TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

IT4735

Internet of Things and Applications

Giảng viên: TS. Phạm Ngọc Hưng
Viện Công nghệ Thông tin và Truyền thông
hungpn@soict.hust.edu.vn

Nội dung

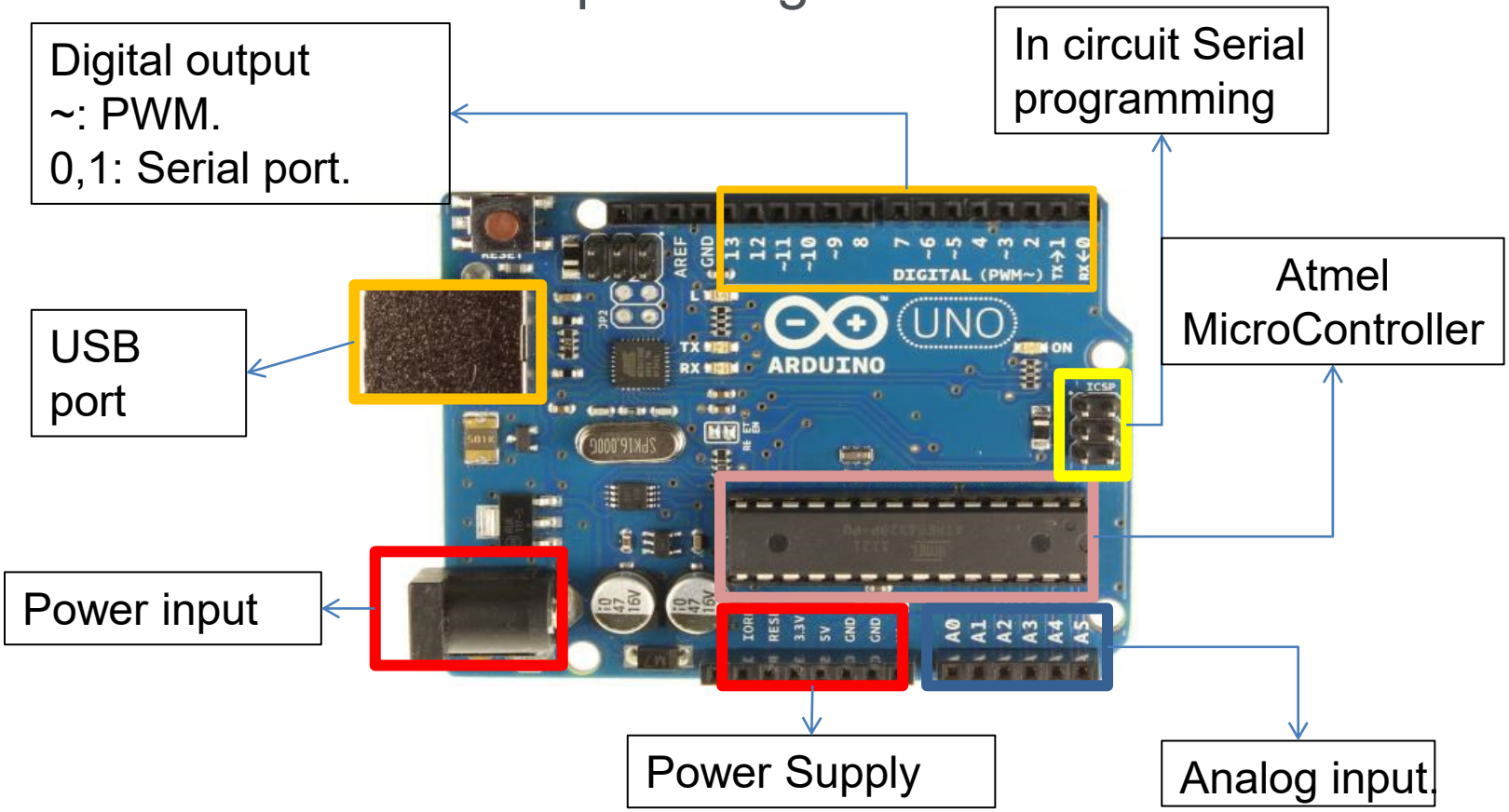
- Chương 1. Tổng quan về IoT
- Chương 2. Hệ thống IoT và các công nghệ
- Chương 3. Lập trình IoT
- Chương 4. An toàn và Bảo mật IoT
- Chương 5. Xây dựng ứng dụng IoT

Chương 3. Lập trình IoT

- 3.1. Lập trình với thiết bị IoT
  - 3.1.1. Lập trình với thiết bị Arduino
  - 3.1.2. Lập trình với thiết bị ESP32
  - 3.1.3. Lập trình với thiết bị Rasberry Pi
  - 3.1.4. Truyền dữ liệu ESP32 – Raspberry Pi
- 3.2. Dịch vụ IoT trên máy chủ, nền tảng đám mây
- 3.3. Thiết kế một hệ thống ứng dụng IoT

3.1.1. Lập trình thiết bị Arduino

- Lập trình với Arduino:
  - VĐK ATMega 8 bit, không có hệ điều hành
  - Arduino Uno: Microchip ATmega328



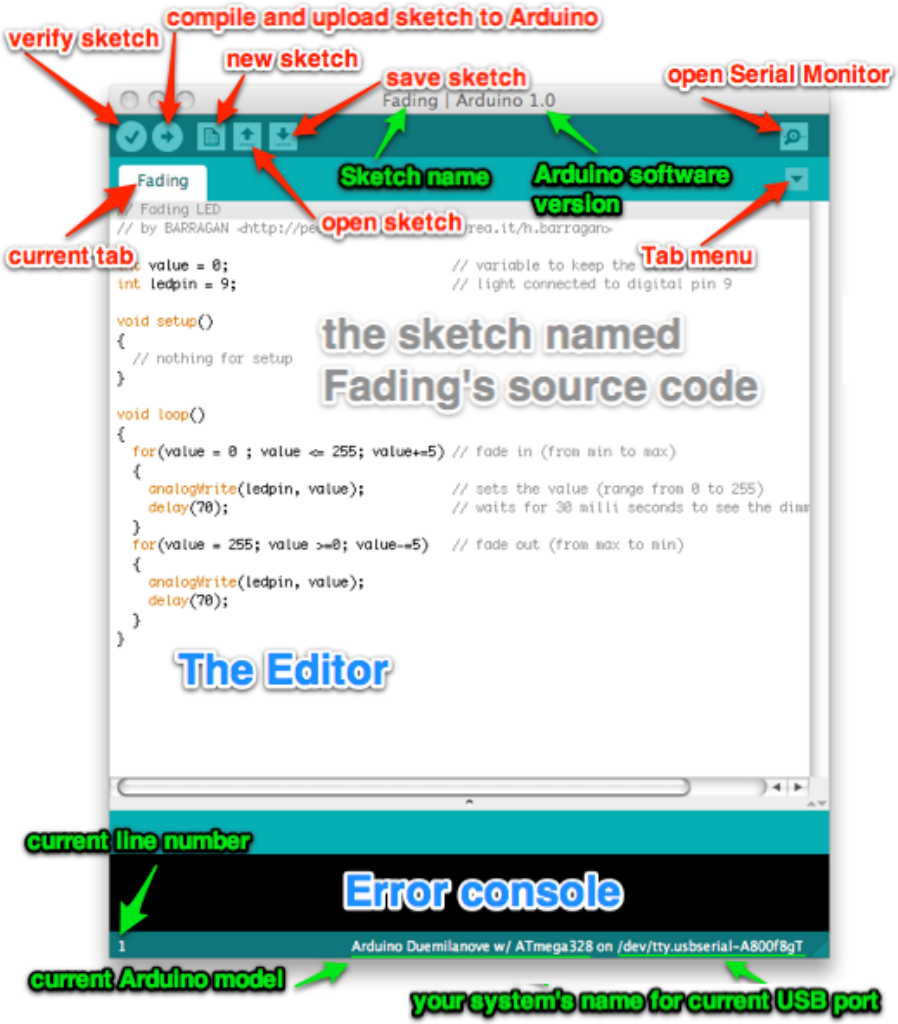
Lập trình Arduino

- Cài đặt Arduino IDE
  - https://www.arduino.cc/en/guide/windows
- Cấu trúc code

```
void setup()
{
        //Used to indicate the initial values of system on  starting.
}

void loop()
{
        Contains the statements that will run whenever the
system is powered after setup.
}
```

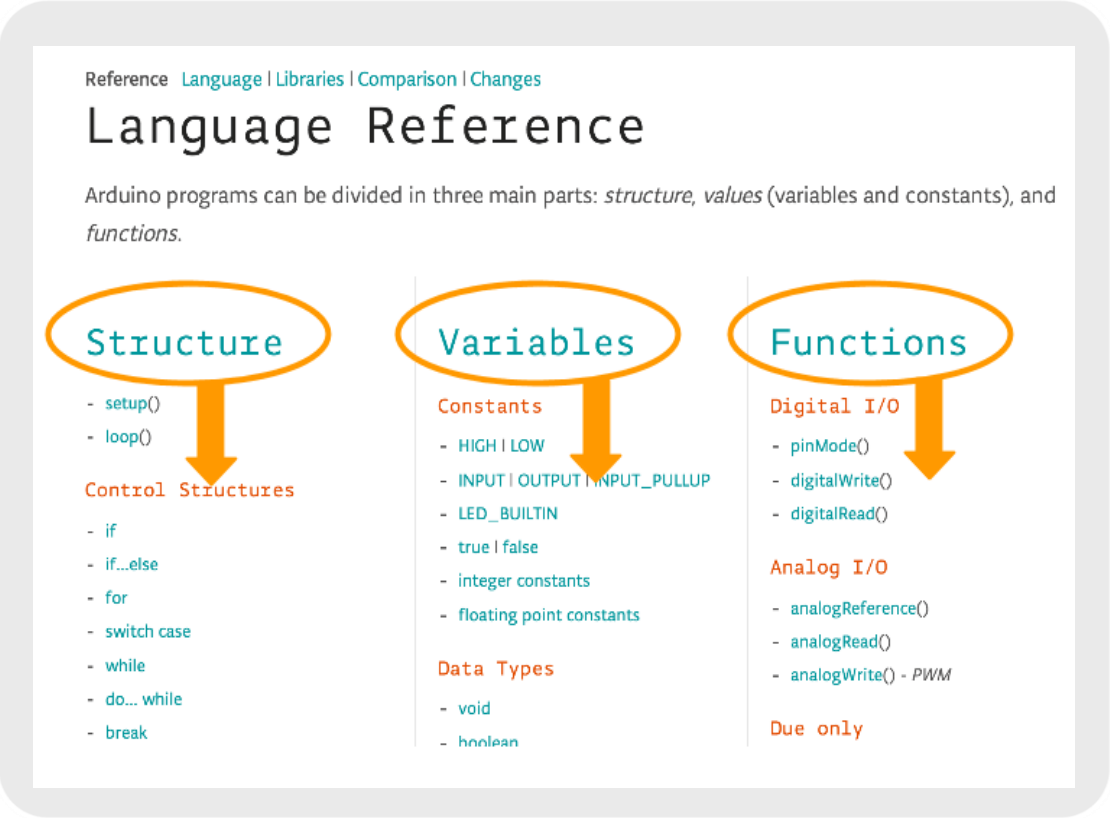
Lập trình Arduino

- Program used to code and upload it to arduino boards (using PC)
- Editor (for code edit)
- Sketch (piece of program)
-



Lập trình Arduino

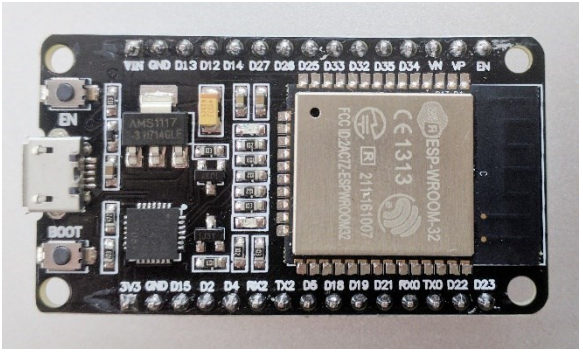
- Tài liệu: Arduino References
- http://arduino.cc/en/Reference/HomePage

