

Individual Assignment Report

SWE30011-IOT PROGRAMMING

LE MINH LONG - 104180139

Contents

I. Summary:	2
II. Conceptual Design:	2
a. Block Diagrams	2
b. UML Diagrams	4
III. Implementation:	5
a. Sensors:	6
b. Actuators:	8
c. Software/Libraries:	12
IV. Resources:	18
V. Appendix:	19
a. Sketches	19
b. Scripts	19

I. Summary:

The objective of this individual assignment project is to utilize practical knowledge gained during tutorial sessions and self-learning to rapidly prototype an IoT proof-of-concept. The chosen IoT system is an environmental monitoring and control system designed to monitor indoor climate conditions and implement control actions based on sensor data. The system includes sensors to measure temperature, distance, and fire detection, along with alert systems. Additionally, a Raspberry Pi 3 is integrated into the system to serve as the edge device, facilitating data storage, simple edge analytics, and user interface functionalities. The project aims to demonstrate the practical application of IoT principles in developing an environmental monitoring and control system, showcasing its potential in enhancing environmental awareness, safety, and sustainability.

II. Conceptual Design:

This section provides a high-level overview of the system architecture, outlining the key components and their interactions. By conceptualizing the design upfront, we can better understand the system's functionality and plan for its implementation effectively.

a. Block Diagrams

These diagrams will illustrate the hardware and software components involved in the IoT system and how they interact with each other to achieve the desired functionality. With clear visual representations, we can gain insights into the system's structure and identify potential areas for optimization or refinement.

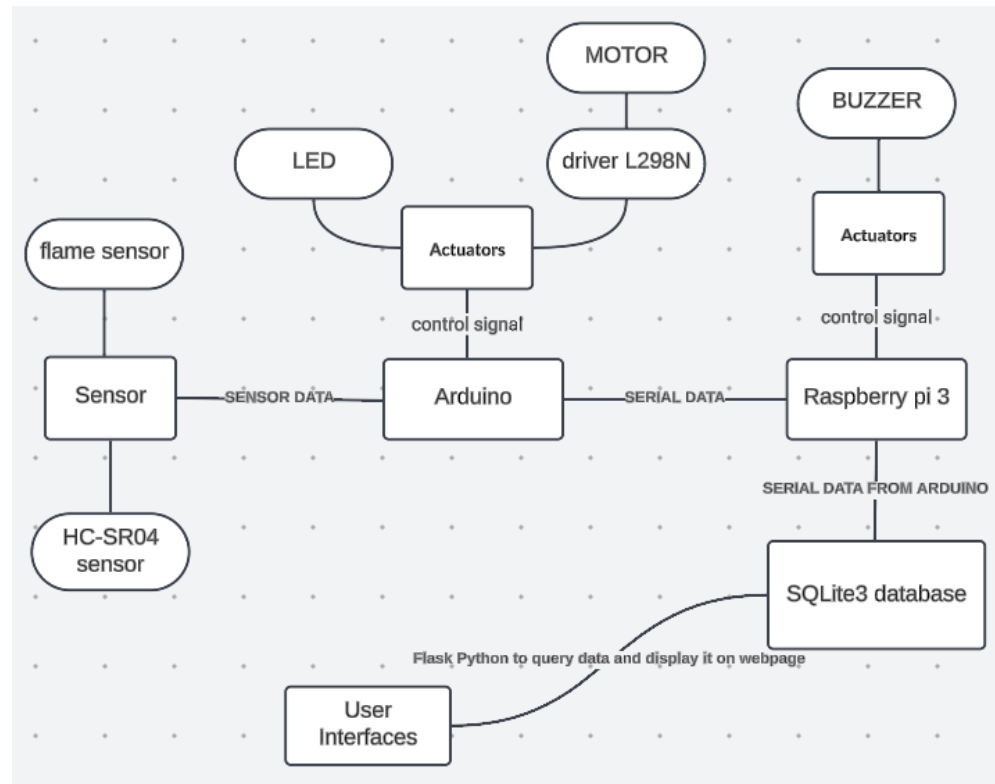


Figure 1 - IOT product block diagram

The IoT environmental monitoring and control system consists of the following components:

- **Arduino Uno:** The central microcontroller unit responsible for interfacing with sensors and actuators.
- **Sensors:** Including temperature sensor, ultrasonic sensor for distance measurement, and flame sensor for fire detection.
- **Actuators:** Consisting of an LED indicator, buzzer for audible alerts, and a motor controlled by an L298N driver module.
- **Raspberry Pi 3:** Serving as the edge device for data storage, analytics, and user interface.
- **Communication Interface:** UART serial communication between Arduino Uno and Raspberry Pi 3 for data exchange.
- **Database:** SQLite database integrated into Raspberry Pi 3 for storing sensor data.
- **Web Interface:** A web application hosted on Raspberry Pi 3 for data visualization and user interaction.

b. UML Diagrams

UML provides a standardized way to represent system components, their relationships, and the flow of activities within the system. By employing use case diagrams, activity diagrams, and class diagrams, we can capture the system's behavior and organization in a concise and structured manner.

