UACM

Universidad Autónoma de la Ciudad de México

NADA HUMANO ME ES AJENO

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

Desarrollo de una tarjeta electrónica basada en Esp32, para control de acceso de una chapa convencional

T E S I S

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADOS EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

P R E S E N T A N

EFRAIN CORONA JUAN APOLINAR

MARTIN ROBLES BECERRA

D I R E C T O R

M. EN I. CHRISTIAN AGUSTÍN VÁZQUEZ VILLANUEVA

Ciudad de México, julio de 2026.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

La presente obra y cada uno de sus elementos está protegido por la Ley Federal del Derecho de Autor; por la Ley de la Universidad Autónoma de la Ciudad de México, así como lo dispuesto por el Estatuto General Orgánico de la Universidad Autónoma de la Ciudad de México; del mismo modo por lo establecido en el Acuerdo por el cual se aprueba la Norma mediante la que se Modifican, Adicionan y Derogan Diversas Disposiciones del Estatuto Orgánico de la Universidad de la Ciudad de México, aprobado por el Consejo de Gobierno el 29 de enero de 2002, con el objeto de definir las atribuciones de las diferentes unidades que forman la estructura de la Universidad Autónoma de la Ciudad de México como organismo público autónomo y lo establecido en el Reglamento de Titulación de la Universidad Autónoma de la Ciudad de México.

Por lo que el uso de su contenido, así como cada una de las partes que lo integran y que están bajo la tutela de la Ley Federal de Derecho de Autor, obliga a quien haga uso de la presente obra a considerar que solo lo realizará si es para fines educativos, académicos, de investigación o informativos y se compromete a citar esta fuente, así como a su autor ó autores. Por lo tanto, queda prohibida su reproducción total o parcial y cualquier uso diferente a los ya mencionados, los cuales serán reclamados por el titular de los derechos y sancionados conforme a la legislación aplicable.

Dedico este trabajo a

mi padre Jose Mateo Juan Hernandez

mi madre Celerina Apolinar Uribe

y a mis hermanos Abraham Juan Apolinar y Nancy Juan Apolinar, quienes me apoyaron de manera incondicional, me brindaron su ayuda y consejos a lo largo de este camino, sin duda, este esfuerzo dio sus frutos.

Efrain Corona Juan Apolinar

“Nada humano me es ajeno”

Dedicado este trabajo a

mi madre Leticia Becerra Lugo

a mi padre Martin Robles Lugo

a mis tías Yolanda Robles Lugo y Lina Robles Lugo

y a mi hermano Alexis Robles Becerra, por su apoyo incondicional.

Martin Robles Becerra

Agradecimientos.

Mis más sinceros agradecimientos a mi familia que estuvo para mi apoyándome a pesar de las dificultades que presentamos y el saber que estamos unidos considero que fue parte pilar de mi finalización de licenciatura.

Mi papá y mamá que estuvieron a mi lado diciéndome palabras de apoyo con el corazón en la mano gracias.

Mi tía Yolanda y tía Lina por igual, estuvieron en mis momentos más difíciles de mi vida y dándome ánimos para finalizar esta etapa tan importante en mi vida.

A mis compañeros que fui compartiendo cursos y sobre todo a mis amigos que compartimos diversidades en los cursos, que pudimos estrechar lazos muy fuertes.

Tanto a los docentes e institución a lo largo de mi trayecto aprendí bastante no solo académicamente si no personalmente, cada docente tuvo su granito de arena que siempre voy a llevar en el corazón, son grandes personas que siempre recordaré y a la institución por darme la oportunidad de cumplir uno de mis sueños de la vida con empatía y siempre con el lema:

“Nada humano me es ajeno”

Martin Robles Becerra

Primero que nada, agradezco a mi familia por todo el apoyo y motivación que me han brindado durante todo este tiempo, son parte fundamental en mi vida y gracias ellos he podido cumplir con mis metas.

Agradezco la Universidad Autónoma de la Ciudad de Mexico (UACM) por darme la oportunidad de continuar con mis estudios universitarios en la carrera de ingeniería en Sistemas en Electrónicos Industriales, sin esta, no hubiera podido continuar con mis estudios, además de contar con áreas en las que se puede convivir con los amigos y compañeros, que sirven para el des estrés y convivencia dentro de la universidad, fuera de lo escolar.

Agradezco a los profesores que fueron mis docentes a lo largo de la carrera, me enseñaron muchas cosas, tanto escolares como también, dándome consejos personales.

Agradezco a mis amigos y compañeros de clase, con los que llegue a compartir muchos momentos buenos y momentos de mucha presión, de una u otra manera, salimos adelante con todos los trabajos, siempre teniendo la mentalidad positiva de aprobar las materias y cumplir nuestras mentas.

Agradezco a todos y cada uno, de los que formaron parte de mi vida universitaria. ¡Muchas Gracias a Todos!

Efrain Corona Juan Apolinar

“Nada humano me es ajeno”

INDICE GENERAL

Agradecimientos. *2*

I. Introducción. *10*

II. Planteamiento del problema. *11*

IV. Objetivos. *12*

Capítulo 1. Marco teórico *14*

1.1. Introducción a los microcontroladores. *14*

1.2. Arquitectura general de un microcontrolador. *14*

1.3. Microcontrolador ESP32. ..... *17*

1.4. Etapa de Control y Procesamiento. *18*

1.4.1. Arquitectura de Microcontroladores para el Internet de las Cosas (IoT). *18*

1.4.2. Conectividad y consumo energético. *20*

1.4.3. Arquitectura ESPNOW. *21*

1.4.4. Servidor en la Nube e Internet de las Cosas (Blynk IoT). *21*

Mecanismo de pines virtuales. *22*

1.4.5. Tarjeta Identificación por Radiofrecuencia (RFID) como control de acceso. *23*

1.4.6. Bluetooth de Bajo Consumo (BLE - Bluetooth Low Energy) *24*

1.5. Etapa de Acoplamiento y Potencia. *27*

1.5.1. Aislamiento de Señal: El Optoacoplador. *27*

1.5.2. Concepto de aislamiento. *27*

1.6. El relevador: actuación electromecánica de potencia. *29*

1.6.1. Constitución y principio de funcionamiento. *29*

1.7. Etapa de Amplificación de Corriente y Conmutación: El Transistor Bipolar de Juntura (BJT). *31*

1.7.1. Principio de Operación en Conmutación (Corte y Saturación) 31

Capítulo 2. 36

Programación y configuración del software. 36

Programación: software de programación utilizado para el ESP32. 36

2.1. Instalación de software. 36

2.1.1. Configuración de Arduino IDE. 36

2.2. Código del programa. 39

2.2.1. Etapas de funcionamiento. 60

2.2.1.1. librerías. 60

2.2.1.2. Cinemática del mecanismo. 61

2.2.1.3. Lazo de Control No Bloqueante (loop). 62

2.2.1.4. Interfaz visual 62

2.3. Descarga y configuración de aplicaciones (APK). 63

2.3.1. Serial Bluetooth Terminal. 63

2.3.2. Blynk IoT. 66

Capítulo 3. 70

Diseño de los prototipos. 70

3.1. Placa PCB. 70

3.1.2. Resultado del esquemático. 72

3.2. Acomodo, enrutamiento de pistas y resultados. 74

3.2.1. Criterios de acomodo de componentes. 74

3.2.1.1. Resultado del acomodo de componentes. 74

3.2.2. Estrategia de Enrutamiento de Pistas 76

3.2.2.1 Enrutamiento de pistas para PCB 76

3.2.2.2. resultado de enrutamiento 78

3.2.3. Elaboración de maqueta de la chapa. 79

Capítulo 4 83

Pruebas individuales físicas de los prototipos y sus módulos. 83

4.1Prueba de módulos de conexión. 83

4.1.1. Módulo de Procesamiento Central ESP32. 83

4.1.2. (Lector RFID-RC522)..... 84

4.1.3. Módulo de Aislamiento y Conmutación (Optoacoplador y Transistor BC547). 85

4.1.4. Módulo de Actuación Mecánica (Servomotor MG995 y Chapa Phillips) 85

4.1.5 Control remoto (ESPNOW). 86

4.1.6. Salidas de control de acceso..... 87

4.1.7. Resultado final del proyecto. 87

INDICE DE TABLAS

Tabla 1. Tabla comparativa de ESP32. 18

Tabla 2. Tabla de diferentes operaciones para bajo consumo. 21

Tabla 3 Comparativa de un Bluetooth Clásico vs. BLE..... 26

Tabla 4. Tabla de posibles sustitutos de relevadores y sus características. 30

Tabla 5. Partes de la maqueta con su respectiva función..... 80

INDICE DE FIGURAS

Figura 1. Flujo Operativo de la Etapa de Potencia Completa. 29

Figura 2. BOARDS MANAGER..... 37

Figura 3. Administrador de dispositivos de windows 11. 37

Figura 4. Tools de Arduino IDE. 38

Figura 5. Selección de versión de la ESP32. 38

Figura 6. Diagrama de flujo. *39*

Figura 7. APK en la app store. *63*

Figura 8. Interfaz principal de Serial Bluetooth Terminal. *64*

Figura 9. Ejemplo de conexión via bluetooth en un dispositivo movil. *64*

Figura 10. Configuración de bluetooth. *65*

Figura 11. Dispositivos para vincular. *65*

Figura 12. Apk Blynk IoT en play store. *66*

Figura 13. Apartado para crear plantillas. *66*

Figura 14. Valores a cambiar para el pin virtual. *67*

Figura 15. Ajustes del pin virtual. *68*

Figura 16. Diagrama esquemático principal en el entorno de desarrollo easyEDA. *72*

Figura 17. Diagrama esquemático del control remoto en el entorno de desarrollo easyEDA. *73*

Figura 18. Diseño de cada componente en la placa PCB. *75*

Figura 19. Diseño del control remoto. *75*

Figura 20. Diseño completo de la PCB. *78*

Figura 21. Diseño de control remoto PCB. *79*

Figura 22. Vista general de la maqueta de pruebas con la integración final de los componentes mecánicos *81*

Figura 23. Módulo de Procesamiento Central ESP32.Módulo de Autenticación *84*

Figura 24. Lector RFID en funcionamiento. *85*

Figura 25. Chapa y servomotor. *86*

Figura 26. Diseño final de la placa del control remoto. *87*

Figura 27. Placa PCB y accesos. *88*

Figura 28. Placa electrónica instalada en la chapa. 88

Conclusión. 90

Apéndice A. 92

Bibliografía. 100

I. Introducción.

En la actualidad, la facilidad en el acceso con sistemas de control se ha convertido en un aspecto fundamental en diversos entornos como viviendas, oficinas, laboratorios y establecimientos comerciales. Tradicionalmente, el control de acceso se ha basado en el uso de cerraduras mecánicas que funcionan mediante llaves físicas. Aunque este método ha sido ampliamente utilizado durante muchos años, presenta diversas limitaciones, entre las que destacan la pérdida o duplicación no autorizada de llaves, la falta de control sobre las personas que ingresan a un espacio y relativamente menos aplicado a zonas de pueblos originarios de México.

Con el avance de la tecnología y el desarrollo de los sistemas electrónicos, han surgido nuevas alternativas que permiten mejorar significativamente los mecanismos de seguridad y control. En este contexto, los sistemas basados en microcontroladores se han convertido en herramientas fundamentales para el desarrollo de soluciones tecnológicas aplicadas a la automatización y al control de dispositivos electrónicos. Estos sistemas permiten integrar sensores, actuadores y módulos de comunicación para crear dispositivos capaces de interactuar con su entorno y ofrecer funciones más avanzadas que los sistemas tradicionales.

Dentro de estas tecnologías destaca el microcontrolador ESP32, el cual se ha posicionado como una plataforma ampliamente utilizada en proyectos de electrónica y sistemas embebidos debido a sus múltiples características, como su capacidad de procesamiento, bajo consumo de energía y la integración de conectividad inalámbrica mediante Wi-Fi y Bluetooth. Estas características permiten desarrollar aplicaciones de control, monitoreo y automatización que pueden ser gestionadas de manera local o remota, lo que amplía considerablemente las posibilidades de implementación en sistemas de seguridad.

II. Planteamiento del problema.

En muchos hogares el acceso aún depende de chapas mecánicas convencionales que requieren llaves físicas. Este método presenta diversas limitaciones: riesgo de pérdida o duplicación de llaves, falta de control sobre quién accede, y baja capacidad de integración con sistemas digitales. Además, en entornos donde varias personas requieren acceso, la gestión manual de llaves se vuelve ineficiente y poco segura. Por otro lado, los sistemas comerciales de control de acceso suelen tener costos elevados, requerir infraestructura especializada o ser difíciles de adaptar a chapas convencionales ya instaladas. Esto genera la necesidad de soluciones económicas, seguras, flexibles y de fácil implementación que permitan modernizar el control de acceso sin reemplazar completamente la cerradura existente puesto que es mucho más barato y automatizado diseñar una tarjeta que mueva el mecanismo existente que obligar al usuario a comprar una cerradura electrónica comercial. Ante esta situación, surge la propuesta de desarrollar una tarjeta electrónica basada en ESP32 capaz de controlar una chapa convencional mediante distintos métodos de autenticación y monitoreo, brindando una solución práctica que mejore la seguridad, la confiabilidad y la facilidad de instalación en diferentes aplicaciones residenciales, académicas y comerciales.

III. Justificación.

El desarrollo de este proyecto nace de una necesidad muy clara en el día a día: la seguridad y el control de acceso en nuestros espacios. A diferencia del uso de la chapa convencional y posiblemente el uso de un candado común permite restringir el acceso de manera más eficiente. Al eliminar la dependencia de una llave física, se reduce el riesgo de una posible pérdida o duplicado y se abre la posibilidad de interacción de diversas formas como RFID, Bluetooth, Wi-Fi y ESPNOW.

Muchas de las soluciones comerciales de seguridad en el mercado actual son costosas cerrando la posibilidad de modificar o reparar ante cualquier situación que se presenta. El diseño de esta chapa demuestra que es posible construir un

sistema seguro, confiable y de alta calidad utilizando componentes electrónicos accesibles y programación que puede ser muy versátil.

IV. Objetivos.

Objetivo general.

Desarrollar una tarjeta electrónica basada en ESP32 que permita controlar el acceso mediante una chapa convencional, integrando métodos de autenticación, accionamiento seguro del mecanismo de apertura, con enfoque en seguridad, confiabilidad y facilidad de instalación.

Objetivos específicos.

- Definir los requisitos del sistema: tipo de chapa, alimentación disponible, tiempo de apertura, modos de acceso y condiciones de seguridad (fallo de energía, uso manual).
- Diseñar la arquitectura electrónica: selección de fuente de alimentación, regulación (5V) y comunicación del ESP32.
- Diseñar la etapa de potencia para accionar la chapa: relé según corriente, diodo flyback para cargas inductivas, aislamiento mediante optoacoplador.
- Implementar el método de autenticación:
    1. Acceso por RFID.
    2. Acceso vía bluethooth.
    3. Acceso ESPNOW.
    4. Conexión Wi-Fi.
- Agregar interfaz de usuario local: LEDs, buzzer o display para indicar estado, acceso concedido o denegado.
- Desarrollar el firmware del ESP32: control de estados (Abierto/Cerrado), lectura de credenciales, lógica anti-rebote y temporización de apertura.
- Diseñar el PCB y el ensamble: ruteo, separación de potencia y señal, conectores y adaptación para instalación en una chapa convencional.
- Validar el sistema con pruebas: consumo, rango de comunicación, tiempo de respuesta y operación continua.

Capítulo 1. Marco teórico.

1.1. Introducción a los microcontroladores.

Los microcontroladores son dispositivos electrónicos programables diseñados para realizar tareas de control específicas dentro de sistemas embebidos. Están formados por un microprocesador, memoria (RAM, ROM, Flash) y periféricos de entrada/salida integrados en un solo circuito integrado.

El propósito principal de un microcontrolador es leer señales del entorno (entradas), procesar esa información bajo una lógica programada y generar una respuesta (salidas). Esta capacidad de "decisión" lo hace ideal para la automatización de la chapa electrónica, donde debe validar una credencial y activar un actuador físico.

La diferencia entre microprocesador y microcontrolador es:

- **Microprocesador (CPU):** Diseñado para aplicaciones generales de alto rendimiento. Requiere memoria RAM, almacenamiento y periféricos externos.
- **Microcontrolador (MCU):** Optimizado para bajo consumo y tareas de control. Incluye todos los componentes necesarios dentro del mismo encapsulado, lo que reduce el tamaño del circuito y el costo de fabricación.

