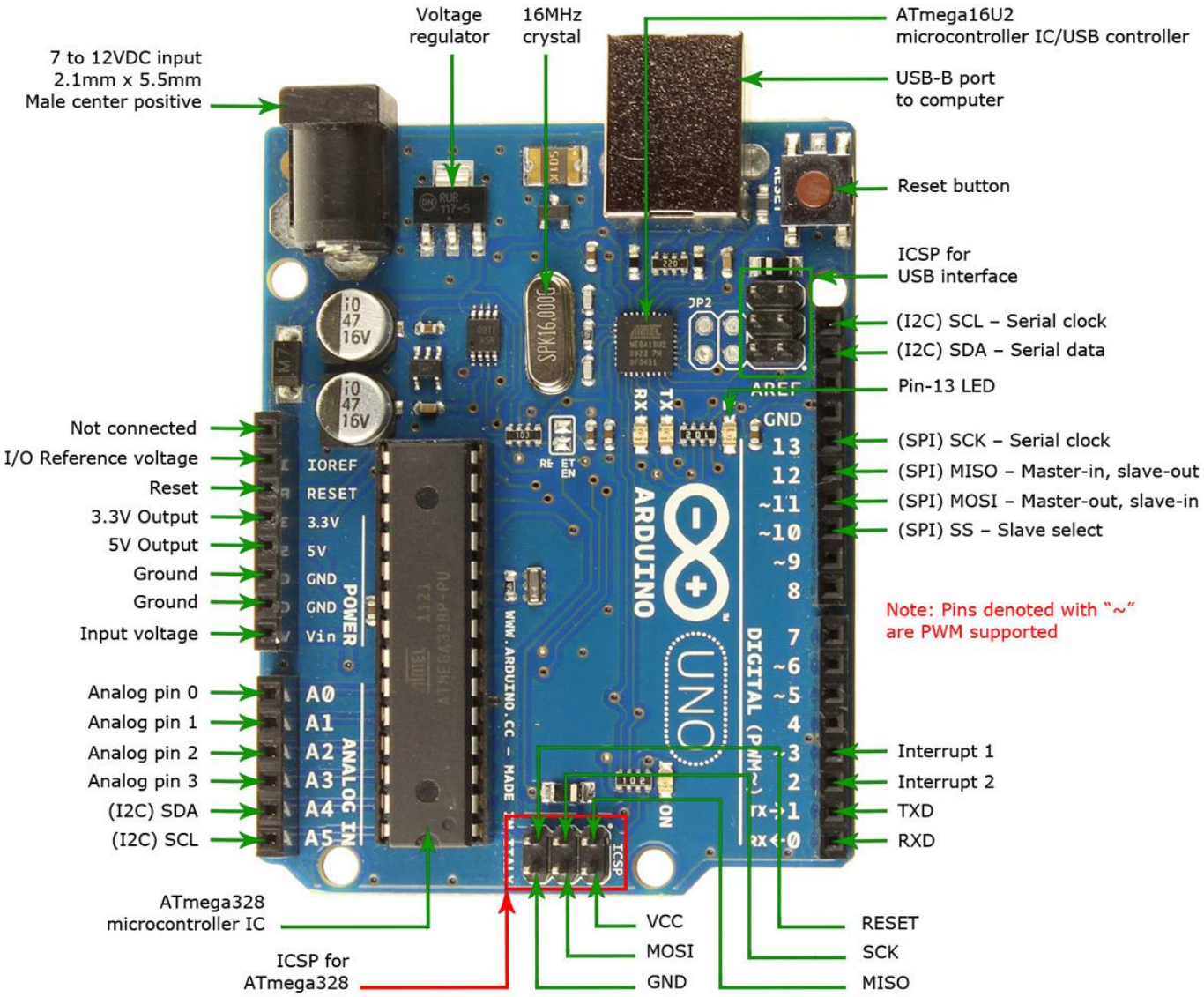
I. Phần Cứng



1. Chip vi điều khiển ATmega328P

Bộ xử lý trung tâm (CPU):

