

5. Lập trình ứng dụng IoT cơ bản với Raspberry Pi

- Mục đích
 - Lập trình trao đổi dữ liệu với Server sử dụng ngôn ngữ Python thông qua giao thức MQTT, HTTP POST/GET giữa Raspberry Pi với Server ThingSpeak.
 - Nâng cao khả năng tạo giao diện với ngôn ngữ HTML cơ bản và Node-RED cơ bản.
 - Lập trình giao tiếp, thu thập dữ liệu và điều khiển các thiết bị ngoại vi cơ bản với Raspberry Pi.
 - Xây dựng các ứng dụng IoT cơ bản với Raspberry Pi.
- Yêu cầu
 - Nắm vững cấu trúc phần cứng của Raspberry Pi 4B và các thiết bị ngoại vi sử dụng.
 - Biết cách kết nối các thiết bị ngoại vi với Raspberry Pi.
 - Biết cách tạo ra một giao diện Web cho các ứng dụng IoT cơ bản sử dụng ngôn ngữ HTML và Node-RED.
 - Biết cách thiết kế các ứng dụng IOT cơ bản với Raspberry Pi để điều khiển và thu thập dữ liệu thông qua giao thức HTTP/MQTT từ giao diện Web.
 - Biết đọc tài liệu tiếng Anh liên quan.

5.1. Giới thiệu

5.1.1. Chuẩn bị các phần cứng và phần mềm cần thiết

Để giúp cho việc viết mã nguồn sử dụng ngôn ngữ Python cho Raspberry Pi được đơn giản và nhanh chóng, thì trong phạm vi các bài thực hành sau đây chúng ta sẽ nhận sự hỗ trợ từ một số thư viện được cài đặt mặc định trong hệ điều hành hoặc phải cài đặt bổ sung vào thêm trong quá trình sử dụng Raspberry Pi. Chẳng hạn như: thư viện GPIOZero (thư viện mặc định có sẵn), thư viện hỗ trợ riêng cho từng loại cảm biến (đã hướng dẫn cài đặt trong phần trước), thư viện hỗ trợ các giao thức truyền nhận dữ liệu với Server như HTTP, MQTT (thư viện phải cài đặt bổ sung).

Trong một số trường hợp chúng ta muốn hiển thị các dữ liệu lưu trữ tên Server lên giao diện web dưới dạng đồ thị, khi đó ThingSpeak cung cấp cho chúng ta một khả năng nhúng đồ thị dữ liệu vào giao diện web rất đơn giản. Để thực hiện việc này, đầu tiên chúng ta cần chỉnh lại chế độ "**Public**" cho kênh lưu trữ dữ liệu cần nhúng. Mặc định thì các kênh lưu trữ dữ liệu sẽ được đặt ở chế độ "**Private**" (sẽ có một logo hình ổ khóa đóng ở phần đầu của tên kênh lưu trữ, xem minh họa trong hình bên dưới), để chuyển kênh này sang chế độ "**Public**" thì đầu tiên ta đăng nhập vào kênh lưu trữ và chọn vào thẻ "**Sharing**".

Name	Created	Updated
Test_Data_Server	2019-06-05	2019-06-21 02:56

New Channel Search by tag

Private Test_Data_Server Created Updated

Sharing

Tiếp theo, ta chọn mục "**Share channel view with everyone**".

Test_Data_Server

Channel ID: 794872
Author: iotlabiuh
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Channel Sharing Settings

- Keep channel view private
- Share channel view with everyone
- Share channel view only with the following users:

Email Address Enter email here Add User

Help

ThingSpeak allows the settings on this page to be changed without requiring the app to be re-installed.

Channel Sh

* Keep channel view private

Như vậy là ta hoàn thành xong việc chuyển kênh lưu trữ sang chế độ "**Public**" (sẽ có một logo hình ổ khóa mở ở phần đầu của tên kênh lưu trữ, xem minh họa trong hình bên dưới).

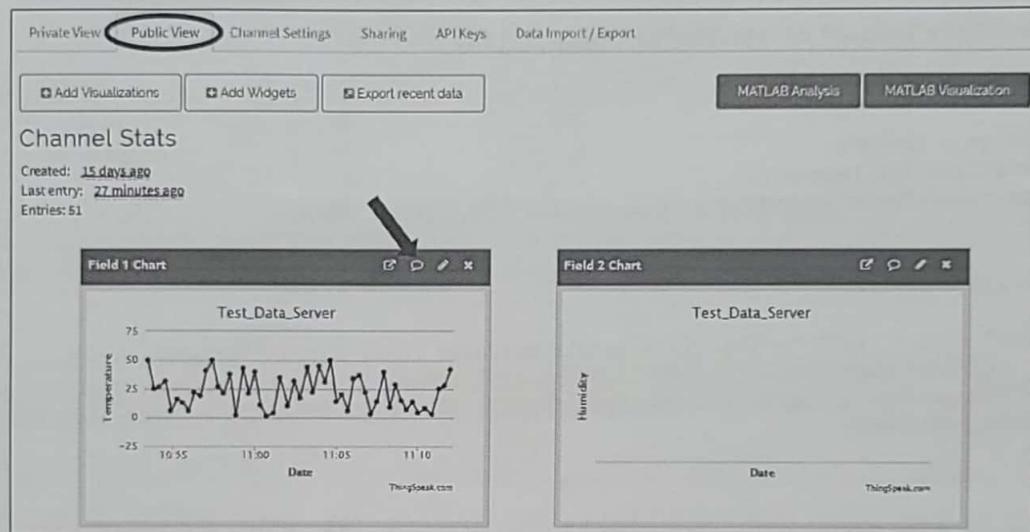
Name	Created	Updated
Test_Data_Server	2019-06-05	2019-06-21 02:55

New Channel Search by tag

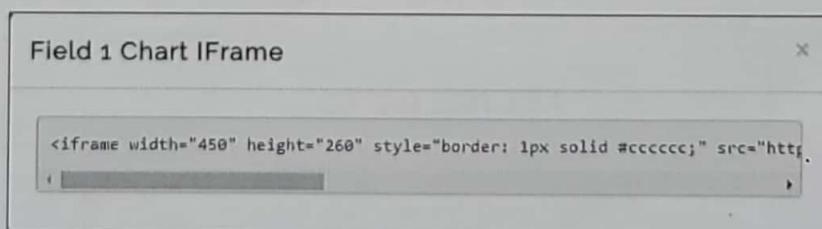
Public Test_Data_Server Created Updated

Sharing

Ké tiếp, để nhúng đồ thị dữ liệu của kênh lưu trữ trên Server ThingSpeak vào trang web thì ta thực hiện như sau, đầu tiên ta đăng nhập vào kênh lưu trữ và chọn vào thẻ “**Public View**”. Sau đó trong cửa sổ của đồ thị mà ta cần nhúng vào web, giả sử trong trường hợp minh họa này ta sẽ nhúng đồ thị của trường lưu trữ “**Field1**” vào web, khi đó ta sẽ chọn vào biểu tượng “**Field 1 Chart IFrame**”, xem minh họa trong hình bên dưới.



Trên màn hình máy tính sẽ xuất hiện hộp thông tin “**Field 1 Chart IFrame**”, bên trong hộp thông tin này sẽ chứa đoạn mã HTML để cho phép chúng ta nhúng đồ thị của trường lưu trữ này vào giao diện web, xem hình minh họa hộp thông tin và đoạn mã HTML được tạo ra trong hình bên dưới.



```
<iframe width="450" height="260" style="border: 1px solid #cccccc;" src="https://thingspeak.com/channels/794872/charts/1?bgcolor=%23ffff&color=%23d62020&dynamic=true&results=60&type=line&update=15"></iframe>
```

5.2. Xây dựng giao diện web cơ bản trên nền tảng HTML

5.2.1. Nhúng đồ thị dữ liệu trên Server ThingSpeak vào Web

Để nhúng đồ thị dữ liệu trên Server ThingSpeak vào Web, ta sao chép toàn bộ đoạn mã HTML được cung cấp này và dán vào trong đoạn mã nguồn HTML của trang web tại vị trí mong muốn. Xem minh họa trong đoạn mã nguồn HTML ví dụ bên dưới,

phần đánh dấu in đậm chính là đoạn mã HTML dùng để cho phép nhúng đồ thị dữ liệu vào giao diện web.

