	BỘ MÔN KỸ THUẬT MÁY TÍNH – VIỄN THÔNG **CƠ SỞ VÀ ỨNG DỤNG IOTS** **MMH: ITFA436064** **Thời gian thực hiện: 01 buổi**

Nhóm:
Lê Quang Thương -21161367
Võ Minh Thuận - 21161366
Lê Thị Tuyết Nhi - 21161344
Trần Thị Xuân Hy - 21161323
Bùi Khánh Hưng - 21161319

1. So sánh chuẩn truyền thông Wifi và chuẩn truyền thông Zigbee, Lora, Lorawan.

WIFI - Tốc độ truyền: wifi thường có tốc độ truyền dữ liệu cao (lên đến 1.3mbps/s), phù hợp cho các ứng dụng yêu cầu băng thông lớn, truyền tải dữ liệu nhanh. - Phạm vi hoạt động: thường hạn chế trong một khu vực nhất định (dùng cục router) như trong một căn nhà, tầng siêu thị,... - Tiêu thụ năng lượng: khá lớn so với các chuẩn truyền thông khác, đặc biệt là trong các ứng dụng yêu cầu hoạt động liên tục.	***ZIGBEE*** - Tốc độ truyền dữ liệu: thấp và không cao như wifi (250kbps/s), phù hợp cho các ứng dụng yêu cầu băng thông thấp như cảm biến. - Tiêu thụ năng lượng: tiêu thụ ít năng lượng, phù hợp cho các thiết bị IoT yêu cầu tuổi thọ pin dài. - Phạm vi hoạt động: khoảng cách ngắn (tối đa 75m) chủ yếu dùng môi trường trong nhà - Băng tần: 2.4GHz - Chi phí thấp
LORA - Tốc độ truyền: thấp (300mpbs/s), phù hợp với những ứng dụng có phạm vi truyền tải rộng, khả năng chống nhiễu vượt trội. - Tiêu thụ năng lượng: tiêu thụ ít năng lượng, phù hợp cho các thiết bị IoT yêu cầu tuổi thọ pin dài (thời lượng pin có thể lên đến 10 năm) - Phạm vi hoạt động: lớn (đến 15km) - Băng tần: 2.4GHz trên toàn thế giới - Chi phí thấp	***LORAWAN***: *là giao thức mạng cải tiến từ Lora* - Tốc độ truyền: 10mbps/s cho phép kết nối thiết bị sử dụng Lora - Tiêu thụ năng lượng: ít tốn năng lượng (20dBm) - Phạm vi hoạt động: truyền dẫn tối đa 10km - Cung cấp định danh, quản lý năng lượng và bảo mật - Cung cấp mức độ bảo mật như xác thực thiết bị, mã hoá dữ liệu

2. So sánh thông số kỹ thuật của ESP32 và ESP8266

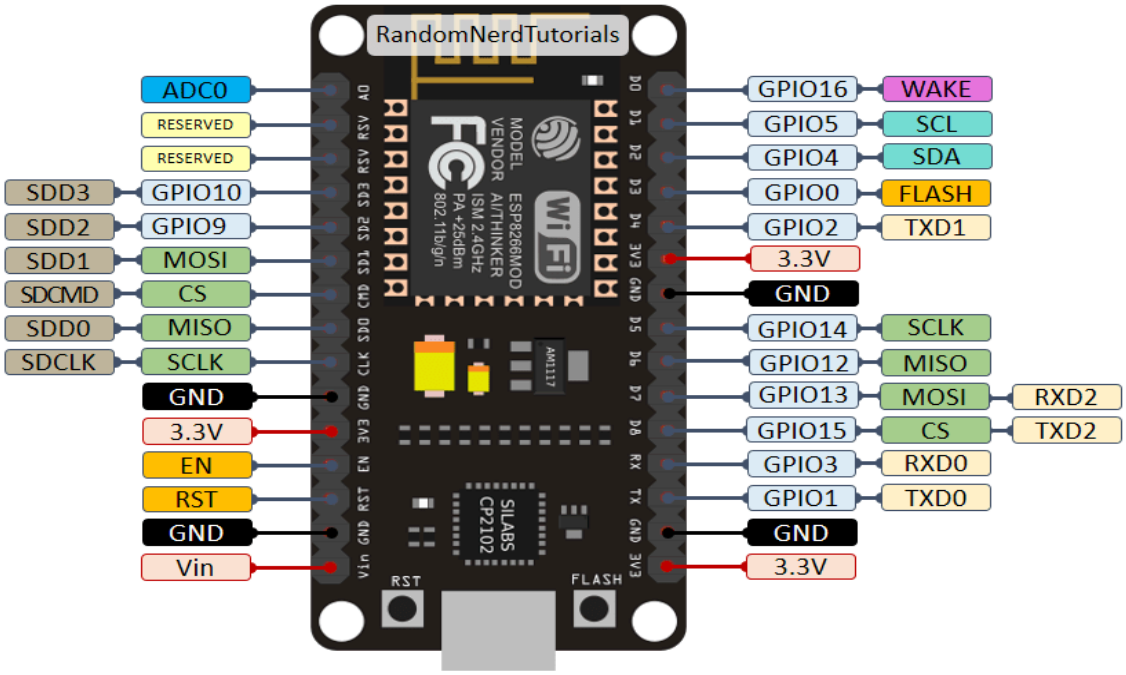
Giống : ESP32 và ESP8266 là hai vi điều khiển phổ biến được sử dụng trong các dự án IoT. Cả hai đều có khả năng kết nối Wi-Fi, giá cả phải chăng và dễ sử dụng. Tuy nhiên, chúng cũng có một số điểm khác biệt quan trọng.

