



# UNIVERSIDAD POLITÉCNICA DE JUVENTINO ROSAS

## Proyecto Integrador

**“Desarrollo de un sistema de control por medio de tecnología  
RFID para los laboratorios de la Universidad Politécnica de  
Juventino Rosas ”**

*Presenta:*

Brenda Shamiel Olivares Hernández  
Gilberto Yañez Ramírez  
Noelia Cacique Pérez

*Asesor:*

Luis Rey Lara Gonzalez

Santa Cruz de Juventino Rosas, Gto. 13 de Junio de 2024.

# Índice general

<b>Índice de figuras</b>	<b>6</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Abstrac . . . . .	1
1.2. Definición del Problema . . . . .	2
1.3. Justificación . . . . .	2
1.4. Objetivos del proyecto . . . . .	2
1.4.1. Objetivo General . . . . .	2
1.4.2. Objetivos Específicos . . . . .	3
<b>2. Estado del Arte</b>	<b>4</b>
<b>3. Marco Teórico</b>	<b>6</b>
<b>4. Desarrollo</b>	<b>8</b>
4.1. ESP32 . . . . .	8
4.1.1. Memorias Internas . . . . .	11
4.2. Periféricos . . . . .	12

4.3. Pines digitales . . . . .	14
4.4. Conversor analógico digital . . . . .	15
4.5. Controlador SD/SDIO/MMC . . . . .	15
4.6. UART . . . . .	15
4.7. Pines I2C ESP32 . . . . .	16
4.8. PWM . . . . .	16
4.9. SPI . . . . .	16
4.10. Chips y módulos con empaquetados QFN . . . . .	17
4.11. Módulos PCB . . . . .	18
4.12. Placas de desarrollo ESP32 . . . . .	19
4.13. ESP32 a utilizar . . . . .	20
4.14. RFID . . . . .	21
4.15. RFID a utilizar . . . . .	24
4.16. VPS . . . . .	25
4.17. Peticiones HTTP . . . . .	26
4.18. Pluma.php . . . . .	27
4.19. MQTT . . . . .	27
4.20. Mosquitto . . . . .	29
4.21. Base de datos . . . . .	30
4.22. Base de datos a utilizar . . . . .	32
<b>5. Análisis y Diseño</b>	<b>33</b>
5.1. Análisis de Requerimientos . . . . .	33
5.2. Aplicación del protocolo mqtt en Arduino IDE . . . . .	36

5.3.	Sistema de Control RFID . . . . .	38
5.4.	Mayor productividad + Mayor eficiencia = Beneficio . . . . .	38
5.5.	MQTT . . . . .	39
5.6.	PHP . . . . .	41
5.6.1.	Características al utilizar PHP: . . . . .	41
5.6.2.	Ventajas al utilizar PHP: . . . . .	42
5.6.3.	Desventajas al utilizar PHP: . . . . .	43
5.7.	Aplicación del protocolo mqtt en Visual Studio Code . . . . .	44
5.8.	Desarrollo de la Base de Datos . . . . .	46
5.9.	DIAGRAMA DE LO REALIZADO EN EL PARCIAL II . . . . .	47
<b>6.</b>	<b>Migración e Implementación del sistema de control en Oracle Cloud</b>	<b>48</b>
6.1.	Migración a Oracle Cloud . . . . .	48
6.1.1.	Oracle Cloud . . . . .	49
6.1.2.	Características principales de Oracle Cloud: . . . . .	49
6.1.3.	Ventajas de usar Oracle Cloud: . . . . .	50
6.1.4.	Justificación de la migración . . . . .	50
6.1.5.	Preparación para la migración . . . . .	52
6.1.6.	Configuración del entorno en Oracle Cloud . . . . .	52
6.1.7.	Proceso de migración . . . . .	52
6.1.8.	Transferencia de los programas a la nube . . . . .	53
6.2.	Aplicación Móvil . . . . .	54
6.3.	Configuración de MQTT . . . . .	54
6.3.1.	Dependencias . . . . .	54

6.3.2. Conexión al Broker . . . . .	54
6.4. Suscripción a Temas . . . . .	55
6.5. Recepción de Mensajes . . . . .	55
6.6. Publicación de Mensajes . . . . .	55
6.7. Desconexión del Broker . . . . .	55
6.8. Manejo de Errores y Reconexión . . . . .	56
6.9. Flujo General . . . . .	56
<b>7. Desarrollo de un Sistema Web y Aplicación Móvil para la Gestión de Registros mediante Tecnología RFID</b>	<b>57</b>
7.1. Tecnologías Utilizadas . . . . .	58
7.2. Página web . . . . .	59
7.3. Arquitectura del Sistema . . . . .	59
7.4. Página principal . . . . .	60
7.5. Selección de Espacios Monitoreados . . . . .	61
7.6. Control de Accesos y Usuarios . . . . .	62
7.7. Base de Datos . . . . .	63
7.8. Autenticación y Seguridad . . . . .	63
7.9. Filtro por Fecha y Hora . . . . .	64
7.10. Generación de Reportes en PDF . . . . .	65
7.11. Aplicación móvil . . . . .	65
7.12. Detalle del código Flutter . . . . .	66
7.13. Visualización de Imágenes en Tiempo Real . . . . .	67
7.14. Integración con el Sistema RFID . . . . .	67

7.15. Próximos Pasos . . . . .	68
--------------------------------	----

# Índice de figuras

4.1. Conectividad . . . . .	9
4.2. Memorias . . . . .	10
4.3. Modulo Periférico de ESP32. . . . .	13
4.4. Pines Digitales en ESP32. . . . .	14
4.5. Empaquetado en ESP32 . . . . .	18
4.6. ESP32 DEV KIT DOIT . . . . .	21
4.7. Datasheet DEV KIT DOIT . . . . .	21
4.8. Lector de proximidad SYRIS SYSKR600E . . . . .	25
4.9. VPS . . . . .	26
4.10. Mosquitto . . . . .	30
4.11. Base de datos (MYSQL) . . . . .	32
5.1. Comunicación RFID . . . . .	34
5.2. Comunicación RFID . . . . .	35
5.3. Función Callback . . . . .	36
5.4. Función para Reconectar al Broker MQTT . . . . .	37
5.5. Codigo Parte 1 Realizado en Visual Studio Code . . . . .	44

5.6. Código Parte 2 Realizado en Visual Studio Code . . . . .	45
5.7. Tabla 1 Datos. . . . .	46
5.8. Tabla 2 Estudiantes. . . . .	47
5.9. Trabajo realizado. . . . .	47
6.1. Comunicación RFID . . . . .	49
7.1. Diagrama de flujo . . . . .	58
7.2. Base de datos (MYSQL) . . . . .	60
7.3. Datos almacenados en la base de datos . . . . .	60
7.4. Página espacios monitoreados . . . . .	61
7.5. Topic cambiado . . . . .	62
7.6. Espacios asignados . . . . .	63
7.7. Datos mostrados en la pagina web . . . . .	64



# Capítulo 1

## Introducción

En este capítulo se presentan los antecedentes del proyecto ya realizados, y el contexto de la problemática a la cual se dará solución para su entendimiento. De esta manera, el lector RFID comprenderá el escenario real y deseado con la solución propuesta y sus objetivos, particular y generales, así como la organización del documento.

### 1.1. Abstrac

This chapter presents the background of projects already carried out, and the context of the problem to which a solution will be given for understanding. In this way, the RFID reader will understand the real and desired scenario with the proposed solution and its objectives, particular and general, as well as the organization of the document.

## **1.2. Definición del Problema**

La Universidad Politécnica de Juventino Rosas actualmente no cuentan con un sistema de control, aunque este proyecto ha sido diseñado enteramente como un desarrollo a la medida servira para poder tener un mejor control en entradas de laboratorios, auditorios, cafeteria, y entrada a la universidad. Se describe el desarrollo y la implementación de un sistema de tecnologia RID con base de datos en MYSQL, para realizar el control requerido en la Universidad.

## **1.3. Justificación**

La tecnología de RFID ha emergido como una herramienta esencial en la gestión y control de recursos en diversos entornos, incluyendo los laboratorios universitarios. Los laboratorios de investigación y docencia en las universidades suelen contar con una amplia variedad de equipos, reactivos y materiales que requieren una gestión eficiente para asegurar su disponibilidad, correcto uso y mantenimiento. La implementación de un sistema RFID en estos laboratorios puede ofrecer una solución innovadora y eficiente para estos desafíos.

## **1.4. Objetivos del proyecto**

### **1.4.1. Objetivo General**

Desarrollo e implementación de un sistema de control por medio de tecnología RFID y MQTT para los laboratorios de la Universidad Politécnica de Juventino Rosas.

### **1.4.2. Objetivos Específicos**

- Investigar y elaborar lo requerido sobre la tecnología RFID.
- Investigación del protocolo MQTT.
- Aplicar tecnologías de Sistemas Embebidos y comunicación entre redes.

# Capítulo 2

## Estado del Arte

- Diseño del sistema de control de asistencia en la agencia mercado de la caja Huancayo aplicando tarjetas de proximidad de tecnología de RFID muestra el diseño de un sistema de control de asistencia en la agencia mercado de la Caja Huancayo aplicando Tarjetas de proximidad de Tecnología RFID específicamente para mejorar el control de los registros de asistencia del personal de la Caja Huancayo con la finalidad de agilizar este proceso que actualmente se conlleva con malas practicas de los colaboradores, el diseño planteado como solución se baso en la metodología de descomposición funcional que conllevo a el análisis e identificación de requerimientos y funcionalidades del sistema propuesto tanto a nivel de hardware y software así como al diseño de la solución en los diferentes niveles y/o aspectos, se llegaron a elegir y proponer los componentes del sistema luego de una comparativa.
- Control de registro de asistencia con control de acceso Colegio Verbo Divino Chicureo. Este proyecto describe la forma en la cual se apoyara, desde la innovación y tecnología, la construcción de un sistema de registro de asistencia. El registro de

asistencia actualmente se realiza de forma manual en una planilla impresa. Durante la jornada la planilla de asistencia es retirada por los inspectores de ciclo, los que posteriormente se encargan de registrarla en el sistema. El trabajo presentado en este proyecto busca la realización de un prototipo de sistema capaz de simplificar las labores diarias de los profesores e inspectores, registrando la asistencia diaria y que esta pueda ser visualizada en tiempo real por los usuarios encargados de registrarla en el sistema, así como también aplicar las sanciones correspondientes en caso de faltas u atrasos.

# Capítulo 3

## Marco Teórico

En este capítulo se presentan los conceptos fundamentales necesarios para el desarrollo del presente trabajo, tanto para la aplicación de escritorio. Esto permitirá al lector familiarizarse con el tema, así como una mejor comprensión del mismo debido a los conceptos que se manejan. Lo que conocemos de nuestro proyecto es que principalmente este está basado en llevar un control dentro de la escuela por medio de las credenciales escolares, tecnología RFID, programación y base de datos que es donde se guardaran los registros ya cuando este se encuentre funcionando. RFID o identificación por Radiofrecuencia. Lo que permite hacer RFID es una tecnología capaz de almacenar, recuperar y registrar una inmensa cantidad de datos (hasta cuatro millones de caracteres y millares de bytes) en un pequeño chip y transmitirlos a través de ondas de radio o radiofrecuencia. RFID es, en definitiva, un sistema de comunicación inalámbrica entre un emisor y un receptor permite grabar, almacenar y transmitir una enorme cantidad de datos de forma automática en segundos e incorporarlos en tiempo real a los programas de gestión. Las investigaciones más recientes del tema han sido sobre conocer el funcionamiento de RFID como es que

puedo llevar el control por medio de una base de datos como es que puede trabajar y también cuales son los conocimientos requeridos para el desempeño y la realización de este proyecto.