- ✧ Arduino Uno R3 sử dụng vi điều khiển ATmega328P, thuộc dòng AVR của hãng Atmel (hiện thuộc Microchip). Đây là một vi điều khiển 8-bit, có khả năng thực hiện các phép toán logic, xử lý tín hiệu, và điều khiển các thiết bị ngoại vi thông qua các chân I/O.
- ✧ Tốc độ xử lý: Chip hoạt động ở tần số 16 MHz, đủ mạnh để xử lý các dự án từ đơn giản đến phức tạp. Nó có thể thực hiện các tác vụ như đo lường, điều khiển và xử lý tín hiệu từ cảm biến một cách hiệu quả.

Bộ nhớ:

- ✧ Flash: 32 KB, trong đó có 0.5 KB dành cho bootloader. Bộ nhớ Flash lưu trữ chương trình của bạn.
- ✧ SRAM: 2 KB, dùng để lưu trữ các biến trong quá trình chương trình chạy. SRAM tương tự như RAM trên máy tính, nơi lưu trữ dữ liệu tạm thời.
- ✧ EEPROM: 1 KB, dùng để lưu trữ dữ liệu không mất khi tắt nguồn. EEPROM tương tự như ROM hoặc SSD trên máy tính, nơi lưu trữ dữ liệu lâu dài.

Chú ý:

- ✧ Kích thước file: Khi lập trình, cần chú ý đến kích thước file chương trình (sketch) để đảm bảo rằng nó không vượt quá dung lượng bộ nhớ Flash. Nếu chương trình quá lớn, có thể gây lỗi hoặc chương trình không chạy đúng.
- ✧ Độ phức tạp của thuật toán: Phải cân nhắc độ phức tạp của thuật toán và cách sử dụng bộ nhớ để tránh việc vi điều khiển bị quá tải, dẫn đến treo hoặc hoạt động không ổn định.
- ✧ Bạn có thể kiểm tra kích thước của chương trình khi biên dịch bằng Arduino IDE để đảm bảo rằng nó nằm trong giới hạn bộ nhớ của vi điều khiển. Đồng thời, cũng nên tối ưu hóa mã nguồn để giảm sử dụng bộ nhớ và tăng hiệu suất.

2. Nhóm chân chức năng của Arduino Uno R3

2.1 Chân Digital:

-Arduino Uno R3 có tổng cộng 14 chân digital được đánh số từ D0 đến D13. Những chân này có thể hoạt động ở hai chế độ: Input (nhận tín hiệu) và Output (phát tín hiệu). Chúng được sử dụng để điều khiển hoặc nhận tín hiệu từ các thiết bị như đèn LED, nút nhấn, relay, cảm biến, v.v.

-Trạng thái hoạt động:

- ✧ Cao (HIGH): Điện áp 5V.
- ✧ Thấp (LOW): Điện áp 0V.

-Sử dụng:

- ✧ Điều khiển (Output): Các chân digital có thể phát tín hiệu để điều khiển các thiết bị ngoại vi. Sử dụng lệnh digitalWrite(Pin, HIGH); để xuất tín hiệu mức cao (5V) hoặc digitalWrite(Pin, LOW); để xuất tín hiệu mức thấp (0V).
- ✧ Ứng dụng: Điều khiển bật/tắt đèn LED, relay, hoặc điều khiển động cơ qua module relay hoặc transistor.
- ✧ Đọc tín hiệu (Input): Các chân digital có thể nhận tín hiệu từ các thiết bị ngoại vi như nút nhấn hoặc cảm biến. Sử dụng lệnh digitalRead(Pin); để đọc giá trị từ chân digital. Kết quả trả về sẽ là HIGH (5V) hoặc LOW (0V), tùy thuộc vào trạng thái tín hiệu.
- ✧ Ứng dụng: Đọc trạng thái của nút nhấn, công tắc, hoặc các cảm biến số (digital sensor).

