**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

Viện Công nghệ thông tin và Truyền thông



**Báo cáo bài tập lớn**

**Học phần Hệ nhúng - IT4210**

*Mã lớp: 115665*

***Đề tài:***

*Xe cần cẩu điều khiển bằng cảm biến gia tốc góc và bluetooth*

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiện: | Lê Trọng Nhân | MSSV: 20173292 |
| Trần Phương Thảo | MSSV: 20173383 |
| Mạc Quang Huy | MSSV: 20173169 |
| Giảng viên hướng dẫn: | TS. Ngô Lam Trung | |

***Hà Nội, ngày 15 tháng 7 năm 2020***

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

Viện Công nghệ thông tin và Truyền thông



**Báo cáo bài tập lớn**

**Học phần Hệ nhúng - IT4210**

*Mã lớp: 115665*

***Đề tài:***

*Xe cần cẩu điều khiển bằng cảm biến gia tốc góc và bluetooth*

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiện: | Lê Trọng Nhân | MSSV: 20173292 |
| Trần Phương Thảo | MSSV: 20173383 |
| Mạc Quang Huy | MSSV: 20173169 |
| Giảng viên hướng dẫn: | TS. Ngô Lam Trung | |

***Hà Nội, ngày 15 tháng 7 năm 2020***

# MỤC LỤC

[MỤC LỤC 1](#_Toc45673640)

[MỤC LỤC HÌNH VẼ 3](#_Toc45673641)

[1. Mô tả bài toán 4](#_Toc45673642)

[2. Yêu cầu của sản phẩm 4](#_Toc45673643)

[2.1. Yêu cầu chức năng 4](#_Toc45673644)

[2.2. Yêu cầu phi chức năng 5](#_Toc45673645)

[3. Thiết kế phần cứng 6](#_Toc45673646)

[3.1. Giới thiệu tổng quan các thành phần phần cứng 6](#_Toc45673647)

[3.1.1. Arduino Uno 6](#_Toc45673648)

[3.1.2. Arduino Nano 8](#_Toc45673649)

[3.1.3. Cảm biến gia tốc MPU6050 11](#_Toc45673650)

[3.1.4. Bluetooth HC-05 12](#_Toc45673651)

[3.1.5. Motor Shield L293D 15](#_Toc45673652)

[3.2. Thiết bị điều khiển 16](#_Toc45673653)

[3.2.1. Sơ đồ thiết kế 16](#_Toc45673654)

[3.2.2. Sơ đồ nguyên lý hoạt động 17](#_Toc45673655)

[3.3. Xe điều khiển 18](#_Toc45673656)

[3.3.1. Sơ đồ thiết kế 18](#_Toc45673657)

[3.3.2. Sơ đồ nguyên lý hoạt động 19](#_Toc45673658)

[3.4. Cánh tay robot 19](#_Toc45673659)

[3.4.1. Sơ đồ thiết kế 19](#_Toc45673660)

[3.4.2. Nguyên lý hoạt động 19](#_Toc45673661)

[4. Thiết kế phần mềm 20](#_Toc45673662)

[4.1. Xe điều khiển: 20](#_Toc45673663)

[4.2. Cánh tay robot 22](#_Toc45673664)

[4.3. Thiết bị điều khiển 30](#_Toc45673665)

[5. Kết quả 35](#_Toc45673666)

[5.1. Kết quả đạt được: 35](#_Toc45673667)

[5.2. Hạn chế 36](#_Toc45673668)

[6. Tài liệu tham khảo 36](#_Toc45673669)

# MỤC LỤC HÌNH VẼ

[Hình 1. Arduino Uno 5](#_Toc45673584)

[Hình 2. Arduino nano 7](#_Toc45673585)

[Hình 3. MPU6050 10](#_Toc45673586)

[Hình 4. HC-05 11](#_Toc45673587)

[Hình 5. L293D 14](#_Toc45673588)

[Hình 6. Sơ đồ mạch thiết bị điều khiển 16](#_Toc45673589)

[Hình 7. Sơ đồ nguyên lý hoạt động thiết bị điều khiển 16](#_Toc45673590)

[Hình 8. Sơ đồ mạch điều khiển 17](#_Toc45673591)

[Hình 9. Nguyên lý hoạt động xe điều khiển 18](#_Toc45673592)

[Hình 10. Sơ đồ mạch cánh tay Robot 18](#_Toc45673593)

# 1. Mô tả bài toán

Xã hội phát triển kéo theo những cải tiến không ngừng về công nghệ và sự ra đời của nhiều giải pháp hiện đại giúp nâng cao chất lượng sống, trong đó có xe thông minh – smart car. Xe thông minh là xu hướng phổ biến trên thế giới khi hỗ trợ tiết kiệm thời gian và công sức nhờ áp dụng công nghệ hiện đại. Tại Việt Nam, ứng dụng này cũng ngày càng được lòng người dùng và trở thành lựa chọn ưu tiên.

Với nhu cầu ngày càng tăng về ứng dụng công nghệ thông tin trong đời sống, kết hợp với hiện trạng thực tế khảo sát được và kiến thức được giảng dạy trên lớp, nhóm chúng em quyết định tiến hành thực hiện đề tài xe cần cẩu điều khiển bằng cảm biến gia tốc góc và bluetooth. Đây là cơ hội để nhóm em thực hành và hiểu sâu hơn về những kiến thức đã học, đồng thời biết cách áp dụng kiến thức đã học vào điều kiện môi trường thực tế. Chính đề tài lần này sẽ là cơ sở, là bước đệm để nhóm chúng em thực hiện những dự án lớn hơn sau này liên quan đến xe thông minh.

# 2. Yêu cầu của sản phẩm

## 2.1. Yêu cầu chức năng

- Các chức năng của từng thành phần trong xe cần cẩu điều khiển bằng cảm biến gia tốc góc và Bluetooth:

+ Cảm biến gia tốc góc: Đọc dữ liệu từ môi trường thực tế.

+ Arduino Nano: Nhận dữ liệu từ cảm biến gia tốc góc và xử lý dữ liệu tương ứng để gửi đi thông qua bluetooth, đồng thời hiển thị mode điều khiển thông qua hai đèn led.

+ Bluetooth: một bluetooth gửi dữ liệu đọc được từ Arduino nano và bluetooth còn lại nhận dữ liệu tương ứng gửi về cho Arduino Uno.

+ Arduino Nano: xử lý dữ liệu nhận được để điều khiển xe và cánh tay gắp thả vật tương ứng với các mode điều khiển.

- Về tổng quan, sản phẩm có các chức năng:

+ Gửi nhận dữ liệu.

+ Điều khiển các phần khác nhau của sản phẩm dựa trên mode điều khiển người dùng chọn.

## 2.2. Yêu cầu phi chức năng

- Các yêu cầu phi chức năng của sản phẩm:

+ Độ tin cậy: Sản phẩm có khả năng chạy ổn định, lâu dài, chịu lỗi.

