Proyecto Final Microcontroladores

Indice (Autogenerado por Obsidian)

Esquema de componentes

LED RGB (x2)

Los LED permiten mostrar diferentes estados de la habitación mediante colores. En el proyecto se utilizan dos unidades para ofrecer indicadores de estado visibles.

Motivo del componente

Se utiliza para proporcionar señales visuales claras y rápidas, como:

- Alerta de temperatura fuera de rango
- Aviso de incendio o humo
- Indicación de acceso permitido/denegado
- Estado operativo general encendido/apagado

Código y Esquema eléctrico

Vamos a usar para esto el Sketch *Blink*

```
const int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Pantalla LCD y OLED

Las pantallas LCD y OLED nos permiten mostrar información esencial del sistema.

Motivo del componente

Se utiliza para proporcionar información en tiempo real facilitando la monitorización, como:

- Temperatura y Humedad
- Que usuario está entrando/saliendo y si tiene permiso

Código y Esquema eléctrico

Este código de ejemplo utiliza la librería de AdaFruit

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

Adafruit_SSD1306 display = Adafruit_SSD1306(128, 32, &WIRE);

void setup() {
  Serial.begin(9600);

  Serial.println("OLED test");
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  Serial.println("OLED Arrancado");
  display.display();
  delay(1000);
  display.clearDisplay();
  display.display();

  Serial.println("IO test");
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  display.print("Connecting to SSID\n'adafruit':");
  display.print("Agustin que haces que no\n estas estudiando");
  display.println("IP: 10.0.1.23");
  display.println("Sending val #0");
  display.setCursor(0,0);
  display.display();
}
```

WEMOS (ESP8266/ESP32)

Módulo de desarrollo WIFI integrado utilizado para la conexión del sistema al servidor MQTT. Actúa como puente entre los sensores/actuadores y la plataforma de control remoto (en el servidor también, puede accederse desde (Control de Acceso - MSanchez))

Motivo del componente

Permite Comunicación inalámbrica con el servidor, enviando datos de sensores y recibiendo órdenes (por ejemplo → Abrir puerta, activar ventilador).

Código y Esquema eléctrico

Código de ejemplo de conexión del wemos (a nuestro mqtt)

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHTesp.h"

// --- Sensor DHT11 en GPIO2 (D2)
DHTesp dht;
// --- Configuración WiFi ---
const char* ssid = "iPhone de manue";
const char* password = "pitocaca";

// --- Configuración MQTT ---
const char* mqtt_server = "msanchez.ovh";
const int mqtt_port = 1883;
const char* mqtt_client_id = "ESP8266_Control_MSanchez";

const char* mqtt_user = "ardu";
const char* mqtt_password = "JMMAMicro";

// --- Topics MQTT ---
const char* TOPIC_DOOR_STATUS = "casa/puerta/estado";
const char* TOPIC_TEMP = "casa/sensor/temperatura";
const char* TOPIC_HUM = "casa/sensor/humedad";
const char* TOPIC_NFC_USER = "casa/nfc/usuario";

const char* TOPIC_ORDER_DOOR = "casa/puerta/orden";
const char* TOPIC_ORDER_ALARM = "casa/alarma/orden";
const char* TOPIC_ORDER_VENTILATION = "casa/ventilacion/orden";

const char* topic_subscriptions[] = {
    TOPIC_ORDER_DOOR,
    TOPIC_ORDER_ALARM,
    TOPIC_ORDER_VENTILATION
};
```

```
const int num_subscriptions = 3;

WiFiClient espClient;
PubSubClient client(espClient);

long lastPublish = 0;
int door_state = 0;

// ----------------------- CALLBACK MQTT -----------------------
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("ORDEN RECIBIDA en [");
  Serial.print(topic);
  Serial.print("]: ");

  String command;
  for (int i = 0; i < length; i++) command += (char)payload[i];
  Serial.println(command);

