_izquierda,x_derecha,y_bajo)
    cambiarEstado()
  else:
    print("no hay dato")
    nueva_imagen = Image.open('imagenServidor.jpeg').convert('RGB')
    box_coords = info[['xmin', 'ymin', 'xmax', 'ymax']].to_numpy(dtype=int)[0]
    roi = np.array(nueva_imagen)[box_coords[1]:box_coords[3], box_coords[0]:box_coords[2], :]
    cv2.imwrite("soloCesped.png", cv2.cvtColor(roi, cv2.COLOR_RGB2BGR))
    nueva_imagen = nueva_imagen.resize((350, 200))
    nueva_imgCam2 = ImageTk.PhotoImage(nueva_imagen)
    lblImgCam2.configure(image=nueva_imgCam2)
    imagenSoloCesped = cv2.imread('soloCesped.png')
    imagen_solocesped_hsv = cv2.cvtColor(imagenSoloCesped, cv2.COLOR_BGR2HSV)
    factor_saturacion = 5
    imagen_solocesped_hsv[:, :, 1] = np.clip(imagen_solocesped_hsv[:, :, 1] * factor_saturacion, 0,
255).astype(np.uint8)
    imagen_corregida = cv2.cvtColor(imagen_solocesped_hsv, cv2.COLOR_HSV2BGR)
    cv2.imwrite('corregida.png', imagen_corregida)
    imagenSoloCesped_1 = cv2.imread('resultado.png')
    imagen_solocesped_hsv_1 = cv2.cvtColor(imagenSoloCesped_1, cv2.COLOR_BGR2HSV)
    factor_saturacion_1 = 5
    imagen_solocesped_hsv_1[:, :, 1] = np.clip(imagen_solocesped_hsv_1[:, :, 1] * factor_saturacion_1, 0,
255).astype(np.uint8)
    imagen_corregida_1 = cv2.cvtColor(imagen_solocesped_hsv_1, cv2.COLOR_HSV2BGR)
    cv2.imwrite('corregida1.png', imagen_corregida_1)
    imagenCorregida1 = cv2.imread('corregida.png')
    imagen_hsv_corr = cv2.cvtColor(imagenCorregida1, cv2.COLOR_BGR2HSV)
    verde_bajo_imgCor = np.array([0, 0,50], np.uint8)
    verde_alto_imgCor = np.array([120, 255, 255], np.uint8)
    red_bajo_imgCor=np.array([150, 0, 0], np.uint8)
    red_alto_imgCor=np.array([179, 255, 255], np.uint8)
    mascara_verde_corr = cv2.inRange(imagen_hsv_corr, verde_bajo_imgCor, verde_alto_imgCor)
    mask_red_corr = cv2.inRange(imagen_hsv_corr, red_bajo_imgCor, red_alto_imgCor)
    maskCompleta_corr = cv2.add(mascara_verde_corr, mask_red_corr)
    pixeles_verdes_completo = cv2.bitwise_and(imagenCorregida1, imagenCorregida1,
mask=maskCompleta_corr)
    contornos, _ = cv2.findContours(mascara_verde_corr, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(pixeles_verdes_completo, contornos, -1, (0, 255, 0), 2)
    mascara_completa = cv2.bitwise_or(mascara_verde_corr, mask_red_corr)
    pixeles_verdes_completo = np.zeros_like(imagenCorregida1)
    pixeles_verdes_completo[mascara_completa != 0] = imagenCorregida1[mascara_completa != 0]
    cv2.imwrite('verdesCompleto.png', pixeles_verdes_completo)
    imagenCorregida1_1 = cv2.imread('corregida.png')
    imagen_hsv_corr_1 = cv2.cvtColor(imagenCorregida1_1, cv2.COLOR_BGR2HSV)
    verde_bajo_imgCor_1 = np.array([0, 0,50], np.uint8)
    verde_alto_imgCor_1 = np.array([120, 255, 255], np.uint8)
    red_bajo_imgCor_1=np.array([150, 0, 0], np.uint8)
    red_alto_imgCor_1=np.array([179, 255, 255], np.uint8)
    mascara_verde_corr_1 = cv2.inRange(imagen_hsv_corr_1, verde_bajo_imgCor_1, verde_alto_imgCor_1)
    mask_red_corr_1 = cv2.inRange(imagen_hsv_corr_1, red_bajo_imgCor_1, red_alto_imgCor_1)
    maskCompleta_corr_1 = cv2.add(mascara_verde_corr_1, mask_red_corr_1)
    pixeles_verdes_completo_1 = cv2.bitwise_and(imagenCorregida1_1, imagenCorregida1_1,
mask=maskCompleta_corr_1)
```

```
    contornos_1, _ = cv2.findContours(mascara_verde_corr_1, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(pixeles_verdes_completo_1, contornos_1, -1, (0, 255, 0), 2)
    mascara_completa_1 = cv2.bitwise_or(mascara_verde_corr_1, mask_red_corr_1)
    pixeles_verdes_completo_1 = np.zeros_like(imagenCorregida1_1)
    pixeles_verdes_completo_1[mascara_completa_1 != 0] = imagenCorregida1_1[mascara_completa_1 != 0]
    cv2.imwrite('verdesCompleto1.png', pixeles_verdes_completo_1)
    imagen_VC = cv2.imread('verdesCompleto.png')
    imagen_gris_VC = cv2.cvtColor(imagen_VC, cv2.COLOR_BGR2GRAY)
    imagen_umbral_VC = cv2.adaptiveThreshold(imagen_gris_VC, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 11, 4)
    contornos_VC, _ = cv2.findContours(imagen_umbral_VC, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    area_minima_VC = 9000
    contornos_filtradosVC = [cnt for cnt in contornos_VC if cv2.contourArea(cnt) > area_minima_VC]
    cv2.drawContours(imagen_VC, contornos_filtradosVC, -1, (0, 255, 0), 3)
    cv2.imwrite('contornoVerdeCompleto.png', imagen_VC)
    cesped_imagen = Image.open('contornoVerdeCompleto.png')
    nueva_imagen2 = cesped_imagen.resize((300, 200))
    cesped_imgCam1 = ImageTk.PhotoImage(nueva_imagen2)
    lblImgCam4.configure(image=cesped_imgCam1)
    lblImgCam4.image = cesped_imgCam1
    imagenObjeto = cv2.imread('verdesCompleto1.png')
    imagen_grisObjeto = cv2.cvtColor(imagenObjeto, cv2.COLOR_BGR2GRAY)
    imagen_umbralObjeto = cv2.adaptiveThreshold(imagen_grisObjeto, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 11, 4)
    contornosObjeto, _ = cv2.findContours(imagen_umbralObjeto, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    area_minimaObjeto = 2000
    altura_minimaObjeto = 130
    altura_maximaObjeto = 260
    contornos_filtradosObjeto = [cnt for cnt in contornosObjeto if cv2.contourArea(cnt) > area_minimaObjeto and altura_minimaObjeto < cnt[0][0][1] < altura_maximaObjeto]
    imagen_contornosObjeto = imagenObjeto.copy()
    cv2.drawContours(imagen_contornosObjeto, contornos_filtradosObjeto, -1, (0, 0, 255), 3)
    cv2.imwrite('contornoObjeto.png', imagen_contornosObjeto)
    cv2.imwrite("C:/xampp/htdocs/sistema/contorno.png", imagen_contornosObjeto)
    objeto_imagen = Image.open('contornoObjeto.png')
    objeto_imagen2 = objeto_imagen.resize((300, 200))
    objeto_imgCam1 = ImageTk.PhotoImage(objeto_imagen2)
    lblImgCam5.configure(image=objeto_imgCam1)
    lblImgCam5.image = objeto_imgCam1
    objeto_imagen_so = Image.open('soloCesped.png')
    objeto_imagen2_so = objeto_imagen_so.resize((300, 200))
    objeto_imgCam1_so = ImageTk.PhotoImage(objeto_imagen2_so)
    lblImgCam3.configure(image=objeto_imgCam1_so)
    lblImgCam3.image = objeto_imgCam1_so
    try:
        if contornos_filtradosObjeto:
            alto, ancho, _ = imagenObjeto.shape
            punto_mas_bajo = tuple(contornos_filtradosObjeto[0][contornos_filtradosObjeto[0][:, :, 1].argmax()][0])
            punto_mas_izquierda = tuple(contornos_filtradosObjeto[0][contornos_filtradosObjeto[0][:, :, 0].argmin()][0])
            punto_mas_derecha = tuple(contornos_filtradosObjeto[0][contornos_filtradosObjeto[0][:, :, 0].argmax()][0])
            nuevo_ancho = 180
            nuevo_alto = 80
            factor_escala_ancho = nuevo_ancho / ancho
            factor_escala_alto = nuevo_alto / alto
            contorno_transformado = contornos_filtradosObjeto[0] * [factor_escala_ancho, factor_escala_alto]
            punto_mas_bajo_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 1].argmax()][0]))
            punto_mas_izquierda_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 0].argmin()][0]))
```