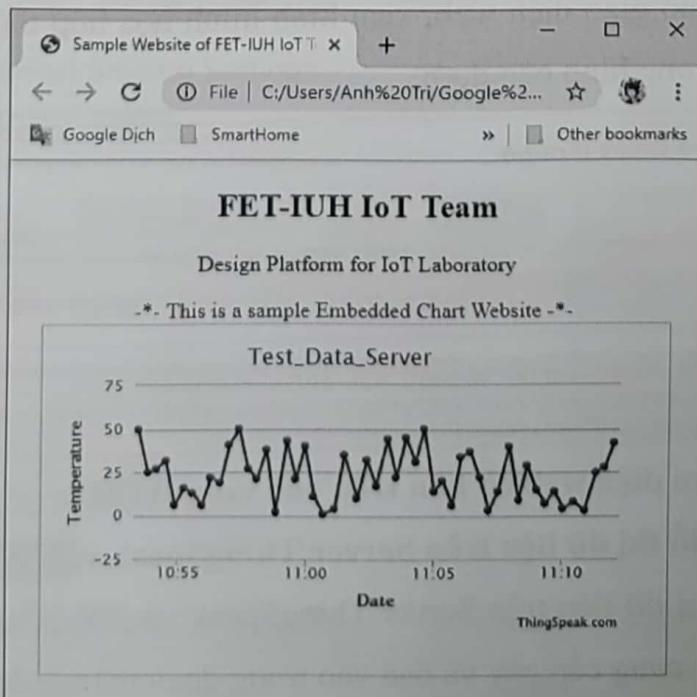
```
from time import sleep
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="generator" content="FET-IUH IoT Team">
    <meta name="dcterms.created" content="T6, 14 Thg6 2019 01:37:13 GMT">
    <meta name="description" content="Design Platform for IoT Laboratory">
    <meta name="keywords" content="Internet of things, LoraWan, Raspberry Pi 4">

    <title>Sample Website of FET-IUH IoT Team</title>
  </head>

  <body>
    <div align = center>
      <h2>FET-IUH IoT Team</h2>
      <font color="blue">Design Platform for IoT Laboratory</font>
    </div>

    <div align = center>
      <p>
        <font color="red">-*- This is a sample Embedded Chart Website -*-</font> <br>
        <iframe width="450" height="260" style="border: 1px solid #cccccc;" src="https://thingspeak.com/channels/794872/charts/1?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=60&type=line&update=15"></iframe>
      </p>
    </div>
  </body>
</html>
```

Kết quả của ví dụ trên ta sẽ thấy xuất hiện một đồ thị dữ liệu đã được nhúng vào giao diện của trang web này.



5.2.2. Tạo thanh trượt trên web và gửi dữ liệu lên Server ThingSpeak

Trong một số trường hợp chúng ta muốn tạo ra các dữ liệu từ một thanh trượt (Slider) nhằm giúp cho giao diện web được sinh động hơn và cập nhật các dữ liệu này lên Server. Để thực hiện được việc đó ta cần thực hiện hai công việc như sau: thứ nhất là tạo ra một thanh trượt trên giao diện web cùng khoảng giới hạn giá trị của nó và thứ hai là gửi các giá trị này lên Server đã chọn.

Đoạn mã HTML dưới đây dùng để tạo một thanh trượt trên giao diện web, thanh trượt này sẽ tạo ra các giá trị nằm trong giới hạn từ 0 (min) đến 100 (max) và có giá trị ban đầu là 0 (value).

```
<input type="range" min="0" max="100" value="0" class="slider" id="myRange"
onchange="updateTextInput(this.value);">
<input type="text" id="textInput" value="0">

<script>
    function updateTextInput(val)
    {
        document.getElementById('textInput').value = val;
    }
</script>
```

Đoạn mã HTML dưới đây dùng để tạo ra một nút nhấn dùng để cập nhật giá trị tạo ra từ thanh trượt bên trên lưu trữ “**Field1**” của Server ThingSpeak đã được tạo ra trước đây.

```
<button onclick="Data_Slider()">Send</button>

<script>
    function Data_Slider()
    {
        var url = "https://api.thingspeak.com/update?";
        var value_slider = document.getElementById('myRange').value;
        var params_slider = "api_key=VN9L7BBX6YI4LWJ3&field1="+value_slider;
        var xhr = new XMLHttpRequest();
        xhr.open("POST", url, true);
        xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xhr.send(params_slider);
    }
</script>
```

Đoạn mã HTML đầy đủ minh họa cho một ví dụ về việc nhúng đồ thị dữ liệu trên Server ThingSpeak vào web và tạo thanh trượt trên web để gửi dữ liệu lên Server.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="generator" content="FET-IUH IoT Team">
        <meta name="dcterms.created" content="T6, 14 Thg6 2019 01:37:13 GMT">
        <meta name="description" content="Design Platform for IoT Laboratory">
        <meta name="keywords" content="Internet of things, LoRaWan, Raspberry Pi 4">
```

```

<title>Sample Website of FET-IUH IoT Team</title>
</head>

<body>
    <div align = center>
        <h2>FET-IUH IoT Team</h2>
        <font color="blue">Design Platform for IoT Laboratory</font>
    </div>

    <div align = center>
        <p>
            <font color="red">-*- This is a sample Embedded Chart Website -*-</font> <br>
            <iframe width="450" height="260" style="border: 1px solid
#cccccc;" src="https://thingspeak.com/channels/794872/charts/1?bgcolor=%23fffff&color=%23d62020&dy
namic=true&results=60&type=line&update=15"></iframe>
        </p>
    </div>

    <div align = center>
        <p>
            <font color="red">-*- This is a sample Data Slider -*-</font> <br>
            <input type="range" min="0" max="100" value="0" class="slider" id="myRange"
onchange="updateTextInput(this.value);">
            <input type="text" id="textInput" value="0">

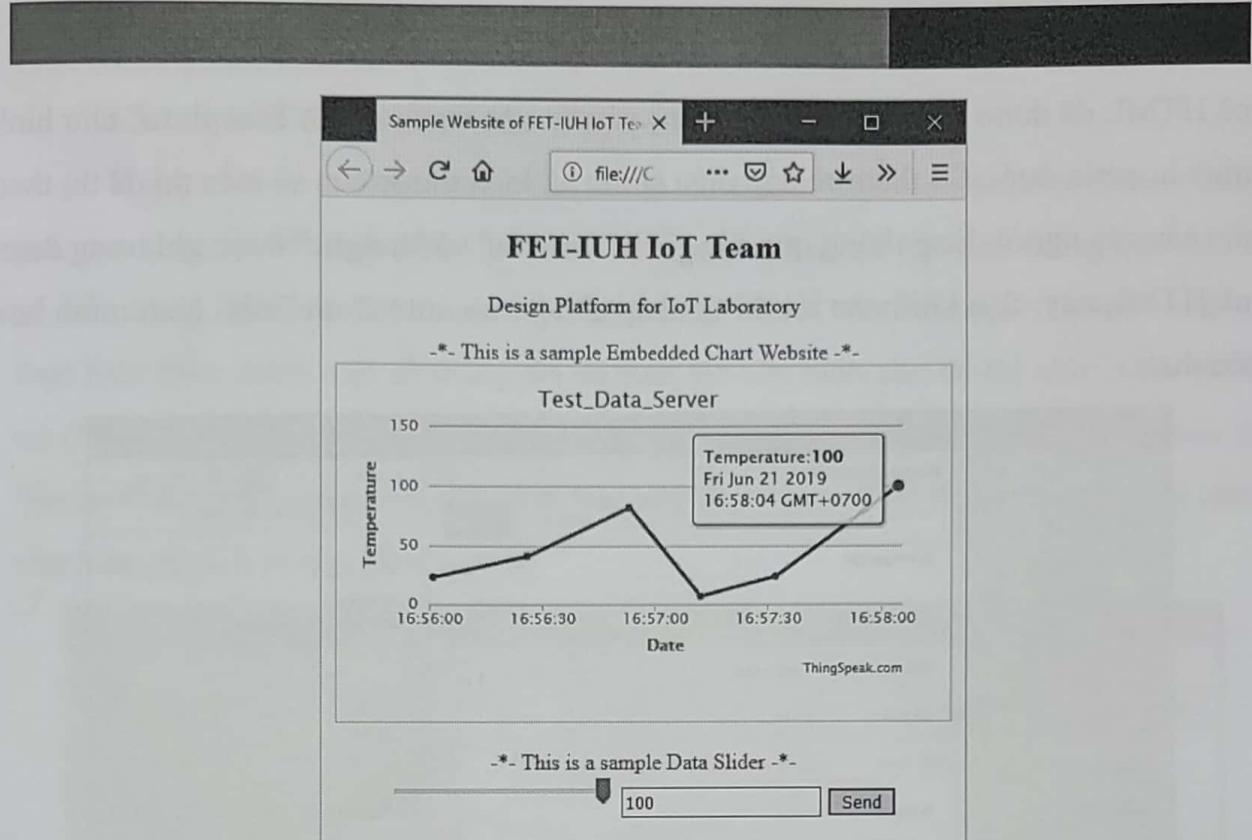
            <button onclick="Data_Slider()">Send</button>
        </p>
    </div>

    <script>
        function updateTextInput(val)
        {
            document.getElementById('textInput').value = val;
        }