+ Hiệu suất: Sản phẩm có hiệu suất ổn định, thời gian phản hổi của việc gửi, nhận và xử lý dữ liệu giữa các thành phần là nhanh chóng và ổn định, không ảnh hưởng đến trải nghiệm của người dùng.

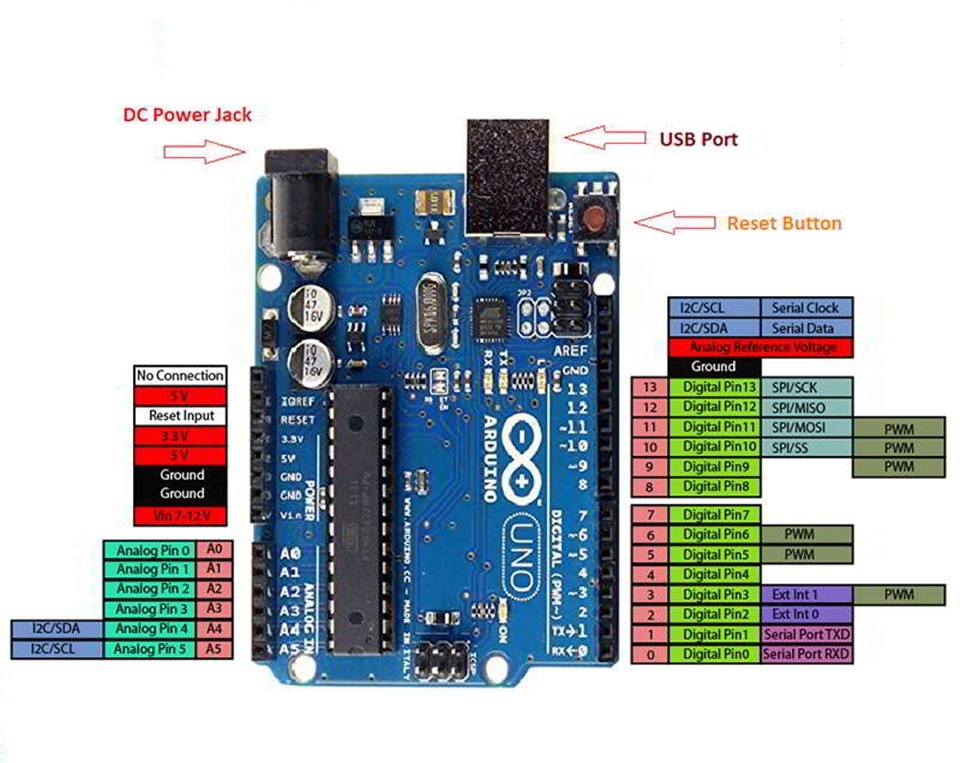
+ Khả năng sử dụng và tương tác: Sản phẩm thân thiện với người dùng, có khả năng tương tác tốt, mang đến trải nghiệm người dùng ổn định và thoải mái.

+ Khả năng bảo trì và mở rộng: Sản phẩm có khả năng bảo trì và mở rộng sau này.

# 3. Thiết kế phần cứng

## 3.1. Giới thiệu tổng quan các thành phần phần cứng

### 3.1.1. Arduino Uno



Hình . Arduino Uno

Arduino Uno R3 là một board mạch vi điều khiển được phát triển bởi Arduino.cc, một nền tảng điện tử mã nguồn mở chủ yếu dựa trên vi điều khiển AVR Atmega328P. Với Arduino chúng ta có thể xây dựng các ứng dụng điện tử tương tác với nhau thông qua phần mềm và phần cứng hỗ trợ.

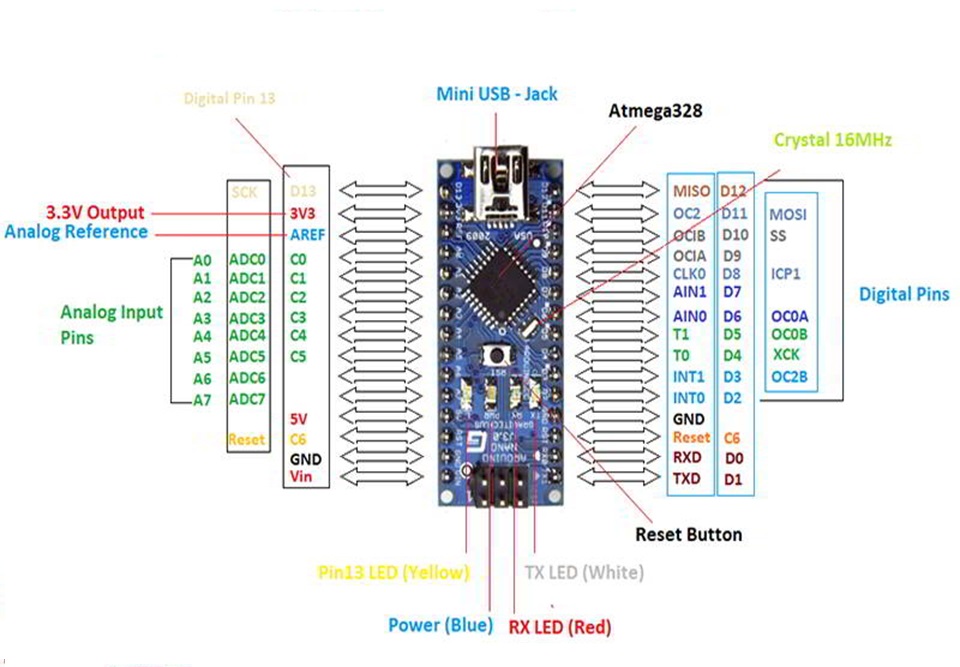
Thư viện mã nguồn mở của arduino ngày một tăng lên, giúp ích cho rất nhiều người mới biết đến Arduino cũng như những chuyên viên lập trình nhúng và chuyên gia cùng tham khảo và xây dựng tiếp nối. Để có thể học và sử dụng, từ đơn giản đến phức tạp bạn cần sử dụng ngôn ngữ lập trình Arduino dựa trên sơ đồ, hệ thống của bạn thiết kế, thông qua phần mềm Arduino IDE, để thực hiện những yêu cầu đó đưa về bộ phận xử lý trung tâm (Arduino).

