**TRƯỜNG ĐẠI HỌC KHOA HỌC HUẾ**
**KHOA CÔNG NGHỆ THÔNG TIN**

---

# ĐỀ TÀI:

# HỆ THỐNG KHÓA CỬA THÔNG MINH DỰA TRÊN ESP32 VÀ RFID

**HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG IOT - NHÓM 6**

**Giảng viên hướng dẫn: Võ Việt Dũng**

**Sinh viên thực hiện: Văn Huỳnh Tường An**

*HUẾ, 04/2025*

# LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn chân thành đến thầy Võ Việt Dũng người đã hướng dẫn và hỗ trợ trong suốt quá trình học tập để em có thể hoàn thành bài tiểu luận này.

Bên cạnh đó, em cũng xin bày tỏ lòng biết ơn đến những người bạn ở trường Đại học và các anh chị em ở khắp mọi nơi trên mạng xã hội đã chia sẻ những kinh nghiệm quý báu để hỗ trợ em trong quá trình nghiên cứu đề tài tiểu luận.

Cuối cùng, em xin gửi lời cảm ơn sâu sắc nhất đến gia đình, bạn bè và tất cả những người đã dành thời gian và nỗ lực để giúp đỡ về mặt tinh thần cũng như sức khỏe để em có thể hoàn thành đề tài này.

Trong quá trình nghiên cứu và thu thập kết quả để hoàn thiện bài tiểu luận về đề tài ***Hệ thống khóa cửa thông minh dựa trên ESP32 và RFID***, vì hạn chế về mặt kiến thức chuyên môn cũng như kinh nghiệm thực tế do đó bài báo cáo của em khó tránh khỏi những sai sót nhất định. Em rất mong có được những ý kiến đóng góp của thầy để bài báo cáo và bản thân em hoàn thiện một cách chỉnh chu hơn.

Em xin chân thành cảm ơn!

Sinh viên

**Văn Huỳnh Tường An**

# DANH MỤC CÁC HÌNH ẢNH

# MỤC LỤC

# PHẦN MỞ ĐẦU

Trong bối cảnh công nghệ phát triển mạnh mẽ như hiện nay, nhu cầu về an ninh và sự tiện lợi trong cuộc sống hàng ngày ngày càng trở nên cấp thiết. Các hệ thống khóa cửa truyền thống sử dụng chìa khóa cơ học tuy vẫn phổ biến, nhưng đã bộc lộ nhiều hạn chế như dễ bị thất lạc, sao chép hoặc phá khóa. Chính vì vậy, các hệ thống khóa cửa thông minh đã và đang dần thay thế, mang lại giải pháp an toàn, hiện đại và thuận tiện hơn cho người sử dụng.

Đề tài Nghiên cứu ***Hệ thống khóa cửa thông minh dựa trên ESP32 và RFID*** tập trung xây dựng một mô hình khóa cửa thông minh sử dụng vi điều khiển ESP32, kết hợp với cảm biến thẻ RFID RC522 để xác thực người dùng và relay để điều khiển việc đóng/mở khóa. Bên cạnh đó, hệ thống còn sử dụng đèn LED và buzzer để cung cấp phản hồi trực quan, giúp người dùng dễ dàng nhận biết trạng thái xác thực (thành công hoặc thất bại).

Toàn bộ hệ thống được mô phỏng bằng nền tảng Wokwi, một công cụ trực tuyến tiện lợi cho việc thiết kế, kiểm thử và triển khai các ứng dụng nhúng trong môi trường giả lập. Việc sử dụng Wokwi không chỉ giúp tiết kiệm chi phí phần cứng trong giai đoạn thử nghiệm mà còn hỗ trợ sinh viên nhanh chóng nắm bắt nguyên lý hoạt động của hệ thống.

# PHẦN NỘI DUNG

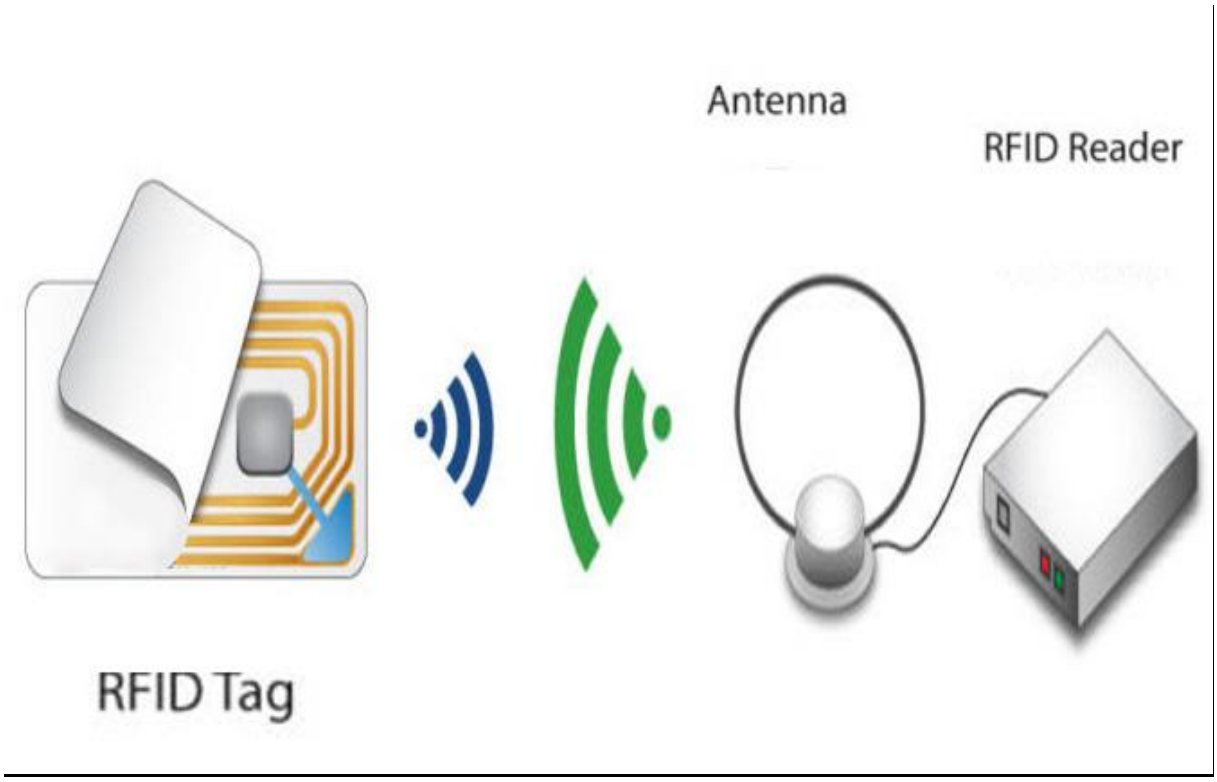
# CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG

## 1.1. RFID (Radio Frequency Identification) :


*Hình 1*

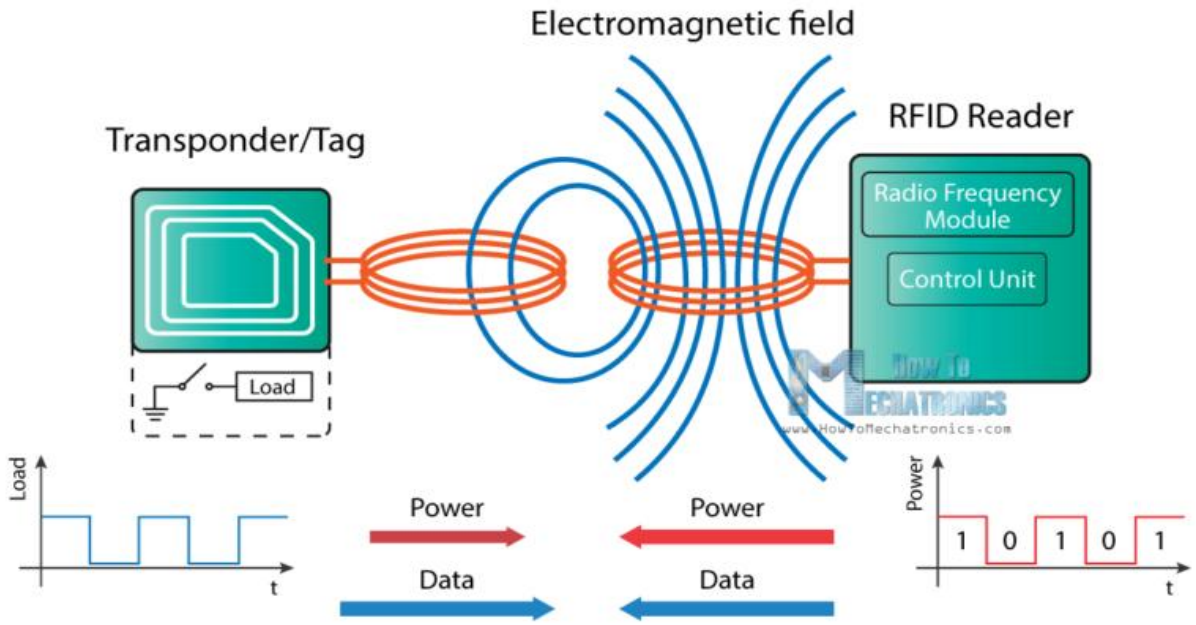
**1.1.1. Khái niệm:** RFID là công nghệ nhận dạng đối tượng bằng sóng vô tuyến (tần số radio). Khi đó cả hai thiết bị hoạt động thu phát sóng trong cùng tần số và tần số đó thường được sử dụng trong RFID là 125Khz hoặc 900Mhz.

- Cách thức hoạt động: Một thiết bị RFID được cấu tạo bởi 2 thành phần chính là thiết bị đọc và thiết bị phát mã có gắn chip. Trong đó thiết bị đọc được gắn antenna thu phát sóng điện từ, còn thiết bị phát mã RFID được gắn với vật cần nhận dạng, mỗi thiết bị RFID có chứa một mã số nhất định sao cho không trùng lặp với nhau.




*Hình 2*

- Nguyên tắc làm việc của RFID: Khi thẻ được cấp nguồn, nó có thể trích xuất thông tin được truyền từ đầu đọc và gửi tin nhắn trở lại đầu đọc, nó sử dụng một kỹ thuật gọi là thao tác tải. Bật và tắt tải ở ăng-ten của thẻ sẽ ảnh hưởng đến mức tiêu thụ điện của ăng-ten của bộ đọc có thể được đo là sụt áp. Sự thay đổi điện áp này sẽ được ghi lại dưới dạng 0 và 1 và đó là cách dữ liệu được truyền từ thẻ đến đầu đọc.




*Hình 3*

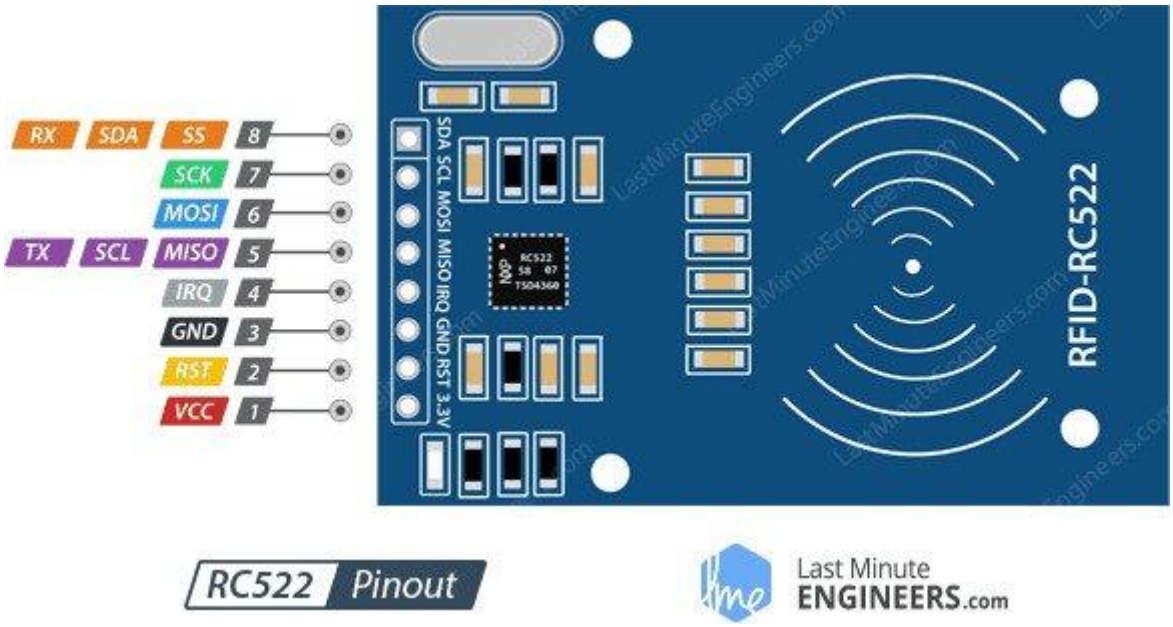
### 1.1.2. RFID RC-522:

Module RFID RC522 sử dụng IC MFRC522 của Phillip dùng để đọc và ghi dữ liệu cho thẻ NFC tần số 13.56MHz, giao tiếp với vi điều khiển thông qua chuẩn SPI và có tốc độ truyền dữ liệu đa 10Mbit/s.

**Thông số kỹ thuật:**

| | |
|---|---|
| Frequency Range | 13.56 MHz ISM Band |
| Host Interface | SPI / I2C / UART |
| Operating Supply Voltage | 2.5 V to 3.3 V |
| Max. Operating Current | 13-26mA |
| Min. Current(Power down) | 10µA |
| Logic Inputs | 5V Tolerant |
| Read Range | 5 cm |

**Sơ đồ chân Module RFID RC522**




*Hình 4*

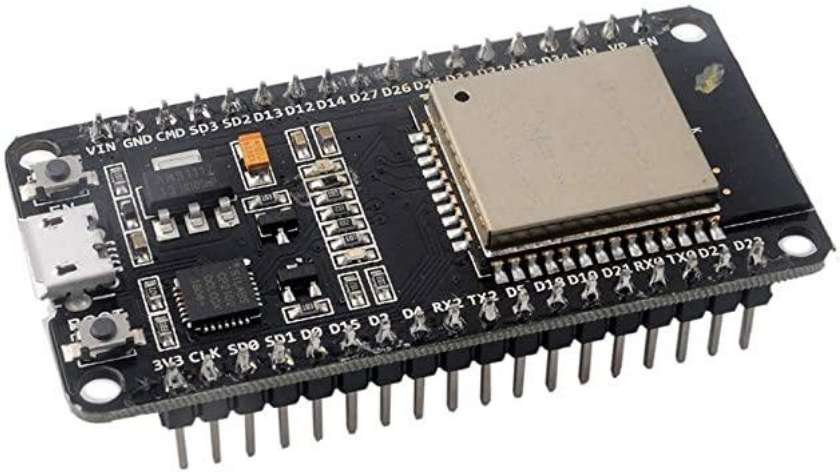
- **DA (Serial Data):** Dùng để truyền và nhận dữ liệu giữa module RFID RC522 Arduino.
- **SCK (Serial Clock):** Dùng để đồng bộ hóa truyền thông SPI giữa module RFID RC522 và Arduino.
- **MOSI (Master Output Slave Input):** Dùng để truyền dữ liệu từ Arduino tới module RFID RC522.
- **MISO (Master Input Slave Output):** Chân đầu vào của Arduino nối với chân MISO trên module RFID RC522. Dùng để nhận dữ liệu từ module RFID RC522 về Arduino qua giao thức SPI.
- **IRQ (Interrupt Request):** Chân ngắt không được sử dụng trong một số ứng dụng của module RFID RC522. Nếu sử dụng, chân này được sử dụng để xác định xem module RFID RC522 có sự kiện cần xử lý hay không.
- **GND (Ground):** Chân đất kết nối với chân GND trên Arduino.
- **RST (Reset):** Chân đặt lại (reset) module RFID RC522. Khi chân này được kích hoạt, module sẽ được đặt lại về trạng thái ban đầu.
- **3.3V:** Nguồn cấp 3.3V cho module.