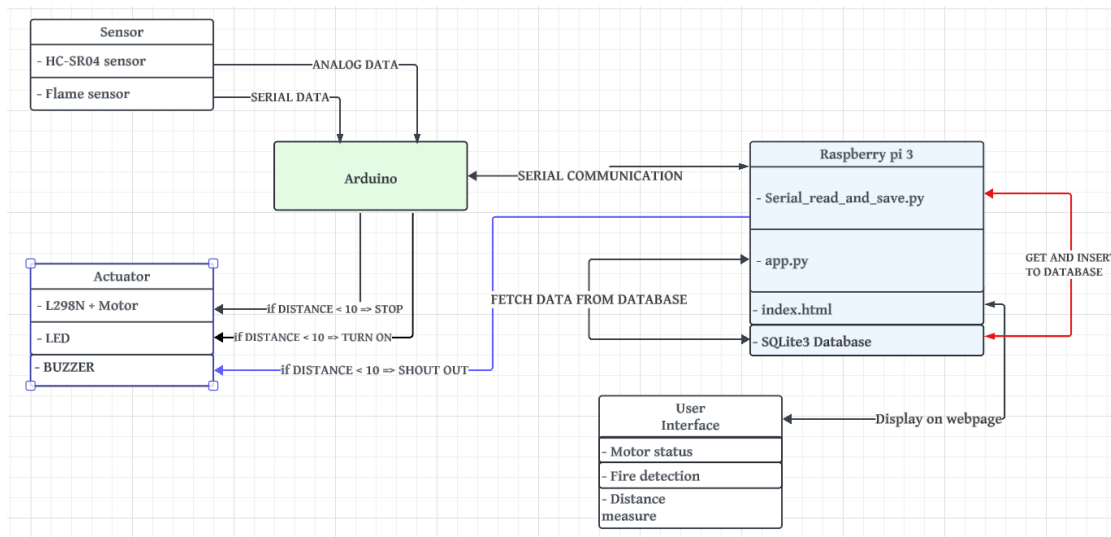


Figure 2 - IOT Node UML Diagram

- **Use case diagram:** The primary use cases include data collection from sensors, data storage in the database, data visualization through the web interface, and control actions based on sensor readings.
- **Activity diagram:** Illustrating the sequence of actions involved in sensor data acquisition, database operations, and user interaction with the web interface.
- **Class diagram:** Representing the classes and their relationships in the system, such as sensor, actuator, database handler, and web server.

III. Implementation:

In implementing the environmental monitoring and control system, a meticulous approach was taken to select and integrate various components, ensuring the system's functionality and reliability. The system comprises sensors, actuators, microcontrollers, and software components, all working seamlessly to monitor environmental conditions and respond appropriately to detected events.

At its core is an Arduino microcontroller, responsible for interfacing with various sensors, including the HC-SR04 ultrasonic sensor for distance measurement and the flame sensor for fire detection. Additionally, actuators such as the LED indicator, buzzer for audible alerts, and motor controlled by the L298N driver module provide real-time feedback and execute control commands. The Raspberry Pi 3 serves as the edge device, facilitating communication between the Arduino and the database for data storage and analysis. A crucial aspect of the system is the communication protocol established between the Arduino and the Raspberry Pi, enabling seamless data exchange.

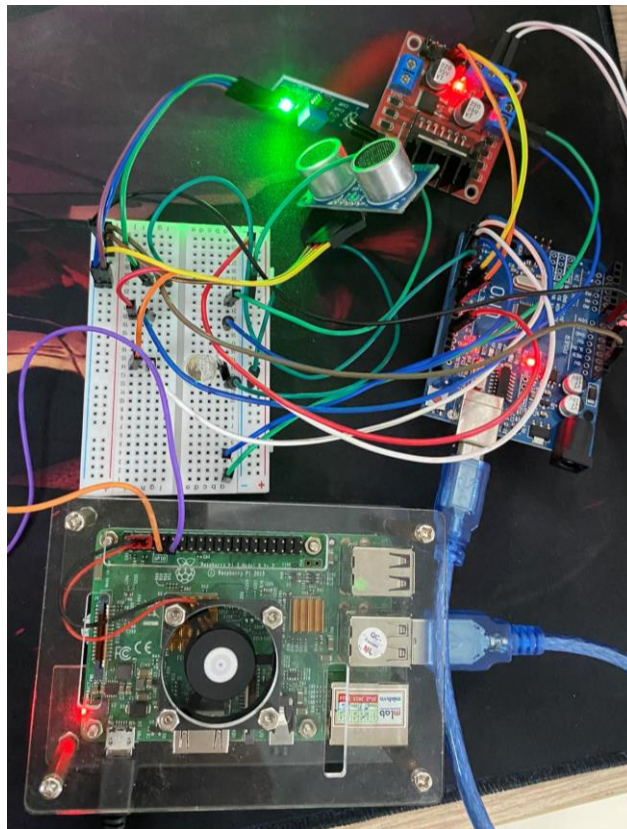


Figure 3 - Full System

a. Sensors:

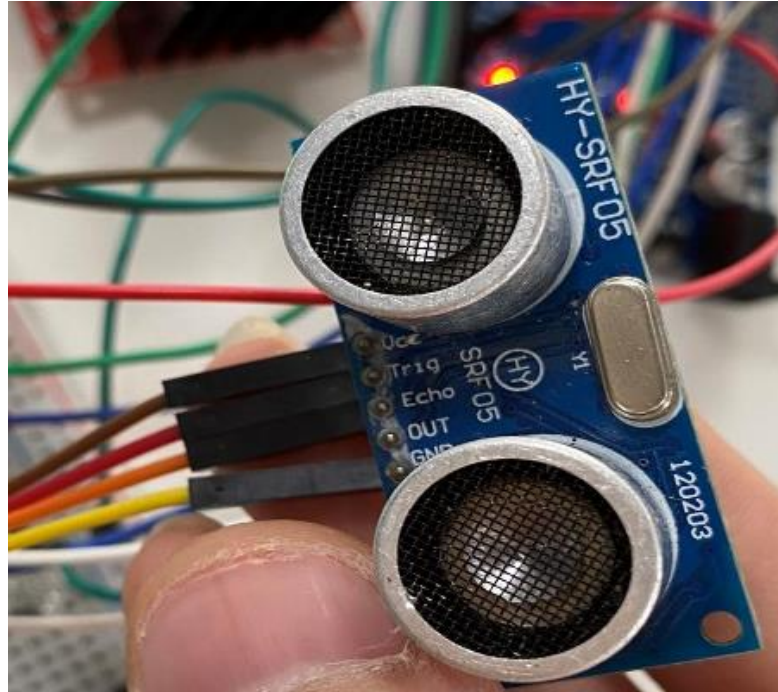


Figure 4 - HC-SR04 sensor

The HC-SR04 ultrasonic sensor plays a vital role in measuring distance within the environment. This sensor operates by emitting ultrasonic waves and calculating the time taken for the waves to return after hitting an object. By measuring the time interval between emission and reception, the HC-SR04 sensor accurately determines the distance to nearby objects with high precision. In the environmental monitoring and control system, the HC-SR04 sensor provides real-time distance data, enabling the system to assess spatial constraints and detect obstacles or obstructions within the monitored area effectively.



Figure 5- *Flame sensor*

The flame sensor serves as a critical component for fire detection within the environment. This sensor detects the presence of fire or flames by analyzing changes in infrared radiation emitted by the surrounding environment. When exposed to flames or intense heat sources, the flame sensor generates electrical signals indicative of potential fire hazards. In the environmental monitoring and control system, the flame sensor serves as an early warning system, alerting users to the presence of fire and facilitating prompt action to mitigate risks and ensure safety. By integrating the flame sensor into the system, the environmental monitoring and control capabilities are enhanced, enabling proactive response to fire-related incidents and improving overall environmental safety.

b. Actuators:

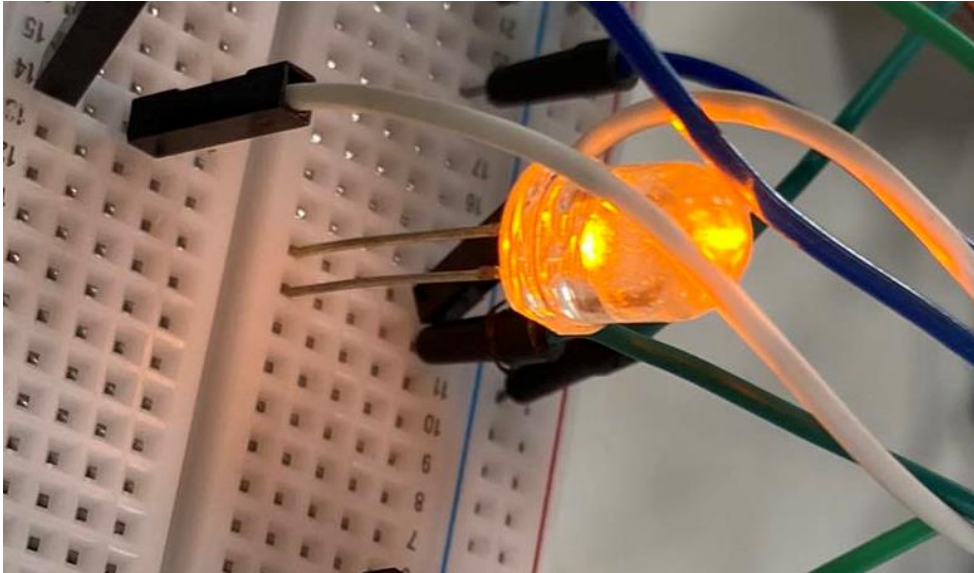


Figure 6 – Led

The LED (Light Emitting Diode) serves as a visual indicator within the environmental monitoring and control system. When activated, the LED emits light, providing a clear visual cue to users regarding system status or detected events. In the context of the system's functionality, the LED serves multiple purposes, including indicating system readiness, signaling alarm conditions such as fire detection or proximity alerts, and conveying operational status during data acquisition and processing. Through appropriate control signals from the microcontroller (Arduino), the LED can be turned on or off dynamically based on predefined conditions or user commands, ensuring effective communication of system information and enhancing user awareness of environmental conditions.



Figure 7 – Buzzer

The buzzer serves as an audible alerting mechanism within the environmental monitoring and control system, designed to provide auditory notifications to users in response to critical events or conditions. Connected to the Raspberry Pi, the buzzer's functionality is triggered by signals received from the Arduino microcontroller, which continuously monitors sensor data and transmits relevant information to the Raspberry Pi for processing. Specifically, when the Arduino detects that the distance measured by the HC-SR04 ultrasonic sensor is less than 10 cm, it sends a corresponding signal to the Raspberry Pi. Upon receiving this signal, the Raspberry Pi evaluates the condition and activates the buzzer to emit audible alerts, signaling the presence of potential hazards or alarm-worthy situations. By leveraging the combined capabilities of the Arduino and Raspberry Pi, the buzzer enhances situational awareness and facilitates timely responses to detected events, thereby contributing to the overall effectiveness and safety of the environmental monitoring and control system.