1.2. Arquitectura general de un microcontrolador.

Para tener un mejor entendimiento sobre cómo funciona y gestiona la chapa electrónica por medio de la ESP32, se va a desglosar los bloques internos que permiten su funcionamiento. La arquitectura más común en microcontroladores modernos es la Arquitectura Harvard, donde existen buses separados para las instrucciones y para los datos, permitiendo una ejecución más rápida.

A. Unidad Central de Procesamiento (CPU)

Es la parte central del dispositivo. Se encarga de buscar las instrucciones en la memoria, decodificarlas y ejecutarlas. En el caso del ESP32, contamos con un procesador Xtensa® Dual-Core LX6 de 32 bits, lo que le permite realizar multitarea (por ejemplo, gestionar la conexión Wi-Fi en un núcleo mientras controla la chapa en el otro).

B. Unidades de Memoria

- Memoria ROM (Read Only Memory): Es un tipo de memoria física (un chip) que viene de fábrica. Contiene el software básico y las instrucciones permanentes para que tu equipo pueda encender y reconocer sus componentes.
- Memoria RAM: Se utiliza para almacenar datos temporales y variables durante la ejecución del programa. Es mucho más rápida pero volátil.
- EEPROM / NVS: Utilizada para guardar datos que no deben perderse (como el listado de tarjetas RFID autorizadas o contraseñas de Wi-Fi).

C. Periféricos de Entrada y Salida (I/O)

Son dispositivos externos que se conectan al microcontrolador.

- GPIO (General Purpose Input/Output): Pines que pueden configurarse para leer un botón o enviar una señal de activación al relé de la chapa.
- ADC (Convertidor Analógico a Digital): Transforma voltajes analógicos (sensores) en números binarios que el CPU entiende.
- Temporizadores (Timers): Permiten controlar cuánto tiempo debe permanecer abierta la chapa antes de bloquearse de nuevo.

D. Módulos de Comunicación

Permiten que el microcontrolador "hable" con otros dispositivos:

- UART: Comunicación serial (usada para programar el ESP32 y ver el monitor serie).

- SPI / I2C: Protocolos para conectar sensores de huella digital, lectores RFID o pantallas LCD. Conectividad Dual
- Su mayor ventaja competitiva es la integración nativa de:
- Wi-Fi (802.11 b/g/n): Permite conectar la chapa a una red local o a la nube para monitoreo remoto.
- Bluetooth de Modo Dual: Soporta Bluetooth Classic (para transferencia de datos convencional) y Bluetooth Low Energy (BLE), que es ideal para interactuar con aplicaciones móviles consumiendo el mínimo de energía.
- Seguridad Integrada

El ESP32 incluye por hardware:

- Aceleradores para criptografía (AES, SHA-2, RSA, ECC).
- Arranque seguro (Secure Boot) y cifrado de memoria Flash, lo que impide que alguien robe el código o las llaves de acceso clonando el chip.
- Módulos Inalámbricos (específico del ESP32): Pilas de protocolos integradas para Wi-Fi y Bluetooth.

E. Reloj del Sistema (Oscilador)

Genera los pulsos de sincronización necesarios para que todos los componentes internos trabajen al mismo ritmo. El ESP32 puede trabajar a frecuencias de hasta 240 MHz, lo que garantiza una respuesta casi instantánea en sistemas de control.

1.3. Microcontrolador ESP32.

El ESP32 es una serie de sistemas en chip (SoC) de bajo costo y bajo consumo de energía. Su arquitectura está diseñada para ser robusta, eficiente y, sobre todo, conectada.

La arquitectura y núcleos a diferencia de los microcontroladores básicos de un solo núcleo, la mayoría de las variantes del ESP32 (como el WROOM-32) cuentan con el microprocesador Xtensa® Dual-Core LX6 de 32 bits.

A diferencia de los microcontroladores básicos de un solo núcleo, la mayoría de las variantes del ESP32 (como el WROOM-32) cuentan con el microprocesador **Xtensa® Dual-Core LX6 de 32 bits**.

- **Doble Núcleo:** Permite que un núcleo se dedique exclusivamente a la gestión de la conectividad (Wi-Fi/Bluetooth) mientras el otro ejecuta la lógica de control del usuario (leer el sensor RFID o activar la chapa). Esto evita retardos o "cuelgues" en el sistema de seguridad.
- **Coprocesador de Ultra Bajo Consumo (ULP):** Es un pequeño procesador adicional que puede realizar tareas básicas (como vigilar un sensor de movimiento) mientras los núcleos principales están en "modo sueño", ahorrando batería significativamente.

Caracteristica	**ESP32 (WROOM**	**Arduino uno (Atmega328P)**	**ESP8266 (NodeMCU)**	**STM (Blue Pill)**
Arquitectura	32 bits	8 bits	32 bits	32 bits
Velocidad de reloj	160 a 240 MHz	16MHz	80 MHz	72 MHz
Núcleos	2(Dual-Core)	1	1	1
Memoria RAM	520 KB	2KB	160 KB	20 KB
Memeria Flash	4 MB (ext. Hasta 16)	32 KB	4 MB	64 KB

Wi-Fi/BT	Integrado (Ambos)	No (requiere módulos)	Solo Wi-Fi	No (requiere módulos)
Pines GPIO	34	14	17	37
Resolución ADC	12 bits (4096 niveles)	10 bits (1024 niveles)	10 bits (1 solo pin)	12 bits (4096 niveles)
Voltaje lógico	3.3 V	5 V	3.3 V	3.3 V
Ideal para…	IoT, seguridad, audio	Educación. Prototipos simples	IoT básicos, sensores	Control industrial, precisión

Tabla 1. Tabla comparativa de ESP32.

1.4. Etapa de Control y Procesamiento.

La etapa de control y procesamiento constituye la parte central, encargada de coordinar las interfaces de lectura, ejecutar los algoritmos de validación y tener control en los actuadores electromecánicos. La totalidad de esta lógica digital se centraliza en el ESP32, específicamente bajo el módulo interconectado ESP32-WROOM.

1.4.1. Arquitectura de Microcontroladores para el Internet de las Cosas (IoT).

En el diseño de sistemas embebidos orientados al Internet de las Cosas (IoT), la selección de la unidad central de procesamiento es un factor crítico que define la viabilidad, el costo y el rendimiento del prototipo. Para el desarrollo del presente sistema automatizado de control de acceso, se seleccionó el circuito integrado **ESP32**, un Sistema en Chip (SoC, por sus siglas en inglés: *System on Chip*) de bajo costo y consumo energético diseñado por la firma Espressif Systems.

La integración de múltiples periféricos de comunicación y procesamiento en una sola oblea de silicio clasifica al ESP32 como una solución robusta frente a los microcontroladores convencionales de 8 bits, al ofrecer capacidades de cómputo en paralelo y conectividad inalámbrica nativa.

1.4.2. Conectividad y consumo energético.

Conectividad Concurrente: El hardware integra soporte nativo para los estándares de red Wi-Fi 802.11 b/g/n y Bluetooth v4.2 BR/EDR y BLE (Bluetooth Low Energy). Esto permite implementar, sin necesidad de hardware adicional, un esquema de control híbrido donde coexisten la gestión en la nube, el control local de respaldo y la comunicación inter-dispositivo de baja latencia mediante el protocolo propietario.

En la tabla 1.1 donde las aplicaciones necesitan que el sistema opere mediante fuentes de energías limitadas, como celdas solares de ion-Litio de 3.7, el SoC ofrece una estructura que administra la energía de una manera avanzada que se basa en diferentes modos de operación-

Modo de Operación	**Elementos Activos**	**Consumo de Corriente Típico**	**Aplicación Teórica**
Active Mode	CPU Cores activos, Wi-Fi/BT encendidos, periféricos energizados.	160 mA a 260mA	Procesamiento de tramas de red y movimiento del servo actuador.
Modem-sleep	CPU operativo, periféricos activos, Wi-Fi/BT apagados temporalmente.	20mA a 68mA	Espera de lectura de tarjetas RFID locales.
Light-sleep	CPU suspendida, periféricos y RAM retenidos, reloj interno pausado.	0.8mA	Modo de espera general de la cerradura por la noche.

Modo de Operación	Elementos Activos	Consumo de Corriente Típico	Aplicación Teórica
Deep-sleep	Todo el chip apagado a excepción del coprocesador ULP y memoria RTC.	10 microA a 150 microA	Sistemas autónomos de monitoreo pasivo.

Tabla 2. Tabla de diferentes operaciones para bajo consumo.

1.4.3. Arquitectura ESPNOW.

A diferencia de las conexiones Wi-Fi convencionales que requieren un proceso de negociación en tres pasos, autenticación mediante contraseñas complejas (WPA2) y la asignación de una dirección IP a través de un servidor DHCP, ESP-NOW utiliza una arquitectura sin conexión basada en tramas de acción rápida del estándar IEEE 802.11.

Direccionamiento por MAC: Los dispositivos se vinculan y reconocen mutuamente en el código de programa utilizando la dirección física fija de su tarjeta de red o dirección MAC (Media Access Control).

Velocidad de Respuesta: Al eliminar las capas superiores del modelo OSI que ralentizan la conexión, el tiempo que transcurre desde que el transmisor envía un comando de apertura hasta que el receptor lo procesa disminuye de los típicos 3 a 5 segundos de una red Wi-Fi tradicional a un rango de 2 a 5 milisegundos.

1.4.4. Servidor en la Nube e Internet de las Cosas (Blynk IoT).

El funcionamiento de Blynk es fundamentado en un modelo de arquitectura cliente-servidor que se compone por 3 elementos esenciales que son la aplicación móvil que sirve como interfaz del usuario, el Blynk Cloud Server (servidor en la nube que enruta los datos) y el hardware embebido (ESP32).

Para la seguridad que ofrece por medio de un token de autenticación puesto que la comunicación entre el ESP32 y la nube se cifra y valida de forma unívoca mediante un *Auth Token* generado por la plataforma (por ejemplo, el token 5ju2QuiOnE5oB44gczrmv5FiXaG_YhN2 configurado en el firmware). Esto evita que dispositivos externos intercepten el canal de control.

A través de la función Blynk.run(), el firmware realiza solicitudes periódicas al servidor para verificar si el usuario interactuó con la aplicación. Se dejó apartada esta función dentro de una condición lógica de monitoreo que se ejecuta en segundo plano, impidiendo que el microcontrolador detenga sus funciones críticas (en el movimiento general del servomotor) si la señal de internet llega a perderse temporalmente.

Mecanismo de pines virtuales.

Para la parte más relevante para implementación de control a través de pines virtuales (identificados en el desarrollo como V32). A diferencia de los pines físicos del ESP32 (GPIO), los pines virtuales son canales de datos abstractos basados en software que se ejecutan sobre la pila de red TCP/IP.

Abstracción de Hardware: El pin virtual V32 actúa como una memoria intermedia en el servidor de la nube. Cuando el usuario presiona el botón digital en su teléfono, se envía un evento que activa la macro BLYNK_WRITE(V32) en el ESP32.

Ventaja: El uso de pines virtuales evita conflictos físicos de hardware en el microcontrolador. Permite disparar la función analógica y temporal ejecutarAcceso() sin necesidad de cablear interruptores físicos adicionales y sin interferir con las señales de los buses SPI del RFID o I2C de la pantalla OLED, maximizando la eficiencia en la distribución de la memoria y los puertos del SoC.

1.4.5. Tarjeta Identificación por Radiofrecuencia (RFID) como control de acceso.

La transferencia de datos como identificación sin contacto es uno de los más usados para sistemas de control de acceso.

En esta parte se realizó la tecnología RFID un método de almacenamiento y recuperación de datos remoto que emplea ondas electromagnéticas para transmitir la identidad de un objeto de forma inalámbrica y sin requerir línea de vista directa entre los componentes.

- El Transceptor (Lector/Escritor): Representado en el prototipo por el circuito integrado MFRC522, encargado de emitir energía y procesar las señales de respuesta.
- El Transpondedor (Tarjeta o Tag): El dispositivo pasivo que porta los datos de identidad únicos del usuario.

Generación del Campo Electromagnético: El lector RC522, a través de su antena impresa en el circuito, emite constantemente una señal de radiofrecuencia de 13.56 MHz que genera un campo magnético de corto alcance (típicamente de 0 a 5 cm).

Inducción de Voltaje (Ley de Faraday): Cuando el usuario aproxima la tarjeta al lector, el campo magnético atraviesa la antena interna de la tarjeta (una bobina de espiras de cobre). Por la ley de inducción de Faraday, se induce una fuerza electromotriz (voltaje) entre las terminales de esta bobina.

Alimentación del Microchip: El voltaje inducido es rectificado y regulado por los circuitos internos de la tarjeta. Una vez que el voltaje alcanza el umbral de operación (aproximadamente 3.3V), el microchip embebido en el plástico "despierta" y se energiza.

Modulación de Carga (Transmisión de Datos): Para responder, el microchip de la tarjeta altera la carga eléctrica de su propia antena en función de los bits binarios que desea transmitir (su código de identidad). Esta variación de carga perturba armónicamente el campo magnético del lector, un fenómeno conocido como

modulación de carga, el cual es detectado y demodulado por el RC522 para recuperar la información digital.

El Identificador Único (UID)

El bloque más crítico para la seguridad y el funcionamiento del firmware desarrollado es el Bloque 0 del Sector 0, denominado técnicamente como el Bloque del Fabricante.

En este bloque se almacenan los primeros 4 bytes correspondientes al UID o Identificador Único de la tarjeta.

Restricción de Hardware: Este bloque es grabado en la fábrica de silicio y se configura con atributos de "solo lectura" mediante hardware. No puede ser modificado por software convencional, lo que garantiza que el UID funcione como una firma digital infalsificable para el control de acceso en la chapa.

1.4.6. Bluetooth de Bajo Consumo (BLE - Bluetooth Low Energy)

El ESP32 integra un transceptor de radiofrecuencia compatible con el estándar Bluetooth v4.2, el cual contiene una tecnología Bluetooth Low Energy (BLE) o Bluetooth de bajo consumo. El BLE fue diseñado específicamente para el internet de las cosas (IoT) priorizando la transferencia de ráfagas cortas de información con una demanda energética muy reducida.

Para optimizar la autonomía del dispositivo, la pila de protocolos BLE del ESP32 gestiona el ciclo de vida de la conexión mediante una máquina de estados optimizada:

- Modo de Anuncio (Advertising Mode): Al iniciar el sistema, el ESP32 no establece un enlace directo; en su lugar, transmite periódicamente paquetes de anuncio (Advertising Packets) al espacio libre en los canales de radiofrecuencia 37, 38 y 39. Estos paquetes contienen el nombre del dispositivo (configurado en el firmware como "ESP32_SERVO_LLAVE") y su UUID. Durante este estado, el consumo del chip es intermitente y reducido.

- Modo de Conexión (Connected Mode): Una vez que el cliente vincula el dispositivo, la frecuencia de transmisión se sincroniza mediante un parámetro conocido como Connection Interval. Al finalizar la transferencia del comando de apertura, el enlace puede suspenderse inmediatamente para retornar al estado de reposo.

Para el flujo de información desde el hardware hasta el firmware, la pila de protocolos de BLE se divide en tres capas principales:

1. El Controlador (Controller): Es la capa física (hardware) que gestiona el transceptor de radiofrecuencia de 2.4 GHz, la modulación de la señal y los paquetes de bits que viajan por el aire.
2. El Anfitrión (Host): Capa lógica superior encargada del enrutamiento. Aquí residen el protocolo L2CAP (encargado de la multiplexación de datos) y el SMP (Manager de Seguridad).
3. La Aplicación: El código de usuario desarrollado en el IDE de Arduino que interactúa directamente con los servicios para ejecutar la apertura de la chapa.

Para tener un mejor entendimiento sobre la comparativa de el bluetooth clásico y el que se implemente en el microcontrolador ESP32 se muestra en la siguiente tabla:

Parámetro Técnico	Bluetooth Clásico (SPP)	Bluetooth Low Energy (BLE)	Ventaja para la Chapa Electrónica
Consumo en Transmisión	30mA a 40mA	**15mA a 20mA**	Duplica la eficiencia energética del SoC.
Tiempo de Conexión	100ms a 300ms	**3ms a 6ms**	Autenticación y apertura casi instantáneas.
Estructura de Datos	Flujo continuo de bytes (Stream)	**Atributos estructurados (GATT)**	Permite mapear funciones específicas a variables de software de forma ordenada.
Soporte de Dispositivos	Limitado en iOS (requiere certificación MFi)	**Universal (Android e iOS nativo)**	Hace al sistema multiplataforma sin modificar el hardware.

Tabla 3 Comparativa de un Bluetooth Clásico vs. BLE.