Arduino Uno R3 được kết nối trực tiếp với máy tính thông qua USB để giao tiếp với phần mềm lập trình IDE, tương thích với Windows, MAC hoặc Linux Systems, tuy nhiên, Windows thích hợp hơn để sử dụng. Các ngôn ngữ lập trình như C và C ++ được sử dụng trong IDE. Ngoài USB, người dùng có thể dùng nguồn điện ngoài để cấp nguồn cho bo mạch

Thông số kỹ thuật:

* Chíp ATMEGA328P-PU
* Nguồn cấp: 7-12V
* Dòng Max chân 5V: 500mA
* Dòng Max 3.3V: 50mA
* Dòng Max Chân I/O: 30mA
* 14 Chân Digital I/O: (6 chân PWM)
* 6 Chân Analog Inputs
* 32k Flash Memory
* 16Mhz Clock Speed
* SRAM 2 KB
* EEPROM 1 KB
* Đèn LED: Arduino Uno đi kèm với đèn LED tích hợp được kết nối thông qua chân 13. Cung cấp mức logic HIGH tương ứng ON và LOW tương ứng tắt.
* Vin: Đây là điện áp đầu vào được cung cấp cho board mạch Arduino. Khác với 5V được cung cấp qua cổng USB. Pin này được sử dụng để cung cấp điện áp toàn mạch thông qua jack nguồn, thông thường khoảng 7-12VDC
* 5V: Chân 5V được sử dụng để cung cấp điện áp đầu ra. Arduino được cấp nguồn bằng ba cách đó là USB, chân Vin của bo mạch hoặc giắc nguồn DC.
* USB: Hỗ trợ điện áp khoảng 5V trong khi Vin và Power Jack hỗ trợ dải điện áp trong khoảng từ 7V đến 20V.
* GND: Chân mass chung cho toàn mạch Arduino
* Reset : Chân reset để thiết lập lại về ban đầu
* IOREF: Chân này rất hữu ích để cung cấp tham chiếu điện áp cho Arduino
* PWM: PWM được cung cấp bởi các chân 3,5,6,9,10, 11. Các chân này được cấu hình để cung cấp PWM đầu ra 8 bit.
* SPI: Chân này được gọi là giao diện ngoại vi nối tiếp. Các chân 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) cung cấp liên lạc SPI với sự trợ giúp của thư viện SPI.
* AREF: Chân này được gọi là tham chiếu tương tự, được sử dụng để cung cấp điện áp tham chiếu cho các đầu vào tương tự.
* TWI: Chân Giao tiếp TWI được truy cập thông qua thư viện dây. Chân A4 và A5 được sử dụng cho mục đích này.
* Serial Communication: Giao tiếp nối tiếp được thực hiện thông qua hai chân 0 (Rx) và 1 (Tx).
* Rx: Chân này được sử dụng để nhận dữ liệu trong khi chân Tx được sử dụng để truyền dữ liệu.
* External Interrupts (Ngắt ngoài): Chân 2 và 3 được sử dụng để cung cấp các ngắt ngoài.

### 3.1.2. Arduino Nano



Hình . Arduino nano

Arduino Nano USB Mini là board mạch sử dụng vi điều khiển ATmega328P hoặc ATmega168 tích hợp các chân I/O đơn giản nhỏ gọn dựa trên mã nguồn mở được phát triển bởi Arduino.cc, có lợi thế lớn về kích thước so với phiên bản Arduino Uno và Arduino Mega. Arduino Nano có thể hoạt động độc lập và tương tác hiệu quả với các thiết bị điện tử, cũng có thể giúp những người mới tìm hiểu về Arduino có thể kết nối với PC, phối hợp với Flash, xử lý, Max/Msp, PD,  và các phần mềm khác một cách dễ dàng. Điều này giúp Arduino Nano là sự lựa chọn ưa thích khi muốn thực hiện một projects mà yếu cầu kết nối với các thiết bị ngoại vi ít và đơn giản.

Các chức năng rất giống giống với phiên bản Arduino Uno nhưng kích thước nhỏ gọn hơn. Arduino Nano hoạt động với điện áp 5V. Tuy nhiên nguồn cấp điện áp đầu vào có thể thay đổi từ 7 đến 12V. Arduino Nano bo gồm tất cả 14 chân Digital, 8 chân Analog, 2 chân Reset và 6 chân nguồn.

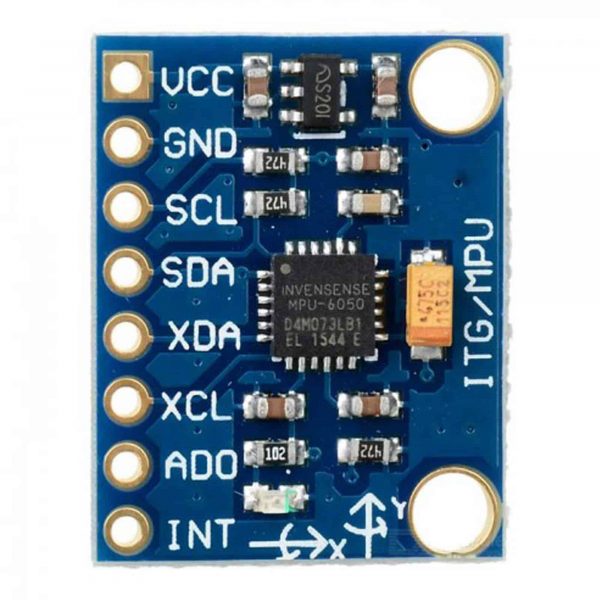
Thông số kỹ thuật:

* Chip: ATmega328P
* Điện áp logic: 5V
* Điện áp hoạt động: 7-12V
* Các chân I/O: 14 chân ( Bao gồm 6 chân PWM)
* Dòng diện I/O: 40mA
* Bộ nhớ Flash: 32Kb
* SRAM: 2Kb
* EEPROM: 1Kb
* Tần số dao động: 16Mhz

Chức năng các chân:

|  |  |  |  |
| --- | --- | --- | --- |
| Thứ tự chân | Tên Pin | Kiểu | Chức năng |
| 1 | D1/TX | I/O | Ngõ vào/ra số  Chân TX-truyền dữ liệu |
| 2 | D0/RX | I/O | Ngõ vào/ra số  Chân RX-nhận dữ liệu |
| 3 | RESET | Đầu vào | Chân reset, hoạt động ở mức thấp |
| 4 | GND | Nguồn | Chân nối mass |
| 5 | D2 | I/O | Ngõ vào/ra digital |
| 6 | D3 | I/O | Ngõ vào/ra digital |
| 7 | D4 | I/O | Ngõ vào/ra digital |
| 8 | D5 | I/O | Ngõ vào/ra digital |
| 9 | D6 | I/O | Ngõ vào/ra digital |
| 10 | D7 | I/O | Ngõ vào/ra digital |
| 11 | D8 | I/O | Ngõ vào/ra digital |
| 12 | D9 | I/O | Ngõ vào/ra digital |
| 13 | D10 | I/O | Ngõ vào/ra digital |
| 14 | D11 | I/O | Ngõ vào/ra digital |
| 15 | D12 | I/O | Ngõ vào/ra digital |
| 16 | D13 | I/O | Ngõ vào/ra digital |
| 17 | 3V3 | Đầu ra | Đầu ra 3.3V (từ FTDI) |
| 18 | AREF | Đầu vào | Tham chiếu ADC |
| 19 | A0 | Đầu vào | Kênh đầu vào tương tự kênh 0 |
| 20 | A1 | Đầu vào | Kênh đầu vào tương tự kênh 1 |
| 21 | A2 | Đầu vào | Kênh đầu vào tương tự kênh 2 |
| 22 | A3 | Đầu vào | Kênh đầu vào tương tự kênh 3 |
| 23 | A4 | Đầu vào | Kênh đầu vào tương tự kênh 4 |
| 24 | A5 | Đầu vào | Kênh đầu vào tương tự kênh 5 |
| 25 | A6 | Đầu vào | Kênh đầu vào tương tự kênh 6 |
| 26 | A7 | Đầu vào | Kênh đầu vào tương tự kênh 7 |
| 27 | +5V | Đầu ra hoặc đầu vào | + Đầu ra 5V (từ bộ điều chỉnh On-board) hoặc  + 5V (đầu vào từ nguồn điện bên ngoài) |
| 28 | RESET | Đầu vào | Chân đặt lại, hoạt động ở mức thấp |
| 29 | GND | Nguồn | Chân nối mass |
| 30 | VIN | Nguồn | Chân nối với nguồn vào |

