Khoa Mạng máy tính và Truyền thông - UIT

3

CÔNG NGHỆ
INTERNET OF THINGS
HIỆN ĐẠI

Làm quen với HTTP và MQTT trong mô hình IoT

Lưu hành nội bộ

A. MỤC TIÊU

- Giới thiệu giao thức **HTTP** và **MQTT** được sử dụng trong mô hình Internet of Things.
- Tiếp cận với **Wemos D1** và **Raspberry Pi** để thực hành với các phương thức truyền thông dữ liệu.
- Cài đặt công cụ VNC để điều khiển, sử dụng thiết bị từ xa.
- Thực hiện một số kịch bản trên các thiết bị.

B. GIỚI THIỆU

1. Một số giao thức được sử dụng trong mô hình IoT

HTTP

HTTP – Hyper Text Transfer Protocol là giao thức được thiết kế và hoạt động theo kiểu Client - Server. Giao tiếp giữa client và server dựa vào một cặp là HTTP Request - HTTP Response. Khi một client đưa ra request, server trả lời bằng các response ngay sau đó.

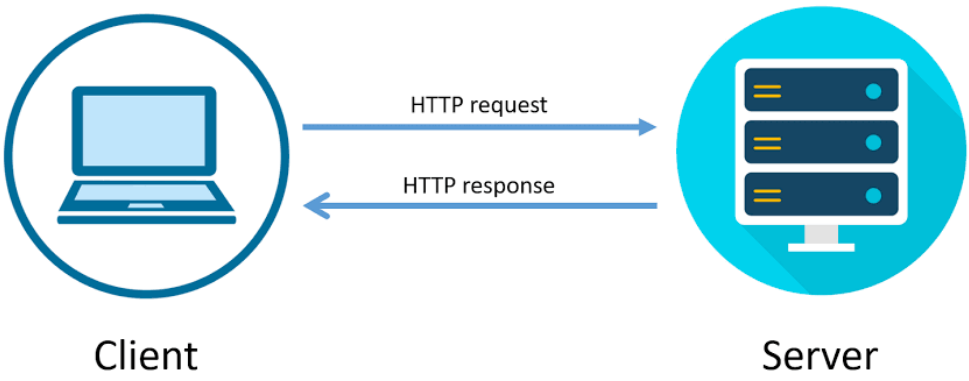
HTTP Client là một phần mềm (trình duyệt, hoặc bất kỳ máy khác) thiết lập kết nối tới server nhằm mục đích gửi một hoặc nhiều thông điệp HTTP Request.

HTTP Server là một chương trình chấp nhận các kết nối để phục vụ các HTTP Request bằng cách gửi các HTTP Response.

HTTP Request là thông điệp được gửi từ HTTP Client để yêu cầu server thực hiện một công việc nào đó. HTTP Request sẽ có hai thành phần chính là Request Method và Request URI:

- Request Method có hai phương thức chính thường được sử dụng là GET và POST.
- Request URI xác định tài nguyên để áp dụng cho các request.

HTTP Response là thông điệp phản hồi từ HTTP Server sau khi nhận được HTTP Request, phản hồi này có thể chứa dữ liệu được yêu cầu hoặc một thông báo.

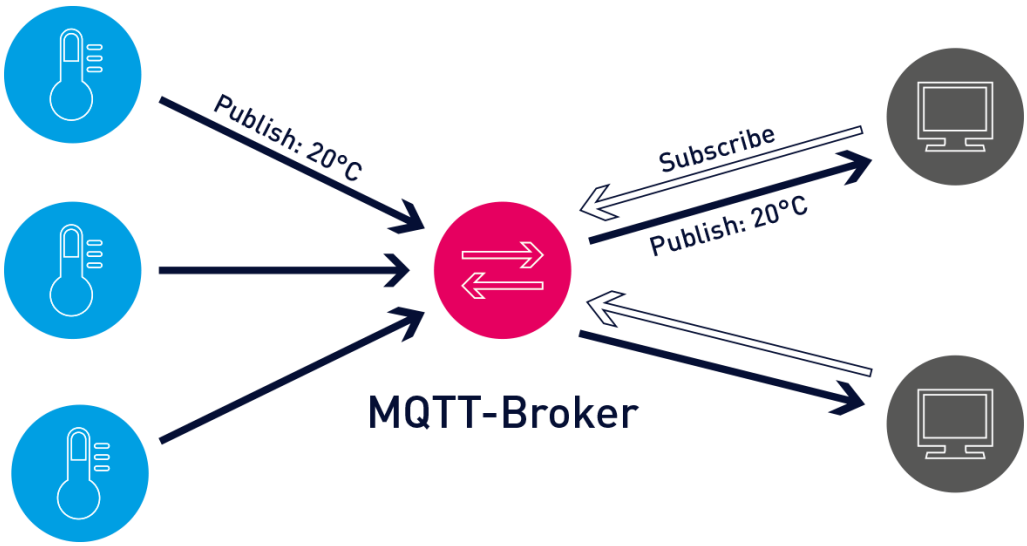


Hình 1. Mô tả mô hình HTTP

MQTT

MQTT là một giao thức truyền dữ liệu giữa nhiều thiết bị thông qua một trạm trung gian (gọi là Broker – là một application chạy trên máy tính Server/Cloud).

Về cơ bản, các thiết bị IoT sẽ đóng vai trò client và đăng ký với Broker dữ liệu nó muốn gửi lên hoặc dữ liệu nó muốn nhận được. Nhiệm vụ của Broker là thiết bị trung gian nhận dữ liệu truyền lên từ các thiết bị client và gửi dữ liệu đến các client muốn nhận tương ứng.



Hình 2. Mô tả mô hình MQTT

Các thiết bị client có thể đăng ký nhận dữ liệu hoặc ngừng nhận dữ liệu từ Broker trong runtime tùy yêu cầu của ứng dụng. Ngoài ra các thiết bị client gửi dữ liệu cũng không cần đăng ký trước loại dữ liệu mà nó sẽ gửi. Do đó, có thể nói giao thức MQTT là mô hình hình

sao và cho phép nhiều thiết bị trao đổi dữ liệu khá linh hoạt cả đối với thiết bị gửi dữ liệu và thiết bị nhận dữ liệu.

Luồng làm việc của giao thức MQTT có thể được mô tả một cách đơn giản như sau:

- Kết nối đến một Broker.
- Đăng ký loại dữ liệu muốn nhận (nếu thiết bị có nhu cầu nhận dữ liệu).
- Gửi dữ liệu lên Broker (khi thiết bị có dữ liệu và muốn gửi tới các thiết bị khác).

Sau khi nắm được nguyên lý của giao thức MQTT, chúng ta sẽ giới thiệu các khái niệm cơ bản của MQTT như sau:

- Client: là thiết bị IoT muốn gửi/nhận dữ liệu trong mạng.
- Broker: là thiết bị trung gian nhận dữ liệu từ các client muốn gửi và gửi dữ liệu đó tới các client muốn nhận.
- Topic: tượng trưng cho loại dữ liệu mà các thiết bị client gửi/nhận thông qua MQTT.

Topic là một chuỗi UTF-8 text tượng trưng cho tên loại dữ liệu. Ví dụ của topic như: "example/topic1", "demo_esp/sensor/accel", "demo_esp/sensor/gyro",... . Trong topic, chúng ta có thể phân chia cấp dùng ký tự '/' và từ đó có thể đăng ký nhận nhiều loại dữ liệu dùng các ký tự đặc biệt như '#' hoặc '+' như sau:

- '+': đăng ký nhiều topic chỉ khác nhau một vị trí phân cấp trong text của topic. Ví dụ như "demo/sensor/+" sẽ đăng ký nhận dữ liệu từ các topic như "demo/sensor/accel", "demo/sensor/gyro",...
- '#': đăng ký nhiều topic khác nhau nhiều vị trí phân cấp trong text của topic. Ví dụ như "demo/#" sẽ đăng ký nhận dữ liệu từ tất cả topic bắt đầu bằng "demo/" như "demo/test", "demo/test/test1", "demo/test1",...

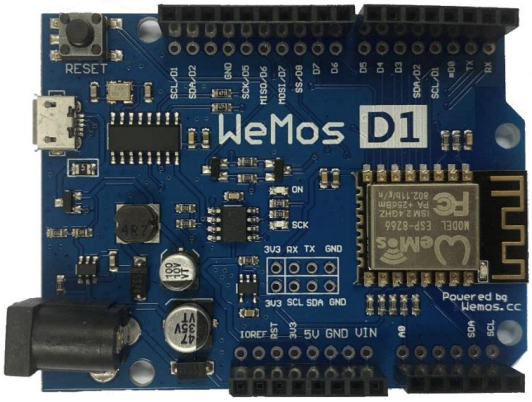
- Publish: là bước gửi dữ liệu từ một thiết bị client đến Broker. Trong bước này, thiết bị IoT sẽ xác định topic (là loại dữ liệu muốn gửi) và giá trị của topic đó.