1.5. Etapa de Acoplamiento y Potencia.

El optoacoplador, también denominado optoaislador, es un componente que permite la transmisión de una señal eléctrica utilizando luz como medio de transporte, logrando así un aislamiento galvánico entre la etapa de control (ESP32) y la etapa de potencia (chapa eléctrica). Según Boylestad y Nashelsky (2009), este dispositivo es esencial en sistemas donde se requiere proteger los componentes digitales de voltajes transitorios y ruido eléctrico provenientes de cargas inductivas.

1.5.1. Aislamiento de Señal: El Optoacoplador.

En el ámbito de sistemas de control, la combinación de circuitos digitales de muy alta velocidad con etapas de potencia mecánica representa un riesgo crítico de falla de hardware. Para tener un menor riesgo en el sistema de control de acceso, se integró un optoacoplador como una primera línea de defensa entre los pines de salida del microcontrolador y la etapa de conmutación del transistor.

El optoacoplador actúa como un repetidor de señal unidireccional que transfiere un pulso lógico digital utilizando luz como medio de transmisión, eliminando la necesidad de una conexión eléctrica directa entre la zona de control y la zona de potencia.

1.5.2. Concepto de aislamiento.

Para el aislamiento esto significa separar físicamente dos circuitos para que no compartan una ruta de retorno de corriente común (tierra) ni líneas de alimentación directa.

Unas de las razones por el cual el aislamiento es obligatorio son las siguientes:

Protección contra tensiones inversas, el servomotor es una carga inductiva que jala un mecanismo mecánico. Al arrancar o frenar bruscamente, el motor genera picos de voltaje inducido inverso (fuerza contraelectromotriz). Si el circuito estuviera acoplado directamente, estos picos viajarían hacia atrás e inyectarían

sobrevoltajes en el pin GPIO 14, destruyendo instantáneamente el canal de la ESP32.

- Supresión de bucles de tierra (Ground Loops): Al activar cargas de alta corriente, la tierra de potencia puede sufrir elevaciones de potencial momentáneas. Separar la tierra digital (GND_ESP32) de la tierra de potencia (GND_POTENCIA) mediante el optoacoplador evita que el ruido eléctrico se propague por la línea de tierra, previniendo reinicios inesperados o congelamientos en el procesador.
- Incompatibilidad de Niveles Lógicos: Permite que la señal de 3.3V del ESP32 excite de forma segura una etapa transistorizada polarizada a 5V o más, funcionando simultáneamente como un traductor de nivel de voltaje sin cargar la línea de control.

Para la corriente del led emisor y encender el LED infrarrojo interno de forma segura a través del GPIO del ESP32, se coloca una resistencia limitadora en serie (Rin). Sabiendo que el pin entrega 3.3V y el LED del optoacoplador tiene una caída de voltaje típica de 1.2V(VF), si deseamos una corriente segura de operación de 10mA.

Para la corriente de salida (Ic) se supone que un optoacoplador estándar con un CTR mínimo del 100%, al inyectar 10mA en la entrada, el fototransistor interno será capaz de entregar un máximo de 10mA en su salida. Esta corriente es más que suficiente para saturar con total firmeza la base del transistor de potencia BC547, garantizando una conmutación rápida y limpia.

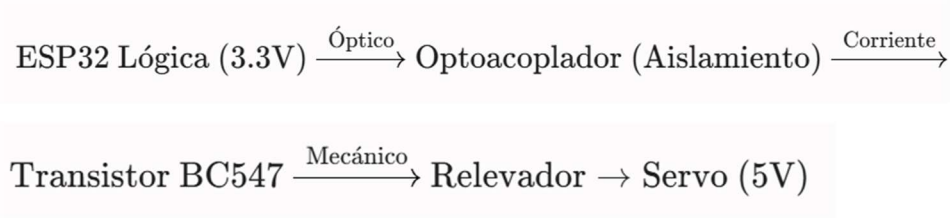


Figura 1. Flujo Operativo de la Etapa de Potencia Completa.

1.6. El relevador: actuación electromecánica de potencia.

El relevador es un dispositivo de conmutación electromecánica que funciona como un interruptor controlado por electricidad. Su función es cerrar el circuito de alimentación de la chapa eléctrica usando 5V cuando recibe una señal lógica del microcontrolador. Según Boylestad y Nashelsky (2009), el relevador permite el control de una carga de alta corriente mediante una corriente de bobina relativamente pequeña.

1.6.1. Constitución y principio de funcionamiento.

Un relevador estándar consta de los siguientes elementos fundamentales:

1. Bobina (Electroimán): Un enrollamiento de hilo de cobre que, al circular una corriente, genera un campo magnético.

2. Armadura: Una pieza móvil de hierro que es atraída por el campo magnético de la bobina.

3. Contactos: Platinos físicos que abren o cierran el circuito de salida. Generalmente se presentan en configuración NC (Normalmente Cerrado) y NA (Normalmente Abierto).

Cuando el ESP32 activa el optoacoplador, este permite el paso de corriente hacia la bobina del relevador. El flujo magnético resultante atrae la armadura, cambiando la posición de los contactos y permitiendo que la energía fluya hacia la chapa eléctrica. Al cesar la corriente, un resorte interno devuelve los contactos a su posición original (Malvino y Bates, 2016).

Para el control de una chapa electrónica, se utiliza normalmente el contacto NA (Normalmente Abierto). Esto garantiza que, en caso de una falla total de energía en el sistema de control, la puerta permanezca cerrada.

Modelo de Relevador	**Voltaje de Bobina**	**Configuración de Contactos**	**Capacidad de Carga Máxima**	**Resistencia de la Bobina (Aprox.)**	**Tipo de Encapsulado / Montaje**
RAS-0510	5V CC	SPDT (1 Form C)	10A a 120V CA / 24V CD	70 ohms	Sellado (Through-Hole)
SRD-05VDC-SL-C	5V CC	SPDT (1 Form C)	10A a 250V CA / 30V CD	70 ohms	Sellado (Cubo azul clásico)
HF32F-G/005-HS	5V CC	SPDT (1 Form C)	10ª a 250V CA / 30V CD	125 ohms	Sellado de alta densidad
G5LE-1-DC5	5V CC	SPDT (1 Form C)	10A a 120V CA / 30V CD	63ohm	Grado Industrial Sellado

Tabla 4. Tabla de posibles sustitutos de relevadores y sus características.

1.7. Etapa de Amplificación de Corriente y Conmutación: El Transistor Bipolar de Juntura (BJT).

Los puertos de salida de propósito general (GPIO) de los microcontroladores modernos basados en tecnologías de silicio CMOS de alta densidad presentan limitaciones físicas estrictas respecto a la entrega de corriente eléctrica. Para el caso específico de la ESP32, los fabricantes estipulan una capacidad de corriente máxima recomendada que oscila entre los 12mA y 40mA por pin, operando a un nivel lógico de 3.3V. Debido a que los actuadores mecánicos y los elementos de conmutación electromecánica (como la bobina de un relevador) demandan corrientes operativas significativamente mayores, se vuelve indispensable el uso de una etapa intermedia de acoplamiento y amplificación. Para este propósito, se implementó el transistor bipolar de juntura (BJT, por sus siglas en inglés: Bipolar Junction Transistor) BC547, un dispositivo semiconductor de tres terminales (Colector, Base y Emisor) de estructura tipo NPN.

1.7.1. Principio de Operación en Conmutación (Corte y Saturación)

A diferencia de las aplicaciones analógicas de amplificación lineal (donde el dispositivo opera en la región activa para amplificar señales variables en el tiempo), en los entornos digitales y de potencia el BJT se configura para trabajar exclusivamente en sus dos regiones extremas de la curva de operación. Esta técnica permite emular el comportamiento de un interruptor electrónico ideal controlado por corriente.

- Región de Corte (Interruptor Abierto)

Esta condición se establece cuando el pin de control del microcontrolador entrega un nivel lógico bajo (LOW=0V) Al no superarse la tensión umbral de la juntura base-emisor (VBE < 0.7V), la corriente de base (IB) se anula por completo. De acuerdo con Malvino y Bates (2021), en este estado el transistor interrumpe el flujo de electrones a través del canal principal, reduciendo la corriente de colector a un valor despreciable de fuga (IC=0). El circuito de salida se comporta como un lazo abierto, manteniendo desenergizada la bobina del relevador.

- Región de Saturación (Interruptor Cerrado)

Se presenta cuando el microcontrolador conmuta a un nivel lógico alto (HIGH=3.3V). Esta tensión polariza la juntura base-emisor en directo (VBE= 0.7V), inyectando una corriente de base (IB) que provoca un flujo máximo de portadores de carga desde el emisor hacia el colector. En esta región, la corriente de colector (IC) queda limitada únicamente por la impedancia de la carga externa conectada y la caída de tensión interna entre colector y emisor decae a su valor mínimo nominal, denominado voltaje de saturación colector-emisor (VCE(sat)=0.1V a 0.3V) (Boylestad & Nashelsky, 2018). El transistor actúa como un interruptor hacia la referencia de tierra (GND), energizando plenamente la carga de potencia.

La configuración como conmutador de lado bajo (*Low-Side Switch*) ofrece las siguientes ventajas técnicas ante las etapas requeridas en el circuito.

- Estabilidad del voltaje de umbral: Al fijar el emisor permanentemente a 0V, la tensión requerida para activar el semiconductor (VBE) se mantiene constante en 0.7V, independientemente de las fluctuaciones de corriente o de la caída de voltaje que experimente la carga en el colector.
- Compatibilidad de niveles de voltaje: Permite utilizar una señal de control digital de baja tensión proveniente de la lógica del microprocesador para gobernar el encendido y apagado de una red de potencia operada a voltajes sustancialmente mayores, logrando una interfaz de acoplamiento eficiente y de bajo costo.

Para garantizar que el transistor BC547 actúe de forma segura como un interruptor electrónico (saturación), permitiendo el paso de corriente hacia el relevador sin forzar al microcontrolador ESP32, es indispensable colocar una resistencia en su pin de base (RB).El objetivo de esta resistencia es limitar la corriente que sale del pin del ESP32 para no dañarlo y, al mismo tiempo, entregarle al transistor la corriente suficiente para que se active por completo.

Voltaje de salida del ESP32 (VCC): 3.3V (Voltaje del pin digital en estado ALTO).

Voltaje Base-Emisor del transistor (VBE): 0.7V (Voltaje necesario para que el silicio del transistor empiece a conducir).

Corriente de la carga (IC): aproximadamente de 50mA, que es la corriente aproximada que consume la bobina del relevador de 5V para activarse.

Ganancia de corriente mínima (β o hFE): 110 (Valor mínimo garantizado por el fabricante para el BC547 en condiciones de conmutación).

Calcular la corriente teórica necesaria en la base (IB): Usamos la fórmula fundamental del transistor para saber cuánta corriente de control necesitamos:

$$IB = \frac{Ic}{hFE}$$

$$IB = \frac{50mA}{110} = 0.454mA$$

Aplicar el factor de saturación.

para asegurar que el transistor se cierre por completo y no trabaje en la zona lineal (lo que causaría que se caliente o no active bien el relevador), se aplica un factor de sobreconducción (generalmente de 3 a 5). Se multiplicó la corriente por un factor para asegurar la saturación:

$$IB(sat) = IB * 4 = 0.454mA * \ 4 = 1.81mA$$

Esta corriente de 1.81mA es bajísima y totalmente segura para el pin del ESP32, el cual puede entregar hasta 12mA sin problemas.

Calcular el valor de la resistencia usando la Ley de Ohm

El voltaje que cae en la resistencia es el voltaje del ESP32 (3.3V) menos la caída interna del transistor (0.7V):

$$RB = \frac{VCC - VBE}{IB(sat)}$$

$$RB\frac{3.3V - 0.7V}{0.00181A} = \frac{2.6V}{0.00181A} = 1436\Omega$$

El cálculo teórico nos indica que cualquier resistencia menor o igual a 1436Ω asegurará la saturación del transistor de forma correcta. Al seleccionar comercialmente una resistencia de 510Ω, aumentamos la corriente de base a:

$$IB = \frac{2.6V}{510\Omega} = 5.09mA$$

Asegura que el transistor se comporte como un interruptor cerrado perfecto, reduciendo al mínimo la pérdida de voltaje interno entre el Colector y el Emisor, entregándole los 5V limpios y completos a la bobina del relevador.

Se mantiene muy por debajo del límite máximo de corriente por pin del microcontrolador 12mA, evitando cualquier desgaste o daño en el silicio del procesador.

Capítulo 2.

Programación y configuración del software.

Programación: software de programación utilizado para el ESP32.

El Arduino IDE (Integrated Development Enviroment) es un software de programación de código abierto puesto que se usa para escribir, cargar el código y depurar programas en múltiples placas de desarrollo que en este caso es del ESP32. Es usado por su versatilidad de uso y compatibilidad con diversas placas.

El programa usado se usa un código de lenguaje que se basa en C/C++, se gestionan diversas librerías y herramientas necesarias para su programación de dichos microcontroladores. Algunas de sus características principales son:

- Interfaz simple y amigable.
- Verificación de errores en el código antes de cargarlo.
- Comunicación con las placas mediante un puerto USB.
- Monitor serie para recibir, enviar y visualizar datos durante la ejecución del programa.

2.1. Instalación de software.

Para su instalación se usó el ejecutable de windows, para proceder a la instalación del software se puede descargar en la pagina oficial de Arduino https://www.arduino.cc/en/software/.

2.1.1. Configuración de Arduino IDE.

Al momento de haber instalado el software se procede a hacer la configuración del programa empezando por agregar la placa ESP32 en el apartado de BOARDS MANAGER la librería que dice by Espressif Systems con una versión anteriores en este caso usando la de 3.3.7 como se muestra en la figura 2.

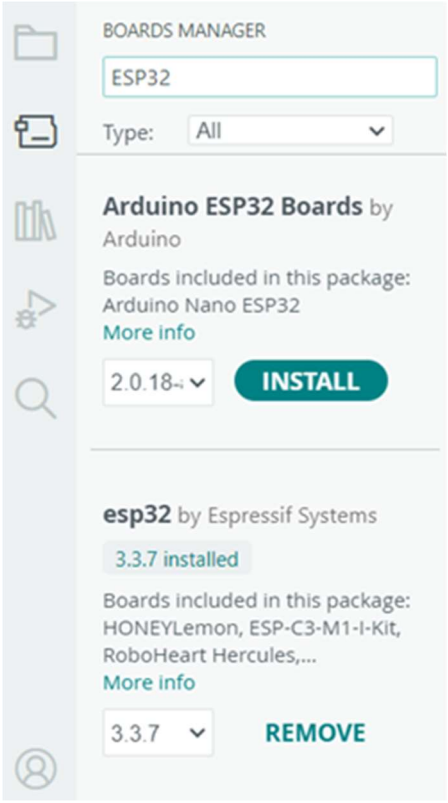


Figura 2. BOARDS MANAGER.

A continuación, se procede a configurar el puerto COM usado, entrando a administrador de dispositivos del sistema operativo Windows 11 para ver que puerto COM está conectado la placa de desarrollo ESP32 como se muestra en la figura 3.

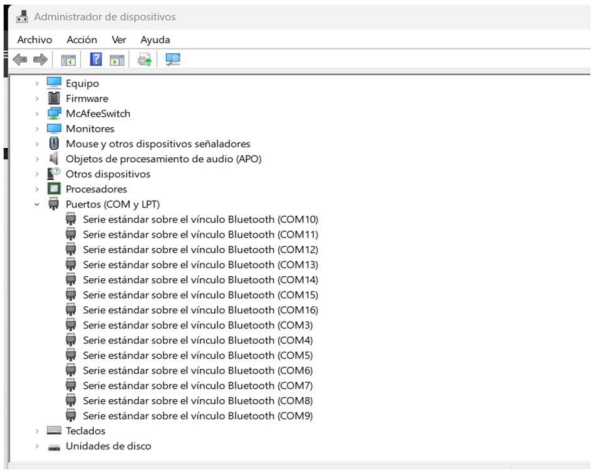


Figura 3. Administrador de dispositivos de windows 11.

Cuando ya se tiene conocimiento del puerto COM asignado a la placa ESP32 se procede a configurarlo dentro de Arduino IDE en en apartado de tools > Port: (Aquí se pone el puerto COM) comos se muestra en la figura 4.

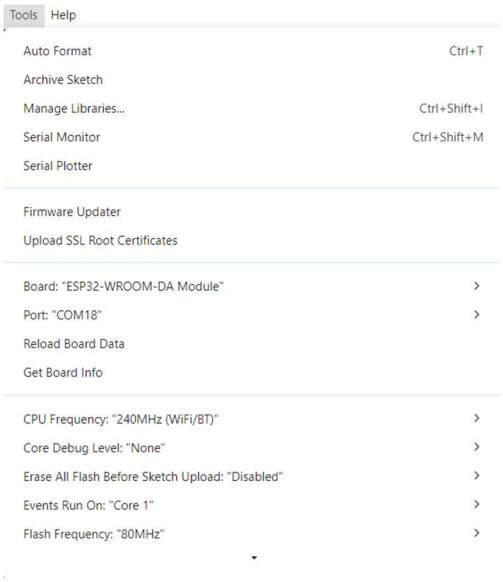


Figura 4. Tools de Arduino IDE.