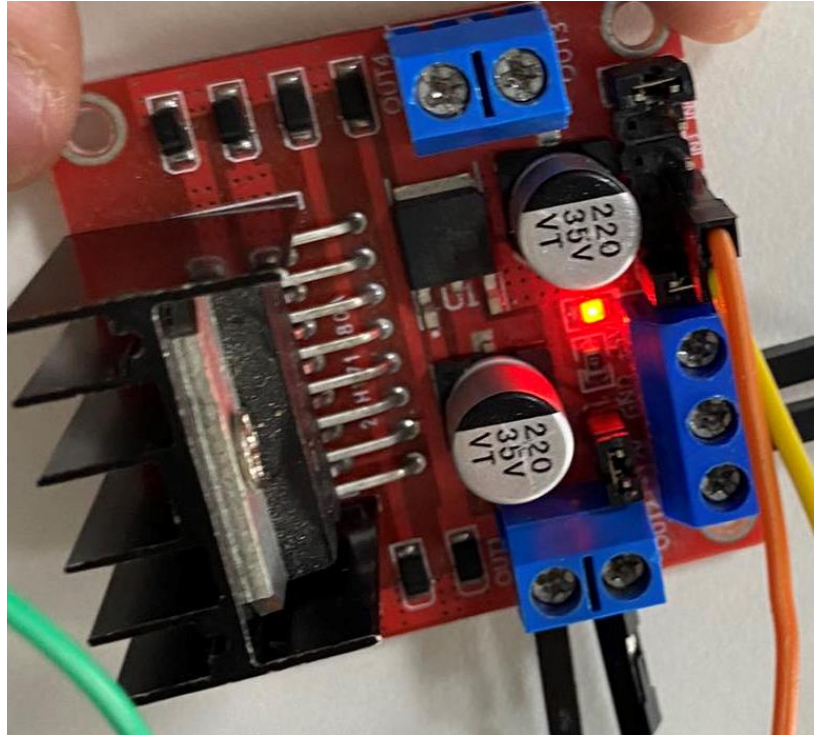


Figure 8 - L298N

The L298N motor driver module and motor constitute crucial components of the environmental monitoring and control system, enabling physical control actions in response to detected environmental conditions. The L298N motor driver module facilitates the control of the motor's speed and direction, allowing for precise adjustments based on input signals received from the Arduino microcontroller. Connected to the Arduino, the L298N motor driver module interprets commands related to motor operation and translates them into appropriate voltage outputs to drive the motor accordingly.



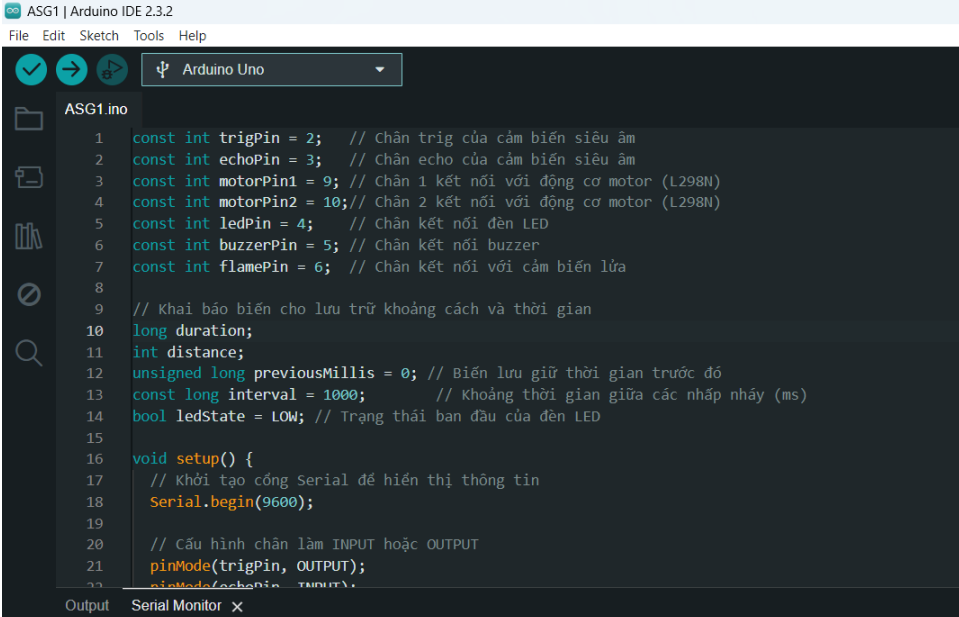
Figure 9 – Motor

The motor, controlled by the L298N motor driver module, responds to commands from the Arduino, which monitors sensor data. When the HC-SR04 ultrasonic sensor detects a distance less than 10 cm, signaling a critical event, the Arduino sends commands to the L298N module to stop the motor. This action helps mitigate potential hazards in the environment. The coordinated interaction between the Arduino, L298N module, and motor allows for dynamic adjustments based on real-time sensor inputs, enhancing system responsiveness.

c. Software/Libraries:

- Arduino Programming with Arduino IDE:

The Arduino IDE is utilized for programming the Arduino Uno microcontroller board. Arduino sketches are written in C/C++ to define the behavior of the IoT system, including sensor data acquisition, actuator control, and serial communication protocols.



