HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

XÂY DỰNG CÁC HỆ THỐNG NHÚNG

ĐỀ TÀI : HỆ THỐNG ĐIỀU KHIỂN Ô TÔ BẰNG GIỌNG NÓI, TỰ ĐỘNG BẬT TẮT ĐÈN KHI TRỜI TỐI,TỰ ĐỘNG CẢM ỨNG VẬT CẢN

Giảng viên hướng dẫn: Đỗ Tiến Dũng

Nhóm học phần: Nhóm 02

Nhóm bài tập: Nhóm 04

Thành viên: Lê Xuân Tấn – B20DCCN591

Phạm Văn Lực – B20DCCN411

Đỗ Việt Phương – B20DCCN519

Nguyễn Long Vũ – B20DCCN746

Hà Nội – 2024

MỤC LỤC

I. Tổng quan về hệ thống

1.1. Lý do chọn đề tài:

Tiện lợi và an toàn: Việc điều khiển ô tô bằng giọng nói có thể giúp giảm sự phụ thuộc vào các gói tay lái và các công tắc điều khiển truyền thống, tạo ra một phương tiện đi lại tiện lợi hơn và giảm nguy cơ tai nạn do sự phân tâm khi lái xe.

Tiềm năng ứng dụng rộng rãi: Công nghệ điều khiển bằng giọng nói có thể được áp dụng không chỉ trong xe hơi cá nhân mà còn trong xe hơi tự lái và các ứng dụng thương mại khác như xe buýt hoặc xe tải.

Tiến bộ trong trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên: Công nghệ hiện đại đã đạt được tiến bộ đáng kể trong việc nhận dạng và hiểu ngôn ngữ tự nhiên, mở ra cơ hội để ứng dụng trong việc điều khiển ô tô.

Sự phát triển của ô tô tự lái và xe kết nối: Đề tài này liên quan mật thiết đến sự phát triển của ô tô tự lái và các hệ thống xe kết nối, nơi mà việc giao tiếp giữa người dùng và xe sẽ trở nên ngày càng quan trọng.

Thách thức kỹ thuật và nghiên cứu: Việc phát triển một hệ thống điều khiển ô tô bằng giọng nói đòi hỏi sự kết hợp của nhiều lĩnh vực như xử lý tín hiệu, trí tuệ nhân tạo, và kỹ thuật nhúng, tạo ra một thách thức hấp dẫn cho các nhà nghiên cứu và kỹ sư.

1.2. Mục đích hệ thống

Hệ thống điều khiển ô tô bằng giọng nói là một công nghệ tiên tiến được tích hợp trong các phương tiện giao thông để cho phép người lái điều khiển các chức năng của xe thông qua lệnh giọng nói. Hệ thống này sử dụng công nghệ nhận dạng giọng nói và xử lý ngôn ngữ tự nhiên để hiểu và thực hiện các yêu cầu từ người lái một cách tự nhiên và hiệu quả.

Hệ thống được tạo ra một cách tiện lợi để người lái có thể tương tác với các chức năng trong xe mà không cần phải sử dụng tay, giúp tập trung hơn vào việc lái xe và giảm nguy cơ xảy ra tai nạn do sự phân tâm.

- Hệ thống này cung cấp khả năng tương tác tự nhiên thông qua giọng nói, cho phép người lái yêu cầu và điều chỉnh các chức năng trong xe một cách linh hoạt và nhanh chóng.
- Hệ thống còn giúp xe tự động bật đèn khi trời tối hoặc tầm nhìn bị giảm sút giúp người lái có thể lái xe an toàn hơn.
- Hệ thống điều khiển ô tô tự động có khả năng phát hiện và tránh các vật cản trong quá trình di chuyển, giúp giảm nguy cơ tai nạn giao thông do va chạm.

1.3. Hành vi hệ thống

Mô tả chi tiết về hành vi của hệ thống:

Điều khiển ô tô bằng giọng nói:

- Hệ thống điều khiển ô tô bằng giọng nói sử dụng một bộ khung xe Robot 4 bánh, mạch ESP32 30 chân, mô-đun điều khiển động cơ L298N (board màu đỏ) và sử dụng trợ lý ảo Google Assistant để nhận diện và xử lý lệnh giọng nói.
- Mạch ESP32 kết nối với mạng WiFi để giao tiếp với trợ lý ảo Google Assistant và giao diện người dùng là một trang web, thực hiện điều khiển toàn bộ hệ thống như việc di chuyển xe theo các hướng.
- Giao diện người dùng cho phép nhập lệnh giọng nói, Google Assistant sẽ sử dụng công nghệ nhận dạng giọng nói để nhận dạng lệnh của người dùng (VD: Tiến lên, đi lùi, …) rồi gửi đến server (dữ liệu văn bản). Sau đó, server sẽ nhận diện yêu cầu của người dùng dựa trên model học máy đã được train từ trước. Sau khi nhận dạng xong, server sẽ gửi đúng yêu cầu của người dùng đến ESP32. ESP32 nhận được yêu cầu và gửi tín hiệu điều khiển tới L298N điều khiển động cơ ô tô thông qua việc đặt các chân IN_1, IN_2, IN_3, IN_4 ở 2 trạng thái là cao (HIGH) và thấp (LOW).
- Động cơ L298N nhận được tín hiệu điều khiển thực hiện di chuyển xe theo khoảng thời gian (delay) đã được cài đặt trong code. kết thúc thời gian di chuyển thì gọi đến hàm stopRobot() trong code để dừng xe.

Tự động bật đèn khi trời tối hoặc tầm nhìn giảm:

- Hệ thống dùng ESP32 kết nối vào 1 quang cảm đề đọc ra được giá trị từ cảm biến ánh sáng.
- Kết nối ESP32 cho bóng đèn của ô tô, sau khi giá trị được gửi về ESP32 sẽ tiếp nhận thông tin cường độ ánh sáng và so sánh xem có sự biến đổi bất thường nào không để tự động tắt bật đèn cho phù hợp với hệ thống lái xe.

Tự động cảm ứng vật cản:

- Hệ thống dùng dữ liệu khoảng cách vật cản gần nhất với ô tô được lấy từ môi trường lưu trữ ThingSpeak (dữ liệu được thu thập từ cảm biến siêu âm) để hiển thị trực tiếp đến người dùng.
- Dựa trên dữ liệu Time Series được lấy từ Thingspeak , hệ thống có thể phân tích được là có vật thể đang lại gần xe ô tô hay không -> Từ đó đưa ra cảnh báo đến người dùng cuối.
- Hệ thống hiển thị trạng thái di chuyển hiện tại của xe và có đang gặp vật cản hay không.

1.4. Yêu cầu hệ thống

Khả năng nhận diện được yêu cầu của người dùng thông qua giọng nói: Ứng dụng sử dụng mô hình google microphone để ghi lại giọng nói và sử dụng mô hình *học máy* để nhận diện được yêu cầu từ người dùng.