        function Data_Slider()
        {
            var url = "https://api.thingspeak.com/update?";
            var value_slider = document.getElementById('myRange').value;
            var params_slider = "api_key=VN9L7BBX6YI4LWJ3&field1="+value_slider;
            var xhr = new XMLHttpRequest();
            xhr.open("POST", url, true);
            xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
            xhr.send(params_slider);
        }
    </script>
</body>
</html>

```

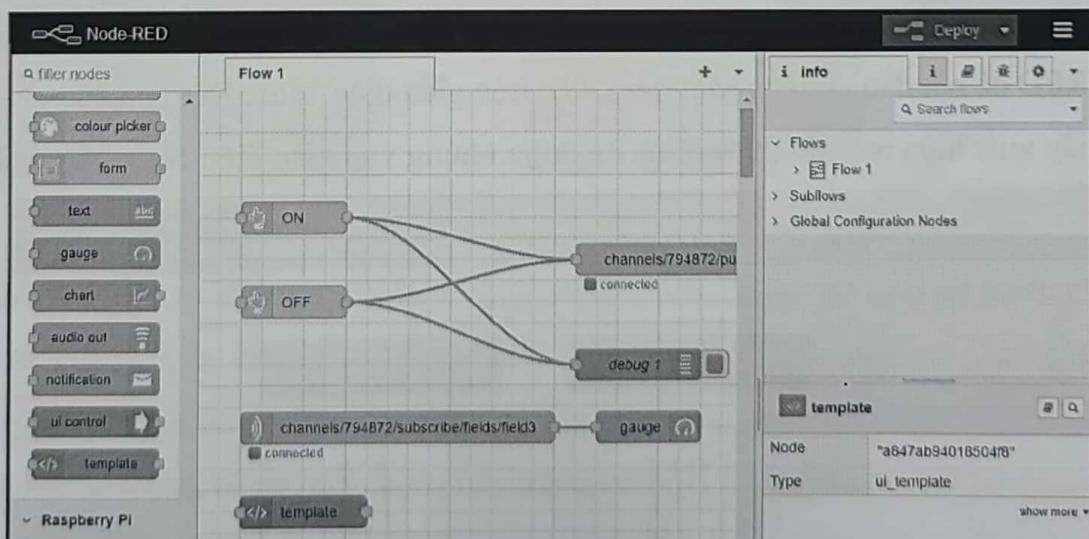
Kết quả của ví dụ trên ta sẽ thấy xuất hiện một giao diện web như sau:



5.3. Xây dựng giao diện web cơ bản trên nền tảng Node-RED

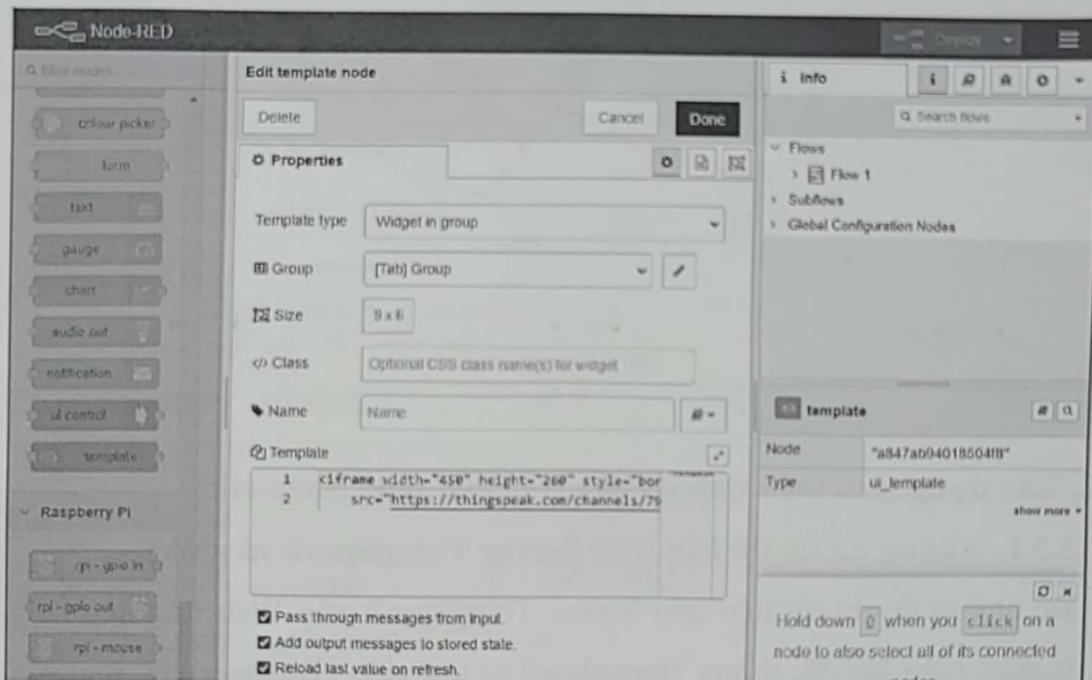
5.3.1. Nhúng đồ thị dữ liệu trên Server ThingSpeak vào Web

Để nhúng đồ thị dữ liệu trên Server ThingSpeak vào Web trên nền tảng Node-RED, ta thực hiện lấy đối tượng “template” từ thư viện “Dashboard” như hình minh họa bên dưới.

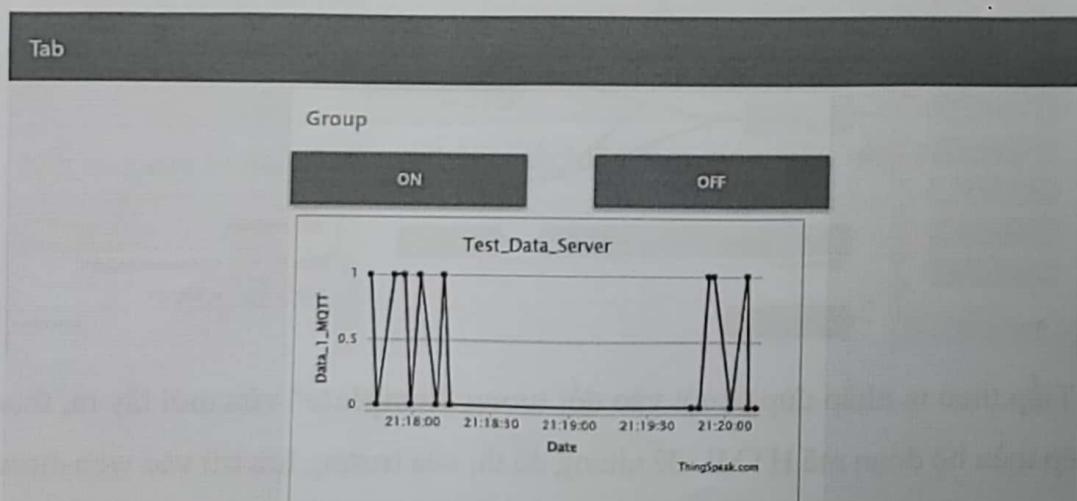


Tiếp theo ta nhấp đúp chuột vào đối tượng “template” vừa mới lấy ra, thực hiện sao chép toàn bộ đoạn mã HTML để nhúng đồ thị của trường lưu trữ vào web được cung cấp từ Server ThingSpeak đã tạo ra từ bài học trước (xem lại hướng dẫn cách lấy đoạn

mã HTML đã được trình bày ở phần trên) và dán vào trong mục “Template” như hình minh họa bên dưới. Có thể trực tiếp điều chỉnh lại kích thước cửa sổ hiển thị đồ thị theo nhu cầu của người dùng thông qua các giá trị “width” và “height” được ghi trong đoạn mã HTML này. Sau khi hoàn tất thì ta nhấp chuột vào nút “Done” như hình minh họa bên dưới.

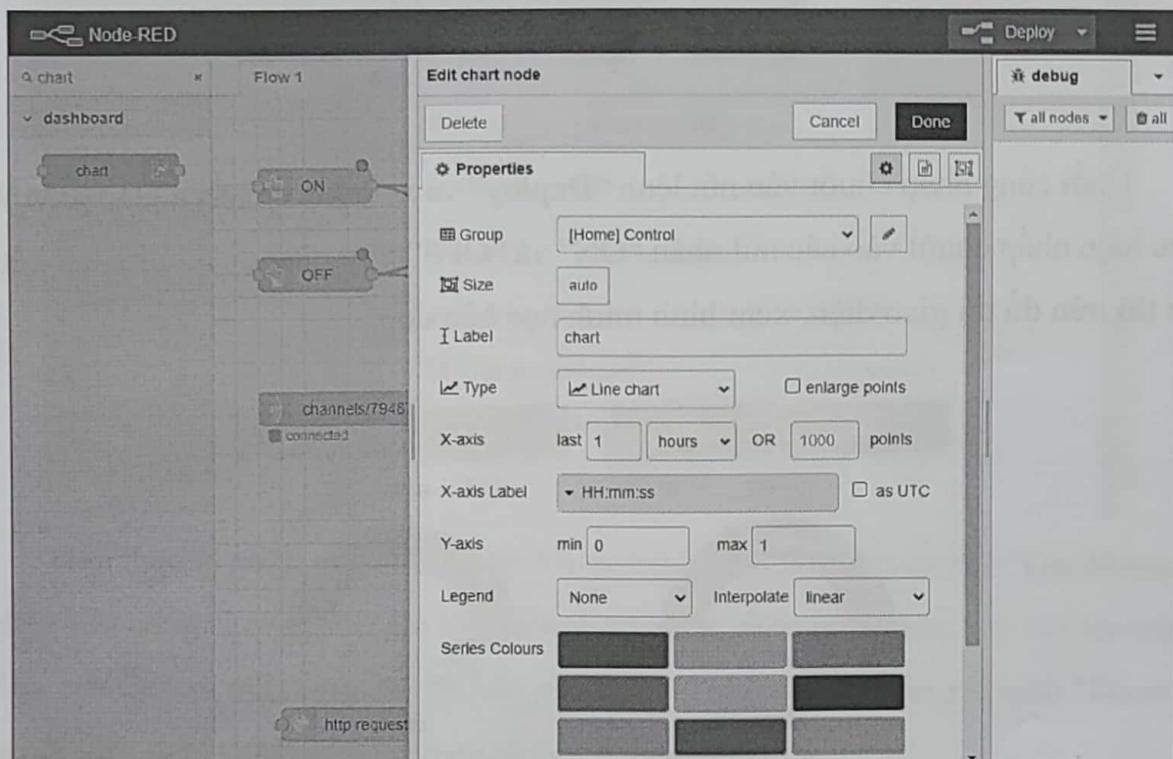


Đến đây thì chúng ta đã hoàn tất việc nhúng đồ thị dữ liệu trên Server ThingSpeak vào Web trên nền tảng Node-RED, nhấp chuột vào nút lệnh “Deploy” trên giao diện Node-RED để bắt đầu chuyển sang làm việc trên giao diện Web. Kết quả của ví dụ trên ta sẽ thấy xuất hiện một đồ thị dữ liệu đã được nhúng vào giao diện của trang web này.

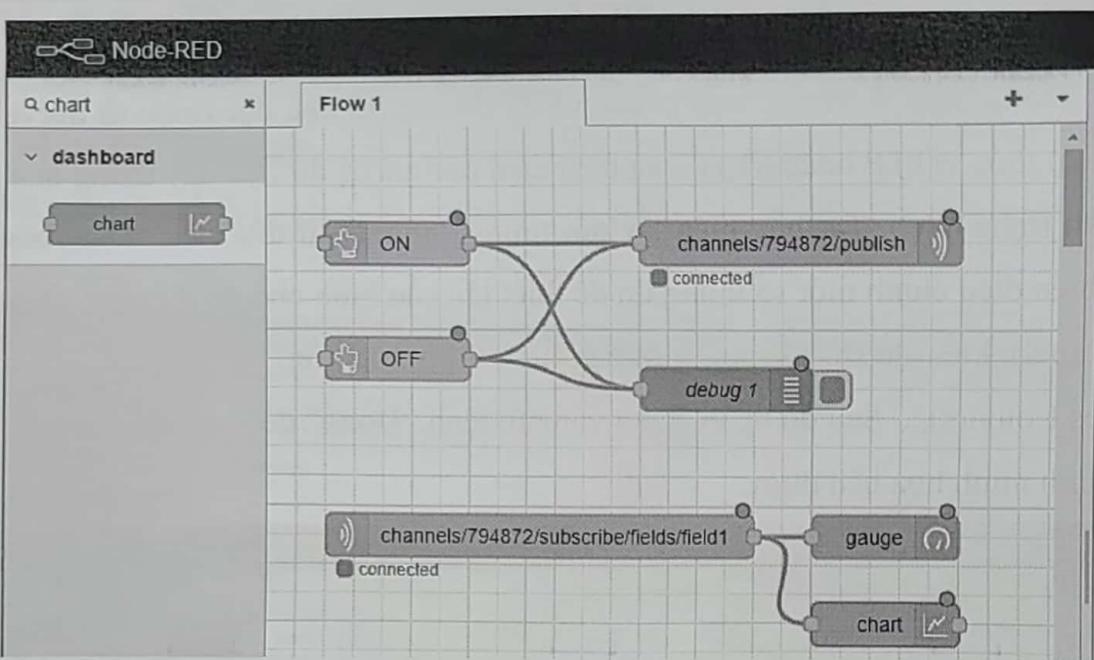


5.3.2. Vẽ đồ thị dữ liệu trên Web

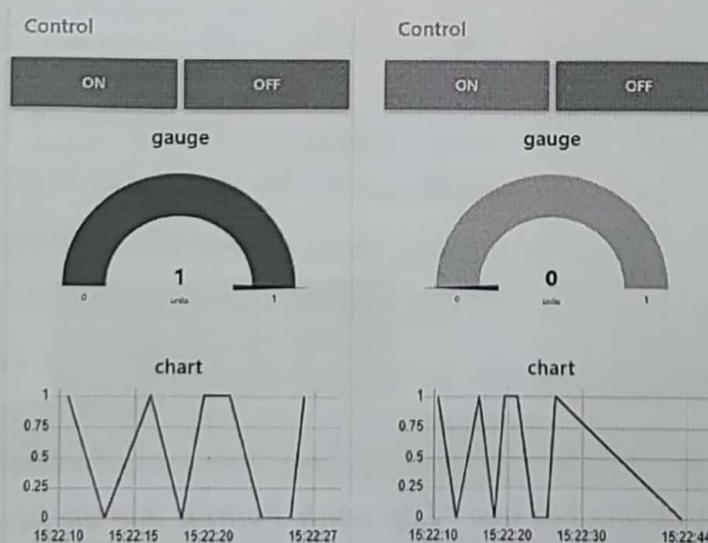
Trong một số trường hợp ta cần vẽ đồ thị của dữ liệu đọc được từ Server thông qua giao thức HTTP hoặc MQTT và biểu diễn các giá trị dữ liệu này thông qua một đồ thị trên trang Web. Ta thực hiện lấy đối tượng “chart” từ thư viện “Dashboard” và thực hiện điều chỉnh một số thông tin để thiết lập cấu hình cho đồ thị như: nhãn tên đồ thị (Label), kiểu đồ thị (Type), giới hạn hiển thị giá trị trên đồ thị (X-axis), màu sắc (Series Colours),... Sau đó nhấp chuột vào nút lệnh “Done” để hoàn tất việc điều chỉnh như hình minh họa bên dưới.



Giả sử ta cần vẽ đồ thị của dữ liệu đọc được từ Server ThingSpeak thông qua giao thức MQTT đã tạo ra trong bài học “**Lập trình Python đọc dữ liệu từ Server**”. Thực hiện vẽ đường liên kết ngõ vào của đối tượng “chart” với ngõ ra của đối tượng “mqtt_in” dùng xuất dữ liệu đọc được từ Server thông qua giao thức MQTT (xem lại cách thiết lập thông số cho đối tượng này trong bài học “**Tạo giao diện Web dùng Node-RED**”) như hình minh họa bên dưới.