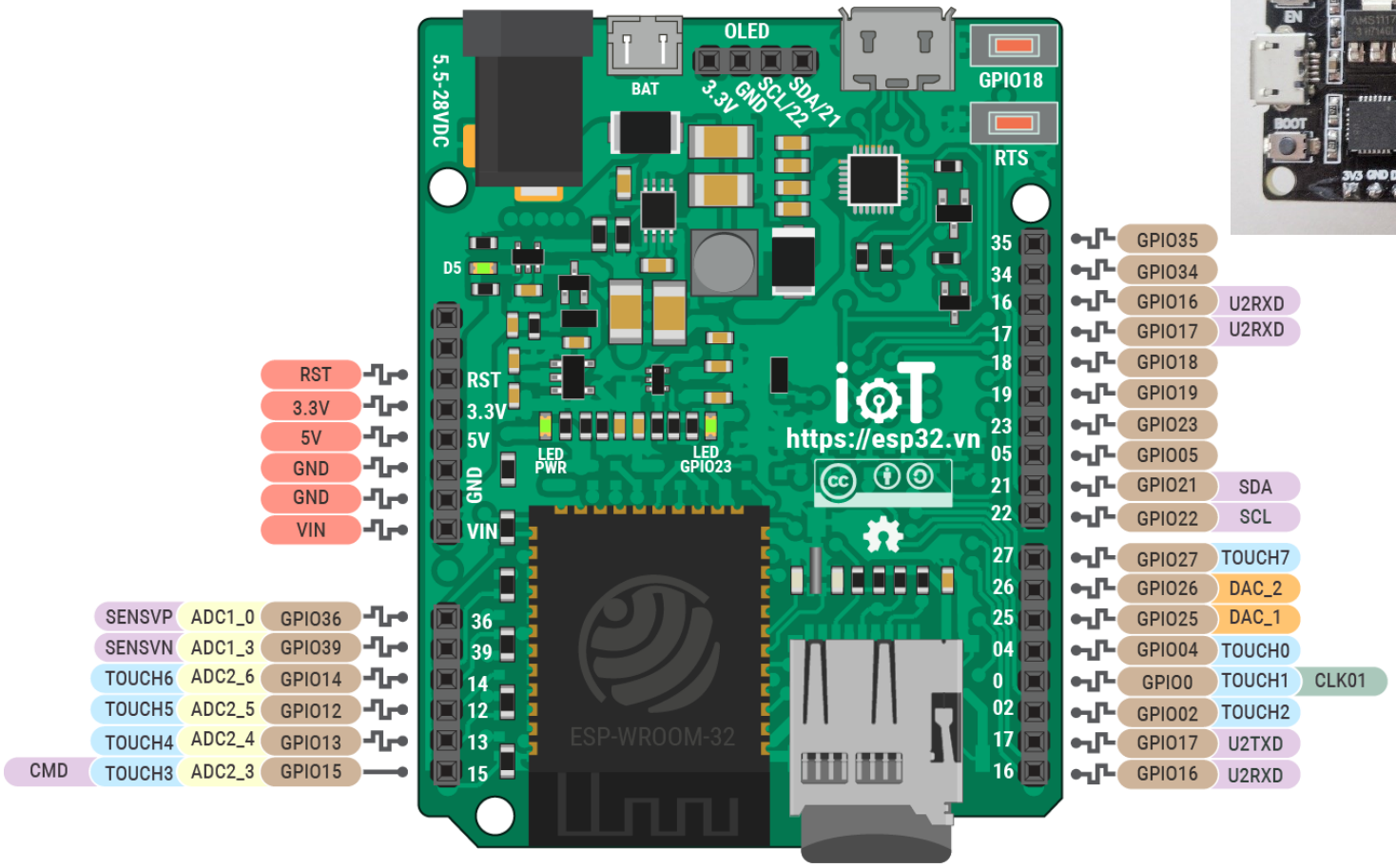


3.1.2. Lập trình thiết bị ESP32

- Lập trình với EPS32 có HĐH FreeRTOS

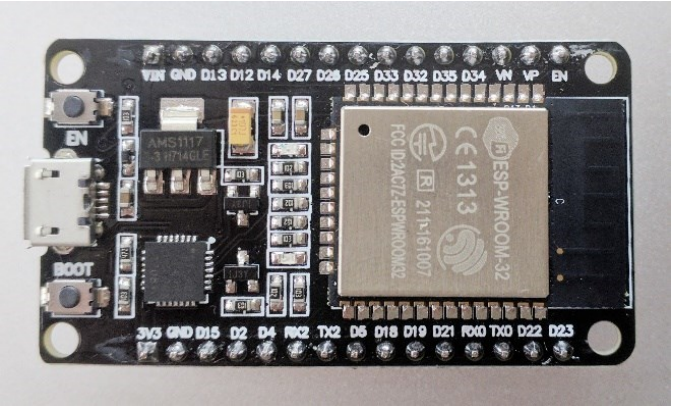
https://esp32.vn/index.html





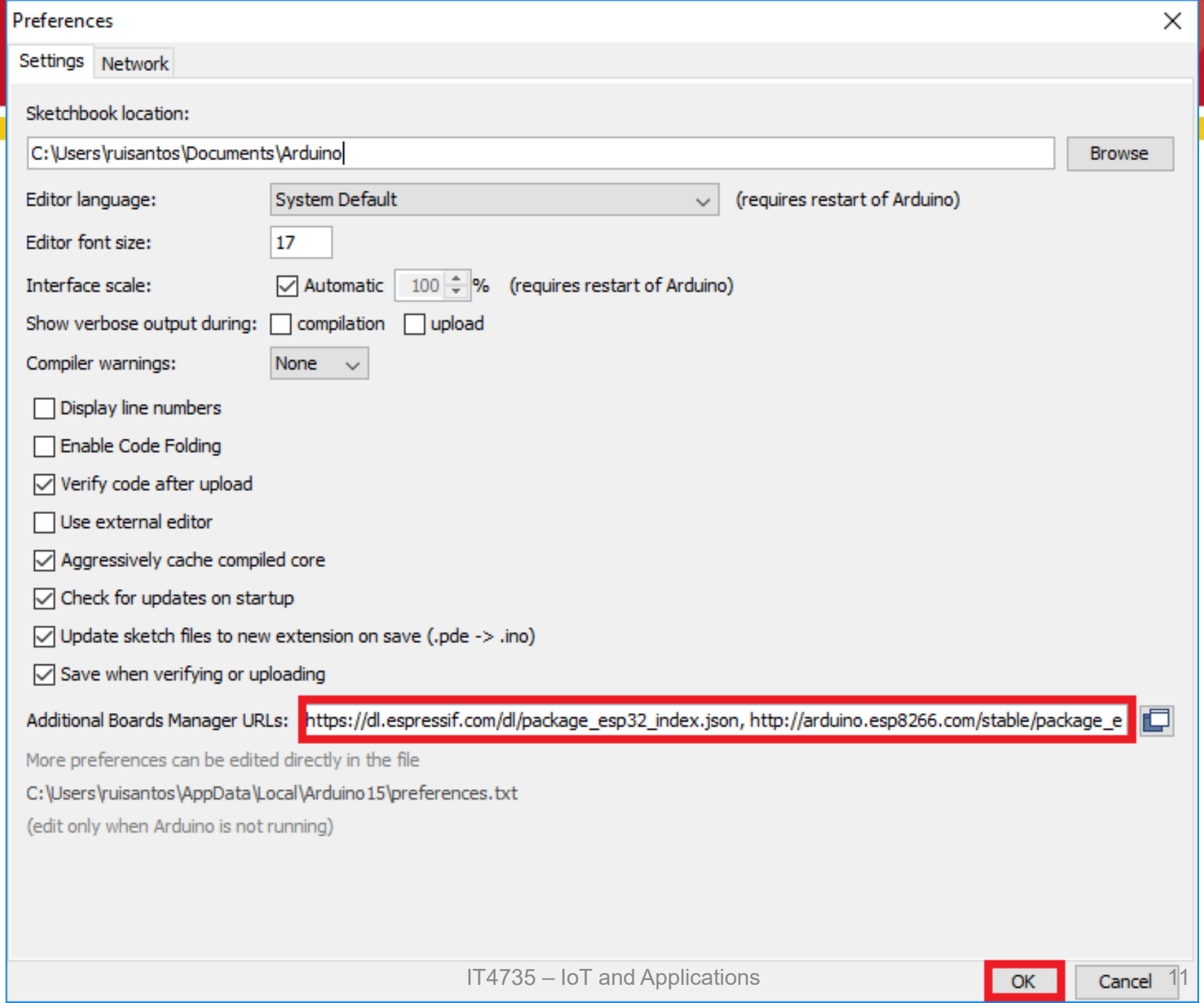
Lập trình thiết bị ESP32

- A. Cài đặt môi trường Arduino IDE cho ESP32
- B. Lập trình GPIO ví dụ LED Blinky
- C. Lập trình kết nối Wifi



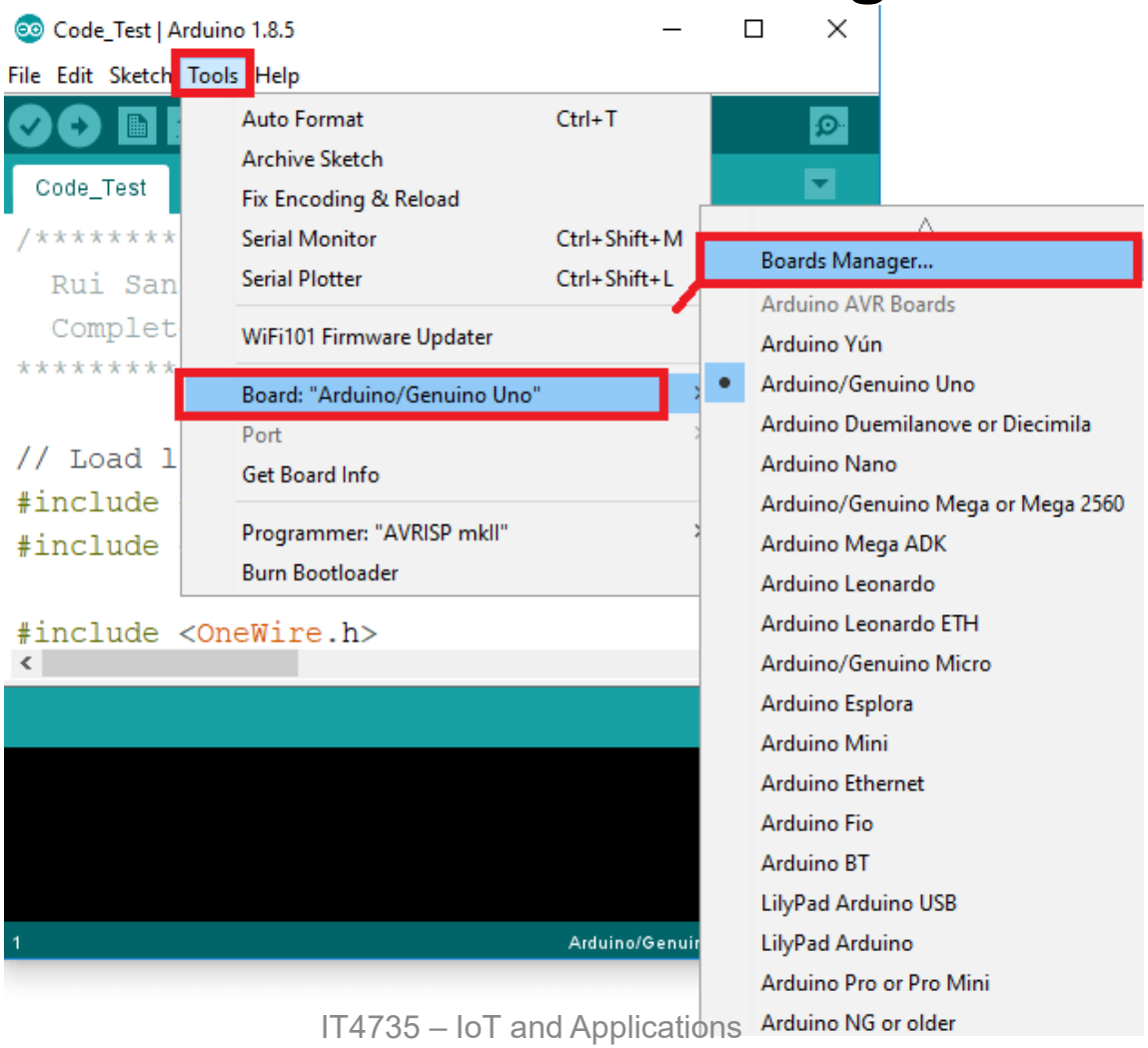
A. Cài đặt môi trường Arduino IDE cho ESP32

- Cài đặt Add-on ESP32 Board trên Arduino IDE:
  - 1) Mở cửa sổ tùy chọn từ IDE Arduino. Chuyển đến **File> Preferences**
  - 2) Nhập https://dl.espressif.com/dl/package_esp32_index.json vào trường “Additional Board Manager URLs” như trong hình bên. Sau đó, nhấp vào nút “OK”


Preferences
Settings
Network
Sketchbook location:
C:\Users\ruisantos\Documents\Arduino
Browse
Editor language:
System Default
(requires restart of Arduino)
Editor font size:
17
Interface scale:
Automatic
100
%
(requires restart of Arduino)
Show verbose output during:
compilation
upload
Compiler warnings:
None
Display line numbers
Enable Code Folding
Verify code after upload
Use external editor
Aggressively cache compiled core
Check for updates on startup
Update sketch files to new extension on save (.pde -> .ino)
Save when verifying or uploading
Additional Boards Manager URLs:
https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_e
More preferences can be edited directly in the file
C:\Users\ruisantos\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)
OK
Cancel

