HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

-------📖-------

BÀI TẬP LỚN

Đề tài: Mô Hình Ô Tô Đồ Chơi Điều Khiển Từ Xa Tích Hợp Điều Khiển Giọng Nói và Cảm Biến Vật Cản

Môn học : IoT và ứng dụng

Giảng viên: Trần Thị Thanh Thuỷ

Nhóm lớp học phần : 04

Nhóm bài tập lớn : 02 (TTT23)

Sinh viên thực hiện:

Trần Đức Lộc ***MSV: B21DCCN492***

Hoàng Ngọc Minh ***MSV: B21DCCN084***

Hà Nội – 2024

Lời cảm ơn

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến cô Trần Thị Thanh Thủy, đã dành thời gian và công sức để hướng dẫn và giảng dạy cho chúng em trong suốt quá trình học môn IoT và ứng dụng. Sự tận tình và sự hỗ trợ của cô đã giúp chúng em hiểu sâu hơn về IoT và áp dụng nó vào trong đời sống hàng ngày.

Chúng em cũng muốn bày tỏ lòng biết ơn đối với cơ hội mà cô đã tạo ra cho chúng em để thực hiện bài tập lớn này. Qua việc thực hành và áp dụng những kiến thức được học vào thực tế, chúng em đã có cơ hội trải nghiệm và học hỏi nhiều điều mới mẻ và bổ ích.

Tuy nhiên, sau tất cả những nỗ lực và cố gắng, chúng em nhận thấy rằng bài báo cáo của chúng em vẫn còn nhiều thiếu sót và hạn chế. Với vốn kiến thức và kinh nghiệm ít ỏi, chúng em hiểu rằng còn phải nỗ lực và rèn luyện thêm nhiều để có thể hoàn thiện hơn trong tương lai.

Vì vậy, chúng em xin kính mong sự thông cảm và góp ý từ phía cô để chúng em có thể rút kinh nghiệm và cải thiện bản thân mình mỗi ngày. Những đóng góp và ý kiến phản hồi của cô sẽ là động lực quan trọng giúp chúng em tiến xa hơn trên con đường học tập và nghiên cứu.

Chúng em xin chân thành cảm ơn cô!

Mục lục

Danh mục hình ảnh

Mở đầu

1. Giới thiệu

Đề tài "Mô Hình Ô Tô Đồ Chơi Điều Khiển Từ Xa Tích Hợp Điều Khiển Giọng Nói và Cảm Biến Vật Cản". Bọn em chọn đề tài này nhằm phát triển một mô hình ô tô đồ chơi có khả năng điều khiển từ xa với các tính năng thông minh. Mục tiêu của đề tài là ứng dụng các công nghệ điều khiển và cảm biến đơn giản để xây dựng một hệ thống ô tô mô hình có thể đáp ứng lệnh điều khiển của người dùng thông qua giọng nói và có khả năng phát hiện chướng ngại vật trên đường di chuyển.

Mô hình ô tô này không chỉ mang lại tính thực tiễn trong việc điều khiển từ xa mà còn giúp người thực hiện hiểu rõ hơn về các khái niệm cơ bản trong lĩnh vực tự động hóa và điều khiển thông minh. Cụ thể, ô tô được thiết kế để nhận lệnh giọng nói, tạo ra một phương thức điều khiển dễ dàng và thuận tiện. Ngoài ra, cảm biến vật cản được tích hợp vào mô hình giúp xe tự động phát hiện và tránh va chạm, nâng cao tính an toàn trong quá trình vận hành.

Đề tài này giúp chúng em rèn luyện kỹ năng thiết kế và triển khai hệ thống điều khiển tích hợp, nắm bắt các kiến thức và kỹ thuật cơ bản để áp dụng vào các ứng dụng thực tế khác trong tương lai. Đây là bước đầu tiên giúp bọn em làm quen với các hệ thống điều khiển và cảm biến, đồng thời cũng mang lại trải nghiệm thú vị và học hỏi được rất nhiều thứ.

2. Lý do chọn đề tài

Trong thời đại công nghệ số hiện nay, các thiết bị thông minh và điều khiển tự động đang ngày càng trở nên phổ biến và thiết thực trong đời sống. Điều này không chỉ mang lại sự tiện lợi mà còn giúp con người tiết kiệm thời gian, công sức và gia tăng tính an toàn khi vận hành. Bọn em chọn đề tài này nhằm mục đích kết hợp giữa kiến thức lý thuyết và thực hành trong lĩnh vực điều khiển từ xa và công nghệ cảm biến.

Mục tiêu của đề tài là xây dựng một hệ thống đơn giản nhưng có ý nghĩa thực tiễn, giúp người thực hiện có cơ hội áp dụng kiến thức về điều khiển, lập trình và cảm biến vào một sản phẩm cụ thể – ô tô mô hình. Mô hình ô tô đồ chơi điều khiển từ xa không chỉ tạo ra trải nghiệm thú vị mà còn là cách tiếp cận thân thiện cho việc nghiên cứu và phát triển các hệ thống tự động hóa ở quy mô nhỏ. Việc tích hợp chức năng điều khiển giọng nói giúp tăng tính linh hoạt trong điều khiển, tạo cảm giác hiện đại và tiện

ích, trong khi cảm biến vật cản giúp xe tránh được các chướng ngại vật trên đường đi, đảm bảo an toàn cho hệ thống.

Ngoài ra, đề tài này còn là cơ hội để bọn em nâng cao các kỹ năng về lập trình vi điều khiển, xử lý tín hiệu Bluetooth, và xử lý dữ liệu cảm biến, từ đó làm nền tảng cho những dự án phức tạp hơn trong tương lai. Hệ thống này không chỉ đơn thuần là một bài tập lớn, mà còn là một bước đầu giúp bọn em thực hiện làm quen với các công nghệ thông minh và ứng dụng của chúng trong thực tiễn. Qua đó, đề tài có ý nghĩa giáo dục và thực hành rõ rệt, đóng góp tích cực vào quá trình học tập và phát triển kỹ năng nghề nghiệp của bọn em sau này.

3. Mục tiêu nghiên cứu

- Xây dựng hệ thống điều khiển từ xa cho mô hình ô tô đồ chơi: Phát triển một mô hình ô tô có thể điều khiển từ xa bằng các thiết bị di động như điện thoại thông minh hoặc máy tính bảng thông qua kết nối Bluetooth.
- Tích hợp điều khiển giọng nói: Nghiên cứu và triển khai chức năng điều khiển giọng nói, cho phép người dùng điều khiển xe bằng các lệnh giọng nói cơ bản, tạo sự linh hoạt và tiện lợi trong quá trình sử dụng.
- Ứng dụng cảm biến vật cản để đảm bảo an toàn: Tích hợp cảm biến vật cản vào mô hình ô tô nhằm giúp xe có khả năng tự động phát hiện và tránh chướng ngại vật trên đường di chuyển, từ đó tăng cường tính an toàn và khả năng phản ứng của hệ thống.
- Đánh giá hiệu quả hoạt động của hệ thống: Thực hiện các thử nghiệm để kiểm tra và đánh giá hiệu quả hoạt động của mô hình, từ khả năng nhận diện và phản hồi lệnh giọng nói, độ chính xác của cảm biến vật cản, cho đến tính ổn định của hệ thống điều khiển từ xa.

Chương 1: Cơ sở lý thuyết

1.1. Intenet of thing (IOT)

Internet of Things (IoT) ra đời vào cuối những năm 1990 khi các nhà nghiên cứu bắt đầu khám phá khả năng kết nối các thiết bị với Internet. Ban đầu, ý tưởng này tập trung vào việc làm cho các thiết bị hằng ngày "thông minh" hơn bằng cách gắn cảm biến và khả năng giao tiếp từ xa. Tuy nhiên, chỉ đến những năm 2010, khi công nghệ di động và đám mây phát triển mạnh, IoT mới thực sự bùng nổ.

Hình 1. 1: IOT và ứng dụng

IoT là một mạng lưới các thiết bị kết nối với Internet, chia sẻ dữ liệu với nhau và hoạt động tự động hoặc thông qua sự điều khiển từ xa. Các thiết bị này có thể là cảm biến, thiết bị gia dụng, ô tô hoặc máy móc công nghiệp. Với khả năng thu thập và phân tích dữ liệu theo thời gian thực, IoT đã thay đổi cách chúng ta tương tác với môi trường xung quanh.

IoT được ứng dụng rất đa dạng, từ ngôi nhà thông minh (smart home), thành phố thông minh (smart city) đến quản lý chuỗi cung ứng, chăm sóc sức khỏe, và nông nghiệp.

Trong công nghiệp, IoT giúp tối ưu hóa quy trình sản xuất và giám sát máy móc. Trong đời sống hàng ngày, nó tạo ra trải nghiệm cá nhân hóa và tiện lợi hơn cho người dùng.

1.2. Các thiết bị phần cứng

1.2.1. Vi điều khiển ESP32-WROOM-32 NodeMCU-32S

Module ESP32-WROOM-32 là một vi điều khiển mạnh mẽ được phát triển bởi Espressif Systems. Với khả năng tích hợp kết nối Wi-Fi và Bluetooth, cùng với sức mạnh từ vi xử lý lõi kép, khả năng xử lý song song, cùng với nhiều tính năng tiên tiến khác, ESP32-WROOM-32 đã trở thành sự lựa chọn cho các dự án IoT, tự động hóa, và nghiên cứu công nghệ.

Trên cơ sở module ESP32-WROOM-32, các nhà sản xuất đã tạo ra nhiều bảng phát triển, trong đó NodeMCU-32S là một bảng phổ biến, thiết kế với kích thước nhỏ gọn, tích hợp các giao diện và kết nối cần thiết, thuận tiện cho việc nghiên cứu, học tập, và phát triển sản phẩm.