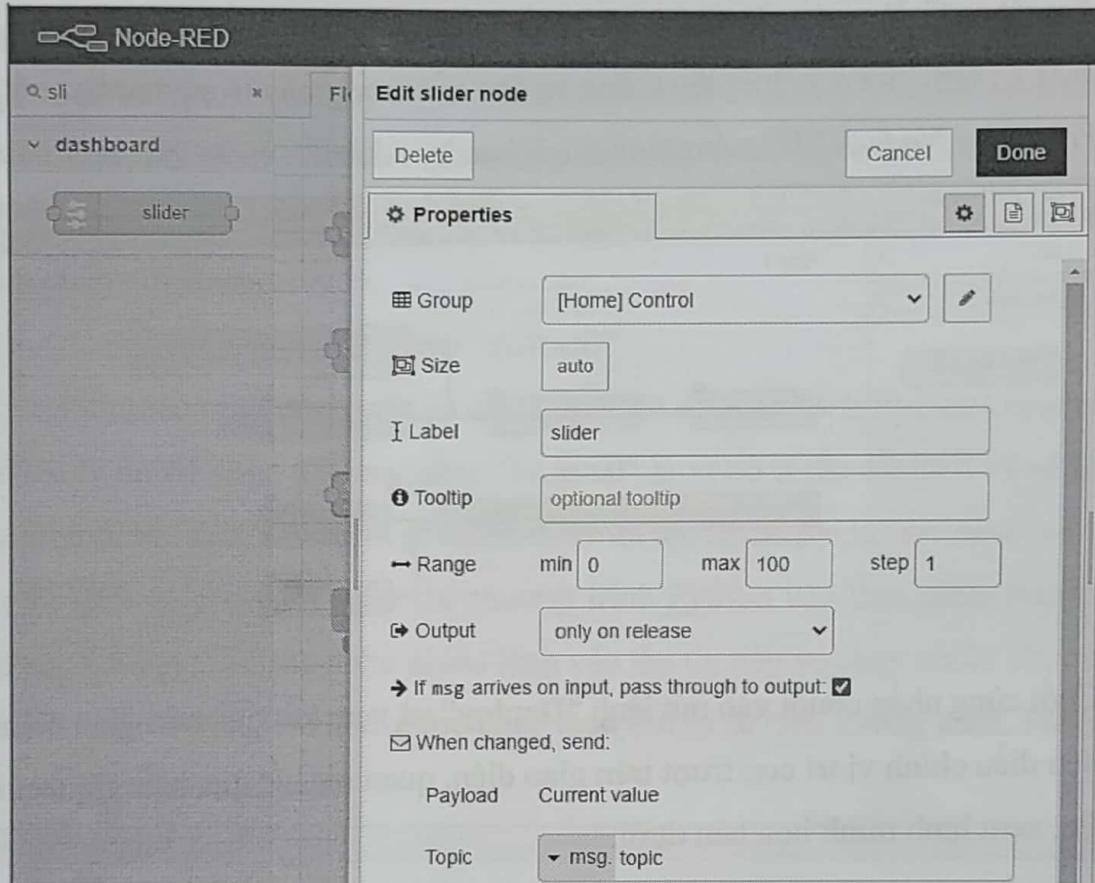
Cuối cùng nhấp chuột vào nút lệnh “Deploy” và xem kết quả trên giao diện Web. Thực hiện nhấp chuột vào các nút nhấn “ON” và “OFF” trên giao diện, quan sát kết quả hiển thị trên đồ thị giao diện, xem hình minh họa bên dưới.



5.3.3. Tạo thanh trượt trên web và gửi dữ liệu lên Server ThingSpeak

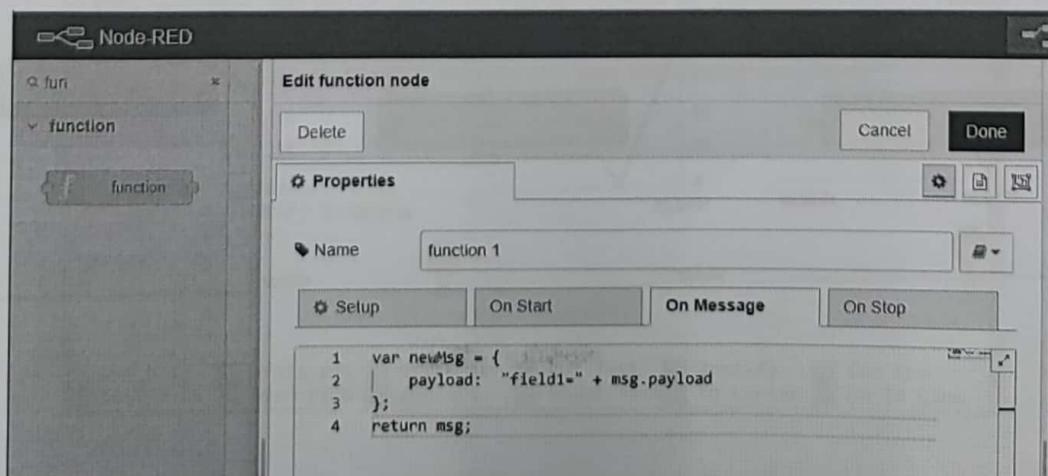
Trong trường hợp chúng ta muốn tạo ra các dữ liệu từ một thanh trượt (Slider) nhằm giúp cho giao diện web được sinh động hơn và cập nhật các dữ liệu này lên Server, để thực hiện được việc đó ta cần lấy 2 đối tượng “slider” và “function” từ thư viện “Dashboard”. Đầu tiên ta nhấp đúp chuột vào đối tượng “slider” và thực hiện điều chỉnh một số tham số như: tên thanh trượt (Name), giới hạn điều chỉnh giá trị (Range),

bước điều chỉnh giá trị (Step), kiểu xuất dữ liệu (Output),... Sau đó nhấp chuột vào nút lệnh “Done” để hoàn tất việc điều chỉnh như hình minh họa bên dưới.

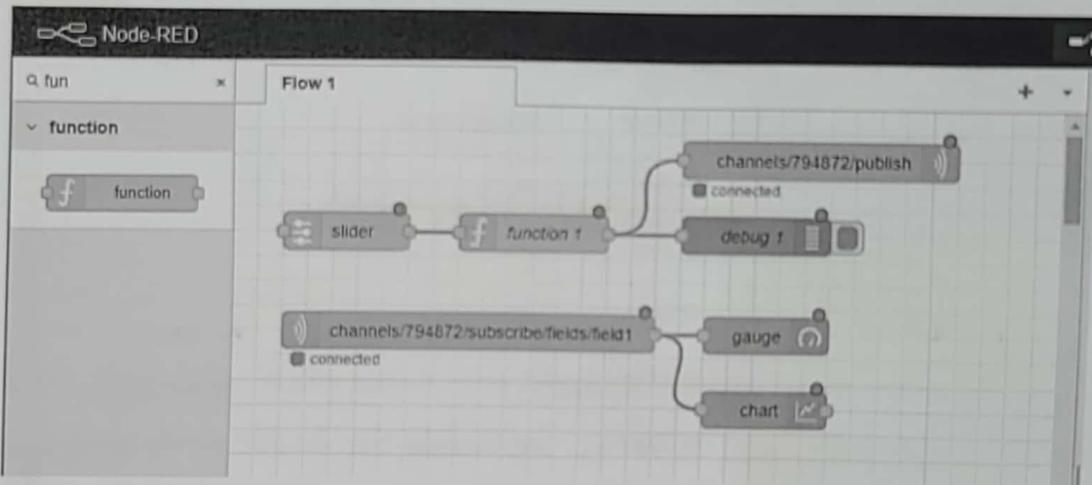


Tiếp theo ta nhấp đúp chuột vào đối tượng “function”, trong thẻ “On Message” ta nhập vào dòng lệnh để đọc giá trị từ thanh trượt và ghép nối giá trị này với tên trường dữ liệu của Server mà ta cần tải dữ liệu lên. Sau đó nhấp chuột vào nút lệnh “Done” để hoàn tất việc điều chỉnh như hình minh họa bên dưới.

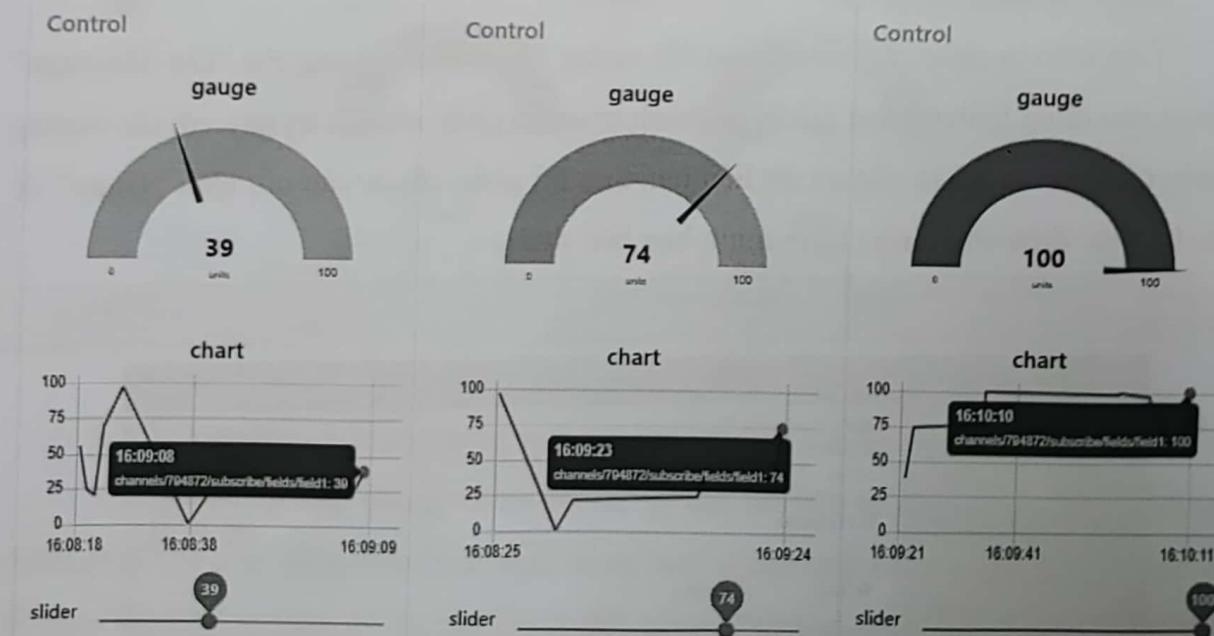
```
var newMsg = { payload: "field1=" + msg.payload };
return msg;
```



Thực hiện vẽ đường liên kết ngõ vào của đối tượng “slider” với ngõ vào của đối tượng “function”, sau đó kết nối ngõ ra của đối tượng “function” với ngõ vào của đối tượng “mqtt out” dùng để gửi dữ liệu đọc được từ thanh trượt lên Server thông qua giao thức MQTT (xem lại cách thiết lập thông số cho đối tượng này trong bài học “*Tạo giao diện Web dùng Node-RED*”) như hình minh họa bên dưới.



Cuối cùng nhấp chuột vào nút lệnh “Deploy” và xem kết quả trên giao diện Web. Thực hiện điều chỉnh vị trí con trượt trên giao diện, quan sát kết quả hiển thị trên đồ thị giao diện, xem hình minh họa bên dưới.



5.4. Phương pháp làm cho các chương trình Python tự động thực thi khi khởi động Raspberry Pi

Trong một số trường hợp chúng ta mong muốn một chương trình Python nào đó sẽ tự động được vận hành ngay khi hệ điều hành của Raspberry Pi được khởi động. Để thực hiện việc này sẽ có rất nhiều phương pháp khác nhau tuy nhiên trong phạm vi tài liệu này sẽ giới thiệu một số cách thức đơn giản và phù hợp với những người mới bắt đầu làm quen với Raspberry Pi.

5.4.1. Sử dụng phương pháp "rc.local"

Có rất nhiều cách để thực hiện việc chạy một chương trình Python khi khởi động. Tuy nhiên lý do để chọn phương pháp "rc.local" là vì nó ít rắc rối nhất và sẽ hiệu quả với người mới bắt đầu. Chúng ta sẽ chỉnh sửa một tập tin có tên là "rc.local" và bổ sung thêm một lệnh để cho phép thực thi chương trình Python vào thời điểm Raspberry Pi khởi động. Chúng ta có thể thêm nhiều lệnh vào tập tin này và chạy nhiều chương trình khác nhau, mỗi chương trình sẽ được chạy theo thứ tự từ trên xuống dưới. Đầu tiên tại Terminal nhập lệnh như sau:

```
~$ sudo nano /etc/rc.local
```

Trên màn hình sẽ xuất hiện một cửa sổ soạn thảo Nano, bên trong là nội dung của tập tin "rc.local" đã có sẵn một đoạn mã ví dụ, xem minh họa trong hình bên dưới.

```
pi@pqtr2002: ~
File Edit Tabs Help
GNU nano 2.7.4          File: /etc/rc.local          Modified
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will return 0 on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "%s" "$_IP"
fi

exit 0
```

Tiếp theo chúng ta sẽ xóa đoạn mã này đi, khi xóa ta cần lưu ý rằng chỉ xóa đi phần đoạn mã được đánh dấu như trong hình và nhớ phải giữ lại dòng "**exit 0**".

```
pi@pqtri2002: ~
File Edit Tabs Help
GNU nano 2.7.4          File: /etc/rc.local          Modified
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will exit 0 on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$IP" ]; then
    printf "My IP address is %s\n" "$IP"
fi

exit 0

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit      ^R Read File ^V Replace   ^U Uncut Text ^T To Linter ^L Go To Line
```

Tiếp theo chúng ta sẽ bổ sung vào trong đó dòng lệnh thực thi chương trình Python mà ta muốn nó sẽ được chạy vào thời điểm khởi động. Giả sử ở đây ta có đường dẫn đến thư mục lưu trữ tập tin là “**/home/pi/Desktop/StartupRasp**” và tập tin Python có tên là “**LED_flash_loop.py**”, chi tiết xem minh họa trong hình bên dưới.

