HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA CÔNG NGHỆ THÔNG TIN

CÔNG NGHỆ PHẦM MỀM NHÚNG
XE TỰ HÀNH ĐIỀU KHIỂN BẰNG BLUETOOTH

Giảng viên hướng dẫn: **Ths. Lê Đức Thuận**

Sinh viên thực hiện: *Trần Gia Lương* *CT030433*

Phạm Trà My *CT030435*

Phan Hoàng Sơn *CT030442*

Hà Nội, 2021

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

CHƯƠNG 1. GIỚI THIỆU

1.1. Mục đích tài liệu

Trong thời đại công nghiệp hóa hiện đại hóa đất nước, việc chú trọng phát triển các ngành công nghiệp cũng như các ngành dịch vụ đang dần dần được coi trọng. Và như một nhu cầu thiết yếu, việc phát triển ra các hệ thống nhúng để đưa vào hỗ trợ trong công nghiệp, dịch vụ sẽ đem lại khoản lợi nhuận khổng lồ khi mà với cùng bằng một sức lao động người sử dụng các hệ thống nhúng tạo ra được nhiều giá trị sức lao động hơn.

Tài liệu này được tạo ra với mục đích phân tích, thiết kế ra một hệ thống nhúng có tên là *Xe tự hành điều khiển bằng bluetooth* . Xe này sẽ làm tiền đề để có thể phát triển thành các sản phẩm sử dụng trong công nghiệp, ví dụ như xe chuyển hàng giữa các kho bãi với nhau ở trong một công xưởng. Có thể thấy hiện nay, việc luân chuyển hàng hóa giữa các khu vực sản xuất trong một công xưởng vẫn cần một nhân công điều khiển xe, khi ta áp dụng công nghệ xe tự hành điều khiển bằng bluetooth thì có thể giảm được chi phí nhân công điều khiển xe, hơn nữa là một người có thể điều khiển nhiều xe làm nhiệm vụ, từ đó gia tăng lợi nhuận cho công ty sản xuất.

1.2. Phạm vi tài liệu

Tài liệu này được giới hạn trong phạm vi môn học công nghệ phầm mềm, cũng như công nghệ phần mềm nhúng. Tài liệu sử dụng bao gồm các ca sử dụng, các đặc tả ca sử dụng, biểu đồ hoạt động, … , các tài liệu, mô tả liên quan đến các thiết bị nhúng như bảng mạch Arduino, dộng cơ, các hình mô tả cách nối mạch và các chương trình nhúng, … .

1.3. Các thiết bị được sử dụng

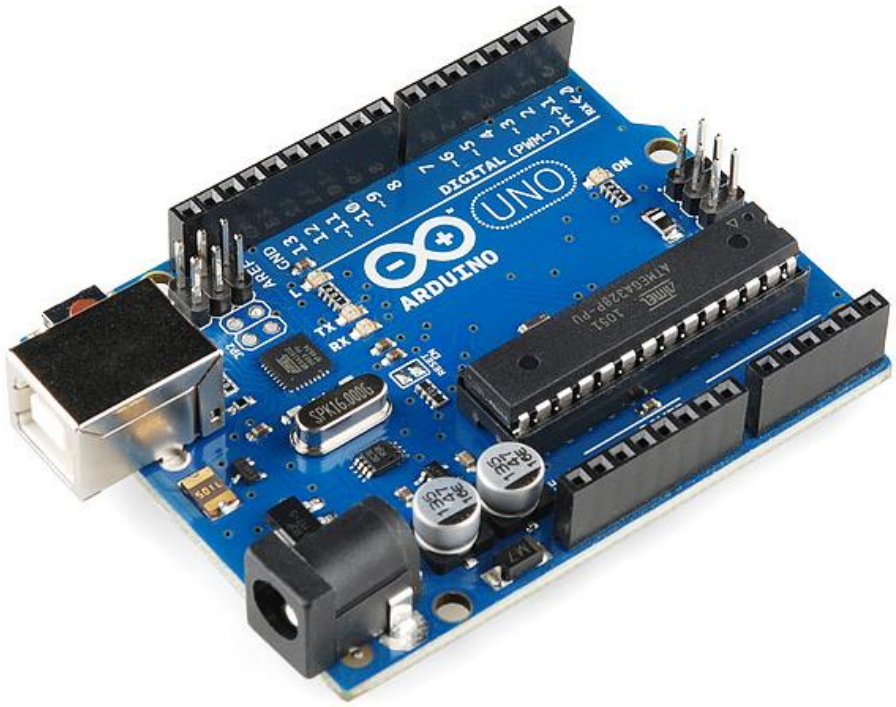
1.3.1. Arduino UNO R3

Arduino UNO R3 là một bảng mạch vi điều khiển mã nguồn mở, sử dụng chip ATmega328P và được phát triển bởi Arduino.cc. Bảng mạch được thiết kế với tập các chân vào/ra theo hai dạng là Digital hoặc Analog, những chân này giúp bảng mạch Arduino có thể giao tiếp với các thiết bị ngoại vi, cũng như là với các bảng mạch khác. Arduino UNO R3 có các thông số kỹ thuật như sau:

Loại thông số	**Giá trị**
Chip Điều khiển	Atmega328P
Điện áp hoạt động	5V
Điện áp đầu vào (khuyên dùng)	7-12V
Điện áp đầu vào (giới hạn)	6-20V

Số chân Digital	14 (trong đó có 6 chân PWN)
Số chân Analog	6
Dòng điện DC trên mỗi chân I/O	20mA
Dòng điện DC trên chân 3.3V	50mA
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Tốc độ xung	16 MHZ
LED_BUILIN	13
Chiều dài	68.6 mm
Chiều rộng	53.4 mm
Cân nặng	25 g

Bảng 1.1: Thông số Arduino UNO R3



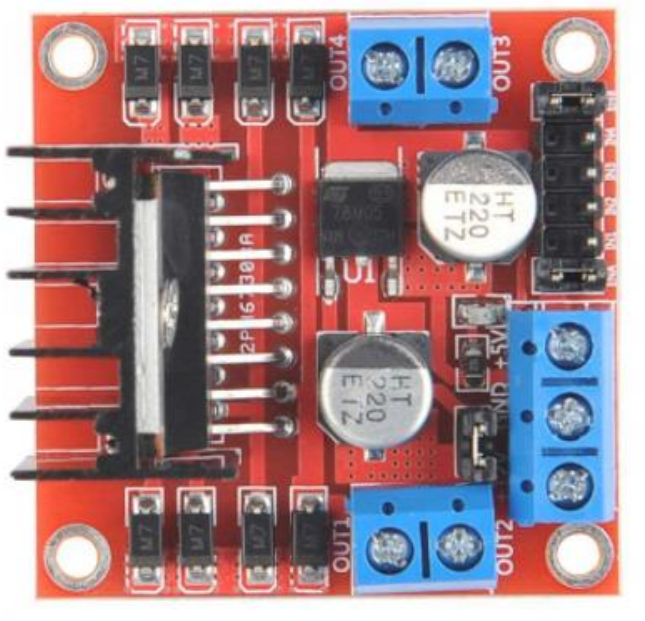
Hình 1.1: Bảng mạch Arduino UNO R3

1.3.2. Motor Driver L298N

Motor Driver L298N là một module dùng để điều khiển động cơ dùng ở trong arduino, nó có thể điều khiển tối đa lên đến bốn động cơ riêng biệt, hoặc hai động cơ riêng biệt theo hai hướng quay khác nhau.
Motor Driver L298N có các chân như sau:

Chân	Chức năng
IN1 & IN2	Chân đầu vào cho motor 1
IN3 & IN4	Chân đầu vào cho motor 2
ENA	Cho phép các chân liên quan đến motor 1 hoạt động
ENB	Cho phép các chân liên quan đến motor 2 hoạt động
OUT1 & OUT2	Chân đầu ra của motor 1
OUT3 & OUT4	Chân đầu ra của motor 2
12V	Chân nguồn 12V
5V	Chân nguồn hỗ trợ mạch bên trong L298N
GND	Chân nối đất

Bảng 1.2: Chi tiết chân L298N



Hình 1.2: Module Motor Driver L298N

1.3.3. DC Motor

DC Motor là động cơ chính giúp xe chuyển động. DC Motor thường có hai đầu dẫn, một cho cực dương, một cho cực âm. Để giúp Motor quay, chúng ta nối hai dầu dẫn vào PIN. Nếu đổi hai đầu dẫn với nhau, Motor xẽ quay theo chiều ngược lại.

Hình 1.3: Động cơ DC Motor

1.3.4. Module HC-05

Module HC-05 là một module bluetooth sử dụng giao thức Serial Port, nó được thiết kế để thiết lập kết nối nối tiếp không dây. Module này dựa trên IC Bluetooth chip đơn BC417, tuân thủ tiêu chuẩn Bluetooth v2.0 và hỗ trợ cho cả giao diện UART và USB.

Đặc điểm kỹ thuật:

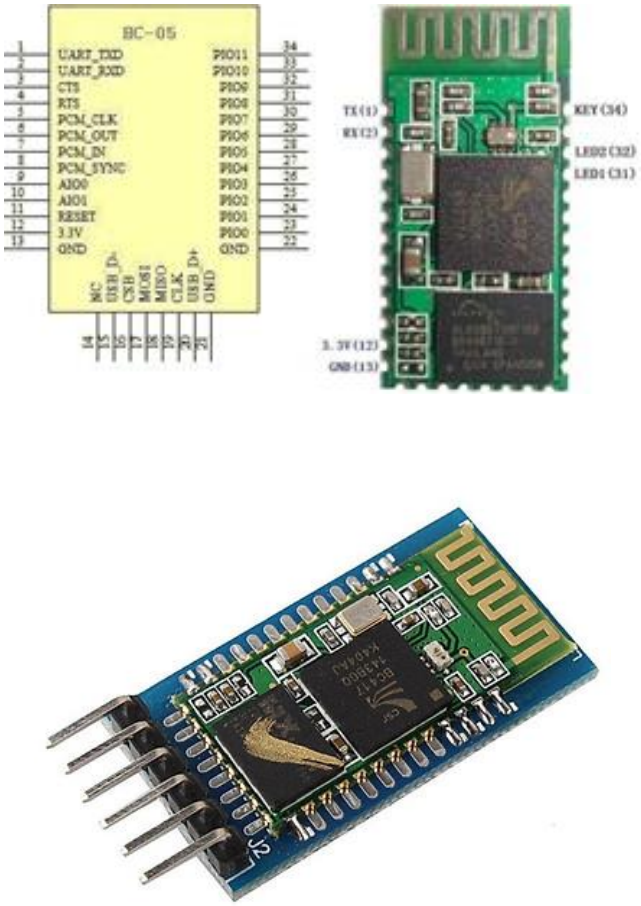
- Chuẩn Bluetooth: V2.0+EDR.
- Điện áp hoạt động: 3.3-4VDC, 30mA
- Kích thước: 28mm x 15mm x 2.35mm
- Tần số: 2.4G
- Tốc độ: 2.1Mbs(Max) / 160kbps
- Tốc độ baud mặc định: 9600, 8bit dữ liệu, 1bit Stop. Hỗ trợ tốc độ baud: 9600, 19200, 38400, 57600, 115200, 230400, 460800.
- Nhiệt độ làm việc: -20 ~ 75ºC
- Độ nhạy: -80dBm
- Module có hai chế độ làm việc

Chi tiết các chân của Module HC-05

Chân	**Chức năng**
EN	Cho phép Module hoạt động khi chân này được bỏ trống hoặc được kết nối với chân 3.3V
+5V	Chân cấp nguồn
GND	Chân nối đất
TX	Chân Output của giao tiếp UART
RX	Chân Input của giao tiếp UART

STATE	Chân báo trạng thái. Pin ở mức thấp khi Module không được kết nối với thiết bị nào, ngược lại chân ở mức cao khi có thiết bị kết nối

Bảng 1.3: Chi tiết các chân của Module HC-05



Hình 1.4: Module HC-05

1.3.5. Module HC-SR04

Cảm biến khoảng cách HC-SR04 là cảm biến dùng để xác định khoảng cách trong phạm vi nhỏ bằng cách phát sóng siêu âm. Cảm biến có độ chính xác khá cao và độ ổn định trong quá trình sử dụng, đồng thời dễ dàng kết nối với các bảng mạch như là Arduino.

Chi tiết chân của Module HC-SR04:

Chân	Chi tiết
Vcc	Chân nguồn 5V
Trig	Chân nhận tín hiệu phát sóng

Echo	Chân xuất tín hiệu xung HIGH, chiều rộng của xung bằng với thời gian sóng siêu âm được phát từ cảm biến và quay về

Bảng 1.4: Chi tiết chân Module HC-SR04

Hình 1.5: Module HC-SR04

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Tìm hiểu về Bluetooth

Bluetooth là một thuật ngữ công nghệ dùng để chỉ một phương thức kết nối và truyền tải dữ liệu không dây tầm gần giữa các thiết bị điện tử. Công nghệ này hỗ trợ việc truyền dữ liệu qua các khoảng cách ngắn giữa các thiết bị di động và cố định, tạo nên các mạng cá nhân không dây.