### 1.2. ESP32:

NodeMCU ESP32 được phát triển trên nền ESP32 với công nghệ WiFi, BLE và nhân ARM SoC tích hợp mới nhất hiện nay được ứng dụng cho IoT và thu thập dữ liệu qua mạng. NodeMCU ESP32 hỗ trợ nhiều chân GPIO, hỗ trợ nhiều giao thức như SPI, I2C,

UART, ... Điều quan trọng nhất của NodeMCU ESP32 là có thể điều khiển và giám sát thiết bị từ xa thông qua internet với giá thành thấp.

### 1.2.1. Cấu hình của ESP32:




*Hình 5*

**CPU**

- CPU: Xtensa Dual-Core LX6 microprocessor.
- Chạy hệ 32 bit
- Tốc độ xử lý 160MHZ up to 240 MHz
- Tốc độ xung nhịp đọc flash chip 40mhz --> 80mhz (tùy chỉnh khi lập trình)
- RAM: 520 KByte SRAM
- 520 KB SRAM liền chip –(trong đó 8 KB RAM RTC tốc độ cao – 8 KB RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).

**Hỗ trợ 2 giao tiếp không dây**

- Wi-Fi: 802.11 b/g/n/e/i
- Bluetooth: v4.2 BR/EDR and BLE

**Hỗ trợ tất cả các loại giao tiếp**

- 8-bit DACs( digital to analog) 2 cổng
- Analog(ADC) 12-bit 16 cổng.
- $I^2C$ – 2 cổng
- UART – 3 cổng
- SPI – 3 cổng (1 cổng cho chip FLASH )
- $I^2S$ – 2 cổng
- SD card /SDIO/MMC host
- Slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN bus 2.0

- IR (TX/RX)
- Băm xung PWM (tất cả các chân )
- Ultra low power analog pre-amplifier’

**Cảm biến tích hợp trên chip esp32**

- 1 cảm biến Hall (cảm biến từ trường)
- 1 cảm biến đo nhiệt độ
- Cảm biến chạm (điện dung) với 10 đầu vào khác nhau.

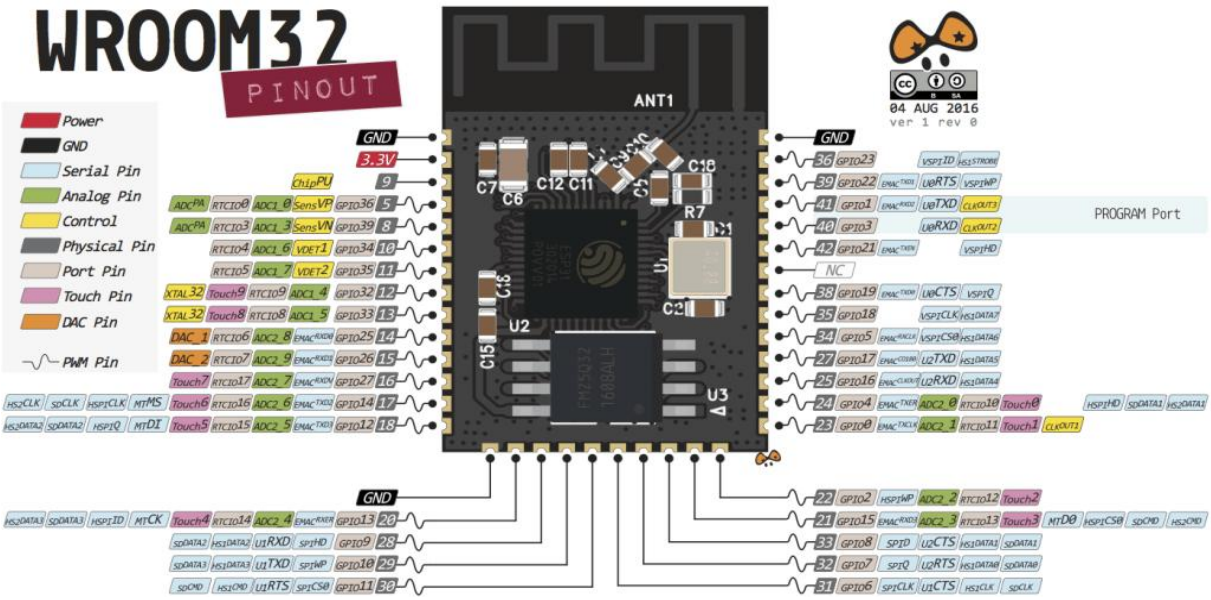
**Bảo mật**

- IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

**Nguồn điện hoạt động**

- Nhiệt độ hoạt động -40 + 85C
- Điện áp hoạt động: 2.2-3.6V
- Số cổng GPIOs : 34

**1.2.3. Sơ đồ chân ESP32:**




*Hình 6*

**Chân Input Only**

GPIO từ 34 đến 39 là GPI – chân chỉ đầu vào. Các chân này không có điện trở kéo lên hoặc kéo xuống bên trong. Chúng không thể được sử dụng làm đầu ra, vì vậy chỉ sử dụng các chân này làm đầu vào:

- GPIO 34
- GPIO 35
- GPIO 36
- GPIO 39

**Chân tích hợp Flash trên ESP32**

GPIO 6 đến GPIO 11 dùng để kết nối Flash SPI, không khuyến khích sử dụng trong các ứng dụng khác

- GPIO 6 (SCK/CLK)
- GPIO 7 (SDO/SD0)
- GPIO 8 (SDI/SD1)
- GPIO 9 (SHD/SD2)
- GPIO 10 (SWP/SD3)
- GPIO 11 (CSC/CMD)

**Chân cảm biến điện dung**

Các chân ESP32 này có chức năng như 1 nút nhấn cảm ứng, có thể phát hiện sự thay đổi về điện áp cảm ứng trên chân.

Các cảm biến cảm ứng bên trong đó được kết nối với các GPIO sau:

- T0 (GPIO 4)
- T1 (GPIO 0)
- T2 (GPIO 2)
- T3 (GPIO 15)
- T4 (GPIO 13)
- T5 (GPIO 12)
- T6 (GPIO 14)
- T7 (GPIO 27)
- T8 (GPIO 33)
- T9 (GPIO 32)

**Analog to Digital Converter (ADC)**

ESP32 có các kênh đầu vào ADC 18 x 12 bit (trong khi ESP8266 chỉ có ADC 1x 10 bit). Đây là các GPIO có thể được sử dụng làm ADC và các kênh tương ứng:

- ADC1_CH0 (GPIO 36)

- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)
- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

Các kênh đầu vào ADC có độ phân giải 12 bit. Điều này có nghĩa là bạn có thể nhận được các số đọc tương tự từ 0 đến 4095, trong đó 0 tương ứng với 0V và 4095 đến 3,3V. Bạn cũng có thể lập trình độ phân giải của các kênh của mình trên code.

**Digital to Analog Converter (DAC)**

Có các kênh DAC 2 x 8 bit trên ESP32 để chuyển đổi tín hiệu kỹ thuật số thành đầu ra tín hiệu điện áp tương tự. Các kênh này chỉ có độ phân giải 8 bit, nghĩa là có giá trị từ 0 – 255 tương ứng với 0 – 3.3V

Đây là các kênh DAC:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

**Các chân thời gian thực RTC:**

Các chân này có tác dụng đánh thức ESP32 khi trong chế độ Low Power Mode. Sử dụng như 1 chân ngắt ngoài.

