Ngày 2: Làm quen với lập trình Arduino trên module ESP8266 – ESP32

Nội dung chính

- Giới thiệu về IoT, các ứng dụng của IoT trong các lĩnh vực khác nhau, đặc biệt SmartHome, SmartFactory...
- Giới thiệu về bộ kit ESP (Arduino)
- Cài đặt phần mềm
- Lập trình cơ bản trên phần mềm Arduino sử dụng ngôn ngữ C/C++
- Lập trình làm việc với GPIO, nút nhấn, ngắt… trên bộ kit ESP
- Tìm hiểu về các chuẩn truyền tải dữ liệu UART, I2C, SPI…
- Lập trình làm việc với cảm biến trên bộ kit ESP
- Lập trình kết hợp cảm biến và GPIO cho bài toán cảnh báo (cháy, bụi, ánh sáng…)

1

Giới thiệu về IoT

1. Giới thiệu về IoT

Định nghĩa IoT

IoT là một mạng lưới kết nối ba yếu tố môi trường riêng biệt (Con người, Đồ vật và Dịch vụ) thông qua internet và không cần con người can thiệp, và kết hợp một hệ thông minh gồm cảm biến, kết nối mạng và xử lý.

Cùng với IoT, Đám mây, Big Data và Trí tuệ nhân tạo sẽ tạo ra giá trị và dịch vụ mới

Chuyển hóa thành dịch vụ

Thiết bị được kết nối
(thiết bị được kết nối với internet)

+

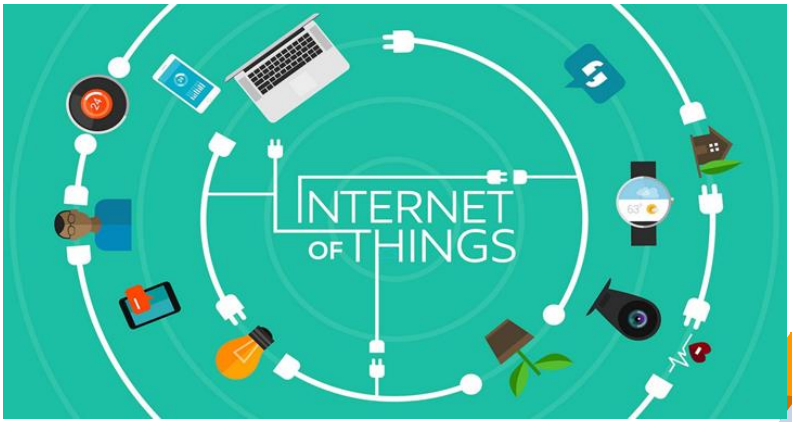
Trí thông minh
(Cảm biến, Kết nối mạng, Xử lý)

(Tạo ra giá trị mới)

1. Giới thiệu về IoT

IoT là gì?

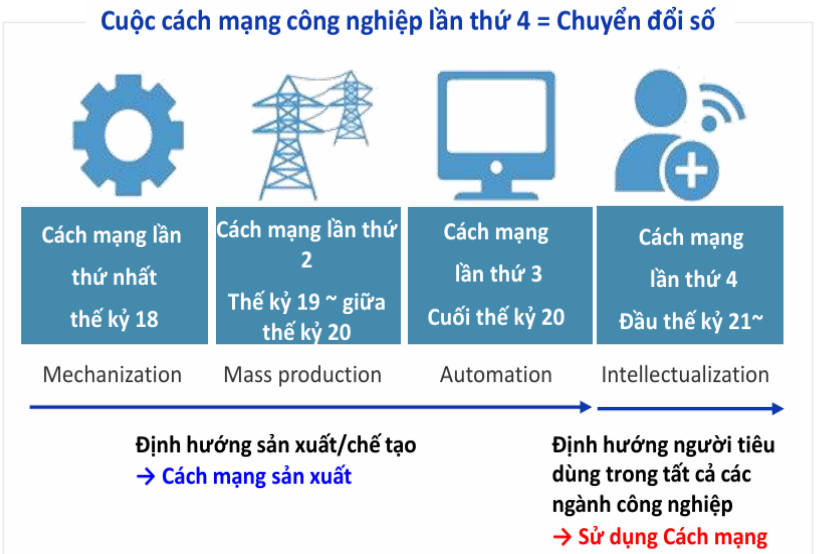
– IoT (tiếng Anh: Internet of Things) là là một mạng lưới mà trong đó, mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính.



1. Giới thiệu về IoT

Cuộc cách mạng công nghiệp lần thứ 4 và sự thay đổi mô hình

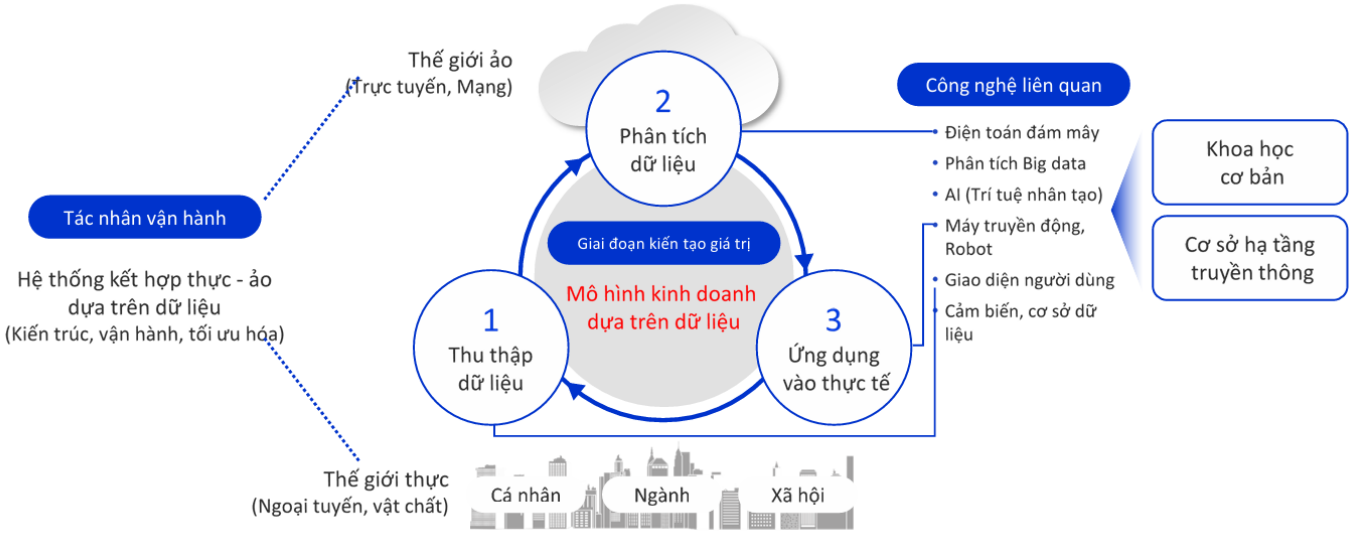
Sự hội tụ của các công nghệ thông tin và truyền thông tiên tiến như Trí tuệ nhân tạo, Internet vạn vật (IoT), Dữ liệu lớn, Di động vào nền kinh tế và xã hội để mang lại sự đổi mới



- **Cuộc cách mạng công nghiệp lần thứ nhất** là cuộc cách mạng cơ khí hóa do động cơ hơi nước gây ra;
- **Cuộc cách mạng công nghiệp lần thứ 2** cho phép sản xuất hàng loạt hàng hóa sản xuất dựa trên năng lượng điện;
- **Cuộc cách mạng công nghiệp lần thứ 3** được đặc trưng bởi sự lan rộng của tự động hóa dựa trên máy tính; và
- **Cách mạng công nghiệp lần thứ 4** là cuộc cách mạng sử dụng dựa trên công nghệ số như IoT, AI, Big Data kết hợp với các ngành công nghiệp khác.
- Cuộc cách mạng lần thứ 4 đang tạo ra sự thay đổi lớn đối với xã hội chúng ta so với các cuộc cách mạng trước đó về phạm vi, tốc độ và hiệu ứng lan tỏa.

1. Giới thiệu về IoT

Tác nhân vận hành, Kiến tạo giá trị và Công nghệ



1. Giới thiệu về IoT

Công nghệ M2M (Kết nối máy móc)

Bốn bước của ứng dụng IoT

- Các sản phẩm thông minh, được kết nối cho phép các danh mục khả năng mới, với mỗi công trình được xây dựng trên các lớp trước đó

Phân loại	RFID/USN/M2M		IoT
Viễn thông/Mạng	Mạng cục bộ, lấy mạng di động làm trung tâm	→	Lấy Internet làm trung tâm
Loại thiết bị	Lấy cảm biến làm trung tâm	→	Sự vật ảo bao gồm sự vật vật chất của cảm biến và máy truyền động, dữ liệu và quy trình
Mức hoạt động của thiết bị	Thu thập thông tin đơn giản/Bị động	→	Sở hữu thông minh tự quyết/tự chủ
Nền tảng dịch vụ	Giám sát xử lý thông tin	→	Giám sát và kiểm soát tự chủ dựa trên ngữ nghĩa
Quy mô quản lý dịch vụ	Hàng chục triệu đối tượng	→	Hàng trăm tỷ đối tượng
Khả năng thích ứng dịch vụ	Dịch vụ tạm thời	→	Cung cấp dịch vụ thông minh tức thì

1. Giới thiệu về IoT

RFID (Radio Frequency Identification) là gì?

RFID = **Nhận dạng bằng tần số vô tuyến**. Đây là công nghệ dùng **sóng radio** để **nhận dạng và trao đổi dữ liệu** giữa một **thẻ (tag)** gắn trên vật cần quản lý và một **thiết bị đọc (reader)**.

1. Cấu tạo cơ bản của hệ thống RFID

- **Thẻ RFID (RFID tag)**: gắn trên sản phẩm, thẻ nhân viên, động vật... Thẻ này chứa **chip** và **ăng-ten** để lưu trữ và truyền dữ liệu.
- **Bộ đọc RFID (RFID reader)**: phát sóng radio để kích hoạt thẻ và nhận dữ liệu từ thẻ.
- **Hệ thống xử lý dữ liệu**: kết nối với máy tính hoặc đám mây để quản lý thông tin.

