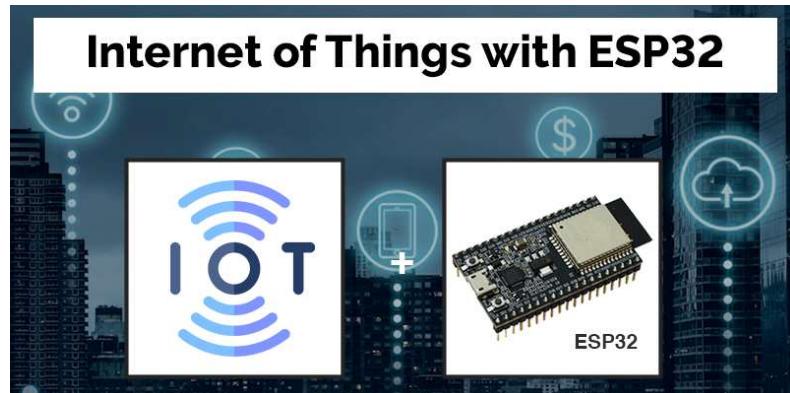




TÀI LIỆU THỰC HÀNH

CÔNG NGHỆ IoT

Giảng viên: Đường Khánh Sơn



MỤC LỤC

MỤC LỤC.....	0
THIẾT BỊ THỰC HÀNH.....	2
BÀI 1 : LED ĐƠN.....	8
BÀI 2 : CHỨC NĂNG INPUT, OUTPUT VÀ NGẮT NGOÀI TRÊN ESP32.....	13
BÀI 3 : LẬP TRÌNH GIAO TIẾP I2C, ADC VỚI CÁC CẢM BIẾN THÔNG DỤNG.....	24
BÀI 4 : NỀN TẢNG BLYNK VỚI ESP32.....	42
BÀI 5 : ARDUINO CLOUD với ESP32.....	61
BÀI 6 : THINGSPEAK VỚI ESP32	70
TÀI LIỆU THAM KHẢO	76

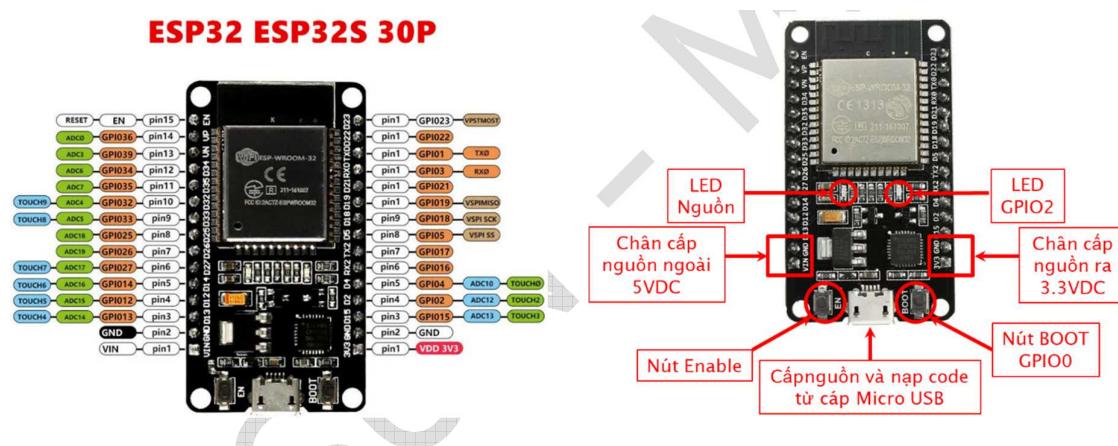
THIẾT BỊ THỰC HÀNH

ESP32 là một mạch điều khiển mạnh mẽ được phát triển bởi Espressif Systems, sở hữu khả năng kết nối không dây WiFi lẫn Bluetooth. Ngoài ra, chúng cũng tích hợp nhiều linh kiện điện tử đa dạng cho dự án như:

- Ăng ten tích hợp
 - Bộ khuếch đại công suất
 - Bộ khuếch đại tiếng ồn thấp
 - Module quản lý năng lượng...

Sơ đồ chân ESP32

ESP32 có nhiều chân để kết nối với các thiết bị ngoại vi. Hiểu rõ sơ đồ chân là rất quan trọng để sử dụng ESP32 một cách hiệu quả.



Vị trí led, nút ấn và chân cấp nguồn trên board ESP32 DEVKIT V1

XEM THÊM TÀI LIỆU CHI TIẾT SƠ ĐỒ CHÂN CỦA ESP32

Cài đặt phần mềm và nạp chương trình cho esp32



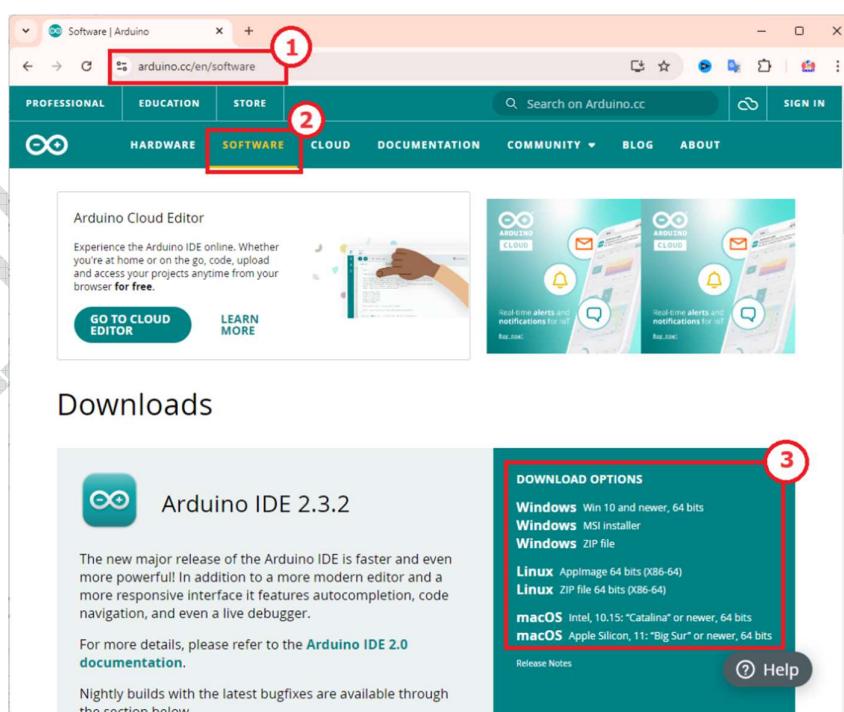
Để bắt đầu lập trình ESP32, chúng ta cần cài đặt một số phần mềm cần thiết:

1. Cài đặt driver giao tiếp máy tính

- Kết nối ESP32 với máy tính bằng cáp USB.
- Tìm kiếm và cài đặt driver giao tiếp máy tính phù hợp với hệ điều hành của bạn CP2102 hoặc CH340.
- Link download driver: [CP2102](#)

2. Tải và cài đặt Arduino IDE

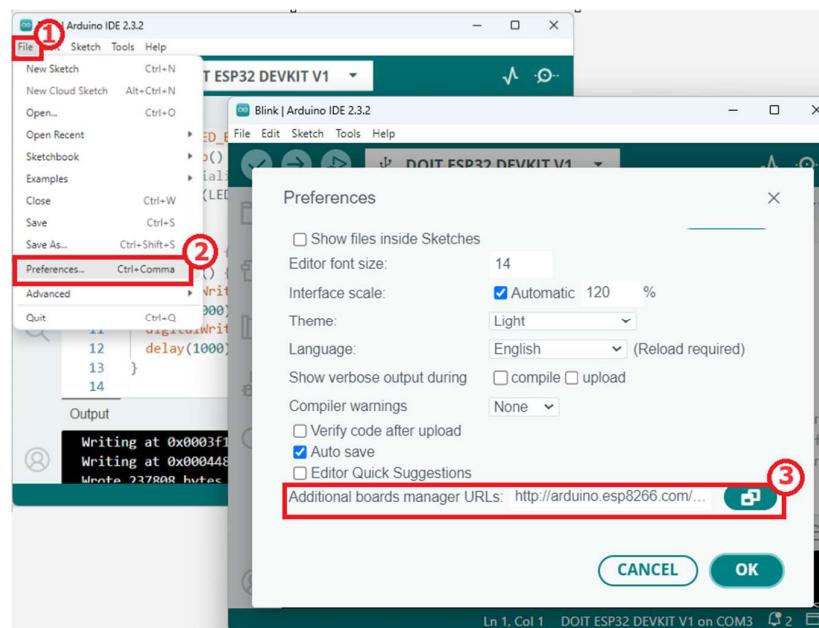
- Truy cập trang web chính thức của Arduino IDE: <https://www.arduino.cc/en/software>
- Tải xuống phiên bản phù hợp với hệ điều hành của bạn.
- Chạy trình cài đặt và làm theo hướng dẫn trên màn hình.



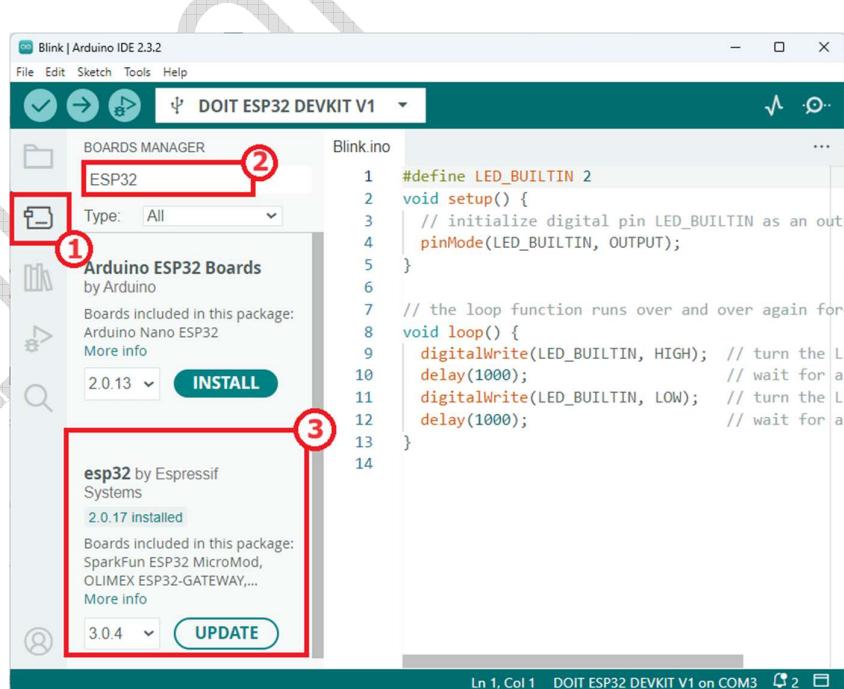
3. Cài đặt board ESP32

Mở Arduino IDE vào mục **File ->Preferences** ở mục **Additional boards manager URLs** nhập đường dẫn sau:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



Vào **Tools ->Board -> Boards manager** tìm kiếm và cài đặt board ESP32 vào phần mềm Arduino IDE.



Nạp chương trình cho ESP32

Sau khi cài đặt xong phần mềm, chúng ta sẽ tiến hành nạp chương trình cho ESP32.

Bắt đầu lập trình ESP32 cơ bản – Hello World

Dự án làm quen đầu tiên với ESP32 : dự án “Hello World”.

Chuẩn bị

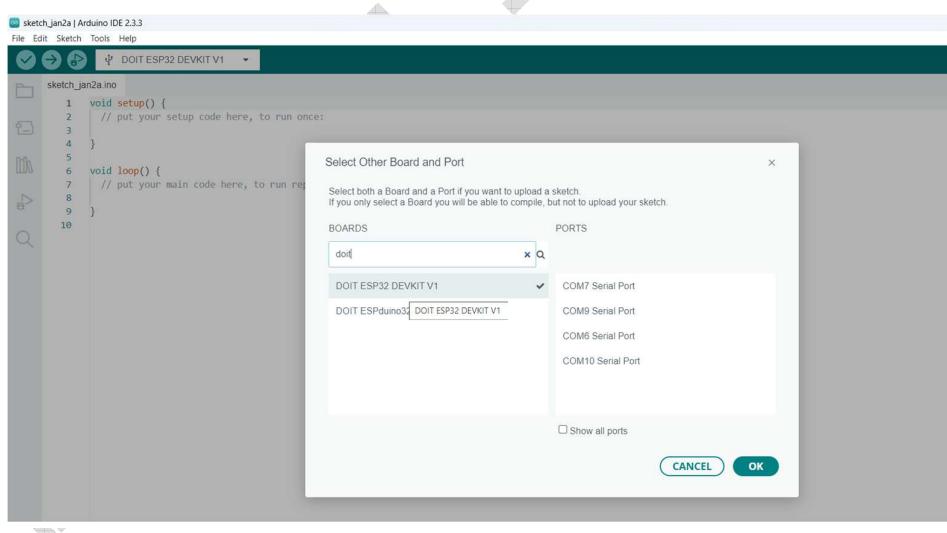
Mạch ESP32

- Dây cáp Type C
- Breadboard
- Dây Jumper

Hướng dẫn ESP32 Hello World trên Serial Monitor

Chương trình mặc định

- Mở Arduino IDE và chọn board như hình dưới:



- Kết nối ESP32 với máy tính qua cáp USB
- Chọn cổng COM và mạch ESP32
- Copy đoạn code sau và paste vào Arduino IDE

```

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

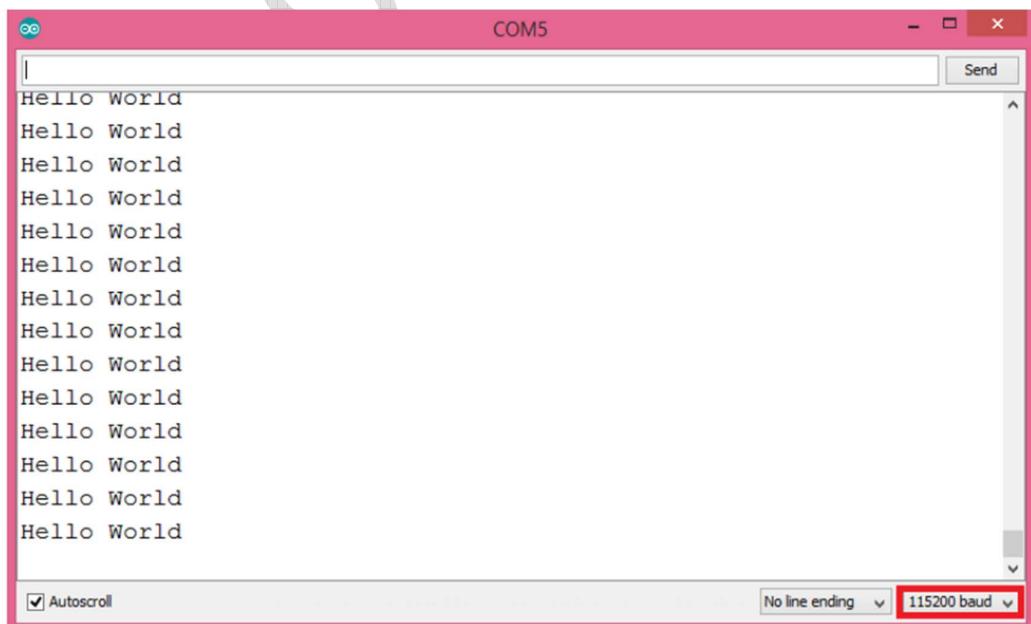
void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!");
  delay(500);
}

```

- Click vào nút nạp chương trình (Upload) trên Arduino IDE
- Mở Serial Monitor trên Arduino IDE



- Trên màn hình Serial sẽ hiển thị chuỗi kết quả Hello World như bên dưới, vậy là dự án ESP32 Hello World đã thành công!



Thực hiện dự án ESP32 Hello World thành công

Tùy chỉnh code

- Thay đổi “Hello World” trong đoạn code thành bất kỳ dòng chữ nào, ví dụ như “CONG NGHE IoT”
- Upload code vào mạch ESP32
- Xem kết quả hiển thị trên Serial Monitor.

ĐK SON - MLAB

BÀI 1 : LED ĐƠN

1.1.Nhiệm vụ

Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào Esp32 và sử dụng mô hình thí nghiệm để kiểm chứng.

- Điều khiển thiết bị ngoại vi bằng các GPIO của Esp32.
- Điều khiển việc hiển thị bằng cách sử dụng LED đơn.
- SV hoàn thành các bài tập 1.1 đến 1.4.

1.2.Yêu cầu

- Nắm vững các tập lệnh điều khiển GPIO.
- Biết cách viết các chương trình điều khiển LED đơn ở các chế độ khác nhau.
- Nắm được sơ đồ và nguyên lý hoạt động của khói LED đơn .
- Biết cách viết các chương trình tạo thời gian trễ với các khoảng thời gian bất kỳ.

1.3.Giới thiệu chung

LED (viết tắt của Light Emitting Diode, có nghĩa là điốt phát quang) là các điốt có khả năng phát ra ánh sáng hay tia hồng ngoại, tử ngoại. Cũng giống như điốt, LED được cấu tạo từ một khói bán dẫn loại p ghép với một khói bán dẫn loại n.

Hoạt động của LED giống với nhiều loại điốt bán dẫn. Khối bán dẫn loại p chứa nhiều lỗ trống tự do mang điện tích dương nên khi ghép với khói bán dẫn n (chứa các điện tử tự do) thì các lỗ trống này có xu hướng chuyển động khuếch tán sang khói n. Cùng lúc khói p lại nhận thêm các điện tử (điện tích âm) từ khói n chuyển sang. Kết quả là khói p tích điện âm (thiểu hụt lỗ trống và dư thừa điện tử) trong khi khói n tích điện dương (thiểu hụt điện tử và dư thừa lỗ trống). Ở biên giới hai bên mặt tiếp giáp, một số điện tử bị lỗ trống thu hút và khi chúng tiến lại gần nhau, chúng có xu hướng kết hợp với nhau tạo thành các nguyên tử trung hòa.

Quá trình này có thể giải phóng năng lượng dưới dạng ánh sáng (hay các bức xạ điện từ có bước sóng gần đó).

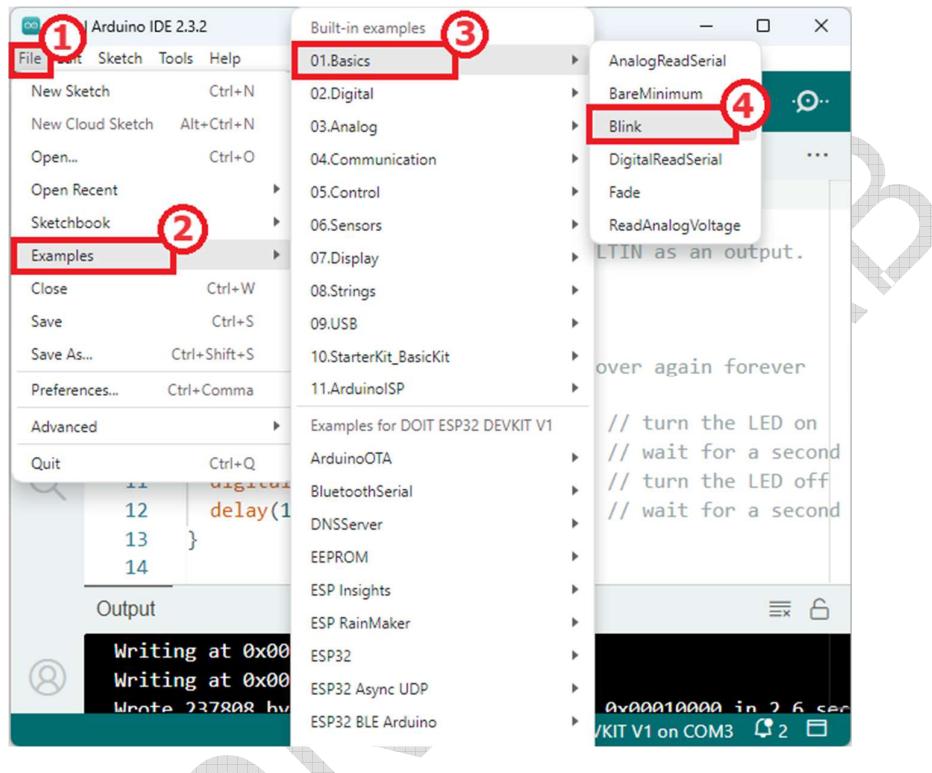
Tùy theo mức năng lượng giải phóng cao hay thấp mà bước sóng ánh sáng phát ra khác nhau (tức màu sắc của LED sẽ khác nhau). Mức năng lượng (và màu sắc của LED) hoàn toàn phụ thuộc vào cấu trúc năng lượng của các nguyên tử chất bán dẫn. LED thường có điện thế phân cực thuận cao hơn điốt thông thường, trong khoảng 1,5 đến 3 V. Nhưng điện thế phân cực nghịch ở LED thì không cao. Do đó, LED rất dễ bị hư hỏng do điện thế ngược gây ra.

Loại LED	Điện thế phân cực thuận
Đỏ	1,4 - 1,8V
Vàng	2 - 2,5V
Xanh lá cây	2 - 2,8V

- ❖ **Bài tập 1.1:** Viết chương trình điều khiển chớp tắt led ở chân GPIO2 của Esp32.

B1. Mở ví dụ code mẫu

- Trong Arduino IDE, vào mục **File -> Examples -> Basics** mở ví dụ code mẫu “**Blink**”.



B2. Sửa đổi code (nếu cần)

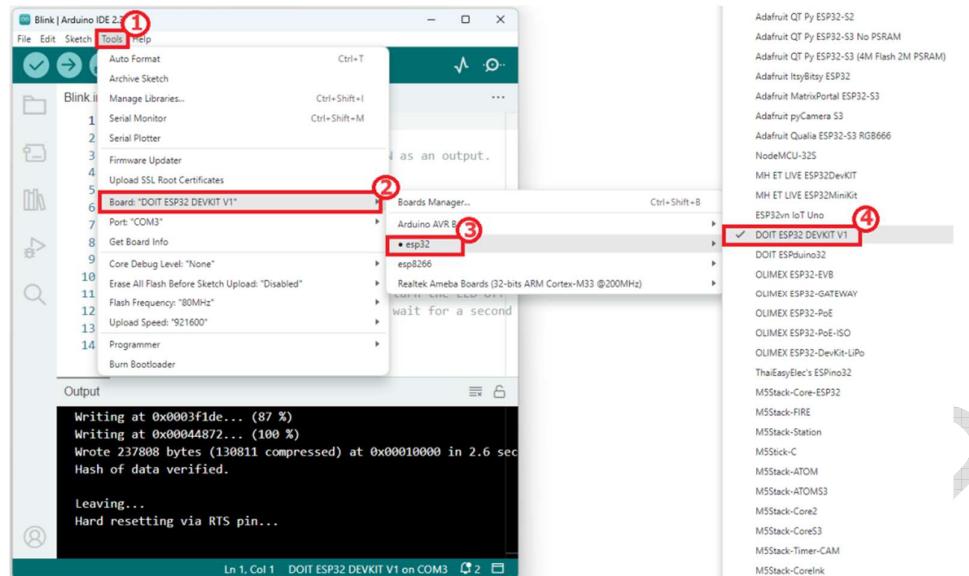
- Nếu muốn điều khiển chân khác trên ESP32, thay đổi giá trị của biến **LED_BUILTIN** trong code.

```
#define LED_BUILTIN 2
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

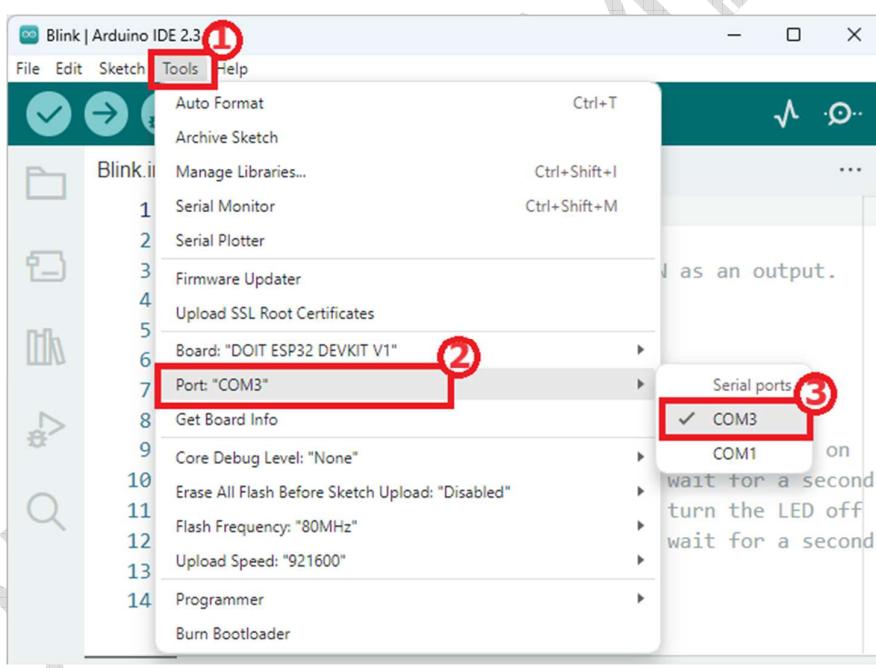
// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on
    delay(1000);                  // wait for a second
    digitalWrite(LED_BUILTIN, LOW); // turn the LED off
    delay(1000);                  // wait for a second
}
```

B3. Chọn board và cổng COM

Vào Tools -> Board và chọn “**DOIT ESP32 DEVKIT V1**”.

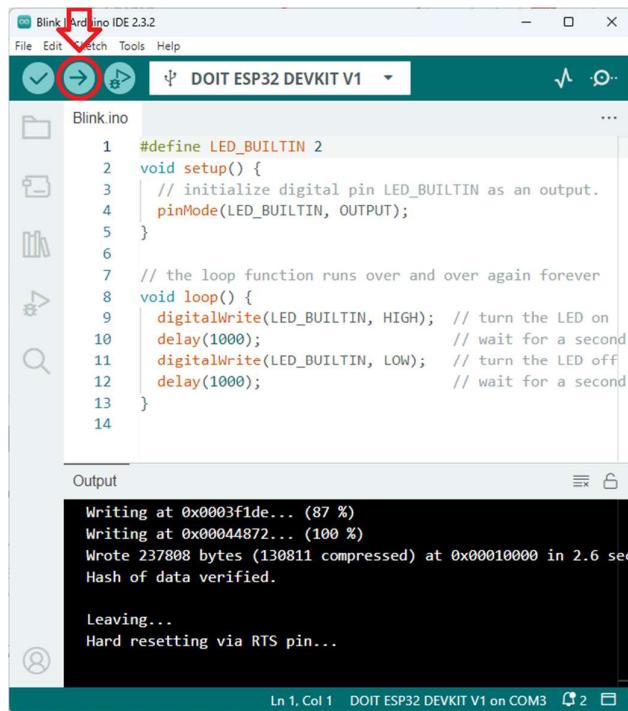


Vào Tools -> Port và chọn cổng COM đang kết nối với ESP32.