Các chân RTC:

- RTC_GPIO0 (GPIO36)
- RTC_GPIO3 (GPIO39)
- RTC_GPIO4 (GPIO34)

- RTC_GPIO5 (GPIO35)
- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)
- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)
- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

**Chân PWM**

ESP32 LED PWM có 16 kênh độc lập có thể được định cấu hình để tạo tín hiệu PWM với các thuộc tính khác nhau. Tất cả các chân có thể hoạt động như đầu ra đều có thể được sử dụng làm chân PWM (GPIO từ 34 đến 39 không thể tạo PWM).

Để xuất PWM, bạn cần xác định các thông số này trong code:

- Frequency – tần số
- Duty cycle
- Kênh PWM
- Chân GPIO nơi bạn muốn xuất tín hiệu

**Chân I2C**

ESP32 có hai kênh I2C và bất kỳ chân nào cũng có thể được đặt làm SDA hoặc SCL. Khi sử dụng ESP32 với Arduino IDE, các chân I2C mặc định là:

- GPIO 21 (SDA)
- GPIO 22 (SCL)

Nếu các bạn muốn sử dụng chân khác cho việc điều khiển I2C có thể sử dụng code: Wire.begin(SDA, SCL);

**Chân ngắt ngoài**: Tất cả các chân ESP32 đều có thể sử dụng ngắt ngoài

### 1.3. Các phần cứng khác:

#### 1.3.1. Relay (Rơ-le):

**Khái niệm:** Relay là một công tắc điện từ có thể đóng/mở mạch điện tự động hoặc bằng tín hiệu điều khiển từ bên ngoài. Nó hoạt động dựa trên nguyên lý cảm ứng điện từ, cho phép điều khiển một mạch điện công suất lớn bằng một tín hiệu điện áp nhỏ.

*Hình 7*

**Cấu tạo của Relay**

Một relay thường gồm các thành phần chính:

- Cuộn dây (Coil):
  - Khi có dòng điện chạy qua, sinh ra từ trường hút tiếp điểm.
- Lõi sắt từ (Core):
  - Tăng cường từ trường khi cuộn dây được cấp điện.
- Tiếp điểm (Contacts):
  - NO (Normally Open): Mở khi không có điện, đóng khi có điện.
  - NC (Normally Closed): Đóng khi không có điện, mở khi có điện.
  - COM (Common): Chân trung tâm kết nối với NO/NC.

**Các ứng dụng thực tế của Relay**

- Nhà thông minh (Smart Home): Bật/tắt đèn, điều khiển rèm cửa.
- Công nghiệp: Điều khiển động cơ, máy bơm.
- An ninh: **Hệ thống khóa cửa thông minh**, báo động.

#### 1.3.2. Buzzer

*Hình 8*

**Khái niệm:** Buzzer (còi báo) là một thiết bị phát âm thanh dùng để tạo tín hiệu cảnh báo hoặc thông báo trong các hệ thống điện tử. Trong **Hệ thống khóa cửa thông minh**, buzzer được sử dụng để báo hiệu trạng thái truy cập (hợp lệ hoặc không hợp lệ).

**Nguyên lý hoạt động:**

- Buzzer chủ động
  - Khi cấp điện (HIGH), buzzer phát ra âm thanh.
  - Khi ngắt điện (LOW), buzzer ngừng kêu.
- Buzzer bị động
  - Cần tín hiệu PWM (xung vuông) để điều chỉnh âm thanh.
  - Có thể tạo nhạc bằng cách thay đổi tần số.

### 1.3.3.LED (Light Emitting Diode - Điốt Phát Quang)

**Khái niệm:** Đèn LED là linh kiện bán dẫn phát ra ánh sáng khi có dòng điện chạy qua. Trong **Hệ thống khóa cửa thông minh** LED được sử dụng để hiển thị trạng thái truy cập một cách trực quan.(LED xanh: Truy cập thành công, LED đỏ: Truy cập thất bại).

**Các thông số kỹ thuật quan trọng**

| Thông số | LED Đỏ | LED Xanh |
|---|---|---|
| Điện áp hoạt động | 1.8-2.2V | 3.0-3.4V |
| Dòng điện tối đa | 20mA | 20mA |
| Góc phát sáng | 120° | 120° |
| Cường độ sáng | 5000-7000mcd | 8000-10000mcd |

**Đặc điểm kỹ thuật của đèn LED**

- Nguyên lý hoạt động: Dựa trên hiệu ứng điện phát quang trong chất bán dẫn.
- Ưu điểm:

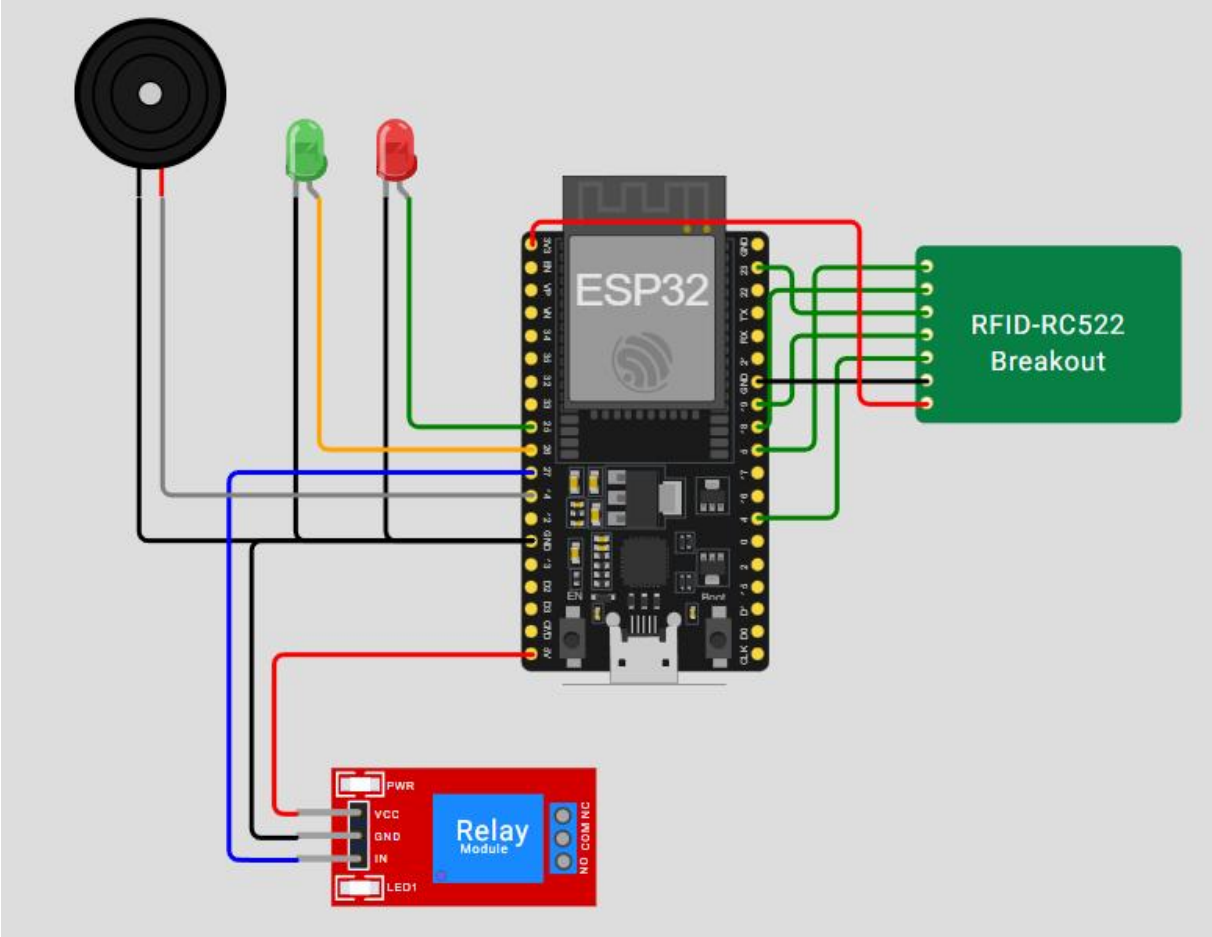
- Tiêu thụ ít điện năng
- Tuổi thọ cao (50,000 - 100,000 giờ)
- Kích thước nhỏ gọn
- Độ sáng cao