Khác:

Tính năng	ESP32	ESP8266
Bộ xử lý	Lõi kép Xtensa 160 MHz	Lõi đơn Tensilica Xtensa LX106 80 MHz
Bộ nhớ	4 MB Flash, 520 KB SRAM	4 MB Flash, 80 KB SRAM
Kết nối	Wi-Fi 802.11 b/g/n, Bluetooth 4.2 BLE	Wi-Fi 802.11 b/g/n
Chân GPIO	34	17
Giao tiếp	UART, SPI, I2C, JTAG, SDIO, TWAI	UART, SPI, I2C, JTAG, SDIO
Cảm biến	Cảm biến cảm ứng, hiệu ứng Hall, cảm biến nhiệt độ	Không có
Mức tiêu thụ điện năng	Cao hơn	Thấp hơn
Giá thành	Cao hơn	Thấp hơn

3. Sơ đồ chân kết nối ESP32 (hoặc ESP8266) (**lập bảng** chức năng các chân).

Chân	Chức năng của chân
VCC	Cung cấp điện áp 3.3V cho ESP hoạt động
GND	Mass
GPIO0(TX)	Chân truyền dữ liệu UART
GPIO2,GPIO4,GPIO5,GPIO6,GPIO7,GPIO8,GPIO9, GPIO10	Chân GPIO đa năng
GPIO12(MISO)	Chân nhận dữ liệu SPI
GPIO13(MOSI)	Chân gửi dữ liệu SPI
GPIO15(SCLK)	Chân xung nhịp SPI
GPIO14(SS)	Chân chọn chip SPI
Reset	Chân reset ESP266
RX	Chân nhận dữ liệu UART



4. Các bước cài đặt và giao tiếp ESP32 (hoặc ESP 8266) với phần mềm Arduino IDE

- Cài board Esp32 vào Arduino Ide.

Bước 1: Mở Arduino Ide -> chọn file -> chọn Preferences.

Bước 2: sao chép dòng trên

"https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json" vào ô Additional Board Manager URLs. Rồi chọn Ok.

Bước 3: Chọn Tool-> Boards Manager -> ghi ESPP32 -> Chọn Install.

- Giao tiếp Esp32 với Arduino Ide

Bước 1: Cắm usb kết nối Esp32 với Laptop chứa Arsuino Ide.

Bước 2: Chọn tool -> board -> chọn board Esp32 mà bạn đã kết nối với Laptop.