Các chuẩn kết nối Bluetooth:

Công nghệ Bluetooth sử dụng sóng Radio với tần số 2.4GHz. Tuy sử dụng cùng tần số với công nghệ Wifi nhưng chúng không hề xung đột với nhau vì Bluetooth có bước sóng ngắn hơn. Bluetooth là một chuẩn điện tử, nó bắt buộc các hãng sản xuất các thiết bị phải tuân thủ các tiêu chuẩn kỹ thuật chung nhằm tạo sự tương thích tối ưu giữ các thiết bị với nhau. Các chuẩn kết nối Bluetooth từ trước đến nay bao gồm:

- **Bluetooth 1.0**: Đạt tốc độ xấp xỉ 1Mbps nhưng gặp nhiều vấn đề về tính tương thích.
- **Bluetooth 1.1**: Đây là bản sửa lỗi tương thích cho phiên bản 1.0.
- **Bluetooth 1.2**: Giảm thời gian dò tìm và tăng tốc độ kết nối so với phiên bản 1.1.
- **Bluetooth 2.0 +ERD**: Được công bố vào tháng 7/2007, phiên bản ổn định hơn và cho tốc độ chia sẻ nhanh hơn, đồng thời giảm mức độ tiêu thụ năng lượng.
- **Bluetooth 2.1 +ERD**: Có thêm cơ chế kết nối phạm vi nhỏ.
- **Bluetooth 3.0 + HS (High Speed)**: Được giới thiệu vào 21/4/2009, chuẩn kết nối này có tốc độ lý thuyết lên đến 24Mbps. Những thiết bị hỗ trợ Bluetooth 3.0 nhưng không có +HS sẽ không đạt được tốc độ trên. Tuy tốc độ cao nhưng Bluetooth vẫn chỉ hỗ trợ nhu cầu chia sẻ nhanh file có dung lượng thấp hay kết nối với loa, tai nghe…
- **Bluetooth 4.0**: Được cho ra mắt vào ngày 30/6/2010, Bluetooth 4.0 là sự kết hợp của "classis Bluetooth" (Bluetooth 2.1 và 3.0), " Bluetooth high speed" (Bluetooth 3.0+ HS) và " Bluetooth low energy" (Bluetooth smart ready / Bluetooth smart). Bluetooth 4.0 với cách thức hoạt động hoàn toàn mới để những kết nối đơn giản được thực hiện nhanh chóng, chuẩn mới còn hỗ trợ truyền tải nhanh hơn, tiết kiệm pin hơn.
- **Bluetooth 4.2**: Ra mắt chính thức vào tháng 12/2014, phiên bản này có tốc độ truyền dữ liệu đã được nâng lên gấp 2,5 lần, cùng với đó là việc tiêu thụ điện năng ít hơn, bảo mật cao hơn và đặc biệt là có thể kết nối trực tiếp Internet thông qua chuẩn IPv6.
- **Bluetooth 5.0**: Là công nghệ Bluetooth phổ thông hiện đại nhất hiện nay, nó giảm thiểu mức độ tiêu thụ điện năng tối đa gấp 2,5 lần so với bluetooth 4.2. Ngoài ra, công nghệ này còn thể hiện độ phủ sóng kết nối ở phạm vi gấp bốn lần so với

Bluetooth 4.0 trước kia - tương đương khoảng 300 mét trong thực tế. Hầu hết các dòng smartphone hiện đại nhất hiện nay như iPhone 11 Pro Max hay Samsung Galaxy Note 20 Ultra đều đang được tích hợp Bluetooth 5.0.

- **Bluetooth 5.1**: Đây là phiên bản mới nhất của công nghệ Bluetooth hiện nay, cho phép bạn trao đổi dữ liệu giữa các thiết bị cách nhau một khoảng ngắn. So với phiên bản cũ, Bluetooth 5.1 được trang bị thêm tính năng Randomized Advertising Channel Indexing, cho phép thiết bị Bluetooth tự thông báo khi nó sẵn sàng kết nối. Ngoài ra không giống như Bluetooth 5.0 yêu cầu các thiết bị phải đi cùng với một dữ liệu gói cụ thể, Bluetooth 5.1 cho phép các thiết bị chọn các kênh một cách ngẫu nhiên. Tính năng mới này rất hữu ích, khi bạn ở khu vực có nhiều thiết bị thông báo rằng chúng đã sẵn sàng kết nối. Tuy vậy Bluetooth 5.1 vẫn cần thêm một thời gian phát triển nữa thì mới có thể trở nên đại trà như Bluetooth 5.0 hiện nay

Hoạt động Bluetooth là chuẩn kết nối không dây tầm ngắn, thiết kế cho các kết nối thiết bị cá nhân hay mạng cục bộ nhỏ trong phạm vi băng tần từ 2.4GHz đến 2.485GHz. Bluetooth được thiết kế hoạt động trên 79 tần số đơn lẻ. Khi kết nối , nó sẽ tự động tìm ra tần số tương thích để di chuyển đến thiết bị cần kết nối trong khu vực nhằm đảm bảo sự liên tục

Hình 2.1: Ứng dụng của bluetooth

Các ưu điểm của bluetooth:

- Thay thế hoàn toàn dây nối.
- Hoàn toàn không nguy hại đến sức khoẻ con người.
- Bảo mật an toàn với công nghệ mã hóa trong. Một khi kết nối được thiết lập thì khó có một thiết bị nào có thể nghe trộm hoặc lấy cắp dữ liệu .
- Các thiết bị có thể kết nối với nhau trong vòng 20m mà không cần trực diện (hiện nay có loại Bluetooth kết nối lên đến 100m).
- Kết nối điện thoại và tai nghe Bluetooth khiến cho việc nghe máy khi lái xe hoặc bận việc dễ dàng.
- Giá thành rẻ.
- Tốn ít năng lượng, chờ tốn 0.3mAh, tối đa 30mAh trong chế độ truyền dữ liệu.

- Không gây nhiễu các thiết bị không dây khác.
- Tính tương thích cao nên được nhiều nhà sản xuất phần cứng và phần mềm hỗ trợ.

Nhược điểm của bluetooth:

- Tốc độ thấp, khoảng 720kbps tối đa.
- Bắt sóng kém khi có vật cản.
- Thời gian thiết lập lâu

Một số yếu tố ảnh hưởng đến phạm vi thiết bị Bluetooth:

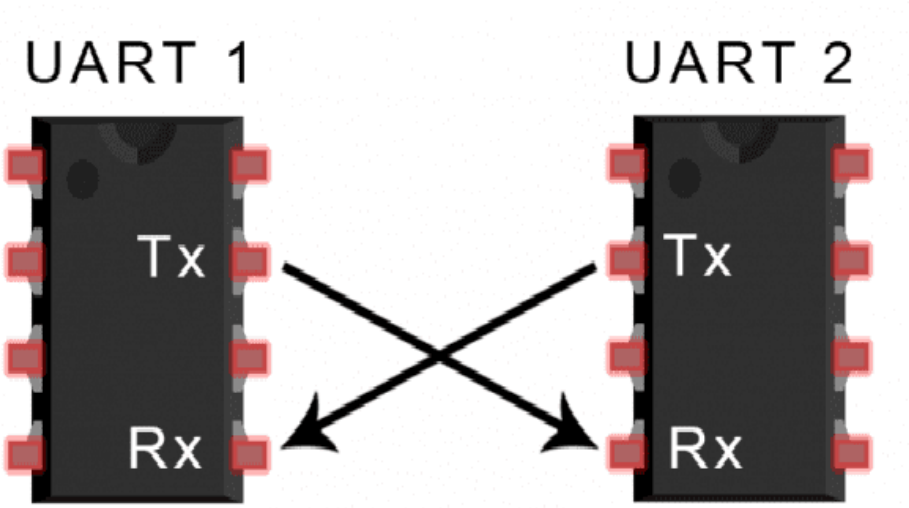
- **Phổ vô tuyến** : Dải tần của công nghệ Bluetooth làm cho nó trở thành một lựa chọn tốt cho giao tiếp không dây.
- **Lớp vật lý (PHY)** : Lớp này xác định một số khía cạnh chính về cách radio được sử dụng để truyền và nhận dữ liệu như tốc độ dữ liệu, cách phát hiện và sửa lỗi được thực hiện, bảo vệ chống nhiễu và các kỹ thuật khác ảnh hưởng đến độ rõ ràng của tín hiệu trên các phạm vi khác nhau.
- **Độ nhạy của máy thu** : Phép đo cường độ tín hiệu tối thiểu mà máy thu vẫn có thể nhận và giải mã dữ liệu một cách chính xác.
- **Công suất truyền** : Như bạn có thể mong đợi, cường độ tín hiệu truyền càng cao, phạm vi có thể đạt được càng dài. Nhưng việc tăng công suất truyền tải cũng sẽ làm cạn kiệt pin của bạn nhanh hơn.
- **Độ lợi của ăng-ten** : Về cơ bản, đây là thay đổi tín hiệu điện từ máy phát thành sóng vô tuyến và trở lại một lần nữa ở đầu nhận.
- **Mất đường truyền** : Một số yếu tố có thể làm suy yếu tín hiệu, bao gồm khoảng cách, độ ẩm và môi trường mà tín hiệu truyền qua (chẳng hạn như gỗ, bê tông hoặc kim loại).

Các giải pháp an toàn bảo mật trong Bluetooth:

- Chỉ mở Bluetooth khi cần thiết.
- Giữ thiết bị ở chế độ ‘hidden’.
- Kiểm tra định kỳ danh sách các thiết bị đã paired.
- Nên mã hóa khi thiết lập Bluetooth với máy tính.
- Sử dụng các phần mềm diệt virus, quét virus định kỳ.

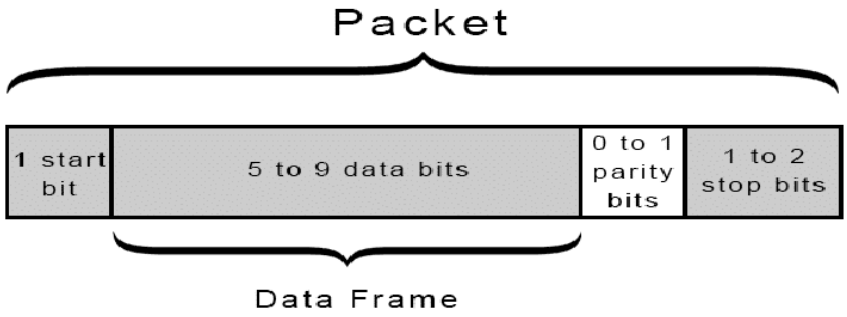
2.2. Giao thức UART

UART là viết tắt của Universal Asynchronous Receiver – Transmitter có nghĩa là truyền dữ liệu nối tiếp bất đồng bộ. Truyền dữ liệu nối tiếp bất đồng bộ có 1 đường phát dữ liệu và 1 đường nhận dữ liệu, không có tín hiệu xung clock nên gọi là bất đồng bộ.



Hình 2.2: Cách mắc nối hai thiết bị sử dụng UART

Dữ liệu truyền qua UART được tập hợp thành gói (packet). Mỗi gói chứa 1 bit start, 5 đến 9 bit dữ liệu (tùy thuộc vào UART), một bit chẵn lẻ (parity bit) tùy chọn và 1 hoặc 2 bit stop.



Hình 2.3: Chi tiết khung dữ liệu giao thức UART

- Start Bit: Là bit bắt đầu trong khung truyền Bit này nhằm mục đích báo cho thiết bị nhận biết quá trình truyền bắt đầu.
- Data: Dữ liệu cần truyền Data không nhất thiết phải 8 bit. Trong UART bit LSB được truyền đi trước, Bit MSB được truyền đi sau.
- Parity bits: Sau khi UART nhận đọc khung dữ liệu, nó sẽ đếm số bit có giá trị là 1 và kiểm tra xem tổng số là số chẵn hay lẻ. Nếu bit chẵn lẻ là 0 (tính chẵn), thì tổng các bit 1 trong khung dữ liệu phải là một số chẵn. Nếu bit chẵn lẻ là 1 (tính lẻ), các bit 1 trong khung dữ liệu sẽ tổng thành một số lẻ. Khi bit chẵn lẻ khớp với dữ liệu, UART sẽ biết rằng quá trình truyền không có lỗi. Nhưng nếu bit chẵn lẻ là 0 và tổng là số lẻ; hoặc bit chẵn lẻ là 1 và tổng số là chẵn, UART sẽ biết rằng các bit trong khung dữ liệu đã thay đổi.
- Stop Bit: là bit báo cáo kết thúc khung truyền, có thể có 1 hoặc 2 bit stop.

Các bước truyền :

- UART truyền nhận dữ liệu song song từ bus dữ liệu.

- UART truyền thêm bit start, bit chẵn lẻ và bit dừng vào khung dữ liệu.
- Toàn bộ gói được gửi nối tiếp từ UART truyền đến UART nhận. UART nhận lấy mẫu đường dữ liệu ở tốc độ truyền được định cấu hình trước.
- UART nhận loại bỏ bit start, bit chẵn lẻ và bit stop khỏi khung dữ liệu.
- UART nhận chuyển đổi dữ liệu nối tiếp trở lại thành song song và chuyển nó đến bus dữ liệu ở đầu nhận.

2.3. Hệ điều hành nhúng thời gian thực FreeRTOS

FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi RichardBarry. FreeRTOS được thiết kế đủ nhỏ để chạy trên một vi xử lý, do FreeRTOS chỉ cung cấp chức năng lập lịch thời gian thực lõi, truyền thông liên tác vụ, định thời và đồng bộ.

Một điểm mạnh nữa trong hệ điều hành nhúng thời gian thực FreeRTOS là nó cho phép lập lịch đa tác vụ. Đa tác vụ trong hệ thống nhúng thời gian thực là khái niệm giống với đa tác vụ trong các hệ thống để bàn ở chỗ nó mô tả nhiều luồng thực thi bằng cách sử dụng cùng một bộ xử lý đơn. Tuy nhiên, mục tiêu của hệ thống nhúng thời gian thực lại khá khác với hệ thống để bàn đó, đặc biệt khi hệ thống nhúng kỳ vọng cung cấp hành vi "thời gian thực cứng" (hard real time).

Nhân (kernel) của hệ điều hành giao cho CPU thực hiện việc xử lý các task theo những khoảng thời gian. Nó cũng liên tục kiểm tra mức ưu tiên của các task, xắp xếp các thông điệp từ task và tiến hành lập lịch. Trong một hệ thống chạy RTOS, cũng có các tài nguyên dùng chung cùng với phần được cấp phát riêng cho mỗi task.

Kernel là trái tim của mỗi hệ điều hành. Nó thực hiện chức năng quản lý và lập lịch các quá trình sử dụng CPU và điều phối tài nguyên. Mỗi task chỉ được thực thi bởi CPU trong một khoảng thời gian, trong các khoảng thời gian còn lại thì task được quản lý bởi các dịch vụ quản lý của hệ điều hành.

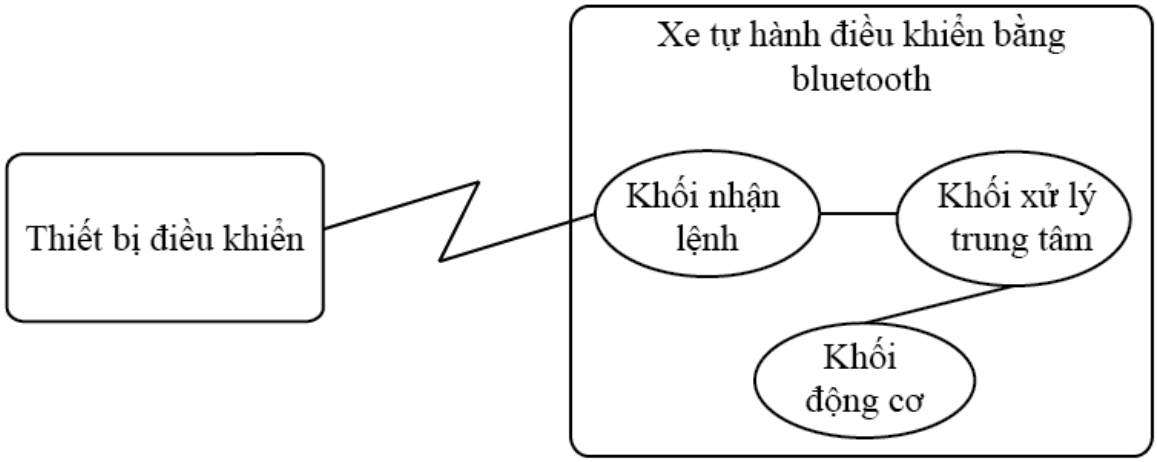
Để khởi tạo các tác vụ ở trong FreeRTOS, ta dùng hàm `xTaskCreate()` và sau khi tạo ra các tác vụ, ta khởi động bộ lập lịch bằng hàm `vTaskStartScheduler()`.