- Nhược điểm:

- Nhạy cảm với nhiệt độ cao
- Cần điện trở hạn dòng để bảo vệ

# CHƯƠNG 2. HỆ THỐNG KHÓA CỬA THÔNG MINH DỰA TRÊN ESP32 VÀ RFID

## 2.1. Mô phỏng hệ thống bằng sơ đồ Wowki:




*Hình 9*

**Các chân được dùng trong sơ đồ mạch**

| Thiết bị | Chân thiết bị | Nối với chân ESP32 | Chức năng / Giải thích |
|---|---|---|---|
| **RFID RC522** | VCC | 3V3 | Cấp nguồn 3.3V cho module RFID |
| | GND | GND | GND (mass) – nối đất chung |
| | SDA | D5 | Slave Select (SS), dùng để chọn thiết bị trong giao tiếp SPI |
| | SCK | D18 | Serial Clock (SPI clock) |
| | MOSI | D23 | Master Out Slave In – truyền dữ liệu từ ESP32 đến RFID |
| | MISO | D19 | Master In Slave Out – nhận dữ liệu từ RFID về ESP32 |

| | | | |
|---|---|---|---|
| | RST | D4 | Reset – khởi động lại module RFID nếu cần |
| **LED xanh (LED1)** | A (Anode) | D26 | Điều khiển sáng LED xanh – báo **truy cập hợp lệ** |
| | C (Cathode) | GND | Nối đất chung |
| **LED đỏ (LED2)** | A | D25 | Điều khiển sáng LED đỏ – báo **truy cập không hợp lệ** |
| | C | GND | Nối đất chung |
| **Buzzer** | + (Chân 2) | D14 | Buzzer phát tiếng khi cần cảnh báo hoặc xác nhận |
| | - (Chân 1) | GND | Nối đất |
| **Relay module** | VCC | 5V | Cấp nguồn 5V cho relay hoạt động |
| | GND | GND | Nối đất chung |
| | IN | D27 | Điều khiển bật/tắt relay (kích hoạt khi ESP32 xuất tín hiệu) |

**2.2. Code mô phỏng:**

```
#include <WiFi.h>

#define BLYNK_TEMPLATE_ID "TMPL6jZZ4VJ1l"
#define BLYNK_TEMPLATE_NAME "ESP32 SMARTKEY"
#define BLYNK_AUTH_TOKEN "Xw8KAMieyc_tcoSWg3B2HY2Kz3DupDbd"

#include <BlynkSimpleEsp32.h>

// Các chân kết nối phần cứng (RFID, Relay, LED...)
#define SS_PIN 21
#define RST_PIN 22
#define RELAY_PIN 27
#define BUZZER_PIN 14
#define GREEN_LED 26
#define RED_LED 25

// Thông tin WiFi
char ssid[] = "Wokwi-GUEST";
```

```
char pass[] = "";

// Các Virtual Pins Blynk
#define VIRTUAL_STATUS V0
#define VIRTUAL_UID    V1
#define VIRTUAL_INPUT  V2  // Text Input từ Blynk app

int failedAttempts = 0;  // Biến đếm số lần nhập sai
unsigned long lockTime = 0;  // Thời gian khóa
const unsigned long lockDuration = 30000;  // Thời gian khóa (30 giây)

// Xử lý dữ liệu từ Blynk Text Input
BLYNK_WRITE(VIRTUAL_INPUT) {
  String input = param.asString();
  input.trim();

  if (failedAttempts >= 5 && (millis() - lockTime) < lockDuration) {
    unsigned long remainingTime = lockDuration - (millis() - lockTime);
    Serial.println("Quá nhiều lần nhập sai. Vui lòng đợi " + String(remainingTime / 1000)
+ " giây.");
    Blynk.virtualWrite(VIRTUAL_STATUS, "Quá nhiều lần nhập sai. Vui lòng đợi " +
String(remainingTime / 1000) + " giây.");
    return;
  }

  if (input.length() > 0) {
    if (isValidFormat(input)) {
      if (input.equals("123456")) {  // ID hợp lệ
        accessGranted(input);
        failedAttempts = 0;  // Đặt lại bộ đếm sau khi nhập đúng
      } else {
        accessDenied(input);
        failedAttempts++;  // Tăng bộ đếm khi nhập sai
        if (failedAttempts >= 5) {
          lockTime = millis();  // Lưu thời gian khi nhập sai 5 lần
          Serial.println("Đã nhập sai 5 lần. Hệ thống sẽ khóa trong 30 giây.");
```

```
        Blynk.virtualWrite(VIRTUAL_STATUS, "Đã nhập sai 5 lần. Hệ thống khóa trong 30 giây.");
      }
    }
   } else {
    Serial.println("Định dạng không hợp lệ. Nhập e-KTP ID (6 số liền nhau, ví dụ: 123456):");
    Blynk.virtualWrite(VIRTUAL_STATUS, "Sai định dạng. Nhập 6 số liền nhau.");
   }
  }
}

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

  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);

  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(RED_LED, LOW);
  digitalWrite(RELAY_PIN, LOW);

  WiFi.begin(ssid, pass);
  Serial.print("Đang kết nối WiFi");
  while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }
  Serial.println("\nWiFi đã kết nối");

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  Serial.println("Nhập e-KTP ID (6 số liền nhau, ví dụ: 123456):");
}
```