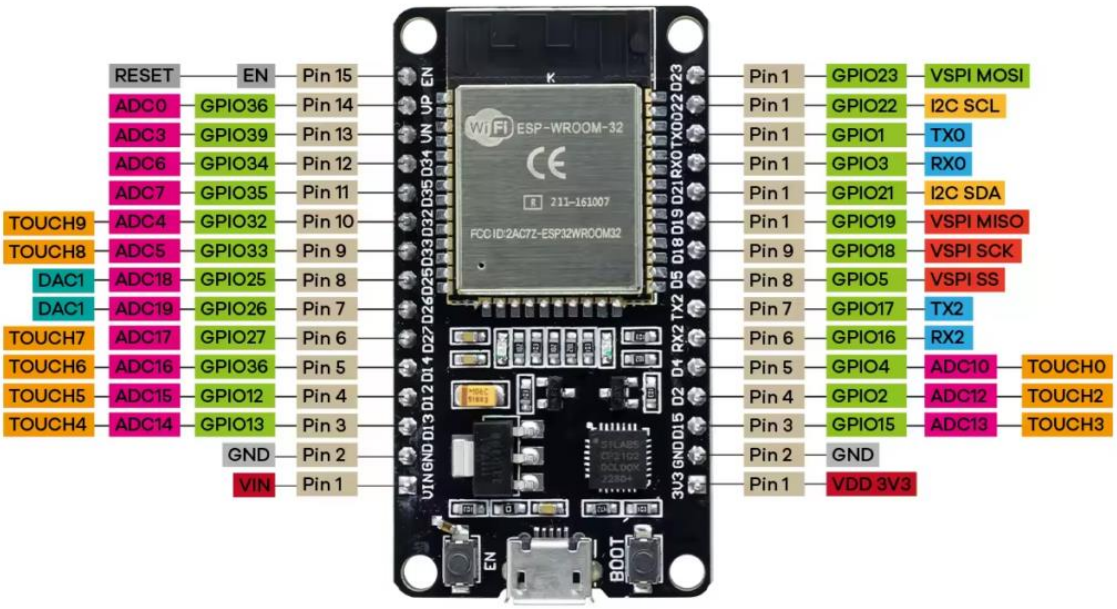
Hình 1. 2: Vi điều khiển ESP32-WROOM-32 NodeMCU-32S

a. Nguyên lý hoạt động

ESP32-WROM-32 NodeMCU 32S hoạt động dựa trên nguyên lý điều khiển điện tử. Khi một tín hiệu điện được gửi đến các chân GPIO, vi điều khiển sẽ thực hiện các tác vụ tương ứng, như bật/tắt thiết bị hoặc đọc dữ liệu từ cảm biến. Các chân VIN và GND cung cấp nguồn cho toàn bộ mạch, trong khi chân VP cho phép vi điều khiển nhận giá trị điện áp từ các cảm biến analog. Việc tích hợp Wi-Fi và Bluetooth cho phép ESP32 kết nối với mạng không dây và giao tiếp với các thiết bị khác, mở ra nhiều khả năng cho các ứng dụng IoT. Nhờ vào bộ vi xử lý mạnh mẽ và khả năng kết nối linh hoạt, ESP32-

WROM-32 NodeMCU 32S là lựa chọn lý tưởng cho các dự án điện tử phức tạp, giúp người dùng phát triển và triển khai các ứng dụng một cách hiệu quả.

b. Thông số kỹ thuật



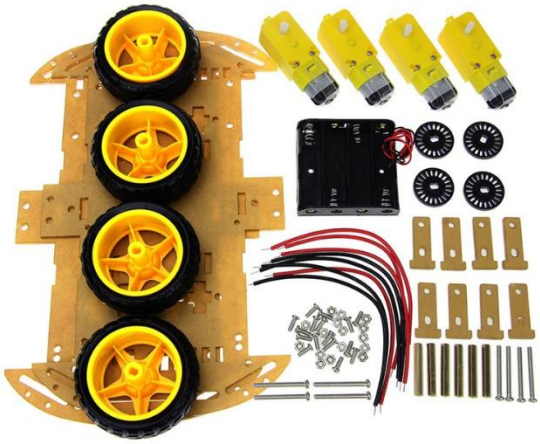
Hình 1. 3: Sơ đồ chân ESP32-WROOM NodeMCU-32S

- CPU**:** Bộ vi xử lý lõi kép, hoạt động với tần số 240 MHz, giúp xử lý nhanh chóng và đồng thời nhiều tác vụ phức tạp.
- Bộ nhớ:
  - RAM: 520 KB SRAM.
  - Flash Memory: 4MB
- Wi-Fi: Hỗ trợ chuẩn 802.11 b/g/n, với băng tần 2.4 GHz
- Bluetooth: Bluetooth v4.2
- GPIO (General Purpose Input/Output): Tổng cộng có 34 chân GPIO đa dụng, mỗi chân có thể cấu hình cho các chức năng khác nhau:
  - Digital Input/Output: Các chân GPIO có thể được lập trình để hoạt động như các chân đầu vào hoặc đầu ra, giúp ESP32 có thể đọc hoặc điều khiển các thiết bị ngoại vi như LED, nút nhấn, relay,….
  - PWM (Pulse Width Modulation): Điều khiển độ sáng của LED, động cơ hoặc các thiết bị cần điều chỉnh công suất một cách linh hoạt.
  - I2C, SPI, UART: Là các giao thức giao tiếp phổ biến, giúp ESP32 kết nối và giao tiếp với các thiết bị như cảm biến, module hiển thị, hoặc các vi điều khiển khác.
  - Touch Sensor: Được cấu hình để nhận tín hiệu từ các bề mặt cảm ứng

- ADC (Analog to Digital Converter): ESP32 có 18 kênh ADC 12-bit, cho phép đọc tín hiệu analog từ cảm biến nhiệt độ, độ ẩm,…
- DAC (Digital to Analog Converter): Có 2 kênh DAC 8-bit, cho phép xuất tín hiệu analog để điều khiển các thiết bị như loa, tín hiệu điều khiển động cơ,..

1.2.2. Xe ô tô giả lập (xe mô hình)

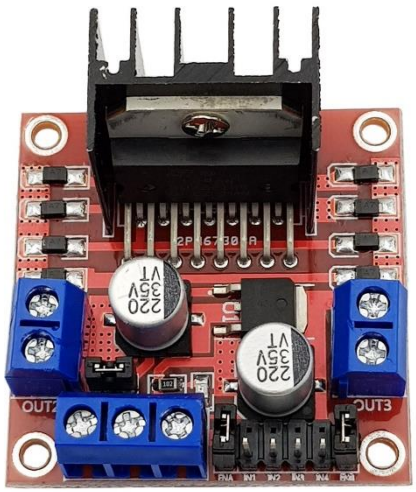
Gồm 4 động cơ hộp giảm tốc, điện áp cung cấp cho động cơ 3-6V.



Hình 1. 4: Bộ khung xe bằng mica

1.2.3. Module điều khiển động cơ L298N

Module L298N là một mạch tích hợp (IC) được sử dụng rộng rãi trong các dự án điện tử, đặc biệt là các dự án liên quan đến điều khiển động cơ DC. Nó cho phép bạn điều khiển một hoặc hai động cơ DC một chiều, hoặc một động cơ bước 4 pha.



Hình 1. 5: Module điều khiển động cơ L298N

a. Thông số kỹ thuật

- Điện áp cấp cho module: 5-7V dùng để điều khiển các chân IN1, IN2, IN3, IN4 và 5-35V điện áp cấp cho động cơ.
- Dòng điện tối đa mỗi kênh: 2A.
- Công suất tối đa mỗi kênh: 25W
- Điện áp hoạt động 3.3V – 5V.
- Nhiệt độ hoạt động: -25°C đến +130°C
- IC điều khiển: L298N
- Module có các chân sau:
    - 12V in: Cung cấp nguồn điện cho động cơ. Điện áp này thường dao động từ 5V đến 35V, tùy thuộc vào động cơ bạn sử dụng.
    - 5V in: Cung cấp nguồn điện 5V cho phần logic của module.
    - GND: Chân nối đất chung cho cả module và động cơ.
    - IN1, IN2, IN3, IN4: Các chân này nhận tín hiệu điều khiển từ vi điều khiển (như Arduino) để quyết định chiều quay và tốc độ của động cơ. Bằng cách kết hợp các mức cao (HIGH) và thấp (LOW) tại các chân này, bạn có thể điều khiển động cơ theo ý muốn.
    - Enable A, Enable B: Các chân này cho phép bạn bật hoặc tắt từng kênh điều khiển riêng biệt. Khi chân Enable được đặt ở mức cao, kênh tương ứng sẽ hoạt động.
    - OUT1, OUT2, OUT3, OUT4: Các chân này kết nối trực tiếp với các cực của động cơ. Bằng cách điều khiển các transistor bên trong module, bạn có thể thay đổi chiều dòng điện qua động cơ, từ đó làm cho động cơ quay theo chiều mong muốn.

b. Nguyên lý hoạt động

Module L298N hoạt động dựa trên cấu trúc cầu H, cho phép điều khiển chiều quay và tốc độ của động cơ DC. Khi nhận được tín hiệu điều khiển từ vi điều khiển, các transistor bên trong module sẽ đóng mở linh hoạt, tạo ra các đường dẫn cho dòng điện đi qua động cơ theo hai chiều ngược nhau. Bằng cách thay đổi tổ hợp đóng mở của các transistor, ta có thể làm cho động cơ quay thuận hoặc ngược chiều. Ngoài ra, bằng cách điều chỉnh độ rộng xung (PWM) của tín hiệu điều khiển, ta có thể điều chỉnh tốc độ quay của động cơ một cách chính xác. Nhờ vậy, module L298N trở thành một công cụ hữu ích trong nhiều ứng dụng điều khiển động cơ trong các dự án điện tử.

1.2.4. Module cảm biến Siêu Âm HC-SR04:

Cảm biến siêu âm Ultrasonic HC-SR04 được sử dụng để nhận biết khoảng cách từ vật thể đến cảm biến nhờ sóng siêu âm, cảm biến có thời gian phản hồi nhanh, độ chính xác cao, phù hợp cho các ứng dụng phát hiện vật cản, đo khoảng cách bằng sóng siêu âm.