Yêu cầu phân tích dữ liệu: Dữ liệu từ các cảm biến được thiết kế trên ô tô được lưu trữ trên nền tảng đám mây và được hệ thống cập nhật và phân tích liên tục để hiển thị cho người dùng.

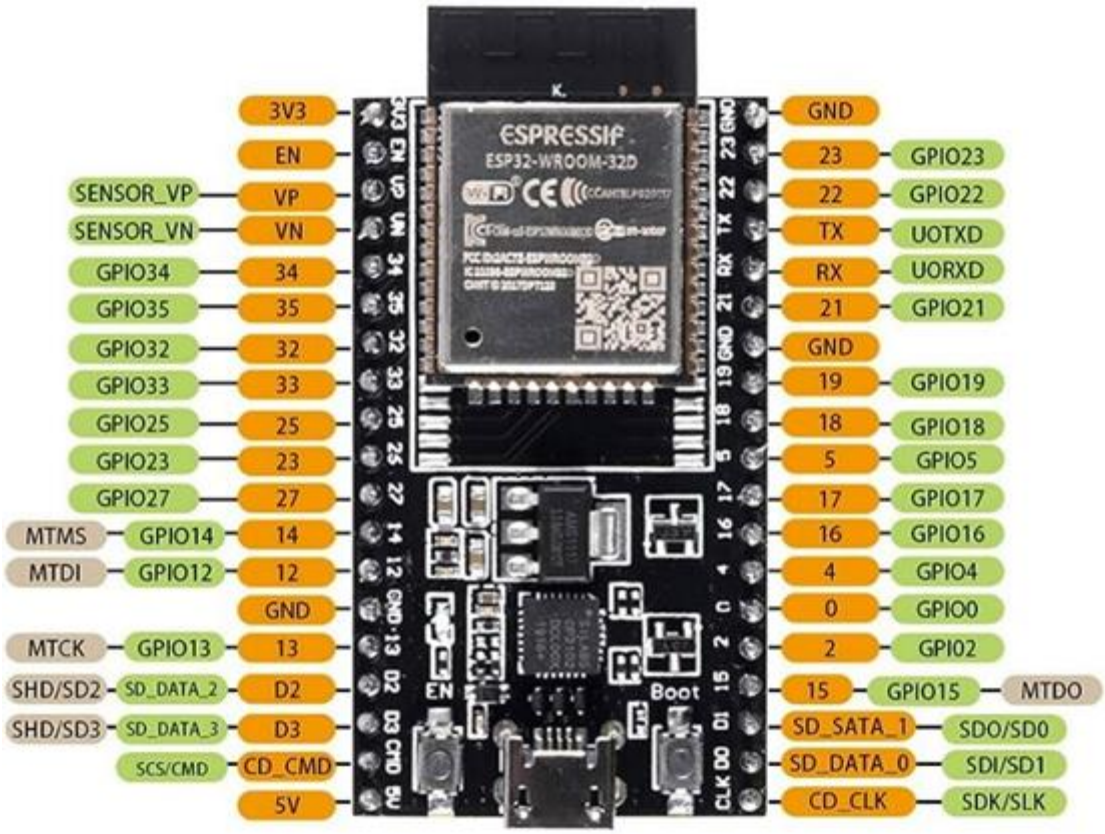
Yêu cầu triển khai ứng dụng: Triển khai trên nền tảng web.

1.5. Công cụ và các thiết bị sử dụng

1.5.1 Phần cứng:

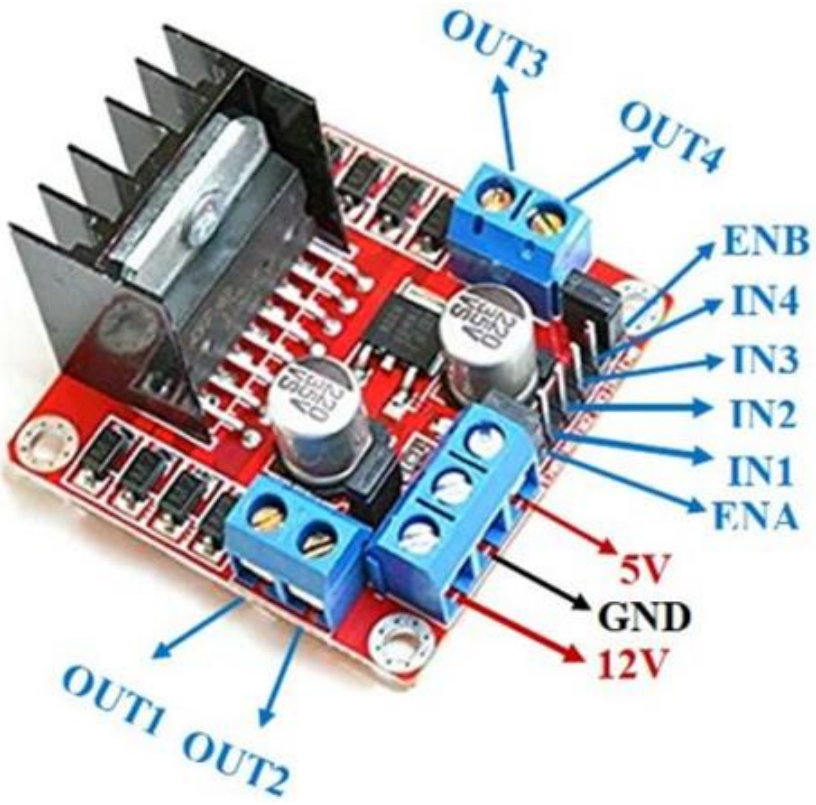
- 1 xe ô tô giả lập (xe mô hình) tự chế gồm 4 động cơ hộp giảm tốc, điện áp cung cấp cho động cơ 6-9V.

- 1 mạch ESP32-WROOM-32D



ESP32-WROOM-32D

- 1 module điều khiển động cơ 12V L298N.



- 1 bộ quang cảm LDR (cảm biến ánh sáng)

- 1 Cảm Biến Siêu Âm HC-SR04:

- 4 cục pin Maxell: mỗi cục 1,5V

- Laptop có kết nối Internet

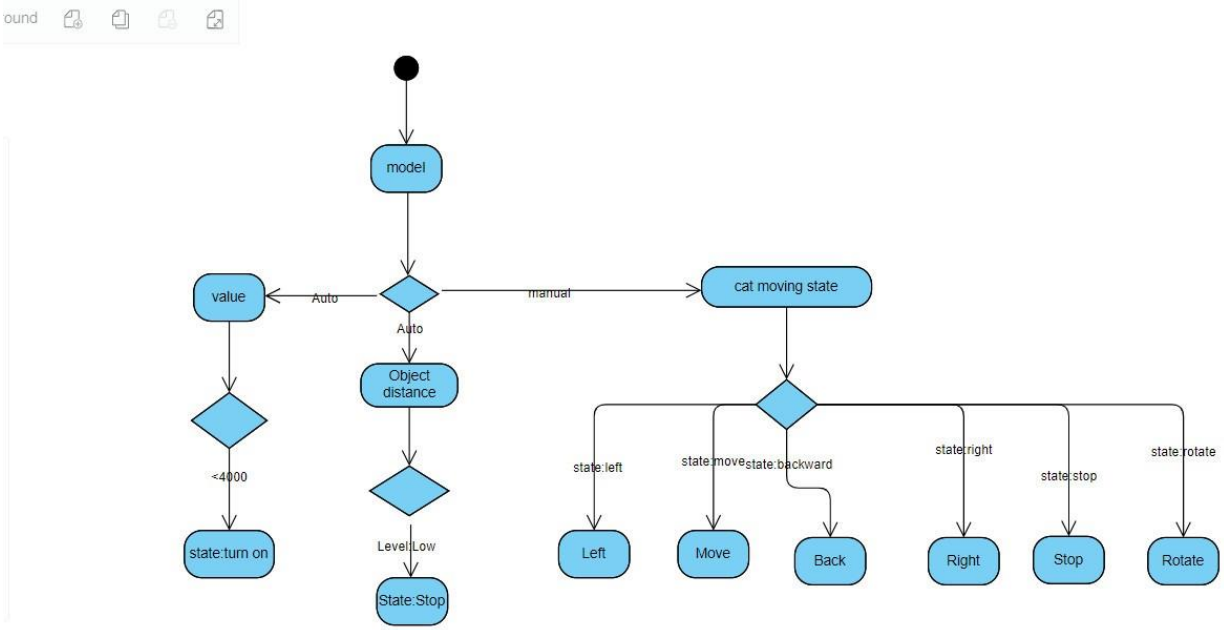
- Dây nối, đèn LED, công tắc


1.5.2 Phần mềm:

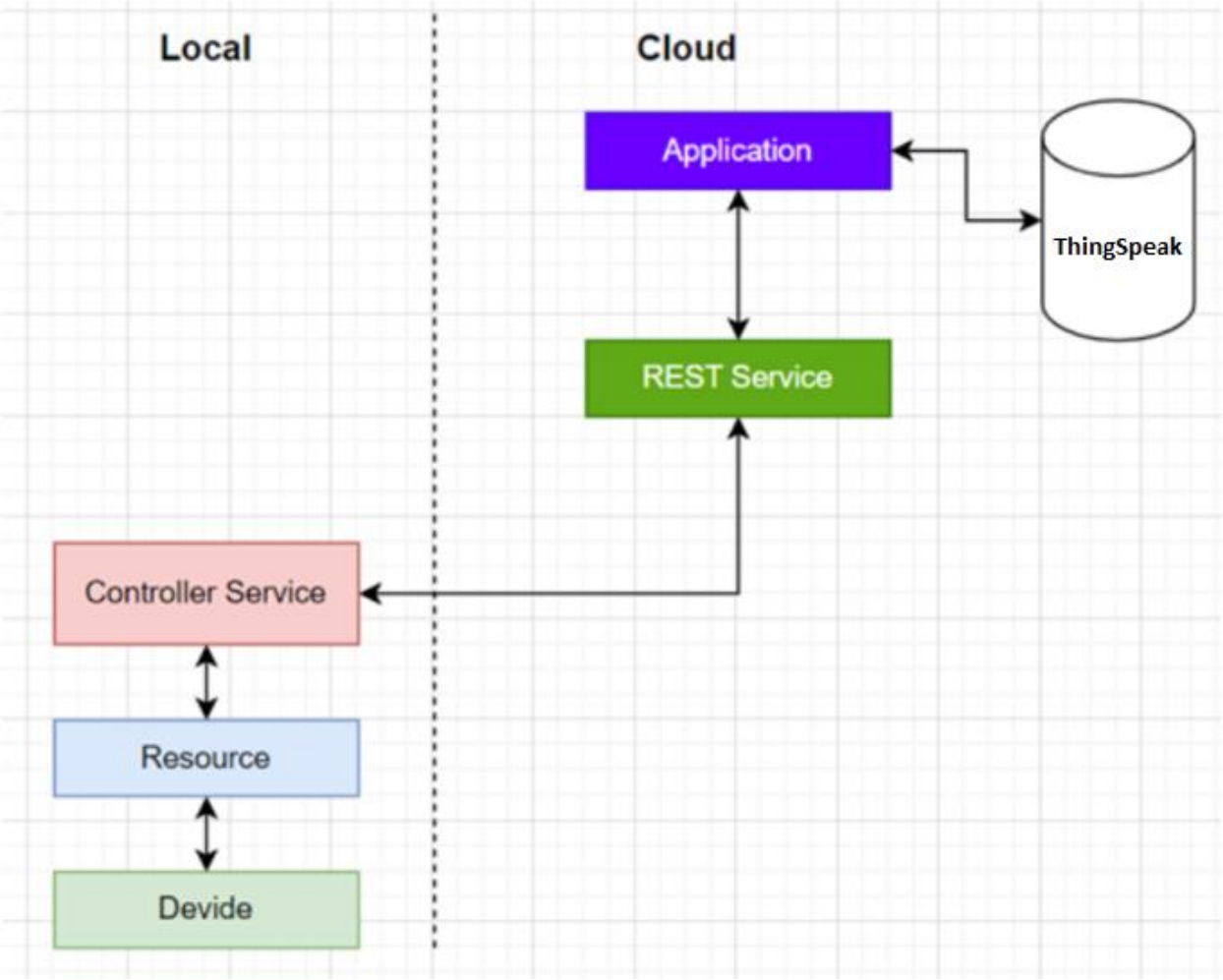
- Giao diện người dùng: HTML, CSS, JavaScript
- Server: Flask Python
- Giao thức: MQTT
- Lưu trữ: ThingSpeak
- Arduino IDE