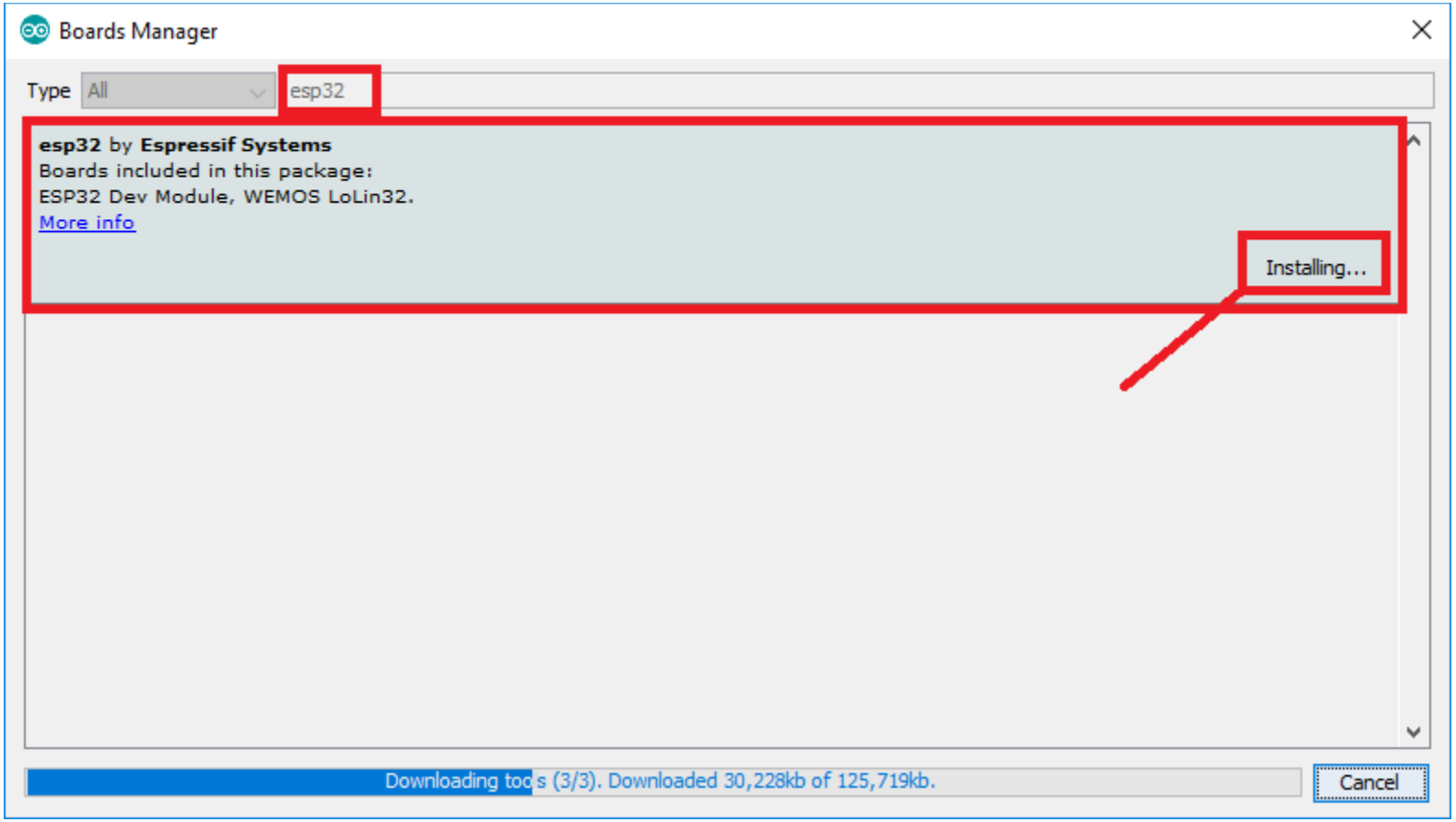
Cài đặt môi trường Arduino IDE cho ESP32

3) Mở board quản lý. Chuyển đến **Tools > Board > Boards Manager…**



Cài đặt môi trường Arduino IDE cho ESP32

3) Mở board quản lý. Chuyển đến **Tools > Board > Boards Manager…**

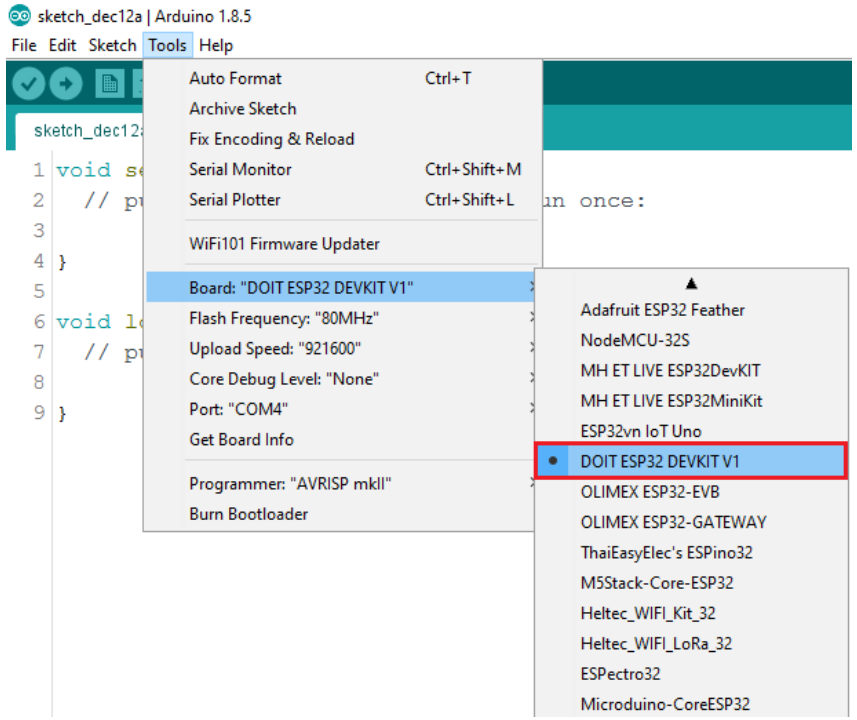


Cài đặt môi trường Arduino IDE cho ESP32

- Kết nối bo mạch ESP32 với máy tính.

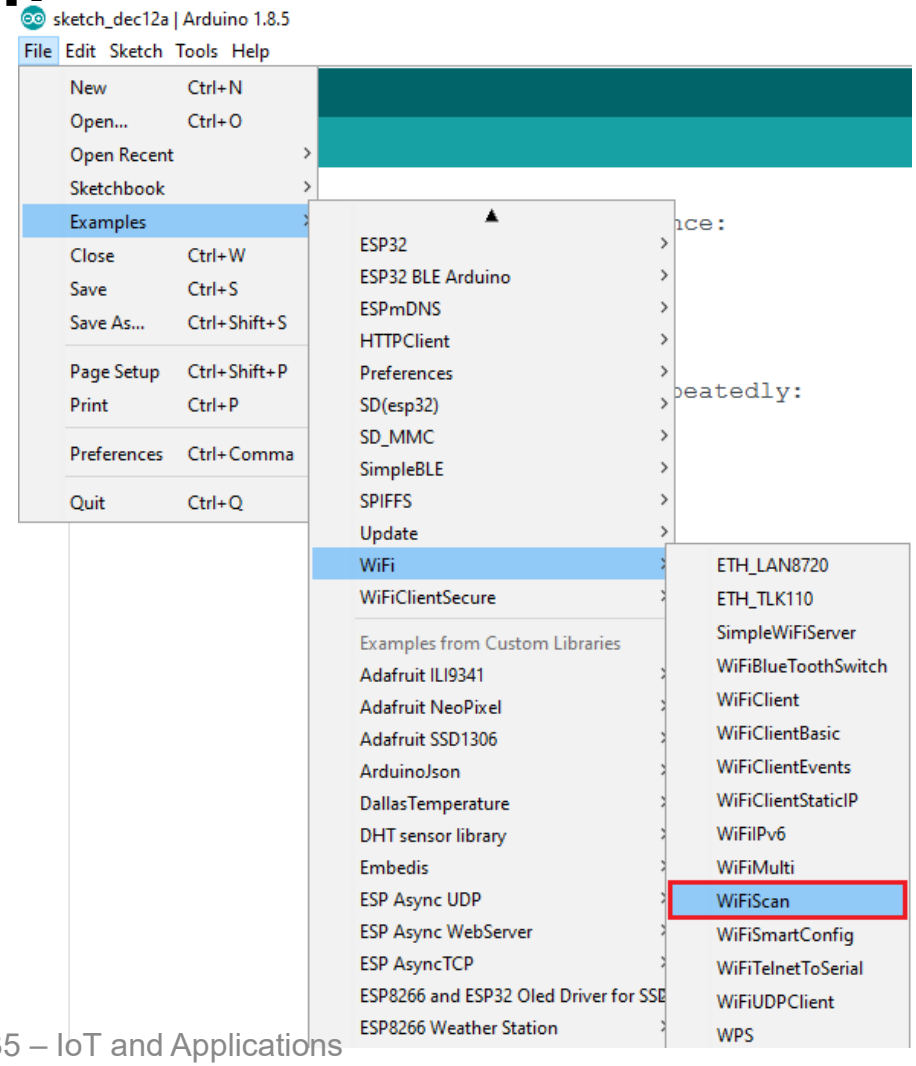
1) Mở Arduino IDE

2) Chọn Board trong **Tools** > **Board** menu (trong trường hợp ví dụ là **DOIT ESP32 DEVKIT V1**)



Cài đặt môi trường Arduino IDE cho ESP32

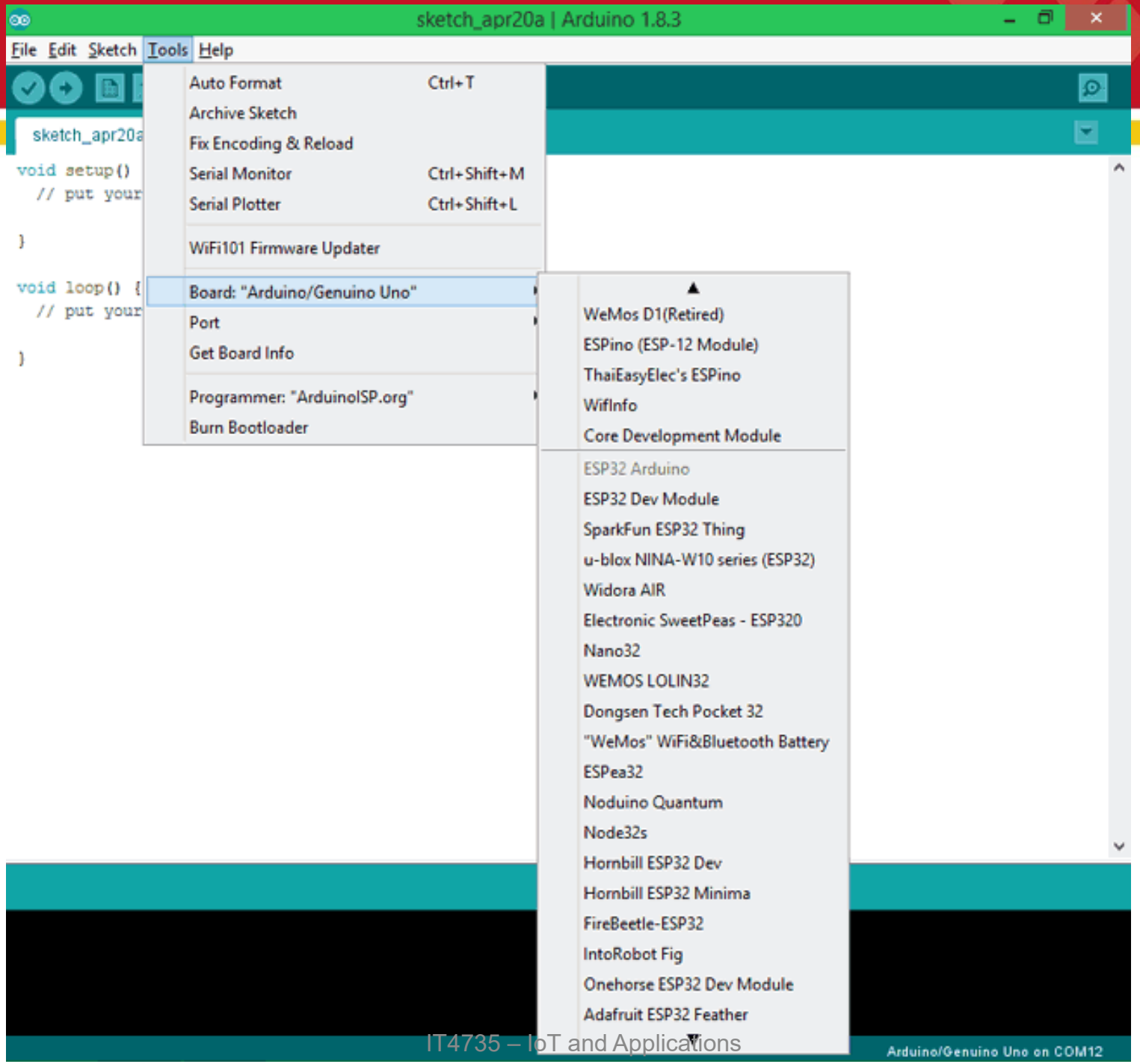
- Mở ví dụ sau trong **File** > **Examples** > **WiFi (ESP32)** > **WiFi Scan**