# Capítulo 4

## Desarrollo

### 4.1. ESP32

ESP32 es una familia de microcontroladores de la empresa Espressif Systems. Su analogía mas clara es la de un ESP8266 con asteroides, que incluye Wifi, Bluetooth y otras potentes características extra.ESP32 es una serie de SoC (por sus siglas en ingles, System on Chip) y módulos de bajo costo y bajo consumo de energía creado por Espressif Systems. Esta nueva familia es la sucesora del conocido ESP8266 y su característica mas notable es que, ademas de Wi-Fi, también soporta Bluetooth. Los ESP32 poseen un alto nivel de integración.En su pequeño encapsulado se incluyen:

- Interruptores de antena.
- Balun de RF.
- Amplificador de potencia.
- Amplificador de recepción de bajo ruido.



- Filtros y módulos de administración de energía.

Además de todo eso, logra un consumo de energía muy bajo a través de funciones de ahorro de energía que incluyen sincronización on de reloj y múltiples modos de operación. El ESP32 viene con pines sensibles al tacto que se pueden usar para despertar al ESP32 de modo deep sleep (sueño profundo) y un sensor de efecto Hall incorporado. En la siguiente imagen, tomada de la hoja de datos, se muestran todos los bloques funcionales que conforman un SoC ESP32. El chip cuenta con conectividad WiFi, siendo compatible con 802.11 b/g/n en la banda de los 2.4GHz, alcanzando velocidades de hasta 150 Mbits/s. También incluye comunicación Bluetooth compatible con Bluetooth v4.2 y Bluetooth Low Energy (BLE).

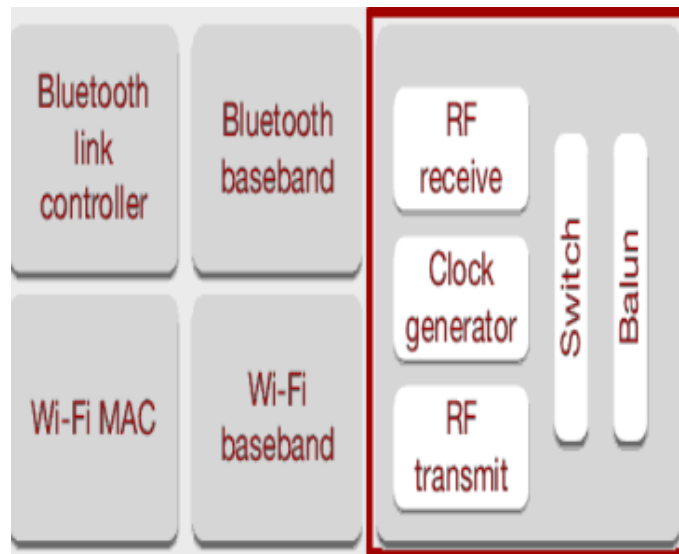


Figura 4.1: Conectividad

El bloque de radio está estrechamente ligado a los módulos de comunicación inalámbricos. De hecho, este es el que realmente transmite y recibe la información. Es decir, toma

los datos digitales provenientes de los módulos Wifi y Bluetooth; y los convierte en señales electromagnéticas que viajan por el aire para comunicarse con tu teléfono móvil o tu router. Ojo, que también realiza la operación inversa: traducir las ondas electromagnéticas generadas por otros dispositivos en datos digitales que los módulos Wifi y Bluetooth son capaces de interpretar. El ESP32 cuenta con dos microprocesadores de bajo consumo Tensilica Xtensa de 32 bits LX6. Pero además, cuenta con un co-procesador de ultra bajo consumo que es utilizado para realizar conversiones analógico-digital y otras operaciones mientras el dispositivo se encuentra funcionando en el modo de bajo consumo deep sleep. De esta forma, se consigue un consumo muy bajo por parte del SoC. Las memorias internas son aquellas que se encuentran ya incluidas en el SoC, y las externas son aquellas que se pueden adicionar para expandir la capacidad del sistema. Muchas placas de desarrollo basadas en ESP32 añaden memorias externas para lograr un sistema con mejores prestaciones.

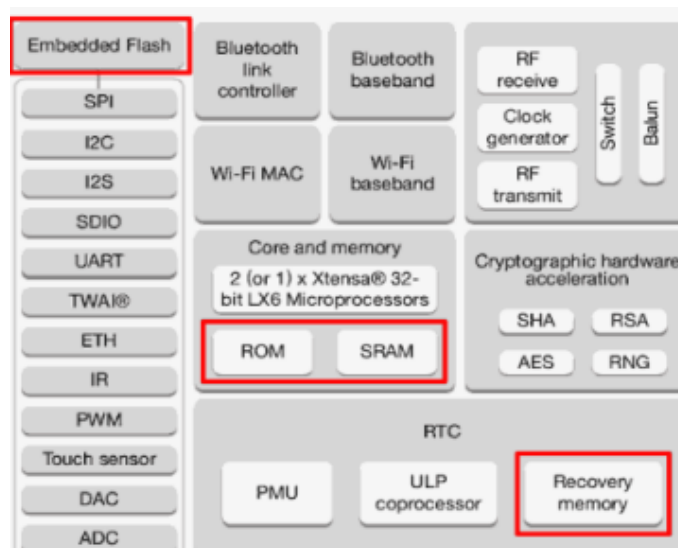


Figura 4.2: Memorias

### 4.1.1. Memorias Internas

En las memorias internas se encuentran:

- Memoria ROM (448 KiB), esta memoria es de solo escritura, es decir que no la puedes reprogramar. Aquí es donde se almacenan los códigos que manejan la pila Bluetooth, el control de la capa física de la Wifi, algunas rutinas de propósito general y el cargador de arranque (bootloader) para iniciar el código de la memoria externa.
- Memoria SRAMinterna (520 KiB), esta memoria es utilizada por el procesador para almacenar tanto datos como instrucciones. Su ventaja es que, para el procesador, es mucho mas fácil acceder a esta que a la SRAM externa.
- RTC SRAM (16 KiB), esta memoria es utilizada por el co-procesador cuando el dispositivo opera en modo deep sleep.
- Efuse (1 Kilobit), 256 bits de esta memoria son utilizados por el propio sistema y los 768 bits restantes están reservados para otras aplicaciones.
- Flash empotrada (Embedded flash), en esta memoria es donde se almacena el código de nuestra aplicación. La cantidad de memoria varía en dependencia del chip utilizado:
  - 0 MiB (chips ESP32-D0WDQ6, ESP32-D0WD y ESP32-S0WD)
  - 2 MiB (chip ESP32-D2WD)
  - 4 MiB (módulo SiP ESP32-PICO-D4)

## 4.2. Periféricos

El ESP32 cuenta con un gran conjunto de periféricos. Aunque son similares a los que poseen otros microcontroladores presentan sus diferencias. Se refiere a cualquier componente o módulo adicional que se integra con el microcontrolador para realizar tareas específicas. Estos periféricos pueden ser internos al chip o externos y se conectan a través de diversos interfaces. Algunos de los periféricos comunes que se encuentran en el ESP32 son:

- GPIO (General Purpose Input/Output): Pinos configurables que pueden servir como entradas o salidas digitales. Son útiles para conectar LEDs, botones, relés, entre otros.
- PWM (Pulse Width Modulation): Permite generar señales con modulación de ancho de pulso, útil para controlar la velocidad de motores o la intensidad de LEDs.
- ADC (Analog-to-Digital Converter): Convierte señales analógicas a digitales. Esto es útil para leer valores de sensores analógicos.
- DAC (Digital-to-Analog Converter): Convierte señales digitales a analógicas, lo que permite generar señales analógicas a partir de datos digitales.
- I2C (Inter-Integrated Circuit): Un bus de comunicación serial que permite conectar múltiples dispositivos con solo dos líneas: SCL (Clock) y SDA (Data).
- SPI (Serial Peripheral Interface): Un protocolo de comunicación serial que permite la comunicación rápida con periféricos como sensores, pantallas y memorias.

- UART (Universal Asynchronous Receiver-Transmitter): Una interfaz de comunicación serial asíncrona que permite la comunicación con dispositivos externos, como módulos Bluetooth, GPS, etc.
- Wi-Fi: El ESP32 tiene un módulo Wi-Fi integrado que permite conectividad a redes inalámbricas.
- Bluetooth: Incluye soporte tanto para Bluetooth clásico como para BLE (Bluetooth Low Energy), lo que facilita la comunicación con otros dispositivos Bluetooth.
- Timer: Contadores que pueden utilizarse para medir intervalos de tiempo, generar señales periódicas o realizar tareas programadas.
- RTC (Real-Time Clock): Un reloj de tiempo real que permite mantener la hora y la fecha, incluso cuando el ESP32 está en modo de bajo consumo.

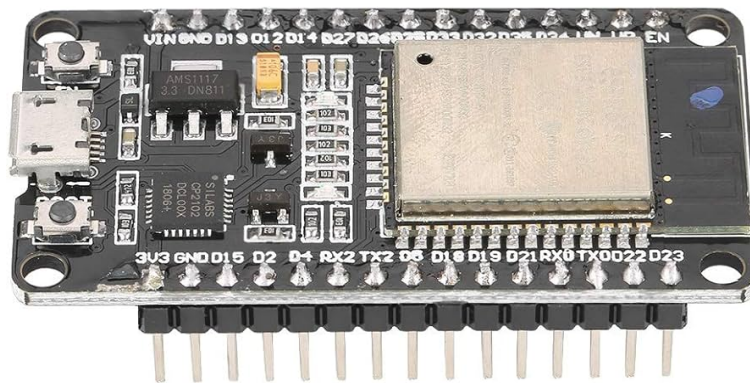


Figura 4.3: Módulo Periférico de ESP32.

### 4.3. Pines digitales

El ESP32 cuenta con un total de 34 pines digitales. Estos pines, al igual que en cualquier placa Arduino, permiten agregar LEDs, botones, zumbadores, y un largo etc. a nuestros proyectos. La mayoría de estos pines admiten el uso de pull-up , pull-down internos y también el estado de alta impedancia. Esto los hace ideales para conectar botones y teclados matriciales, así como para aplicar técnicas de control de LEDs como la conocida Charlieplexin.

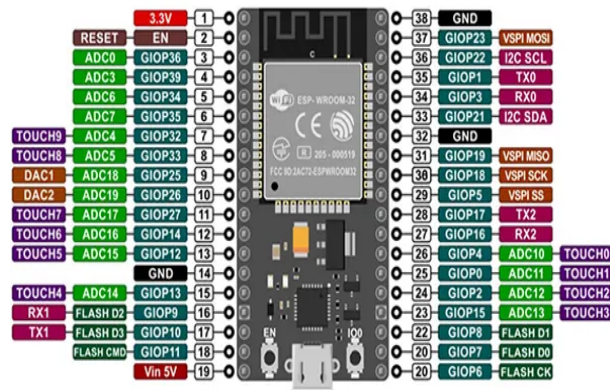


Figura 4.4: Pines Digitales en ESP32.

## 4.4. Conversor analógico digital

Algunos de los pines también pueden ser utilizados para interactuar con sensores analógicos, es decir, como si fueran los pines analógicos de una placa Arduino. Para esto el ESP32 cuenta con un conversor analógico digital de 12-bits y 18 canales, es decir, que puedes tomar lecturas de hasta 18 sensores analógicos. Esto te permite desarrollar aplicaciones conectadas muy compactas, incluso cuando se empleen varios sensores analógicos.

