BÀI THỰC HÀNH SỐ 05
LẬP TRÌNH MỘT SERVER ĐƠN GIẢN VỚI FLASK

1 Yêu cầu

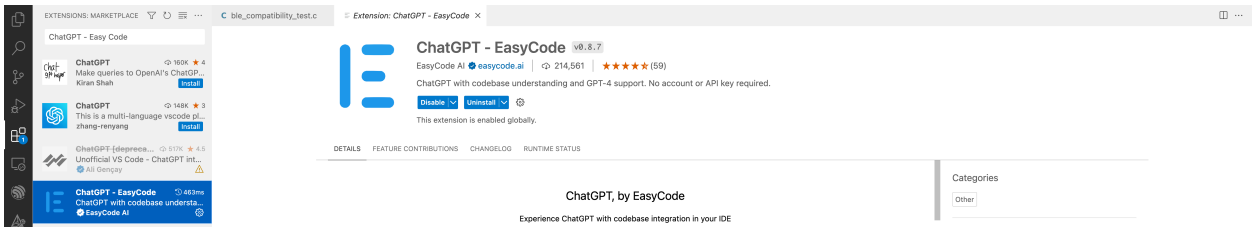
- Sinh viên có kiến thức về lập trình
- Sinh viên có kiến thức cơ bản về ngôn ngữ lập trình python
- Sinh viên có kiến thức cơ bản về ngôn ngữ ký hiệu HTML

2 Tích hợp ChatGPT vào VSCode

2.1 Cài đặt ChatGPT – EasyCode

Vào cuối năm 2022, ChatGPT – một công cụ AI mạnh mẽ đã được công ty OpenAI cho ra mắt phiên bản thử nghiệm. Công cụ này đã tạo nên một cơn sốt trên toàn cầu khi có khả năng sinh ngôn ngữ dựa trên ngữ cảnh vô cùng mạnh mẽ. Không dừng lại ở đó, với bộ dữ liệu được huấn luyện phong phú, ChatGPT có khả năng hỗ trợ lập trình rất tốt.

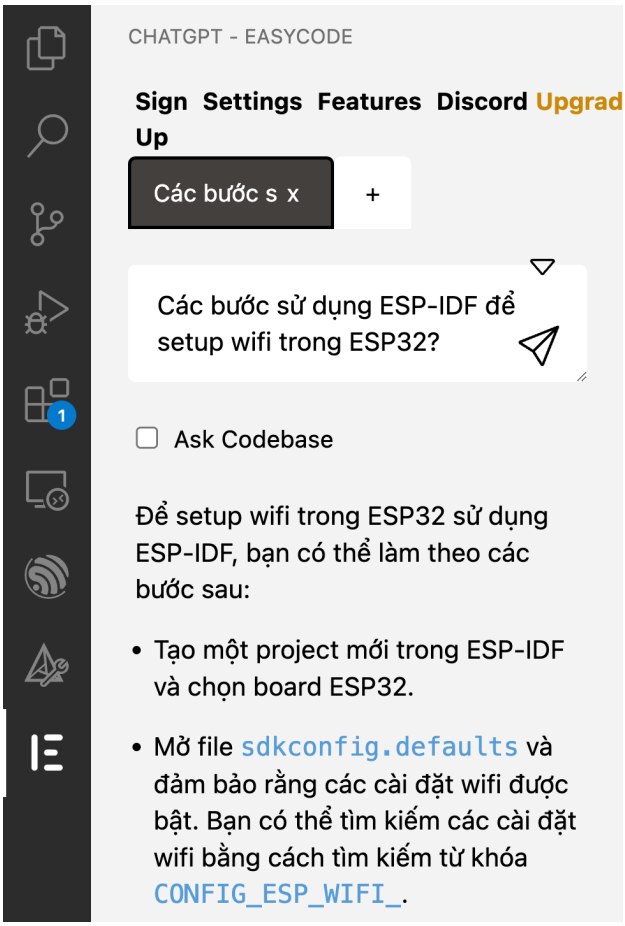
Trong bài thực hành này, chúng ta sẽ thực hiện tích hợp công cụ này vào VSCode để hỗ trợ việc lập trình. Hiện nay, có rất nhiều extension tích hợp ChatGPT trên VSCode, tuy nhiên để thuận tiện trong việc đào tạo, extension ChatGPT – Easy Code được đề xuất do extension này không yêu cầu người dùng phải có tài khoản ChatGPT để sử dụng.