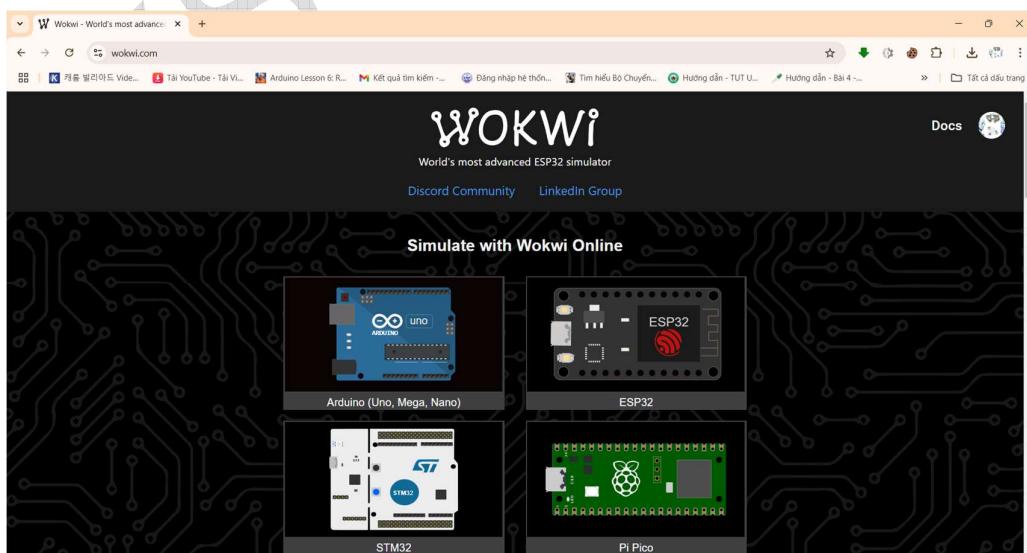
B4. Nạp chương trình

- Nhấn nút “Upload” trên Arduino IDE để nạp chương trình vào ESP32.
- Đợi quá trình nạp chương trình hoàn tất.
- Nhấn nút EN trên ESP32 để khởi động lại.



Lúc này led trên board ESP32 đã bật tắt theo như code lập trình.

Trang web hỗ trợ mô phỏng Esp32 :
<https://wokwi.com/>



- ❖ Bài tập 1.2 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển chớp tắt led ở chân GPIO2 với thời gian delay 1s.
- ❖ Bài tập 1.3 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển 3 led đỏ, vàng, xanh với thời gian sáng led đỏ 5s, sáng led vàng 1s, sáng led xanh 4s.
- ❖ Bài tập 1.4 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển 6 led sáng tắt dần.

DK SON - MLAB

BÀI 2 : CHỨC NĂNG INPUT, OUTPUT VÀ NGẮT NGOÀI TRÊN ESP32

2.1.Nhiệm vụ

Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào Esp32 và sử dụng mô hình thí nghiệm để kiểm chứng.

- Lập trình các chức năng input, output và ngắt ngoài của Esp32.
- Điều khiển và sử dụng các linh kiện cơ bản như nút nhấn, relay, còi...
- SV hoàn thành các bài tập 2.1 đến 2.5.

2.2.Yêu cầu

- Nắm vững các tập lệnh điều khiển input, output và ngắt ngoài của Esp32.
- Biết cách viết các chương trình điều khiển các thiết bị thông dụng như led, còi, đèn 220 VAC...
- Nắm được sơ đồ và nguyên lý hoạt động của các thiết bị LCD, relay, button, buzzers...

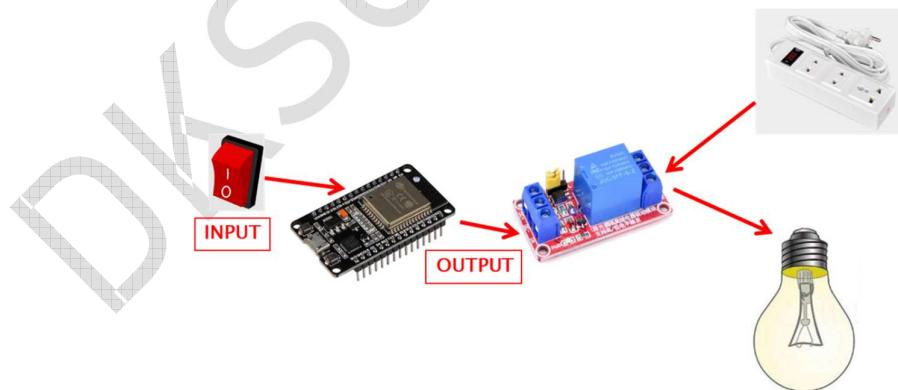
2.3. Giới thiệu

Input và output cho phép chúng ta tương tác với thế giới bên ngoài, đọc dữ liệu từ các cảm biến và điều khiển các thiết bị. Ngắt ngoài cung cấp một cách hiệu quả để phản ứng nhanh chóng với các sự kiện xảy ra, ví dụ như khi một nút nhấn được ấn hoặc một cảm biến phát hiện chuyển động.

Chức năng Input và Output

Input: Dùng để đọc dữ liệu từ các thiết bị ngoại vi như nút nhấn, cảm biến,...

Output: Dùng để điều khiển các thiết bị như đèn LED, động cơ, relay...



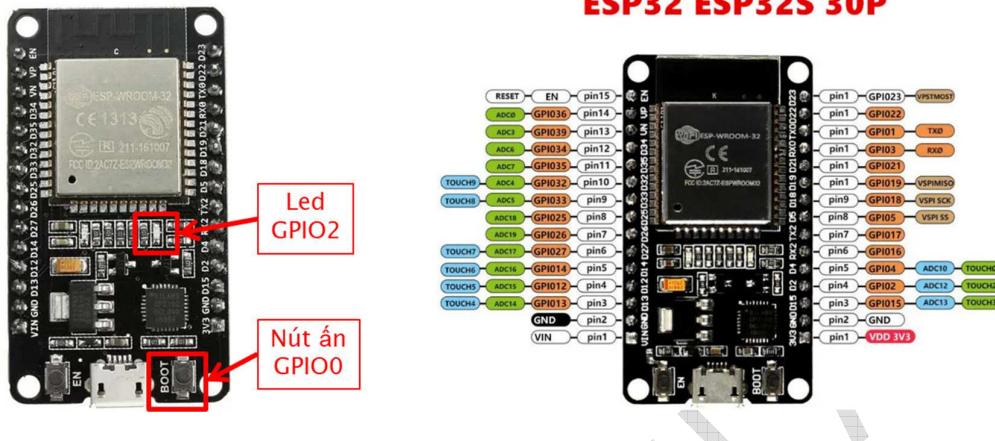
Các lệnh thiết lập chức năng input/output:

pinMode(pin, mode): Thiết lập chế độ cho chân (**INPUT**, **INPUT_PULLUP**, **INPUT_PULLDOWN** hoặc **OUTPUT**).

digitalRead(pin): Đọc giá trị số (0 hoặc 1) từ chân input/output.

digitalWrite(pin, value): Viết giá trị số (0 hoặc 1) ra chân output.

Ví dụ: Dùng chức năng input và output trên ESP32 điều khiển đèn LED bằng nút nhấn



Code chương trình:

```
const int buttonPin = 0; // Chân kết nối cho nút nhấn
const int ledPin = 2; // Chân kết nối cho đèn LED

void setup() {
    pinMode(buttonPin, INPUT_PULLUP); // Đặt chế độ chân là INPUT_PULLUP
    pinMode(ledPin, OUTPUT); // Đặt chế độ chân ledPin là OUTPUT
}

void loop() {
    int buttonState = digitalRead(buttonPin); // Đọc trạng thái nút nhấn
    if (buttonState == HIGH) {
        digitalWrite(ledPin, HIGH); // Bật đèn LED
    } else {
        digitalWrite(ledPin, LOW); // Tắt đèn LED
    }
}
```

Lưu ý khi dùng chức năng INPUT và OUTPUT trên ESP32

Khi sử dụng chức năng **INPUT** và **OUTPUT** trên ESP32 chúng ta cần chú ý lựa chọn chân cho phù hợp để không ảnh hưởng đến các chế độ hoạt động trên ESP32. Dưới đây là bảng tra cứu tham khảo:

GPIO	Input	Output	Notes
0	pulled up	OK	outputs PWM signal at boot, must be LOW to enter flashing mode
1	TX pin	OK	debug output at boot

2	OK	OK	connected to on-board LED, must be left floating or LOW to enter flashing mode
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot, strapping pin
6	x	x	connected to the integrated SPI flash
7	x	x	connected to the integrated SPI flash
8	x	x	connected to the integrated SPI flash
9	x	x	connected to the integrated SPI flash
10	x	x	connected to the integrated SPI flash
11	x	x	connected to the integrated SPI flash
12	OK	OK	boot fails if pulled high, strapping pin
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot, strapping pin
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	

21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

Khi dùng các chân trên ESP32 làm INPUT hoặc OUTPUT chúng ta cần lưu ý đến 5 chân đặc biệt để tránh ảnh hưởng đến quá trình khởi động cũng như nạp chương trình:

GPIO 0: Phải ở mức LOW để vào chế độ bootloader.

GPIO 2: Phải ở trạng thái floating hoặc LOW trong quá trình khởi động.

GPIO 5: Phải ở mức HIGH trong quá trình khởi động.

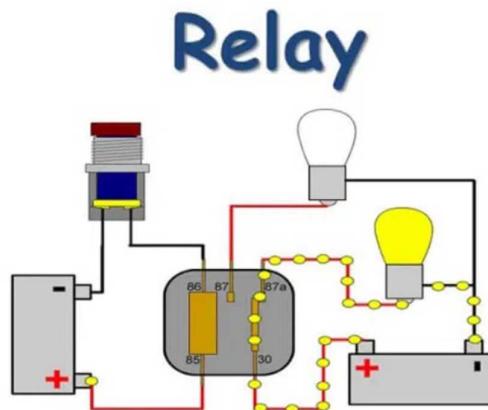
GPIO 12: Phải ở mức LOW trong quá trình khởi động.

GPIO 15: Phải ở mức HIGH trong quá trình khởi động.

Relay là gì? Các loại Relay trên thị trường

Relay là gì? Đây là một thiết bị điện nhỏ nhưng khá quen thuộc và quan trọng trong hệ thống điện, là một thành phần không thể thiếu để tạo nên tính liên tục cho mạch điện cả dân dụng hay công nghiệp.

Tổng quan về Relay



Relay (thường được gọi là rơ-le) là thiết bị công tắc điện tử (khóa K) đóng/ ngắt dòng điện, có khả năng bật/tắt linh hoạt để ngăn cho cường độ lớn dòng điện đi qua một thiết bị đang vận hành.

Có thể hiểu theo cách nghĩ đơn giản, relay hoạt động như một đòn bẩy điện để chuyển mạch, dùng cho dòng điện có cường độ nhỏ, nhưng nó khác với công tắc thông thường, là Relay được kích hoạt bằng điện thay vì hoạt động khi có lực trực tiếp tác động như công tắc.

Bản thân Relay là tên gọi vay mượn từ tiếng Pháp, nên chúng ta hay gọi đơn giản, ngắn gọn cho dễ nhớ là rơ-le. Bản chất của relay là 1 nam châm điện và hệ thống các tiếp điểm đóng cắt thiết kế theo kiểu modem để dễ dàng lắp đặt sử dụng hơn.

Cấu tạo của relay gồm 3 bộ phận chính:

- Khối tiếp thu – Nhận tín hiệu đầu vào để cung cấp năng lượng và truyền tín hiệu cho khối trung gian, đồng thời chuyển đổi chúng thành đại lượng cần thiết.
- Khối trung gian – Tiếp nhận tín hiệu từ khối tiếp thu và thay đổi thành đại lượng cần thiết cho relay
- Khối chấp hành – Thực hiện nhiệm vụ từ tín hiệu được truyền đi từ khối trung gian, và phát tín hiệu cho mạch điều khiển.

Nguyên lý hoạt động của Relay:

Khi dòng điện công suất nhỏ chạy qua mạch điện sẽ kích hoạt nam châm điện là dây thép quấn quanh lõi, tạo ra từ trường và truyền đi tín hiệu. Từ trường này sẽ thu hút 1 tiếp điểm chính và kích hoạt mạch điện thứ 2, cho phép thiết bị có thể kết nối và sử dụng dòng điện cường độ lớn hơn nhiều.

Khi dòng điện ngắt hoàn toàn, nam châm cũng ngừng hoạt động, không tạo ra từ trường nữa, lúc này các tiếp điểm sẽ bị lực kéo của lò xo ban đầu kéo về vị trí cũ, khiến mạch điện thứ 2 bị ngắt.

Như vậy relay đã thu hẹp khoảng cách, tiết diện của cường độ dòng điện, cho phép dòng điện nhỏ kích hoạt dòng điện cường độ lớn hơn. Công tắc relay cho phép thiết bị hay bộ máy lớn có thể sử dụng dòng điện lớn hơn với cường độ dòng ban đầu khá nhỏ.

Relay dùng làm gì?

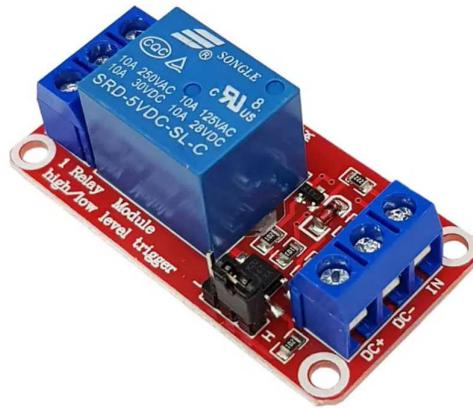
Relay có thể dùng trong các trường hợp như sau:

- Để đóng/ ngắt cho thiết bị điện cần thường xuyên đóng mở.
- Relay hoạt động như một thiết bị bảo vệ, phát hiện dòng quá tải, hay dưới dòng,... để đảm bảo thiết bị điện được an toàn, không bị tác động xấu khi cường độ dòng điện biến đổi đột ngột.
- Relay được sử dụng để chuyển đổi tín hiệu âm thanh, ánh sáng trong điều khiển đèn hay cảnh báo âm thanh.

Tổng quan về Module Relay

Sau khi tìm hiểu khái niệm **Relay là gì?** giờ chúng ta sẽ tìm hiểu về Module Relay (Mô-đun role):

Module Relay là gì?



Relay là công tắc điện tử, còn Module relay là tổ hợp một board mạch điện trong đó có sử dụng relay và mạch điều khiển relay. Module relay thường dùng trong thiết kế điều khiển trực tiếp của nhiều loại vi điều khiển như: Arduino, AVR, PIC, ARM, ... Sử dụng tín hiệu điện áp thấp (3,3 – 5V) để điều khiển relay.

Module relay thường được dùng trong mạch điều khiển hệ thống điện tự động, cho phép thiết bị có thể điều khiển ở tải ngõ ra điện áp cao với dòng điện lớn, trong khi ngõ vào là dòng tín hiệu điều khiển rất nhỏ.

Ưu điểm của Module relay là có thể áp dụng với mọi loại vi điều khiển, kích thước nhỏ, dễ lắp đặt, có giá thành rất rẻ, không cần tốn công làm mạch.

Các chân của module relay bao gồm:

- VCC: Chân cấp nguồn điện đầu vào cho mạch điều khiển và cuộn dây relay
- GND: Chân nguồn tham chiếu 0V.
- IN: ngõ vào nhận tín hiệu.
- NO: Tiếp điểm ngõ ra thường mở của rơ le
- COM: Chân tiếp điểm chung của rơ le
- NC: Tiếp điểm ngõ ra thường đóng của rơ le.

Các thông số của bộ Module Relay là gì?

Một module relay có cấu tạo chính gồm 2 linh kiện thu động cơ bản là relay và transistor.

Nên module relay có những thông số cơ bản như sau:

Hiệu điện thế kích hoạt tối ưu

Có rất nhiều loại module relay có mức hiệu điện thế tối ưu, tương ứng với cường độ dòng điện tối đa khác nhau.

Các mức hiệu điện thế tối đa và cường độ dòng điện tối đa tương ứng trong thiết bị điện khi đấu nối vào module rơ-le bao gồm:

- SRD-05VDC-SL-C: Hiệu điện thế kích hoạt tối ưu là 5V.
- 10A – 250VAC: Cường độ dòng điện tối đa qua các tiếp điểm của relay với hiệu điện thế $\leq 250V$ (AC) là 10A.
- 10A – 30VDC: Cường độ dòng điện tối đa qua các tiếp điểm của relay với hiệu điện thế $\leq 30V$ (DC) là 10A.
- 10A – 125VAC: Cường độ dòng điện tối đa qua các tiếp điểm của relay với hiệu điện thế $\leq 125V$ (AC) là 10A.
- 10A – 28VDC: Cường độ dòng điện tối đa qua các tiếp điểm của relay với hiệu điện thế $\leq 28V$ (DC) là 10A.

Cách sử dụng module relay

Relay bình thường có 6 chân, trong đó 3 chân dùng để kích, 3 chân còn lại dùng để đấu nối với thiết bị điện công suất cao.

3 chân dùng để kích trong sơ đồ điện bao gồm:

- Chân +: cấp hiệu điện thế kích tối ưu.
- Chân -: nối với cực âm thiết bị
- Chân S (data): chân tín hiệu, tùy vào loại module relay mà nó sẽ làm nhiệm vụ kích relay. Nếu bạn đang dùng module relay kích ở mức mức cao thì chân S phải được cấp điện thế dương, và ngược lại.

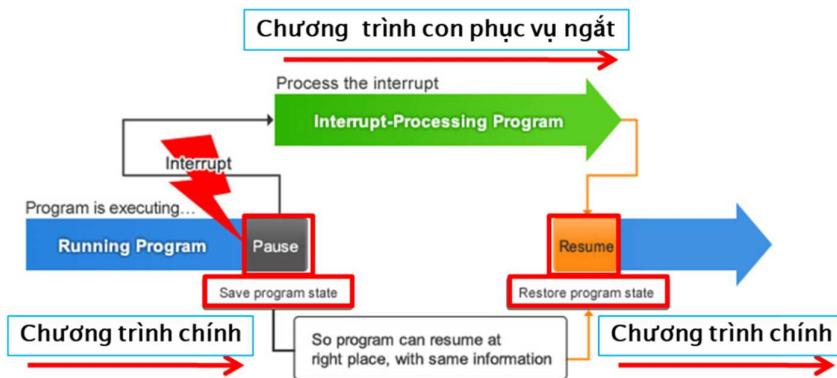
3 chân còn lại nối với thiết bị điện công suất cao:

- COM: nối với 1 chân bất kỳ của thiết bị điện.

- ON hoặc NO: nối với dây pha nếu dùng điện xoay chiều, còn nối với cực dương của nguồn nếu dòng điện một chiều.
- OFF hoặc NC: nối với dây trung tính (trung hòa) nếu dùng điện xoay chiều, còn là nối với cực âm nếu dùng điện một chiều.

Ngắt ngoài trên ESP32

Ngắt ngoài cho phép chương trình của bạn phản ứng nhanh chóng với các sự kiện xảy ra trên một chân GPIO. Khi sự kiện xảy ra, một hàm callback sẽ được gọi để thực hiện các tác vụ cần thiết.



Cách sử dụng ngắt ngoài trên ESP32:

pinMode(pin, mode): Thiết lập chế độ INPUT/INPUT_PULLUP

attachInterrupt(pin, ISR, mode): Thiết lập ngắt cho một chân GPIO.

pin: Chân GPIO cần thiết lập ngắt.

ISR: Hàm callback sẽ được gọi khi ngắt xảy ra.

mode: Chế độ ngắt (RISING, FALLING, CHANGE).

Ví dụ: Bật còi báo động khi phát hiện chuyển động dùng ngắt ngoài trên ESP32



Code chương trình:

```
// Khai báo các chân kết nối
const int pirPin = 13; // Chân kết nối cảm biến PIR
const int buzzerPin = 14; // Chân kết nối còi báo

void setup() {
    // Thiết lập chế độ cho các chân kết nối
    pinMode(pirPin, INPUT); // Thiết lập chân PIR là đầu vào
    pinMode(buzzerPin, OUTPUT); // Thiết lập chân còi báo là đầu ra

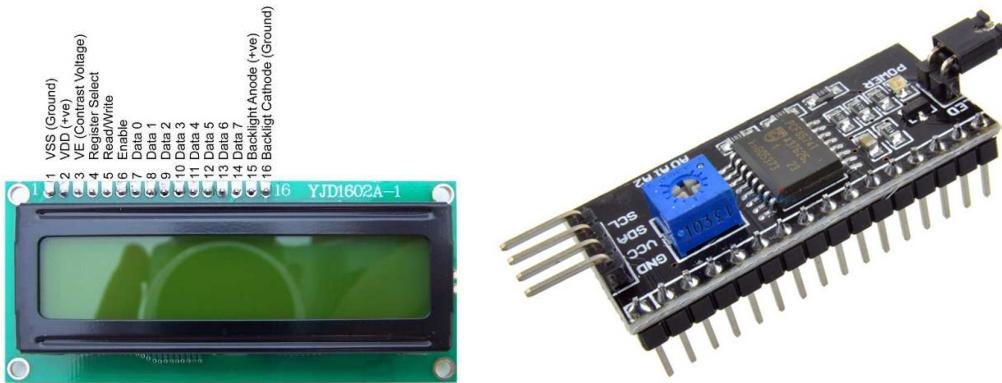
    // Gắn ngắt ngoài cho chân PIR, gọi hàm motionDetected khi có tín hiệu RISING
    attachInterrupt(pirPin, motionDetected, RISING);
}

void loop() {
    // Chương trình chính (để tránh vì xử lý ngắt ngoài)
}

// Hàm xử lý ngắt ngoài khi phát hiện chuyển động
ICACHE_RAM_ATTR void motionDetected() {
    digitalWrite(buzzerPin, HIGH); // Bật còi báo
    delay(1000); // Chờ 1 giây
    digitalWrite(buzzerPin, LOW); // Tắt còi báo
}
```

Giới thiệu về Màn hình LCD 16x2 và giao tiếp I2C:

- Màn hình LCD 16x2: Là một linh kiện phổ biến trong các dự án điện tử và lập trình. Nó có 16 chân, trong đó 8 chân dữ liệu (D0 - D7) và 3 chân điều khiển (RS, RW, EN). 5 chân còn lại dùng để cấp nguồn và đèn nền cho LCD 16x2.

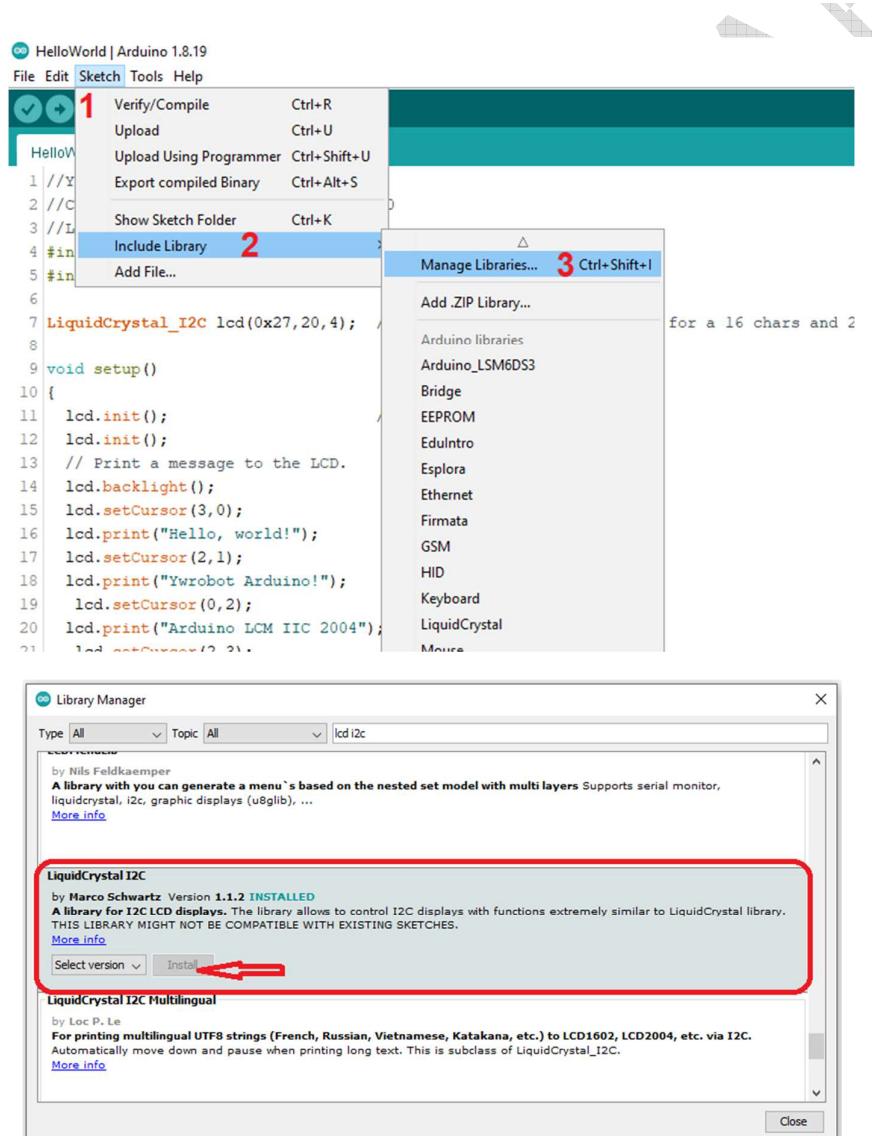


- LCD 16x2 có 16 chân trong đó 8 chân dữ liệu (D0 – D7) và 3 chân điều khiển (RS, RW, EN).
- 5 chân còn lại dùng để cấp nguồn và đèn nền cho LCD 16x2.
- Các chân điều khiển giúp ta dễ dàng cấu hình LCD ở chế độ lệnh hoặc chế độ dữ liệu.
- Chúng còn giúp ta cấu hình ở chế độ đọc hoặc ghi.
- LCD 16x2 có thể sử dụng ở chế độ 4 bit hoặc 8 bit tùy theo ứng dụng ta đang làm.

- Module I2C LCD: Để giảm số lượng chân kết nối, bạn có thể sử dụng module I2C LCD. Thay vì cần 6 chân để kết nối với LCD 16x2, module I2C chỉ cần 2 chân (SCL, SDA). Module này hỗ trợ các loại LCD sử dụng driver HD44780 (LCD 16x2, LCD 20x4, ...).

