ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

HỌC PHẦN: NHẬP MÔN LẬP TRÌNH KẾT NỐI VẠN VẬT

Chủ đề: Hệ thống theo dõi nhiệt độ và độ ẩm

Giáo viên hướng dẫn:
Thầy Nguyễn Đức Hoàng Hạ

Sinh viên thực hiện:
Nguyễn Duy Quang - 20120360

01/2024

Mục lục

Danh sách hình vẽ

Danh sách mã nguồn

1 Giới thiệu

1.1 Bối cảnh

IoT (Internet of Things) là một lĩnh vực đang nổi lên trong thời gian gần đây. Nhìn chung, thay vì chỉ xem rằng mỗi thiết bị là độc lập, cô lập lẫn nhau thì IoT sẽ kết nối các thiết bị lại với nhau, giúp các thiết bị có khả năng giao tiếp và trao đổi thông tin lẫn nhau. Điều này giúp:

- Tự động hóa: Các thiết bị có thể tự trao đổi và vận hành mà không cần sự nhúng tay của con người.
- Thời gian thực: Bởi vì thông tin đều được xử lý bằng máy móc, thời gian xử lý thông tin sẽ giảm đi đáng kể, dẫn đến việc nhiều tác vụ có thể thực hiện thời gian thực.
- Tối ưu hóa: Ta có thể thiết kế một hệ thống thông minh từ các thiết bị, giúp tạo ra các dụng cụ thông minh và có tính tối ưu cao hơn.

Ở một hướng khác, nhiệt độ và độ ẩm thường là hai yếu tố thường bị bỏ qua, cho dù cả hai yếu tố trên đều quan trọng trong việc đảm bảo chất lượng cuộc sống. Chúng đều có ảnh hưởng lên máy móc là một thành phần quan trọng trong cuộc sống của mỗi người, và bản thân nhiệt độ và độ ẩm cũng là một phần không nhỏ trong đời sống.

1.2 Mục tiêu của dự án

Mục tiêu mà dự án nhắm đến là:

- Ứng dụng các kiến thức đã học từ học phần.
- Hệ thống đơn giản, dễ dàng tái hiện.
- Hệ thống có thể ứng dụng được phần nào trong thực tế.
- Hệ thống có chi phí thấp, nằm trong giới hạn của sinh viên.

2 Đặc tả yêu cầu

Ta có các yêu cầu chức năng sau:

- Yêu cầu cảm biến: Bao gồm các yêu cầu về thiết bị đầu vào (cảm biến).
- Yêu cầu máy chủ: Bao gồm các yêu cầu về máy chủ (Flask).
- Yêu cầu giao diện: Bao gồm các yêu cầu về giao diện hiển thị cho người dùng.

Ngoài ra, đi cùng với đó là các yêu cầu phi chức năng:

- Hệ thống có khả năng mở rộng trong tương lai.
- Hệ thống có cập nhật thông tin theo thời gian thực.
- Hệ thống có tính bảo mật.

2.1 Yêu cầu chức năng

2.1.1 Yêu cầu cảm biến

Yêu cầu này bao gồm các yêu cầu về thiết bị đầu vào của hệ thống, trong đó:

- Cảm biến cần phải có khả năng đọc được thông tin về nhiệt độ và độ ẩm của môi trường một cách liên tục và đều đặn một cách ổn định.
- Cảm biến cần đọc thông tin một cách chính xác, với sai số thấp và nằm trong khoảng ước lượng được.
- Cảm biến cần có khả năng trao đổi thông tin với máy chủ một cách đều đặn và ổn định.

2.1.2 Yêu cầu máy chủ

Yêu cầu này bao gồm các yêu cầu về máy chủ của hệ thống, trong đó:

- Máy chủ cần phải có khả năng hoạt động ổn định.
- Máy chủ cần phải có khả năng nhận được thông tin từ cảm biến một cách đều đặn và ổn định.
- Máy chủ cần phải có khả năng hiểu được thông tin từ cảm biến.
- Máy chủ cần phải có khả năng cung cấp thông tin cho thành phần giao diện một cách đều đặn và ổn định.