Hình 2.1. Extension ChatGPT - EasyCode trong VS Code

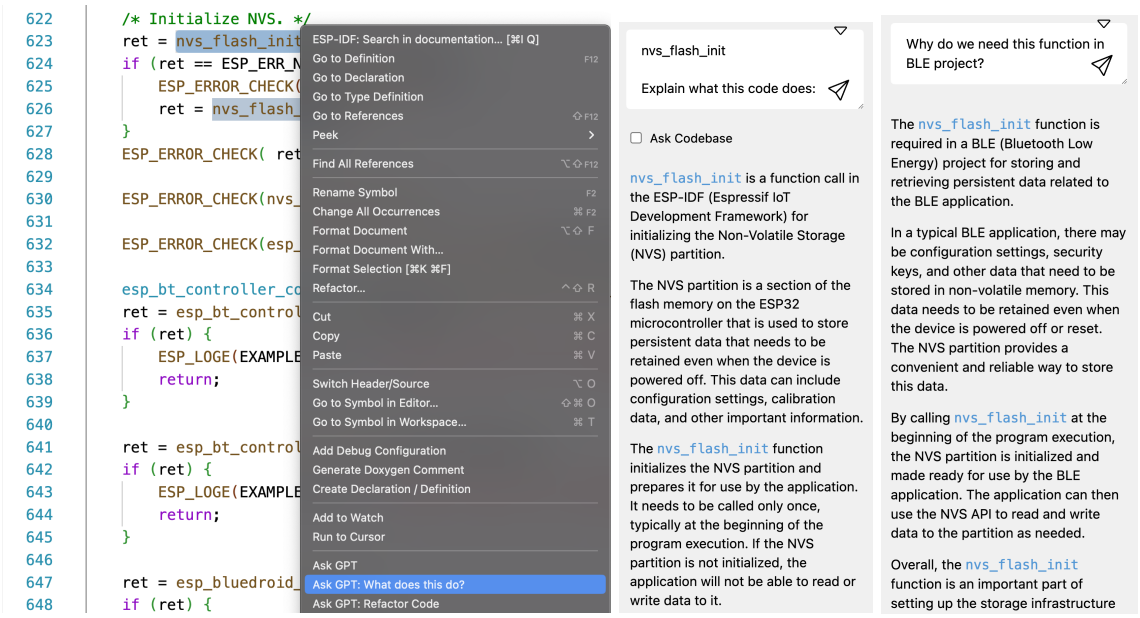
2.2 Sử dụng ChatGPT – EasyCode

Việc sử dụng ChatGPT – EasyCode khá đơn giản, ta chỉ cần bấm vào biểu tượng của ChatGPT – EasyCode, và bắt đầu đặt câu hỏi. Lưu ý rằng, kết quả trả về chỉ là đề xuất của ChatGPT, người dùng cần tự cân nhắc trước khi sử dụng. Bên cạnh đó, hãy luôn nhớ rằng ChatGPT là một công cụ hỗ trợ, chúng ta có thể sử dụng để hỗ trợ xử lý công việc nhanh và hiệu quả hơn, tuy nhiên nếu quá lạm dụng thì sẽ phản tác dụng. Việc hiểu rõ bản chất của vấn đề luôn là yếu tố cốt lõi để có thể làm việc hiệu quả và lâu dài trong mọi lĩnh vực.



Hình 2.2. Sử dụng ChatGPT – EasyCode

Ngoài ra, công cụ này còn hỗ trợ giải thích source code trực tiếp bằng cách chọn đoạn code muốn giải thích, bấm chuột phải và chọn **"Ask GPT: What does this do?"**.



Hình 2.3. Yêu cầu ChatGPT giải thích code

Đối với sinh viên chưa có nhiều kinh nghiệm làm việc với ngôn ngữ Python và HTML, cũng như chưa từng làm việc với server, ChatGPT sẽ làm công cụ hỗ trợ đắc lực để sinh viên có thể hoàn thành một project IoT hoàn chỉnh.

3 Xây dựng ứng dụng web và kết nối đến MQTT broker

3.1 Flask framework

3.1.1 Cài đặt Flask framework

Flask là một framework ứng dụng web nhẹ và mạnh mẽ dành cho ngôn ngữ lập trình Python. Với Flask, người dùng có thể xây dựng các ứng dụng web linh hoạt và mở rộng từ những dự án đơn giản đến những ứng dụng phức tạp.