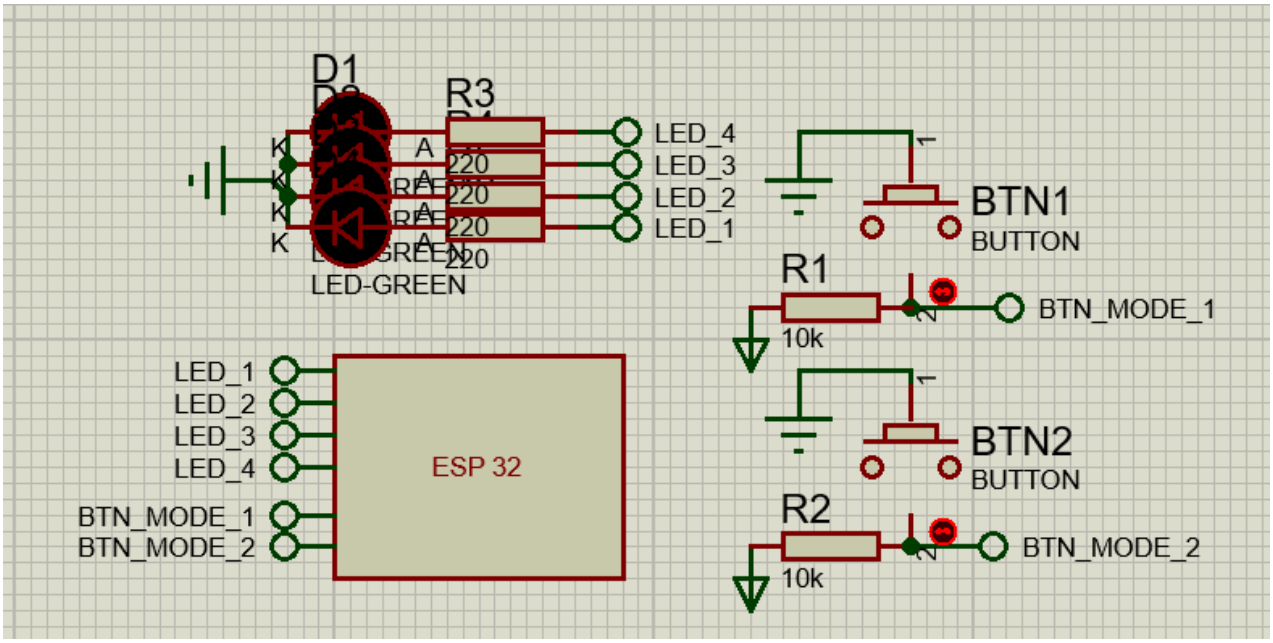
Bước 3: Chọn tool -> Port -> chọn cổng com nhận diện của Esp32 với laptop.

Bước 4: Lập trình điều khiển Esp

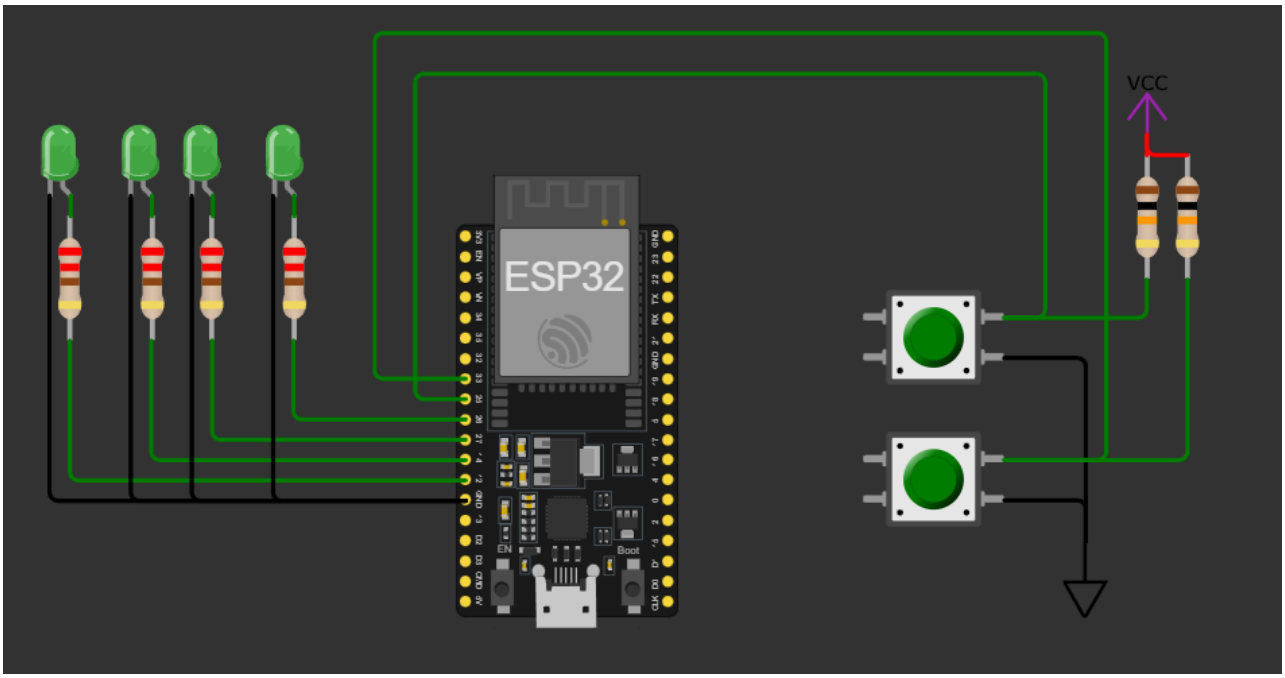
Bước 5: Chọn Upload và chờ vài giây để tải chương trình vào Esp32.

