



UNIVERSIDAD TÉCNICA DE AMBATO

**FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E
INDUSTRIAL**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA Y
COMUNICACIONES**

TEMA:

“AUTÓMATA CONTROLADO POR IMPULSOS NEURONALES ALFA Y BETA
CON FPGA DE HARDWARE LIBRE”

Trabajo de Graduación. Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Electrónica y Comunicaciones

LINEAS DE INVESTIGACION: Física y Electrónica

AUTOR: Anabel Cristina Carrera Proaño

TUTOR: Ing. Víctor Santiago Manzano Villafuerte, Mg.

Ambato - Ecuador

Enero, 2017

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Investigación sobre el Tema: "AUTÓMATA CONTROLADO POR IMPULSOS NEURONALES, ALFA Y BETA CON FPGA DE HARDWARE LIBRE", de la señora Anabel Cristina Carrera Proaño estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato, Enero de 2017

EL TUTOR

Ing. Víctor Santiago Manzano Villafuerte, Mg

AUTORÍA

El presente Proyecto de Investigación titulado: AUTÓMATA CONTROLADO POR IMPULSOS NEURONALES ALFA Y BETA CON FPGA DE HARDWARE LIBRE. Es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, Enero de 2017

Anabel Cristina Carrera Proaño

CC: 1803551066

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato, Enero de 2017

Anabel Cristina Carrera Proaño

CC: 1803551066

APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ingenieros: Ing. Mg. Julio Enrique Cuji Rodríguez, Ing. Mg. Geovanni Brito Moncayo, revisó y aprobó el Informe Final del Proyecto de Investigación titulado AUTÓMATA CONTROLADO POR IMPULSOS NEURONALES ALFA Y BETA CON FPGA DE HARDWARE LIBRE, presentado por la señora Anabel Cristina Carrera Proaño de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ing. José Vicente Morales Lozada
PRESIDENTE DEL TRIBUNAL

Ing. Mg. Julio Enrique Cuji Rodríguez
DOCENTE CALIFICADOR

Ing. Mg. Geovanni Brito Moncayo
DOCENTE CALIFICADOR

DEDICATORIA

A mis padres, Gustavo Carrera y Amparito Proaño, al ser un ejemplo de amor, dedicación, educación, y esforzarse siempre por sus hijos para que no nos falte nunca nada.

A mi hermano, Christian Carrera por ser mi compañero de juegos, peleas, y mi inspiración para ser un buen ejemplo.

A mi esposo Alex Vargas, que me ha tomado de la mano, y me ha enseñado a no rendirme y que todo esfuerzo tiene su recompensa, es una bendición en mi vida.

A mis abuelitos, tíos, primos, por ser tan buenos conmigo, ser un apoyo y darme consejos cuando lo necesitaba.

A mis amigos y profesores de Tenis de Mesa, que han sido mi segunda familia, y me enseñaron que con perseverancia y disciplina todo es posible.

A mis compañeros, profesores, por ser los pilares fundamentales en mi educación académica.

Anabel Carrera Proaño

AGRADECIMIENTO

A mis padres, abuelitos y hermano, por inculcarme valores, disciplina, respeto y que la familia debe estar siempre unida.

A mi esposo, por su apoyo incondicional, su ayuda, su paciencia, y amor, ser mi compañero de vida y comprenderme siempre.

A la Ing. Pamela Espejo, por ser como una hermana, y ser quien me apoyó me escuchó y me enseñó lo necesario y lo bueno para cada etapa de mi vida.

A mis compañeros, en especial, Lizbeth, Daniel, Luis, David, Jairo, por su amistad, y ayuda en mi formación educativa.

A mi tutor, el Ingeniero Santiago Manzano, que a lo largo del presente proyecto me ha brindado su ayuda, su tiempo y mucha paciencia.

Anabel Carrera Proaño

INDICE

APROBACIÓN DEL TUTOR.....	ii
AUTORÍA.....	iii
DERECHOS DE AUTOR.....	iv
APROBACIÓN DE LA COMISIÓN CALIFICADORA	v
DEDICATORIA.....	vi
AGRADECIMIENTO.....	vii
INDICE	viii
RESUMEN	xiv
ABSTRACT.....	xv
GLOSARIO DE TÉRMINOS Y ACRÓNIMOS	xvi
INTRODUCCIÓN	xviii
CAPÍTULO 1 EL PROBLEMA	1
1.1. Tema	1
1.2. Planteamiento del problema.....	1
1.3. Delimitación	2
1.4. Justificación	3
1.5. Objetivos	4
1.5.1. Objetivo General.....	4
1.5.2. Objetivos Específicos	5
CAPÍTULO 2 MARCO TEÓRICO	6
2.1. Antecedentes investigativos	6
2.2. Fundamentación Teórica.....	9
2.2.1. Autómata	9
2.2.2 Comunicaciones Inalámbricas.....	12
2.2.3 Señales EEG(Electroencefalograficas).....	17

2.2.4	Adquisición de señales	20
2.2.5	Tarjetas de Adquisición.....	21
2.3.	Propuesta de Solución.....	29
CAPÍTULO 3 METODOLOGÍA		30
3.1.	Modalidad de la investigación.....	30
3.2.	Población y muestra.....	30
3.3.	Recolección de información.....	30
3.4.	Procesamiento y análisis de datos.....	31
3.5.	Desarrollo del Proyecto.....	31
CAPÍTULO 4 DESARROLLO DE LA PROPUESTA		33
4.1.	Análisis del proceso técnico de reconocimiento de ondas cerebrales	35
4.1.1.	Análisis de las ondas cerebrales, alfa y beta	37
4.3.	Diseño del control de un Autómata con un FPGA de hardware libre	47
4.3.1.	Adquisición de las señales neuronales.....	47
4.3.2.	Sistema de comunicación entre la adquisición de señales neuronales y el procesamiento de las mismas.	50
4.3.3.	Procesamiento de señales neuronales.....	56
4.3.4.	Sistema de comunicación entre el procesador de señales y autómata	74
4.3.5.	Receptor de señales	75
CAPÍTULO 5 CONCLUSIONES Y RECOMENDACIONES		82
5.1.	Conclusiones	82
5.2.	Recomendaciones	83
BIBLIOGRAFÍA Y REFERENCIAS		84
ANEXOS.....		89

INDICE DE TABLAS

Tabla 4.1 Análisis técnico comparativo entre diademas EEG.....	43
Tabla 4.2 Comparación entre Arduino y Mojo V3.....	44
Tabla 4.3 Análisis técnico entre tarjetas FPGA.....	45
Tabla 4.4 Comparación técnica de dispositivos de comunicación.....	46
Tabla 4.5 Significado led del MindWave.....	49
Tabla 4.6 Descripción de pines del Bluetooth HC-05 [54].....	51
Tabla 4.7 Conexión de pines entre el Módulo HC-05 y el conversor USB-TTL.....	53
Tabla 4.8 Comandos básicos de configuración.....	54
Tabla 4.9 Comandos para configurar la paridad entre el Bluetooth y MindWave.....	55
Tabla 4.10. Características de los usuarios.....	77
Tabla 4.11. Presupuesto.....	81

ÍNDICE DE FIGURAS

Figura 2.1 Clasificación de los robots por su generación [12].	10
Figura 2.2 Robots Inteligentes [13].	11
Figura 2.3 Clasificación de los robots por su lenguaje de programación [14].	12
Figura 2.4 Logo de bluetooth [20].	14
Figura 2.5 Formato de trama del protocolo UART para el envío de datos [22].	15
Figura 2.6 Conexión de periféricos mediante Wifi [24].	15
Figura 2.7 Modulo Xbee [27].	16
Figura 2.8 Electroencefalograma de contacto [30].	18
Figura 2.9 Electroencefalograma de malla [30].	18
Figura 2.10 Electroencefalograma Quirúrgico [30].	18
Figura 2.11 Ondas cerebrales [31].	19
Figura 2.12 Dispositivo de método de adquisición de señales invasivos [30].	20
Figura 2.13 Casco Mindwave de NeuroSky, instrumento no invasivo [33].	21
Figura 2.14 Esquema básico de una tarjeta de adquisición de datos [36].	22
Figura 2.15 Arquitectura PXI [36].	22
Figura 2.16 Sistema hibrido de adquisición de datos [36].	23
Figura 2.17 Arduino [38].	23
Figura 2.18 Rasberry Pi [40].	24
Figura 2.19 Esquema FPGA [42].	25
Figura 2.20 Estructura básica de una descripción VHDL [46].	27
Figura 2.21 Estructura de una descripción Verilog [48].	28
Figura 2.22 Estructura básica de una descripción de ABEL [49].	29
Figura 4.1 Esquema del sistema para el control del autómatas.	33
Figura 4.2 Esquema general del sistema de control.	34
Figura 4.3 Electroodos. [52].	36
Figura 4.4 Puntos Principales de la cabeza de acuerdo al sistema 10-20 [52].	36
Figura 4.5 Puntos en el cerebro donde se detectan las ondas cerebrales [51].	37
Figura 4.6 Ondas Alfa [51].	38
Figura 4.7 Lectura EEG, visualización de alfa en estado de relajación [53].	38
Figura 4.8 Ondas Beta [51].	39

Figura 4.9 Lectura EEG, visualización de beta en estado de concentración [53].....	39
Figura 4.10 Comparación de espectros [53]	40
Figura 4.11 Comparación de espectros [53]	41
Figura 4.12 Análisis de espectro [53]	42
Figura 4.13 Mojo vs Arduino	44
Figura 4.14 Partes externas de la diadema MindWave [53]	47
Figura 4.15 Componentes internos de la diadema.....	48
Figura 4.16 Vista del anverso y reverso del circuito Mindwave	48
Figura 4.17 Representación del espesor en el circuito Mindwave.....	48
Figura 4.18 Especificaciones del módulo bluetooth.....	50
Figura 4.19 Descripción de pines en Bluetooth HC-05	51
Figura 4.20 Conexión Módulo HC-05 y Conversor USB-TTL.....	52
Figura 4.21 Respuesta de IDE Arduino a comandos AT.....	53
Figura 4.22 Propiedades de MindWave Mobile	54
Figura 4.23 Comprobación de Pareo entre Bluetooth y MindWave.....	55
Figura 4.24 Mojo V3	56
Figura 4.25. Diagrama de bloques de la programación en la tarjeta FPGA	57
Figura 4.26. Diagrama de flujo de la lógica de programación.....	59
Figura 4.27 Esquema de las conexiones del sistema	60
Figura 4.28 Descripción de la programación en Mojo V3	61
Figura 4.29 Chip SPARTAN-6.....	61
Figura 4.30 Partes de la programación en mojo IDE.....	62
Figura 4.31 Entorno y descripción de los Constraints en mojo IDE	62
Figura 4.32 Entorno y descripción de los Components en mojo IDE	63
Figura 4.33 Entorno y descripción de los Source en mojo IDE	66
Figura 4.34 Chip ATmega32U4	70
Figura 4.35 Descripción de los pines de ATmega32U4	70
Figura 4.36 Descripción de los archivos en Arduino IDE	71
Figura 4.37 Leds que muestran las salidas y el modo de funcionamiento.....	74
Figura 4.38 Placa de control sin modificar Fuente: Investigadora	74
Figura 4.39 Modificación en señales de entrada.....	74
Figura 4.40 Modificación de la alimentación	75

Figura 4.41 Autómata con forma de araña [56].....	75
Figura 4.42 Diagrama del controlador de motor que se utiliza el autómata [56]	75
Figura 4.43 Colocación MindWave.....	76
Figura 4.44. Gráfico de datos obtenidos del funcionamiento en la persona 1.....	77
Figura 4.45. Prueba de funcionamiento y conexión con Labview en la persona 1.....	78
Figura 4.46. Gráfico de datos obtenidos del funcionamiento en la persona 5.....	79
Figura 4.47. Prueba de funcionamiento y conexión con Labview en la persona 5.....	80

RESUMEN

En la presente investigación se plantea el desarrollo de un sistema de comunicación encaminado al control de dispositivos por medio de ondas cerebrales. Posibilitando a los usuarios con limitaciones físicas y las que no las poseen, un apoyo para realizar actividades sin el uso de sus extremidades, ya sea por ocupación o por restricciones de terreno, clima, gravedad, etc., cumpliendo con las expectativas de funcionalidad previstas.

Para el cual se ha optado por el uso de dispositivos de bajo costo y necesarios para la adquisición y envío de señales entre las diferentes etapas del sistema con el ánimo de tener una comunicación confiable y de una velocidad de respuesta alta.

El control del autómatas se lo realizó con la diadema Mindwave para la adquisición de datos, a lo referente de señales neuronales, y el procesamiento con la tarjeta Mojo V3. Todo el sistema se une para implementar el control de un autómatas hexápodo por medio de infrarrojo, con la comodidad de ser inalámbrico y de fácil uso.

Palabras Claves: EEG, Mindwave, Neurosky, Mojo V3, FPGA, Ondas cerebrales

ABSTRACT

In the present investigation the United Nations Development Communication System aimed to control devices via brain waves were planted. Enabling a user's with physical and those that do not possess, support of the United Nations sin Perform activities using their limbs, and by sea or land occupancy restrictions, weather, gravity, etc., meeting the expectations Limitations Functionality provided.

For which we have opted for the use of low-cost devices and requirements for the acquisition and Sending signals between different stages of the System with the aim of having a reliable communication and high response speed.

PLC control is performed it with Mindwave For data acquisition a relation of neural signals, and Mojo Card Processing with V3 diadem. The whole system is bound to implement control of the United Nations PLC (spider) with the convenience of being Wireless and easy to use.

Keywords: EEG, Mindwave, Neurosky, Mojo V3, FPGA, brainwaves

GLOSARIO DE TÉRMINOS Y ACRÓNIMOS

- **ADC:** Analog to Digital Converter
- **Arduino:** Plataforma de hardware libre, constituida por una placa con un microcontrolador AVR con pines de entrada y salida y un entorno de desarrollo de libre acceso de baja complejidad de uso.
- **BCI:** Son las interfaces cerebro-computadora (Brain Computer Interfaces), representan una tecnología que se trata básicamente en la adquisición de las ondas cerebrales para que posteriormente puedan ser procesadas e interpretadas por un computador
- **BLE:** Bluetooth Low Energy
- **EEG:** Electroencefalograma
- **Hardware:** Conjunto de elementos o materiales físicos que constituyen un sistema electrónico informático
- **IDE:** Integrated Development Environment
- **Interfaz:** Medio que permite comunicar dos o más dispositivos o maquinas.
Extensión para la comunicación entre un sistema o subsistemas.
- **LED:** Light-emitting diode
- **Microcontrolador:** Circuito integrado programable, capaz de almacenar y ejecutar ordenes pres grabados en su memoria.
- **PCB:** Printed Circuit Board
- **Tiempo Real:** término utilizado para describir la presentación o reacción ante un evento generado en el momento de aparición del mismo.
- **Sensor:** dispositivo capaz de detectar magnitudes físicas o químicas y convertirlas en variables de naturaleza eléctrica.

- **SRAM:** Static Random Access Memory
- **TTL:** Transistor-Transistor Logic
- **UART:** Universal Asynchronous Receiver-Transmitter
- **USB:** Universal Serial Bus

INTRODUCCIÓN

En el presente proyecto se implementa un sistema de control para un autómata, construido con placas de hardware libre. Este prototipo electrónico opera mediante el uso de un sensor que proporciona información sobre las señales neuronales, una tarjeta de adquisición que ayuda a la recepción de ondas cerebrales y envío de señales hacia el control del autómata, presentando un sistema donde es posible obtener una medida inmediata de nuestros niveles de Atención y Meditación. A continuación se hace una breve descripción de los capítulos que componen esta investigación.

En el primer capítulo se describe el problema de investigación, explicando las causas y consecuencias que lo originan, y por último los objetivos de la investigación.

El segundo capítulo presenta los antecedentes y se realiza una introducción a todo el marco teórico necesario para el desarrollo del proyecto. Se trata de una forma general los conceptos relacionados con la robótica, lenguajes de programación, y etapas de adquisición de ondas neuronales.

El Tercer capítulo muestra las diferentes técnicas de investigación utilizadas, la forma de obtención y análisis de la información, además de los mecanismos que llevaron a la construcción del sistema para el control de dispositivos electrónicos.

El cuarto capítulo detalla paso a paso las etapas para la construcción del prototipo, además se describe las pruebas de funcionamiento del sistema completo.

En el Quinto capítulo se muestran las conclusiones y recomendaciones que se han podido extraer durante todo el desarrollo del sistema de control mediante impulsos neuronales para un autómata.

CAPÍTULO 1

EL PROBLEMA

1.1. Tema

Autómata controlado por impulsos neuronales alfa y beta con FPGA de hardware libre

1.2. Planteamiento del problema

A nivel mundial hoy en día la tecnología electrónica, no solo desarrolla sistemas electrónicos dirigidos con sensores que detectan diferentes cambios del medio que los rodea, sino que actualmente existen sistemas desarrollados con comunicación teledirigida por impulsos cerebrales de un ser humano quien emite información por medio de interfaces biométricas multisensoriales.

El profesor de la Escuela Politécnica de Lausana-Suiza, José Millán, especializado en exoesqueletos, ha desarrollado una silla de ruedas que es controlada por el pensamiento humano, el objetivo es controlar el avance, el retroceso y giro de la máquina. [1]

Las experiencias que se han realizado con algunas personas, a través de las señales emitidas por el cerebro con el uso de electrodos, han permitido crear autos inteligentes autónomos y con estos estudios se han demostrado que el control mental sobre un dispositivo, supondría proporcionar una libertad impensable a los pacientes medulares, ayudándoles a romper las cadenas fisiológicas que los retienen.

En el Ecuador existen investigaciones de neurotecnología, centradas a la orientación de personas que posean algún tipo de capacidad especial, buscando mejorar su movilidad. También se debe señalar que los dispositivos electrónicos neurotransmisores comenzaron

a utilizarse en el país con el objetivo de conocer ciertas preferencias de los clientes y su reacción frente a estimulaciones sensoriales. Realizando algunas demostraciones de las facultades aplicativas de esta neurotecnología se ha encontrado grandes aportaciones en el campo de la robótica, electrónica de consumo, marketing electrónico, juegos y sobre todo para mejorar la calidad de vida de las personas con capacidades especiales.

Las investigaciones de neurotecnología e interacción del pensamiento con una máquina no se enfoca simplemente a personas con capacidades especiales, sino se mira de una manera global, para no centrarse en un problema específico, se necesita valorar la posibilidad de interactuar con las señales cerebrales, haciendo que el pensamiento sea capaz de mover máquinas.

En la ciudad de Ambato, se cuenta con investigaciones de robots autónomos, tratando de perfeccionar las reacciones de los mismos o poder reemplazar algunas actividades que son realizados por los humanos, “pero en algunos campos (donde implica tomar decisiones, según los factores ambientales o sociales), más fácil es enseñar a una persona, que programar un autómatas” [2], por lo tanto el presente estudio surge ante la necesidad de buscar que los robots no sustituyan a los humanos sino que deben cooperar a los mismos, y ayudar con esta nueva tecnología en relación con el plan del buen vivir que busca mejorar la calidad de vida de la población.

1.3. Delimitación

Delimitación de Contenidos

Área académica: Física y Electrónica

Línea de investigación: Sistemas Electrónicos

Sublínea de investigación: Sistemas embebidos

Delimitación Espacial:

La presente investigación se desarrolló en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato.

Delimitación Temporal:

El proyecto se realizó en el transcurso del periodo académico Abril-Septiembre del 2016 después de su aprobación por parte del Honorable Consejo Directivo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

1.4. Justificación

La ejecución de una investigación encaminada a contribuir con un sueño tecnológico sencillamente es fabuloso, el control mental siempre ha sido el deseo de muchos seres humanos y también el obtener una libertad indispensable a los pacientes medulares, ayudándoles a interactuar con el mundo aunque posean cadenas fisiológicas que lo retengan. La interfaz perfecta entre máquina y humano, no se ha culminado, cada vez se realiza más investigaciones con respecto a este tema.

El interés de esta investigación es el diseño, desarrollo y construcción de un dispositivo electrónico encaminado a obtener diferentes señales emitidas por el cerebro, para posteriormente interpretarlas y crear una interfaz entre las señales neuronales emitidas por seres humanos y el control del autómata, cabe indicar que estas señales pueden variar según el estado de meditación y concentración de las personas (Señales alfa y beta).

La importancia de este proyecto, se enmarca en el apoyo que un dispositivo controlado por impulsos neuronales puede generar a personas que poseen inmovilidad en sus extremidades, la confianza de poder acercar un autómata y utilizarlo como un instrumento de interacción con su entorno. Para personas que no tienen estas limitaciones, ayudará como un apoyo para realizar actividades sin el uso de sus extremidades, ya sea por ocupación de las mismas o por restricciones de terreno, clima, gravedad, etc. Permitiendo el desarrollo de aplicaciones electrónicas que sin lugar a duda serán de gran utilidad en

áreas como la medicina, agricultura, medio ambiente, dispositivos electrónicos, entre otras.

El impacto social indiscutiblemente es importante, porque la sociedad de hoy exige numerosos retos y desafíos, en la actualidad las personas con capacidades especiales ya no pueden estar confinadas en sus hogares y apartadas de las actividades cotidianas, este control del autómeta les permitiría con la ayuda del mismo explorar nuevas formas de movilización e interacción por si solos; con todo esto se ponderaría el uso de la tecnología como una herramienta de inclusión, eliminando brechas de desigualdad que por décadas se han venido acumulando.

El beneficio de esta investigación es obtener información y conocimientos necesarios sobre la neurotecnología, para seguir experimentando y tener avances prometedores en el ámbito de la cooperación máquina-humano. Todos los seres se favorecen ya que se obtendrá una nueva posibilidad de interacción teniendo “una mano extra”, adaptando los dispositivos a las necesidades propias de cada persona otorgando la utilidad requerida.

En consecuencia, la presente investigación se enfoca en el control de un autómeta mediante ondas cerebrales obtenidas por medio de electrodos y señales emitidas por el cerebro, con el fin de poseer un elemento extra de nuestro cuerpo que cumpla órdenes con solo pensarlas y enfocarlas. Con el objetivo que en futuros proyectos, se pueda mejorar, aunque todavía el gran desafío en este caso sería que la interfaz cerebral ayudará de forma natural al movimiento del cuerpo.

1.5. Objetivos

1.5.1. Objetivo General

Controlar un Autómeta con FPGA de hardware libre por medio de ondas cerebrales, alfa y beta

1.5.2. Objetivos Específicos

- Analizar el proceso técnico de reconocimiento de ondas cerebrales, alfa y beta
- Determinar el medio de comunicación entre las ondas cerebrales, alfa y beta y el dispositivo electrónico
- Diseñar el control de un Autómata con un FPGA de hardware libre

CAPÍTULO 2

MARCO TEÓRICO

2.1. Antecedentes investigativos

Al realizar la investigación acerca de la tecnología EEG (Electro Encéfalo Grafía) mediante reconocimiento de impulsos, se han encontrado varios proyectos que poseen un grado de similitud, los cuales se detallan a continuación:

- En el 2014 en la Universidad Técnica de Ambato José Luis Varela desarrollo el proyecto denominado “Sistema de Control Automático para el Posicionamiento de una Silla de Ruedas Eléctrica”, el cual consiste en crear un sistema que permita el control del desplazamiento de una silla móvil de manera automática, de tal forma que el usuario pueda movilizarse sin ningún problema hacia un destino previamente programado, mediante algoritmos de control que tienen como objeto optimizar la movilidad del usuario. Además consta de un circuito de control en la silla de ruedas el cual permite monitorear y controlar las entradas de dicha silla desde un computador, teniendo como entradas de control la velocidad lineal y velocidad angular. Es por ello que el proyecto busca mejorar la vida de las personas que utilizan silla de ruedas ya sea por problemas de movilidad e incluso personas de la tercera edad [3].

- En el mismo año en la Universidad Católica de Santiago de Guayaquil Ángel Yaguana Hernández realizo el proyecto de titulación denominado “Desarrollo e implementación de una interfaz de comunicación que permita la interacción entre un usuario y las señales emitidas por sus ondas cerebrales usando un dispositivo de EEG de NeuroSky para controlar periféricos electrónicos” el cual consiste en la implementación de un sistema BCI (Interfaz Cerebro-Computadora), para ello se desarrolla una interfaz biométrica neurosensorial con el fin de controlar

periféricos electrónicos. Esto se logra mediante el módulo de comunicación TGAM1 de NeuroSky que se encarga de enlazar al dispositivo EEG colocado en un individuo a través de un firmware que actúa como un puente entre la BCI y el usuario final. Además este módulo lee, procesa y decodifica las señales emitidas por el cerebro de tal manera que se determina los estados mentales que este presenta, esto se demuestra mediante el uso de un hardware electrónico [4].

- Otro proyecto similar se realizó en el año 2015, en la Universidad Tecnológica Equinoccial por Edison Maila Andrango denominado “Prototipo de silla de ruedas controlada mediante señales eléctricas producidas por el cerebro (Electrograma)” para lo cual se desarrolló un sistema basado en una interfaz cerebro-computadora con el fin de controlar una silla de ruedas con el fin de ser utilizada por personas que presentan discapacidades o que hayan sufrido algún tipo de accidente. Para el proyecto se utilizó el equipo Emotiv® Neuroheadset EEG el cual fue enlazado hacia una computadora, para después ser controlada mediante una interfaz a través de la plataforma de Labview y algunas librerías de Arduino, dicha interfaz se encarga de controlar el prototipo mediante la utilización de una tarjeta de adquisición de datos de forma inalámbrica. Además se realizó el prototipo de la silla de ruedas y una tarjeta de potencia para controlar dos motores DC.

Para evaluar el sistema se utilizó indicadores de eficiencia, efectividad, eficacia y el índice de carga de trabajo que el usuario presenta al intentar mover el prototipo. Obteniendo como resultado una efectividad mayor al 85% e índice de carga de trabajo inferior al 50%, por lo que se determinó que la ejecución del sistema permite que el prototipo pueda ser controlado por los usuarios mediante señales eléctricas producidas por el cerebro [5].

En el exterior existen algunos trabajos relacionados acerca de la tecnología EEG, los cuales se muestran a continuación:

- El Grupo de Investigación de Inteligencia Artificial conformado por F. Tanco, C. Verrastro, D. Grinberg y J. Roitman de la Universidad Tecnológica Nacional de

Buenos Aires realizó el trabajo denominado “Implementación de redes neuronales artificiales en hardware para aplicación de detección automática de fulguraciones solares” el cual se basa en la descripción y comparación de dos modelos de neuronas artificiales de tal forma que permita implementar redes neuronales mediante circuitos digitales aplicadas a la detección automática de fulguraciones solares. Dicha implementación se lleva a cabo sobre dispositivos de lógica programable (FPGA), usando un esquema de entrenamiento off-line (mediante software), esto facilita la implementación debido a su capacidad de reprogramación [6].

- De igual forma en el año 2009 Eugenio Rodríguez de la empresa Toyota, realizó el proyecto “Sistema de detección cerebral que controla el movimiento de una silla de ruedas mediante la lectura de los pensamientos del usuario”, esto se consigue mediante tecnología BMI (Brain Machine Interface) la cual se basa en el reconocimiento y transformación de los patrones de las ondas cerebrales, de tal forma que se impulse la silla de ruedas hacia diversas direcciones, esto sin tener ningún retraso en el pensamiento y el movimiento debido a que el sistema de interfaz cerebral ofrece una respuesta de un pensamiento en tan solo 125 milésimas de segundo, lo que lo hace mucho más rápido que cualquier otra tecnología existente.

Dicho proceso se realiza mediante el uso de cinco sensores colocados encima de las regiones del cerebro que se encargan de interpretar los patrones de circulación de las señales generadas por el usuario. Además, el software se adapta a los patrones del pensamiento de cada persona, es por ello que posee una exactitud del 95% según las pruebas realizadas [7].

- En el año 2014 Yasin Munir, Dris Amrani e Hicham Buchiji, estudiantes de cuarto año de la Escuela Marroquí de Ciencias de Ingeniería de Rabat crearon un robot que se maneja con la vista llamado “Red Silence”, el cual se apoya en una pequeña plataforma rodante, dicho robot requiere el movimiento de los ojos o el parpadeo para su respectivo funcionamiento, con lo cual puede realizar tareas como: el cierre de ventanas, apagado de la luz o incluso de ciertas actividades relacionadas

con la alimentación. Además le permite al usuario estar en contacto con sus familiares a través de una opción de control conectada al internet [8].

- Otro proyecto similar fue realizado por la empresa OCZ, el cual permite controlar la PC con el poder de la mente, dicho dispositivo se llama “OCZ NIA”, este dispositivo consiste en una banda la cual se coloca alrededor de la cabeza, que se conecta a una pequeña caja que se encarga de decodificar las señales enviadas por la banda, y de igual manera enviar los datos hacia la computadora por medio del puerto USB. Es decir su funcionamiento se basa en la detección de señales eléctricas del cerebro y movimientos musculares de la cara y ojos, y enviando dichas señales a un procesador que se encarga de detectar patrones preconfigurados. Aunque por ahora está orientado a video-juegos, puede ser utilizado en personas con discapacidades que no pueden mover sus extremidades [9].

2.2. Fundamentación Teórica

2.2.1. Autómata

Se conoce como autómata al dispositivo capaz de funcionar en todo o casi todo por sus propios medios, debido a que posee un mecanismo que le permite realizar tareas y acciones de manera autosuficiente mediante la realización de determinados movimientos. De igual manera las máquinas son capaces de recrear e imitar los movimientos de un ser animado, tal es el caso de los robots, el cual posee un sistema electromecánico que le ayuda a realizar diferentes acciones previamente programadas en su memoria [10].

“Un robot está formado por elementos principales como son: transmisores, estructura mecánica y eléctrica, sistema de accionamiento, sistema sensorial, sistema de control y elementos terminales” [11].

Clasificación de los Autómatas

Uno de los factores que determinan la utilidad y flexibilidad de un robot es la potencia del software en el controlador, de acuerdo a este parámetro los robots han sido

clasificados de la siguiente manera: de acuerdo a su generación, lenguaje de programación, nivel de inteligencia, nivel de control y nivel de lenguaje de programación.

a. Generación. La generación de un robot se establece mediante el orden histórico de desarrollo en la robótica. Para lo cual se ha establecido cinco generaciones que normalmente son asignadas a los robots, la cual se muestra en la figura 2.1 que se muestra a continuación:

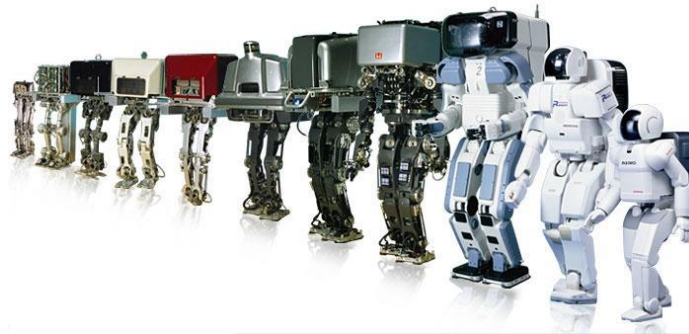


Figura 2.1 Clasificación de los robots por su generación [12].

- **Robots Playback.** regeneran una secuencia de instrucciones grabadas.
 - **Robots controlados por sensores.** Toman decisiones basados en datos obtenidos mediante el uso de sensores.
 - **Robots controlados por visión.** Manipulan un objeto al utilizar información obtenida mediante un sistema de visión.
 - **Robots controlados adaptablemente.** Pueden reprogramar automáticamente sus acciones mediante una base de los datos obtenidos por medio de sensores.
 - **Robots con inteligencia artificial.** Utilizan las técnicas de inteligencia artificial para hacer sus propias decisiones y resolver problemas.
- b. Nivel de inteligencia.** Existen robots que pueden interactuar con el usuario en base a aprendizajes, es por ello que se ha clasificado a los robots dentro de seis clases sobre la base de su nivel de inteligencia como se puede observar en la figura 2.2 que se muestra a continuación:



Figura 2.2 Robots Inteligentes [13].

- **Dispositivos de manejo manual.** Controlados por un usuario.
 - **Robot de secuencia arreglada.**
 - **Robot de secuencia variable.** Se puede modificar su secuencia.
 - **Robot regenerador.** El usuario conduce el robot en su tarea.
 - **Robot de control numérico.** El usuario programa el movimiento, hasta mostrar manualmente la tarea.
 - **Robot inteligente.** Interactúan con cambios del medio ambiente.
- c. **Nivel de control.** Como su nombre lo indica se agrupan de acuerdo al nivel de control que realizan los robots, de acuerdo a estos parámetros se clasifican en tres grupos los cuales se detallan a continuación:
- **Nivel de inteligencia artificial.** Realiza las tareas mediante comandos de bajo nivel.
 - **Nivel de modo de control.** Se obtiene una interacción entre los diferentes mecanismos, trayectorias planeadas y puntos de asignación.
 - **Niveles de servosistemas.** Se controlan los parámetros de los mecanismos mediante retroalimentación de los datos captados por los sensores.
- d. **Nivel de lenguaje de programación.** Se considera el nivel de lenguaje de programación la herramienta indispensable para el desarrollo de una aplicación efectiva, es decir crear un lenguaje de alto nivel. Los sistemas de programación de robots se encuentran distribuidos en tres clases, tal como se muestra en la siguiente figura :



Figura 2.3 Clasificación de los robots por su lenguaje de programación (Robot AIB) [14]

- **Sistemas guiados.** El usuario conduce el robot mediante movimientos que deben ser realizados.
 - **Sistemas de programación de nivel-robot.** El usuario escribe el programa para determinar el movimiento y el sensado.
 - **Sistemas de programación de nivel-tarea.** El usuario especifica la operación por sus acciones sobre los objetos que el robot manipula [15].
- e. Aplicación.** En la actualidad los robots son de gran utilidad en la vida diaria, ya que son utilizados en diversas actividades, tales como robots en los salones de clases, robots soldadores en la industria automotriz, hasta brazos operadores. Algunas de las áreas en donde se utiliza la robótica son:
- Agricultura.
 - Educación.
 - Laboratorios y cirugías.
 - Manipuladores cinemáticas.
 - Espacio.
 - Vehículos.
 - Androides.
 - Robots móviles.