```
            punto_mas_derecha_nuevo = tuple(map(int, contorno_transformado[contorno_transformado[:, :, 0].argmax()][0]))
            x_bajo, y_bajo = punto_mas_bajo_nuevo
            x_izquierda, y_izquierda = punto_mas_izquierda_nuevo
            x_derecha, y_derecha = punto_mas_derecha_nuevo
         else:
            y_bajo=0
            x_izquierda=0
            x_derecha=0
      except Exception as e:
         y_bajo=0
         x_izquierda=0
         x_derecha=0
      y_bajo=(math.floor(y_bajo / 10) * 10)-8
      x_izquierda=(math.floor(x_izquierda / 10) * 10 )
      x_derecha=(math.ceil(x_derecha / 10) * 10)
      actualizarParametros(70,0,180,x_izquierda,x_derecha,y_bajo)
      cambiarEstado()
   actualizarPorcentaje(20)
image1 = Image.open('prueba1.jpg')
def obtener_codigo_html(url):
   try:
      response = requests.get(url)
      if response.status_code == 200:
         return response.text
      else:
         print(f"Error al obtener el código HTML. Código de estado: {response.status_code}")
         return None
   except requests.exceptions.RequestException as e:
      print(f"Error al hacer la solicitud HTTP: {e}")
      return None
def clima():fuera esta funcion
   global labelClima
   global labelTiempo
   global labelHumedadActual
   global labelLluvia
   global labelViento
   while True:
      url_pagina = 'https://www.msn.com/es-xl/el-tiempo/pronostico/in-Ambato,Tungurahua?loc=eyJsIjoiQW1iYXRvIiwiciI6IlR1bmd1cmFodWEiLCJjIjoiRWN1YWRvciIsImkiOiJFQyIsImciOiJlcy14bCIsIngiOiItNzguNTkwNTYxNzMiLCJ5IjoiLTEuMjQyMjQyMjQyMTUifQ%3D%3D&weadegreetype=C&ocid=msedgntp&cvid=5f1f209fdbd94174aa01b9412f0be61f&content=TeaserTempRecord_wxnwtsrec_greeting'
      codigo_html = obtener_codigo_html(url_pagina)
      tiempo="_"
      temperatura="0"
      humedad="0"
      lluvia="0"
      viento="0"
      if codigo_html:
         soup = BeautifulSoup(codigo_html, 'html.parser')
         elemento = soup.find('div', id='CurrentWeatherFeedback')
         if elemento:
            valor_numerico = elemento.get_text().split(' ')[0]
            temperatura=valor_numerico
            labelClima.config(text=f"{valor_numerico}")
         elemento = soup.find('div', id='OverviewCurrentTemperature')
         if elemento:
            elemento = soup.find('div', class_='u1SummaryCaptionCompact-E1_1')
            if elemento:
               descripcion = elemento.get_text(strip=True)
               tiempo=descripcion
               labelTiempo.config(text=f"{descripcion}")
            else:
               print("Clima: N/A")
         else:
```

```
            print("Precipitaciones: N/A")
        elemento = soup.find('div', id='CurrentDetailLineHumidityValue')
        if elemento:
            valHumedad = elemento.get_text().split(' ')[0]
            humedad=valHumedad
            labelHumedadActual.config(text=f"Humedad Ambiental: {valHumedad}")
        elemento = soup.find('div', id='ForecastDays')
        if elemento:
            elemento = soup.find('div', class_='precipitationV3-E1_1')
            if elemento:
                valLluvia = elemento.get_text(strip=True)
                lluvia=valLluvia
                labelLluvia.config(text=f"Probabilidad de Lluvia: {valLluvia}")
            else:
                print("Lluvia: N/A")
        elemento = soup.find('div', id='CurrentDetailLineWindValue')
        if elemento:
            valViento = elemento.get_text().split(' ')[0]
            viento=valViento
            labelViento.config(text=f"Velocidad del Viento: {valViento} Km/h")
        actualizarClima(tiempo,temperatura,humedad,lluvia,viento)
    time.sleep(4)
def abrirPrincipal():
    global entryInicioUsuario
    global entryInicioContrasena
    usuario = entryInicioUsuario.get()
    contrasena = entryInicioContrasena.get()
    controlIS = iniciarSesion(usuario, contrasena)
    if 'Si' in controlIS.get('mensaje', ''):
        hilo_proceso = threading.Thread(target=proceso)
        hilo_proceso.start()
        valoresClima = threading.Thread(target=clima)
        valoresClima.start()
        datos.state(newstate="zoomed")
        ventana.state(newstate="withdraw")
    else:
        messagebox.showerror("Error", "Los datos son incorrectos")
def guardarDatos():
    global entryNombre
    global entryUsuario
    global entryContrasena
    global entryRepetirContrasena
    global entryEmail
    nombre = entryNombre.get()
    usuario = entryUsuario.get()
    contrasena = entryContrasena.get()
    confirmar = entryRepetirContrasena.get()
    email = entryEmail.get()
    if not nombre or not usuario or not contrasena or not confirmar or not email:
        messagebox.showerror("Error", "Todos los datos son obligatorios")
        return
    if contrasena != confirmar:
        messagebox.showerror("Error", "Las contraseñas no son iguales")
        return
    controlRegistro = registrarUsuario(nombre,usuario,email,contrasena)
    if 'Si' in controlRegistro.get('mensaje', ''):
        messagebox.showerror("Información", "Registro Correcto")
        abrirCaratula()
    else:
        messagebox.showerror("Error", controlRegistro.get('mensaje', ''))
def abrirRegistrar():
    ventana.state(newstate="withdraw")
    registrar.state(newstate="zoomed")
def abrirCaratula():
    registrar.state(newstate="withdraw")
    ventana.state(newstate="zoomed")
```