2. Cách hoạt động

1. Reader phát sóng radio.
2. Thẻ RFID trong vùng phủ sóng sẽ phản hồi lại bằng cách truyền dữ liệu lưu trên chip (ID, thông tin sản phẩm...).
3. Reader thu tín hiệu và gửi dữ liệu về hệ thống quản lý.

3. Ứng dụng thực tế

- **Siêu thị / kho hàng**: quản lý hàng hóa nhanh chóng, không cần quét mã vạch từng sản phẩm.
- **Thẻ nhân viên / thẻ ra vào**: mở cửa tự động khi đưa thẻ gần máy quét.
- **Thu phí đường bộ tự động (ETC)**: xe gắn thẻ RFID, trạm đọc nhận diện và trừ tiền.
- **Theo dõi động vật**: gắn thẻ RFID để quản lý gia súc, thú cưng.
- **Y tế**: quản lý thuốc, thiết bị y tế, bệnh nhân.

1. Giới thiệu về IoT

USN (Ubiquitous Sensor Network)

- USN = **Mạng cảm biến phổ cập / ở khắp mọi nơi**.
- Là hệ thống **kết nối rất nhiều cảm biến** được triển khai trong môi trường, thu thập thông tin (nhiệt độ, độ ẩm, ánh sáng, chuyển động, vị trí…) rồi truyền dữ liệu về trung tâm xử lý.
- Ý tưởng: bất cứ đâu, bất cứ lúc nào, các cảm biến đều có thể **giám sát, thu thập dữ liệu và kết nối qua mạng**.

1. Thành phần cơ bản

- **Nút cảm biến (Sensor Node)**: gồm cảm biến + bộ xử lý nhỏ + module truyền thông (RF, ZigBee, LoRa, Wi-Fi…).
- **Gateway (cổng)**: thu thập dữ liệu từ nhiều sensor node và gửi lên server hoặc cloud.
- **Hệ thống xử lý trung tâm**: phân tích, lưu trữ, hiển thị dữ liệu cho người dùng.

1. Giới thiệu về IoT

2. Đặc điểm

- **Quy mô lớn**: nhiều nút cảm biến triển khai rộng khắp.
- **Tự động**: cảm biến tự gửi dữ liệu, không cần con người thao tác nhiều.
- **Kết nối liên tục**: dữ liệu được cập nhật thời gian thực.
- **Là nền tảng của IoT**: USN chính là hạ tầng quan trọng để triển khai Internet of Things.

3. Ứng dụng thực tế của USN

- **Nông nghiệp thông minh**: cảm biến đất, nhiệt độ, độ ẩm giúp điều khiển tưới tiêu tự động.
- **Thành phố thông minh (Smart City)**: giám sát giao thông, chiếu sáng đường phố, chất lượng không khí.
- **Quân sự**: hệ thống giám sát chiến trường bằng mạng cảm biến.
- **Môi trường**: giám sát mưa, lũ, cháy rừng.

1. Giới thiệu về IoT

M2M (Machine to Machine) là công nghệ cho phép **các thiết bị, máy móc, cảm biến… giao tiếp và trao đổi dữ liệu trực tiếp với nhau** mà **không cần sự can thiệp thường xuyên của con người**.

- "Machine to Machine" = máy với máy.
- M2M giúp thiết bị **tự động thu thập dữ liệu, truyền đi và phản hồi** qua mạng (có thể là **Wi-Fi, 4G/5G, ZigBee, LoRa, Bluetooth, …**).

- **RFID**: nhận dạng & theo dõi bằng sóng radio (đối tượng riêng lẻ).
- **M2M**: máy móc giao tiếp trực tiếp với nhau.
- **USN**: mạng cảm biến ở khắp nơi, tự động thu thập & chia sẻ dữ liệu.

1. Giới thiệu về IoT

- **Ứng dụng của IoT**
  - ➢ Nhà thông minh (Smart Home)

1. Giới thiệu về IoT

Ứng dụng của IoT

- Nhà thông minh (Smart Home)

- Hệ thống chiếu sáng
- Các thiết bị gia dụng
- Giải trí
- An ninh

1. Giới thiệu về IoT

Ứng dụng của IoT

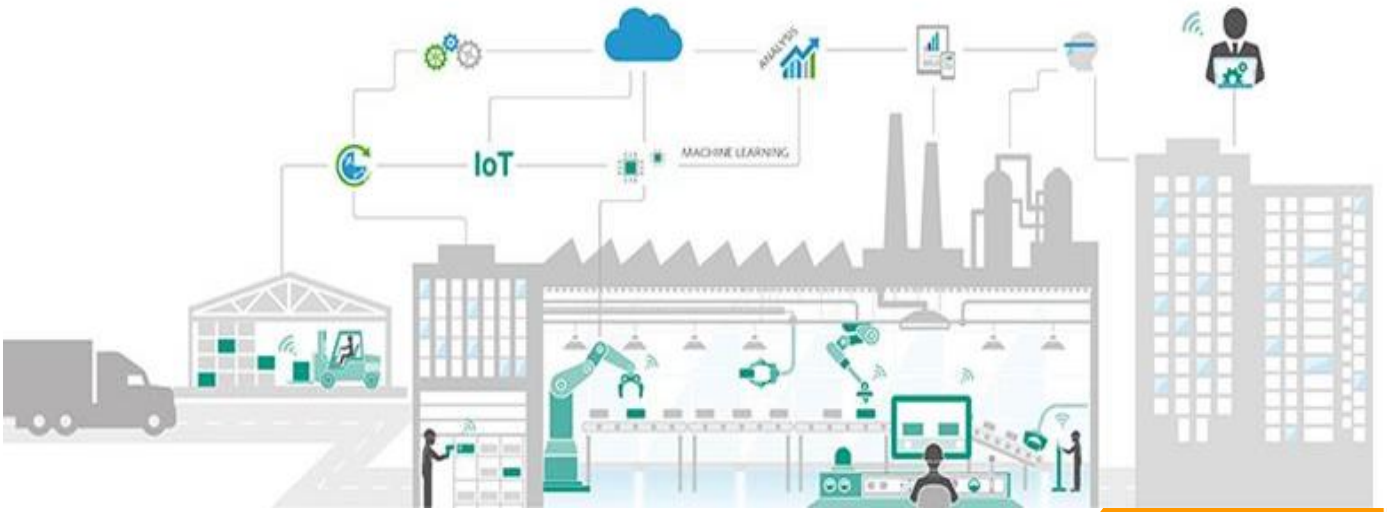
- Nông nghiệp thông minh (Smart Farm)

Áp dụng các hệ thống cảm biến, theo dõi, tự động hoá vào các quá trình từ chuẩn bị, nuôi dưỡng đến phân phối

1. Giới thiệu về IoT

Ứng dụng của IoT

- Công nghiệp thông minh, nhà máy thông minh



2

Tổng quan về Arduino

2. Tổng quan về Arduino

Arduino là gì?

- Arduino là một nền tảng gồm có phần cứng là các board mạch vi điều khiển và phần mềm là hệ thống IDE, thư viện
- Có 2 thành phần chính để phân biệt giữa các board Arduino với nhau: Vi điều khiển và Board mạch
- Vi điều khiển là một máy tính nhỏ, với CPU, RAM, ROM, các thiết bị ngoại vi (I/O), timers, counters,…tất cả được nhúng vào trong một mạch tích hợp (Intergraed Circuit/IC) - nơi tất cả các khối này được kết hợp vào trong một board thông qua hệ thống bus
- Một board arduino là một mạch điện tử hoàn chỉnh, với trái tim là một vi điều khiển AVR, board sẽ có nhiệm vụ cung cấp các điều kiện cần thiết để vi điều khiển có thể hoạt động.

2. Tổng quan về Arduino

Arduino là gì?

Arduino là một **nền tảng mã nguồn mở** bao gồm:

•**Phần cứng**: Các board mạch vi điều khiển (ví dụ: Arduino Uno, Nano, Mega, hoặc các module tích hợp như ESP8266, ESP32 có thể lập trình bằng Arduino).

•**Phần mềm**: Arduino IDE – một môi trường lập trình tích hợp (Integrated Development Environment) hỗ trợ viết code bằng ngôn ngữ **C/C++** và nạp vào board.

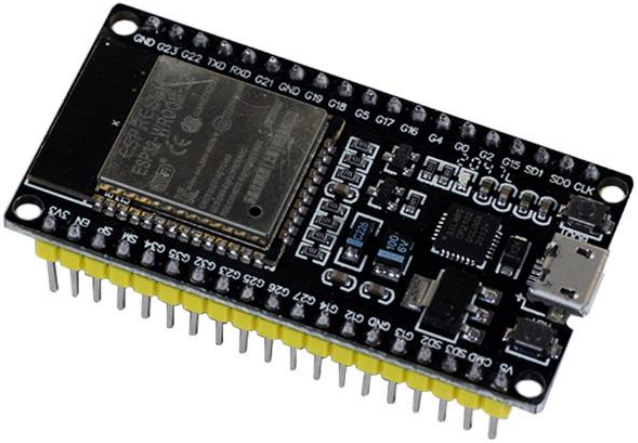
•Mỗi board Arduino là một **mạch điện tử hoàn chỉnh**: ngoài vi điều khiển, còn có cổng USB, nguồn, thạch anh dao động, các chân kết nối, giúp người dùng dễ dàng nạp chương trình và kết nối với cảm biến, đèn LED, động cơ, hay các module khác.

2. Tổng quan về Arduino

Arduino phổ biến

- **Dễ học, dễ dùng**: Cộng đồng đông đảo, có sẵn rất nhiều ví dụ và thư viện hỗ trợ.
- **Giá rẻ, đa dạng**: Có nhiều dòng board, từ cơ bản (Uno, Nano) đến mạnh mẽ (Mega, ESP32).
- **Ứng dụng rộng**: Từ **mô hình giáo dục** (học lập trình, IoT) đến **thực tế** (nhà thông minh, robot, điều khiển thiết bị công nghiệp nhỏ).
- Khi lập trình Arduino, chương trình **luôn có 2 hàm chính**:
  - setup(): chạy **một lần duy nhất** khi khởi động → dùng để khai báo.
  - loop(): chạy **lặp đi lặp lại** → dùng để xử lý liên tục (như nhấp nháy LED, đọc cảm biến).