II. Thiết kế hệ thống

2.1. Đặc tả tiến trình



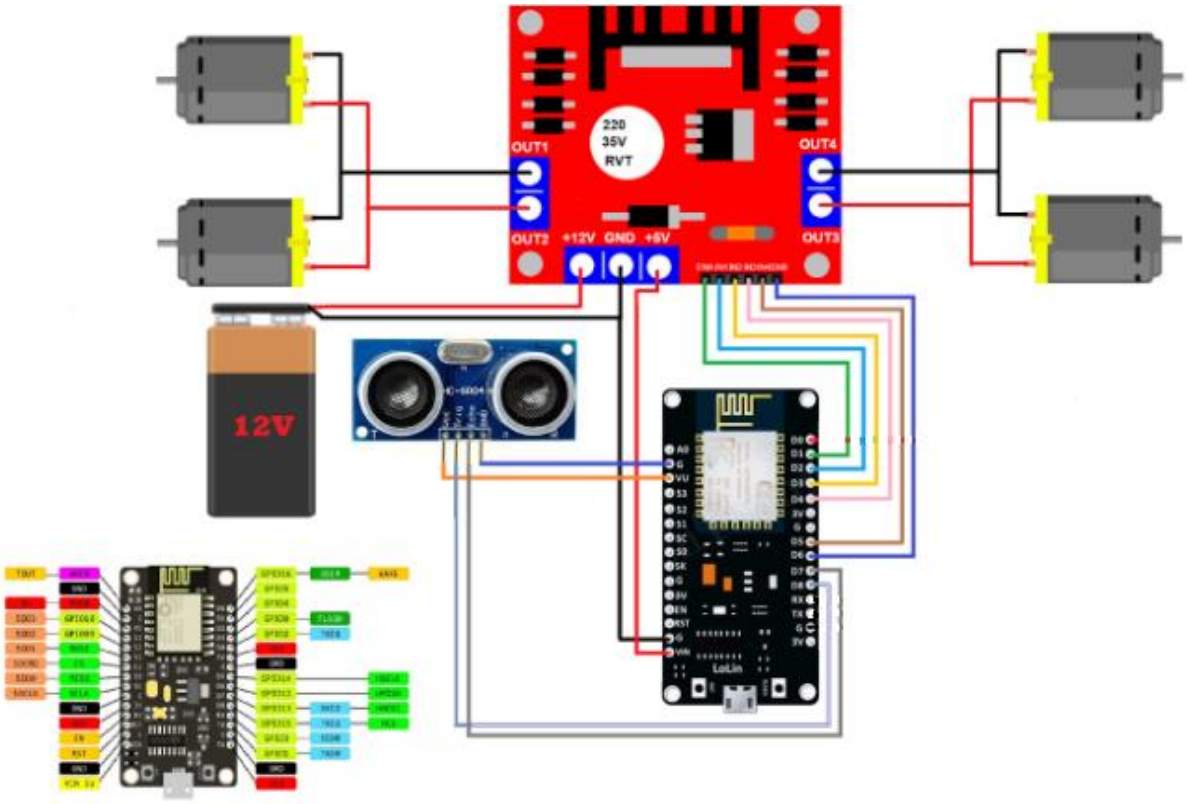
2.2. Đặc tả thành phần chức năng



Luồng xử lý:

- Người dùng gửi yêu cầu trên dịch vụ webservice
- Yêu cầu của người dùng được gửi đến server (Flask)
- Server thực hiện yêu cầu của người dùng: Xử lý dữ liệu, xử lý yêu cầu, gửi yêu cầu của người dùng đến ESP32.
- ESP32 thực hiện yêu cầu từ server, điều khiển động cơ ô tô L298N
- Với cảm biến siêu âm, dữ liệu khoảng cách giữa vật cản gần nhất được đẩy lên Cloud (ThingSpeak) liên tục.
- Server lấy dữ liệu từ ThingSpeak và hiển thị lên cho người dùng cuối các thông số giám sát và cảnh báo.

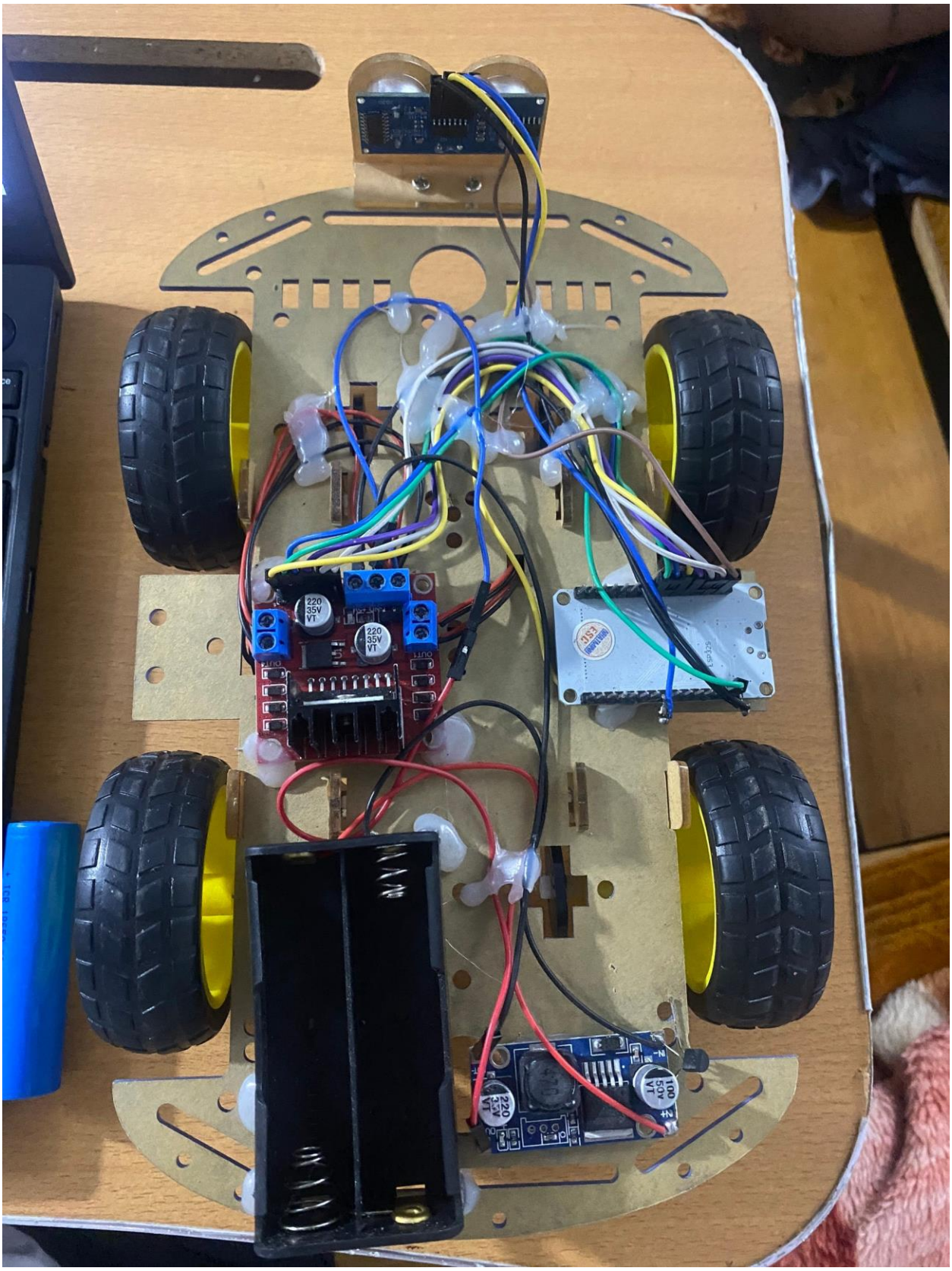
2.3. Sơ đồ tổng thể phần cứng



III. Kết quả

3.1 Phần cứng

Hình minh họa :


220
35V
VT
220
35V
VT
OUT
100
50V
VT
220
35V
VT

Chức năng điều khiển :

- Ở module điều khiển động cơ, nối chân 12V vào công tắc sau đó nối liền vào cực dương của pin, chân GND vào cực âm của pin để cấp điện cho module điều khiển động cơ và vào chân GND của ESP32.
- Nối 4 mô tơ vào 2 chân cấp điện cho mô tơ của module điều khiển động cơ OUT1 – 4 sao cho 2 mô tơ cùng bên được mắc song song với nhau và 2 cặp mô tơ 2 bên nối tiếp nhau. Lắp như vậy để 2 mô tơ cùng bên hoạt động giống nhau giúp xe có thể quay trái, quay phải.
- Nối chân 5V của module điều khiển động cơ vào chân VIN của ESP32, chân GND của module điều khiển động cơ vào chân GND của ESP32 để cấp điện cho ESP32 hoạt động.
- Nối các chân của module điều khiển động cơ ENA, IN1, IN2, IN3, IN4, ENB với các chân tượng ứng của ESP32: D5, D16, D17, D18, D19, D4 để điều khiển các mô tơ và tốc độ của chúng.