B. Lập trình GPIO cho ESP32

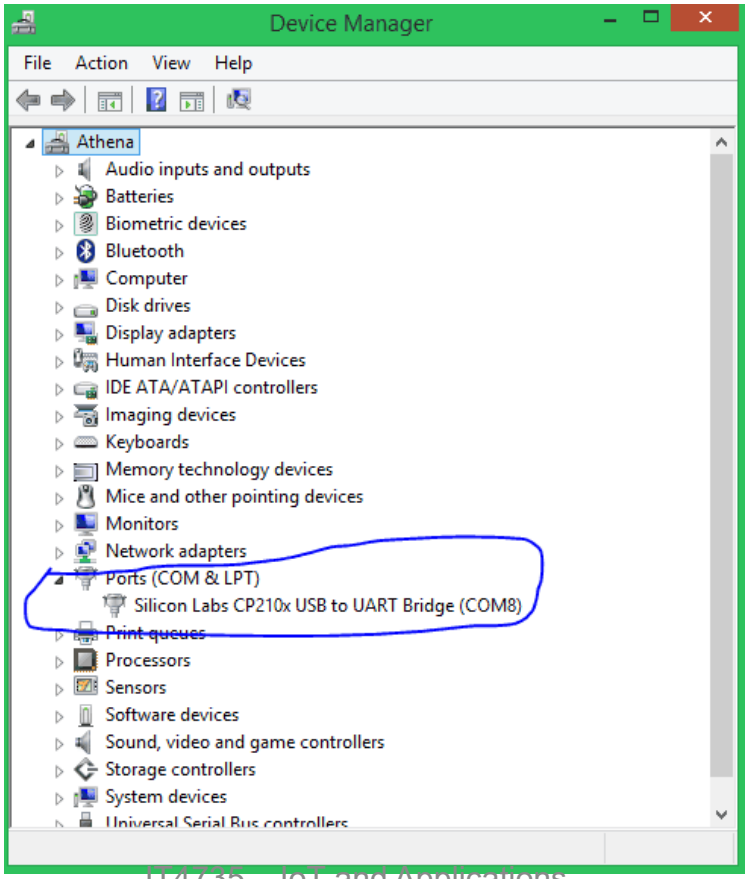
Ứng dụng LED Blinky

- **STEP 1:** Connect your ESP32 board to your computer through the micro-USB cable. Make sure the red LED goes high on the module to ensure power supply.
- **STEP2:** Start the Arduino IDE and navigate to *Tools -> Boards and select ESP32Dev* board as shown below


sketch_apr20a | Arduino 1.8.3
File Edit Sketch Tools Help
sketch_apr20a
void setup()
// put your
}
void loop() {
// put your
}
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "ArduinoISP.org"
Burn Bootloader
WeMos D1(Retired)
ESPino (ESP-12 Module)
ThaiEasyElec's ESPino
WifInfo
Core Development Module
ESP32 Arduino
ESP32 Dev Module
SparkFun ESP32 Thing
u-blox NINA-W10 series (ESP32)
Widora AIR
Electronic SweetPeas - ESP320
Nano32
WEMOS LOLIN32
Dongsen Tech Pocket 32
"WeMos" WiFi&Bluetooth Battery
ESPea32
Noduino Quantum
Node32s
Hornbill ESP32 Dev
Hornbill ESP32 Minima
FireBeetle-ESP32
IntoRobot Fig
Onehorse ESP32 Dev Module
Adafruit ESP32 Feather
Arduino/Genuino Uno on COM12

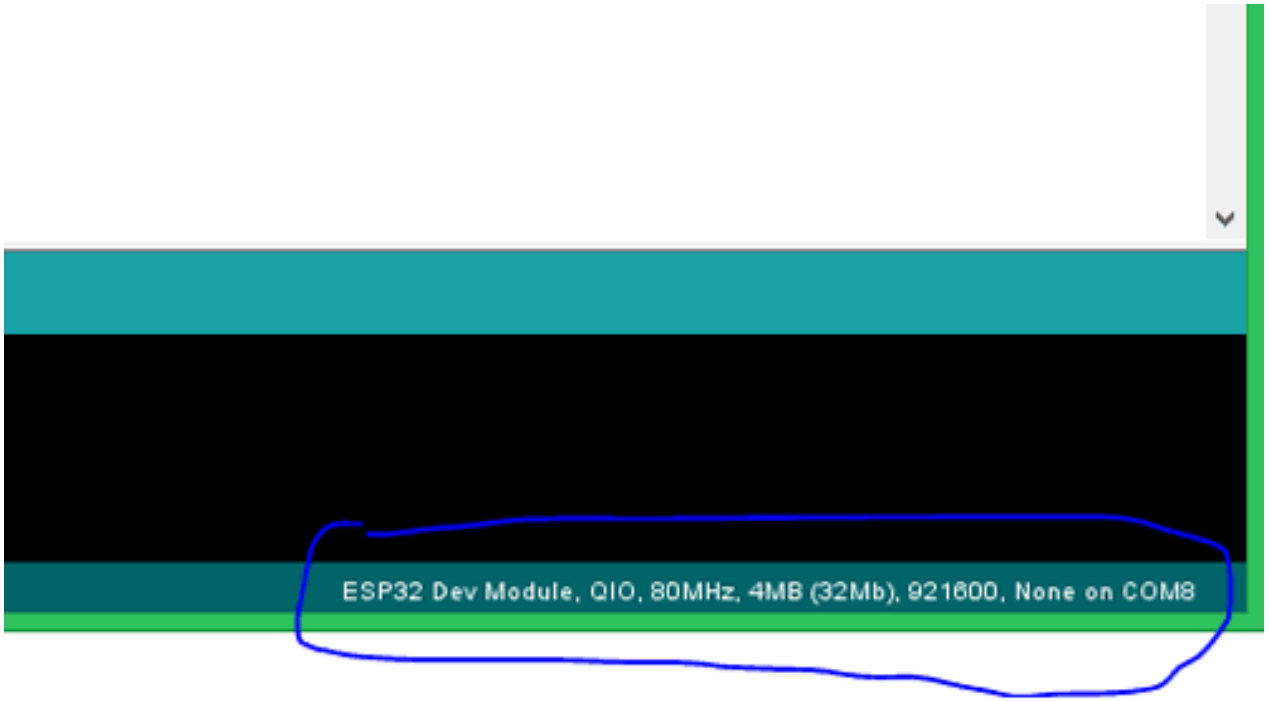
Ứng dụng LED blinky

- **STEP 3:** Open device manager and check to which com port your ESP32 is connected to. Mine is connected to COM 8 as shown below.



Ứng dụng LED blinky

- **STEP 4:** Go back to Arduino IDE and under *Tools -> Port select* the Port to which your ESP is connected to. Once selected you should see something like this on the bottom left corner of the IDE.



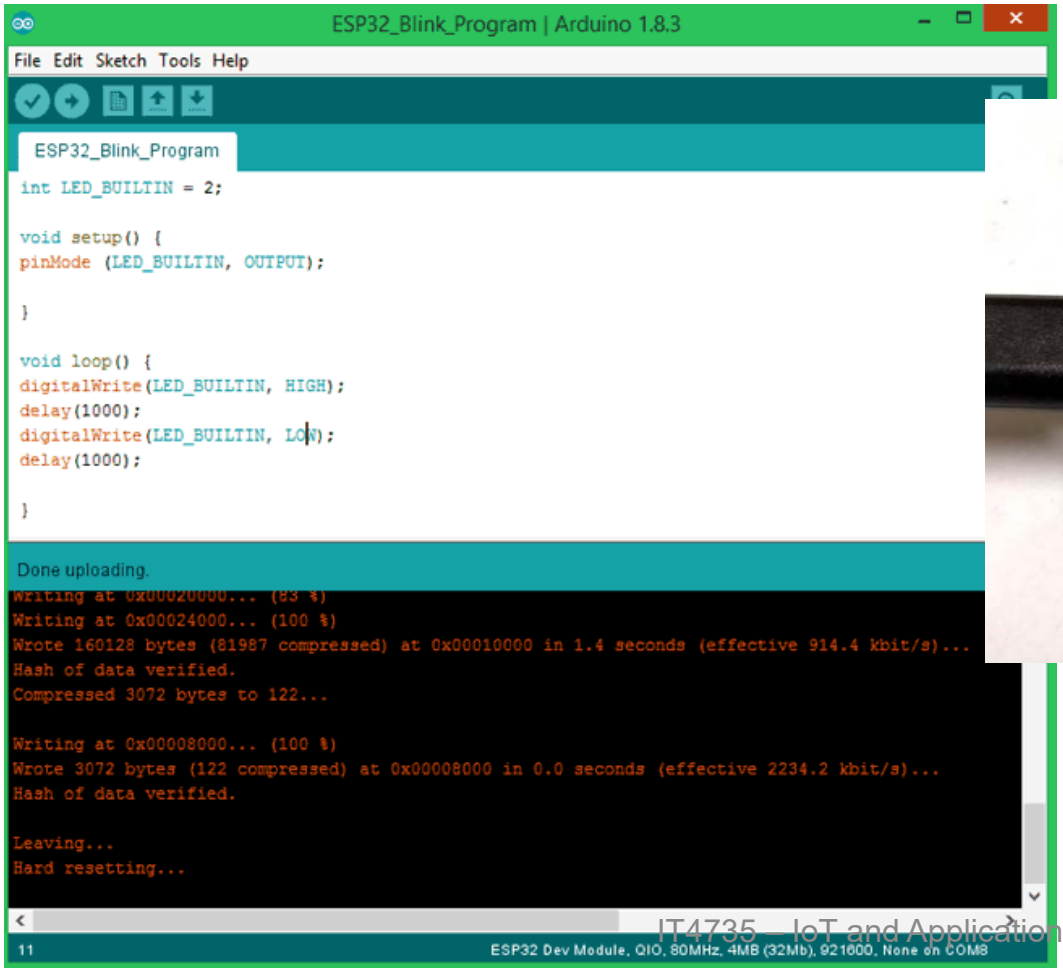
Ứng dụng LED blinky

- **STEP 5:** Let's upload the Blink Program, to check if we are able to program our ESP32 module. This program should blink the LED at an interval of 1 second.

```
int LED_BUILTIN = 2;
void setup() {
        pinMode (LED_BUILTIN, OUTPUT);
}
void loop() {
        digitalWrite(LED_BUILTIN, HIGH);        delay(1000);
        digitalWrite(LED_BUILTIN, LOW);         delay(1000);
}
```

Ứng dụng LED blinky

- **STEP 6:** To upload the code, just click on upload and you should see the Arduino console displaying the following if everything works as expected.

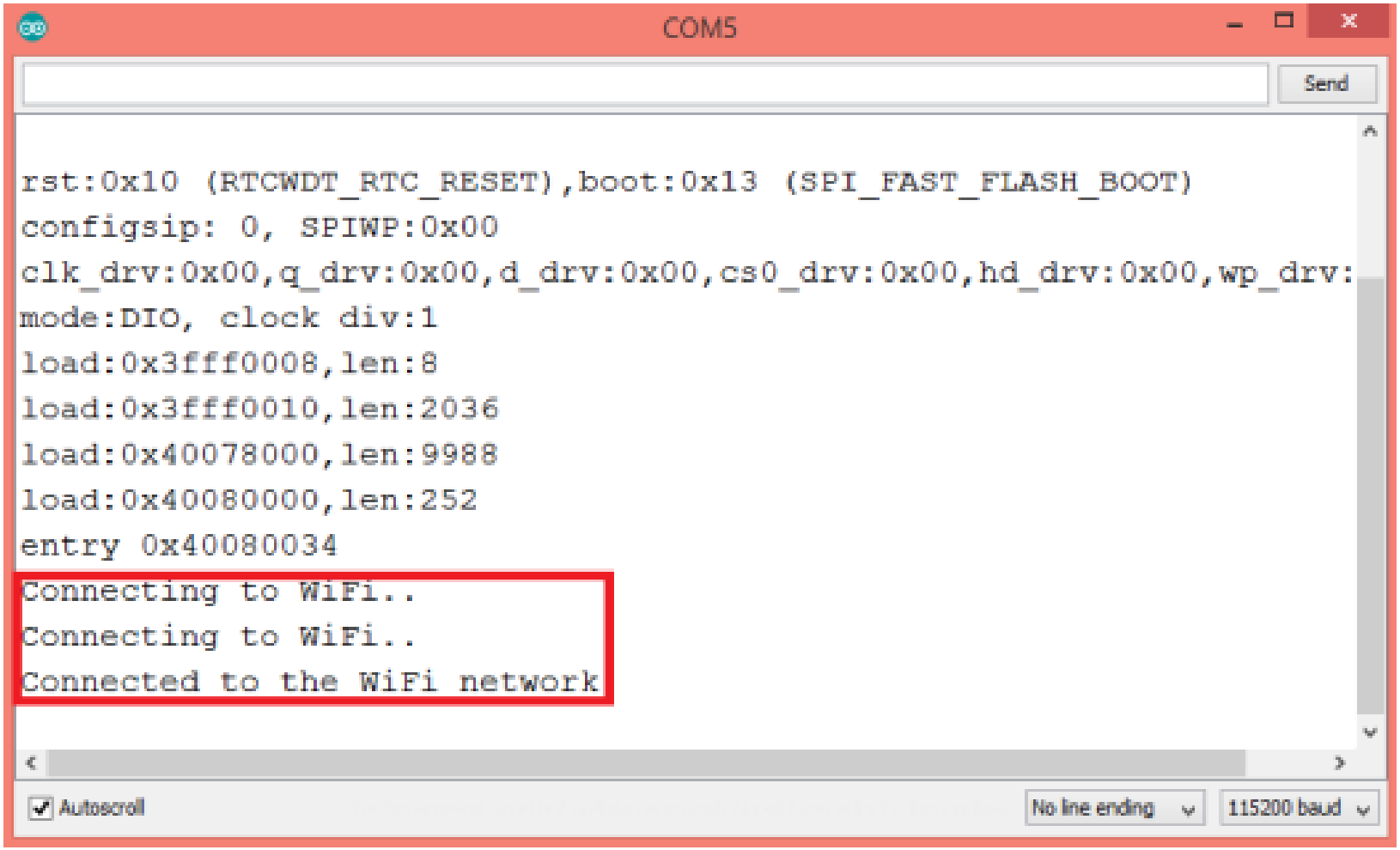


C. Ứng dụng kết nối Wifi

```
#include <WiFi.h>

const char* ssid = "yourNetworkName";
const char* password =  "yourNetworkPass";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }

  Serial.println("Connected to the WiFi network");

}

void loop() {
}
```