- Subscribe: là bước đăng ký nhận dữ liệu từ Broker của một thiết bị Client. Trong bước này, thiết bị IoT sẽ xác định loại dữ liệu mà nó muốn nhận. Khi Broker nhận được loại dữ liệu này từ một thiết bị client khác, nó sẽ gửi dữ liệu này tới thiết bị client đã đăng ký nhận.
- Unsubscribe: là bước thông báo với Broker là thiết bị Client không muốn tiếp tục nhận dữ liệu nữa.
- QoS: https://code.google.com/archive/p/mqtt4erl/wikis/QualityOfServiceUseCases.wiki
- Retain: Nếu RETAIN được set bằng 1, khi gói tin được publish từ client, Broker PHẢI lưu trữ lại gói tin với QoS, và nó sẽ được gởi đến bất kỳ client nào subscribe cùng kênh trong tương lai. Khi một client kết nối tới Broker và subscribe, nó sẽ nhận được gói tin cuối cùng có RETAIN = 1 với bất kỳ topic nào mà nó đăng ký trùng. Tuy nhiên, nếu Broker nhận được gói tin mà có QoS = 0 và RETAIN = 1, nó sẽ huỷ tất cả các gói tin có RETAIN = 1 trước đó. Và phải lưu gói tin này lại, nhưng hoàn toàn có thể huỷ bất kỳ lúc nào.

2. Các thiết bị liên quan

Wemos D1

Wemos D1 là một board mạch có thiết kế giống như Arduino UNO, sử dụng vi điều khiển ESP8266 hỗ trợ kết nối Wifi chuẩn b/g/n giúp cho việc phát triển các ứng dụng có liên quan tới WiFi/Internet.



Hình 3. Wemos D1

Các chân GPIO: các chân trên Wemos D1 được đánh số khác so với Arduino Uno. Điều này có nghĩa là để bật chân D1 (tương tự với các chân khác) trên Wemos ta phải sử dụng một số khác không phải là một như Arduino Uno. Trong thư viện của board Wemos đã định nghĩa các hằng số cho các chân với số tương ứng với chân đó.

Wemos	ESP8266
D0	GPIO16
D1	GPIO5
D2	GPIO4
D3	GPIO0
D4	GPIO2
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15
TX	GPIO3
RX	GPIO1

Như vậy ví dụ để khai báo cho chân D5 trên Wemos D1 chúng ta có hai cách như sau:

```
pinMode(14, OUTPUT);
pinMode(D5, OUTPUT);
```

Raspberry Pi

Raspberry Pi là một máy tính có kích thước nhỏ gọn, giá thành rẻ có thể cắm vào màn hình máy tính và sử dụng chuột, bàn phím tiêu chuẩn. Nó là một thiết bị nhỏ có khả năng cho phép mọi người khám phá máy tính và học cách lập trình bằng các ngôn ngữ như Scratch và Python. Nó có khả năng làm được hầu hết những gì chúng ta cần trên một máy tính thông thường như truy cập Internet, xem video, soạn thảo, chơi trò chơi, làm các server nhỏ trong gia đình,…

Hơn nữa, Raspberry Pi có khả năng tương tác với thế giới bên ngoài thông qua các chân điều khiển.

Hình 4. Raspberry Pi

3. Phần mềm

VNC Viewer

VNC Viewer là một phần mềm dùng để điều khiển máy tính, thiết bị từ xa. VNC sẽ giúp người dùng hiển thị được màn hình của máy tính hoặc hệ thống ở xa ngay trên máy tính local của người dùng và có thể điều khiển, thao tác qua kết nối mạng.

Hình 5. VNC Viewer

Khi kết hợp với Raspberry Pi, Raspberry Pi lúc này đóng vai trò là VNC Server. Nhờ vào ứng dụng của VNC Viewer, thay vì phải sử dụng chuột, bàn phím, màn hình,... kết nối vào Raspberry Pi để sử dụng thì chúng ta có thể tận dụng các phần cứng này trên máy tính cá nhân của người dùng và thông qua VNC Viewer để điều khiển, sử dụng Raspberry Pi.

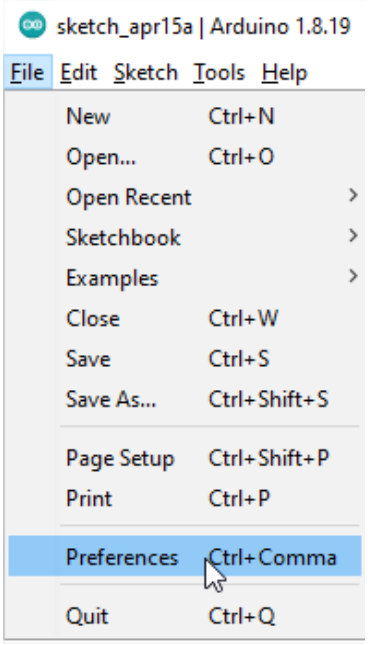
C. THỰC HÀNH

1. Tương tác với đèn LED thông qua HTTP Server

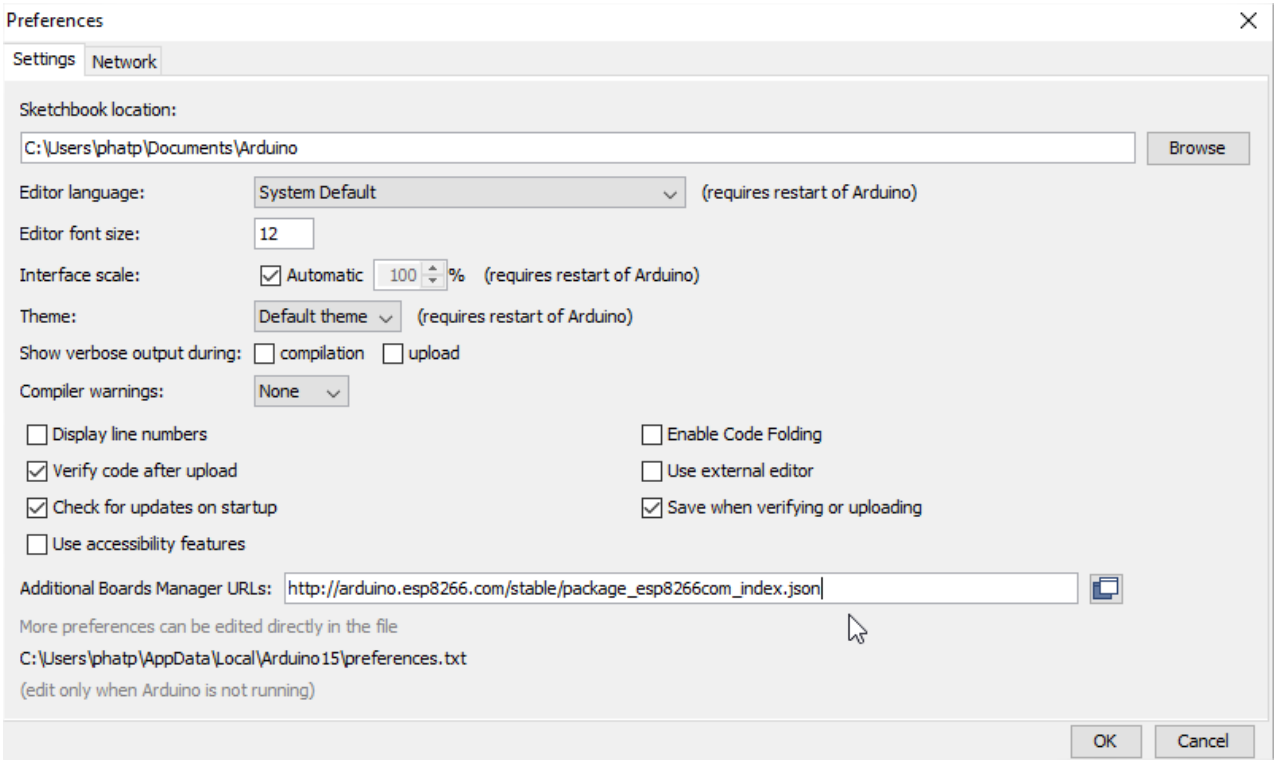
Mô tả: Tại phần thực hành này, sinh viên thực hiện dựng 1 HTTP Server trên Wemos D1 và tiến hành điều khiển bật tắt đèn LED thông qua giao diện web.