Để truyền thông liên tác vụ, hệ điều hành FreeRTOS cung cấp các hàng đợi cùng các hàm API để thao tác với hàng đợi. Để tạo ra một hàng đợi, ta sử dụng hàm `xQueueCreate()`, hàm này trả về một biến có kiểu là `xQueueHandler` và biến này sẽ được sử dụng để tham chiếu tới hàng đợi vừa được tạo. Và để đẩy thông tin vào hàng đợi, ta dùng hàm `xQueueSend()`. Để lấy dữ liệu ra khỏi hàng đợi, ta dùng hàm `xQueueReceive()`. Để lấy dữ liệu ra khỏi hàng đợi mà không xóa dữ liệu đó ra khỏi hàng đợi, ta dùng hàm `xQueuePeek()`.

CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG

3.1. Phân tích kiến trúc

Kiến trúc đơn giản của *Xe tự hành điều khiển bằng bluetooth* được chia làm hai phần. Phần thứ nhất là thiết bị điều khiển, ở đây có thể dùng điện thoại di động có kết nối bluetooth làm thiết bị điều khiển. Phần thứ hai là xe được điều khiển, xe được điều khiển sẽ được chia tiếp thành 3 khối lần lượt là khối nhận lệnh, khối điều khiển và khối thực thi.



Hình 3.1: Kiến trúc đơn giản của hệ thống

Thiết bị điều khiển ở đây được chọn là điện thoại di động chạy hệ điều hành android, thiết bị sẽ được cài đặt một phầm mềm android có giao diện hỗ trợ các chức năng điều khiển xe. Sau đó thiết bị sẽ được kết nối và có thể điều kiển xe thông qua giao thức bluetooth.

Ở đầu bên kia là Xe tự hành điều khiển bằng bluetooth. Xe sẽ có một khối chuyên nhận tín hiệu điều khiển được gửi từ Thiết bị điều khiển sang, có rất nhiều Module hỗ trợ nhận tín hiệu bluetooth và nhóm đã lựa chọn sử dụng Module HC-05. Module này sẽ nhận tín hiệu và truyền tín hiệu sang Arduino bằng giao thức UART.

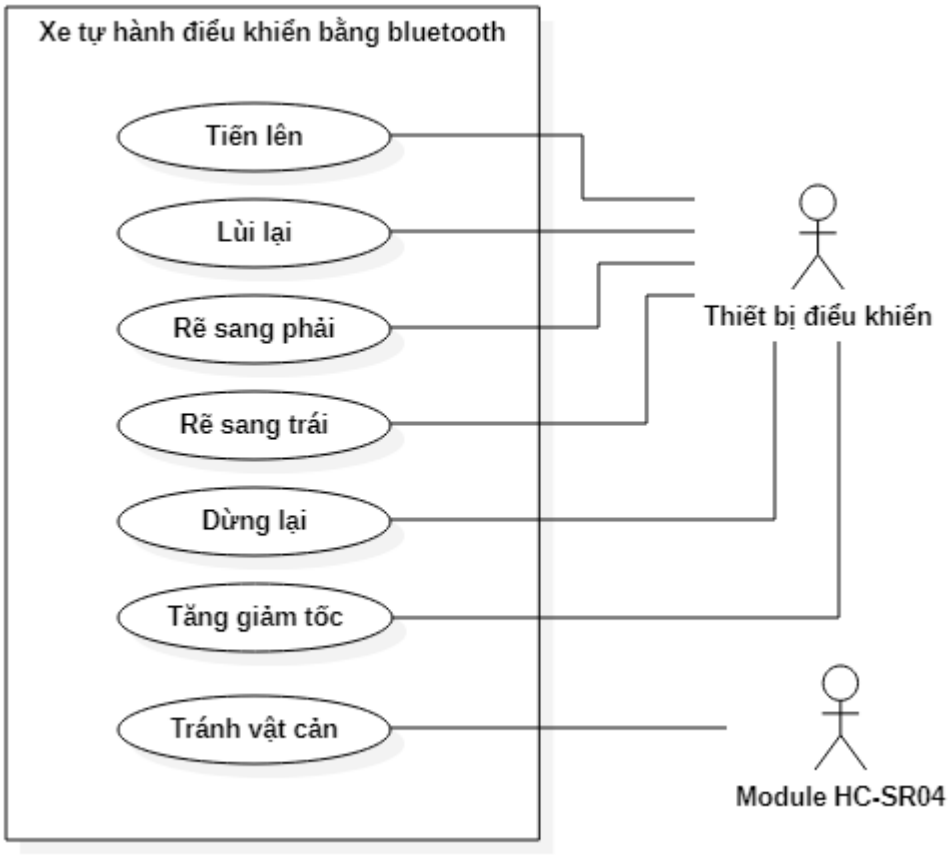
Khối điều khiển của xe tự hành điều khiển bằng bluetooth đóng vai trò như là một bộ não của xe, nó sẽ nhận mọi tín hiện từ ngoài ngửi vào và sẽ đi phân tích xử lý dữ liệu đấy. Kết quả của việc phân tích đó chính là đưa ra những quyết định điều kiển xe, ví dụ như là tiến lên hay lùi lại.

Cuối cùng ở đây là khối vận hành, khối vận hành bao gồm các động cơ và Module Motor Driver L298N. Do động cơ ở đây sử dụng hiệu điện thế và cường độ dòng điện lớn hơn nhiều so với khối điều khiển, vì vậy cần phải dùng một thiết bị điều khiển trung gian đó là Module Motor Driver L298N. Module này sẽ nhận tín hiệu đầu vào từ arduino, và sẽ điều khiển tốc độ nhanh, chậm của Motor thông qua cách điều

khiển cường độ dòng điện chạy qua động cơ bằng các Trasistor được thiết kế sẵn bên trong Module và vì vậy, Module Motor Driver L298N có thể điều khiển dòng điện lên tới 39V.

3.2. Phân tích xe tự hành điều khiển bằng bluetooth

3.2.1. Biểu đồ ca sử dụng của hệ thống



Hình 3.2: Biểu đồ ca sử dụng của hệ thống

Về mặt tổng quan, hệ thống xe tự hành điều khiển bằng bluetooth có bao gồm hai Actor đó chính là Thiết bị điều khiển và Module HC-SR04, kèm theo đó hệ thống có tổng cộng bảy ca sử dụng đó chính là Tiến lên, Lùi lại, Rẽ sang phải, Rẽ sang trái, Tăng giảm tốc độ, Dừng lại và tránh vật cản, đó là bảy chứng năng cơ bản để tạo thành một hệ thống xe tự hành điều khiển bằng bluethooth.

Đối với Actor là Thiết bị điều khiển, nó được phép thực hiện sáu chức năng ứng với bốn ca sử dụng đó là Tiến lên, Lùi lại, Rẽ sang phải, Rẽ sang trái, Dừng lại, Tăng giảm tốc độ. Với sáu ca sử dụng này, Thiết bị điều khiển có thể tùy ý điều khiển cách thức hoạt động của xe để đạt được mục đích ví dụ như là cho xe di chuyển từ A sang B.

Đối với Actor là Module HC-SR04, nó được phép thực hiện một chức năng ứng với ca sử dụng Tránh vật cản, ca sử dụng này được tạo ra với mục đích là để tránh tai nạn không đáng có xảy ra khi xe gặp chướng ngại vật, HC-SR04 sẽ truyền thông tin

khoảng cách của xe so với HC-SR04 về bộ xử lý trung tâm, từ đó bộ xử lý trung tâm sẽ đưa ra quyết định dừng xe và thông báo cho Thiết bị điều khiển.

3.2.2. Đặc tả các ca sử dụng

3.2.2.1. Ca sử dụng tiến lên

Use Case	Tiến lên
Actor	Thiết bị điều khiển
Brief Description	Điều khiển cho xe tiến lên
Pre-condition	Thiết bị đã được kết nối Bluetooth
Basic Flows	1. Người điều khiển gửi tín hiệu tiến lên ở trên thiết bị điều khiển. 2. Module HC-05 nhận thông tin điều khiển tiến lên và chuyển tín hiệu điều khiển cho arduino. 3. Arduino nhận tín hiệu điều khiển tiến và xuất tín hiệu khởi động động cơ hai bên ra các chân cần thiết, và tốc độ của 2 bên động cơ bằng nhau. 4. Động cơ được khởi động và xe bắt đầu tiến lên.
Alternative Flows	

Bảng 3.1 Ca sử dụng Tiến lên

2.2.2.2. Ca sử dụng lùi lại

Use Case	Lùi lại
Actor	Thiết bị điều khiển
Brief Description	Điều khiển cho xe lùi lại
Pre-condition	Thiết bị đã được kết nối Bluetooth
Basic Flows	1. Người điều khiển ra tín hiệu lùi lại ở trên thiết bị điều khiển. 2. Module HC-05 nhận thông tin điều lùi lại phải và chuyển tín hiệu điều khiển cho arduino. 3. Arduino nhận tín hiệu điều khiển lùi lại và xuất tín hiệu khởi động động cơ hai bên lùi lại. 4. Động cơ được khởi động và xe đi lùi lại.

Alternative Flows	

Bảng 3.2: Ca sử dụng lùi lại

2.2.2.3. Ca sử dụng rẽ sang phải

Use Case	Rẽ sang phải
Actor	Thiết bị điều khiển
Brief Description	Điều khiển cho xe rẽ sang phải
Pre-condition	Thiết bị đã được kết nối Bluetooth
Basic Flows	1. Người điều khiển ra tín hiệu rẽ sang phải ở trên thiết bị điều khiển. 2. Module HC-05 nhận thông tin điều khiển rẽ sang phải và chuyển tín hiệu điều khiển cho arduino. 3. Arduino nhận tín hiệu khoảng cách các vật thể bên phải 4. Nếu không có vật thể bên phải.Arduino nhận tín hiệu điều khiển rẽ sang phải và xuất tín hiệu khởi động động cơ bên phải chạy chậm hơn động cơ bên trái. 5. Động cơ được khởi động.
Alternative Flows	4.1. Nếu khoảng cách vật thể bên phải nhỏ hơn 25cm thì Arduino sẽ dừng động cơ.

Bảng 3.3: Ca sử dụng Rẽ sang phải

2.2.2.4. Ca sử dụng rẽ sang trái

Use Case	Rẽ sang trái
Actor	Thiết bị điều khiển
Brief Description	Điều khiển cho xe rẽ sang trái
Pre-condition	Thiết bị đã được kết nối Bluetooth
Basic Flows	1. Người điều khiển ra tín hiệu rẽ sang trái ở trên thiết bị điều khiển. 2. Module HC-05 nhận thông tin điều khiển rẽ sang trái và chuyển tín hiệu điều khiển cho arduino.

	3. Arduino nhận tín hiệu khoảng cách các vật thể bên trái 4. Nếu không có vật thể bên trái. Arduino nhận tín hiệu điều khiển rẽ sang trái và xuất tín hiệu khởi động động cơ bên trái chạy chậm hơn động cơ bên phải. 5. Động cơ được khởi động.
Alternative Flows	4.1. Nếu khoảng cách vật thể bên trái nhỏ hơn 25cm thì Arduino sẽ dừng động cơ.

Bảng 3.4: Ca sử dụng Rẽ sang trái

2.2.2.5. Ca sử dụng Dừng lại

Use Case	Dừng lại
Actor	Thiết bị điều khiển
Brief Description	Điều khiển cho xe rẽ sang trái
Pre-condition	Thiết bị đã được kết nối Bluetooth
Basic Flows	1. Người điều khiển ra tín hiệu dừng ở trên thiết bị điều khiển. 2. Module HC-05 nhận thông tin điều khiển dừng lại và chuyển tín hiệu điều khiển cho arduino. 3. Động cơ dừng lại.
Alternative Flows	

Bảng 3.5: Ca sử dụng lùi lại

2.2.2.6. Ca sử dụng tăng giảm tốc độ

Use Case	Tăng giảm tốc độ
Actor	Thiết bị điều khiển
Brief Description	Điều khiển cho xe tăng giảm tốc độ
Pre-condition	Thiết bị đã được kết nối Bluetooth
Basic Flows	1. Người điều khiển ra tín hiệu tăng giảm tốc độ ở trên thiết bị điều khiển.

	2. Module HC-05 nhận thông tin điều khiển tăng giảm tốc độ và chuyển tín hiệu điều khiển cho arduino. 3. Arduino gửi tín hiệu tốc độ đến cho L298N để điều khiển tốc độ Motor.
Alternative Flows	

Bảng 3.6: Ca sử dụng tăng giảm tốc độ

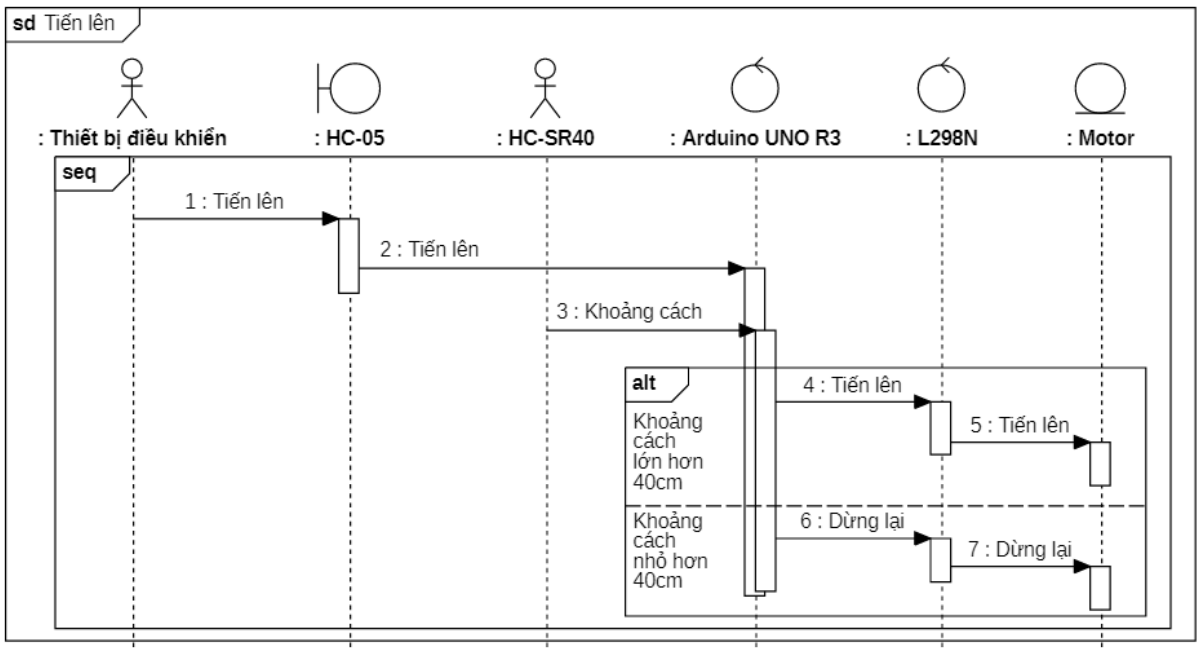
2.2.2.7. Ca sử dụng tránh vật cản

Use Case	Tránh vật cản
Actor	Module HC-SR04
Brief Description	Cảm biến đo khoản cách để tránh vật cản
Pre-condition	
Basic Flows	1. Cảm biến đo khoảng cách từ xe cho tới vật cản ở phía trước. 2. Cảm biến gửi thông tin khoảng cách vật cản cho arduino. 3. Nếu khoảng cách giữa xe và vật cản nhỏ hơn 5cm, xe sẽ không thể tiến lên phía trước, Thiết bị điều khiển cần điều khiển xe tránh sang hướng khác
Alternative Flows	3.1. Nếu khoảng cách giữa xe và vật cản lơn hơn 40cm, xe có thể tiến lên tiếp.