### 3.1.3. Cảm biến gia tốc MPU6050



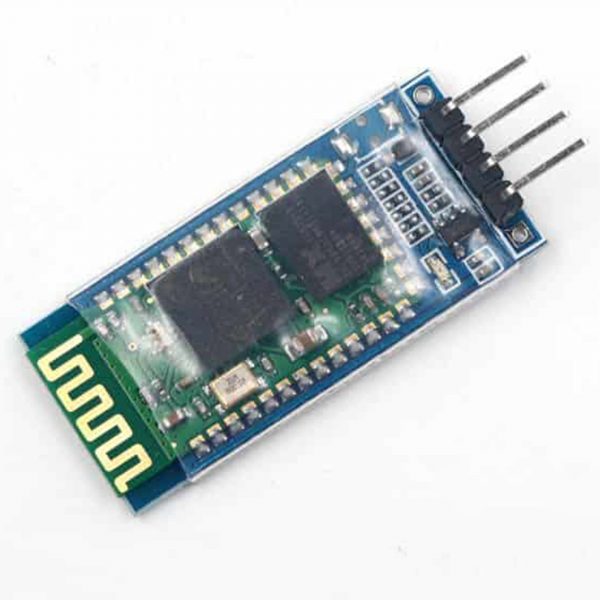
Hình . MPU6050

Cảm biến gia tốc GY-521 6DOF IMU MPU6050 được sử dụng để đo 6 thông số: 3 trục Góc quay (Gyro), 3 trục gia tốc hướng (Accelerometer), là loại cảm biến gia tốc phổ biến nhất trên thị trường hiện nay, ví dụ và code dành cho nó rất nhiều và hầu như có trên mọi loại vi điều khiển.

Thông số kỹ thuật:

* Điện áp sử dụng: 3~5VDC
* Điện áp giao tiếp: 3~5VDC
* Chuẩn giao tiếp: I2C
* Giá trị Gyroscopes trong khoảng: +/- 250 500 1000 2000 degree/sec
* Giá trị Acceleration trong khoảng: +/- 2g, +/- 4g, +/- 8g, +/- 16g
* Board mạch mạ vàng, linh kiện hàn tự động bằng máy chất lượng tốt nhất.

### 3.1.4. Bluetooth HC-05



Hình . HC-05

Module thu phát Bluetooth HC-05 dùng để thiết lập kết nối Serial giữa 2 thiết bị bằng sóng bluetooth. Điểm đặc biệt của module bluetooth HC-05 là module có thể hoạt động được ở 2 chế độ: MASTER hoặc SLAVE.

- Ở chế độ SLAVE: bạn cần thiết lập kết nối từ smartphone, laptop, usb bluetooth để dò tìm module sau đó pair với mã PIN là 1234. Sau khi pair thành công, bạn đã có 1 cổng serial từ xa hoạt động ở baud rate 9600.

- Ở chế độ MASTER: module sẽ tự động dò tìm thiết bị bluetooth khác (1 module bluetooth HC-06, usb bluetooth, bluetooth của laptop…) và tiến hành pair chủ động mà không cần thiết lập gì từ máy tính hoặc smartphone.

Module Bluetooth thu phát HC-05 được thiết kế nhỏ gọn ra chân tín hiệu giao tiếp cơ bản và nút bấm để vào chế độ AT COMMAND, mạch được thiết kế để có thể cấp nguồn và giao tiếp qua 3.3VDC hoặc 5VDC, thích hợp cho nhiều ứng dụng khác nhau: Robot Bluetooth, điều khiển thiết bị qua Bluetooth,….

Thông số kỹ thuật:

* Điện áp hoạt động: 3.3 ~ 5VDC
* Mức điện áp chân giao tiếp: TTL tương thích 3.3VDC và 5VDC.
* Dòng điện khi hoạt động: khi Pairing 30 mA, sau khi pairing hoạt động truyền nhận bình thường 8 mA.
* Baudrate UART có thể chọn được: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
* Cấu hình: Bluetooth serial port (master và slave)
* Giao thức bluetooth: Bluetooth v2.0 + EDR
* Tần số: 2.4 GHz ISM band
* Điều chế: GFSK
* Nguồn: =4 dBm, class 2
* Kích thước: 15.2 x 35.7 x 5.6mm

Cách kết nối 2 bluetooth HC-05 version 3.0-20170601:

Lưu ý: HC-05 version 3.0-20170601 có hai chế độ AT mode, khác với HC-05 version 2 chỉ có một chế độ AT mode.

a) Ở chế độ AT mode thứ nhất (Đèn led trên HC-05 ở chế độ nháy chậm):

- Cách vào chế độ AT mode thứ nhất:

+ Chân RX, TX trên HC-05 nối tương ứng với chân TX, RX trên Arduino Uno hoặc Arduino Nano; chân GND, +5V trên HC-05 nối tương ứng với GND và 3.3V trên Arduino Uno và Arduino Nano.

+ Source code dùng cho HC-05 (có thể dùng cho cả Master và Slave): hc-05.ino

+ Nhấn giữ nút Reset trên HC-05 đồng thời rút chân +5V của HC-05 nối với chân nguồn 3.3V trên mạch ra, sau đó cắm lại chân nguồn vừa rút trong lúc vẫn nhấn giữ nút Reset. Lúc này, đèn led trên HC-05 đã nháy chậm lại, HC-05 đã vào chế độ AT mode thứ nhất. Để chắc chắn, mở Serial Monitor lên và kiểm tra bằng cách gõ lệnh AT, nếu kết quả là OK thì chế độ AT mode đã được thiết lập đúng.