## 4.5. Controlador SD/SDIO/MMC

Este periférico permite al ESP32 interactuar con tarjetas SD y MMC directamente. De hecho, combinando este controlador con el conversor digital analógico es posible mejorar nuestro pequeño reproductor de audio.

## 4.6. UART

Muchos microcontroladores cuentan con módulos UART, que en Arduino son conocidos como puertos Serial. Estos permiten establecer comunicaciones asíncronas entre dos dispositivos utilizando solamente dos pines.

El ESP32 cuenta con tres puertos UART:

- UART0
- UART1
- UART2

Todos estos son compatibles con los protocolos RS-232, RS-485 e IrDA.

## 4.7. Pines I2C ESP32

Los ESP32 cuentan con dos interfaces I2C o TWI que admiten los modos de operación maestro y esclavo. Entre sus características destacan:

- Modo estándar (100 Kbit/s)
- Modo rápido (400 Kbit/s)
- Direccionamiento de 7 y 10 bits

## 4.8. PWM

Al igual que el ESP8266, el ESP32 también soporta el uso de salidas analógicas utilizando PWM. La gran diferencia es que en un ESP32 es posible utilizar hasta 16 pines como salidas PWM, una gran diferencia ante un ESP8266 que solo admite 8 o una placa Arduino UNO que solo soporta 6.

## 4.9. SPI

El ESP32 también cuenta con comunicación SPI. Cuenta con tres buses totalmente funcionales:

- Cuatro modos de transferencia: esto significa que es compatible con todos o casi todos los dispositivos SPI y QSPI disponibles en el mercado.
- Todos los puertos SPI son capaces de alcanzar altas velocidades (teóricamente hasta 80 MHz).



- Buffer de 64 bytes para transmisión y recepción.

Un Soc reúne todos los componentes de un ordenador en un chip. Mientras, que un SIP se compone de varios circuitos integrados en el mismo paquete. De modo general, lo puedes encontrar en cuatro formatos:

- SoC ( sistema en chip )
- SiP ( sistema en paquete )
- Módulos PCB
- Placas de desarrollo

Si bien es cierto que lo normal es comprar una placa de desarrollo y no un chip o módulo aislado, es necesario conocer los chips y sus características. Esto se debe a que las prestaciones de las placas de desarrollo dependen, en gran medida, del chip que utilizan.

## **4.10. Chips y módulos con empaquetados QFN**

Todos los SoCs y SiPs ESP32 utilizan el empaquetado QFN. Este es un encapsulado de circuito integrado para montaje superficial en el que los conectores se ubican por los cuatro lados, pero sin extenderse fuera del chip. Este permite obtener diseños muy compactos. Es por eso que muchos microcontroladores y microprocesadores se suelen fabricar con este empaquetado. Su desventaja radica en que es muy complicado, por no decir casi imposible, de utilizar de forma casera.

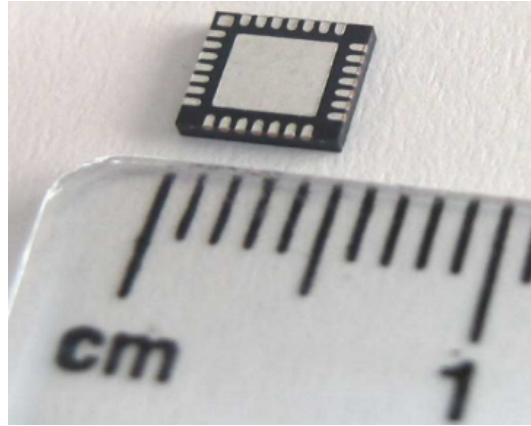


Figura 4.5: Empaquetado en ESP32

## 4.11. Módulos PCB

- ESP-WROOM-32
- ESP-WROOM-32D
- ESP-WROOM-32U
- ESP32-WROVER-B

El inconveniente de utilizar los chips anteriores es que es necesario agregar toda la electrónica externa. Además, teniendo en cuenta que se trata de chips de montaje superficial esa tarea más que complicada puede resultar imposible si no se cuenta con las herramientas adecuadas. Teniendo en cuenta esto, varias compañías se han dado a la tarea de fabricar módulos PCB basados en esos chips. Cada módulo es una pequeña placa de circuito impreso sobre la que se sueldan chips y otros componentes y se tratan como un conjunto inseparable para integrarse en otra PCB. Todos los módulos contienen un

chip ESP32, un cristal oscilador de 40 MHz y un chip de memoria flash, pero varían en características como:

- Configuración de la antena: algunos traen la antena integrada y otros disponen de un conector.
- Capacidad del chip de memoria flash: puede variar desde 1 MiB hasta 16 MiB.
- Blindaje EM: en algunos de estos módulos se ha agregado un blindaje para reducir las interferencias electromagnéticas.

Estos módulos están diseñados para ser incorporados a productos finales, por lo tanto, carecen de los componentes típicos de las placas de desarrollo como conversores USB-Serie o reguladores de voltaje.

## **4.12. Placas de desarrollo ESP32**

- ESP32 DEV KIT DOIT
- Adafruit HUZZAH32
- ESP32 Thing
- LOLIN32 con soporte de batería
- ESP32 LOLIN OLED
- SX1278 ESP32 (LoRa)

El principal inconveniente de todos los chips y módulos ESP32 vistos hasta ahora, es que no permiten una forma amigable de ser programados. Es decir, no poseen una interfaz USB al igual que las placas Arduino para ser programados. Además, tampoco cuentan con cabezales de pines que permitan realizar prototipos de forma fácil sin necesidad de crear un circuito impreso propio. Para tales propósitos existen las placas de desarrollo. Cuentan con conectores y otros componentes que te permitirán crear tus proyectos IoT aprovechando al máximo las características del ESP32.

## 4.13. ESP32 a utilizar

### ESP32 DEV KIT DOIT

Existen dos versiones de esta placa: una con 30 y otra con 36 GPIOs. Ambas se basan en el módulo PCB ESP32-WROOM-32 y son similares, siendo la distribución de pines su única diferencia. La versión de 36 GPIOs permite acceder a los 6 pines utilizados para la memoria externa. Teniendo en cuenta que Espressif recomienda no utilizar estos pines, lo mejor es utilizar la versión de 30 GPIOs. Los pines están etiquetados en la parte superior de la placa, lo que facilita la conexión con otros dispositivos. También cuenta con una interfaz USB-Serie y regulador de voltaje que permiten programar fácilmente la placa utilizando Arduino IDE o cualquier otra plataforma de desarrollo compatible. La placa puede ser alimentada utilizando el conector micro-USB, el pin VIN o el pin 3.3V.



Figura 4.6: ESP32 DEV KIT DOIT

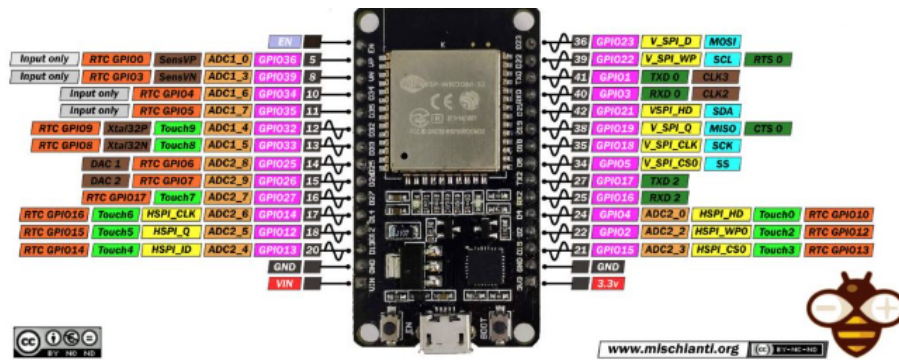


Figura 4.7: Datasheet DEV KIT DOIT

## 4.14. RFID

La tecnología RFID nace como una evolución del código de barras para agilizar el intercambio de información por medio del uso de ondas de radio. RFID es un sistema

de identificación entre objetos que utiliza radiofrecuencia. Esta tecnología permite que los datos que se encuentran codificados en etiquetas RFID inteligentes sean capturados por un lector RFID mediante ondas de radio. La principal ventaja es que la posibilidad de identificar objetos individuales de forma única, rápida y de manera eficiente. Esta tecnología es usada por ejemplo en el control de inventario, trazabilidad, control de acceso a áreas restringidas, logística, control de vehículos, rastreo de artículos, gestión de la cadena de suministro, e incluso permite prevenir la falsificación de ciertos productos. Para poder llevar a cabo la identificación de objetos mediante radiofrecuencia, es necesario tener una serie de componentes. Por ejemplo, de manera sencilla, la etiqueta RFID almacena información en su memoria interna. La antena RFID recibe la información almacenada en la etiqueta y la envía al lector. El lector RFID recibe la información y la transmite hacia el ordenador. En el ordenador, los datos obtenidos son almacenados en una base de datos, para que luego un software pueda analizarlos. Hay diferentes tipos de RFID que varían en frecuencia de operación, alcance, capacidad de almacenamiento y aplicación específica. Entre estos tipos más comunes están:

#### **Baja Frecuencia (LF):**

- Frecuencia de operación: 125-134 kHz.
- Alcance corto (generalmente hasta 10 cm).
- Menor interferencia por objetos metálicos o líquidos.
- Usos comunes incluyen sistemas de control de acceso, identificación de mascotas y sistemas de pago.

#### **Alta Frecuencia (HF):**

- Frecuencia de operación: 13.56 MHz.
- Alcance moderado (hasta 1 metro).
- Se utiliza en aplicaciones como tarjetas de acceso, pagos sin contacto, y seguimiento de inventario.

#### **Ultra Alta Frecuencia (UHF):**

- Frecuencia de operación: 860-960 MHz.
- Alcance extendido (puede superar los 10 metros).
- Alta velocidad de lectura y escritura.
- Se emplea en logística, seguimiento de activos, gestión de inventario y etiquetado de productos.

#### **Frecuencia Extremadamente Alta (EHF):**

- Frecuencia de operación: 30-300 GHz.
- En desarrollo para aplicaciones de alta velocidad y alta precisión, como seguimiento de inventario en tiempo real en almacenes y entornos industriales.

#### **Pasivo vs. Activo:**

- RFID pasivo: No tiene alimentación interna y se activa por la señal del lector. Más económico y se usa comúnmente en etiquetas de inventario y control de acceso.
- RFID activo: Tiene su propia fuente de energía y transmite datos de forma independiente. Mayor alcance y capacidad de almacenamiento, utilizado en aplicaciones de seguimiento de vehículos, contenedores y activos de alto valor.

### **Semi-pasivo:**

- También conocido como batería asistida pasiva, tiene una fuente de energía limitada que ayuda en la comunicación con el lector. Se utiliza en aplicaciones donde se necesita un alcance extendido y una mayor potencia de transmisión que las etiquetas pasivas.

## **4.15. RFID a utilizar**

SYSKR600E Lector de Proximidad EM 125 kHz. El AccessPRO Lector de Tarjeta de Proximidad, Chip, SYS-KR600-E es un dispositivo diseñado para el control básico de acceso en exteriores. Su carcasa está fabricada con acrilonitrilo butadieno estireno (ABS) y cuenta con una clasificación IP65 que lo hace resistente al polvo y a la lluvia.

Características:

- Para uso en exterior.
- Wiegand a 26 bits.
- Sonido audible.
- Plástico ABS.
- Compatible con tarjetas ACCESISOCARD, ACCESSPROXCARD y ATR261BO
- Frecuencia de operación 125 kHz.