  // --- Puerta ---
  if (strcmp(topic, TOPIC_ORDER_DOOR) == 0) {
    if (command == "ABRIR") {
      door_state = 1;
      client.publish(TOPIC_DOOR_STATUS, "ABIERTA");
    } else if (command == "CERRAR") {
      door_state = 0;
      client.publish(TOPIC_DOOR_STATUS, "CERRADA");
    }
  }

  // --- Alarma ---
  else if (strcmp(topic, TOPIC_ORDER_ALARM) == 0) {
    Serial.println(command == "ON" ? "ALARMA ON" : "ALARMA OFF");
  }

  // --- Ventilación ---
  else if (strcmp(topic, TOPIC_ORDER_VENTILATION) == 0) {
    Serial.println(command == "ON" ? "VENTILACIÓN ON" : "VENTILACIÓN OFF");
  }
}

// ----------------------- RECONNECT MQTT -----------------------
void reconnect() {
  while (!client.connected()) {
    Serial.print("Intentando conexión MQTT...");
    if (client.connect(mqtt_client_id, mqtt_user, mqtt_password)) {
      Serial.println("conectado!");
```

```
      for (int i = 0; i < num_subscriptions; i++) {
        client.subscribe(topic_subscriptions[i]);
      }

      client.publish(TOPIC_DOOR_STATUS, door_state == 1 ? "ABIERTA" : "CERRADA");
    } else {
      Serial.print("falló, rc=");
      Serial.println(client.state());
      delay(3000);
    }
  }
}

// ------------------------ WIFI ------------------------
void setup_wifi() {
  Serial.print("Conectando a ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(300);
    Serial.print(".");
  }

  Serial.println("\nConectado!");
  Serial.println(WiFi.localIP());
}

// ------------------------ SETUP ------------------------
void setup() {
  Serial.begin(115200);

  setup_wifi();

  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);

  // ---- Inicializar DHT ----
  dht.setup(2, DHTesp::DHT11);  // GPIO2 = D2
  Serial.println("DHT11 inicializado");
}

// ------------------------ LOOP ------------------------
void loop() {
  if (!client.connected()) reconnect();
```

```
  client.loop();

  long now = millis();
  if (now - lastPublish > 10000) {
    lastPublish = now;

    TempAndHumidity data = dht.getTempAndHumidity();

    if (isnan(data.humidity) || isnan(data.temperature)) {
      Serial.println("Error leyendo DHT11");
      return;
    }

    char tempString[10];
    char humString[10];

    dtostrf(data.temperature, 4, 1, tempString);
    dtostrf(data.humidity, 4, 1, humString);

    client.publish(TOPIC_TEMP, tempString);
    client.publish(TOPIC_HUM, humString);

    Serial.print("Publicado -> T: ");
    Serial.print(tempString);
    Serial.print(" °C  |  H: ");
    Serial.println(humString);
  }
}
```

Sensor DHT11

Sensor digital básica para medir temperatura y humedad dentro de la habitación frigorífica.

Motivo del componente

Imprescindible para simular el control térmico de la habitación, generando valores que pueden activar el ventilador, alertar sobre fallos o enviarse al servidor.

Código y Esquema eléctrico

```
#include "DHT.h"

// ----- Configuración del sensor -----
#define DHTPIN 7        // Pin digital donde está conectado el DHT11
#define DHTTYPE DHT11   // Tipo de sensor (DHT11 azul)
```

```
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  Serial.println("Iniciando sensor DHT11...");

  dht.begin();

  Serial.println("Status\tHumedad (%)\tTemp (C)\tTemp (F)\tHeatIndex (C)\t(F)");
}

void loop() {
  // El DHT11 requiere al menos 1 segundo entre lecturas
  delay(2000);

  float humidity = dht.readHumidity();
  float temperatureC = dht.readTemperature();       // Celsius
  float temperatureF = dht.readTemperature(true);   // Fahrenheit

  // Comprobar si la lectura falló
  if (isnan(humidity) || isnan(temperatureC)) {
    Serial.println("Error leyendo del DHT11");
    return;
  }