Bước 6: Nếu thành công sẽ thông báo Done Uploading trên terminal của Arduino Ide.

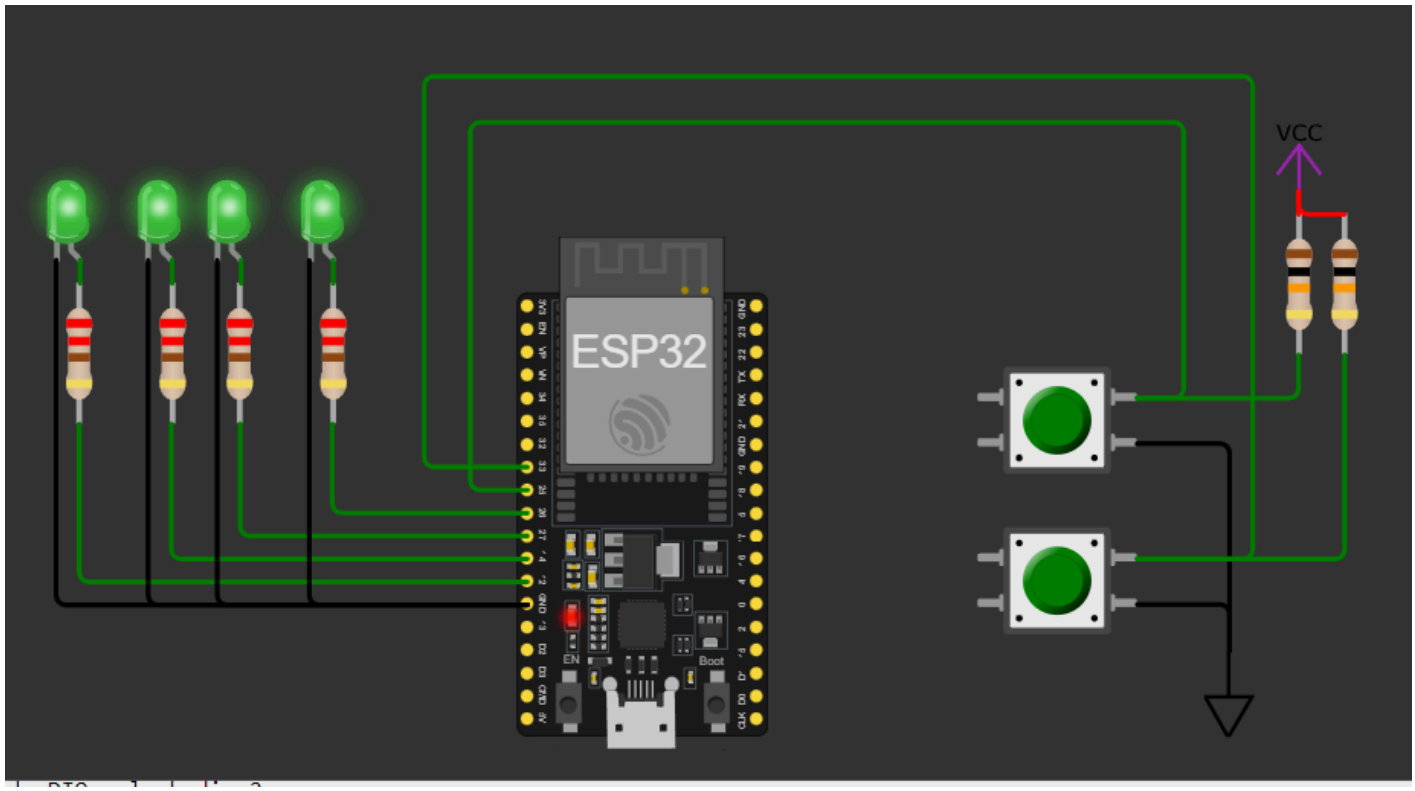
5. Các **bước thực hiện** và **giải thích code** chương trình chớp tắt LED (4 Led) kết hợp giao tiếp nút nhấn để chọn Mode chớp tắt trên ESP32 (hoặc ESP8266) và hình ảnh chụp các kết quả.
Vẽ sơ đồ nguyên lý của mạch.