- Trong chế độ này, ta thiết lập các thông số cần thiết cho HC-05 trước khi tiến vào chế độ AT mode thứ hai để kết nối với nhau:

+ AT+NAME=”…”: thiết lập tên cho thiết bị nếu cần thiết để kiểm tra.

+ AT+RMAAD: xóa các kết nối trước đó

+ AT+UART: kiểm tra thông số baudrate được cấu hình, nếu cần chỉnh sửa dùng command AT+UART=?,?,? với ? được thay bằng thông số phù hợp.

+ AT+CMODE=1: chấp nhận mọi loại kết nối

+ AT+ROLE=?: ? tương ứng với 1 nếu đó là master, 0 nếu đó là slave.

+ AT+INQM=0,5,5: thiết lập vào chế độ inquiry standard mode, với tối đa 5 kết nối với timeout tối đa là 5s.

+ AT+PSWD=”…”: thiết lập mật khẩu chung tùy ý trên 2 bluetooth HC-05.

+ AT+ADDR?: kiểm tra địa chỉ MAC trên 2 thiết bị HC-05, dùng để thiết lập kết nối sau đó.

b) Ở chế độ AT mode thứ hai (Đèn led trên HC-05 ở chế độ nháy nhanh như bình thường): Thiết lập kết nối cho 2 bluetooth HC-05 ở chế độ này.

- Trong Serial Monitor, dùng lệnh AT+STATE để kiểm tra trạng thái của HC-05, cả 2 bluetooth Master và Slave nên ở trạng thái INQUIRING.

- Trên Serial Monitor của slave hiện lên chuỗi các địa chỉ có dạng xyz:xy:xyzw,931F00,7FFF. Trong đó địa chỉ là chuỗi xyz:xy:xyzw.

Để sử dụng địa chỉ, chuyển “:” thành “,”.

Kiểm tra lại tên thiết bị đang kết nối bằng lệnh AT+RNAME? xyz,xy,xyzw: tên thiết bị là tên thiết bị đang được kết nối.

- Kết nối 2 bluetooth HC-05, trên Serial Monitor của master, sử dụng các lệnh

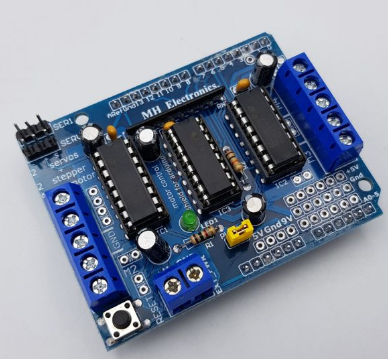
+ AT+PAIR= xyz,xy,xyzw,5: xyz,xy,xyzw là địa chỉ của slave

+ AT+BIND= xyz,xy,xyzw

+ AT+LINK= xyz,xy,xyzw

- Lúc này, 2 bluetooth HC-05 đã kết nối được với nhau, kiểm tra lại bằng lệnh: AT+STATE?, lúc này trạng thái của 2 thiết bị là PAIRED hoặc CONNECTED. Lúc này có thể tiến hành gửi nhận thông điệp giữa 2 bluetooth HC-05.

### 3.1.5. Motor Shield L293D



Hình . L293D

Arduino Motor Shield L293D là một phần board mở rộng cho các board arduino, dùng để điều khiển các loại động cơ DC, động cơ bước và động cơ servo. Arduino Motor Shield được thiết kế gọn gàng, đẹp mắt và tương thích hoàn toàn với các board Arduino: arduino uno r3, arduino leonardo, arduino mega2560, có thể sử dụng và điều khiển một cách dễ dàng và nhanh chóng.

Arduino Motor Shield sử dụng 2 IC cầu H L293D hoàn chỉnh với các chế độ bảo vệ và 1 IC logic 74HC595 để điều khiển các động cơ.

Arduino Motor Shield L293D có thể điều khiển nhiều loại motor khác nhau như step motor, servo motor, motor DC, với mức áp lên đến 36V, dòng tối đa 600mA cho mỗi kênh điều khiển.

Thông số kỹ thuật:

Chân điều khiển 2 RC servo được kết nối với chân số 9 và 10. Nguồn cung cấp được lấy trực tiếp từ board Arduino.

* Motor 1 nối với chân 11
* Motor 2 nối với chân 3
* Motor 3 nối với chân 5
* Motor 4 nối với chân 6
* Chân 4, 7, 8, 12 dùng điều khiển motor thông qua IC 74HC595

Ngoài ra để tiện cho việc sử dụng nguồn cắm ngoài, trên Arduino Motor Shield sử dụng 1 jumper PWR mục đích để lấy nguồn ngoài thông qua jack DC của board arduino để cung cấp nguồn cho motor hoạt động. Nếu trong trường hợp chúng ta không sử dụng jumper này thỉ phải cấp 1 nguồn riêng vào chân EXT\_PWR để cấp nguồn cho motor hoạt động.

## 3.2. Thiết bị điều khiển

### 3.2.1. Sơ đồ thiết kế

Ảnh có chứa mạch, đồng hồ

Mô tả được tạo tự động

Hình . Sơ đồ mạch thiết bị điều khiển

Bao gồm:

* 1 Arduino Nano
* 1 cảm biến gia tốc MPU6050
* 1 Bluetooth HC-05 để gửi tín hiệu
* 1 nút bấm
* 1 điện trở
* 2 đèn LED