```
def proceso():
    global botonAutomatico
    global botonManual
    global botonIniciar
    global botonDetener
    global valorAutomatico
    global valorEstado
    global valorEstado
    global labelSensorHumedad
    global on_image_tk
    global off_image_tk
    global iniciarD_image_tk
    global detenerA_image_tk
    global labelSensorHumedadDescripcion
    global imgElectro
    global desativado_image_tk
    global ativado_image_tk
    global labelProceso
    while True:
        tDatos = traerProceso()
        a = np.array(json.loads(tDatos))
        for x in a:
            dato = x['proceso']
            datoA = x['estado']
            porcentaje = x['porcentaje']
            datoHumedad = x['humedad']
            electro = x['electro']
            labelSensorHumedad.config(text=f"Humedad de la Tierra: {datoHumedad} %")
            mensajeHumedad=obtener_mensaje(float(datoHumedad))
            labelSensorHumedadDescripcion.config(text=f"{mensajeHumedad}")
            labelProceso.config(text=f"{porcentaje} %")
            if (electro=="1"):
                imgElectro.config(image=ativado_image_tk)
            else:
                imgElectro.config(image=desativado_image_tk)
            if (datoA=="1"):
                valorAutomatico = 1
                botonAutomatico.config(image=on_image_tk)
                botonManual.config(image=off_image_tk)
            else:
                valorAutomatico = 0
                botonAutomatico.config(image=off_image_tk)
                botonManual.config(image=on_image_tk)
            if int(dato) >= 1:
                valorEstado = 1
                botonIniciar.config(image=iniciarD_image_tk)
                botonDetener.config(image=detenerA_image_tk)
                botonIniciar.configure(state='disabled')
                botonDetener.configure(state='normal')
                if int(dato) == 1:
                    capturarImagen()
                if int(dato) == 2:
                    ventanaPrincipal()
            else:
                valorEstado = 0
                actualizarPorcentaje(0)
                botonIniciar.config(image=iniciarA_image_tk)
                botonDetener.config(image=detenerD_image_tk)
                botonIniciar.configure(state='normal')
                botonDetener.configure(state='disabled')
        time.sleep(3)
def obtener_mensaje(datoHumedad):
    global imgVerde
    global imgAmarillo
    global imgNaranja
    global imgRojo
```

```
    if datoHumedad > float(40):
        imgVerde.configure(state='normal')
        imgAmarillo.configure(state='disabled')
        imgNaranja.configure(state='disabled')
        imgRojo.configure(state='disabled')
        return "Nivel de humedad alto. No es necesario regar."
    elif datoHumedad > float(30):
        imgVerde.configure(state='disabled')
        imgAmarillo.configure(state='normal')
        imgNaranja.configure(state='disabled')
        imgRojo.configure(state='disabled')
        return "Nivel de humedad moderado. Puede esperar para regar."
    elif datoHumedad > float(10):
        imgVerde.configure(state='disabled')
        imgAmarillo.configure(state='disabled')
        imgNaranja.configure(state='normal')
        imgRojo.configure(state='disabled')
        return "Nivel de humedad bajo. Se considera regar."
    elif datoHumedad >= float(0):
        imgVerde.configure(state='disabled')
        imgAmarillo.configure(state='disabled')
        imgNaranja.configure(state='disabled')
        imgRojo.configure(state='normal')
        return "Hora de regar. El nivel de humedad es muy bajo."
    else:
        return "Error: El valor del sensor de humedad no es válido."
subVen = tkinter.LabelFrame(ventana, text='')
subVen.place(relx=0, rely=0, relwidth=1, relheight=1)
image_path = 'portada.png'
image = Image.open(image_path)
widthIm, heightIm = ventana.winfo_screenwidth(), ventana.winfo_screenheight()
resized_image = image.resize((widthIm, heightIm))
tk_image = ImageTk.PhotoImage(resized_image)
image_reg = 'portadaReg.png'
imageReg = Image.open(image_reg)
widthReg, heightReg = registrar.winfo_screenwidth(), registrar.winfo_screenheight()
resized_imageReg = imageReg.resize((widthReg, heightReg))
tk_image_reg = ImageTk.PhotoImage(resized_imageReg)
imgLogoReg = tkinter.Label(registrar, image=tk_image_reg, bg='#000')
imgLogoReg.place(x=0, y=0, relwidth=1, relheight=1)
entryInicioUsuario = tkinter.Entry(ventana, font=('Arial', 16))
entryInicioUsuario.place(relx=0.5, rely=0.45, anchor='center', relwidth=0.4, height=40)
entryInicioContrasena = tkinter.Entry(ventana, font=('Arial', 16), show='*')
entryInicioContrasena.place(relx=0.5, rely=0.55, anchor='center', relwidth=0.4, height=40)
labelUsuario = tkinter.Label(
    ventana, text="Usuario:", font=('Times 15'))
labelUsuario.place(relx=0.2, rely=0.42, relwidth=0.1, height=40)
labelContra = tkinter.Label(
    ventana, text="Contraseña:", font=('Times 15'))
labelContra.place(relx=0.2, rely=0.52, relwidth=0.1, height=40)
entryNombre = tkinter.Entry(registrar, font=('Arial', 16))
entryNombre.place(relx=0.5, rely=0.35, anchor='center', relwidth=0.4, height=40)
entryUsuario = tkinter.Entry(registrar, font=('Arial', 16))
entryUsuario.place(relx=0.5, rely=0.45, anchor='center', relwidth=0.4, height=40)
entryEmail = tkinter.Entry(registrar, font=('Arial', 16))
entryEmail.place(relx=0.5, rely=0.55, anchor='center', relwidth=0.4, height=40)
entryContrasena = tkinter.Entry(registrar, font=('Arial', 16), show='*')
entryContrasena.place(relx=0.5, rely=0.65, anchor='center', relwidth=0.4, height=40)
entryRepetirContrasena = tkinter.Entry(registrar, font=('Arial', 16), show='*')
entryRepetirContrasena.place(relx=0.53, rely=0.75, anchor='center', relwidth=0.35, height=40)
labelNombre = tkinter.Label(
    registrar, text="Nombre:", font=('Times 15'))
labelNombre.place(relx=0.2, rely=0.32, relwidth=0.1, height=40)
labelUsuario = tkinter.Label(
    registrar, text="Usuario:", font=('Times 15'))
labelUsuario.place(relx=0.2, rely=0.42, relwidth=0.1, height=40)
```