2.2.2 Comunicaciones Inalámbricas

Las comunicaciones inalámbricas se consiguen mediante ondas electromagnéticas, a través de ondas de radio, microondas e infrarrojos, en cualquiera caso se requiere de una antena para transmitir la señal y de igual forma otra para recibirla. La comunicación inalámbrica resulta de gran utilidad en zonas de difícil acceso, ya que se puede alcanzar

mayor distancia dependiendo de los parámetros de potencia del transmisor de la frecuencia de transmisión [16].

Radiofrecuencia (RF)

El término radiofrecuencia (RF) denominado como espectro de radiofrecuencia, comprende las frecuencias a la que la radiación de energía electromagnética es útil para propósitos de comunicación, se extiende desde los 100KHz y 100GHz. Dichas ondas pertenecientes a este espectro se pueden transmitir aplicando corriente alterna la cual es generada por una antena.

El rango de las microondas está incluido en las bandas de radiofrecuencia, concretamente en las de UHF (ultra-high frequency - frecuencia ultra alta) 0,3-3 GHz, SHF (super-high frequency - frecuencia súper alta) 3-30 GHz y EHF (extremely-high frequency - frecuencia extremadamente alta) 30-300 GHz [17].

El hercio es la unidad de medida de la frecuencia de las ondas, y corresponde a un ciclo por segundo. Las ondas electromagnéticas de esta región del espectro, se pueden transmitir aplicando la corriente alterna originada en un generador a una antena [4].

Equipos de Radiofrecuencia

Para que exista una eficaz comunicación, es indispensable contar con equipos de Radio-Frecuencia adecuados, es por ello que se debe contar con dos módulos, un transmisor y un receptor inalámbricos. Dado que sólo uno de ellos es un equipo de transmisión, la comunicación de datos sólo funcionará en un sentido, por lo cual se necesitan dos pares de diferentes frecuencias para poder actuar como un transmisor/receptor [18].

Bluetooth

Es un sistema de comunicación inalámbrica diseñado para ser utilizado en cortas distancias, diseñado por IBM y otras empresas con el fin de reducir la cantidad de cables que se encuentran en un ambiente laboral. Se trata en un método de enlace que en lugar de cables utiliza ondas de radio, utiliza la banda de frecuencias de 2,4GHz y funciona en dispositivos que se encuentren en un radio máximo de 10 metros, su logo se muestra en la figura 2.4 que se detalla a continuación [19].

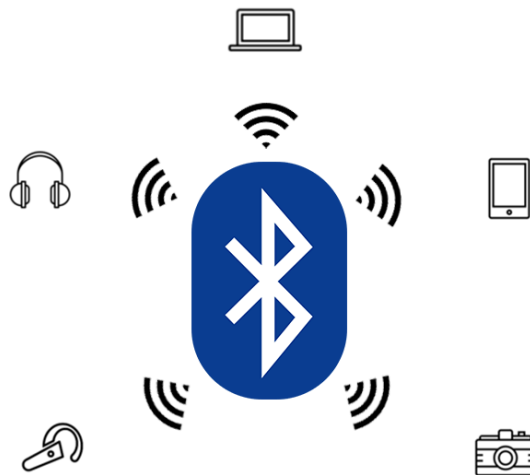


Figura 2.4 Logo de bluetooth [20].

Funcionamiento. El estándar bluetooth opera como maestro/esclavo, se conoce como piconet a la red formada por un dispositivo y todos los dispositivos que se encuentra dentro de su rango, generalmente pueden coexistir hasta 10 piconets dentro de una misma área de cobertura, además un dispositivo maestro puede conectarse simultáneamente hasta con 7 dispositivos esclavos activos y 255 cuando se encuentran en modo de espera. Los dispositivos que se encuentran en el modo en espera se sincronizan, pero no tienen su propia dirección física en la piconet [21].

Universal Asynchronous Receivers/Transmitters (UARTs)

El Receptor/Transmisor Asíncrono Universal (UART) es un circuito integrado que se utiliza para comunicaciones vía serie con el estándar RS-232C. Es un protocolo sencillo y universal utilizado para comunicaciones entre periféricos y un PC. La denominación de asíncrono, en este caso no significa que no necesite un sincronismo, al contrario, al no llevarlo explícitamente en una línea debe tenerlo implícitamente en su funcionamiento y por tanto es mucho más rígido.

El Microprocesador LM3S8962 incluye dos 16C550 (UART0 y UART1) totalmente programables, con una velocidad de transferencia de hasta 3,125 Mbps. La UART0 es utilizada internamente para el envío y recepción de datos en el depurado de las aplicaciones. Este módulo se ha utilizado para realizar diversas pruebas y visualizar los datos a través del Hyper Terminal de Windows. [22]

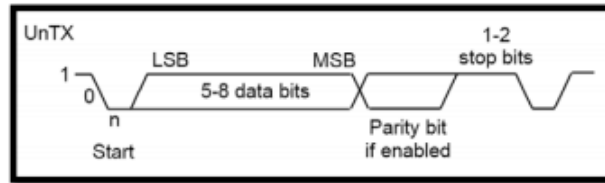


Figura 2.5 Formato de trama del protocolo UART para el envío de datos [22]

Características principales del Módulo UART:

- FIFO recepción de 16x12 bits y FIFO transmisión de 16x8 bits
- Programmable baud rate generator
- Generación automática de los bits “start”, “stop”, y “parity”
- Interface serie programable
 - 5, 6, 7, o 8 bits de datos
 - Generación y detección de bits “even”, “odd”, “stick”, o “no parity”
 - 1 o 2 bits de “stop”
- IrDA serial-IR (SIR) “encoder/decoder”
- DMA interface [22]

Wifi

La tecnología Wifi define una forma de intercambiar datos entre dos equipos mediante la utilización de ondas de radio, esto se conoce como medio radioeléctrico. Este sistema tiene la particularidad de ser muy resistente a las interferencias provenientes de otras fuentes de radio así como de los efectos del eco, lo que le permite coexistir con otros sistemas de radiofrecuencia si verse muy afectado. Además al permitir la interconexión de equipos permite compartir los servicios de comunicaciones, como el acceso a internet, compartir ordenadores, periféricos, etc. Como se muestra en la figura 2.6 [23].



Figura 2.6 Conexión de periféricos mediante Wifi [24].

Funcionamiento. La tecnología de acceso directo inalámbrico denominada Wifi se aplica a redes de ordenadores o terminales de datos conectados mediante ondas de radio basadas en el estándar 802.11, su funcionamiento es similar al de los teléfonos móviles, ya que permite una conexión inalámbrica entre distintos equipos en un radio geográfico restringido por una estación base. Además permite conexiones a altas velocidades de hasta 600Mbps con tecnología 802.11n y acceso a redes de intranet de las empresas, siendo también un medio innovador de suministro de acceso inalámbrico hacia internet [25].

Zigbee

El stack de protocolos Zigbee se apoya sobre IEEE 802.15.4-2003 para la comunicación entre dispositivos cercanos, es decir, el acceso al medio y el intercambio de mensajes por éste. Por sobre esta base, define un nivel de routing (NWK) que permite que los dispositivos con funcionalidad de router puedan transportar mensajes para otro destinatario, extendiendo el alcance total de la red. [26]

El módulo Xbee 802.15.4, son pequeños dispositivos de color azul tal como se muestra en la figura 2.7, los cuales permiten la comunicación entre sí de forma inalámbrica, son fabricados por Digi Internacional, los cuales ofrecen una gran variedad de combinaciones de hardware, protocolos, antenas y potencias de transmisión.



Figura 2.7 Modulo Xbee [27].

Estos módulos permiten configurar mucho de sus pines, para funcionar como entradas analógicas, además puede operar en uno de dos modos:

- **Peer to peer.** Cada módulo habla con cualquier otro modulo, emitiendo broadcasts o direccionamiento remoto, de modo que todos los módulos deben tener el receptor encendido de forma que cualquiera puede recibir un mensaje en cualquier instante.
- **Con coordinador.** Uno de los módulos es configurado como coordinador, de tal forma que este siempre alerta, pudiendo los remotos permanecer en modo de bajo consumo por un tiempo determinado.

Existen dos modos de operación:

- **Transparente.** El modulo envia al remoto los mensajes por su puerto serie y presenta los mensajes del modulo remoto.
- **API.** Recomendado para el modulo que cumple el rol de coordinador, ya que permite recibir y enviar mensajes desde multiples modulos [28].

2.2.3 Señales EEG(Electroencefalograficas)

La señal EEG es la información neurofisiológica obtenida por medio del registro de la actividad bioeléctrica cerebral bajo ciertas condiciones, se utiliza fundamentalmente para el diagnóstico de alguna enfermedad o de posibles lesiones. Además el EEG registra la actividad eléctrica producida en la corteza del cerebro, ya que dicha actividad se determina en microvoltios (uV), se debe amplificar en un factor de 1.000.000 para que pueda ser visualizado en la pantalla de un ordenador o en una hoja impresa [29].

Métodos de medición mediante EEG.

Los EEG son registros de los potenciales eléctricos generados por ondas cerebrales que se hallan en el rango de 1uV a 100uV en adultos, los cuales son recogidos a través de sensores que captan las señales bioeléctricas emitidas por el cerebro como una respuesta integrada de las estructuras cortico-subcorticales reflejadas en las capas más superficiales de la corteza craneal.

Una electroencefalografía se puede captar a través de diversos procedimientos: sobre el cuero cabelludo, en la base del craneo, en el cabello expuesto o en localizaciones cerebrales profundas. Para ello se utilizan diferentes tipos de electrodos:

1) **Superficiales.** Estos a su vez se subdividen en:

Adheridos. Pequeños discos metálicos de 5mm de diámetro, que se adhieren con pasta conductora.

De contacto. Pequeños tubos de plata enroscados en soportes de plástico, se colocan en contacto con almohadillas humedecidas con solución conductora, se sujetan al cráneo con bandas elásticas y se conectan con pinzas, como se muestra en la figura 2.8.

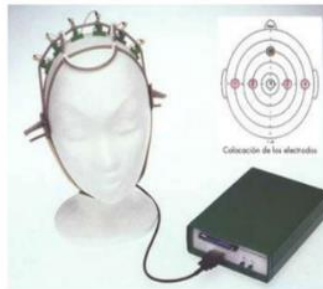


Figura 2.8 Electroencefalograma de contacto [30].

En caso de malla. Existen en diferentes tamaños, de acuerdo a la talla del paciente, se sujetan con cintas a una banda torácica., como se ve en la figura 2.9



Figura 2.9 Electroencefalograma de malla [30].

De aguja. Se emplea en recién nacidos, puede ser de uno solo o de uso múltiple.

2) **Quirúrgicos.** Se utilizan durante la cirugía y son manipulados por un neurocirujano, tal como se muestra en la figura 2.10 [30].

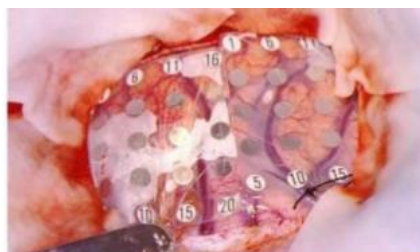


Figura 2.10 Electroencefalograma Quirúrgico [30].

Según la forma de captación de este tipo de método, el registro de la actividad cerebral recibe nombres distintos:

- Electroencefalograma (EGG). Mediante el uso de electrodos de superficie o basales.
- Electrocoagulograma (ECoG). Se utiliza con electrodos quirúrgicos en la superficie de la corteza.
- Estéreo Electroencefalograma (E-EEG). Se realiza a través de electrodos quirúrgicos de aplicación profunda [30].

Comportamiento de la señal EEG.

“Poseen amplitudes que van desde los 10 mV en registros sobre el córtex, a 100 mV en la superficie del cuero cabelludo. Las frecuencias de estas ondas se mueven entre 0,5 y 100 Hz y dependen mucho del grado de actividad del córtex cerebral. La mayoría de las veces estas ondas no poseen ninguna forma determinada, en algunas son ritmos normales que suelen clasificarse en ritmos a, b, q y d. En otras poseen características muy específicas de patologías cerebrales como la epilepsia”. [30].

A partir de la señal del EEG es posible diferenciar ondas clasificadas en términos de su frecuencia de oscilación y que son denominadas alfa, beta, gama, delta, y theta, tal como se muestra en la figura 2.11.

Ondas cerebrales	Frecuencia	Estado mental
Onda delta	0,5 - 3 Hz	sueño profundo
Onda theta	4 - 7 Hz	sueño ligero
Onda alfa	8 - 13 Hz	despierto, relajado
Onda beta	14 Hz	despierto, excitado

Figura 2.11 Ondas cerebrales [31].

“Las ondas alfa poseen frecuencias entre 8 y 13 Hz. Se registran en sujetos normales despiertos, sin ninguna actividad y con los ojos cerrados, localizándose sobre todo en la zona occipital; su amplitud está comprendida entre 20 y 200 mV.

Las ondas beta poseen frecuencias entre 14 y 30 Hz, aunque pueden llegar hasta los 50 Hz; se registran fundamentalmente en las regiones parietal y frontal. Se dividen en dos tipos fundamentales, de comportamiento muy distinto, b1 y b2.

Las ondas theta poseen frecuencias entre 4 y 7 Hz y se presentan en la infancia aunque también pueden presentarlas los adultos en períodos de stress emocional y frustración. Se localizan en las zonas parietal y temporal.

Las ondas delta poseen frecuencias inferiores a 3,5 Hz y se presentan durante el sueño profundo, en la infancia y en enfermedades orgánicas cerebrales graves” [30].

2.2.4 Adquisición de señales

La mayor parte de los BCIs obtienen la información relevante a partir de la electroencefalografía debido a la elevada resolución temporal, bajo coste, alta portabilidad y escaso riesgo para los usuarios. Sin embargo, la eficacia a la hora de obtener resultados depende de múltiples factores; el cuero cabelludo, cráneo, ruido de fondo proveniente de otros dispositivos o señales del propio cerebro, son factores que pueden dificultar la necesaria obtención de las señales deseadas, introduciendo grandes componentes de ruido [32].

Existen dos aproximaciones para hacer frente al problema de la captación de la información deseada:

Métodos invasivos

Este tipo de procedimiento se centra en la obtención de señales limitando el ruido y los obstáculos, es por ello que típicamente se realiza a través del cuero cabelludo hacia las zonas del cerebro de donde se pretende obtener la información., como se muestra en la figura 2.12.

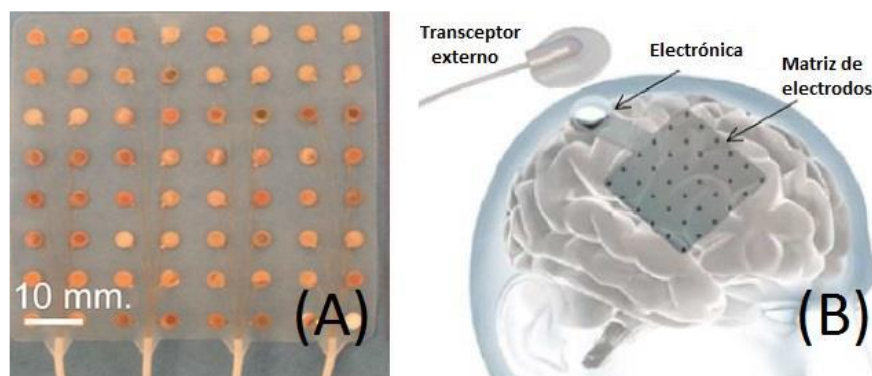


Figura 2.12 Dispositivo de método de adquisición de señales invasivos [30].

Métodos no invasivos.

Este tipo de métodos limitan el impacto de la adquisición de la señal sobre el usuario ya que se realiza una toma de señales desde el exterior de la cabeza del usuario. Uno de los métodos no invasivos más utilizados es la electroencefalografía la cual consiste en la colocación de electrodos directamente sobre el cuero cabelludo con ayuda de un gel o líquido conductor como se muestra en la figura 2.13.



Figura 2.13 Casco Mindwave de NeuroSky, instrumento no invasivo [33].

2.2.5 Tarjetas de Adquisición

Las tarjetas de adquisición de datos poseen una memoria en la que se almacenan las muestras adquiridas o las que van a ser generadas, es de tipo FIFO. Además posee un espacio de memoria para transferir los datos de la tarjeta hacia el computador. En la transferencia de datos entre la tarjeta y el ordenador existe mayor flujo de datos en función de la aplicación que se realice. En el ámbito de las tarjetas de adquisición de datos existen numerosos fabricantes como National Instruments, Addi Data, Advantech, etc los cuales comercializan tarjetas con diversas interfaces de conexión [34]- [35].

Clasificación

Estas pueden ser de tipo plug-in, sistemas externos y sistemas híbridos.

Tarjetas de tipo plug-in. Son diseñadas para ser insertadas en los slots internos de una computadora, actualmente son diseñadas para el bus PCI aunque existen varias versiones como: EISA, IBM Micro Channel, Buses Appel, USB, etc.

Poseen alta velocidad que va desde los 100KHz a 1GHz, poseen gran disponibilidad de un amplio abanico de funciones: A/D, D/A, E/S digitales, contadores, temporizadores y funciones específicas, tal como se muestra en la figura 2.14.

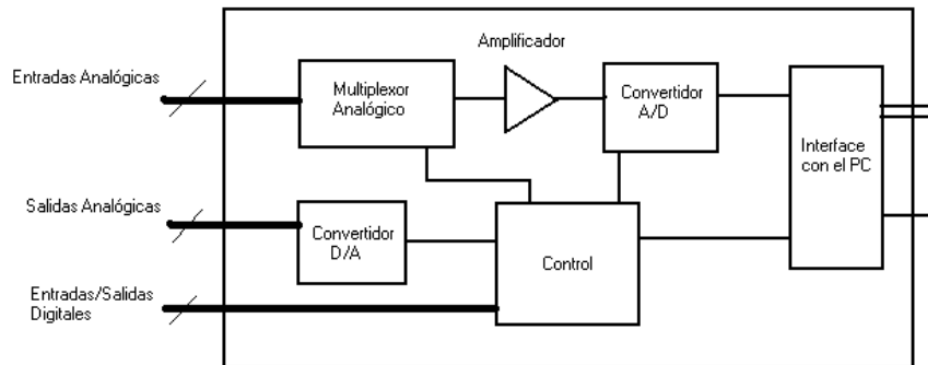


Figura 2.14 Esquema básico de una tarjeta de adquisición de datos [36].

Sistemas externos de adquisición de datos. En un inicio consistían en un equipo autónomo conectado a un computador mediante una interfaz estándar. Poseen mayor velocidad y versatilidad, y cuentan con un entorno eléctrico más protegido del ruido. Actualmente son equipos autónomos orientados a aplicaciones industriales como procesos de control en tiempo real, para proyectos con alta sensibilidad para bajos niveles de tensión en las entradas.

Cuenta con las siguientes arquitecturas:

- PXI (PCI extensión for Instrumentation), mostrada en la figura 2.15
- VISA (Virtual Instrumentation Software Architecture)
- VXI (VME extensión for Instrumentation)



Figura 2.15 Arquitectura PXI [36].

Sistemas híbridos de adquisición de datos. Utilizan una combinación de tarjetas de adquisición de datos e instrumentación específica de test, como se muestra en la figura 2.16 [36].



Figura 2.16 Sistema híbrido de adquisición de datos [36].

Arduino

Arduino es una plataforma de prototipos de electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar el cual se muestra en la figura 2.17, una de sus ventajas es el que puede sentir el entorno mediante la recepción de entradas desde una variedad de sensores, además puede afectar su alrededor mediante el control de luces, motores y otros artefactos. El microcontrolador de la placa se programa usando el *Arduino Programming Language* (basado en Wiring) y el *Arduino Development Environment* (basado en Processing). Los proyectos de Arduino pueden ser autónomos o se pueden comunicar con software en ejecución en un ordenador [37].

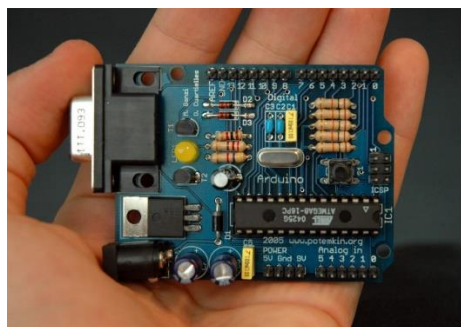


Figura 2.17 Arduino [38].

Funcionamiento

Como pasa con la mayoría de las placas microcontroladoras las funciones de Arduino pueden resumirse en tres. En primera instancia, tenemos una interfaz de entrada, que

puede estar directamente unida a los periféricos, o conectarse a ellos por medio de los puertos, el objetivo de esa interfaz de entrada es llevar la información al microcontrolador, el cual varía dependiendo de las necesidades del proyecto en el que se desea usar la placa, y hay una buena variedad de fabricantes y versiones disponibles.

Por último, tenemos una interfaz de salida, que lleva la información procesada a los periféricos encargadas de hacer el uso final de esos datos [37].

Raspberry Pi

Se trata de una diminuta placa base de 85 x 54 milímetros (un poco más grande que una cajetilla de tabaco) en el que se aloja un chip Broadcom BCM2835 con procesador ARM hasta a 1 GHz de velocidad, GPU VideoCore IV y hasta 512 Mbytes de memoria RAM, tal como se muestra en la figura 2.18 [39].

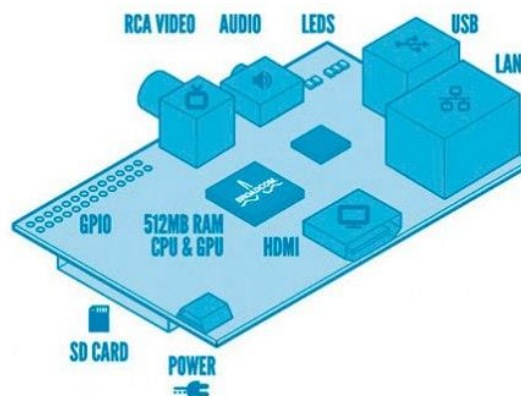


Figura 2.18 Raspberry Pi [40].

Las Raspberry Pi son consideradas una buena opción para aprovechar la potencia de un ordenador pero recurriendo a un tamaño sumamente más pequeño, Los podemos utilizar como servidor de contenidos, conectados al televisor y, por supuesto, como ordenador al uso. Raspberry Pi es uno de los productos más populares para estos fines, tanto por su atractivo precio como por las enormes opciones que trae consigo [39].

FPGA

Las siglas FPGA significan Field Programmable Gate Array, como su nombre indica, se trata de un dispositivo compuesto por una serie de bloques lógicos (puertas, registros, memorias, flip/flops, etc) programables, es decir, la interconexión entre estos bloques

lógicos y su funcionalidad no viene predefinida sino que se puede programar y reprogramar, su esquema se muestra en la figura 2.19 [41].

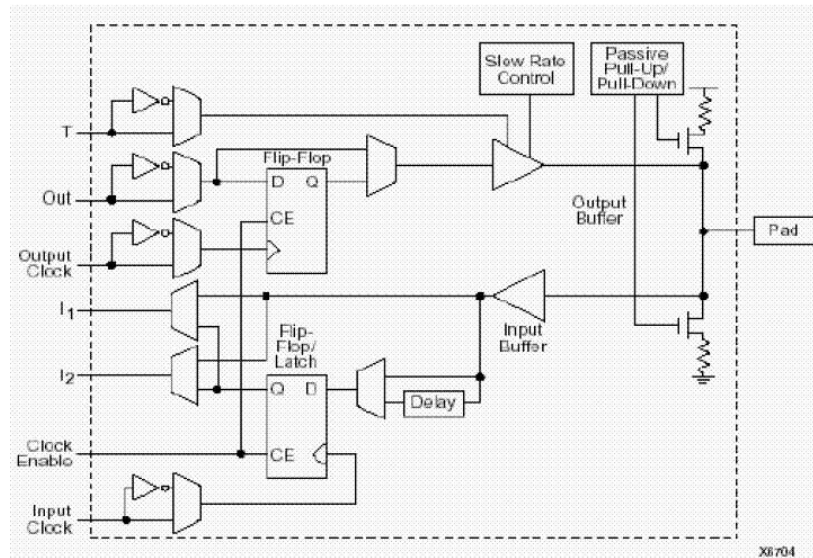


Figura 2.19 Esquema FPGA [42].

Funcionamiento

La adopción de chips FPGA en la industria ha sido impulsada por el hecho de que los FPGAs combinan lo mejor de los ASICs (Circuito Integrado para Aplicaciones Específicas) y de los sistemas basados en procesadores. Ofrecen velocidades temporizadas por hardware y fiabilidad, pero sin requerir altos volúmenes de recursos para compensar el gran gasto que genera un diseño personalizado de ASIC. El silicio reprogramable tiene la misma capacidad de ajustarse que un software que se ejecuta en un sistema basado en procesadores, pero no está limitado por el número de núcleos disponibles. A diferencia de los procesadores, los FPGAs llevan a cabo diferentes operaciones de manera paralela, por lo que éstas no necesitan competir por los mismos recursos. Cada tarea de procesos independientes se asigna a una sección dedicada del chip, y puede ejecutarse de manera autónoma sin ser afectada por otros bloques de lógica. Como resultado, el rendimiento de una parte de la aplicación no se ve afectado cuando se agregan otros procesos [43].

Características de los FPGA's.

Esencialmente, los FPGA's están compuestos por elementos con recursos no comprometidos que pueden ser seleccionados, configurados e interconectados; el PLD

(Dispositivo lógico programable) no requiere del proceso de interconexión dado que sus elementos internos (registros, arreglos lógicos y macro celdas de entrada/salida) son fijos en relación de unos con otros. Los FPGA's actualmente en uso presentan algunas características fundamentales, entre las que destacan:

- Todos estos dispositivos están compuestos de un cierto número de módulos lógicos relativamente independientes entre sí, que pueden interconectarse para formar un circuito mayor; estos módulos pueden ser grandes bloques configurables o pequeños elementos de función fija formados por algunas compuertas.
- El tamaño óptimo de los módulos lógicos y los requerimientos de interconexión son completamente dependientes del tipo de aplicación que se desea implementar en el dispositivo. Por ejemplo, una aplicación compuesta de estructuras regulares o semindependientes puede implementarse en forma más eficiente en un FPGA con módulos lógicos grandes, mientras que si se tiene un diseño en el que predominen funciones lógicas aleatorias interrelacionadas se puede emplear un dispositivo con menos módulos y más recursos de interconexión.
- Los módulos en un FPGA se interconectan por medio de canales configurables, mediante un proceso conocido como enrutamiento. El enrutamiento consiste básicamente en determinar, ya sea en forma manual o a través de herramientas de cómputo, una estrategia de interconexión eficiente.
- El rango en tamaño de los FPGA's abarca desde 1,200 hasta 20,000 compuertas equivalentes, mientras que en los PLD's es desde algunos cientos hasta 2,000 compuertas. Dado que los FPGA's se están incrementando constantemente en cuanto a densidad, se puede esperar que sus aplicaciones típicas serán en orden de magnitud más compleja que para un simple PLD [44].

Lenguajes de programación

La FPGA posee celdas que se configuran con una función específica ya sea como multiplexor, memoria o como una función lógica, por lo que para programarla se debe definir la función lógica que se desea que realice. Esto se consigue mediante el uso de un lenguaje de programación, estos lenguajes son conocidos como HDL (lenguaje de descripción de hardware), de los cuales los más utilizados son:

VHDL (Lenguaje para descripción y modelado de circuitos)

VHDL fue estandarizado por la IEEE en 1987 y extendido en 1993, el cual permite diseñar, simular y sintetizar cualquier proyecto, que va desde un circuito combinacional simple hasta un sistema microprocesador completo en un chip [45].

Toda descripción creada en VHDL está conformada por tres elementos fundamentales:

- **LIBRARY.** Son bibliotecas que contienen elementos que se pueden usar en la descripción del programa.
- **ENTITY.** Es el encargado de especificar las entradas y salidas.
- **ARCHITECTURE.** Contiene el código VHDL que describe el funcionamiento del circuito.

En la figura 2.20 se observa los elementos fundamentales que deben intervenir en una descripción básica de VHDL [46].

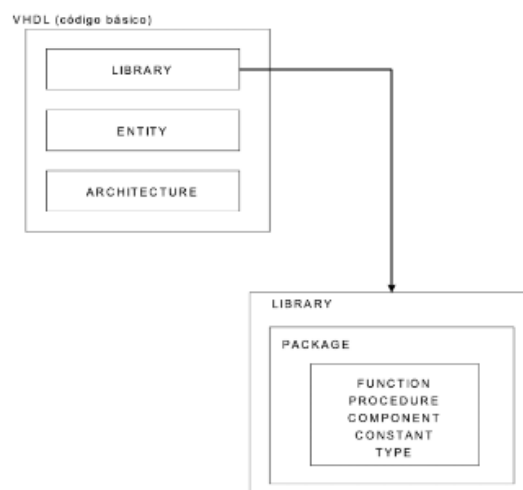


Figura 2.20 Estructura básica de una descripción VHDL [46].

Verilog

Es un lenguaje HDL diseñado por CADENCE Design Systems, usado para modelar sistemas electrónicos, soporta el diseño, prueba e implementación de circuitos analógicos, digitales y de señal mixta a diferentes niveles de abstracción. El diseño de programas en Verilog consiste en crear una jerarquía de módulos, los cuales son definidos con conjuntos de puertos de entrada, salida y bidireccionales.

Internamente cada módulo contiene sentencias que definen su comportamiento, además establece la conexión entre los puertos, cables y registros. Un módulo puede contener una o más instancias de otro módulo para definir un sub-comportamiento [47].

Las descripciones en Verilog están estructuradas tal como se muestra en la figura 2.21 que se muestra a continuación:

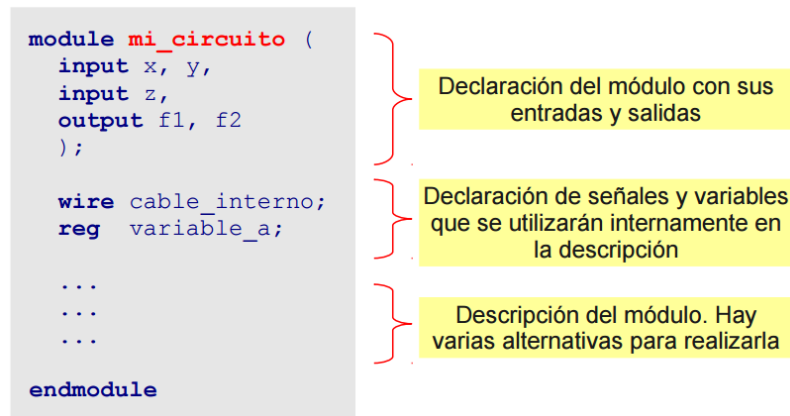


Figura 2.21 Estructura de una descripción Verilog [48].

ABEL

Abel es un lenguaje de descripción de hardware (HDL), diseñado con el fin de permitir a los diseñadores especificar funciones lógicas para su realización en los PLD. Utiliza un compilador el cual se encarga de traducir el archivo de texto de ABEL a un patrón de fusión que puede ser descargado en un PLD físico, permitiéndole expresar sus funciones mediante tablas de verdad o simples instrucciones [45].

La estructura básica de descripción de Abel se detalla en la figura 2.22.

Encabezado	Module <i>module_name</i> title <i>string</i>
Declaraciones	<i>DeviceID</i> device <i>deviceType</i> Declaraciones de pin otras declaraciones
Descripción lógica	Equations <i>Ecuaciones</i>
Vectores de Test	test_vectors <i>vectores de test</i>
	end <i>module_name</i>

Figura 2.22 Estructura básica de una descripción de ABEL [49].

LUCID - DISEÑO DIGITAL EXPLICADO

Lucid es una nueva HDL (Hardware Description Language, Lenguaje de descripción de hardware) que está diseñado para que sea más fácil trabajar con FPGAs. Esto se realiza por no depender de "modelos" de flip-flops y máquinas de estados finitos, lo que reduce la cantidad de código que tiene que ser por escrito, y la comprobación de errores en tiempo real a través de la IDE Mojo (Software realizado por la compañía Mojo para la programación de la placa con el mismo nombre).

Lucid se basa en Verilog pero con sintaxis de C++ / Java [50].

2.3.Propuesta de Solución

El Autómata controlado por impulsos neuronales con FPGA de hardware libre ayudará a controlar dispositivos electrónicos mediante neuro señales emitidas por el ser humano.