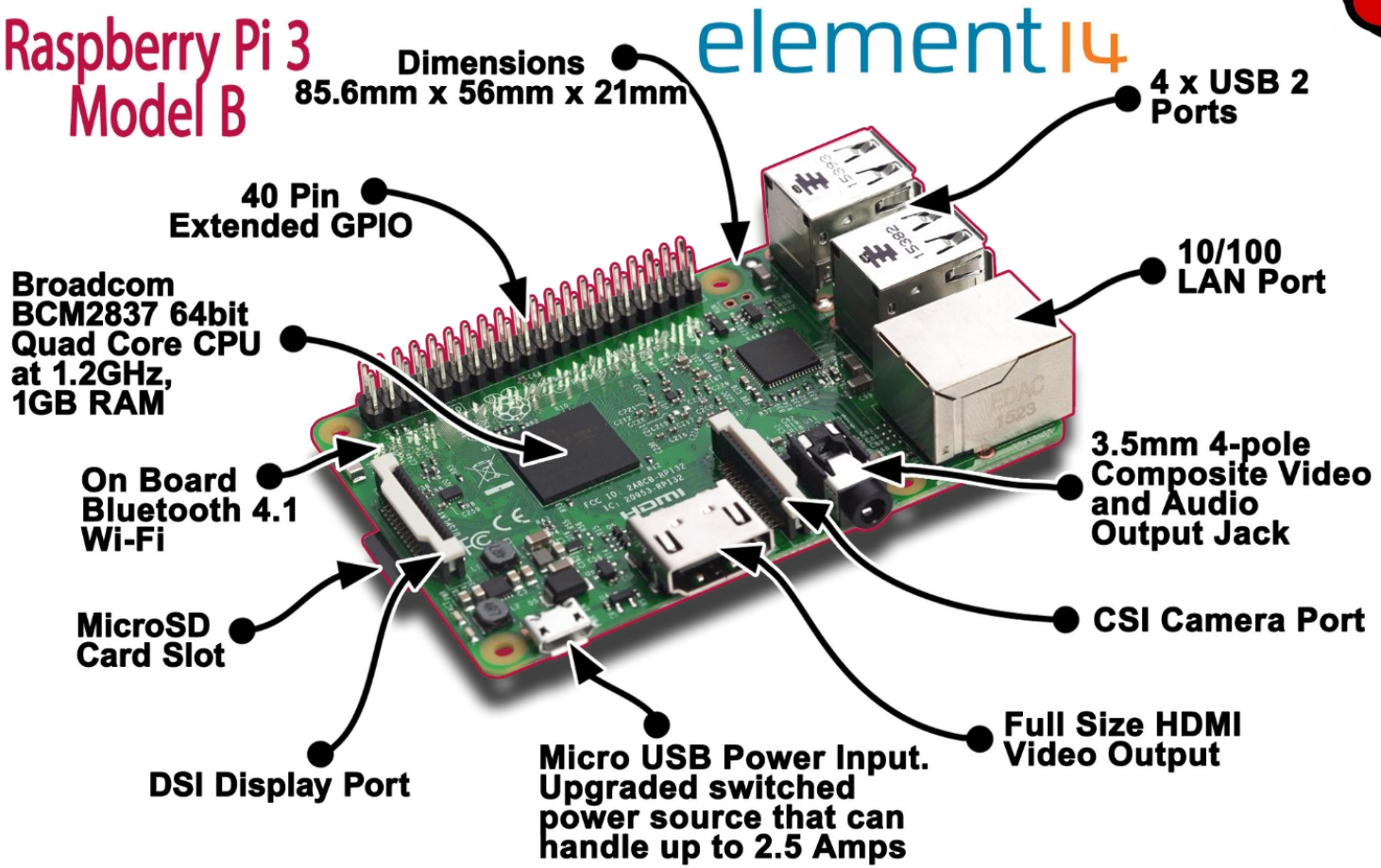
Ứng dụng kết nối Wifi

- Kết quả



3.1.3. Lập trình thiết bị Raspberry Pi

- Raspberry Pi

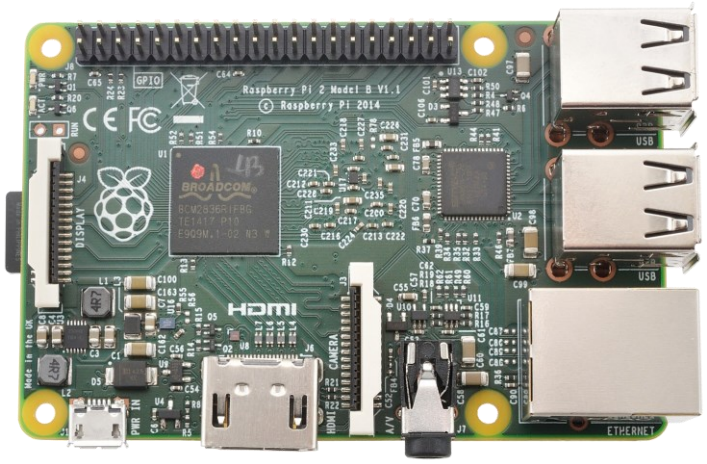


Lập trình thiết bị Raspberry Pi

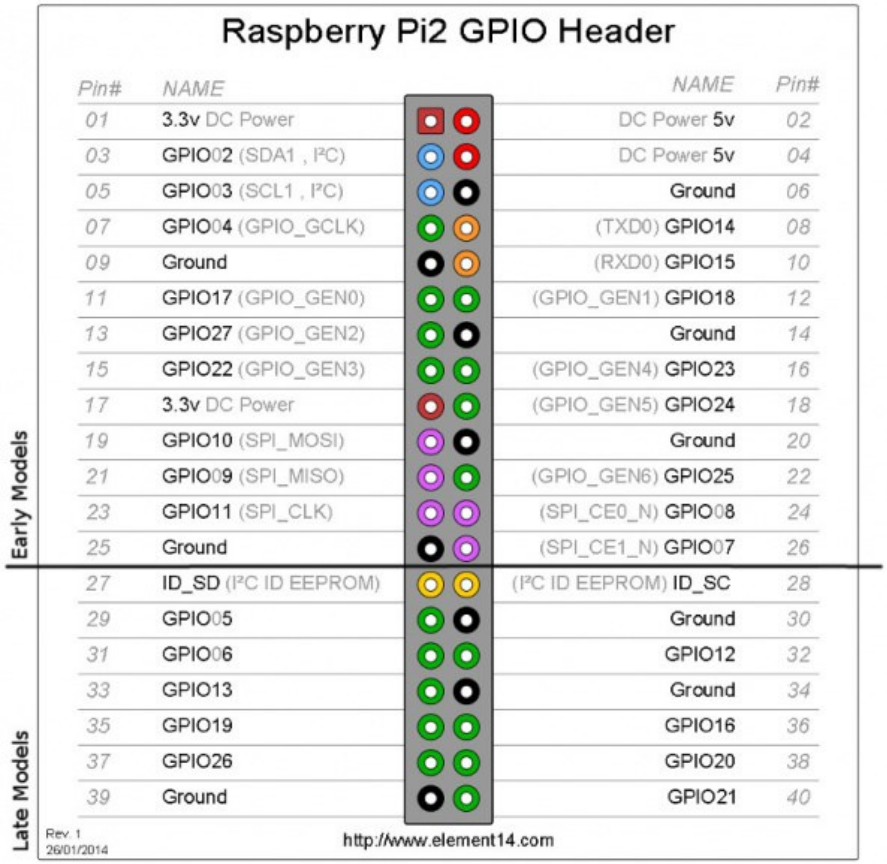
- A. Lập trình GPIO với thư viện Python RPi.GPIO
- B. Lập trình web server với Python Flask framework

A. Lập trình GPIO trên Raspberry Pi

- Lập trình GPIO



Raspberry Pi 3

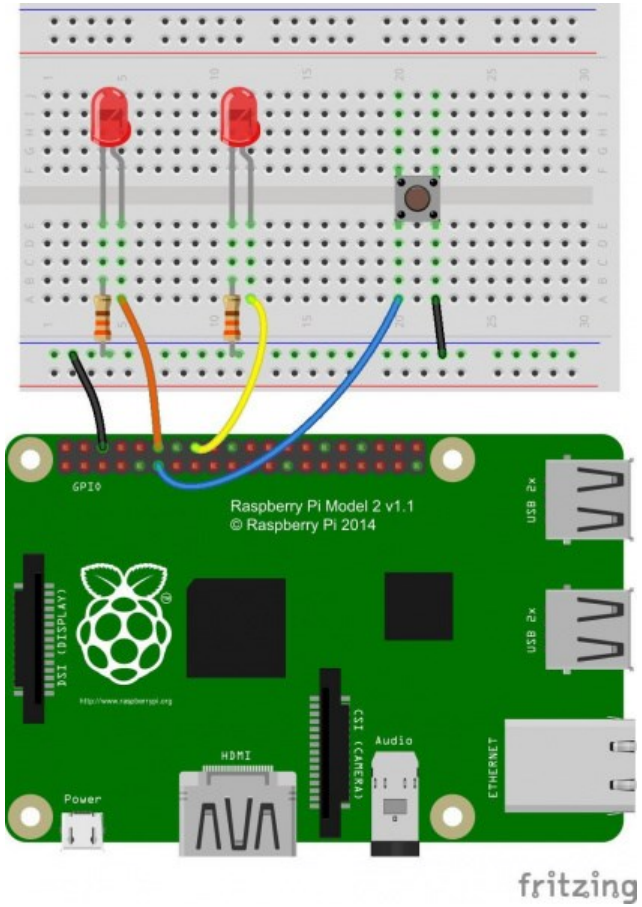


https://learn.sparkfun.com/tutorials/raspberry-gpio

Lập trình GPIO trên Raspberry Pi

Ví dụ kết nối LEDs

- **2 LEDs** are connected to the **Pi's GPIO 18 and GPIO 23** (Broadcom chip-specific numbers = BCM pin number)
- On header P1 (BOARD pin number):
  - GPIO 18 = Pin 12
  - GPIO 23 = Pin 16
- 1 **button** is connected to Broadcom **GPIO 17**, aka P1 pin 11.



Lập trình GPIO trên Raspberry Pi

- Sử dụng thư viện Python RPi.GPIO (mặc định đi kèm trong HĐH Raspbian)
- Nếu cần cài đặt: $ sudo pip install RPi.GPIO
- Các chức năng thư viện cung cấp:
  - Configure pins as input/output
  - Read inputs (high/low)
  - Set outputs (high low)
  - Wait for edge (wait for input to go high/low)
  - Pin event detection (callback on input pin change)

Python GPIO API (RPi.GPIO)

Step 1: Import RPi.GPIO package:

```
import RPi.GPIO as GPIO
print(GPIO.RPI_INFOR)
```

Step 2: Pin Numbering Declaration

- Xác định kiểu số hiệu chân (pin) muốn sử dụng:
  - GPIO.BOARD – Sử dụng số hiệu chân như header của bo mạch

```
GPIO.setmode(GPIO.BOARD)
```

  - GPIO.BCM – Sử dụng số hiệu chân theo đặc tả của Broadcom.

```
GPIO.setmode(GPIO.BCM)
```

Python GPIO API (RPi.GPIO)

Step 3: Setting a Pin Mode

- setup([pin], [GPIO.IN, GPIO.OUT])
- Example: GPIO.setup(18, GPIO.OUT)

Step 4: Read Inputs, Give Outputs

Output:

- **Digital Output:**
  - Use the GPIO.output([pin], [GPIO.LOW, GPIO.HIGH]) function.
  - For example, if you want to set pin 18 high, write:

    GPIO.output(18, GPIO.HIGH)
  - GPIO.HIGH = 1 = True
  - GPIO.LOW = 0 = False

Python GPIO API (RPi.GPIO)

Output:

- PWM (“Analog”) Output:
  - To initialize PWM, use GPIO.PWM([pin], [frequency]) function.
  - Then use pwm.start([duty cycle]) function to set an initial value.
  - For example:

```
pwm = GPIO.PWM(18, 1000)
pwm.start(50)
```

  - To change PWM duty cycle use function:

```
pwm.ChangeDutyCycle([duty cycle]).  (0-100%)
pwm.ChangeDutyCycle(75)
```

  - To turn PWM on that pin off, use the pwm.stop() command.