El lector de proximidad SYRIS SYSKR600E es un dispositivo utilizado para la lectura de tarjetas de proximidad. Este tipo de dispositivos se utilizan comúnmente en sistemas



de control de acceso, como en empresas, instituciones educativas o edificios residenciales, donde se necesita autenticar la identidad de una persona para permitir su acceso a ciertas áreas. El SYSKR600E es fabricado por SYRIS, una empresa especializada en sistemas de seguridad y control de acceso. Este lector de proximidad utiliza tecnología RFID (Identificación por Radio Frecuencia) para leer tarjetas o etiquetas de proximidad. Cuando se presenta una tarjeta válida ante el lector, este verifica la información contenida en la tarjeta y muestra en una base de datos la información de la persona.



Figura 4.8: Lector de proximidad SYRIS SYSKR600E

## 4.16. VPS

Un VPS, o servidor virtual privado, es una forma de hosting en la nube de servicio multi-inquilino en el que los recursos del servidor virtualizado están disponibles para un usuario final a través de Internet, mediante un proveedor de hosting o de nube. Cada VPS se instala en una máquina física, operada por el proveedor de hosting o de nube, que

ejecuta varios VPS. Al crear un sitio web o una aplicación web, los clientes generalmente necesitan configurar una base de datos, configurar un servidor web y añadir su código. La gestión del hardware del servidor físico puede ser compleja y costosa. Para resolver este problema, los proveedores de alojamiento administran el hardware subyacente y permiten a los usuarios consumir estos recursos. En el alojamiento VPS, cada usuario recibe una máquina virtual con recursos dedicados que está lista para desplegar y configurar su aplicación o sitio web. De este modo, los clientes que utilizan el alojamiento VPS pueden centrarse en sus aplicaciones o sitios web sin tener que perder tiempo y energía ocupándose de los servidores físicos que alojan su código. Los proveedores de alojamiento VPS ofrecen un rendimiento seguro, fiable y constante para sus sitios web.



Figura 4.9: VPS

## 4.17. Peticiones HTTP

Las peticiones HTTP (HyperText Transfer Protocol) son solicitudes que los clientes (como navegadores web, aplicaciones móviles, etc.) envían a los servidores para recibir o enviar información a través de la web. HTTP es el protocolo fundamental utilizado para

la transmisión de datos. Se refiere a cualquier componente o módulo adicional que se integra con el microcontrolador para realizar tareas específicas. Estos periféricos pueden ser internos al chip o externos y se conectan a través de diversos interfaces.

## 4.18. Pluma.php

Pluma.php probablemente sea un archivo PHP ubicado en un servidor web. PHP es un lenguaje de programación ampliamente utilizado para desarrollar aplicaciones web dinámicas. El archivo pluma.php podría contener código que procesa la solicitud HTTP que se le envía, realiza operaciones en una base de datos, genera una respuesta y la devuelve al cliente que realizó la solicitud. En este caso específico, el archivo pluma.php parece estar diseñado para recibir datos enviados desde un dispositivo y procesarlos de alguna manera.

## 4.19. MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligero y de baja latencia diseñado para dispositivos y redes con recursos limitados, especialmente en entornos de Internet de las Cosas (IoT). Fue desarrollado por IBM y luego estandarizado por OASIS (Organization for the Advancement of Structured Information Standards)

- Publicar/Suscribir: Utiliza un modelo de comunicación publicar/suscribir (publish/-subscribe) en lugar de un modelo tradicional cliente/servidor. Esto significa que los clientes pueden publicar mensajes a un tema otros clientes pueden suscribirse a esos

temas para recibir mensajes.

- Bajo Ancho de Banda: Está diseñado para ser eficiente en el uso del ancho de banda y la batería, lo que lo hace ideal para dispositivos con limitaciones de recursos y redes con alta latencia o baja fiabilidad.

### **Calidad de Servicio (QoS)**

MQTT ofrece tres niveles de calidad de servicio para la entrega de mensajes:

- QoS 0 (Al máximo una vez): El mensaje es enviado al receptor una sola vez y no se requiere confirmación de recepción. Este nivel es el más rápido pero menos fiable.
- QoS 1 (Al menos una vez): El mensaje es enviado al receptor al menos una vez, con confirmación de recepción. Puede ocurrir que el mensaje se entregue más de una vez.
- QoS 2 (Exactamente una vez): El mensaje es entregado exactamente una vez mediante un proceso de confirmación en dos pasos. Este nivel es el más seguro pero también el más lento.

### **Componentes Principales**

- Broker: El broker MQTT actúa como intermediario para todos los mensajes que pasan entre los clientes. Es responsable de recibir todos los mensajes, filtrarlos por tema, y distribuirlos a los suscriptores adecuados.
- Clientes: Los dispositivos o aplicaciones que se conectan al broker. Los clientes pueden ser tanto publicadores como suscriptores de mensajes.

### **Clientes**

- Los clientes se conectan al broker MQTT usando un protocolo TCP/IP. Para conexiones seguras, se puede usar SSL/TLS.
- Durante la conexión, los clientes envían un mensaje CONNECT al broker, que incluye información de autenticación (si es necesario).

### **Publicación de mensajes**

- Un cliente publica un mensaje a un tema específico enviando un mensaje PUBLISH al broker.
- Un tema es una cadena jerárquica que se usa para categorizar los mensajes, como `sensors/temperature` o `home/livingroom/light`.

### **Distribución de mensajes**

- El broker recibe el mensaje publicado y lo distribuye a todos los clientes que están suscritos al tema correspondiente.
- Publish es una acción clave en el protocolo MQTT, que se refiere a la publicación de mensajes por parte de un cliente a un tema específico. En el contexto de MQTT, publicar significa que un cliente envía un mensaje al broker, especificando un tema, para que el broker distribuya ese mensaje a otros clientes que están suscritos a ese tema.

## **4.20. Mosquitto**

Mosquitto es un broker de mensajes de código abierto que implementa el protocolo MQTT (Message Queuing Telemetry Transport). MQTT es un protocolo de mensajería

ligero y eficiente, diseñado para conectar dispositivos en redes con limitaciones de ancho de banda y alta latencia. Este protocolo es ampliamente utilizado en aplicaciones de Internet de las Cosas (IoT) para comunicación entre dispositivos y sensores. Mosquitto es un broker MQTT que facilita la comunicación entre dispositivos mediante un modelo de publicación/suscripción. En lugar de que los dispositivos se comuniquen directamente entre sí, utilizan un broker (Mosquitto) como intermediario. Los dispositivos se conectan al broker y pueden publicar mensajes en temas o suscribirse a esos temas para recibir mensajes.

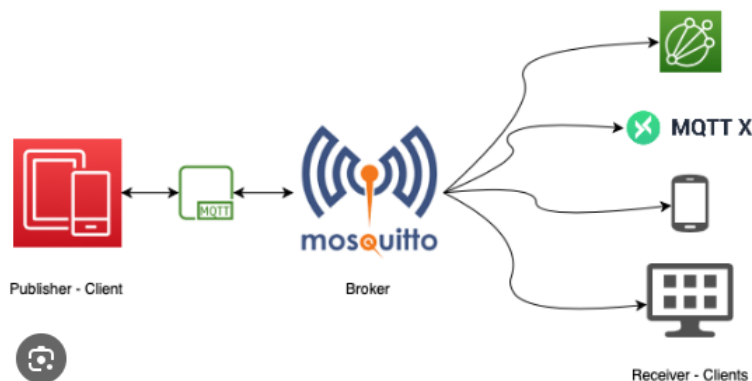


Figura 4.10: Mosquitto

## 4.21. Base de datos

Una base de datos es un sistema organizado de recopilación, almacenamiento y administración de datos de manera que se pueden acceder, gestionar y actualizar fácilmente. Las bases de datos permiten almacenar grandes cantidades de información de forma estructurada, lo que facilita su consulta y manipulación.

- Estructura : Los datos en una base de datos se organizan en tablas, que consisten en filas y columnas. Cada fila representa un registro único, y cada columna representa un campo específico de datos dentro del registro.
- Sistema de Gestión de Bases de Datos (DBMS) : Es un software que se utiliza para crear, gestionar y manipular bases de datos. Los ejemplos de DBMS incluyen MySQL, PostgreSQL, Oracle, Microsoft SQL Server y SQLite.
- Lenguaje de Consulta : Para interactuar con la base de datos, se utiliza un lenguaje de consulta, siendo el más común el SQL (Structured Query Language).

### **Tipos de base de datos**

- Relacionales : Utilizan una estructura de tablas con relaciones entre ellas. Ejemplos: MySQL, PostgreSQL, Oracle.
- NoSQL : Diseñadas para manejar grandes volúmenes de datos no estructurados o semiestructurados.
- Orientadas a Objetos : Almacenan datos en forma de objetos, como en la programación orientada a objetos.
- Bases de Datos en Memoria : Almacenan datos en la memoria principal para acceso rápido
- Operaciones Básicas : Las operaciones básicas que se pueden realizar en una base de datos incluyen Crear (Create), Leer (Read), Actualizar (Update) y Eliminar (Delete), combinadas conocidas por sus siglas en inglés CRUD.

## 4.22. Base de datos a utilizar

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto basado en el lenguaje de consulta estructurado (SQL). Es uno de los RDBMS más populares y es ampliamente utilizado en aplicaciones web, software empresarial y proyectos de código abierto. Código Abierto: MySQL es gratuito y de código abierto, aunque también ofrece versiones comerciales con características adicionales y soporte técnico. MySQL es conocido por su alto rendimiento y capacidad para manejar grandes volúmenes de datos y altas tasas de transacciones. MySQL puede manejar múltiples usuarios accediendo y manipulando la base de datos simultáneamente sin conflictos.



Figura 4.11: Base de datos (MYSQL)



# Capítulo 5

## Análisis y Diseño

El desarrollo del capítulo abarca la presentación de conceptos vinculados a la metodología de desarrollo de software aplicada.