Bảng 3.7: Ca sử dụng tránh vật cản

2.2.3. Phân tích các ca sử dụng

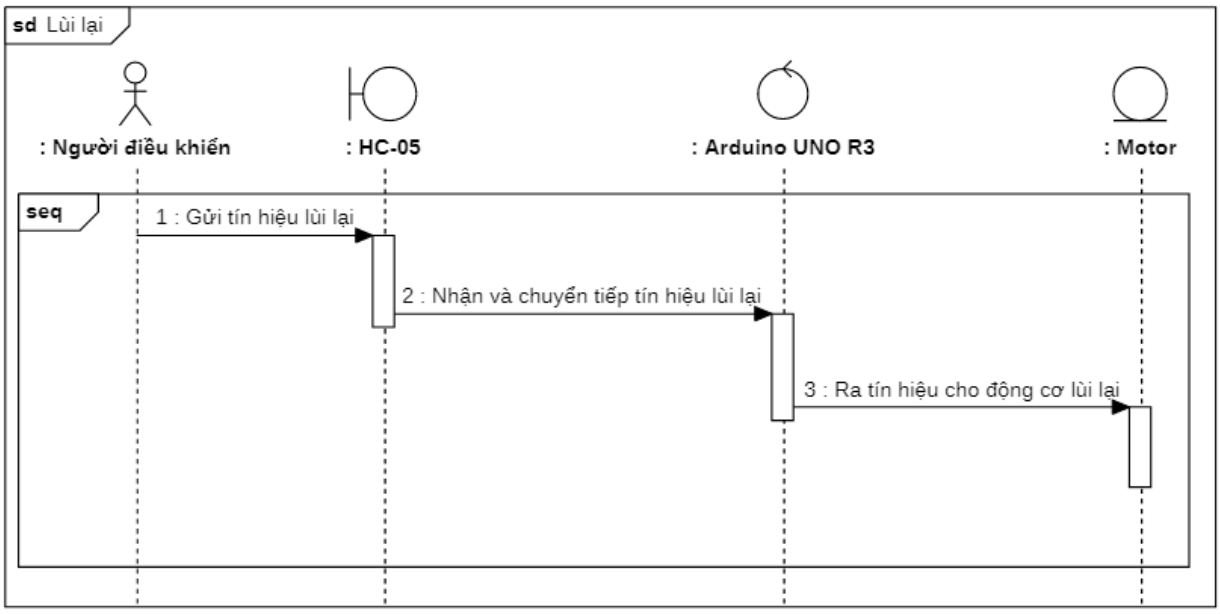
2.2.3.1. Ca sử dụng tiến lên



Hình 3.3: Ca sử dụng tiến lên

Thiết bị điều khiển ra tín hiệu tiến lên ở trên thiết bị điều khiển, Module HC-05 nhận thông tin điều khiển tiến lên và chuyển tín hiệu điều khiển cho arduino.Arduino nhận tín hiệu điều khiển tiến và xuất tín hiệu khởi động động cơ hai bên ra các chân cần thiết, và tốc độ của 2 bên động cơ bằng nhau.Động cơ được khởi động và xe bắt đầu tiến lên

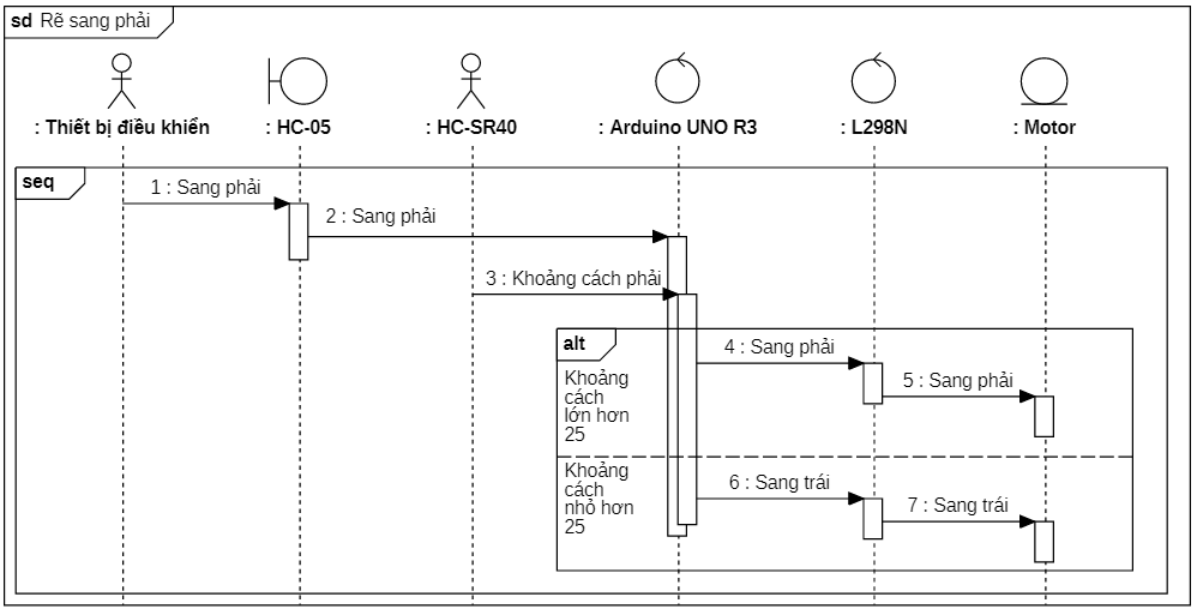
2.2.3.2. Ca sử dụng lùi lại



Hình 3.4: Biểu đồ tuần tự ca sử dụng Lùi lại

Thiết bị điều khiển ra tín hiệu lùi lại ở trên thiết bị điều khiển.Module HC-05 nhận thông tin điều lùi lại phải và chuyển tín hiệu điều khiển cho arduino.Arduino nhận tín hiệu điều khiển lùi lại và xuất tín hiệu khởi động động cơ hai bên lùi lại.Động cơ được khởi động và xe đi lùi lại

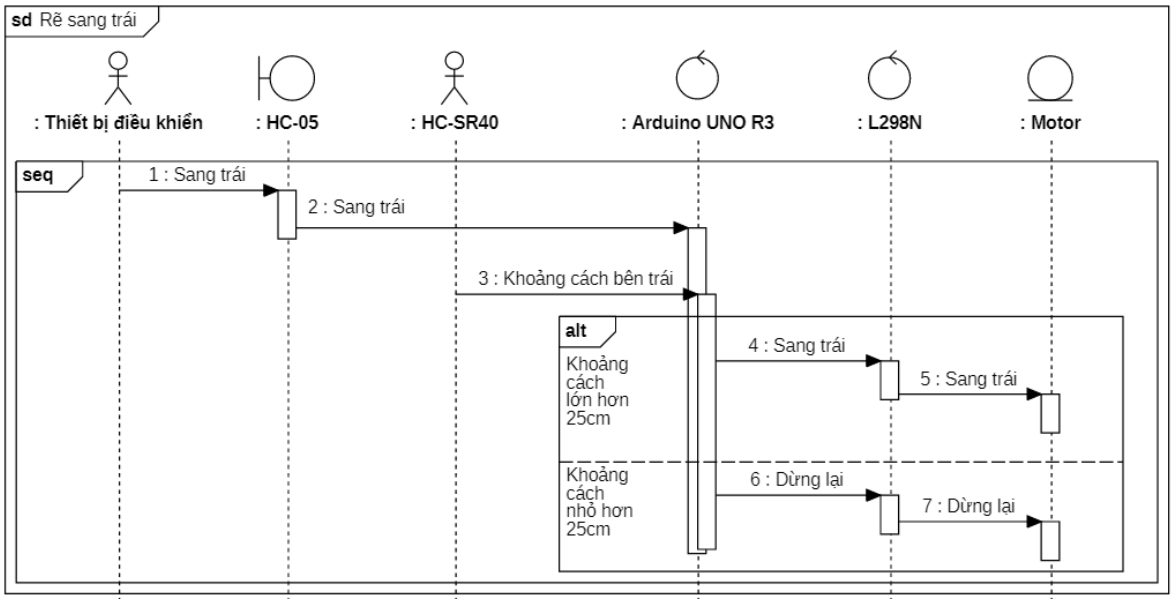
2.2.3.3. Ca sử dụng sang phải



Hình 3.5: Biểu đồ tuần tự ca sử dụng Rẽ sang phải

Thiết bị điều khiển ra tín hiệu rẽ sang phải ở trên thiết bị điều khiển.Module HC-05 nhận thông tin điều khiển rẽ sang phải và chuyển tín hiệu điều khiển cho Arduino. Arduino nhận tín hiệu điều khiển rẽ sang phải và tín hiệu vật cản từ HC-SR04 bên phải, nếu bên phải không có vật cản thì Arduino xuất tín hiệu khởi động động cơ bên phải chạy chậm hơn động cơ bên trái. Động cơ được khởi động

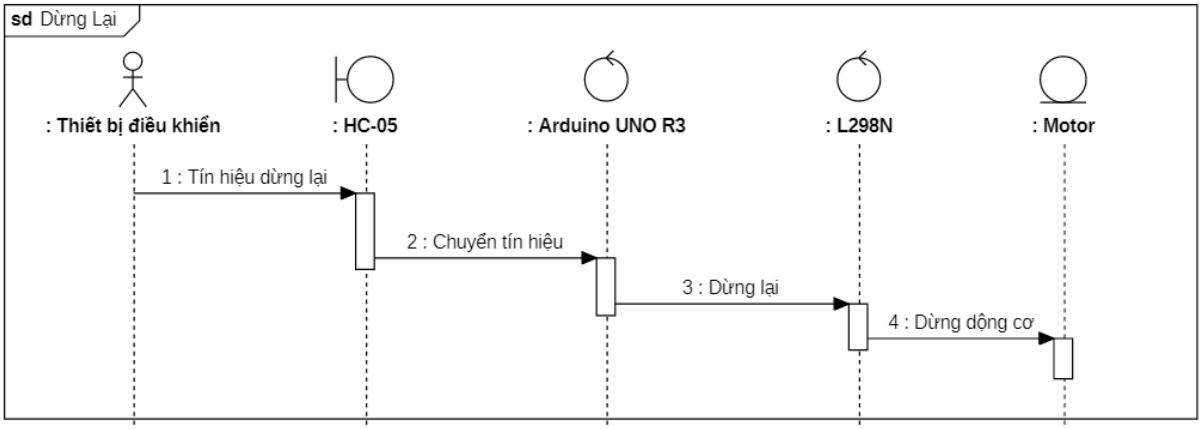
2.2.3.4. Ca sử dụng sang trái



Hình 3.6: Biểu đồ tuần tự ca sử dụng Rẽ sang trái

Thiết bị điều khiển ra tín hiệu rẽ sang trái ở trên thiết bị điều khiển. Module HC-05 nhận thông tin điều khiển rẽ sang trái và chuyển tín hiệu điều khiển cho arduino. Arduino nhận tín hiệu điều khiển rẽ sang trái và xuất tín hiệu khởi động động cơ bên trái chạy chậm hơn động cơ bên phải. Động cơ được khởi động

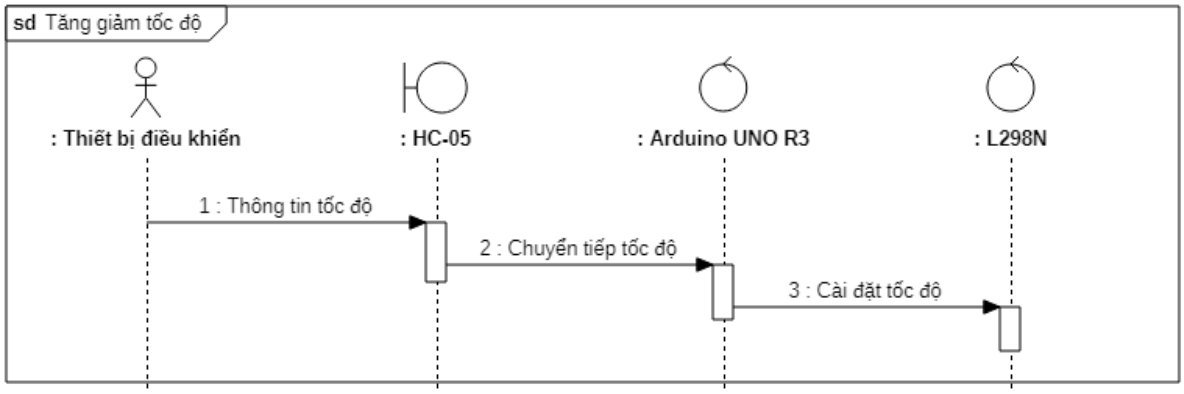
2.2.2.5. Ca sử dụng Dừng lại



Hình 3.7: Biểu đồ tuần tự ca sử dụng dừng lại

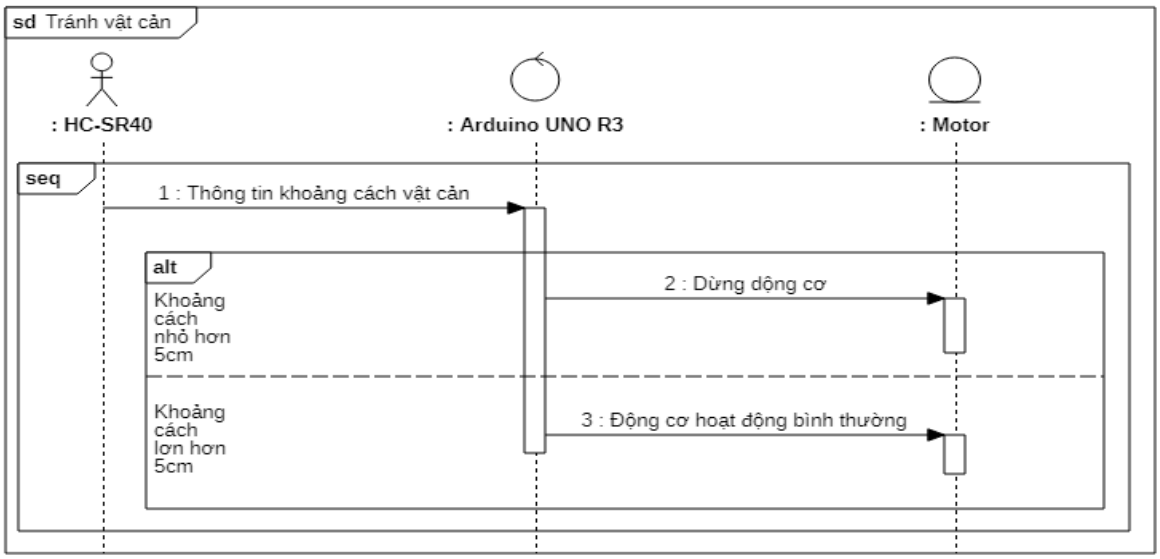
Khi thiết bị người dùng gửi tín hiệu dừng lại thông qua bluetooth, HC-05 sẽ nhận được tín hiệu và gửi tiếp tín hiệu đến Arduino UNO R3, Arduino sẽ nhận tín hiệu, kèm theo là gửi tín hiệu điều khiển ra các chân để dừng động cơ lại.