Chức năng tự động bật tắt đèn:

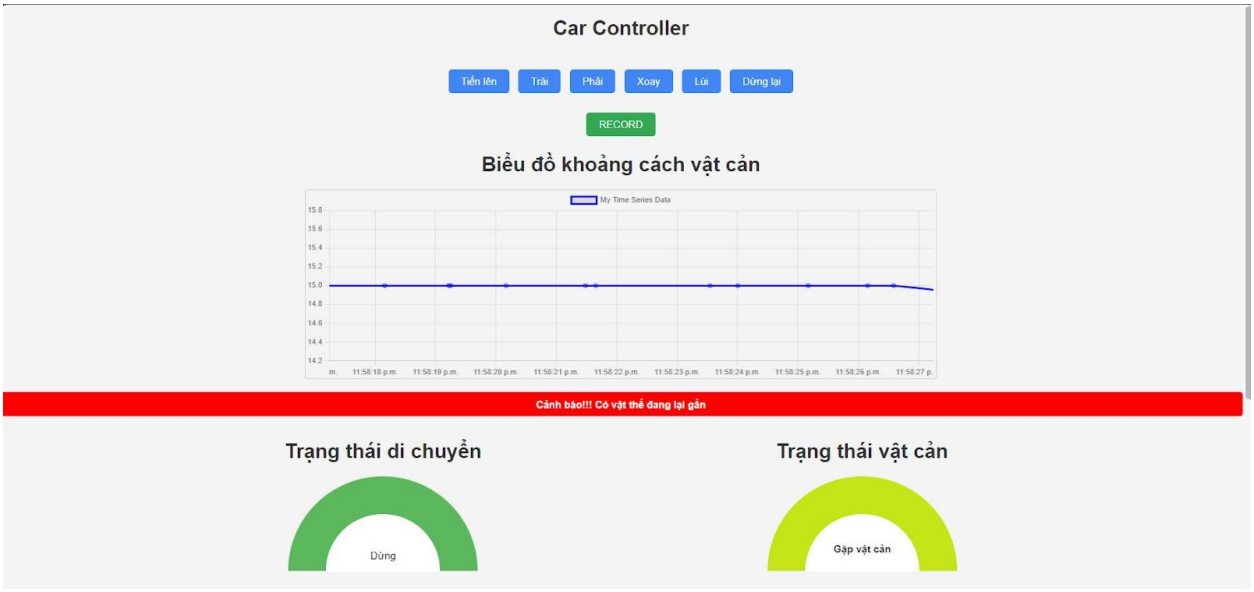
- Kết nối LDR(Quang cảm) :
- Kết nối một chân của cảm biến ánh sáng (LDR) với chân 32 trên ESP32 (hoặc chân ADC mà bạn chọn)
- Kết nối chân còn lại của LDR với chân 3V3 trên ESP32.
- Kết nối LED:
- Sử dụng một đèn LED và một resistor để giới hạn dòng điện qua LED (ví dụ: resistor 220 ohm cho LED thông thường).
- Kết nối chân dương của LED với một chân GPIO (Chân D2 trên ESP32).
- Kết nối chân âm của LED với chân GND trên ESP32.

Chức năng tự động trách vật cản:

- Nối chân VCC, GND, Trig, Echo của cảm biến vật thể bằng sóng siêu âm HC-SR04 tương ứng vào chân VU, GND, D8, D7 của module điều khiển động cơ để điều khiển hoạt động của cảm biến vật thể bằng sóng siêu âm HC-SR04.

3.2 Phần mềm

Giao diện chính của trang Web



Mô tả:

- Hệ thống bao gồm 6 nút: Tiến lên, trái, phải, xoa ,lùi, dừng lại để người dùng có thể trực tiếp điều khiển ô tô thông qua click.
- Button Record cho phép hệ thống thu âm giọng nói từ người dùng và yêu cầu ô tô di chuyển theo yêu cầu từ người dùng.
- Biểu đồ khoảng cách vật cản: Dữ liệu khoảng cách vật cản gần nhất với ô tô được lấy từ môi trường lưu trữ ThingSpeak (dữ liệu được thu thập từ cảm biến siêu âm) để hiển thị trực tiếp đến người dùng.
- Dựa trên dữ liệu Time Series được lấy từ đám mây, hệ thống có thể phân tích được là có vật thể đang lại gần xe ô tô hay không -> Từ đó đưa ra cảnh báo đến người dùng cuối
- Hiển thị trạng thái di chuyển hiện tại của xe và có đang gặp vật cản hay không.

Tạo kết nối MQTT đến Arduino

```
import paho.mqtt.client as mqtt
import time

mqtt_broker = "broker.hivemq.com"  # Use the HiveMQ broker address
mqtt_port = 1883
mqtt_topic = "iot_crvt4722"  # Specify the same MQTT topic used in Arduino code

```

```
client = mqtt.Client()

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe(mqtt_topic)

def on_message(client, userdata, msg):
    command = msg.payload.decode()
    print(f"Received command: {command}")
    control_car(command)

def control_car(command):
    # Add logic to control the car based on the received command
    if command == "forward":
        print("Moving forward")
        # Add your forward motion code here
    elif command == "backward":
        print("Moving backward")
        # Add your backward motion code here
    elif command == "left":
        print("Turning left")
        # Add your left turn code here
    elif command == "right":
        print("Turning right")
        # Add your right turn code here
    elif command == "stop":
        print("Stopping")
        # Add your stop code here
    elif command == "rotate":
        print("rotate")
    else:
        print("Unknown command")

client.on_connect = on_connect
client.on_message = on_message

client.connect(mqtt_broker, mqtt_port, 60)
client.loop_start()
def send_request_to_arduino(command):
    client.publish(mqtt_topic, command)
```