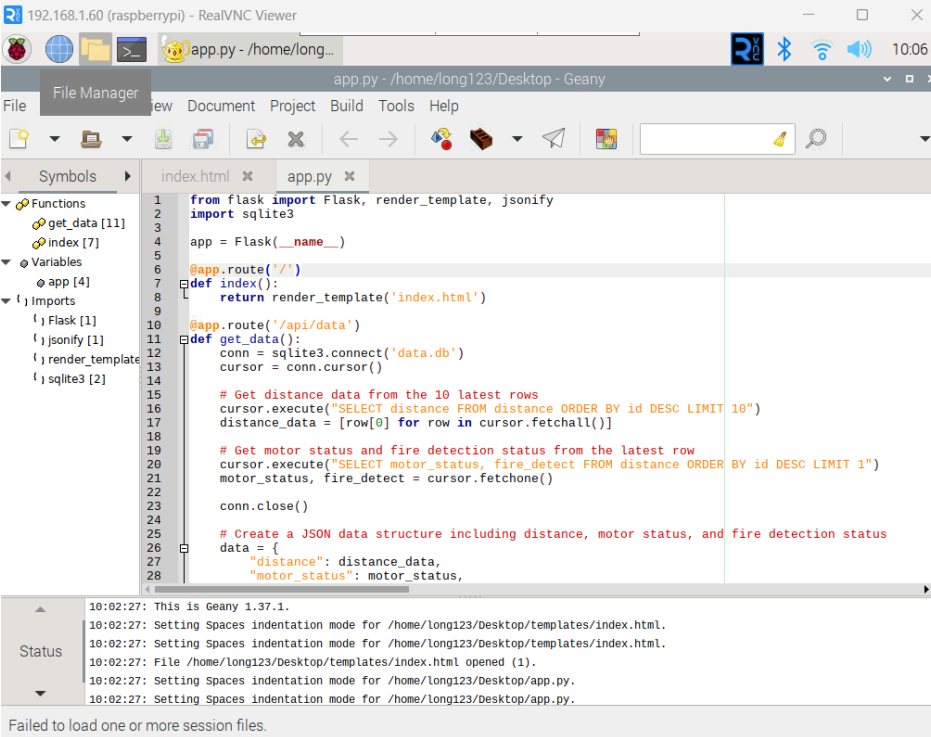
```
ASG1 | Arduino IDE 2.3.2
File Edit Sketch Tools Help

ASG1.ino
1 const int trigPin = 2; // Chân trig của cảm biến siêu âm
2 const int echoPin = 3; // Chân echo của cảm biến siêu âm
3 const int motorPin1 = 9; // Chân 1 kết nối với động cơ motor (L298N)
4 const int motorPin2 = 10; // Chân 2 kết nối với động cơ motor (L298N)
5 const int ledPin = 4; // Chân kết nối đèn LED
6 const int buzzerPin = 5; // Chân kết nối buzzer
7 const int flamePin = 6; // Chân kết nối với cảm biến lửa
8
9 // Khai báo biến cho lưu trữ khoảng cách và thời gian
10 long duration;
11 int distance;
12 unsigned long previousMillis = 0; // Biến lưu giữ thời gian trước đó
13 const long interval = 1000; // Khoảng thời gian giữa các nhấp nháy (ms)
14 bool ledState = LOW; // Trạng thái ban đầu của đèn LED
15
16 void setup() {
17 // Khởi tạo cổng Serial để hiển thị thông tin
18 Serial.begin(9600);
19
20 // Cấu hình chân làm INPUT hoặc OUTPUT
21 pinMode(trigPin, OUTPUT);
22 pinMode(echoPin, INPUT);
23 pinMode(motorPin1, OUTPUT);
24 pinMode(motorPin2, OUTPUT);
25 pinMode(ledPin, OUTPUT);
26 pinMode(buzzerPin, OUTPUT);
27 pinMode(flamePin, INPUT);
28
29 }
30
31 void loop() {
32 // Kiểm tra cảm biến siêu âm
33 digitalWrite(trigPin, LOW);
34 delayMicroseconds(2);
35 digitalWrite(trigPin, HIGH);
36 delayMicroseconds(10);
37 digitalWrite(trigPin, LOW);
38 duration = pulseIn(echoPin, HIGH);
39 distance = duration * 0.034 / 2;
40 Serial.print("Distance: ");
41 Serial.println(distance);
42
43 // Kiểm tra cảm biến lửa
44 if (digitalRead(flamePin) == HIGH) {
45 digitalWrite(buzzerPin, HIGH);
46 digitalWrite(ledPin, HIGH);
47 } else {
48 digitalWrite(buzzerPin, LOW);
49 digitalWrite(ledPin, LOW);
50 }
51
52 // Kiểm tra cảm biến siêu âm và điều khiển động cơ
53 if (distance < 10) {
54 digitalWrite(motorPin1, HIGH);
55 digitalWrite(motorPin2, LOW);
56 } else if (distance > 100) {
57 digitalWrite(motorPin1, LOW);
58 digitalWrite(motorPin2, HIGH);
59 } else {
60 digitalWrite(motorPin1, LOW);
61 digitalWrite(motorPin2, LOW);
62 }
63
64 // Kiểm tra thời gian và nhấp nháy đèn LED
65 if (previousMillis + interval < millis()) {
66 previousMillis = millis();
67 ledState = !ledState;
68 digitalWrite(ledPin, ledState);
69 }
70 }
71
72 Output Serial Monitor x
```

Figure 10 - Arduino IDE

- Raspberry Pi Programming with Python and Geany:

Python scripts are developed using the Geany text editor on the Raspberry Pi. These scripts leverage the Flask framework to create a web application, allowing for the development of a user interface for data visualization and system control. Additionally, Python is used to integrate SQLite3 for database management and PySerial for serial communication with the Arduino.



```

1  from flask import Flask, render_template, jsonify
2  import sqlite3
3
4  app = Flask(__name__)
5
6  @app.route('/')
7  def index():
8      return render_template('index.html')
9
10 @app.route('/api/data')
11 def get_data():
12     conn = sqlite3.connect('data.db')
13     cursor = conn.cursor()
14
15     # Get distance data from the 10 latest rows
16     cursor.execute("SELECT distance FROM distance ORDER BY id DESC LIMIT 10")
17     distance_data = [row[0] for row in cursor.fetchall()]
18
19     # Get motor status and fire detection status from the latest row
20     cursor.execute("SELECT motor_status, fire_detect FROM distance ORDER BY id DESC LIMIT 1")
21     motor_status, fire_detect = cursor.fetchone()
22
23     conn.close()
24
25     # Create a JSON data structure including distance, motor status, and fire detection status
26     data = {
27         "distance": distance_data,
28         "motor_status": motor_status,