* *Để có thể biên dịch và nạp code cho Wemos D1 chúng ta cần thêm board mạch Wemos D1 và trong Arduino IDE.*

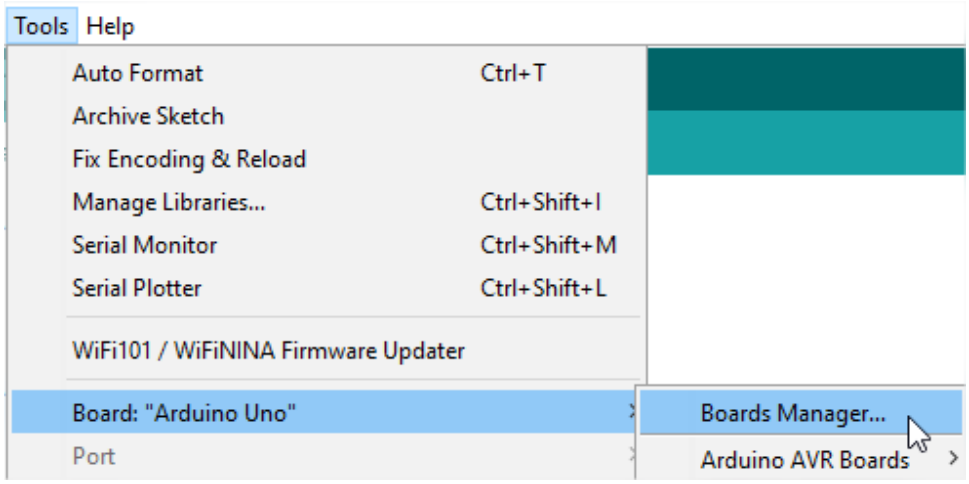
- **Bước 1:** Mở Arduino IDE chọn **File → Preferences**.



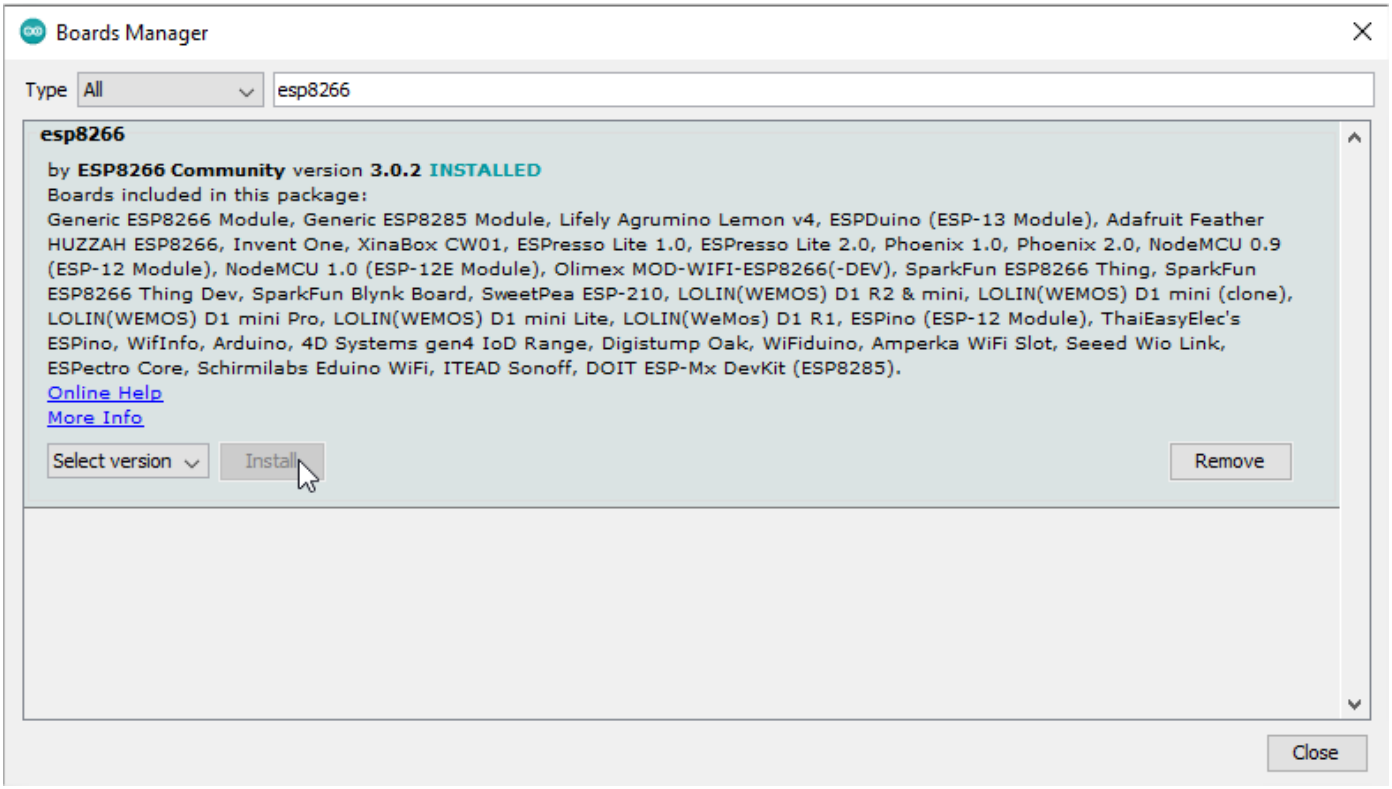
- **Bước 2:** Tại cửa sổ **Preferences**, thêm đường dẫn: http://arduino.esp8266.com/stable/package_esp8266com_index.json tại phần **Additional Boards Manager URLs.**



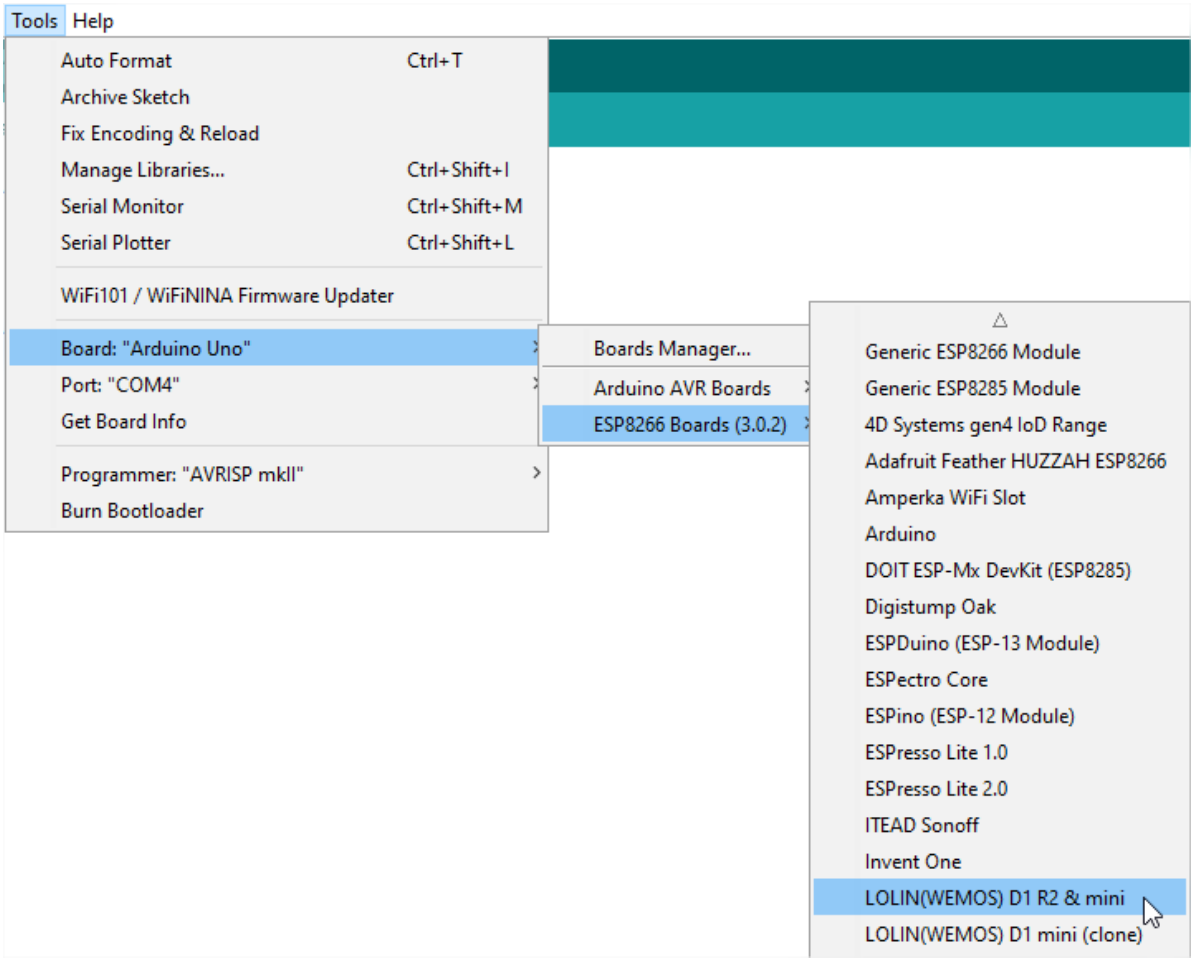
- **Bước 3:** Tiếp tục vào **Tools → Board → Boards Manager...**



- **Bước 4:** Tìm kiếm **esp8266** và chọn **Install**.