Tạo các API bằng Python Flask kết nối điều khiển đến Arduino thông qua giao thức MQTT

```python
from flask import Flask, url_for, render_template, request, redirect, url_for,
session, flash, jsonify
import joblib
from request_to_arduino_sender import send_request_to_arduino
from flask_cors import CORS
from pulldatatest import get_data_from_think_speak, get_all_data_from_think_speak
import datetime
from urllib.parse import quote

app = Flask(__name__)
CORS(app)

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

@app.route('/think_speak_data', methods=['GET'])
def fetch_think_speak_data():
    start_date = str(datetime.datetime.now(datetime.timezone.utc) -
datetime.timedelta(minutes=5))
    end_date = str(datetime.datetime.now(datetime.timezone.utc))

    start_date = '%20'.join(start_date.split(' '))
    end_date = '%20'.join(end_date.split(' '))

    return get_data_from_think_speak(start_date=start_date, end_date=end_date)

@app.route('/all_think_speak_data', methods=['GET'])
def fetch_all_think_speak_data():
    start_date = str(datetime.datetime.now(datetime.timezone.utc) -
datetime.timedelta(minutes=5))
    end_date = str(datetime.datetime.now(datetime.timezone.utc))

    start_date = '%20'.join(start_date.split(' '))
    end_date = '%20'.join(end_date.split(' '))

    return get_all_data_from_think_speak(start_date=start_date,
end_date=end_date)

@app.route('/controll', methods=['POST'])
def control_car():
    # If the request is JSON, handle it as in your original code
    data = request.get_json()

    try:
```

```
        speech = list()
        speech.append(data['speech'])

        switcher = {
            1: "forward",
            2: "left",
            3: "right",
            4: "backward",
            5: "rotate",
            6: "stop",
        }

        controll_require = switcher.get(speech, "Unknown command")
        print(controll_require)
        send_request_to_arduino(controll_require)

        return jsonify({"controll_require": controll_require})
    except KeyError:
        command = data['command']
        send_request_to_arduino(command=command)

        return jsonify({"command": command})

    except:
        print("Exception")
        return "Invalid request data", 400

    return "Invalid request data", 400

if __name__ == "__main__":
    app.run(debug=True, host = '0.0.0.0')
```

Code Arduino

```
#include "DHT.h"
#include <NewPing.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <string.h>
```

```
#include <ESP32Servo.h>
#include <HTTPClient.h>

// const char* ssid ="crvt4722";
// const char* password ="00000000";

const char* ssid ="xuantan04";
const char* password ="123456789";

// Thinkspeak
String UrlThingspeak =
"http://api.thingspeak.com/update?api_key=J4M93WWZ30RDZO7A";
String httpGETRequest(const char* Url);

//MQTT
#define MQTT_SERVER "broker.hivemq.com"
#define MQTT_PORT 1883
#define MQTT_USER "XuanTan"
#define MQTT_PASSWORD "20042002"
WiFiClient wifiClient;
PubSubClient client(wifiClient);
//END_MQTT
int cnt_to_cloud = 1;

NewPing sonar(21, 5);

const int enA = 13;
const int in1 = 12;
const int in2 = 14;
const int enB = 27;
const int in3 = 26;
const int in4 = 25;
const int ldrPin = 32;  // Chân analog kết nối với LDR
const int ledPin = 2 ;  // Chân GPIO kết nối với LED
int sensorValue = 0;    // Biến lưu trữ giá trị đọc từ LDR
void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(115200);
}
```

```
void setup()
{
  Serial.begin(115200);     // giao tiếp Serial với baudrate 115200

   pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

  pinMode(enB, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);

  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);

  //************************setup wifi
  WiFi.begin(ssid,password);
  Serial.println("conecting");
  while(WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
  //END_wifi

  //MQTT
  client.setServer(MQTT_SERVER, MQTT_PORT );
  client.setCallback(callback);
  connect_to_broker();
}

void callback(char* topic, byte *payload, unsigned int length) {
  Serial.println("-------new message from broker-----");
  Serial.print("topic: ");
```

```
  Serial.println(topic);
  Serial.print("message: ");
  Serial.write(payload, length);
  Serial.println();

  String message = "";

  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }

  if(message == "forward") dithang();
  else if (message == "left") retrai();
  else if (message == "right") rephai();
  else if (message == "stop") stopMotors();
  else if (message == "backward") lui();
  else if (message == "rotate") xoay();
}
void loop() {
  sensorValue = analogRead(ldrPin); // Đọc giá trị từ cảm biến ánh sáng
  Serial.print("Giá trị đọc từ cảm biến ánh sáng: ");
  Serial.println(sensorValue);
  // Kiểm tra giá trị đọc từ cảm biến để bật/tắt đèn LED
  if (sensorValue < 4090) {
    digitalWrite(ledPin, HIGH); // Bật đèn LED
  } else {
    digitalWrite(ledPin, LOW);  // Tắt đèn LED
  }
  delay(1000); // Chờ 1 giây trước khi đọc lại giá trị
}

void loop()
{

  client.loop();
  if (!client.connected()) {
    connect_to_broker();
  }

   int distance = sonar.ping_cm();
```

```
   Serial.print(distance);
   Serial.println(" cm");

  if (distance > 1 && distance < 20) luikhigapvancan();

  int count_object_check = distance < 20? 1:0;
  if(cnt_to_cloud %2 ==0){
    String url = UrlThingspeak + "&field1=" + String(distance) +  "&field2=" +
String(count_object_check);
    HTTPClient http;
    http.begin(url);  // Specify the URL
    int httpCode = http.GET();  // Make the GET request

    // Check for a successful response
    if (httpCode > 0) {
      Serial.printf("[HTTP] GET... code: %d\n", httpCode);
      if (httpCode == HTTP_CODE_OK) {
        String payload = http.getString();
        Serial.println(payload);
      }
    } else {
      Serial.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
    }

    if (cnt_to_cloud > 100000) cnt_to_cloud = 1;
    http.end();
  }

  cnt_to_cloud++;


  delay(700);
}



void connect_to_broker() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
```

```
    String clientId = "ESP32";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str(), MQTT_USER, MQTT_PASSWORD)) {
      Serial.println("connected");
      client.subscribe("iot_crvt4722");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 2 seconds");
      delay(2000);
    }
  }
}


void dithang() {
  // Quay theo chiều kim đồng hồ
  // myServo.write(90);
  analogWrite(enA, 230);  // Giá trị PWM từ 0 đến 255
  analogWrite(enB, 230);  // Giá trị PWM từ 0 đến 255
  Serial.println("Đi thẳng");
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);

  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void retrai() {
  // Rẽ trái
  // myServo.write(135);
  analogWrite(enA, 250);  // Giá trị PWM từ 0 đến 255
  analogWrite(enB, 0);  // Giá trị PWM từ 0 đến 255
  Serial.println("Rẽ trái");
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);

  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  delay(2000);  // Đợi 1 giây, bạn có thể điều chỉnh thời gian để rẽ chính xác 45 độ
```