  // Cálculo de heat index
  float heatIndexC = dht.computeHeatIndex(temperatureC, humidity, false);
  float heatIndexF = dht.computeHeatIndex(temperatureF, humidity, true);

  // Impresión
  Serial.print("OK\t");
  Serial.print(humidity);
  Serial.print("\t\t");
  Serial.print(temperatureC);
  Serial.print("\t\t");
  Serial.print(temperatureF);
  Serial.print("\t\t");
  Serial.print(heatIndexC);
  Serial.print("\t\t");
  Serial.println(heatIndexF);
}
```

Sensor de incendios (KY-026)

Sensor analógico/digital capaz de detectar presencia de fuego. Permite identificar situaciones peligrosas dentro de la habitación frigorífica

Motivo del componente

Se integra en el sistema de alarma de seguridad, activando zumbadores y notificando al servidor.

Código y Esquema eléctrico

```
int buzzer = 11;      // selecciona el pin para el zumbador
int valorSensor = 0;  // variable para almacenar el valor proveniente del sensor
void setup() {
 //Seteo de la velocidad del puerto serial
 Serial.begin(9600);
 // declarar buzzer como una SALIDA:
  pinMode(buzzer, OUTPUT);
}
void loop() {
  // leer el valor del sensor:
  valorSensor = analogRead(A0);
   Serial.println(valorSensor);
  // activa el buzzer
  if (valorSensor < 500){
   digitalWrite(buzzer, HIGH);
   delay(100);
  // Desactiva el  buzzer
  digitalWrite(buzzer, LOW);
  delay(50);
    }
 }
```

Zumbador

Dispositivo sonoro capaz de emitir avisos acústicos.

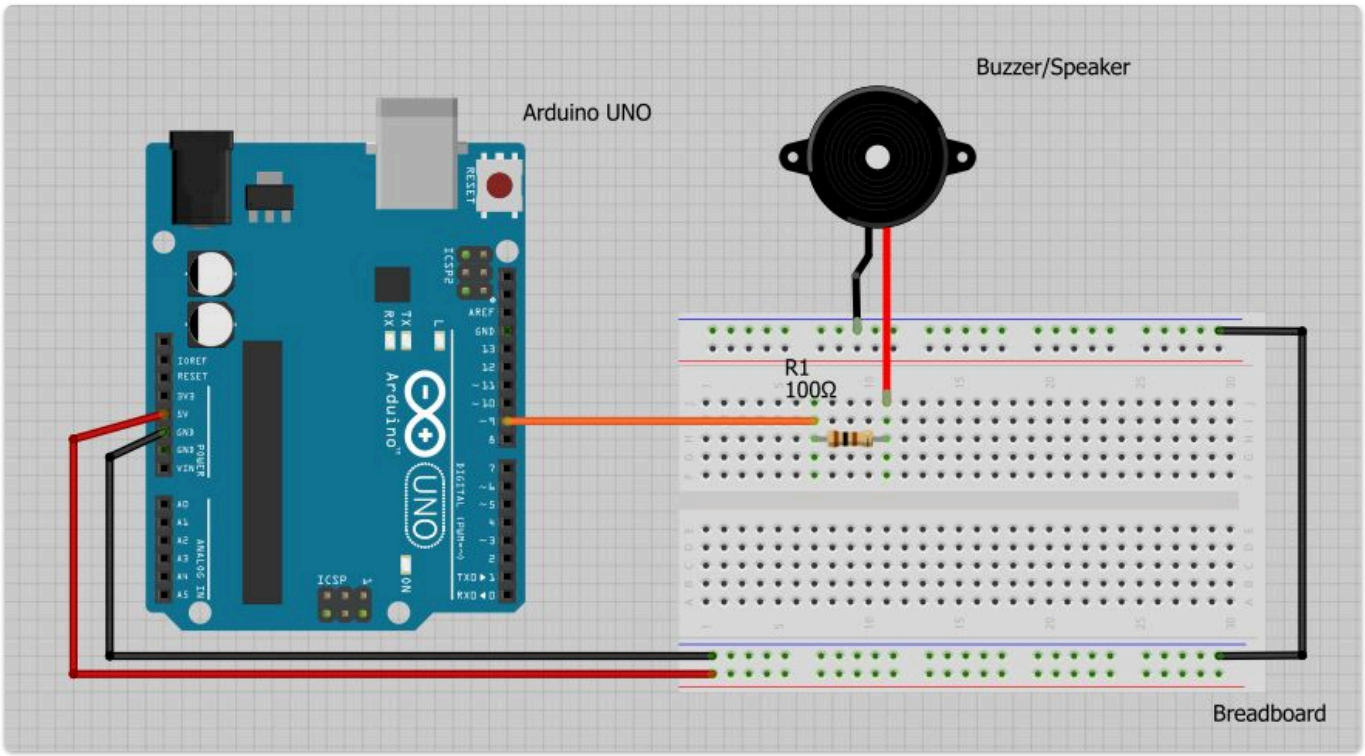
Motivo del componente

Proporciona alertas sonoras para situaciones críticas como:

- Incendio o humo
- Acceso no autorizado
- Fallos en sensores
- Sobre temperatura

Código y Esquema eléctrico