```

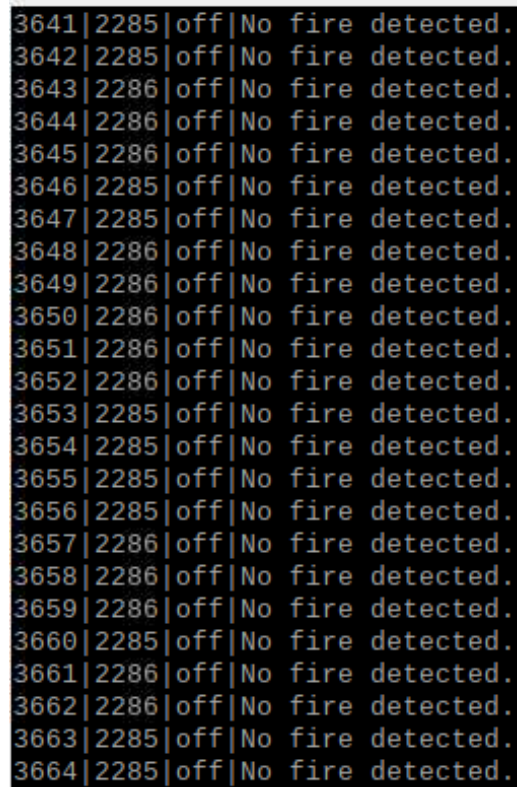
10:02:27: This is Geany 1.37.1.
10:02:27: Setting Spaces indentation mode for /home/long123/Desktop/templates/index.html.
10:02:27: Setting Spaces indentation mode for /home/long123/Desktop/templates/index.html.
10:02:27: File /home/long123/Desktop/templates/index.html opened (1).
10:02:27: Setting Spaces indentation mode for /home/long123/Desktop/app.py.
10:02:27: Setting Spaces indentation mode for /home/long123/Desktop/app.py.

Failed to load one or more session files.

Figure 11 – Geany

- Database Management with SQLite3:

SQLite3 is employed for database management within the Raspberry Pi environment. Python scripts interact with the SQLite3 database to store and retrieve sensor data efficiently, facilitating data analysis and system functionality.



```
3641|2285|off|No fire detected.
3642|2285|off|No fire detected.
3643|2286|off|No fire detected.
3644|2286|off|No fire detected.
3645|2286|off|No fire detected.
3646|2285|off|No fire detected.
3647|2285|off|No fire detected.
3648|2286|off|No fire detected.
3649|2286|off|No fire detected.
3650|2286|off|No fire detected.
3651|2286|off|No fire detected.
3652|2286|off|No fire detected.
3653|2285|off|No fire detected.
3654|2285|off|No fire detected.
3655|2285|off|No fire detected.
3656|2285|off|No fire detected.
3657|2286|off|No fire detected.
3658|2286|off|No fire detected.
3659|2286|off|No fire detected.
3660|2285|off|No fire detected.
3661|2286|off|No fire detected.
3662|2286|off|No fire detected.
3663|2285|off|No fire detected.
3664|2285|off|No fire detected.
```

Figure 12 - SQLite3 Database

- Serial Communication with PySerial:

PySerial library enables serial communication between the Arduino and Raspberry Pi. Bidirectional data exchange is established, allowing the Arduino to transmit sensor data to the Raspberry Pi, while the Raspberry Pi can send control commands back to the Arduino. This communication protocol ensures seamless interaction between the two components of the IoT system.