Se debe de poner específicamente el modelo del ES32 que se está usando puesto que hay varias versiones que son comerciales y para tener una recopilación del programa correcto, en este caso se usó la versión de la placa llamada: ESP32 WROOM DA Module como se muestra en la figura 5.

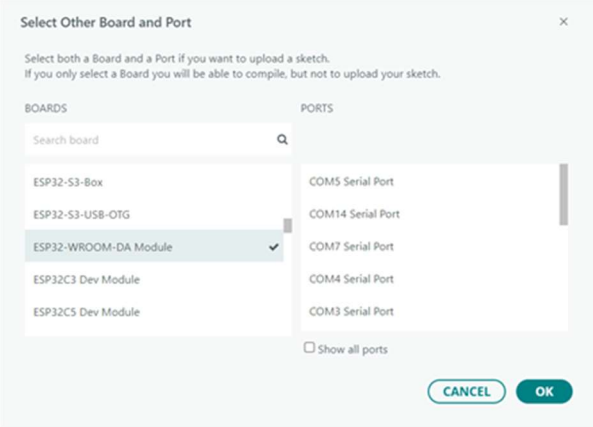


Figura 5. Selección de versión de la ESP32.

2.2. Código del programa.

Para el código del programa se va a implementar el siguiente mecanismo controlado por varios accesos de apertura.

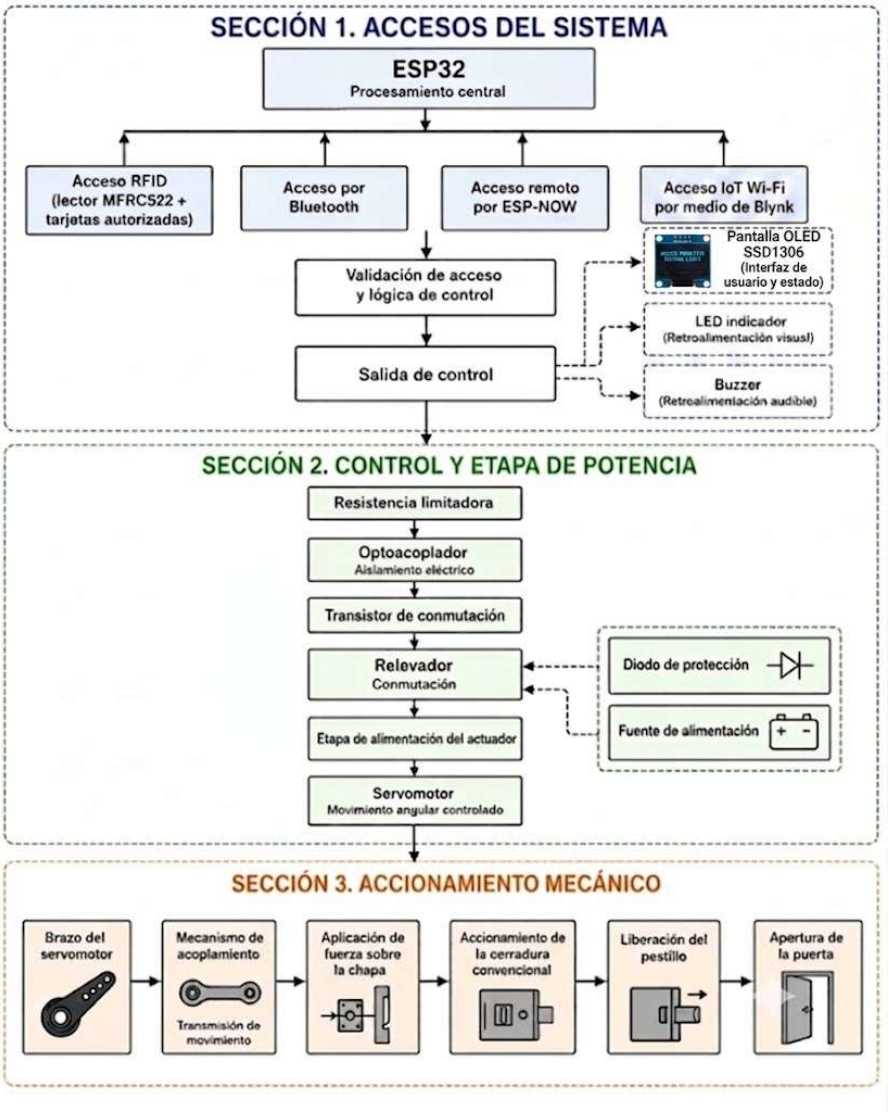


Figura 6. Diagrama de flujo.

Código del control de la chapa electrónica.

```
#define BLYNK_TEMPLATE_ID "TMPL2ZtnIuvxu"
#define BLYNK_TEMPLATE_NAME "llave"
#define BLYNK_AUTH_TOKEN "e3szWadpKk2XS8xqz4RvH1aWZWQRRv5O"

#include <SPI.h>
#include <MFRC522.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <esp_now.h>
#include <esp_wifi.h>
#include <ESP32Servo.h>
#include <BlynkSimpleEsp32.h>
#include "BluetoothSerial.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "time.h"

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
#define SCREEN_ADDRESS 0x3C

#define I2C_SDA_PIN   21
#define I2C_SCL_PIN   17

TwoWire I2COLED = TwoWire(0);
```

```
Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT, &I2COLED,
OLED_RESET);

const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = -21600;
const int   daylightOffset_sec = 0;

struct RegistroAcceso {
  String hora;
  String origen;
};

RegistroAcceso historial[3] = {
  {"--:--:--", "VACIO"},
  {"--:--:--", "VACIO"},
  {"--:--:--", "VACIO"}
};
int indiceHistorial = 0;

char network[]  = "POCO X3 NFC";
char password[] = "12345678";

#define ESPNOW_CHANNEL 6

#define SS_PIN         5
#define RST_PIN        22
#define LED_ROJO_RFID  25
#define PIN_COMPARTIDO  32
#define SERVO_PIN       4
```

```
#define BUZZER_PIN      33

#define BUZZER_CHANNEL  0
const int RELAY_ACTIVO = LOW;
const int RELAY_REPOSO = HIGH;

const int OPEN_ANGLE = 50;
const int BASE_ANGLE = 180;
const int BASE_STEP_DELAY_MS = 3;
const int BASE_MOVE_TIME_MS = BASE_ANGLE * BASE_STEP_DELAY_MS;

MFRC522 rfid(SS_PIN, RST_PIN);
Servo miServo;
BluetoothSerial SerialBT;

bool ledIsOn = false;
bool servoEnMovimiento = false;
bool enPeriodoDeEspera = false;
bool bluetoothActivo = false;
unsigned long tiempoInicioEspera = 0;
const unsigned long tiempoEspera = 3000;

unsigned long ultimoCampioPantallaEspera = 0;
const unsigned long intervaloIntercalado = 5000;
bool mostrarLogoUACM = true;

volatile bool comandoEspNowPendiente = false;
volatile uint8_t comandoEspNow = 255;
```

```
byte tarjetaPermitida1[4] = {0x13, 0x23, 0x4E, 0x1C};
byte tarjetaPermitida2[4] = {0x82, 0x56, 0xA9, 0x1C};

unsigned long lastWifiAttempt = 0;
unsigned long lastBlynkAttempt = 0;
unsigned long lastBlynkRun = 0;

const unsigned long WIFI_RETRY_MS = 20000;
const unsigned long BLYNK_RETRY_MS = 15000;
const unsigned long BLYNK_RUN_MS = 50;

void atenderEspNow();
void atenderBluetooth();
void atenderRFID();
void maintainWiFiAndBlynk();
void diagnosticoWiFiInicial();
void imprimirCanalReal();
bool verificarUID(byte *uid, byte longitud);
void ejecutarSecuenciaServoYLed(String origen);
void beepPermitido();
void beepDenegado();
void mostrarMensajeOLED(String titulo, String msg1, String msg2, uint16_t
delayMs = 0);
void gestionarPantallaEspera();
void mostrarEsperaUACM();
void mostrarEsperaHistorial();
void agregarAlHistorial(String origen);
String obtenerHoraActual();
```

```
void dibujarCandadoVerticalIzquierda();

void onEspNowRecv(const uint8_t *mac, const uint8_t *data, int len) {
  if (len >= 1) {
    comandoEspNow = data[0];
    comandoEspNowPendiente = true;
  }
}

BLYNK_WRITE(V32) {
  int valor = param.asInt();
  if (valor == 1) {
    if (!servoEnMovimiento && !enPeriodoDeEspera) {
      ejecutarSecuenciaServoYLed("WIFI");
    }
  }
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  I2COLED.begin(I2C_SDA_PIN, I2C_SCL_PIN, 100000);

  if(!oled.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("Error: SSD1306 no detectado."));
  } else {
    oled.clearDisplay();
```

```
  oled.setRotation(3);
  oled.display();
}

mostrarMensajeOLED("INICIO", "SISTEMA", "COMPLETO", 1000);

pinMode(LED_ROJO_RFID, OUTPUT);
pinMode(PIN_COMPARTIDO, OUTPUT);
digitalWrite(LED_ROJO_RFID, LOW);
digitalWrite(PIN_COMPARTIDO, RELAY_REPOSO);
ledIsOn = false;

ledcSetup(BUZZER_CHANNEL, 2000, 8);
pinMode(BUZZER_PIN, OUTPUT);
digitalWrite(BUZZER_PIN, LOW);

miServo.write(0);
SPI.begin();
rfid.PCD_Init();

bluetoothActivo = SerialBT.begin("ESP32_SERVO_LLAVE");

WiFi.mode(WIFI_STA);
WiFi.setSleep(true);
WiFi.begin(network, password);
lastWifiAttempt = millis();

diagnosticoWiFiInicial();
```

```
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

  esp_wifi_set_promiscuous(true);
  esp_wifi_set_channel(ESPNOW_CHANNEL, WIFI_SECOND_CHAN_NONE);
  esp_wifi_set_promiscuous(false);

  imprimirCanalReal();
  Blynk.config(BLYNK_AUTH_TOKEN);

  if (esp_now_init() == ESP_OK) {
    esp_now_register_recv_cb(onEspNowRecv);
  }
}

void loop() {
  atenderEspNow();
  if (enPeriodoDeEspera && millis() - tiempoInicioEspera >= tiempoEspera) {
    enPeriodoDeEspera = false;
  }
  atenderRFID();
  atenderBluetooth();
  digitalWrite(PIN_COMPARTIDO, ledIsOn ? RELAY_ACTIVO :
RELAY_REPOSO);
  maintainWiFiAndBlynk();
  gestionarPantallaEspera();
}

String obtenerHoraActual() {
  struct tm timeinfo;
```

```
  if(!getLocalTime(&timeinfo)){ return "--:--:--"; }
  char horaFormateada[9];
  strftime(horaFormateada, sizeof(horaFormateada), "%H:%M:%S", &timeinfo);
  return String(horaFormateada);
}

void agregarAlHistorial(String origen) {
  historial[indiceHistorial].hora = obtenerHoraActual();
  historial[indiceHistorial].origen = origen;
  indiceHistorial = (indiceHistorial + 1) % 3;
}

void gestionarPantallaEspera() {
  if (servoEnMovimiento || enPeriodoDeEspera) return;
  if (millis() - ultimoCampioPantallaEspera >= intervaloIntercalado) {
    ultimoCampioPantallaEspera = millis();
    mostrarLogoUACM = !mostrarLogoUACM;
  }
  if (mostrarLogoUACM) { mostrarEsperaUACM(); } else {
mostrarEsperaHistorial(); }
}

void mostrarEsperaUACM() {
  oled.clearDisplay();
  oled.setTextSize(2);
  oled.setTextColor(SSD1306_WHITE);
  oled.setCursor(8, 30);
  oled.print(F("UACM"));
  oled.drawFastHLine(4, 65, 56, SSD1306_WHITE);
```

```
  oled.setTextSize(1);
  if (WiFi.status() == WL_CONNECTED) {
    oled.setCursor(14, 80); oled.print(F("SISTEMA"));
    oled.setCursor(14, 94); oled.print(F("ESPERA"));
  } else {
    oled.fillRect(2, 78, 60, 26, SSD1306_WHITE);
    oled.setTextColor(SSD1306_BLACK);
    oled.setCursor(11, 83); oled.print(F("OFFLINE"));
    oled.setCursor(11, 94); oled.print(F("NO-NET"));
  }
  oled.display();
}

void mostrarEsperaHistorial() {
  oled.clearDisplay();
  oled.fillRect(0, 0, 64, 14, SSD1306_WHITE);
  oled.setTextSize(1); oled.setTextColor(SSD1306_BLACK);
  oled.setCursor(8, 3); oled.print(F("ACCESOS"));
  oled.setTextColor(SSD1306_WHITE);
  int r1 = (indiceHistorial + 2) % 3;
  int r2 = (indiceHistorial + 1) % 3;
  int r3 = indiceHistorial;
  oled.setCursor(0, 22); oled.print("1." + historial[r1].origen);
  oled.setCursor(8, 32); oled.print(historial[r1].hora);
  oled.setCursor(0, 48); oled.print("2." + historial[r2].origen);
  oled.setCursor(8, 58); oled.print(historial[r2].hora);
  oled.setCursor(0, 74); oled.print("3." + historial[r3].origen);
  oled.setCursor(8, 84); oled.print(historial[r3].hora);
```

```
  oled.display();
}

void diagnosticoWiFiInicial() {
  unsigned long inicioWiFi = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - inicioWiFi < 5000) {
delay(500); }
  if (WiFi.status() == WL_CONNECTED) {
    mostrarMensajeOLED("WIFI", "OK", "ONLINE", 1200);
  } else {
    mostrarMensajeOLED("WIFI", "FALLO", "OFFLINE", 1200);
  }
}

void imprimirCanalReal() {
  uint8_t canalPrimario;
  wifi_second_chan_t canalSecundario;
  esp_wifi_get_channel(&canalPrimario, &canalSecundario);
  Serial.print(F("Canal real del receptor: "));
  Serial.println(canalPrimario);
}

void atenderEspNow() {
  if (comandoEspNowPendiente) {
    comandoEspNowPendiente = false;
    uint8_t cmd = comandoEspNow;
    if (cmd == 1 && !servoEnMovimiento && !enPeriodoDeEspera) {
      ejecutarSecuenciaServoYLed("CTRL REM");
    }
```

```
  }
}

void atenderBluetooth() {
  if (!bluetoothActivo) return;
  if (SerialBT.available()) {
    char c = SerialBT.read();
    if ((c == 'H' || c == 'h') && !servoEnMovimiento && !enPeriodoDeEspera) {
      ejecutarSecuenciaServoYLed("BT");
    }
  }
}

void atenderRFID() {
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
    bool uidValido = verificarUID(rfid.uid.uidByte, rfid.uid.size);
    if (uidValido && !servoEnMovimiento && !enPeriodoDeEspera) {
      ejecutarSecuenciaServoYLed("TARJETA");
    }
    else if (!uidValido) {
      mostrarMensajeOLED("DENEGADO", "TARJETA", "INVALIDA");
      agregarAlHistorial("RECHAZADO");
      ledIsOn = false;
      digitalWrite(PIN_COMPARTIDO, RELAY_REPOSO);
      digitalWrite(LED_ROJO_RFID, HIGH);
      beepDenegado(); delay(300); digitalWrite(LED_ROJO_RFID, LOW);
    }
    rfid.PICC_HaltA(); rfid.PCD_StopCrypto1();
```

```
  }
}

void maintainWiFiAndBlynk() {
  unsigned long now = millis();
  if (WiFi.status() != WL_CONNECTED) {
    if (now - lastWifiAttempt >= WIFI_RETRY_MS) {
      lastWifiAttempt = now;
      WiFi.begin(network, password);
    }
    return;
  }

  uint8_t canalActual;
  wifi_second_chan_t ch2;
  esp_wifi_get_channel(&canalActual, &ch2);
  if (canalActual != ESPNOW_CHANNEL) {
    esp_wifi_set_promiscuous(true);
    esp_wifi_set_channel(ESPNOW_CHANNEL, WIFI_SECOND_CHAN_NONE);
    esp_wifi_set_promiscuous(false);
  }

  if (!Blynk.connected()) {
    if (now - lastBlynkAttempt >= BLYNK_RETRY_MS) {
      lastBlynkAttempt = now;
      Blynk.connect(1000);
    }
    return;
```

```
  }
  if (now - lastBlynkRun >= BLYNK_RUN_MS) {
    lastBlynkRun = now;
    Blynk.run();
  }
}

bool verificarUID(byte *uid, byte longitud) {
  if (longitud != 4) return false;
  if (memcmp(uid, tarjetaPermitida1, 4) == 0) return true;
  if (memcmp(uid, tarjetaPermitida2, 4) == 0) return true;
  return false;
}

void beepPermitido() {
  ledcAttachPin(BUZZER_PIN, BUZZER_CHANNEL);
  ledcWriteTone(BUZZER_CHANNEL, 2500); delay(80);
  ledcWriteTone(BUZZER_CHANNEL, 0);    delay(50);
  ledcWriteTone(BUZZER_CHANNEL, 2500); delay(80);
  ledcWriteTone(BUZZER_CHANNEL, 0);
  ledcDetachPin(BUZZER_PIN);
  pinMode(BUZZER_PIN, OUTPUT); digitalWrite(BUZZER_PIN, LOW);
}

void beepDenegado() {
  ledcAttachPin(BUZZER_PIN, BUZZER_CHANNEL);
  ledcWriteTone(BUZZER_CHANNEL, 800); delay(500);
ledcWriteTone(BUZZER_CHANNEL, 0);
  ledcDetachPin(BUZZER_PIN);
```

```
  pinMode(BUZZER_PIN, OUTPUT); digitalWrite(BUZZER_PIN, LOW);
}

void mostrarMensajeOLED(String titulo, String msg1, String msg2, uint16_t
delayMs) {
  oled.clearDisplay();
  oled.fillRect(0, 0, 64, 16, SSD1306_WHITE);
  oled.setTextSize(1); oled.setTextColor(SSD1306_BLACK);
  oled.setCursor(2, 4); oled.print(titulo);
  oled.setTextColor(SSD1306_WHITE);
  oled.setCursor(0, 30); oled.println(msg1);
  oled.setCursor(0, 50); oled.println(msg2);
  oled.display();
  if(delayMs > 0) { delay(delayMs); }
}

void dibujarCandadoVerticalIzquierda() {
  oled.fillRect(20, 74, 24, 16, SSD1306_WHITE);
  oled.fillCircle(32, 80, 2, SSD1306_BLACK);
  oled.drawFastVLine(32, 81, 4, SSD1306_BLACK);
  oled.drawCircle(27, 67, 7, SSD1306_WHITE);
  oled.fillRect(15, 68, 25, 7, SSD1306_BLACK);
  oled.drawFastVLine(20, 67, 8, SSD1306_WHITE);
  oled.drawFastVLine(34, 67, 3, SSD1306_WHITE);
}

void ejecutarSecuenciaServoYLed(String origen) {
  servoEnMovimiento = true;
  digitalWrite(LED_ROJO_RFID, LOW);
```

```
  agregarAlHistorial(origen);
  mostrarMensajeOLED("PERMISO", "ORIGEN:", origen);

  ledIsOn = true;
  digitalWrite(PIN_COMPARTIDO, RELAY_ACTIVO);
  beepPermitido();
  delay(400);

  miServo.attach(SERVO_PIN);
  miServo.write(0);
  delay(100);

  int stepDelayMs = BASE_MOVE_TIME_MS / max(1, OPEN_ANGLE);
  if (stepDelayMs < 1) stepDelayMs = 1;

  oled.clearDisplay();
  oled.setTextSize(1); oled.setTextColor(SSD1306_WHITE);
  oled.setCursor(8, 28); oled.print(F("ABRIENDO"));
  dibujarCandadoVerticalIzquierda();
  oled.display();

  for (int ang = 0; ang <= OPEN_ANGLE; ang++) { miServo.write(ang);
delay(stepDelayMs); }
  delay(tiempoEspera);
  for (int ang = OPEN_ANGLE; ang >= 0; ang--) { miServo.write(ang);
delay(stepDelayMs); }
  delay(500);

  miServo.detach(); delay(50);
```

```
  ledIsOn = false;
  digitalWrite(PIN_COMPARTIDO, RELAY_REPOSO);

  enPeriodoDeEspera = true;
  tiempoInicioEspera = millis();
  servoEnMovimiento = false;

  mostrarLogoUACM = true;
  ultimoCampioPantallaEspera = millis();
  mostrarEsperaUACM();
}
```