### 3.2.2. Sơ đồ nguyên lý hoạt động

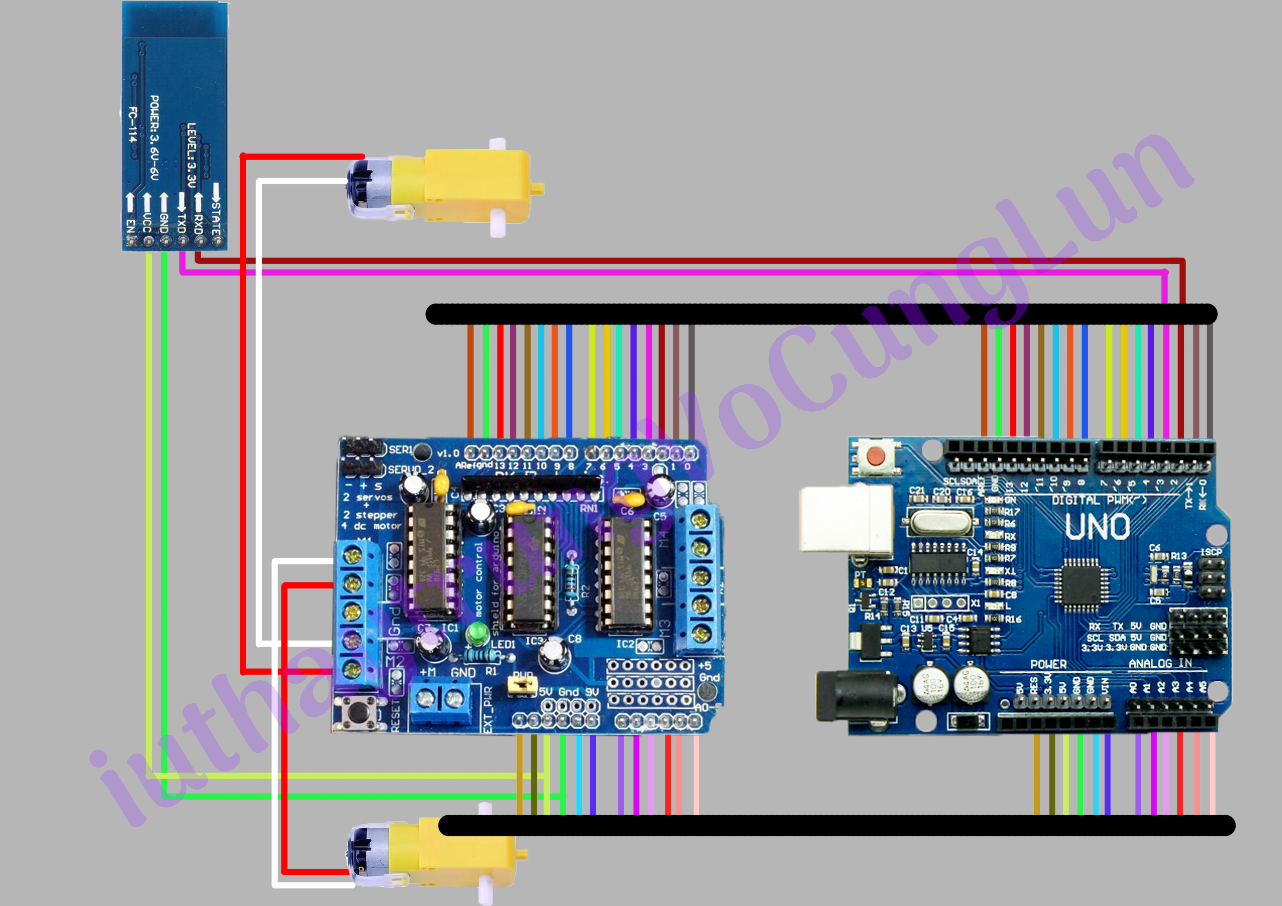
Ảnh có chứa mạch

Mô tả được tạo tự động

Hình . Sơ đồ nguyên lý hoạt động thiết bị điều khiển

## 3.3. Xe điều khiển

### 3.3.1. Sơ đồ thiết kế



Hình . Sơ đồ mạch điều khiển

Bao gồm:

* 1 Arduino Uno
* 1 motor Shield L293D
* 1 module bluetooth HC-05
* 2 motor giảm tốc 3V - 9V

### 3.3.2. Sơ đồ nguyên lý hoạt động

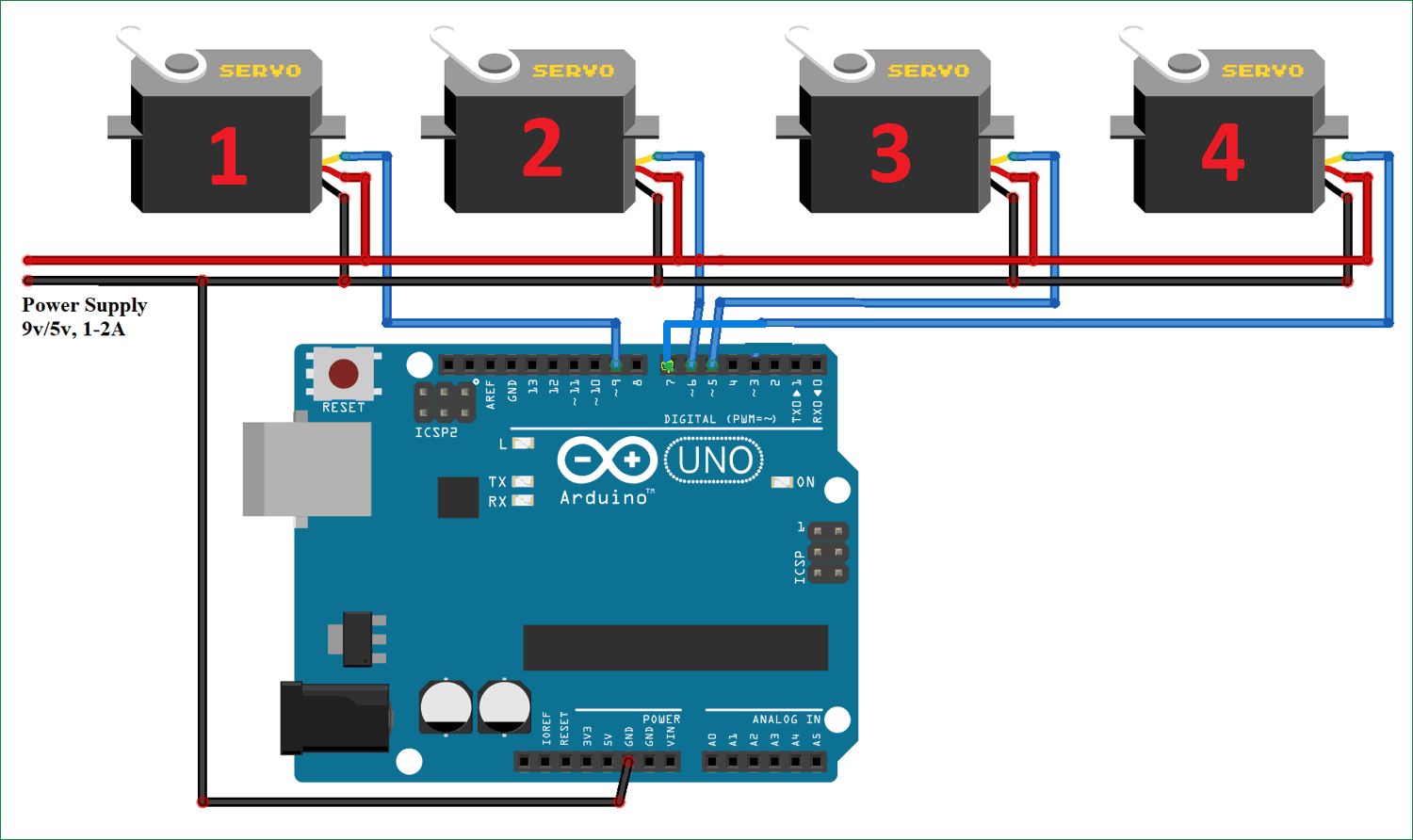
Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Hình . Nguyên lý hoạt động xe điều khiển