2. Tổng quan về Arduino



Module Wifi BLE ESP32 Node MCU LuaNode32 CP2102

☆☆☆☆☆ Đánh giá sản phẩm

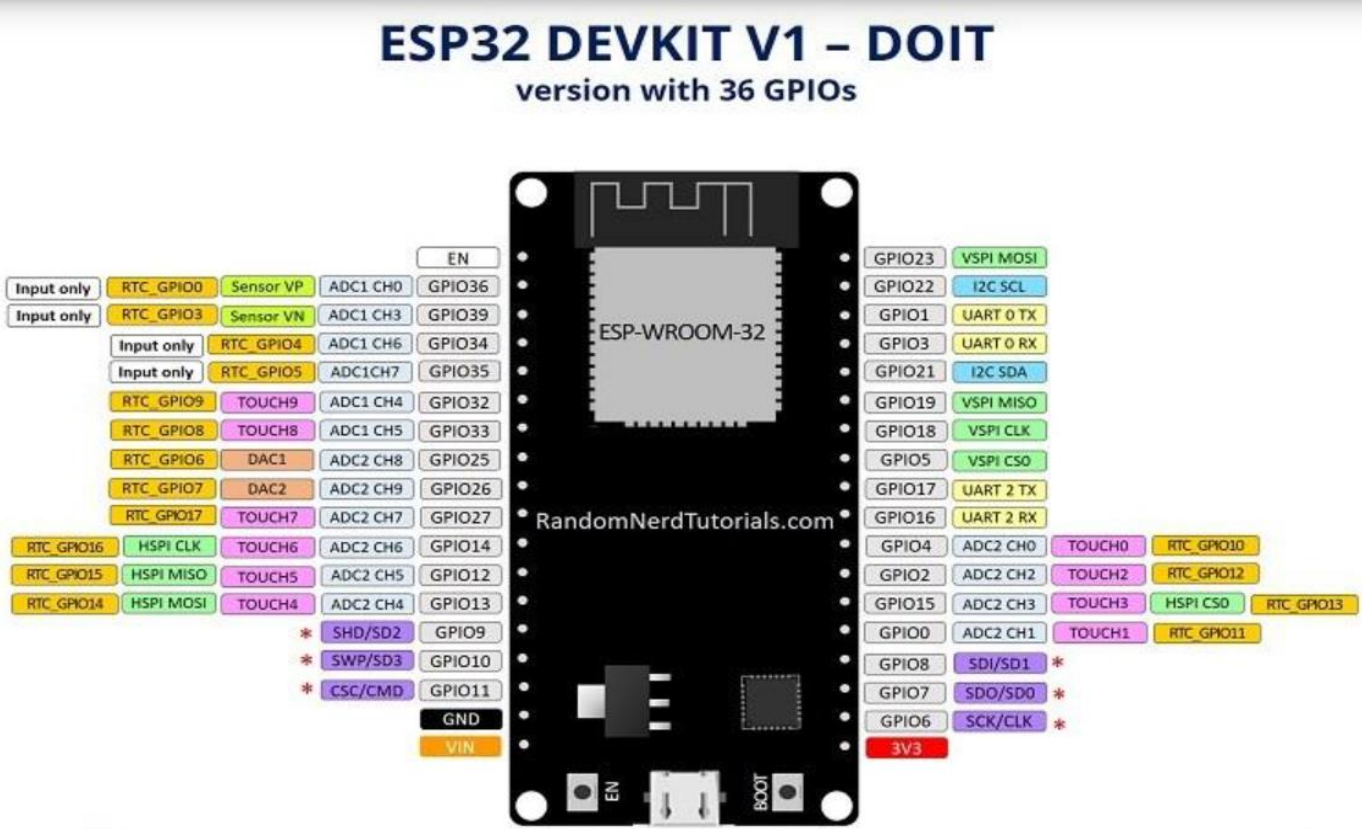
119.000₫ ~~300.000₫~~

Mã sản phẩm: PVX82512

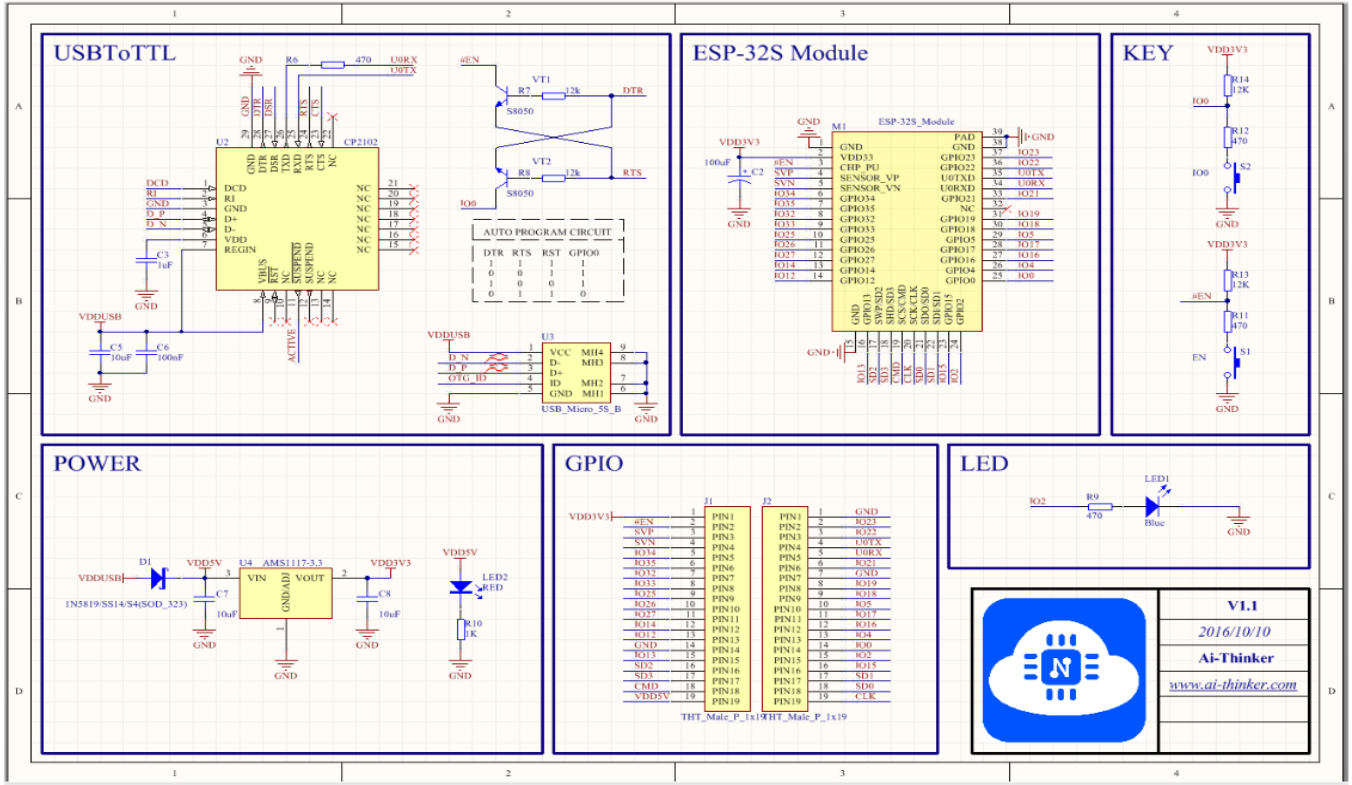
Tags: BLE , esp32 , ESP32 , esp32 module , esp32 wifi , mua esp32 , WIFI

- Về vi xử lí: Xtensa lõi kép 32-bit LX6, tốc độ hoạt động là 240MHz
- Bộ nhớ (SRAM): 512 KB
- Kết nối không dây:
  - Chuẩn Wifi 802. 11 b/g/n
  - Bluetooth (v4.2 BR/EDR và BLE)