Chú ý:

- ✧ Các chân này có thể cấp nguồn cho một số thiết bị như servo (không tải), cảm biến, nhưng nên hạn chế cho các thiết bị có tải nặng vì dòng điện chỉ khoảng 20-40 mA.
- ✧ Trong một số trường hợp, khi đọc tín hiệu từ input, cần kéo trở Pull-Up/Pull-Down để đảm bảo tín hiệu được đọc chính xác. Thường dùng các giá trị trở phổ biến như 1kΩ, 4.7kΩ, 10kΩ.

2.2 Chân Analog:

-Arduino Uno R3 có 6 chân analog được đánh số từ A0 đến A5. Những chân này chủ yếu dùng để đọc tín hiệu tương tự (analog) và chuyển đổi chúng thành tín hiệu số (digital) thông qua bộ chuyển đổi tương tự - số (ADC - Analog to Digital Converter).

-Chuyển đổi tín hiệu: Các chân analog của Arduino Uno có bộ chuyển đổi ADC 10-bit, tức là chúng có thể chia dải điện áp từ 0V đến 5V thành 1024 mức (từ 0 đến 1023).

-Ví dụ: Nếu tín hiệu vào là 2.5V, Arduino sẽ đọc giá trị khoảng 512 (vì 2.5V là giữa dải điện áp 0-5V).

-Cách sử dụng:

- ✧ Đọc tín hiệu: Dùng lệnh analogRead(Pin); để đọc tín hiệu từ chân analog. Kết quả trả về sẽ là giá trị từ 0 đến 1023, tương ứng với điện áp đầu vào.
- ✧ Ứng dụng: Chân analog thường dùng để đọc tín hiệu từ các cảm biến như cảm biến nhiệt độ, ánh sáng, độ ẩm, hoặc bất kỳ thiết bị nào xuất ra tín hiệu điện áp tương tự.

Chú ý: Các chân A1-A5 chỉ dùng để đọc các tín hiệu analog từ cảm biến và không có chức năng điều khiển (Output).

2.3 Các chân PWM:

-Arduino Uno có 6 chân có khả năng tạo tín hiệu PWM (Pulse Width Modulation), đó là các chân 3, 5, 6, 9, 10, 11. PWM là một kỹ thuật điều khiển tín hiệu số bằng cách thay đổi chu kỳ làm việc của xung (tỷ lệ thời gian bật và tắt của tín hiệu).

-Chân PWM: Được đánh dấu bằng ký hiệu dấu ~ (gợn sóng) bên cạnh số chân (ví dụ: ~3, ~5, ~6, ~9, ~10, ~11).

-Chức năng: PWM được sử dụng để mô phỏng tín hiệu tương tự bằng cách thay đổi chu kỳ làm việc của một tín hiệu số. Điều này cho phép Arduino điều khiển các thiết bị như đèn LED, động cơ, loa một cách mềm dẻo.

-Điều chế độ rộng xung: Chu kỳ làm việc của PWM được điều chỉnh với giá trị từ 0 đến 255, tương ứng với 0% đến 100% của chu kỳ làm việc:

- ✧ 0: Tín hiệu luôn ở mức thấp (0V).
- ✧ 255: Tín hiệu luôn ở mức cao (5V).
- ✧ 128: Tín hiệu ở mức cao 50% thời gian và mức thấp 50% thời gian (tương ứng với 2.5V trung bình).

-Cách sử dụng: Để điều khiển chân PWM, dùng lệnh `analogWrite(Pin, value);` trong đó:

- ✧ Pin: là số của chân PWM (3, 5, 6, 9, 10, 11).
- ✧ Value: là giá trị từ 0 đến 255, tương ứng với chu kỳ làm việc.

-Ứng dụng:

- ✧ Làm mờ đèn LED: Bằng cách thay đổi độ sáng từ 0 (tắt) đến 255 (sáng nhất).
- ✧ Điều khiển tốc độ động cơ: PWM được dùng để điều chỉnh mức độ cung cấp điện áp trung bình cho động cơ, giúp điều khiển tốc độ quay.
- ✧ Phát âm thanh: PWM cũng được dùng để tạo ra các tần số âm thanh khác nhau từ loa hoặc buzzer, bằng cách thay đổi tần số của tín hiệu PWM, từ đó tạo ra các âm thanh có cao độ khác nhau.
- ✧

2.4 Các chân đặc biệt trên Arduino Uno R3

2.4.1 UART (Serial):

-RX (D0): Chân nhận dữ liệu (Receive) trong giao tiếp UART (Serial).