CAPÍTULO 3

METODOLOGÍA

3.1. Modalidad de la investigación.

El proyecto tuvo una investigación aplicada, poniendo en práctica los conocimientos técnicos y científicos adquiridos durante la formación académica, además la información sobre la tecnología actual que sirvió para dar solución al problema planteado en la investigación.

El proyecto de investigación se desarrolló con una modalidad bibliográfica, debido a que fue necesario la recopilación de información sobre el tema planteado, se consultó en libros, revistas, artículos científicos, tesis y publicaciones existentes, siendo esta la mejor manera de obtener información.

Se empleó la investigación experimental debido a que se desarrolló varias pruebas hasta conseguir la configuración apropiada para el reconocimiento de los impulsos neuronales, detectados por el dispositivo de reconocimiento.

3.2. Población y muestra.

La presente investigación por sus características no se necesitó población y muestra.

3.3. Recolección de información.

Se realizó la revisión de documentos y luego se procedió a seleccionar la información mediante la observación, con el fin de ser utilizados dentro de la investigación.

3.4. Procesamiento y análisis de datos.

Una vez selecciona la información necesaria del proyecto de investigación se efectuó los siguientes pasos:

- Organizar la información obtenida.
- Revisar la información seleccionada.
- Análisis de la información y encontrar la solución al problema planteado.
- Interpretar los resultados.

3.5. Desarrollo del Proyecto.

En el proyecto investigación se realizó el siguiente procedimiento:

- Análisis de funcionalidades de dispositivos de control electrónicos disponibles en el mercado
- Análisis económico y de mercado sobre los dispositivos seleccionados para la adquisición de frecuencias neuronales, procesamiento, y transmisión hacia el autómata.
- Investigación sobre el proceso técnico de comunicación entre la tecnología EEG y el FPGA.
- Diseño de la comunicación para la adquisición, procesamiento y envío de señales neuronales
- Acondicionamiento de las señales de frecuencia producida por el lector de impulsos neuronales

- Enlace mediante radiofrecuencia (Bluetooth), entre el sistema de adquisición de las frecuencias neuronales y el FPGA.
- Programación del algoritmo de FPGA
- Enlace mediante radiofrecuencia, entre el FPGA y el control del autómata
- Implementación del sistema de control remoto del Autómata
- Programación de interfaz de usuario para la visualización de cada frecuencia y su comportamiento
- Pruebas de confiabilidad del funcionamiento del sistema de control neuronal del autómata
- Análisis de Resultados obtenidos de las pruebas del prototipo

CAPÍTULO 4

DESARROLLO DE LA PROPUESTA

El presente proyecto está orientado al diseño y construcción de un sistema de control electrónico mediante la adquisición y análisis de ondas cerebrales, para tener una mayor interacción con el medio sin necesidad de ningún movimiento físico, facilitando el control mediante frecuencias de la actividad cerebral; la construcción del sistema se basa en la unión de partes básicas e indispensables como lector de señales cerebrales, tarjetas de adquisición, líneas de conexión e interfaces conectados entre sí, utilizando diversas tecnologías.

El sistema de control electrónico mediante ondas cerebrales para un robot, necesita dispositivos que interactúen entre sí, y cumpla con la función requerida. Para poder lograrlo se desarrolla un esquema general del prototipo (ver figura 4.1.) que permite la adquisición de las señales de frecuencia alfa y beta y la interacción con el control del autómatas.



Figura 4.1 Esquema del sistema para el control del autómatas
Fuente: Investigadora

La figura 4.2 indica el desarrollo sistemático que permitió la implementación del dispositivo, en estas secciones se muestra de una manera detallada los procesos que ayudaron al control del dispositivo.

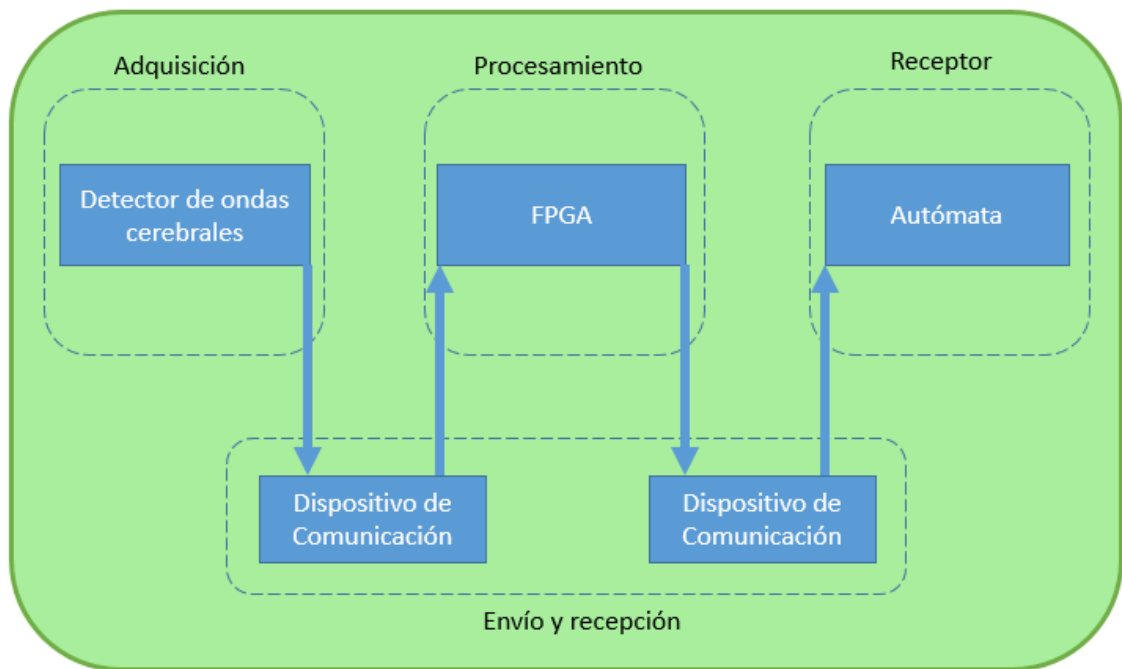


Figura 4.2 Esquema general del sistema de control
Fuente: Investigadora

El presente esquema ayuda a la comprensión de las diferentes etapas como es la adquisición, el procesamiento y la recepción. Se ha realizado una reducción de la información para disminuir la complejidad y dar un mejor panorama de la propuesta a realizarse.

- **Adquisición de ondas cerebrales:** La adquisición de ondas cerebrales, es la primera etapa del sistema de control, donde se obtendrá las señales EEG.
- **Emisión de ondas cerebrales:** Al adquirir las señales, se necesita enviarlas, por el cual se debe analizar, la forma de envío de los mismos.
- **Recepción de ondas cerebrales:** Para la recepción de las señales adquiridas, se tiene que ver la compatibilidad con el transmisor.
- **Procesamiento de las señales recibidas:** El procesamiento de las señales se realiza con una FPGA, la cual se analiza que cumpla con los requerimientos de la propuesta que se plantea, el cual es hardware libre.

- **Recepción de las señales ya procesadas y control en el autómata:** La recepción de señales se realiza de acuerdo a la forma de transmisión hacia el autómata.

4.1. Análisis del proceso técnico de reconocimiento de ondas cerebrales

La adquisición de las señales cerebrales es la parte más importantes de los dispositivos de interacción humano-máquina. La utilidad más conocida de esta adquisición, es la herramienta en el diagnóstico y tratamiento en el ámbito de la salud neurológica.

Todas las acciones que realiza el cuerpo humano necesitan señales que nacen en el cerebro, las mismas son transmitidas por medio de las terminaciones nerviosas hacia los músculos y órganos, para que cumplan los movimientos o funciones específicas.

Electroencefalograma

La electroencefalografía fue descubierta por Hans Berger en 1924, y consiste en obtener una señal eléctrica del funcionamiento del cerebro.

La actividad eléctrica, se produce cuando las neuronas se comunican. El evento más simple es conocido como potencial de acción, y se relaciona a una descarga provocada por la apertura y cierre rápido de canales iónicos Na^+ (Sodio) y K^+ (Potasio) en la membrana de la neurona. Si la membrana se despolariza hasta cierto umbral, la neurona se activa. El seguimiento de estas descargas en el tiempo revela la actividad cerebral [51].

Obtención de Señales electroencefalografías

Los electrodos son utilizados para la obtención de señales electroencefalografías, existen diferentes dispositivos con varios nombres, pero depende de la forma de captación del registro.

Mientras más cerca se realiza la detección de las señales será más limpia y libre de ruido, y de esta manera la señal tiene una mejor resolución, haciendo uso de la técnica no invasiva, todo esto se realiza con ayuda de los electrodos [52]

Electrodos

El principal uso en el análisis de las señales EEG, son los electrodos, estos dispositivos se colocan en la superficie de la cabeza, están compuestas por ciertos componentes químicos, que permiten convertir las corrientes iónicas existentes en la superficie de la piel, en corriente eléctrica. [52]



Figura 4.3 Electrodo. [52]

Colocación de los electrodos

La colocación de los electrodos se realiza mediante el sistema 10-20 (Figura 4.4) determinado por partes de la Federación Internacional de Sociedades de Electroencefalografía, este protocolo se normaliza a partir de las referencias atómicas Inion y Nasion longitudinalmente y los arcos auriculares. [52]

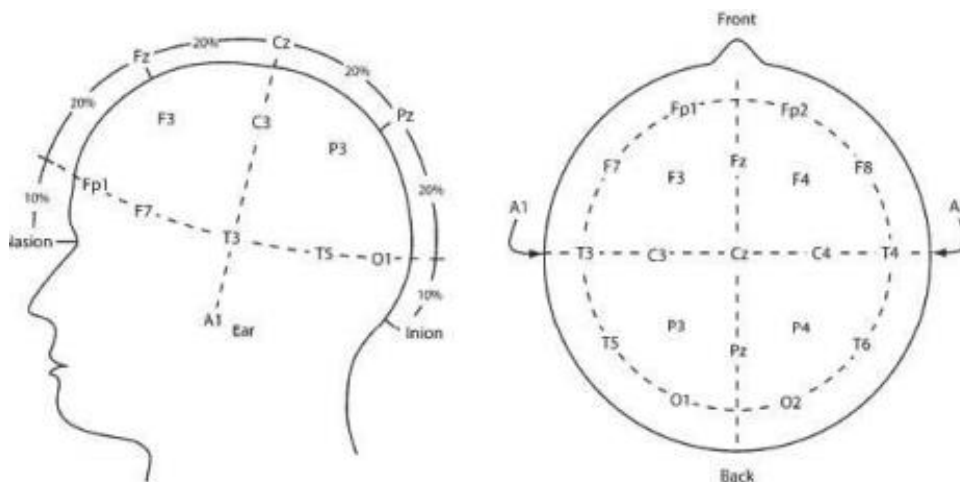


Figura 4.4 Puntos Principales de la cabeza de acuerdo al sistema 10-20 [52]

Ondas cerebrales

Onda cerebral es la actividad eléctrica producida por el cerebro. Estas ondas pueden ser detectadas mediante el electroencefalógrafo y se clasifican en:

- **Delta:** Estas ondas van de 0,5 a 4 Hz. Son las ondas más lentas y están presentes mientras una persona duerme.
- **Theta:** Fluctúa entre 4 y 7,5 Hz, están vinculados a la ineficiencia y el soñar despierto.
- **Alfa:** Oscilan de 8 a 13 Hz, son más lentas y asociadas con la relajación y desconexión.
- **Beta:** Están en la gama de frecuencias de entre 14 y 26 Hz, pero a menudo se las divide en beta bajo y beta alto para conseguir un análisis más específico. Las ondas son pequeñas y rápidas, asociadas con la concentración enfocada.
- **Gamma:** Estas ondas están en el rango de frecuencias mayores a 30 Hz. Su amplitud es muy pequeña, y su ocurrencia es rara, por lo que se las relaciona con ciertas enfermedades del cerebro. [51]

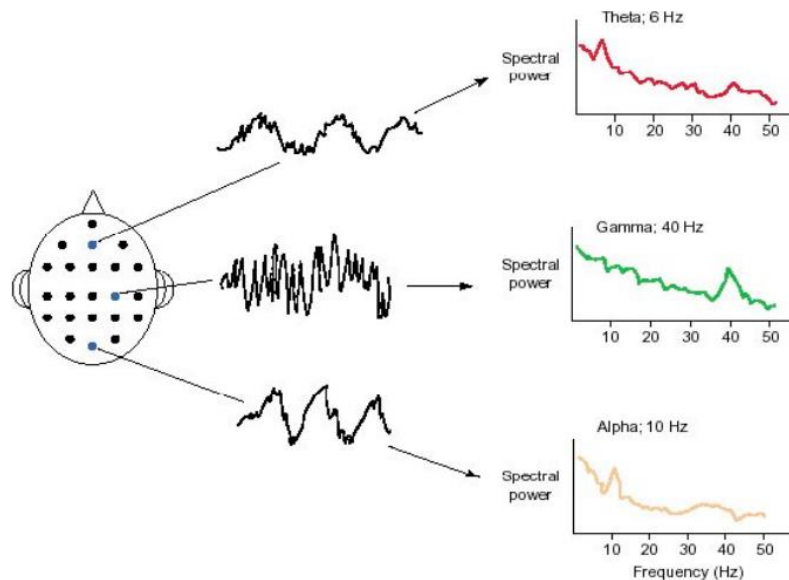


Figura 4.5 Puntos en el cerebro donde se detectan las ondas cerebrales [51]

4.1.1. Análisis de las ondas cerebrales, alfa y beta

Al realizar el análisis de cada una de las ondas cerebrales que posee un humano se puede concluir que ante la necesidad del proyecto, la cual es la lectura de ondas alfa y beta, se

va a tener como resultado, la meditación, que es el resultado de encontrarse entre las frecuencias (8-13Hz) y la concentración que se realiza cuando se encuentra en las frecuencias (14- 26 HZ).

Banda Alfa

Las ondas alfa, se dice que representan el límite entre lo inconsciente y consciente. Están presentes en estado de relajamiento, por lo que el tener más ondas de este tipo ayuda a disminuir la ansiedad y de esta forma fortalecer el sistema inmune.

La figura 4.6, indica la forma típica de la onda alfa.

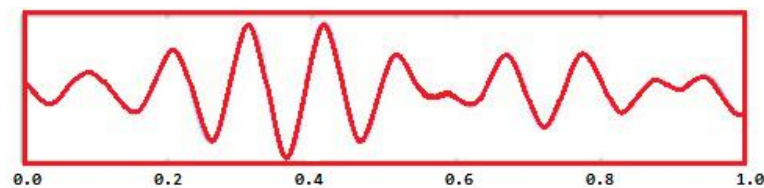


Figura 4.6 Ondas Alfa [51]

Ritmo alfa

En el electroencefalograma, mostrado en la figura 4.7, se observa una franja horizontal de energía cerca de 10 Hz, incluyen algunas ondas Alfa que el cerebro genera simplemente cerrando los ojos.

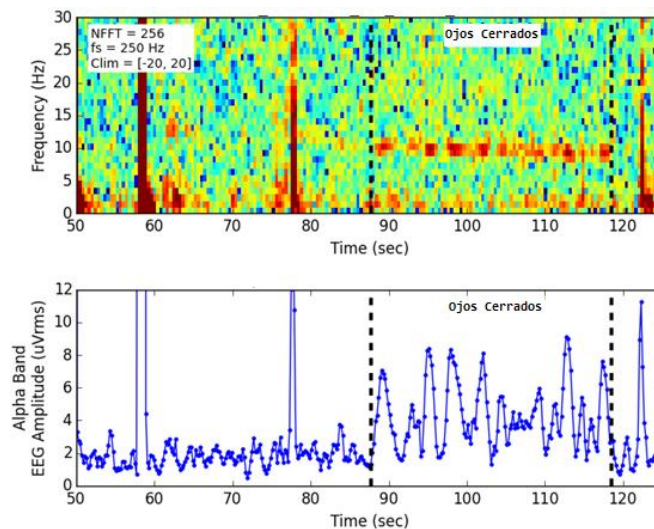


Figura 4.7 Lectura EEG, visualización de alfa en estado de relajación [53]

Frecuencia: 8 – 13 Hz

Localización: posterior

Estado de conciencia: vigil, reposo con ojos cerrados

Atenuación: somnolencia, concentración, estímulo o fijación visual.

Amplitud: 40 a 50 uV, en el adulto

Ausente en personas con discapacidad visual permanente

Se lo observa mejor en montaje referencial

Banda Beta

Las ondas beta (figura 4.8) se suelen clasificar en: beta baja o ritmos sensomotores de 14 a 15 Hz, beta media de 15 a 18 Hz, y beta alta de 18 a 26 Hz. Las ondas beta bajas y medias se relacionan con un estado de alerta bajo, con las actividades normales de la persona cuando está despierta; mientras que las ondas Beta altas se relacionan con un estado de alerta máximo, se presentan en estados de enojo, sorpresa e inquietud

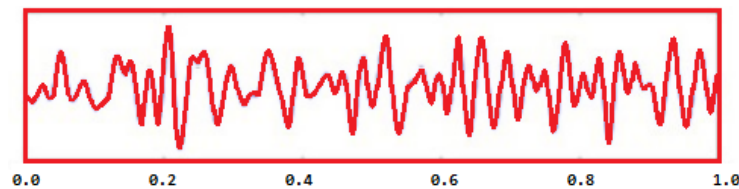


Figura 4.8 Ondas Beta [51]

Ritmo Beta

Concentración se ve generalmente como una mayor actividad en las frecuencias más altas de EEG - las denominadas ondas beta (13-30 Hz). Como se puede observar en la figura 4.9, se muestra frecuencias de 0 a 100 Hz. En esta trama, se puede ver claramente que hay más actividad EEG cuando se encuentra en estado de concentración en comparación con cuando no lo está.

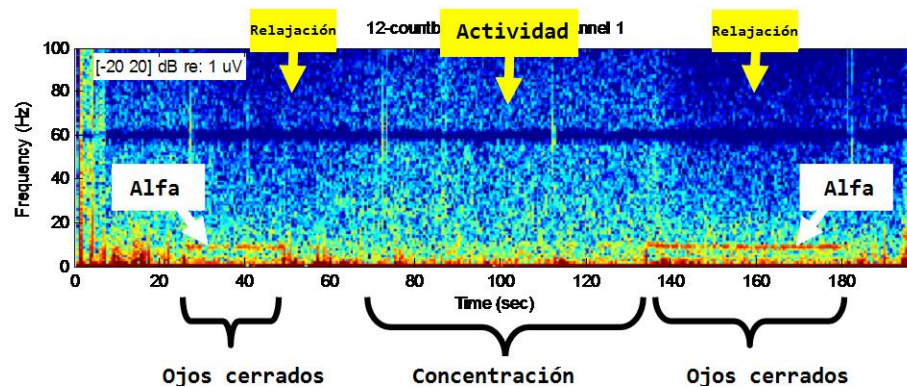


Figura 4.9 Lectura EEG, visualización de beta en estado de concentración [53]

Otro nombre: actividad rápida difusa

Descripción: en vigilia: < 20 uV. .

Frecuencia: 13 Hz.

Duración: 77 mseg.

Actividad gama: > 30 Hz.

Distribución: frontocentral

Comparación de los espectros

Mientras espectrogramas como las figuras anteriores son útiles para rápidas vistas cualitativos de tiempo y de frecuencia, es difícil de ser cuantitativos con un espectrograma. Así, que en la figura 4.10, se muestra el espectro promedio para un período de fuerte concentración (t = 90-130 seg) y se muestra el espectro promedio de un período en el que se encuentra con ojos cerrados y la mente tranquila (t = 155 -178 seg). Como se puede observar a continuación, los dos espectros son definitivamente diferente, especialmente para frecuencias superiores a 22 Hz.

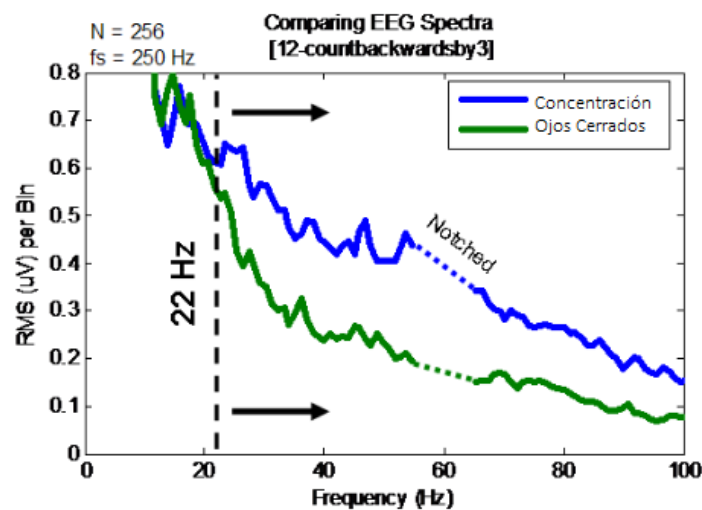


Figura 4.10 Comparación de espectros [53]

La detección de Concentración y de Relajación.

Concentración:

Con el conocimiento de que, "concentración" comienza a manifestarse como un aumento de la energía del EEG por encima de 22 Hz, ahora se puede contemplar el dispositivo que posea un detector de concentración. En la figura 4.11, se visualiza los datos que se

necesita, la trama superior es el espectrograma de EEG, la figura siguiente es lo que sucede cuando se filtra los datos de EEG para mostrar la intensidad sólo para frecuencias entre 22 y 100 Hz.

Para la concentración sostenida ($t = 90-130$ seg), la señal del EEG filtrada se está ejecutando alrededor de 3,4 uVrms. Entonces, cuando se detecta el cierre de ojos y relajarse (después de $t = 140$ seg), las señales EEG descienden hasta unos 2,0 uVrms. Así que, si se tuviera que definir un umbral para la detección de la concentración, que se podría poner en algún lugar en el medio ... por ejemplo, alrededor de 2,7 uVrms.

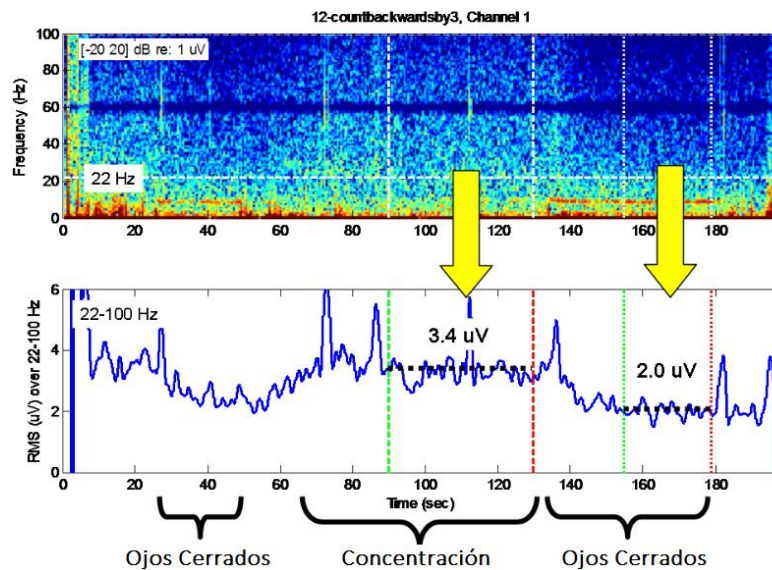


Figura 4.11 Comparación de espectros [53]

Relajación:

En la figura 4.12, se puede observar que existe una tendencia en la intensidad de la señal del EEG, pero que no está relacionada con la apertura de los ojos. Al principio, cuando los ojos se cierran, las señales EEG de alta frecuencia son bastante bajas en 1,9 uVrms. Entonces, cuando se abre los ojos ($t = 210$ seg), la intensidad EEG aumenta sólo ligeramente a 2,0 uVrms y se mantiene así durante bastante tiempo. Esta es una fuerte evidencia de que la simple apertura de sus ojos no se activa específicamente el aumento de la actividad Beta.

En la segunda mitad del periodo mientras están los ojos-abiertos, se observa que la intensidad EEG se desplaza hacia arriba. Con el tiempo, el promedio es de alrededor de 2,5 uVrms. Se nota que incluso el aumento de 2,5 uVrms todavía no exceda el umbral de

2,7 uVrms. Por lo tanto, este pequeño aumento no cumple con los criterios de "concentración".

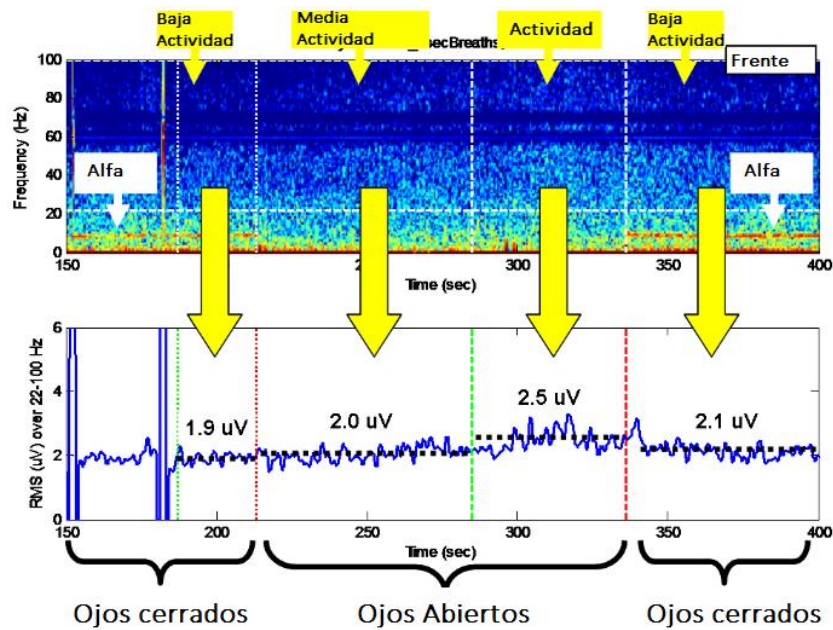




Figura 4.12 Análisis de espectro [53]

4.2. Análisis técnico del hardware requerido

Adquisición de ondas cerebrales

Para adquisición de las actividades cerebrales se necesita un electroencefalograma (EEG). Existen varios dispositivos para realizarlo pero de todos se ha escogido un producto ya fabricado y probado, esto se recomienda porque al estar ya fabricado es más fácil para el usuario colocarse los electrodos en el lugar óptimo sin tener conocimientos en el mismo. En el mercado existen varios dispositivos, pero entre ellos se destacan por su librería actualizada en el software para la interfaz (Labview): La diadema de Emotiv y la diadema de Neurosky.

Tabla 4.1 Análisis técnico comparativo entre diademas EEG.

Diadema EEG		
	EMOTIV EPOC	MINDWAVE NEUROSKY
PARÁMETROS TÉCNICOS		
Número de electrodos	14	1
Bits ADC	16	14
Interpretación de los sensores	3 estados mentales (sobre la base de las ondas cerebrales), 13 pensamientos conscientes, expresiones faciales, movimientos de la cabeza (detectadas por 2 giroscopios)	2 estados mentales (basado en 4 ondas cerebrales), el parpadeo
Conectividad	Bluetooth	Bluetooth
Productor	Emotiv	Neurosky
Precio	\$ 799	\$ 99,99

Fuente: Investigadora

Tomando en cuenta el menor costo y la mayor simplicidad en la adquisición e interpretación de datos, la diadema de Mindwave tiene ventajas para el proyecto.

Las ventajas son:

- Los EEG tienen filtros para las distintas frecuencias y amplitudes de las señales cerebrales (ondas alfa, beta, gama, etc).
- Se realizan interpretaciones de las frecuencias al mostrar el porcentaje de atención y meditación del usuario
- La empresa Neurosky está desarrollando dispositivos para realizar lecturas de EMG, ECG y GSR, por lo que hay la posibilidad de tener todas estas variables leídas en un solo sistema embebido.

Comparación entre Microcontroladores y FPGA´s

La presente comparación se realiza para analizar ventajas y desventajas de las tarjetas de adquisición FPGA frente a microcontroladores como lo es Arduino.

Mojo V3 vs Arduino



Figura 4.13 Mojo vs Arduino
Fuente: Investigadora

Los dispositivos Mojo V3 y Arduino son muy diferentes, con un microcontrolador, como un Arduino, el chip ya está diseñado, sólo tiene que escribir algún tipo de software, por lo general en C o C ++, y compila en un archivo hexadecimal que ha cargado en el microcontrolador. El microcontrolador almacena el programa en la memoria flash y lo almacena hasta que se borra o reemplaza y se tiene control sobre el software.

Tabla 4.2 Comparación entre Arduino y Mojo V3

Arduino Vs Mojo V3		
Características	Arduino	Mojo V3
Velocidad de respuesta		
Tamaño	Mediano	Mediano
Costo	\$35	\$75
Programación	Re-programable	Re-programable
Velocidad de diseño	Normal	Rápido

Seguro	No	Si
Secuencial	Si	No
Concurrente	No	Si
Fuente	Abierta	Abierta

Fuente: Investigadora



Los FPGAs son diferentes a los microprocesadores por la razón que aquí se diseña el circuito digital. No hay procesador para ejecutar software, al menos hasta que diseñe uno. Configurar un FPGA puede ser algo tan simple como configurar una compuerta, o algo tan complejo como un procesador multi-núcleo. Con las FPGA's se tiene control sobre el hardware.

Al realizar el análisis y la comparación respectiva, se puede concluir que aparte de la velocidad de respuesta alta, la FPGA también ofrece una gran ventaja por su diseño abierto para realizar circuitos digitales y existe la posibilidad de realizar diferentes proyectos y una amplia gama de investigaciones en diferentes ramas.

FPGA de hardware libre

En el mercado existen varias placas destinadas al procesamiento con arreglos de compuertas programables en campo (FPGA), tomando en cuenta la principal característica del requerimiento que es que posea hardware libre, se ha determinado dos FPGA como preselección. Los FPGA son Papilio y Mojo V3.

Tabla 4.3 Análisis técnico entre tarjetas FPGA.

Tarjeta FPGA		
	FPGA Mojo v3	FPGA Papilio
PARÁMETROS TÉCNICOS		
Costo	\$74,99	84,99
Tarjeta FPGA	Spartan 6 XC6SLX9	Spartan 6 LX9
Regulación de voltaje	4.8-12V	5V, 3.3V, 2.5V, y 1.2V
Pines de entrada y salida digitales	84	48
Oscilador	32Mhz	32Mhz
Suministración de energía	Por USB mini y por fuente	Por USB mini

Fuente: Investigadora

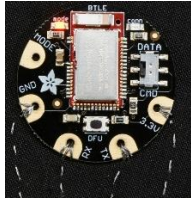


Al terminar el análisis de las diferentes tarjetas de adquisición, por su costo, facilidad de obtención en el mercado, software con actualización libre, y su funcionalidad, se determina que el Mojo V3, es el dispositivo elegido.

Después que la FPGA está configurada con éxito el ATmega entra en el modo esclavo. Esto permite que el FPGA pueda conectarse con el microcontrolador; que le da acceso al puerto serie y las entradas analógicas, otra ventaja de la misma es la compatibilidad de programación con IDE de Arduino.

Dispositivos de comunicaciones

Al hacer la observación de la comunicación necesaria entre la diadema EEG y la tarjeta FPGA, es obligatorio un módulo, que sea compatible con la FPGA seleccionada y con la comunicación de emisión de la diadema EEG.

Tabla 4.4 Comparación técnica de dispositivos de comunicación

Dispositivos de comunicación			
	Módulo Flora Bluefruit LE	Bluetooth XBee	Módulo Bluetooth HC05
PARÁMETROS TÉCNICOS			
Costo	17.50 USD	22,95 USD	15.00 USD
Accesibilidad	Determinadas Zonas geográficas	Determinadas Zonas geográficas	Cualquier parte del mundo
Dispositivo interno	Módulo MDBT40	Módulo CSR BC417143	Módulo CSR BC417143
Protocolo	Bluetooth V4 BLE	Bluetooth v2.0 + EDR	Bluetooth v2.0 + EDR
Compatibilidad	Android 4.3 o superior	Todas las distribuciones Android	Todas las distribuciones Android
Tasa de Baudios por defecto	9600 baud	9600 baud	9600 baud
Protocolo de comunicaciones	Beacon -UART simulado	UART	UART
Comunicaciones seriales	9600 bps	1200 ~ 1382400 Bps	1200 ~ 1382400 bps
Voltaje de operaciones	2.7V ~ 3.6V	3.3V	3.3V
Consumo	3.4mA(MDBT40)	50mA	30~40mA
Frecuencia de trabajo	2.4 ~ 2.483GHz	2.4 ~ 2.48GHz	2.4GHz banda ISM
Temperatura de operación	-25 °C ~ +75 °C	-25 °C - +75°C	-20 °C a +75 °C
Antena	Interna	Interna	Interna

Fuente: Investigadora

Analizando los módulos, la accesibilidad y la compatibilidad de comunicación con la diadema Mindwave se han descartado la utilización de módulos xbee y del Módulo Flora Bluefruit LE. En consecuencia, se ha optado por el desarrollo de un módulo de comunicación bluetooth a partir del dispositivo HC-05, por cuestiones de su participación de tanto de Maestro como de Esclavo.