2.1.3 Yêu cầu giao diện

Yêu cầu này bao gồm các yêu cầu về giao diện cho người dùng, trong đó:

- Giao diện cần phải dễ hiểu, dễ tương tác cho người dùng.
- Giao diện cần phải cập nhật thông tin từ máy chủ một cách đều đặn và ổn định.
- Giao diện cần phải cung cấp thông tin từ máy chủ cho người dùng một cách dễ hiểu.

2.2 Yêu cầu phi chức năng

Ngoài các yêu cầu chức năng đã nêu trên, ứng dụng còn cần các yêu cầu phi chức năng khác.

- Hệ thống có khả năng mở rộng trong tương lai: Yêu cầu này bao gồm theo nhiều hướng: Yêu cầu về phần cứng (thêm cảm biến, vi điều khiển mạnh hơn...) và phần mềm (các yêu cầu về hệ thống máy chủ), quy mô (kết hợp nhiều loại cảm biến thành một hệ thống)...
- Hệ thống có cập nhật thông tin theo thời gian thực: Thông tin từ ứng dụng phải đáng tin cậy. Tức là ngoài việc yêu cầu thông tin được cung cấp từ ứng dụng phải chính xác, thông tin cần phải được cập nhật thường xuyên để phù hợp với môi trường hiện tại.
- Hệ thống có tính bảo mật: Yêu cầu này nhắm tới bảo mật thông tin của hệ thống. Tức là: Thông tin của hệ thống chỉ có thể do một thành phần nào đó trong hệ thống đọc được, thông tin không thể đọc được nếu đó không phải là một thành phần trong hệ thống.

3 Thiết kế hệ thống

3.1 Phần cứng

Các yêu cầu phần cứng tối thiểu cần phải có là:

- Một vi điều khiển: vi điều khiển yêu cầu cần phải có khả năng kết nối với cảm biến, bao gồm trao đổi và hiểu được thông tin từ cảm biến. Ngoài ra, vi điều khiển còn cần có khả năng trao đổi thông tin với máy chủ. Trong ứng dụng này, vi điều khiển được chọn là ESP32 vì các lý do:
  - Khả năng xử lý ổn.
  - Có khả năng kết nối Bluetooth hoặc WiFi.
  - Tiêu hao thấp.
  - Cho phép xử lý nhiều loại thông tin khác nhau, phù hợp với việc mở rộng hệ thống.
  - Là một thiết bị phổ biến, có nhiều tài liệu hỗ trợ.
  - Giá thành phù hợp túi tiền sinh viên (xấp xỉ 200.000 VNĐ).
- Một cảm biến: Cảm biến phải có khả năng đọc được nhiệt độ và độ ẩm, và phải đọc được nó một cách đều đặn và ổn định. Trong ứng dụng này, cảm biến được chọn là DHT11 bởi vì giá thành của nó.
- Một hệ thống mạng.
- Một máy chủ.
- Thiết bị của người dùng.

Ngoài ra, các phần cứng còn được sử dụng trong ứng dụng là:

- Board cắm.
- Mạch chuyển.

3.2 Phần mềm

Hai phần mà ta cần chú ý ở đây là ứng dụng dành cho phần mềm của bộ vi điều khiển và ứng dụng dành cho máy chủ, vì vậy các công nghệ được sử dụng là:

- Arduino: Là một nền tảng phổ biến và thân thiện với người mới, Arduino là một lựa chọn hàng đầu cho các hệ thống nhỏ và vừa. Ngoài ra, Arduino còn có một môi trường phát triển dễ dùng

cho người mới (Arduino IDE) cùng với một cộng đồng lớn, nhiều tài liệu tham khảo. Arduino IDE là phần mềm được khuyên dùng.