2.2.2.6. Ca sử dụng tăng giảm tốc độ



Người điều khiển kéo chọn tốc độ ở trên màn hình điều khiển, Thiết bị sẽ bắt sự kiện và gửi tín hiệu thông qua bluetooth, Arduino nhận và chỉnh lại tốc độ xe.

2.2.3.7. Ca sử dụng tránh vật cản



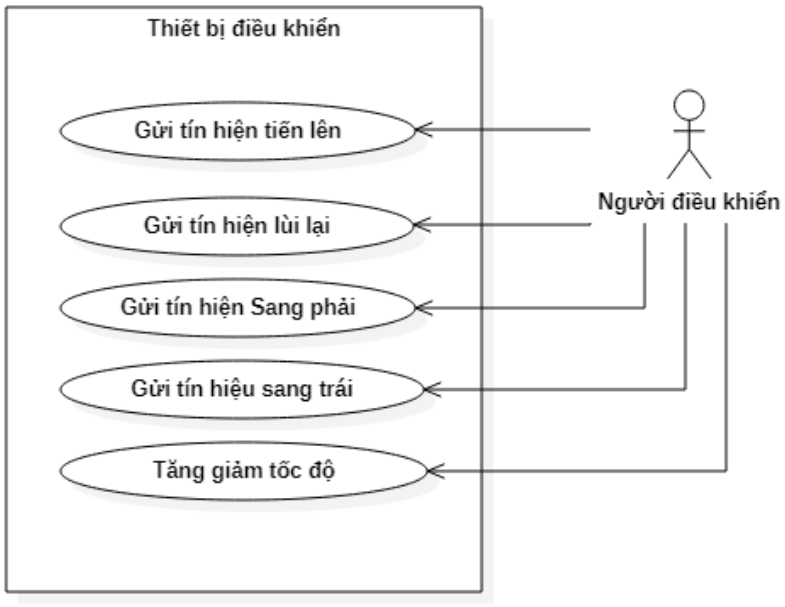
Hình 3.8: Biểu đồ tuần tự ca sử dụng Tránh vật cản

Cảm biến đo khoảng cách từ xe cho tới vật cản ở phía trước. Cảm biến gửi thông tin khoảng cách vật cản cho arduino. Nếu khoảng cách giữa xe và vật cản nhỏ hơn 5cm, xe sẽ không thể tiến lên phía trước, thiết bị điều khiển cần điều khiển xe tránh sang hướng khác

3.3. Phân tích phầm mềm điều khiển

3.3.1. Biểu đồ ca sử dụng của phần mềm điều khiển

Phầm mềm điều khiển được thiết kế để chạy trên nền tảng hệ điều hành android, nó sẽ có các chức năng bao gồm kết nối bluetooth tới thiết bị , gửi tín hiệu điều khiển tiến lên, điều khiển lùi lại, điều khiển rẽ sang trái và điều khiển rẽ sang phải.



Hình 3.9: Biểu đồ ca sử dụng của phần mềm điều khiển

3.3.2. Đặc tả các ca sử dụng

2.3.2.1. Ca sử dụng gửi tín hiệu tiến lên

Use Case	Gửi tín hiện tiến lên
Actor	Người điều khiển
Brief Description	Thiết bị điều khiển gửi tín hiện điều khiển tiến lên
Pre-condition	Thiết bị đã kết nối Bluetooth
Basic Flows	1. Người dùng ấn nút tiến lên ở trên màn hình. 2. Thiết bị bắt sự kiện ACTION_DOWN. 3. Thiết bị gửi tín hiệu tiến lên bằng blueooth. 4. Người dùng thả nút tiến lên. 5. Thiết bị bắt sự kiện ACTION_UP. 6. Thiết bị gửi tín hiệu dừng bằng bluetooth.
Alternative Flows	

Bảng 3.8: Ca sử dụng gửi tín hiệu tiến lên

2.3.2.2. Ca sử dụng gửi tín hiệu lùi xuống

Use Case	Gửi tín hiện lùi xuống
Actor	Người điều khiển
Brief Description	Thiết bị điều khiển gửi tín hiện điều khiển lùi xuống
Pre-condition	Thiết bị đã kết nối Bluetooth
Basic Flows	1. Người dùng ấn và giữ nút lùi xuống ở trên màn hình. 2. Thiết bị bắt sự kiện ACTION_DOWN. 3. Thiết bị gửi tín hiệu lùi lại thông qua bluetooth. 4. Người dùng thả nút ấn ra. 5. Thiết bị bắt sự kiện ACTION_UP. 6. Thiết bị gửi tín hiệu dừng xe lại.
Alternative Flows	

Bảng 3.9: Ca sử dụng gửi tín hiệu lùi xuống

2.3.2.3. Ca sử dụng gửi tín hiệu sang phải

Use Case	Gửi tín hiệu sang phải
Actor	Người điều khiển
Brief Description	Thiết bị điều khiển gửi tín hiện điều khiển sang phải
Pre-condition	Thiết bị đã kết nối Bluetooth
Basic Flows	1. Người dùng ấn và giữ nút sang phải ở trên màn hình. 2. Thiết bị bắt sự kiện ACTION_DOWN. 3. Thiết bị gửi tín hiệu sang phải bằng bluetooth. 4. Người dùng thả nút ấn ra. 5. Thiết bị bắt sự kiện ACTION_UP. 6. Thiết bị gửi tín hiệu dừng xe lại.
Alternative Flows	

Bảng 3.10: Ca sử dụng gửi tín hiệu sang phải

2.3.2.4. Ca sử dụng gửi tín hiệu sang trái

Use Case	Gửi tín hiệu sang trái
Actor	Người điều khiển
Brief Description	Thiết bị điều khiển gửi tín hiện điều khiển sang trái
Pre-condition	Thiết bị đã kết nối Bluetooth
Basic Flows	1. Người dùng ấn và giữ nút sang trái ở trên màn hình. 2. Thiết bị bắt sự kiện ACTION_DOWN. 3. Thiết bị gửi tín hiệu sang trái bằng bluetooth. 4. Người dùng thả nút ấn ra. 5. Thiết bị bắt sự kiện ACTION_UP. 6. Thiết bị gửi tín hiệu dừng lại bằng bluetooth.
Alternative Flows	

Bảng 3.11: Ca sử dụng gửi tín hiệu sang trái

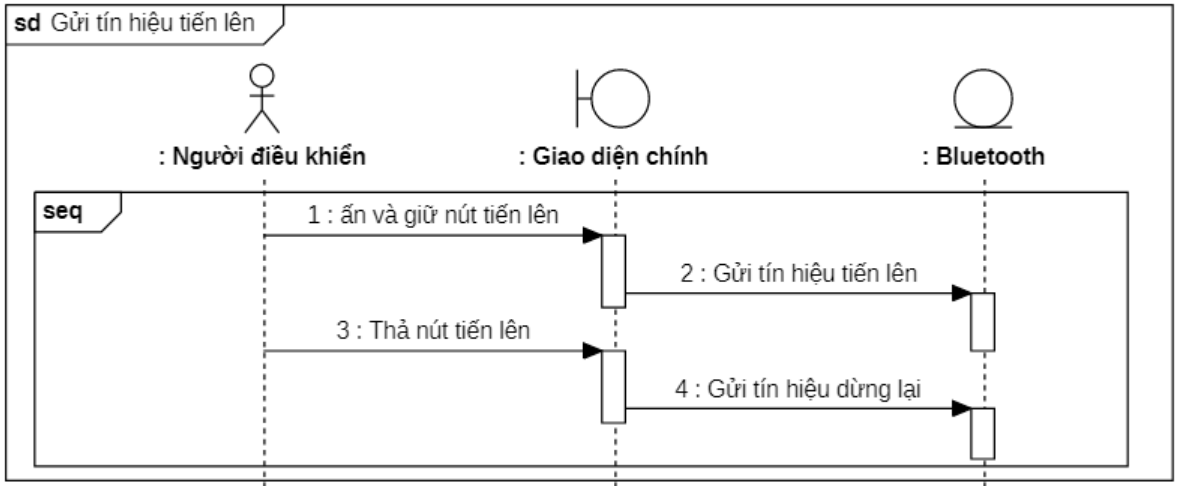
2.3.2.5. Ca sử dụng tăng giảm tốc độ

Use Case	Tăng giảm tốc độ
Actor	Người điều khiển
Brief Description	Thiết bị điều khiển gửi tín hiện điều tăng giảm tốc độ
Pre-condition	Thiết bị đã kết nối Bluetooth
Basic Flows	1. Người dùng chọn tốc độ nhờ thanh trượt trên màn hình 2. Thiết bị bắt sự kiện onChange. 3. Thiết bị gửi tín hiệu tốc độ bằng bluetooth.
Alternative Flows	

Bảng 3.12: Ca sử dụng tăng giảm tốc độ

3.3.3. Phân tích các ca sử dụng

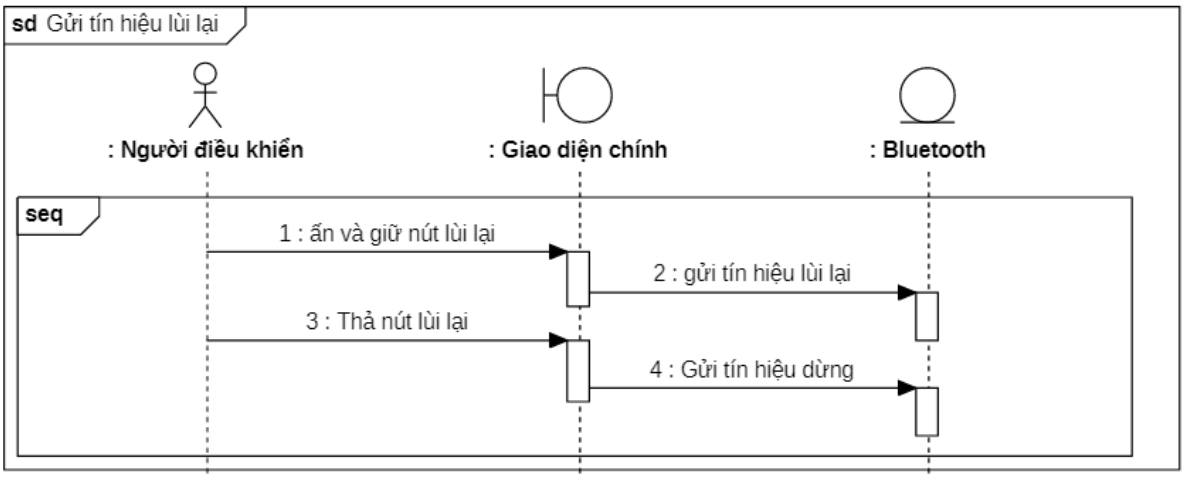
3.3.3.1. Ca sử dụng gửi tín hiệu tiến lên



Hình 3.10: Biểu đồ tuần tự ca sử dụng gửi tín hiệu tiến lên

Người dùng ấn nút tiến lên ở trên màn hình. Thiết bị gửi thông tin tiến lên thông qua bluetooth.

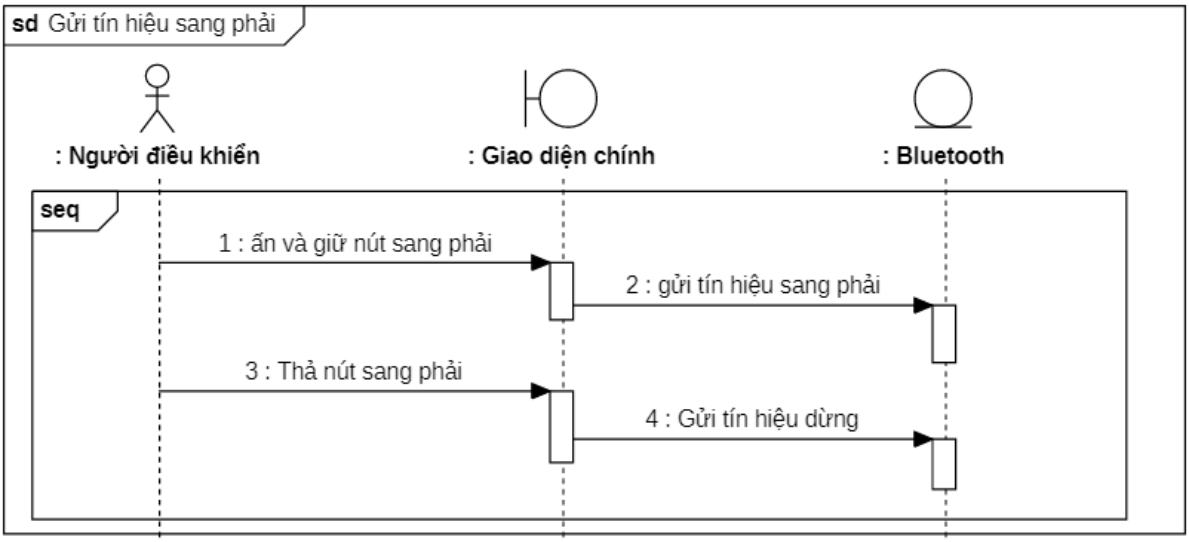
3.3.3.2. Ca sử dụng gửi tín hiệu lùi lại



Hình 3.11: Ca sử dụng gửi tín hiệu lùi lại

Người dùng ấn nút lùi xuống ở trên màn hình. Thiết bị gửi thông tin lùi xuống thông qua bluetooth.