```
labelEmail = tkinter.Label(
   registrar, text="Email:", font=('Times 15'))
labelEmail.place(relx=0.2, rely=0.52, relwidth=0.1, height=40)
labelContra = tkinter.Label(
   registrar, text="Contraseña:", font=('Times 15'))
labelContra.place(relx=0.2, rely=0.62, relwidth=0.1, height=40)
labelContraRep = tkinter.Label(
   registrar, text="Repetir Contraseña:", font=('Times 15'))
labelContraRep.place(relx=0.2, rely=0.72, relwidth=0.15, height=40)
imgLogo = tkinter.Label(subVen, image=tk_image, bg='#000')
imgLogo.place(x=0, y=0, relwidth=1, relheight=1)
btnContinuar = tkinter.Button(ventana, text="Ingresar", command=abrirPrincipal, fg="#000000", bg="#ffffff")
btnContinuar.place(relx=0.5, rely=0.8, anchor='s', relwidth=0.2, height=40)
btnContinuar['font'] = font.Font(size=18, weight="bold")
btnRegistrar = tkinter.Button(ventana, text="Registrar", command=abrirRegistrar, fg="#000000", bg="#ffffff")
btnRegistrar.place(relx=0.5, rely=0.9, anchor='s', relwidth=0.2, height=40)
btnRegistrar['font'] = font.Font(size=18, weight="bold")
btnCrear = tkinter.Button(registrar, text="Registrar", command=guardarDatos, fg="#000000", bg="#ffffff")
btnCrear.place(relx=0.5, rely=0.87, anchor='s', relwidth=0.2, height=40)
btnCrear['font'] = font.Font(size=18, weight="bold")
btnSalir = tkinter.Button(registrar, text="Cancelar", command=abrirCaratula, fg="#000000", bg="#ffffff")
btnSalir.place(relx=0.5, rely=0.95, anchor='s', relwidth=0.2, height=40)
btnSalir['font'] = font.Font(size=18, weight="bold")
frameClima = tkinter.LabelFrame(datos, text='Clima', bg='#d9d9d9')
frameClima.place(x=25, y=22, width=210, height=280)
labelTiempo = tkinter.Label(
   frameClima, text="Muy Nublado", font=('Times 12'), bg='#d9d9d9', fg='black')
labelTiempo.place(x=10, y=10, width=180, height=40)
labelClima = tkinter.Label(
   frameClima, text="25°C", font=('Times 40'), bg='#d9d9d9', fg='black')
labelClima.place(x=10, y=45, width=180, height=80)
labelHumedadActual = tkinter.Label(
   frameClima, text="Humedad Ambiental: 0", font=('Times 12'), bg='#d9d9d9', fg='black', anchor='w')
labelHumedadActual.place(x=10, y=120, width=180, height=40)
labelLluvia = tkinter.Label(
   frameClima, text="Probabilidad de Lluvia: 0", font=('Times 12'), bg='#d9d9d9', fg='black', anchor='w')
labelLluvia.place(x=10, y=170, width=180, height=40)
labelViento = tkinter.Label(
   frameClima, text="Velocidad del Viento: 0", font=('Times 10'), bg='#d9d9d9', fg='black', anchor='w')
labelViento.place(x=10, y=220, width=180, height=40)
frameControl = tkinter.LabelFrame(datos, text='Control', bg='#d9d9d9')
frameControl.place(x=250, y=22, width=280, height=280)
labelAutomatico = tkinter.Label(
   frameControl, text="Control Automático", font=('Times 12'), bg='#d9d9d9', fg='black', anchor='w')
labelAutomatico.place(x=10, y=5, width=150, height=40)
labelManual = tkinter.Label(
   frameControl, text="Control Manual", font=('Times 12'), bg='#d9d9d9', fg='black', anchor='w')
labelManual.place(x=10, y=60, width=150, height=40)
labelElectro = tkinter.Label(
   frameControl, text="Electroválvula", font=('Times 12'), bg='#d9d9d9', fg='black', anchor='w')
labelElectro.place(x=10, y=110, width=150, height=40)
labelDistanciaFondo = tkinter.Label(
   frameControl, text="Distancia Largo(m)", font=('Times 12'), bg='#d9d9d9', fg='black', anchor='w')
labelDistanciaFondo.place(x=10, y=160, width=150, height=40)
valor_predeterminado = tkinter.StringVar(value="9")
entryDistancia = tkinter.Entry(frameControl, font=('Arial', 12), textvariable=valor_predeterminado)
entryDistancia.place(x=200, y=180, anchor='center', width=90, height=35)
on_image = Image.open("on.png")
off_image = Image.open("off.png")
activado_image = Image.open("activado.png")
desactivado_image = Image.open("desactivado.png")
iniciarA_image = Image.open("iniciarA.png")
iniciarD_image = Image.open("iniciarD.png")
detenerA_image = Image.open("detenerA.png")
detenerD_image = Image.open("detenerD.png")
width, height = 90, 35
```

```
on_image = on_image.resize((width, height))
off_image = off_image.resize((width, height))
activado_image = activado_image.resize((width, height))
desactivado_image = desactivado_image.resize((width, height))
iniciarA_image = iniciarA_image.resize((width, height))
iniciarD_image = iniciarD_image.resize((width, height))
detenerA_image = detenerA_image.resize((width, height))
detenerD_image = detenerD_image.resize((width, height))
on_image_tk = ImageTk.PhotoImage(on_image)
off_image_tk = ImageTk.PhotoImage(off_image)
ativado_image_tk = ImageTk.PhotoImage(activado_image)
desativado_image_tk = ImageTk.PhotoImage(desactivado_image)
iniciarA_image_tk = ImageTk.PhotoImage(iniciarA_image)
iniciarD_image_tk = ImageTk.PhotoImage(iniciarD_image)
detenerA_image_tk = ImageTk.PhotoImage(detenerA_image)
detenerD_image_tk = ImageTk.PhotoImage(detenerD_image)
botonAutomatico = tkinter.Button(frameControl, image=on_image_tk,command=cambiarAutomatico, bd=0,
bg="#d9d9d9")
botonAutomatico.place(x=150, y=10, width=width, height=height)
botonManual = tkinter.Button(frameControl, image=off_image_tk,command=cambiarAutomatico, bd=0,
bg="#d9d9d9")
botonManual.place(x=150, y=60, width=width, height=height)
imgElectro = tkinter.Label(frameControl, image=ativado_image_tk, bg='#d9d9d9')
imgElectro.place(x=150, y=110, width=width, height=height)
botonIniciar = tkinter.Button(frameControl, image=iniciarA_image_tk,command=cambiarEstado, bd=0,
bg="#d9d9d9")
botonIniciar.place(x=20, y=220, width=width, height=height)
botonDetener = tkinter.Button(frameControl, image=detenerD_image_tk, command=actualizarEstado, bd=0,
bg="#d9d9d9")
botonDetener.place(x=150, y=220, width=width, height=height)
botonDetener.configure(state='disabled')
frameSuelo = tkinter.LabelFrame(datos, text='Suelo', bg='#d9d9d9')
frameSuelo.place(x=25, y=310, width=400, height=120)
verde_image = Image.open("verde.png")
amarillo_image = Image.open("amarillo.png")
naranja_image = Image.open("naranja.png")
rojo_image = Image.open("rojo.png")
width, height = 30, 30
verde_image = verde_image.resize((width, height))
amarillo_image = amarillo_image.resize((width, height))
naranja_image = naranja_image.resize((width, height))
rojo_image = rojo_image.resize((width, height))
verde_image_tk = ImageTk.PhotoImage(verde_image)
amarillo_image_tk = ImageTk.PhotoImage(amarillo_image)
naranja_image_tk = ImageTk.PhotoImage(naranja_image)
rojo_image_tk = ImageTk.PhotoImage(rojo_image)
labelSensorHumedad = tkinter.Label(
    frameSuelo, text="Humedad de la Tierra: 15 %", font=('Times 12'), bg='#d9d9d9', fg='black')
labelSensorHumedad.place(x=10, y=5, width=350, height=40)
labelSensorHumedadDescripcion = tkinter.Label(
    frameSuelo, text="Nivel de humedad bajo. Se considera regar.", font=('Times 10'), bg='#d9d9d9', fg='black')
labelSensorHumedadDescripcion.place(x=10, y=30, width=350, height=40)
imgVerde = tkinter.Label(frameSuelo, image=verde_image_tk, bg='#d9d9d9')
imgVerde.place(x=120, y=65, width=width, height=height)
imgAmarillo = tkinter.Label(frameSuelo, image=amarillo_image_tk, bg='#d9d9d9')
imgAmarillo.place(x=160, y=65, width=width, height=height)
imgNaranja = tkinter.Label(frameSuelo, image=naranja_image_tk, bg='#d9d9d9')
imgNaranja.place(x=200, y=65, width=width, height=height)
imgRojo = tkinter.Label(frameSuelo, image=rojo_image_tk, bg='#d9d9d9')
imgRojo.place(x=240, y=65, width=width, height=height)
imgAmarillo.configure(state='disabled')
imgNaranja.configure(state='disabled')
imgRojo.configure(state='disabled')
frameProceso = tkinter.LabelFrame(datos, text='Proceso', bg='#d9d9d9')
frameProceso.place(x=430, y=310, width=110, height=120)
labelProceso = tkinter.Label(
```