Una vez seleccionados cada uno de los componentes que darán vida al sistema de control del autómatas, se precisa el desarrollo de la etapa de software que se encargara de la gestión de los datos desde el punto de recolección de la señal eléctrica generada por el dispositivo de adquisición de señales neuronales, su acondicionamiento, procesamiento y posterior envío hacia el control del autómatas. Cumpliendo con el esquema principal presentados en la Figura 4.2.

4.3. Diseño del control de un Autómata con un FPGA de hardware libre

4.3.1. Adquisición de las señales neuronales

Mindwave de NeuroSky

Se determinó que el detector de señales neuronales elegido es Mindwave de la compañía de NeuroSky. Los auriculares de lectura, de ondas cerebrales pueden determinar diferentes estados emocionales con un solo sensor en la frente. Los datos de los biosensores son analizados para comprender cómo el usuario realiza los cambios de estados.

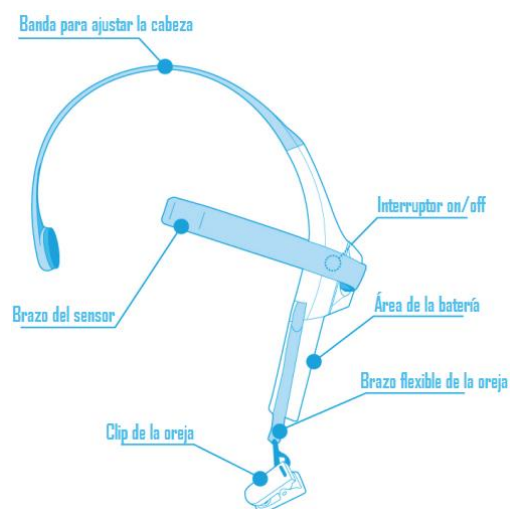


Figura 4.14 Partes externas de la diadema MindWave [53]

En la figura 4.15., se presenta los componentes en el interior del auricular MindWave.



Figura 4.15 Componentes internos de la diadema
Fuente: La investigadora

El circuito que se encuentra en el interior es presentado a continuación

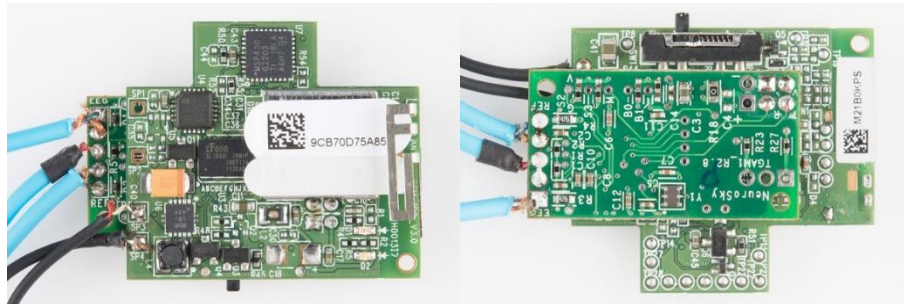


Figura 4.16 Vista del anverso y reverso del circuito Mindwave
Fuente: La investigadora

El gráfico de la parte mecánica y su espesor se representa en la figura 4.17.

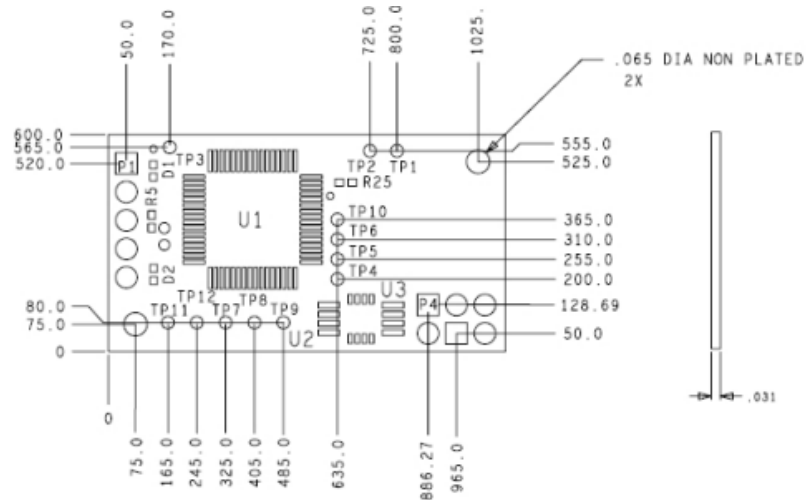


Figura 4.17 Representación del espesor en el circuito Mindwave
Fuente: La investigadora

Descripción de la tecnología NeuroSky

El último siglo de investigación en neurociencias ha aumentado considerablemente nuestro conocimiento sobre el cerebro y las señales eléctricas emitidas por las neuronas que se disparan en el cerebro. Los patrones y las frecuencias de estas señales eléctricas pueden medirse mediante la colocación de un sensor en el cuero cabelludo. La línea de productos de la empresa posee la tecnología NeuroSky ThinkGear™, que mide las señales eléctricas analógicas, comúnmente conocidas como ondas cerebrales, y los procesa en señales digitales.

ThinkGear

ThinkGear es la tecnología dentro de cada producto NeuroSky, que permite a un dispositivo interactuar con las ondas cerebrales que usa la diadema. Incluye el sensor que toca la frente, los puntos de contacto y la referencia situados en el clip de oreja, y el chip que procesa todos los datos. Tanto las ondas cerebrales primas y los Metros eSense (atención y meditación) se calculan en el chip ThinkGear.

eSense

eSense™ es un algoritmo de NeuroSky, para la interpretación de los estados mentales. Para calcular eSense, la tecnología NeuroSky ThinkGear amplía la señal de las ondas cerebrales, elimina el ruido del ambiente y del movimiento muscular. El algoritmo eSense se aplica entonces a la señal restante, dando como resultado los valores interpretados por el medidor de eSense. Se debe tener en cuenta que los valores del medidor eSense no describen un número exacto, pero sí un rango de la actividad cerebral.

Funcionamiento

Luz LED

La luz LED de la MindWave móvil tiene dos colores: rojo y azul.

Tabla 4.5 Significado led del MindWave

Luz	Estado de MindWave	Significado
Apagada	Alimentación apagada	MindWave está apagado o no tiene batería
Doble luz azul parpadeante	Modo de emparejamiento	MindWaveis listo para ser emparejado con un dispositivo

Rojo sólido	Desapareado	MindWave necesita ser sincronizado con el dispositivo
Solo una luz azul parpadeante	Listo	MindWave está esperando que un dispositivo se vincule
Azul sólido	Conectado	MindWave se está comunicando con un dispositivo
Doble Rojo intermitente	Batería baja	Reemplazar la batería MindWave

Fuente: Investigadora

4.3.2. Sistema de comunicación entre la adquisición de señales neuronales y el procesamiento de las mismas.

HC-05 FC-114

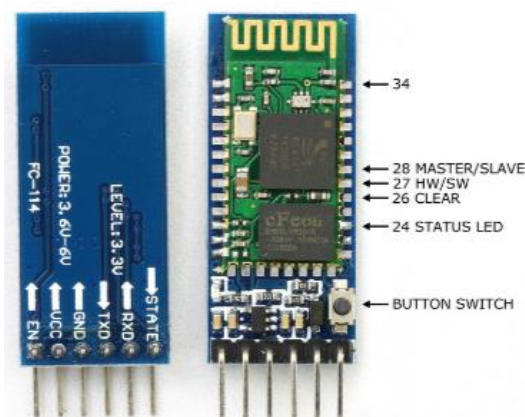


Figura 4.18 Especificaciones del módulo bluetooth [54]

Bluetooth hc-05

El módulo HC - 05 es una herramienta fácil de usar, diseñado para la configuración de la conexión serie inalámbrica.

Especificaciones:

Las características de hardware

- Sensibilidad típica de -80 dBm.
- Hasta + 4dBm RF potencia de transmisión
- Operación con energía baja de 1,8 a 3,6 V / S
- Control de PIO (Programación entrada/salida)
- Interfaz UART con velocidad de transmisión programable
- Con antena integrada
- Con el conector incluido

Las características del software

- Velocidad de transmisión por defecto : 38400
- Bits de datos : 8
- Bit de parada : 1
- Paridad: Sin paridad
- Control de datos: Si tiene.
- Velocidad soportada: 9600, 19200, 38400, 57600, 115200, 230400,460800.

Descripción de los pines

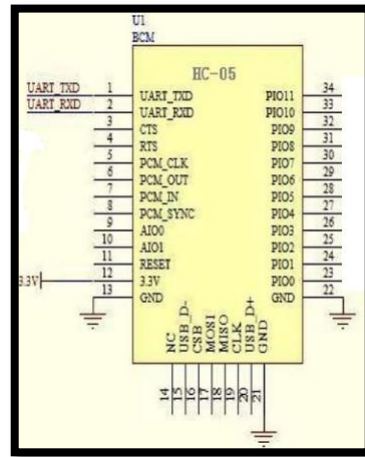


Figura 4.19 Descripción de pines en Bluetooth HC-05
Fuente: Investigadora

Tabla 4.6 Descripción de pines del Bluetooth HC-05 [54]

Nombre del pin	# de pin	Tipo de pin	Descripción
Tierra	13	VSS	Pin de tierra
	21		
	22		
3.3 VCC	12	3.3V	Suministro de voltaje
AIO0	9	Bi-Direccional	Línea programable de entrada y salida
AIO1	10		
PIO0	23	Bi-Direccional RX	Entrada y salida programable
PIO1	24	Bi-Direccional TX	Entrada y salida programable
PIO2	25	Bi-Direccional	Línea programable de entrada y salida
PIO3	26		
PIO4	27		
PIO5	28		
PIO6	29		

PIO7	30		
PIO8	31		
PIO9	32		
PIO10	33		
PIO11	34		
RESETB	11	CMOS de entrada, pull-up	Reestablecer en entrada baja
UART_RTS	4	CMOS de salida, tri-estable, pull-up	UART, solicitud de envío activa baja
UART_CTS	3	CMOS de entrada, pull-down	UART, listo para enviar bajo activo
UART_RX	2	CMOS de entrada, pull-down	UART, entrada de datos
UART_TX	1	CMOS de salida, estable pull-up	UART, salida de datos
SPI_MOSI	17	CMOS de entrada, pull-down	Entrada de datos de interfaz periférica
SPI_CSB	16	CMOS de entrada pull-up	Selección de chip de interfaz periférica en serie , bajo activo
SPI_CLK	19	CMOS de intrada, pull down	Reloj de la interfaz periférica en serie
SPI_MISO	18	CMOS de entrada, pull-down	Interfaz periférica en serie de salida de datos
USB_-	15	Bi-direccional	
USB_+	20	Bi-direccional	
NC	14		
PCM_CLK	5	Bi-direccional	Sincronización de datos de reloj
PCM_OUT	6	CMOS salida	Sincronización de datos de salida
PCM_IN	7	CMOS entrada	Sincronización de datos de entrada
PCM_SYNC	8	Bi-direccional	Sincronización estroboscópica síncrona de datos

Configuración de bluetooth

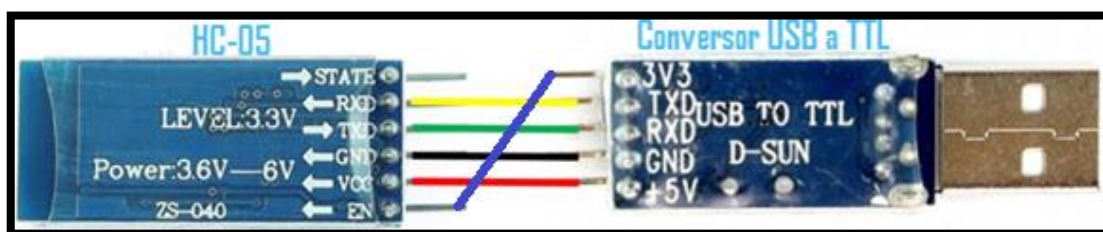







Figura 4.20 Conexión Módulo HC-05 y Conversor USB-TTL
Fuente: Investigadora

El módulo HC-05 se debe programar por medio de los comandos AT, para esto es necesario que el pin en Key, este High (alto), se conecta el pin Vcc del módulo HC-05 al pin 5V del conversor.

Al terminar el anterior paso e iniciar el programa (IDE Arduino), entrará en el modo de comandos AT.

Tabla 4.7 Conexión de pines entre el Módulo HC-05 y el conversor USB-TTL

Pin del módulo HC-05	Se conecta a:	Pin de Conversor USB TTL
RXD		TXD
TXD		RXD
GND		GND
VCC		+/- 5V
ENABLE		3V3

Fuente: Investigadora

- El pin STATE refleja la situación en la que se encuentra el módulo y por el momento de la configuración no se utilizará.

Modo AT, y programación

Para ingresar en modo AT, es necesario desconectar el conversor un momento y volverlo a conectar, se observará que el led del bluetooth se prende y se apagara cada dos segundos aproximadamente.

Se procede a ejecutar el IDE de Arduino, se ingresa en /herramientas/monitor serial, se modifica los valores de velocidad a “9600 baud” y a “Ambos NL y CR”.

Para comprobar el modo AT, se ingresa el comando “AT” y deberá la respuesta debe ser OK.



Figura 4.21 Respuesta de IDE Arduino a comandos AT

Fuente: Investigadora

Los siguientes comandos presentados en la tabla 4.8 sirven para una configuración básica del módulo, se registrará, el nombre, contraseña, modo de operación, etc.

Tabla 4.8 Comandos básicos de configuración

Comando	Descripción	Observación
AT+NAME?	Consulta el nombre del módulo	
AT+NAME=	Cambia el nombre por defecto	AT+NAME= Pantech
AT+ROLE?	Si contesta "0" está en modo esclavo. Si contesta "1" está en modo maestro.	
AT+ROLE=1	Cambia a modo maestro	Es el modo que se necesita para el proyecto
AT+PSWD?	Consulta la contraseña del HC-05	Por defecto suele venir "0000"
AT+UART?	Muestra la configuración	

Fuente: Investigadora

Comandos para la conexión con Mindwave

Los anteriores comandos, se realizan para una configuración básica, también se debe configurar el módulo bluetooth con los comandos indicados en la Tabla 4.8, para el reconocimiento de la diadema Mindwave de la empresa de Neurosky.

Para poder realizar la configuración, se debe verificar la serie del identificador único, de la diadema detectora de ondas cerebrales.

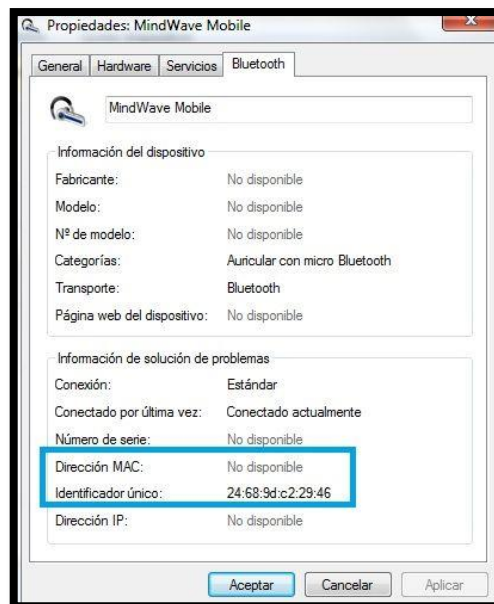


Figura 4.22 Propiedades de MindWave Mobile
Fuente: Investigadora

Se debe ejecutar los siguientes comandos en el siguiente orden:

Tabla 4.9 Comandos para configurar la paridad entre el Bluetooth y MindWave

Comando	Descripción
at+init\r\n	Inicializar la librería SPP (no se puede repetir)
at+iac=9e8b33\r\n	Pregunta si el dispositivo Bluetooth dispone de un código de acceso
at+class=0\r\n	Indaga el tipo de dispositivo Bluetooth
at+inqm=1, 9, 48\r\n	Arranca el modo de preguntar : 1) Intensidad de la señal RSSI. 2) Deja de preguntar si hay más de 9 respuestas de Bluetooth 3) Pregunta si el tiempo limitado es de $48 * 1,28 = 61.44s$.
at+inq\r\n	Pregunta los dispositivos que se encuentran alrededor del dispositivo.
at+pair= 2468,9d,c22946	Parea el módulo con el dispositivo Mindwave
at+link=2468,9d,c22946	Conecta los dos dispositivos

Fuente: Investigadora

Se desconecta el cable de key y se reinicia el módulo.

Culminada la configuración se realiza la comprobación de vínculo entre el bluetooth y el casco mindwave, se energiza y se pone la diadema en modo pareo para comprobar la conexión. Para que la vinculación sea exitosa el dispositivo de la empresa Neurosky debe tener su led en azul sólido.



Figura 4.23 Comprobación de Pareo entre Bluetooth y MindWave

Fuente: Investigadora

4.3.3. Procesamiento de señales neuronales

FPGA

Las FPGA´s (Field Programmable Gate Arrays) son dispositivos que permiten crear con mayor facilidad circuitos digitales, se pueden interpretar como una pizarra en blanco, un FPGA por sí solo no hace nada, todo depende de la programación que se realice en la misma. Una vez cargada la FPGA se comportará como el circuito digital diseñado.

Las FPGA´s a diferencia de un ASIC (Application Specific Integrate Circuit) donde el diseño del circuito se graba por una sola vez, en las tarjetas FPGA se puede volver a configurar tantas veces como sea necesario.

Mojo v3

El Mojo es una tarjeta de desarrollo FPGA, ha sido diseñada para que el entendimiento de la misma y su programación sean fácil.

El objetivo del Mojo es empezar con la programación de FPGA y el diseño digital lo más fácil posible.

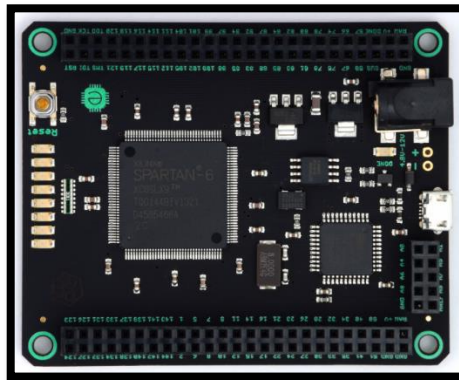


Figura 4.24 Mojo V3
Fuente: Investigadora

Las características incluyen Mojo:

- Spartan FPGA 6 XC6SLX9
- 84 pines IO digitales
- 8 entradas analógicas
- 8 LED de propósito general

- 1 Botón de reinicio
- 1 LED para mostrar cuando el FPGA está configurado correctamente
- La regulación de voltaje, puede manejar 4.8V - 12V
- Un microcontrolador (ATmega32U4) utilizado para la configuración de la FPGA, comunicaciones USB, y la lectura de los pines analógicos
- Gestor de arranque, compatible con Arduino para controlar el microcontrolador.
- Memoria flash integrada para almacenar el archivo de configuración de la FPGA

Análisis de la programación.

En la figura 4.25 se plantea un diagrama de bloques, el cual describe los pasos que se debe seguir para la programación de la comunicación entre el Mojo V3 y el módulo bluetooth, además de la salida de control para el autómata.

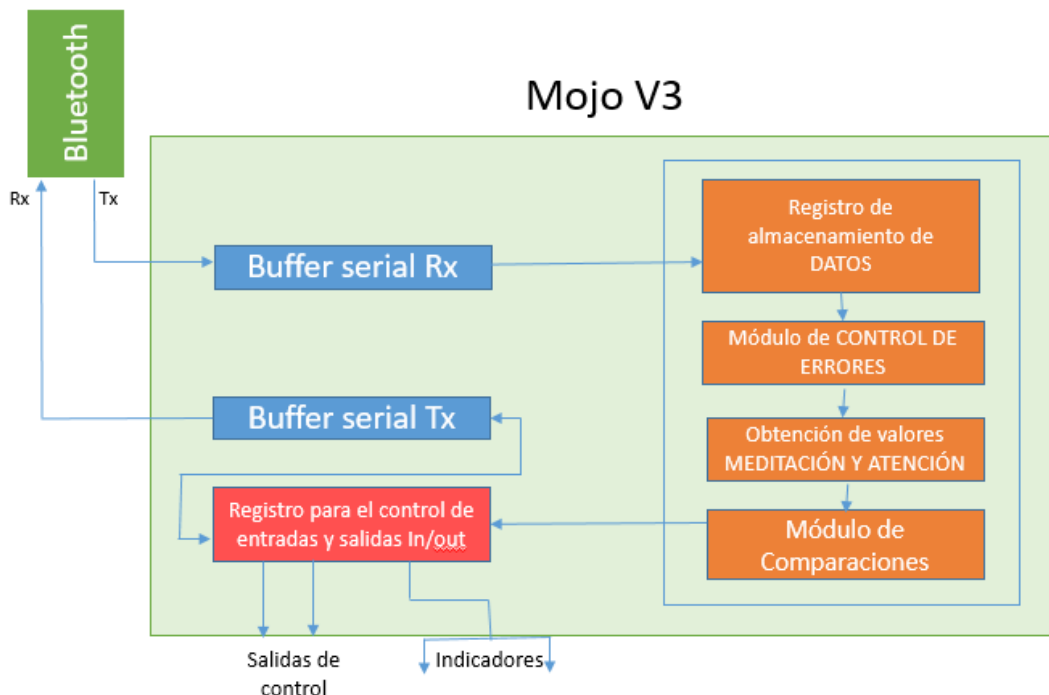
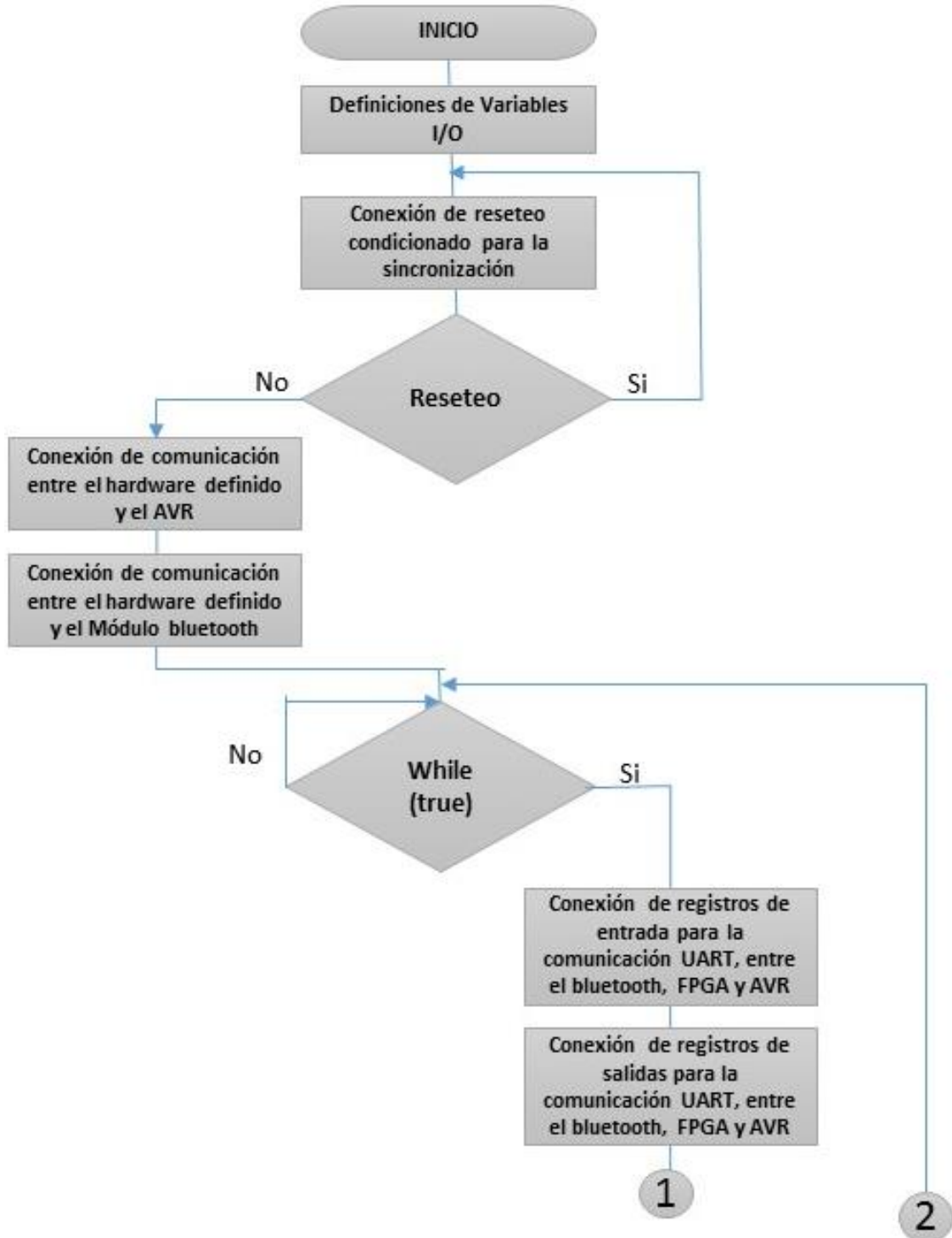


Figura 4.25. Diagrama de bloques de la programación en la tarjeta FPGA

Fuente: Investigadora

Diagrama de flujo

Al tener claro cada paso que se realiza en la programación, se extiende la descripción en un diagrama de flujo indicando como se realiza la lectura de datos, y las decisiones tomadas con cada situación presentada.



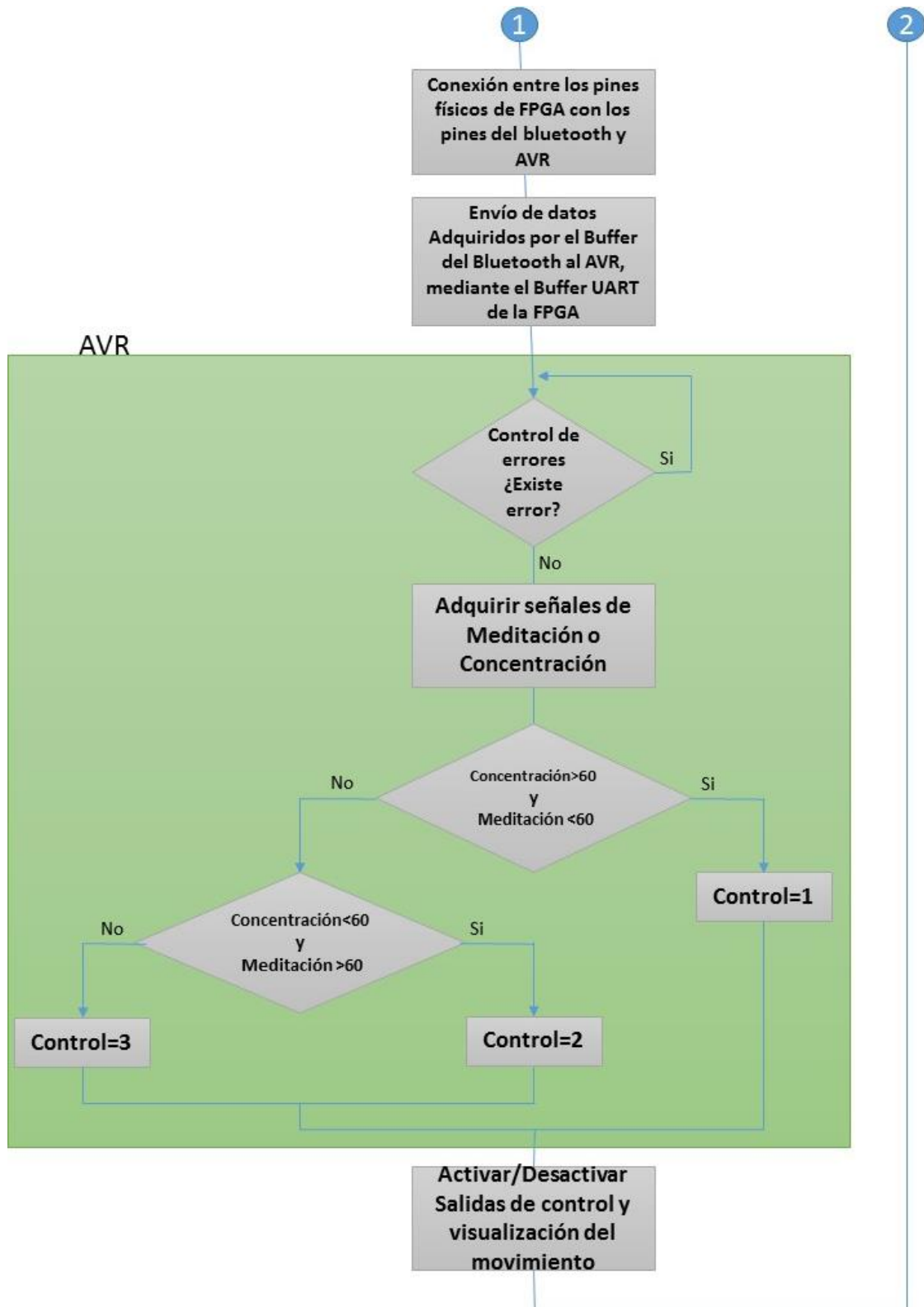


Figura 4.26. Diagrama de flujo de la lógica de programación
Fuente: Investigadora

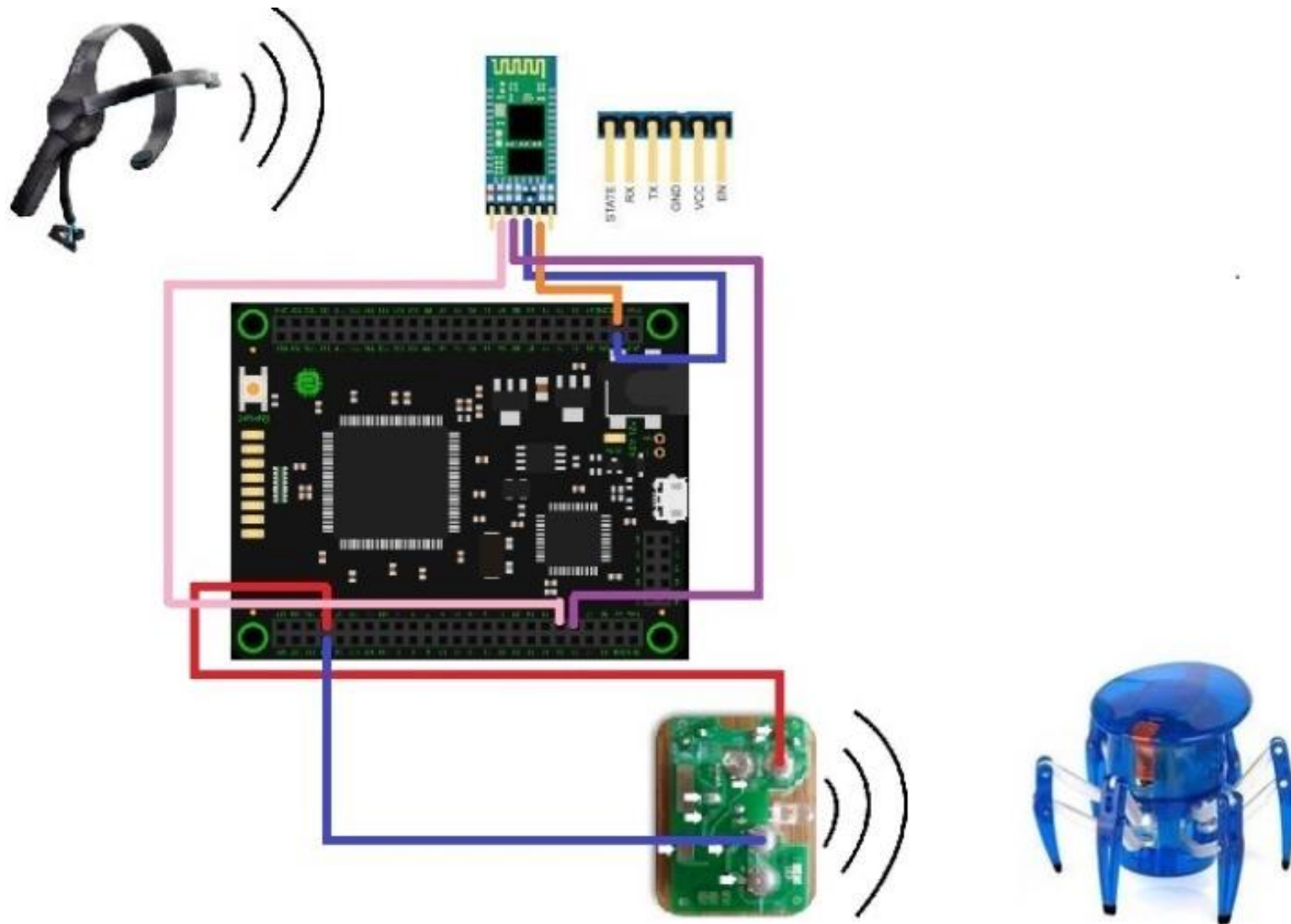


Figura 4.27 Esquema de las conexiones del sistema

En la figura 4.27 se representa las conexiones físicas del sistema, en la cual se observa las fases inalámbricas y alámbricas, los pines utilizados tanto de entrada como de salida, y la función de los componentes antes analizados.