Python GPIO API (RPi.GPIO)

Input:

- If a pin is configured as an input, you can use the GPIO.input([pin]) function to read its value.
- The input() function will return either a True or False indicating whether the pin is HIGH or LOW.
- You can use an if statement to test this, for example

```
if GPIO.input(17):
    print("Pin 11 is HIGH")
else:
    print("Pin 11 is LOW")
```

Python GPIO API (RPi.GPIO)

Step 5: Clean up GPIO and exit

- Giải phóng chân để tránh xung đột khi sử dụng lại chân đó trong chương trình khác

  GPIO.cleanup()

- Hoặc tắt cảnh báo bằng hàm:

  GPIO.setwarnings(False).

- Sử dụng Delay trong thư viện time của python

```
import time
time.sleep([seconds])
time.sleep(0.25)
```

Ví dụ 1. Chương trình LED Blinky

```
from RPi import GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
led = 2
GPIO.setup(led, GPIO.OUT)
while True:
    GPIO.output(led, True)
    sleep(1)
    GPIO.output(led, False)
    sleep(1)
```

Ví dụ 2: Turn LED on 2s, off 1s, loop forever

```
import RPi.GPIO as GPIO
import time

def main():
    GPIO.cleanup()
    GPIO.setmode(GPIO.BOARD) # to use Raspberry Pi board pin numbers
    GPIO.setup(11, GPIO.OUT)          # set up GPIO output channel
    while True:
        GPIO.output(11, GPIO.LOW) # set RPi board pin 11 low. Turn off LED.
        time.sleep(1)
        GPIO.output(11, GPIO.HIGH) # set RPi board pin 11 high. Turn on LED.
        time.sleep(2)

main()
```

Ví dụ 3: Đọc trạng thái nút bấm trên Pin 7

```
import RPi.GPIO as GPIO
import time
butPin = 7
GPIO.setmode(GPIO.BOARD)
# normally 0 when connected 1
GPIO.setup(butPin, GPIO.IN, GPIO.PUD_DOWN)
try:
        while(True):
                print(GPIO.input(butPin))
                time.sleep(1)
except KeyboardInterrupt:
        GPIO.cleanup()
print("Exiting")
```

Đọc trạng thái nút bấm bằng ngắt (Interrupt)

- Đọc trạng thái nút bấm:
  - Phương pháp thăm dò (Polling): Chương trình phải định kỳ kiểm tra trạng thái nút bấm (chân vào)
  - Phương pháp Ngắt (Interrupts): hay “event-detect’, lắng nghe sự kiện ngắt xảy ra trên một chân vào bằng cách sử dụng:

    GPIO.add_event_detect(channel, event, callback = my_callback, bouncetime = timeinmilliseconds)
  - channel: số hiệu chân vào (input pin)
  - events: GPIO.RISING, GPIO.FALLING, GPIO.BOTH
  - my_callback: Chương trình con xử lý ngắt (chạy trong một luồng riêng)
  - Bouncetime: quãng thời gian tối thiểu giữa 2 lần phát sinh sự kiện

Đọc trạng thái nút bấm bằng ngắt (Interrupt)

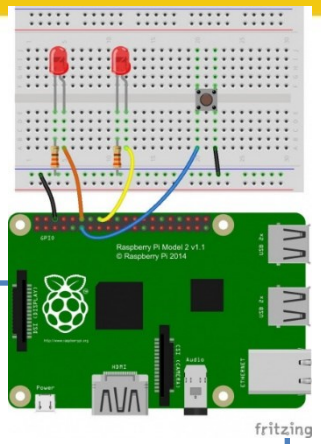
Chương trình đọc trạng thái nút bấm bằng xử lý ngắt

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time
led = 8
ledstate = True
swtch = 7
GPIO.setmode(GPIO.BOARD)
GPIO.setup(swtch,GPIO.IN, GPIO.PUD_DOWN)
GPIO.setup(led,GPIO.OUT,initial=ledstate)

#this method will be invoked
when the event occurs
```

```
def switchledstate(channel):
        global ledstate
        global led
        ledstate = not(ledstate)
        GPIO.output(led,ledstate)

# adding event detect to the switch pin
GPIO.add_event_detect(swtch, GPIO.BOTH,
switchledstate, 600)
try:
    while(True):
    #to avoid 100% CPU usage
        time.sleep(1)
except KeyboardInterrupt:
#cleanup GPIO settings before exiting
GPIO.cleanup()
```

Ví dụ tổng hợp

Ví dụ minh họa tổng hợp: đọc nút bấm, điều khiển 1 LED digital, 1 LED bằng PWM

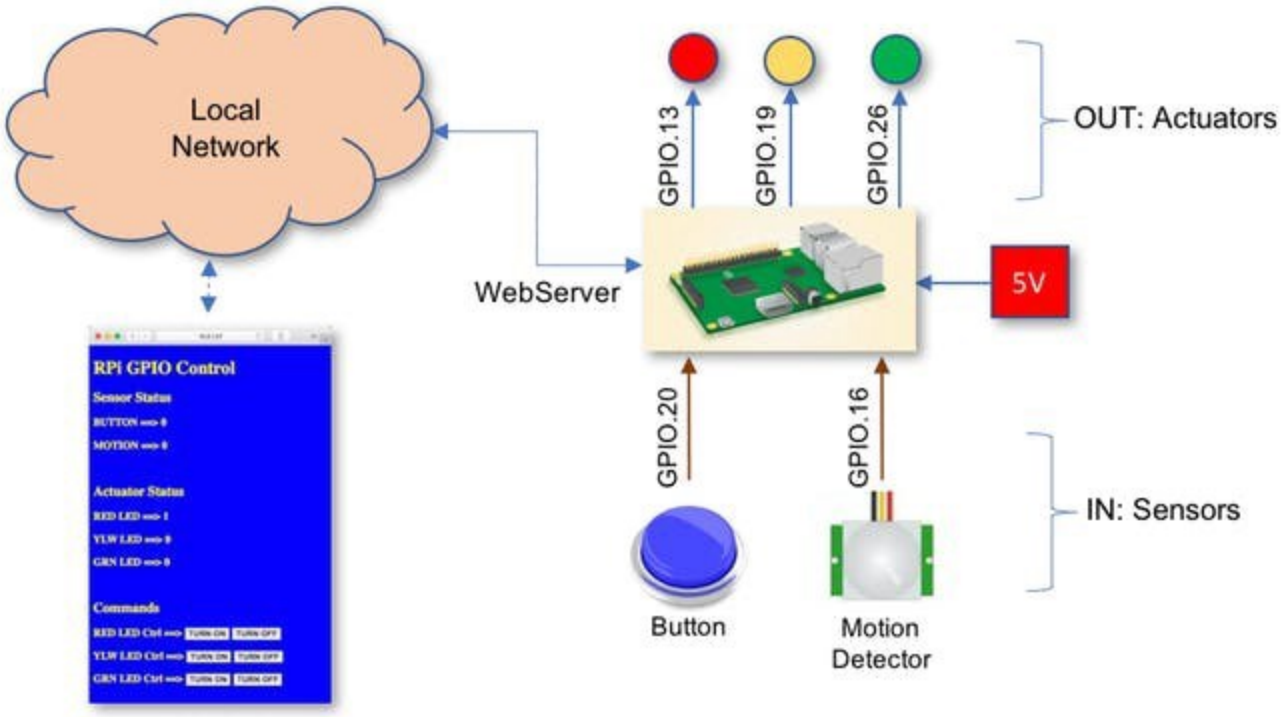



```
# External module imports
import RPi.GPIO as GPIO
import time

# Pin Definitons:
pwmPin = 18 # Broadcom pin 18 (P1 pin 12)
ledPin = 23 # Broadcom pin 23 (P1 pin 16)
butPin = 17 # Broadcom pin 17 (P1 pin 11)

dc = 95 # duty cycle (0-100) for PWM pin

# Pin Setup:
GPIO.setmode(GPIO.BCM) # Broadcom pin-numbering scheme
GPIO.setup(ledPin, GPIO.OUT) # LED pin set as output
GPIO.setup(pwmPin, GPIO.OUT) # PWM pin set as output
pwm = GPIO.PWM(pwmPin, 100)  # Initialize PWM on pwmPin
100Hz frequency
GPIO.setup(butPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# Button pin set as input w/ pull-up
```

```
# Initial state for LEDs:
GPIO.output(ledPin, GPIO.LOW)
pwm.start(dc)

print("Here we go! Press CTRL+C to exit")
try:
    while 1:
        if GPIO.input(butPin): # button is released
            pwm.ChangeDutyCycle(dc)
            GPIO.output(ledPin, GPIO.LOW)
        else: # button is pressed:
            pwm.ChangeDutyCycle(100-dc)
            GPIO.output(ledPin, GPIO.HIGH)
            time.sleep(0.075)
            GPIO.output(ledPin, GPIO.LOW)
            time.sleep(0.075)
except KeyboardInterrupt: # If CTRL+C is pressed, exit
cleanly:
    pwm.stop() # stop PWM
    GPIO.cleanup() # cleanup all GPIO
```

B. Lập trình Web server trên Raspberry Pi

- Lập trình Web server trên Pi sử dụng Python Flask framework



https://towardsdatascience.com/python-webserver-with-flask-and-raspberry-pi-398423cc6f5d

Lập trình Web server trên Raspberry Pi

- Xây dựng ứng dụng Web page cho phép người dùng điều khiển các chân GPIO trên PI thông qua internet
  - Ví dụ: điều khiển tắt/bật đèn, thiết bị trong nhà qua internet
- Sử dụng Python Flask Microframework xây dựng Pi trở thành một Local Web Server
  - Python to talk to the web server
  - HTML, CSS to make web page

Lập trình Web server trên Raspberry Pi

- **What we will learn**
  - How to Make Python-powered web server
  - How to build a basic web app with Flask and run it as a local website on your Raspberry Pi
  - How routes are used to map URLs to web pages
  - How to configure Flask and make your website accessible to other devices on your local network

RPi as a web server Using Flask and Python

- Install Flask on Raspberry Pi:

```
sudo apt-get install python3-flask
```

- Create folder for local web server on RPi, inside **/home/pi/Documents**

```
pi@raspberrypi:~ $ cd Documents
pi@raspberrypi:~/Documents $ mkdir myFirstApp
```

```
/myFirstApp
        /static
        /templates
```

RPi as a web server Using Flask and Python

- Creat a simple web server on RPi
  - Inside **/home/pi/Documents/myFirstApp,** make an **app.py** file using nano editor:

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index(): return 'Hello world'
if __name__ == '__main__':
        app.run(debug=True, host='0.0.0.0')
```

RPi as a web server Using Flask and Python

- Simple web server (app.py):
  - First import the Flask class, create an instance of this class.
  - use the route()decorator to tell Flask what URL should trigger our function index( )
  - The function is given a name(index) which is also used to generate URLs for that particular function, and returns the message we want to display in the user's browser..
  - NOTE: Note here the host='0.0.0.0' means the web app will be accessible to any device on the network.

Running web server

- Simple web server (app.py):
  - Run this web server: python3 app.py
  - Output:

```
* Running on
* Restarting with reloader
```

  - Open the Pi's web browser from the taskbar or application menu and navigate to http://127.0.0.1:5000/. You should see a white screen with the words Hello world:

The route of web pages

- This route is made up of three parts:

```
@app.route('/')
def index(): return 'Hello world'
```

  - @app.route('/'): this determines the entry point; the / means the root of the website, so just http://127.0.0.1:5000/.
  - def index(): this is the name we give to the route. Here it was called index because it's the index of the website.
  - return 'Hello world': this is the content of the web page, which is returned when the user browses the index of the website.

The route of web pages

- Create a new route:

```
@app.route('/cakes')
def cakes(): return 'Yummy cakes!'
```

Using HTML template

- Create html template for web server (place in templates folder)

```
/home/pi/Documents/myFirstApp/templates
```

```
/myFirstApp
        /static
        /templates
```

- Create file index.html:

```
<html>
<body>
<h1>Hello from a
template!</h1> </body>
</html>
```

- Modify file app.py:

```
from flask import Flask, render_template
@app.route('/')
def index(): return render_template('index.html')
```

Flask will look for index.html in a directory called templates, in the same directory as the app.py file.