### 5.1. Análisis de Requerimientos

Creamos el código en arduino IDE, lo primero que hicimos fue hacer el código que iba a hacer que el RFID leyera las credenciales:

Cuando el RFID lee la credencial lo único que hace es mostrar en el monitor 7 dígitos los cuales son el identificador de la credencial. Este código utiliza un lector de tarjetas RFID para leer códigos RFID y luego los envía a través de una conexión HTTP. Lo primero es configurar las librerías necesarias para manejar el lector de tarjetas RFID y la conexión HTTP. En la función `setup()`, se inicializa la comunicación serial y se prepara el lector de tarjetas RFID. En la función `loop()`, se verifica si hay una tarjeta RFID presente. Si se detecta una tarjeta, se lee el código RFID y se envía a través de una conexión HTTP a

un servidor. Si no se detecta ninguna tarjeta, el programa espera hasta que se presente una tarjeta. Una vez hecho esto y el aver entendido el código lo siguiente que realizamos fue desarrollar el código el cual iba hacer que el ESP32 se conectara a internet, el cual fue el siguiente:

```
RFID_COMPLETO.ino
1  #include <WiFi.h>
2  #include <Wiegand.h>
3  #include <HTTPClient.h>
4
5  const char *ssid = "TPLINK5";
6  const char *password = "Luisrey2024";
7
8  WIEGAND wg;
9  void WiFi_Setup()
10 {
11     Serial.print("Conectando a ");
12     Serial.println(ssid);
13     WiFi.begin(ssid, password);
14     int contador = 0;
15     while (WiFi.status() != WL_CONNECTED && contador < 20 ) {
16         delay(500);
17         Serial.print(".");
18         contador += 1;
19     }
20     if (WiFi.status() == WL_CONNECTED) {
21         Serial.println("");
22         Serial.println("WiFi conectado");
23     } else {
24         ESP.restart();
25     }
26     wg.begin();
27 }
```

Figura 5.1: Comunicación RFID

Se incluye la librería `WiFi.h` que permite al dispositivo conectarse a una red WiFi. Se definen las variables `ssid` y `password` que contienen el nombre de la red WiFi y la contraseña, respectivamente. En la función `setup()`, se inicia la comunicación serial y se espera un segundo. Se imprime un mensaje indicando que el dispositivo está intentando conectarse a la red WiFi. Se utiliza `WiFi.begin(ssid, password)` para intentar conectarse a la red WiFi con el nombre y la contraseña proporcionados. Se utiliza un bucle `while` para esperar a que la conexión se establezca. Mientras la conexión esté en proceso, se

imprime un mensaje indicando que se está intentando conectar y se espera un segundo. Una vez que la conexión se establece, se imprime un mensaje indicando que el dispositivo está conectado a la red WiFi y se muestra la dirección IP asignada al dispositivo en la red. Comprendiendo estos codigos juntamos los dos agregando otra parte para completarlo lo cual fue lo siguiente:

```
void loop()
{
  if (wg.available() )
  {
    HTTPClient http;
    String RFID = String (wg.getCode());
    String datos_a_enviar = "rfid=" + RFID;
    Serial.println(datos_a_enviar);
    http.begin("http://141.148.173.116/pluma.php");
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
    int codigo_respuesta = http.POST(datos_a_enviar);
    if (codigo_respuesta > 0)
    {
      if (codigo_respuesta == 200)
      {
        String cuerpo_respuesta = http.getString();
        Serial.println(cuerpo_respuesta);
      }
    }
    else
    {
      Serial.println(codigo_respuesta);
    }
    http.end();
  }
}
```

Figura 5.2: Comunicación RFID

La finalidad de este fue mandar el identificador de la credencial leída al VPS para que posteriormente si el código de respuesta es 200, significa que la solicitud fue exitosa y el servidor ha devuelto una respuesta.

## 5.2. Aplicación del protocolo mqtt en Arduino IDE

Esta función se llama cuando se recibe un mensaje MQTT. Convierte el mensaje recibido en un entero (resultI) y lo asigna a la variable var. También construye una cadena resultS a partir del mensaje recibido.

```
void callback(char* topic, byte* payload, unsigned int length) {  
    Serial.print("Mensaje recibido [");  
    Serial.print(topic);  
    Serial.print("] ");  
  
    char payload_string[length + 1];  
  
    int resultI;  
  
    memcpy(payload_string, payload, length);  
    payload_string[length] = '\0';  
    resultI = atoi(payload_string);  
  
    var = resultI;  
  
    resultS = "";  
  
    for(int i=0;i<length;i++){  
        resultS= resultS + (char)payload[i];  
    }  
  
    Serial.println();  
}
```

Figura 5.3: Función Callback

Esta función intenta reconectar al broker MQTT si la conexión se pierde.

```

void reconnect(){
  while (!mqttClient.connected()){
    Serial.print ("Intentando conectarse MQTT...");

    if (mqttClient.connect("arduinoClient"))
    {
      Serial.println("Conectando");
      mqttClient.subscribe("Entrada/01");
    }
    else {
      Serial.print("Fallo, rc=");
      Serial.print(mqttClient.state());
      Serial.println("Intentar de nuevo en 5 segundos");
      //Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

```

Figura 5.4: Función para Reconectar al Broker MQTT

Una vez teniendo en cuenta que esta el programa en IDE, y el hacer las pruebas, nos dimos cuenta que al incluir el broker es mas complejo de identificar las rfid y/o matriculas subscritas los topicos especificados.

### 5.3. Sistema de Control RFID

El sistema de RFID es un sistema que es similar al de los c'ódigos de barras tradicionales, pero tienen grandes ventajas frente a esta tecnología. Esta tecnología no requiere identificar una etiqueta de un producto mediante una imagen, la tecnología RFID utiliza ondas de radio con las que se comunican con un microchip, el cual se puede montar en muchos soportes tales como tag o etiqueta RFID, una tarjeta o un transpondedor, su lectura es instantánea, a distancia y no se requiere una línea de visión, todo esto permite que los operarios pueden trabajar con más libertad y una mayor eficiencia, también alarga la vida útil de las etiquetas, porque se tendrá una lectura completamente sin contacto. Los controles de entradas y salidas es lo más común para el uso de la tecnología RFID. Esto permitirá que se pueda controlar la entrada y salida de todo tipo de elementos y personas, desde los productos terminados hasta los semielaborados. También permite que se controle la entrada y salida de las instalaciones o eventos.

### 5.4. Mayor productividad + Mayor eficiencia = Beneficio

Con las soluciones RFID, al automatizar procesos de lecturas de etiquetas, se puede reducir el número de personas necesarias para gestionar tu almacén. También permiten que los tiempos que pueden gastar los empleados en la identificación de los productos y mercancías se reduzcan considerablemente. Además, al ser un sistema de identificación inteligente, es más fiable que otros sistemas y los errores de lectura serán mínimos. Las

incidencias en la manipulación de los stocks y preparación de pedidos serán muy bajas. Todo esto implica una gran mejora en la productividad y en la eficiencia que tendrá tu almacén.

Esto te permitirá que se pueda dedicar más tiempo a otras tareas que aporten un mayor valor a tu empresa, porque la intervención humana en los procesos serva menor. Esto permite controlar el número de personas que entran, así como para evitar que la entrada con un acceso falso pueda acceder.

## 5.5. MQTT

MQTT es un protocolo de mensajería basado en estándares, o un conjunto de reglas, que se utiliza para la comunicación de un equipo a otro. Los sensores inteligentes, los dispositivos portátiles y otros dispositivos de Internet de las cosas (IoT) generalmente tienen que transmitir y recibir datos a través de una red con recursos restringidos y un ancho de banda limitado. Estos dispositivos IoT utilizan MQTT para la transmisión de datos, ya que resulta fácil de implementar y puede comunicar datos IoT de manera eficiente. MQTT admite la mensajería entre dispositivos a la nube y la nube al dispositivo. MQTT se ejecuta sobre TCP/IP utilizando una topología PUSH/SUBSCRIBE. En la arquitectura MQTT, existen dos tipos de sistemas: clientes y brókeres. Un bróker es el servidor con el que se comunican los clientes: recibe comunicaciones de unos y se las envía a otros. Los clientes no se comunican directamente entre sí, sino que se conectan con el bróker. Cada cliente puede ser un editor, un suscriptor o ambos. MQTT es un protocolo controlado por eventos, donde no hay transmisión de datos periódica o continua. Así se mantiene

el volumen de transmisión al mínimo. Un cliente sólo publica cuando hay información para enviar, y un bróker sólo envía información a los suscriptores cuando llegan nuevos datos. Para mantener el protocolo al mínimo, sólo se pueden efectuar cuatro acciones en cualquier comunicación: publicar, suscribirse, cancelar suscripción o hacer ping.

**Publicar:** Envía un bloque de datos que contiene el mensaje que se va a enviar. Estos datos son específicos de cada implementación, pero pueden ser algo tan simple como una indicación de encendido/apagado o un valor de un determinado sensor, como temperatura, presión, etc. En el caso de que el tema que se está publicando no exista, este se crea en el bróker.

**Suscribirse:** Convierte a un cliente en suscriptor de un tema. Se puede suscribir a temas en concreto o mediante comodines, que permiten suscripciones a toda una rama de temas o a parte de ella. Para suscribirse, un cliente envía un paquete SUBSCRIBE y recibe un paquete SUBACK a cambio. Si hay un mensaje retenido para el tema, el nuevo suscriptor también lo recibe.

**Ping:** Un cliente puede hacer ping al bróker. El suscriptor envía un paquete PINGREQ y, como respuesta, se recibe un paquete PINGRESP. Se pueden utilizar pings para garantizar que la conexión siga funcionando y que la sesión TCP no haya sido cerrada inesperadamente por otro equipo de red, como un router o una puerta de enlace.

**Desconectar:** Un suscriptor o editor puede enviar un mensaje de DISCONNECT al bróker. Este mensaje informa al bróker de que ya no necesitará enviar o poner en cola mensajes para un suscriptor y que ya no recibirá datos de un editor. Este tipo de cierre permite al cliente volver a conectarse utilizando la misma identidad de cliente que en ocasiones anteriores. Cuando un cliente se desconecta sin enviar un mensaje de desconexión,



se envía su última voluntad y testamento a los suscriptores.

## 5.6. PHP

El lenguaje PHP, es un lenguaje de programación que permite desarrollar webs y aplicaciones web dinámicas, PHP, que es un lenguaje flexible, se integra perfectamente con HTML, lo que es esencial para entender qué es PHP en programación, siempre siguiendo algunas reglas establecidas, además, el lenguaje PHP favorece a la conexión entre el servidor y la interfaz del usuario.

PHP es un lenguaje versátil que se utiliza en una amplia gama de aplicaciones en la programación PHP. ¿Qué significa que sea un lenguaje de código abierto? Un lenguaje de código abierto permite ser modificado por desarrolladores y se adapta a diversos proyectos. Esto demuestra la importancia de entender qué es PHP en programación.

### 5.6.1. Características al utilizar PHP:

- Documentación y plantillas: Este lenguaje, al ser abierto y gratuito, permite el desarrollo y crecimiento de una comunidad que, además de ser muy activa, ofrece sus conocimientos al momento de trabajar con PHP con la comunidad.
- Oferta laboral: El lenguaje permite el desarrollo de sitios webs complejos y dinámicos. Por ello, cada vez son más las empresas e instituciones que requieren de este servicio en particular para sus páginas y plataformas.
- Orientado a objetos: Permite que el procesamiento de datos sea mucho más rápido

debido a que los scripts en métodos.

- Separación de estructuras: Este lenguaje permite una separación de códigos. Es decir, mientras que uno se encarga de mover y actualizar datos, otros se quedan arraigados a la interfaz del sitio. Permitiendo un código mucho más limpio y estable.
- Libertad de operación: Como se mencionó anteriormente, PHP es un lenguaje que puede ser ejecutado en casi cualquier servidor o sistema operativo. El único requisito es que el computador pueda ejecutar código sin problema.

### **5.6.2. Ventajas al utilizar PHP:**

- Es un lenguaje de código abierto: esto implica que cualquier programador tiene acceso a él, sin costos extras.
- Es fácil de aprender: PHP es un lenguaje cuya curva de aprendizaje es baja y comprensible. Su red de tutoriales, videos, códigos documentados y gran comunidad simplifican el proceso de aprendizaje.
- Es fácil para instalarse en los servidores o sistemas operativos: cada entorno de desarrollo posee un alto porcentaje de configuración, y el acceso a la base de datos es rápido y sencillo.
- Se adapta al lenguaje HTML, con respecto al desarrollo de páginas web y aplicaciones complejas, este lenguaje es perfecto, pudiendo adaptarse al lenguaje HTML, para crear un código mucho más sólido y dinámico.
- Soporta gran cantidad de datos: una de las ventajas de PHP es que permite procesar

de forma inteligente y con extrema eficiencia los registros de datos generados por usuarios al interactuar con la aplicación.