Hình 1. 6: Cảm biến siêu âm HC-SR04

a. Thông số kĩ thuật

- Điện áp hoạt động : 5V
- Dòng điện tiêu thụ : < 2mA
- Khoảng cách đo được: 2cm – 450cm
- Góc tối đa: 15 độ
- Sai số: 3mm
- Các chân của cảm biến Siêu Âm HC-SR04:
    - Chân Vcc: Chân này dùng để cấp nguồn cho cảm biến. Thông thường, cảm biến hoạt động tốt nhất với điện áp 5V DC
    - Chân Trig: Chân này là chân kích hoạt. Khi bạn gửi một xung điện ngắn (ví dụ: 10 micro giây) đến chân này, cảm biến sẽ phát ra một sóng siêu âm.
    - Chân Echo: Chân này là chân nhận. Khi sóng siêu âm dội lại và được cảm biến thu nhận, chân này sẽ xuất ra một xung điện. Thời gian dài của xung điện này tỷ lệ thuận với khoảng cách từ cảm biến đến vật cản.
    - Chân GND: Chân này là chân nối đất, dùng để tạo một đường dẫn cho dòng điện trở về nguồn.

b. Nguyên lý hoạt động của cảm biến Siêu Âm HC-SR04

Chỉ cần cung cấp một xung ngắn 10uS cho đầu vào kích hoạt để bắt đầu đo khoảng cách, sau đó mô-đun sẽ gửi ra một xung siêu âm 8 chu kỳ ở tần số 40 kHz và tăng tiếng vang của nó. Tiếng vang là một đối tượng khoảng cách có độ rộng xung và

phạm vi theo tỷ lệ. Có thể tính toán phạm vi thông qua khoảng thời gian giữa việc gửi tín hiệu kích hoạt và nhận tín hiệu tiếng vang.

Công thức: uS / 58 = cm hoặc uS / 148 = inch; hoặc: phạm vi = thời gian mức cao * vận tốc (340M/S) / 2.

1.3. Bluetooth và các giao thức

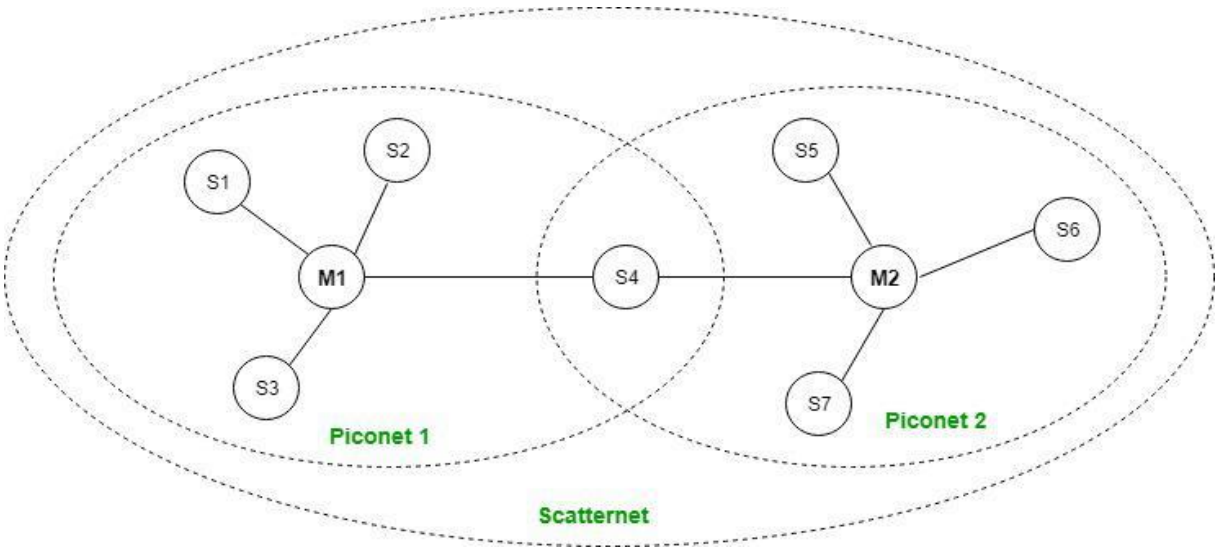
1.3.1. Bluetooth

a. Khái niệm

Bluetooth được sử dụng để truyền thông không dây trong phạm vi ngắn đối với dữ liệu và giọng nói. Đây là một công nghệ mạng khu vực cá nhân không dây (WPAN) và được sử dụng cho truyền thông dữ liệu ở khoảng cách ngắn. Công nghệ này được phát minh bởi Ericsson vào năm 1994. Nó hoạt động trong các băng tần không cần cấp phép dành cho công nghiệp, khoa học và y tế (ISM) từ 2.4 GHz đến 2.485 GHz.

Phạm vi hoạt động của Bluetooth lên đến 10 mét. Tùy thuộc vào phiên bản, nó cung cấp tốc độ truyền dữ liệu ít nhất là 1 Mbps hoặc 3 Mbps. Phương pháp trải phổ mà Bluetooth sử dụng là FHSS (Frequency-Hopping Spread Spectrum - Phương pháp Trải Phổ Nhảy Tần). Một mạng Bluetooth được gọi là "piconet", và một nhóm các piconet kết nối với nhau được gọi là "scatternet".

b. Kiến trúc

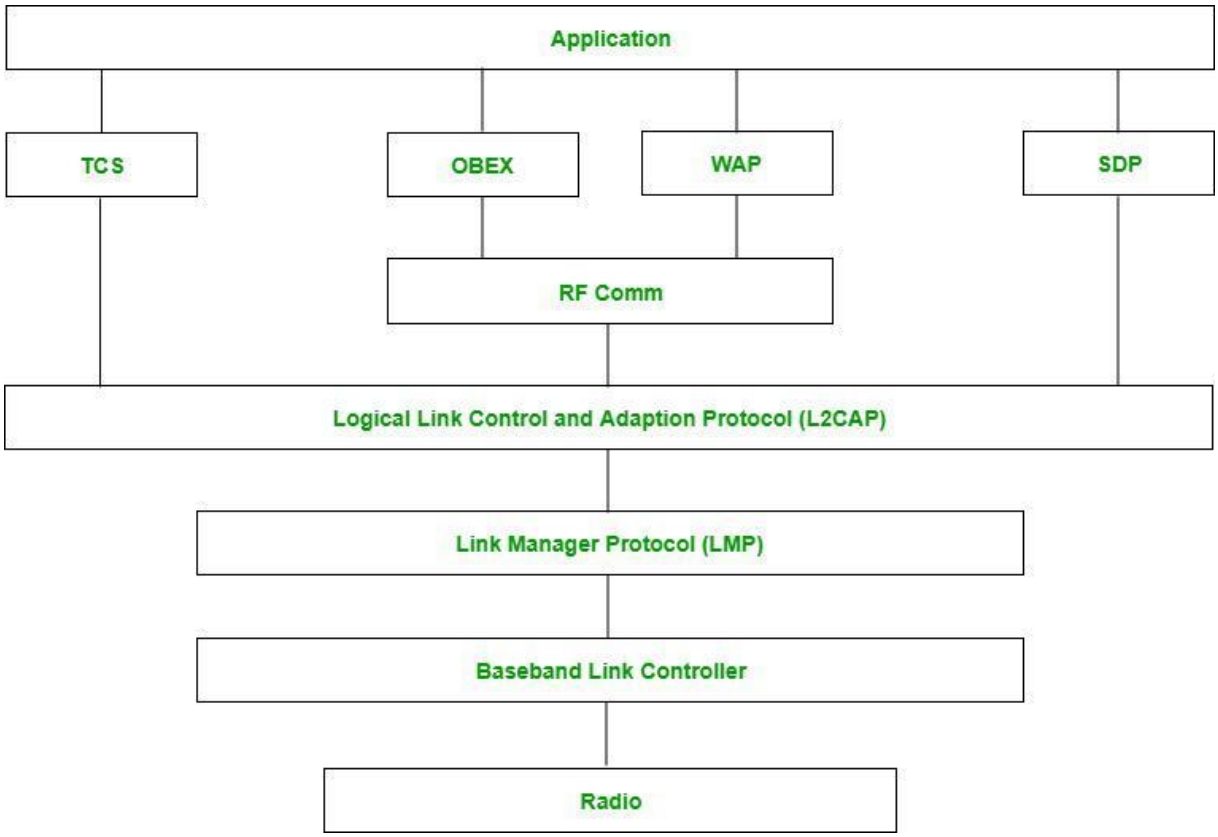


Hình 1. 7: Kiến trúc BlueTooth

Piconet là một loại mạng Bluetooth bao gồm một nút chính, được gọi là "master" và bảy nút phụ đang hoạt động, được gọi là "slave". Giao tiếp giữa nút chính và các nút phụ có thể là một-nhiều hoặc một-một. Tuy nhiên, giao tiếp chỉ có thể diễn ra giữa master và slave; giao tiếp giữa các slave với nhau là không thể. Ngoài ra, piconet còn có 255 nút ở trạng thái chờ (parked nodes), là các nút phụ và không thể tham gia giao tiếp cho đến khi được chuyển sang trạng thái hoạt động.

Scatternet được tạo ra bằng cách kết hợp nhiều piconet. Một nút slave trong một piconet có thể đóng vai trò là master hoặc nút chính trong một piconet khác. Loại nút này có thể nhận tin nhắn từ master trong một piconet và chuyển tin nhắn đó tới slave của nó trong piconet khác mà nó đóng vai trò master. Loại nút này được gọi là "nút cầu nối" (bridge node). Một trạm không thể là master trong hai piconet cùng một lúc.

c. Ngăn xếp giao thức



Hình 1. 8: Ngăn xếp giao thức

Radio RF Layer: Bao gồm tần số, cách sử dụng kỹ thuật nhảy tần (frequency hopping), và công suất truyền. Nó thực hiện điều chế và giải điều chế dữ liệu thành các tín hiệu RF. Tầng này định nghĩa các đặc điểm vật lý của các bộ thu phát Bluetooth. Nó

xác định hai loại liên kết vật lý: kết nối không định hướng (connection-less) và kết nối có định hướng (connection-oriented).