- **Bước 5:** Sau khi đã cài đặt board mạch, vào **Tools** → **Board** → Chọn **LOLIN(WEMOS) D1 R2 & mini**.



- **Bước 6:** Sau khi đã chọn **Board** và **Port**, ta tiến hành biên dịch và nạp code vào board mạch.

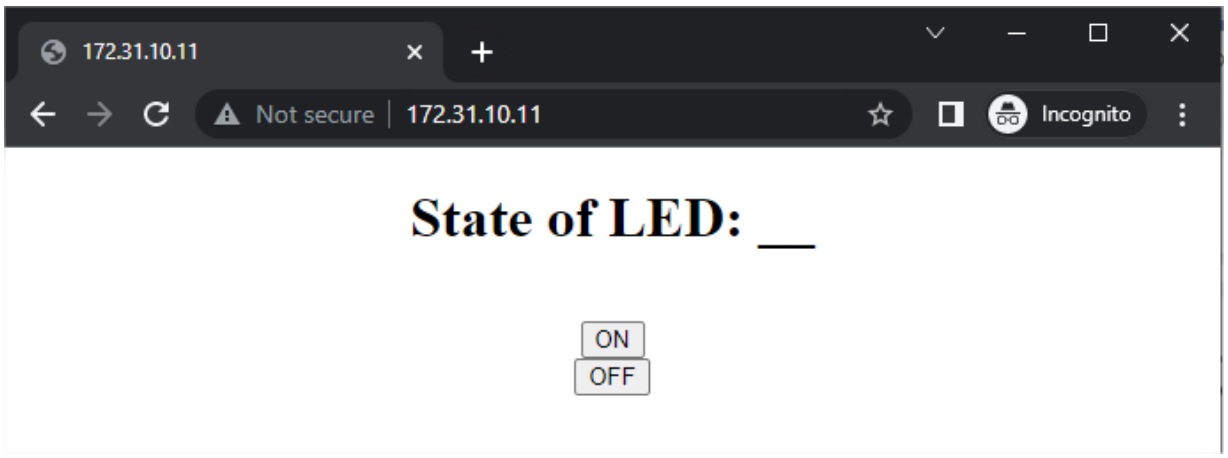
```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

//ESP Web Server Library to host a web page
#include <ESP8266WebServer.h>

//---------------------------------------------------------------
//Our HTML webpage contents in program memory

//Main
const char MAIN_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<body>
<center>
<h1>State of LED: __</h1><br>
```

```
<a href="ledOn"><button>ON</button></a><br>
<a href="ledOff"><button>OFF</button></a><br>
</center>

</body>
</html>
)=====";

//Led on
const char LEDON_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<body>
<center>
<h1>State of LED: ON</h1><br>
<a href="ledOn"><button>ON</button></a><br>
<a href="ledOff"><button>OFF</button></a><br>
</center>

</body>
</html>
)=====";

//Led off
const char LEDOFF_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<body>
<center>
<h1>State of LED: OFF</h1><br>
<a href="ledOn"><button>ON</button></a><br>
<a href="ledOff"><button>OFF</button></a><br>
</center>
</body>
</html>
)=====";
//---------------------------------------------------------------
//On board LED Connected to GPIO2

#define LED LED_BUILTIN

//SSID and Password of your WiFi router
const char* ssid = "UiTiOt-E3.1";
const char* password = "UiTiOtAP";

//Declare a global object variable from the ESP8266WebServer class.
```

```
ESP8266WebServer server(80); //Server on port 80

//==============================================================
// This routine is executed when you open its IP in browser
//==============================================================
void handleRoot() {
 Serial.println("You called root page");
 String html = MAIN_page; //Read HTML contents
 server.send(200, "text/html", html); //Send web page
}

void handleLEDon() {
 Serial.println("LED on page");
 digitalWrite(LED,LOW); //LED is connected in reverse
 String html = LEDON_page; //Read HTML contents
 server.send(200, "text/html", html); //Send ADC value only to client ajax request
}

void handleLEDoff() {
 Serial.println("LED off page");
 digitalWrite(LED,HIGH); //LED off
 String html = LEDOFF_page; //Read HTML contents
 server.send(200, "text/html", html); //Send ADC value only to client ajax request
}
//==============================================================
//                  SETUP
//==============================================================
void setup(){
  Serial.begin(115200);

  WiFi.begin(ssid, password);     //Connect to your WiFi router
  Serial.println("");

  //Onboard LED port Direction output
  pinMode(LED,OUTPUT);
  //Power on LED state off
  digitalWrite(LED,HIGH);

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  //If connection successful show IP address in serial monitor
  Serial.println("");
```

```
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());  //IP address assigned to your ESP

  server.on("/", handleRoot);      //Which routine to handle at root location. This is
display page
  server.on("/ledOn", handleLEDon); //as Per  <a href="ledOn">, Subroutine to be called
  server.on("/ledOff", handleLEDoff);

  server.begin();                  //Start server
  Serial.println("HTTP server started");
}
//==============================================================
//                     LOOP
//==============================================================
void loop(){
  server.handleClient();          //Handle client requests
}
```

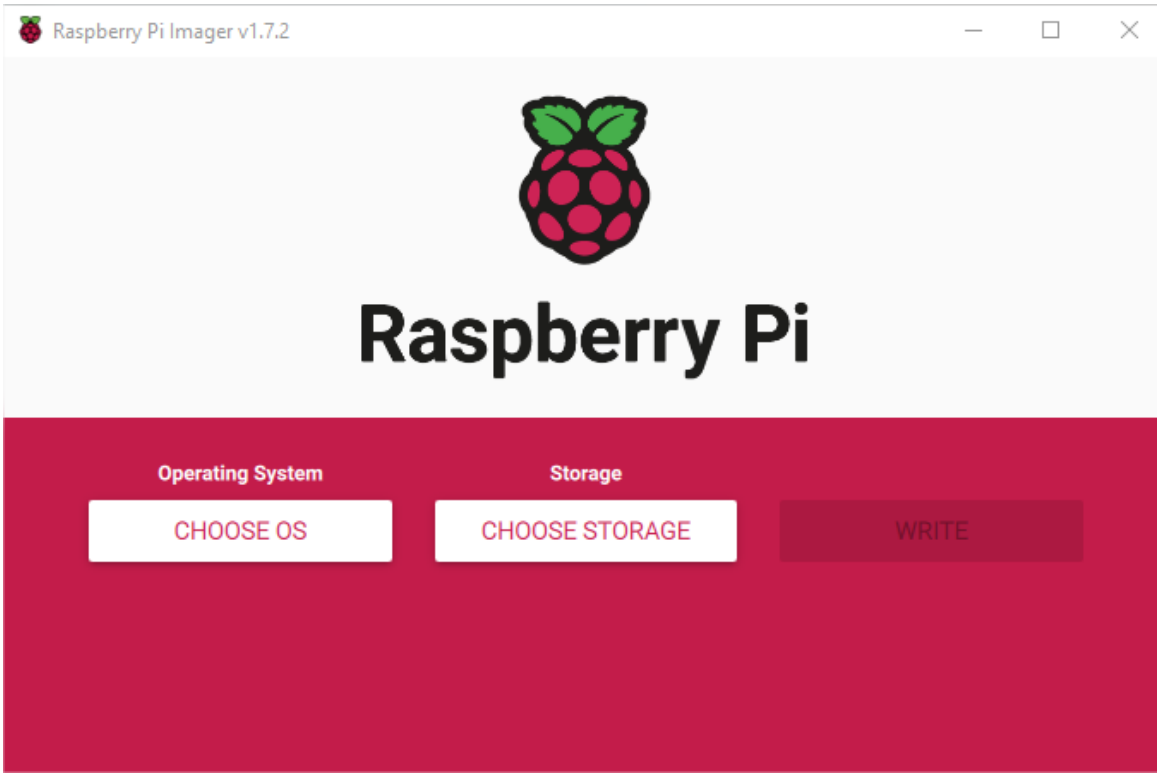
- **Bước 7:** Chọn vào các nút **ON** và **OFF** trên giao diện và quan sát kết quả trên board mạch.