Al terminar el análisis, de los pasos necesarios y observar las conexiones físicas para la programación se concluye que el proyecto en el FPGA Mojo V3, se realiza en dos partes y con diferentes softwar.



Figura 4.28 Descripción de la programación en Mojo V3
Fuente: Investigadora

SPARTAN-6

La familia FPGA Spartan®-6 de Xilinx ofrece un equilibrio óptimo de bajo riesgo, bajo costo, bajo consumo y rendimiento para aplicaciones sensibles al costo. Estos FPGA utilizan una tecnología de proceso de 45 nm de bajo consumo comprobado. Además, la serie Spartan-6 ofrece una tecnología de gestión avanzada de energía, hasta a 150 k de celdas lógicas, bloques integrados PCI Express®, soporte de memoria avanzada, partes DSP de 250 MHz y transceptores de baja potencia de 3.2 Gbps. [55]

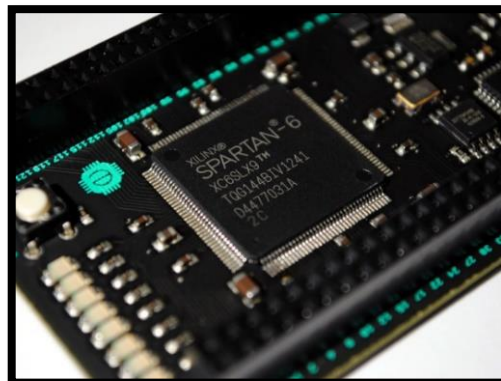


Figura 4.29 Chip SPARTAN-6
Fuente: Investigadora

Programación para la tarjeta FPGA SPARTAN-6 utilizando Lucid

Introducción a Lucid

Para empezar a trabajar con lucid que necesita el IDE Mojo. También se tiene que instalar la herramienta de Xilinx ISE llamada para poder construir sus proyectos y programar el FPGA.

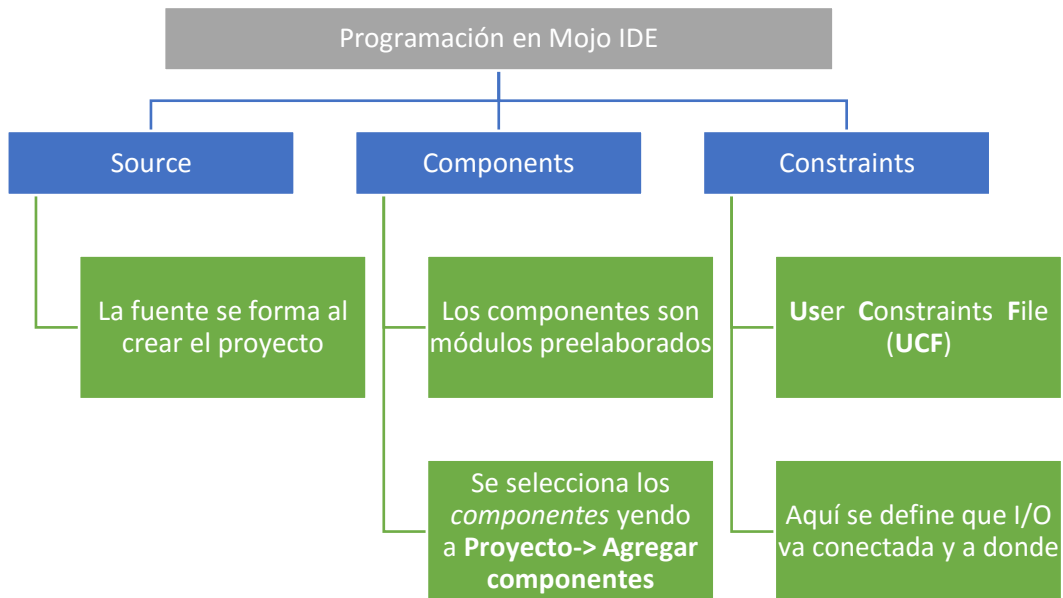


Figura 4.30 Partes de la programación en mojo IDE

Fuente: Investigadora

Constraints

Los Constraints son definiciones específicas de Entradas y salidas físicas que se van a utilizar de la tarjeta.

La Figura 4.31., indica el entorno en el cual se programa el FPGA, como se puede observar el segmento que constituye los Constraints tiene dos archivos, mojo.ucf y deb_bt.ucf.

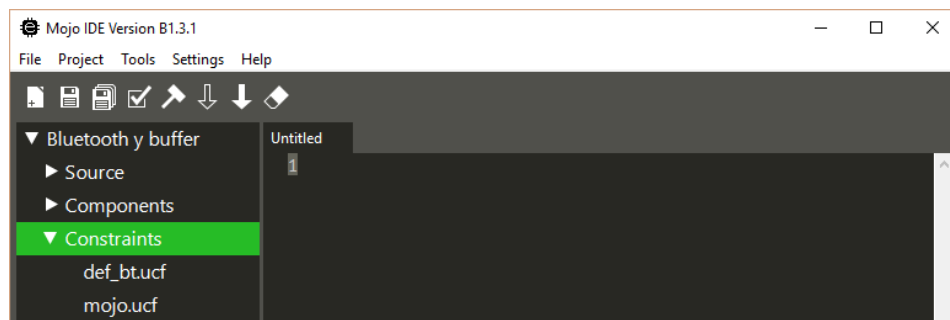


Figura 4.31 Entorno y descripción de los Constraints en mojo IDE

Fuente: Investigadora basado en IDE Mojo

mojo.ufc: es el archivo por defecto que se crea cuando se comienza un nuevo proyecto, su configuración es predeterminada.

deb_bt.ufc, el código completo se encuentra en el Anexo O: este archivo es creado para la realización del proyecto, (Click derecho en Constraints -> New constraints -> user constraints), el código tipado para el desarrollo del proyecto en este archivo se muestra a continuación.

```
NET "bt_tx" LOC = P34 | IOSTANDARD = LVTTTL; //conectar al pin de transmisión del módulo bluetooth
NET "bt_rx" LOC = P32 | IOSTANDARD = LVTTTL; //conectar al pin de recepción del módulo bluetooth
NET "control<0>" LOC = P126 | IOSTANDARD = LVTTTL; //pin para controlar el control remoto del autómeta
NET "control<1>" LOC = P123 | IOSTANDARD = LVTTTL; //pin para controlar el control remoto del autómeta
```

Components:

En el segmento de los componentes se declaran las interfaces que se van a utilizar, en esta sección no puede ser declarados nuevos componentes, por lo que se trabaja con los predeterminados por el desarrollador.

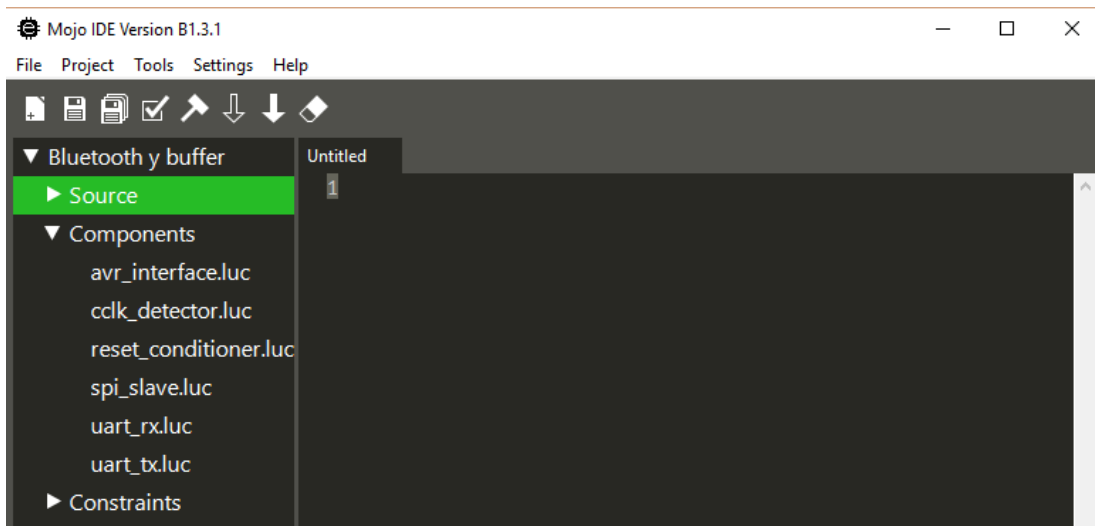


Figura 4.32 Entorno y descripción de los Components en mojo IDE

Fuente: Investigadora basado en IDE Mojo

avr_interface.luc, el código completo se encuentra en el Anexo I

Es la interfaz que define el vínculo FPGA y el AVR, en esta sección se define como se comporta el FPGA para comunicarse con el AVR.

```

//inputs, conexión de entradas
cclk_detector.cclk = cclk;
spi_slave.sck = spi_sck;
spi_slave.mosi = spi_mosi;
spi_slave.data_in = hff;
spi_slave.ss = spi_ss;
uart_rx.rx = rx;
uart_tx.data = tx_data;
uart_tx.new_data = new_tx_data;
uart_tx.block = busy.q;
block.d = c{block.q[2:0], tx_block};

// outputs, conexión de salidas
new_sample = newSampleReg.q;
sample = sampleReg.q;
sample_channel = sampleChannelReg.q;

tx_busy = uart_tx.busy;
rx_data = uart_rx.data;
new_rx_data = uart_rx.new_data;

spi_channel = cclk_detector.ready ? channel : bzzzz;
spi_miso = cclk_detector.ready && !spi_ss ? spi_slave.miso : bz;
tx = cclk_detector.ready ? uart_tx.tx : bz;

```

cclk_detector.luc, el código completo se encuentra en el Anexo J

En la sección de cclk_detector, se realiza la sincronía entre FPGA y AVR para la configuración tanto del FPGA o el AVR.

```

const CTR_SIZE = $clog2(CLK_FREQ/5000); // need to wait about 200uS

.clk(clk), .rst(rst) {
  dff ctr[CTR_SIZE];
}

```

reset_conditioner.luc, el código completo se encuentra en el Anexo K

En esta sección el componente reset_conditioner, lee un pin y envía el reseteo al sistema q se está trabajando.

```

dff stage[STAGES] (.clk(clk), .rst(in), #INIT(STAGESx{1}));

```

spi_slave.luc, el código completo se encuentra en el Anexo L

Método utilizado para SPI, que indica al AVR que canal análogo se va a utilizar

```

// connect to buffer output
miso = miso_reg.q;
done = done_reg.q;
data_out = data_out_reg.q;

// read in buffered inputs

```

```

ss_reg.d = ss;
mosi_reg.d = mosi;
sck_reg.d = c{sck_reg.q[0], sck}; // save old sck

```

```

done_reg.d = 0; // default to not done

```

uart_rx.luc, el código completo se encuentra en el Anexo M

Comunicación UART de recepción, en esta sección se programa según el protocolo UART, la recepción de la trama de datos

```

case (state.q) {
state.IDLE:
bitCtr.d = 0; // reset counter
ctr.d = 0; // reset counter
if (rx.d.q == 0) // if rx line is low (start bit)
state.d = state.WAIT_HALF; // switch state

state.WAIT_HALF:
ctr.d = ctr.q + 1; // increment the counter
if (ctr.q == (CLK_PER_BIT >> 1)) { // if counter is the max value
ctr.d = 0; // reset counter
state.d = state.WAIT_FULL; // switch state
}

state.WAIT_FULL:
ctr.d = ctr.q + 1; // increment counter
if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
savedData.d = c{rx.d.q, savedData.q[7:1]}; // shift in new data
bitCtr.d = bitCtr.q + 1; // increment bit counter
ctr.d = 0; // reset counter
if (bitCtr.q == 7) { // if we have received 8 bits
state.d = state.WAIT_HIGH; // switch state
newData.d = 1; // signal new byte received
}
}

state.WAIT_HIGH:
if (rx.d.q == 1) // wait for input to go high (idle)
state.d = state.IDLE; // switch state

```

uart_tx.luc, el código completo se encuentra en el Anexo N

Comunicación UART de transmisión, en esta sección se programa según el protocolo UART, la transmisión de la trama de datos

```

case (state.q) { // FSM
state.IDLE:
txReg.d = 1; // idle high (UART standard)
if (!blockFlag.q) {
busy = 0; // not busy
bitCtr.d = 0; // reset counter
ctr.d = 0; // reset counter
if (new_data) { // request to send data?

```

```

        savedData.d = data; // save the data
        state.d = state.START_BIT; // switch states
    }
}

state.START_BIT:
    ctr.d = ctr.q + 1; // increment counter
    txReg.d = 0; // start bit is low
    if (ctr.q == CLK_PER_BIT - 1){ // if ctr is the max value
        ctr.d = 0; // reset counter
        state.d = state.DATA; // switch states
    }

state.DATA:
    txReg.d = savedData.q[bitCtr.q]; // output the data bit
    ctr.d = ctr.q + 1; // increment counter
    if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
        ctr.d = 0; // reset counter
        bitCtr.d = bitCtr.q + 1; // increase bit counter
        if (bitCtr.q == 7) // if we have sent all the bits
            state.d = state.STOP_BIT; // switch states
    }

state.STOP_BIT:
    txReg.d = 1; // stop bit is high
    ctr.d = ctr.q + 1; // increase counter
    if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
        state.d = state.IDLE; // switch states
    }

default: state.d = state.IDLE; // if state is invalid reset to idle

```

Source

En el segmento de la fuente, se ha creado por defecto el archivo `mojo_top.luc` donde se visualiza toda la programación uniendo las anteriores secciones, en la fuente se observa tres archivos más, `bt_interface.luc`, `bt_rx.luc`, `bt_tx.luc`, estos son componentes que han sido creados en este segmento, ya que en la sección “Components”, no se pueden crear nuevos y ante la necesidad de componentes para la comunicación del bluetooth.

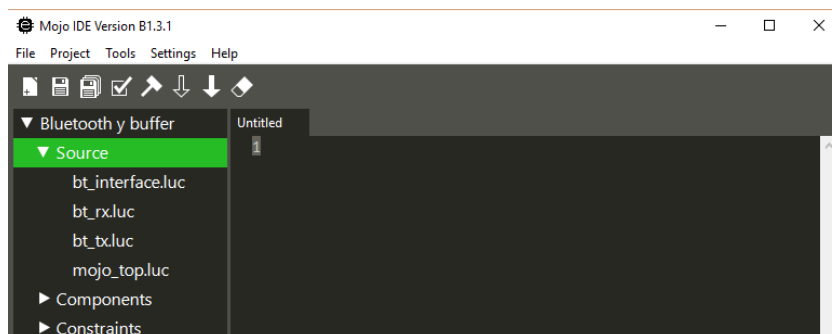


Figura 4.33 Entorno y descripción de los Source en mojo IDE
Fuente: Investigadora basado en IDE Mojo

Componentes creados en la fuente

bt_interface.luc, el código completo se encuentra en el Anexo F

```
// Interfaz de TX Serial
input tx_data[8], // registro para datos para enviar
input new_tx_data, // Bandera de deteccion de dato nuevo (1 = dato nuevo)
output tx_busy, // Bandera para deteccion de transmisor ocupado (1 =
ocupado)
input tx_block, // Bandera para el bloqueo del transmisor (1 = bloqueo)
conectar al bt_rx_busy

// Interfaz de RX Serial
output rx_data[8], // registro para datos recibidos
output new_rx_data // Bandera de deteccion de nuevo dato recibido (1 = dato
nuevo)
```

bt_rx.luc, el código completo se encuentra en el Anexo G

```
state.IDLE:
    bitCtr.d = 0; // reset counter
    ctr.d = 0; // reset counter
    if (rx.d.q == 0) // if rx line is low (start bit)
        state.d = state.WAIT_HALF; // switch state

state.WAIT_HALF:
    ctr.d = ctr.q + 1; // increment the counter
    if (ctr.q == (CLK_PER_BIT >> 1)) { // if counter is the max value
        ctr.d = 0; // reset counter
        state.d = state.WAIT_FULL; // switch state
    }

state.WAIT_FULL:
    ctr.d = ctr.q + 1; // increment counter
    if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
        savedData.d = c{rx.d.q, savedData.q[7:1]}; // shift in new data
        bitCtr.d = bitCtr.q + 1; // increment bit counter
        ctr.d = 0; // reset counter
        if (bitCtr.q == 7) { // if we have received 8 bits
            state.d = state.WAIT_HIGH; // switch state
            newData.d = 1; // signal new byte received
        }
    }

state.WAIT_HIGH:
    if (rx.d.q == 1) // wait for input to go high (idle)
        state.d = state.IDLE;
```

bt_tx.luc, el código completo se encuentra en el Anexo H

```
state.IDLE:
    txReg.d = 1; // idle high (UART standard)
    if (!blockFlag.q) {
        busy = 0; // not busy
        bitCtr.d = 0; // reset counter
        ctr.d = 0; // reset counter
```

```

    if (new_data) {          // request to send data?
        savedData.d = data; // save the data
        state.d = state.START_BIT; // switch states
    }
}

state.START_BIT:
    ctr.d = ctr.q + 1;      // increment counter
    txReg.d = 0;           // start bit is low
    if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
        ctr.d = 0;         // reset counter
        state.d = state.DATA; // switch states
    }

state.DATA:
    txReg.d = savedData.q[bitCtr.q]; // output the data bit
    ctr.d = ctr.q + 1;           // increment counter
    if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
        ctr.d = 0;             // reset counter
        bitCtr.d = bitCtr.q + 1; // increase bit counter
        if (bitCtr.q == 7) // if we have sent all the bits
            state.d = state.STOP_BIT; // switch states
    }

state.STOP_BIT:
    txReg.d = 1;             // stop bit is high
    ctr.d = ctr.q + 1;      // increase counter
    if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
        state.d = state.IDLE; // switch states
    }
}

```

Fuente

mojo_top.luc, el código completo se encuentra en el Anexo E

Declaración de las entradas y salidas para el control del módulo HC-05

```

//*****control hc05*****
input bt_tx, // BT TX se conecta directamente al pin de transmisión del módulo
(def_bt.ufc)
output bt_rx, // BT RX se conecta directamente al pin de recepción del módulo
(def_bt.ufc)
input bt_rx_busy, // BT Bandera que indica que el buffer de comunicación está
lleno

```

Pines de salida para el control del autómata, aunque no se utilizan el array completo de bits, se realiza esto para que no ocurra una pérdida de información

```

//*****pines de salida para control*****
output control [8] // Array de salidas para el control del Autómata

```

Conexión de entradas de los pines para el bluetooth

```

//////////BLUETOOTH//////////

```

```

    bt.cclk = cclk;      // conecta reloj de configuración general con el que se necesita
                        // en el módulo de la comunicación BT.
    bt.rx = bt_tx;      // Conecta el pin de la FPGA al que está conectado dispositivo
                        // bluetooth con la interfaz del HARDWARE, DEFINIDO en el módulo bt_interface.

```

Conexión de entradas de los pines para el AVR

```

/////////////////////////////////AVR/////////////////////////////////
    avr.cclk = cclk;      // conecta reloj de configuración general con el que se
                        // necesita en el módulo del AVR.
    avr.spi_ss = spi_ss; // avr.spi_ss = bz; // conecta pines físicos con los del
                        // HARDWARE DEFINIDO en avr_interface
    avr.spi_mosi = spi_mosi; // avr.spi_mosi = bz; // NO SE UTILIZA EN LA
                        // APLICACION DEL PRESENTE
    avr.spi_sck = spi_sck; // avr.spi_sck = bz; // pueden dejarse definidos o setear
                        // bz para desactivar el mismo
    avr.rx = avr_tx;     // conectar pin físico de TX del AVR con el RX del HARDWARE
                        // DEFINIDO en avr_interface

```

Conexión de salidas de los pines para el bluetooth

```

/////////////////////////////////BLUETOOTH/////////////////////////////////
    bt_rx = bt.tx;      // conectar pin fisico de RX del bluetooth con el TX del
                        // HARDWARE DEFINIDO en bt_interface
    bt.tx_block = bt_rx_busy; // bloquea cuando el modulo este recibiendo

```

Conexión de salidas de los pines para el AVR

```

/////////////////////////////////AVR/////////////////////////////////
    spi_miso = avr.spi_miso; // conectar pin fisico MISO de comunicacion SPI con el
                        // del HARDWARE DEFINIDO
    spi_channel = bzzzz;    // spi_channel = avr.spi_channel; puede conectarse el
                        // pin fisico o desactivarse al no usarse
    avr_rx = avr.tx;       // conectar pin fisico de RX del AVR con el TX del
                        // HARDWARE DEFINIDO en avr_interface
    avr.channel = hf;      // desactivar ADC
    avr.tx_block = avr_rx_busy; // Se bloquea la transmision cuando el AVR se encuentre
                        // ocupado

```

Conexión entre los transmisores y receptores del bluetooth y el ATMEGA

```

//*****conectar tx con rx*****

//genero un eco (es decir conectar la TX del AVR con el RX del mismo para
//obtener de retorno el dato que se envía)
    avr.tx_data = avr.rx_data;
    avr.new_tx_data = avr.new_rx_data;

/////////////////////////////////bluetooth/////////////////////////////////

    bt.new_tx_data = bz;      // desactivado (no envía)

```

```

bt.tx_data = bz;           // desactivado (no envíe)
bt.tx_block = bz;        // desactivado (no envíe)

////////////////////////////////////
avr.new_tx_data = bt.new_rx_data; // si se detecta un dato en el Serial del
Bluetooth enviar para el AVR(habilitación TX)
avr.tx_data = bt.rx_data; // dato del buffer del bluetooth enviarlo por el buffer
del SerialAVR
avr.tx_block = avr.tx_busy; // bloquear la transmisión mientras se envíen todos
los datos(evita colapsos)Importante

```

ATmega32U4

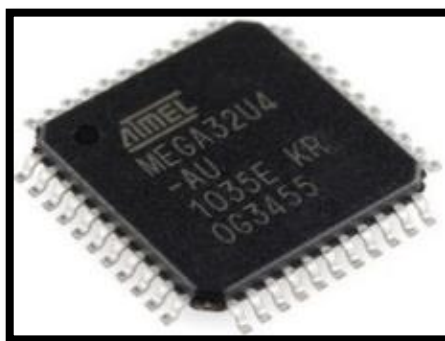


Figura 4.34 Chip ATmega32U4
Fuente: Investigadora

El ATmega32U4 es un CMOS de bajo consumo de 8 bits basado en el microcontrolador AVR mejorado arquitectura RISC. Mediante la ejecución de instrucciones de gran alcance en un solo ciclo de reloj, el ATmega32U4 logra rendimientos se acerca a 1 MIPS por MHz que permite al diseñador del sistema para optimizar el consumo de energía en comparación con la velocidad de procesamiento.

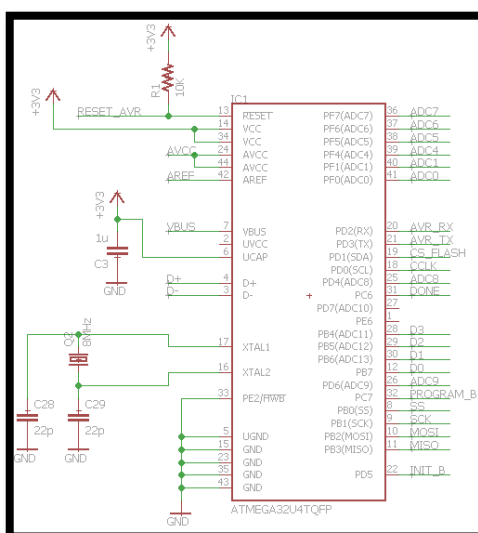


Figura 4.35 Descripción de los pines de ATmega32U4
Fuente: Investigadora

Programación en ATmega32U4

La programación en el Chip ATmega32U4, se lo realiza en el IDE de Arduino modificado, el cual consta de 6 archivos, que se explican en la figura 4.36.

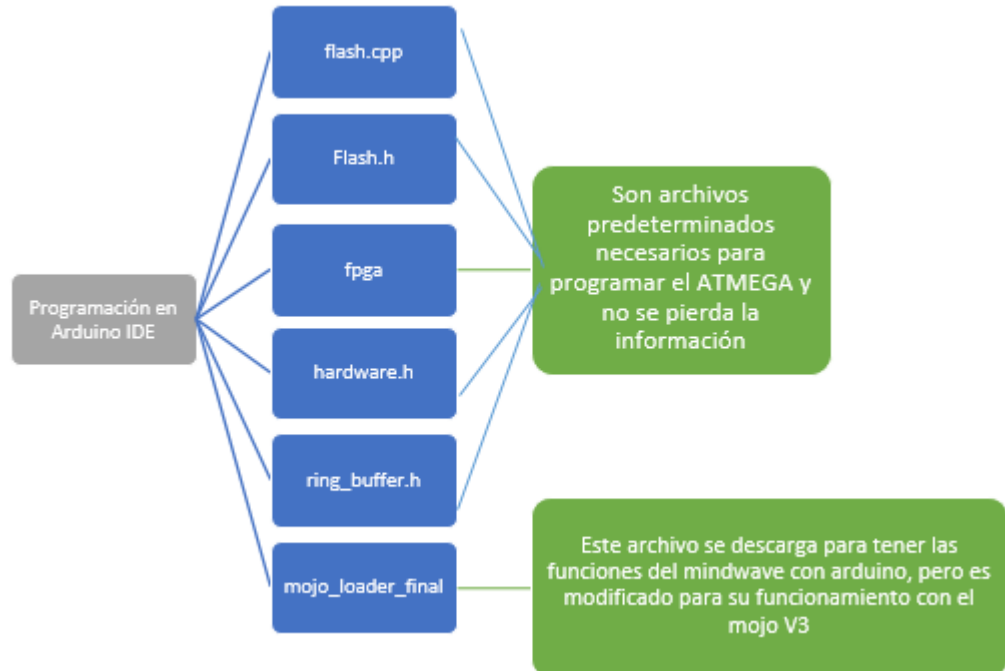


Figura 4.36 Descripción de los archivos en Arduino IDE

Fuente: Investigadora

mojo_loader_final, el código completo se encuentra en el Anexo P

Las modificaciones realizadas, se hacen sobre el casco de Mindwave, para la comunicación con el ATMEGA.

```
/*******MINDWAVE CASCO*****  
#include <SoftwareSerial.h>  
#define DEBUGOUTPUT 0  
  
// checksum variables  
byte generatedChecksum = 0;  
byte checksum = 0;  
int payloadLength = 0;  
byte payloadData[64] = {0};  
byte poorQuality = 0;  
byte attention = 0;  
byte meditation = 0;  
  
// system variables  
long lastReceivedPacket = 0;  
boolean bigPacket = false;  
  
byte ReadOneByte()  
{
```

```

int ByteRead;

while(!Serial1.available());
ByteRead = Serial1.read();

#if DEBUGOUTPUT
// Serial1.print((char)ByteRead); // echo the same byte out the USB serial (for debug
purposes)
#endif

return ByteRead;
}

```

El siguiente tipeado de comandos es necesario para adquirir los datos del casco, revisar los bits de sincronismo y cálculo del checksum, para conocer si existe o no existe la comunicación, obtener los datos necesarios y conocidos.

```

//*****Adquiere los datos del CASCO
MINDWAVE*****

if(ReadOneByte() == 170)
{
if(ReadOneByte() == 170)
{
payloadLength = ReadOneByte();
if(payloadLength > 169) //Payload length cannot be greater than 169
return;

generatedChecksum = 0;
for(int i = 0; i < payloadLength; i++)
{
payloadData[i] = ReadOneByte(); //Read payload into memory
generatedChecksum += payloadData[i];
}

checksum = ReadOneByte(); //Read checksum byte from stream
generatedChecksum = 255 - generatedChecksum; //Take one's compliment of
generated checksum

if(checksum == generatedChecksum)
{
poorQuality = 200;
attention = 0;
meditation = 0;

for(int i = 0; i < payloadLength; i++) // Parse the payload
{
switch (payloadData[i])
{
case 2:
i++;
poorQuality = payloadData[i];
}
}
}
}
}

```

```

        bigPacket = true;
        break;
    case 4:
        i++;
        attention = payloadData[i];
        break;
    case 5:
        i++;
        meditation = payloadData[i];
        break;
    case 0x80:
        i = i + 3;
        break;
    case 0x83:
        i = i + 25;
        break;
    default:
        break;
    }
}
#endif
#if !DEBUGOUTPUT
// si el checksum es el correcto se puede acceder a los datos que envía el mindwave
// variables = "poorQuality" "attention" "meditation" "lastReceivedPacket"
if(bigPacket)
{
    Serial.print("CalidadBaja: ");
    Serial.print(poorQuality, DEC);
    Serial.print(" Atencion: ");
    Serial.print(attention, DEC);
    Serial.print(" Meditacion: ");
    Serial.print(meditation, DEC);
    Serial.print(" Tiempo entre paquetes: ");
    Serial.print(millis() - lastReceivedPacket, DEC);
    lastReceivedPacket = millis();
    Serial.print("\n");

    if(attention>=60 && meditation<60)
    {
        //Serial.print(1,DEC);
        Serial1.print(1,DEC);
    }
    else if(attention<60 && meditation>=60)
    {
        //Serial.print(2,DEC);
        Serial1.print(2,DEC);
    }
    else
    {
        //Serial.print(3,DEC);
        Serial1.print(3,DEC);
    }
}
}
#endif

```

```

bigPacket = false;
}
}
}

```

La configuración de las salidas se representa en una trama de 8 bits, los mismos que son visualizados con 8 leds como se observa en la figura 4.37, de los cuales se observa en los pines 1 y 2 sus cambios de estado, según sea la adquisición: meditación o concentración.



Figura 4.37 Leds que muestran las salidas y el modo de funcionamiento
Fuente: Investigadora basado en IDE Mojo

4.3.4. Sistema de comunicación entre el procesador de señales y el autómata Control

Para el sistema de control del autómata se realizarán modificaciones físicas para que pase de un control manual, a un control automático, el cual utilice las dos señales que van a ser enviadas por el procesador de datos para las funciones de girar y avanzar, y que la alimentación de voltaje, se realice con la tarjeta de Mojo V3, deshabilitando la utilización de pilas.



Figura 4.38 Placa de control sin modificar
Fuente: Investigadora



Figura 4.39 Modificación en señales de entrada
Fuente: Investigadora



Figura 4.40 Modificación de la alimentación
Fuente: Investigadora

4.3.5. Receptor de señales

Tanto el transmisor de mano y el receptor en el robot se basan en la AT88EB un microcontrolador de chip (Alpha Microelectrónica Corp), con el propio robot también contiene un controlador de ST1155A H Bridge.

Autómata



Figura 4.41 Autómata con forma de araña [56]

El autómata tiene la capacidad de girar su cabeza y caminar en todas las direcciones, tiene un control a distancia para que pueda ser guiado.

Tiene cuatro entradas lógicas, dos para cada motor. Para cada motor, una entrada controla la dirección y el otro gira el motor encendido y apagado.

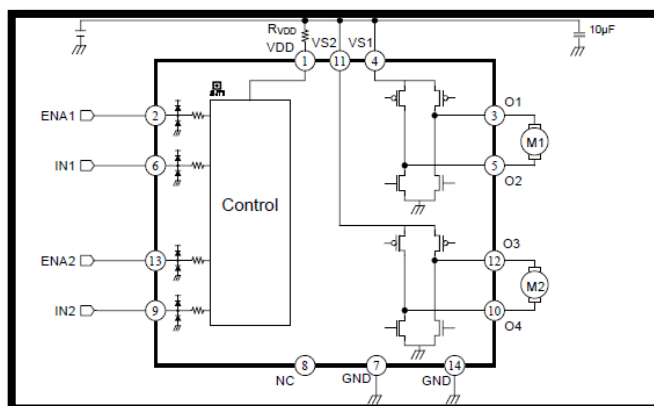


Figura 4.42 Diagrama del controlador de motor que se utiliza el autómata [56]

Para controlar cada motor, sólo tiene que conducir una señal lógica en el pin apropiado, para seleccionar la dirección y una señal lógica "alta" en el pasador ENA apropiada para arrancar el motor.

El poder eléctrico, se ejecuta con 4,5 V, suministrada desde tres pilas de 1,5V cada una.

4.4.Pruebas de funcionamiento

La comprobación del funcionamiento del sistema de control, describe la comunicación en cada una de sus partes, el desempeño en la adquisición de ondas neuronales y el control del autómeta, además se puede observar que el sistema cumple con las características propuestas en la presente investigación.

El sistema está diseñado para que sea cómodo y totalmente inalámbrico en cada una de sus etapas, la figura 4.38. Indica la colocación de la diadema Mindwave.



Figura 4.43 Colocación MindWave
Fuente: Investigadora

4.4.1. Pruebas del funcionamiento del prototipo para el control del autómeta

Las pruebas realizadas sobre el funcionamiento del sistema de control del autómeta, ayudan a verificar la velocidad de respuesta en diferentes individuos. La tabla 4.10, indica el número de personas que realizaron las pruebas y sus características.