Using HTML template

- Make sure the web app is still running.
- If it stop, run **python3 app.py** from myFirstApp directory.
- Reload the route in web browser (go to the base route at http://127.0.0.1:5000/) to see new HTML template being displayed.

Adding CSS to web page

- Make file style.css, place in folder static of web app

```
/myFirstApp
        /static
        /templates
```

- A simple style.css:

```
body {
        background: red;
        color: yellow;
}
```

Adding CSS to web page

- Using css in html page:

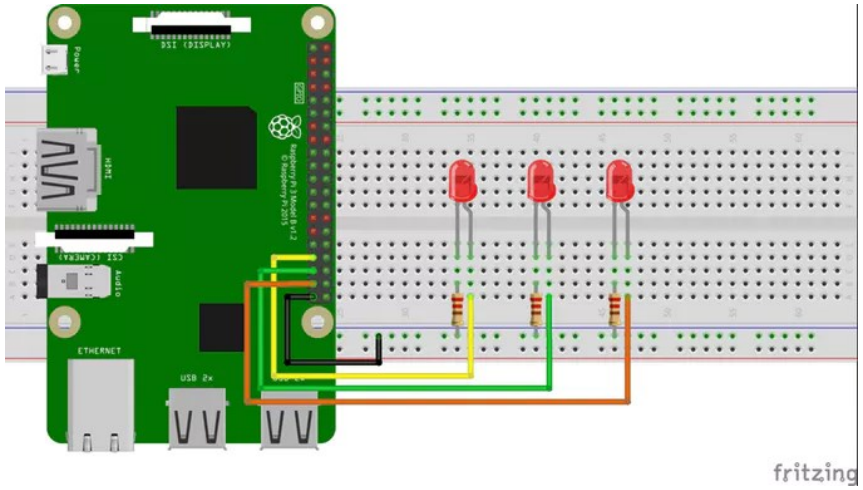
```
<html>
<head>
<link rel="stylesheet" href='../static/style.css' />
</head>
<body>
<h1>Hello from a template!</h1>
</body>
</html>
```

Browsing on other devices

- Require the IP address of the RPi
- For example: 192.168.1.3
- Browser URL = http://192.168.1.3:5000/:

Control RPi GPIO over the internet

- Control 3 RPi GPIO pins over the internet
- Use 6 buttons (turn on/off) on the web page)



Control RPi GPIO over the internet

- Python script:

```
# Raspberry Pi 3 GPIO Pins Status And Control Using
Flask Web Server and Python
import RPi.GPIO as GPIO
from flask import Flask, render_template, request
app = Flask(__name__)
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
ledRed = 13
ledYellow= 19
ledGreen= 26
ledRedSts = 0
ledYellowSts = 0
ledGreenSts = 0
GPIO.setup(ledRed, GPIO.OUT)
GPIO.setup(ledYellow,GPIO.OUT)
GPIO.setup(ledGreen, GPIO.OUT)
GPIO.output(ledRed, GPIO.LOW)
GPIO.output(ledYellow, GPIO.LOW)
GPIO.output(ledGreen, GPIO.LOW)
@app.route('/')
```

Control RPi GPIO over the internet

- Python script (cont.):

```
def index():
        ledRedSts = GPIO.input(ledRed)
        ledYellowSts = GPIO.input(ledYellow)
        ledGreenSts = GPIO.input(ledGreen)
        templateData = { 'ledRed' : ledRedSts,
        'ledYellow' : ledYellowSts, 'ledGreen' : ledGreenSts }
        return render_template('index.html', **templateData)

@app.route('/<deviceName>/<action>')
def do(deviceName, action):
        if deviceName == "ledRed":
                actuator = ledRed
        if deviceName == "ledYellow":
                actuator = ledYellow
        if deviceName == "ledGreen":
                actuator = ledGreen
        if action == "on":
                GPIO.output(actuator, GPIO.HIGH)
        if action == "off":
                GPIO.output(actuator, GPIO.LOW)
```

Control RPi GPIO over the internet

- Python script (cont.):

```
def do(deviceName, action):
…
            ledRedSts = GPIO.input(ledRed)
            ledYellowSts = GPIO.input(ledYellow)
            ledGreenSts = GPIO.input(ledGreen)
            templateData = { 'ledRed' : ledRedSts,
            'ledYellow' : ledYellowSts,
            'ledGreen' : ledGreenSts }
            return render_template('index.html', **templateData
)

if __name__ == "__main__":
            app.run(host = '0.0.0.0', debug=True)
```

Control RPi GPIO over the internet

- HTML template

```
<!DOCTYPE html> <html>
 <head> <title> GPIO Control Web App</title>
<link rel="styleSheet" href="/static/style.css"/>
</head>
<body>
<img src="/static/dogMeme.jpg" width="300px" height="300px" alt="Because You Normie"/>
<h1>Actuators</h1><br>
<h2>Status</h2>
<h3> RED LED --> {{ledRed}}</h3>
 <h3> YELLOW LED --> {{ledYellow}}</h3>
<h3> GREEN LED --> {{ledGreen}}</h3><br>
<h2>Led Control</h2>
<h3> RED LED CNTRL ==> <a href ="/ledRed/on" class="button">TURN ON</a>
<a href="/ledRed/off" class = "button">TURN OFF</a></h3>
<h3> YELLOW LED CNTRL ==> <a href="/ledYellow/on" class="button">TURN ON</a>
<a href="/ledYellow/off" class="button">TURN OFF</a></h3>
<h3> GREEN LED CNTRL ==> <a href="/ledGreen/on" class="button">TURN ON</a>
 <a href="/ledGreen/off" class="button">TURN OFF</a></h3>
 </body>
 </html>
```

Control RPi GPIO over the internet

- HTML template

```
<!DOCTYPE html> <html>
 <head> <title> GPIO Control Web App</title>
<link rel="styleSheet" href="/static/style.css"/>
</head>
<body>
<img src="/static/dogMeme.jpg" width="300px" height="300px" alt="Because You Normie"/>
<h1>Actuators</h1><br>
<h2>Status</h2>
<h3> RED LED --> {{ledRed}}</h3>
 <h3> YELLOW LED --> {{ledYellow}}</h3>
<h3> GREEN LED --> {{ledGreen}}</h3><br>
<h2>Led Control</h2>
<h3> RED LED CNTRL ==> <a href ="/ledRed/on" class="button">TURN ON</a>
<a href="/ledRed/off" class = "button">TURN OFF</a></h3>
<h3> YELLOW LED CNTRL ==> <a href="/ledYellow/on" class="button">TURN ON</a>
<a href="/ledYellow/off" class="button">TURN OFF</a></h3>
<h3> GREEN LED CNTRL ==> <a href="/ledGreen/on" class="button">TURN ON</a>
 <a href="/ledGreen/off" class="button">TURN OFF</a></h3>
 </body>
 </html>
```

Control RPi GPIO over the internet

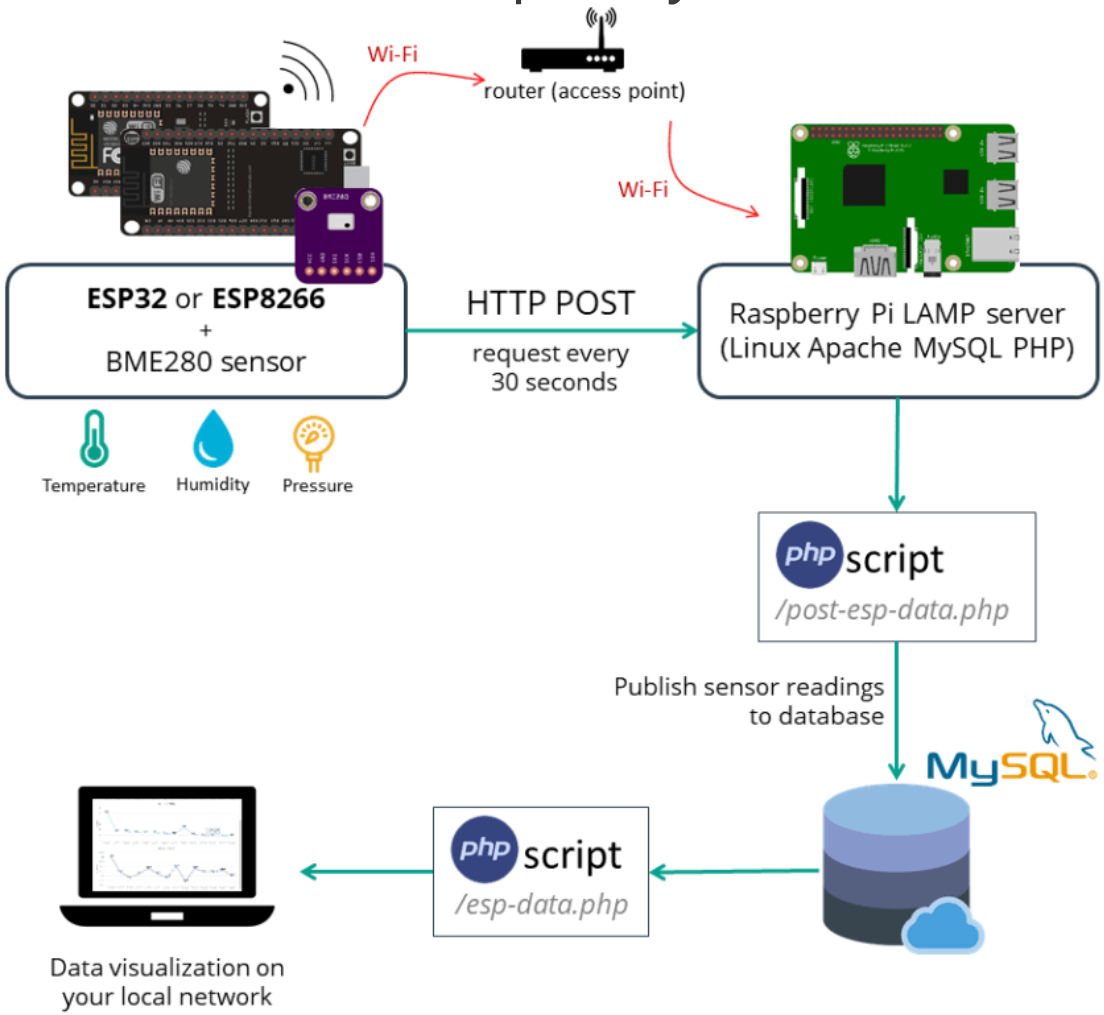
- CSS code

```
body {
        text-align: center;
        background: #54ff9f;
        color: #fff5ee;
}
.button{
        font: bold 16px Arial;
        background-color:#EEEEEE;
        padding: 1px;
        border: 1px solid #CCCCCC;
}
```

Reference:
https://pythonhosted.org/energenie/examples/web/

3.1.4. Truyền dữ liệu ESP32 – Raspberry Pi

- ESP32 Publish Data to Raspberry Pi LAMP Server



https://randomnerdtutorials.com/esp32-esp8266-raspberry-pi-lamp-server/

3.2. Dịch vụ IoT trên máy chủ, nền tảng đám mây

- Các dịch vụ cần thiết:
  - Lưu trữ dữ liệu (database)
  - Trình bày dữ liệu (representation), theo dõi giám sát (Dashboard)
  - Phân tích dữ liệu
- 2 giải pháp:
  - Xây dựng phần mềm trên máy chủ
  - Sử dụng các nền tảng IoT cloud services