Cài đặt thư viện LiquidCrystal_I2C:

- Để sử dụng màn hình LCD giao tiếp I2C với Arduino, bạn cần cài đặt thư viện LiquidCrystal_I2C.
- Bạn có thể cài đặt thư viện này bằng cách mở Arduino IDE, chọn “Sketch” > “Include Library” > “Manage Libraries”. Tìm kiếm “LiquidCrystal_I2C” và cài đặt phiên bản mới nhất.



- ❖ Bài tập 2.1 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển bật tắt 3 đèn led xanh, đỏ, vàng dùng 3 nút nhấn.
- ❖ Bài tập 2.2 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển bật tắt đèn led thông qua module relay.
- ❖ Bài tập 2.3 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển bật tắt đèn led bằng nút nhấn hiển thị trạng thái trên LCD I2C 1602.
- ❖ Bài tập 2.4 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển hiển thị số lần nhấn nút trên LCD I2C 1602.
- ❖ Bài tập 2.5 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển bật tắt led và còi báo bằng nút nhấn dùng ngắt ngoài của Esp32.

DK SON - MILAB

BÀI 3 : LẬP TRÌNH GIAO TIẾP I2C, ADC VỚI CÁC CẢM BIẾN THÔNG DỤNG

3.1.Nhiệm vụ

Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào Esp32 và sử dụng mô hình thí nghiệm để kiểm chứng.

- Lập trình các chức năng ADC, giao tiếp I2C của Esp32.
- Điều khiển và sử dụng các linh kiện cơ bản như Servo, biến trờ, keypad, màn hình OLED...
- SV hoàn thành các bài tập 3.1 đến 3.5.

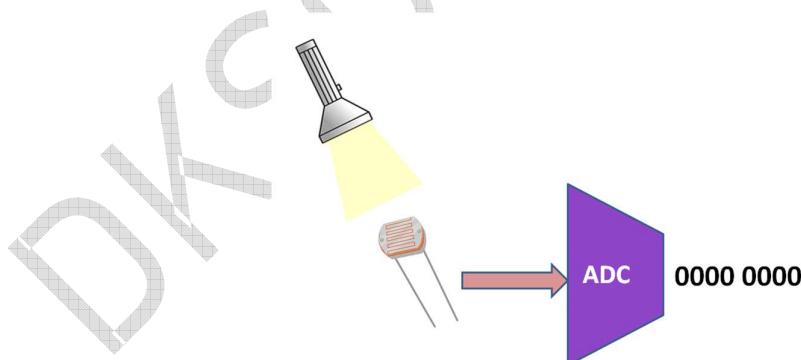
3.2.Yêu cầu

- Nắm vững các tập lệnh giao tiếp của Esp32.
- Biết cách viết các chương trình điều khiển các thiết bị thông dụng.
- Nắm được sơ đồ và nguyên lý hoạt động của các thiết bị như Servo, biến trờ, keypad, màn hình OLED...

3.3. Giới thiệu :

➤ Khái niệm chuyển đổi tương tự sang số ADC

Chuyển đổi tương tự sang số hay ADC (Analog-to-Digital Converter) là hệ thống mạch thực hiện chuyển đổi một tín hiệu analog (tín hiệu tương tự) liên tục, ví dụ như tín hiệu âm thanh thanh micro, hay tín hiệu ánh sáng trong máy ảnh kỹ thuật số, thành tín hiệu số. Một hệ thống ADC là một bộ phận phần cứng (như một bộ tính toán độc lập) làm nhiệm vụ chuyển đổi tín hiệu analog (dưới dạng điện áp hay dòng điện) thành các giá trị số (digital) đại diện cho cường độ điện áp hay tín hiệu đó.

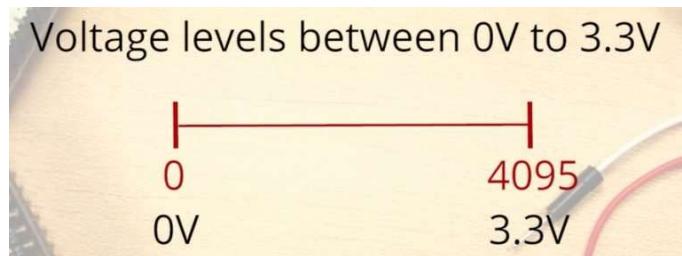


Có 2 tham số quan trọng của bộ ADC cần lưu ý:

- Tốc độ lấy mẫu (sampling) được tính theo số chu kỳ chuyển đổi
- Độ phân giải: Tính theo Bit bộ ADC có độ phân giải 10 Bit sẽ có $2^{10} = 1024$ giá trị

Đọc giá trị tương tự với ESP32 có nghĩa là bạn có thể đo các mức điện áp khác nhau giữa 0V và 3,3V. Điện áp đo được sau đó được gán cho giá trị từ 0 đến 4095, trong đó

0 tương ứng với 0V và 3,3V tương ứng với 4095. Bất kỳ điện áp nào từ 0V đến 3,3V sẽ nhận giá trị tương ứng ở giữa.



Analog Input trong ESP32

Đọc đầu vào tín hiệu tương tự cũng giống như đọc đầu vào tín hiệu số chúng ta chỉ cần một hàm đơn giản đó là [analogRead\(\)](#) với tham số truyền vào là số chân GPIO đó.

[analogRead\(GPIO\);](#)

example: `int value = analogRead(2);`

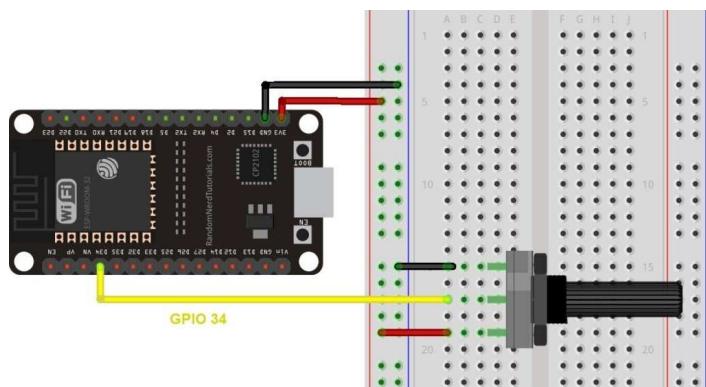
Ngoài ra để có thể cấu hình chi tiết cho Analog Input. Chúng ta có thể sử dụng thêm các hàm sau:

- `analogReadResolution(resolution)`: thiết lập độ phân giải. Giá trị từ 9 (0 – 511) đến 12 bit (0 – 4095). Mặc định là độ phân giải 12 bit.
- `analogSetWidth(width)`: thiết lập độ phân giải. Giá trị từ 9 (0 – 511) đến 12 bit (0 – 4095). Mặc định là độ phân giải 12 bit
- `analogSetCycles(cycles)`: thiết lập số chu kỳ cho một lần lấy mẫu. Mặc định là 8. Phạm vi: 1 đến 255.
- `analogSetSamples(samples)`: đặt số lượng mẫu trong một lần lấy mẫu. Mặc định là 1 mẫu. Nó có tác dụng làm tăng độ nhạy.
- `analogSetClockDiv(attenuation)`: đặt bộ chia cho clock ADC. Mặc định là 1. Phạm vi: 1 đến 255.
- `adcAttachPin(pin)`: Gắn một chân vào chế độ ADC (cũng xóa bất kỳ chế độ tương tự nào khác được bật trên chân đó). Trả về kết quả TRUE hoặc FALSE.
- `adcStart(pin)`: bắt đầu chuyển đổi ADC trên chân
- `adcBusy(pin)`: Kiểm tra xem chân có đang bộn chuyển đổi không (trả về TRUE hoặc FALSE).
- `resultadcEnd(pin)`: Nhận kết quả của chuyển đổi: trả về số nguyên 16 bit.

Đọc giá trị analog từ chiết áp với ESP32.

Chúng ta sẽ làm một ví dụ đơn giản đọc giá trị analog từ chiết áp (potentiometer).

Sơ đồ mạch



Code

```
// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)

const int potPin = 34;

// variable for storing the potentiometer value

int potValue = 0;

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

    delay(1000);

}

void loop() {
    // Reading potentiometer value

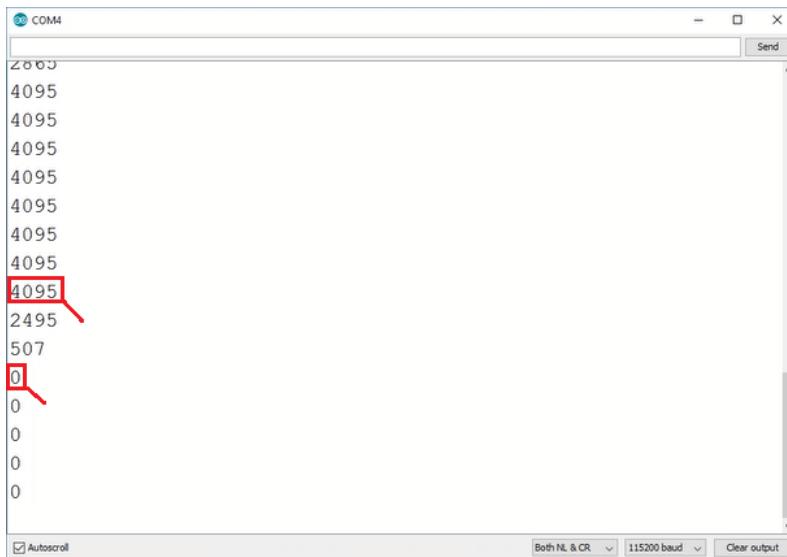
    potValue = analogRead(potPin);

    Serial.println(potValue);

    delay(500);
}
```

Kiểm tra Example

Giá trị lớn nhất là 4095 và giá trị nhỏ nhất là 0.



Giới thiệu cảm biến quang trở

Cảm biến quang trở giúp chúng ta nhận biết được ánh sáng, cường độ ánh sáng được ứng dụng rộng rãi trong đời sống như : Hỗ trợ cho tấm Pin năng lượng mặt trời, tự động bật tắt đèn khi trời tối, điều chỉnh cường độ sáng để phù hợp với môi trường.

Tùy vào cách kết nối mà đọc giá trị cảm biến về khác nhau, nếu chỉ cần nhận biết sáng tối thì kết nối theo kiểu so sánh đưa ngõ ra mức “0” hoặc “1” còn nếu muốn đọc hiển thị thì kết nối theo cầu phân áp và dùng vi điều khiển đọc giá trị ADC.

Cấu tạo và cách hoạt động

Thành phần chính để tạo nên quang trở đó chính là Cadmium Sulphide (CdS) được sử dụng là chất quang dẫn, thường không chứa hoặc có rất ít các hạt electron khi không được ánh sáng chiếu vào.

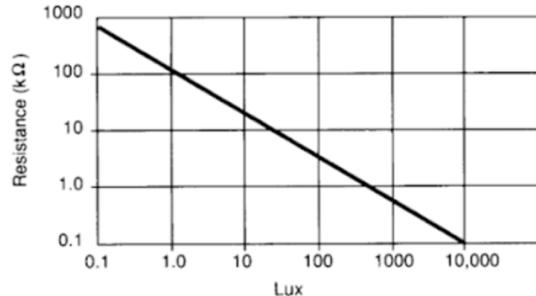
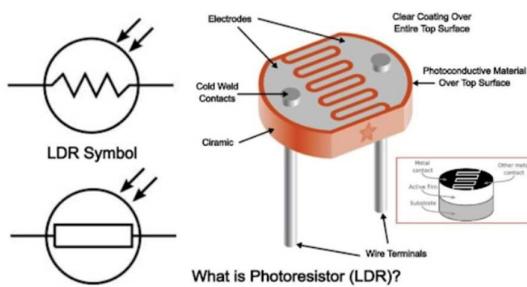
Một quang trở khi được đóng gói sẽ có thêm 2 màng kim loại và được đặt trong một hộp nhựa có thể tiếp xúc được với ánh sáng, và cấu tạo theo hình tròn dẹt để có thể tiếp xúc tối đa với ánh sáng chiếu vào (như hình ảnh bên dưới).

Nguyên lý hoạt động

Khi không có ánh sáng chiếu vào, quang trở thường không chứa hoặc có rất ít các hạt electron, vì vậy lúc này nó sẽ dẫn điện kém, tương đương với việc điện trở bản thân nó lúc này khá cao, lên đến vài $M\Omega$ (Mega-Ohm).

Ngược lại, nếu như có ánh sáng chiếu vào mạnh thì số lượng electron tăng lên, quang trở trở nên dẫn điện tốt, và điện trở có thể giảm xuống đến vài trăm Ω (Theo mình test thì khoảng 200Ω khi đặt sát bóng đèn học).

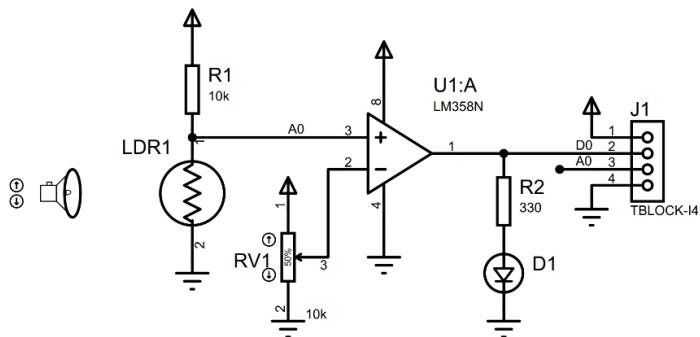
Với sự thay đổi của cường độ ánh sáng chiếu vào, cường độ ánh sáng chiếu vào tăng thì điện trở sẽ giảm, nhưng biểu đồ thay đổi sẽ có dạng tuyến tính hoặc không tuyến tính (tùy loại), ví dụ như bên dưới (Các bạn có thể tìm kiếm và xem trong LDR datasheet của từng loại riêng).



MODULE QUANG TRỞ LM393 RA 4 CHÂN

THÔNG SỐ KỸ THUẬT :

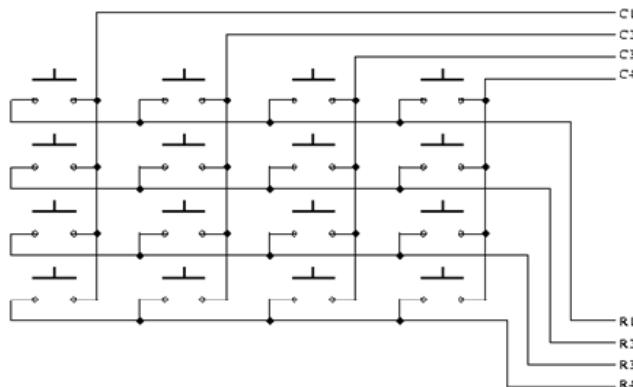
- Điện áp hoạt động 3.3 – 5 V.
- Kết nối 4 chân với 2 chân cấp nguồn (VCC và GND) và 2 chân tín hiệu ngõ ra (AO và DO).
- Hỗ trợ cả 2 dạng tín hiệu ra Analog và TTL. Ngõ ra Analog 0 – 5V tỷ lệ thuận với cường độ ánh sáng, ngõ TTL tích cực mức thấp.
- Độ nhạy cao với ánh sáng được tùy chỉnh bằng biến trở .



KeyPad 4x4

KeyPad là một thiết bị nhập chứa các nút bấm cho phép người dùng nhập các chữ số, chữ cái hoặc ký tự điều khiển. KeyPad không chứa tất cả bảng mã ASCII như keyboard vì thế nó thường được sử dụng trong các ứng dụng chuyên dụng và tương đối đơn giản, ở đó, số lượng nút nhấn thay đổi phụ thuộc vào ứng dụng.

KeyPad 4x4 là bàn phím gồm 16 nút nhấn, được xếp thành 4 hàng, mỗi hàng gồm 4 phím bấm như hình dưới đây:



Cấu tạo và nguyên lý hoạt động

Cấu tạo

KeyPad 4x4 gồm: 8 đầu vào/ra và 16 phím nhấn kết nối theo sơ đồ dưới đây:

Các phím bấm được chia thành 4 hàng và 4 cột, 1 đầu của nút bấm được nối với đầu vào cột(C), đầu kia được nối với đầu vào hàng(R).

Nguyên lý hoạt động

Để làm việc với KeyPad 4x4, người lập trình thường sử dụng giải thuật “quét phím”. Giải thuật này yêu cầu VĐK liên tục đưa các tín hiệu đầu ra ở hàng (hoặc cột) và thu lại đầu vào ở cột (hoặc hàng), nếu phím được bấm, đầu phát tín hiệu sẽ được kết nối với đầu thu, từ đó xác định được phím đã bấm.

Việc lựa chọn đầu ra/vào hình thành 2 phương pháp quét phím: theo chiều dọc và theo chiều ngang. Trong báo cáo này, tín hiệu xuất ra ở các hàng và thu lại ở các cột.

Giả sử một nút ‘2’ được nhấn, khi đó đường C và 2 được nối với nhau. Nếu đường C được nối với GND, khi đó, điện áp ở chân số 2 sẽ mang điện áp 0V. Tương tự như thế với các phím cùng hàng C.

Thuật toán

Bước 1: set các chân ROW1, ROW2, ROW3, ROW4 như các chân Output và giữ chúng ở mức cao, các chân COL1, COL2, COL3, COL4 như các chân input có điện trở kéo lên.

Bước 2: đưa tín hiệu đầu ra ở các chân $ROW1 = 1$, $ROW2 = 1$, $ROW3 = 1$ và $ROW4 = 1$. Kiểm tra tín hiệu ở các chân COL1, COL2, COL3, COL4 luôn bằng 1 dù có phím nào được nhấn hay không.

Bước 3: đưa tín hiệu đầu ra ở các chân $ROW1 = 0$, $ROW2 = 1$, $ROW3 = 1$ và $ROW4 = 1$.

Kiểm tra COL1, 2, 3 và 4, nếu phím thuộc hàng 1 được nhấn sẽ giá trị COL nhận được bằng 0, ví dụ: ON/C được nhấn, $COL1 = 0$, $COL2, 3, 4 = 1$. Nếu phím thuộc các hàng khác (2, 3, 4) được nhấn, các chân COL1, COL2, COL3, COL4 luôn bằng 1.

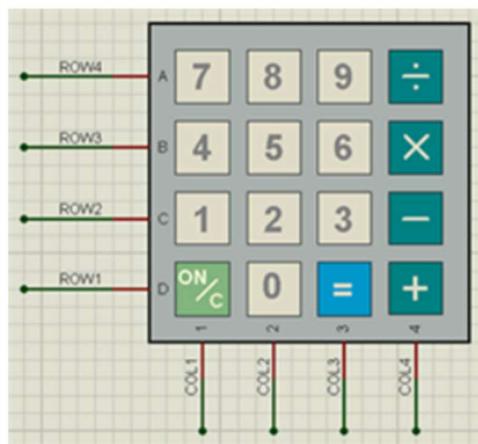
Bằng thao tác trên, sẽ xác định phím đã được nhấn nếu nó ở hàng 1.

Bước 4: tiếp tục đưa tín hiệu đầu ra ở các chân ROW1 = 1, ROW2 = 0, ROW3 = 1, ROW4 = 1 để xác định phím bấm được nhấn nếu nó ở hàng 2.

Bước 5: thực hiện quá trình dịch chân đầu ra mang điện áp mức 0 một cách liên tục và xác định phím được bấm.

Giao tiếp với KeyPad 4x4

Để thực hiện quá trình giao tiếp với KeyPad 4x4, ta ghép nối sơ đồ sau:



Vì cấu tạo KeyPad bao gồm các nút bấm nên không tránh khỏi nhiễu do việc bấm nút, người dùng có thể sử dụng chống nhiễu bằng phần mềm và kết hợp thêm một mạch chống nhiễu đơn giản bằng cách nối đất tụ gồm, giá trị tùy vào điện trở treo (thường là 104) từ các chân COL1, 2, 3 và 4.

Động cơ Servo

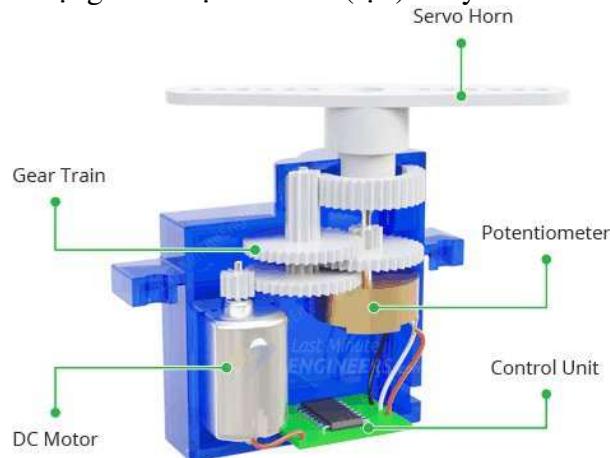
Servo là một dạng động cơ điện đặc biệt. Không giống như động cơ thông thường cứ cắm điện vào là quay liên tục, servo chỉ quay khi được điều khiển (bằng xung PPM) với góc quay nằm trong khoảng bất kì từ 0° - 180° . Mỗi loại servo có kích thước, khối lượng và cấu tạo khác nhau. Có loại thì nặng chỉ 9g (chủ yếu dùng trên máy bay mô hình), có loại thì sở hữu một moment lực lớn (vài chục Newton/m).

Động cơ servo được thiết kế những hệ thống hồi tiếp vòng kín. Tín hiệu ra của động cơ được nối với một mạch điều khiển. Khi động cơ quay, vận tốc và vị trí sẽ được hồi tiếp về mạch điều khiển này. Nếu có bất kỳ lý do nào ngăn cản chuyển động quay của động cơ, cơ cấu hồi tiếp sẽ nhận thấy tín hiệu ra chưa đạt được vị trí mong muốn. Mạch điều khiển tiếp tục chỉnh sai lệch cho động cơ đạt được điểm chính xác. Các động cơ servo điều khiển bằng liên lạc vô tuyến được gọi là động cơ servo RC (radio-controlled). Trong thực tế, bản thân động cơ servo không phải được điều khiển bằng vô tuyến, nó chỉ nối với máy thu vô tuyến trên máy bay hay xe hơi. Động cơ servo nhận tín hiệu từ máy thu này.

Động cơ Servo SG90 Arduino là một loại động cơ có kích thước khá nhỏ, giá thành thấp và được sử dụng rộng rãi trong lập trình Arduino. Nó được sử dụng trong một số dự án như: **Robot dò line tránh vật cản, cánh tay Robot 4 bậc...**

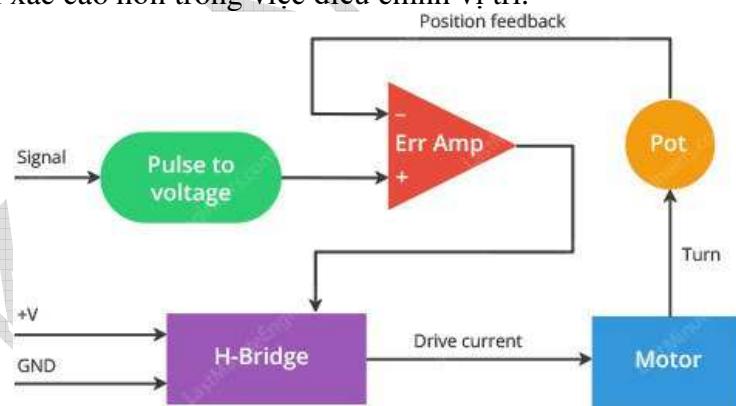
Cấu tạo bên trong động cơ Servo SG90 Arduino bao gồm:

Động cơ DC: Động cơ DC chịu trách nhiệm tạo ra chuyển động quay của trục đầu ra. Điện áp được cấp vào động cơ để tạo ra xoắn (lực) xoay.



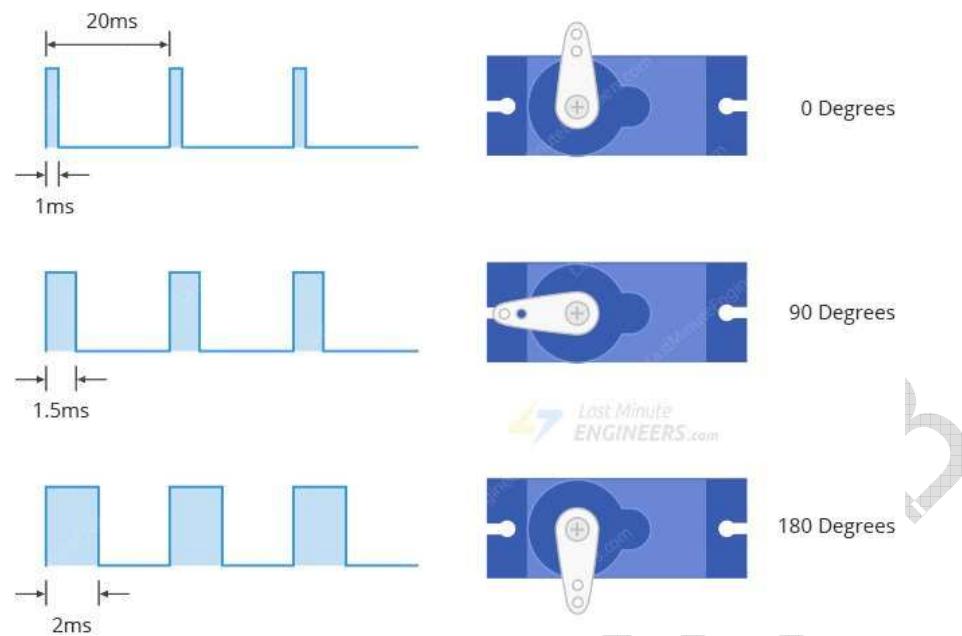
Hệ thống điều khiển: Hệ thống điều khiển bao gồm bộ điều khiển và mạch phản hồi. Bộ điều khiển nhận tín hiệu điều khiển từ nguồn điều khiển (như Arduino) và điều chỉnh tín hiệu đầu vào để kiểm soát chuyển động của động cơ. Mạch phản hồi (potentiometer) được gắn kết với trục đầu ra để cung cấp thông tin về vị trí hiện tại của động cơ.

Hệ thống giảm tốc: Động cơ Servo SG90 thường đi kèm với hệ thống giảm tốc để tăng lực xoắn và giảm tốc độ quay của động cơ. Hệ thống giảm tốc giúp đạt được độ chính xác cao hơn trong việc điều chỉnh vị trí.



Quá trình hoạt động của **động cơ Servo SG90** như sau:

Nhận tín hiệu điều khiển: Tín hiệu điều khiển PWM (Pulse Width Modulation) được cung cấp từ nguồn điều khiển, như Arduino. Tín hiệu này có thể được điều chỉnh trong khoảng từ 1 ms đến 2 ms và có chu kỳ là 20 ms.



