

# Índice general

<b>Resumen</b>	<b>I</b>
<b>1. Introducción general</b>	<b>1</b>
1.1. Descripción del sistema	1
1.2. Motivación	2
1.3. Estado del arte	2
1.4. Objetivos y alcance	3
<b>2. Introducción específica</b>	<b>5</b>
2.1. Internet de las cosas	5
2.2. Protocolos de comunicación	5
2.2.1. Protocolo HTTP	5
2.2.2. Protocolo MQTT	6
2.2.3. Broker Eclipse Mosquitto	7
2.3. Bases de datos	7
2.3.1. Bases de datos relacionales	7
2.3.2. Sistema de gestión de base de datos MySQL	7
2.4. Tecnologías backend	8
2.4.1. API RESTFul	8
2.4.2. Node.js	9
2.4.3. Express	9
2.4.4. Sequelize	9
2.5. Tecnologías frontend	10
2.5.1. Ionic framework	10
2.6. Tecnologías de servidor	11
2.6.1. Docker y Docker Compose	11
2.6.2. Servidor web Nginx	11
2.7. Hardware utilizado	11
2.7.1. Raspberry Pi	11
2.7.2. NodeMCU Esp32	12
2.7.3. Identificación por radiofrecuencia	13
2.7.4. Módulo RC522	13
2.7.5. Buzzer sonoro	13
<b>3. Diseño e implementación</b>	<b>15</b>
3.1. Análisis del software	15
<b>4. Ensayos y resultados</b>	<b>17</b>
4.1. Pruebas funcionales del hardware	17
<b>5. Conclusiones</b>	<b>19</b>
5.1. Conclusiones generales	19
5.2. Próximos pasos	19

# Índice general

<b>Resumen</b>	<b>I</b>
<b>1. Introducción general</b>	<b>1</b>
1.1. Descripción del sistema	1
1.2. Motivación	2
1.3. Estado del arte	2
1.4. Objetivos y alcance	3
<b>2. Introducción específica</b>	<b>5</b>
2.1. Internet de las cosas	5
2.2. Protocolos de comunicación	5
2.2.1. Protocolo HTTP	5
2.2.2. Protocolo MQTT	6
2.2.3. Broker Eclipse Mosquitto	7
2.3. Bases de datos	7
2.3.1. Bases de datos relacionales	7
2.3.2. Sistema de gestión de base de datos MySQL	7
2.4. Tecnologías backend	8
2.4.1. API RESTFul	8
2.4.2. Node.js	9
2.4.3. Express	9
2.4.4. Sequelize	9
2.5. Tecnologías frontend	10
2.5.1. Ionic framework	10
2.6. Tecnologías de servidor	11
2.6.1. Docker y Docker Compose	11
2.6.2. Servidor web Nginx	11
2.7. Hardware utilizado	11
2.7.1. Raspberry Pi	11
2.7.2. NodeMCU Esp32	12
2.7.3. Identificación por radiofrecuencia	13
2.7.4. Módulo RC522	13
2.7.5. Buzzer sonoro	13
<b>3. Diseño e implementación</b>	<b>15</b>
3.1. Arquitectura general del sistema	15
<b>4. Ensayos y resultados</b>	<b>17</b>
4.1. Pruebas funcionales del hardware	17
<b>5. Conclusiones</b>	<b>19</b>
5.1. Conclusiones generales	19
5.2. Próximos pasos	19

Principales características del modelo Raspberry Pi 4 seleccionado para este trabajo:

- Procesador Broadcom BCM2711 de cuatro núcleos ARM Cortex-A72 a 1,5 GHz.
- Procesador gráfico VideoCore VI con soporte para OpenGL ES 3.x.
- 8 GB de memoria RAM LPDDR4-3200.
- Bluetooth 5.0.
- Wi-Fi de doble banda 802.11ac.
- Gigabit Ethernet.

Puertos:

- Dos puertos micro-HDMI que pueden soportar dos pantallas con resolución de hasta 4K a 60 fps.
- Dos puertos USB 3.0.
- Dos puertos USB 2.0
- Un puerto GPIO de 40 pines
- Un puerto CSI para la conexión de una cámara
- Un puerto DSI para la conexión de una pantalla táctil
- Un puerto de audio de 3,5 mm.