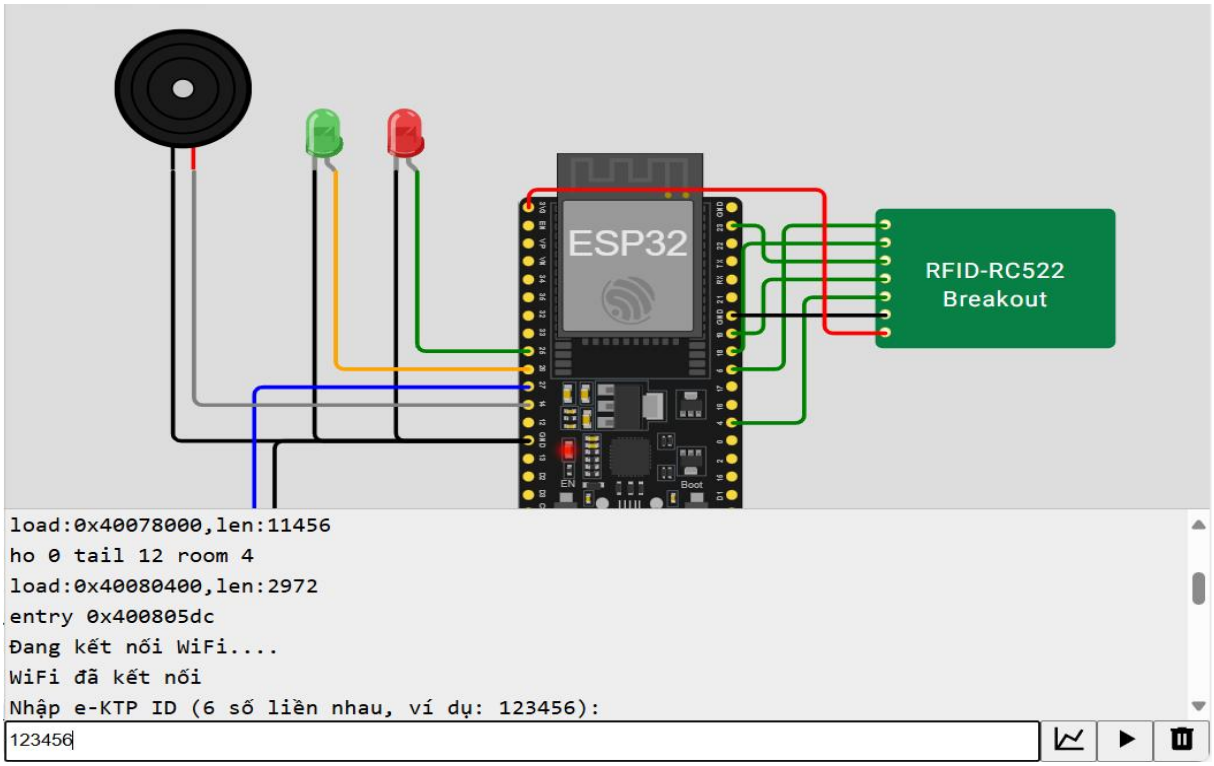
```
void loop() {
  Blynk.run();

  // Nhập qua Serial nếu có
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    input.trim();

    if (failedAttempts >= 5 && (millis() - lockTime) < lockDuration) {
      unsigned long remainingTime = lockDuration - (millis() - lockTime);
      Serial.println("Quá nhiều lần nhập sai. Vui lòng đợi " + String(remainingTime / 1000)
+ " giây.");
      return;
    }

    if (input.length() > 0) {
      if (isValidFormat(input)) {
        if (input.equals("123456")) {
          accessGranted(input);
          failedAttempts = 0;  // Đặt lại bộ đếm sau khi nhập đúng
        } else {
          accessDenied(input);
          failedAttempts++;  // Tăng bộ đếm khi nhập sai
          if (failedAttempts >= 5) {
            lockTime = millis();  // Lưu thời gian khi nhập sai 5 lần
            Serial.println("Đã nhập sai 5 lần. Hệ thống sẽ khóa trong 30 giây.");
          }
        }
      } else {
        Serial.println("Sai định dạng. Nhập e-KTP ID (6 số liền nhau, ví dụ: 123456):");
      }
    }
  }
}

// Hàm kiểm tra định dạng e-KTP mới: 6 chữ số liền nhau
bool isValidFormat(String input) {
```

```
  if (input.length() != 6) return false;
  for (int i = 0; i < 6; i++) {
    if (!isDigit(input.charAt(i))) return false;
  }
  return true;
}

void accessGranted(String uid) {
  Serial.println("✅ Truy cập được chấp nhận!");
  digitalWrite(BUZZER_PIN, HIGH);
  digitalWrite(GREEN_LED, HIGH);
  digitalWrite(RELAY_PIN, HIGH);

  Blynk.virtualWrite(VIRTUAL_STATUS, "✅ Truy cập được chấp nhận!");
  Blynk.virtualWrite(VIRTUAL_UID, uid);

  delay(5000);

  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(RELAY_PIN, LOW);
}

void accessDenied(String uid) {
  Serial.println("❌ Truy cập bị từ chối!");
  digitalWrite(BUZZER_PIN, HIGH);
  digitalWrite(RED_LED, HIGH);

  Blynk.virtualWrite(VIRTUAL_STATUS, "❌ Truy cập bị từ chối!");
  Blynk.virtualWrite(VIRTUAL_UID, uid);

  delay(5000);

  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(RED_LED, LOW);
}
```

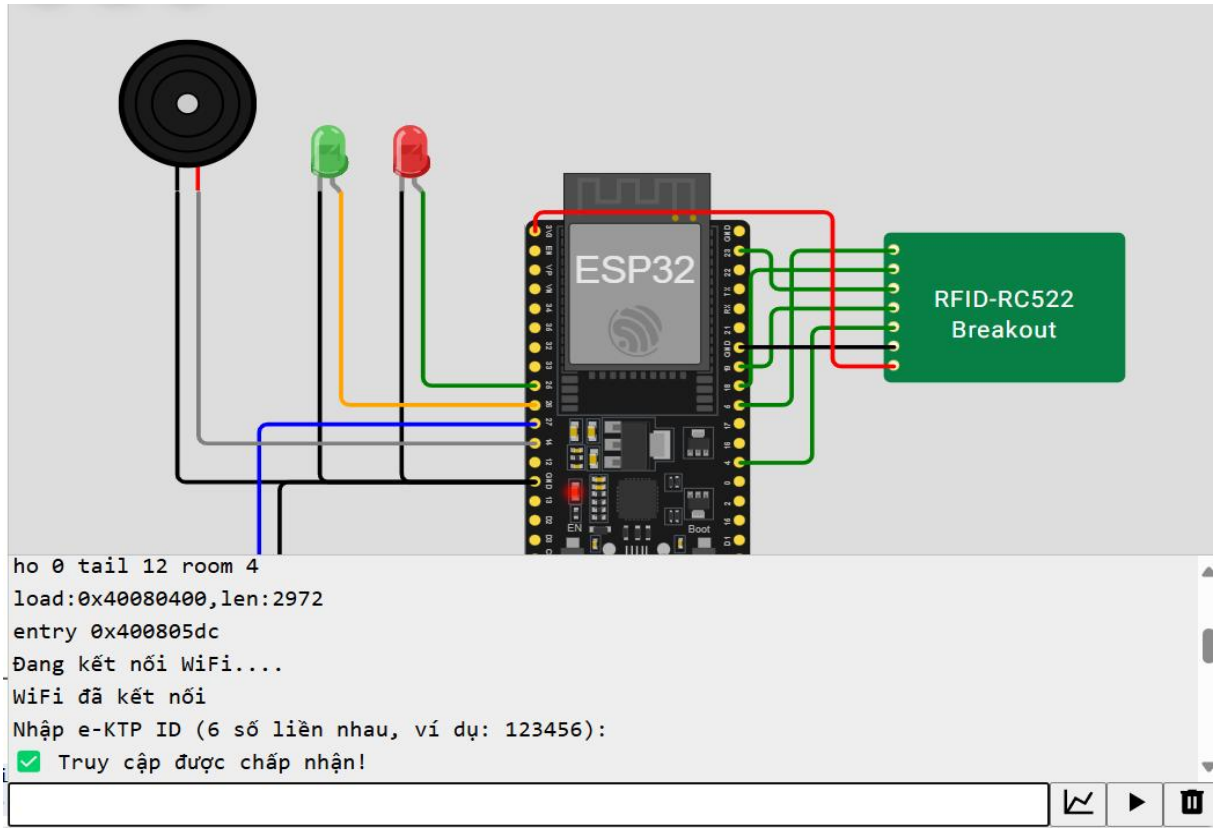
## 2.3. Kết quả mô phỏng:

### 2.3.1. Truy cập hệ thống thành công:

***Chạy ở Wokwi:***

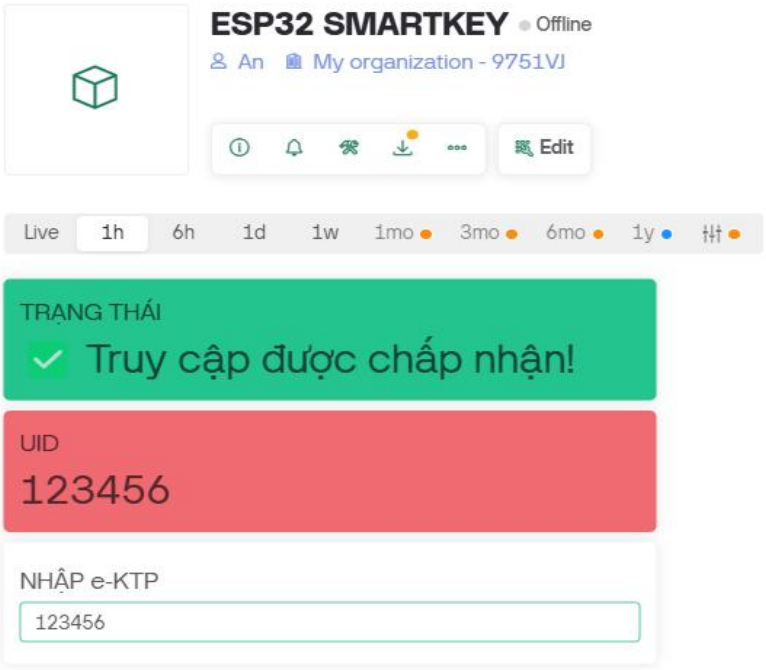



*Hình 10*




*Hình 11*

***Chạy trên Blynk:***

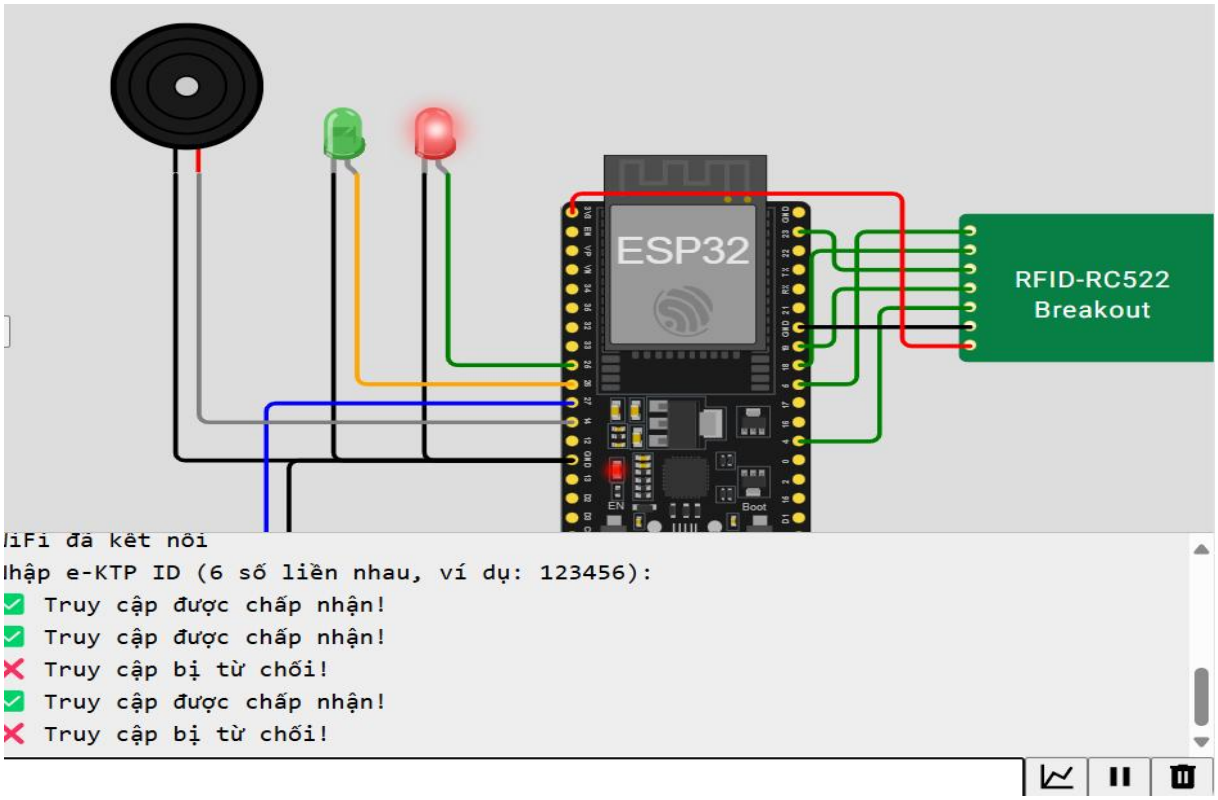



*Hình 12*

### 2.3.1. Truy cập hệ thống thất bại:

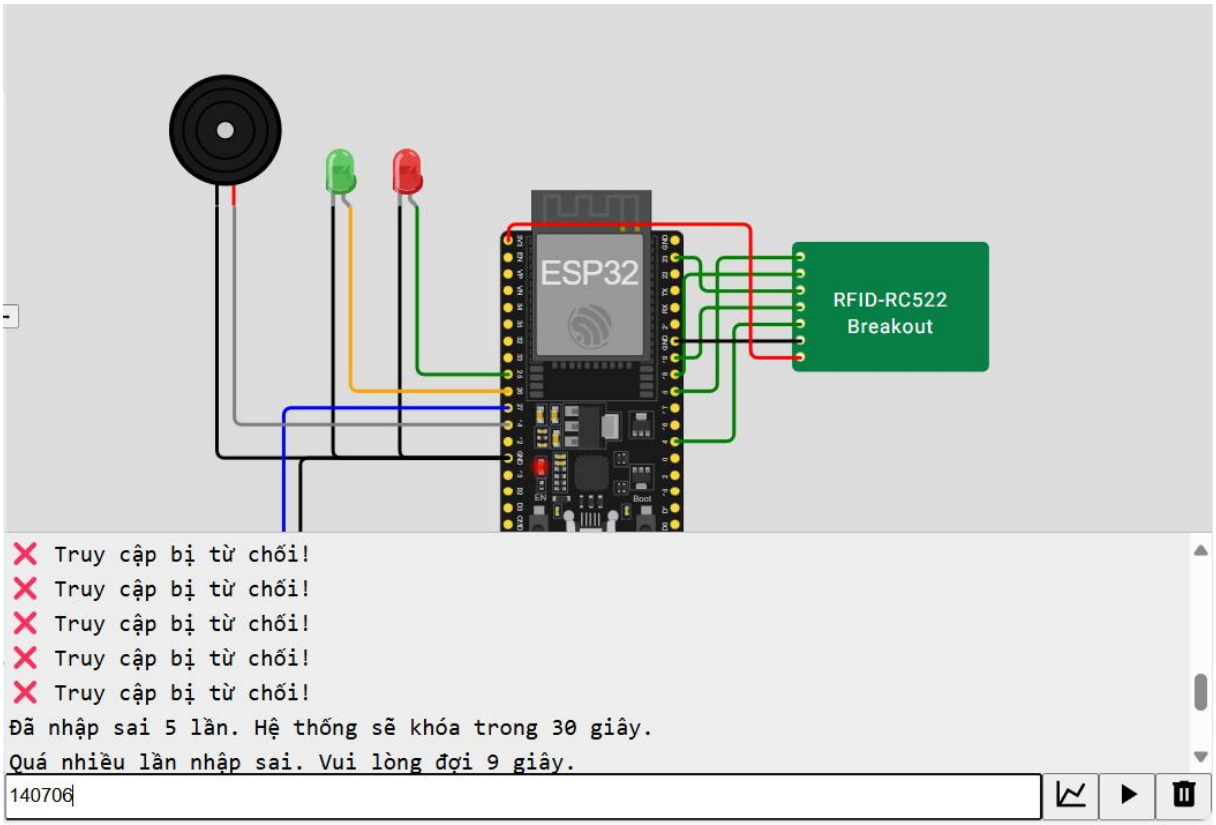
***Chạy trên Wokwi***

**Nhập:** 140706 (ID sai)



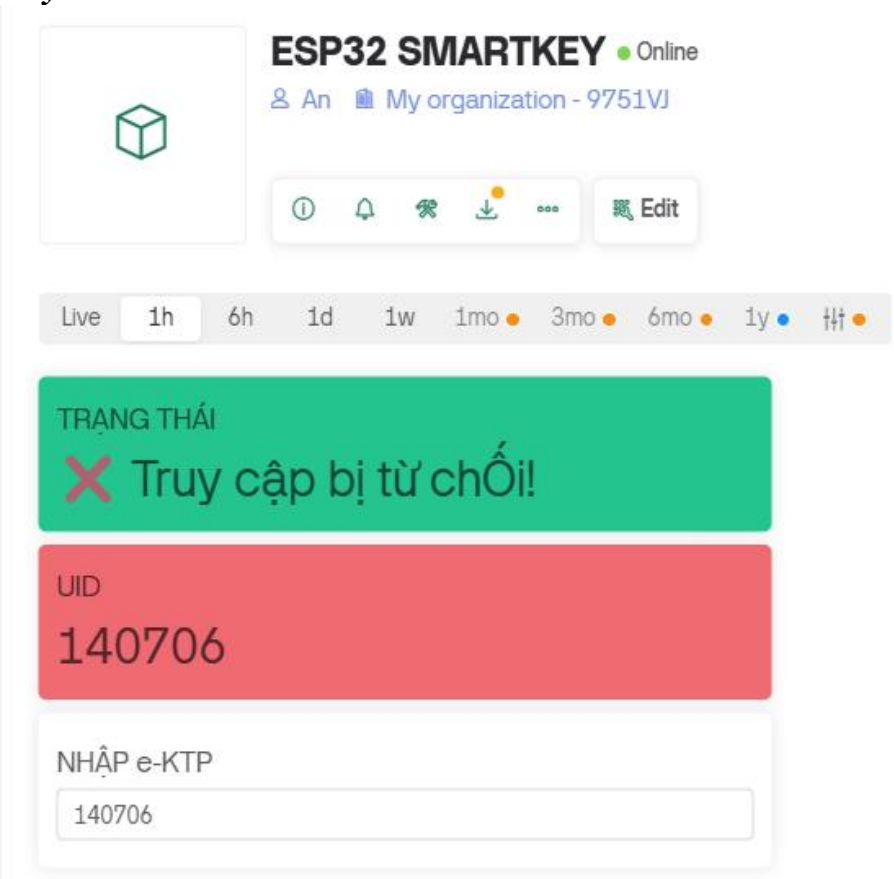

*Hình 13*

**Nhập sai quá 5 lần:**




*Hình 14*

***Chạy trên Blynk:***




*Hình 15*

### 2.4. Nguyên lý hoạt động của hệ thống:

#### 2.4.1. Kết nối WiFi và Blynk

- Kết nối WiFi: Thiết bị kết nối với mạng WiFi "Wokwi-GUEST" (mạng mô phỏng)
- Kết nối Blynk: Sử dụng token xác thực để kết nối với nền tảng Blynk
- Khởi tạo phần cứng: Cấu hình các chân GPIO cho relay, đèn LED, buzzer

#### 2.4.2. Nhận diện truy cập

Hệ thống nhận input từ 2 nguồn:

- Blynk App: Qua Virtual Pin V2 (Text Input)
- Serial Monitor: Qua cổng serial (115200 baud)

#### 2.4.3. Kiểm tra định dạng và xác thực

- Kiểm tra định dạng: Input phải là chuỗi 6 chữ số liền nhau (định dạng e-KTP giả lập)
- Xác thực: So sánh với ID hợp lệ ("123456" trong code mẫu)

#### 2.4.4. Xử lý kết quả

- Truy cập hợp lệ:

- Kích hoạt Relay (mở khóa)
- Bật đèn LED xanh và Buzzer
- Hiển thị thông báo "✅ Truy cập được chấp nhận!" trên Blynk và Serial
- Reset bộ đếm lần nhập sai

- Truy cập không hợp lệ:

- Bật đèn LED đỏ và buzzer
- Hiển thị thông báo "❌ Truy cập bị từ chối!" trên Blynk và Serial
- Tăng bộ đếm lần nhập sai

#### 2.4.5. Cơ chế bảo vệ chống brute-force

- Sau 5 lần nhập sai liên tiếp:

- Hệ thống khóa trong 30 giây
- Hiển thị thời gian đếm ngược còn lại
- Không nhận thêm input trong thời gian này

### 2.4.6. Giao tiếp hai chiều với Blynk

- Gửi dữ liệu lên Blynk:

- Trạng thái truy cập (V0)
- UID nhập vào (V1)

- Nhận lệnh từ Blynk: Qua Virtual Pin V2 (Text Input)

### 2.4.7. Phần cứng điều khiển

Relay: Điều khiển khóa điện (chân 27)

LED: Xanh (26) - truy cập hợp lệ, Đỏ (25) - truy cập không hợp lệ

Buzzer: Phát tín hiệu âm thanh (chân 14)