2. Tổng quan về Arduino

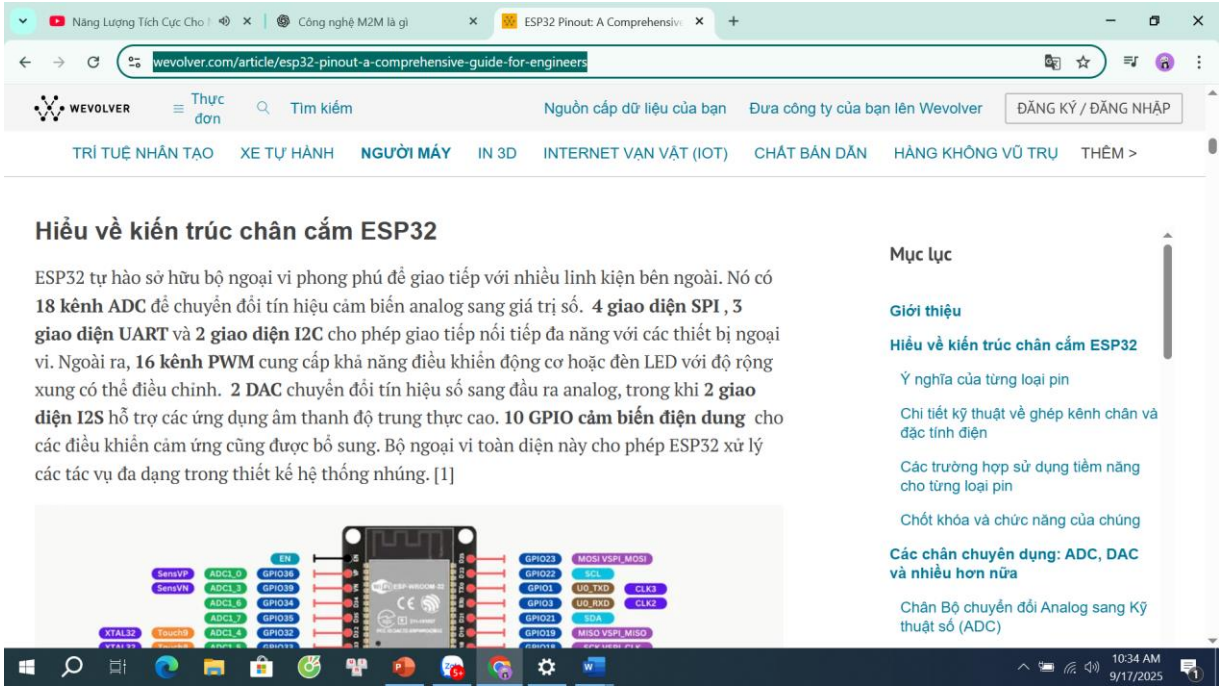


2. Tổng quan về Arduino



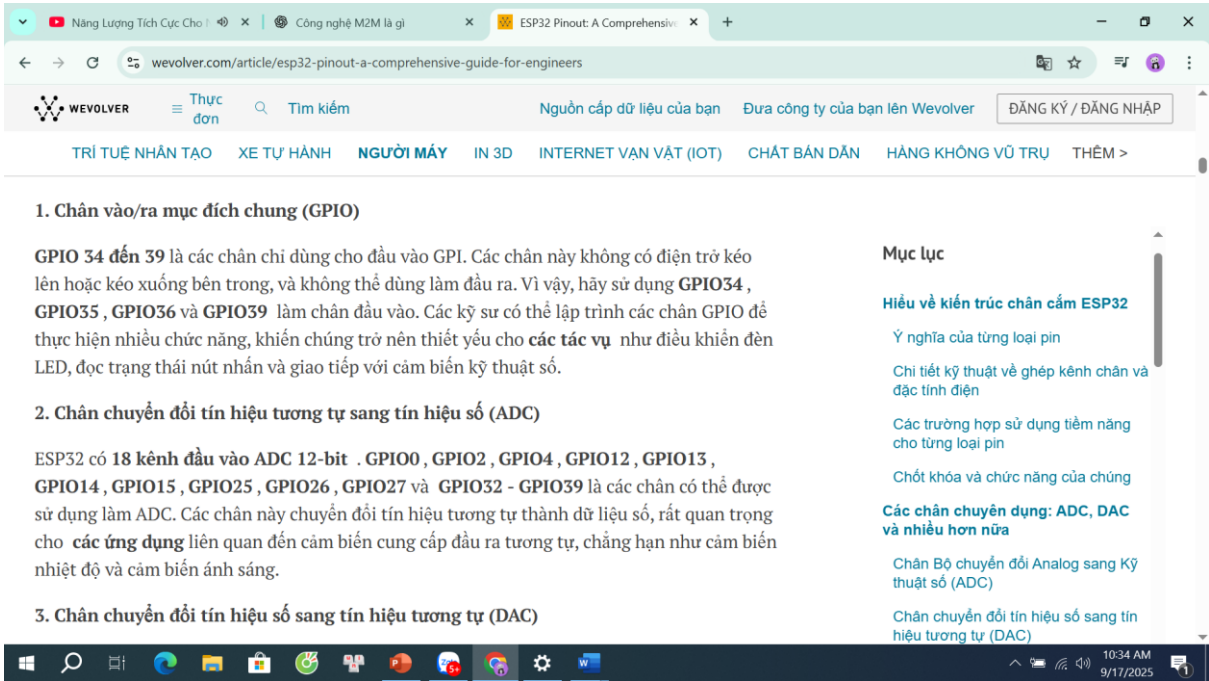
2. Tổng quan về Arduino

https://www.wevolver.com/article/esp32-pinout-a-comprehensive-guide-for-engineers



2. Tổng quan về Arduino

https://www.wevolver.com/article/esp32-pinout-a-comprehensive-guide-for-engineers



2. Tổng quan về Arduino

1. Các loại ESP32 phổ biến

ESP32 được chia thành **chip** và **module/board** (được tích hợp sẵn để dễ dùng):

a) Chip ESP32 chính

- **ESP32**: Phiên bản cơ bản, 2 nhân Xtensa LX6, xung nhịp đến 240 MHz, tích hợp Wi-Fi + Bluetooth.
- **ESP32-S2**: Tối ưu cho bảo mật, USB OTG, chỉ có Wi-Fi (không có Bluetooth).
- **ESP32-S3**: Hỗ trợ AI (vector instructions), USB OTG, Wi-Fi + BLE 5.0.
- **ESP32-C3**: Nhân RISC-V 32-bit, tiết kiệm năng lượng, hỗ trợ Wi-Fi + BLE 5.0.
- **ESP32-C6**: Bản nâng cấp C3, có **Wi-Fi 6**, BLE 5.2.
- **ESP32-H2**: Tập trung cho IoT Mesh, hỗ trợ **Thread / Zigbee**, không có Wi-Fi.

2. Tổng quan về Arduino

b) Module ESP32 (dạng tích hợp Wi-Fi/Bluetooth + anten + flash)

Một số module tiêu biểu:

- **ESP-WROOM-32**: Module phổ biến nhất, tích hợp chip ESP32, flash 4MB.
- **ESP32-WROVER**: Có thêm RAM PSRAM, phù hợp cho AI, camera.
- **ESP32-S2/S3-WROOM, ESP32-C3-WROOM**: Module dựa trên chip thế hệ mới.

c) Board phát triển (Development Boards)

Các hãng đã thiết kế bo mạch tiện lợi:

- **ESP32 DevKit V1** (Doit, NodeMCU-32S) → phổ biến nhất.
- **ESP32-WROVER-KIT** → phục vụ AI, camera, RAM lớn.
- **ESP32-CAM** → tích hợp camera OV2640, dùng cho giám sát.

2. Tổng quan về Arduino

2. Tính năng nổi bật

- **Kết nối mạnh mẽ**: Wi-Fi 2.4 GHz (802.11 b/g/n), Bluetooth Classic + BLE 4.2/5.0/5.2.
- **Xử lý mạnh**: 2 nhân (dual-core), xung nhịp 160–240 MHz.
- **Bộ nhớ**: SRAM 320 KB, Flash ngoài đến 16MB, PSRAM đến 8MB (tùy model).
- **GPIO phong phú**: 30+ chân, hỗ trợ ADC (12-bit), DAC, I2C, SPI, UART, PWM, CAN bus, Ethernet MAC.
- **AI & bảo mật** (trên S3, C3, C6): Hỗ trợ AI accelerator, mã hóa AES, RSA, SHA, secure boot.
- **Tiết kiệm năng lượng**: Có nhiều chế độ ngủ (Deep Sleep, Light Sleep).

2. Tổng quan về Arduino

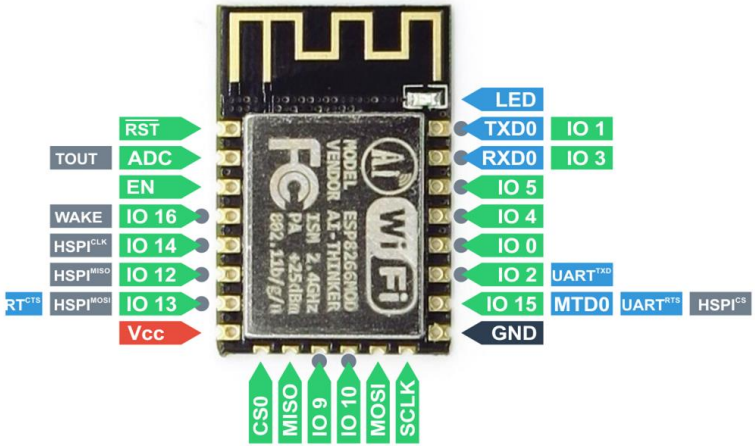
🔷 3. Ứng dụng của ESP32

ESP32 được dùng rộng rãi trong **IoT (Internet of Things)** và các ứng dụng nhúng:

1. **Nhà thông minh (Smart Home)**
   - Điều khiển đèn, quạt, ổ cắm qua Wi-Fi/Bluetooth.
   - Hệ thống an ninh: ESP32-CAM giám sát bằng camera.
2. **Công nghiệp (IIoT)**
   - Giám sát máy móc, cảm biến nhiệt độ/áp suất.
   - Gửi dữ liệu về server hoặc MQTT broker.
3. **Thiết bị đeo (Wearable, Health Monitoring)**
   - ESP32-S3/C3 với BLE dùng cho smartwatch, đo nhịp tim.
4. **Robot – Drone**
   - Kết hợp cảm biến, camera, truyền dữ liệu thời gian thực.
5. **Giáo dục – Nghiên cứu**
   - Board rẻ, dễ lập trình bằng **Arduino IDE**, **MicroPython**, **ESP-IDF**.
6. **Ứng dụng đặc biệt**
   - ESP32-H2 cho mạng Zigbee/Thread → IoT mesh.
   - ESP32-C6 với Wi-Fi 6 → tốc độ cao, tiết kiệm điện.