-TX (D1): Chân truyền dữ liệu (Transmit) trong giao tiếp UART (Serial).

-Chức năng: Hai chân này được sử dụng để truyền và nhận dữ liệu qua giao tiếp Serial giữa Arduino và các thiết bị khác, hoặc giữa Arduino và máy tính qua cổng USB (giao tiếp UART).

Chú ý: Vì các chân này cũng được dùng cho việc tải chương trình từ máy tính lên Arduino, nên khi sử dụng chúng cho mục đích khác, bạn cần thận trọng để tránh xung đột trong quá trình nạp code.

2.4.2 SPI:

-SS (D10): Chân chọn thiết bị (Slave Select).

-MOSI (D11): Chân dữ liệu từ Master đến Slave (Master Out Slave In).

-MISO (D12): Chân dữ liệu từ Slave đến Master (Master In Slave Out).

-SCK (D13): Chân clock trong giao tiếp SPI.

-Chức năng: Các chân này hỗ trợ giao tiếp SPI (Serial Peripheral Interface), một giao thức truyền thông tốc độ cao để trao đổi dữ liệu giữa Arduino và các thiết bị như cảm biến, bộ nhớ, hoặc các thiết bị điều khiển khác.

Chú ý: Thực tế khi sử dụng SPI sẽ cần thêm các chân GND và DC. Chúng ta có thể kết nối nhiều thiết bị với giao tiếp SPI bằng cách On/Off các chân SS (Slave Select) để chọn thiết bị cụ thể, trong khi các chân còn lại có thể kết nối song song giữa nhiều thiết bị.

2.4.3 I2C:

-SCL (A5): Chân clock trong giao tiếp I2C.

-SDA (A4): Chân dữ liệu trong giao tiếp I2C.

-Chức năng: Hai chân này hỗ trợ giao tiếp I2C, một giao thức truyền thông hai dây để kết nối Arduino với các thiết bị ngoại vi như cảm biến, module màn hình, và bộ nhớ EEPROM. SCL dùng để truyền tín hiệu clock, còn SDA dùng để truyền dữ liệu.

Chú ý: Các thiết bị kết nối qua I2C có một địa chỉ riêng, nên bạn có thể kết nối cùng lúc nhiều thiết bị. Tuy nhiên, tốc độ của I2C không nhanh bằng SPI.

2.4.4 Chân VIN, GND, 3.3V và 5V:

-VIN: Chân này cho phép bạn cung cấp điện áp từ 7-12V cho bo mạch Arduino khi không sử dụng cổng USB để cấp nguồn.

-GND: Các chân GND là chân nối đất (Ground), cần kết nối khi cấp nguồn hoặc giao tiếp với các thiết bị ngoại vi.

-3.3V: Cung cấp điện áp 3.3V cho các thiết bị ngoại vi cần điện áp thấp hơn.

-5V: Cung cấp điện áp 5V cho các thiết bị ngoại vi.

Chú ý:

- ✧ Tuyệt đối hạn chế cấp ngược chân VIN và GND hoặc chập chậm, vì có thể dẫn đến hỏng hóc các linh kiện, đặc biệt là chip USB trên bo mạch (có thể gây tổn thất tài chính).
- ✧ Hạn chế cấp nguồn từ 3.3V hoặc 5V cho các thiết bị có công suất lớn. Nếu cần, nên cấp nguồn ngoài cho các thiết bị và kéo chân GND từ nguồn ngoài vào Arduino để đảm bảo an toàn và ổn định.

3. Các thiết bị ngoại vi

-Hiện nay, thị trường cung cấp nhiều loại thiết bị ngoại vi với chức năng đa dạng. Phần lớn các thiết bị này đã được tích hợp thành các module sẵn có, dễ dàng kết nối và sử dụng với vi điều khiển. Các thiết bị ngoại vi thường được chia thành hai nhóm chính: nhóm đọc tín hiệu và nhóm điều khiển.