Baseband Link Layer: Đây là động cơ kỹ thuật số của hệ thống Bluetooth và tương đương với tầng con MAC trong mạng LAN. Nó thực hiện việc thiết lập kết nối trong một piconet, định địa chỉ, định dạng gói tin, điều chỉnh thời gian và điều khiển công suất.

Link Manager Protocol Layer: Tầng này thực hiện việc quản lý các liên kết đã được thiết lập, bao gồm các quy trình xác thực và mã hóa. Nó chịu trách nhiệm tạo liên kết, giám sát tình trạng của chúng, và kết thúc kết nối một cách an toàn khi có lệnh hoặc khi xảy ra lỗi.

L2CAP: Còn được gọi là trung tâm của ngăn xếp giao thức Bluetooth. Nó cho phép giao tiếp giữa các tầng trên và dưới của ngăn xếp giao thức Bluetooth. Tầng này đóng gói các gói dữ liệu từ các tầng trên thành định dạng mà các tầng dưới yêu cầu. Nó cũng thực hiện phân đoạn và ghép kênh.

SDP: Đây là viết tắt của Service Discovery Protocol. Tầng này cho phép khám phá các dịch vụ có sẵn trên một thiết bị khác có hỗ trợ Bluetooth.

RF Comm Layer: Là giao thức thay thế cho kết nối có dây. Viết tắt của Radio Frontend Component. Nó cung cấp giao diện nối tiếp với WAP và OBEX. Tầng này cũng cung cấp khả năng mô phỏng các cổng nối tiếp thông qua giao thức L2CAP. Giao thức này dựa trên tiêu chuẩn ETSI TS 07.10.

OBEX: Viết tắt của Object Exchange. Đây là một giao thức truyền thông để trao đổi các đối tượng (dữ liệu) giữa hai thiết bị.

WAP: Viết tắt của Wireless Access Protocol. Được sử dụng để truy cập internet.

TCS: Viết tắt của Telephony Control Protocol. Nó cung cấp dịch vụ điện thoại. Chức năng cơ bản của tầng này là điều khiển cuộc gọi (thiết lập và giải phóng cuộc gọi) và quản lý nhóm cho cổng kết nối phục vụ nhiều thiết bị.

Tầng Ứng dụng (Application Layer): Cho phép người dùng tương tác với ứng dụng.

d. Giao thức Bluetooth Serial Port Profile (SPP)

Bluetooth SPP là một phần của ngăn xếp giao thức Bluetooth cổ điển, cho phép các thiết bị Bluetooth truyền dữ liệu qua kết nối mô phỏng cổng nối tiếp, giống như cách truyền dữ liệu qua cổng COM hoặc UART.

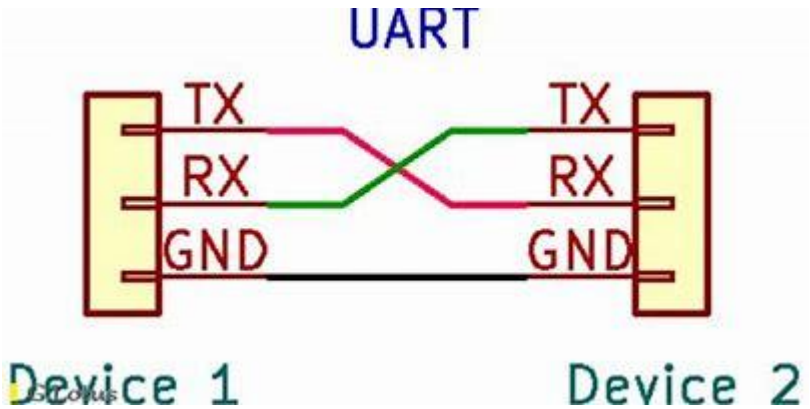
Giao tiếp Point-to-Point: Bluetooth SPP cho phép thiết lập một kết nối trực tiếp giữa hai thiết bị (ESP32 và server) để truyền dữ liệu hai chiều một cách liên tục và ổn định.

Giao thức nền tảng (Underlying Protocol): Bluetooth SPP sử dụng RFCOMM – một giao thức trong ngăn xếp Bluetooth giúp mô phỏng kết nối cổng nối tiếp. RFCOMM được xây dựng dựa trên chuẩn ETSI TS 07.10, cho phép truyền dữ liệu nối tiếp qua Bluetooth.

Ứng dụng: Bluetooth SPP phù hợp cho các ứng dụng truyền dữ liệu nối tiếp, liên tục và theo chuỗi, giữa các thiết bị IoT với máy tính hoặc các thiết bị khác, đặc biệt trong các ứng dụng yêu cầu kết nối đơn giản và hiệu quả.

Tính năng nổi bật: SPP cung cấp một phương thức đơn giản để gửi và nhận dữ liệu serial mà không cần dây nối, với khả năng dễ dàng thiết lập và tương thích với các thiết bị có hỗ trợ Bluetooth.

1.3.2. UART



Hình 1. 9: Giao thức UART

UART (Universal Asynchronous Receiver-Transmitter) là một giao thức truyền thông không đồng bộ, được sử dụng để truyền và nhận dữ liệu nối tiếp giữa hai thiết bị, chẳng hạn như vi điều khiển và module ngoại vi. Nó phổ biến trong các hệ thống nhúng vì đơn giản và dễ triển khai.

Cách thức hoạt động của giao thức UART :

- Không đồng bộ: UART không sử dụng tín hiệu xung nhịp (clock) chung giữa các thiết bị. Thay vào đó, các thiết bị phải được cấu hình

để sử dụng cùng tốc độ truyền dữ liệu (baud rate), ví dụ: 9600, 115200 bps.

- TX (Transmit): Chân truyền dữ liệu.
- RX (Receive): Chân nhận dữ liệu.

Cấu trúc khung dữ liệu của giao thức UART :

- Start bit: Được gửi đầu tiên để báo hiệu bắt đầu truyền.
- Data bits: Thường từ 7-9 bit, chứa dữ liệu chính.
- Parity bit (tùy chọn): Kiểm tra lỗi (even hoặc odd parity).
- Stop bit: Kết thúc khung dữ liệu (1 hoặc 2 bit).

Nhìn chung UART là giao thức đơn giản, dễ sử dụng, cần ít dây nối. Tuy nhiên UART lại có độ trễ khá cao, chỉ hỗ trợ giao tiếp 1:1. Do vậy UART thường được sử dụng trong các hệ thống nhúng nhỏ, đơn giản.

1.4. Học máy

1.4.1. Xử lý dữ liệu

CountVectorizer là một công cụ trong thư viện scikit-learn dùng để chuyển đổi văn bản thành một ma trận số đếm các từ (Document-Term Matrix), phù hợp cho các mô hình học máy. Cụ thể, CountVectorizer sẽ tách từ trong các tài liệu văn bản, đếm số lần xuất hiện của mỗi từ và biểu diễn mỗi tài liệu bằng một vector tần suất từ. Công cụ này hữu ích trong các bài toán như phân loại văn bản, phân tích cảm xúc, và truy vấn thông tin.

Các tham số của CountVectorizer:

- Input : {‘filename’, ‘file’, ‘content’}, default=’content’: Xác định loại đầu vào
  - 'filename': Đầu vào là danh sách tên tệp.
  - 'file': Đầu vào là các đối tượng giống file.
  - 'content': Đầu vào là một chuỗi các phần tử dạng văn bản hoặc byte (mặc định).
- Encoding : str, default=’utf-8’: Chuẩn mã hóa được dùng để giải mã dữ liệu văn bản, mặc định là 'utf-8'.
- decode_error : {‘strict’, ‘ignore’, ‘replace’}, default=’strict’: Xác định nếu dữ liệu byte chứa ký tự không đúng mã hóa.
- strip_accents : {‘ascii’, ‘unicode’} or callable, default=None: Xác định có loại bỏ dấu và chuẩn hóa ký tự hay không.

- Lowercase : bool, default=True: Chuyển tất cả ký tự thành chữ thường trước khi tách từ (mặc định là True).
- ngram_range : tuple (min_n, max_n), default=(1, 1): Định nghĩa phạm vi n-grams từ hoặc ký tự được tạo ra, dưới dạng một tuple (min_n, max_n). Ví dụ, (1, 1) chỉ tạo ra unigrams (từ đơn), (1, 2) tạo ra cả unigrams và bigrams.
- Analyzer : {'word', 'char', 'char_wb'} or callable, default='word': Xác định đơn vị để tạo n-grams
- max_df : float in range [0.0, 1.0] or int, default=1.0: Bỏ qua các từ có tần suất tài liệu cao hơn ngưỡng đã cho, dùng để loại bỏ các từ phổ biến không mang ý nghĩa.
- min_df : float in range [0.0, 1.0] or int, default=1: Bỏ qua các từ có tần suất tài liệu thấp hơn ngưỡng đã cho, dùng để loại bỏ các từ hiếm.
- max_features : int, default=None: Giới hạn số từ đặc trưng dựa trên tần suất từ cao nhất. Nếu None, sử dụng tất cả từ trong từ điển.
- binary : bool, default=False: Nếu là True, tất cả số đếm khác 0 được đặt thành 1. Thích hợp cho các mô hình xử lý sự kiện nhị phân.