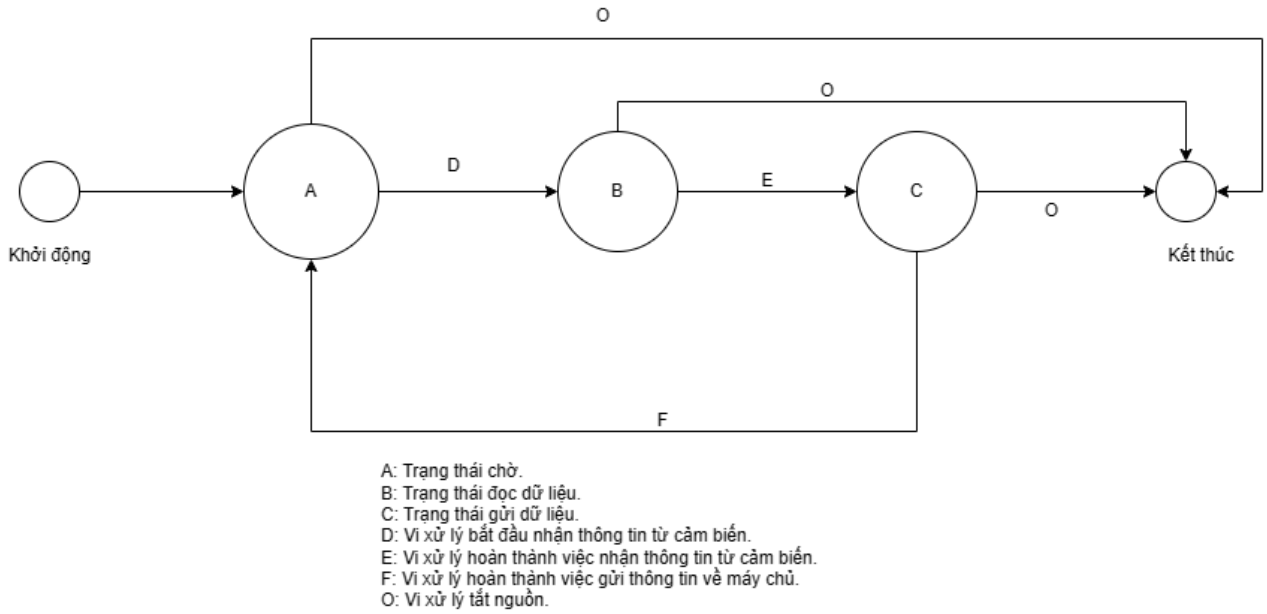
- Flask: Flask là một thư viện dành cho thiết kế web sử dụng Python. Flask nhẹ, đơn giản, tối giản nhưng lại có khả năng mở rộng tốt, là một lựa chọn cho việc thiết kế các máy chủ web từ nhỏ, vừa và lớn. Visual Studio hoặc Visual Studio Code là hai phần mềm được khuyên dùng, hoặc Pycharm là một lựa chọn chuyên sâu hơn dành cho những người đã thành thạo Python.