Với khả năng khởi tạo nhanh chóng và dễ dàng, Flask giúp người dùng triển khai ứng dụng web một cách nhanh nhất có thể. Nó cung cấp một tập hợp các công cụ và thư viện cần thiết để xử lý các yêu cầu và phản hồi HTTP, đồng thời hỗ trợ các công nghệ mạnh mẽ như URL routing, bảo mật,...

Một trong những đặc điểm nổi bật của Flask là tính linh hoạt. Framework không ép buộc bất kỳ phụ thuộc hay cấu trúc dự án cụ thể nào, cho phép người dùng tự do lựa chọn công cụ và thư viện phù hợp với nhu cầu của họ. Điều này giúp tận dụng các thành phần có sẵn trong hệ sinh thái Python và tích hợp chúng vào dự án của mình.

Một điểm mạnh khác của Flask là cộng đồng phát triển rộng lớn xung quanh nó. Có rất nhiều tiện ích mở rộng do cộng đồng cung cấp, giúp người dùng mở rộng chức năng

của ứng dụng một cách dễ dàng. Người dùng có thể tìm thấy các tiện ích cho xác thực người dùng, xử lý biểu mẫu, giao tiếp với cơ sở dữ liệu và rất nhiều lĩnh vực khác.

Để cài đặt Flask, từ terminal của VSCode, ta thực hiện lệnh:

```
pip install -U Flask
```

3.1.2 Tạo một ứng dụng web đơn giản với Flask

Mặc dù Flask không yêu cầu một cấu trúc project cụ thể để triển khai, tuy nhiên, để người mới dễ tiếp cận, ta sẽ sử dụng cấu trúc thư mục sau khi làm việc với Flask.

```
project/
├── app/
│   ├── __init__.py
│   ├── models.py
│   ├── routes.py
│   ├── static/
│   └── templates/
│   └── app.py
├── config.py
├── requirements.txt
```

Trong đó:

- app/ : Thư mục này chứa mã nguồn chính của ứng dụng.
- __init.py__: Khởi tạo ứng dụng Flask và cài đặt các cấu hình cần thiết.
- models.py: Chứa các mô hình cơ sở dữ liệu cho ứng dụng.
- routes.py: Chứa các đường dẫn (routes) và các view cho ứng dụng.
- static/: Chứa các tệp tĩnh như CSS, JavaScript và hình ảnh.
- templates/: Chứa các mẫu HTML cho ứng dụng.
- config.py: Chứa các biến cấu hình cho ứng dụng.
- requirements.txt: Liệt kê các gói Python cần thiết cho ứng dụng.
- app.py: Chạy ứng dụng Flask.

Để tạo một Flask server, đặt chương trình đơn giản dưới đây vào file **app.py**.

```
from flask import Flask

app = Flask(__name__)
```

```
@app.route('/')
def hello():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run()
```

Để khởi động server, ta thực hiện lệnh sau đây trong thư mục project:

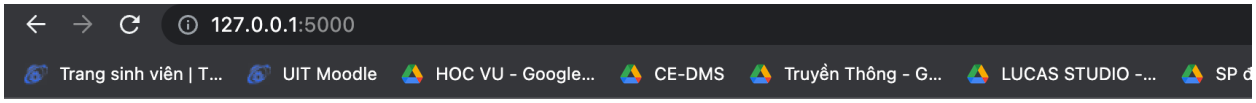
```
python app.py
```

Trong terminal sẽ hiện lên thông tin của server khi chạy như Hình 3.1. Trong hình, ta sẽ thấy project đang chạy với "Debug mode: off". Nếu muốn chạy project với chế độ Debug, ta sửa `app.run()` thành `app.run(debug=True)`.

```
(base) hoangloctran@MacBook-Pro-5 LAB05_FlaskServer % python run.py
 * Serving Flask app "run" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [01/Jun/2023 10:42:32] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/Jun/2023 10:42:32] "GET /favicon.ico HTTP/1.1" 404 -
```

Hình 3.1. Kết quả khi chạy server flask

Để xem kết quả trên web application, mở browser và truy cập vào IP http://127.0.0.1:5000/ ta sẽ có kết quả như Hình 3.2.