## 3.4. Cánh tay robot

### 3.4.1. Sơ đồ thiết kế



Hình . Sơ đồ mạch cánh tay Robot

### 3.4.2. Nguyên lý hoạt động

* Tại mode 1: Motor 1 và Motor 2 hoạt động
* Tại mode 2: Motor 1 và Motor 3 hoạt động
* Tại mode 3: Motor 1 và Motor 4 hoạt động

# 4. Thiết kế phần mềm

## 4.1. Xe điều khiển:

Chương trình chính của xe, nhận tín hiệu điều khiển và thực hiện:

|  |
| --- |
| #include <SoftwareSerial.h>  #include "DriveMotor.h"  #define GOING\_FORWARD 'F'  #define GOING\_FORWARD\_SLOW 'f'  #define GOING\_FORWARD\_FAST 'w'  #define GOING\_BACKWARD 'B'  #define GOING\_BACKWARD\_SLOW 'b'  #define GOING\_BACKWARD\_FAST 's'  #define TURNING\_LEFT 'L'  #define TURNING\_LEFT\_SLOW 'l'  #define TURNING\_LEFT\_FAST 'a'  #define TURNING\_RIGHT 'R'  #define TURNING\_RIGHT\_SLOW 'r'  #define TURNING\_RIGHT\_FAST 'd'  #define STOP 'S'  SoftwareSerial BTserial(2, 3); // RX | TX  char d = 0;  int mode = 0;  void setup() {  Serial.begin(38400);  Serial.println("Arduino is ready");  // HC-05 default serial speed for AT mode is 38400  BTserial.begin(38400);  motorSetup();  mode = 0;  }  void loop() {  if (BTserial.available() > 0) {  d = BTserial.read();  Serial.println(mode);  Serial.println(d);  if (d >= '0' && d < '4') {  mode = (int)(d - '0');  } else if (d == GOING\_FORWARD) {  goForward(mode, 1);  } else if (d == GOING\_FORWARD\_SLOW){  goForward(mode, 0);  } else if (d == GOING\_FORWARD\_FAST){  goForward(mode, 2);  }  else if (d == TURNING\_RIGHT) {  turnRight(mode, 1);  } else if (d == TURNING\_RIGHT\_SLOW){  turnRight(mode, 0);  } else if (d == TURNING\_RIGHT\_FAST){  turnRight(mode, 2);  }  else if (d == GOING\_BACKWARD){  goBackward(mode, 1);  } else if (d == GOING\_BACKWARD\_SLOW){  goBackward(mode, 0);  } else if (d == GOING\_BACKWARD\_FAST){  goBackward(mode, 2);  }  else if (d == TURNING\_LEFT) {  turnLeft(mode, 1);  } else if (d == TURNING\_LEFT\_SLOW){  turnLeft(mode, 0);  } else if (d == TURNING\_LEFT\_FAST){  turnLeft(mode, 2);  }  else if (d == STOP) {  stopCar();  }  Serial.println(mode);  }  } |

## 4.2. Cánh tay robot

Đoạn chương trình điều khiển cánh tay robot:

|  |
| --- |
| // 4DOF ARM CONTROLLER library  // copyright iuthaytrungVoCungLun @ 2020  // this code is private domain, do not use  // without our permission!!  #include <Servo.h>  //||---------------------------------------||  //||---------------------------------------||  //||---------CODE FOR HAND CONFIG----------||  //||---------------------------------------||  //||---------------------------------------||  // PIN CONFIGURATION  #define JAW\_SERVO\_PIN 7  #define HEIGHT\_SERVO\_PIN 5  #define RANGE\_SERVO\_PIN 6  #define ROTATE\_SERVO\_PIN 9  // JAW SERVO CONFIGURATION  #define MAX\_JAW\_SERVO 180  #define MIN\_JAW\_SERVO 80  // HEIGHT SERVO CONFIGURATION  #define MAX\_HEIGHT\_SERVO 180  #define MIN\_HEIGHT\_SERVO 0  // RANGE SERVO CONFIGURATION  #define MAX\_RANGE\_SERVO 180  #define MIN\_RANGE\_SERVO 0  // ROTATE SERVO CONFIGURATION  #define MAX\_ROTATE\_SERVO 180  // 4DOF ARM CONTROLLER library  // copyright iuthaytrungVoCungLun @ 2020  // this code is private domain, do not use  // without our permission!!  #include <Servo.h>  //||---------------------------------------||  //||---------------------------------------||  //||---------CODE FOR HAND CONFIG----------||  //||---------------------------------------||  //||---------------------------------------||  // PIN CONFIGURATION  #define JAW\_SERVO\_PIN 7  #define HEIGHT\_SERVO\_PIN 5  #define RANGE\_SERVO\_PIN 6  #define ROTATE\_SERVO\_PIN 9  // JAW SERVO CONFIGURATION  #define MAX\_JAW\_SERVO 180  #define MIN\_JAW\_SERVO 80  // HEIGHT SERVO CONFIGURATION  #define MAX\_HEIGHT\_SERVO 180  #define MIN\_HEIGHT\_SERVO 0  // RANGE SERVO CONFIGURATION  #define MAX\_RANGE\_SERVO 180  #define MIN\_RANGE\_SERVO 0  // ROTATE SERVO CONFIGURATION  #define MAX\_ROTATE\_SERVO 180  #define MIN\_ROTATE\_SERVO 0  //||---------------------------------------||  //||---------------------------------------||  //||-------------HAND VARIABLE-------------||  //||---------------------------------------||  //||---------------------------------------||  Servo jawServo;  Servo heightServo;  Servo rangeServo;  Servo rotateServo;  // variable to store the servo position  int jaw\_pos = 100;  int height\_pos = 120;  int range\_pos = 40;  int rotate\_pos = 90;  void initState() {  jawServo.write(jaw\_pos);  jawServo.attach(JAW\_SERVO\_PIN);  heightServo.write(height\_pos);  heightServo.attach(HEIGHT\_SERVO\_PIN);  rangeServo.write(range\_pos);  rangeServo.attach(RANGE\_SERVO\_PIN);  rotateServo.write(rotate\_pos);  rotateServo.attach(ROTATE\_SERVO\_PIN);  }  //||---------------------------------------||  //||---------------------------------------||  //||---------CODE FOR HAND ACTION----------||  //||---------------------------------------||  //||---------------------------------------||  void rangeDecrease(int speed\_mode) {  if (range\_pos <= MIN\_RANGE\_SERVO) {  rangeServo.write(MIN\_RANGE\_SERVO);  return;  } else {  int i;  for(i = 0; i < speed\_mode; i++){  range\_pos--;  rangeServo.write(range\_pos);  }  }  }  void rangeIncrease(int speed\_mode) {  if (range\_pos >= MAX\_RANGE\_SERVO) {  rangeServo.write(MAX\_RANGE\_SERVO);  return;  } else {  int i;  for(i = 0; i < speed\_mode; i++){  range\_pos++;  rangeServo.write(range\_pos);  }  }  }  void heightIncrease(int speed\_mode) {  if (height\_pos >= MAX\_HEIGHT\_SERVO) {  heightServo.write(MAX\_HEIGHT\_SERVO);  return;  } else {  int i;  for(i = 0; i < speed\_mode; i++){  height\_pos++;  heightServo.write(height\_pos);  }  }  }  void heightDecrease(int speed\_mode) {  if (height\_pos <= MIN\_HEIGHT\_SERVO) {  heightServo.write(MIN\_HEIGHT\_SERVO);  return;  } else {  int i;  for(i = 0; i < speed\_mode; i++){  height\_pos--;  heightServo.write(height\_pos);  }  }  }  void rotateRight(int speed\_mode) {  if (rotate\_pos <= MIN\_ROTATE\_SERVO) {  rotateServo.write(MIN\_ROTATE\_SERVO);  return;  } else {  int i;  for(i = 0; i < speed\_mode; i++){  rotate\_pos--;  rotateServo.write(rotate\_pos);  }  }  }  void rotateLeft(int speed\_mode) {  if (rotate\_pos >= MAX\_ROTATE\_SERVO) {  rotateServo.write(MAX\_ROTATE\_SERVO);  return;  } else {  int i;  for(i = 0; i < speed\_mode; i++){  rotate\_pos++;  rotateServo.write(rotate\_pos);  }  }  }  void jawBind(int speed\_mode) {  if (jaw\_pos <= MIN\_JAW\_SERVO) {  jawServo.write(MIN\_JAW\_SERVO);  return;  } else {  int i;  for(i = 0; i < speed\_mode; i++){  jaw\_pos--;  jawServo.write(jaw\_pos);  }  }  }  void jawRelease(int speed\_mode) {  if (jaw\_pos >= MAX\_JAW\_SERVO) {  jawServo.write(MAX\_JAW\_SERVO);  return;  } else {  int i;  for(i = 0; i < speed\_mode; i++){  jaw\_pos++;  jawServo.write(jaw\_pos);  }  }  } |