-Nhóm đọc tín hiệu: Bao gồm các thiết bị như cảm biến (sensor), công tắc (switch), nút nhấn (button),... được dùng để thu thập dữ liệu từ môi trường xung quanh.

-Nhóm điều khiển: Bao gồm các thiết bị như màn hình TFT/LCD, rơ-le (relay), L298N (điều khiển động cơ DC), nRF24L01 (thu-phát tín hiệu RF),... được dùng để điều khiển các tác vụ hoặc hiển thị thông tin.

3.1 Các Module Đọc Tín Hiệu

-Các module cảm biến thường được chia thành hai loại dựa trên loại tín hiệu mà chúng xuất ra:

-Cảm biến tín hiệu analog:

- ✧ Ví dụ: cảm biến ánh sáng, cảm biến hồng ngoại, cảm biến pH, cảm biến nhiệt độ.
- ✧ Những cảm biến này gửi tín hiệu dạng analog về vi điều khiển (như Arduino). Arduino có thể đọc tín hiệu này và chuyển đổi nó thành các giá trị số từ 0 đến 1023, do nó hỗ trợ độ phân giải 10-bit. Giá trị này thường cần ánh xạ sang các thang đo phù hợp.
- ✧ Ví dụ, với cảm biến nhiệt độ, giá trị từ 0 đến 1023 có thể được ánh xạ thành nhiệt độ thực tế sử dụng các công thức hoặc hàm chuyển đổi.

-Cảm biến tín hiệu digital:

Ví dụ: các công tắc, cảm biến tiếp xúc, cảm biến chuyển động.

Các cảm biến này xuất ra tín hiệu dạng nhị phân (0 hoặc 1), cho phép kiểm tra trạng thái bật hoặc tắt.

3.2 Các Module Điều Khiển

-Các module điều khiển cho phép vi điều khiển tác động lên các hệ thống hoặc hiển thị thông tin ra bên ngoài. Chúng thường được sử dụng để điều khiển các thiết bị như động cơ, màn hình, rơ-le hoặc giao tiếp với các thiết bị khác qua sóng radio, sóng RF, hoặc các giao thức truyền thông khác.

-Module màn hình (LCD, TFT, OLED):

- ✧ LCD: Loại màn hình đơn giản, phổ biến nhất, thường dùng để hiển thị thông tin dạng văn bản (ví dụ: màn hình 16x2 hoặc 20x4).
- ✧ TFT và OLED: Có độ phân giải cao hơn, hiển thị đồ họa và màu sắc, thường dùng trong các ứng dụng yêu cầu hiển thị phức tạp như biểu đồ, hình ảnh, hoặc giao diện đồ họa cho người dùng.
- ✧ Giao tiếp: Các màn hình thường được điều khiển qua các giao thức như I2C, SPI hoặc GPIO và yêu cầu thư viện hỗ trợ để dễ dàng thao tác với vi điều khiển như Arduino hoặc ESP.

-Module rơ-le (Relay):

- ✧ Rơ-le là thiết bị đóng/ngắt mạch điện dựa trên tín hiệu điều khiển từ vi điều khiển, thường được sử dụng để điều khiển các thiết bị điện áp cao hoặc dòng điện lớn mà vi điều khiển không thể trực tiếp xử lý (ví dụ: đèn, quạt, máy bơm nước).

-Module điều khiển động cơ (Motor Driver):

- ✧ Các module điều khiển động cơ, như L298N, cho phép điều khiển động cơ DC, động cơ bước (stepper motor), hoặc động cơ servo.
- ✧ L298N: Module phổ biến cho động cơ DC, có khả năng điều khiển đồng thời hai động cơ với dòng điện tương đối lớn.
- ✧ Servo motor: Được điều khiển bởi tín hiệu PWM (Pulse Width Modulation) và thường dùng trong các ứng dụng yêu cầu điều khiển chính xác vị trí như robot hoặc hệ thống tự động hóa.