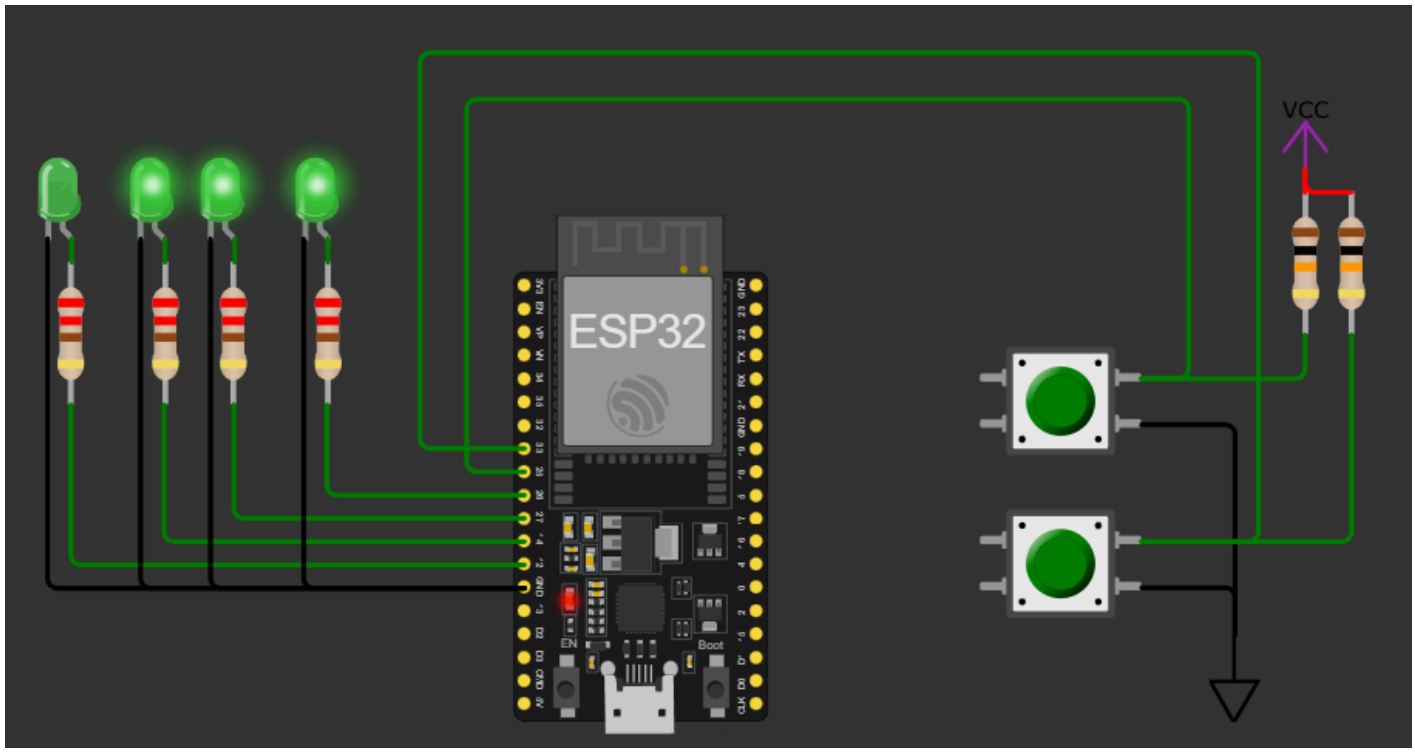
Sơ đồ kết nối



Hình kết quả mode 1



Hình kết quả mode 2



Chương trình code và giải thích

#define led_1 26

#define led_2 27

#define led_3 14

#define led_4 12

```
#define btn_mode_1 25
#define btn_mode_2 33
bool state_mode_1 = false;
bool state_mode_2 = false;
int count;
int ddta[4] = {led_1, led_2, led_3, led_4};
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
  pinMode(led_1, OUTPUT);
  pinMode(led_2, OUTPUT);
  pinMode(led_3, OUTPUT);
  pinMode(led_4, OUTPUT);
  pinMode(btn_mode_1, INPUT);
  pinMode(btn_mode_2, INPUT);
  count = 0;
}
```

```
/*
Hàm kiểm tra nút nhấn
Khi nhấn nút hàm kiểm tra nút nhấn nếu đảo giá trị đọc bằng 1 thì chờ 20ms và kiểm tra bằng 1 lại nếu bằng 1 thì
chờ nhả phím khi nhả phím thì trả về true, các trường hợp còn lại trả về false.
*/
bool check_btn(int pinName){
  if(!digitalRead(pinName)){
    delay(20);
    if(!digitalRead(pinName)){
      while(!digitalRead(pinName));
      return true;
    }
    return false;
  }
  return false;
}

void loop() {