2. Chuẩn bị các cài đặt để sử dụng Raspberry Pi

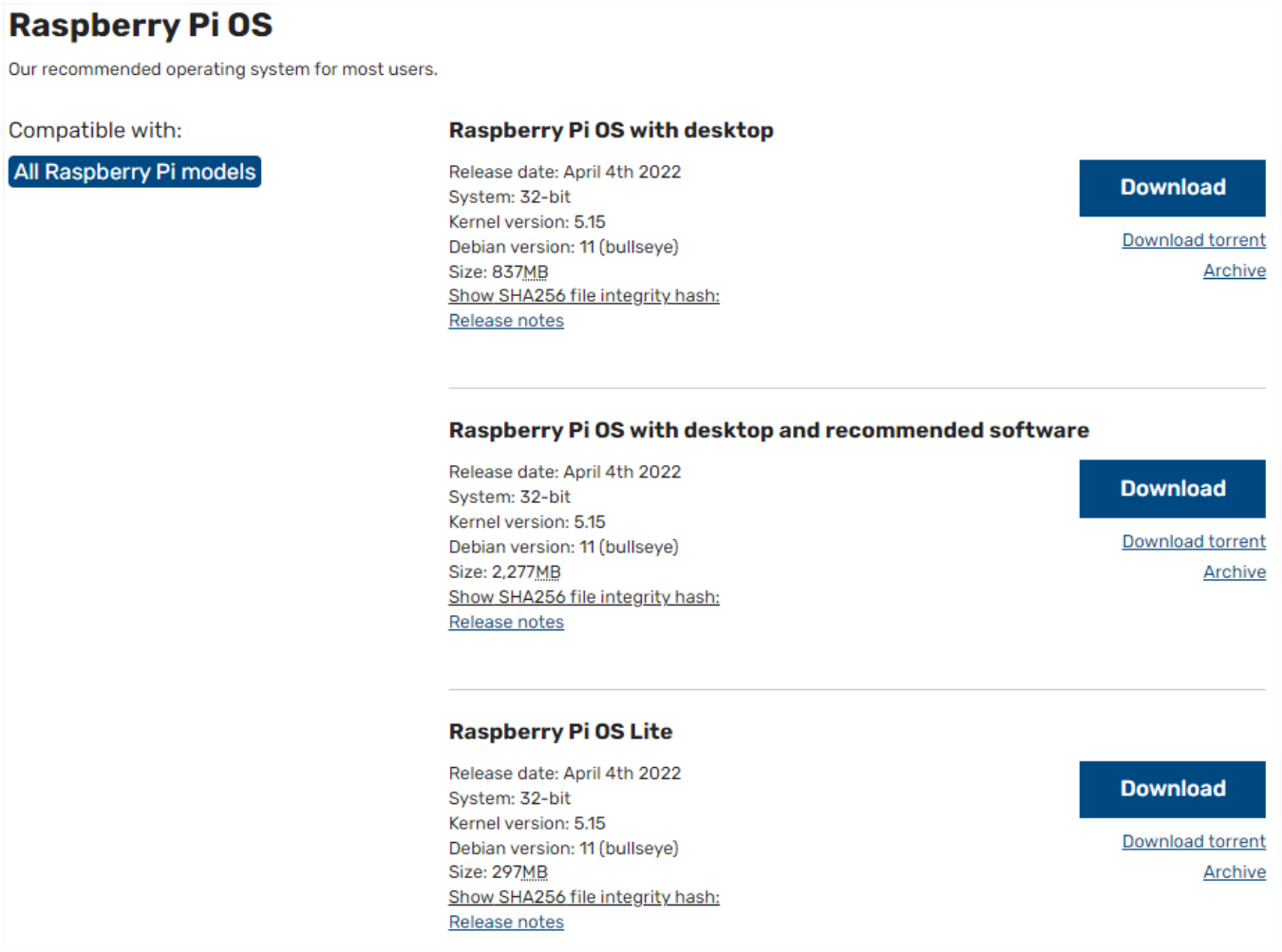
Mô tả: Tại phần thực hành này, sinh viên tiến hành cài đặt hệ điều hành, thiết đặt SSH, kết nối Wifi và thiết lập VNC trên Raspberry Pi để sử dụng thiết bị từ xa.

- **Bước 1:** Tải phần mềm Raspberry Pi Imager tại https://www.raspberrypi.com/software/

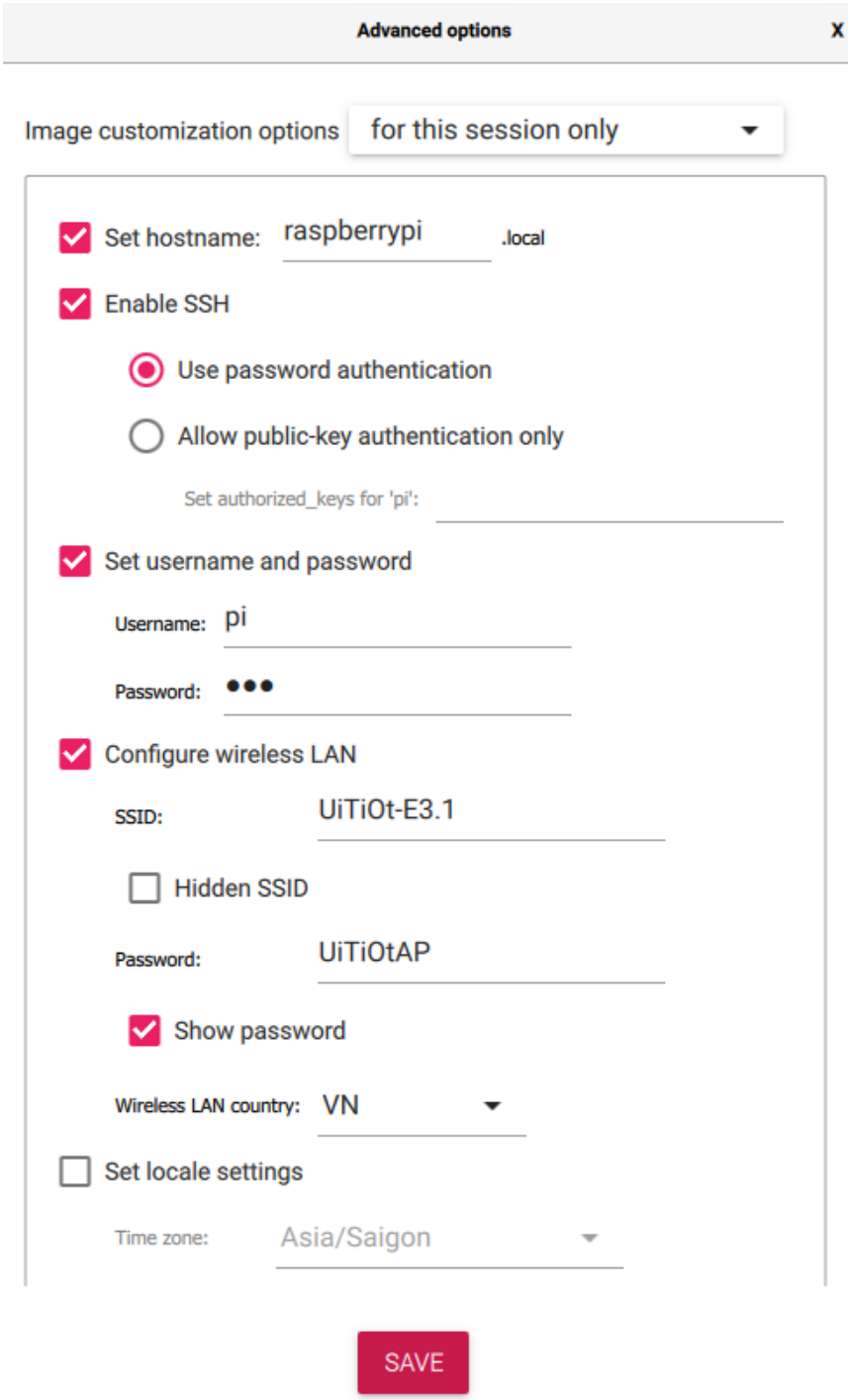


- **Bước 2:** Tải hệ điều hành cho Raspberry Pi tại: https://www.raspberrypi.com/software/operating-systems/