La placa es compatible con diferentes sistemas operativos, incluyendo Raspberry Pi OS (anteriormente llamado Raspbian), Ubuntu, Windows 10 IoT Core y otros sistemas operativos basados en Linux. También es compatible con diferentes lenguajes de programación como Python, C++, Java y más, lo que la hace ideal para proyectos de IoT, robótica, automatización del hogar, entre otros.

### 2.7.2. NodeMCU Esp32

La NodeMCU ESP32 [35] es una placa de desarrollo basada en el microcontrolador ESP32, que es ampliamente utilizado en proyectos de Internet de las cosas (IoT).

Detalles técnicos del modelo NodeMCU ESP32 WROOM 32 seleccionado para este trabajo:

- Microcontrolador ESP32 de doble núcleo con una velocidad de reloj de hasta 240 MHz y 512 KB de memoria RAM.
- 4 MB de memoria flash integrada para almacenamiento de programas y datos.
- Conectividad inalámbrica Wi-Fi 802.11 b/g/n y Bluetooth 4.2 BLE.
- Interfaz de programación USB integrada para programar la placa y proporcionar una conexión de depuración.
- GPIO de 30 pines para la conexión de sensores, actuadores y otros dispositivos periféricos.

Principales características del modelo Raspberry Pi 4 seleccionado para este trabajo:

- Procesador Broadcom BCM2711 de cuatro núcleos ARM Cortex-A72 a 1,5 GHz.
- Procesador gráfico VideoCore VI con soporte para OpenGL ES 3.x.
- 8 GB de memoria RAM LPDDR4-3200.
- Bluetooth 5.0.
- Wi-Fi de doble banda 802.11ac.
- Gigabit Ethernet.

Puertos:

- Dos puertos micro-HDMI que pueden soportar dos pantallas con resolución de hasta 4K a 60 fps.
- Dos puertos USB 3.0.
- Dos puertos USB 2.0
- Un puerto GPIO de 40 pines
- Un puerto CSI para la conexión de una cámara
- Un puerto DSI para la conexión de una pantalla táctil
- Un puerto de audio de 3,5 mm.

La placa es compatible con diferentes sistemas operativos, incluyendo Raspberry Pi OS (anteriormente llamado Raspbian), Ubuntu, Windows 10 IoT Core y otros sistemas operativos basados en Linux. También es compatible con diferentes lenguajes de programación como Python, C++, Java y más, lo que la hace ideal para proyectos de IoT, robótica, automatización del hogar, entre otros.

### 2.7.2. NodeMCU Esp32

La NodeMCU ESP32 [35] es una placa de desarrollo basada en el microcontrolador ESP32, que es ampliamente utilizado en proyectos de Internet de las cosas (IoT).

Detalles técnicos del modelo NodeMCU ESP32 WROOM 32 seleccionado para este trabajo:

- Microcontrolador ESP32 de doble núcleo con una velocidad de reloj de hasta 240 MHz y 512 kB de memoria RAM.
- 4 MB de memoria flash integrada para almacenamiento de programas y datos.
- Conectividad inalámbrica Wi-Fi 802.11 b/g/n y Bluetooth 4.2 BLE.
- Interfaz de programación USB integrada para programar la placa y proporcionar una conexión de depuración.
- GPIO de 30 pines para la conexión de sensores, actuadores y otros dispositivos periféricos.

- Interfaces I2C, SPI, UART, PWM y ADC integradas.
- Soporte para el entorno de programación Arduino, así como para MicroPython y Lua.
- Compatible con una amplia gama de bibliotecas y herramientas de desarrollo de código abierto.

### 2.7.3. Identificación por radiofrecuencia

RFID [36] son las siglas en inglés de *Radio Frequency Identification* o *Identificación por Radiofrecuencia* en español. Es una tecnología de identificación automática que utiliza ondas de radio para leer y capturar información almacenada en etiquetas o *tags* RFID.

Esta tecnología consta de tres componentes básicos: el tag RFID que comunmente es una tarjeta o llavero pequeño, el lector RFID y un sistema informático que gestiona la información capturada por el lector.

Los tags RFID contienen antenas y circuitos integrados que permiten la comunicación inalámbrica con los lectores, los cuales envían señales de radio para alimentar y activar los tags, y recibir la información almacenada en ellos. Pueden ser pasivos, activos o semi-activos, dependiendo de si necesitan o no una fuente de alimentación externa.