Các phương thức chính của CountVectorizer:

- fit(raw_documents): Xây dựng từ điển của các token từ tập tài liệu.
  - raw_documents: Iterable chứa các tài liệu văn bản hoặc đối tượng file.
  - Trả về: Đối tượng CountVectorizer đã học được từ điển.
- fit_transform(raw_documents): Học từ điển và trả về ma trận tài liệu "từ".
  - raw_documents: Iterable chứa các tài liệu văn bản hoặc đối tượng file.
  - Trả về: Ma trận tài liệu "từ" với kích thước (n_samples, n_features).
- transform(raw_documents): Chuyển đổi tài liệu thành ma trận tài liệu "từ" sử dụng từ điển đã học.
  - raw_documents: Iterable chứa các tài liệu văn bản hoặc đối tượng file.
  - Trả về: Ma trận tài liệu "từ" dạng thưa (sparse matrix).
- get_feature_names_out(): Lấy tên các đặc trưng đầu ra (tức các từ hoặc n-grams đã được trích xuất).
  - Trả về: Mảng các tên đặc trưng.
- inverse_transform(X): Trả về các từ trong mỗi tài liệu tương ứng với các mục không bằng 0 trong X.
  - X: Ma trận tài liệu "từ" dạng thưa.
  - Trả về: Danh sách các từ cho mỗi tài liệu.

- build_analyzer(): Trả về một hàm gọi được để xử lý dữ liệu đầu vào, bao gồm tiền xử lý, tách từ và tạo n-grams.
- build_preprocessor(): Trả về hàm tiền xử lý văn bản trước khi tách từ.
- build_tokenizer(): Trả về hàm tách từ thành một chuỗi các token.

```
from sklearn.feature_extraction.text import CountVectorizer
corpus = [
    'This is the first document.',
    'This document is the second document.',
    'And this is the third one.',
    'Is this the first document?',
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
vectorizer.get_feature_names_out()
# Kết quả: array(['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this'], ...)
print(X.toarray())
# Kết quả:
# [[0 1 1 1 0 0 1 0 1]
#  [0 2 0 1 0 1 1 0 1]
#  [1 0 0 1 1 0 1 1 1]
#  [0 1 1 1 0 0 1 0 1]]
vectorizer2 = CountVectorizer(analyzer='word', ngram_range=(2, 2))
X2 = vectorizer2.fit_transform(corpus)
vectorizer2.get_feature_names_out()
# Kết quả: array(['and this', 'document is', 'first document', 'is the', 'is this',
#        'second document', 'the first', 'the second', 'the third', 'third one',
#        'this document', 'this is', 'this the'], ...)
print(X2.toarray())
# Kết quả:
# [[0 0 1 1 0 0 1 0 0 0 0 1 0]
#  [0 1 0 1 0 1 0 1 0 0 1 0 0]
#  [1 0 0 1 0 0 0 0 1 1 0 1 0]
#  [0 0 1 0 1 0 1 0 0 0 0 0 1]]
```

Hình 1. 10: Xử lý dữ liệu đầu vào

1.4.2. Model

Multinomial Naive Bayes là là một thuật toán học máy phân loại xác suất để tính toán phân phối xác suất của dữ liệu văn bản, rất phù hợp với dữ liệu có các đặc điểm biểu diễn tần suất rời rạc hoặc số lượng sự kiện trong nhiều tác vụ xử lý ngôn ngữ tự nhiên.

Thuật ngữ “Multinomial” đề cập đến loại phân phối dữ liệu được mô hình giả định. Các tính năng trong phân loại văn bản thường là số lượng từ hoặc tần suất thuật ngữ. Phân phối multinomial được sử dụng để ước tính khả năng nhìn thấy một tập hợp số lượng từ cụ thể trong một tài liệu.

Hàm tính xác suất (PMF) của phân phối multinomial được sử dụng để mô hình hóa khả năng quan sát một tập hợp số lượng từ cụ thể trong một tài liệu.

$$P(D|c) = \frac{T_c!}{\prod_{i=1}^{V}(x_i!)} \prod_{i=1}^{V} \left(\frac{\theta_{c,i}^{x_i}}{x_i!} \right)$$

Hình 1. 11: Hàm tính xác suất

- c: Lớp (nhãn) của tài liệu, ví dụ như Spam hoặc Not Spam
- D: Tài liệu (Document), là văn bản cụ thể mà chúng ta đang phân loại.
- w_i: Một từ (Word) trong từ vựng.
- x_i: Số lượng từ x_i xuất hiện trong tài liệu D.
- T_c: Tổng số từ trong các tài liệu thuộc lớp c.

$$\theta_{c,i} = \frac{\text{count}(w_i, c) + 1}{\sum_w (\text{count}(w, c) + 1)}$$

Hình 1. 12: Công thức tính xác suất của một từ trong lớp

- $\theta_{c,i}$: Xác suất của từ w_i xuất hiện trong một tài liệu thuộc lớp c.

VD: Phân loại một tài liệu thành lớp 'Spam' hoặc 'Not Spam' và từ vựng chứa 3 từ: { buy, now, free } và phân loại một tài liệu mới theo số lượng từ.

Class	buy	now	free	Total Words
Spam	20	5	10	35
Not Spam	5	15	5	25

Hình 1. 13: Ví dụ về bài toán

Với nhãn là spam ta có các xác suất sau:

$\theta_{spam,buy} = \frac{20+1}{35+3} = \frac{21}{38}$

$\theta_{spam,now} = \frac{5+1}{35+3} = \frac{6}{38}$

$\theta_{spam,free} = \frac{10+1}{35+3} = \frac{11}{38}$

Với nhãn là not spam ta có các xác suất sau:

$\theta_{not\ spam,buy} = \frac{5+1}{25+3} = \frac{6}{28}$

$\theta_{not\ spam,now} = \frac{15+1}{25+3} = \frac{16}{28}$

$\theta_{not\ spam,free} = \frac{5+1}{25+3} = \frac{6}{28}$

Giả sử có D mới cần phân loại với số lần xuất hiện các từ như sau: {buy:1, now:0, free: 2}

$P(D|Spam) = (\frac{21}{38})^1 * (\frac{6}{38})^0 * (\frac{11}{38})^2$

$P(D|Not\ Spam) = (\frac{6}{28})^1 * (\frac{16}{28})^0 * (\frac{6}{28})^2$

Áp dụng định lý Bayes để tính xác suất hậu nghiệm của mỗi lớp:

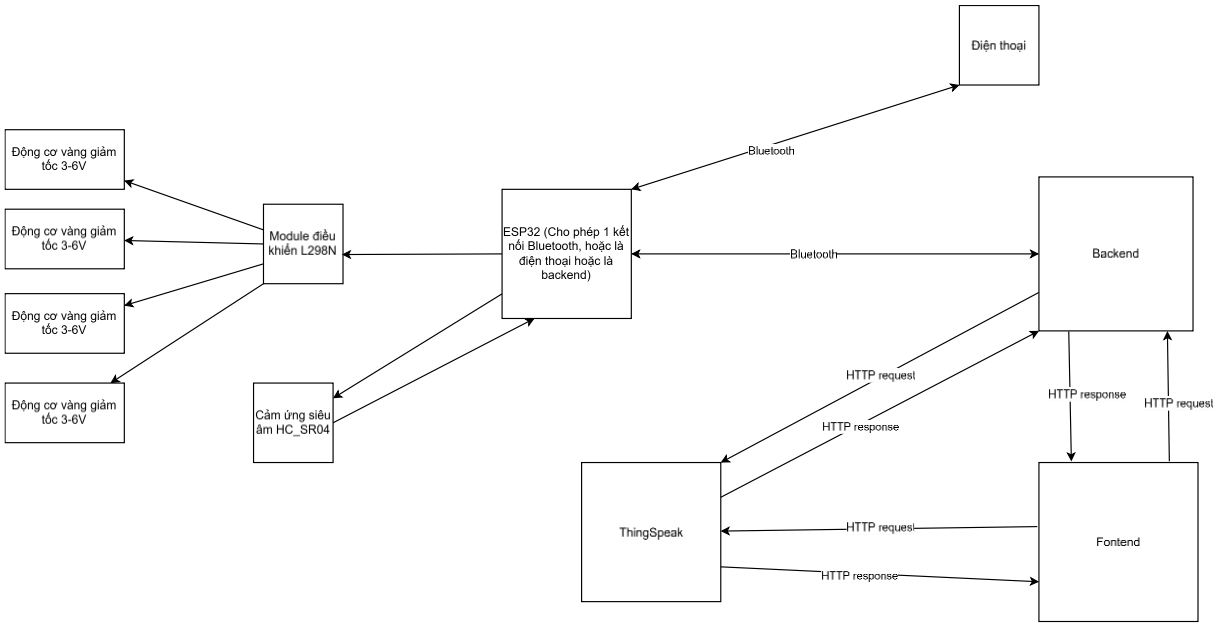
$P(Spam|D) = \frac{P(D|Spam)P(Spam)}{P(D)}$

$P(Not\ Spam|D) = \frac{P(D|Not\ Spam)P(Not\ Spam)}{P(D)}$

So sánh các xác suất và phân loại tài liệu vào lớp có xác suất cao hơn.