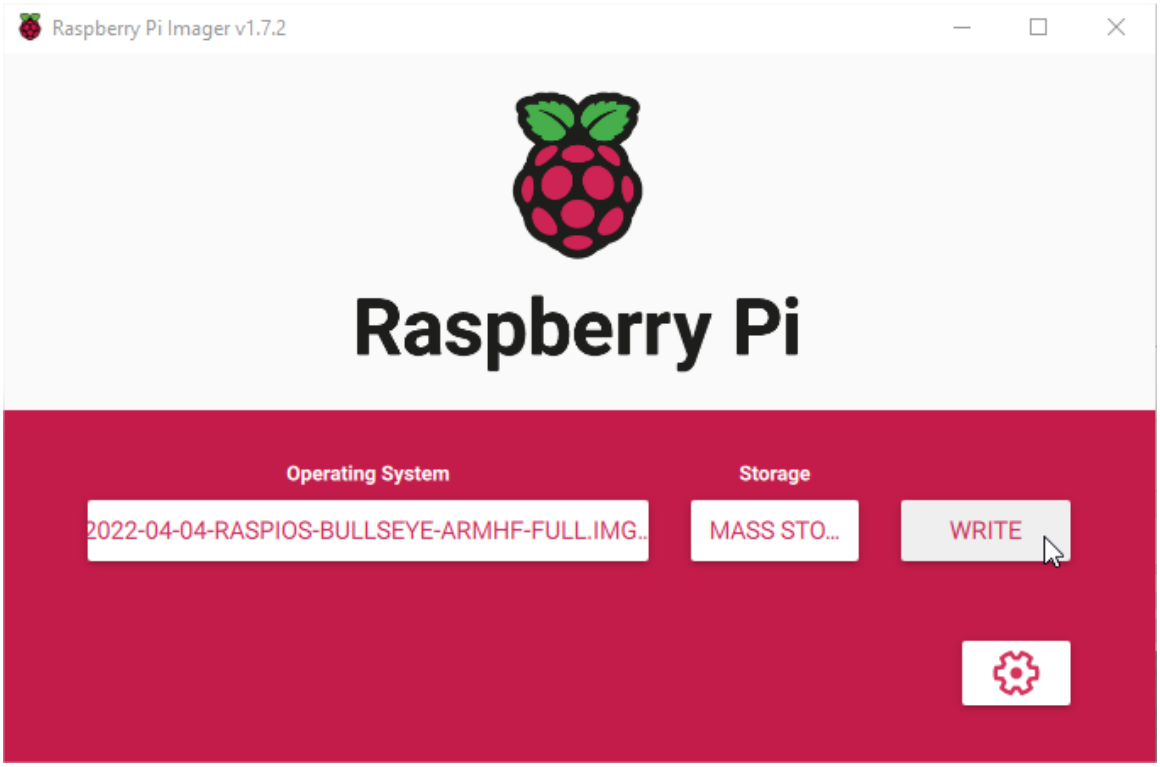
- **Bước 3:** Tại đây, ta có nhiều lựa chọn cho hệ điều hành của Raspberry Pi, tùy theo nhu cầu của người dùng để sử dụng hợp lý. Trong phần thực hành này, sinh viên sử dụng bản **Raspberry Pi OS with desktop and recommended software**:



- **Bước 4:** Sử dụng đầu đọc thẻ MicroSD, thẻ nhớ và phần mềm Raspberry Pi Imager để cài đặt phiên bản hệ điều hành vừa tải lên thẻ nhớ.
- **Bước 5:** Sử dụng tổ hợp phím **Ctrl + Shift + X** để mở hộp thoại **Advanced options**.
- **Bước 6:** Tiến hành đặt hostname, cho phép SSH, tạo user với username và password, kết nối Wifi cho Raspberry Pi.

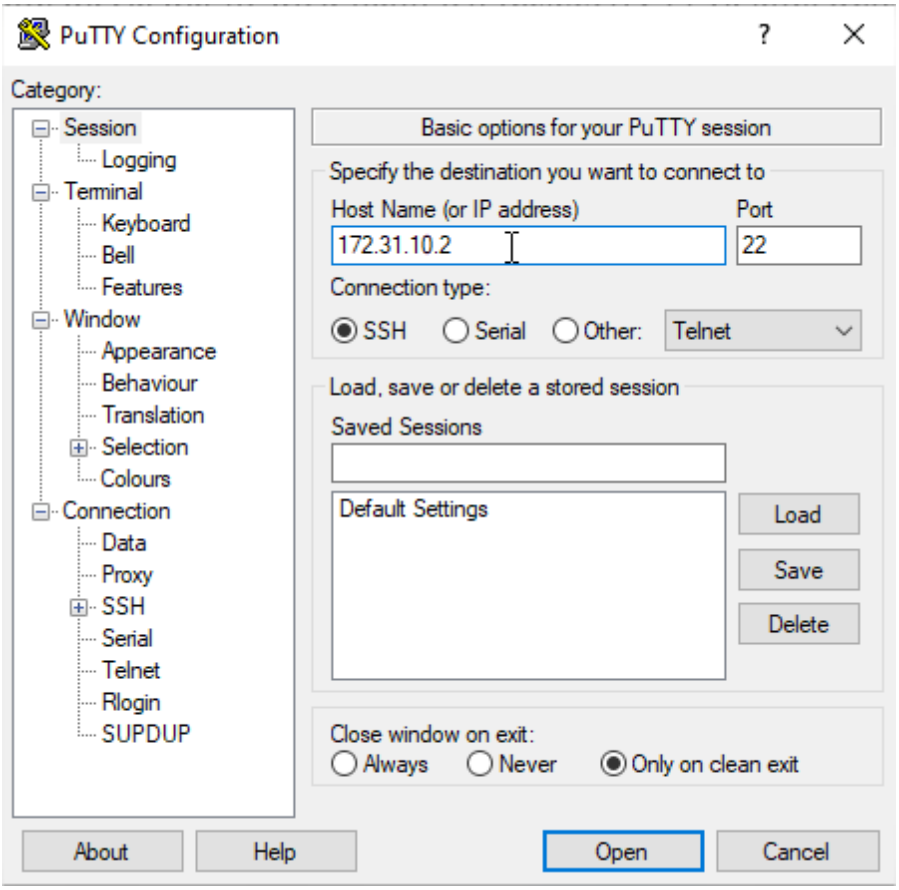


- **Bước 7:** Sau khi đã chọn **Operating System** (phiên bản hệ điều hành vừa tải ở **Bước 2**) và **Storage** (thẻ nhớ) chọn **WRITE** để quá trình cài đặt diễn ra.



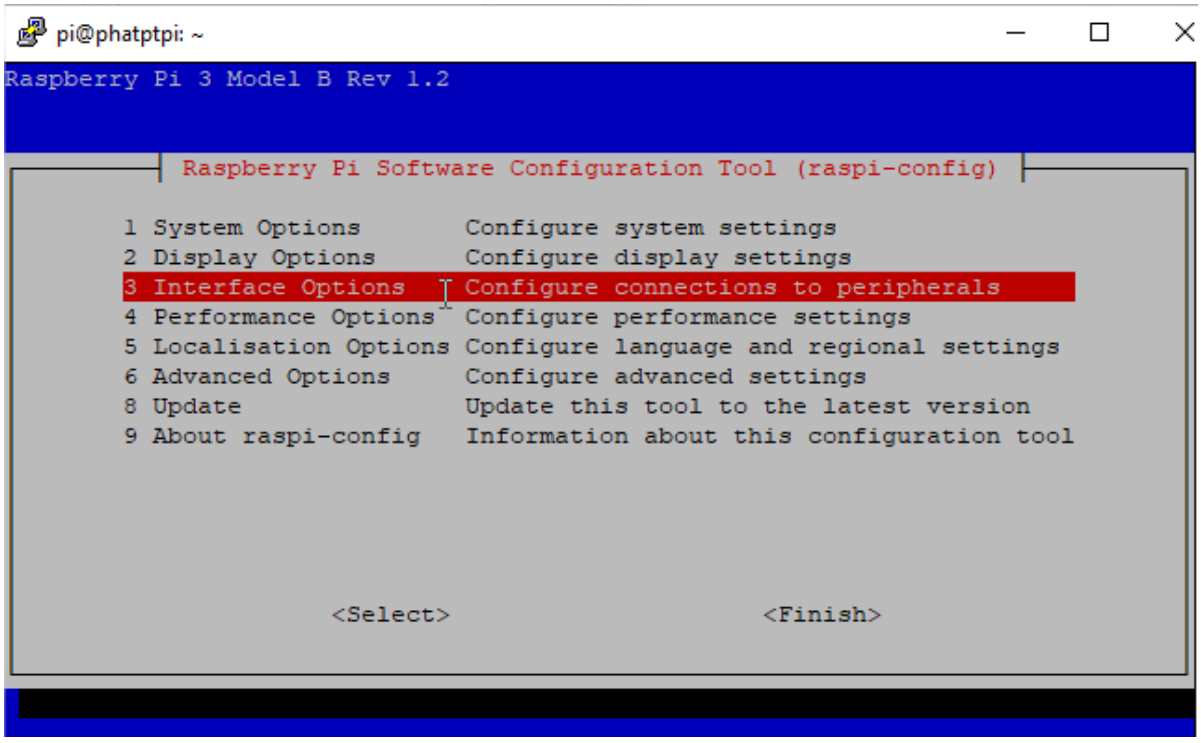
- **Bước 8:** Kết nối thẻ nhớ đã cài đặt hệ điều hành lên Raspberry Pi và khởi động nguồn cho thiết bị.