```
pi@pqtri2002: ~
File Edit Tabs Help
GNU nano 2.7.4          File: /etc/rc.local          Modified
#
# By default this script does nothing.
#
# Print the IP address
_IP=$(hostname -I) || true
if [ "$IP" ]; then
    printf "My IP address is %s\n" "$IP"
fi

python /home/pi/Desktop/StartupRasp/LED_flash_loop.py

exit 0

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit      ^R Read File ^V Replace   ^U Uncut Text ^T To Linter ^L Go To Line
```

Một điểm rất quan trọng là nếu chương trình Python của chúng ta có ý định tiếp tục chạy trong nền hệ điều hành vô thời hạn (hoặc trong một khoảng thời gian dài). Dấu

"&" sẽ được thêm vào ở phía cuối cùng của dòng lệnh trên, khi đó nó sẽ được sử dụng để rẽ nhánh quá trình và cho phép hệ điều hành tiếp tục thực hiện chương trình Python. Vì vậy, nếu chương trình Python đơn giản chỉ được chạy trong một hoặc hai giây khi Raspberry Pi khởi động hoặc bên trong chương trình Python đã có lệnh vòng lặp thì chúng ta không thêm dấu này vào.

Tiếp theo chúng ta đóng cửa sổ soạn thảo Nano và lưu lại tập tin "rc.local" bằng cách nhấn "Ctrl + X" để thoát soạn thảo và sau đó nhấn "Y" để lưu lại tập tin. Cuối cùng sẽ tiến hành khởi động lại Raspberry Pi và quan sát kết quả đạt được, tại Terminal nhập lệnh như sau:

```
~$ sudo reboot
```

5.4.2. Sử dụng phần mềm “Supervisor”

Cách tiếp theo là chúng ta sử dụng thêm một phần mềm hỗ trợ có tên là “Supervisor”. Đầu tiên ta cần phải cài đặt phần mềm này vào Raspberry Pi, tại Terminal nhập lệnh như sau:

```
~$ sudo apt-get install supervisor
```

Tiếp theo ta cần xác định tập tin Python sẽ tự động được thực thi và xác định đường dẫn đến thư mục lưu trữ tập tin này. Giả sử ở đây ta có đường dẫn đến thư mục lưu trữ tập tin là “/home/pi/Desktop/StartupRasp” và tập tin Python có tên là “LED_flash_loop.py”. Sau khi đã có đủ các thông tin nêu trên thì ta sẽ tạo một tập tin thực thi để chạy tập tin Python có nội dung như sau:

```
#!/bin/bash  
cd /home/pi/Desktop/StartupRasp      # Duong dan den tap tin .py  
python -u LED_flashLoop.py           # Chay chuong trinh
```

Cần lưu ý rằng, tập tin thực thi này phải được lưu trữ vào mô-đun Raspberry với phần tên do chúng ta tự đặt tùy ý, nhưng phần mở rộng thì bắt buộc phải là “.sh”. Giả sử ở đây ta sẽ đặt tên cho tập tin thực thi này là “AutorunProg.sh”. Tiếp theo, ta sẽ tạo một tập tin khai báo có nội dung như sau:

```
[program: AutorunProg]      # Ten chuong trinh dat tuy y  
command=/home/pi/Desktop/StartupRasp/AutorunProg.sh  
autostart=true                # Duong dan den tap tin .sh  
autorestart=true  
startsecs=60                   # Khoi dong sau 60 giay khi he dieu hanh khoi dong xong  
stderrLogFile=/home/pi/Desktop/StartupRasp/Log/error.Log    # Luu tap tin Err Log  
stdoutLogFile=/home/pi/Desktop/StartupRasp/Log/LogFile.Log    # Luu tap tin Log
```

Cũng tương tự như trên, tập tin khai báo này phải được lưu trữ vào mô-đun Raspberry với phần tên do chúng ta tự đặt tùy ý, nhưng phần mở rộng thì bắt buộc phải là “**.conf**”. Giả sử ở đây ta sẽ đặt tên cho tập tin thực thi này là “**AutorunProg.conf**”. Chú ý thêm là thư mục “**Log**” cần phải được tạo ra trước và nó phải được đặt trong cùng thư mục với tập tin “**.conf**”. Tiếp theo, ta sao chép tập tin “**.conf**” vào thư mục “**/etc/supervisor/conf.d/**”, tại Terminal nhập lệnh như sau:

```
pi@raspberrypi:~/Desktop/StartupRasp sudo cp -f AutorunProg.conf /etc/supervisor/conf.d
```

Tiếp theo, ta sẽ phân quyền cho tập tin “**AutorunProg.conf**” vừa được sao chép vào trong thư mục “**/etc/supervisor/conf.d/**”, tại Terminal nhập lệnh như sau:

```
~$ cd /etc/supervisor/conf.d  
pi@raspberrypi:~/Desktop/StartupRasp sudo chmod 777 Autorun.conf
```

Cuối cùng kiểm tra lại tập tin Python này đã được cho phép hoạt động hay chưa bằng cách tại Terminal nhập lệnh như sau:

```
~$ sudo service supervisor restart  
~$ sudo supervisorctl
```

Nếu hệ thống thông báo chương trình “**LED_flash_loop**” đang hoạt động, quan sát trong cửa sổ Terminal sẽ xuất hiện dòng thông tin “**AutorunProg STARTING**”, thì có nghĩa là mọi thiết lập của chúng ta đã thành công. Cuối cùng ta khởi động lại hệ điều hành của Raspberry Pi và quan sát kết quả đạt được. Thiết lập tương tự cho các chương trình Python khác nếu muốn chúng khởi động cùng hệ điều hành.

Khi chúng ta không còn muốn chương trình Python này được khởi động cùng với hệ điều hành nữa, thì đơn giản chúng ta chỉ cần xóa đi tập tin “**AutorunProg.conf**” vừa mới được sao chép vào trong thư mục “**/etc/supervisor/conf.d/**”.

5. Phương pháp dừng các chương trình Python chạy nền trong hệ điều hành Raspian của Raspberry Pi

Thông thường khi chạy các chương trình Python từ cửa sổ Terminal thì việc dừng thực thi và thoát ra khỏi chu trình làm việc của các chương trình Python này rất đơn giản, chúng ta chỉ cần nhấn tổ hợp phím “**Ctrl + C**” trên bàn phím. Hoặc đơn giản hơn nữa là chúng ta đóng cửa sổ Terminal đó lại.

Tuy nhiên, khi các chương trình Python đã chạy nền trong hệ điều hành Raspian của Raspberry Pi thì việc dừng thực thi và thoát khỏi chu trình làm việc của các chương

trình này không thể thực hiện bằng cách nêu trên. Trong trường hợp này ta cần phải thực hiện một số thao tác lệnh, đầu tiên tại Terminal nhập lệnh như sau:

```
~$ ps -ef|grep python
```

Trên màn hình Terminal sẽ xuất hiện thông tin về các chương trình Python đang chạy nền trong hệ điều hành, một ví dụ minh họa có nội dung như sau:

```
~$ ps -ef|grep python
pi      663      1  0 Jul01 ?        00:00:02 /usr/bin/python3 /usr/bin/blueman-applet
pi      9715    9712  0 10:34 ?        00:00:00 python -u LED_flash_loop.py
pi      9749    9722  0 10:36 pts/0    00:00:00 grep --color=auto python
~$
```

Trong hình minh họa bên trên chúng ta thấy có một chương trình Python đang chạy có tên là "**LED_flash_loop.py**", đặt trường hợp chúng ta muốn dừng thực thi và thoát khỏi chu trình làm việc của chương trình này. Để thực hiện, chúng ta quan sát cột thông tin thứ 2 từ trái sang sẽ thấy có một giá trị số là **9715**, con số này chính là mã PID của chương trình "**LED_flash_loop.py**" đang chạy. Mã PID này sẽ được dùng để bổ sung vào trong lệnh "**kill**" để cho phép dừng thực thi chương trình tương ứng, tại Terminal nhập lệnh như sau:

```
~$ kill -9 9715
```

Sau khi thực thi xong lệnh trên ta có thể kiểm tra lại thì thấy chương trình Python có tên "**LED_flash_loop.py**" đã không còn hoạt động nữa, minh họa chi tiết trong hình bên dưới.