Chương 2: Thiết kế hệ thống

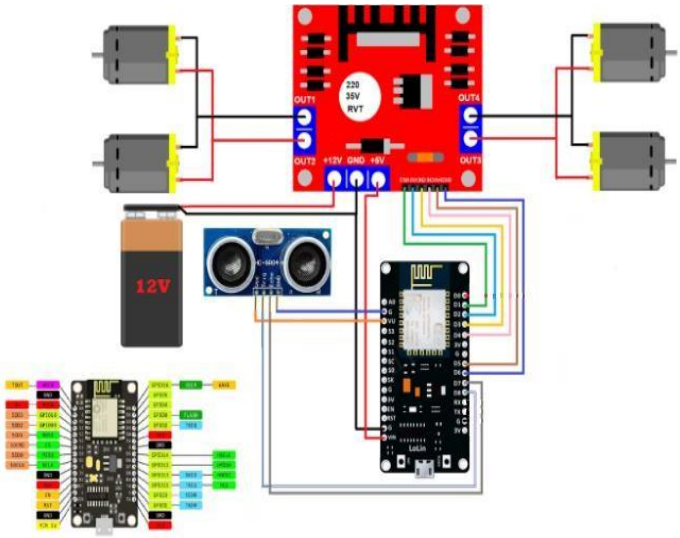
1.1. Sơ đồ khối hệ thống



Hình 2. 1: Sơ đồ khối hệ thống

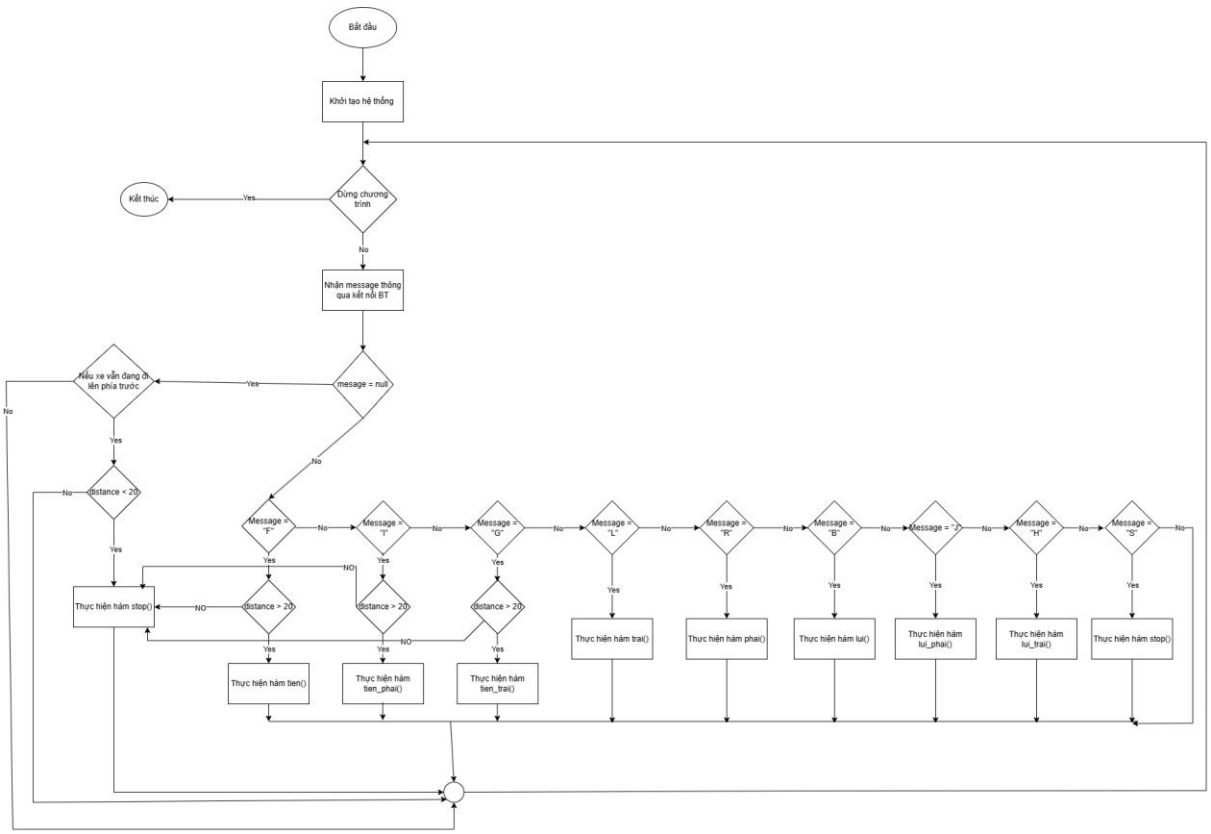
- ESP32: Đây là vi điều khiển trung tâm, nhận và xử lý dữ liệu từ các cảm biến và điều khiển động cơ thông qua module L298N.
- Module điều khiển L298N: Đóng vai trò điều khiển các động cơ DC, cho phép ESP32 điều khiển hướng quay và tốc độ của động cơ.
- Động cơ DC giảm tốc 3-6V: Các động cơ này được điều khiển bởi module L298N để thực hiện các hoạt động di chuyển.
- Cảm biến siêu âm HC-SR04: Đo khoảng cách phía trước, gửi dữ liệu về cho ESP32.
- ThingSpeak: Dịch vụ đám mây để lưu trữ và phân tích dữ liệu từ ESP32, có thể được sử dụng để hiển thị thông tin từ cảm biến hoặc trạng thái của hệ thống.
- Backend và Frontend: Phần Backend chịu trách nhiệm xử lý dữ liệu và gửi các lệnh điều khiển đến ESP32. Phần Frontend là giao diện người dùng, cho phép người dùng theo dõi và điều khiển hệ thống từ xa.

2.2. Sơ đồ mạch kết nối thiết bị phần cứng



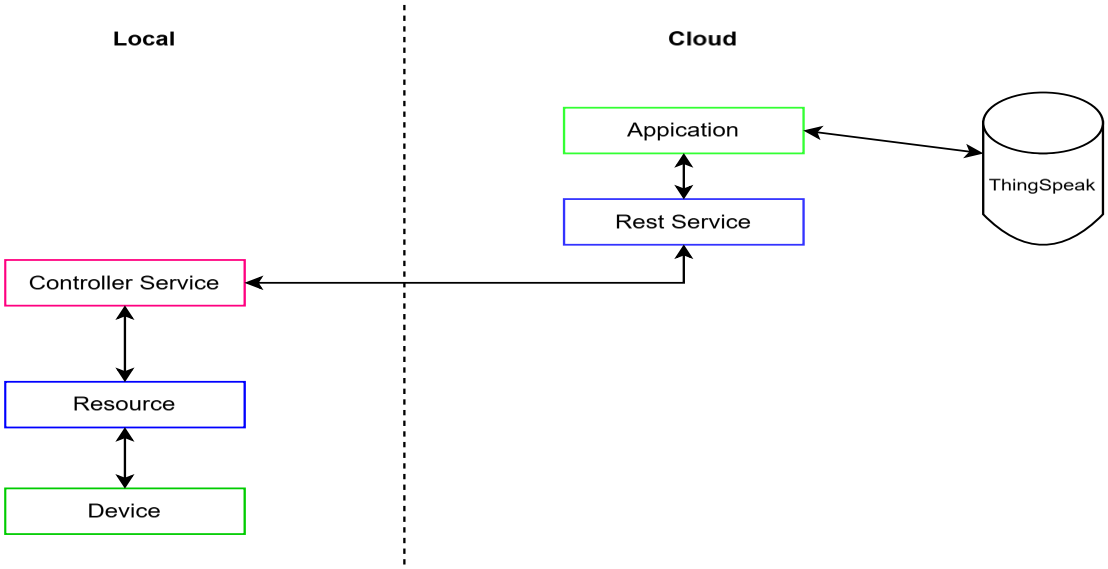
Hình 2. 2: Sơ đồ mạch kết nối phần cứng

2.3. Đặc tả tiến trình



Hình 2. 3: Đặc tả tiến trình

2.4. Đặc tả cấp độ IOT



Hình 2. 4: Đặc tả cấp độ IoT

2.5. Code chương trình điều khiển

2.5.1. Code phần cứng

a. Khai báo thư viện và hằng số

Khai báo các thư viện cần thiết cho việc kết nối Bluetooth, đo khoảng cách vật cản.

```
#include "BluetoothSerial.h"
#include <NewPing.h>
```

Hình 2. 5: Code khai báo thư viện

Khởi tạo đối tượng Bluetooth Serial, định nghĩa các chân cảm biến, chân điều khiển động cơ

```
BluetoothSerial SerialBT; // Khởi tạo đối tượng Bluetooth Serial

// Cảm biến siêu âm
#define TRIG_PIN 18
#define ECHO_PIN 5

// Định nghĩa chân điều khiển động cơ
const int enA = 14;
const int in1 = 27;
const int in2 = 26;
const int in3 = 25;
const int in4 = 33;
const int enB = 32;

int speed = 200;
volatile int distance = 0; // Biến đo khoảng cách
char dieu_khien;

// Task handle cho task đo khoảng cách
TaskHandle_t distanceTaskHandle;
```

Hình 2. 6: Khai báo các thông số và định nghĩa chân cảm biến, chân động cơ

b. Cấu hình cho các thiết bị khi esp32 khởi động

```
void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32_BT");
  Serial.println("Bluetooth is ready. Pair with ESP32_BT to start!");
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

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

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  analogWrite(enA, speed);
  analogWrite(enB, speed);

  // Tạo task đo khoảng cách trên lõi 0
  xTaskCreatePinnedToCore(
    measureDistanceTask, // Hàm thực hiện đo khoảng cách
    "Distance Task",      // Tên task
    1000,                 // Kích thước stack
    NULL,                 // Tham số cho task (không cần)
    1,                    // Mức ưu tiên
    &distanceTaskHandle,  // Task handle
    0                     // Chạy trên lõi 0
  );
}
```

Hình 2. 7: Setup cấu hình khi ESP32 khởi động

c. Hàm loop() - vòng lặp chính

```
void loop() {
  // Xử lý tín hiệu Bluetooth từ điện thoại
   if (distance > 1 && distance < 20) {
    if (digitalRead(in1) == LOW && digitalRead(in2) == HIGH && digitalRead(in3) == LOW && digitalRead(in4) == HIGH) {

    }
    else{
      Stop();
    }
  }
  if (SerialBT.available()) {
    dieu_khien = SerialBT.read();
    Serial.println(dieu_khien);
    switch (dieu_khien)
    {
      case 'F':
        if (distance > 1 && distance < 20) Stop();
        else{
              tien();
              SerialBT.println(distance);
        }
        break;
      case 'B':
        SerialBT.println(distance);
        lui();
        break;
      case 'L':
        trai();
        SerialBT.println(distance);
        break;
      case 'R':
        phai();
        SerialBT.println(distance);
        break;
      case 'I':
        tien_phai();
        SerialBT.println(distance);
        break;
      case 'G':
        tien_trai();
        SerialBT.println(distance);
        break;
      case 'J':
        lui_phai();
        SerialBT.println(distance);
        break;
      case 'H':
        lui_trai();
        SerialBT.println(distance);
        break;
      case 'S':
        Stop();
        SerialBT.println(distance);
        break;
      case '0':
        SerialBT.println(distance);
        Serial.print("Distance send to server");
        Serial.println(distance);
        break;
    }
  }
}
```