```
const int buzzer = 9;
void setup(){
  pinMode(buzzer, OUTPUT);
}

void   loop(){
  Zumba();
}

void Zumba() {
  tone(buzzer, 1000);
  delay(100);
  noTone(buzzer);
  delay(100);
}
```



Sensor de nivel de agua

Sensor que detecta presencia o nivel de agua en la base de la habitación.

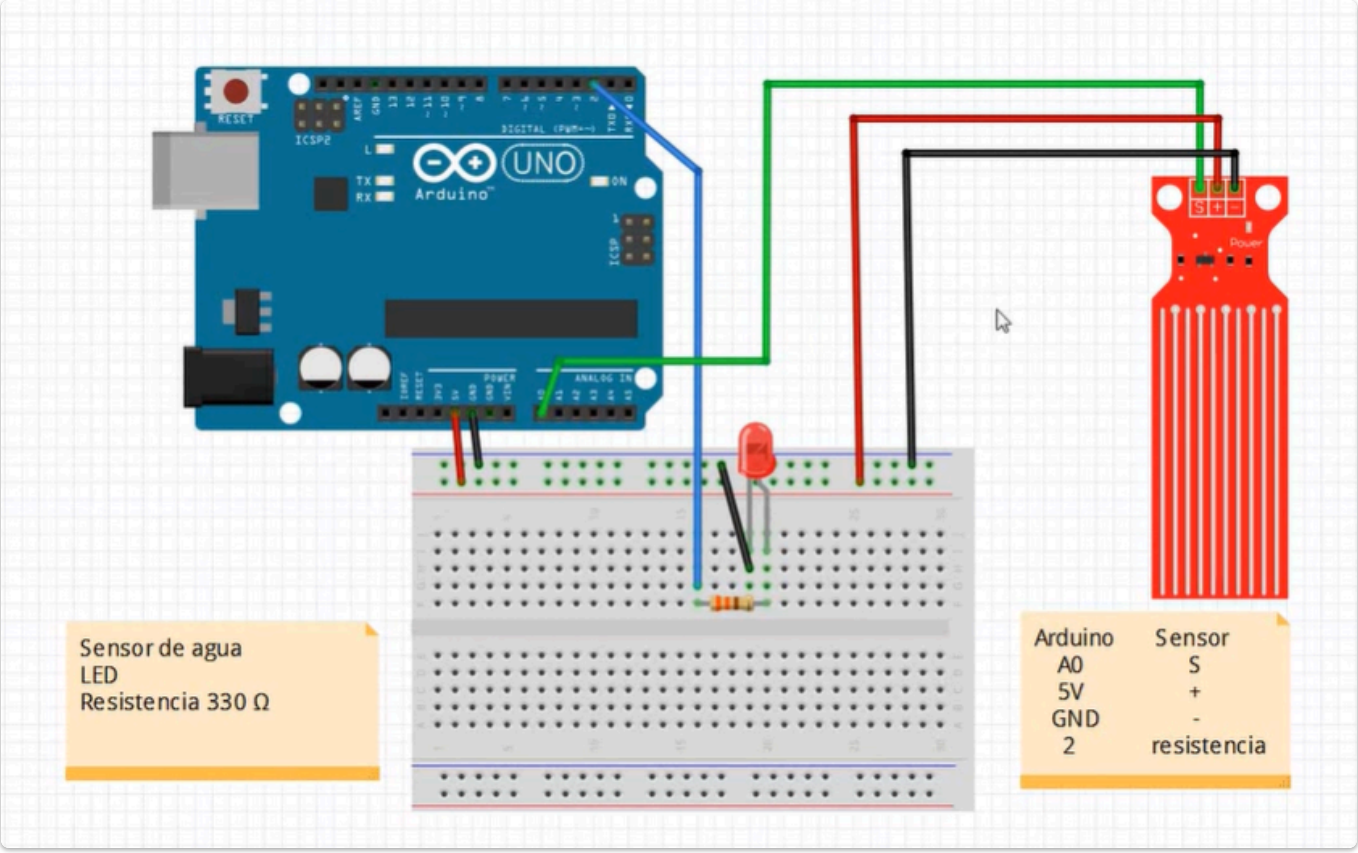
Motivo del componente

Simula un sistema de detección de fugas de agua, que podría indicar deterioro de la instalación o un fallo en el sistema de refrigeración

Código y Esquema eléctrico

El código es lo más básico posible.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(analogRead(5));
  delay(1000);
}
```