2. Tổng quan về Arduino

Phần cứng

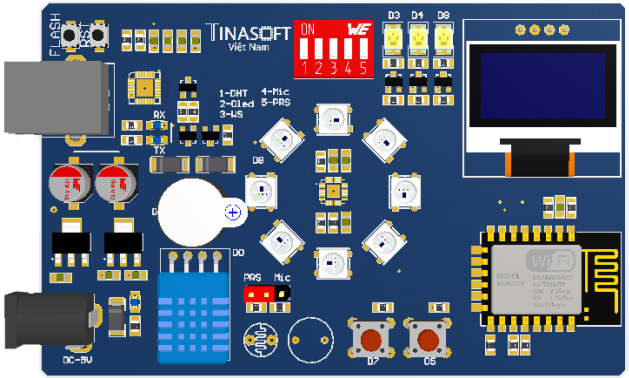


- MCU: ESP8266EX, Wifi: 2.4GHz, 802.11 b/g/n
- Chế độ hoạt động: AP, STA và (AP + STA)
- Kích thước: 16mm x 24mm
- Số chân I/O: 11 (tất cả các chân I/O đều có Interrupt/PWM/I2C/One-wire, trừ chân D0)

2. Tổng quan về Arduino

Bộ kit LHV.IoT (Tina soff)

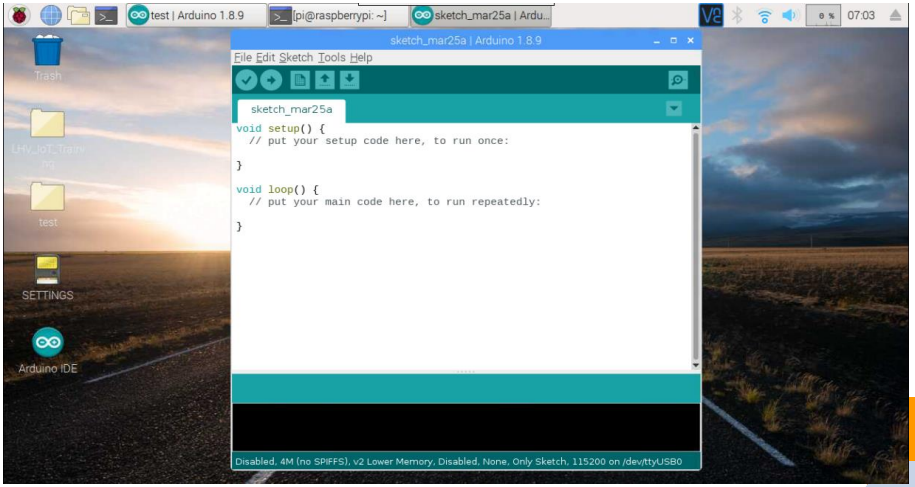
- Module ESP8266
- Màn hình OLED
- Cảm biến nhiệt độ, độ ẩm
- Quang trở
- Còi
- LED RGB
- Cảm biến gia tốc
- LED
- Mic
- Nút nhấn



2. Tổng quan về Arduino

Phần mềm

- Ngôn ngữ lập trình chính của Arduino là C/C++
- Người lập trình được cung cấp vô số các thư viện
- Việc lập trình sẽ được thực hiện trên một công cụ được gọi là IDE (Intergrated Development Environment). Công cụ này được đội ngũ kỹ sư của Arduino xây dựng và phát triển.



3

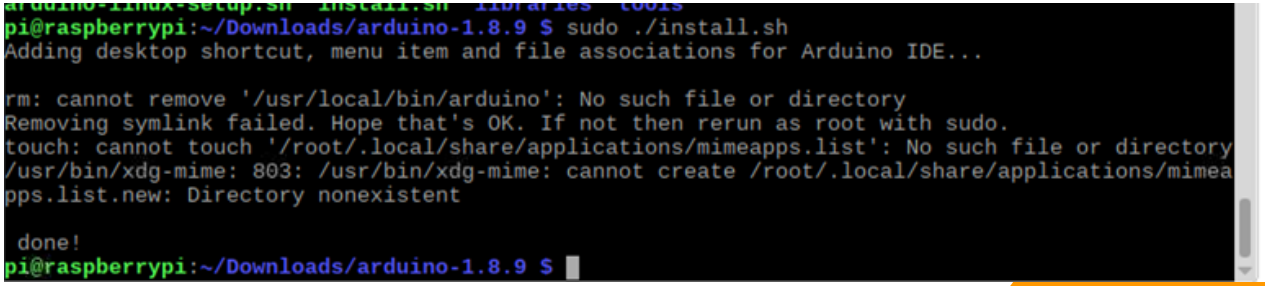
Cài đặt Arduino IDE và setup môi trường lập trình

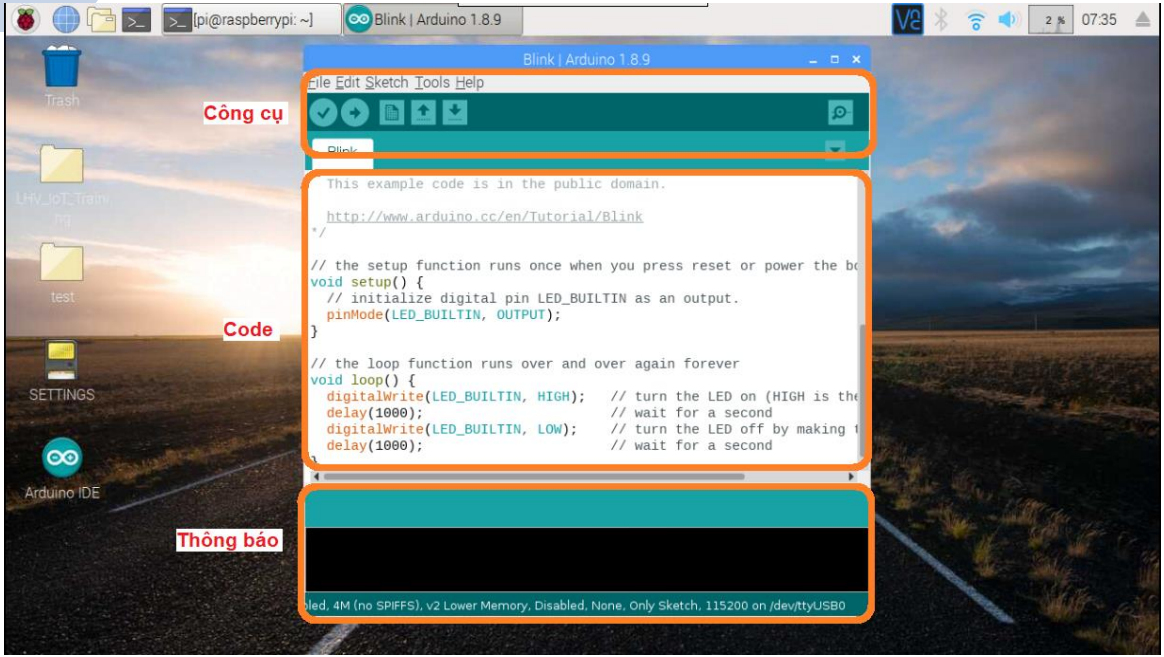
3. Cài đặt Arduino IDE và setup môi trường lập trình

Cài đặt IDE

- Mở trình duyệt web của Raspbian lên
- Truy cập vào trang chủ của arduino: https://www.arduino.cc/
- Trên thanh menu của trang web chọn mục Software -> Downloads
- Kéo xuống phần “Download the Arduino IDE”, chọn phiên bản Linux ARM 32bit
- Giải nén file cài đặt được thư mục cài đặt
- Mở Terminal, vào thư mục vừa giải nén được, chạy lệnh dưới để cài:

$ sudo ./install.sh





- Ô công cụ: gồm có các nút lệnh menu (File, Edit, Sketch, Tools, Help).
- Ô code: Đây là nơi chúng ta sẽ viết chương trình của mình
- Ô thông báo: Nơi hiển thị tất cả các thông báo của IDE trong quá trình biên dịch/upload chương trình.

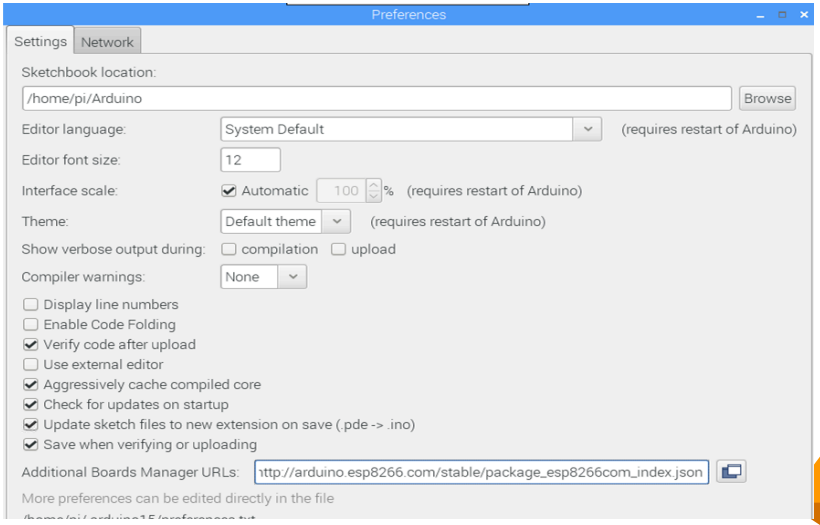
3. Cài đặt Arduino IDE và setup môi trường lập trình

Setup cho bộ kit

- Mở IDE lên, chọn File -> Preferences
- Ở cửa sổ hiện lên, thêm vào ô "*Additional Board Manager URLs*" dòng sau:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

- Chọn "*OK*"



3. Cài đặt Arduino IDE và setup môi trường lập trình

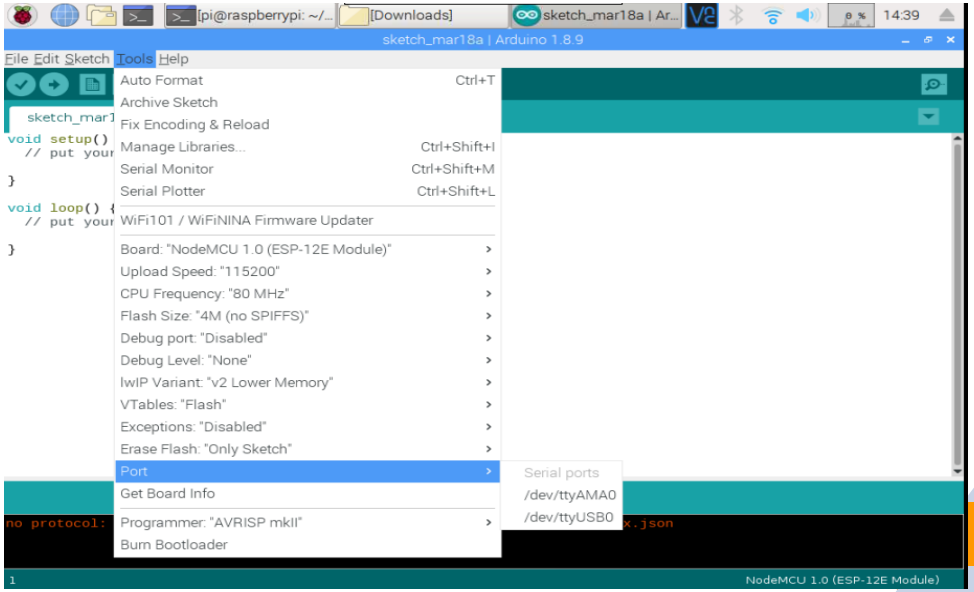
Setup cho bộ kit

- Mở *Tools -> Boards -> Boards Manager*. Ở cửa sổ hiện lên, tìm "*esp8266*", chọn phiên bản *Community* rồi chọn *Install*