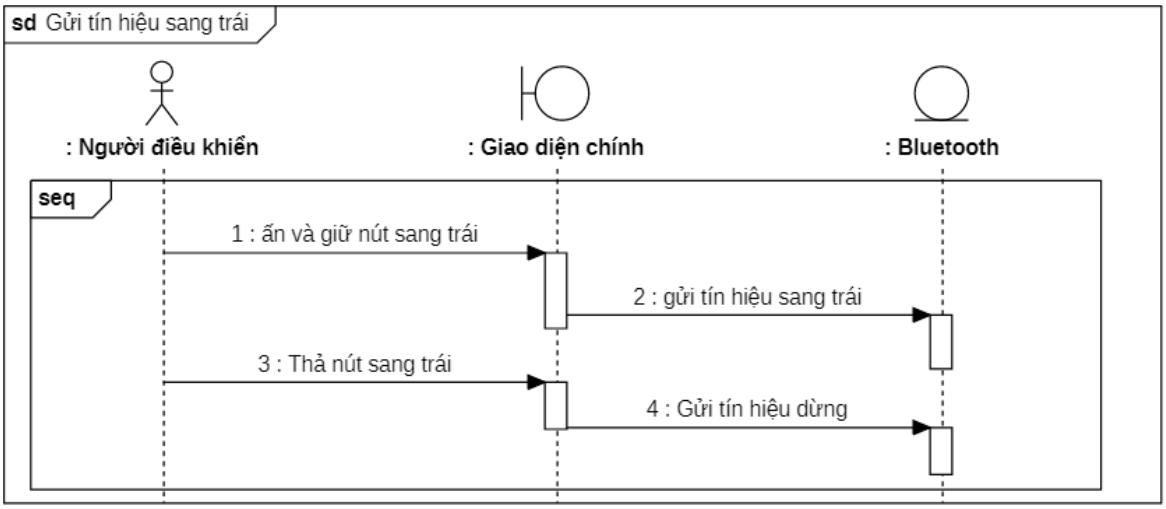
3.3.3.3. Ca sử dụng gửi tín hiệu sang phải



Hình 3.12: Biểu đồ tuần tự ca sử dụng gửi tín hiệu sang phải

Người dùng ấn nút sang phải ở trên màn hình. Thiết bị gửi thông tin sang phải thông qua bluetooth.

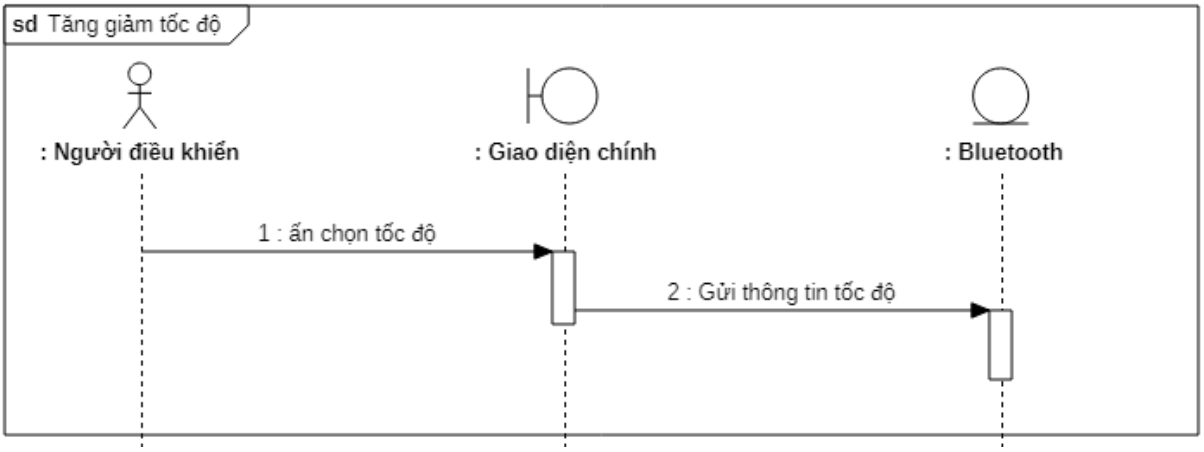
3.3.3.4. Ca sử dụng gửi tín hiệu sang trái



Hình 3.13: Biểu đồ tuần tự ca sử dụng gửi tín hiệu sang trái

Người dùng ấn nút sang trái ở trên màn hình. Thiết bị gửi thông tin sang trái thông qua bluetooth.

3.3.3.5. Ca sử dụng thay đổi tốc độ



Người dùng ấn chọn tốc độ dựa vào thanh tốc độ hiển thị ở trên màn hình, thiết bị sẽ bắt sự kiện ấn nút và gửi thông tin tốc độ thông qua bluetooth.

CHƯƠNG 4. THIẾT KẾ HỆ THỐNG

4.1. Thiết kế xe tự hành điều khiển bằng bluetooth

4.1.1. Thiết kế phần mềm

4.1.1.1. Tổng quan vấn đề thiết kế phần mềm

Đối với bất cứ hệ thống nào, phần mềm hệ thống có ý nghĩa như là một bộ não, chi phối mọi hành động được đưa ra. Ngoài yếu tố phần cứng, sự thành công của một hệ thống nhúng còn được quyết định bởi sự hoàn hảo của phần mềm mà nó được cài đặt. Ở đây khi cấu trúc linh kiện phần cứng là như nhau đối với mỗi loại xe, vậy điểm tạo nên sự khác biệt giữa các xe đó nằm ở phần mềm mỗi loại xe sử dụng.

Và tất nhiên, vấn đề thời gian thực là vấn đề hầu hết các hệ thống nhúng cần đáp ứng, nhất là đối với hệ thống xe tự hành nói chung. Sẽ thật là nguy hiểm khi mà một xe tự hành gặp vật cản nhưng tác vụ quản lý hệ thống phanh lại không đáp ứng được trong một khoảng thời gian an toàn.

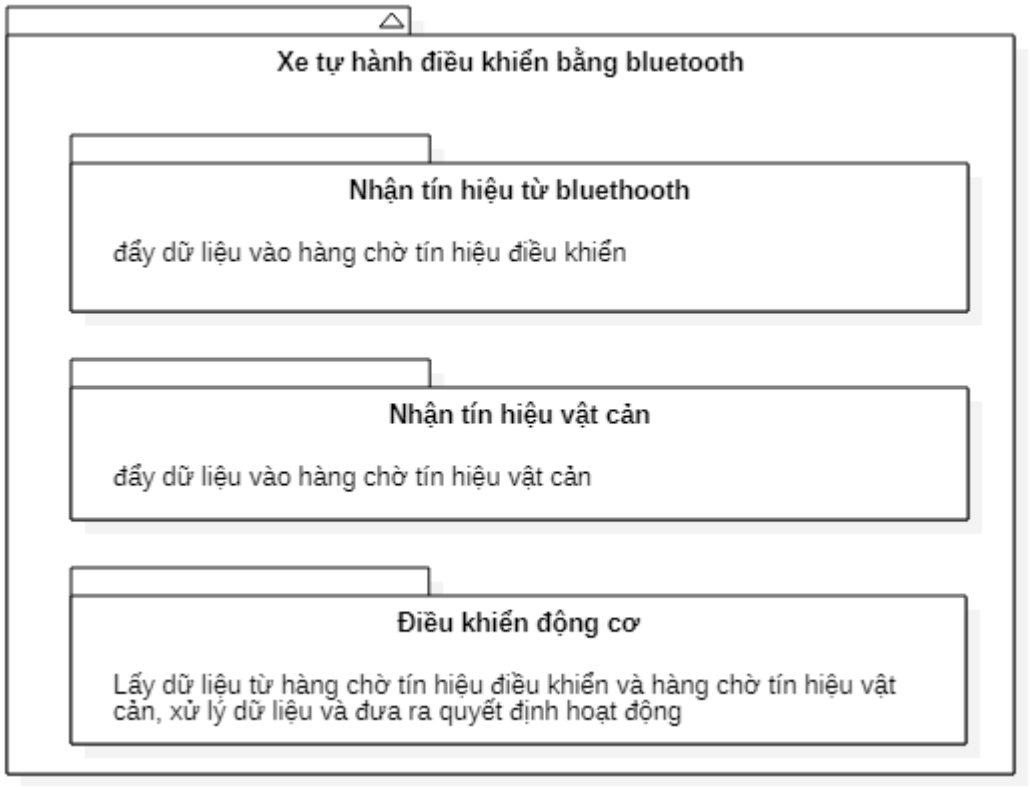
Để đáp ứng vấn đề thời gian thực trong hệ thống nhúng, tốc độ thực thi và độ đơn giản của mã lệnh được đặt lên hàng đầu, do đó ngôn ngữ phổ biến nhất được sử dụng để tạo lên các hệ thống nhúng là C/C++, lý do bởi vì C/C++ là ngôn ngữ lập trình gần với hợp ngữ và ngôn ngữ máy, nên tốc độ thực thi nhanh hơn nhiều so với một số loại ngôn ngữ khác.

Một phần quan trọng nữa trong hệ thống nhúng đó chính là hệ điều hành thời gian thực. Các hệ thống nhúng không thể sử dụng được các hệ điều hành lớn và công kềnh như là Window, thay vào đó nó sử dụng các hệ điều hành chuyên biệt khác để điều phối cấp phát tài nguyên hệ thống. Hệ điều hành FreeRTOS là một trong các hệ điều hành chuyên biệt đó, nó được viết và thực thi dựa trên ngôn ngữ C và nó giúp ra tạo, quản lý các tác vụ trong hệ thống, kèm theo đó là những thư viện hỗ trợ liên quan giúp ta xây dựng hệ thống nhúng một cách đơn giản và hiệu quả.

Hình 4.1: Hệ điều hành FreeRTOS

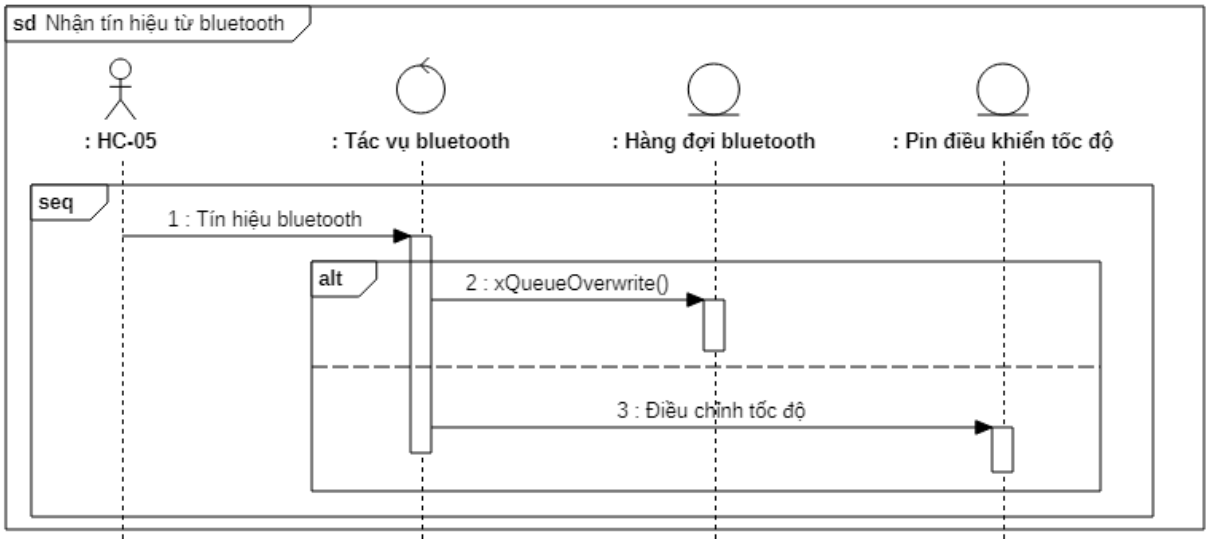
4.1.1.2. Xây dựng các tác vụ trong hệ thống

Các tác vụ trong hệ thống xe tự hành điều khiển bằng bluethooth được chia làm ba tác vụ chính là nhận tín hiệu đều khiển từ bluethooth, nhận tín hiệu vật cản và điều khiển động cơ hoạt động.



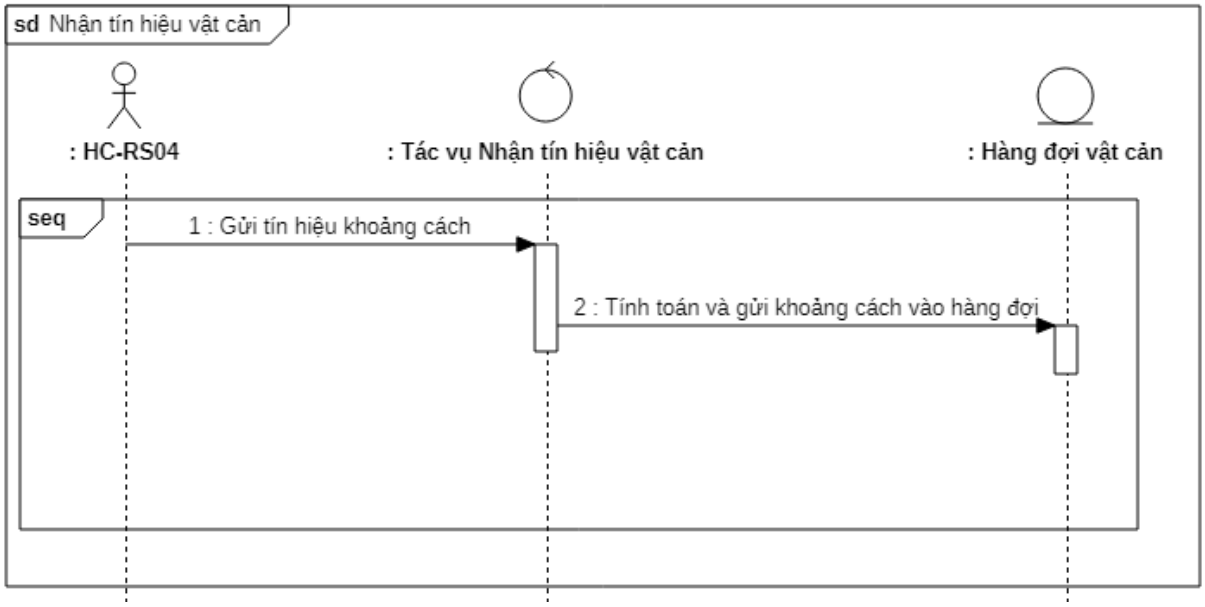
Hình 4.2: Tổng quan các tác vụ ở trong hệ thống