Hình 2. 8: Vòng lặp chính

d. Các hàm điều khiển xe

```
// Các hàm điều khiển động cơ
void dieuKhienDongCo(bool in1_val, bool in2_val, bool in3_val, bool in4_val) {
  digitalWrite(in1, in1_val);
  digitalWrite(in2, in2_val);
  digitalWrite(in3, in3_val);
  digitalWrite(in4, in4_val);
}
void tien() { dieuKhienDongCo(HIGH, LOW, HIGH, LOW); }
void lui() { dieuKhienDongCo(LOW, HIGH, LOW, HIGH); }
void phai() { dieuKhienDongCo(HIGH, LOW, LOW, HIGH); }
void trai() { dieuKhienDongCo(LOW, HIGH, HIGH, LOW); }
void Stop() { dieuKhienDongCo(LOW, LOW, LOW, LOW); }
void tien_trai() { dieuKhienDongCo(LOW, LOW, HIGH, LOW);}
void tien_phai() { dieuKhienDongCo(HIGH, LOW, LOW, LOW); }
void lui_phai() { dieuKhienDongCo(LOW, HIGH, LOW, LOW);}
void lui_trai() { dieuKhienDongCo(LOW, LOW, LOW, HIGH);}
```

Hình 2. 9: Các hàm điều khiển động cơ

e. Hàm đo khoảng cách vật cản

```
// Hàm đo khoảng cách từ cảm biến siêu âm
void measureDistanceTask(void * parameter) {
  for (;;) {
    distance = getDistance();
    vTaskDelay(100 / portTICK_PERIOD_MS); // Kiểm tra mỗi 100ms
  }
}

// Hàm đo khoảng cách từ cảm biến siêu âm (không dùng blocking)
int getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  int distance = duration * 0.0343 / 2; // Tính khoảng cách (cm)
  return distance;
}
```

Hình 2. 10: Hàm đo khoảng cách vật cản

2.5.2. Code webserver

a. Fontend

```
function sendCommandToBackend(command) {
  fetch('http://127.0.0.1:5000/controll', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ speech: command }),
  })
    .then((response) => response.json())
    .then((data) => {
      console.log('Phản hồi từ API:', data);
      document.getElementById(
        'status',
      ).textContent = `Đã gửi lệnh: ${data.speech}`;
    })
    .catch((error) => {
      console.error('Lỗi khi gửi dữ liệu:', error);
      document.getElementById('status').textContent =
        'Có lỗi xảy ra khi gửi lệnh.';
    });
}
```

Hình 2. 11: Hàm gửi lệnh từ FE về BE

```
function recordVoice() {
  const recordButton = document.querySelector('.record-btn');
  const statusElement = document.getElementById('status');
  statusElement.textContent = 'Đang thu âm...';
  recordButton.classList.add('recording');

  // Thêm spinner
  const spinner = document.createElement('div');
  spinner.classList.add('spinner');
  recordButton.appendChild(spinner);

  if (!('webkitSpeechRecognition' in window)) {
    console.log('Trình duyệt không hỗ trợ Web Speech API');
    statusElement.textContent = 'Trình duyệt không hỗ trợ Web Speech API';

    // Xóa hiệu ứng khi gặp lỗi
    recordButton.classList.remove('recording');
    if (spinner) recordButton.removeChild(spinner);
  } else {
    const recognition = new webkitSpeechRecognition();
    recognition.lang = 'vi-VN'; // Đặt ngôn ngữ là tiếng Việt
    recognition.interimResults = false;
    recognition.maxAlternatives = 1;

    // Khi bắt đầu thu âm
    recognition.onstart = function () {
      console.log('Đang thu âm...');
    };

    // Khi nhận diện giọng nói thành công
    recognition.onresult = function (event) {
      const transcript = event.results[0][0].transcript;
      console.log('Đã nhận diện giọng nói: ' + transcript);
      statusElement.textContent = 'Đã thu âm: ' + transcript;

      // Xóa hiệu ứng sau khi hoàn thành thu âm
      recordButton.classList.remove('recording');
      if (spinner) recordButton.removeChild(spinner);
      sendCommandToBackend(transcript);
    };

    // Khi gặp lỗi trong quá trình thu âm
    recognition.onerror = function (event) {
      console.error('Lỗi thu âm: ' + event.error);
      statusElement.textContent = 'Lỗi thu âm: ' + event.error;

      // Xóa hiệu ứng khi gặp lỗi
      recordButton.classList.remove('recording');
      if (spinner) recordButton.removeChild(spinner);
    };

    // Bắt đầu quá trình thu âm
    recognition.start();
  }
}
```

Hình 2. 12: Hàm thu âm giọng nói người dùng

```
let chart;
async function fetchSensorData() {
  const response = await fetch(
    'https://api.thingspeak.com/channels/2681700/feeds.json?api_key=AQNH2YUMJGLPHMNO&results=10',
  );
  const data = await response.json();
  return data.feeds.map((feed) => ({
    time: feed.created_at,
    value: feed.field1,
  }));
}
```

Hình 2. 13: Hàm lấy dữ liệu khoảng cách từ ThinkSpeak

```
// Hàm để vẽ hoặc cập nhật biểu đồ
async function drawChart() {
  const sensorData = await fetchSensorData();
  const labels = sensorData.map((data) => {
    const date = new Date(data.time);
    return date.getHours() + ':' + date.getMinutes(); // Hiển thị giờ:phút
  });
  const values = sensorData.map((data) => parseFloat(data.value));
  const alertBox = document.getElementById('alert-box');
  const ctx = document.getElementById('obstacleChart').getContext('2d');

  // Nếu biểu đồ đã được tạo, cập nhật dữ liệu
  if (chart) {
    chart.data.labels = labels;
    chart.data.datasets[0].data = values;
    chart.update();
  } else {
    // Tạo biểu đồ nếu chưa tồn tại
    chart = new Chart(ctx, {
      type: 'line',
      data: {···
      options: {···
    });
  }
}
```

Hình 2. 14: Hàm vẽ biểu đồ

b.Backend

```
model = joblib.load('./models/action_classification.joblib')
vectorizer = joblib.load('./models/text_vectorizer.joblib')
#du doan hanh dong
def classify_command(text_command):
    command_vector = vectorizer.transform([text_command])
    probabilities = model.predict_proba(command_vector)
    predicted_label_index = probabilities.argmax()
    predicted_label = model.classes_[predicted_label_index]
    max_probability = probabilities[0][predicted_label_index]
    print(max_probability)
    if max_probability < 0.3:
        return "S"
    else:
        return predicted_label
```

Hình 2. 15: Hàm dự đoán lệnh thông qua học máy

```
@app.route('/controll', methods=['POST'])
def control_car():
    data = request.get_json()

    if not data or 'speech' not in data:
        return jsonify({"error": "No speech command received"}), 400

    speech_command = data['speech']
    speech_command = speech_command.rstrip('.')

    print(f"Received speech command: {classify_command(speech_command)}")
    send_brl(classify_command(speech_command))
    return jsonify({"speech": classify_command(speech_command)})
```

Hình 2. 16: Hàm nhận dữ liệu từ FE gửi về và gửi đến ESP32

```python
import serial
import time

# Hàm gửi dữ liệu tới ESP32
def send_brl(data):
    try:
        # Mở kết nối Bluetooth
        BT = serial.Serial('COM7', 115200, timeout=1)
        print("Connected to COM7")

        # Gửi dữ liệu tới ESP32
        BT.write(data.encode('utf-8'))
        print(f"Sent: {data}")
        time.sleep(0.5)

        # Kiểm tra và nhận phản hồi từ ESP32 nếu có
        if BT.in_waiting > 0:  # Kiểm tra xem có dữ liệu nào trong hàng đợi không
            response = BT.readline().decode('utf-8').strip()
            print(f"Received response from ESP32: {response}")
            return response  # Trả về phản hồi cho người gọi hàm

    except serial.SerialException as e:
        print(f"Error sending data: {e}")
        return f"Error: {e}"
    finally:
        # Đóng kết nối sau khi gửi dữ liệu
        BT.close()
        print("Bluetooth connection closed")
```