```
    frameProceso, text="0 %", font=('Times 19'), bg='#d9d9d9', fg='black')
labelProceso.place(x=10, y=20, width=80, height=40)
frameCapturada = tkinter.LabelFrame(datos, text='Imagen Capturada', bg='#d9d9d9')
frameCapturada.place(x=550, y=22, width=360, height=280)
frameDeteccion = tkinter.LabelFrame(datos, text='Detección', bg='#d9d9d9')
frameDeteccion.place(x=930, y=22, width=360, height=280)
img1 = image1.resize((350, 200))
imgCam1 = ImageTk.PhotoImage(img1)
lblImgCam1 = tkinter.Label(frameCapturada, image=imgCam1, bg="#d9d9d9")
lblImgCam1.place(x=5, y=5, width=340, height=250)
img2 = image1.resize((350, 200))
imgCam2 = ImageTk.PhotoImage(img2)
lblImgCam2 = tkinter.Label(frameDeteccion, image=imgCam2, bg="#d9d9d9")
lblImgCam2.place(x=5, y=5, width=340, height=250)
frameContorno = tkinter.LabelFrame(datos, text='Contorno Césped', bg='#d9d9d9')
frameContorno.place(x=150, y=470, width=320, height=230)
frameObjeto = tkinter.LabelFrame(datos, text='Detección de Objeto', bg='#d9d9d9')
frameObjeto.place(x=550, y=470, width=320, height=230)
frameSeparacion = tkinter.LabelFrame(datos, text='Contorno Objeto', bg='#d9d9d9')
frameSeparacion.place(x=900, y=470, width=320, height=230)
img4 = image1.resize((300, 200))
imgCam4 = ImageTk.PhotoImage(img4)
lblImgCam4 = tkinter.Label(frameContorno, image=imgCam4, bg="#d9d9d9")
lblImgCam4.place(x=5, y=5, width=300, height=200)
img3 = image1.resize((300, 200))
imgCam3 = ImageTk.PhotoImage(img3)
lblImgCam3 = tkinter.Label(frameObjeto, image=imgCam3, bg="#d9d9d9")
lblImgCam3.place(x=5, y=5, width=300, height=200)
img5 = image1.resize((300, 200))
imgCam5 = ImageTk.PhotoImage(img5)
lblImgCam5 = tkinter.Label(frameSeparacion, image=imgCam5, bg="#d9d9d9")
lblImgCam5.place(x=5, y=5, width=300, height=200)
ventana.mainloop()
```

Anexo I. Código de control de la ESP32

```
#include <WiFi.h>
#include <ArduinoJson.h>
#include <HTTPClient.h>
#include <base64.h>
#include <Wire.h>
#include <ESP32Servo.h>
const char* ssid = "RED_MARTINEZ_Plus";
const char* password = "12041996";
String ip = "http://192.168.214.124/sistema/datos";
String phpUrl = ip + "/procesoEsp.php";
String phpUrlEnviar = ip + "/cambiarestado.php";
String phpUrlEnviarHumedad = ip + "/actualizarHumedad.php";
String phpUrlEnviarEstado = ip + "/actualizarEstadoEsp.php";
String phpUrlEnviarPorcentaje = ip + "/actualizarPorcentajeE.php";
const int sensorPin1 = 33;
#define SERVO_PINH 2
#define SERVO_PINV 4
Servo servoMotorH;
Servo servoMotorV;
int procesoge;
int anguloH;
int anguloV;
int valorMaxX;
int valorMaxY;
int valorMinX;
int humedad;
int valorSensor1;
int anguloMaximoOriginal;
int anguloMaximoH;
int anguloMinimoH;
int anguloMaximo;
```

```
int anguloMinimoHOriginal;
int electro=0;
int distanciaV=9;
int distanciaH=4;
int originalAV=80;
int anguloTotal=0;

void setup() {
  Serial.begin(115200);
  Serial.println();
  delay(5000);
  digitalWrite(22, HIGH);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connectando a WiFi...");
  }
  Serial.println("WiFi Contectado");
  pinMode(sensorPin1, INPUT_PULLDOWN);
  servoMotorH.attach(SERVO_PINH,500,2500);
  servoMotorV.attach(SERVO_PINV,500,2500);
  servoMotorH.write(0);
  delay(3000);
  servoMotorV.write(0);
  pinMode(22, OUTPUT);
  digitalWrite(22, HIGH);
  getValor();
  electro=0;
}
void loop() {
  getValor();
  actualizarHumedad();
  if(procesoge==3){
    regreCero();
    if(anguloH==0 && anguloV==0){
      int nuevoH=0;
      int nuevoV=0;
      int puntoVA=0;
      int puntoHA=0;
      int puntoHAD=0;
      actualizarPorcentaje(30);
      puntoHA=round(90-((90*(distanciaH/2))/4));
      puntoHAD=round(180-(90-((90*(distanciaH/2))/4)));
      int puntoHA2=round(90+((90*(distanciaH))/4));
      anguloTotal=round((originalAV*distanciaV)/9);
      puntoVA=20;
      if(0<valorMinX || 0>valorMaxX){
      }else{
         if(puntoVA>= valorMaxY){
           puntoVA=valorMaxY;
         }
      }
      if(puntoHA>=0){
        puntoVA=round(((distanciaH/2)*puntoVA)/4);
      }
      getValor();
      if(procesoge==3){
        if((distanciaH/2)==4){
          digitalWrite(22, LOW);
          regarPunto(0,puntoVA);
          actualizarPorcentaje(40);
          electro=1;
          actualizarHumedad();
        }
      }
      puntoVA=40;
```