Código del control remoto vía ESPNOW.

```
#include <WiFi.h>
#include <esp_now.h>
#include <esp_wifi.h>

#define BUTTON_PIN 17
#define LED_PIN    21

#define CANAL_FIJO 6

uint8_t receptorAddress[] = {0x6C, 0xC8, 0x40, 0x93, 0xE7, 0x2C};

const unsigned long TIEMPO_ENTRE_ENVIOS = 3000;
const unsigned long tiempoAntirebote = 200;

unsigned long ultimoEnvio = 0;
unsigned long ultimoRebote = 0;
bool botonAnterior = HIGH;
uint32_t contadorPaquete = 0;

typedef struct __attribute__((packed)) {
  uint8_t comando;
  uint32_t numeroPaquete;
} MensajeEspNow;

void onDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  if (status == ESP_NOW_SEND_SUCCESS) {
    Serial.println(F(" -> [ACK] ¡Recibido por el destino!"));
```

```
  } else {
    Serial.println(F(" -> [FAIL] No se encontró el receptor en este canal"));
  }
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  Serial.println(F("\n========================================"));
  Serial.println(F(" TRANSMISOR: CANAL FIJO 6 INSTANTÁNEO "));
  Serial.println(F("========================================"));

  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);

  WiFi.mode(WIFI_STA);
  WiFi.setSleep(false);

  esp_wifi_set_promiscuous(true);
  esp_wifi_set_channel(CANAL_FIJO, WIFI_SECOND_CHAN_NONE);
  esp_wifi_set_promiscuous(false);

  if (esp_now_init() != ESP_OK) {
    Serial.println(F("Error al iniciar ESP-NOW"));
    return;
  }
```

```
  esp_now_register_send_cb(onDataSent);

  esp_now_peer_info_t peer = {};
  memcpy(peer.peer_addr, receptorAddress, 6);
  peer.channel = CANAL_FIJO;
  peer.encrypt = false;
  peer.ifidx = WIFI_IF_STA;

  if (esp_now_add_peer(&peer) != ESP_OK) {
    Serial.println(F("Error al agregar receptor"));
    return;
  }

  Serial.println(F("Transmisor listo en Canal 6. Esperando pulsación..."));
}

void loop() {
  bool botonActual = digitalRead(BUTTON_PIN);

  if (botonAnterior == HIGH && botonActual == LOW) {
    unsigned long tiempoActual = millis();

    if (tiempoActual - ultimoRebote > tiempoAntirebote) {
      ultimoRebote = tiempoActual;

      if (tiempoActual - ultimoEnvio >= TIEMPO_ENTRE_ENVIOS || ultimoEnvio
== 0) {
```

```
        contadorPaquete++;
        ultimoEnvio = tiempoActual;

        MensajeEspNow msg;
        msg.comando = 1;
        msg.numeroPaquete = contadorPaquete;

        Serial.print(F("Enviando paquete ID: "));
        Serial.println(contadorPaquete);

        esp_now_send(receptorAddress, (uint8_t *)&msg, sizeof(msg));

        digitalWrite(LED_PIN, HIGH);
        delay(100);
        digitalWrite(LED_PIN, LOW);

      } else {
        unsigned long tiempoRestante = (TIEMPO_ENTRE_ENVIOS -
(tiempoActual - ultimoEnvio)) / 1000;
        Serial.print(F("Espera ")); Serial.print(tiempoRestante); Serial.println(F("
seg..."));
      }
    }
  }
  botonAnterior = botonActual;
}
```

2.2.1. Etapas de funcionamiento.

2.2.1.1. librerías.

Las librerías para su correcto funcionamiento con todos los accesos y pantalla que se usaron son las siguientes:

<Wire.h>: Administra el protocolo de comunicación I2C por hardware. Es la encargada de abrir el canal de datos para comunicarse con la pantalla OLED usando solo dos pines (SDA y SCL).

<Adafruit_GFX.h>: Es la librería maestra de gráficos de Adafruit. Proporciona las funciones matemáticas para dibujar figuras geométricas básicas en la pantalla, como las líneas (drawLine), círculos (drawCircle) y rectángulos (drawRect) de tus animaciones.

<Adafruit_SSD1306.h>: Es el controlador específico para el chip de tu pantalla OLED. Traduce los gráficos de la librería anterior en encendido y apagado de píxeles reales dentro de la pantalla de 28x64.

<SPI.h>: Administra el protocolo de comunicación serial de alta velocidad SPI. Es indispensable para enviar y recibir ráfagas de datos rápidas hacia el lector de tarjetas RFID.

<MFRC522.h>: Contiene los comandos necesarios para controlar el lector RFID-RC522. Permite encender la antena del sensor, detectar si hay un llavero cerca y leer su código de identificación único.

<WiFi.h>: Enciende y controla el módulo de radiofrecuencia de 2.4 GHz integrado en el ESP32. Permite que el microcontrolador busque y se conecte de forma inalámbrica al módem de tu casa (red local).

<BlynkSimpleEsp32.h>: Conecta el código con los servidores en la nube de Blynk mediante internet. Te permite recibir en tiempo real las órdenes de apertura que mandas desde el botón de la aplicación en el celular y en computadora.

"BluetoothSerial.h": Configura el Bluetooth clásico del ESP32 para que funcione como un puerto serie virtual. Gracias a ella, se puede recibir el carácter 'H' enviado desde cualquier aplicación de terminal Bluetooth en el teléfono.

<ESP32Servo.h>: Controla los temporizadores (*timers*) internos del ESP32 para generar señales PWM (Modulación por Ancho de Pulso) estables. Le dice exactamente a qué ángulo moverse al servomotor sin que vibre.

<esp_now.h>: Activa el protocolo de comunicación inalámbrico directo y exclusivo de Espressif. Te permite enlazar dos módulos ESP32 entre sí para mandarse comandos instantáneos sin necesidad de conectarse a un módem de internet.

2.2.1.2. Cinemática del mecanismo.

// ===== Ajuste de carrera y velocidad del mecanismo =====

const int OPEN_ANGLE = 50;

const int BASE_ANGLE = 180;

const int BASE_STEP_DELAY_MS = 3;

const int BASE_MOVE_TIME_MS = BASE_ANGLE * BASE_STEP_DELAY_MS;
const int OPEN_ANGLE = 200:

Define el límite físico o la carrera útil a la que debe girar el servomotor para retraer por completo el pestillo y liberar la chapa electrónica. Está configurado a 50°. Al ser una constante (const), este valor almacena en la memoria Flash del ESP32 para asegurar que el motor siempre viaje exactamente a la misma posición angular en cada ciclo de acceso permitido, evitando descalibrarse.

2.2.1.3. Lazo de Control No Bloqueante (loop).

- Asincronismo por hardware: Sustituye el uso de retardos convencionales (delay) por un contador de marcas de tiempo basado en la función de reloj interno millis().
- Procesamiento libre: Permite que las animaciones visuales de la pantalla OLED y los efectos de pulso se refresquen en segundo plano sin congelar el microcontrolador.
- Escaneo en paralelo: Asegura que el ESP32 tenga ciclos de reloj libres para leer continuamente el sensor RFID y procesar llamadas de Blynk o Bluetooth al mismo instante.
- Evita congelamientos: Impide pérdidas de conexión con los servidores de la nube IoT al no bloquear nunca el flujo principal del programa.

2.2.1.4. Interfaz visual.

- Gestión de buses: Integra las librerías Wire.h y SPI.h para coordinar la comunicación síncrona nativa por hardware a través de los canales I2C y SPI del chip.
- Búfer gráfico: Acopla las herramientas de Adafruit (Adafruit_SSD1306.h y Adafruit_GFX.h) para controlar los píxeles de la pantalla y el renderizado geométrico de vectores.
- Sincronización electromecánica: Utiliza ESP32Servo.h para tomar el control de los temporizadores internos (*timers*) del ESP32, generando señales PWM estables y precisas para mover el motor.

2.3. Descarga y configuración de aplicaciones (APK).

2.3.1. Serial Bluetooth Terminal.

Para la conexión vía bluetooth (BLE) se debe de descargar una aplicación que se encuentra en la app store de Android llamada Serial Bluetooth Terminal, ya que se va a encargar de la interfaz de comunicación para la apertura de la chapa por medio de un comando escrito como se muestra en la figura 7.

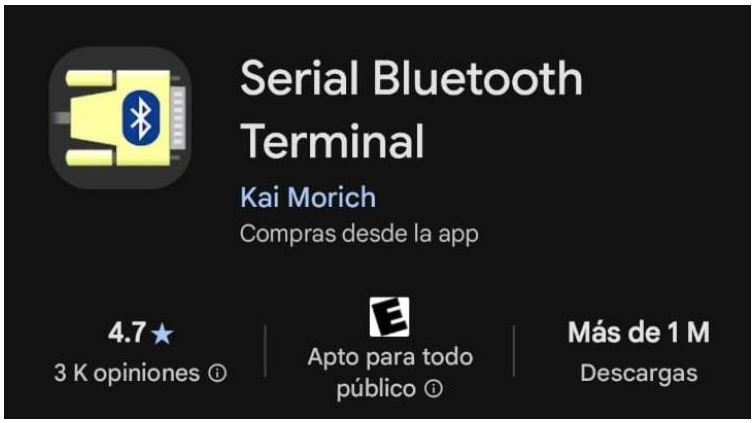


Figura 7. APK en la app store.

Se procede a descargar y a abrir la aplicación, por consiguiente se muestra la primera parte de la figura 8.

Figura 8. Interfaz principal de Serial Bluetooth Terminal.

Para poder tener la conexión primero se debe de conectar a través de la configuración del celular que se está usando como se muestra en la figura 9 y posteriormente conectarlo por medio de la apk en el apartado Devices como se muestra en la figura 10.

Figura 9. Ejemplo de conexión via bluetooth en un dispositivo movil.

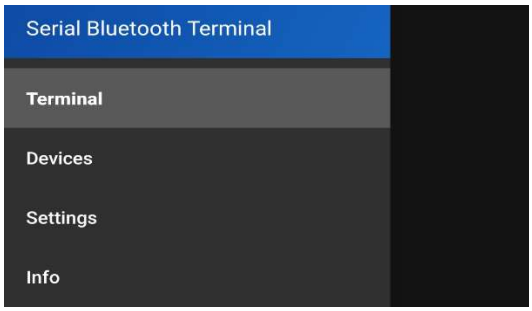


Figura 10. Configuración de bluetooth.

Aquí se encuentran los dispositivos enlazados posteriormente en la configuración del dispositivo para que lo reconozca en este caso el nombre que se le puso en la programacion, en este caso, se puso el nombre de ESP32_CHAPA_TESIS como se muestra en la figura 11 o en nombre que se haya asignado en la programación previamente.

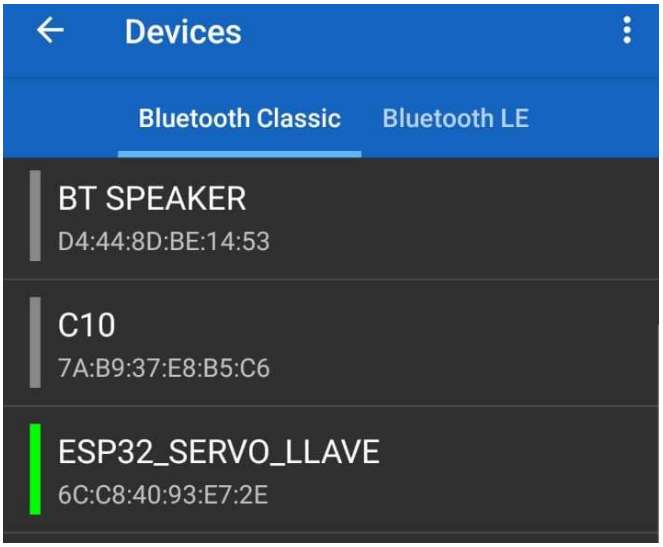


Figura 11. Dispositivos para vincular.

2.3.2. Blynk IoT.

Para la parte de conexión vía internet tenemos una aplicación llamada Blynk IoT que sirve como comunicación con la ESP32 y como interfaz de usuario por medio de botones virtuales.

Se descarga por medio de play store del dispositivo que tenemos con el nombre y logo que se muestra en la figura 12:

Figura 12. Apk Blynk IoT en play store.

Para esta aplicación se creo una cuenta en con un correo de gmail para acceder a la aplicación.

El paso siguiente es crear una plantilla en el apartado que se muestra en la figura 13,donde van a formar parte la interfaz con los botones virtuales y su respectiva configuración.