3.3 Mô tả hệ thống

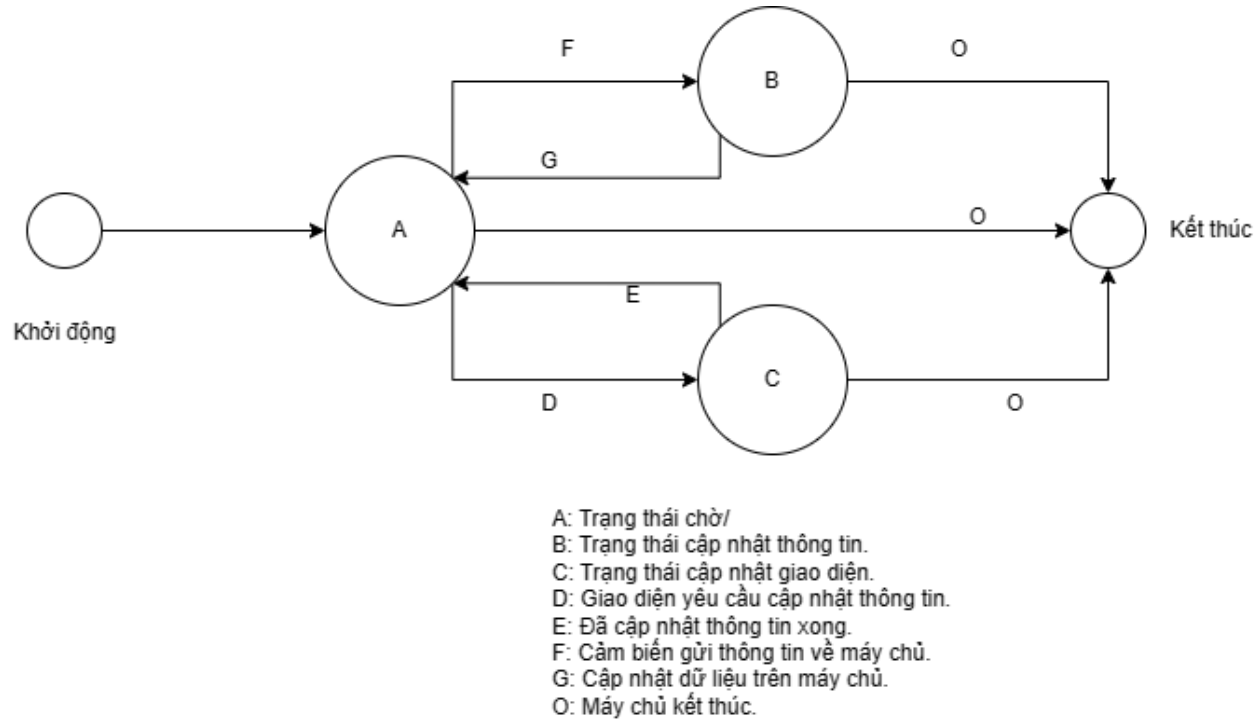
Dựa theo yêu cầu chức năng của ứng dụng, chúng ta sẽ chia hệ thống làm ba thành phần.

- Cảm biến: Thành phần này chịu trách nhiệm thu thập thông tin từ môi trường và gửi đến máy chủ.
- Máy chủ: Thành phần này chịu trách nhiệm xử lý thông tin, và gửi thông tin cho giao diện.
- Giao diện: Thành phần này chịu trách nhiệm hiển thị thông tin cho người dùng.

3.4 Sơ đồ FSM (Finite State Machine)



Hình 1: Sơ đồ FSM của cảm biến



Hình 2: Sơ đồ FSM của máy chủ

4 Cài đặt

Chú thích:

4.1 Mạch vi điều khiển

4.2 Mã nguồn vi điều khiển

Ứng dụng sử dụng các thư viện sau để chạy nên hãy đảm bảo ứng dụng đã được cài đặt các thư viện trước khi chạy.

Mã nguồn 1: Thư viện

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <WiFi.h>
#include <HttpClient.h>
```

Để vi điều khiển có thể đọc thông tin từ cảm biến và gửi thông tin đến máy chủ, ta sẽ cần tối thiểu các biến/tham số sau.

Mã nguồn 2: Khởi tạo

```
DHT_Unified dht(DHTPIN, DHTTYPE); // Sensor
uint32_t delayMS; // Delay time after request
```

```
unsigned long lastTime = 0; // Last time send request
float temperature;
float humidity;
```

Ta có thể lấy thông tin từ cảm biến bằng cách

Mã nguồn 3: Lấy thông tin cảm biến

```
sensors_event_t event;
dht.temperature().getEvent(&event);
if (isnan(event.temperature)) {
    Serial.println(F("Error reading temperature!"));
}
else {
    temperature = event.temperature;
}
```

Và gửi thông tin đến máy chủ thông qua một request. Để đơn giản, ta sẽ gửi thông tin bằng request GET thông qua các thông số.

Mã nguồn 4: Gửi thông tin

```
String data = "?temperature=" + String(temperature) + "&humidity=" + String(
    humidity);
String sendingToServer = "http://" + serverName + ":" + port + "/update/" + data
    ;
client.beginRequest();
client.get(sendingToServer);
client.endRequest();
```

Chú ý: Có thể tách riêng các phần định nghĩa cổng, loại cảm biến hay thông tin mạng thành một file tiêu đề (.h) và sử dụng chúng như môi trường để tăng tính bảo mật.

Mã nguồn 5: Môi trường