- Permite automatizar tareas: existen algunas tareas que los developers tienen que hacer cuando desarrollan una web que pueden ser automatizadas fácilmente con este lenguaje.

### **5.6.3. Desventajas al utilizar PHP:**

- Aprender sobre framework de PHP, si bien tener entornos de trabajo o frameworks permite incluir funciones sin escribir código adicional, para un desarrollador esto supone tener que aprender las utilidades del framework.
- Mala calidad de manejo de errores, una de las principales desventajas de PHP es que no tiene herramientas de depuración, que son necesarias para buscar errores y advertencias.
- Seguridad baja, PHP es uno de los lenguajes de programación menos seguros. Para garantizar la seguridad de los datos, debes tener conocimientos técnicos profundos y/o trabajar con SQL.
- Se necesita un servidor web, para ejecutar el software en Internet hay que contar con un hosting.

**¿Cuáles son las ventajas de PHP y MySQL juntos?** MySQL proporciona la parte de base de datos, mientras que PHP proporciona la parte de aplicación de la base de datos web.

## 5.7. Aplicación del protocolo mqtt en Visual Studio Code

En línea `require("vendor/autoload.php")`, incluye el archivo de autoloading desde el directorio "vendor", que este se utiliza para cargar automáticamente las clases y funciones necesarias el proyecto PHP. (El autoloading es una práctica común en proyectos modernos para evitar la necesidad de incluir manualmente cada archivo de clase, ya que este lo que hace es hacerlo automáticamente.), (`use /Bluerhinos/phpMQTT`) En la línea 5 se declara el espacio de nombres ("namespace") y se importa la clase "phpMQTT" del espacio de nombres (`/Bluerhinos`). Después se crea una clase "phpMQTT" con los parámetros: ("Sserver"): La dirección IP o el nombre de host del servidor MQTT. En la línea 26 se muestra la función "procmgs", esta función se encargará de manejar los mensajes entrantes en los usuarios suscritos a los topics.

```
1 <?php
2
3 require('vendor/autoload.php');
4
5 use \Bluerhinos\phpMQTT;
6
7 // Conexion MQTT
8 $server      = '192.168.50.75';
9 $port        = 1883;
10 $username    = 'luismqtt';
11 $password    = '2024';
12 $clientId    = 'phpMQTT-' . rand(5,15);
13 $topicPublish = 'test';
14 $topicSubscribe = 'test';
15
16 $c = new phpMQTT($server, $port, 'hola', null);
17
18 $conn_result = $c -> connect(false, null, $username, $password);
19
20 if ($conn_result) {
21     $c->publish($topicPublish, 'ya salio', 0, false);
22     $topics[$topicSubscribe] = array("qos">=0, "function">="procmgs");
23     $c ->subscribe($topics);
24 }
25
26 // Función para manejar mensajes entrantes
27 function procmgs($topic, $msg){
28     echo 'topic: '.$topic.' msg: '.$msg;
29
30     // $json = json_decode($msg, true);
31     $json = json_decode($msg);
32 }
```

Figura 5.5: Código Parte 1 Realizado en Visual Studio Code

En la siguiente imagen se observan las variables `"host,dbname"`, `username` y `password`, estas almacenan los detalles de la conexión a la base de datos MySQL. Estos valores se utilizan para establecer una conexión con el servidor MySQL. Después están las funciones `foo()`. en la línea 2. La función `foo()`. es una función definida por el usuario. Acepta tres argumentos, (`arg1`, `arg2`, `arg3`), aunque no se utilizan en esta función. La función `bar()` ubicada en la línea 11 también es una función definida por el usuario. Imprime. Existo desde el momento inmediato que comenzó el programa. Se llama inmediatamente después de su definición.

```
$host      = 'localhost';
$dbname    = 'integrador';
$username  = 'root';
$password  = '';

try {
    $conn = new mysqli($host, $username, $password, $dbname);

    echo "Connected to $dbname at $host successfully";
    $id=null;

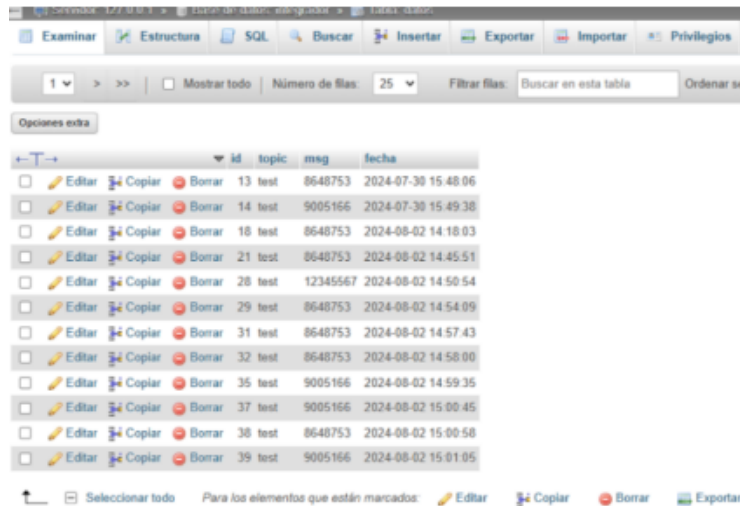
    $query1 = "insert into datos (topic,msg)values('$topic','$msg')";
    echo $query1;
    if(!$conn->query($query1))
    {
        printf("Error: $query1 <br> $conn->error");
    }

    $conn = null;
} catch (PDOException $error) {
    die("Could not connect to the DB: ". $error->getMessage());
}
```

Figura 5.6: Código Parte 2 Realizado en Visual Studio Code

## 5.8. Desarrollo de la Base de Datos

A continuación se muestra el avance de lo realizado en la Base de Datos. En esta primer tabla llamada DATOS se muestra el topic, el mensaje que envia que en este caso es el RFID, y la fecha y hora en la que se pasa la credencial. En la segunda tabla llamada ESTUDIANTES solo se hace la relación de los RFID de las credenciales, con el numero de matricula, para que al pasar la credencial por el lector, mande llamar el numero de matricula, y asi, que se aparezca la fotografia, el nombre, fecha y hora, en pantalla.



	id	topic	msg	fecha
<input type="checkbox"/>	13	test	8648753	2024-07-30 15:48:06
<input type="checkbox"/>	14	test	9005166	2024-07-30 15:49:38
<input type="checkbox"/>	18	test	8648753	2024-08-02 14:18:03
<input type="checkbox"/>	21	test	8648753	2024-08-02 14:45:51
<input type="checkbox"/>	28	test	12345567	2024-08-02 14:50:54
<input type="checkbox"/>	29	test	8648753	2024-08-02 14:54:09
<input type="checkbox"/>	31	test	8648753	2024-08-02 14:57:43
<input type="checkbox"/>	32	test	8648753	2024-08-02 14:58:00
<input type="checkbox"/>	35	test	9005166	2024-08-02 14:59:35
<input type="checkbox"/>	37	test	9005166	2024-08-02 15:00:45
<input type="checkbox"/>	38	test	8648753	2024-08-02 15:00:58
<input type="checkbox"/>	39	test	9005166	2024-08-02 15:01:05

Figura 5.7: Tabla 1 Datos.

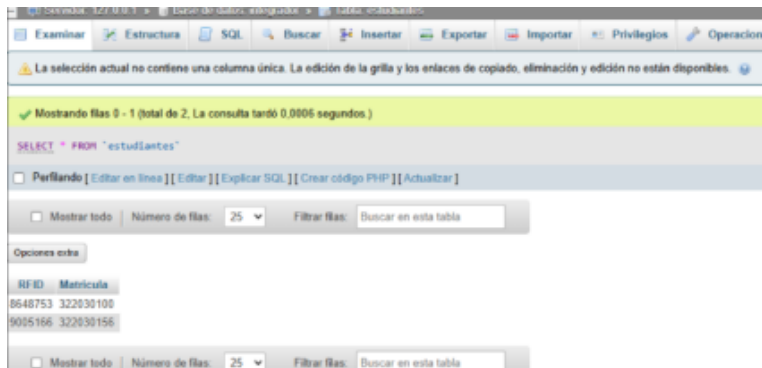


Figura 5.8: Tabla 2 Estudiantes.

## 5.9. DIAGRAMA DE LO REALIZADO EN EL PARCIAL II

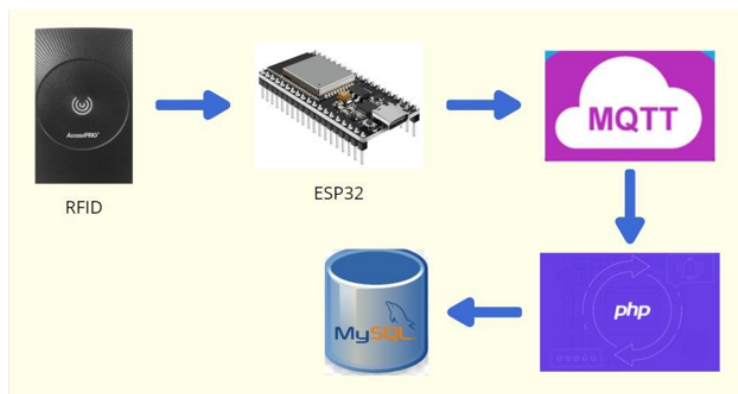


Figura 5.9: Trabajo realizado.

## Capítulo 6

# Migración e Implementación del sistema de control en Oracle Cloud

### 6.1. Migración a Oracle Cloud

Para el manejo del sistema de control por RFID para los laboratorios de la Universidad Politécnica de Juventino Rosas, se ha decidido migrar la infraestructura existente a Oracle Cloud. Este proceso implica trasladar el sistema actual a una máquina virtual, la cual nos proporcionará un entorno controlado.



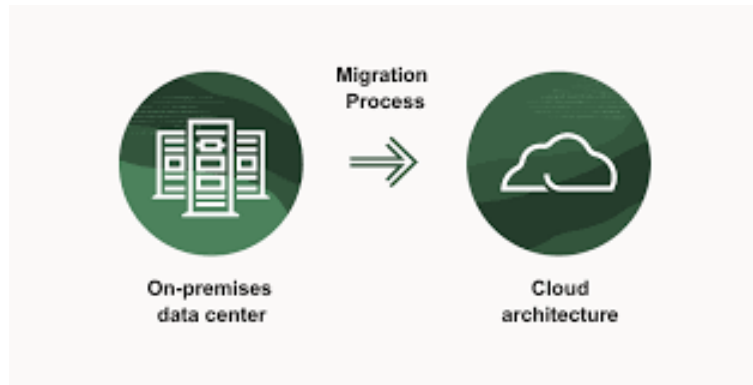


Figura 6.1: Comunicación RFID

### 6.1.1. Oracle Cloud

Oracle Cloud es una plataforma de computación en la nube desarrollada por Oracle Corporation que ofrece una amplia gama de servicios y herramientas diseñadas para empresas y desarrolladores. Su propósito principal es proporcionar infraestructura, aplicaciones y servicios para alojar, gestionar y analizar datos y aplicaciones de manera eficiente, escalable y segura.

### 6.1.2. Características principales de Oracle Cloud:

- Infraestructura como Servicio (IaaS):

Proporciona recursos virtuales como servidores, almacenamiento y redes para que las empresas puedan alojar sus aplicaciones y bases de datos.

- Plataforma como Servicio (PaaS):

Ofrece herramientas para desarrollar, probar y desplegar aplicaciones. Incluye servicios como bases de datos, análisis de datos y gestión de APIs.

- Software como Servicio (SaaS):

Permite utilizar aplicaciones empresariales como ERP, CRM y HCM sin necesidad de instalarlas en servidores locales.