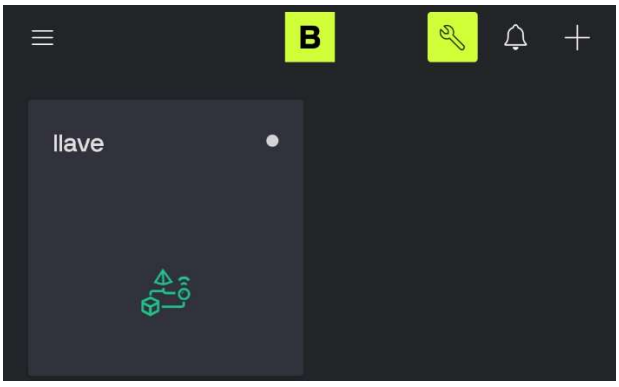


Figura 13. Apartado para crear plantillas.

Al crear la platilla especificando que es para ESP32 se procede a crear el pin virtual que se va a utilizar para mandar la señal de acceso permitido al microcontrolador configurando el boton en la parte de DATA TYPE es importante ponerlo en integer, en el minimo 0 y en el máximo 1, puesto que solo va a mandar un valor virtualmente y además de tenerlo como V32 como se muestra en la figura 14.

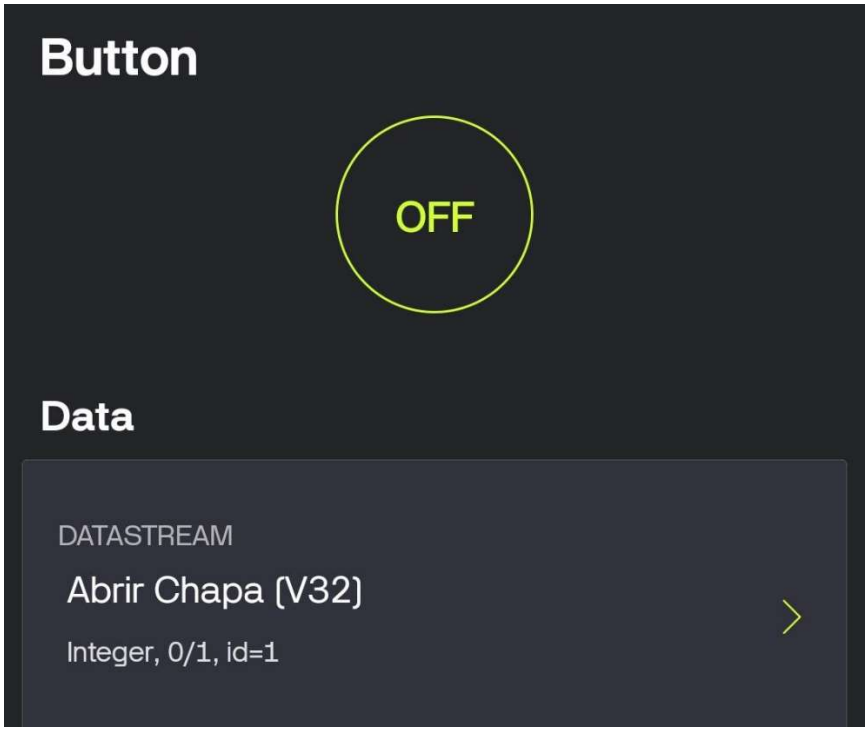


Figura 14. Valores a cambiar para el pin virtual.

En la siguiente parte se debe de tener en el modo Push y ponerle el nombre que se desea para que sea mas amigable en la interfaz como se muestra en la figura 15.

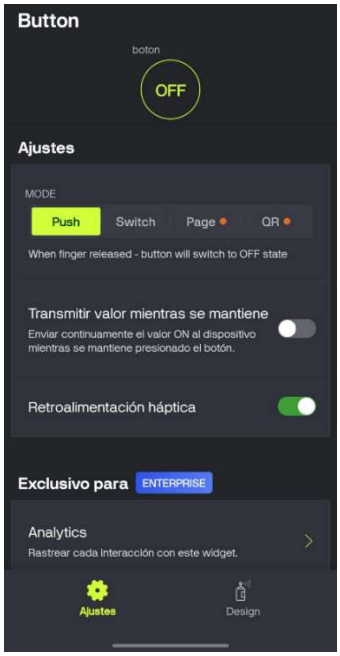


Figura 15. Ajustes del pin virtual.

Capítulo 3.

Diseño de los prototipos.

3.1. Placa PCB.

Durante las primeras fases de diseño, es común y de gran utilidad emplear tablillas de pruebas provisionales (conocidas como protoboards) para conectar los componentes mediante cables sueltos. Si bien este método es excelente para validar que la programación funcione y los componentes respondan, carece por completo de la estabilidad necesaria para que el sistema opere de forma definitiva y segura en el mundo real.

Para tomar la decisión de diseñar y fabricar una placa de circuito impreso son, para tres partes fundamentales de la placa a diseñar.

- Eliminación de falsos contactos: Los cables sueltos y las conexiones a presión de las tablillas de prueba tienden a aflojarse con extrema facilidad ante cualquier movimiento, vibración o cambio de temperatura. Al soldar cada componente directamente sobre una placa sólida, las conexiones se vuelven permanentes y definitivas, garantizando que el sistema no falle inesperadamente por un cable flojo.
- Orden, reducción de espacio y estética: Una cerradura electrónica debe ser un dispositivo compacto y estético para poder integrarse en una puerta o gabinete. La PCB permite reemplazar todo el desorden de cables flotantes por "caminos" de cobre grabados sobre la placa. Esto reduce drásticamente el tamaño total del circuito, permitiendo acomodar el ESP32, la pantalla y los sensores en un espacio sumamente reducido y ordenado.
- Limpieza y orden en la energía: Un servomotor exige corriente para moverse y vencer la fuerza mecánica de la chapa. Si esa energía viaja por cables delgados y largos, se genera una saturación eléctrica que puede confundir al cerebro del sistema (el ESP32) y provocar que se reinicie o se congele. La placa PCB permite diseñar caminos más anchos y exclusivos para la corriente del motor, asegurando que este reciba toda la fuerza

necesaria, sin incomodar ni alterar el funcionamiento digital de las antenas WiFi o Bluetooth.

3.1.1. Creación del diagrama esquemático.

Primordialmente para el desarrollo del hardware de la chapa electrónica radica en la estructura del diagrama esquemático. Este proceso define las interconexiones lógicas para los diferentes pines del microcontrolador y sus periféricos y también por su parte las reglas de aislamiento acoplamiento y flujo de corriente que van a gobernar el comportamiento físico del circuito.

Para la realización de esta etapa, se seleccionó como entorno de desarrollo de software la plataforma EasyEDA (Easy Electronic Design Automation).

Se eligió este entorno de desarrollo por los siguientes criterios.

- Integración de Librerías y Modelado de Componentes: EasyEDA cuenta con una vinculación directa a las bases de datos de LCSC Electronics y JLCPCB. Esto permitió importar de manera exacta los símbolos esquemáticos y los encapsulados físicos (footprints) estandarizados de componentes críticos para el proyecto, tales como el módulo ESP32-WROOM-32E, el transceptor RFID-RC522 y el transistor de potencia BC547, garantizando una coincidencia del 100% entre el plano teórico y los componentes reales.
- Gestión de Redes Estructuradas: En lugar de realizar un enrutamiento visualmente saturado mediante líneas continuas flotantes, el esquema se diseñó utilizando etiquetas de red. Esta técnica de diseño asistido por computadora permite agrupar los buses de datos síncronos (como las líneas MISO, MOSI, SCK y SS del bus SPI para el RFID, y SDA/SCL para el bus I2C de la pantalla OLED) de forma limpia y modular, reduciendo la probabilidad de errores de conexión y facilitando la posterior auditoría del circuito.
- Verificación de Reglas de Diseño Eléctrico: Mediante el motor de análisis automatizado de EasyEDA, se corrieron pruebas de Verificación de Reglas

Eléctricas (ERC, Electrical Rules Check). Esta herramienta analizó el esquema en busca de pines flotantes, salidas en cortocircuito o conflictos de impedancia antes de proceder a la creación de la PCB, asegurando la integridad lógica del sistema.

3.1.2. Resultado del esquemático.

Al tener en cuenta todos los criterios a seguir se tuvo un resultado favorable al hacer las interconexiones combinando la parte de potencia y de control, todos los componentes usados se muestran en la figura 16, con estas conexiones se concluye la parte de elaboración del esquemático.

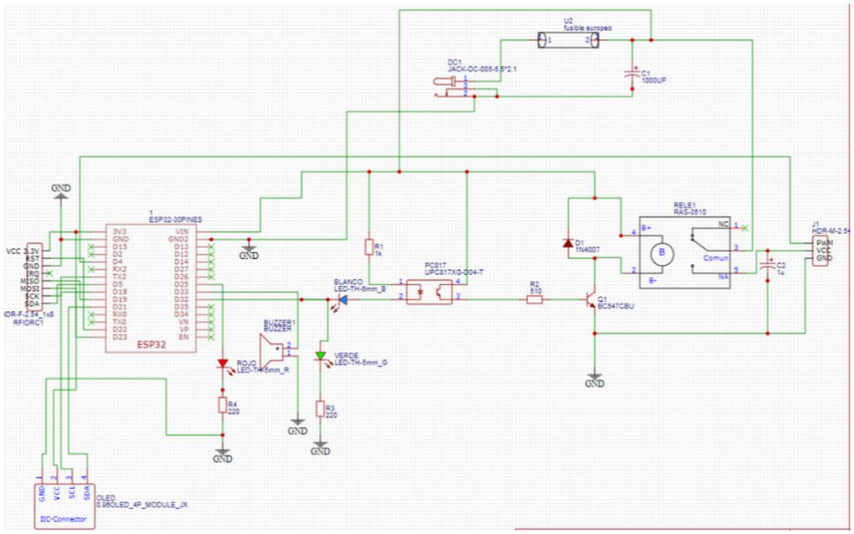


Figura 16. Diagrama esquemático principal en el entorno de desarrollo easyEDA.

Para el control remoto se usó el diagrama esquemático que se muestra en la figura 17.

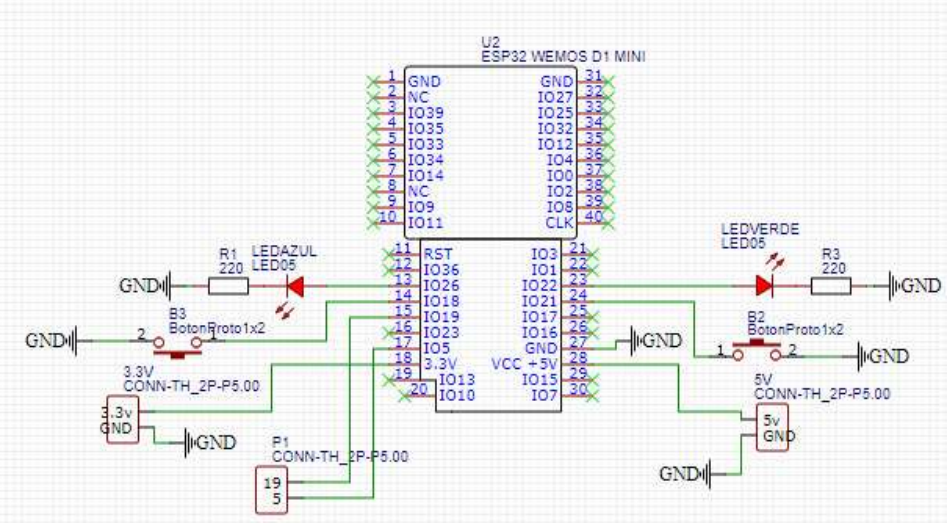


Figura 17. Diagrama esquemático del control remoto en el entorno de desarrollo easyEDA.

Para permitir la apertura de la chapa inteligente a distancia sin necesidad de cables ni conexión a Internet, se diseñó el circuito de un control remoto portátil. En la Figura 16 se muestra el diagrama esquemático de este mando, el cual utiliza el microcontrolador ESP32 (WEMOS D1 MINI) para enviar órdenes inalámbricas directas a la cerradura principal mediante el protocolo de comunicación rápida ESP-NOW.

Botón B3 (Apertura): Está conectado al pin IO26 del ESP32. Al presionarlo, el procesador se activa y envía un paquete de datos inmediato por ESP-NOW hacia la chapa principal indicándole que encienda el motor para retirar el pasador Phillips.

Botón B2 (Alarma): Se encuentra conectado al pin IO17. Este botón se configuró para enviar la orden inversa; al pulsarlo, se manda la instrucción inalámbrica para que la cerradura principal active una alarma

LED Azul: Conectado al pin IO18 con una resistencia de protección de 220 Ohms (R1). Este foco se enciende brevemente cuando el control está transmitiendo la señal, avisando al usuario que el mando está funcionando.

LED Verde: Conectado al pin IO22 con su resistencia de 220 Ohms (R3). Gracias a que el protocolo ESP-NOW permite recibir una confirmación de recibido, este LED verde se enciende únicamente cuando la chapa principal le contesta al control remoto que la orden fue recibida y ejecutada correctamente.

Línea de 3.3V y 5V: El control está preparado para recibir energía regulada (por ejemplo, de una batería portátil o recargable) conectada a sus pines de voltaje y tierra común (GND), lo que mantiene el circuito encendido con un consumo mínimo de batería.

3.2. Acomodo, enrutamiento de pistas y resultados.

Una vez que se confirme la prueba del diagrama esquemático, se procede a la etapa del diseño en físico para una placa PCB y dos partes importantes son el acomodo de componentes y el enrutamiento de pistas.

3.2.1. Criterios de acomodo de componentes.

Para el acomodo se siguió una serie de condiciones:

Los componentes que son de interacción directa con el usuario (como RFID, la pantalla OLED) se pusieron estratégicamente en las orillas del perímetro de la placa. Esto da la facilidad a las conexiones externas como cables externos y evita que los conductores crucen por encima de la zona central del circuito.

Todos los componentes discretos de dos terminales (resistencias, diodos LED y capacitores) se orientaron bajo un mismo eje de simetría (ya sea completamente horizontales o verticales). Mantener una orientación homogénea facilita el proceso de ensamble.

Proximidad de Alta Frecuencia: Los elementos que operan con señales síncronas de reloj (como el reloj que operan en el RFID y pantalla OLED) o buses de datos rápidos (en la pantalla OLED) se colocaron lo más cerca posible de los pines correspondientes del ESP32. Reducir la distancia física de estas conexiones minimiza la inductancia parásita de las pistas y reduce la probabilidad

de que actúen como antenas receptoras de ruido externo.

3.2.1.1. Resultado del acomodo de componentes.

En la figura 18 se muestra el resultado del diseño a base de los criterios anteriores mencionados para tener una tarjeta de alta fidelidad y la estructura de compacta.

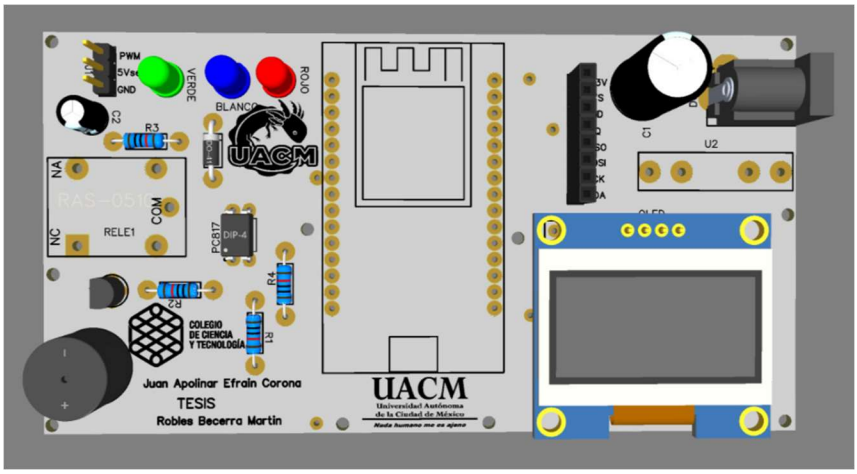


Figura 18. Diseño de cada componente en la placa PCB.

Para el diseño de componentes del control remoto se logró un resultado poniendo los componentes alrededor del microcontrolador como se muestra en la figura 19 con sus respectivos LEDs indicadores y los dos push buttom para mandar la instrucciones de acceso permitido y acceso denegado.