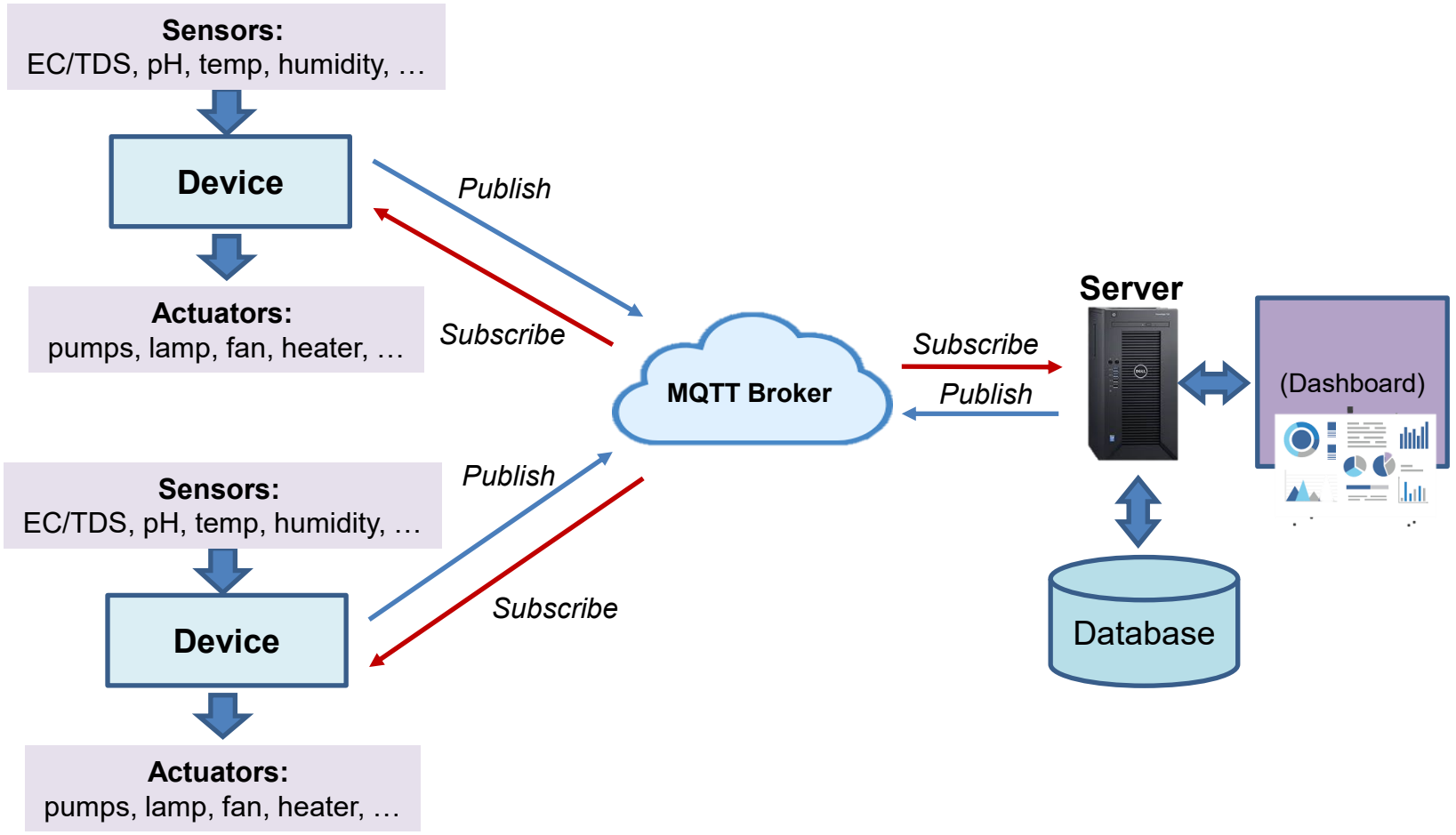
3.3. Thiết kế một hệ thống ứng dụng IoT

- **Thiết kế một hệ thống ứng dụng IoT trong thủy canh**

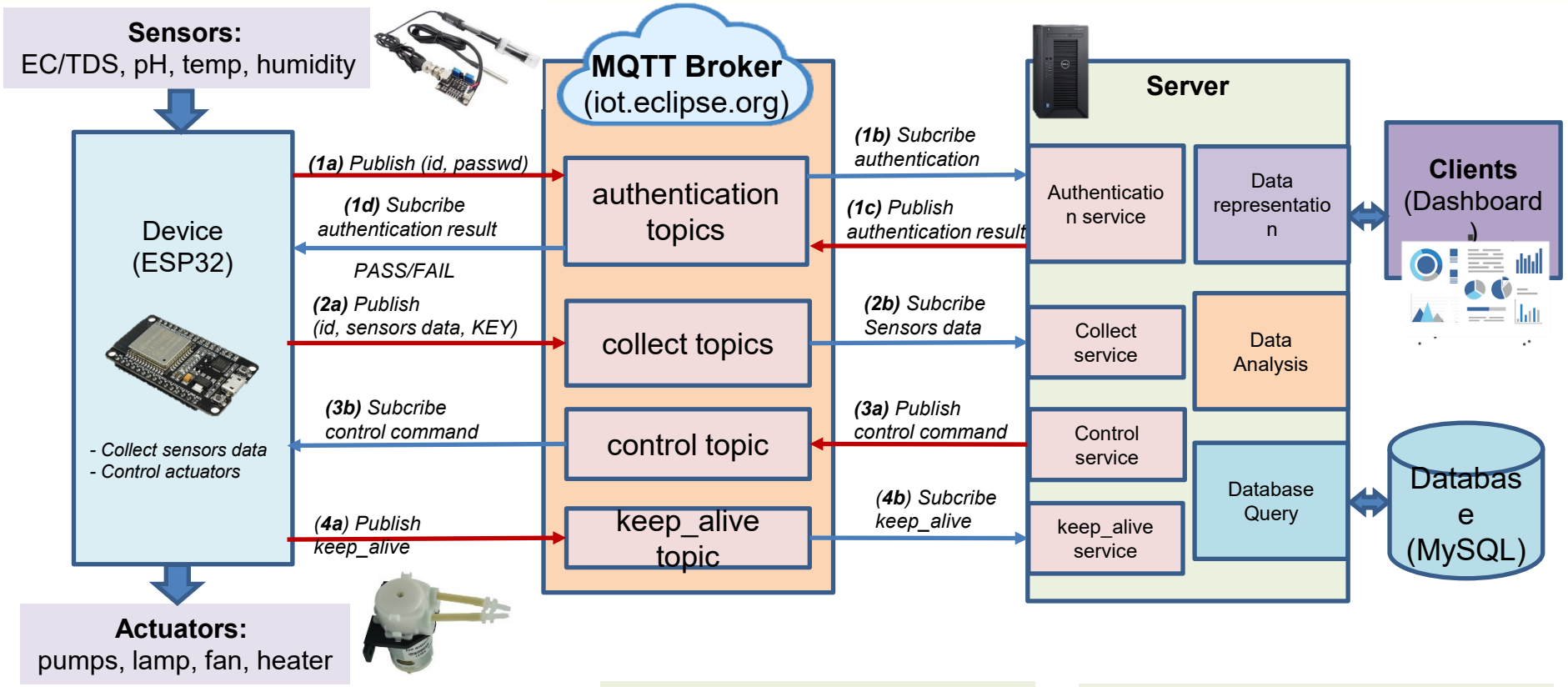
- **Collecting** hydroponic environment parameters (pH, TDS/EC, temp, …)
- **Monitoring** via web-based dash board, mobile application
- **Controlling** actuators (pumps, lamp, fan, heater, …)
- **Analyzing** data

Thiết kế tổng quan

- Hệ thống ứng dụng IoT trong thủy canh



Thiết kế kiến trúc hệ thống

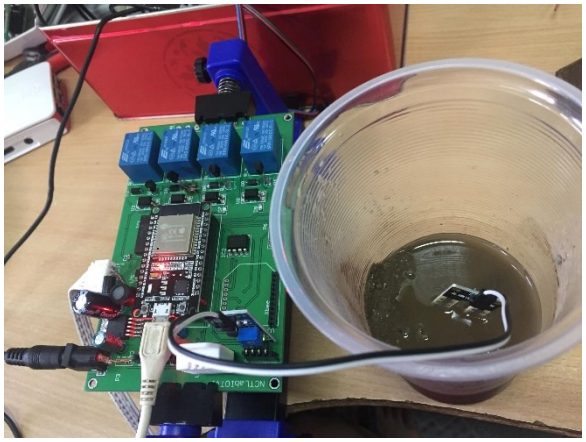


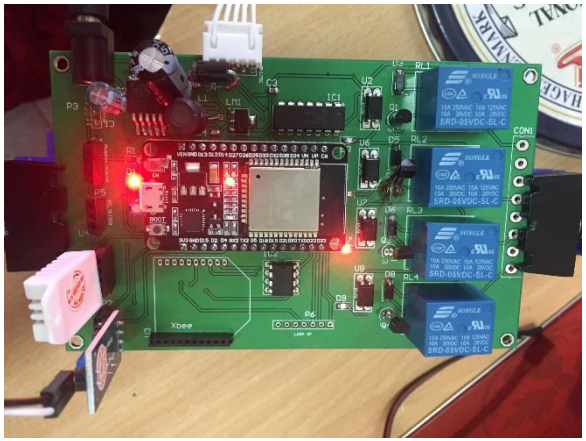
Part 1: Device (ESP32):
1) Firmware: collect task, control task (Using light sleep or power save mode)
2) Design hardware (pcb, box)

Part 2: mqtt services on Server
1) Authentication service
2) Collect (sensors data) service
3) Control service
4) keep_alive service

Part 3: Web server app
- Data representation (views, graphs)
- Data management (database query)
- Control GUI (command to ESP32)

Thiết kế thiết bị thu thập điều khiển dùng ESP32



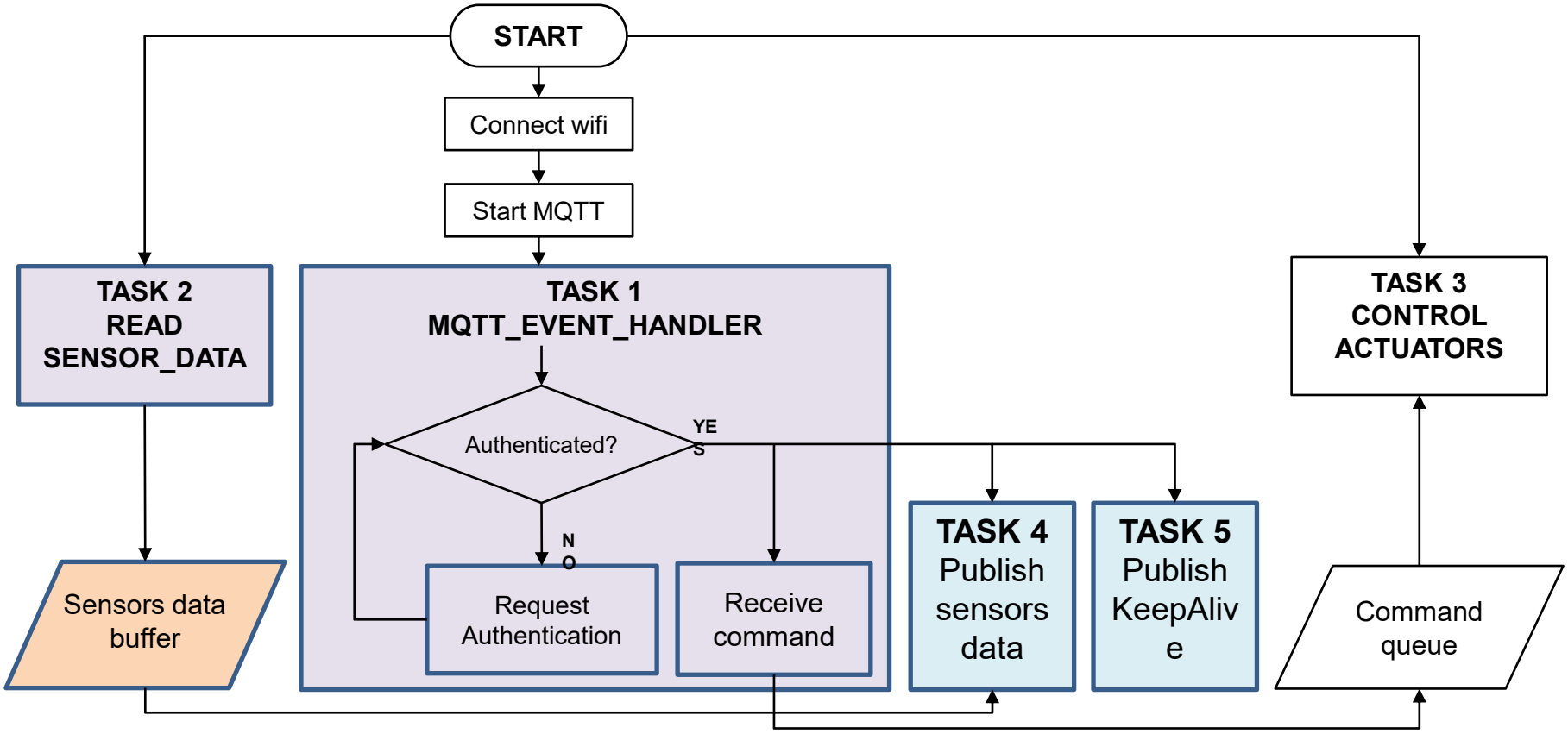


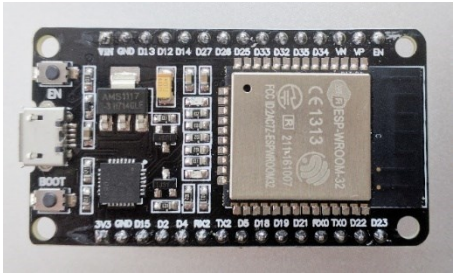
Prototype of collecting/controlling devices

- Design hardware prototype of devices for collecting and controlling
  - ESP32 computer

  (SoC with Wifi, FreeRTOS)
  - Interface to sensors (pH, TDS/EC)
  - Interface to actuator by relay

Kiến trúc Firmware trên thiết bị ESP32





Lập trình Firmware trên thiết bị ESP32

- Task 1. Giao tiếp qua MQTT với server
- Task 2. Đọc dữ liệu từ sensor
- Task 3. Publish dữ liệu đến server

(Tham khảo: Mã nguồn)