```
if(20<valorMinX || 20>valorMaxX){
}else{
  if(puntoVA>= valorMaxY){
    puntoVA=valorMaxY;
  }
}
if(puntoHA>=20){
  puntoVA=round(((distanciaH/2)*puntoVA)/4)-10;
}
getValor();
if(procesoge==3){
  if((distanciaH/2)==4){
    regarPunto(20,puntoVA);
    actualizarPorcentaje(45);
    digitalWrite(22, LOW);
    electro=1;
    actualizarHumedad();
  }
}
puntoVA=60;
if(40<valorMinX || 40>valorMaxX){
}else{
  if(puntoVA>= valorMaxY){
    puntoVA=valorMaxY;
  }
}
if(puntoHA>=40){
  puntoVA=round(((distanciaH/2)*puntoVA)/4)-10;
}
getValor();
if(procesoge==3){
  if((distanciaH/2)==4){
    regarPunto(40,puntoVA);
    actualizarPorcentaje(50);
    digitalWrite(22, LOW);
    electro=1;
    actualizarHumedad();
  }
}
puntoVA=70;
if(50<valorMinX || 50>valorMaxX){
}else{
  if(puntoVA>= valorMaxY){
    puntoVA=valorMaxY;
  }
}
if(puntoHA>=50){
  puntoVA=round(((distanciaH/2)*puntoVA)/4)-30;
}
if((distanciaH/2)<4){
  puntoVA=puntoVA-50;
}
getValor();
if(procesoge==3){
    regarPunto(50,0);
    digitalWrite(22, LOW);
    regarPunto(50,puntoVA);
    actualizarPorcentaje(55);
    electro=1;
    actualizarHumedad();
}
puntoVA=80;
if(60>=puntoHA){
  if(60<valorMinX || 60>valorMaxX){
  }else{
    if(puntoVA>= valorMaxY){
```

```
      puntoVA=valorMaxY;
    }
  }
  if((distanciaH/2)<4){
    puntoVA=puntoVA-50;
  }
  getValor();
  if(procesoge==3){
    /llamar a la funcion para iniciar riego, dando como parámetros posiciones máximos en servo
    regarPunto(60,puntoVA);
    actualizarPorcentaje(60);
    electro=1;
    actualizarHumedad();
  }
}
puntoVA=70;
if(70>=puntoHA){
  if(70<valorMinX || 70>valorMaxX){
  }else{
    if(puntoVA>= valorMaxY){
      puntoVA=valorMaxY;
    }
  }
   if((distanciaH/2)<4){
    puntoVA=puntoVA-30;
  }
  getValor();
  if(procesoge==3){
    /llamar a la funcion para iniciar riego, dando como parámetros posiciones máximos en servo
    regarPunto(70,puntoVA);
    actualizarPorcentaje(70);
    electro=1;
    actualizarHumedad();
  }

}
puntoVA=80;
if(80>=puntoHA){
  if(80<valorMinX || 80>valorMaxX){
  }else{
    if(puntoVA>= valorMaxY){
      puntoVA=valorMaxY;
    }
  }
  if((distanciaH/2)<4){
    puntoVA=puntoVA-10;
  }
  /llamar a la funcion getValor que traerá valores de los parámetros principales desde el servidor
  getValor();
  if(procesoge==3){
    regarPunto(80,puntoVA);
    actualizarPorcentaje(75);
    electro=1;
    actualizarHumedad();
  }
}
puntoVA=80;
if(90<valorMinX || 90>valorMaxX){
}else{
    if(puntoVA>= valorMaxY){
      puntoVA=valorMaxY;
    }
  }
  if((distanciaH/2)<4){
    puntoVA=puntoVA-10;
  }
```

```
  getValor();
  if(procesoge==3){
    regarPunto(90,puntoVA);
    actualizarPorcentaje(78);
    electro=1;
    actualizarHumedad();
  }
puntoVA=80;
if(110<valorMinX || 110>valorMaxX){
}else{
  if(puntoVA>= valorMaxY){
    puntoVA=valorMaxY;
  }
}
if(puntoHAD<=110){
      puntoVA=puntoVA-10;
}
  if((distanciaH/2)<4){
    puntoVA=puntoVA-40;
  }
  getValor();
  if(procesoge==3){
    regarPunto(110,puntoVA);
    actualizarPorcentaje(80);
    electro=1;
    actualizarHumedad();
  }
puntoVA=70;
if(120<=puntoHA2){
  if(120<valorMinX || 120>valorMaxX){
    }else{
    if(puntoVA>= valorMaxY){
      puntoVA=valorMaxY;
    }
  }
  if(puntoHAD<=120){
      puntoVA=puntoVA-20;
  }
  if((distanciaH/2)<4){
    puntoVA=puntoVA-40;
  }
  getValor();
  if(procesoge==3){
    regarPunto(120,puntoVA);
    actualizarPorcentaje(83);
  }
}
puntoVA=60;
  if(140<valorMinX || 140>valorMaxX){
    }else{
    if(puntoVA>= valorMaxY){
      puntoVA=valorMaxY;
    }
  }
  if(puntoHAD<=140){
      puntoVA=round(((distanciaH/2)*puntoVA)/4);
  }
  if((distanciaH/2)<4){
    puntoVA=puntoVA-15;
  }
  getValor();
  if(procesoge==3){
    regarPunto(140,puntoVA);
    actualizarPorcentaje(85);
    electro=1;
    actualizarHumedad();
```

```
      }
     puntoVA=50;
     if(160<valorMinX || 160>valorMaxX){
     }else{
      if(puntoVA>= valorMaxY){
        puntoVA=valorMaxY;
      }
     }
     if(puntoHAD<=160){
      puntoVA=round(((distanciaH/2)*puntoVA)/4);
     }
     if((distanciaH/2)<4){
        puntoVA=puntoVA-30;
     }
      getValor();
     if(procesoge==3){
      if((distanciaH/2)==4){
        regarPunto(160,puntoVA);
        actualizarPorcentaje(90);
        electro=1;
        actualizarHumedad();
      }
     }
     puntoVA=10;
     if(180<valorMinX || 180>valorMaxX){
     }else{
      if(puntoVA>= valorMaxY){
        puntoVA=valorMaxY;
      }
     }
     if(puntoHAD<=180){
      puntoVA=round(((distanciaH/2)*puntoVA)/4)-5;
     }
     getValor();
     if(procesoge==3){
      if((distanciaH/2)==4){
        regarPunto(180,puntoVA);
        actualizarPorcentaje(95);
        electro=1;
        actualizarHumedad();
      }
     }
     electro=0;
     digitalWrite(22, HIGH);
     actualizarPorcentaje(98);
     actualizarEstado();
     regreCero();
     actualizarPorcentaje(0);
    }
  }else{
    digitalWrite(22, HIGH);
  }
  delay(1000);
}
void regreCero(){
  if(anguloH>0){
     regresarH(0);
    }
    if(anguloV>0){
     regresarV();
    }
    anguloH=0;
    anguloV=0;
}
void regarPunto( int valorHorizontal,int valorVertical){
  int proceso=valorVertical/10;
```