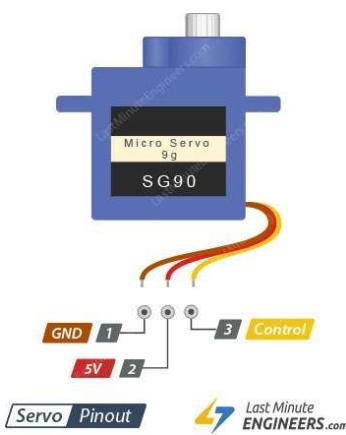
Điều chỉnh vị trí: Bộ điều khiển nhận tín hiệu điều khiển và so sánh nó với vị trí hiện tại của động cơ được xác định bởi mạch phản hồi. Dựa trên sự khác biệt giữa vị trí yêu cầu và vị trí hiện tại, bộ điều khiển điều chỉnh tín hiệu điều khiển để tạo ra chuyển động quay của động cơ.

Phản hồi vị trí: Mạch phản hồi (potentiometer) gửi thông tin về vị trí hiện tại của động cơ cho bộ điều khiển. Bộ điều khiển sử dụng thông tin này để đảm bảo rằng động cơ đạt được vị trí yêu cầu và duy trì nó trong quá trình hoạt động.

Điều chỉnh liên tục: Quá trình điều chỉnh vị trí và phản hồi được thực hiện liên tục để duy trì độ chính xác và ổn định của vị trí đầu ra của động cơ.



Sơ đồ chân động cơ Servo SG90 Arduino



- **5V:** Chân dương dùng để cấp nguồn cho động cơ servo.
- **GND:** Chân âm nối đất.
- **Control:** Chân tín hiệu điều khiển để điều chỉnh vị trí của động cơ. Tín hiệu điều khiển được gửi dưới dạng xung PWM.

Màn hình OLED là gì?

OLED (viết tắt bởi Organic Light Emitting Diode: Diode phát sáng hữu cơ) đang trở thành đối thủ cạnh tranh cũng như ứng cử viên sáng giá thay thế màn hình LCD.

Màn hình OLED gồm những lớp như tấm nền, Anode, lớp hữu cơ, cathode. Và phát ra ánh sáng theo cách tương tự như đèn LED. Quá trình trên được gọi là phát lân quang điện tử.



Những ưu điểm có thể kể đến trên màn hình OLED là những lớp hữu cơ nhựa mỏng, nhẹ mềm dẻo hơn những lớp tinh thể trên LED hay LCD nhờ vậy mà có thể ứng dụng OLED để chế tạo màn hình gấp cuộn được. Độ sáng của OLED cũng tốt hơn LED và không cần đèn nền như trên LCD nên sử dụng pin ít hơn. Góc nhìn cũng cải thiện hơn những công nghệ tiền nhiệm, khoảng 170 độ.

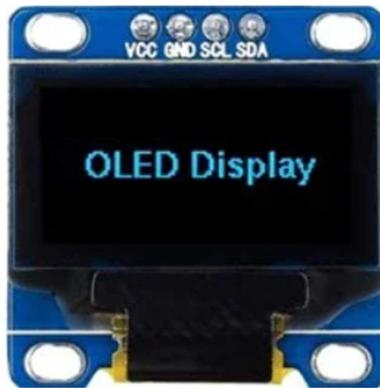


Nhược điểm có thể kể tới là tuổi thọ màn này khá thấp, giá thành sản xuất cao và rất dễ hỏng khi gặp nước. Nên dễ hiểu màn hình này chưa được ứng dụng nhiều.

Tổng quan về màn hình OLED 0.96

Màn hình OLED 0.96 là một loại màn hình hiển thị sử dụng công nghệ Organic Light Emitting Diode (OLED). Kích thước màn hình thường là 0.96 inch, đây là kích thước phổ biến được sử dụng trong nhiều ứng dụng nhỏ gọn.

OLED 0.96 có độ phân giải thường là 128×64 pixel, tức là có 128 điểm ảnh trên chiều ngang và 64 điểm ảnh trên chiều dọc. Với độ phân giải này, màn hình có thể hiển thị các đồ họa, ký tự và biểu đồ đơn giản.



Một trong những đặc điểm nổi bật của màn hình OLED là khả năng tự phát sáng của các pixel. Mỗi pixel trên màn hình OLED có thể tự phát ra ánh sáng mà không cần đèn nền phụ trợ như các loại màn hình khác. Điều này cho phép màn hình OLED có độ tương phản cao, màu sắc tươi sáng và góc nhìn rộng.

Màn hình OLED 0.96 thường được kết nối với Arduino thông qua giao tiếp I2C hoặc SPI. Có sẵn các thư viện hỗ trợ để điều khiển màn hình OLED trên Arduino IDE.

Nguồn (Power)

Màn hình OLED không yêu cầu đèn nền như màn hình LCD ký tự, vì nó tự phát sáng. Điều này giúp màn hình OLED có độ tương phản cao, góc nhìn rộng và khả năng hiển thị màu đen sâu. Bởi vì không có đèn nền, năng lượng tiêu thụ của màn hình giảm đáng kể. Trung bình, màn hình OLED sử dụng khoảng 20mA, nhưng điều này có thể thay đổi tùy thuộc vào độ sáng của màn hình.

Bộ điều khiển SSD1306 hoạt động ở điện áp từ 1,65V đến 3,3V, trong khi tấm nền OLED yêu cầu điện áp nguồn từ 7V đến 15V. Các yêu cầu nguồn cấp khác nhau này được đáp ứng bởi mạch bơm sạc (**Charge pump**) bên trong màn hình. Điều này cho phép kết nối màn hình OLED với Arduino hoặc bất kỳ bộ vi điều khiển logic 5V nào khác mà không cần sử dụng bộ chuyển đổi mức logic.

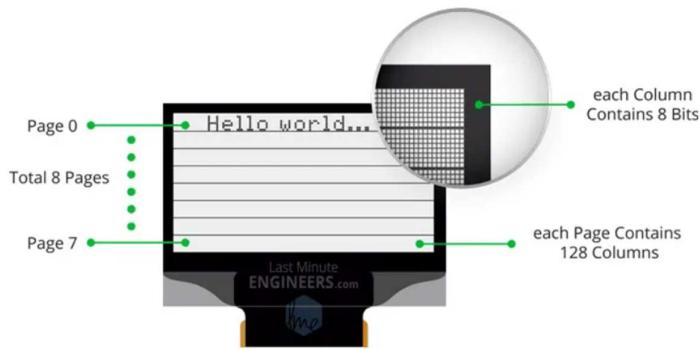
Bản đồ bộ nhớ OLED (OLED Memory Map)

Để điều khiển màn hình OLED, việc hiểu bản đồ bộ nhớ là rất quan trọng.

Dù kích thước của màn hình OLED là bao nhiêu, bộ điều khiển SSD1306 bao gồm một RAM Dữ liệu Hiển thị Đồ họa (GDDRAM) có dung lượng 1KB để lưu trữ các mảng bit sẽ được hiển thị trên màn hình. Vùng nhớ 1KB này được chia thành 8 trang (từ 0 đến 7). Mỗi trang có 128 cột/đoạn (khối 0 đến 127). Và, mỗi cột có thể lưu trữ 8 bit dữ liệu (từ 0 đến 7). Tổng cộng:

$$8 \text{ trang} \times 128 \text{ cột} \times 8 \text{ bit} = 8192 \text{ bit} = 1\text{KB bộ nhớ}$$

Toàn bộ bộ nhớ 1KB này, bao gồm các trang, cột và dữ liệu, được đánh dấu như sau:



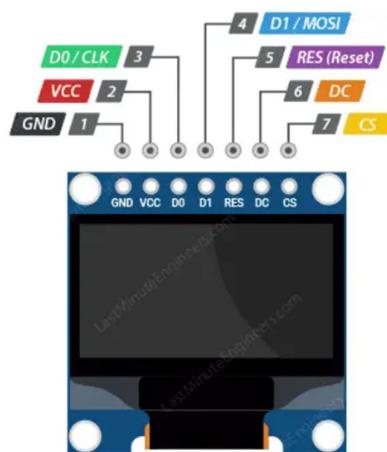
Mỗi bit đại diện cho một pixel OLED duy nhất trên màn hình và có thể được chương trình BẬT hoặc TẮT.

Màn hình OLED 0.96 I2C



- **VCC:** là nguồn cấp cho màn hình có thể là 3.3V hoặc 5V
- **GND:** chân nối đất
- **SCL:** chân xung clock
- **SDA:** chân dữ liệu nối tiếp

Màn hình OLED 0.96 SPI



- **VCC:** là nguồn cấp cho màn hình có thể là 3.3V hoặc 5V

- **GND:** chân nối đất
- **D0/CLK:** chân xung clock
- **D1/MOSI:** được sử dụng để gửi dữ liệu từ Arduino đến màn hình OLED
- **RES (Reset):** được sử dụng để thiết lập lại màn hình OLED
- **CS:** được sử dụng để chọn màn hình OLED trong giao tiếp SPI
- **DC:** được sử dụng để xác định liệu dữ liệu đang được gửi qua bus SPI là một lệnh hay dữ liệu hiển thị.

Để lập trình hiển thị được trên màn hình Oled chúng ta cần xác định loại giao tiếp của màn hình và cài các thư viện cần thiết.

Đối với màn hình OLED giao tiếp I2C trên Wokwi sử dụng thư viện của Adafruit.

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

Chip DS1307

DS1307 là chip đồng hồ thời gian thực (RTC : Real-time clock), khái niệm thời gian thực ở đây được dùng với ý nghĩa thời gian tuyệt đối mà con người đang sử dụng, tinh bằng giây, phút, giờ... DS1307 là một sản phẩm của Dallas Semiconductor (một công ty thuộc Maxim Integrated Products). Chip này có 7 thanh ghi 8-bit chứa thời gian là: giây, phút, giờ, thứ (trong tuần), ngày, tháng, năm. Ngoài ra DS1307 còn có 1 thanh ghi điều khiển ngõ ra phụ và 56 thanh ghi trống có thể dùng như RAM. DS1307 được đọc và ghi thông qua giao diện nối tiếp I2C (TWI của AVR) nên cấu tạo bên ngoài rất đơn giản. DS1307 xuất hiện ở 2 gói SOIC và DIP có 8 chân như trong hình 1.

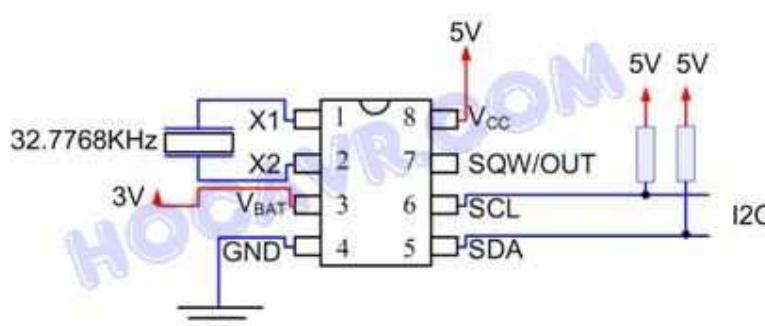


Hình 1. Hai gói cấu tạo chip DS1307.

Các chân của DS1307 được mô tả như sau:

- **X1** và **X2:** là 2 ngõ kết nối với 1 thạch anh 32.768KHz làm nguồn tạo dao động cho chip.
- **V_{BAT}:** cực dương của một nguồn pin 3V nuôi chip.

- **GND:** chân mass chung cho cả pin 3V và Vcc.
- **Vcc:** nguồn cho giao diện I2C, thường là 5V và dùng chung với vi điều khiển. Chú ý là nếu Vcc không được cấp nguồn nhưng VBAT được cấp thì DS1307 vẫn đang hoạt động (nhưng không ghi và đọc được).
- **SQW/OUT:** một ngõ phụ tạo xung vuông (Square Wave / Output Driver), tần số của xung được tạo có thể được lập trình. Như vậy chân này hầu như không liên quan đến chức năng của DS1307 là đồng hồ thời gian thực, chúng ta sẽ bỏ trống chân này khi nối mạch.
- **SCL và SDA** là 2 đường giao xung nhịp và dữ liệu của giao diện I2C mà chúng ta đã tìm hiểu trong bài TWI của AVR. Có thể kết nối DS1307 bằng một mạch điện đơn giản như trong hình 2.



Hình 2. Mạch ứng dụng đơn giản của DS1307.

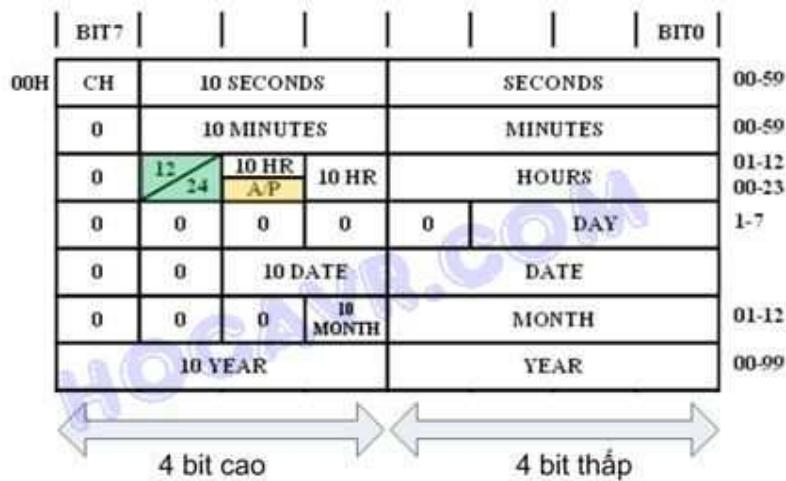
Cấu tạo bên trong DS1307 bao gồm một số thành phần như mạch nguồn, mạch dao động, mạch điều khiển logic, mạch giao diện I2C, con trỏ địa chỉ và các thanh ghi (hay RAM). Do đa số các thành phần bên trong DS1307 là thành phần “cứng” nên chúng ta không có quá nhiều việc khi sử dụng DS1307. Sử dụng DS1307 chủ yếu là ghi và đọc các thanh ghi của chip này. Vì thế cần hiểu rõ 2 vấn đề cơ bản đó là cấu trúc các thanh ghi và cách truy xuất các thanh ghi này thông qua giao diện I2C. Phần này chúng ta tìm hiểu cấu trúc các thanh ghi trước và cách truy xuất chúng sẽ tìm hiểu trong phần 2, điều khiển DS1307 bằng AVR.

Như tôi đã trình bày, bộ nhớ DS1307 có tất cả 64 thanh ghi 8-bit được đánh địa chỉ từ 0 đến 63 (từ 0x00 đến 0x3F theo hệ hexadecimal). Tuy nhiên, thực chất chỉ có 8 thanh ghi đầu là dùng cho chức năng “đồng hồ” (tôi sẽ gọi là RTC) còn lại 56 thanh ghi bỏ trống có thể được dùng chứa biến tạm như RAM nếu muốn. Bảy thanh ghi đầu tiên chứa thông tin về thời gian của đồng hồ bao gồm: giây (SECONDS), phút (MINUETS), giờ (HOURS), thứ (DAY), ngày (DATE), tháng (MONTH) và năm (YEAR). Việc ghi giá trị vào 7 thanh ghi này tương đương với việc “cài đặt” thời gian khởi động cho RTC. Việc đọc giá từ 7 thanh ghi là đọc thời gian thực mà chip tạo ra. Ví dụ, lúc khởi động chương trình, chúng ta ghi vào thanh ghi “giây” giá trị 42, sau đó 12s chúng ta đọc thanh ghi này, chúng ta thu được giá trị 54. Thanh ghi thứ 8 (CONTROL) là thanh ghi điều khiển xung ngõ ra SQW/OUT (chân 6). Tuy nhiên, do chúng ta không dùng chân SQW/OUT nên có thể bỏ qua thanh ghi thứ 8. Tổ chức bộ nhớ của DS1307 được trình bày trong hình 3.



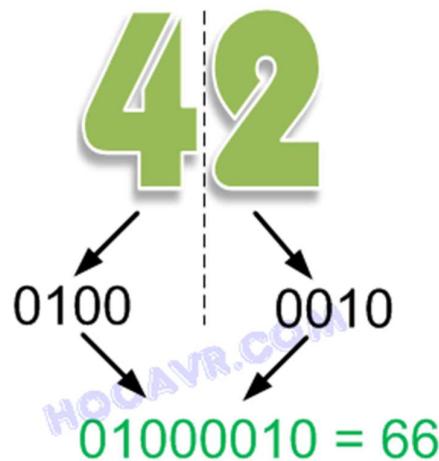
Hình 3. Tổ chức bộ nhớ của DS1307.

Vì 7 thanh ghi đầu tiên là quan trọng nhất trong hoạt động của DS1307, chúng ta sẽ khảo sát các thanh ghi này một cách chi tiết. Trước hết hãy quan sát tổ chức theo từng bit của các thanh ghi này như trong hình 4.



Hình 4. Tổ chức các thanh ghi thời gian.

Điều đầu tiên cần chú ý là giá trị thời gian lưu trong các thanh ghi theo dạng BCD. BCD là viết tắt của cụm từ Binary-Coded Decimal, tạm dịch là các số thập phân theo mã nhị phân. Ví dụ bạn muốn cài đặt cho thanh ghi MINUTES giá trị 42. Nếu quy đổi 42 sang mã thập lục phân thì chúng ta thu được $42=0x2A$. Theo cách hiểu thông thường chúng ta chỉ cần gán MINUTES=42 hoặc MINUTES=0x2A, tuy nhiên vì các thanh ghi này chứa giá trị BCD nên mọi chuyện sẽ khác, tôi sẽ diễn giải bằng hình 5.



Hình 5. Số BCD.

Với số 42, trước hết nó được tách thành 2 chữ số (digit) 4 và 2. Mỗi chữ số sau đó được đổi sang mã nhị phân 4-bit. Chữ số 4 được đổi sang mã nhị phân 4-bit là 0100 trong khi 2 được đổi thành 0010. Ghép mã nhị phân của 2 chữ số lại chúng ta thu được một số 8 bit, đó là số BCD. Với trường hợp này, số BCD thu được là 01000010 (nhị phân) = 66. Như vậy, để đặt số phút 42 cho DS1307 chúng ta cần ghi vào thanh ghi MINUTES giá trị 66 (mã BCD của 42). Tất cả các phần mềm lập trình hay thanh ghi của chip điều khiển đều sử dụng mã nhị phân thông thường, không phải mã BCD, do đó chúng ta cần viết các chương trình con để quy đổi từ số thập nhị phân (hoặc thập phân thường) sang BCD, phần này sẽ được trình bày trong lúc lập trình giao tiếp với DS1307. Thoạt nhìn, mọi người đều cho rằng số BCD chỉ làm vẩn đền thêm rắc rối, tuy nhiên số BCD rất có ưu điểm trong việc hiển thị nhất là khi hiển thị từng chữ số như hiển thị bằng LED 7 đoạn chẵng hạn. Quay lại ví dụ 42 phút, giả sử chúng ta dùng 2 LED 7-doạn để hiện thị 2 chữ số của số phút. Khi đọc thanh ghi MINUTES chúng ta thu được giá trị 66 (mã BCD của 42), do 66=01000010 (nhị phân), để hiển thị chúng ta chỉ cần dùng phương pháp tách bit thông thường để tách số 01000010 thành 2 nhóm 0100 và 0010 (tách bằng toán tử shift ">>" của C hoặc instruction LSL, LSR trong asm) và xuất trực tiếp 2 nhóm này ra LED vì 0100 = 4 và 0010 = 2, rất nhanh chóng. Thậm chí, nếu chúng ta nối 2 LED 7-doạn trong cùng 1 PORT, việc tách ra từng digit là không cần thiết, để hiển thị cả số, chỉ cần xuất trực tiếp ra PORT. Như vậy, với số BCD, việc tách và hiển thị digit được thực hiện rất dễ dàng, không cần thực hiện phép chia (rất tốn thời gian thực thi) cho cơ số 10, 100, 1000...như trong trường hợp số thập phân.

Thanh ghi giây (SECONDS): thanh ghi này là thanh ghi đầu tiên trong bộ nhớ của DS1307, địa chỉ của nó là 0x00. Bốn bit thấp của thanh ghi này chứa mã BCD 4-bit của chữ số hàng đơn vị của giá trị giây. Do giá trị cao nhất của chữ số hàng chục là 5 (không có giây 60 !) nên chỉ cần 3 bit (các bit SECONDS6:4) là có thể mã hóa được (số 5 = 101, 3 bit). Bit cao nhất, bit 7, trong thanh ghi này là 1 điều khiển có tên CH (Clock halt – treo đồng hồ), nếu bit này được set bằng 1 bộ dao động trong chip bị vô hiệu hóa, đồng hồ không hoạt động. Vì vậy, nhất thiết phải reset bit này xuống 0 ngay từ đầu.

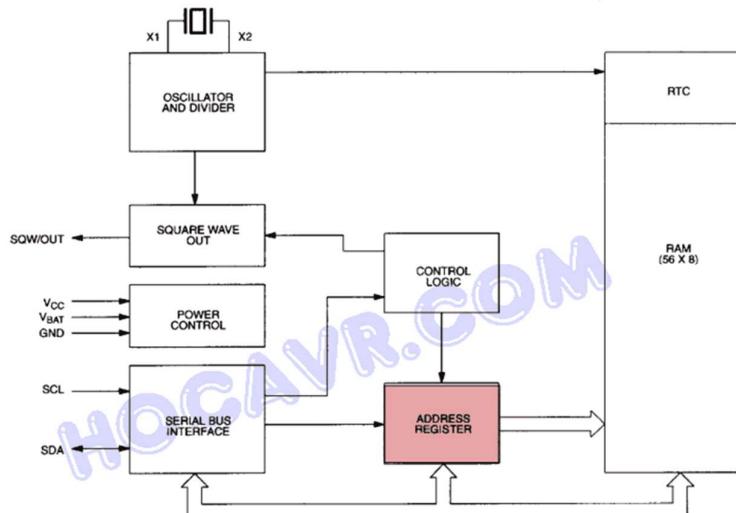
Thanh ghi phút (MINUTES): có địa chỉ 0x01, chứa giá trị phút của đồng hồ. Tương tự thanh ghi SECONDS, chỉ có 7 bit của thanh ghi này được dùng lưu mã BCD của phút, bit 7 luôn luôn bằng 0.

Thanh ghi giờ (HOURS): có thể nói đây là thanh ghi phức tạp nhất trong DS1307. Thanh ghi này có địa chỉ 0x02. Trước hết 4-bits thấp của thanh ghi này được dùng cho chữ số hàng đơn vị của giờ. Do DS1307 hỗ trợ 2 loại hệ thống hiển thị giờ (gọi là mode) là 12h (1h đến 12h) và 24h (1h đến 24h) giờ, bit6 (màu green trong hình 4) xác lập hệ thống giờ. Nếu bit6=0 thì hệ thống 24h được chọn, khi đó 2 bit cao 5 và 4 dùng mã hóa chữ số hàng chục của giá trị giờ. Do giá trị lớn nhất của chữ số hàng chục trong trường hợp này là 2 (=10, nhị phân) nên 2 bit 5 và 4 là đủ để mã hóa. Nếu bit6=1 thì hệ thống 12h được chọn, với trường hợp này chỉ có bit 4 dùng mã hóa chữ số hàng chục của giờ, bit 5 (màu orange trong hình 4) chỉ buổi trong ngày, AM hoặc PM. Bit5 =0 là AM và bit5=1 là PM. Bit 7 luôn bằng 0. (thiết kế này hơi dở, nếu đổi hẳn 2 bit mode và A-P sang 2 bit 7 và 6 thì sẽ đơn giản hơn).

Thanh ghi thứ (DAY – ngày trong tuần): nằm ở địa chỉ 0x03. Thanh ghi DAY chỉ mang giá trị từ 1 đến 7 tương ứng từ Chủ nhật đến thứ 7 trong 1 tuần. Vì thế, chỉ có 3 bit thấp trong thanh ghi này có nghĩa.

Các thanh ghi còn lại có cấu trúc tương tự, **DATE** chứa ngày trong tháng (1 đến 31), **MONTH** chứa tháng (1 đến 12) và **YEAR** chứa năm (00 đến 99). Chú ý, DS1307 chỉ dùng cho 100 năm, nên giá trị năm chỉ có 2 chữ số, phần đầu của năm do người dùng tự thêm vào (ví dụ 20xx).

Ngoài các thanh ghi trong bộ nhớ, DS1307 còn có một thanh ghi khác nằm riêng gọi là **con trỏ địa chỉ hay thanh ghi địa chỉ** (Address Register). Giá trị của thanh ghi này là địa chỉ của thanh ghi trong bộ nhớ mà người dùng muốn truy cập. Giá trị của thanh ghi địa chỉ (tức địa chỉ của bộ nhớ) được set trong lệnh Write mà chúng ta sẽ khảo sát trong phần tiếp theo, AVR và DS1307. Thanh ghi địa chỉ được tô đỏ trong hình 6, cấu trúc DS1307.



Hình 6. Cấu trúc DS1307.

- ❖ Bài tập 3.1 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình đọc giá trị analog từ biến trở và hiển thị lên LCD I2C 1602.
- ❖ Bài tập 3.2 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển bật tắt các led vàng, xanh , đỏ theo mức độ tín hiệu nhận được từ cảm biến LDR, hiển thị các thông số lên màn hình LCD I2C 1602.
- ❖ Bài tập 3.3 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển servo quay theo các góc từ 0 -180 bằng tín hiệu nhận được từ biến trở.
- ❖ Bài tập 3.4 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển mở khóa cửa bằng mật khẩu (phần cứng bao gồm LCD I2C 1602, servo, keypad 4x4).
- ❖ Bài tập 3.5 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình đồng hồ số hiển thị trên màn hình OLED.

DK SON - MLAB

BÀI 4 : NỀN TẢNG BLYNK VỚI ESP32

4.1.Nhiệm vụ

Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào Esp32 và sử dụng mô hình thí nghiệm để kiểm chứng.

- Lập trình các ứng dụng IoT cloud dùng Esp32.
- Điều khiển và sử dụng các linh kiện cơ bản như nút nhấn, relay, còi...
- SV hoàn thành các bài tập 4.1 đến 4.3.

4.2.Yêu cầu

- Nắm vững cách lập trình và thiết kế các ứng dụng IoT trên nền tảng Blynk.
- Biết cách viết các chương trình điều khiển gửi và nhận dữ liệu trên các nền tảng cloud.
- Nắm được sơ đồ và nguyên lý hoạt động của các cảm biến HC – SR04, DHT22...

4.3. Giới thiệu

Cảm biến siêu âm là gì?



Cảm biến siêu âm HC-SR04 được sử dụng rất phổ biến để xác định khoảng cách vì RẺ và CHÍNH XÁC. Cảm biến HC-SR04 sử dụng sóng siêu âm và có thể đo khoảng cách trong khoảng từ 2 -> 300cm, với độ chính xác gần như chỉ phụ thuộc vào cách lập trình.