- Escalabilidad y Elasticidad:

Los recursos pueden ajustarse en tiempo real según las necesidades del proyecto, lo que permite un uso eficiente de los costos.

### **6.1.3. Ventajas de usar Oracle Cloud:**

- Alto rendimiento:

Ideal para manejar grandes volúmenes de datos y ejecutar aplicaciones críticas.

- Compatibilidad:

Soporta múltiples lenguajes de programación y sistemas operativos.

- Innovación tecnológica:

Integra herramientas avanzadas como inteligencia artificial, machine learning y análisis de datos.

### **6.1.4. Justificación de la migración**

La decisión de migrar el sistema de control basado en tecnología RFID a Oracle Cloud se fundamenta en las siguientes razones:

- El sistema RFID genera una cantidad significativa de datos relacionados con accesos y uso de laboratorios. Oracle Cloud permite escalar los recursos, como almacenamiento y capacidad de procesamiento, en función de las necesidades actuales y

futuras del proyecto. Esta escalabilidad asegura que el sistema pueda adaptarse al crecimiento de la institución, tanto en número de usuarios como en complejidad de operaciones.

- Oracle Cloud proporciona niveles avanzados de seguridad, como cifrado de datos, controles de acceso y auditorías. Esto es esencial para proteger la información sensible de estudiantes, personal y actividades dentro de los laboratorios. La seguridad integral minimiza el riesgo de accesos no autorizados y garantiza el cumplimiento de normativas de protección de datos.
- Al migrar a la nube, se garantiza que el sistema esté disponible 24/7 sin interrupciones, incluso en caso de fallas en la infraestructura local. Oracle Cloud ofrece opciones de recuperación ante desastres, replicación de datos y redundancia geográfica, asegurando que el sistema RFID funcione de manera ininterrumpida.
- Con Oracle Cloud, las bases de datos pueden ser gestionadas y optimizadas automáticamente. Esto reduce los tiempos de consulta y mejora la experiencia del usuario, especialmente al manejar grandes volúmenes de datos generados por el sistema RFID.
- Al estar en la nube, el sistema RFID puede ser gestionado desde cualquier lugar con conexión a internet, facilitando el monitoreo y la administración por parte del personal autorizado.

### **6.1.5. Preparación para la migración**

La migración de un sistema local a Oracle Cloud requiere una planificación detallada para garantizar que los datos y las aplicaciones se transfieran correctamente, minimizando riesgos y tiempos de inactividad. A continuación, se describen los pasos clave en la preparación para la migración del sistema RFID. Los puntos a considerar fueron:

- Identificar los componentes clave del sistema RFID, como la base de datos, las aplicaciones relacionadas y las integraciones existentes.
- Revisar la estructura de la base de datos actual y asegurarse de que sea compatible con los servicios de Oracle Cloud.

### **6.1.6. Configuración del entorno en Oracle Cloud**

- Creamos una cuenta de Oracle Cloud: Configuramos una cuenta con acceso a los servicios necesarios.
- Asignación de recursos: Dimensionar recursos de almacenamiento, memoria y procesamiento de acuerdo con el análisis previo.
- Configuración de seguridad: Definir reglas de acceso, políticas de encriptación y mecanismos de autenticación para proteger los datos.

### **6.1.7. Proceso de migración**

El proceso de migración del sistema RFID comenzó con la creación de la base de datos en Oracle Cloud y continuó con una serie de pasos estratégicos para garantizar una transición fluida y exitosa.

- Durante esta etapa, se definieron las tablas necesarias para replicar la estructura del sistema existente, como:
  - Tabla donde se relaciona la matricula y Rfid de los estudiantes
  - Tabla donde se almacena el RFID, la hora y fecha de entrada

La segunda etapa del proceso de migración implicó la subida y configuración de los programas necesarios para que el sistema RFID funcionara directamente en el entorno de Oracle Cloud. Se identificaron los programas críticos del sistema RFID que debían ejecutarse en la nube. Estos incluyen:

- Aplicaciones encargadas de registrar accesos mediante RFID.
- Interfaces para la gestión de usuarios y configuraciones del sistema. Scripts o módulos que interactúan directamente con la base de datos para consultas y actualizaciones.
- Se realizaron pruebas locales para garantizar que el código estuviera actualizado y compatible con el entorno de Oracle Cloud.

#### **6.1.8. Transferencia de los programas a la nube**

- Subida de archivos: Los archivos de las aplicaciones se comprimieron y se subieron al servidor en Oracle Cloud.
- Despliegue: Se descomprimieron los archivos en las ubicaciones apropiadas dentro del servidor en la nube.

Se configuraron las rutas de acceso para que los programas pudieran ejecutarse correctamente.

## 6.2. Aplicación Móvil

Una aplicación móvil en Flutter puede integrar el protocolo MQTT (Message Queuing Telemetry Transport) para la comunicación en tiempo real. MQTT es un protocolo ligero de mensajería utilizado en aplicaciones IoT y otros sistemas donde se requiere una comunicación eficiente entre dispositivos.

## 6.3. Configuración de MQTT

El cliente MQTT en la aplicación Flutter se conecta a un servidor llamado *broker*. Este broker gestiona la comunicación entre dispositivos. Para integrar MQTT en Flutter, se utiliza la librería `mqtt_client`.

### 6.3.1. Dependencias

Se debe agregar la dependencia `mqtt_client` en el archivo `pubspec.yaml`

### 6.3.2. Conexión al Broker

El cliente MQTT se conecta al broker utilizando parámetros como la URL del broker, el puerto y las credenciales necesarias.

## 6.4. Suscripción a Temas

Una vez conectado al broker, la aplicación se suscribe a uno o más *temas* (topics) para recibir mensajes en tiempo real. Los temas son canales a través de los cuales se envían y reciben los mensajes.

## 6.5. Recepción de Mensajes

Cuando el cliente recibe un mensaje, se debe definir un *callback* que manejará el mensaje recibido y tomará las acciones necesarias, como actualizar la interfaz de usuario (UI).

## 6.6. Publicación de Mensajes

La aplicación también puede publicar mensajes en un tema para enviar datos a otros dispositivos o aplicaciones.

## 6.7. Desconexión del Broker

Cuando la aplicación ya no necesita recibir ni enviar mensajes, debe desconectarse del broker de manera segura.

## 6.8. Manejo de Errores y Reconexión

Es importante manejar los errores de conexión, como las desconexiones inesperadas. MQTT permite reconectar el cliente automáticamente o manualmente si la conexión se pierde, garantizando que el cliente continúe recibiendo los mensajes en tiempo real.

## 6.9. Flujo General

El flujo de la aplicación es el siguiente:

1. La aplicación se conecta al broker MQTT.
2. Se suscribe a uno o más temas para recibir mensajes en tiempo real.
3. La aplicación recibe los mensajes y toma las acciones necesarias, como actualizar la interfaz de usuario.
4. También puede publicar mensajes en temas específicos para interactuar con otros dispositivos o usuarios.
5. Finalmente, cuando la comunicación ya no es necesaria, la aplicación se desconecta del broker.

Este flujo permite a la aplicación Flutter gestionar la comunicación bidireccional en tiempo real a través de MQTT, ideal para aplicaciones de monitoreo, control de dispositivos IoT y notificaciones en tiempo real.



## Capítulo 7

# Desarrollo de un Sistema Web y Aplicación Móvil para la Gestión de Registros mediante Tecnología RFID

En este documento se describe el diseño y desarrollo de un sistema web basado en PHP, cuyo objetivo principal es gestionar el acceso y monitoreo de registros mediante tecnología RFID en los laboratorios de la Universidad. Este sistema permitirá a los usuarios autenticarse, consultar registros en tiempo real y generar reportes en formato PDF.

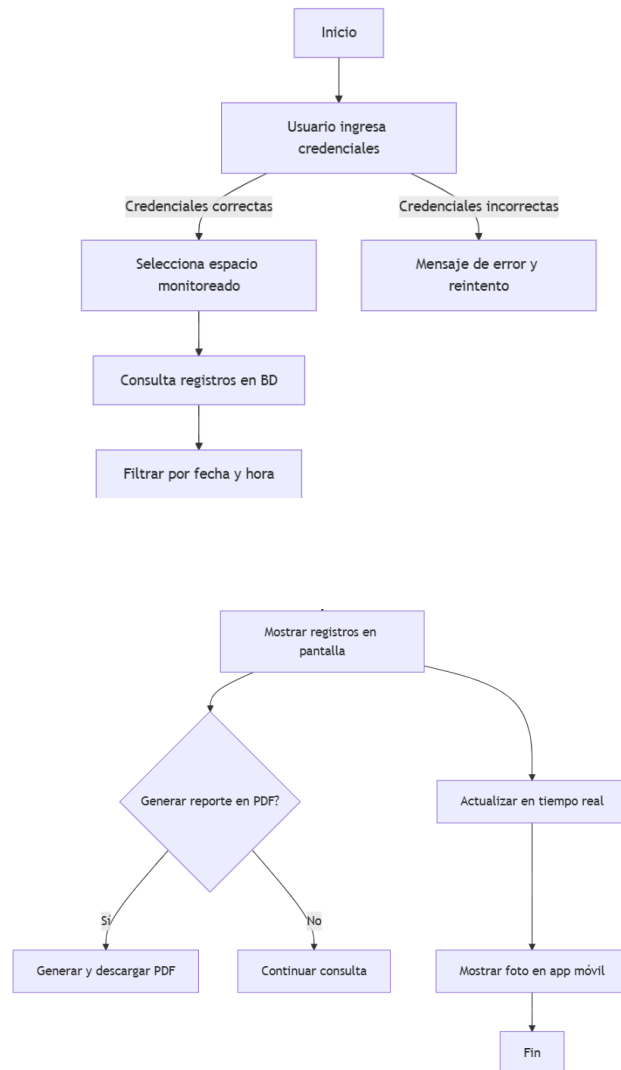


Figura 7.1: Diagrama de flujo

## 7.1. Tecnologías Utilizadas

Para el desarrollo del sistema, se han utilizado las siguientes tecnologías:

- **PHP:** Lenguaje de programación para la implementación del backend.

- **MySQL:** Base de datos utilizada para el almacenamiento de registros.
- **HTML, CSS y JavaScript:** Para la creación de la interfaz de usuario.
- **FPDF:** Librería en PHP para la generación de reportes en formato PDF.
- **Flutter:** Para el desarrollo de la aplicación móvil.
- **MQTT:** Protocolo de comunicación para la transferencia de datos en tiempo real.

## 7.2. Página web

La página web será desarrollada en Visual Studio Code y utilizará PHP como lenguaje de programación principal, complementado con bases de datos para el almacenamiento y gestión de información. Se implementará un sistema de autenticación para garantizar el acceso seguro a la plataforma y permitir a los usuarios gestionar los registros de manera eficiente.

## 7.3. Arquitectura del Sistema

El sistema se compone de varios módulos interconectados:

- **Módulo de autenticación:** Permite el inicio de sesión de usuarios.
- **Módulo de gestión de accesos:** Administra los permisos de los usuarios.
- **Módulo de monitoreo:** Permite visualizar los accesos registrados.
- **Módulo de generación de reportes:** Facilita la exportación de datos en formato PDF.