```
  int punto=0;
  Serial.println("punto nuevo horizontal: " + String(valorHorizontal) + ", vertical: " + String(valorVertical));
  subirH(valorHorizontal);
  subirV(valorVertical);
  regresarV();
}
void subirH(int nuevo){
  int pos=0;
  for(pos = anguloH; pos<=nuevo;pos+=1){
      servoMotorH.write(pos);
      delay(180);
    }
  anguloH=nuevo;
}
void subirV(int nuevo){
  int pos=0;
  for(pos = 0; pos<=nuevo;pos+=1){
    servoMotorV.write(pos);
    delay(180);
  }
  anguloV=nuevo;
}
void regresarH(int valor){
  int pos=0;
  for(pos = anguloH; pos>=valor;pos-=1){
      servoMotorH.write(pos);
      delay(180);
    }
  anguloH=valor;
}
void regresarV(){
  int pos=0;
  for(pos = anguloV; pos>=0;pos-=1){
    servoMotorV.write(pos);
    delay(180);
  }
  anguloV=0;
}
int getValor() {
  HTTPClient http;
  http.begin(phpUrl);
  int httpResponseCode = http.GET();
  if (httpResponseCode == 200) {
    String payload = http.getString();
    Serial.print(payload);
    int valor = parseJson(payload);
    http.end();
    return valor;
  } else {
    Serial.print("Error en la solicitud HTTP. Código de respuesta: ");
    Serial.println(httpResponseCode);
    return -1;
  }
}
int parseJson(String payload) {
  DynamicJsonDocument doc(1024);
  deserializeJson(doc, payload);
  estado = doc[0]["estado"];
  procesoge = doc[0]["proceso"];
  anguloMaximoOriginal = doc[0]["anguloMaximo"];
  anguloMaximoH = doc[0]["anguloMaximoH"];
  anguloMinimoH = doc[0]["anguloMinimoH"];
  valorMaxX = doc[0]["valorMaxX"];
  valorMinX = doc[0]["valorMinX"];
  valorMaxY = doc[0]["valorMaxY"];
  anguloMaximo=30;
```

```
  anguloMinimoHOriginal=anguloMinimoH;
   return procesoge;
}
void enviarEstado() {
  HTTPClient http;
  http.begin(phpUrlEnviar);
  http.addHeader("Content-Type", "application/json");
  String para = "";
  String jsonData = "{\"parametro\":\"" + para + "\"}";
 int httpResponseCode = http.POST(jsonData);
  if (httpResponseCode == 200) {
    String respuesta = http.getString();
    Serial.println("Respuesta del servidor: " + respuesta);
  } else {
    Serial.print("Error en la solicitud HTTP. Código de respuesta: ");
    Serial.println(httpResponseCode);
  }
  http.end();
}
void actualizarHumedad(){
  valorSensor1 = analogRead(sensorPin1);
  int porcentajeHumedad1 = map(valorSensor1, 0, 4000, 100, 0);
  humedad=porcentajeHumedad1;
  actualizarHumedadHTTP();
  if(estado==1){
    if(procesoge==0){
      if (porcentajeHumedad1 < 30) {
          enviarEstado();
      }
    }

  }
}
void actualizarHumedadHTTP() {
  HTTPClient http;
  http.begin(phpUrlEnviarHumedad);
  http.addHeader("Content-Type", "application/json");
  String jsonData = "{\"valor\":\"" + String(humedad) + "\", \"electro\":\"" + String(electro) + "\"}";
  int httpResponseCode = http.POST(jsonData);
  if (httpResponseCode == 200) {
    String respuesta = http.getString();
    Serial.println("Respuesta del servidor: " + respuesta);
  } else {
    Serial.print("Error en la solicitud HTTP. Código de respuesta: ");
    Serial.println(httpResponseCode);
  }
  http.end();

}
void actualizarEstado() {
  HTTPClient http;
  http.begin(phpUrlEnviarEstado);
  http.addHeader("Content-Type", "application/json");
  String jsonData = "{\"valor\":\"" + String("0") + "\"}";
  int httpResponseCode = http.POST(jsonData);
  if (httpResponseCode == 200) {
    String respuesta = http.getString();
    Serial.println("Respuesta del servidor: " + respuesta);
  } else {
    Serial.print("Error en la solicitud HTTP. Código de respuesta: ");
    Serial.println(httpResponseCode);
  }
  http.end();
}
void actualizarPorcentaje(int valor) {
  HTTPClient http;
```