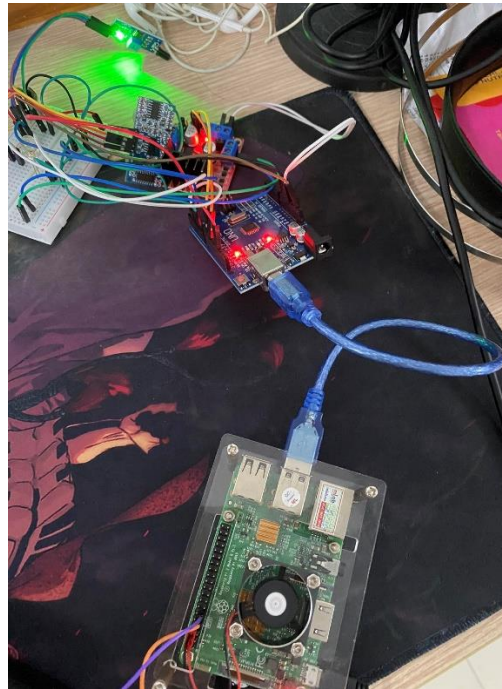


Figure 13 - Arduino and IOT node connect to raspberry pi 3

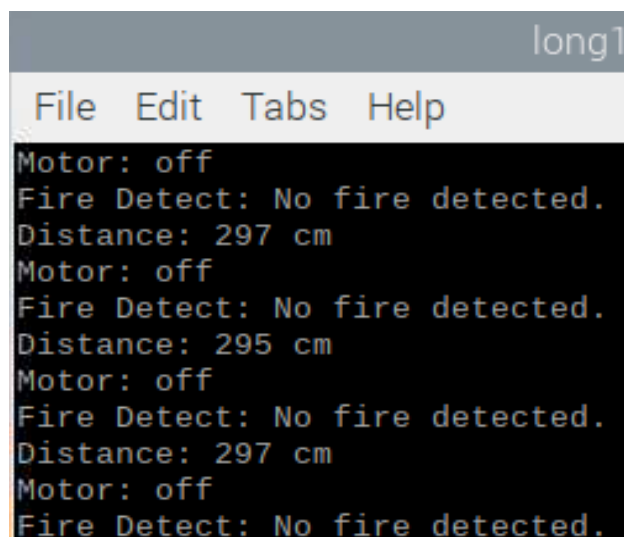
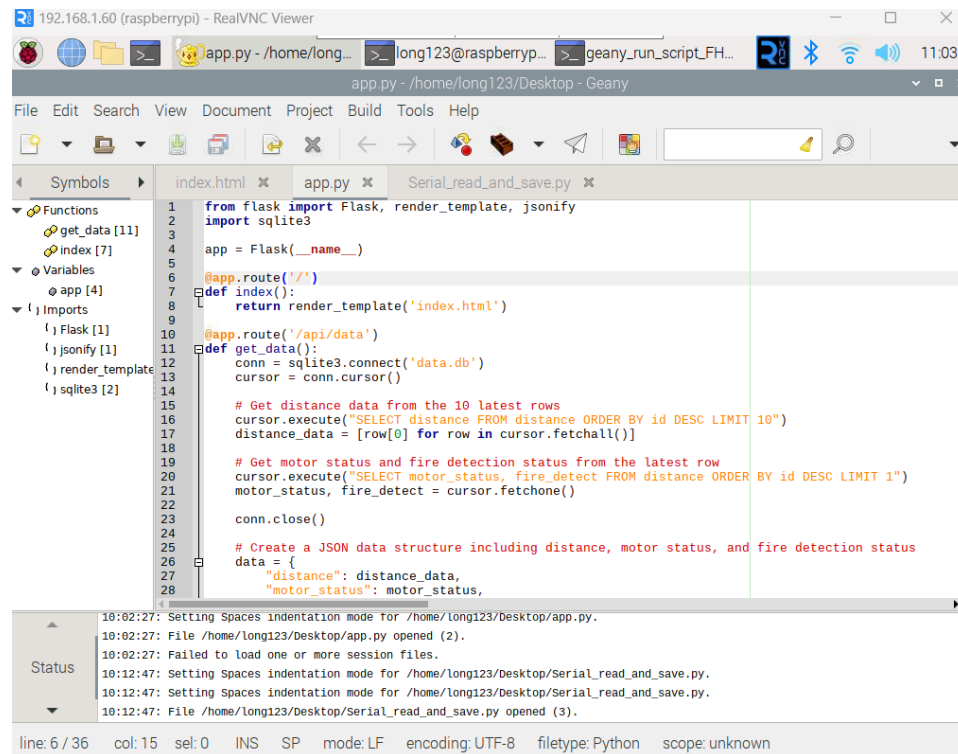


Figure 14 - Data receive on raspberry pi 3

- Web Development with Flask:

The Flask library is utilized to develop a web server on the Raspberry Pi, facilitating the creation of a user interface. Through Flask, data fetched from the SQLite3 database can be routed and displayed on web pages, providing users with real-time access to sensor data and system controls.



```

1 from flask import Flask, render_template, jsonify
2 import sqlite3
3
4 app = Flask(__name__)
5
6 @app.route('/')
7 def index():
8     return render_template('index.html')
9
10 @app.route('/api/data')
11 def get_data():
12     conn = sqlite3.connect('data.db')
13     cursor = conn.cursor()
14
15     # Get distance data from the 10 latest rows
16     cursor.execute("SELECT distance FROM distance ORDER BY id DESC LIMIT 10")
17     distance_data = [row[0] for row in cursor.fetchall()]
18
19     # Get motor status and fire detection status from the latest row
20     cursor.execute("SELECT motor_status, fire_detect FROM distance ORDER BY id DESC LIMIT 1")
21     motor_status, fire_detect = cursor.fetchone()
22
23     conn.close()
24
25     # Create a JSON data structure including distance, motor status, and fire detection status
26     data = {
27         "distance": distance_data,
28         "motor_status": motor_status,

```

Status

10:02:27: Setting Spaces indentation mode for /home/long123/Desktop/app.py.
 10:02:27: File /home/long123/Desktop/app.py opened (2).
 10:02:27: Failed to load one or more session files.
 10:12:47: Setting Spaces indentation mode for /home/long123/Desktop/Serial_read_and_save.py.
 10:12:47: Setting Spaces indentation mode for /home/long123/Desktop/Serial_read_and_save.py.
 10:12:47: File /home/long123/Desktop/Serial_read_and_save.py opened (3).

line: 6 / 36 col: 15 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: unknown

Figure 15 - Flask library

The user interface is a pivotal element of the IoT system, offering real-time access to sensor data and system status. It features a dynamic chart displaying live distance measurements from the HC-SR04 ultrasonic sensor, enabling users to assess environmental conditions at a glance. Additionally, a dedicated section displays motor and fire detection statuses, providing clear indicators of system activity and potential hazards. Interactive controls may be incorporated to empower users with the ability to adjust settings and initiate control actions, ensuring effective monitoring and proactive response to emerging situations.