## Funciones principales

### 7.4. Página principal

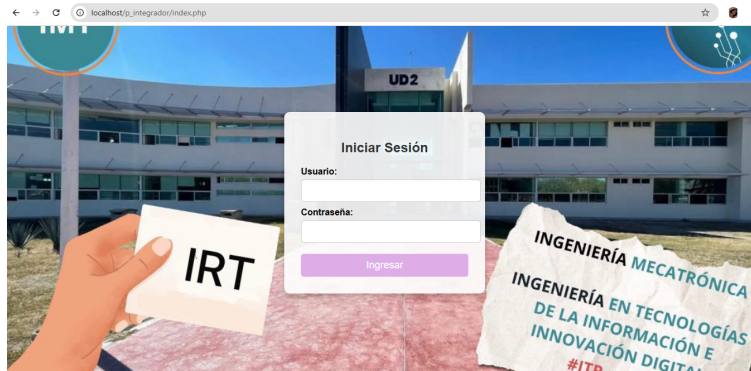


Figura 7.2: Base de datos (MYSQL)

+ Opciones

				id	usuario	password
<input type="checkbox"/>	Editar	Copiar	Borrar	7	gil	0987
<input type="checkbox"/>	Editar	Copiar	Borrar	8	bren	12345
<input type="checkbox"/>	Editar	Copiar	Borrar	9	noelia	noelia
<input type="checkbox"/>	Editar	Copiar	Borrar	10	luis	luis
<input type="checkbox"/>	Editar	Copiar	Borrar	11	erick	erick
<input type="checkbox"/>	Editar	Copiar	Borrar	12	fernando	fernando

Figura 7.3: Datos almacenados en la base de datos

- La interfaz principal contará con un menú llamado LOGIN, donde los usuarios deberán ingresar un nombre de usuario y una contraseña.

- La información de acceso estará almacenada en una base de datos, y el sistema validará el usuario ingresado antes de permitir el acceso.
- En caso de que el usuario sea incorrecto, se mostrará un mensaje de error y se restringirá el acceso.

## 7.5. Selección de Espacios Monitoreados

**Bienvenido, noelia**

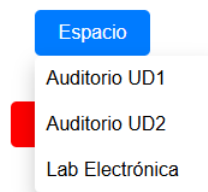
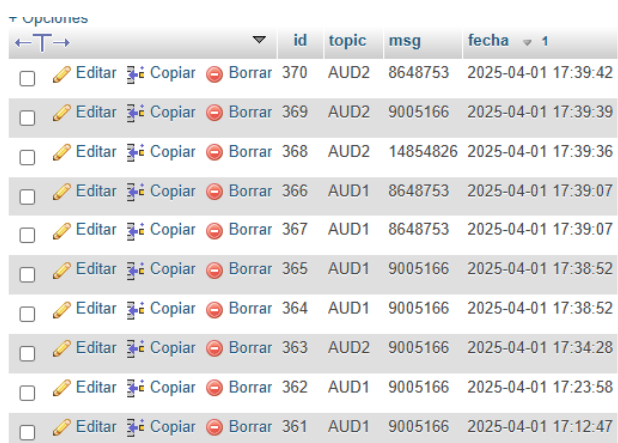


Figura 7.4: Página espacios monitoreados

Una vez iniciada la sesión, el usuario tendrá acceso a una interfaz con las opciones correspondientes a los espacios monitoreados. En este caso solo tendremos las siguientes:

- Auditorio
- Laboratorio de móviles
- Laboratorio electrónica

Al seleccionar cualquiera de estos espacios, el sistema desplegará los registros almacenados en la base de datos correspondientes a la ubicación elegida. Además, solo los administradores tendrán acceso a estos espacios y podrán gestionar su contenido.



		id	topic	msg	fecha
<input type="checkbox"/>	Editar Copiar Borrar	370	AUD2	8648753	2025-04-01 17:39:42
<input type="checkbox"/>	Editar Copiar Borrar	369	AUD2	9005166	2025-04-01 17:39:39
<input type="checkbox"/>	Editar Copiar Borrar	368	AUD2	14854826	2025-04-01 17:39:36
<input type="checkbox"/>	Editar Copiar Borrar	366	AUD1	8648753	2025-04-01 17:39:07
<input type="checkbox"/>	Editar Copiar Borrar	367	AUD1	8648753	2025-04-01 17:39:07
<input type="checkbox"/>	Editar Copiar Borrar	365	AUD1	9005166	2025-04-01 17:38:52
<input type="checkbox"/>	Editar Copiar Borrar	364	AUD1	9005166	2025-04-01 17:38:52
<input type="checkbox"/>	Editar Copiar Borrar	363	AUD2	9005166	2025-04-01 17:34:28
<input type="checkbox"/>	Editar Copiar Borrar	362	AUD1	9005166	2025-04-01 17:23:58
<input type="checkbox"/>	Editar Copiar Borrar	361	AUD1	9005166	2025-04-01 17:12:47

Figura 7.5: Topic cambiado

## 7.6. Control de Accesos y Usuarios

- Se implementó un sistema de permisos donde solo los administradores pueden gestionar los usuarios.
- Solo los administradores tienen acceso a los espacios monitoreados y pueden agregar nuevos usuarios.

							id	usuario	password	espacio_asignado
<input type="checkbox"/>		Editar		Copiar		Borrar	7	gil	0987	NULL
<input type="checkbox"/>		Editar		Copiar		Borrar	8	bren	12345	NULL
<input type="checkbox"/>		Editar		Copiar		Borrar	9	noelia	noelia	NULL
<input type="checkbox"/>		Editar		Copiar		Borrar	10	luis	luis	Auditorio UD2
<input type="checkbox"/>		Editar		Copiar		Borrar	11	erick	erick	Lab Electronica
<input type="checkbox"/>		Editar		Copiar		Borrar	12	fernando	fernando	Auditorio UD1
<input type="checkbox"/>		Editar		Copiar		Borrar	13	bertha	123	Lab redes

Figura 7.6: Espacios asignados

## 7.7. Base de Datos

La base de datos está diseñada para almacenar los registros de acceso y la información de los usuarios. Algunas de sus tablas principales son:

- **usuarios:** Contiene información de los usuarios y sus credenciales.
- **accesos:** Registra los accesos con fecha, hora y ubicación.
- **espacios:** Contiene los espacios disponibles y sus permisos de acceso.

## 7.8. Autenticación y Seguridad

La seguridad del sistema se implementa mediante:

- Uso de contraseñas para proteger los registros.
- **Autorización basada en roles** para restringir accesos.

## 7.9. Filtro por Fecha y Hora

Para mejorar la búsqueda de registros, se ha implementado un filtro de fecha y hora. Esto permite consultar accesos dentro de un intervalo de tiempo específico, optimizando la gestión de datos.

- Para facilitar la búsqueda de registros específicos, se implementó un filtro de fecha y hora.
- Los usuarios podrán seleccionar un rango de fechas y horas para visualizar únicamente los registros que correspondan a dicho intervalo.
- Esto permitirá un acceso rápido a la información relevante y mejorará la gestión de datos históricos.



The screenshot displays the 'Registros de Auditorio 1' interface. At the top, there is a filter section with 'Fecha inicio:' and 'Fecha fin:' labels, each followed by a date input field in 'dd/mm/aaaa' format and a calendar icon. A blue 'Filtrar' button is positioned to the right. Below the filter is a table with two columns: 'Fecha' and 'Matricula'. The table contains 11 rows of data, all showing the date '2025-03-31' and the matriculation number '322030156'.

Fecha	Matricula
2025-03-31 21:31:53	322030156
2025-03-31 21:42:54	322030156
2025-03-31 21:51:00	322030156
2025-03-31 22:02:05	322030156
2025-03-31 22:08:46	322030156
2025-03-31 22:09:03	322030156
2025-03-31 22:14:08	322030156
2025-03-31 22:25:37	322030156
2025-03-31 22:33:58	322030156
2025-03-31 22:49:06	322030156

Figura 7.7: Datos mostrados en la pagina web



## 7.10. Generación de Reportes en PDF

Los reportes se generan en formato PDF utilizando la librería FPDF. Estos reportes incluyen información detallada de los accesos y permiten documentar eventos relevantes para auditorías o revisiones.

- El sistema incluirá una opción para generar reportes en formato PDF con los registros mostrados en pantalla.
- Esta funcionalidad será útil para documentar eventos, respaldar información y generar informes para auditorías o revisiones.
- El reporte incluirá detalles como fecha, hora, espacio seleccionado y otras especificaciones que se agregarán más adelante, presentados en un formato estructurado y profesional.

## 7.11. Aplicación móvil

La aplicación móvil será desarrollada en Flutter dentro del entorno de desarrollo Visual Studio Code. Su objetivo será servir como un dispositivo compacto para mostrar información visual relacionada con el sistema RFID, eliminando la necesidad de hardware adicional.

### **Funcionalidades de la Aplicación Móvil**

La aplicación móvil desarrollada, la cual se conecta a un broker MQTT para recibir y enviar mensajes en tiempo real. La aplicación se utiliza para mostrar información visual relacionada con el sistema RFID, permitiendo la validación inmediata de acceso a través

de la visualización de imágenes. Su objetivo será servir como un dispositivo compacto para mostrar información visual relacionada con el sistema RFID, eliminando la necesidad de hardware adicional.

## 7.12. Detalle del código Flutter

### Estado de la Aplicación `MyAppState`)

Variables Importantes:

- `client`: Instancia de `MqttServerClient` para gestionar la conexión con el broker MQTT.
- `mensaje`: Cadena que muestra el mensaje recibido. Inicialmente, Esperando mensaje....
- `imageUrl`: Ruta de la imagen a mostrar. Se inicia con una imagen por defecto ubicada en `assets/UPJR.jpg`.
- Inicialización: En el método `initState()`, se llama a `connectMQTT()` para establecer la conexión con el broker.

### Conexión MQTT y Manejo de Mensajes

Configuración del Cliente: Se crea una instancia de `MqttServerClient` utilizando una dirección IP y un identificador único basado en el tiempo. Se configuran opciones como el período de mantenimiento de la conexión (`keepAlivePeriod`) y se definen callbacks para eventos de conexión y desconexión.

Mensaje de Conexión: Se utiliza un `MqttConnectMessage` para configurar parámetros como el identificador del cliente y la calidad del servicio (QoS).

Suscripción y Actualización: La aplicación se suscribe al tópico `lab`.

Cuando se recibe un mensaje, se extrae su contenido y se actualiza el estado del widget. Esto incluye la modificación del texto del mensaje y la ruta de la imagen, que se actualiza dinámicamente para mostrar la fotografía correspondiente.

## 7.13. Visualización de Imágenes en Tiempo Real

- Cuando una persona pase su credencial RFID, la aplicación móvil mostrará su fotografía en la pantalla del dispositivo.
- Esto permitirá una validación visual inmediata del acceso sin requerir equipos adicionales.

Esta funcionalidad se implementa actualizando la ruta de la imagen (`imageUrl`) en el método de recepción de mensajes MQTT, permitiendo así mostrar la imagen específica relacionada con el mensaje recibido.

## 7.14. Integración con el Sistema RFID

- La aplicación estará conectada en tiempo real con el sistema de monitoreo de registros.
- Recibirá la información de la base de datos y la presentará en la pantalla de manera automática.

Mediante el uso de MQTT, la aplicación se integra con el sistema RFID para recibir datos que indican la identidad o la acción (por ejemplo, el paso de una credencial RFID) y responder mostrando la imagen correspondiente. Esta integración permite una respuesta inmediata y visual del sistema de acceso.

## **7.15. Próximos Pasos**

Se seguirá optimizando el sistema y añadiendo mejoras de seguridad y experiencia de usuario. La pagina web tendra un formato mas especificado lo cual es con base a como se les pide a los laboratoristas.