3. Cài đặt Arduino IDE và setup môi trường lập trình

Setup cho bộ kit

- Tại IDE, chọn *Tools,* phần *Board* chọn "*NodeMCU 1.0 (ESP-12E Module)*", phần *Port* chọn cổng COM tương ứng với cổng đang được kết nối, thường có dạng "*dev/ttyUSB*"



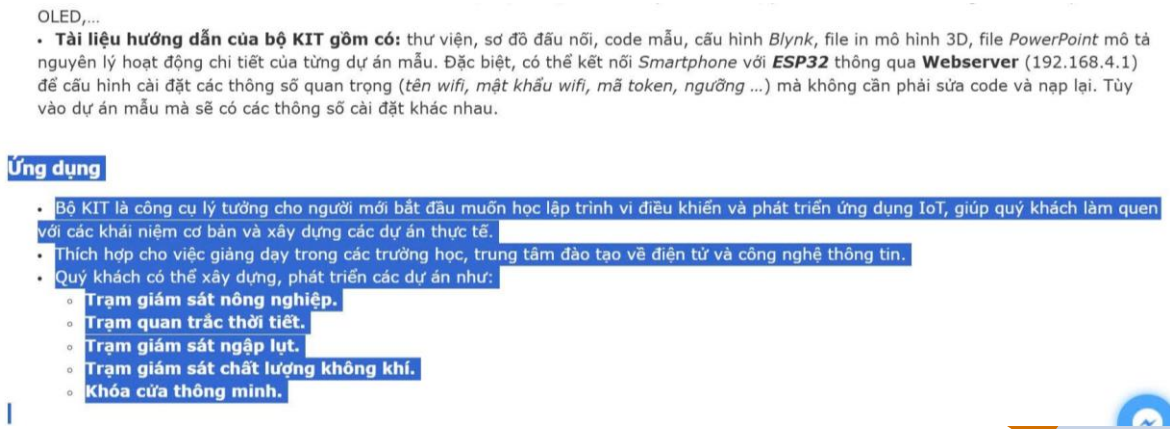
3. Cài đặt Arduino IDE và setup môi trường lập trình

- Cài đặt môi trường Arduino, kết nối ESP32 , load chương trình

https://www.youtube.com/watch?v=sJtJJU3c3xE

Bài 1. Cách viết code và nạp chương trình cho ESP32

https://banlinhkien.com/bo-kit-hoc-tap-esp32-iot-co-ban-blk-p38420583.html



OLED,...

- **Tài liệu hướng dẫn của bộ KIT gồm có:** thư viện, sơ đồ đấu nối, code mẫu, cấu hình *Blynk*, file in mô hình 3D, file *PowerPoint* mô tả nguyên lý hoạt động chi tiết của từng dự án mẫu. Đặc biệt, có thể kết nối *Smartphone* với ***ESP32*** thông qua **Webserver** (192.168.4.1) để cấu hình cài đặt các thông số quan trọng (*tên wifi, mật khẩu wifi, mã token, ngưỡng ...*) mà không cần phải sửa code và nạp lại. Tùy vào dự án mẫu mà sẽ có các thông số cài đặt khác nhau.

Ứng dụng

- Bộ KIT là công cụ lý tưởng cho người mới bắt đầu muốn học lập trình vi điều khiển và phát triển ứng dụng IoT, giúp quý khách làm quen với các khái niệm cơ bản và xây dựng các dự án thực tế.
- Thích hợp cho việc giảng dạy trong các trường học, trung tâm đào tạo về điện tử và công nghệ thông tin.
- Quý khách có thể xây dựng, phát triển các dự án như:
  - **Trạm giám sát nông nghiệp.**
  - **Trạm quan trắc thời tiết.**
  - **Trạm giám sát ngập lụt.**
  - **Trạm giám sát chất lượng không khí.**
  - **Khóa cửa thông minh.**

ĐỒ ÁN

Một số yêu cầu !

- **Đúng form : > 20tr**
- **Đảm bảo nội dung từng hạn mục**
- **Khái quát mô hình nghiên cứu**
- **Thiết kế, Thi công …**
- **Tập trung ESP8266, ESP32, Raspberry Pi…**
- **Lập trình C/C++, Python…**
- **Tính thực tế, ứng dụng**
- **Trình bày**

Một số yêu cầu !

4

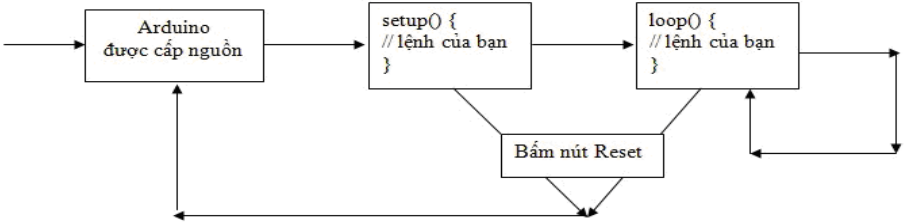
Cấu trúc một chương trình

4. Cấu trúc một chương trình

- Trong một chương trình, có 2 hàm cố định luôn phải có là hàm *setup()* và hàm *loop()*

- Hàm *setup()* là hàm chạy 1 lần khi bắt đầu chương trình, dùng để khai báo các tham số, khai báo các thư viện, thiết lập các chân, …
- Hàm *loop()* sẽ được chạy ngay sau hàm *setup()*. Hàm này sẽ được chạy lặp đi lặp lại liên tục đến khi bị ngắt nguồn

```
sketch_mar17a

void setup() {
  // put your setup code here, to run once:

}

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

}
```



4. Cấu trúc một chương trình

- Chúng ta có thể viết thêm nhiều hàm con khác bên cạnh 2 hàm chính

```
Kiểu_của_hàm Tên_hàm(Các_tham_số){
  /*bắt đầu thân hàm*/
  Thân_hàm
  /*kết thúc hàm*/
  }
```

- Việc đặt tên hàm và biến phải tuân thủ theo các quy tắc đặt tên trong ngôn ngữ C

5

Một số chương trình cơ bản

1. Blink

- Hàm “*pinMode( );” :* cấu hình một chân bất kỳ hoạt động như là một đầu vào tín hiệu (INPUT) hoặc đầu ra tín hiệu (OUTPUT).
- Hàm “*digitalWrite( );” :* để xuất tín hiệu ra một chân digital.
- Hàm *“delay( );”* : có tác dụng dừng/giữ nguyên trạng thái chương trình trong một khoảng thời gian nhất định tính bằng mili giây
- Khi ta vừa làm cho led sáng, hàm *delay(1000);* này sẽ giữ cho led sáng 1s, tương tự khi led tắt cũng vậy

Blink §

```
int LED = D3;

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

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

1. Blink

➢ **<u>Ví dụ:</u> 3 led D3, D4, D8 cùng nhấp nháy, sáng lần lượt, tắt lần lượt**

```
void setup() {
  pinMode(D3, OUTPUT);
  pinMode(D4, OUTPUT);
  pinMode(D8, OUTPUT);
}

void loop() {
  cung_nhap_nhay();
  sang_lan_luot();
  tat_lan_luot();
}

void cung_nhap_nhay(){
  digitalWrite(D3, HIGH);
  digitalWrite(D4, HIGH);
  digitalWrite(D8, HIGH);
  delay(1000);
  digitalWrite(D3, LOW);
  digitalWrite(D4, LOW);
  digitalWrite(D8, LOW);
  delay(1000);
  }
```

```
void sang_lan_luot(){
  digitalWrite(D3, HIGH);
  delay(1000);

  digitalWrite(D4, HIGH);
  delay(1000);

  digitalWrite(D8, HIGH);
  delay(1000);

  }

void tat_lan_luot(){
  digitalWrite(D3, LOW);
  delay(1000);

  digitalWrite(D4, LOW);
  delay(1000);

  digitalWrite(D8, LOW);
  delay(1000);
  }
```

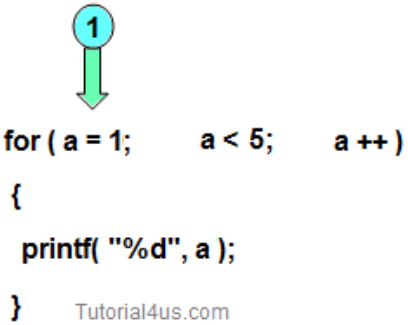
2. Vòng lặp

❖ Vòng lặp For

- Lặp đi lặp lại một số lần xác định trước của một hoặc một chuỗi hành động nào đó.

for (khởi đầu; điều kiện lặp; bước nhảy){
khối lệnh;
};

- Khối lệnh sẽ được thực hiện lặp đi lặp lại cho đến khi biểu thức khởi đầu không còn thoả mãn theo điều kiện lặp nữa



a	Output
1	

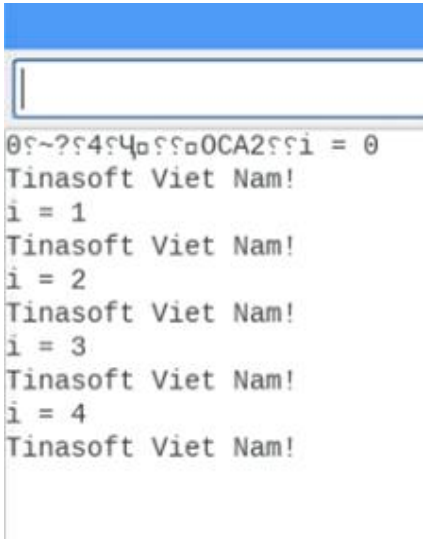
2. Vòng lặp

➢ **<u>Ví dụ:</u> Sử dụng vòng for, in ra dòng chữ "Tinasoft Viet Nam!" 5 lần.**

Code