  // put your main code here, to run repeatedly:

  // Kiểm tra nút nhấn 1 nếu có nhấn thì chuyển mode 1
  if(check_btn(btn_mode_1) == true){
    state_mode_1 = true;
    state_mode_2 = false;
  }
  // Kiểm tra nút nhấn 2 nếu có nhấn thì chuyển mode 2
  if(check_btn(btn_mode_2) == true){
    state_mode_1 = false;
    state_mode_2 = true;
  }
```

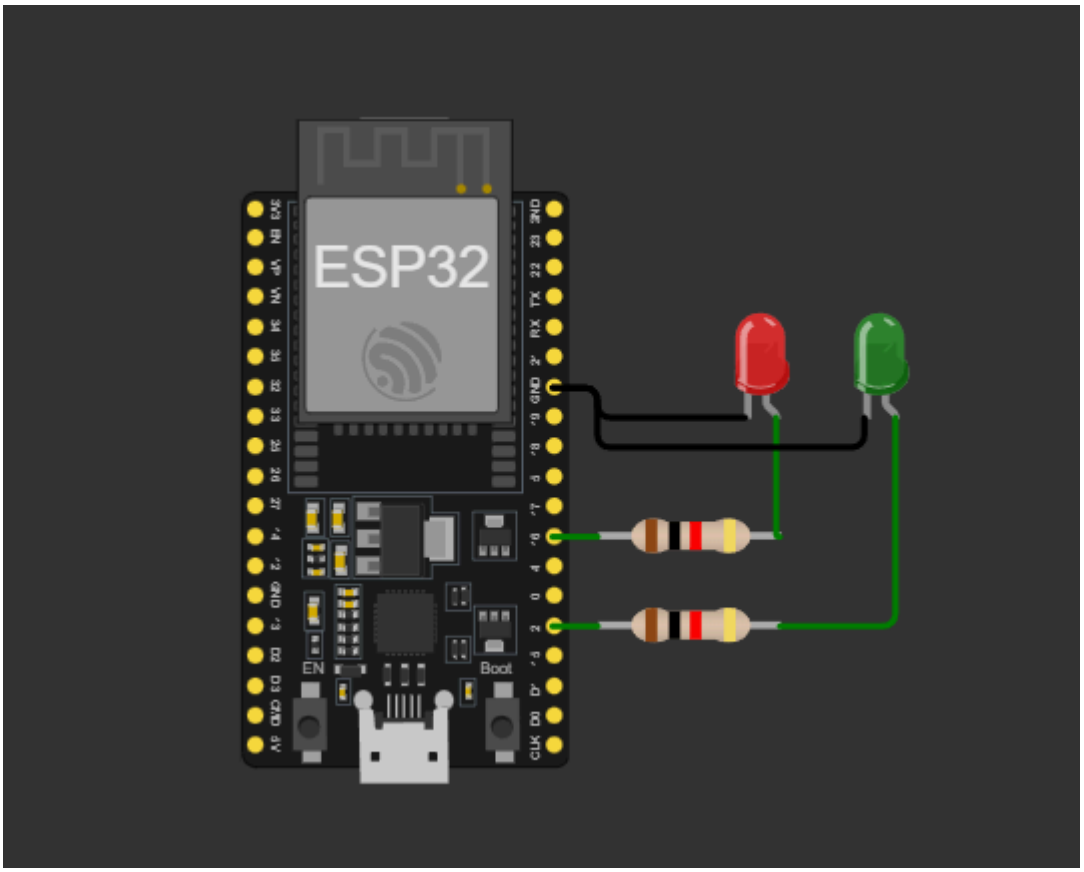
```
  // Mode 1 thì 4 led chớp tắt 4leds 300ms
  if(state_mode_1 == true){
    digitalWrite(led_1, 1);
    digitalWrite(led_2, 1);
    digitalWrite(led_3, 1);
    digitalWrite(led_4, 1);
    delay(300);
    digitalWrite(led_1, 0);
    digitalWrite(led_2, 0);
    digitalWrite(led_3, 0);
    digitalWrite(led_4, 0);
    delay(300);
  }
  // Mode 2 thì 4 sáng dần tắt hết.
  if(state_mode_2 == true){
    if(count < 5){
      delay(300);
      digitalWrite(ddta[count],1);
      count++;
    }
    else{
      digitalWrite(led_1, 0);
      digitalWrite(led_2, 0);
      digitalWrite(led_3, 0);
      digitalWrite(led_4, 0);
      count = 0;
    }
  }
}
```