Servomotor

Motor encargado de bloquear el acceso a la puerta de manera automática

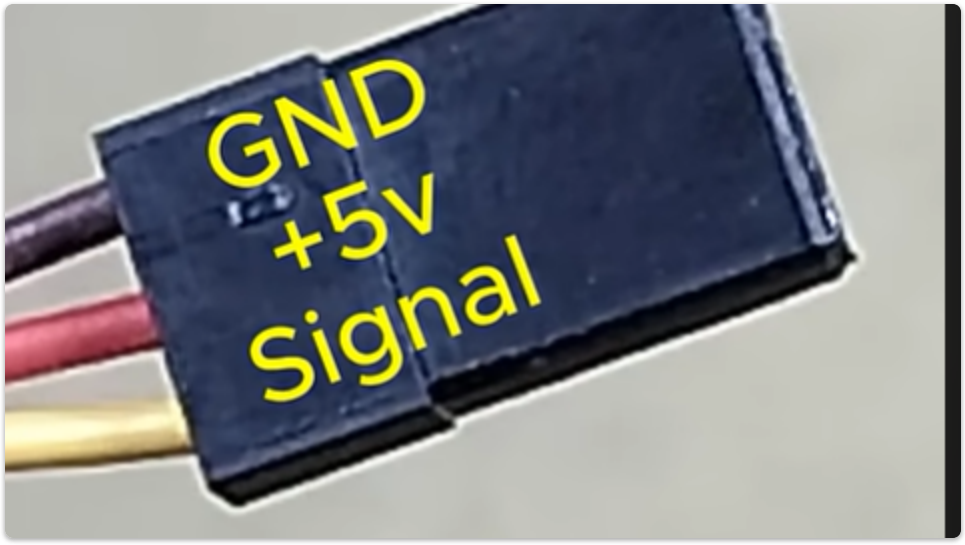
Motivo del componente

Simula un sistema de detección de fugas de agua, que podría indicar deterioro de la instalación o un fallo en el sistema de refrigeración

Código y Esquema eléctrico

Hay que usar la biblioteca Servo

```
#include <Servo.h>
Servo myservo;  // create Servo object to control a servo

int potpin = A0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the Servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer
(value between 0 and 1023)
  val = 110; // Cerrado     // scale it for use with the servo (value between 0 and
180)
  myservo.write(val);                  // sets the servo position according to the
scaled value
  delay(15);                           // waits for the servo to get there
}
```



Ventilador

Ventilador eléctrico encargado de simular el sistema de refrigeración interna y circulación del aire

Motivo del componente

Permite demostrar un control térmico básico, activándose cuando la temperatura supera un umbral y apagándose al normalizarse, o cuando el servidor envía una orden.