Tabla 4.10. Características de los usuarios

Características Usuarios	Género	Edad	Comprensión de uso del sistema			Control para el cambio de estados		
			Fácil	Medio	Difícil	Fácil	Medio	Difícil
Persona 1	Masculino	8 años		x		x		
Persona 2	Masculino	10 años	x			x		
Persona 3	Masculino	21 años	x				x	
Persona 4	Masculino	25 años	x				x	
Persona 5	Femenino	25 años	x				x	
Persona 6	Femenino	33 años	x			x		
Persona 7	Masculino	35 años		x			x	
Persona 8	Femenino	40 años		x				x
Persona 9	Femenino	49 años		x			x	
Persona 10	Masculino	83 años		x				x

Fuente: Investigadora

Se realiza las pruebas durante 20 segundos, indicando a los usuarios, que deben concentrarse en mover el autómeta, exactamente que avance, los resultados de los mismos se presentan en la figura 4.44.

Para realizar un análisis cuantitativo de los estados, se ha valorado cada uno de ellos de la siguiente manera.

Estado	Valor
Meditación	1
Neutro	2
Concentración	3

Persona 1																				
Segundos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Estado	2	3	2	1	2	2	1	1	1	1	1	1	3	3	3	3	3	3	3	3

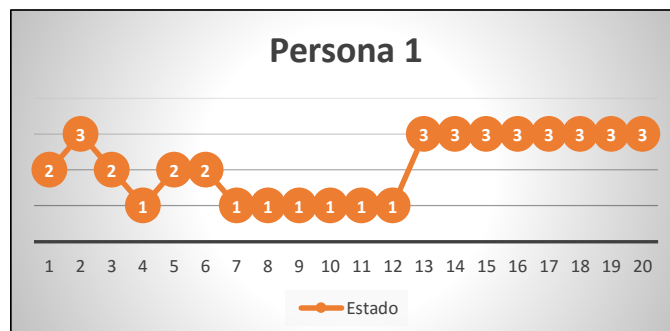


Figura 4.44. Gráfico de datos obtenidos del funcionamiento en la persona 1

Fuente: Investigadora



Figura 4.45. Prueba de funcionamiento y conexión con Labview en la persona 1
Fuente: Investigadora

Las figuras 4.45 y 4.47 Muestra el comportamiento de las señales en alfa y beta, de manera gráfica y texto, la interpretación de acuerdo a la concentración y meditación, el estado de la señal recibida y el puerto que reconoce la presencia de la diadema Mindwave como dispositivo transmisor.

La persona 1 Muestra un alto grado de estabilidad en el estado de atención, aunque al principio de las pruebas haya mostrado un estado de meditación en un corto periodo. Por lo cual se concluye, que con un poco de motivación realiza la tarea de concentrarse y mantener el autómata en movimiento

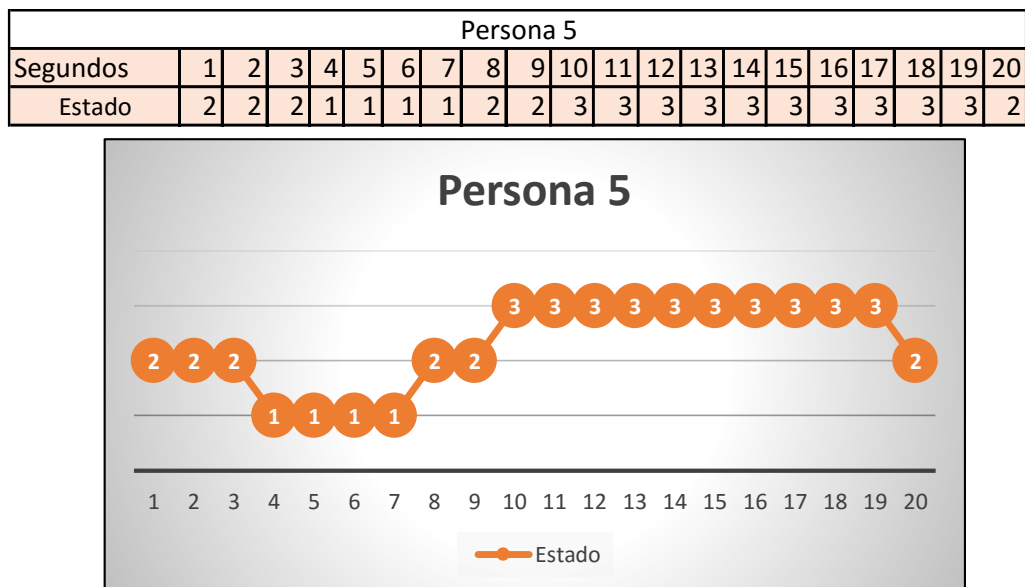


Figura 4.46. Gráfico de datos obtenidos del funcionamiento en la persona 5
Fuente: Investigadora

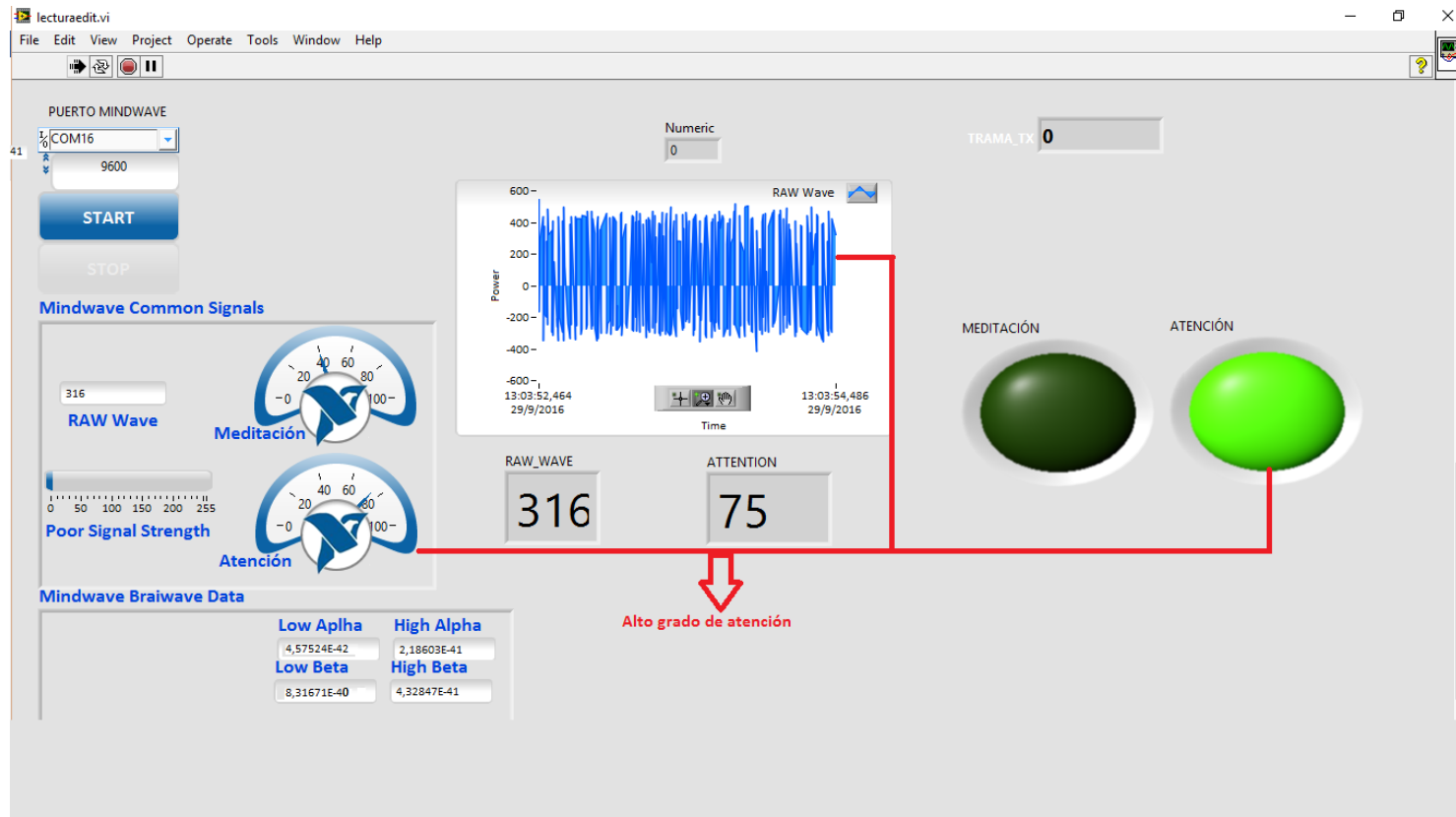


Figura 4.47. Prueba de funcionamiento y conexión con Labview en la persona 5
Fuente: Investigadora

La persona 5 a diferencia de la persona 1, se demora en realizar el movimiento de avanzar, debido a que no llega a la cumbre de la concentración, hasta después de un tiempo ligeramente largo, pero termina en un estado de concentración. Por lo cual se concluye, que la persona necesita más motivación e indicaciones para que pueda realizar el ejercicio propuesto.

4.5. Presupuesto

Analizando algunos proveedores de equipos, tiendas online y observando los requerimientos necesarios, se obtiene el siguiente presupuesto para el sistema de control del autómatas por medio de impulsos neuronales alfa y beta con FPGA de hardware libre.

Tabla 4.11. Presupuesto

CANTIDAD	DETALLE	PRECIO	TOTAL
1	Mindwave de la empresa de NeuroSky	\$ 160,00	\$ 160,00
1	Mojo V3	\$ 100,00	\$ 100,00
1	Hexbug Araña	\$ 30,00	\$ 30,00
1	Bluetooth HC-05	\$ 15,00	\$ 15,00
1	Estructura del FPGA	\$ 15,00	\$ 15,00
1	Kit de componentes electrónicos	\$ 20,00	\$ 20,00
Subtotal			\$ 340,00
I.V.A. (12%)			\$ 40,80
Total			\$ 380,80

Fuente: La investigadora

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- El análisis de la variedad de equipos que existen en el mercado y cada una de sus diferentes aplicaciones en diversos proyectos, sea en los campos de la medicina como la ingeniería, es de mucha importancia, ya que con los mismos se puede identificar los valores, la manipulación y la adquisición de ondas EEG, que son importantes para una posterior adecuación y mejoramiento del proyecto.

- Se desarrolló un sistema inalámbrico para la interconexión y comunicación con los datos adquiridos por la diadema Mindwave y el procesador de datos Mojo V3, mediante el protocolo de bluetooth, la cual está diseñada para aplicaciones que requieren una comunicación sincronizada con el dispositivo de adquisición de señales neuronales de la empresa Neurosky.

- Para el correcto funcionamiento y vinculación de la diadema Mindwave y el módulo Bluetooth HC-05, la información de las señales neuronales adquiridas por el dispositivo deberán estar sincronizadas a una velocidad 9600 Baudios, la cual se almacena en un buffer por medio del protocolo UART, y son sincronizadas por medio de “flags”, que ayudan a conocer su estado de memoria, y disponibilidad.

- El módulo UART, contiene una línea transmisora Tx (convertidor paralelo-serie) y una línea receptora Rx (convertidor serie- paralelo) de forma independiente, es decir, que puede transmitir y recibir datos simultáneamente porque consta de una comunicación full-duplex, permitiendo la comunicación con el bluetooth y el procesador Mojo V3.

5.2. Recomendaciones

- Si el requerimiento de señales neuronales es mayor a dos (Alfa y beta), se recomienda el uso de otros dispositivos de adquisición de ondas cerebrales, que posea mayor número de electrodos, aunque el precio es más elevado como se muestra en la comparación de hardware.
- Se debe tener en cuenta el cuidado y mantenimiento de los dispositivos que componen el sistema, el electrodo ubicado en el brazo de la diadema mindwave debe siempre estar limpio para impedir su desgaste, al momento de utilizarlo se debe tomar las precauciones de la colocación sobre todo de la pinza en la oreja para realizar tierra y optimizar los datos adquiridos.
- Antes de la utilización de cualquier componente para la comunicación entre el dispositivo de adquisición de frecuencias cerebrales y la tarjeta de procesamiento de datos, se debe realizar el análisis de compatibilidad.
- Al realizar aplicaciones donde se vinculen ondas neuronales, es necesario el conocimiento de la velocidad de transmisión, el protocolo a utilizarse y acoplar las señales a un patrón general, para la correcta captura de la trama enviada por la diadema Mindwave.

BIBLIOGRAFÍA Y REFERENCIAS

- [1] E. Comercio, «El futuro de la robótica: anticipar intenciones humanas y procesar órdenes cerebrales,» 2014. [En línea]. Available: <http://www.elcomercio.com/guaifai/futuro-robotica-anticipar-intenciones-humanas.html>.
- [2] L. M. Ariza, «Pensé que podría retomar el viejo sueño de la humanidad: Controlar cosas con el pensamiento,» *EL PAÍS*, p. 25, 6 Febrero 2012.
- [3] J. Varela, Sistema de Control Automático para el Posicionamiento de una silla de ruedas electrica, Repositorio Universidad Técnica de Ambato, Ambato, 2014 .
- [4] A. Yaguana, Desarrollo e Implementacion de una interfaz de comunicación que permita la interacción entre el usuario y las señales emitidas por sus ondas cerebrales usando un dispositivo de EGG de NeuroSky para controlar periféricos electrónicos, Repositorio Escuela Politécnica del Litoral Guayaquil, 2014.
- [5] E. Maila, Prototipo de silla de ruedas controlada mediante señales eléctricas producidas por el cerebro, Repositorio de la Escuela Politécnica Nacional, Quito, 2015.
- [6] Tanco F., Verrastro C., Grinberg D., Roitman J., «Implementación de redes neuronales artificiales en hardware para aplicación de detección automática de fulguraciones solares,» [En línea]. Available: <http://www.secyt.frba.utn.edu.ar/gia/redesneuronales.pdf>.
- [7] E. Rodriguez, «Sillas de ruedas controladas por la mente,» 30 junio 2009. [En línea]. Available: <http://www.fierasdelaingenieria.com/sillas-de-ruedas-controladas-por-la-mente/>.
- [8] El Comercio, «Red Silence, el robot marroquí que se maneja con la vista,» Edición Quito, 26 Mayo 2014.
- [9] J. Elias, «Controla tu PC con la mente con el OCZ NIA,» 21 Julio 2008. [En línea]. Available: http://www.eliax.com/index.cfm?post_id=5077.
- [10] Definicion ABC, «Definicion de automata,» 2016. [En línea]. Available: <http://www.definicionabc.com/general/automata.php>.
- [11] «Morfología del robot,» UNiversidad de Atacama, [En línea]. Available: <http://www.industriaynegocios.cl/academicos/alexanderborger/docts%20docencia/semi>

nario%20de%20aut/trabajos/2004/rob%C3%B3tica/seminario%202004%20robotica/seminario_robotica/documentos/morfolog%C3%8Da%20del%20robot.htm.

- [12] N. Manzanilla, «Robotica,» Marzo 2014. [En línea]. Available: <http://es.slideshare.net/neydamanzanilla/la-robotica-33060172>.
- [13] D. Ernaque, «Robotica,» 13 Julio 2014. [En línea]. Available: <https://roboticaela.wordpress.com/2014/07/13/robots-automatas-y-simple-maquinas/>.
- [14] V. Vera, «Lenguaje y programacion de robots,» 10 Agosto 2010. [En línea]. Available: <http://lenguaje-programacion-robots.blogspot.com/>.
- [15] D. Murcia, «Robotica,» [En línea]. Available: http://diegomurcia-robotica-1101.blogspot.com/p/clasificacion-de-los-robots_02.html.
- [16] J. Barrio, Ciencias para el mundo contemporáneo, Editex, España, Agosto 2013.
- [17] T. Kotsos, «Ondas Cerebrales,» 8 Julio 2008. [En línea]. Available: http://www.bibliotecapleyades.net/ciencia/ciencia_brain69.htm..
- [18] A. Urzelai, Manual basico de logistica integral, Buenos Aires: Ediciones Diaz de Santos, 2016.
- [19] Staff Editorial de Electrónica y Servicio, Electrónica y Servicio: Autoetéreos con bluetooth y USB,2014.
- [20] J. Fish, «How Bluetooth Works,» 10 Agosto 2015. [En línea]. Available: <https://www.thetrackr.com/blog/technology/how-bluetooth-works-a-dive-into-the-emerging-technology>.
- [21] «CCM,» Julio 2006. [En línea]. Available: <http://es.ccm.net/contents/69-como-funciona-bluetooth>.
- [22] J. G. Rodríguez, «upcommons,» 11 Febrero 2011. [En línea]. Available: <http://upcommons.upc.edu/bitstream/handle/2099.1/9257/Mem%C3%B2ria.pdf>.
- [23] J. Caballar, Wi-Fi : lo que se necesita conocer, España: Rc Libros, 2013.
- [24] S. Martinez, «¿Se pueden robar los datos transmitidos por wi-fi?,» 24 Noviembre 2013. [En línea]. Available: <http://equipo8wifi.blogspot.com/>.
- [25] X. Muñoz, Manual de Derecho de las Telecomunicaciones, Madrid: Legal Link, 2006.
- [26] S. R. Caprile, «EQUISBÍ,» de *Desarrollo de aplicaciones con comunicación remota basadas en módulos Zigbee y 802.15.4*, Buenos Aires, Editores-GAE, 2012, p. 135.

- [27] Mecatrónica UASLP, «Tutorial Xbee parte 1: ¿Qué es un Xbee y qué es necesario?,» 4 Julio 2013. [En línea]. Available: <https://mecatronicauaslp.wordpress.com/2013/07/04/xbee-parte-1-que-es-un-xbee-y-que-es-necesario/>.
- [28] S. Caprile, Equisbí: Desarrollo de aplicaciones con comunicación remota, Buenos Aires, 2009.
- [29] J. Rowan, Conceptos Basicos Sobre EEG, New York: Elsevier, 2014.
- [30] R. Barea, Universidad Alcala, [En línea]. Available: <http://www.bioingenieria.edu.ar/academica/catedras/bioingenieria2/archivos/apuntes/tema%205%20-%20electroencefalografia.pdf>.
- [31] Omicrona, «¿Cómo se leen las ondas cerebrales?,» 30 Octubre 2012. [En línea]. Available: <http://www.omicrono.com/2012/10/como-se-leen-las-ondas-cerebrales/>.
- [32] A. Moran, Diseño de interfaces cerebro-maquina controlados mediante registros de EGG, Madrid, 2015.
- [33] Alvy, «Unobrain: un gimnasio cerebral con un casco que «lee la mente»,» 22 Noviembre 2012. [En línea]. Available: <http://www.microservos.com/archivo/gadgets/unobrain-gimnasio-mental.html>.
- [34] J. Lajara, LabVIEW: Entorno gráfico de programación, España: Marcrombo, 2011.
- [35] J. Molina, Programación Gráfica para Ingenieros, Barcelona: Marcombo, 2010.
- [36] Universidad del pais Vasco, «Sistemas de adquisicion de datos,» [En línea]. Available: [http://www.sc.ehu.es/acwamurc/Transparencias/\(4\)TAD.pdf](http://www.sc.ehu.es/acwamurc/Transparencias/(4)TAD.pdf).
- [37] Arduino, «¿Que es un arduino?,» [En línea]. Available: <http://arduino.cl/que-es-arduino/>.
- [38] J. Torres, «Hardware para novatos (VII): Arduino ¿qué es y cómo funciona?,» 17 Marzo 2014. [En línea]. Available: <https://hipertextual.com/archivo/2014/03/hardware-novatos-arduino/>.
- [39] Computer. [En línea]. Available: <http://computerhoy.com/noticias/hardware/que-es-raspberry-pi-donde-comprarla-como-usarla-8614>.
- [40] A. Castro, «¿Qué es Raspberry Pi, dónde comprarla y cómo usarla?,» 23 Enero 2014. [En línea]. Available: <http://computerhoy.com/noticias/hardware/que-es-raspberry-pi-donde-comprarla-como-usarla-8614>.

- [41] Electrouni, «Electro-Digital,» 2015. [En línea]. Available: <http://electrounidigital.blogspot.com/2015/07/principales-caracteristicas-de-una-fpga.html>.
- [42] J. Reyes, «Dispositivos Logicos Programables,» 27 Enero 2009. [En línea]. Available: <http://iindustrial.obolog.es/dispositivos-logicos-programables-parte-2-210378>.
- [43] National Instruments. [En línea]. Available: <http://www.ni.com/white-paper/6984/es/>.
- [44] A. Leyton, «Clasificaciones del Hardware Libre,» 15 Diciembre 2015. [En línea]. Available: <https://plus.google.com/wm/trollface-meme-troll-gif-pics-lol-funny/106922021380369095667/posts/BH9pZVAzFKC>.
- [45] J. Wakerly, Diseño digital: principios y prácticas, Mexico : Pearson educacion de Mexico , 2011.
- [46] A. Grediaga, Diseño de procesadores con VHDL, Universidad de Alicante: Textos Docentes.
- [47] C. Rivera, «Verilog,» 7 Diciembre 2012. [En línea]. Available: <http://carloseljunior.blogspot.com/>.
- [48] P. Ruiz, «Introduccion a HDL Verilog,» Departamento de Tecnología Electrónica, Noviembre 2012. [En línea]. Available: <https://www.dte.us.es/Members/paulino/Verilog-Intro.pdf>.
- [49] L. Silva, «Universidad Tecnica Federico Santa Maria,» 30 Diciembre 2012. [En línea]. Available: http://www2.elo.utfsm.cl/~lsb/elo211/aplicaciones/Abel/tut_abel.pdf.
- [50] J. Rajewski, «Embedded Micro,» Agosto 2015. [En línea]. Available: <https://embeddedmicro.com>.
- [51] E. Larsen, «Classification of EEG Signals in a Brain-Computer Interface System. MSc in,» Norwegian University of Science and Technology, 2011. [En línea]. Available: <http://www.diva-portal.org/smash/get/diva2:440513/FULLTEXT01.pdf>, 72 pp.
- [52] G. A.R., «Implementación de métodos de procesamiento de señales EEG para aplicaciones de comunicación y control,» de *ECI Perú*, 2013.
- [53] Z. L. Marsan CA, «EEG HACKER,» Indian J Psychiatry., 14 Octubre 2014. [En línea]. Available: <http://eeghacker.com/2014/10/detecting-alpha-waves-threshold.html>.

- [54] Itead, «Robotshop,» 2015. [En línea]. Available: http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth_hc05.pdf.
- [55] D.-K. Electronics, «Digi-Key Electronics,» 2016. [En línea]. Available: <http://www.digikey.com/es/product-highlight/x/xilinx/spartan-6-fpga>.
- [56] I. F. Labs, «Hexbug,» [En línea]. Available: <https://www.hexbug.com/faq>.
- [57] M. Cayre y S. S.-L. C. S. a. A. S. Jordane Malaterre, « «The common properties of neurogenesis in the adult brain: from invertebrates to vertebrates.»» de *Comparative Biochemistry and Physiology Part B: Biochemistry and Molecular Biology*, 2012, pp. 1-15.

ANEXOS

ANEXO A

Datasheet Mindwave-Neurosky

RB-Neu-01

NeuroSky Mindwave EEG Sensor



- Wireless and light weight
- TGAM1 module with TGAT1 ASIC
- 8-hour AAA battery life
- Supports Windows XP / Vista / 7
- Supports Mac OS X 10.5.8 and 10.6.x

The **NeuroSky Mindwave EEG Sensor** turns your computer into a private tutor. The headset takes decades of laboratory brainwave technology and puts it into a bundled software package. It safely measures brainwave signals and monitors the attention levels of students as they interact with math, memory and pattern recognition applications. Ten apps are included with experiences ranging from fun entertainment to serious education.

Measurements:

- Raw signal
- Neuroscience defined EEG power spectrum (Alpha, Beta, etc.)
- eSense meter for Attention
- eSense meter for Meditation
- eSense Blink Detection
- On-head detection

Specifications:

- Weighs 90g
- Sensor arm up: (h)225mm x (w)155mm x (d)92mm
- Sensor Arm down: (h)225mm x (w)155mm x (d)165mm
- 30mW rate power; 50mW max power
- 2.420 - 2.471GHz RF frequency
- 6dBm RF max power
- 250kbit/s RF data rate
- 10m RF range
- 5% packet loss of bytes via wireless
- UART Baudrate: 57,600 Baud
- 1mV pk-pk EEG maximum signal input range
- 3Hz – 100Hz hardware filter range
- 12 bits ADC resolution
- 512Hz sampling rate
- 1Hz eSense calculation rate

Included Software:

- Meditation Journal
- SpeedMath
- BlinkZone
- Schulte
- SpadeA
- Find Number
- MindHunter
- Man.Up
- MindtyAnt
- Jack's Adventure

ANEXO B

Datasheet Módulo Bluetooth HC-05

HC Serial Bluetooth Products

User Instructional Manual

1 Introduction

HC serial Bluetooth products consist of Bluetooth serial interface module and Bluetooth adapter, such as:

(1) Bluetooth serial interface module:

Industrial level: HC-03, HC-04(HC-04-M, HC-04-S)

Civil level: HC-05, HC-06(HC-06-M, HC-06-S)

HC-05-D, HC-06-D (with baseboard, for test and evaluation)

(2) Bluetooth adapter:

HC-M4

HC-M6

This document mainly introduces Bluetooth serial module. Bluetooth serial module is used for converting serial port to Bluetooth. These modules have two modes: master and slaver device. The device named after even number is defined to be master or slaver when out of factory and can't be changed to the other mode. But for the device named after odd number, users can set the work mode (master or slaver) of the device by AT commands.

HC-04 specifically includes:

Master device: HC-04-M, M=master

Slave device: HC-04-S, S=slaver

The default situation of HC-04 is slave mode. If you need master mode, please state it clearly or place an order for HC-04-M directly. The naming rule of HC-06 is same.

When HC-03 and HC-05 are out of factory, one part of parameters are set for activating the device. The work mode is not set, since user can set the mode of HC-03, HC-05 as they want.

The main function of Bluetooth serial module is replacing the serial port line, such as:

1. There are two MCUs want to communicate with each other. One connects to Bluetooth master device while the other one connects to slave device. Their connection can be built once the pair is made. This Bluetooth connection is equivalently liked to a serial port line connection including RXD, TXD

signals. And they can use the Bluetooth serial module to communicate with each other.

2. When MCU has Bluetooth slave module, it can communicate with Bluetooth adapter of computers and smart phones. Then there is a virtual communicable serial port line between MCU and computer or smart phone.

3. The Bluetooth devices in the market mostly are slave devices, such as Bluetooth printer, Bluetooth GPS. So, we can use master module to make pair and communicate with them.

Bluetooth Serial module's operation doesn't need drive, and can communicate with the other Bluetooth device who has the serial. But communication between two Bluetooth modules requires at least two conditions:

- (1) The communication must be between master and slave.
- (2) The password must be correct.

However, the two conditions are not sufficient conditions. There are also some other conditions basing on different device model. Detailed information is provided in the following chapters.

In the following chapters, we will repeatedly refer to Linvor's (Formerly known as Guangzhou HC Information Technology Co., Ltd.) material and photos.

2 Selection of the Module

The Bluetooth serial module named even number is compatible with each other; The slave module is also compatible with each other. In other word, the function of HC-04 and HC-06, HC-03 and HC-05 are mutually compatible with each other. HC-04 and HC-06 are former version that user can't reset the work mode (master or slave). And only a few AT commands and functions can be used, like reset the name of Bluetooth (only the slaver), reset the password, reset the baud rate and check the version number. The command set of HC-03 and HC-05 are more flexible than HC-04 and HC-06's. Generally, the Bluetooth of HC-03/HC-05 is recommended for the user.

Here are the main factory parameters of HC-05 and HC-06. Pay attention to the differences:

HC-05	HC-06
Master and slave mode can be switched	Master and slave mode can't be switched
Bluetooth name: HC-05	Bluetooth name: linvor
Password: 1234	Password: 1234

<p>Master role: have no function to remember the last paired slave device. It can be made paired to any slave device. In other words, just set AT+CMODE=1 when out of factory. If you want HC-05 to remember the last paired slave device address like HC-06, you can set AT+CMODE=0 after paired with the other device. Please refer the command set of HC-05 for the details.</p>	<p>Master role: have paired memory to remember last slave device and only make pair with that device unless KEY (PIN26) is triggered by high level. The default connected PIN26 is low level.</p>
<p>Pairing: The master device can not only make pair with the specified Bluetooth address, like cell-phone, computer adapter, slave device, but also can search and make pair with the slave device automatically.</p> <p>Typical method: On some specific conditions, master device and slave device can make pair with each other automatically. (This is the default method.)</p>	<p>Pairing: Master device search and make pair with the slave device automatically.</p> <p>Typical method: On some specific conditions, master and slave device can make pair with each other automatically.</p>
<p>Multi-device communication: There is only point to point communication for modules, but the adapter can communicate with multi-modules.</p>	<p>Multi-device communication: There is only point to point communication for modules, but the adapter can communicate with multi-modules.</p>
<p>AT Mode 1: After power on, it can enter the AT mode by triggering PIN34 with high level. Then the baud rate for setting AT command is equal to the baud rate in communication, for example: 9600.</p> <p>AT mode 2: First set the PIN34 as high level, or while on powering the module set the PIN34 to be high level, the Baud rate used here is 38400 bps.</p> <p>Notice: All AT commands can be operated only</p>	<p>AT Mode: Before paired, it is at the AT mode. After paired it's at transparent communication.</p>

<p>when the PIN34 is at high level. Only part of the AT commands can be used if PIN34 doesn't keep the high level after entering to the AT mode. Through this kind of designing, set permissions for the module is left to the user's external control circuit, that makes the application of HC-05 is very flexible.</p>	
<p>During the process of communication, the module can enter to AT mode by setting PIN34 to be high level. By releasing PIN34, the module can go back to communication mode in which user can inquire some information dynamically. For example, to inquire the pairing is finished or not.</p>	<p>During the communication mode, the module can't enter to the AT mode.</p>
<p>Default communication baud rate: 9600, 4800-1.3M are settable.</p>	<p>Default communication baud rate: 9600, 1200-1.3M are settable.</p>
<p>KEY: PIN34, for entering to the AT mode.</p>	<p>KEY: PIN26, for master abandons memory.</p>
<p>LED1: PIN31, indicator of Bluetooth mode. Slow flicker (1Hz) represents entering to the AT mode², while fast flicker(2Hz) represents entering to the AT mode¹ or during the communication pairing. Double flicker per second represents pairing is finished, the module is communicable.</p> <p>LED2: PIN32, before pairing is at low level, after the pairing is at high level.</p> <p>The using method of master and slaver's indicator is the same.</p> <p>Notice: The PIN of LED1 and LED2 are connected with LED+.</p>	<p>LED: The flicker frequency of slave device is 102ms. If master device already has the memory of slave device, the flicker frequency during the pairing is 110ms/s. If not, or master has emptied the memory, then the flicker frequency is 750m/s. After pairing, no matter it's a master or slave device, the LED PIN is at high level.</p> <p>Notice: The LED PIN connects to LED+ PIN.</p>
<p>Consumption: During the pairing, the current is</p>	<p>Consumption: During the pairing, the current is</p>

fluctuant in the range of 30-40mA. The mean current is about 25mA. After paring, no matter processing communication or not, the current is 8mA. There is no sleep mode. This parameter is same for all the Bluetooth modules.	fluctuant in the range of 30-40 m. The mean current is about 25mA. After paring, no matter processing communication or not, the current is 8mA. There is no sleep mode. This parameter is same for all the Bluetooth modules.
Reset: PIN11, active if it's input low level. It can be suspended in using.	Reset: PIN11, active if it's input low level. It can be suspended in using.
Level: Civil	Level: Civil

The table above that includes main parameters of two serial modules is a reference for user selection.

HC-03/HC-05 serial product is recommended.

3. Information of Package

The PIN definitions of HC-03, HC-04, HC-05 and HC-06 are kind of different, but the package size is the same: 28mm * 15mm * 2.35mm.

The following figure 1 is a picture of HC-06 and its main PINs. Figure 2 is a picture of HC-05 and its main PINs. Figure 3 is a comparative picture with one coin. Figure 4 is their package size information. When user designs the circuit, you can visit the website of Guangzhou HC Information Technology Co., Ltd. (www.wavesen.com) to download the package library of protle version.