Video chạy code:

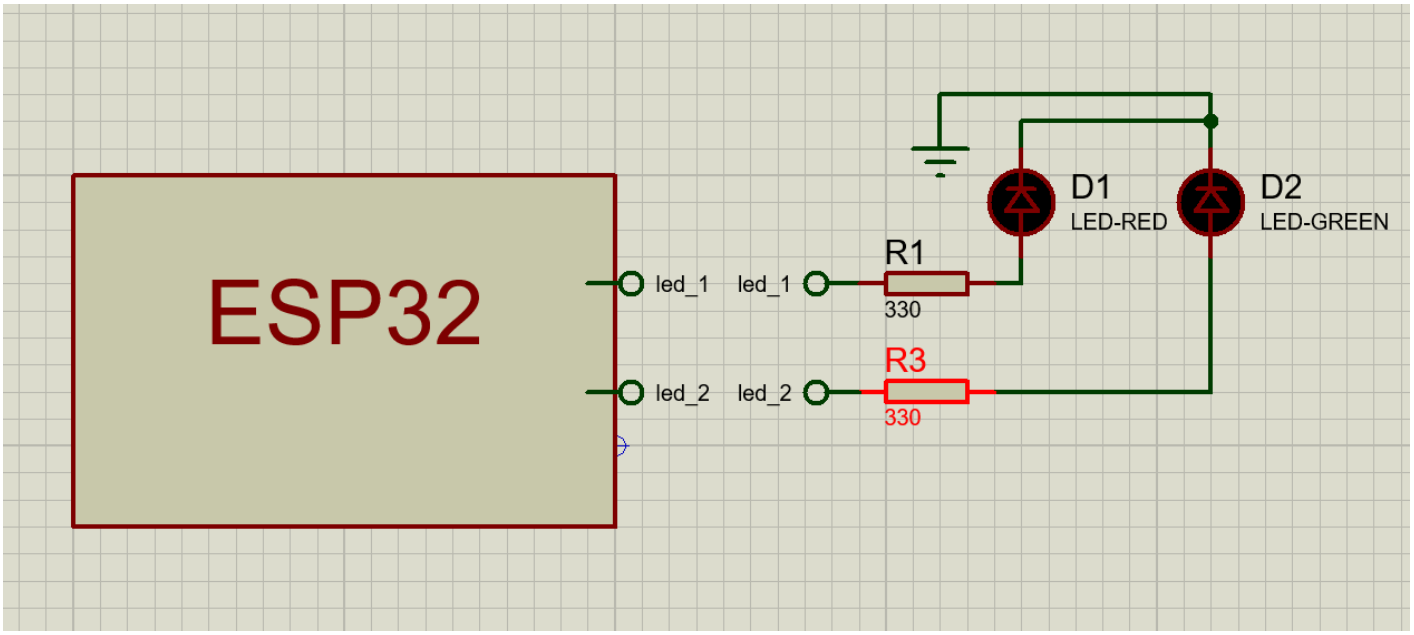
https://drive.google.com/file/d/1Ocukw3-VD9Wto0MLzCjf0aJi7E4RnzxG/view?usp=drive_link

6. Thực hiện giao tiếp ESP32 và 2 Led đơn điều khiển chớp tắt 2 Led (có sử dụng FreeRTOS). Giải thích code và chụp hình ảnh (kèm clip nếu có) kết quả thực hiện.