Cảm biến siêu âm HC-SR04 sử dụng nguyên lý phản xạ sóng siêu âm. Cảm biến gồm 2 module. 1 module phát ra sóng siêu âm và 1 module thu sóng siêu âm phản xạ về. Đầu tiên cảm biến sẽ phát ra 1 sóng siêu âm với tần số 40khz. Nếu có chướng ngại vật trên đường đi, sóng siêu âm sẽ phản xạ lại và tác động lên module nhận sóng. Bằng cách đo thời gian từ lúc phát đến lúc nhận sóng ta sẽ tính được khoảng cách từ cảm biến đến chướng ngại vật.

Thông số kỹ thuật và sơ đồ nối chân.

- Điện áp: 5V DC
- Dòng hoạt động: < 2mA
- Mức cao: 5V
- Mức thấp: 0V
- Góc tối đa: 15 độ
- Khoảng cách: 2cm – 450cm (4.5m)

- Độ chính xác: 3mm

Vcc	5V
Trig	Một chân Digital output
Echo	Một chân Digital input
GND	GND

Sơ đồ nối chân giữa cảm biến siêu âm với Arduino

Đo khoảng cách bằng cảm biến siêu âm.



Để đo khoảng cách, ta sẽ phát 1 xung rất ngắn (5 microSeconds - ú) từ chân **Trig**. Sau đó, cảm biến sẽ tạo ra 1 xung HIGH ở chân **Echo** cho đến khi nhận lại được sóng phản xạ ở pin này. Chiều rộng của xung sẽ bằng với thời gian sóng siêu âm được phát từ cảm biến và quay trở lại.

Tốc độ của âm thanh trong không khí là 340 m/s (hàng số vật lý), tương đương với 29,412 microSeconds/cm (106 / (340*100)). Khi đã tính được thời gian, ta sẽ chia cho 29,412 để nhận được khoảng cách.

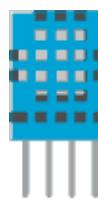
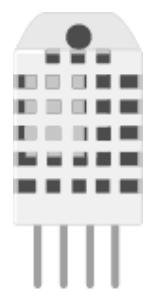
Lưu ý: Cảm biến siêu âm càng xa thì càng bát không chính xác, vì góc quét của cảm biến sẽ mở rộng dần theo hình nón, ngoài ra bề mặt xiên hay xù xì cũng làm giảm độ chính xác của cảm biến, thông số kỹ thuật ghi ở dưới đây là của nhà sản xuất test trong điều kiện lý tưởng, còn thực tế thì tùy theo môi trường làm việc của cảm biến.

Cảm biến DHT11 và DHT22

Cảm biến DHT11 và DHT22 là hai cảm biến được sử dụng phổ biến nhất trong dòng DHTxx. Chúng có vẻ giống nhau và có cùng cấu hình chân kết nối, nhưng thông số kỹ thuật thì khác nhau.

Trong hai loại này, DHT22 đắt hơn và chắc chắn có thông số kỹ thuật tốt hơn. DHT22 có thể đo nhiệt độ từ -40°C đến $+125^{\circ}\text{C}$ với độ chính xác $\pm 0.5^{\circ}\text{C}$, trong khi DHT11 có thể đo nhiệt độ từ 0°C đến 50°C với độ chính xác $\pm 2^{\circ}\text{C}$. Ngoài ra, cảm biến DHT22 có thể đo độ ẩm tương đối từ 0 đến 100% với độ chính xác 2-5%, trong khi cảm biến DHT11 chỉ có thể đo độ ẩm tương đối từ 20 đến 80% với độ chính xác 5%.

Bảng so sánh cấu hình chi tiết cảm biến DHT11 và DHT22:

		
	DHT11	DHT22
Điện áp hoạt động	3-5V	3-5V
Dòng điện hoạt động	2.5mA max	2.5mA max
Phạm vi độ ẩm	20-80% (sai số 5%)	0-100% (sai số 2-5%)
Phạm vi nhiệt độ	0-50°C / ± 2°C	-40 tới 80°C / ± 0.5°C
Tần suất lấy mẫu	1 Hz (1 giây / lần)	0.5 Hz (2 giây / lần)
Kích thước	15.5mm x 12mm x 5.5mm	15.1mm x 25mm x 7.7mm
Ưu điểm	Giá rẻ hơn, tốc độ lấy mẫu cao hơn	Chính xác hơn

Mặc dù DHT22 có độ chính xác cao hơn và có khả năng hoạt động trong một dải nhiệt độ và độ ẩm rộng hơn, nhưng có những ưu điểm mà DHT11 hoàn toàn vượt trội hơn DHT22. Đó là giá thành rẻ hơn, kích thước nhỏ gọn hơn và có tốc độ lấy mẫu cao hơn. DHT11 thực hiện việc đọc dữ liệu một lần mỗi giây (hoặc tốc độ lấy mẫu 1Hz), trong khi DHT22 thực hiện việc đọc dữ liệu một lần mỗi hai giây (hoặc tốc độ lấy mẫu 0.5Hz).

Mặc dù có những khác biệt này, điện áp hoạt động của cả hai cảm biến đều dao động từ 3 đến 5 volt, với dòng điện tối đa là 2.5mA (trong quá trình chuyển đổi). Điểm hay nhất chính là cảm biến DHT11 và DHT22 có cùng chuẩn chân kết nối nên chúng có thể hoán đổi cho nhau, có nghĩa là nếu bạn xây dựng dự án của mình với một cảm biến, bạn có thể dễ dàng rút nó ra và thay thế bằng một cảm biến khác. Mã của bạn có thể cần điều chỉnh một chút, nhưng hệ thống dây vẫn giữ nguyên!

Bên trong cảm biến DHT11 và DHT22

Nếu bạn tháo vỏ cảm biến, bạn sẽ thấy một điện trở nhiệt độ NTC và một cảm biến độ ẩm bên trong.



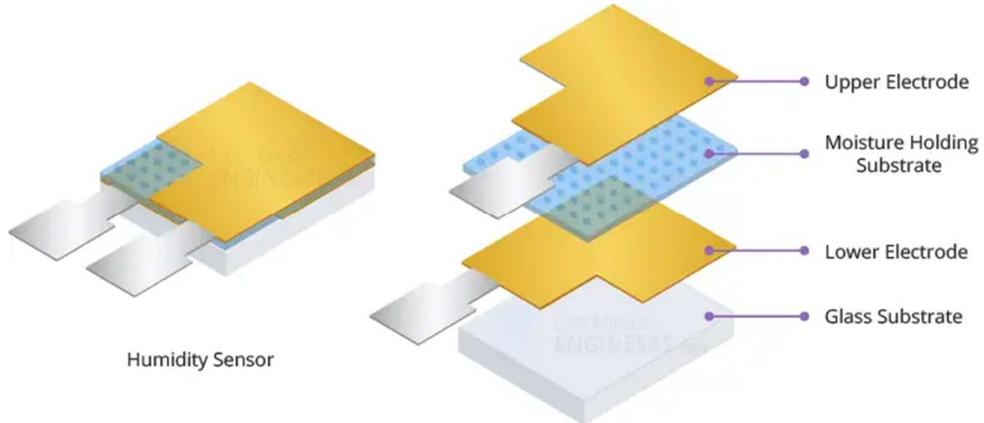
Bên trong cảm biến DHT11



Bên trong cảm biến DHT22

Thành phần cảm biến độ ẩm bên trong DHT11 và DHT22 gồm hai điện cực với một lớp nền giữ ẩm ở giữa (thường là muối hoặc polymer nhựa dẫn điện).

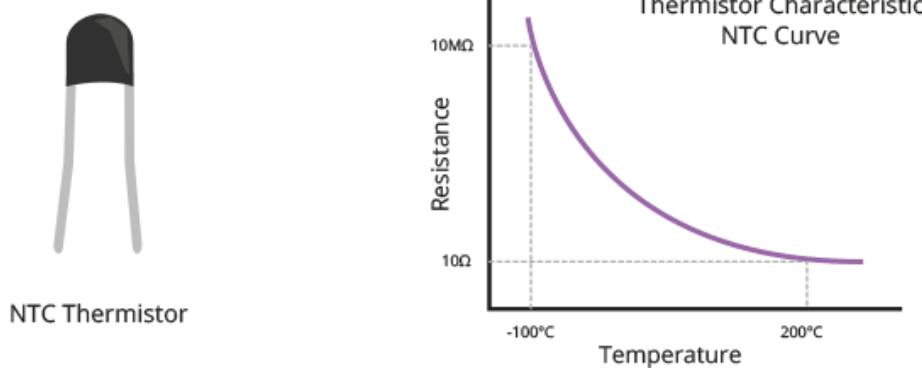
Khi độ ẩm tăng lên, lớp nền sẽ hấp thụ hơi nước, dẫn đến việc giải phóng các ion và làm giảm điện trở giữa hai điện cực. Sự thay đổi điện trở này tỷ lệ thuận với độ ẩm, có thể được đo để ước tính độ ẩm tương đối.



Cấu trúc cảm biến độ ẩm bên trong DHT11 và DHT22

DHT11 và DHT22 cũng bao gồm một điện trở nhiệt NTC để đo nhiệt độ. NTC là một loại điện trở có thể thay đổi giá trị theo nhiệt độ.

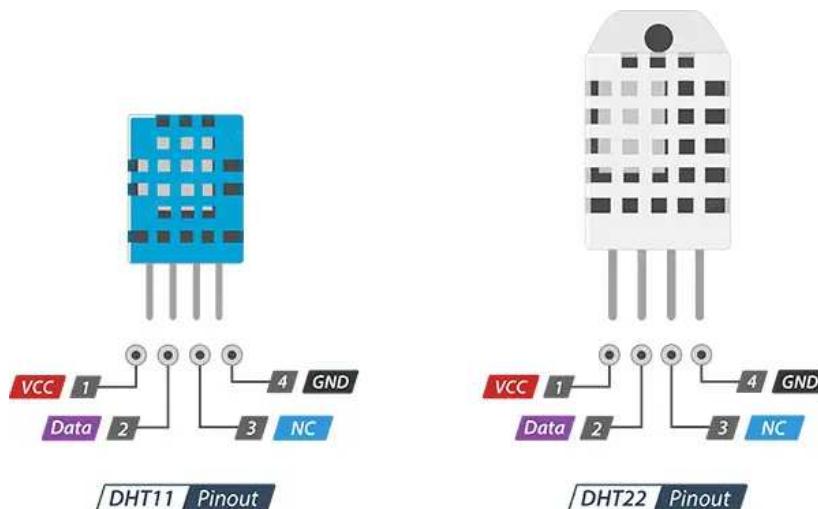
Về mặt kỹ thuật, tất cả các điện trở đều là NTC theo nghĩa là điện trở của chúng thay đổi một chút theo nhiệt độ, nhưng sự thay đổi này thường rất nhỏ và khó đo. Điện trở nhiệt NTC được thiết kế để điện trở của chúng thay đổi đáng kể theo nhiệt độ (khoảng 100 ohms thay đổi mỗi độ). Thuật ngữ “NTC” là viết tắt của “Hệ số nhiệt độ” (Negative Temperature Coefficient), nghĩa là điện trở giảm khi nhiệt độ tăng.



Điện trở nhiệt NTC

Sơ đồ chân cảm biến DHT11 và DHT22

Cảm biến DHT11 và DHT22 đều tương đối đơn giản để kết nối, cảm biến có 4 chân:

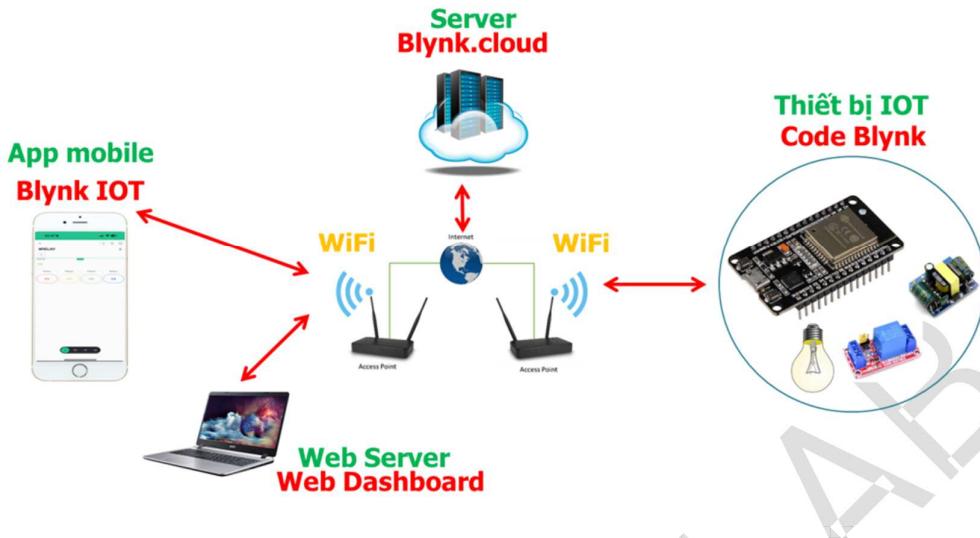


Sơ đồ chân kết nối DHT11 và DHT22

- **VCC(+):** Chân này cung cấp nguồn điện cho cảm biến. Mặc dù điện áp cung cấp có thể dao động từ 3.3V đến 5.5V, nhưng nguồn điện 5V được khuyến nghị sử dụng. Với nguồn cung cấp 5V, cảm biến có thể được đặt cách xa vi điều khiển tới 20m. Với điện áp cung cấp 3.3V, cảm biến chỉ nên được đặt cách 1m nếu không sẽ gây ra hiện tượng sụt áp trên đường dây làm lỗi tín hiệu.
- **Data:** Chân này được sử dụng để giao tiếp giữa cảm biến và vi điều khiển.
- **NC:** không sử dụng (Not Connected)
- **GND(-):** Đây là chân nối đất.

Blynk là gì?

- **Blynk IoT** là một nền tảng IoT platform giúp bạn dễ dàng kết nối và điều khiển các thiết bị IoT từ xa qua internet.
- **Server Blynk** đóng vai trò trung gian, xử lý các yêu cầu từ ứng dụng **Blynk IoT** và các thiết bị IoT như **ESP32**.



Ưu điểm của Blynk

- Dễ sử dụng:** Blynk có giao diện thân thiện và trực quan, giúp bạn dễ dàng tạo các dự án IoT mà không cần nhiều kiến thức chuyên sâu về lập trình.
- Đa nền tảng:** Ứng dụng Blynk IoT hoạt động trên cả Android và iOS, cho phép bạn giám sát và điều khiển thiết bị từ bất kỳ thiết bị di động nào.
- Thời gian thực:** Dữ liệu từ các thiết bị IoT được cập nhật liên tục và hiển thị ngay trên ứng dụng Blynk IoT, giúp bạn giám sát và phản hồi kịp thời.
- Thư viện phong phú:** Blynk hỗ trợ nhiều loại vi điều khiển như ESP32, Arduino, và Raspberry Pi, với thư viện phong phú và dễ tích hợp.
- Bảo mật:** Sử dụng mã xác thực (Auth Token) để kết nối và bảo vệ thông tin giữa ứng dụng Blynk IoT và các thiết bị IoT.

Khuyết điểm của Blynk

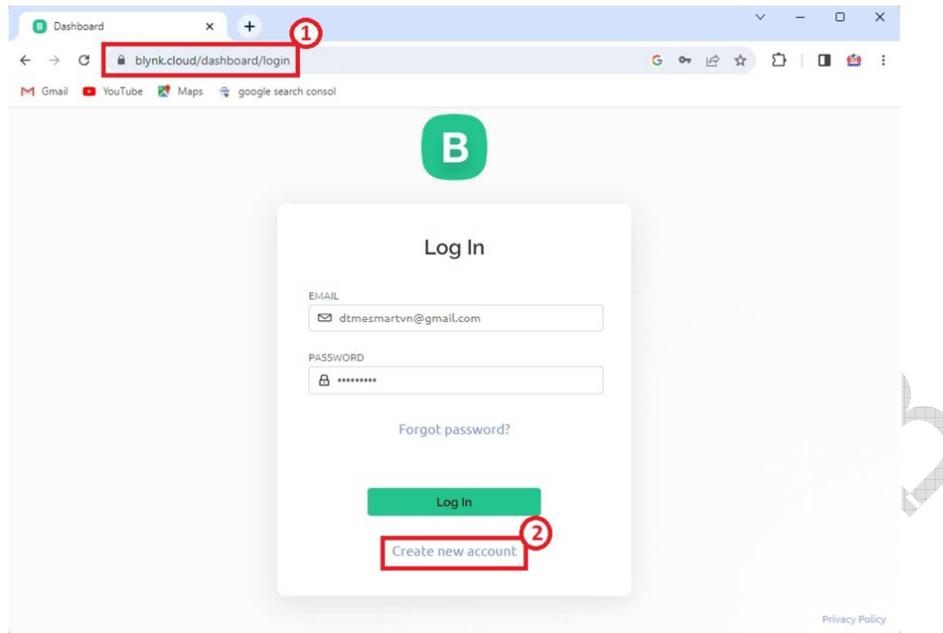
- Giới hạn miễn phí:** Phiên bản miễn phí của Blynk có giới hạn về số lượng widget và thiết bị mà bạn có thể sử dụng. Để sử dụng nhiều hơn, bạn cần nâng cấp lên phiên bản trả phí.
- Phụ thuộc internet:** Blynk yêu cầu kết nối internet liên tục để hoạt động. Điều này không phù hợp cho các ứng dụng cần hoạt động ngoại tuyến.
- Chi phí nâng cấp:** Để sử dụng đầy đủ các tính năng và không bị giới hạn, bạn cần trả phí để nâng cấp tài khoản.

Cách dùng Blynk với ESP32

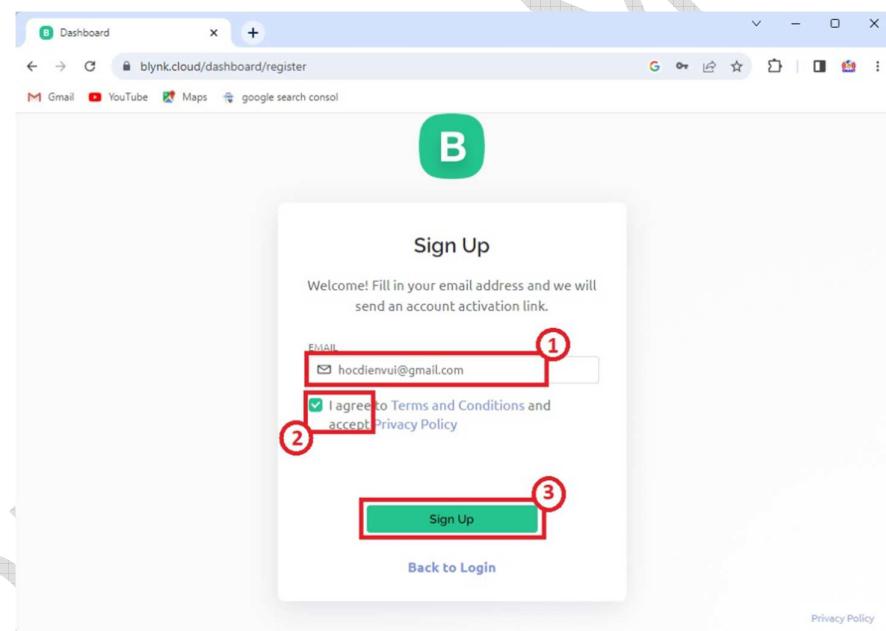
1. Khởi tạo server Blynk

a. Đăng ký tài khoản

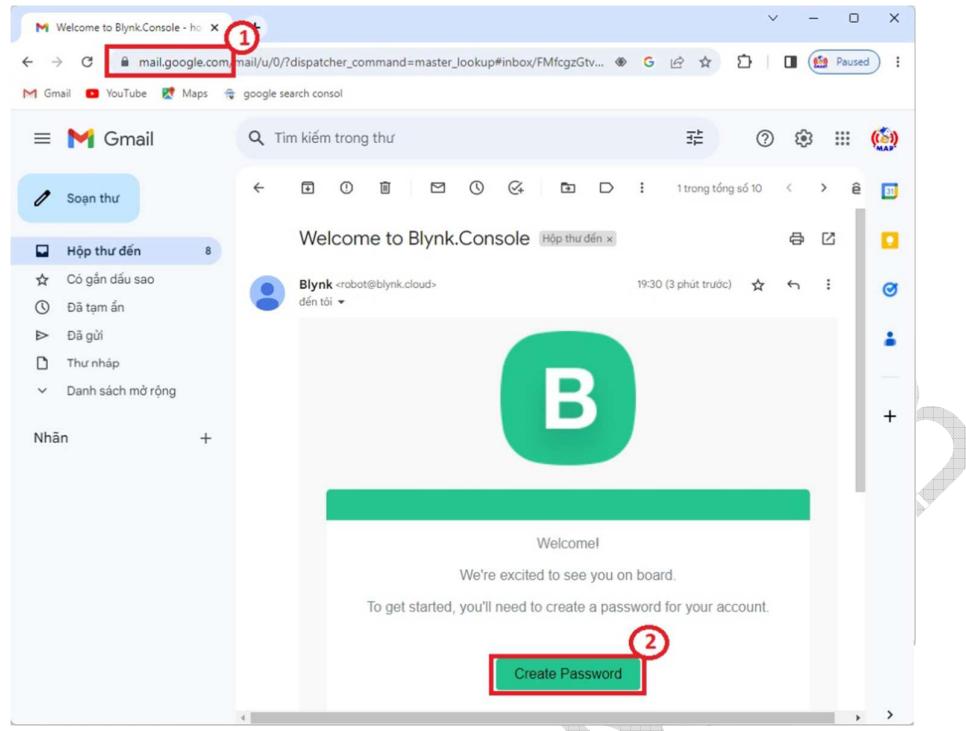
Truy cập vào <https://blynk.cloud/> và chọn **Create new account**



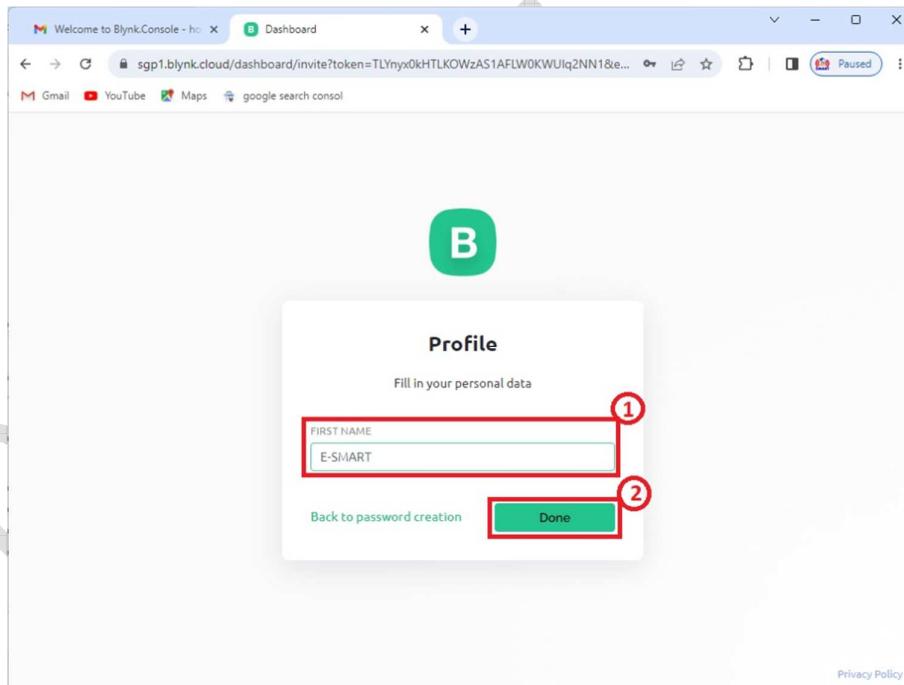
Nhập **email** của bạn và đồng ý với các điều khoản



Sau đó, kiểm tra **hộp thư email** của bạn và nhấn vào nút **Create Password** trong email mà **Blynk** gửi cho bạn.

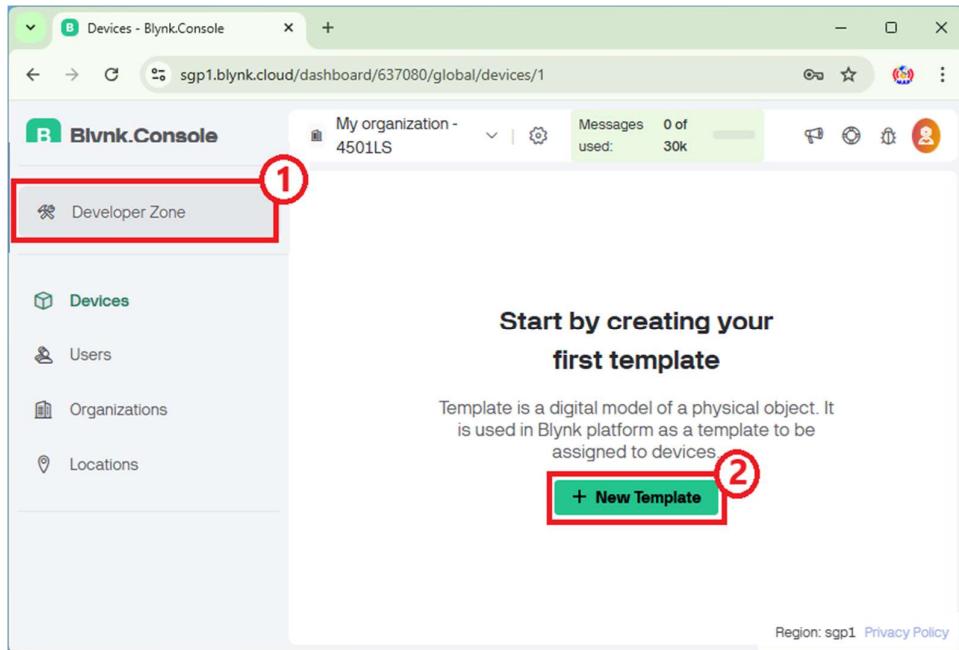


Nhập mật khẩu của bạn và nhấn **Continue**. Sau đó, điền các thông tin cần thiết cho hồ sơ của bạn, như tên, quốc gia, công ty và lĩnh vực hoạt động.