# KẾT LUẬN

Qua quá trình nghiên cứu và thực hiện đề tài "Hệ thống khóa cửa thông minh dựa trên ESP32 và RFID", em đã có cơ hội tìm hiểu sâu hơn về các công nghệ như vi điều khiển ESP32, cảm biến RFID RC522, relay, đèn LED và buzzer. Đặc biệt, việc ứng dụng nền tảng Blynk để điều khiển và giám sát hệ thống từ xa giúp nâng cao tính thực tiễn, hiện đại và tiện lợi cho người dùng.

Hệ thống đã được mô phỏng thành công trên nền tảng Wokwi, giúp tiết kiệm chi phí phần cứng và tạo điều kiện thuận lợi trong việc kiểm thử. Thông qua quá trình mô phỏng, hệ thống có khả năng xác thực người dùng dựa trên mã UID, điều khiển khóa cửa, phản hồi trạng thái truy cập bằng đèn LED và âm thanh, đồng thời thực hiện biện pháp bảo mật chống brute-force thông minh.

Mặc dù hệ thống mới chỉ ở mức mô hình cơ bản và còn hạn chế về tính năng, nhưng nó hoàn toàn có thể mở rộng, tích hợp thêm các chức năng nâng cao như lưu trữ lịch sử truy cập, gửi cảnh báo qua Telegram, hoặc sử dụng cảm biến vân tay, camera AI để tăng cường độ bảo mật.

Đề tài không chỉ giúp em nâng cao kiến thức và kỹ năng lập trình IoT, mà còn rèn luyện tư duy logic, khả năng giải quyết vấn đề thực tiễn. Em hy vọng trong tương lai sẽ có thể tiếp tục phát triển đề tài này thành một sản phẩm hoàn thiện phục vụ đời sống.

# TÀI LIỆU THAM KHẢO

[1] https://viettelstore.vn/tin-tuc/cong-nghe-rfid-la-gi-nguyen-ly-hoat-dong-va-ung-dung-cua-rfid

[2] https://arduinokit.vn/rfid-la-gi-huong-dan-su-dung-module-rfid-rc522-voi-arduino/

[3] https://vietmachine.com.vn/cach-hoat-dong-cua-rfid-va-cach-tao-khoa-cua-dua-tren-arduino.html

[4] https://iotstarters.com/building-an-rfid-rc522-access-system-with-ESP32/

[5] https://arduinokit.vn/rfid-la-gi-huong-dan-su-dung-module-rfid-rc522-voi-arduino/#google_vignette

[6] https://arduinokit.vn/he-thong-khoa-cua-thong-minh-su-dung-rfid-rc-522-va-arduino/

[7] http://arduino.vn/tutorial/1570-gioi-thieu-module-ESP32-va-huong-dan-cai-trinh-bien-dich-tren-arduino-ide

[8] https://khuenguyencreator.com/lap-trinh-esp32-tu-a-toi-z/

[9] https://khuenguyencreator.com/tong-quan-ve-so-do-chan-esp32-va-ngoai-vi/

[10] https://iotstarters.com/building-an-rfid-rc522-access-system-with-esp32/