- **Bước 9:** Sử dụng phần mềm **Putty** với hostname, username và pasword đã cấu hình ở **Bước 6** để SSH vào Raspberry Pi.

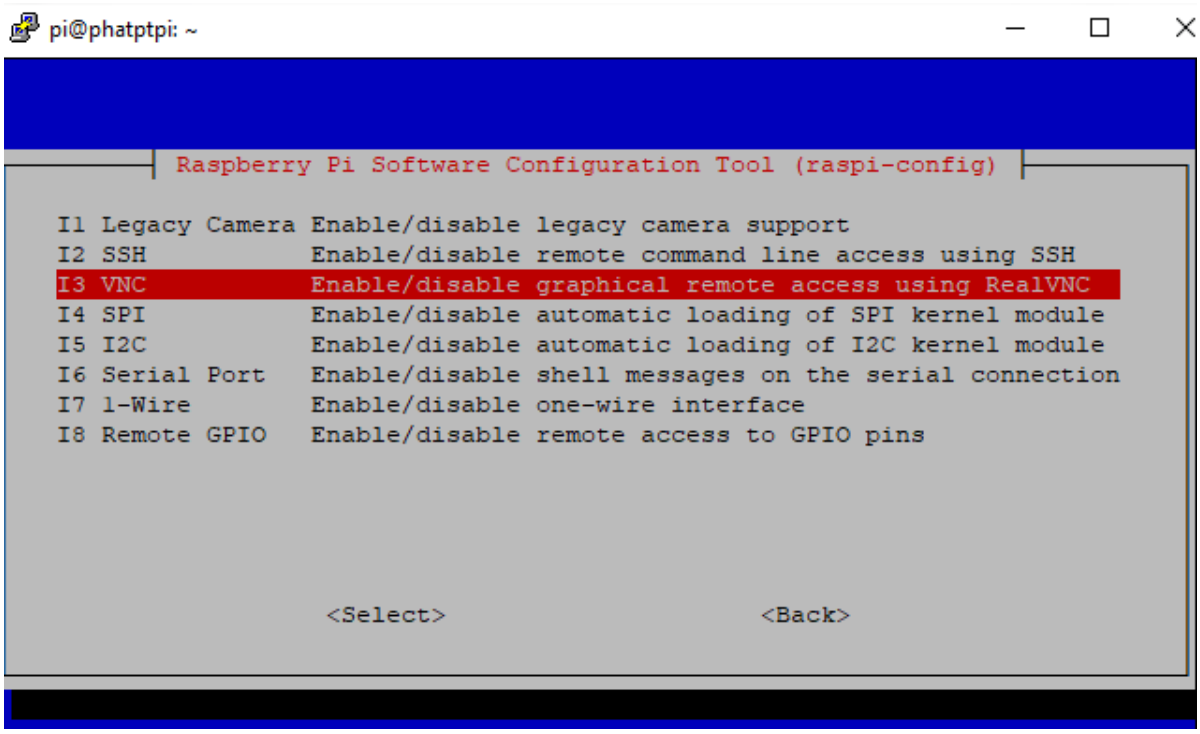


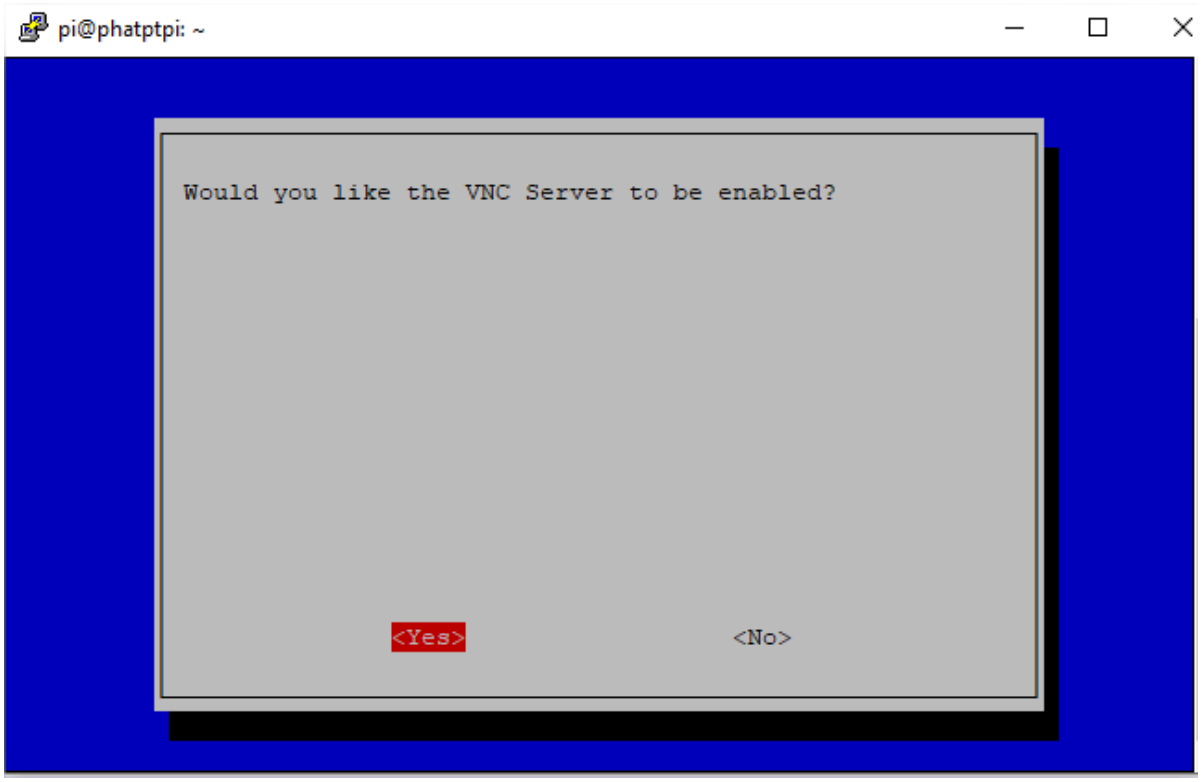
(*Nếu không sử dụng được hostname thì có thể chuyển sang sử dụng địa chỉ IP để thay thế. Sinh viên có thể sử dụng 1 phần mềm Network Scanner bất kỳ để tìm địa chỉ IP của Raspberry Pi hoặc sử dụng cáp Ethernet để kết nối đến thiết bị và tìm địa chỉ IP*).

- **Bước 10:** Sử dụng lệnh `sudo raspi-config` để mở **Raspberry Pi Software Configuration Tool**.
- **Bước 11:** Chọn **Interface Options**.