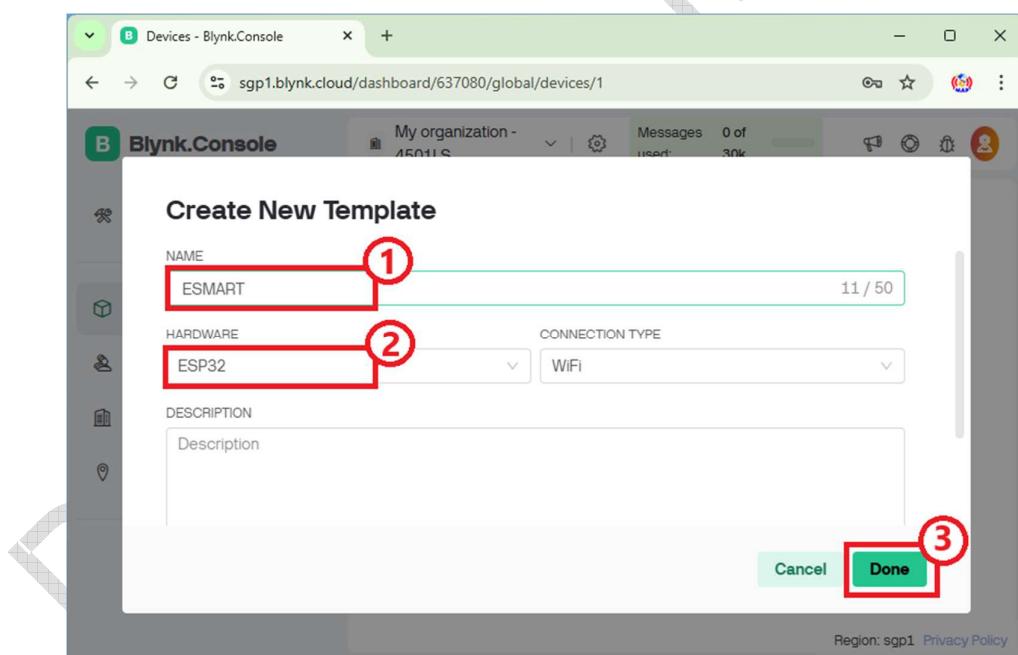


b. Tạo bảng cơ sở dữ liệu Template

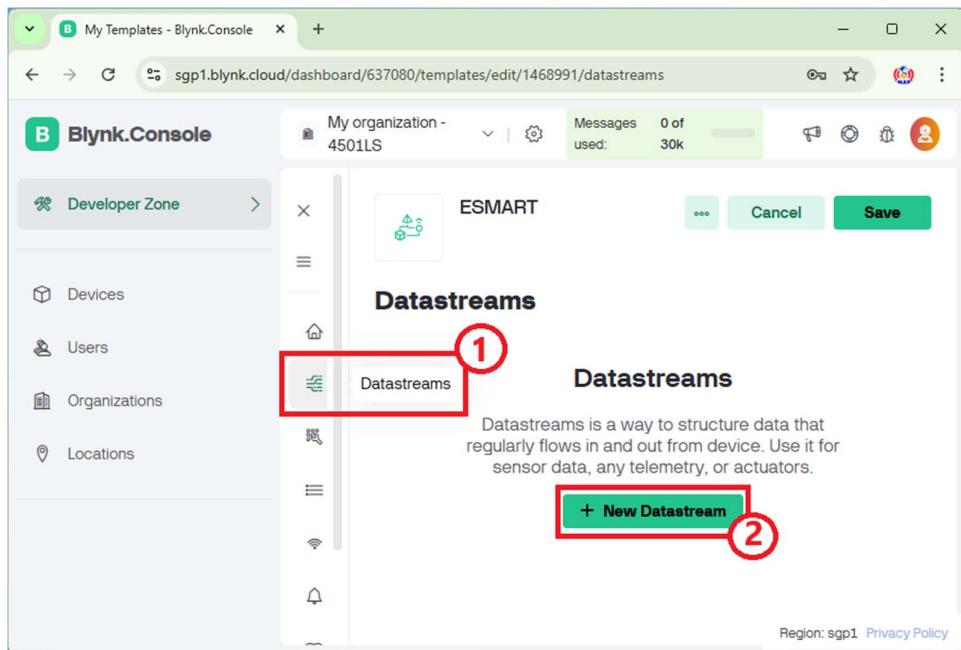
Sau khi hoàn tất hồ sơ, bạn chọn tiếp mục **Developer Zone** -> Chọn **New Template** để tạo 1 bảng cơ sở dữ liệu mới cho project của bạn.



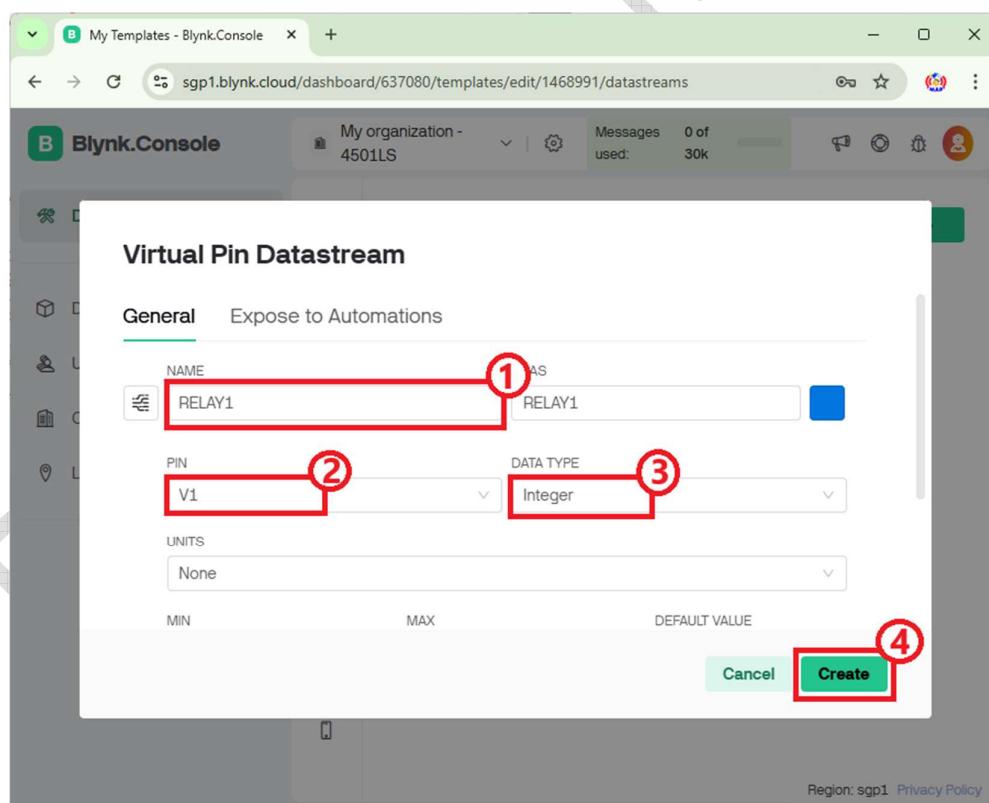
Các bạn điền tên project vào ô **NAME** và chọn **HARDWARE** là **ESP32** sau đó nhấn **Done**.



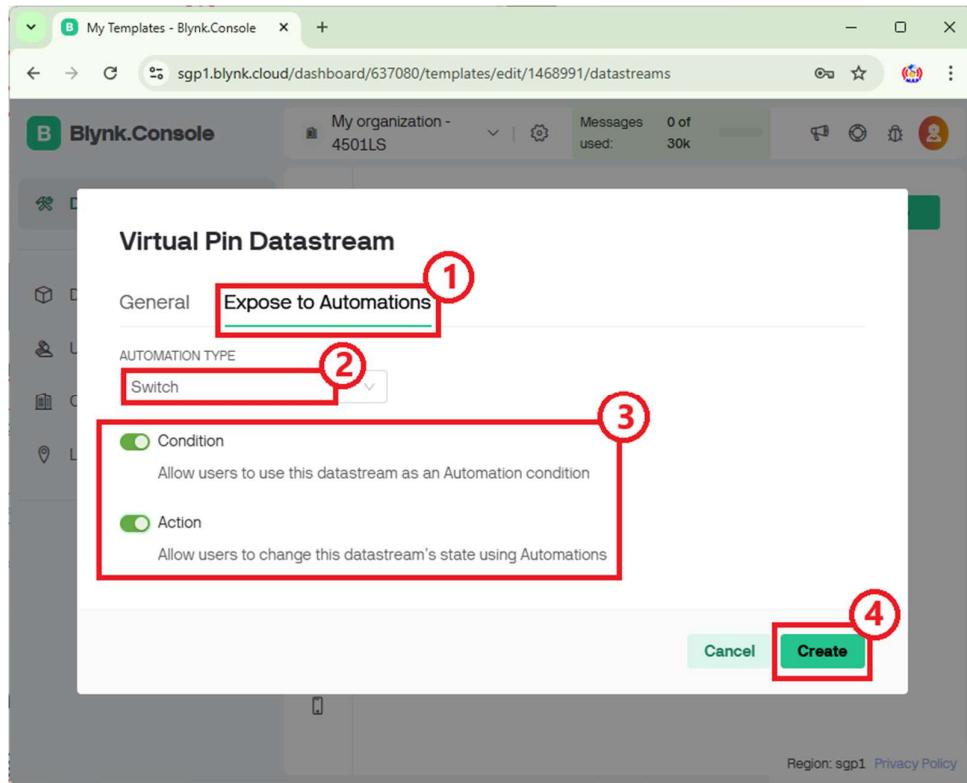
Chúng ta chọn tiếp vào mục **Datastreams** và chọn **New Datastream** để tạo các biến **Virtual Pin** liên kết với bảng cơ sở dữ liệu trên **Blynk**



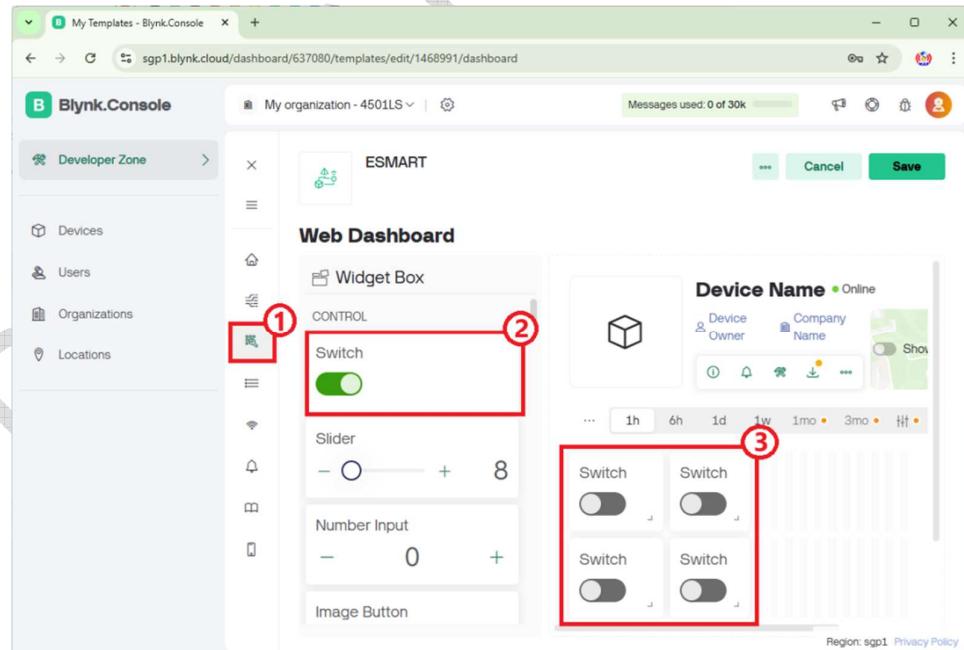
Các bạn sẽ tạo lần lượt các biến **Virtual Pin**, đặt tên và chọn kiểu dữ liệu tương ứng để sử dụng trong quá trình lập trình code trên **ESP32**.



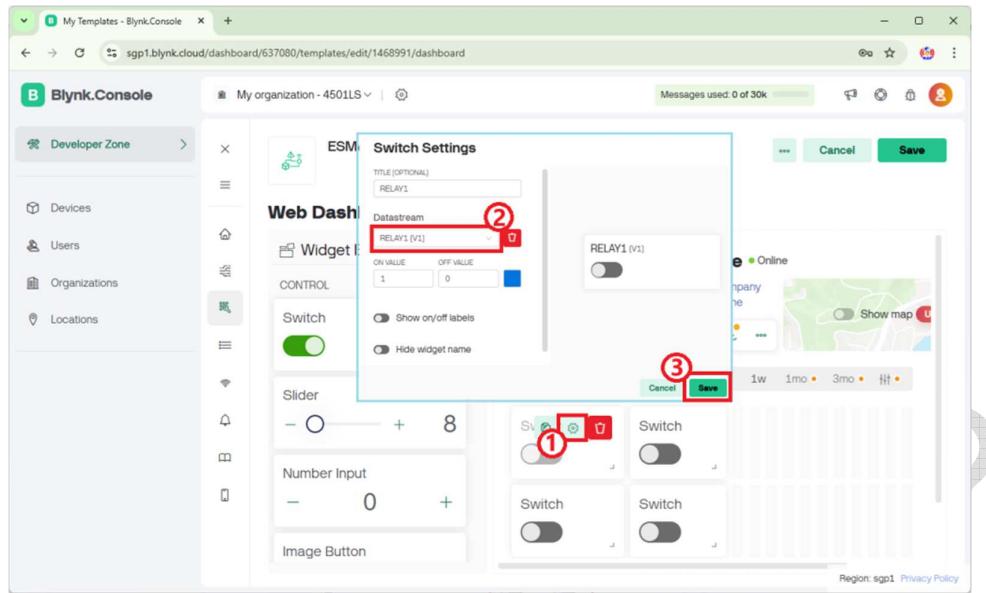
Nếu bạn muốn thêm chức năng tự động hóa cho các biến **Virtual Pin**, bạn cần truy cập vào tab '**Expose to Automations**' để kích hoạt các tính năng tự động hóa. Chức năng này cho phép bạn cài đặt lịch trình, cảnh báo, hoặc điều khiển dựa trên các tình huống cụ thể.



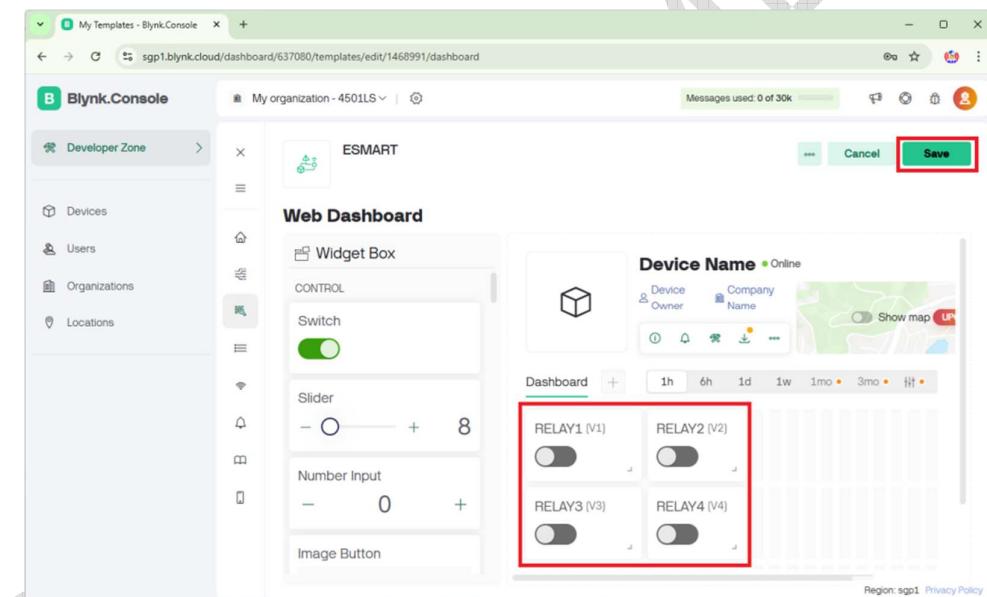
Sau khi tạo các biến **Virtual Pin**, bạn hãy chuyển sang mục **Web Dashboard** để cấu hình giao diện điều khiển và theo dõi qua web. Ta sẽ thêm các **Widget Switch** vào bảng điều khiển để điều khiển các cổng ra trên **ESP32**.



Bạn cần liên kết **Widget Switch** với biến **Virtual Pin** mà bạn đã tạo để có thể điều khiển thông qua cơ sở dữ liệu trên **Blynk**.

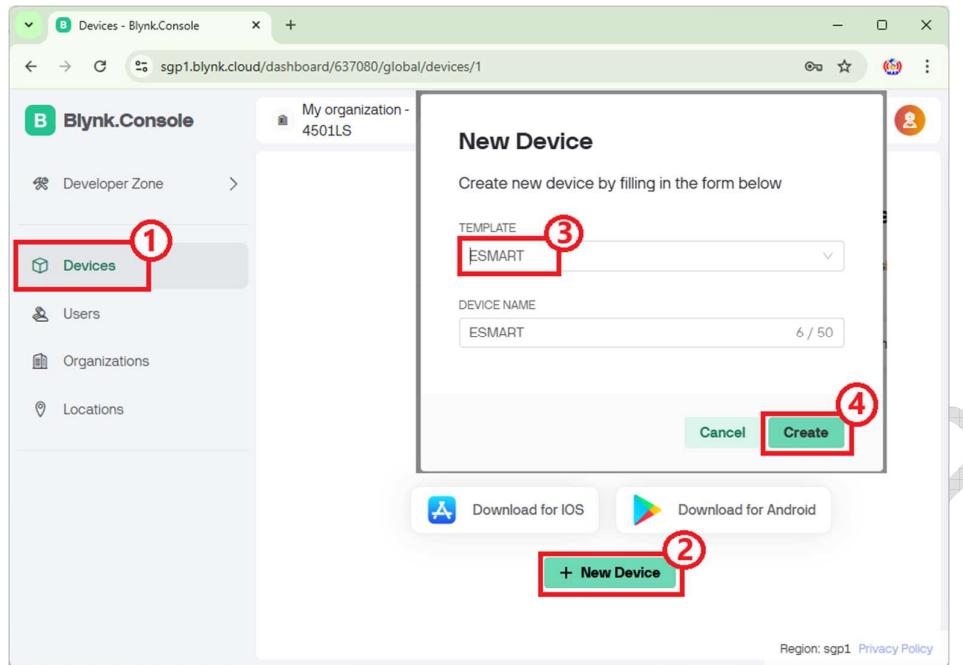


Sau khi hoàn thành xong các bạn ấn vào nút Save để lưu lại Template Blynk.

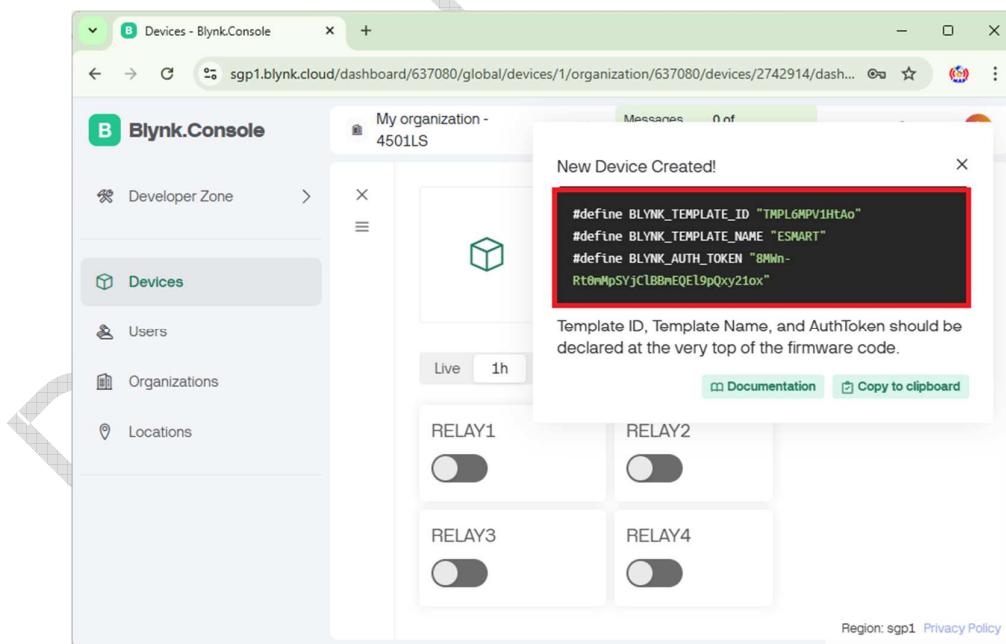


c. Thêm thiết bị mới vào server Blynk

Để thêm thiết bị mới vào server Blynk, hãy chọn ‘Devices’, sau đó ‘New Device’, và ‘From Template’. Chọn Template bạn vừa tạo và nhấn ‘Create’.



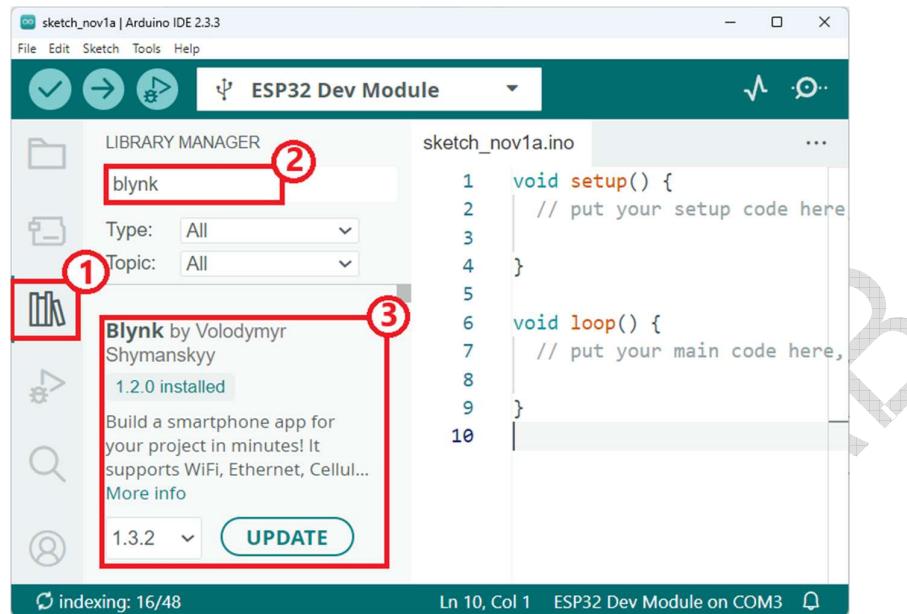
Lúc này, chúng ta đã thêm một thiết bị vào **server Blynk**, và server này sẽ cung cấp các thông tin cần thiết để kết nối với cơ sở dữ liệu của thiết bị, bao gồm: **BLYNK_TEMPLATE_ID**, **BLYNK_TEMPLATE_NAME** và **BLYNK_AUTH_TOKEN**. Bạn cần lưu lại những thông tin này để viết code cho **ESP32**.



2. Kết nối ESP32 với server Blynk

Để kết nối **ESP32** với **server Blynk** chúng ta cần biên soạn và nạp chương trình cho **ESP32**. Trong chương trình đó các bạn cần phải sử dụng thư viện do **Blynk** cung cấp. Ở đây mình sẽ sử dụng phần mềm **Arduino IDE** version 2.3.3 để viết và nạp chương trình cho **ESP32**.

Để cài đặt thư viện **Blynk** vào **Arduino IDE** thì các bạn vào mục **Library Manager** -> tìm từ khóa **Blynk**, sau đó cài đặt thư viện **Blynk**.



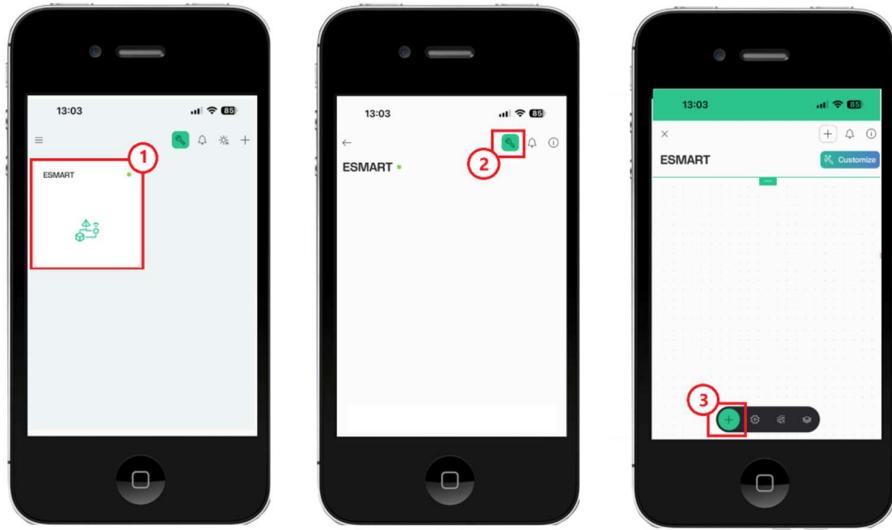
Tiếp theo, để sử dụng được code của **Blynk** thì các bạn có thể truy cập vào website <https://examples.blynk.cc> trên website này sẽ cung cấp cho chúng ta tất cả các code mẫu để sử dụng kết nối với **server Blynk**.

3. Kết nối App Blynk IoT với server Blynk

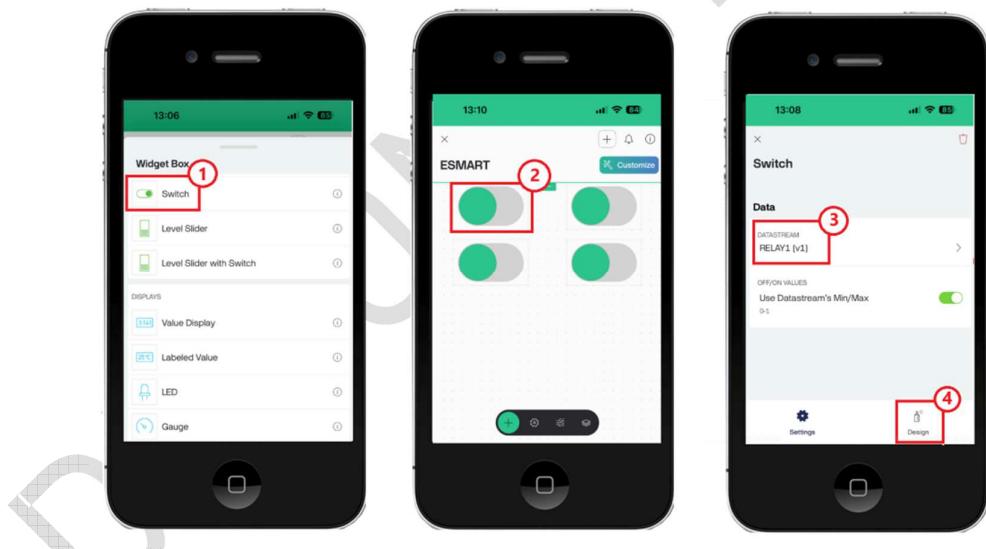
Trên điện thoại, hãy truy cập **CH Play** hoặc **App Store**, tìm kiếm và cài đặt ứng dụng **Blynk IoT**. Khi cài đặt hoàn tất, mở ứng dụng, chọn **Login** và đăng nhập bằng tài khoản **Blynk** đã đăng ký trước đó trên trang web Blynk.cloud.