Código y Esquema eléctrico

```
// Basicamente con 255 es encendido pero con el 0 es apagado.
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {
  analogWrite(3,255);
  delay(1000);
  analogWrite(3,0);
  delay(5000);
}
```

NFC

Módulo de lectura NFC que permite identificar tarjetas o llaveros autorizados. Se comunica con Arduino o Wemos para validación de acceso

Motivo del componente

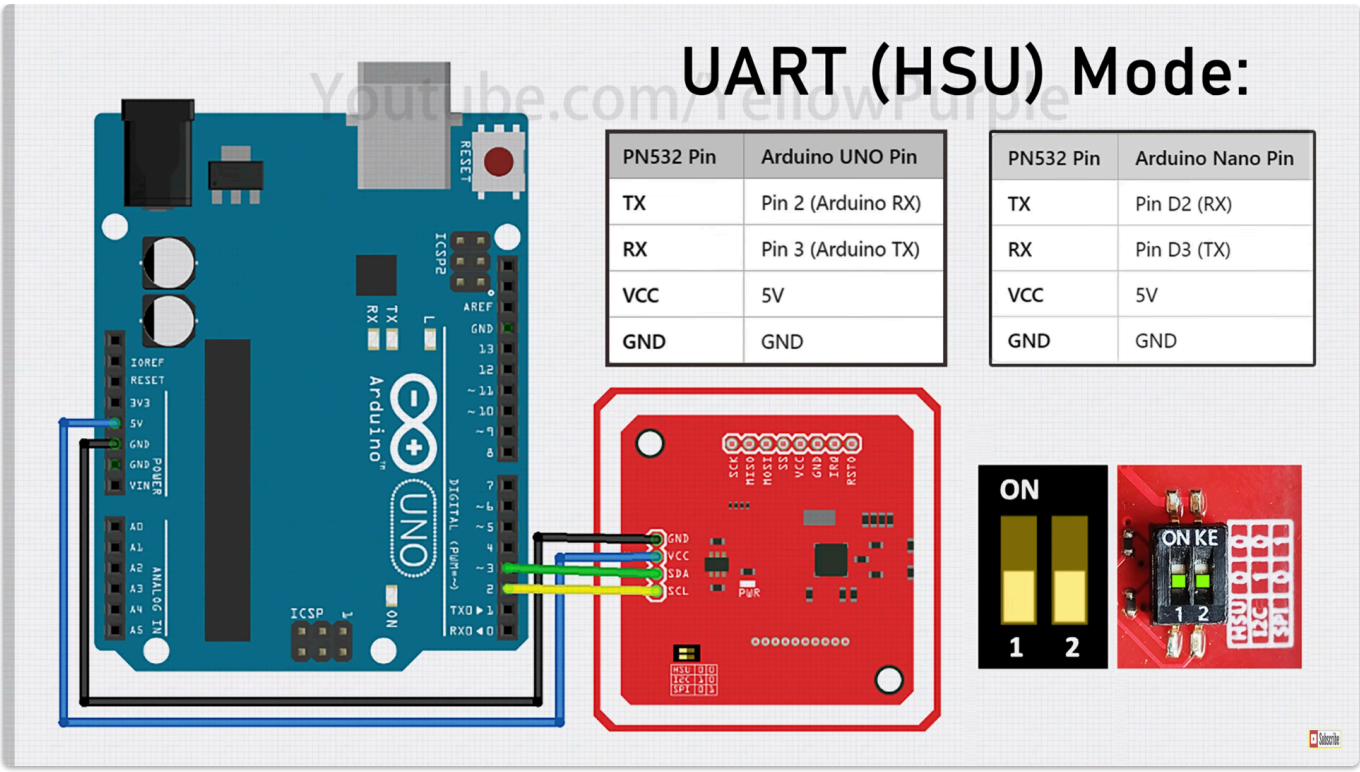
Permite controlar la entrada a la cámara frigorífica, permitiendo el acceso solo a usuarios autorizados. Puede además verificarse contra el servidor MQTT para mayor seguridad.