```
void setup() {
  Serial.begin(9600);
  for(int i=0; i<5; i++){
    Serial.print("i = ");
    Serial.println(i);
    Serial.println("Tinasoft Viet Nam!");
    }
}

void loop() {
}
```

Kết quả



2. Vòng lặp

❖ Vòng lặp While

- Cách vận hành tương tự như vòng for, chỉ khác một chút về cú pháp

while(biểu thức){
* Khối lệnh;*
}

- *Khối lệnh* lặp đi lặp lại liên tục trong khi *biểu thức* còn đúng

code

```
1  a = 1
2  while a < 10:
3   print (a)
4   a += 2
```

output

variables

Code

```
void setup() {
  int i = 0;
  Serial.begin(9600);
  while(i<5){
    Serial.print("i = ");
    Serial.println(i);
    Serial.println("Tinasoft Viet
Nam!");
    i++;
    }
}

void loop() {
}
```

2. Vòng lặp

➢ **<u>Ví dụ:</u> Sử dụng vòng lặp, điều khiển đèn led bất kỳ nhấp nháy 5 lần; từng đèn nháy lần lượt 3 lần.**

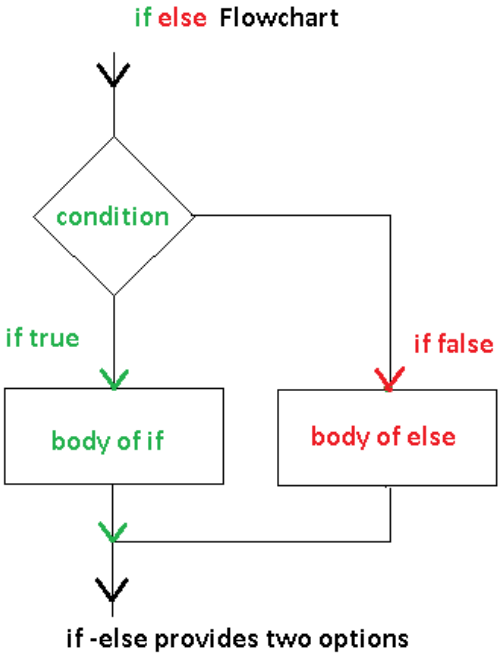
```
void setup() {
  pinMode(D3, OUTPUT);
  pinMode(D4, OUTPUT);
  pinMode(D8, OUTPUT);
}

void loop() {
  for(int i=0; i<5; i++){
    digitalWrite(D4, HIGH);
    delay(500);
    digitalWrite(D4, LOW);
    delay(500);
    };
```

```
for(int i=0; i<3; i++){
    digitalWrite(D3, HIGH);
    delay(500);
    digitalWrite(D3, LOW);
    delay(500);
    digitalWrite(D4, HIGH);
    delay(500);
    digitalWrite(D4, LOW);
    delay(500);
    digitalWrite(D8, HIGH);
    delay(500);
    digitalWrite(D8, LOW);
    delay(500);
    };
}
```

3. Rẽ nhánh

```
a = 1
while a < 7 :
    if(a % 2 == 0):
        print(a, "is even")
    else:
        print(a, "is odd")
    a += 1
```

code output variables www.penjee.com



- Cú pháp:

if(điều kiện){
khối lệnh 1;
}
else{
khối lệnh 2;
}

- Có thể viết hàm *if* không có *else* hoặc nhiều hàm *if* lồng vào với nhau

3. Rẽ nhánh

➢ **<u>Ví dụ 1:</u> Bấm nút thì đèn sáng, không bấm nút thì đèn tắt**

```
int LED = D3;
int BUTTON = D7;
int buttonStatus = 0;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT);
}
void loop() {
  buttonStatus=digitalRead(BUTTON);
  if (buttonStatus == 1){
    digitalWrite(LED, HIGH);
    delay(200);
  }
  if (buttonStatus == 0){
    digitalWrite(LED, LOW);
    delay(200);
  }
}
```

➢ Ví dụ 2: Bấm nút đổi trạng thái đèn

```
int LED = D3;
int BUTTON = D7;
int buttonStatus = 0;
int oldButton = 0;
int ledStatus = 0;

void setup() {
  pinMode(BUTTON, INPUT);
  pinMode(LED, OUTPUT);
}
void loop() {
  buttonStatus = digitalRead(BUTTON);
  if (buttonStatus != oldButton) {
    oldButton = buttonStatus;
    delay(300);
    if (buttonStatus == HIGH) {
      ledStatus = !ledStatus;
    }
  }
  if (ledStatus == 1 ) {
    digitalWrite(LED, HIGH);
  }
  else {
    digitalWrite(LED, LOW);
  }
}
```

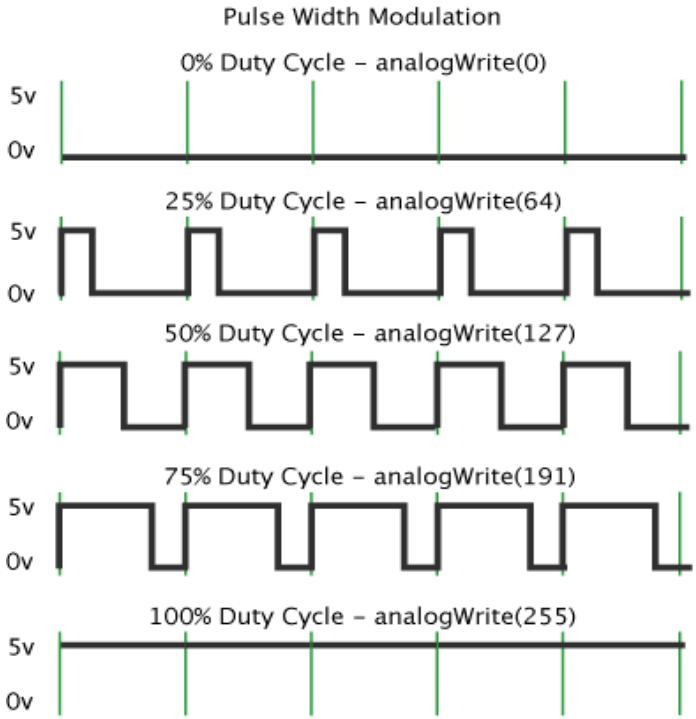
4. XungPWM

- Xung là các trạng thái cao / thấp (HIGH/LOW) về mức điện áp được lặp đi lặp lại. Đại lượng đặc trưng cho 1 xung PWM (Pulse Width Modulation) bao gồm tần số (frequency) và chu kì xung (duty cycle).
- Tần số là số lần lặp lại dao động trong khoảng thời gian 1 giây, đơn vị Hz. Ví dụ, 1Hz = 1 dao động trong 1 giây. 2Hz = 2 dao động trong 1 giây. 16MHz = 16 triệu dao động trong 1 giây.
- Dao động được xác định từ trạng thái bắt đầu và kết thúc ngay trước khi trạng thái bắt đầu được lặp lại.

- Một dao động sẽ bao gồm 2 trạng thái điện: mức cao (x giây) và mức thấp (y giây). Tỉ lệ phần trăm thời gian giữa 2 trạng thái điện này chính là chu kì xung T = x/y

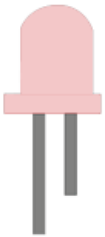
4. Xung PWM

- Xung khi sử dụng với hàm analogWrite() trong Arduino:



➢ Ví dụ 1: Sử dụng xung PWM để làm thay đổi độ sáng đèn LED

```
int LED = D4;
int brightness = 0;     // mặc định độ sáng của đèn là 0
int fadeAmount = 5;     // giá trị thay đổi mỗi lần thay đổi độ sáng

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

void loop()  {
  analogWrite(LED, brightness);

  brightness = brightness + fadeAmount;

  //Nếu độ sáng = 0 hoặc = 255 thì sẽ đổi chiều của biến thay đổi độ
sáng.
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  delay(50);
}
```



➢ Ví dụ 1: Sử dụng xung PWM để làm thay đổi độ sáng đèn LED

1. Khai báo biến

```cpp
int LED = D4;           // Chân D4 được nối với LED
int brightness = 0;     // Biến lưu độ sáng LED, ban đầu = 0 (tắt)
int fadeAmount = 5;     // Bước thay đổi độ sáng mỗi lần lặp
```

- `brightness` nhận giá trị từ 0 → 255 (tương ứng với duty cycle 0% → 100%).
- `fadeAmount` cho biết tăng hoặc giảm độ sáng mỗi lần lặp vòng `loop()`.

---

2. Hàm `setup()`

```cpp
void setup() {
  pinMode(LED, OUTPUT);  // Cấu hình chân LED là ngõ ra
}
```

- Chuẩn bị để xuất tín hiệu PWM điều khiển LED.

➢ **Ví dụ 1: Sử dụng xung PWM để làm thay đổi độ sáng đèn LED**

3. Hàm `loop()`

```cpp
void loop() {
  analogWrite(LED, brightness);    // Xuất tín hiệu PWM với độ sáng hiện tại
```

- Arduino phát xung PWM ra chân LED, độ sáng tùy thuộc vào giá trị `brightness`.

```cpp
  brightness = brightness + fadeAmount;
```

- Cập nhật giá trị độ sáng: tăng thêm 5 đơn vị mỗi vòng lặp.
- Ví dụ: 0 → 5 → 10 → 15 → ... → 255.