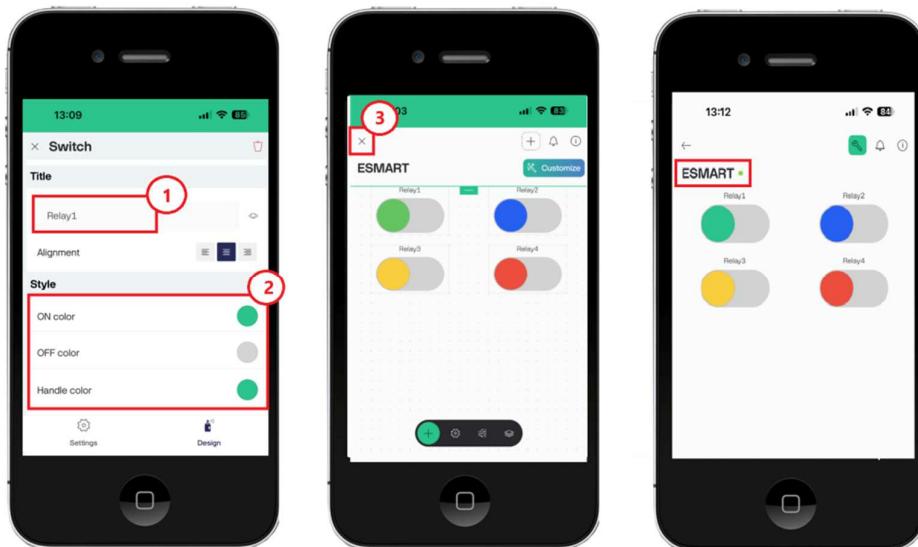
Tiếp tục án vào mục thiết bị có trên màn hình vào chọn vào icon trên góc trên để vào phần thiết lập giao diện. Án tiếp vào mục dấu “+” để thêm các **Widget** vào giao diện điều khiển.



Chọn bốn Widget Switch để tạo giao diện điều khiển cho bốn relay liên kết với **ESP32**. Sau khi chọn và sắp xếp chúng phù hợp, tiến hành thiết lập kết nối tới các biến Virtual Pin đã tạo trên **server Blynk**. Chúng ta sẽ chọn Relay1, Relay2, Relay3, và Relay4 tương ứng. Nhấn vào mục **Design** để cài đặt tile và màu sắc cho mỗi **Switch**.



Sau khi hoàn tất việc thiết lập, chúng ta sẽ có giao diện điều khiển trên ứng dụng **Blynk IoT**. Nếu **ESP32** đã kết nối thành công với **server Blynk**, ứng dụng **Blynk IoT** sẽ hiển thị trạng thái **ONLINE**. Bạn có thể điều khiển các **RELAY** bằng cách sử dụng các công tắc trên giao diện.



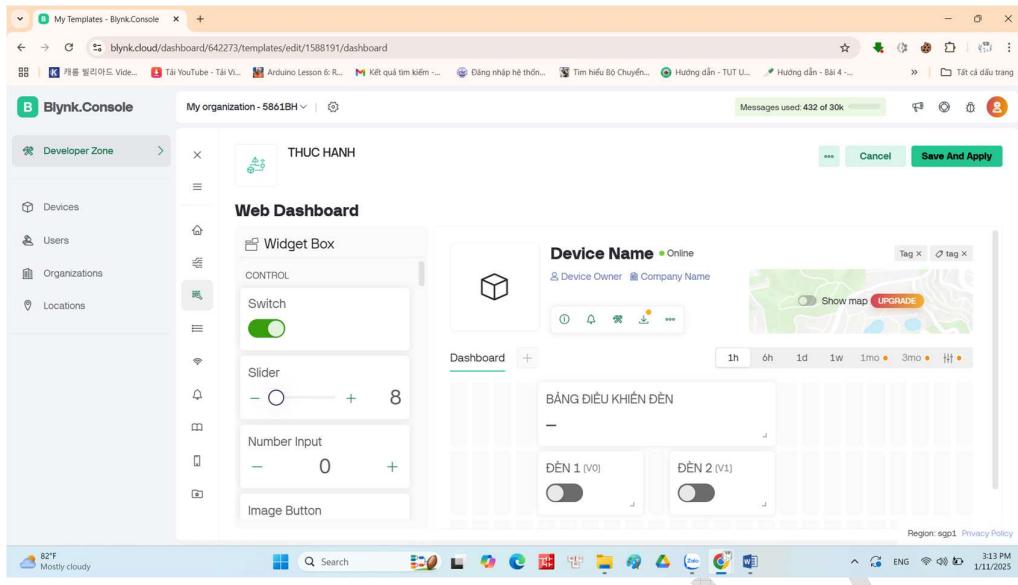
❖ Bài 4.1: Điều khiển 02 led qua Blynk

Tạo 01 Template mới đặt tên tùy ý, sau đó tạo add Device vào (đặt tên tùy ý, chọn loại esp32).

Tạo 02 Pin V0 và V1 với kiểu dữ liệu là integer như hình :

ID	Name	Alias	Color	Pin	Data Type
1	BAT	BAT	Red	V0	Integer
2	TAT	TAT	Green	V1	Integer

Xây dựng giao diện trên Web dashboard như hình dưới: Bao gồm 02 Switch liên kết với 02 virtual pin V0 và V1.

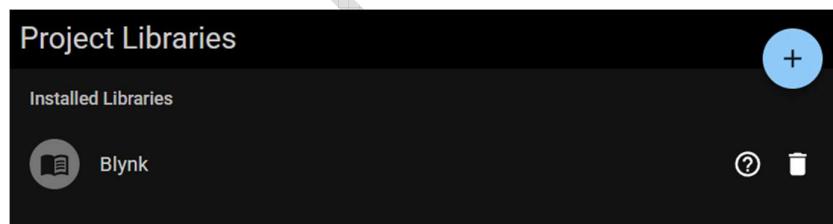


Code trên wokwi :

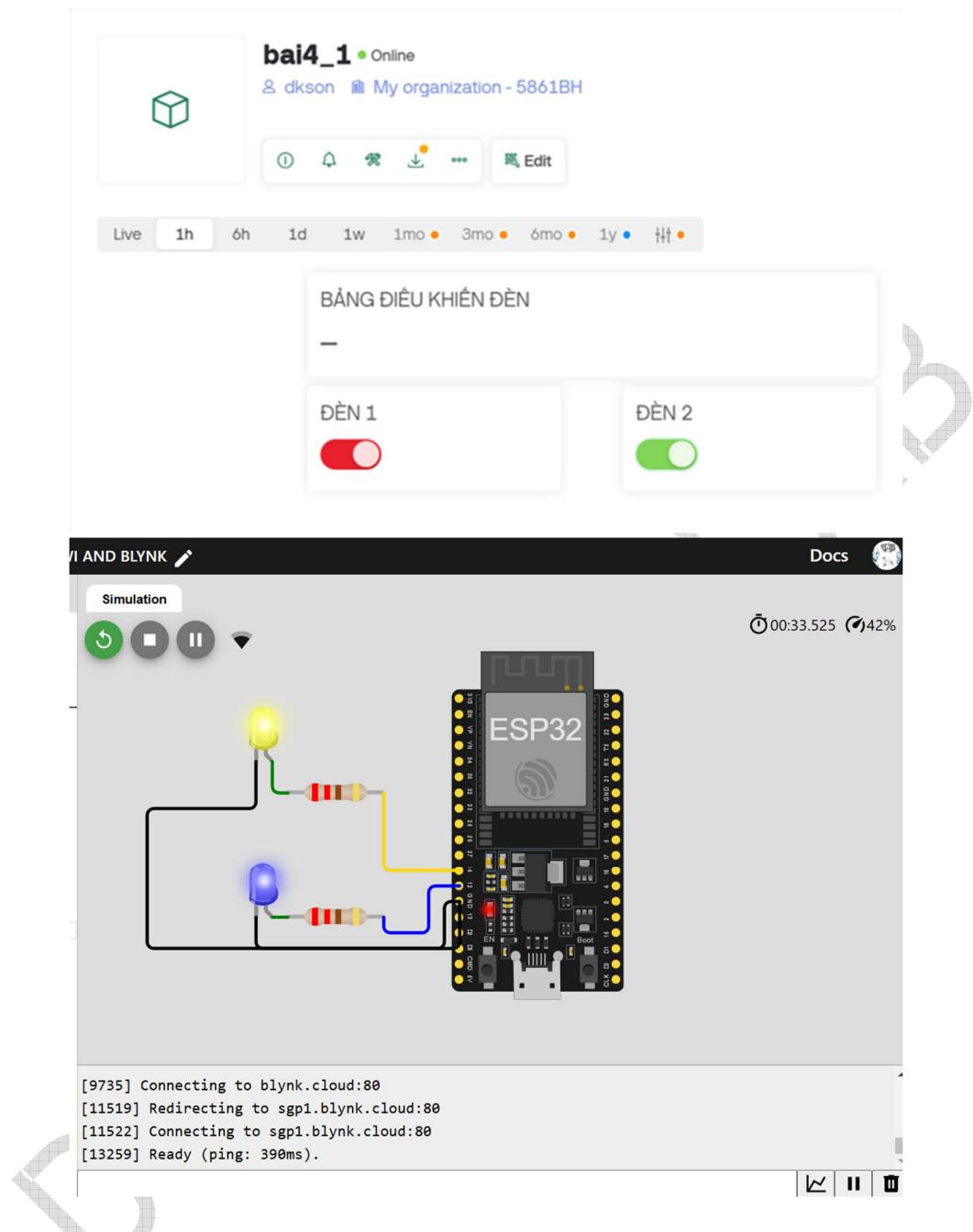
```
#define BLYNK_TEMPLATE_ID "TMPL61qOI8Zx_"
#define BLYNK_TEMPLATE_NAME "THUC HANH"
#define BLYNK_AUTH_TOKEN "1m1-Y25mpN71bL_MC4wJ4-8fxLh8dVaR"
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial
#include <BlynkSimpleEsp32.h>
// Your WiFi credentials.
// Set password to "" for open networks.
//DON'T CHANGE IF YOU USE THIS CODING ON WOKWI
char ssid[] = "Wokwi-GUEST";
char pass[] = "";
int blue = 12;
int green = 14;
BLYNK_WRITE(V0) //for blue led
{
    int buttonState = param.toInt();
    if(buttonState == 1)
    {
        digitalWrite(blue, HIGH);
    }
    if(buttonState == 0)
    {
        digitalWrite(blue, LOW);
    }
}
BLYNK_WRITE(V1) //for green led
{
    int buttonState = param.toInt();
    if(buttonState == 1)
```

```
{  
    digitalWrite(green, HIGH);  
}  
if(buttonState == 0)  
{  
    digitalWrite(green, LOW);  
}  
}  
void setup()  
{  
    Serial.begin(115200);  
    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);  
    pinMode(blue, OUTPUT);  
    pinMode(green, OUTPUT);  
    digitalWrite(blue, LOW);  
    digitalWrite(green, LOW);  
}  
void loop()  
{  
    Blynk.run();  
}
```

Cần add thêm thư viện Blynk trên wokwi:



Kết quả mô phỏng trên Wokwi liên kết với Blynk :



- ❖ Bài tập 4.2 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình giám sát nhiệt độ, độ ẩm vườn rau. Hệ thống gồm cảm biến DHT 22, cập nhật dữ liệu nhiệt độ, độ ẩm lên Blynk.
- ❖ Bài tập 4.3 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình giám sát mực nước trong bồn dùng nền tảng Blynk. Hệ thống gồm cảm biến HC-SR04, còi, led đỏ, relay, chức năng đo mực nước, hiển thị mực nước và gởi cảnh báo (khi mức nước >100 cm) lên ứng dụng, bật tắt relay trên ứng dụng.

BÀI 5 : ARDUINO CLOUD với ESP32

5.1.Nhiệm vụ

Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào Esp32 và sử dụng mô hình thí nghiệm để kiểm chứng.

- Lập trình các ứng dụng IoT cloud dùng Esp32.
- Điều khiển và sử dụng các linh kiện cơ bản như nút nhấn, relay, còi...
- SV hoàn thành các bài tập 5.1 đến 5.2.

5.2.Yêu cầu

• Nắm vững cách lập trình và thiết kế các ứng dụng IoT trên nền tảng arduino cloud.

• Biết cách viết các chương trình điều khiển gửi và nhận dữ liệu trên các nền tảng cloud.

• Nắm được sơ đồ và nguyên lý hoạt động của các cảm biến HC – SR04, DHT22...

5.3. Giới thiệu

Arduino Cloud

Arduino [Cloud](#) là nền tảng để phát triển các dự án Arduino và kết nối chúng với thế giới. Nó hỗ trợ kết nối an toàn với các bo mạch thông qua [Wi-Fi®](#), [LoRa®](#), [Ethernet](#) và [Cellular \(GSM/NB-IoT\)](#) và cho phép bạn tạo một hệ thống để gửi bất kỳ thông tin biến đổi nào từ bo mạch này sang bo mạch khác.

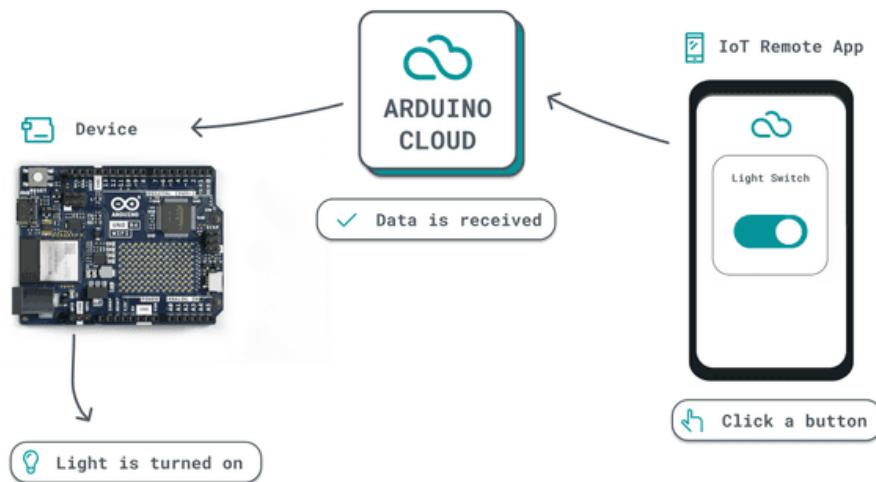
Nền tảng Arduino Cloud bao gồm:

- Môi trường phát triển tích hợp (IDE) để lập trình bo mạch của bạn,
- một dịch vụ đám mây để đồng bộ hóa dữ liệu từ các bo mạch Arduino, cũng như từ các máy khách [Python](#) & [JavaScript](#),
- một công cụ đồ họa (bảng điều khiển) để kiểm soát và giám sát bảng của bạn (cũng như một [ứng dụng di động](#)).
- [REST API](#) và [các công cụ dòng lệnh](#) cho các hoạt động tự động hóa quy mô lớn hơn.

Nói một cách rất đơn giản, với Arduino Cloud, bạn có thể:

1. Tạo một chương trình cho Arduino dựa trên ý tưởng của bạn.
2. Tải chương trình lên bo mạch của bạn và đồng bộ hóa bất kỳ dữ liệu nào bạn muốn (thường là qua Wi-Fi®).
3. Tạo bảng điều khiển để kiểm soát và theo dõi dữ liệu của bạn.

Ví dụ, việc BẬT/TẮT đèn được kết nối với thiết bị sẽ hoạt động như sau:



Tương tác với thiết bị của bạn

Và nếu bạn muốn đọc giá trị của cảm biến được kết nối với thiết bị của mình, nó sẽ hoạt động như thế này:



Theo dõi thiết bị của bạn.

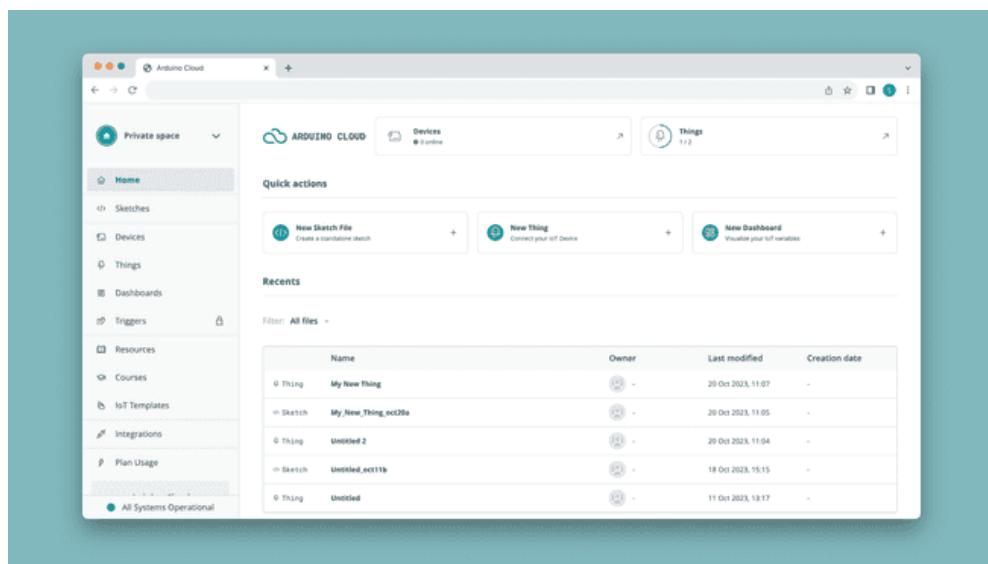
Bo mạch tương thích

Board là phần cứng vật lý hoặc thiết lập ảo (Python/JS) của bạn, còn "device" là cách nó được cấu hình trong Cloud.

Khả năng tương thích với Arduino Cloud được chia thành hai loại:

- **Hỗ trợ Cloud Editor** - bạn có thể lập trình **bắt** kỳ bo mạch Arduino chính thức nào trong Cloud Editor.
- **Hỗ trợ IoT** - bo mạch có mô-đun radio (ví dụ: Wi-Fi®) được hỗ trợ. Bo mạch dựa trên ESP32 cũng được hỗ trợ.

Tổng quan



Trang chủ Arduino Cloud.

Hướng dẫn sử dụng Arduino Cloud

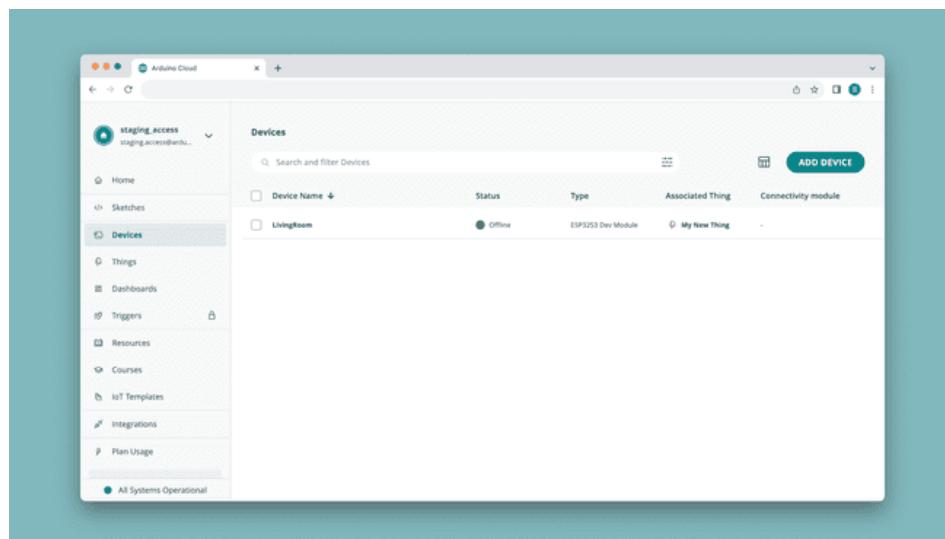
Việc thiết lập một dự án trên Arduino Cloud rất dễ dàng và có thể thực hiện thông qua một vài bước đơn giản được tóm tắt trong phần này.

1. Tạo một tài khoản

Để sử dụng Arduino Cloud, bạn sẽ cần có tài khoản Arduino, bạn có thể đăng ký [tại đây](#).

2. Cấu hình thiết bị

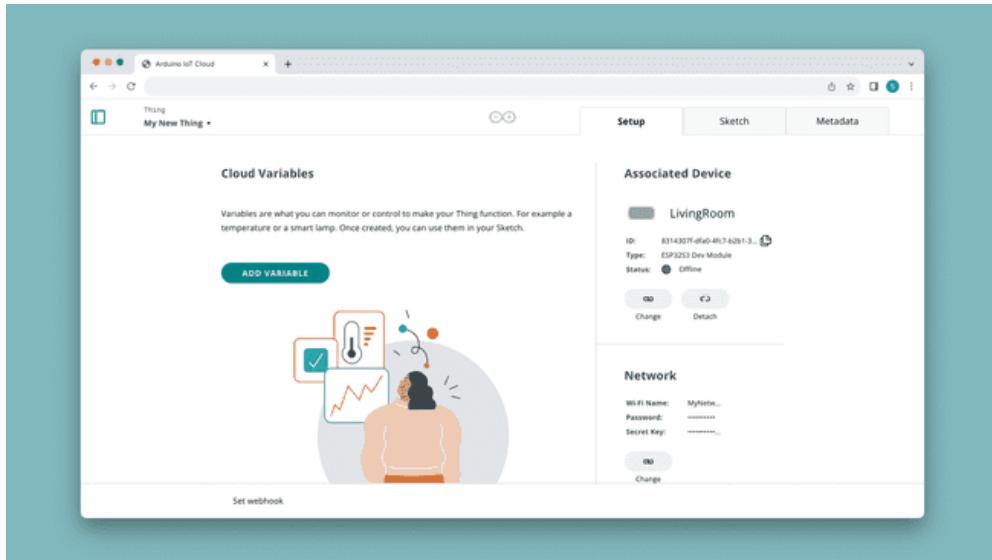
Đầu tiên, bạn cần kết nối bo mạch với máy tính và cấu hình thiết bị trong tab [Thiết bị](#).



Các thiết bị trong Arduino Cloud.

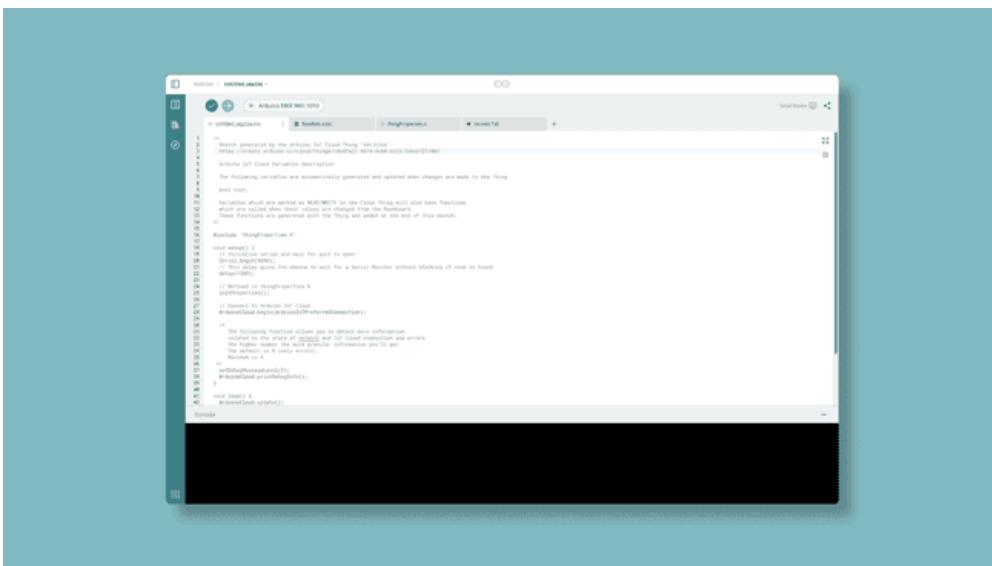
3. Tạo ra một thứ gì đó

Sau khi cấu hình thiết bị, chúng ta có thể tạo Thing, là **bản sao ảo** của bo mạch của bạn. Ở đây chúng ta cấu hình chi tiết mạng, chọn thiết bị muốn liên kết và tạo các biến muôn đồng bộ hóa.



4. Tạo Sketch

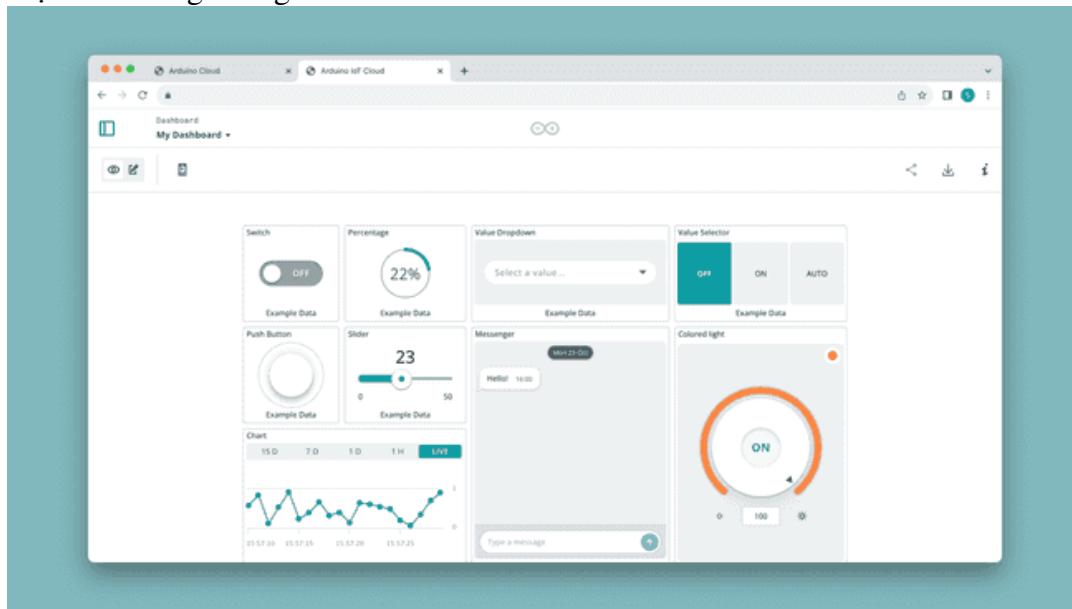
Sau khi bạn đã thực hiện các cấu hình trên, bạn có thể chuyển sang **tạo** Sketch. Đây là nơi *bạn* lập trình cho dự án của mình.