Đối với tác vụ nhận tín hiệu từ bluetooth, nhóm sử dụng thư viện build-in của hệ điều hành FreeRTOS là queue.h để xây dựng lên hàng đợi. Khi nhận được một message của Module HC-05 gửi về, nó sẽ đẩy tín hiệu này vào trong hàng chờ để cho tác vụ điều khiển động cơ đọc và thực thi lệnh, trong trường hợp đó là tín hiệu tăng giảm tốc độ, thì nó sẽ cài đặt tốc độ cho xe bằng API analogWrite()



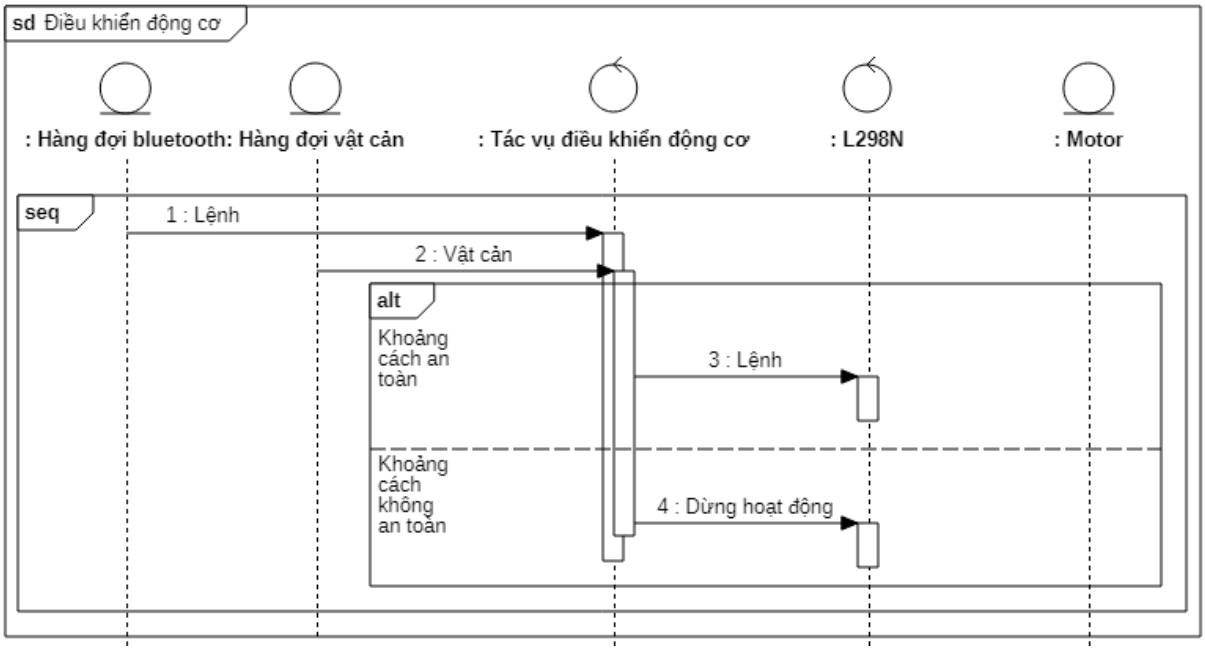
Hình 4.3: Biểu đồ tuần tự tác vụ nhận tín hiệu từ bluetooth

Đối với tách vụ nhận tí hiệu vật cản, Module HC-SR04 sẽ liên tục gửi thông tin khoảng cách của vật cản vào hàng đợi, từ đó tác vụ điều khiển động cơ lấy thông tin tín hiệu ra, phân tích xem có nên cho xe di chuyển tiếp hay không.

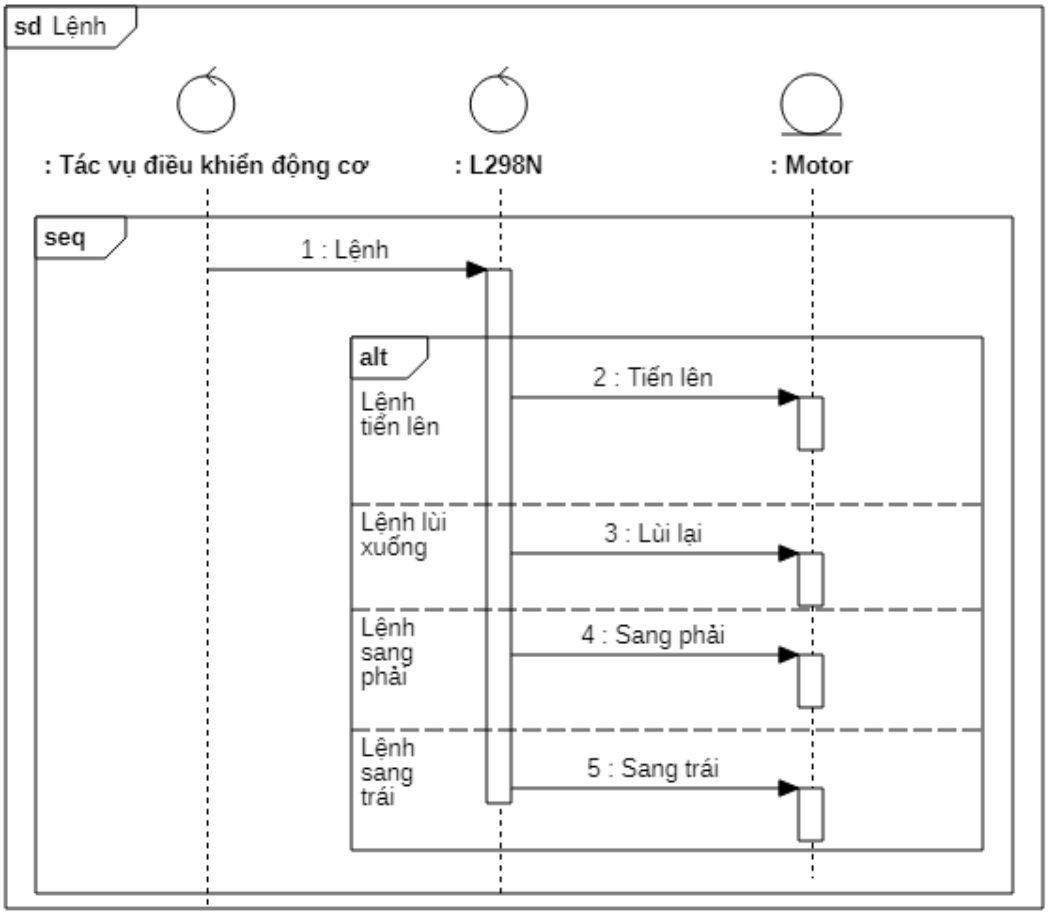


Hình 4.4: Biểu đồ tuần tự tác vụ nhận tín hiệu vật cản

Cuối cùng là tác vụ điều khiển động cơ, tác vụ này sẽ có nghiệm vụ lấy dữ liệu từ hàng đợi bluetooth và hàng đợi vật cản. Đầu tiên, tác vụ xe kiểm tra xem xe có bị vật cản không, nếu không thì sẽ cho động cơ hoạt động theo lệnh được đẩy vào trong hàng đợi bluetooth.

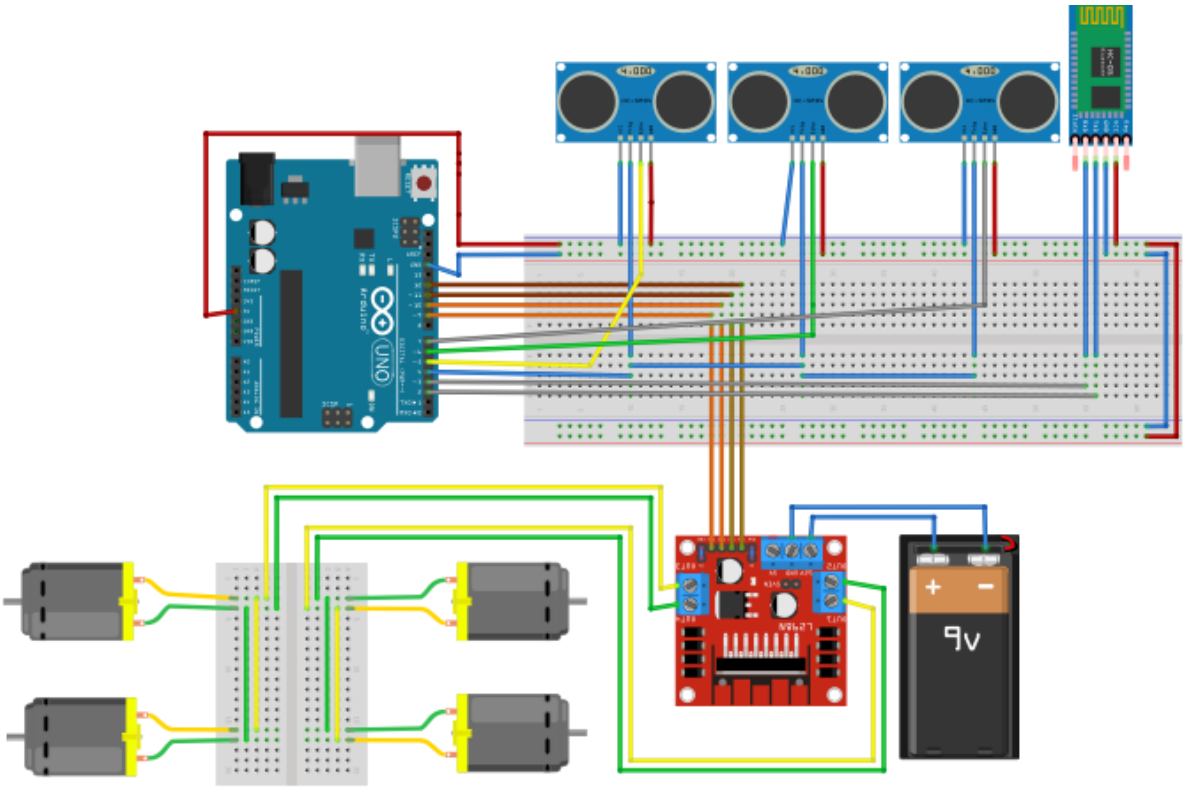


Hình 4.5: Biểu đồ tuần từ tác vụ điều khiển động cơ



Hình 4.6: Biểu đồ tuần tự chi tiết lệnh

4.1.2. Thiết kế phần cứng



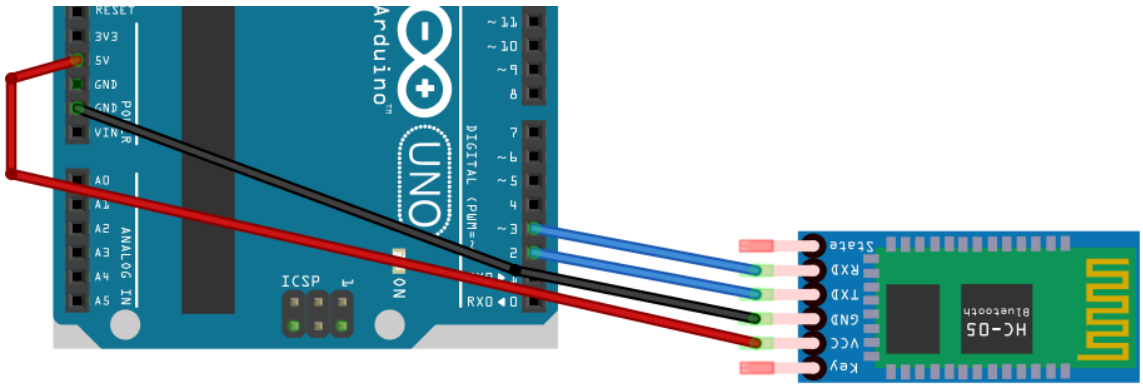
Hình 4.7: Thiết kế mạch của hệ thống

Đối với Module bluetooth HC-05, do khuyến nghị không nên sử dụng hai chân Serial là chân 1 và chân 2 ở trên Arduino, vì dễ dẫn đến xung đột khi nạp chương trình, nên nhóm thiến hành config và sử dụng chân 2,3 lần lượt làm chân TXD và RXD để trao đổi thông tin bluetooth. Arduino có thư viện build-in là SoftwareSerial.h để hỗ trợ chuyển đổi hai chân digital 2 và 3 thành chân Serial.

```
#include <SoftwareSerial.h>
#define TX_PIN 3;
#define RX_PIN 2;
SoftwareSerial bluetooth(TX_PIN, RX_PIN);
```

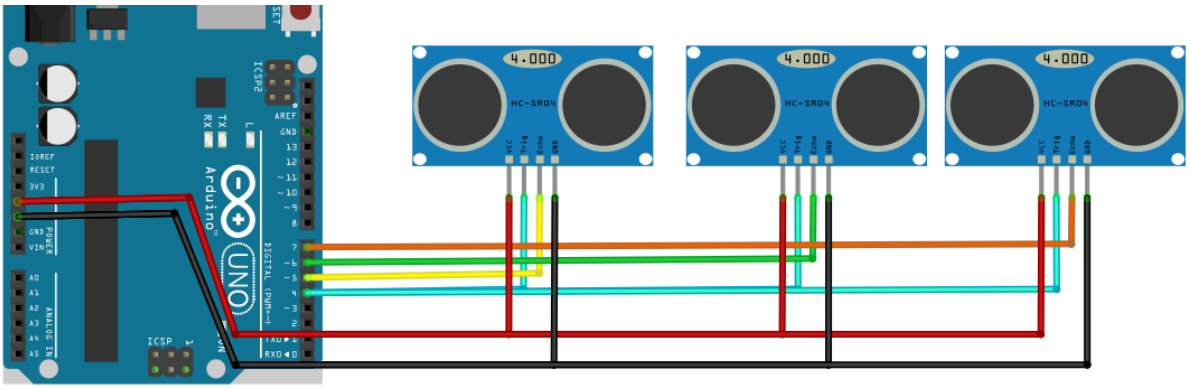
Hình 4.8: Code mô phỏng cách cấu hình chân Serial

Ta lần lượt nối chân 3 của Arduino với chân RXD của Module HC-05, nối chân 2 của Arduino với chân TXD của Module HC-05, tiếp theo tiến hành nối Vcc và GND của Module HC-05 vào Arduino.