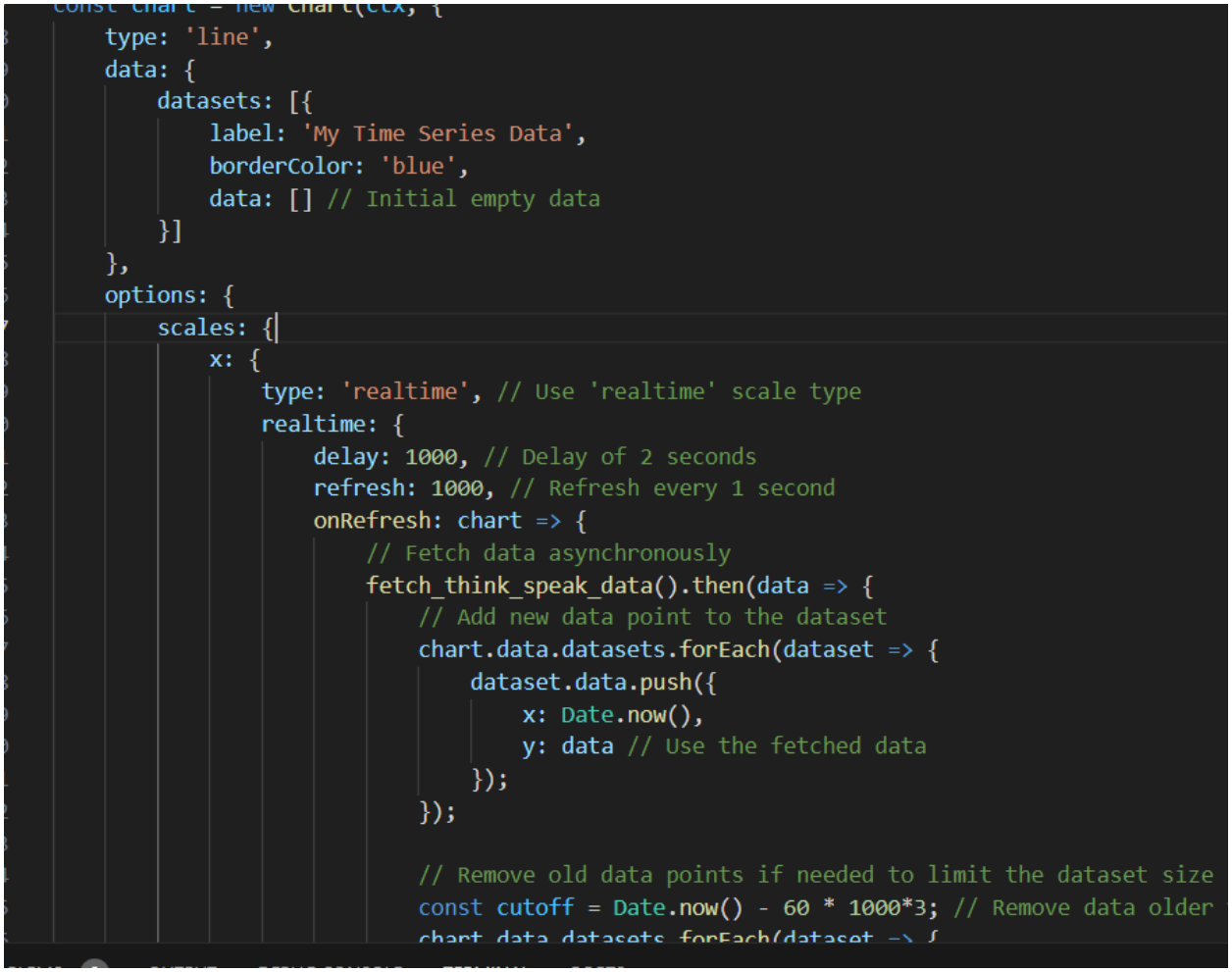
```
  stopMotors();


}
void rephai() {
   // Rẽ phải
  // myServo.write(45);
  analogWrite(enA, 0);  // Giá trị PWM từ 0 đến 255
  analogWrite(enB, 250);  // Giá trị PWM từ 0 đến 255
  Serial.println("Rẽ phải");
  // digitalWrite(in1, HIGH);
  digitalWrite(in1, LOW);
  // digitalWrite(in2, LOW);
  digitalWrite(in2, HIGH);

  // digitalWrite(in3, LOW);
  digitalWrite(in3, HIGH);
  // digitalWrite(in4, HIGH);
  digitalWrite(in4, LOW);
  delay(2000);  // Đợi 1 giây, bạn có thể điều chỉnh thời gian để rẽ chính xác 45 độ
  stopMotors();
}

void luikhigapvancan() {
    // Lùi
  // myServo.write(90);
  analogWrite(enA, 250);  // Giá trị PWM từ 0 đến 255
  analogWrite(enB, 250);  // Giá trị PWM từ 0 đến 255

  Serial.println("Lùi");
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);

  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);

  delay(1000);  // Đợi 1 giây, bạn có thể điều chỉnh thời gian để rẽ chính xác 45 độ
  stopMotors();
}
```

```
void lui() {
    // Lùi
  // myServo.write(90);
  analogWrite(enA, 250);  // Giá trị PWM từ 0 đến 255
  analogWrite(enB, 250);  // Giá trị PWM từ 0 đến 255

  Serial.println("Lùi");
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);

  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
void xoay() {
   // Xoay vòng
  // myServo.write(180);
  analogWrite(enA, 0);  // Giá trị PWM từ 0 đến 255
  analogWrite(enB, 250);  // Giá trị PWM từ 0 đến 255

  Serial.println("Xoay vòng");
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);

  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void stopMotors() {
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);

}
}
```