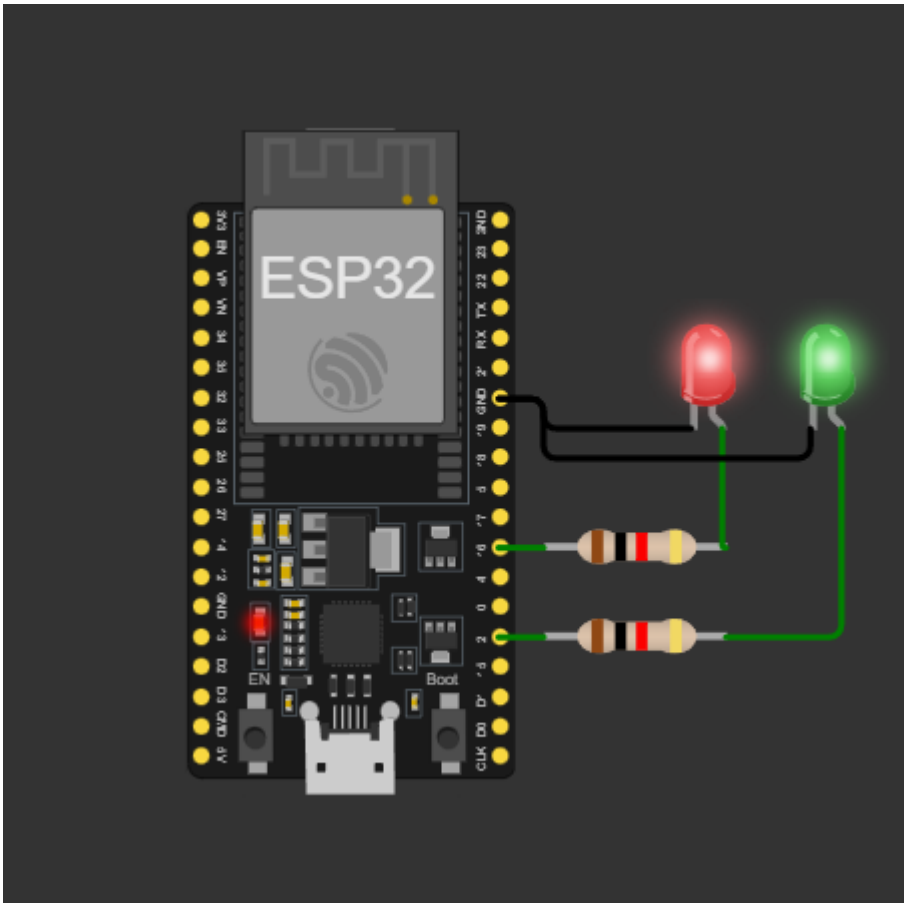
   Sơ đồ nối dây:



   Sơ đồ nguyên lý:



Ảnh kết quả:



Chương trình:

```
#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
// Định nghĩa chân GPIO cho Led
#define LED1_GPIO GPIO_NUM_2
#define LED2_GPIO GPIO_NUM_16
// Task điều khiển Led 1
void taskLed1(void *pvParameters) {
  (void)pvParameters;
  while (true) {
    // Bật Led 1
    digitalWrite(LED1_GPIO, HIGH);
    vTaskDelay(1000); // Chờ 1 giây
```

```
    // Tắt Led 1
    digitalWrite(LED1_GPIO, LOW);
    vTaskDelay(1000); // Chờ 1 giây
  }
}
// Task điều khiển Led 2
void taskLed2(void *pvParameters) {
(void)pvParameters;
  while (true) {
    // Bật Led 2
    digitalWrite(LED2_GPIO, HIGH);
    vTaskDelay(500); // Chờ 0.5 giây
    // Tắt Led 2
    digitalWrite(LED2_GPIO, LOW);
    vTaskDelay(500); // Chờ 0.5 giây
  }
}
void setup() {
  // Khởi tạo Serial
  Serial.begin(115200);
  // Thiết lập GPIO cho Led
  pinMode(LED1_GPIO, OUTPUT);
  pinMode(LED2_GPIO, OUTPUT);
  // Tạo task
  xTaskCreate(taskLed1, "Led1", 2048, NULL, 1, NULL);
  xTaskCreate(taskLed2, "Led2", 2048, NULL, 1, NULL);
}
void loop() {
  // Code loop không cần thiết trong trường hợp này
}
```

link video chạy thử :
https://drive.google.com/file/d/1OayXr7zTLqFv0yll5QOHgnmwZM6v_LjQ/view?usp=drive_link

Hình ảnh làm việc nhóm.