Figure 1 HC-06

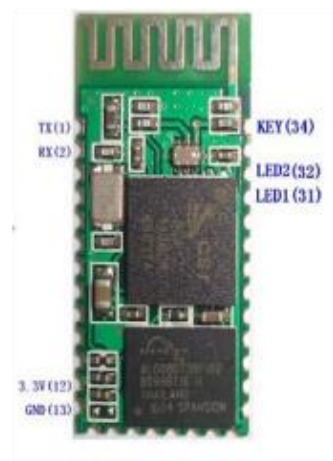


Figure 2 HC-05



Figure 3 Comparative picture with one coin

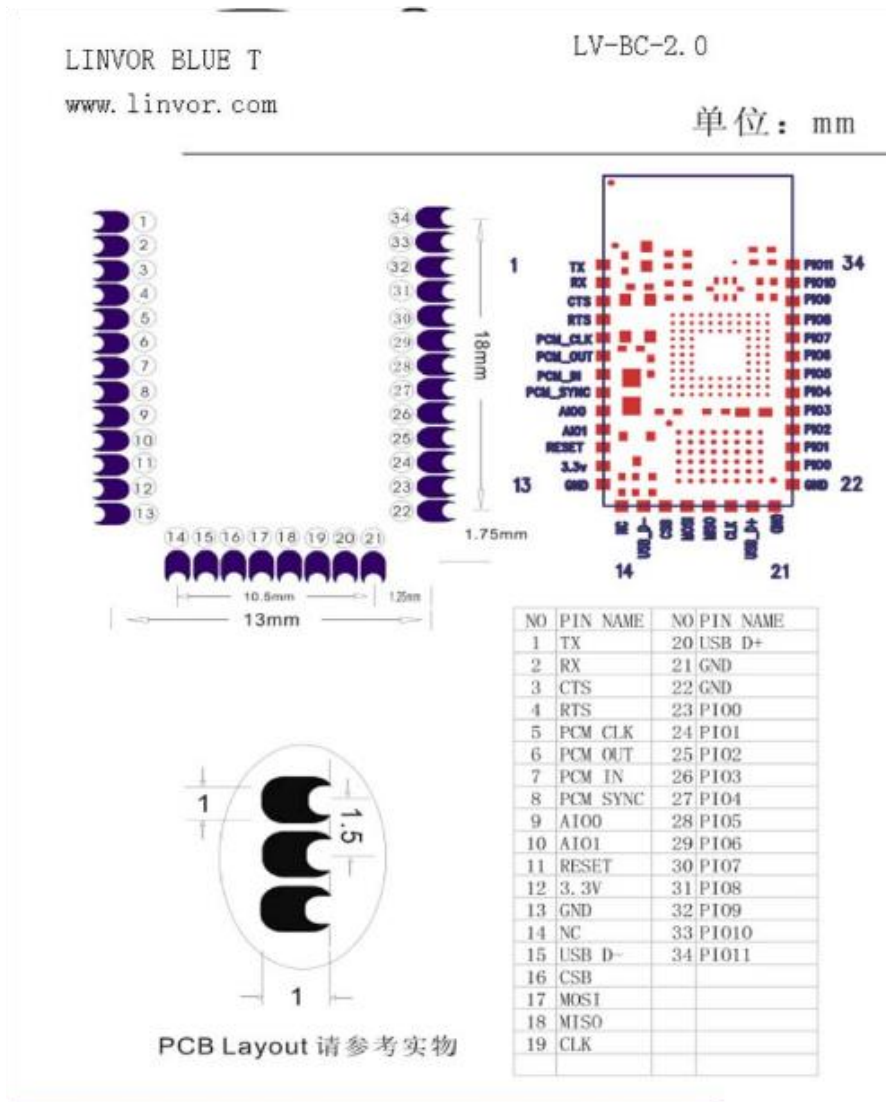


Figure 4 Package size information

ANEXO C

Datasheet Hexbug Spider



HEXBUG® Spider Micro Robotic Creatures

The HEXBUG® Spider is a robotic creature that is an electro mechanical marvel to watch and operate. Each HEXBUG Spider features 360 degree steering, an LED forward eye, and two-channel, user selectable infrared remote control. The two-channel remote control enables the user to operate multiple bugs independently or at the same time. The largest member of the HEXBUG family, the complexity of its internal mechanisms and six-legged crawling motion is simply fascinating to operate and watch. HEXBUG Spider is CPSIA approved for kids ages 8 and up.

How does it work?

The six-legged HEXBUG Spider features 360 degree steering and an LED forward eye, allowing the user to maneuver it around objects and control precisely where it scurries. Three button cell batteries are included in each HEXBUG Spider and two with each remote.

Fits in the palm of your hand for miniature robotic fun at anytime!

- Product Length: 4.5 inches
- Product Width: 3.5 inches
- Product Height: 3 inches
- Product Weight: 0.168 pounds

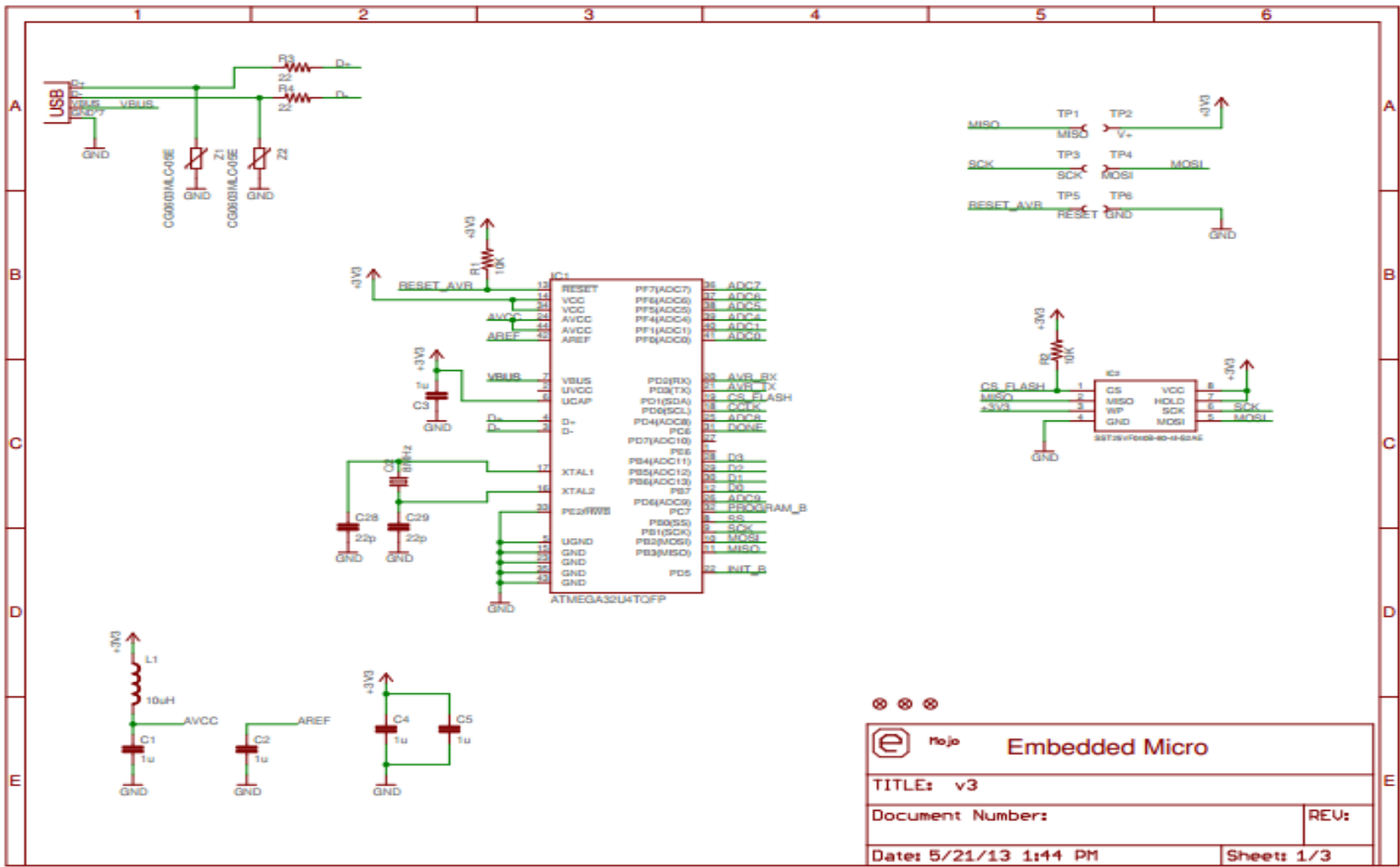
Available in various translucent colors including Red, Green, Blue, Orange and Teal

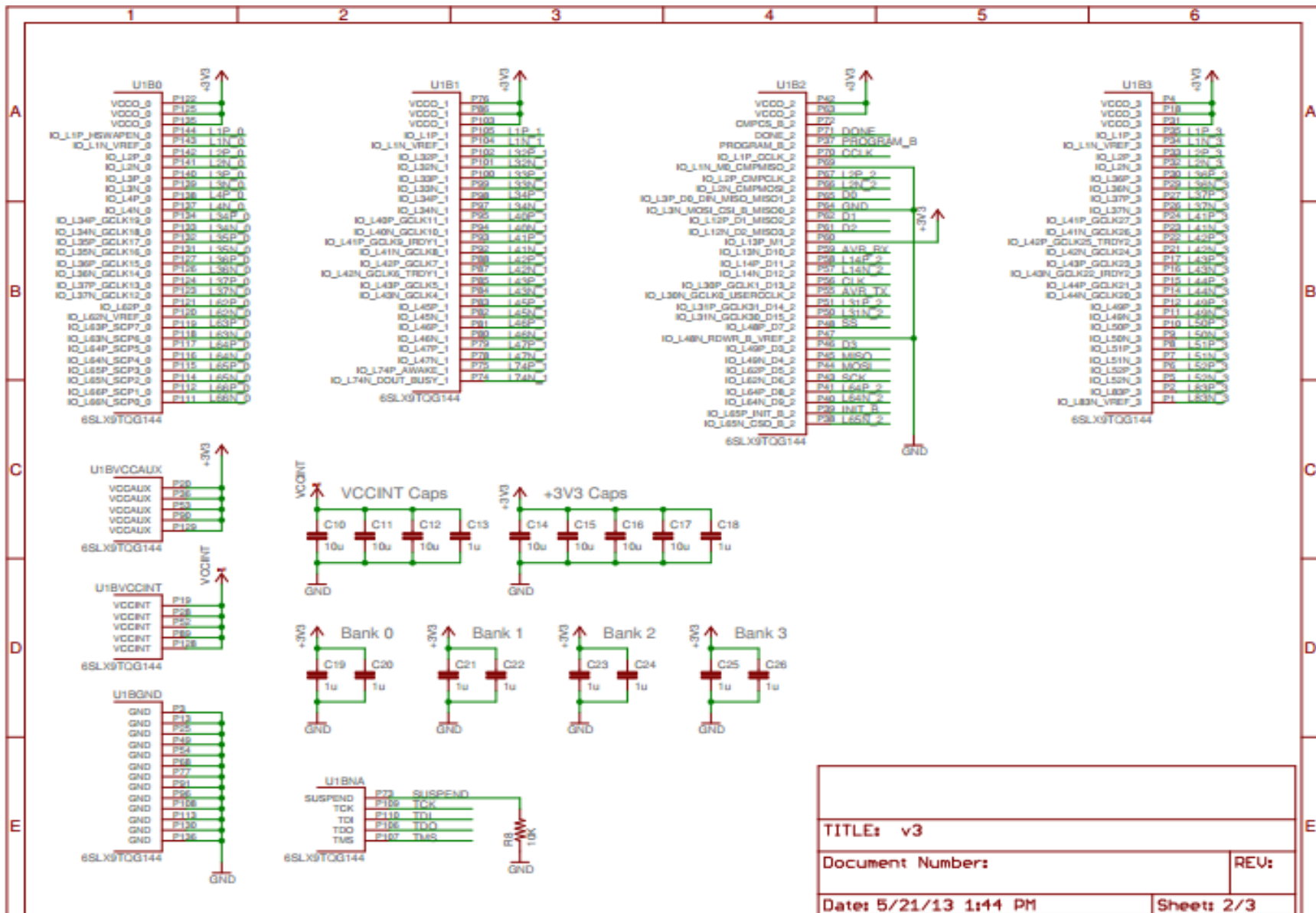
- Pricing varies by country
- Available at www.hexbug.com

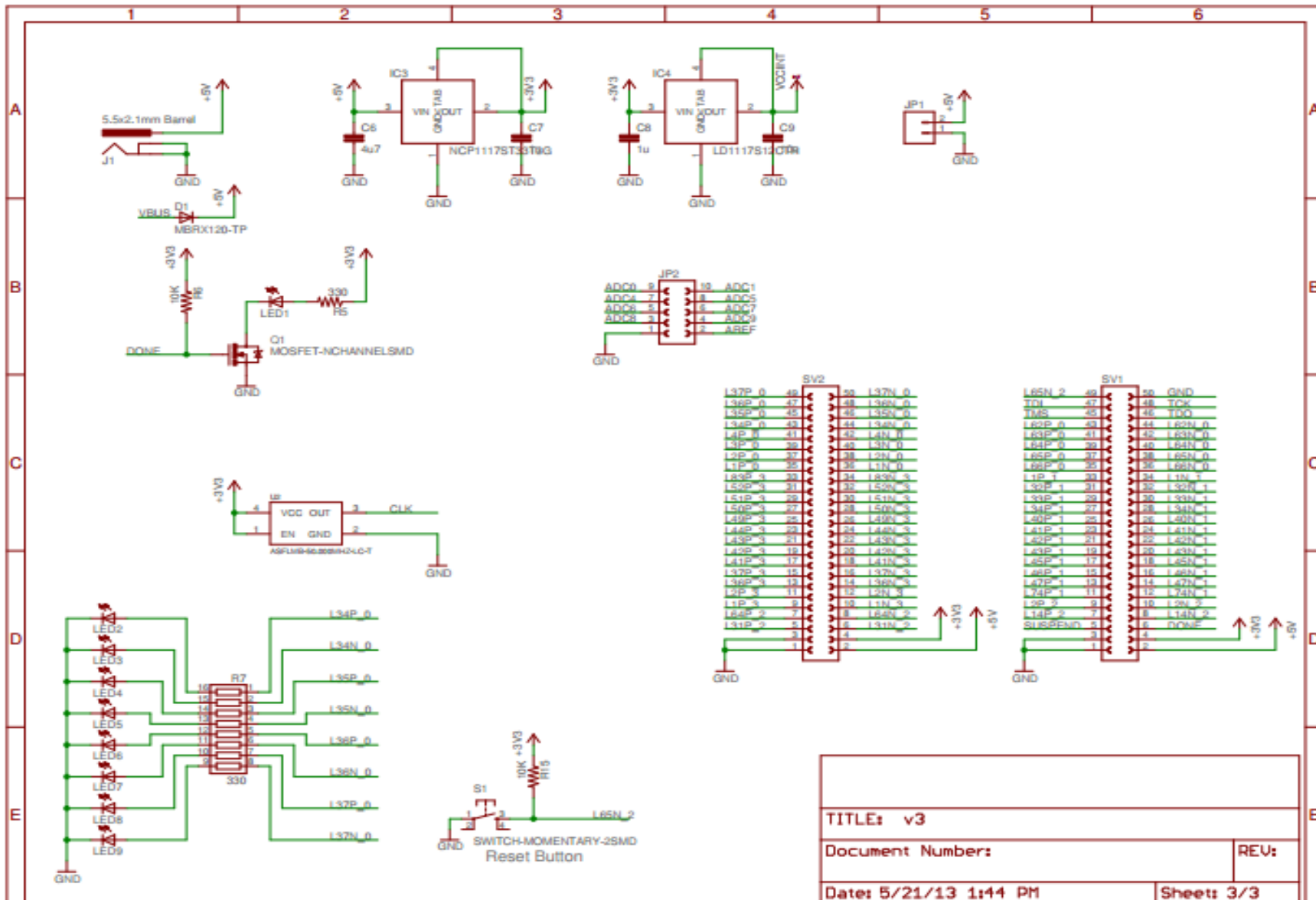


ANEXO D

Esquemático de la tarjeta de adquisición Mojo V3







TITLE: v3	
Document Number:	REV:
Date: 5/21/13 1:44 PM	Sheet: 3/3

ANEXO E

Programación en Mojo IDE; Source; mojo_top.luc

```
module mojo_top
(
  input clk,           // Reloj de 50Mhz
  input rst_n,        // Boton de reseteo(mojo.ufc) logica inversa(activa en bajo)
  output led [8],     // Array de 8 bits al que estan conectados los LEDS(mojo.ufc)
  input cclk,         // Reloj de configuracion, Cuando se pone en ALTO el avr ha finalizado
  // listo.
  output spi_miso,    // AVR SPI MISO
  input spi_ss,       // AVR SPI Slave Select
  input spi_mosi,     // AVR SPI MOSI
  input spi_sck,      // AVR SPI Clock
  output spi_channel[4], // pines de proposito general del AVR (Se usa para seleccionar el
  // canal ADC generalmente)
  input avr_tx,       // AVR TX Se conecta al pin de recepcion de la FPGA (mojo.ufc)
  output avr_rx,      // AVR RX Se conecta al pin de transmision de la FPGA (mojo.ufc)
  input avr_rx_busy,  // AVR Bandera que indica que el buffer de comunicacion esta lleno
  //*****control hc05*****
  input bt_tx,        // BT TX se conecta directamente al pin de transmision del
  // modulo(def_bt.ufc)
  output bt_rx,       // BT RX se conecta directamente al pin de recepcion del
  // modulo(def_bt.ufc)
  input bt_rx_busy,   // BT Bandera que indica que el buffer de comunicacion esta lleno
  //*****pines de salida para control*****
  output control [8] // Array de salidas para el control del HEXAPODO
)
sig rst;             // variable tipo signal para reseteo

// Modulo de conexion para las entradas CLK y RST de todos los m3dulos contenidos en
// las llaves.
// Esto es conveniente ya que la mayoria de los m3dulos requerieren un reloj y
// restablecer la se±al.
.clk(clk)
{
  reset_conditioner reset_cond; //condicionador utilizado para sincronizacion de la fpga y el
  // reseteo de la misma.
  .rst(rst)
  {
    // El modulo avr_interface es usado para hablar con el AVR (acceso al USB y ADC del
    // avr)
    avr_interface avr; // modulo usado para comunicarse con el avr
    bt_interface bt; // modulo usado para comunicarse con el HC05
    dff data[8]; // registro de flip-flops tipo D de 8 bits de salida para almacenar caracteres
  }
}

// Es donde se puede realizar el c±culo y leer las se±ales de lectura / escritura.
// El bloque always recibe su nombre debido a que siempre est± ocurriendo.
// Cuando las herramientas ven un bloque always, generan un circuito digital
// que permite replicar el comportamiento descrito por el bloque.
always //equivalente al while(true) todo se ejecuta en paralelo
{
```



```

    reset_cond.in = ~rst_n; // Señal de entrada para reinicio (Logica Inversa por tanto
    entrada invertida)
    rst = reset_cond.out; // Reseteo condicionado

//*****conectar entradas*****

    ///////////////////////////////////BLUETOOTH////////////////////////////////////
    bt.cclk = cclk; // conecta reloj de configuracion general con el que se necesita en el
    modulo de la comunicacion BT.
    bt.rx = bt_tx; // Conecta el pin de la FPGA al que esta conectado dispositivo
    bluetooth con la interfaz del HARDWARE
    // DEFINIDO en el modulo bt_interface.
    ///////////////////////////////////AVR////////////////////////////////////
    avr.cclk = cclk; // conecta reloj de configuracion general con el que se necesita en el
    modulo del AVR.
    avr.spi_ss = spi_ss; // avr.spi_ss = bz; // conecta pines fisicos con los del
    HARDWARE DEFINIDO en avr_interface
    avr.spi_mosi = spi_mosi; // avr.spi_mosi = bz; // NO SE UTILIZA EN LA APLICACION
    DEL PRESENTE
    avr.spi_sck = spi_sck; // avr.spi_sck = bz; // pueden dejarse definidos o setear bz para
    desactivar el mismo
    avr.rx = avr_tx; // conectar pin fisico de TX del AVR con el RX del HARDWARE
    DEFINIDO en avr_interface
    ///////////////////////////////////

//*****conectar salidas*****

    ///////////////////////////////////AVR////////////////////////////////////
    spi_miso = avr.spi_miso; // conectar pin fisico MISO de comunicacion SPI con el del
    HARDWARE DEFINIDO
    spi_channel = bzzzz; // spi_channel = avr.spi_channel; puede conectarse el pin fisico
    o desactivarse al no usarse
    avr_rx = avr.tx; // conectar pin fisico de RX del AVR con el TX del HARDWARE
    DEFINIDO en avr_interface
    avr.channel = hf; // desactivar ADC
    avr.tx_block = avr_rx_busy; // Se bloquea la transmision cuando el AVR se encuentre
    ocupado
    ///////////////////////////////////BLUETOOTH////////////////////////////////////
    bt_rx = bt_tx; // conectar pin fisico de RX del bluetooth con el TX del HARDWARE
    DEFINIDO en bt_interface
    bt.tx_block = bt_rx_busy; // bloquea cuando el modulo este recibiendo
    ///////////////////////////////////

//*****conectar tx con rx*****

    //genero un eco (es decir conectar la TX del AVR con el RX del mismo para obtener de
    retorno el dato que se envie)
    avr.tx_data = avr.rx_data;
    avr.new_tx_data = avr.new_rx_data;

    ///////////////////////////////////bluetooth////////////////////////////////////

    //bt.new_tx_data = avr.new_rx_data; // si se detecta un dato en el Serial del AVR enviar
    por bluetooth(habilitacion TX)
    //bt.tx_data = avr.rx_data; // dato del buffer SerialAVR enviarlo por el buffer del
    bluetooth
    //bt.tx_block = bt.tx_busy; // bloquear la transmision mientras se envíen todos los
    datos(evita colapsos)Importante

```

```

bt.new_tx_data = bz;           // desactivado (no envie)
bt.tx_data = bz;              // desactivado (no envie)
bt.tx_block = bz;            // desactivado (no envie)

////////////////////////////////////

avr.new_tx_data = bt.new_rx_data; // si se detecta un dato en el Serial del Bluetooth
enviar para el AVR(habilitacion TX)
avr.tx_data = bt.rx_data;      // dato del buffer del bluetooth enviarlo por el buffer del
SerialAVR
avr.tx_block = avr.tx_busy;    // bloquear la transmision mientras se envíen todos los
datos(evita colapsos)Importante
//avr.new_tx_data = bz;        // si se detecta un dato en el Serial del Bluetooth enviar para
el AVR(habilitacion TX)
//avr.tx_data = bz;           // dato del buffer del bluetooth enviarlo por el buffer del
SerialAVR
//avr.tx_block = bz;          // bloquear la transmision mientras se envíen todos los
datos(evita colapsos)Importante

////////////////////////////////////

if (bt.new_rx_data) // si se recibe un nuevo dato por bluetooth
{
//data.d = bt.rx_data; // escribir el dato en los FF data (o utilizar ese dato donde sea
necesario)
}

if (avr.new_rx_data) // si se recibe un nuevo dato que envia el avr
{
data.d = avr.rx_data; // escribir el dato en los FF data
}

led = data.q; // coneccion entre array de leds con las salidas de los FFs
control = data.q; // coneccion entre array para contro con las salidas de los FFs
}

```

ANEXO F

Programación en Mojo IDE; Components; bt_interface.luc

```
module bt_interface #(
    CLK_FREQ = 50000000 : CLK_FREQ > 0,          // definir reloj de 50Mhz
    BAUD = 57600 : BAUD > 0 && BAUD < CLK_FREQ/4 // velocidad de transmision de
datos para el dispositivo bluetooth
)
(
    // definiciones de interfaces de entrada y salida
    input clk,
    input rst,

    input cclk,

    output tx,
    input rx,

    output new_tx,          // nueva bandera para TX

    // Interfaz de TX Serial
    input tx_data[8],      // registro para datos para enviar
    input new_tx_data,     // Bandera de deteccion de dato nuevo (1 = dato nuevo)
    output tx_busy,        // Bandera para deteccion de transmisor ocupado (1 = ocupado)
    input tx_block,        // Bandera para el bloqueo del transmisor (1 = bloqueo) conectar al
bt_rx_busy

    // Interfaz de RX Serial
    output rx_data[8],     // registro para datos recibidos
    output new_rx_data     // Bandera de deteccion de nuevo dato recibido (1 = dato nuevo)
)

sig n_rdy;

.clk(clk)
{
    cclk_detector cclk_detector(#CLK_FREQ(CLK_FREQ), .rst(rst));
    .rst(n_rdy)
    {
        bt_rx blu_rx(#CLK_FREQ(CLK_FREQ), #BAUD(BAUD)); //defino variable BLU con
parametros de bt_rx
        bt_tx blu_tx(#CLK_FREQ(CLK_FREQ), #BAUD(BAUD)); //defino variable BLU con
parametros de bt_tx
    }
    dff block[4]; //registro de flip flop tipo D para almacenar situacion de bloqueo
    dff busy;     //flip flop tipo D para almacenar bit de ocupado
}

always
{
    //conexiones que se ejecutan en paralelo dependiendo directamente de mojo_top.luc
    new_tx=0;
    //

    n_rdy = ~cclk_detector.ready;

    //entradas
```

```

cclk_detector.cclk = cclk;
blu_rx.rx = rx;
blu_tx.data = tx_data;
blu_tx.new_data = new_tx_data;
blu_tx.block = busy.q;
block.d = c{block.q[2:0], tx_block};
//salidas
tx_busy = blu_tx.busy;
rx_data = blu_rx.data;
new_rx_data = blu_rx.new_data;

tx = cclk_detector.ready ? blu_tx.tx : bz;

if (block.q[3] ^ block.q[2]) //si no hay datos setea bandera de bloqueo a libre
    busy.d = 0;

if (!blu_tx.busy && new_tx_data) // si se activa la bandera de transmision o de nuevo dato
de entrada, setea bloqueo a ocupado
    busy.d = 1;
}

```

ANEXO G

Programación en Mojo IDE; Components; bt_rx.luc

```
module bt_rx #(
    CLK_FREQ = 50000000 : CLK_FREQ > 0,          // clock frequency
    BAUD = 57600 : BAUD > 0 && BAUD <= CLK_FREQ/4 // desired baud rate
)(
    input clk,      // clock input
    input rst,     // reset active high
    input rx,      // UART rx input
    output data[8], // received data
    output new_data // new data flag (1 = new data)
)

const CLK_PER_BIT = (CLK_FREQ + BAUD) / BAUD - 1; // clock cycles per bit
const CTR_SIZE = $clog2(CLK_PER_BIT); // bits required to store CLK_PER_BIT - 1

.clk(clk) {
    .rst(rst) {
        fsm state = {IDLE, WAIT_HALF, WAIT_FULL, WAIT_HIGH}; // FSM for receiver
    }
    dff ctr[CTR_SIZE]; // delay counter
    dff bitCtr[3]; // bit counter
    dff savedData[8]; // received data
    dff newData; // new data flag buffer
    dff rxd; // input buffer
}

always {
    rxd.d = rx; // buffer rx
    newData.d = 0; // default to 0

    // outputs
    data = savedData.q;
    new_data = newData.q;

    case (state.q)
    {
        state.IDLE:
            bitCtr.d = 0; // reset counter
            ctr.d = 0; // reset counter
            if (rxd.q == 0) // if rx line is low (start bit)
                state.d = state.WAIT_HALF; // switch state

        state.WAIT_HALF:
            ctr.d = ctr.q + 1; // increment the counter
            if (ctr.q == (CLK_PER_BIT >> 1)) { // if counter is the max value
                ctr.d = 0; // reset counter
                state.d = state.WAIT_FULL; // switch state
            }

        state.WAIT_FULL:
            ctr.d = ctr.q + 1; // increment counter
            if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
                savedData.d = c{rxd.q, savedData.q[7:1]}; // shift in new data
                bitCtr.d = bitCtr.q + 1; // increment bit counter
                ctr.d = 0; // reset counter
                if (bitCtr.q == 7) { // if we have received 8 bits

```

```
    state.d = state.WAIT_HIGH;           // switch state
    newData.d = 1;                       // signal new byte received
}
}

state.WAIT_HIGH:
    if (rx.d.q == 1)                     // wait for input to go high (idle)
        state.d = state.IDLE;           // switch state

default:
    state.d = state.IDLE;                // if in an invalid state, reset to idle
}
}
```

ANEXO H

Programación en Mojo IDE; Components; bt_tx.luc

```
module bt_tx #(
    CLK_FREQ = 50000000 : CLK_FREQ > 0,          // clock frequency
    BAUD = 57600 : BAUD > 0 && BAUD <= CLK_FREQ/2 // desired baud rate
)
(
    input clk,          // clock
    input rst,         // reset active high
    output tx,         // TX output
    input block,       // block transmissions
    output busy,       // module is busy when 1
    input data[8],     // data to send
    input new_data     // flag for new data
)

const CLK_PER_BIT = (CLK_FREQ + BAUD) / BAUD - 1; // clock cycles per bit
const CTR_SIZE = $clog2(CLK_PER_BIT); // bits required to store CLK_PER_BIT - 1

.clk(clk) {
.rst(rst) {
    fsm state = { IDLE, START_BIT, DATA, STOP_BIT }; // FSM for transmitter
}
dff ctr[CTR_SIZE]; // delay counter
dff bitCtr[3]; // bit counter
dff savedData[8]; // transmission data
dff txReg; // output buffer
dff blockFlag; // input buffer
}

always
{
    // Outputs
    tx = txReg.q;
    busy = 1; // default to 1

    blockFlag.d = block; // connect to buffer

    case (state.q) { // FSM
        state.IDLE:
            txReg.d = 1; // idle high (UART standard)
            if (!blockFlag.q) {
                busy = 0; // not busy
                bitCtr.d = 0; // reset counter
                ctr.d = 0; // reset counter
                if (new_data) { // request to send data?
                    savedData.d = data; // save the data
                    state.d = state.START_BIT; // switch states
                }
            }
        state.START_BIT:
            ctr.d = ctr.q + 1; // increment counter
            txReg.d = 0; // start bit is low
            if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
                ctr.d = 0; // reset counter
            }
    }
}
```

```

    state.d = state.DATA;    // switch states
}

state.DATA:
txReg.d = savedData.q[bitCtr.q]; // output the data bit
ctr.d = ctr.q + 1;             // increment counter
if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
    ctr.d = 0;                 // reset counter
    bitCtr.d = bitCtr.q + 1;   // increase bit counter
    if (bitCtr.q == 7)        // if we have sent all the bits
        state.d = state.STOP_BIT; // switch states
}

state.STOP_BIT:
txReg.d = 1;                 // stop bit is high
ctr.d = ctr.q + 1;          // increase counter
if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
    state.d = state.IDLE;     // switch states
}

default: state.d = state.IDLE; // if state is invalid reset to idle
}
}

```


ANEXO I

Programación en Mojo IDE; Components; avr_interface.luc

```
module avr_interface #(
    CLK_FREQ = 50000000 : CLK_FREQ > 0,          // clock frequency
    BAUD = 500000 : BAUD > 0 && BAUD < CLK_FREQ/4 // baud rate
)
    input clk,
    input rst,

    // cclk, or configuration clock is used when the FPGA is begin configured.
    // The AVR will hold cclk high when it has finished initializing.
    // It is important not to drive the lines connecting to the AVR
    // until cclk is high for a short period of time to avoid contention.
    input cclk,

    // AVR SPI Signals
    output spi_miso,      // connect to spi_miso
    input spi_mosi,       // connect to spi_mosi
    input spi_sck,        // connect to spi_sck
    input spi_ss,         // connect to spi_ss
    output spi_channel[4], // connect to spi_channel

    // AVR Serial Signals
    output tx,            // connect to avr_rx (note that tx->rx)
    input rx,            // connect to avr_tx (note that rx->tx)

    // ADC Interface Signals
    input channel[4],     // ADC channel to read from, use hF to disable ADC
    output new_sample,    // new ADC sample flag
    output sample[10],    // ADC sample data
    output sample_channel[4], // channel of the new sample

    // Serial TX User Interface
    input tx_data[8],     // data to send
    input new_tx_data,    // new data flag (1 = new data)
    output tx_busy,       // transmitter is busy flag (1 = busy)
    input tx_block,       // block the transmitter (1 = block) connect to avr_rx_busy

    // Serial Rx User Interface
    output rx_data[8],    // data received
    output new_rx_data    // new data flag (1 = new data)
)

sig n_rdy;

.clk(clk){
    cclk_detector cclk_detector(#CLK_FREQ(CLK_FREQ), .rst(rst));
    .rst(n_rdy) {
        spi_slave spi_slave(#CPOL(0), #CPHA(0));
        uart_rx uart_rx(#CLK_FREQ(CLK_FREQ), #BAUD(BAUD));
        uart_tx uart_tx(#CLK_FREQ(CLK_FREQ), #BAUD(BAUD));

        dff newSampleReg, sampleReg[10], sampleChannelReg[4];
        dff byteCt[2];
    }
    dff block[4];
    dff busy;
```

```

}

always {
    // AVR not ready flag
    n_rdy = ~cclk_detector.ready;

    //inputs
    cclk_detector.cclk = cclk;
    spi_slave.sck = spi_sck;
    spi_slave.mosi = spi_mosi;
    spi_slave.data_in = hff;
    spi_slave.ss = spi_ss;
    uart_rx.rx = rx;
    uart_tx.data = tx_data;
    uart_tx.new_data = new_tx_data;
    uart_tx.block = busy.q;
    block.d = c{block.q[2:0], tx_block};

    // outputs
    new_sample = newSampleReg.q;
    sample = sampleReg.q;
    sample_channel = sampleChannelReg.q;

    tx_busy = uart_tx.busy;
    rx_data = uart_rx.data;
    new_rx_data = uart_rx.new_data;

    spi_channel = cclk_detector.ready ? channel : bzzzz;
    spi_miso = cclk_detector.ready && !spi_ss ? spi_slave.miso : bz;
    tx = cclk_detector.ready ? uart_tx.tx : bz;

    newSampleReg.d = 0; // default to 0

    if (block.q[3] ^ block.q[2])
        busy.d = 0;

    if (!uart_tx.busy && new_tx_data)
        busy.d = 1;

    if (spi_ss) // device is not selected
        byteCt.d = 0; // reset byte count

    if (spi_slave.done) { // sent/received data from SPI
        if (byteCt.q == 0) {
            sampleReg.d[7:0] = spi_slave.data_out; // first byte is the 8 LSB of the sample
            byteCt.d = 1;
        } else if (byteCt.q == 1) {
            sampleReg.d[9:8] = spi_slave.data_out[1:0]; // second byte is the channel 2 MSB of the
sample
            sampleChannelReg.d = spi_slave.data_out[7:4]; // and the channel that was sampled
            byteCt.d = 2; // slave-select must be brought high before the next
transfer
            newSampleReg.d = 1; // signal we have new data
        }
    }
}
}
}