```
~$ kill -9 9715
~$ ps -ef|grep python
pi      663      1  0 Jul01 ?        00:00:02 /usr/bin/python3 /usr/bin/blueman-applet
pi      9888    9722  0 10:56 pts/0    00:00:00 grep --color=auto python
~$
```

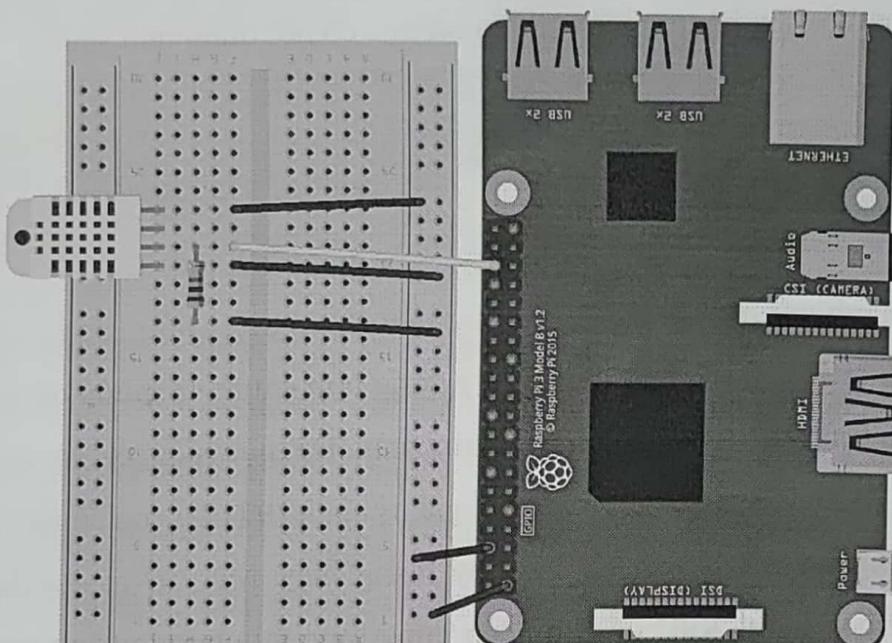
Trong trường hợp chúng ta muốn dừng thực thi tất cả các chương trình Python đang chạy nền thì có thể sử dụng lệnh sau đây, tại Terminal nhập lệnh như sau:

```
~$ pkill -9 python
```

5.6. Bài thí nghiệm 1

Yêu cầu thí nghiệm: Chương trình Python đọc giá trị nhiệt độ và độ ẩm từ cảm biến DHT22 và gửi dữ liệu này lên Server ThingSpeak sau mỗi 30 giây.

5.6.1. Sơ đồ kết nối



Hình 5.1. Sơ đồ kết nối cảm biến DHT22 với Raspberry Pi.

5.6.2. Mã nguồn

Đoạn mã nguồn mẫu dùng để đọc giá trị nhiệt độ và độ ẩm từ cảm biến DHT22 và gửi dữ liệu này lên Server ThingSpeak sau mỗi 30 giây bằng Python:

```
from urllib import request, parse
from time import sleep
from sense_hat import DHT
from datetime import datetime

Ver = 1.00
sensor = DHT('22', 16)

# Channel ID: 794872
# Author: iotLabiu
# User API Key: YUPXLTFPBN4TLX9P
# MQTT API Key: IZWFYCHQJKAHM2MU
# API Key (Write): VN9L7BBX6YI4LWJ3
# API Key (Read): VSOKXS2QDTNC3Z30

# CTC tao chuoi du lieu de gui Len Server
### Input - data - Du Lieu can gui Len Server (noi Luu du Lieu tren Server La "field1")
### Output - params - Chuoi du Lieu duoc tao ra de gui Len Server
def make_param_thingspeak(temp, hum):
    params = parse.urlencode({'field1': temp, 'field2': hum, }).encode()
    return params

# CTC gui chuoi du Lieu Len Server ThingSpeak IoT
```

```

### Input - params - Chuỗi du liệu cần gửi lên Server ThingSpeak IoT
### Output - response_data - Chuỗi du liệu nhận về từ Server ThingSpeak IoT
def thingspeak_post(params):
    # Tao Header trong giao thục HTTP POST
    api_key_write = "U9D4Y90UYD5JYNAJ"
    req = request.Request('https://api.thingspeak.com/update', method="POST")
    req.add_header("Content-Type", "application/x-www-form-urlencoded")
    req.add_header("X-THINGSPEAKAPIKEY", api_key_write)
    r = request.urlopen(req, data = params)
    response_data = r.read()
    return response_data

while True:
    humidity, temperature = sensor.read()

    while(humidity >=100):
        sleep(1)
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

    mytime = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    print mytime
    print "Temperature =",round(temperature,2), " Humidity =",round(humidity,2)

    #ThingSpeak
    params_thingspeak = make_param_thingspeak(round(temperature,2), round(humidity,2))
    thingspeak_post(params_thingspeak)

    sleep(30)  #30s

```

5.7. Bài thí nghiệm 2

Yêu cầu thí nghiệm: Chương trình Python đọc dữ liệu từ Server ThingSpeak sau mỗi 10 giây.

5.7.1. Sơ đồ kết nối

Thực hiện kết nối tương tự bài thí nghiệm 1 bên trên.

5.7.2. Mã nguồn

Đoạn mã nguồn mẫu dùng để đọc dữ liệu từ Server ThingSpeak sau mỗi 10 giây bằng Python:

```

from urllib import request, parse
from time import sleep
from datetime import datetime
import RPi.GPIO as GPIO

# Channel ID: 794872
# Author: iotlabiuh
# User API Key: YUPXLTFPB4TLX9P
# MQTT API Key: IZWFYCHQJKAHM2MU
# API Key (Write): VN9L7BBX6YI4LWJ3
# API Key (Read): VSOKXS2QDTNC3Z30

# CTC nhận dữ liệu cuối cùng được gửi vào Field_1 trên Server ThingSpeak IoT
### Input -
### Output - value - Dữ liệu cuối cùng được gửi vào Field_1 trên Server ThingSpeak IoT
def thingspeak_get():
    api_key_read = "VSOKXS2QDTNC3Z30"
    channel_ID = "794872"

```

```

req = request.Request("https://api.thingspeak.com/channels/%s/fields/1>Last.json?api_key=%s" % (channel_ID,api_key_read), method="GET")
r = request.urlopen(req)
responce_data = r.read().decode()
data1 = json.loads(responce_data)

req = request.Request("https://api.thingspeak.com/channels/%s/fields/1>Last.json?api_key=%s" % (channel_ID,api_key_read), method="GET")
r = request.urlopen(req)
responce_data = r.read().decode()
data2 = json.loads(responce_data)

value1 = data1['field1']
value2 = data2['field2']

return value1,value2

while True:
    #Insys Relay Status
    mytime = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    value1,value2 = thingspeak_get()

    print mytime
    print value1,value2

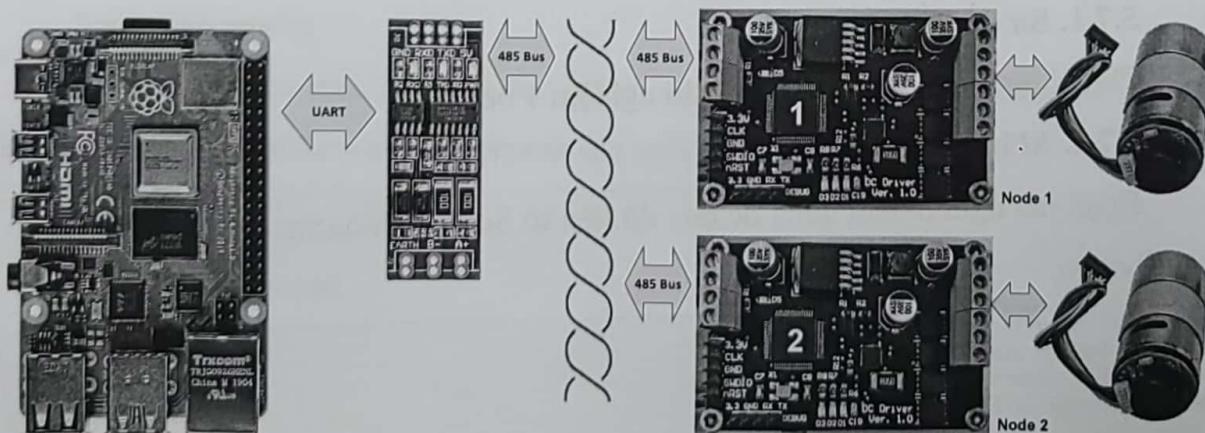
    sleep(10)

```

5.8. Bài thí nghiệm 3

Yêu cầu thí nghiệm: Chương trình Python điều khiển động cơ DC thông qua giao thức RS485.

5.8.1. Sơ đồ kết nối



Hình 5.2. Sơ đồ kết nối động cơ với Raspberry Pi thông qua giao thức RS485.

Với giao thức kết nối RS485 thì ta chỉ cần nối dây A, B của các Node vào chung với đường bus A, B của hệ thống (lưu ý rằng A nối với A và B nối với B). Định dạng dữ liệu (Frame) truyền từ Raspberry Pi xuống các Node như sau:

Start Byte	Node Address	Data High	Data Low
"k"	0	0	0 – 100

Trong đó:

- Start Byte: Ký tự để nhận dạng bắt đầu 1 Frame dữ liệu và được mặc định là ký tự “k”.
- Node Address: Địa chỉ của Node cần điều khiển.
- Data High: Dữ liệu (Byte cao) gửi đến Node cần điều khiển, trong ví dụ này nó sẽ được mặc định giá trị 0.
- Data Low: Dữ liệu (Byte thấp) gửi đến Node cần điều khiển, trong ví dụ này nó sẽ có giá trị từ 0 đến 100 tương ứng với sự thay đổi tốc độ từ 0% (min) đến 100% (max).

5.8.2. Mã nguồn

Đoạn mã nguồn mẫu dùng để điều khiển tốc độ động cơ DC bằng Python:

```
import serial

ser = serial.Serial(
    port='/dev/ttyS0',
    baudrate = 115200,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)
ser.write('k')
mybytes = bytearray([1, 0, 100])      # Node 01 chạy 100% tốc độ
ser.write(mybytes)

ser.write('k')
mybytes = bytearray([2, 0, 50])       # Node 02 chạy 50% tốc độ
ser.write(mybytes)
```

5.9. Bài tập củng cố kiến thức

Yêu cầu bài tập:

- Viết chương trình giám sát các thông số nhiệt độ, độ ẩm và điều khiển bật/tắt đèn, điều khiển tốc độ động cơ DC bằng Raspberry Pi.
- Các thông số được gửi lên Server ThingSpeak mỗi 10s.
- Xây dựng giao diện người dùng trên nền HTML.

Gợi ý thực hiện:

- Bước 1: Tạo Server ThingSpeak để lưu hai tập dữ liệu nhiệt độ và độ ẩm; hai giá trị để điều khiển bật/tắt đèn và điều khiển tốc độ động cơ DC.
- Bước 2: Viết mã nguồn bằng Python cho Raspberry Pi dùng để đọc dữ liệu từ cảm biến và gửi dữ liệu lên Server.
- Bước 3: Xây dựng giao diện người dùng trên nền HTML.
- Bước 4: Chạy thử chương trình và kiểm tra các tính năng, tìm và khắc phục các lỗi phát sinh.

Báo cáo kết quả thực hiện bài tập:

- Trình bày sơ đồ kết nối phần cứng cho bài thí nghiệm.
- Hướng khắc phục khi bị lỗi cảm biến (đọc sai, mất dữ liệu).
- Hướng khắc phục khi bị mất kết nối internet.
- Hướng khắc phục khi cơ cấu chấp hành bị lỗi.
- Trong báo cáo phải thể hiện kết quả của ít nhất 30 phút truyền dữ liệu lên Server liên tục, các trường hợp khi bị lỗi cảm biến, mất kết nối internet khi đang chạy. Tốc độ giao tiếp nhanh nhất giữa Raspberry Pi và Server ThingSpeak là bao nhiêu?
- Trình bày lưu đồ giải thuật, mã nguồn Python trong Raspberry Pi.
- Mã nguồn HTML giao diện người dùng.

5.10. Báo cáo thực hành

Báo cáo được thực hiện theo nhóm thực hành và nộp về cho giáo viên dưới dạng **một tập tin Word** theo yêu cầu:

- Nội dung thực hiện báo cáo thì sinh viên xem trong "**Tài liệu báo cáo thực hành môn Internet vạn vật**".
- Tải tập tin báo cáo lên trang web E-Learning (OCW FET-IUH) của Khoa Công nghệ Điện tử.
- Gửi tập tin báo cáo về địa chỉ mail của giáo viên hướng dẫn.
- **Tiêu đề mail có dạng: IOT_x_Nhom_n_Bai_m**

Trong đó: **x** là ký hiệu buổi học (CT2, ST5, CT5 hoặc CT6).

n là số nhóm thực hành (1, 2, 3,...).

m = 6: Bài “Lập trình ứng dụng IoT cơ bản với Raspberry Pi”.

5.11. Tóm tắt nội dung bài học

Nội dung chính của bài học bao gồm:

- Giới thiệu lập trình ứng dụng IoT cơ bản với Raspberry Pi.
- Xây dựng giao diện web cơ bản trên nền tảng HTML.
- Xây dựng giao diện web cơ bản trên nền tảng Node-RED.
- Cách thức nhúng đồ thị dữ liệu trên Server ThingSpeak vào Web.
- Cách thức tạo thanh trượt trên web và gửi dữ liệu lên Server ThingSpeak.
- Phương pháp làm cho các chương trình Python tự động thực thi khi khởi động Raspberry Pi.
 - Sử dụng phương pháp “rc.local”.
 - Sử dụng phần mềm “Supervisor”.
- Phương pháp dùng các chương trình Python chạy nền trong hệ điều hành Raspian của Raspberry Pi.

2.6. Báo cáo thí nghiệm Buổi 6 – Lập trình ứng dụng IOT cơ bản với Raspberry Pi:

Đọc tài liệu hướng dẫn thực hành và thực hiện các yêu cầu bài tập sau:

- **Bài tập mức độ 1 (3 điểm):**

- Web: Viết giao diện Web có 2 nút nhấn (ON và OFF) và đồ thị thể hiện hai giá trị nhiệt độ và độ ẩm đọc được từ Server.
- Raspberry: Sử dụng mô-đun Raspberry Pi kết hợp với Grove Base Hat và các mô-đun Grove phù hợp để viết chương trình thực hiện việc đọc giá trị nhiệt độ, độ ẩm của môi trường và gửi các dữ liệu này lên Server của ThingSpeak mỗi 20 giây, liên tục trong ít nhất là 15 phút. Ngoài ra, mô-đun Raspberry phải thực hiện giao tiếp với Server để đọc về trạng thái hiện tại của nút nhấn từ trang Web để điều khiển bật/tắt LED đơn.

Sau đó, thực hiện các bước để *chương trình có khả năng tự khởi động khi bật nguồn cho mô-đun Raspberry*.

- **Bài tập mức độ 2 (7 điểm):**

- Web: Viết giao diện Web có
 - 4 nút nhấn (ON, OFF, Auto và Manual). Trong đó: Nút 1 và 2 để chọn chế độ Auto/Manual; Nút 3 và 4 để điều khiển bật/tắt LED đơn.
 - Đồ thị thể hiện hai giá trị nhiệt độ và độ ẩm đọc được từ Server.
 - Có các biểu tượng để thể hiện trạng thái hiện tại của LED là đang bật hoặc tắt.
 - Có cửa sổ hiển thị con số kích thước lớn giá trị nhiệt độ và độ ẩm đọc được từ Server (hiển thị giá trị của lần cập nhật cuối cùng).
 - Có cửa sổ hiển thị thời gian hiện tại.

- Raspberry: Sử dụng mô-đun Raspberry Pi kết hợp với Grove Base Hat và các mô-đun Grove phù hợp để viết chương trình thực hiện việc đọc giá trị nhiệt độ, độ ẩm của môi trường và gửi các dữ liệu này lên Server của ThingSpeak mỗi 20 giây, liên tục trong ít nhất là 30 phút. Ngoài ra, mô-đun Raspberry phải thực hiện giao tiếp với Server để đọc về trạng thái hiện tại của nút nhấn từ trang Web để điều khiển:
 - Nếu ở chế độ Manual, sẽ bật/tắt LED đơn từ điều khiển trên giao diện.
 - Nếu ở chế độ Auto, LED sáng trong khoảng thời gian từ 18h đến 22h, còn lại LED tắt.

Sau đó, thực hiện các bước để *chương trình có khả năng tự khởi động khi bật nguồn cho mô-đun Raspberry*.

- **Bài tập mức độ 3 (10 điểm):**

Sử dụng hai mô-đun Raspberry Pi kết hợp với Grove Base Hat và các mô-đun Grove phù hợp để viết chương trình điều khiển như sau:

- Web: Viết giao diện Web có:
 - 4 nút nhấn (ON, OFF, Auto và Manual). Trong đó: Nút 1 để chọn chế độ Auto/Manual; Nút 2, 3, 4 để điều khiển LED đơn, Buzzer và Relay.
 - Đồ thị thể hiện hai giá trị nhiệt độ và độ ẩm đọc được từ Server.
 - Có các biểu tượng để thể hiện trạng thái hiện tại của LED đơn, Buzzer và Relay là đang bật hoặc tắt.
 - Có cửa sổ hiển thị con số kích thước lớn giá trị nhiệt độ và độ ẩm đọc được từ Server (hiển thị giá trị của lần cập nhật cuối cùng).
 - Có cửa sổ hiển thị thời gian hiện tại.

- **Mô-đun Raspberry 1:** Thực hiện việc đọc giá trị nhiệt độ, độ ẩm của môi trường. Đồng thời gửi dữ liệu nhiệt độ, độ ẩm này lên Server của ThingSpeak mỗi 20 giây cho một gói tin và thực hiện liên tục trong ít nhất là 30 phút. Tùy chọn sử dụng giao thức HTTP hoặc MQTT. Chương trình bắt buộc phải tự khởi động lại (reset) khi xảy ra lỗi cảm biến, rớt mạng.
- **Mô-đun Raspberry 2:** Thực hiện việc đọc dữ liệu nhiệt độ, độ ẩm từ Server và in ra màn hình LCD 16x2. Tùy chọn sử dụng giao thức HTTP hoặc MQTT. Ngoài ra còn có thêm các tính năng sau:
 - Nếu ở chế độ Manual, sẽ bật/tắt LED đơn và Relay từ điều khiển trên giao diện.
 - Nếu ở chế độ Auto:
 - Nếu giá trị nhiệt độ trên 35°C thì bật LED, dưới 31°C thì tắt LED, còn lại các giá trị khác thì giữ nguyên trạng thái hiện tại của LED.
 - Nếu giá trị độ ẩm trên 90% thì bật Relay, dưới 60% thì tắt Relay, còn lại các giá trị khác thì giữ nguyên trạng thái hiện tại của Relay.

Sau đó, thực hiện các bước để **chương trình có khả năng tự khởi động khi bật nguồn cho mô-đun Raspberry**.

Lưu ý:

- Bắt buộc chương trình phải có khả năng tự khởi động khi bật nguồn cho mô-đun Raspberry.
- Bắt buộc chương trình phải tự khởi động lại (reset) khi có lỗi phát sinh trong quá trình hoạt động. Thao tác chạy chương trình thủ công sẽ không được chấp nhận.
- Minh chứng trong báo cáo phải bổ sung thêm phần hình ảnh đồ thị dữ liệu của từng loại cảm biến trên Server ThingSpeak liên tục trong 30 phút.

- Trong cả ba mức độ bài tập, hai gói tin kề nhau thể hiện trong Server có thời gian dưới 90 giây mới được xem là liên tục. Khi không đáp ứng được yêu cầu này thì không được chấp nhận.
- Đối với mức độ 2 và 3, nếu tồn tại dữ liệu bất thường được gửi lên Server sẽ làm mất tính liên tục của dữ liệu. Trong quá trình gửi dữ liệu lên Server, nếu dữ liệu cảm biến bị lỗi thì không được gửi lên Server mà phải đọc lại và gửi dữ liệu khác.