-Module giao tiếp không dây (RF, Bluetooth, Wi-Fi):

- ✧ nRF24L01: Module giao tiếp RF cho phép truyền dữ liệu không dây giữa các vi điều khiển với khoảng cách xa và tiêu thụ điện năng thấp, thường dùng trong hệ thống cảm biến không dây hoặc điều khiển từ xa.
- ✧ Bluetooth và Wi-Fi modules: Các module như HC-05 (Bluetooth) cho phép vi điều khiển kết nối với các thiết bị khác hoặc mạng internet, sử dụng cho các ứng dụng IoT (Internet of Things).

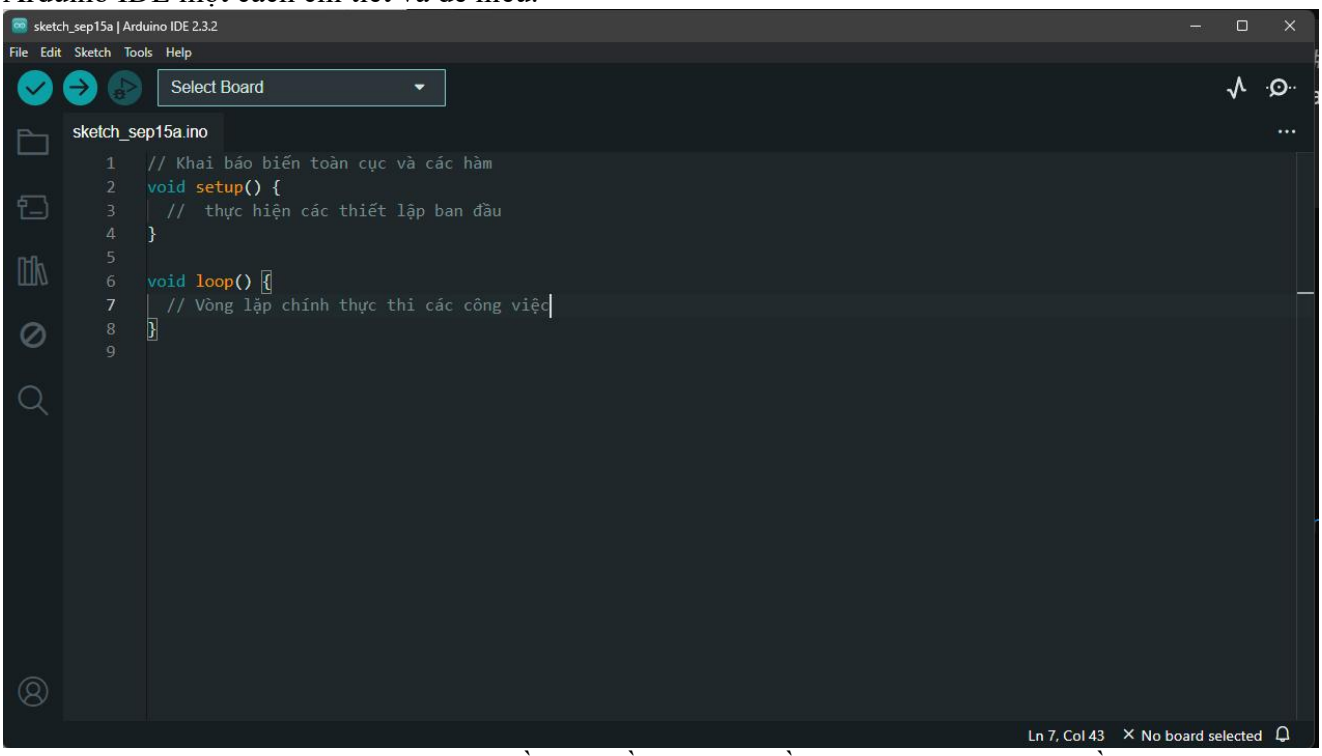
-Module điều khiển LED và dải đèn LED (LED Strip):