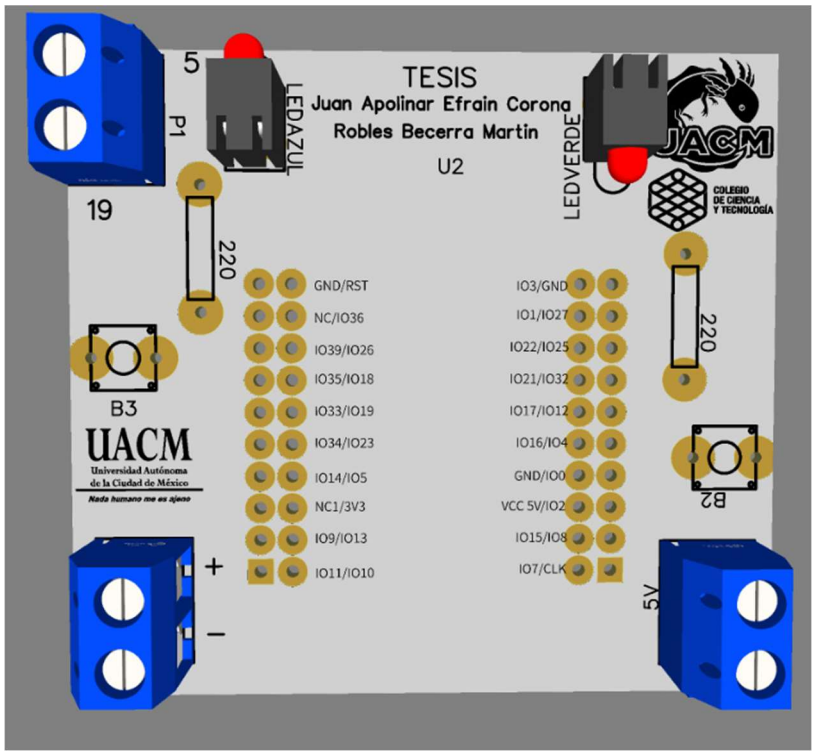


Figura 19. Diseño del control remoto.

3.2.2. Estrategia de Enrutamiento de Pistas.

Para esta parte se tuvo consideración varios puntos importantes para el enrutamiento como cuanta corriente circula y no cortar la pista por falta de espacio o cruzar pistas logrando el 100% de interconexión a través de las pistas de cobre, este criterio no es solamente estético, si no la prevención de fallas críticas en el sistema.

3.2.2.1 Enrutamiento de pistas para PCB.

El trazado de las pistas de la tarjeta se realizó utilizando una sola cara de cobre (diseñada en la capa inferior o Bottom Layer de EasyEDA). Diseñar la placa en una sola cara representa un reto importante, ya que todas las conexiones deben acomodarse en el mismo plano de plástico y cobre sin cruzarse ni tocarse entre sí, acomodando las piezas de forma estratégica y usando el mínimo de puentes de alambre rígidos (jumpers) puesto que se va a dar un poco de prioridad la reducción de tamaño de la placa.

Acomodo Estratégico de las Pistas: Para evitar que las líneas de conexión chocaran entre sí, se buscaron los caminos más cortos y directos. Las señales de datos de los componentes más rápidos, como el lector RFID y la pantalla OLED, se acomodaron rodeando las demás piezas de la placa, asegurando que la información llegue limpia y sin interferencias al microcontrolador ESP32.

Uso de Puentes de Alambre Planificados (Jumpers): En los puntos donde era físicamente imposible conectar una pista sin cruzarse con otra, se planeó el uso de jumpers o puentes hechos con pedazos de alambre rígido de cobre. Estos pequeños puentes se colocaron únicamente en líneas de datos secundarias y nunca en las líneas principales de energía, evitando así pérdidas de voltaje en el circuito. Plano de Tierra Común en una Sola Cara: Todo el espacio vacío que quedó en la placa se rellenó con un polígono de cobre conectado a tierra (GND). Este plano de masa común envuelve todas las pistas, funcionando como un escudo protector contra el ruido eléctrico. Esto es fundamental para proteger la antena del ESP32 y asegurar que la conexión con las conexiones usadas.

3.2.2.2. resultado de enrutamiento

En la figura 20 se muestran las dos partes que se van a aplicar en la placa PCB con sus respectivas pistas.

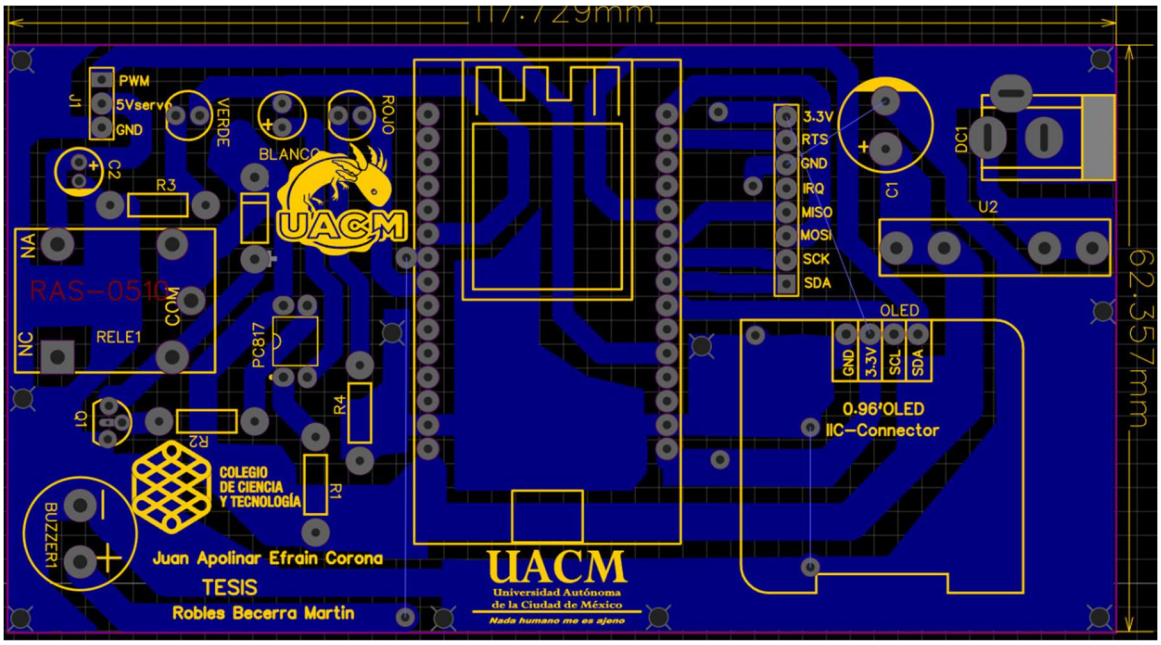


Figura 20. Diseño completo de la PCB.

Para el control remoto se obtuvo el siguiente resultado haciendo mención importante que se realizó de una sola cara puesto que el diagrama es bastante compacto como se muestra en la figura 21.

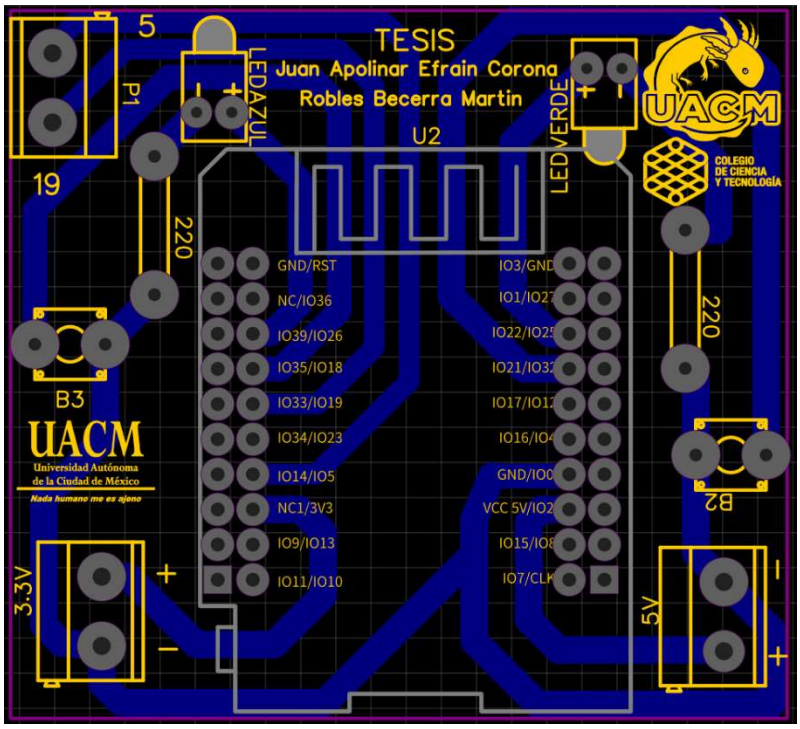


Figura 21. Diseño de control remoto PCB.

3.2.3. Elaboración de maqueta de la chapa.

Para comprobar que el proyecto funciona realmente en la vida real, se construyó la maqueta física que se muestra en la figura 22. Esta maqueta imita el comportamiento de una puerta común y su marco, utilizando dos bloques de madera unidos por bisagras de metal para que la puerta pueda abrirse y cerrarse de verdad.

A continuación, se explica de manera sencilla cómo se acomodaron y para qué sirven las piezas que se ven en la foto:

En lugar de gastar en una cerradura electrónica comercial cara, se utilizó una chapa clásica de sobreponer de la marca Phillips. Esta cerradura se dejó exactamente como viene de fábrica, manteniendo su botón original y la entrada para la llave física por el otro lado.

El tornillo del motor se sujetó directamente a la pieza metálica trasera que mueve el pasador. Esto es una gran ventaja para el proyecto, ya que demuestra que el sistema se adapta a lo que el usuario ya tiene instalado en su casa. Además, funciona como un sistema de seguridad: si se corta la luz o se apaga el circuito,

la puerta se puede seguir abriendo por fuera usando su llave normal de toda la vida.

Pieza en la Maqueta	¿Qué componente se usó?	¿Qué función hace en la práctica?
Base de la maqueta	Bloques de madera con bisagras	Simular la puerta y el marco para ver cómo abre el sistema.
Motor de fuerza	Servomotor con engranes de metal	Hacer la fuerza necesaria para jalar el pasador de la chapa.
Brazo del motor	Brazo de aluminio reforzado	Pasar la fuerza del motor al tornillo sin doblarse.
Pieza de unión	Tornillo de acero ajustable	Convertir el giro del motor en un empuje recto para la chapa.
Cerradura	Chapa de sobreponer Phillips	Servir como el candado físico que asegura la puerta.

Tabla 5. Partes de la maqueta con su respectiva función.

En conclusión, la elaboración de esta maqueta demostró que es completamente viable automatizar una cerradura residencial de forma económica y sin complicaciones. Al unir con éxito la tarjeta electrónica, el servomotor de fuerza y la chapa Phillips en una sola estructura de madera, se logró un prototipo sólido, ordenado y que responde de inmediato a las órdenes de apertura.

El buen funcionamiento de este mecanismo confirma que el diseño cumple con los objetivos de estabilidad, seguridad y facilidad de instalación planteados al inicio del proyecto, quedando listo para pasar a la etapa de pruebas finales de laboratorio y conectividad.

A continuación, en la figura 22, se presenta la evidencia física del sistema completamente armado:

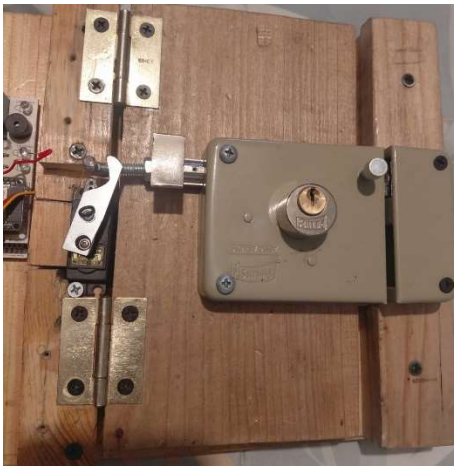


Figura 22. Vista general de la maqueta de pruebas con la integración final de los componentes mecánicos

Capítulo 4

Pruebas individuales físicas de los prototipos y sus módulos.

4.1Prueba de módulos de conexión.

Con la maqueta completamente armada y la tarjeta electrónica ensamblada, se realizaron pruebas de funcionamiento directamente en el hardware para asegurar que cada módulo trabaje de manera correcta y segura antes de poner el sistema en operación continua.

A continuación, se describen las pruebas realizadas a cada uno de los bloques del proyecto:

4.1.1. Módulo de Procesamiento Central ESP32.

Objetivo de la prueba: Verificar la estabilidad eléctrica del chip operando a 3.3V y comprobar la calidad de sus conexiones inalámbricas.

Procedimiento: Se alimentó el módulo ESP32-WROOM a través de la tarjeta y se monitoreo si hubo altas temperaturas en el microcontrolador o encendido de una manera correcta.

Resultado: El regulador mantuvo el voltaje estable en 3.3V sin variaciones. Gracias a que en el diseño de EasyEDA se dejó el área debajo de la antena libre de cobre, el chip se enlazó de inmediato a la red Wi-Fi para comunicarse con Blynk y mantuvo un canal de datos rápido con el protocolo ESP-NOW. El circuito integrado no presentó sobrecalentamiento durante las horas de prueba. En la figura 23 se muestra después de un largo funcionamiento operando sin ningún problema.

Figura 23. Módulo de Procesamiento Central ESP32.Módulo de Autenticación

4.1.2. (Lector RFID-RC522)

Objetivo de la prueba: Validar la lectura correcta de las tarjetas de acceso y llaveros a través del bus de comunicación SPI.

Procedimiento: Se programó el ESP32 para recibir los códigos únicos (UID) de los llaveros y se pasaron diferentes tarjetas sobre el lector a distintas distancias.

Resultado: El sensor detectó las tarjetas autorizadas a una distancia óptima de hasta 3 cm. La comunicación por el bus SPI fue limpia y sin pérdida de datos, reconociendo al instante qué usuario intentaba ingresar para mandar la señal de validación y se muestra en la figura 24.

Figura 24. Lector RFID en funcionamiento.

4.1.3. Módulo de Aislamiento y Conmutación (Optoacoplador y Transistor BC547).

Objetivo de la prueba: Asegurar que la etapa digital de control active la etapa de potencia de forma segura y sin retornos de ruido eléctrico agregándole dos capacitores, uno para el encendido de todo el circuito y otro para la parte del servomotor en caso de que genere picos de voltaje no deseado.

Procedimiento: Se mandó un pulso digital desde un pin del ESP32 hacia el optoacoplador para que este, a su vez, activara la base del transistor BC547.

Resultado: El circuito funcionó perfectamente como un interruptor electrónico. Al recibir el disparo, el transistor en encapsulado conmutó de inmediato para activar el relevador. La prueba confirmó que el plano de tierra común absorbió cualquier tipo de interferencia, protegiendo por completo los pines que son sensibles del ESP32.

4.1.4. Módulo de Actuación Mecánica (Servomotor MG995 y Chapa Phillips)

Objetivo de la prueba: Evaluar el torque del motor y comprobar el correcto arrastre del pasador físico dentro de la maqueta de madera.

Procedimiento: Se mandó la señal de apertura de la cerradura y se midió el comportamiento del servomotor acoplado al tornillo ajustable de acero.

Resultado: El motor entregó la fuerza necesaria para jalar el pasador de la chapa Phillips y vencer la resistencia de su resorte interno, completando la carrera de apertura de forma suave y el mecanismo se muestra en la figura 25.

Figura 25. Chapa y servomotor.

4.1.5 Control remoto (ESPNOW).

Objetivo de la prueba: Evaluar la estabilidad, la velocidad de respuesta y la correcta recepción de datos binarios y de estado entre los módulos ESP32 utilizando el protocolo de comunicación inalámbrica ESP-NOW.

Procedimiento: Se enviaron datos simulando solicitudes de acceso remoto (comandos de apertura) desde el emisor, registrando los tiempos de confirmación de entrega y verificando la prioridad de la señal frente a peticiones simultáneas en el receptor.

Resultado: El enlace inalámbrico no demostró errores y un emparejamiento inmediato, logrando recibir e interpretar correctamente el 100% de los comandos de apertura enviados. Validó con éxito los paquetes de datos y otorgó el acceso sin interferir con las demás tareas de control de la placa, confirmando la estabilidad del canal de comunicación. El diseño final se muestra en la figura 26.