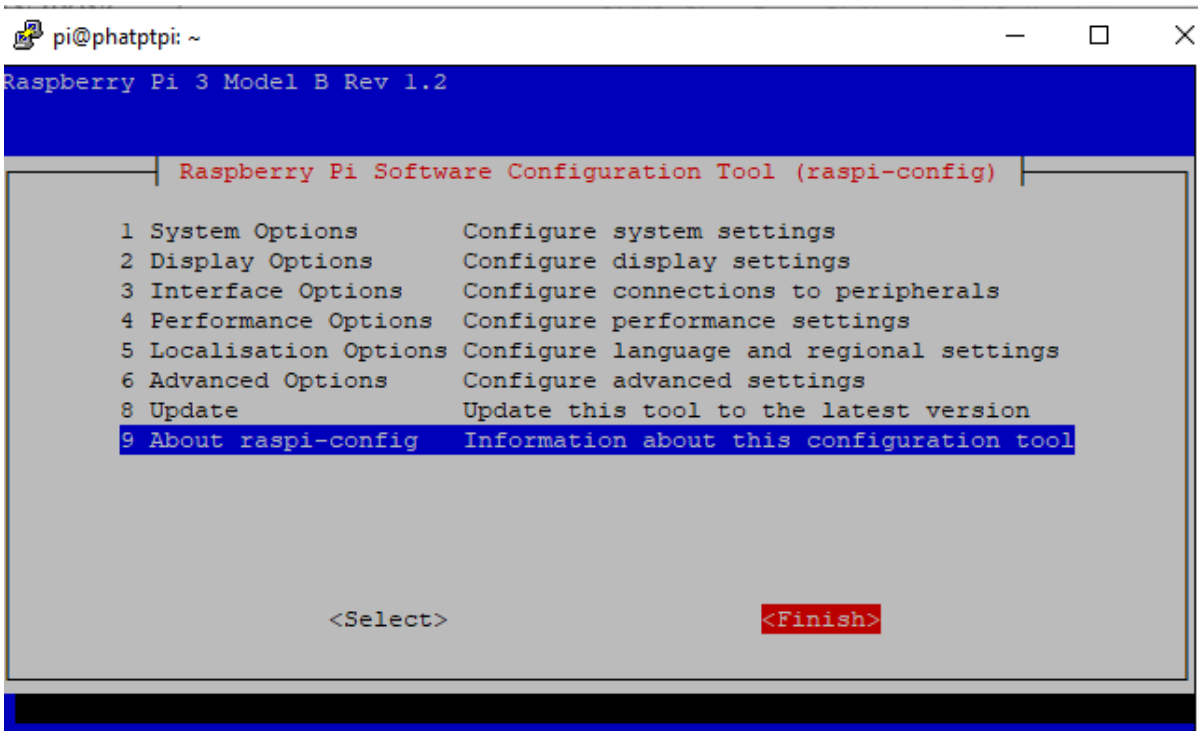


- **Bước 12:** Chọn **VNC** và Enable VNC Server trên Raspberry Pi.



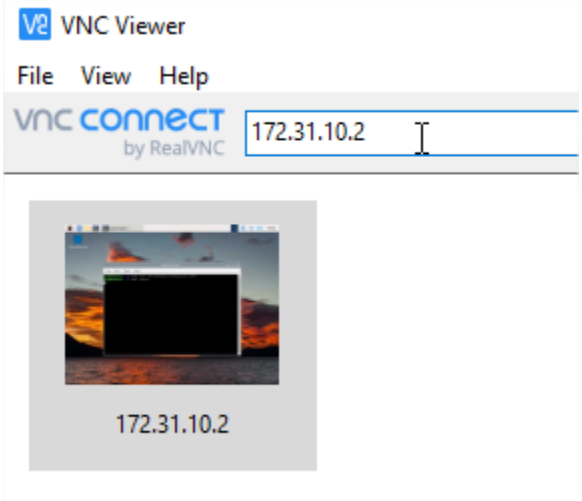


- **Bước 13:** Lưu cấu hình đã thay đổi.



- **Bước 14:** Trên máy tính cá nhân, tải phần mềm VNC Viewer tại https://www.realvnc.com/en/connect/download/viewer/

- **Bước 15:** Gõ địa chỉ IP của Raspberry Pi và bắt đầu sử dụng VNC.



D. YÊU CẦU & NỘP BÀI

1. Yêu cầu

Đối với phần HTTP, sinh viên thực hiện các yêu cầu sau:

1. Lấy ý tưởng từ bài 5 - bài thực hành số 1 "**thử tài đoán số**", xây dựng kịch bản gồm 1 Wemos D1, 5 đèn LED. HTTP Server được dựng trên Wemos D1. Trò chơi được dừng lại khi người chơi không còn điểm. Bắt đầu với 5 đèn LED sáng từ trái sang phải và ngược lại, sau đó trò chơi sẽ ngẫu nhiên 1 số lượng đèn bất kỳ. Đèn được hiển thị trong khoảng thời gian là 2 giây sau đó tắt đi, vị trí của các đèn được thiết đặt một cách ngẫu nhiên. Người chơi cần nhanh trí đếm số lượng đèn và % 3, thực hiện chọn nút tương ứng trên giao diện website trong khoảng 2 giây tiếp theo. Nếu người dùng chọn đúng thì sẽ được cộng điểm và bị trừ điểm nếu chọn sai hoặc hết thời gian. Điểm số và tình trạng của trò chơi sẽ được hiển thị qua giao diện website.

Lưu ý: Sử dụng **URL parameters** (*query string*) để thể hiện việc chọn các con số trên giao diện website.

2. Xây dựng kịch bản thiết bị gồm 1 Wemos D1, 1 cảm biến ánh sáng, 1 cảm biến khoảng cách và 3 đèn LED. Trong đó, máy tính cá nhân như một Server, sử dụng một ngôn ngữ

lập trình bất kỳ để xây dựng các RESTful API (chủ yếu sử dụng hai phương thức GET và POST). Sử dụng Wemos D1 như một HTTP Client, định kỳ mỗi 5s Wemos D1 thu thập giá trị cảm biến ánh sáng và cảm biến khoảng cách, gửi các giá trị cảm biến với định dạng JSON trong payload của POST API ở trên. Đồng thời, Server sẽ trả về số lượng đèn sáng tùy theo cường độ ánh sáng. Nếu có người tiếp cận dựa vào giá trị của cảm biến khoảng cách thì trả về cho người dùng số lượng đèn sáng để cung cấp đủ ánh sáng. Nếu không có người tiếp cận thì tắt các đèn. Khi máy tính nhận được thông điệp thì xuất ra command line hoặc bất kỳ trình dạng nào có thể quan sát được. Các nội dung gửi đi và về đều ở dạng JSON.

Lưu ý: Định dạng của HTTP API:

```
{
  "error": true,
  "message": "this is a message of API",
  "data": {
    "temperature": value,
    "light": value,
  }
}
```

Đối với phần MQTT, sinh viên thực hiện các yêu cầu sau:

3. SSH hoặc VNC vào Raspberry Pi được cung cấp sẵn để tiến hành kiểm tra hoạt động của MQTT Broker. Sử dụng một phần mềm MQTT Client bất kỳ (ví dụ như MQTTLens) để tiến hành publish vào topic "cq21/nhomX/check", với X là số thứ tự nhóm. Trên Raspberry Pi subcribe vào topic trên để nhận thông điệp được gửi từ Client.

4. Xây dựng kịch bản 1 Wemos D1, 1 Raspberry Pi, 1 máy tính cá nhân và 1 đèn LED. Raspberry Pi đóng vai trò là MQTT Broker. Triển khai MQTT Client trên Wemos D1 và máy tính cá nhân. Tại máy tính cá nhân người dùng thông qua topic "cq21/nhomX/led", tiến hành publish các trạng thái của đèn LED. Tại Wemos D1, subscribe vào topic "cq21/nhomX/led" để lắng nghe trạng thái của đèn, từ đó điều khiển bật hoặc tắt đèn LED.

5. Xây dựng kịch bản gồm 1 Wemos D1, 1 Raspberry Pi, 1 máy tính cá nhân, 1 cảm biến nhiệt độ, độ ẩm DHT22 và 1 đèn LED. Raspberry Pi đóng vai trò là MQTT Broker. Tại máy tính cá nhân, sử dụng một ngôn ngữ lập trình bất kỳ để subcribe vào lắng nghe dữ liệu cảm biến từ Wemos D1 thông qua topic "cq21/nhom**X**/dht/value". Cho khoảng giá trị bình thường của nhiệt độ là từ 25 °C - 27°C hoặc độ ẩm 40% - 70%, nếu trong quá trình đo đạc phát hiện giá trị bất thường thì máy tính cá nhân publish tín hiệu bất thường vào topic "cq21/nhom**X**/dht/detected". Wemos D1 subcribe vào topic "cq21/nhom**X**/dht/detected", nếu nhận giá trị phát hiện bất thường thì chớp tắt đèn một cách liên tục để báo hiệu.

2. Yêu cầu nộp bài

- Sinh viên tìm hiểu và thực hành theo hướng dẫn. Thực hiện **nhóm.**
- Sinh viên báo cáo kết quả thực hiện và nộp bài bằng file. Trong đó:
  - Trình bày chi tiết quá trình thực hành và trả lời các câu hỏi nếu có (kèm theo các ảnh chụp màn hình tương ứng).
  - Giải thích các kết quả đạt được.
  - Tải mẫu báo cáo thực hành và trình bày theo mẫu được cung cấp.

Nén tất cả các file và đặt tên file theo định dạng theo mẫu: **NhomY-LabX_MSSV1_MSSV2** Ví dụ: Nhom1-Lab03_25520001_25520002

  - Nộp báo cáo trên theo thời gian đã thống nhất tại website môn học.
  - Các bài nộp không tuân theo yêu cầu sẽ KHÔNG được chấm điểm.

HẾT

Chúc các bạn hoàn thành tốt bài thực hành!