Sketch trên Arduino Cloud.

5. Tạo Dashboard

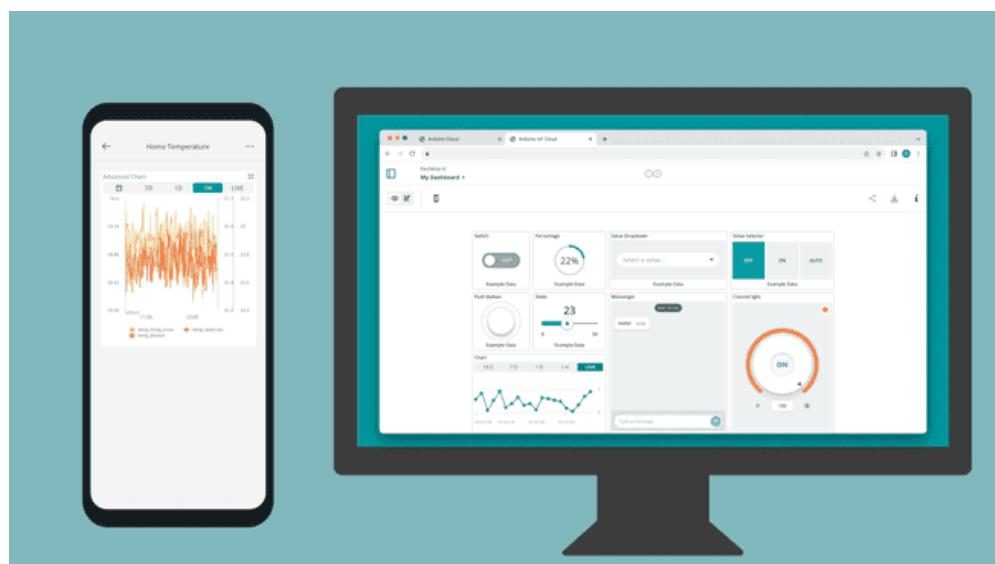
Dashboard cho phép bạn tương tác với thiết bị của mình từ giao diện web hoặc ứng dụng di động. Bảng điều khiển bao gồm **widgets**, mà bạn có thể liên kết đến một biến trong Thing của mình.



Dashboard trong Arduino Cloud.

6. Theo dõi dự án của bạn

Sau khi hoàn tất mọi cấu hình, tạo chương trình và bảng điều khiển, bạn có thể theo dõi và tương tác với dự án của mình từ giao diện web hoặc ứng dụng di động.



Arduino Cloud Dashboard & Ứng dụng từ xa IoT

ESP32 / ESP8266 với Arduino Cloud

Arduino Cloud hỗ trợ nhiều loại bo mạch phát triển dựa trên ESP32 / ESP8266. Các chip ESP rất phù hợp cho bất kỳ dự án IoT nào và chúng có thể được lập trình bằng ngôn ngữ Arduino (C++).

Thiết lập các bo mạch dựa trên ESP trong Arduino Cloud rất nhanh chóng và đơn giản. Nó được thực hiện bằng cách tạo **Device ID** và **Secret Key**, cùng với thông tin xác thực Wi-Fi® của bạn là đủ để kết nối với Arduino Cloud.

Các bo mạch ESP32 và ESP8266 là các bo mạch của bên thứ ba.

Cài đặt

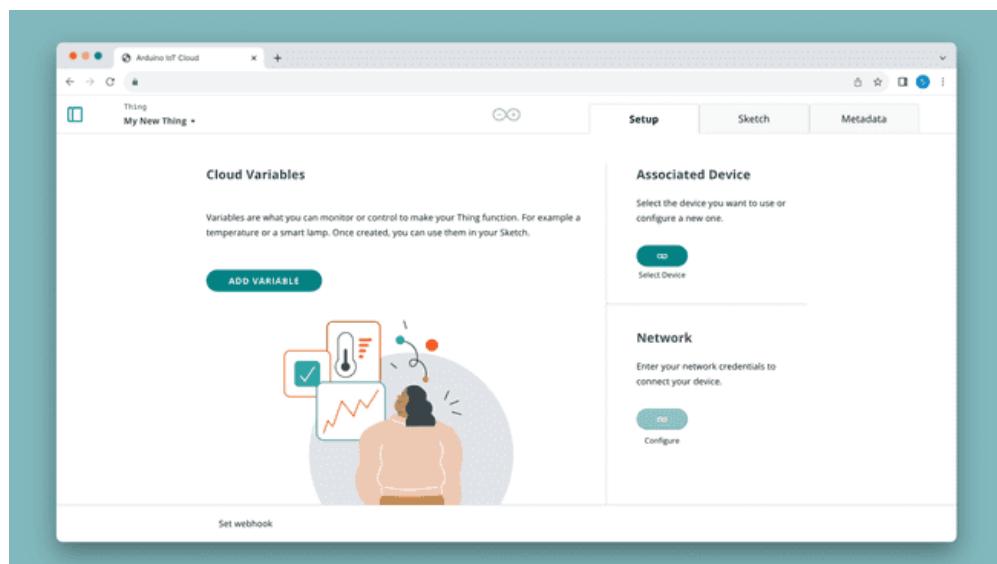
Cấu hình thiết bị

Đầu tiên chọn [Arduino Cloud - Devices](#). Tại đây bạn có thể xem tất cả các thiết bị của mình và cấu hình một thiết bị mới.

1. Nhập vào " **Add Device** "
2. Chọn " **Third Party Device** "
3. Chọn bảng của bạn từ danh sách (nếu không có, hãy chọn ví dụ ESP32S3 Dev Board).
4. Đặt tên cho bảng của bạn, ví dụ:
Device_ESP32_LivingRoom.
5. Lưu **Device ID** và **Secret Key** của bạn .

Cấu hình Thing

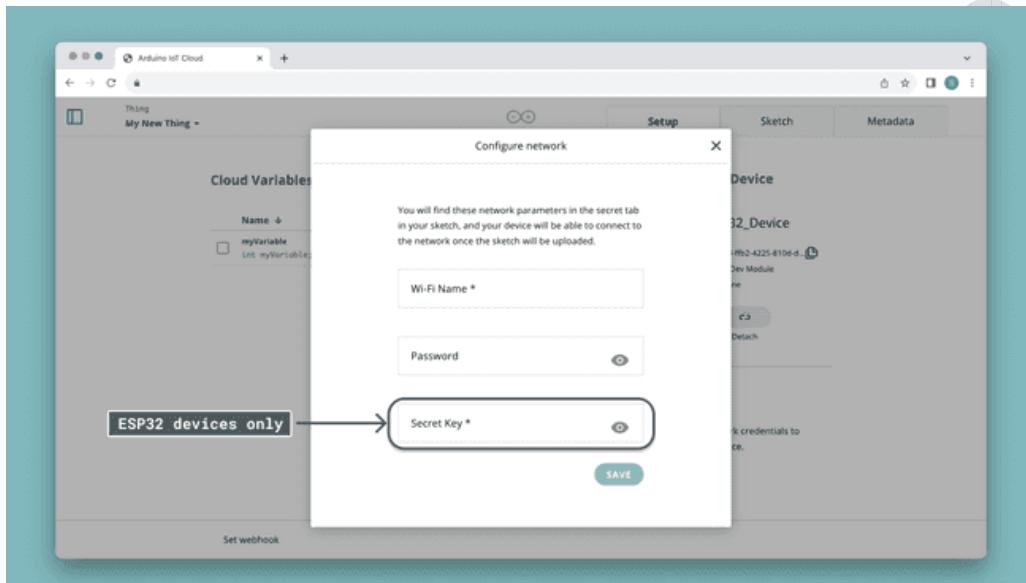
Tiếp theo, chọn tab **Things**. Tại đây, bạn sẽ thấy danh sách Things của mình và nút để tạo Things mới.



Giao diện Arduino Cloud Thing

"Thing" là bản sao ảo của phần cứng của bạn và tại đây chúng ta tạo các biến mà chúng ta muốn đồng bộ hóa giữa Cloud và bo mạch.

1. Trước tiên, hãy kết nối thiết bị mà chúng ta muốn sử dụng bằng cách nhấp vào nút "Select Device" trong phần "Associated Devices" ở bên phải.
2. hãy tạo một biến mới, gọi nó là test và kiểu **boolean** và có quyền **read/write**.
3. Cuối cùng, cấu hình mạng của bạn trong phần **Network**. Tại đây, bạn sẽ nhập thông tin xác thực Wi-Fi® và **Secret Key** của mình , được lấy khi cấu hình thiết bị.



Nhập thông tin đăng nhập mạng.

Sketch được tạo tự động hiện có thể chỉnh sửa. Sketch này bao gồm mọi thứ cần thiết để kết nối với Đám mây và có một hàm gọi lại được tạo cho mỗi biến **đọc/ghi**.

Dưới đây là Sketch được tạo ra cho biến test.

```
#include "thingProperties.h"

void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
    Serial.begin(9600);
    delay(1500);
    initProperties();
    ArduinoCloud.begin(ArduinoIoTPREFERRED_CONNECTION);
    setDebugMessageLevel(2);
    ArduinoCloud.printDebugInfo();
}

void loop() {
```

```

ArduinoCloud.update();

}

void onTestChange() {
    if(test){
        digitalWrite(LED_BUILTIN, HIGH);
    }
    else{
        digitalWrite(LED_BUILTIN, LOW);
    }
}

```

- Sketch sẽ tự động được cập nhật bất cứ khi nào bạn thay đổi Thing (ví dụ: thêm biến, thay đổi thiết bị),
`ArduinoCloud.update()`
 Chức năng đồng bộ hóa dữ liệu giữa bảng và Đám mây.

Compile & Upload

- Trước tiên hãy đảm bảo rằng bạn đã cài đặt [Create Agent](#). Điều này cho phép Arduino Cloud giao tiếp với bo mạch của bạn trong giao diện web.
- Kiểm tra xem bo mạch của bạn đã được kết nối và hiển thị trong menu chọn bo mạch chưa.
- Nhấp vào nút verify/upload.
- Chờ cho đến khi code được tải lên thành công.
- Mở serial monitor để kiểm tra thông báo gỡ lỗi. Nếu bo mạch của bạn không kết nối được, nó sẽ in lỗi ở đây.

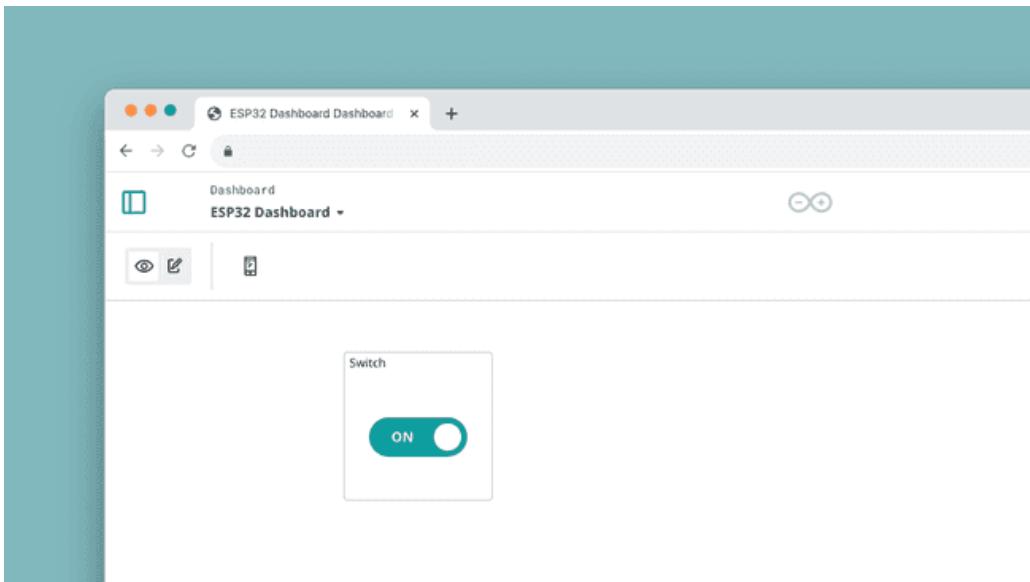
Verify Connection

Sau khi tải lên hoàn tất, bạn có thể Verify Connection bằng cách kiểm tra giao diện Thing. Tại đây, bạn có thể xem giá trị và dấu thời gian mới nhất cũng như trạng thái thiết bị của bạn (trực tuyến/ngoại tuyến).

Để kiểm soát trạng thái của test biến, chúng ta có thể thiết lập **dashboard** và **switch widget**. Điều này sẽ cho phép chúng ta điều khiển đèn LED từ xa.

- Vào **dashboard** và tạo **dashboard** mới.
- Nhấp vào nút chỉnh sửa ở góc trên bên trái, sau đó nhấp vào nút "**Add**". Chọn Thứ bạn muốn liên kết với nó, sau đó nhấp vào "**Create Widgets**".

3. Một switch widget sẽ được tạo ra, nó được liên kết với trạng thái của đèn LED (bật/tắt).



Bảng điều khiển trong Arduino Cloud.

- ❖ Bài tập 5.1 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình điều khiển 3 đèn xanh, đỏ, vàng, một động cơ servo, một cảm biến hồng ngoại PIR. Hệ thống có thể bật tắt các đèn, điều khiển góc quay servo trên ứng dụng hoặc app arduino cloud, cảnh báo khi phát hiện người (dùng cảm biến PIR).
- ❖ Bài tập 5.2 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình giám sát vườn rau dùng nền tảng arduino cloud. Hệ thống gồm cảm biến LDR, DHT 22, relay, chức năng đo độ sáng, nhiệt độ, độ ẩm, hiển thị lên ứng dụng, bật tắt relay trên ứng dụng.

BÀI 6 : THINGSPEAK VỚI ESP32

6.1.Nhiệm vụ

Thực hành lập trình ứng dụng trên máy tính, biên dịch chương trình, nạp vào Esp32 và sử dụng mô hình thí nghiệm để kiểm chứng.

- Lập trình các ứng dụng IoT cloud dùng Esp32.
- Điều khiển và sử dụng các linh kiện cơ bản như led, LCD, relay...
- SV hoàn thành các bài tập 6.1 đến 6.2.

6.2.Yêu cầu

- Nắm vững cách lập trình và thiết kế các ứng dụng IoT trên nền tảng Thingspeak.
- Biết cách viết các chương trình điều khiển gửi và nhận dữ liệu trên các nền tảng cloud.
- Nắm được sơ đồ và nguyên lý hoạt động của các cảm biến DHT22...

6.3. Giới thiệu

Cách sử dụng Thingspeak

Thingspeak là gì?

Thingspeak là một nền tảng mà bạn có thể hiển thị dữ liệu trên Cloud. Bạn có thể truy cập để hiển thị hoặc lấy dữ liệu từ Cloud về thiết bị IOT thông qua giao thức HTTP. Thingspeak hoàn toàn miễn phí và giao diện cũng khá đẹp nên có rất nhiều người trên thế giới sử dụng.

Mô hình của Thingspeak

Cách lấy API Thingspeak

Tương tự như cách giao tiếp với các server khác, Thingspeak sử dụng API và có 1 key để định danh người sử dụng.

Đầu tiên đăng kí và đăng nhập vào <https://thingspeak.com/>

Vào Channels – My channels- New channels

My Channel:

Name	Created	Updated
🔒 SIM808	2021-04-20	2021-04-22 09:49
🔒 Test ESP32	2021-05-20	2021-05-20 01:55
🔒 hanoi_weather	2021-05-20	2021-05-20 06:40
🔒 ESP32_SIM	2021-06-24	2021-06-24 07:52

Trong new channel Tạo tên bảng của bạn. Các trường cần hiển thị. Ở đây mình sử dụng 2 trường là Temparature và Humidity. Nhấn Save channel để hoàn thành.

New Channel

Name	ESP22_Khuenguyencreator
Description	Quản trắc nhiệt độ, độ ẩm
Field 1	Temp <input checked="" type="checkbox"/>
Field 2	Humi <input checked="" type="checkbox"/>
Field 3	<input type="checkbox"/>
Field 4	<input type="checkbox"/>
Field 5	<input type="checkbox"/>
Field 6	<input type="checkbox"/>
Field 7	<input type="checkbox"/>
Field 8	<input type="checkbox"/>

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- Percentage complete: Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- Channel Name: Enter a unique name for the ThingSpeak channel.
- Description: Enter a description of the ThingSpeak channel.
- Fields: Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata: Enter information about channel data, including JSON, XML, or CSV data.
- Tags: Enter keywords that identify the channel. Separate tags with commas.
- Link to External Site: If you have a website that contains information about your ThingSpeak channel, specify the URL.
- Show Channel Location:
 - Latitude: Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.

Chuyển qua tap API Keys để lấy key và example. Các bạn có thể copy Example và dùng postman để kiểm thử

The screenshot shows the Thingspeak API Keys settings page. At the top, there are tabs for Private View, Public View, Channel Settings, Sharing, API Keys (which is selected), and Data Import / Export. Below the tabs, there are two main sections:

- Write API Key:** A text input field contains the key "J69MXGIE0KLMA27C". A red arrow points from the text "lần ghi lên Thingspeak" to this field.
- Read API Keys:** A text input field contains the key "P5RU1206P88KZ9YZ". A red arrow points from the text "lần ghi lên Thingspeak" to this field.

Below these sections are "API Keys Settings" and "API Requests" sections, which are not directly related to the text "lần ghi lên Thingspeak".

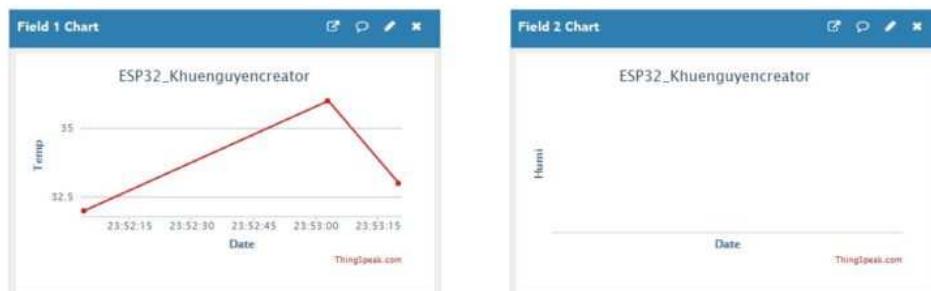
Sử dụng Postman thử gửi 3 nhiệt độ là 32,36 và 33. Phản response trả về sẽ là số lần ghi lên Thingspeak

The screenshot shows a POST request in Postman. The URL is `https://api.thingspeak.com/update?api_key=J69MXGIE0KLMA27C&field1=36`. The "Params" tab is selected, showing two parameters: `api_key` with value `J69MXGIE0KLMA27C` and `field1` with value `36`.

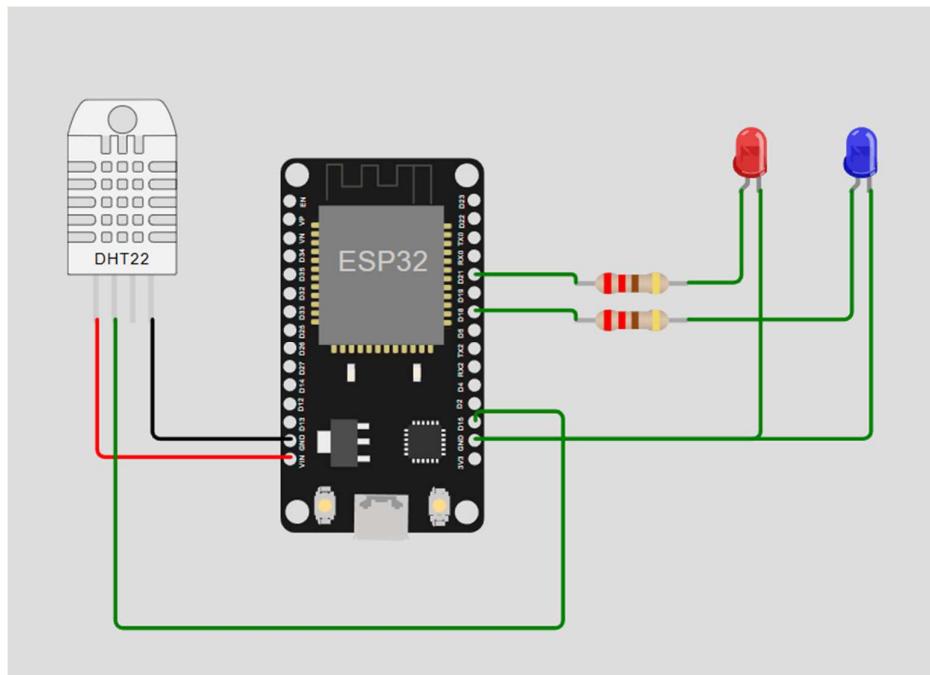
Kết quả

Channel Stats

Created: 12.minutes.ago
Last entry: less.than.a.minute.ago
Entries: 3



Ví dụ : Chương trình liên tục gởi giá trị nhiệt độ, độ ẩm từ cảm biến DHT 22 lên thingspeak, nếu nhiệt độ lớn hơn 50 °C thì bật đèn đỏ, độ ẩm lớn hơn 70 % thì bật đèn xanh.



CODE mẫu trên wokwi :

```
#include <WiFi.h>          // thêm thư viện WIFI cho kết nối mạng
#include "DHT.h"            // thêm thư viện cho cảm biến DHT
#include "ThingSpeak.h"      // thêm thư viện cho việc gửi dữ liệu lên ThingSpeak
#define DHTTYPE DHT22        // định nghĩa sử dụng cảm biến DHT22
#define DHTPIN 15             // định nghĩa chân kết nối cảm biến 15
DHT dht(DHTPIN,DHTTYPE); // tạo dht từ thư viện DHT.h để truy cập đọc nhiệt
độ và độ ẩm từ cảm biến DHT22
//định nghĩa các chân GPIO cho các led
const int LED_RED = 21;
const int LED_BLUE = 18;

const char* WIFI_NAME = "Wokwi-GUEST"; // WiFi SSID
const char* WIFI_PASSWORD = ""; // WiFi Password
const int myChannelNumber = 2786694; // số kênh ThingSpeak liên kết
const char* myApiKey = "HBT5KG2YOP0EK239"; // ThingSpeak API key
const char* server = "api.thingspeak.com"; // Địa chỉ máy chủ ThingSpeak

WiFiClient client; // tạo client wifi
void setup() {
    Serial.begin(115200); // Khởi tạo giao tiếp Serial với tốc độ 115200bps
    //thiết lập chân LED_RED và LED_BLUE là OUTPUT
    pinMode(LED_RED, OUTPUT);
    pinMode(LED_BLUE, OUTPUT);
}
```

```

pinMode(LED_BLUE, OUTPUT);
dht.begin(); //khởi động cảm biến DHT
WiFi.begin(WIFI_NAME, WIFI_PASSWORD); //Kết nối với mạng WIFI
while (WiFi.status() != WL_CONNECTED) // chờ đến khi kết nối WIFI thành
công
{
    delay(1000);
    Serial.println("Wifi not connected"); // in ra thông báo không kết nối
WIFI được
}
Serial.println("Wifi connected !"); //in ra thông báo WIFI đã được kết
nối
Serial.println("Local IP: " + String(WiFi.localIP())); // in ra địa chỉ IP
WiFi.mode(WIFI_STA); // đặt chế độ WIFI thành Station Mode - kết nối với một
điểm truy cập (AP) đã xác định trước
ThingSpeak.begin(client); // khởi động ThingSpeak
}

void loop()
{
    //đọc dữ liệu từ cảm biến DHT22
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();

    // điều khiển bật led đỏ cảnh báo khi nhiệt độ >50 hoặc <-20 và led xanh khi
    độ ẩm >70
    if(temperature >50 || temperature<-20)
        digitalWrite(LED_RED,1);
    else digitalWrite(LED_RED,0);
    if(humidity >70)
        digitalWrite(LED_BLUE,1);
    else digitalWrite(LED_BLUE,0);
    // gửi dữ liệu lên ThingSpeak
    ThingSpeak.setField(1,temperature); // field 1 trong dữ liệu được gửi lên
    ThingSpeak là nhiệt độ
    ThingSpeak.setField(2,humidity); // field 2 trong dữ liệu được gửi lên
    ThingSpeak là độ ẩm
    // gửi dữ liệu lên ThingSpeak và kết quả lưu trữ trong biến status
    int status = ThingSpeak.writeFields(myChannelNumber,myApiKey);

    if(status == 200) // trong giao thức HTTP 200 là mã trạng thái thành
    công,nếu gửi dữ liệu thành công lên ThingSpeak
    {
        Serial.println("Data pushed successfully"); // in ra thông báo "Data
        pushed successfully" đã gửi thành công lên ThingSpeak
    }
    else // nếu có lỗi trong quá trình gửi
    {
}

```

```
    Serial.println("Push error" + String(status)); // in ra thông báo "Push  
error" cùng với mã trạng thái nhận được từ máy chủ ThingSpeak để hiển thị lỗi  
cụ thể  
}  
Serial.println("Temp: " + String(temperature, 2) + "°C"); // in ra giá trị  
nhiệt độ lên Serial Monitor với định dạng 2 chữ số thập phân  
Serial.println("Humidity: " + String(humidity, 2) + "%"); // in ra giá trị  
độ ẩm lên Serial Monitor với định dạng 2 chữ số thập phân  
Serial.println("---"); // in ra --- phân tách các lần đo  
delay(5000); // delay 5s  
}
```

- ❖ Bài tập 6.1 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình liên tục gởi
giá trị nhiệt độ, độ ẩm từ cảm biến DHT 22 lên thingspeak, đồng thời hiển thị
trên màn hình LCD I2C.
- ❖ Bài tập 6.2 : Vẽ mạch mô phỏng / đấu mạch và viết chương trình giám sát vườn
rau. Hệ thống gồm cảm biến DHT 22, 02 relay, chức năng đo nhiệt độ, độ ẩm,
gởi lên thingspeak, nếu nhiệt độ < 20 độ C thì bật relay 1, nếu độ ẩm < 50 % thì
bật relay 2.

DK SON

TÀI LIỆU THAM KHẢO

- [1] https://docs.wokwi.com/?utm_source=wokwi
- [2] <https://cloud.arduino.cc/>
- [3] <https://blynk.io/>
- [4] <https://thingspeak.mathworks.com/>
- [5] <https://khuenguyencreator.com/>
- [6] <https://dienthongminhesmart.com/>
- [7] <https://www.iotzone.vn/>
- [8] <https://www.espressif.com/en>
- [9] BK Tripathy, J Anuradha, Internet of Things (IoT) Technologies, Applications, Challenges and Solutions, CRC Press. 2020.
- [10] Bài giảng môn IoT và ứng dụng, TS. Nguyễn Đức Minh, học viện công nghệ Bưu chính viễn thông.
- [11] Bài giảng môn IoT và ứng dụng, Phạm Ngọc Hưng, đại học bách khoa Hà Nội.