```
// secret.h
#define DHTPIN 2
#define DHTTYPE DHT11

const char* ssid = "Duy Thong";
const char* password = "12345678";
const String serverName = "192.168.1.9";
const int port = 5000;
```

4.3 Mã nguồn máy chủ

Ta sử dụng một module dữ liệu đơn giản để làm nguồn dữ liệu toàn cục cho máy chủ.

Mã nguồn 6: Module dữ liệu

```
// modules/data.py
data = {
    'temperature': 0,
    'humidity': 0
}
def update_data(temperature, humidity):
    data['temperature'] = temperature
    data['humidity'] = humidity
def get_data():
    return data
```

Như vậy, khi cập nhật dữ liệu trên máy chủ, ta chỉ cần gọi lại hàm cập nhật...

Mã nguồn 7: Cập nhật dữ liệu

```
// modules/dht/__init__.py
from flask import Blueprint, request
from modules.data import update_data
bp = Blueprint('dht', __name__)
@bp.route('/update/', methods=['GET']) # Get information from the sensor
def get_data():
    update_data(request.args.get('temperature'), request.args.get('humidity'))
    return 'ok', 200
```

... và hàm lấy dữ liệu mỗi khi ta cần.

Mã nguồn 8: Lấy dữ liệu dữ liệu

```
from modules.data import get_data
data = get_data()
```

Để tiện cho việc minh họa, ta sẽ thiết kế máy chủ đảm nhận cả nhiệm vụ giao diện người dùng. Giao diện sẽ được thiết kế đơn giản bằng html cùng với css.

4.4 Thử nghiệm

5 Tổng kết

5.1 Tổng kết ứng dụng

Nhìn chung, ứng dụng đã đạt được các mục tiêu sau:

- Thiết kế được một hệ thống có thể đọc một cách ổn định nhiệt độ và độ ẩm của môi trường.
- Hệ thống có giá trị tương đối rẻ, nằm trong tầm tay của sinh viên (350.000 VNĐ cho hàng mới).
- Hệ thống có khả năng mở rộng.

Ngoài ra, ứng dụng cũng chưa hoàn thành được mục tiêu:

- Hệ thống chưa có tính bảo mật, bởi vì bất cứ thiết bị nào cũng có thể truy cập vào cảm biến để lấy thông tin.
- Hệ thống còn bị giới hạn: Bản thân cảm biến, khoảng cách (yêu cầu nguồn điện cho vi điều khiển và cảm biến...)...

5.2 Cải thiện và phát triển

Một số cải thiện và phát triển của hệ thống.

- Thay đổi phương thức giao tiếp của hệ thống: Nhìn chung, việc gửi thông tin một cách đều đặn như thế này là không phù hợp với phương pháp sử dụng API (API có khả năng phản hồi thấp). Vì vậy, ta có thể:
  - Sử dụng Web Socket để thay thế.
  - Giới hạn tần suất gửi dữ liệu từ cảm biến. Một ví dụ là chỉ gửi dữ liệu khi mà có sự thay đổi về nhiệt độ hay độ ẩm (lưu lại nhiệt độ và độ ẩm trước đó và so sánh với nhiệt độ hay độ ẩm hiện tại).
- Bảo mật hệ thống: Có nhiều cách để bảo mật hệ thống này: Sử dụng các token được quy định sẵn, sử dụng serial thay vì wi-fi...
- Nâng cấp giao diện: Bởi vì nhìn chung, ta có thể coi ứng dụng này như một ứng dụng web nên việc sử dụng các công nghệ từ lập trình web (sử dụng NextJS + Tailwindcss cho frontend) là khả thi.

Tài liệu

[1] Esp32 dht11/dht22 web server using arduino ide | random nerd tutorials. https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-web-server-arduino-ide/.

[2] arduino-libraries/arduinohttpclient: Arduino http client library. https://github.com/arduino-libraries/ArduinoHttpClient.