- ✧ Các module điều khiển LED như WS2812 hoặc Neopixel cho phép điều khiển nhiều đèn LED với màu sắc và hiệu ứng đa dạng thông qua một chân duy nhất của vi điều khiển, thường dùng trong các ứng dụng chiếu sáng trang trí hoặc tín hiệu.
- ✧ Cách sử dụng các module điều khiển
- ✧ Tất cả các module này yêu cầu vi điều khiển gửi tín hiệu điều khiển tới chúng thông qua các giao thức như GPIO (cho rơ-le, L298N), PWM (cho servo motor), hoặc giao tiếp không dây (cho nRF24L01, Bluetooth).
- ✧ Bạn cũng có thể dùng các giao thức này để giao tiếp giữa các Arduino.
- ✧ Hầu hết các module đều có thư viện hỗ trợ sẵn cho các vi điều khiển như Arduino, ESP32, giúp việc lập trình trở nên dễ dàng hơn. Chỉ cần kết nối phần cứng và sử dụng các hàm điều khiển trong thư viện, bạn có thể nhanh chóng tích hợp các module điều khiển vào dự án của mình.

Chú ý: Các module và cảm biến trên chỉ là một vài loại phổ biến. Cách sử dụng chúng thường tương tự nhau, và nếu không biết cách sử dụng, bạn có thể tham khảo tài liệu từ YouTube và Google.

II. Phần Mềm

Trong phần này, mình sẽ giải thích và hướng dẫn cách viết một file code Arduino bằng phần mềm Arduino IDE một cách chi tiết và dễ hiểu.



Theo mình, một chương trình Arduino gồm 3 phần chính: phần khai báo toàn cục, phần khởi tạo (void setup()), và phần vòng lặp chính (void loop()).

1 Khai báo biến toàn cục

-Phần khai báo biến toàn cục là phần đầu tiên của chương trình Arduino, nơi bạn định nghĩa các biến, hằng số, và thư viện cần sử dụng trong toàn bộ chương trình. Các biến khai báo tại đây có phạm vi toàn cục, nghĩa là chúng có thể được sử dụng ở mọi nơi trong chương trình, cả trong hàm setup() và hàm loop(). Điều này giúp bạn có thể lưu trữ và thay đổi dữ liệu ở nhiều nơi trong chương trình mà không bị giới hạn trong một hàm cụ thể.

-Các hàm xử lý khác cũng có thể viết ở đây và gọi trong các hàm khác như như setup hay loop...

2 Khởi tạo (void setup())

-Hàm void setup() là một trong hai hàm bắt buộc trong bất kỳ chương trình Arduino nào. Hàm này được gọi chỉ một lần khi chương trình bắt đầu chạy hoặc khi Arduino được khởi động lại (reset). Mục đích của hàm setup() là để thực hiện các thiết lập ban đầu cho phần cứng và các biến cần thiết trước khi chương trình bắt đầu thực thi vòng lặp chính trong hàm loop().

3Vòng lặp chính (void loop())

-Hàm void loop() là phần quan trọng nhất của một chương trình Arduino. Đây là nơi chứa các lệnh mà Arduino thực thi liên tục theo thứ tự từ trên xuống dưới và lặp lại mãi mãi, cho đến khi thiết bị bị tắt hoặc được reset. Tất cả các tác vụ chính, như điều khiển thiết bị đầu ra hoặc đọc dữ liệu từ cảm biến, sẽ diễn ra trong vòng lặp này.

Lời kết:

Đây là bài viết đầu tiên của mình về Arduino, vì vậy có thể còn nhiều điểm thiếu sót. Rất mong nhận được sự thông cảm và góp ý từ các bạn để mình có thể cải thiện và bổ sung thêm kiến thức, giúp những người đọc sau có trải nghiệm tốt hơn. Mục tiêu của mình là phát triển cộng đồng Arduino nói chung và IoT nói riêng. Nếu có bất kỳ ý kiến đóng góp nào, các bạn có thể liên hệ với mình qua Facebook cá nhân của mình.

Cảm ơn các bạn đã đọc bài viết!