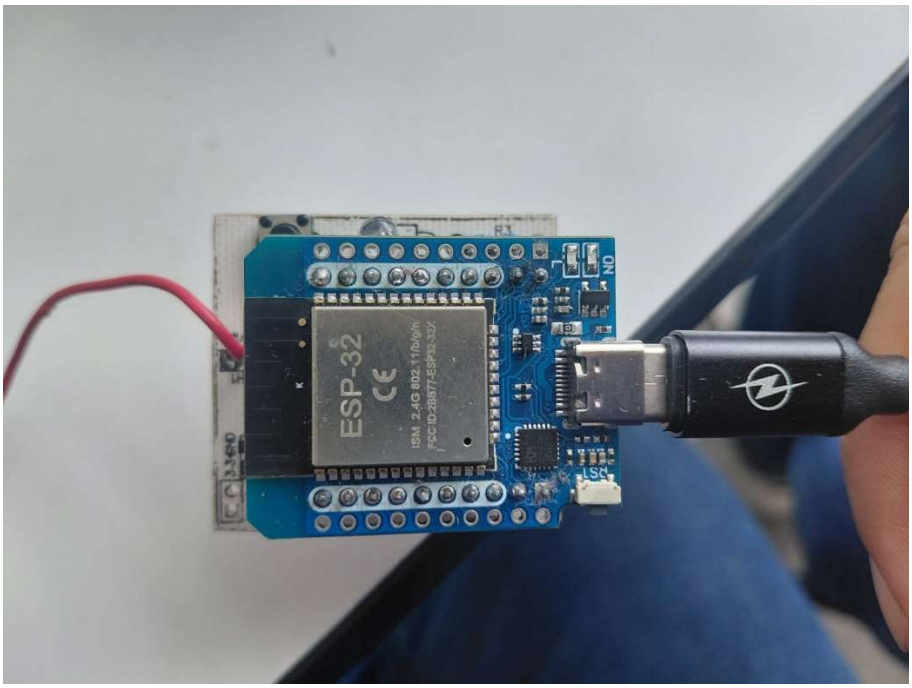


Figura 26. Diseño final de la placa del control remoto.

4.1.6. Salidas de control de acceso.

Objetivo de la prueba: El objetivo es indicar los accesos ya sea permitido o denegado, además de un buzzer que es un indicador auditivo diferente para cada acceso.

Procedimiento: Se enviaron los datos por medio de los diferentes tipos de acceso usados para poder ver las indicaciones visuales y auditivas para darle entender al usuario que función se llevó a cabo.

Resultado: Al momento de mandar los datos se observó satisfactoriamente que cada LED y el buzzer respondían como se configuró.

4.1.7. Resultado final del proyecto.

Como se muestra en la figura 27 y 28 el correcto funcionamiento del control de la chapa electrónica con su respectiva placa PCB y los diferentes tipos de acceso implementados.

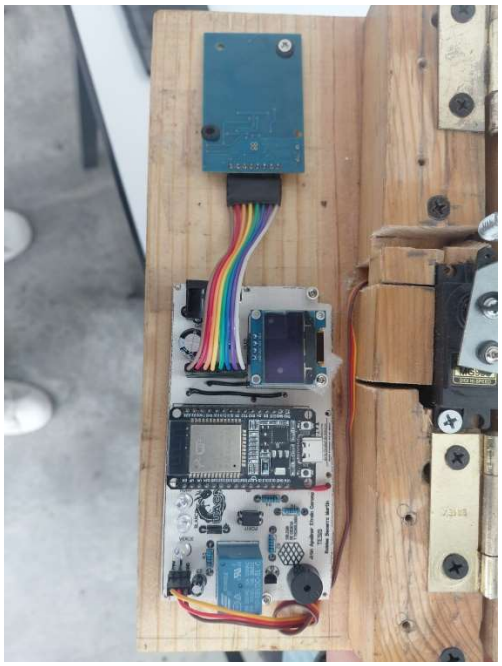


Figura 27. Placa PCB y accesos.

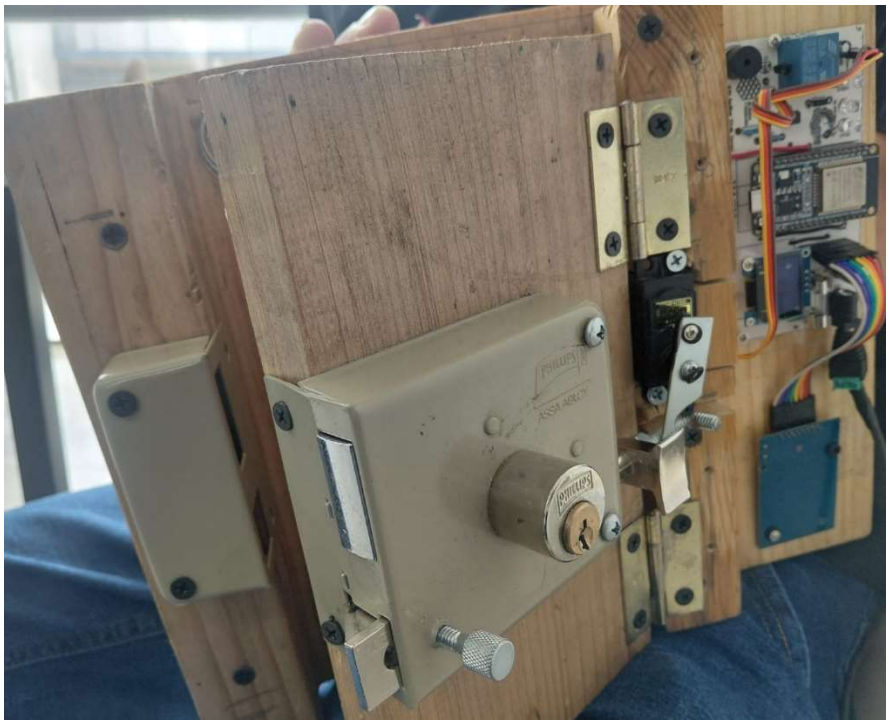


Figura 28. Placa electrónica instalada en la chapa.

Conclusión.

Para concluir, nos encontramos con muchas dificultades a lo largo de todo el proyecto, desde la programación, donde muchas veces el código no funcionaba como queríamos y otras, donde el código no tenía un orden de prioridad al dar los accesos. Con el tiempo fuimos puliendo esa parte, lo que nos dio un código completamente funcional.

También la parte del desarrollo de la placa, tuvimos muchas cosas en contra, porque no siempre quedaban a la primera, teníamos que estar haciendo una tras otra, ver que, si nos servía y ver que no nos servía, para que la placa final contara con lo esencial y es lo que conseguimos al final.

Al hacer este proyecto aprendimos y pulimos nuestras habilidades, tanto de diseño de PCB, ver como acomodaría cada componente en la placa o que tan grueso estarían las pistas, en la parte de la programación, estructurar toda la parte del código de una manera que no interfiriera con cada uno de los accesos y en realizar la placa físicamente con lo que conlleva, soldar, perforar, planchar, son habilidades que me ayudaron. para llevar a cabo este proyecto y que funcione como se planeaba desde un principio.

Por otro lado, el diseño y la elaboración de la placa de circuito impreso (PCB).Al no contar con un resultado perfecto en los primeros intentos, fue necesario realizar múltiples pruebas en la plataforma EasyEDA. Este ejercicio nos obligó a analizar minuciosamente qué elementos eran realmente indispensables, descartando conexiones redundantes y aplicando criterios técnicos esenciales, como el uso de un plano de tierra común.

El resultado final es una maqueta completamente operativa que cumple con los objetivos de bajo costo y facilidad de instalación planteados desde el inicio, demostrando que la persistencia y la aplicación correcta de los criterios de diseño permiten desarrollar soluciones tecnológicas eficientes y profesionales.

Apéndice A.
Especificaciones técnicas de los dispositivos.

Motor MG995.

Especificaciones:

Giro de 180 grados

Voltaje de operación: 4.8v-7.2v

Torque: 11kg/cm (6V)

Velocidad de operación: 0.16 seg/60 grados (6v sin carga)

Temperatura de trabajo: -30°C - 60°C

Conector universal para la mayoría de los receptores de radio control

Incluye las piezas que se pueden ver en la imagen

RFID RC522

- Modelo del Chip:NXP MFRC522
- Frecuencia de Operación:13.56 MHz
- Voltaje de Operación:3.3V DC (Voltaje crítico)
- Corriente de Operación:13-26 mA a 3.3V
- Corriente Máxima:< 30 mA
- Corriente en Modo Sleep:< 80 uA
- Protocolo de Comunicación:SPI (Serial Peripheral Interface), I2C, UART (El módulo de venta común utiliza SPI)
- Velocidad de Transferencia (Máx.):10 Mbit/s (SPI)
- Distancia de Lectura: 0 mm a 60 mm (Depende del tag y del entorno)
- Temperatura de Operación:-20 °C a 80 °C
- Humedad de Operación:5% – 95% de Humedad Relativa
- Tarjetas Compatibles:MIFARE Classic 1K/4K, MIFARE Ultralight, MIFARE DESFire (ISO 14443A)
- Dimensiones: 40 mm x 60 mm.
- Peso total: 20 g.

Tarjeta Tag RFID

Esp32 Devkit V1

Microcontrolador: ESP32 (doble núcleo Xtensa LX6 hasta 240 MHz)

Conectividad inalámbrica: Wi-Fi 2.4 GHz 802.11 b/g/n y Bluetooth v4.2 + BLE

Memoria: 520 KB SRAM interna / 4 MB Flash

GPIO disponibles: Hasta 25 pines de propósito general (dependen de la placa)

Conversión analógica: ADC de 12 bits (múltiples canales), 2 x DAC de 8 bits

Interfaces: UART, SPI, I²C, I²S, PWM, CAN (según librerías)*

Alimentación de la tarjeta: 5 V vía USB-C (regulada a 3.3 V en placa)

Voltaje lógico: 3.3 V (no conectar señales de 5 V directamente a GPIO)

Programación: USB-C, compatible con Arduino IDE y ESP-IDF

Pantalla OLED

Módulo de Pantalla OLED SPI SSD1331 0.95" (7 Pines, 65K Colores)

Descripción de Pines

GND: Tierra

VCC: Voltaje positivo (2.8-5.5V)

SCL: Línea de reloj

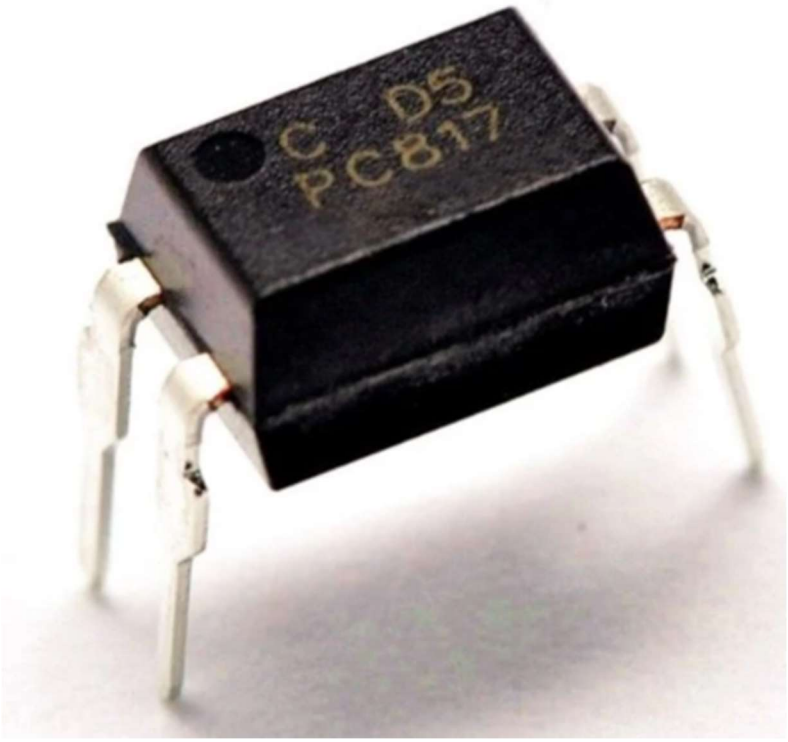
SDA: Línea de datos

RES: Reset

DC: Datos/Demanda

CS: Selección de chip

PC817



Voltaje de Aislamiento: 5000 V RMS [1, 2]

Lado de Entrada (LED Infrarrojo):

Corriente directa nominal recomendada: 5 mA - 20 mA (Máxima: 50 mA)

Voltaje directo: 1.2 V - 1.4 V

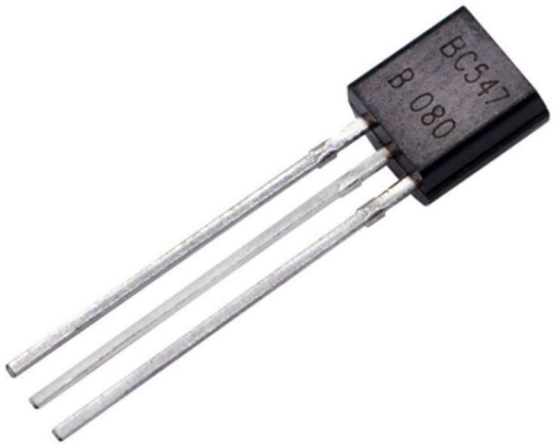
Voltaje inverso máximo: 6 V.

Lado de Salida (Fototransistor):

Voltaje máximo Colector-Emisor (VCEO): 80 V

Corriente máxima del colector (IC): 50 mA (Se recomienda no superar los 20 mA)

Transistor BC547.



Tipo de Transistor: NPN

Encapsulado: TO-92

Voltaje Colector-Emisor (\(V_{CEO}\)): 45 V máx.

Voltaje Colector-Base (\(V_{CBO}\)): 50 V máx.

Voltaje Emisor-Base (\(V_{EBO}\)): 6 V máx.

Corriente de Colector (\(I_{C}\)): 100 mA máx.

Disipación de potencia (\(P_{D}\)): 500 mW

Frecuencia de transición (\(f_{T}\)): 300 MHz

Phillips 650 PS

Dimensiones del producto empacado: 6.00 cm de alto x 14.00 cm de ancho x 19.00 cm de largo.

Peso aproximado: 1.46 kg, debido a su placa reforzada de alta resistencia.

Relevador RAS-05-10.

Voltaje de la bobina: 5 Vcc.

Configuración de contactos: 1 polo, 2 tiros (SPDT).

Capacidad de corriente nominal: 10 A.

Fusible.

Corriente: 3ª.

VCA: 250V.

Bibliografía.

Boylestad, R. L., & Nashelsky, L. (2009). Electrónica: teoría de circuitos y dispositivos electrónicos (10.ª ed.). Pearson Educación.

Espressif Systems. (2022). ESP32 technical reference manual [Manual].

Espressif Systems. (2023). ESP32 series datasheet [Data sheet].

Gupta, S. (2020). IoT essentials: Bluetooth low energy (BLE) fundamentals. IEEE Press.

Maier, A., Sharp, A., & Vaglica, J. (2018). Hands-on Internet of Things with ESP32. Packt Publishing.

Malvino, A. P., & Bates, D. J. (2016). Principios de electrónica (8.ª ed.). McGraw-Hill Education.

Mazidi, M. A., Naimi, S., & Naimi, S. (2010). The AVR microcontroller and embedded systems: Using Assembly and C. Pearson.

Ortiz Arciniega, J. L., Valencia Barahona, F. X., Bosmediano Cárdenas, C. P., Bastidas Jácome, A. D., Aguirre Chagna, V. H., & Jácome Ayala, P. J. (2025). ESP32: Manual básico para estudiantes (1.ª ed.). Alumni Editora.

Pettinato, G. (2019). ESP32: Programación y aplicaciones con Arduino IDE. Editores Técnicos.

Rashid, M. H. (2015). Electrónica de potencia: Dispositivos, circuitos y aplicaciones (4.ª ed.). Pearson Education.

Sedra, A. S., & Smith, K. C. (2014). Microelectronic circuits (7th ed.). Oxford University Press.

Serway, R. A., & Jewett, J. W. (2018). Física para ciencias e ingeniería (10.ª ed.). Cengage Learning.

Syngress. (2002). Bluetooth end to end. Elsevier Science.

Tipler, P. A., & Mosca, G. (2010). Física para la ciencia y la tecnología (Vol. 2, 6.ª ed.). Reverté.

Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). Sistemas digitales: Principios y aplicaciones (12.ª ed.). Pearson Educación.

Toshiba Semiconductor & Storage. (2021). General description of optocouplers/photocouplers [Application note].

Vishay Semiconductors. (2020). 4N25: Optocoupler, phototransistor output, high reliability, 5300 V RMS [Data sheet].

Vishay Semiconductors. (2022). Schottky barrier rectifiers: Technology and applications [Technical note].

Young, H. D., & Freedman, R. A. (2018). Sears y Zemansky: Física universitaria con física moderna (14.ª ed.). Pearson.

Onsemi. (2022). BC546, BC547, BC548, BC549, BC550: Amplifier transistors NPN solicon [Hoja de especificaciones]. Recuperado de https://www.onsemi.com/PDF/datasheet/bc546-d.pdf

Sharp Microelectronics. (2018). PC817 Series: High density mounting type photocoupler [Hoja de especificaciones]. Recuperado de https://global.sharp/products/device/lineup/data/pdf/datasheet/pc817_series_d.pdf