Código y Esquema eléctrico

Usamos la biblioteca Adafruit_PN532

```
#include <Wire.h>
#include <Adafruit_PN532.h>

#define PN532_I2C_ADDR (0x24)
#define PN532_IRQ   (2)  // Pin IRQ (Interrupción)
#define PN532_RST   (3)  // Pin RST (Reset)

Adafruit_PN532 nfc(PN532_IRQ, PN532_RST);

void setup(void) {
  Serial.begin(115200);
  Serial.println("--- Lector PN532 Inicializando ---");

  nfc.begin();

  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata) {
    Serial.print("No se encontró el chip PN532. Verifica las conexiones.");
    while (1);
```

```
  }

  Serial.print("PN532 Encontrado! Versión de Firmware: ");
  Serial.println((versiondata>>16) & 0xFF, HEX);

  nfc.SAMConfig();

  Serial.println("Esperando tarjeta NFC/RFID...");
}


void loop(void) {
  uint8_t success;
  uint8_t uid[7];
  uint8_t uidLength;
  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);

  if (success) {
    Serial.println("\n--- Tarjeta Detectada ---");
    Serial.print("  Longitud del UID: ");
    Serial.print(uidLength, DEC);
    Serial.println(" bytes");

    Serial.print("  UID: ");
    for (uint8_t i=0; i < uidLength; i++) {
      Serial.print(" 0x");
      Serial.print(uid[i], HEX);
    }
    Serial.println("");
    delay(2000);
  }
  // Si no hay tarjeta, el loop continúa buscando.
}
```



Servidor MQTT

El sistema utiliza un servidor MQTT basado en Mosquitto, que actúa como intermediario entre la maqueta (Arduino/Wemos) y la web.

A través de este servidor, la habitación puede publicar datos (temperatura, alarmas, estado de la puerta...) y suscribirse a órdenes enviadas desde un panel de control o aplicación.

Mosquitto se ejecuta como broker y ofrece un canal de comunicación ligero, rápido y adecuado para dispositivos IoT con recursos limitados.

Motivo del componente

El servidor MQTT permite:

- Comunicación bidireccional entre la maqueta y los servicios.
- Control remoto de actuadores.
- Envío de datos en tiempo real.

Tabla de tópics MQTT

Topic	Quién envía	Quién escucha	Mensaje Ejemplo
puerta/orden	Web	Arduino	"ABRIR", "CERRAR"
alarma/orden	Web	Arduino	"ON", "OFF"
ventilacion/orden	Web	Arduino	"ON", "OFF"

Topic	Quién envía	Quién escucha	Mensaje Ejemplo
puerta/estado	Arduino	Web	"ABIERTA", "CERRADA"
sensor/temperatura	Arduino	Web	"24.5"
sensor/humedad	Arduino	Web	"60"
nfc/usuario	Arduino	Web	"Manuel"
puerta/permisos	Web	Arduino	"INCORRECTO", "CORRECTO"
ventilacion/estado	Arduino	Web	"ENCENDIDO","APAGADO"
sensor/fuego	Arduino	Web	"OK","FUEGO"
sensor/agua	Arduino	Web	"OK","MOJADO"

Configuración de Mosquitto

```
listener 1883
protocol mqtt
allow_anonymous false
password_file /etc/mosquitto/passwd
listener 9001
protocol websockets
```