Hình 2. 17: Hàm gửi dữ liệu tới ESP32

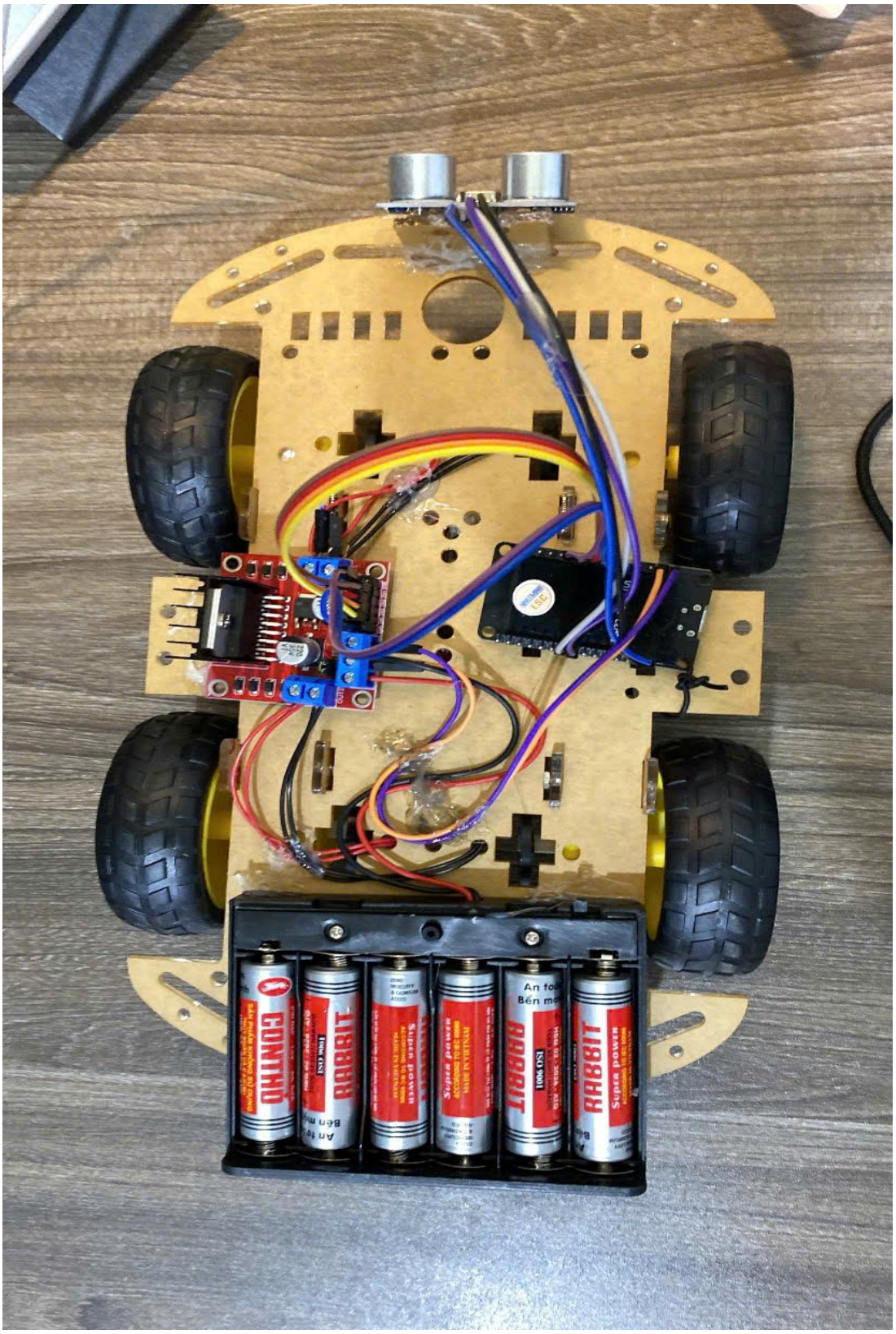
c. Model học máy

```python
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
import joblib

df = pd.read_csv('./data/labels.csv')

text_data = df['label']
labels = df['value']


vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(text_data)

model = MultinomialNB()
model.fit(X_train_vectorized, labels)

joblib.dump(model, './models/action_classification.joblib')
joblib.dump(vectorizer, './models/text_vectorizer.joblib')
```

Hình 2. 18: Model học máy

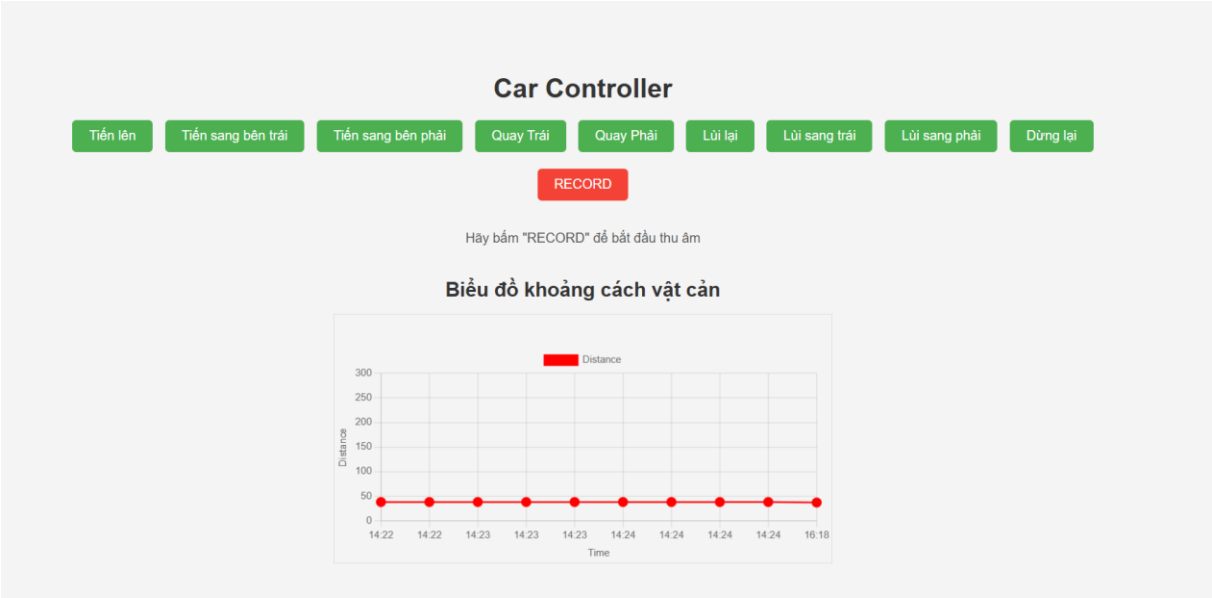
Chương 3: Kết quả thực nghiệm

3.1. Mô hình ô tô



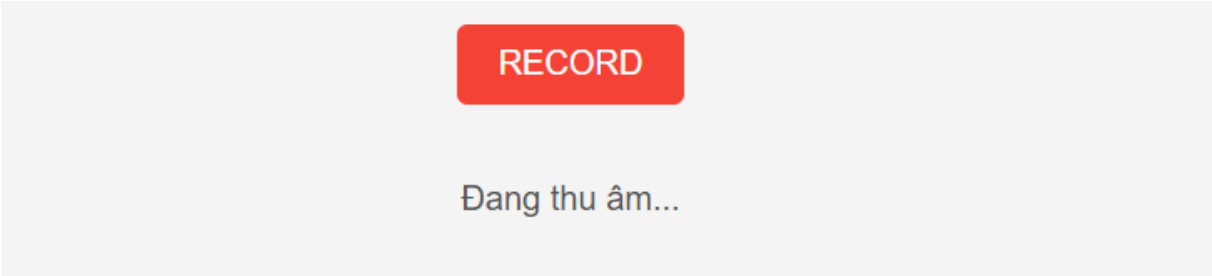
Hình 3. 1: Mô hình ô tô

3.2. Giao diện khi gửi lệnh từ trang web



Hình 3. 2: Giao diện gửi lệnh

3.3. Giao diện khi thu âm



Hình 3. 3: Giao diện khi thu âm

3.4. Giao diện biểu đồ khoảng cách

Hình 3. 4: Giao diện biểu đồ khoảng cách

Kết luận

Dự án đã thành công trong việc kết hợp các công nghệ tiên tiến vào một mô hình ô tô đồ chơi thông minh, đáp ứng được các mục tiêu đề ra ban đầu. Mô hình đã đạt được khả năng điều khiển từ xa qua Bluetooth, nhận lệnh bằng giọng nói, phát hiện vật trên đường di chuyển.

Trong suốt quá trình thực hiện, chúng em đã có cơ hội áp dụng các kiến thức về lập trình vi điều khiển, tích hợp cảm biến, và xử lý tín hiệu không dây Bluetooth. Mặc dù đã đạt được những thành quả nhất định, tuy nhiên dự án vẫn còn tồn tại một số khuyết điểm cần được cải thiện, ví dụ như độ nhạy và phạm vi của cảm biến còn hạn chế, kết nối Bluetooth chưa ổn định, và khả năng nhận diện giọng nói và dự đoán lệnh chưa cao trong điều kiện ồn ào.

Tuy vậy, mô hình này cũng đã mở ra ra cho chúng em nhiều hướng phát triển tiềm năng trong tương lai. Có thể cải thiện khả năng nhận diện giọng nói bằng cách sử dụng AI, nâng cấp cảm biến để tăng độ chính xác và khả năng phát hiện vật cản, cũng như phát triển thêm các chế độ điều khiển tự động. Ngoài ra còn có thể tích hợp GPS, Wi-Fi, và xây dựng một ứng dụng di động để điều khiển, giúp tăng tính ứng dụng của mô hình và mang lại trải nghiệm điều khiển tốt hơn cho người dùng.

Dự án này không chỉ giúp chúng em cải thiện kỹ năng về kỹ thuật, mà còn đóng vai trò như một nền tảng học tập và nghiên cứu hữu ích cho các ứng dụng thực tế về hệ thống điều khiển từ xa và công nghệ IoT. Mô hình ô tô điều khiển từ xa tích hợp điều khiển giọng nói và cảm biến vật cản là một bước đi ban đầu quan trọng, tạo tiền đề cho các dự án phát triển xe tự hành và các hệ thống tự động hóa phức tạp hơn trong tương lai.

Nhìn chung, dự án đã mang lại những kiến thức và kinh nghiệm quý báu, giúp chúng em hiểu rõ hơn về tiềm năng và ứng dụng của công nghệ hiện đại trong đời sống. Với những định hướng và cải tiến hợp lý, mô hình ô tô đồ chơi này có thể trở thành một nền tảng học tập và nghiên cứu vững chắc, không chỉ trong môi trường học tập mà còn có thể mở rộng trong các lĩnh vực thực tiễn khác.

Tài liệu tham khảo

[1] Sibigtroth, J. (2017). *Getting Started with Bluetooth on the ESP32*.

[2] Morón, R., Zamora, A., & García, C. (2020). *Design and Implementation of a Bluetooth Communication System for IoT Devices Using ESP32*. Journal of Embedded Systems, 12(4), 205-217.

[3] Espressif Systems. (2021). *ESP32 Series Datasheet*.

[4] Ultrasonic Sensor HC-SR04. (2021). *Datasheet and User Manual*.

[5] STMicroelectronics. (2016). *L298N Dual Full-Bridge Driver - Datasheet*.

[6] Budiman, M. (2018). *Design and Development of Mobile Robot Using L298N Motor Driver and Arduino Platform*. International Journal of Engineering and Technology, 7(3), 32-38.