```
 http.begin(phpUrlEnviarPorcentaje);
 http.addHeader("Content-Type", "application/json");
 String jsonData = "{\"valor\":\"" + String(valor) + "\"}";
 int httpResponseCode = http.POST(jsonData);
 if (httpResponseCode == 200) {
   String respuesta = http.getString();
   Serial.println("Respuesta del servidor: " + respuesta);
 } else {
   Serial.print("Error en la solicitud HTTP. Código de respuesta: ");
   Serial.println(httpResponseCode);
 }
 http.end();
}
```

Anexo J. Cálculo de consumo hídrico

Todos los cálculos a continuación realizados son en base a [9], [13] y [15]

Latitud de la ciudad de Ambato 1°14′, de acuerdo al porcentaje de horas diarias de sol, hemisferio sur para la latitud de Ambato, se tiene en el mes de diciembre 8,55 [9].

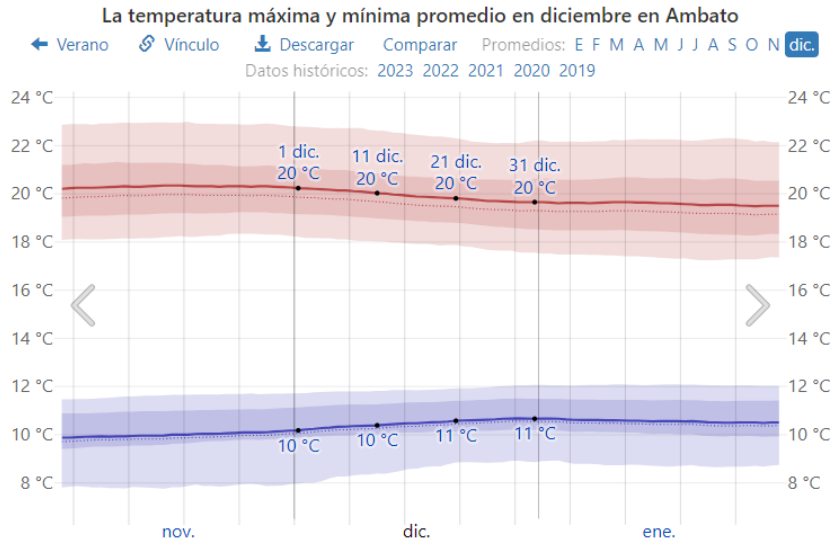


Figura J1. Temperatura del mes de diciembre 2023 [43]

En base a la Figura J1, se determina el factor promedio de temperatura media mensual de diciembre de 10,5 °C

Cálculo del factor de consumo del mes de diciembre

$$f(diciembre) = p(0{,}46\,T + 8{,}13) \qquad (1)$$

$$f(diciembre) = 8{,}55(0{,}46\,(10{,}5) + 8{,}13)$$

$$f(diciembre) = 110{,}80\,mm$$

Cálculo del coeficiente de temperatura del mes de diciembre

$$kt = 0{,}031144 \times T + 0{,}2396 \quad (2)$$

$$kt = 0{,}031144 \times (10{,}5) + 0{,}2396$$

$$kt = 0{,}56$$

Cálculo de la evapotranspiración referencia del cultivo

$$ETo = f \times Kt \quad (3)$$

$$ETo = 110{,}80 \times 0{,}56$$

$$ETo = 62{,}048\, mm$$

Siendo Kc el coeficiente de cultivo, para céspedes de estación fría como es el kikuyo suele tomar el valor de 0,8 [2].

Cálculo de la evapotranspiración real del cultivo

$$ETc = ETo \times Kc \quad (4)$$

$$ETc = 62{,}048 \times 0{,}8$$

$$ETc = 49{,}63\, mm$$

En la Figura J2, se presenta los valores de precipitación del mes de diciembre 2023 donde, se obtiene un promedio de 81,75 mm

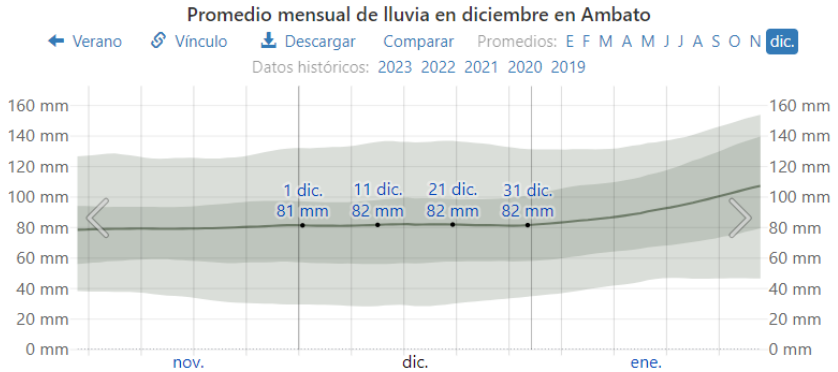


Figura J2. Precipitación mensual de diciembre 2023 [43]

Cálculo de la necesidad de riego

$$Nn = ETo - Pe \quad (5)$$

$$Nn = 110{,}80 - 81{,}75$$

$$Nn = 29{,}05\ mm/mes$$

En base a la necesidad de riego obtenida, se determina que para el mes de diciembre, se debe regar ocasionalmente debido a las precipitaciones constantes.

Cálculo del requerimiento bruto de agua

$$Rb = \frac{Nn}{Ea} \times 100 \quad (6)$$

Donde Ea, corresponde al porcentaje de eficiencia de aplicación del riego, en la Tabla 2, se presenta que el riego por aspersión tiene una eficiencia del 65% al 85%. Por lo cual Ea será el promedio de la eficiencia dando un valor del 75%

$$Rb = \frac{29{,}05}{75\%} \times 100$$

$$Rb = 3873{,}33\ mm/mes$$

Cálculo del módulo de riego del cultivo

$$MR = Rb \times 0{,}004 \quad (7)$$

$$MR = 3873{,}33 \times 0{,}004$$

$$MR = 15{,}49\ l/s/ha$$

Cálculo de caudal continuo

$$Qc = MR \times A \quad (8)$$

Donde A, es la superficie del cultivo en ha (hectareas), se requiere convertir el área de riego del sistema a m2 y posteriormente a hectáreas. Las dimensiones del área de jardín de prueba son de 9m x 8m = 72m2 = 0,0072 ha

$$Qc = MR \times A$$

$$Qc = 15{,}49 \times 0{,}0072$$

$$Qc = 0{,}111528\ l/s$$

Se concluye mediante los cálculos realizados que el consumo de agua es de 15,49 l/s/ha y que, para el área de prueba del sistema de riego, el caudal es de 0,111528 l/s.

Se necesita determinar la cantidad de litros utilizados durante los 6 minutos que toma realizar el riego en todo el jardín de prueba. Dado que 1 minuto equivale a 60 segundos, la cantidad de litros, se puede calcular multiplicando el consumo en l/s por el tiempo en segundos y por los minutos que dura el riego.

$$Caudal\ en\ \ l/min = Caudal \times Tiempo\ en\ seg \qquad (9)$$

$$Caudal\ en\ \ l/min\ \ = 0{,}111528\ l/s\ \times\ 60\ s$$

$$Caudal\ en\ l/min\ \ == 6{,}69168\ l/min$$

$$Caudal\ en\ \ litros = Caudal\ en\ l/min \times T\ riego \qquad (10)$$

$$Cantidad\ en\ litros\ \ = 6{,}69168\ l/min\ x\ 6\ minutos$$

$$Cantidad\ en\ litros\ \ = 40{,}15\ litros$$

Anexo K. Encuesta aplicada

UNIVERSIDAD TECNICA DE AMBATO

FACULTAD DE INGENIERIA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERIA EN TELECOMUNICACIONES

Encuesta desarrollada para el proyecto de investigación "SISTEMA DE RIEGO INTELIGENTE DE CORTO ALCANCE PARA JARDINES A PARTIR DE VISIÓN ARTIFICIAL".

Objetivo: Mediante esta encuesta, se pretende evaluar el nivel de usabilidad de una interfaz móvil para el monitoreo de un sistema de riego, en la cual la persona va a simular ser un usuario y brindara su punto de vista sobre la interfaz.

A continuación, seleccione su edad y género correspondiente.

Edad

• 18-25

• 26-33

• 34 en adelante

Genero

• Masculino

• Femenino

Cuestionario de preguntas

1. ¿Usaría el sistema con frecuencia?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

2. ¿Encuentra el sistema innecesariamente complejo?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

3. ¿La interfaz es fácil de usar?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

4. ¿Necesitaría el apoyo de otra persona para utilizar la interfaz?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

5. ¿Las funciones de la interfaz estaban bien integradas?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

6. ¿Considera que hay inconsistencias en esta interfaz?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

7. ¿Encuentra útiles los gráficos proporcionados?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

8. ¿Encuentra innecesaria la información de la interfaz?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

9. ¿Aprendería a utilizar esta interfaz con rapidez?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo

10. ¿Considera necesario conocer otros aspectos para utilizar la interfaz?

• Totalmente en desacuerdo

• En desacuerdo

• Neutro

• De acuerdo

• Totalmente de acuerdo