### 2.7.4. Módulo RC522

El RFID RC522 [37] es un módulo de lectura-escritura RFID que utiliza la tecnología de comunicación de campo cercano (NFC) para la identificación y el intercambio de datos de manera inalámbrica. Entre sus principales características se encuentran:

- Soporte para frecuencias de operación de 13,56 MHz.
- Comunicación mediante el protocolo SPI (Serial Peripheral Interface).
- Capacidad para leer y escribir etiquetas RFID de tipo MIFARE, que son ampliamente utilizadas en sistemas de acceso, control de inventario, pago sin contacto, entre otros.
- Funciones de autenticación y encriptación para mayor seguridad en la transmisión de datos.
- Bajo consumo de energía y fácil integración con microcontroladores y otros sistemas embebidos.

### 2.7.5. Buzzer sonoro

El módulo buzzer pasivo KY006 [38] es un pequeño dispositivo electrónico que se utiliza para producir sonidos audibles en una variedad de proyectos electrónicos. Se conecta a la placa de control mediante tres pines.

Entre las características técnicas del módulo KY006 se encuentran:

- Tensión de funcionamiento: 5 V DC.
- Corriente de funcionamiento: <25 mA.

- Interfaces I2C, SPI, UART, PWM y ADC integradas.
- Soporte para el entorno de programación Arduino, así como para MicroPython y Lua.
- Compatible con una amplia gama de bibliotecas y herramientas de desarrollo de código abierto.

### 2.7.3. Identificación por radiofrecuencia

RFID [36] son las siglas en inglés de *Radio Frequency Identification* o *identificación por radiofrecuencia* en español. Es una tecnología de identificación automática que utiliza ondas de radio para leer y capturar información almacenada en etiquetas o *tags* RFID.

Esta tecnología consta de tres componentes básicos: el tag RFID que comunmente es una tarjeta o llavero pequeño, el lector RFID y un sistema informático que gestiona la información capturada por el lector.

Los tags RFID contienen antenas y circuitos integrados que permiten la comunicación inalámbrica con los lectores, los cuales envían señales de radio para alimentar y activar los tags, y recibir la información almacenada en ellos. Pueden ser pasivos, activos o semi-activos, dependiendo de si necesitan o no una fuente de alimentación externa.

### 2.7.4. Módulo RC522

El RFID RC522 [37] es un módulo de lectura-escritura RFID que utiliza la tecnología de comunicación de campo cercano (NFC) para la identificación y el intercambio de datos de manera inalámbrica. Entre sus principales características se encuentran:

- Soporte para frecuencias de operación de 13,56 MHz.
- Comunicación mediante el protocolo SPI (*Serial Peripheral Interface*).
- Capacidad para leer y escribir etiquetas RFID de tipo MIFARE, que son ampliamente utilizadas en sistemas de acceso, control de inventario, pago sin contacto, entre otros.
- Funciones de autenticación y encriptación para mayor seguridad en la transmisión de datos.
- Bajo consumo de energía y fácil integración con microcontroladores y otros sistemas embebidos.

### 2.7.5. Buzzer sonoro

El módulo buzzer pasivo KY006 [38] es un pequeño dispositivo electrónico que se utiliza para producir sonidos audibles en una variedad de proyectos electrónicos. Se conecta a la placa de control mediante tres pines.

Entre las características técnicas del módulo KY006 se encuentran:

- Tensión de funcionamiento: 5 V DC.
- Corriente de funcionamiento: <25 mA.

## Capítulo 3

# Diseño e implementación

### 3.1. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno `lstlisting` con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]
las líneas de código irían aquí...
\end{lstlisting}
```

A modo de ejemplo:

```
1 #define MAX_SENSOR_NUMBER 3
2 #define MAX_ALARM_NUMBER 6
3 #define MAX_ACTUATOR_NUMBER 6
4
5 uint32_t sensorValue[MAX_SENSOR_NUMBER];
6 FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
7 state_t alarmState[MAX_ALARM_NUMBER]; //ON or OFF
8 state_t actuatorState[MAX_ACTUATOR_NUMBER]; //ON or OFF
9
10 void vControl() {
11     initGlobalVariables();
12
13     period = 500 ms;
14
15     while(1) {
16
17         ticks = xTaskGetTickCount();
18
19         updateSensors();
20
21         updateAlarms();
22
23         controlActuators();
24
25         vTaskDelayUntil(&ticks, period);
26     }
27 }
28 }
```

CÓDIGO 3.1. Pseudocódigo del lazo principal de control.

## Capítulo 3

# Diseño e implementación

### 3.1. Arquitectura general del sistema