```cpp
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount;
  }
```

- Nếu LED đạt **tối đa (255)** hoặc **tối thiểu (0)** thì đảo chiều thay đổi.
- `fadeAmount` đang là `+5` thì đổi thành `-5`, ngược lại `-5` đổi thành `+5`.
- Kết quả: LED sáng dần lên, rồi lại mờ dần xuống ↓ ạo hiệu ứng **nhấp nháy mượt**

➢ Ví dụ 2: Dùng 2 nút nhấn để tăng/giảm độ sáng đèn LED

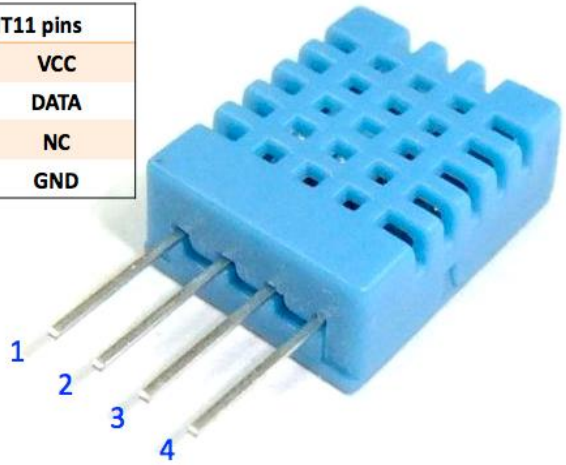
```
int brightness = 0;
int fadeAmount = 15;
int button1 = D7;
int button2 = D5;
int b1Status = 0;
int b2Status = 0;

void setup() {
  pinMode(D7, INPUT);
  pinMode(D5, INPUT);
  pinMode(D4, OUTPUT);
  Serial.begin(9600);
}
```

```
void loop() {
  analogWrite(D4, brightness);
  if (brightness <= 0) {
    brightness = 0;
  }
  if (brightness >= 255) {
    brightness = 255;
  }

  b1Status = digitalRead(D7);
  b2Status = digitalRead(D5);
  if (b1Status == HIGH) {
    brightness = brightness + fadeAmount;
  }
  if (b2Status == HIGH) {
    brightness = brightness - fadeAmount;
  }
  delay(200);
  Serial.println(brightness);
  Serial.println(analogRead(D4));
}
```

Cảm biến nhiệt độ, độ ẩm DHT11

- DHT11 là cảm biến có thể đọc được nhiệt độ và độ ẩm cùng lúc.
- Cấu tạo gồm 4 chân:
  - VCC (chân nguồn dương)
  - DATA (chân tín hiệu)
  - NC (Not Connected - chân này chúng ta không dùng đến)
  - GND (chân nối đất).

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



Cảm biến nhiệt độ, độ ẩm DHT11

- Dữ liệu truyền về của DHT11 gồm 40bit dữ liệu theo thứ tự: 8 bit biểu thị phần nguyên của độ ẩm + 8 bit biểu thị phần thập phân của độ ẩm + 8 bit biểu thị phần nguyên của nhiệt độ + 8 bit biểu thị phần thập phân của nhiệt độ + 8 bit check sum.
- Ví dụ: ta nhận được 40 bit dữ liệu như sau:

  0011 0101 0000 0000 0001 1000 0000 0000 0100 1101
- Tính toán:

  8 bit checksum: 0011 0101 + 0000 0000 + 0001 1000 + 0000 0000

  = 0100 1101
- Độ ẩm: *0011 0101 = 35H = 53%* (ở đây do phần thập phân có giá trị 0000 0000, nên ta bỏ qua không tính phần thập phân)
- Nhiệt độ: *0001 1000 = 18H = 24°C* (ở đây do phần thập phân có giá trị 0000 0000, nên ta bỏ qua không tính phần thập phân)

5. Đọc cảm biến

Cảm biến nhiệt độ, độ ẩm DHT11

Byte (8 bit)	Giá trị nhị phân (binary)	Giá trị thập phân (decimal)	Ý nghĩa cảm biến
Byte 1	`0011 0101`	53	Độ ẩm – phần nguyên = **53%**
Byte 2	`0000 0000`	0	Độ ẩm – phần thập phân = **0** → bỏ qua
Byte 3	`0001 1000`	24	Nhiệt độ – phần nguyên = **24°C**
Byte 4	`0000 0000`	0	Nhiệt độ – phần thập phân = **0** → bỏ qua
Byte 5	`0100 1101`	77	**Checksum** để kiểm tra dữ liệu

Cảm biến nhiệt độ, độ ẩm DHT11

. Checksum kiểm tra

- 53 + 0 + 24 + 0 = 77
- Byte 5 = `0100 1101` = 77

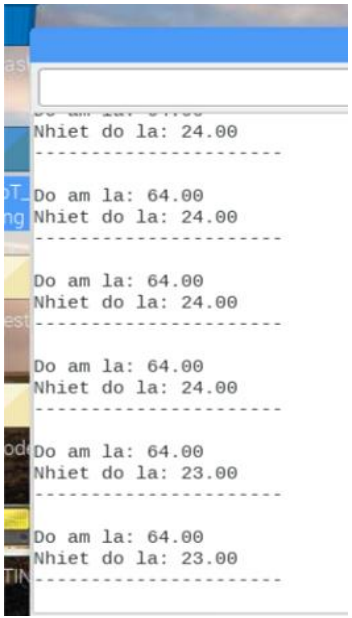
✅ Kết quả hợp lệ.

5. Đọc cảm biến

➢ Ví dụ 1: Đọc giá trị nhiệt độ/độ ẩm từ cảm biến DHT11 và hiển thị lên màn hình Serial

- Trước tiên ta thêm thư viện: chọn *Sketch -> Include Library -> ADD .Zip Library*; chọn đến 2 file thư viện đã tải về là *Adafruit_Sensor-master* và *DHT_sensor_library* rồi thêm vào.

Kết quả



Code

```
#include <DHT.h>
#define DHTPIN D0
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  Serial.println();
  Serial.print("Do am la: ");
  Serial.println(h);
  Serial.print("Nhiet do la: ");
  Serial.println(t);
  Serial.println("--------------------");
  delay(1000);
}
```

5. Đọc cảm biến

➢ **<u>Ví dụ 2:</u> Chương trình cảnh báo nhiệt độ, khi nhiệt độ lên quá cao, vượt một ngưỡng nào đấy thì nháy đèn và có còi cảnh báo.**

```
#include <DHT.h>
#define DHTPIN D0
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
int led = D3;
int coi = D6;

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(led, OUTPUT);
  pinMode(coi, OUTPUT);
}
```

```
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  Serial.println();
  Serial.print("Do am la: ");
  Serial.println(h);
  Serial.print("Nhiet do la: ");
  Serial.println(t);
  Serial.println("---------------");
  if (t > 27) {
    canh_bao();
  }
}

void canh_bao() {
  digitalWrite(led, HIGH);
  digitalWrite(coi, HIGH);
  delay(200);
  digitalWrite(led, LOW);
  digitalWrite(coi, LOW);
  delay(200);
}
```

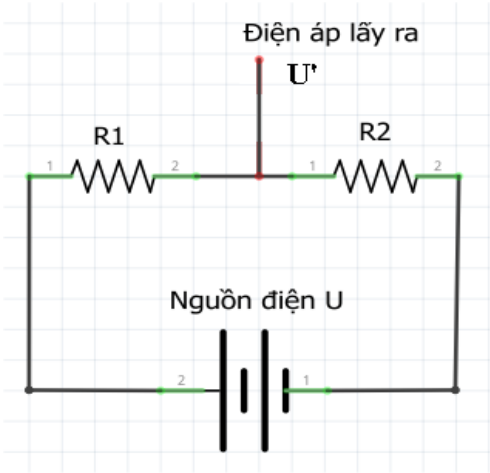
Cảm biến ánh sáng với quang trở

- Quang trở là một linh kiện rất hay gặp và được sử dụng trong những mạch cảm biến ánh sáng.
- Là một loại điện trở có thể thay đổi giá trị theo cường độ ánh sáng. Nếu đặt ở môi trường có ít ánh sáng thì điện trở của quang trở sẽ tăng cao và ngược lại, nếu đặt ở nơi có nhiều ánh sáng thì điện trở sẽ giảm.
- Để lấy được giá trị quang trở, chúng ta sử dụng phương pháp đọc hiệu điến thế từ cầu phân áp
- Cầu phân áp có cấu tạo gần giống như một biến trở bình thường, gồm 2 điện trở nối tiếp nhau có tác dụng phân chia điện áp U từ nguồn điện ra điện áp theo ý muốn của người dùng.

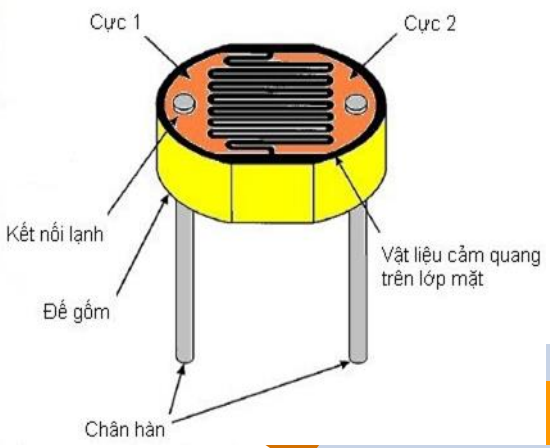
Cảm biến ánh sáng với quang trở

- Điện áp U' lấy ra sẽ được tính bởi công thức:

➢ U' = U / (R1 + R2) x R2. Trong đó I = U / (R1 + R2) chính là cường độ dòng điện trong mạch.



- Đối với board mạch của chúng ta, quang trở có thể được xem là điện trở R1, và phía trước mỗi chân Analog của Arduino có một điện trở, xem đó là điện trở R2 = 10k



6. Đọc quang trở

➢ **Ví dụ 1: Đọc giá trị cường độ ánh sáng thông qua quang trở**

Code

```
int quangtro = A0;
void setup() {
  Serial.begin(9600);
}

void loop() {
  int giatriQuangtro =
analogRead(quangtro);

  Serial.println(giatriQuangtro);
}
```

Kết quả

- Khi rê tay lại gần quang trở (giảm cường độ ánh sáng chiếu vào) thì hiệu điện thế khá nhỏ, giá trị đọc được khoảng 400-500. Cường độ ánh sáng càng lớn thì hiệu điện thế càng cao, giá trị đọc được càng nhỏ.

6. Đọc quang trở

➢ **<u>Ví dụ 2:</u> Tự động bật đèn khi thiếu ánh sáng, tắt đèn khi đủ ánh sáng.**

```
int quangtro = A0;
int led = D4;

void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
}

void loop() {
  int giatriQuangtro = analogRead(quangtro);
  Serial.println(giatriQuangtro);
  if (giatriQuangtro > 700) {
    digitalWrite(led, HIGH);
  }
  else {
    digitalWrite(led, LOW);
  }
}
```

THANK YOU!