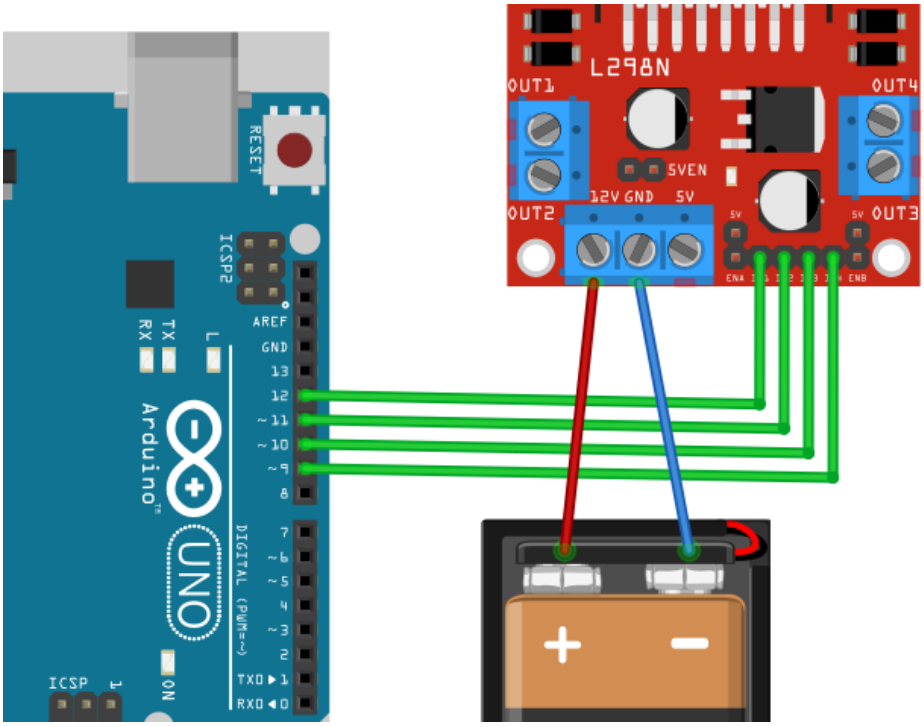
Hình 4.9: Cách nối Module HC-05 vào arduino

Đối với Module HC-SR04, ở đây đã sử dụng 4 module HC-SR04, nối chân trig của cả 3 module chung vào pin 4 của Arduino, nối Vcc và GND cho cả 3 module, và lần lượt nối chân echo của từng module vào chân 5, 6 và 7.



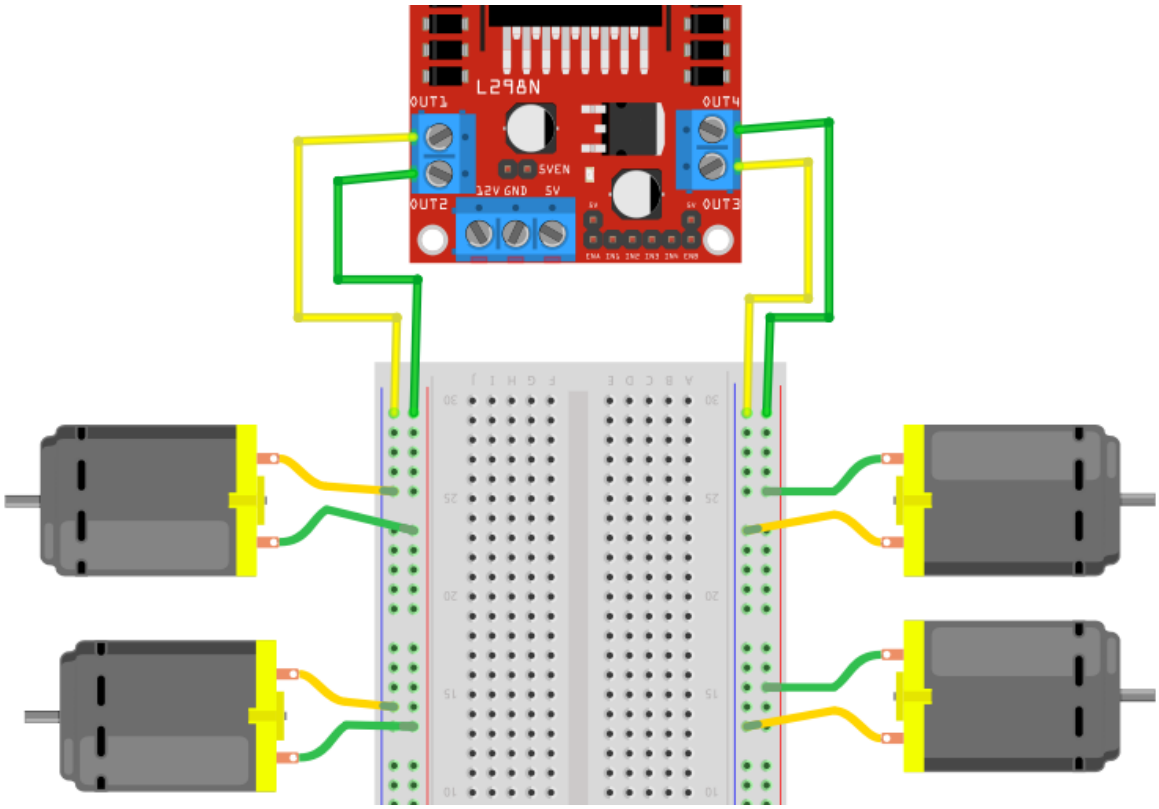
Hình 4.10: Cách mắc Module HC-SR04

Đối với Module L298D, mắc lần lượt bốn chân của Module L298D là IN1, IN2, IN3, IN4 vào bốn chân 9, 10, 11, 12 của arduino, mắc cổng 12V của L298D vào cực dương của pin, mắc chân GND vào cực âm của pin.



Hình 4.11: Cách mắc Module L298D

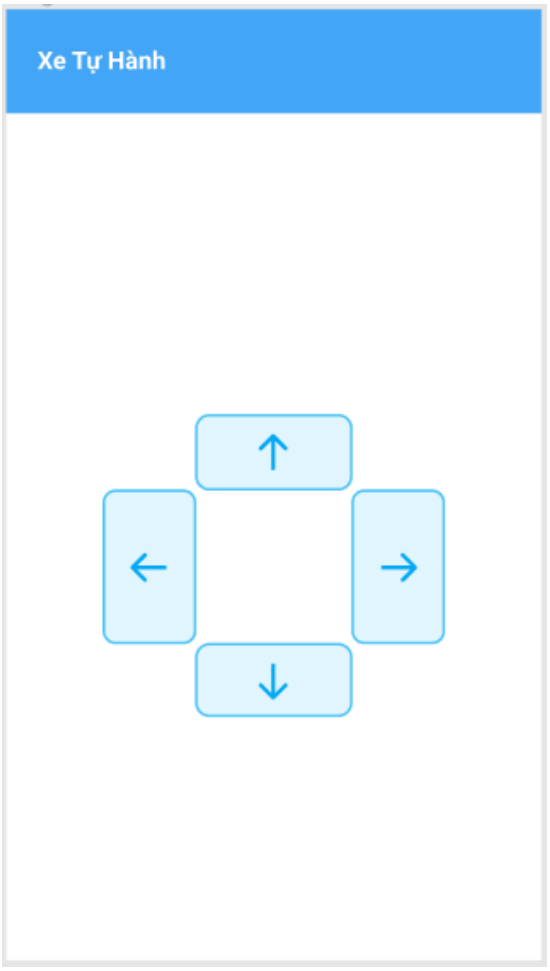
Cuối cùng, mắc lần lượt các động cơ vào Module L298D theo sơ đồ sau



Hình 4.12: Cách mắc Motor vào L298N

4.2. Thiết kế thiết bị điều khiển

Thiết bị điều khiển được dùng là điện thoại chạy hệ điều hành android, được cài phần mềm chuyên biệt để điều khiển xe. Màn hình chứa bao gồm bốn nút là nút điều khiển lên, xuống, sang phải, sang trái.



Hình 4.13: Giao diện phần mềm điều khiển

PHỤ LỤC

CODE Arduino của xe tự hành điều khiển bằng bluetooth:

```
#include<Arduino_FreeRTOS.h>
#include<task.h>
#include<queue.h>
#include<SoftwareSerial.h>

QueueHandle_t frontQueue;
QueueHandle_t rightQueue;
QueueHandle_t leftQueue;
QueueHandle_t commandQueue;

SoftwareSerial bluetooth(2,3);

int trig=4;
int echoRight=5;
int echoLeft=7;
int echoFront=6;

int speedPin = 9;

int rightmotor1=8;
int rightmotor2=10;
int leftmotor1=11;
int leftmotor2=12;

void setup(){
  //=========== active pin and serial ==========
  Serial.begin(9600);
  bluetooth.begin(9600);
  pinMode(trig, OUTPUT);
  pinMode(echoRight, INPUT);
  pinMode(echoFront, INPUT);
  pinMode(echoLeft, INPUT);
  pinMode(speedPin, OUTPUT);
  pinMode(rightmotor1, OUTPUT);
  pinMode(rightmotor2, OUTPUT);
  pinMode(leftmotor1, OUTPUT);
  pinMode(leftmotor2, OUTPUT);
  //=========================================

  analogWrite(speedPin,250);
```

```
  //================ create queue ================
  frontQueue = xQueueCreate(1,sizeof(int));
  rightQueue = xQueueCreate(1,sizeof(int));
  leftQueue = xQueueCreate(1,sizeof(int));
  commandQueue = xQueueCreate(1,sizeof(int));
  //==============================================

  //============== create task ===================
  xTaskCreate(hdsr04,"khoangcach", 200,NULL,1,NULL);
  xTaskCreate(start,"chay", 200,NULL,1,NULL);
  xTaskCreate(command,"cmd",200,NULL,1,NULL);
  vTaskStartScheduler();
  //==============================================
}

void loop(){}

void start(void *param){
  int leftspace;
  int rightspace;
  int frontspace;
  while(1){

    //===== get command from queue =========
    int cmd;
    xQueuePeek(commandQueue,&cmd,(TickType_t)0);

    //======================================

    //======== L398N Controller ============
    switch(cmd){
      case '1':
        xQueuePeek(frontQueue, &frontspace, ( TickType_t )0);
        if(frontspace>40){
          front();
        }else stp();
        break;
      case '2':
        back();
        break;
      case '3':
        xQueuePeek(leftQueue, &leftspace, ( TickType_t )0);
        if(leftspace>25){
          left();
        }else stp();
        break;
      case '4':
```

```
        xQueuePeek(rightQueue, &rightspace, ( TickType_t )0);
        if(rightspace>25){
          right();
        }else stp();
        break;
       case '5':
        stp();
        break;
      default:
        break;
    }
    //====================================
    vTaskDelay(20/ portTICK_PERIOD_MS);
  }
}


void command(void *param){
  while(1){
    if(bluetooth.available()){
      int cmd;
      cmd = bluetooth.read();
      if(cmd< 60 && cmd >40){
        xQueueOverwrite(commandQueue,&cmd);
      }else if (cmd>=100 && cmd<=250){
        analogWrite(speedPin, cmd);
      }
    }
    vTaskDelay(40/ portTICK_PERIOD_MS);
  }
}

void hdsr04(void *param){
  unsigned long duration1;
  unsigned long duration2;
  unsigned long duration3;

  int distance1;
  int distance2;
  int distance3;
  while(1){
    //   right hc sr04

    digitalWrite(trig,LOW);
    vTaskDelay(2/portTICK_PERIOD_MS);
    digitalWrite(trig,HIGH);
    vTaskDelay(10/portTICK_PERIOD_MS);
```

```
    digitalWrite(trig,LOW);
    duration1 = pulseIn(echoRight,HIGH);
    distance1 = int(duration1/2/29.412);
    if(distance1>0){
      xQueueOverwrite(rightQueue,&distance1);
    }
    vTaskDelay(20/portTICK_PERIOD_MS);

    // front hc-sr04

    digitalWrite(trig,LOW);
    vTaskDelay(2/portTICK_PERIOD_MS);
    digitalWrite(trig,HIGH);
    vTaskDelay(10/portTICK_PERIOD_MS);
    digitalWrite(trig,LOW);
    duration2 = pulseIn(echoFront,HIGH);
    distance2 = int(duration2/2/29.412);
    if(distance2>0){
      xQueueOverwrite(frontQueue,&distance2);
    }
    vTaskDelay(20/portTICK_PERIOD_MS);

    // left
    digitalWrite(trig,LOW);
    vTaskDelay(2/portTICK_PERIOD_MS);
    digitalWrite(trig,HIGH);
    vTaskDelay(10/portTICK_PERIOD_MS);
    digitalWrite(trig,LOW);
    duration3 = pulseIn(echoLeft,HIGH);
    distance3 = int(duration3/2/29.412);
    if(distance3>0){
      xQueueOverwrite(leftQueue,&distance3);
    }
    vTaskDelay(20/portTICK_PERIOD_MS);
  }
}

void back(){
  digitalWrite(leftmotor1, HIGH);
  digitalWrite(leftmotor2, LOW);
  digitalWrite(rightmotor1, HIGH);
  digitalWrite(rightmotor2, LOW);
}
void front(){
  digitalWrite(leftmotor1, LOW);
  digitalWrite(leftmotor2, HIGH);
  digitalWrite(rightmotor1, LOW);
```

```
  digitalWrite(rightmotor2, HIGH);
}
void right(){
  digitalWrite(leftmotor1, LOW);
  digitalWrite(leftmotor2, HIGH);
  digitalWrite(rightmotor1, HIGH);
  digitalWrite(rightmotor2, LOW);
}
void left(){
  digitalWrite(leftmotor1, HIGH);
  digitalWrite(leftmotor2, LOW);
  digitalWrite(rightmotor1, LOW);
  digitalWrite(rightmotor2, HIGH);
}

void stp(){
  digitalWrite(leftmotor1, LOW);
  digitalWrite(leftmotor2, LOW);
  digitalWrite(rightmotor1, LOW);
  digitalWrite(rightmotor2, LOW);
}
```

TÀI LIỆU THAM KHẢO

Nguyễn Ngọc Bình, *Công nghệ phần mềm nhúng,* Nhà xuất bản đại học Quốc gia Hà Nội, 2013.

Lê Thị Hồng Vân, *Giáo trình hệ điều hành nhúng thời gian thực.*

http://arduino.vn/bai-viet/42-arduino-uno-r3-la-gi

http://arduino.vn/bai-viet/893-cach-dung-module-dieu-khien-dong-co-l298n-cau-h-de-dieu-khien-dong-co-dc

http://arduino.vn/bai-viet/953-huong-dan-thiet-lap-chi-tiet-cho-module-bluetooth

http://arduino.vn/bai-viet/233-su-dung-cam-bien-khoang-cach-hc-sr04

http://arduino.vn/bai-viet/639-du-xe-dieu-khien-tu-xa-qua-bluetooth

https://www.youtube.com/watch?v=ChRvQ1nh_Og

https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf

http://arduino.vn/bai-viet/953-huong-dan-thiet-lap-chi-tiet-cho-module-bluetooth