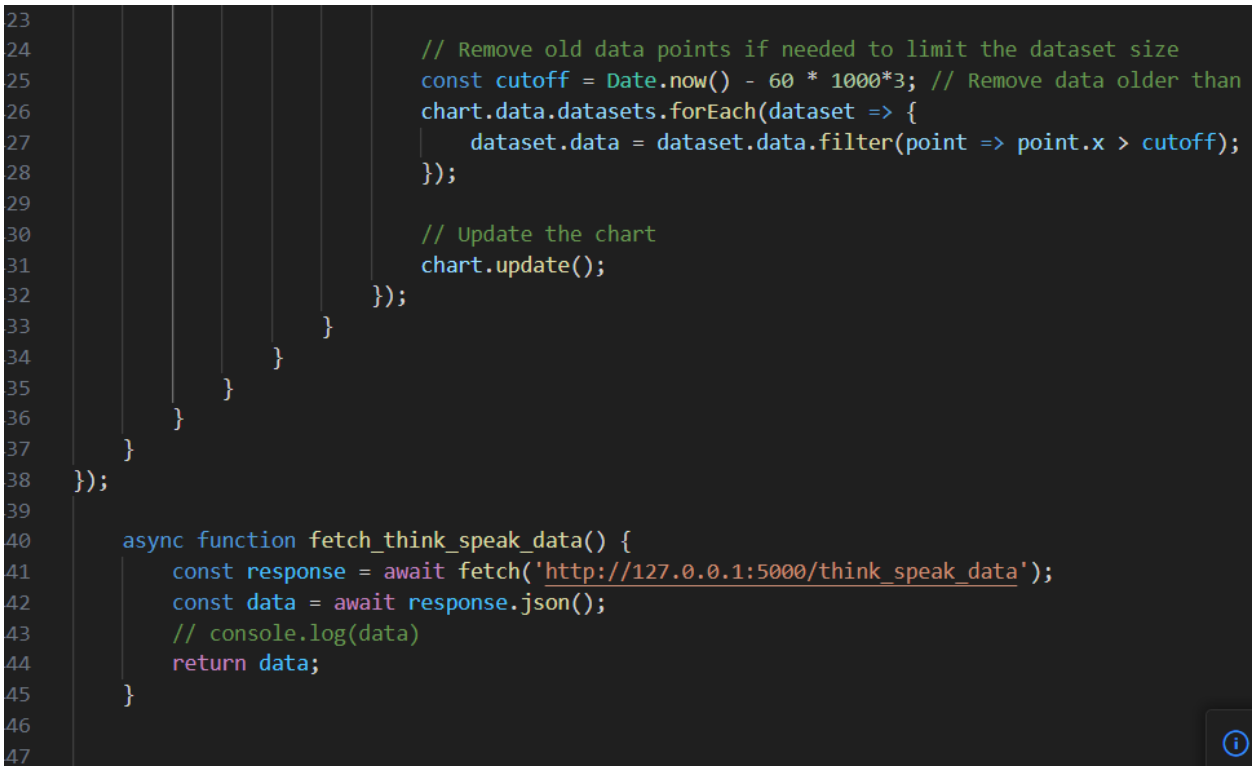
- **Xây dựng hệ thống học máy cho phép phân loại yêu cầu được lấy từ giọng nói của người dùng.**

```python

df = pandas.read_csv('labels.csv')
# print(df)

# Chuẩn bị dữ liệu
text_data = df.label
labels = df.value

# Chia tập dữ liệu
# X_train, X_test, y_train, y_test = train_test_split(text_data, labels, test_size=0.1)

# Vector hóa văn bản
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(text_data)
# X_test_vectorized = vectorizer.transform(X_test)

# Xây dựng và huấn luyện mô hình
model = MultinomialNB()
model.fit(X_train_vectorized, labels)

# Dự đoán và đánh giá mô hình
# y_pred = model.predict(X_test_vectorized)
# accuracy = accuracy_score(y_test, y_pred)
# report = classification_report(y_test, y_pred)

joblib.dump(model,'action_classification.joblib')
# Lưu vectorizer
joblib.dump(vectorizer, 'text_vectorizer.joblib')


```

- **Tạo biểu đồ giám sát realtime lấy dữ liệu từ ThinkSpeak thông qua API server**



```
const chart = new Chart(ctx, {
    type: 'line',
    data: {
        datasets: [{
            label: 'My Time Series Data',
            borderColor: 'blue',
            data: [] // Initial empty data
        }]
    },
    options: {
        scales: {
            x: {
                type: 'realtime', // Use 'realtime' scale type
                realtime: {
                    delay: 1000, // Delay of 2 seconds
                    refresh: 1000, // Refresh every 1 second
                    onRefresh: chart => {
                        // Fetch data asynchronously
                        fetch_think_speak_data().then(data => {
                            // Add new data point to the dataset
                            chart.data.datasets.forEach(dataset => {
                                dataset.data.push({
                                    x: Date.now(),
                                    y: data // Use the fetched data
                                });
                            });

                            // Remove old data points if needed to limit the dataset size
                            const cutoff = Date.now() - 60 * 1000*3; // Remove data older
                            chart.data.datasets.forEach(dataset => {
```



```
                        // Remove old data points if needed to limit the dataset size
                        const cutoff = Date.now() - 60 * 1000*3; // Remove data older than
                        chart.data.datasets.forEach(dataset => {
                            dataset.data = dataset.data.filter(point => point.x > cutoff);
                        });

                        // Update the chart
                        chart.update();
                    });
                }
            }
        }
    }
});

    async function fetch_think_speak_data() {
        const response = await fetch('http://127.0.0.1:5000/think_speak_data');
        const data = await response.json();
        // console.log(data)
        return data;
    }
```

3.3 Kết luận và tài liệu tham khảo

Ưu điểm:

- *Tiện lợi và thoải mái:*

- Người lái có thể điều khiển các chức năng của ô tô mà không cần rời tay khỏi vô lăng hoặc mắt khỏi đường, giúp tăng cường sự an toàn.
- Dễ dàng thực hiện các thao tác như gọi điện thoại, điều chỉnh âm lượng, hoặc thay đổi cài đặt điều hòa chỉ bằng giọng nói.

- *An toàn:*

- Giảm thiểu sự phân tâm của người lái xe, giảm nguy cơ tai nạn do mất tập trung khi thao tác các thiết bị trên xe.
- Cảm ứng vật cản giúp ô tô có thể tự động trách các vật cản giảm thiểu rủi ro tai nạn giao thông

- *Hiệu quả:*

- Tối ưu hóa việc sử dụng thời gian và công sức, cho phép người lái xe thực hiện nhiều nhiệm vụ cùng lúc mà không cần ngừng xe.

- *Khả năng truy cập:*

- Giúp người khuyết tật hoặc những người có hạn chế về thể chất dễ dàng điều khiển các chức năng của ô tô.

- *Công nghệ tiên tiến:*

- Nâng cao giá trị và tính hiện đại của ô tô, đáp ứng nhu cầu ngày càng cao của người tiêu dùng về các tính năng thông minh.

Nhược điểm:

- *Độ chính xác và hiệu quả của nhận diện giọng nói:*

- Hệ thống có thể gặp khó khăn trong việc nhận diện giọng nói khi có nhiều tiếng ồn xung quanh hoặc khi người dùng có giọng nói, cách phát âm, hoặc ngữ điệu khác nhau.
- Một số từ hoặc câu lệnh có thể bị nhận diện sai, gây ra các hành động không mong muốn.

- *Yêu cầu về phần cứng và phần mềm:*

- Cần có các thiết bị phần cứng chuyên dụng như micro chất lượng cao và bộ xử lý mạnh mẽ để đảm bảo hoạt động mượt mà của hệ thống.
- Yêu cầu cập nhật phần mềm thường xuyên để cải thiện độ chính xác và thêm vào các tính năng mới.

- *Bảo mật và quyền riêng tư:*

- Hệ thống điều khiển bằng giọng nói có thể thu thập và lưu trữ dữ liệu âm thanh, tạo ra các lo ngại về bảo mật và quyền riêng tư.
- Nguy cơ bị tấn công hoặc xâm nhập thông qua các lỗ hổng trong hệ thống.

- *Giới hạn ngôn ngữ và ngữ cảnh:*

- Hệ thống có thể không hỗ trợ đầy đủ các ngôn ngữ hoặc phương ngữ khác nhau, gây khó khăn cho người dùng không nói ngôn ngữ được hỗ trợ.
- Khó khăn trong việc hiểu và xử lý các lệnh phức tạp hoặc các ngữ cảnh đặc biệt.

- *Chi phí:*

- Chi phí phát triển và tích hợp hệ thống điều khiển bằng giọng nói có thể khá cao, làm tăng giá thành của ô tô.