```

ANEXO J

Programación en Mojo IDE; Components; cclk_detector.luc

```
module cclk_detector #(
    CLK_FREQ = 50000000 : CLK_FREQ > 0
)(
    input clk,
    input rst,
    input cclk, // cclk input from AVR
    output ready // when 1, the AVR is ready
)

const CTR_SIZE = $clog2(CLK_FREQ/5000); // need to wait about 200uS

.clk(clk), .rst(rst) {
    dff ctr[CTR_SIZE];
}

// ready should only go high once cclk has been high for a while
// if cclk ever falls, ready should go low again
always {
    ready = &ctr.q; // max value

    if (cclk == 0) { // cclk low
        ctr.d = 0; // reset counter
    } else if (!&ctr.q) { // cclk high and ctr isn't maxed
        ctr.d = ctr.q + 1;
    }
}
```

ANEXO K

Programación en Mojo IDE; Components; reset_conditioner.luc

```
module reset_conditioner #(
    STAGES = 4 : STAGES > 1 // number of stages
)(
    input clk, // clock
    input in, // async reset
    output out // snyc reset
){

    dff stage[STAGES] (.clk(clk), .rst(in), #INIT(STAGESx{1}));

    always {
        stage.d = c{stage.q[STAGES-2:0],0};
        out = stage.q[STAGES-1];
    }
}
```

ANEXO L

Programación en Mojo IDE; Components; spi_slave.luc

```
module spi_slave #(
    // clock polarity, 0 = inactive low, 1 = inactive high
    CPOL = 0 : CPOL == 0 || CPOL == 1,

    // clock phase, 0 = valid on leading edge, 1 = valid on trailing edge
    CPHA = 0 : CPHA == 0 || CPHA == 1
)(
    input clk,        // clock
    input rst,        // reset
    input ss,         // SPI slave select
    input mosi,       // SPI MOSI
    output miso,      // SPI MISO
    input sck,        // SPI SCK
    output done,      // transfer done
    input data_in[8], // data to send
    output data_out[8] // data received
)

    .clk(clk) {
    .rst(rst) {
        dff bit_ct[3]; // bit counter
        dff data[8]; // received data
    }
    dff mosi_reg; // mosi buffer
    dff miso_reg; // miso buffer
    dff sck_reg[2]; // sck buffer
    dff ss_reg; // ss buffer
    dff data_out_reg[8]; // data_out buffer
    dff done_reg; // done buffer
}

always {
    // connect to buffer output
    miso = miso_reg.q;
    done = done_reg.q;
    data_out = data_out_reg.q;

    // read in buffered inputs
    ss_reg.d = ss;
    mosi_reg.d = mosi;
    sck_reg.d = c{sck_reg.q[0], sck}; // save old sck

    done_reg.d = 0; // default to not done

    if (ss_reg.q) { // not selected
        bit_ct.d = 3b111; // reset counter
        data.d = data_in; // copy in data for next byte
        miso_reg.d = data_in[7]; // write first bit out
    } else {
        // When CPOL and CPHA are different, we read on the falling edge.
        // When they are the same we read on the rising edge.
        // Therefore we can use XOR to check that and invert the
        // edge detector. If you XOR with 1, the bit is flipped.

        if (sck_reg.q == (b01 ^ 2x{CPOL[0]^CPHA[0]})) { // reading edge
```



```

input rst,    // reset active high
input rx,    // UART rx input
output data[8], // received data
output new_data // new data flag (1 = new data)
)

const CLK_PER_BIT = (CLK_FREQ + BAUD) / BAUD - 1; // clock cycles per bit
const CTR_SIZE = $clog2(CLK_PER_BIT); // bits required to store CLK_PER_BIT - 1

.clk(clk) {
.rst(rst) {
    fsm state = {IDLE, WAIT_HALF, WAIT_FULL, WAIT_HIGH}; // FSM for receiver
}
dff ctr[CTR_SIZE]; // delay counter
dff bitCtr[3]; // bit counter
dff savedData[8]; // received data
dff newData; // new data flag buffer
dff rxd; // input buffer
}

always {
rxd.d = rx; // buffer rx
newData.d = 0; // default to 0

// outputs
data = savedData.q;
new_data = newData.q;

/* When a new byte is being received, the input goes low. This is the
start bit. When the beginning of this bit is detected we need to wait
one and a half bit widths before reading in the first data bit. The
half cycle is so that we read in the middle of the bit for the most
accurate result. WAIT_HALF waits this half cycle and WAIT_FULL waits
a full cycle then reads in a bit. After 8 bits are read, the data is
output and the FSM waits for the RX input to go high again to signal
the end of the transmission. The FSM then returns to IDLE ready to
receive the next byte. */
case (state.q) {
state.IDLE:
    bitCtr.d = 0; // reset counter
    ctr.d = 0; // reset counter
    if (rxd.q == 0) // if rx line is low (start bit)
        state.d = state.WAIT_HALF; // switch state

state.WAIT_HALF:
    ctr.d = ctr.q + 1; // increment the counter
    if (ctr.q == (CLK_PER_BIT >> 1)) { // if counter is the max value
        ctr.d = 0; // reset counter
        state.d = state.WAIT_FULL; // switch state
    }

state.WAIT_FULL:
    ctr.d = ctr.q + 1; // increment counter
    if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
        savedData.d = c{rxd.q, savedData.q[7:1]}; // shift in new data
        bitCtr.d = bitCtr.q + 1; // increment bit counter
        ctr.d = 0; // reset counter
        if (bitCtr.q == 7) { // if we have received 8 bits

```

```
    state.d = state.WAIT_HIGH;           // switch state
    newData.d = 1;                       // signal new byte received
}
}

state.WAIT_HIGH:
    if (rx.d.q == 1)                     // wait for input to go high (idle)
        state.d = state.IDLE;           // switch state

default:
    state.d = state.IDLE;                 // if in an invalid state, reset to idle
}
}
```


ANEXO N

Programación en Mojo IDE; Components; uart_tx.luc

```
module uart_tx #(
    CLK_FREQ = 50000000 : CLK_FREQ > 0,          // clock frequency
    BAUD = 500000 : BAUD > 0 && BAUD <= CLK_FREQ/2 // desired baud rate
)(
    input clk,          // clock
    input rst,         // reset active high
    output tx,         // TX output
    input block,       // block transmissions
    output busy,       // module is busy when 1
    input data[8],     // data to send
    input new_data     // flag for new data
)

const CLK_PER_BIT = (CLK_FREQ + BAUD) / BAUD - 1; // clock cycles per bit
const CTR_SIZE = $clog2(CLK_PER_BIT); // bits required to store CLK_PER_BIT - 1

.clk(clk) {
    .rst(rst) {
        fsm state = { IDLE, START_BIT, DATA, STOP_BIT }; // FSM for transmitter
    }
    dff ctr[CTR_SIZE]; // delay counter
    dff bitCtr[3]; // bit counter
    dff savedData[8]; // transmission data
    dff txReg; // output buffer
    dff blockFlag; // input buffer
}

always {
    // Outputs
    tx = txReg.q;
    busy = 1; // default to 1
    blockFlag.d = block; // connect to buffer
    /* When a new byte is presented to send, that byte is saved so that if the input changes the
    correct data is still sent. We first have to send a 0, which is the start bit. Once the start bit is
    sent, each data bit is sent out. The counter ctr is used to delay between bits to get the correct
    baud rate. The counter bitCounter is used to keep track of what bit to send. After all eight bits
    are sent, we need to send 1, the stop bit. This bit ensures that the line goes high between
    transmissions. When the transmitter is sending out data, the output busy is set to 1. To
    prevent the transmitter from sending data, set block to 1. This can be used for flow control.
    */
    case (state.q) { // FSM
        state.IDLE:
            txReg.d = 1; // idle high (UART standard)
            if (!blockFlag.q) {
                busy = 0; // not busy
                bitCtr.d = 0; // reset counter
                ctr.d = 0; // reset counter
                if (new_data) { // request to send data?
                    savedData.d = data; // save the data
                    state.d = state.START_BIT; // switch states
                }
            }
        state.START_BIT:
            ctr.d = ctr.q + 1; // increment counter
            txReg.d = 0; // start bit is low
    }
}
```

```

if (ctr.q == CLK_PER_BIT - 1){ // if ctr is the max value
    ctr.d = 0; // reset counter
    state.d = state.DATA; // switch states
}

state.DATA:
txReg.d = savedData.q[bitCtr.q]; // output the data bit
ctr.d = ctr.q + 1; // increment counter
if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
    ctr.d = 0; // reset counter
    bitCtr.d = bitCtr.q + 1; // increase bit counter
    if (bitCtr.q == 7) // if we have sent all the bits
        state.d = state.STOP_BIT; // switch states
}

state.STOP_BIT:
txReg.d = 1; // stop bit is high
ctr.d = ctr.q + 1; // increase counter
if (ctr.q == CLK_PER_BIT - 1) { // if ctr is the max value
    state.d = state.IDLE; // switch states
}

default: state.d = state.IDLE; // if state is invalid reset to idle
}
}

```

ANEXO O

Programación en Mojo IDE; Constraints; def_bt.ufc

```
NET "bt_tx" LOC = P34 | IOSTANDARD = LVTTTL; //conectar al pin de transmisión del  
módulo bluetooth
```

```
NET "bt_rx" LOC = P32 | IOSTANDARD = LVTTTL; //conectar al pin de recepción del módulo  
bluetooth
```

```
NET "control<0>" LOC = P126 | IOSTANDARD = LVTTTL; //pin para controlar el control  
remoto del HEXAPODO
```

```
NET "control<1>" LOC = P123 | IOSTANDARD = LVTTTL; //pin para controlar el control remoto  
del HEXAPODO
```

ANEXO P

Programación en ARDUINO IDE; mojo_loader_final

```
#include "hardware.h"
#include "ring_buffer.h"
#include <SPI.h>
#include "flash.h"
/*****MINDWAVE CASCO*****/
#include <SoftwareSerial.h>
#define DEBUGOUTPUT 0
// checksum variables
byte generatedChecksum = 0;
byte checksum = 0;
int payloadLength = 0;
byte payloadData[64] = {0};
byte poorQuality = 0;
byte attention = 0;
byte meditation = 0;
// system variables
long lastReceivedPacket = 0;
boolean bigPacket = false;
byte ReadOneByte()
{
  int ByteRead;
  while(!Serial1.available());
  ByteRead = Serial1.read();
  #if DEBUGOUTPUT
  // Serial1.print((char)ByteRead); // echo the same byte out the USB serial (for debug purposes)
  #endif
  return ByteRead;
}
/*****
typedef enum {
  IDLE,
  READ_SIZE,
  WRITE_TO_FLASH,
  WRITE_TO_FPGA,
  VERIFY_FLASH,
  LOAD_FROM_FLASH
}
loaderState_t;
typedef enum {
  WAIT, START_LOAD, LOAD, SERVICE
}
taskState_t;
#if defined(__AVR_ATmega32U4__) // Mojo V3
#define BUFFER_SIZE 1024
#define SERIAL_STOP 1000
#define SERIAL_CUT 1020
#elif defined(__AVR_ATmega16U4__) // Mojo V2
#define BUFFER_SIZE 512
#define SERIAL_STOP 500
#define SERIAL_CUT 510
#endif
uint8_t loadBuffer[BUFFER_SIZE + 128];
RingBuffer_t adcBuffer, serialBuffer;
volatile taskState_t taskState = SERVICE;
uint8_t adcPort = 0x0F;
```

```

volatile uint8_t convPort = 0x0F;
/* This is where you should add your own code! Feel free to edit anything here.
   This function will work just like the Arduino loop() function in that it will
   be called over and over. You should try to not delay too long in the loop to
   allow the Mojo to enter loading mode when requested. */
int cont;
void userLoop()
{
// uartTask(); //comunicacion usada por defecto por embeddedmicro.com para habilitar el acceso
desde la fpga al PUERTOUSB
// adcTask(); //comunicacion usada por defecto por embeddedmicro.com para habilitar el acceso
desde la fpga a las ADC

//*****Comunicacion serial entre FPGA y AVR*****
// if(Serial1.available()
// {
//   Serial.write(Serial1.read());
// }
// if(Serial.available()
// {
//   Serial1.write(Serial.read());
// }
//*****Comunicacion serial entre AVR y nuevo
vinculo*****
// if(vinculo.available()
// {
//   Serial.print(vinculo.read());
// }
// if(Serial.available()
// {
//   vinculo.write(Serial.read());
// }
//*****Adquiere los datos del CASCO
MINDWAVE*****
// revisar bits de sincronismo y cálculo del checksum necesario para conocer si existe
comunicacion
// y obtener los datos necesarios y conocidos
if(ReadOneByte() == 170)
{
if(ReadOneByte() == 170)
{
payloadLength = ReadOneByte();
if(payloadLength > 169) //Payload length cannot be greater than 169
return;
generatedChecksum = 0;
for(int i = 0; i < payloadLength; i++)
{
payloadData[i] = ReadOneByte(); //Read payload into memory
generatedChecksum += payloadData[i];
}
checksum = ReadOneByte(); //Read checksum byte from stream
generatedChecksum = 255 - generatedChecksum; //Take one's compliment of generated
checksum
if(checksum == generatedChecksum)
{
poorQuality = 200;
attention = 0;
meditation = 0;
}
}
}

```

```

for(int i = 0; i < payloadLength; i++) // Parse the payload
{
  switch (payloadData[i])
  {
    case 2:
      i++;
      poorQuality = payloadData[i];
      bigPacket = true;
      break;
    case 4:
      i++;
      attention = payloadData[i];
      break;
    case 5:
      i++;
      meditation = payloadData[i];
      break;
    case 0x80:
      i = i + 3;
      break;
    case 0x83:
      i = i + 25;
      break;
    default:
      break;
  }
}
#endif !DEBUGOUTPUT
// si el checksum es el correcto puede se acceder a los datos que envia el mindwave
// variables = "poorQuality" "attention" "meditation" "lastReceivedPacket"
if(bigPacket)
{
  Serial.print("CalidadBaja: ");
  Serial.print(poorQuality, DEC);
  Serial.print(" Atencion: ");
  Serial.print(attention, DEC);
  Serial.print(" Meditacion: ");
  Serial.print(meditation, DEC);
  Serial.print(" Tiempo entre paquetes: ");
  Serial.print(millis() - lastReceivedPacket, DEC);
  lastReceivedPacket = millis();
  Serial.print("\n");
  if(attention>=60 && meditation<60)
  {
    //Serial.print(1,DEC);
    Serial1.print(1,DEC);
  }
  else if(attention<60 && meditation>=60)
  {
    //Serial.print(2,DEC);
    Serial1.print(2,DEC);
  }
  else
  {
    //Serial.print(3,DEC);
    Serial1.print(3,DEC);
  }
}
}

```

```

#endif
    bigPacket = false;
  }
}
}
}
/*****
*/
/* this is used to undo any setup you did in initPostLoad */
void disablePostLoad() {
  ADCSRA = 0; // disable ADC
  UCSR1B = 0; // disable serial port
  SPI.end(); // disable SPI
  SET(CCLK, LOW);
  OUT(PROGRAM);
  SET(PROGRAM, LOW); // reset the FPGA
  IN(INIT);
  SET(INIT, HIGH); // pullup on INIT
}
/* Here you can do some setup before entering the userLoop loop */
void initPostLoad() {
  //Serial.flush();
  Serial1.begin(500000); //PUERTO SERIAL1 HABILITADO TRABAJANDO A LA
VELOCIDAD CONFIGURADA
  //PREVIAMENTE EN LA FPGA
  SPI.begin();
  // These buffers are used by the demo ADC/Serial->USB code to prevent dropped samples
  RingBuffer_InitBuffer(&adcBuffer, loadBuffer, 128);
  RingBuffer_InitBuffer(&serialBuffer, loadBuffer + 128, BUFFER_SIZE);
  adcPort = 0x0f; // disable the ADC by default
  ADC_BUS_DDR &= ~ADC_BUS_MASK; // make inputs
  ADC_BUS_PORT &= ~ADC_BUS_MASK; // no pull ups
  /* IMPORTANTE: ESTE CODIGO HA SIDO DESABILITADO DEBIDO A QUE ESTE
CODIGO
* ESTABLECE EL REEMPLAZO AL SERIAL1 QUE TRABAJA A UNA VELOCIDAD DE
500000 BAUDIOS
* PARA EVITAR PROBLEMAS DE PROGRAMACION Y PERMITA UTILIZAR DICHO
BUFFER DE
* COMUNICACIONES DIRECTAMENTE Y PODER HACER USO DE LA LIBRERIA
SOFTSERIAL DE ARDUINO
* ING. PC
// Again, the Arduino libraries didn't offer the functionality we wanted
// so we access the serial port directly. This sets up an interrupt
// that is used with our own buffer to capture serial input from the FPGA
UBRR1 = 1; // 0.5 M Baud
UCSR1C = (1 << UCSZ11) | (1 << UCSZ10);
UCSR1A = (1 << U2X1);
UCSR1B = (1 << TXEN1) | (1 << RXEN1) | (1 << RXCIE1);
*/
// Setup all the SPI pins
SET(CS_FLASH, HIGH);
OUT(SS);
SET(SS, HIGH);
SPI_Setup(); // enable the SPI Port
DDRD |= (1 << 3);
DDRD &= ~(1 << 2);
PORTD |= (1 << 2);
// This pin is used to signal the serial buffer is almost full
OUT(TX_BUSY);

```

```

SET(TX_BUSY, LOW);
// set program as an input so that it's possible to use a JTAG programmer with the Mojo
IN(PROGRAM);
// the FPGA looks for CCLK to be high to know the AVR is ready for data
SET(CCLK, HIGH);
IN(CCLK); // set as pull up so JTAG can work
}
/* We needed more flexibility than the Arduino libraries provide. This sets
the ADC up to free-run and call an interrupt when each new sample is ready */
void configADC(uint8_t preScaler, uint8_t highPower, uint8_t refSelect, uint8_t port) {
  ADCSRA = (0 << ADEN); //disable
  ADMUX = (refSelect << REFS0) | (port & 0x07);
  if (port > 7)
    ADCSRB = (1 << MUX5) | (highPower << ADHSM);
  else
    ADCSRB = (highPower << ADHSM);
  convPort = port;
  ADCSRA = (1 << ADEN) | (1 << ADSC) | (1 << ADATE) | (1 << ADIE)
    | (preScaler << ADPS0);
}
void setup() {
  /* Disable clock division */
  clock_prescale_set(clock_div_1);
  OUT(CS_FLASH);
  SET(CS_FLASH, HIGH);
  OUT(CCLK);
  OUT(PROGRAM);
  /* Disable digital inputs on analog pins */
  DIDR0 = 0xF3;
  DIDR2 = 0x03;
  Serial.begin(115200); // Baud rate does nothing
  sei(); // enable interrupts
  loadFromFlash(); // load on power up
  initPostLoad();
}
void loop() {
  static loaderState_t state = IDLE;
  static int8_t destination;
  static int8_t verify;
  static uint32_t byteCount;
  static uint32_t transferSize;
  int16_t w;
  uint8_t bt;
  uint8_t buffIdx;
  switch (taskState) {
    case WAIT:
      break;
    case START_LOAD: // command to enter loader mode
      disablePostLoad(); // setup peripherals
      taskState = LOAD; // enter loader mode
      state = IDLE; // in idle state
      break;
    case LOAD:
      w = Serial.read();
      bt = (uint8_t) w;
      if (w >= 0) { // if we have data
        switch (state) {
          case IDLE: // in IDLE we are waiting for a command from the PC

```



```

byteCount = 0;
transferSize = 0;
if (bt == 'F') { // write to flash
    destination = 0; // flash
    verify = 0; // don't verify
    state = READ_SIZE;
    Serial.write('R'); // signal we are ready
}
if (bt == 'V') { // write to flash and verify
    destination = 0; // flash
    verify = 1; // verify
    state = READ_SIZE;
    Serial.write('R'); // signal we are ready
}
if (bt == 'R') { // write to RAM
    destination = 1; // ram
    state = READ_SIZE;
    Serial.write('R'); // signal we are ready
}
if (bt == 'E') { //erase
    eraseFlash();
    Serial.write('D'); // signal we are done
}
//Serial.flush();
break;
case READ_SIZE: // we need to read in how many bytes the config data is
transferSize |= ((uint32_t) bt << (byteCount++ * 8));
if (byteCount > 3) {
    byteCount = 0;
    if (destination) {
        state = WRITE_TO_FPGA;
        initLoad(); // get the FPGA read for a load
        startLoad(); // start the load
    }
    else {
        state = WRITE_TO_FLASH;
        eraseFlash();
    }
    Serial.write('O'); // signal the size was read
    //Serial.flush();
}
break;
case WRITE_TO_FLASH:
// we can only use the batch write for even addresses
// so address 5 is written as a single byte
if (byteCount == 0)
    writeByteFlash(5, bt);
buffIdx = (byteCount++ - 1) % 256;
loadBuffer[buffIdx] = bt;
if (buffIdx == 255 && byteCount != 0)
    writeFlash(byteCount + 5 - 256, loadBuffer, 256); // write blocks of 256 bytes at a time for
speed
if (byteCount == transferSize) { // the last block to write
    if (buffIdx != 255) // finish the partial block write
        writeFlash(byteCount + 5 - (buffIdx + 1), loadBuffer,
            buffIdx + 1);
    delayMicroseconds(50); // these are neccary to get reliable writes
    uint32_t size = byteCount + 5;

```

```

    for (uint8_t k = 0; k < 4; k++) {
        writeByteFlash(k + 1, (size >> (k * 8)) & 0xFF); // write the size of the config data to the
flash
        delayMicroseconds(50);
    }
    delayMicroseconds(50);
    writeByteFlash(0, 0xAA); // 0xAA is used to signal the flash has valid data
    Serial.write('D'); // signal we are done
    //Serial.flush(); // make sure it sends
    if (verify) {
        state = VERIFY_FLASH;
    }
    else {
        state = LOAD_FROM_FLASH;
    }
}
break;
case WRITE_TO_FPGA:
    sendByte(bt); // just send the byte!
    if (++byteCount == transferSize) { // if we are done
        sendExtraClocks(); // send some extra clocks to make sure the FPGA is happy
        state = IDLE;
        taskState = SERVICE; // enter user mode
        initPostLoad();
        Serial.write('D'); // signal we are done
        //Serial.flush();
    }
    break;
case VERIFY_FLASH:
    if (bt == 'S') {
        byteCount += 5;
        for (uint32_t k = 0; k < byteCount; k += 256) { // dump all the flash data
            uint16_t s;
            if (k + 256 <= byteCount) {
                s = 256;
            }
            else {
                s = byteCount - k;
            }
            readFlash(loadBuffer, k, s); // read blocks of 256
            uint16_t br = Serial.write((uint8_t*) loadBuffer, s); // dump them to the serial port
            k -= (256 - br); // if all the bytes weren't sent, resend them next round
            //Serial.flush();
            delay(10); // needed to prevent errors in some computers running Windows (give it time
to process the data?)
        }
        state = LOAD_FROM_FLASH;
    }
    break;
case LOAD_FROM_FLASH:
    if (bt == 'L') {
        loadFromFlash(); // load 'er up!
        Serial.write('D'); // loading done
        //Serial.flush();
        state = IDLE;
        taskState = SERVICE;
        initPostLoad();
    }
}

```

```

        break;
    }
}
break;
case SERVICE:
    userLoop(); // loop the user code
    break;
}
}
/* This is called when any control lines on the serial port are changed.
It requires a modification to the Arduino core code to work.
This looks for 5 pulses on the DTR line within 250ms. Checking for 5
makes sure that false triggers won't happen when the serial port is opened. */
void lineStateEvent(unsigned char linestate)
{
    static unsigned long start = 0;
    static uint8_t falling = 0;
    if (!(linestate & LINESTATE_DTR)) {
        if ((millis() - start) < 250) {
            if (++falling >= 5)
                taskState = START_LOAD;
        }
        else {
            start = millis();
            falling = 1;
        }
    }
}
}
//adc puede retirarse ya que no se utilizan variables analogicas
/* This checks to see what port the FPGA is requesting. If it hasn't changed then
no worries, but if it has the ADC needs to be stopped and set to the new port.
It then empties the ADC buffer into the SPI bus so the FPGA can actually have the
ADC data. */
void adcTask() {
    static uint8_t preScaler = 0x05; // 32
    static uint8_t highPower = 1;
    static uint8_t refSelect = 0x01; // AVcc
    uint8_t adc_bus = (ADC_BUS_PIN & ADC_BUS_MASK) >> ADC_BUS_OFFSET;
    if (adc_bus != adcPort) { // did the requested ADC pin change?
        adcPort = adc_bus;
        if (adcPort < 2 || (adcPort < 10 && adcPort > 3)) { // 0,1,4,5,6,7,8,9
            configADC(preScaler, highPower, refSelect, adcPort); // reconfigure ADC
        }
        else { // pin is not valid
            ADCSRA = (0 << ADEN); //disable ADC
        }
    }
}

while (!RingBuffer_IsEmpty(&adcBuffer)) { // for all the samples
    // Grab two bytes from the ring buffer, do it directly for a speed gain
    uint8_t byte1 = *adcBuffer.Out;
    if (++adcBuffer.Out == adcBuffer.End)
        adcBuffer.Out = adcBuffer.Start;
    uint8_t byte2 = *adcBuffer.Out;
    if (++adcBuffer.Out == adcBuffer.End)
        adcBuffer.Out = adcBuffer.Start;
    uint_reg_t CurrentGlobalInt = GetGlobalInterruptMask();
    GlobalInterruptDisable();
}

```

```

    adcBuffer.Count -= 2; // actually remove the two bytes from the buffer
    SetGlobalInterruptMask(CurrentGlobalInt);
    SET(SS, LOW);
    uint8_t keyWord = SPI.transfer(byte1); // each sample is two bytes
    uint8_t config = SPI.transfer(byte2);
    SET(SS, HIGH);
    if (keyWord == 0xAA) { // the keyWord is used by the FPGA to config the ADC
        preScaler = config & 0x07;
        highPower = (config >> 3) & 0x01;
        refSelect = (config >> 4) & 0x03;
        configADC(preScaler, highPower, refSelect, adcPort);
    }
}
}
}
ISR(ADC_vect) { // new ADC sample, save it
    RingBuffer_Insert(&adcBuffer, ADCL );
    RingBuffer_Insert(&adcBuffer, (convPort << 4) | ADCH );
}
void serialRXEnable() {
    UCSR1B |= (1 << RXEN1);
}
void serialRXDisable() {
    UCSR1B &= ~(1 << RXEN1);
}
// ESTO ES UNA MODIFICACION DEL CODIGO DE EMBEDDED MICRO PARA LA
// HABILITACION DEL PUERTO SERIAL1
// EL CUAL REPRESENTA EL VINCULO ENTRE EL ATMEGA32U4 Y LA FPGA. Se excluyen
// las modificaciones
// que implican las configuraciones del puerto serial.
/*
static inline void Serial_SendByte(const char DataByte)
{
    while (!(UCSR1A & (1 << UDRE1)));
    UDR1 = DataByte;
}
/* This function handles all the serial to USB work. It works
   much the same way as the ADC task, but it just forwards data
   from one port to the other instead of the ADC to the FPGA. */
/*
void uartTask() {
    if (Serial) { // does the data have somewhere to go?
        uint16_t ct = RingBuffer_GetCount(&serialBuffer);
        if (ct > 0) { // is there data to send?
            if (serialBuffer.Out + ct <= serialBuffer.End) { // does it loop in our buffer?
                ct = Serial.write(serialBuffer.Out, ct); // dump all the data
                serialBuffer.Out += ct;
                if (serialBuffer.Out == serialBuffer.End)
                    serialBuffer.Out = serialBuffer.Start; // loop the buffer
            }
            else { // it looped the ring buffer
                uint8_t* loopend = serialBuffer.Out + ct;
                uint16_t ct2 = loopend - serialBuffer.End;
                uint16_t ct1 = ct - ct2;
                uint16_t ct1s = Serial.write(serialBuffer.Out, ct1); // dump first block
                if (ct1s == ct1) {
                    ct2 = Serial.write(serialBuffer.Start, ct2); // dump second block
                    serialBuffer.Out = serialBuffer.Start + ct2; // update the pointers
                }
            }
        }
    }
}

```

```

        ct = ct1 + ct2;
    }
    else {
        ct = ct1s;
        serialBuffer.Out += ct;
    }
}
uint_reg_t CurrentGlobalInt = GetGlobalInterruptMask();
GlobalInterruptDisable();

serialBuffer.Count -= ct; // update the count

SetGlobalInterruptMask(CurrentGlobalInt);

int count = RingBuffer_GetCount(&serialBuffer);

//if (count == 0)
// Serial.flush();

if (count < SERIAL_STOP) {
    serialRXEnable();
    TOGGLE(TX_BUSY); // re-enable the serial port
}
}
int16_t w;
while ((w = Serial.read()) >= 0) {
    Serial_SendByte(w);
}
}
}

ISR(USART1_RX_vect) { // new serial data!
*(serialBuffer.In) = UDR1;

if (++serialBuffer.In == serialBuffer.End)
    serialBuffer.In = serialBuffer.Start;

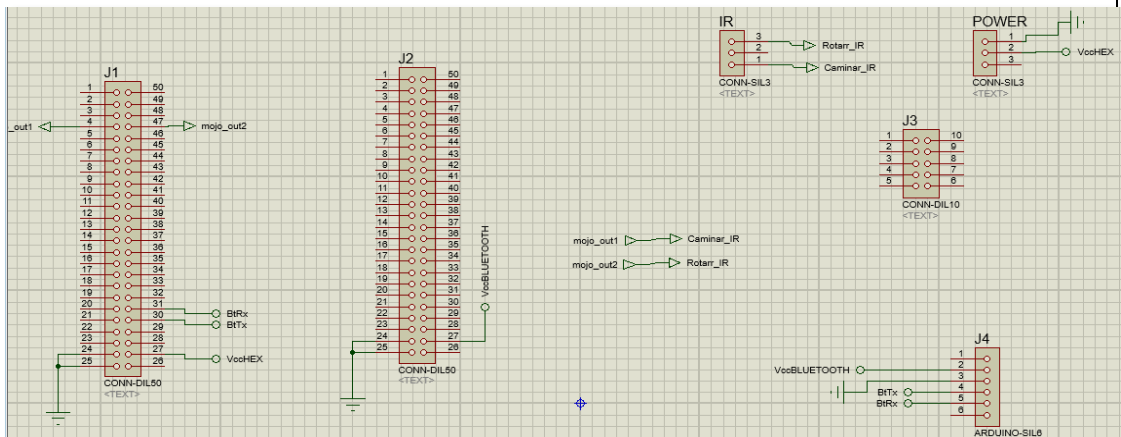
serialBuffer.Count++;
if (serialBuffer.Count >= SERIAL_STOP) { // are we almost out of space?
    if (serialBuffer.Count > SERIAL_CUT)
        serialRXDisable(); // if our flag is ignored disable the serial port so it doesn't clog things up
    } else {
        TOGGLE(TX_BUSY);
    }
}
}
*/

```

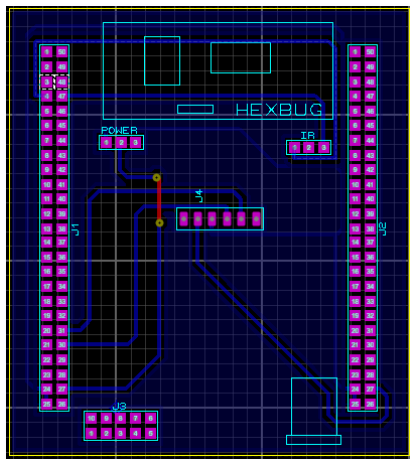
ANEXO Q

Diseño en Proteus del shield de conexión del sistema.

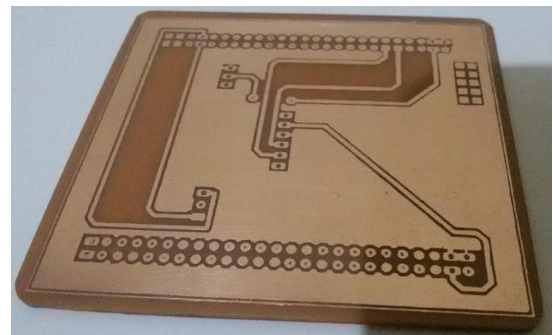
Captura esquemática



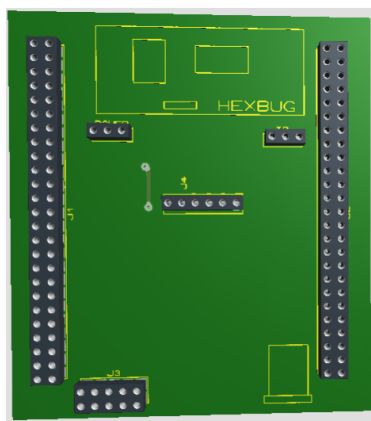
Captura PCB



Captura de placa real



Captura 3D



Captura 3D