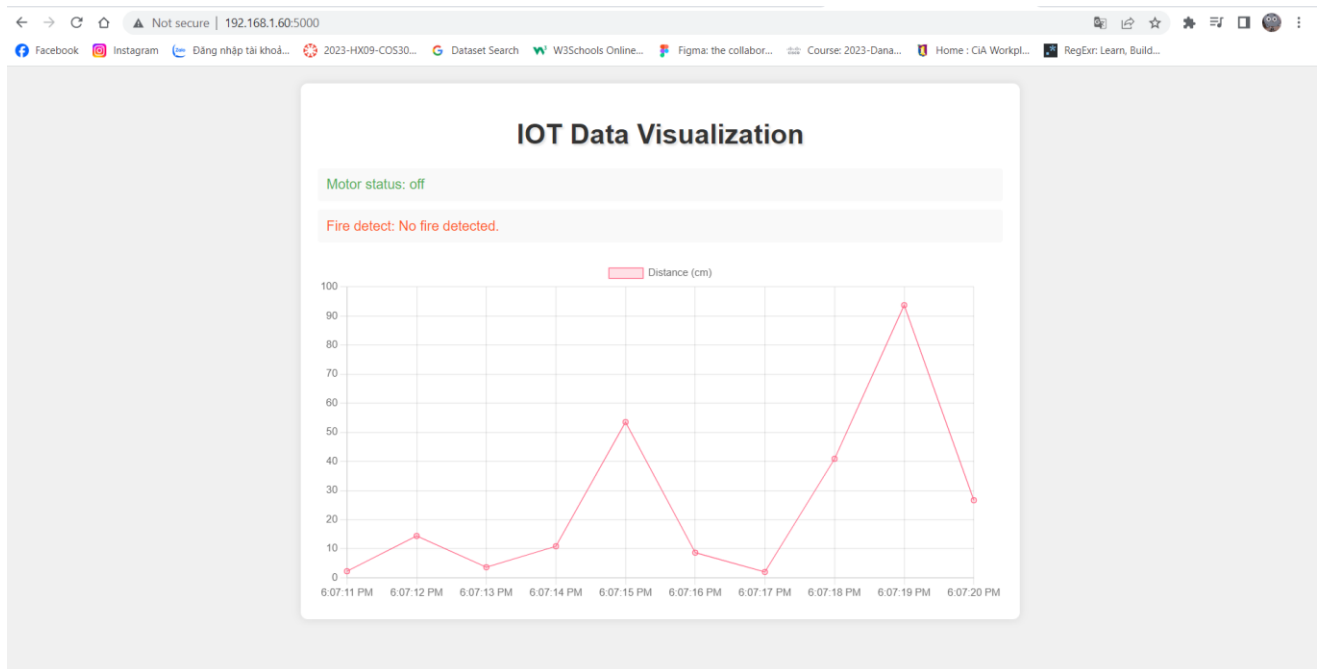


Figure 16 - User interface on web page

IV. Resources:

- *How to install Raspberry Pi OS 'RASPBIAN' operating system - pi4 pi3 pi2* (2021) *YouTube*. Available at: <https://www.youtube.com/watch?v=h6GOB07FjG0> (Accessed: 03 March 2024).
- Söderby, K. (no date) *Using the serial Plotter Tool*, *docs.arduino.cc*. Available at: <https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-plotter/> (Accessed: 03 March 2024).
- **RealVNC**
Remote Access Software for desktop and Mobile: Realvnc (2024) *RealVNC®*. Available at: <https://www.realvnc.com/en/> (Accessed: 03 March 2024).
- **Arduino IDE**
Team, T.A. (no date) *Software*, Arduino. Available at: <https://www.arduino.cc/en/software> (Accessed: 03 March 2024).

V. Appendix:

a. Sketches

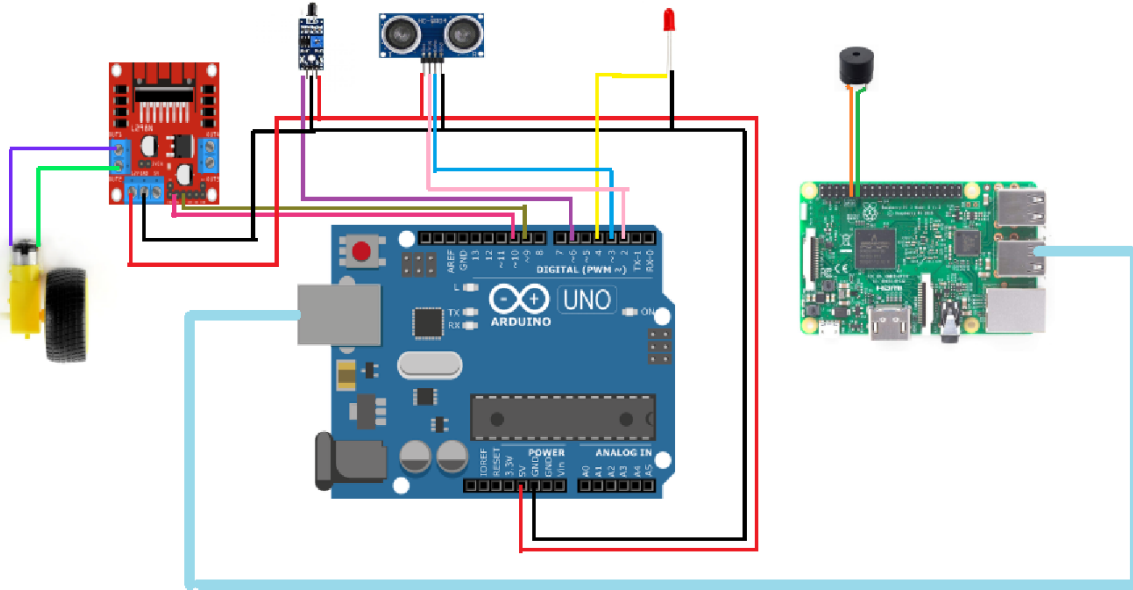


Figure 17 - IOT Node Sketches

b. Scripts

All my script is uploaded to this drive:

<https://drive.google.com/drive/u/0/folders/1x0u4qllex74hq06mrMND7y7tcCb2Fz4WS>

- ASG1.ino file include code for Arduino to run the IOT Node.
- Serial_read_and_save.py file code run on raspberry pi to get the data from Arduino and save to database.
- App.py file code use flask library to fetch the data from the database and route it real-time when the database update to the index.html file in templates folder.