Hello, World!

Hình 3.2. Kết quả hiển thị trên web application

3.1.3 Sử dụng template HTML để tạo giao diện đơn giản

Có thể thấy ứng dụng web còn khá đơn giản khi chỉ in ra chữ "Hello, world!" từ lệnh return. Để có thể tận dụng được thế mạnh của một ứng dụng web, ta sẽ thử in ra một

trang HTML đơn giản bằng cách tạo ra một file **index.html** đặt trong thư mục **templates/** với nội dung như bên dưới:

```
<!DOCTYPE html>
<html>
<head>
    <title>My Flask App</title>
</head>
<body>
    <h1>Hello, World!</h1>
</body>
</html>
```

File HTML trên sẽ tạo một trang với tiêu đề là "My Flask App", và in ra dòng chữ "Hello, World!". Ta sẽ sửa lại file app.py để ứng dụng web in ra file HTML trên:

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Trong Flask project, ta sẽ thường xuyên thấy cấu trúc:

```
@app.route('/')
def index():
    return render_template('index.html')
```

@app.route('/') là một decorator giúp xác định đường dẫn URL cho hàm index. Khi người dùng truy cập ứng dụng web (ví dụ thông qua IP: http://127.0.0.1/5000/) thì ứng dụng sẽ căn cứ vào URL mà người dùng truy cập để gọi đến các hàm tương ứng. Trong ví dụ trên, ta có đối số của **@app.route()** là **'/'** có nghĩa là đường dẫn gốc của ứng dụng web. Nói cách khác, nếu người dụng truy cập http://127.0.0.1/5000/ thì hàm **index()** sẽ được gọi.

Hàm `render_template()` có chức năng tạo ra (render) một trang HTML. Đối số của hàm này là ***'index.html'*** để chỉ file **index.html** mặc định nằm trong thư mục **templates/**.

3.2 Kết nối ứng dụng web đến MQTT broker

Mục tiêu của việc sử dụng ứng dụng web chính là giúp người dùng cuối dễ dàng tiếp cận hệ thống IoT. Như vậy, ứng dụng web cần có chế để kết nối đến MQTT broker, từ đó gửi/lấy các dữ liệu đến/từ các cảm biến hoặc bộ điều khiển – vốn đang kết nối với ESP32.

3.2.1 Cài đặt thư viện cần thiết

Để sử dụng giao thức MQTT trên Flask, ta sẽ cần cài đặt thêm một số gói thư viện sau:

- Flask-Socketio: `pip install flask-socketio`
- Flask-MQTT: `pip install flask-mqtt`

3.2.2 Cài đặt MQTT trong Flask project:

- Cài đặt kết nối đến broker:

```
from flask import Flask
from flask_mqtt import Mqtt

app = Flask(__name__)
app.config['MQTT_BROKER_URL'] = 'mybroker.com'
app.config['MQTT_BROKER_PORT'] = 1883
app.config['MQTT_USERNAME'] = 'user'
app.config['MQTT_PASSWORD'] = 'secret'
app.config['MQTT_REFRESH_TIME'] = 1.0  # refresh time in seconds
mqtt = Mqtt(app)

@app.route('/')
def index():
    return render_template('index.html')
```

- Thiết lập xử lý các sự kiện:

```
@mqtt.on_connect()
def handle_connect(client, userdata, flags, rc):
    mqtt.subscribe('home/mytopic')
```

@mqtt.on_connect() là decorator khai báo nếu sự kiện MQTT Connect xuất hiện thì sẽ gọi hàm `handle_connect()`.

Tương tự, ta xử lý sự kiện khi nhận được dữ liệu từ broker:

```
@mqtt.on_message()
def handle_mqtt_message(client, userdata, message):
    data = dict(
        topic=message.topic,
        payload=message.payload.decode()
    )
```

- Gửi thông điệp hoặc theo dõi một topic
  - Để gửi thông điệp, ta dùng hàm: `mqtt.publish('ten_topic', 'thong diep')`
  - Để theo dõi một topic, ta dùng hàm: `mqtt.subscribe('ten_topic')`

Sinh viên có thể tham khảo thêm cách sử dụng thư viện Flask-MQTT tại đây: https://pypi.org/project/Flask-MQTT/

4 BÀI TẬP

Thực hiện mini project.