## 4.3. Thiết bị điều khiển

Đoạn chương trình gửi tín hiệu điều khiển:

|  |
| --- |
| #define RADIAN 180/M\_PI  #define GOING\_FORWARD 'F'  #define GOING\_FORWARD\_SLOW 'f'  #define GOING\_FORWARD\_FAST 'w'  #define GOING\_BACKWARD 'B'  #define GOING\_BACKWARD\_SLOW 'b'  #define GOING\_BACKWARD\_FAST 's'  #define TURNING\_LEFT 'L'  #define TURNING\_LEFT\_SLOW 'l'  #define TURNING\_LEFT\_FAST 'a'  #define TURNING\_RIGHT 'R'  #define TURNING\_RIGHT\_SLOW 'r'  #define TURNING\_RIGHT\_FAST 'd'  #define STOP 'S'  int turn\_angle; // to make a decision whether to turn left or right  int move\_direction; // to make a decision whether to go forward or backward  mpu.dmpGetQuaternion(&q, fifoBuffer);  mpu.dmpGetGravity(&gravity, &q);  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);  turn\_angle = ypr[1] \* RADIAN;  move\_direction = ypr[2] \* RADIAN;  Serial.print("ypr\t");  Serial.print(ypr[0] \* RADIAN);  Serial.print("\t");  Serial.print(ypr[1] \* RADIAN);  Serial.print("\t");  Serial.println(ypr[2] \* RADIAN);  // For turning right  if (turn\_angle <= -15 && turn\_angle >= -30){  BTSerial.write(TURNING\_RIGHT\_SLOW);  Serial.write(TURNING\_RIGHT\_SLOW);  } else if (turn\_angle < -30 && turn\_angle >= -50){  BTSerial.write(TURNING\_RIGHT);  Serial.write(TURNING\_RIGHT);  } else if (turn\_angle < -50){  BTSerial.write(TURNING\_RIGHT\_FAST);  Serial.write(TURNING\_RIGHT\_FAST);  }  // For turning left  else if (turn\_angle >= 15 && turn\_angle <= 30){  BTSerial.write(TURNING\_LEFT\_SLOW);  Serial.write(TURNING\_LEFT\_SLOW);  } else if (turn\_angle > 30 && turn\_angle <= 50){  BTSerial.write(TURNING\_LEFT);  Serial.write(TURNING\_LEFT);  } else if (turn\_angle > 50){  BTSerial.write(TURNING\_LEFT\_FAST);  Serial.write(TURNING\_LEFT\_FAST);  }  // For going forward  else if (move\_direction <= -15 && move\_direction >= -25){  BTSerial.write(GOING\_FORWARD\_SLOW);  Serial.write(GOING\_FORWARD\_SLOW);  } else if (move\_direction < -25 && move\_direction >= -40){  BTSerial.write(GOING\_FORWARD);  Serial.write(GOING\_FORWARD);  } else if (move\_direction < -40){  BTSerial.write(GOING\_FORWARD\_FAST);  Serial.write(GOING\_FORWARD\_FAST);  }  // For going backward  else if (move\_direction >= 15 && move\_direction <= 25) {  BTSerial.write(GOING\_BACKWARD\_SLOW);  Serial.write(GOING\_BACKWARD\_SLOW);  } else if (move\_direction > 25 && move\_direction <= 40) {  BTSerial.write(GOING\_BACKWARD);  Serial.write(GOING\_BACKWARD);  } else if (move\_direction > 40) {  BTSerial.write(GOING\_BACKWARD\_FAST);  Serial.write(GOING\_BACKWARD\_FAST);  }  else {  BTSerial.write(STOP);  Serial.write(STOP);  } |

Đoạn chương trình để chọn mode điều khiển:

|  |
| --- |
| char modeArr[] = {'0', '1', '2', '3'};  int x = 1;  const byte ledPin1 = 10;  const byte ledPin2 = 6;  const byte interruptPin = 3;  volatile byte state1 = HIGH;  volatile byte state2 = HIGH;  int mode = 0;  //char controlCar = 'c';  //char controlArm = 'a';  void handleInterrupt() {  static unsigned long long last\_interrupt\_time = 0;  unsigned long long interrupt\_time = millis();  if (interrupt\_time - last\_interrupt\_time > 200) {  mode++;  if (mode == 4) mode = 0;  switch(mode){  case 0:  {  state1=LOW;  state2=LOW;  break;  }  case 1:  {  state1=LOW;  state2=HIGH;  break;  }  case 2:  {  state1=HIGH;  state2=LOW;  break;  }  case 3:  {  state1=HIGH;  state2=HIGH;  break;  }  }  for (int i = 0; i < 20; i++) {  Serial.println(modeArr[mode]);  BTSerial.write(modeArr[mode]);  }  }  last\_interrupt\_time = interrupt\_time;  }  void setup() {  Serial.begin(115200);  BTSerial.begin(38400);  pinMode(ledPin1, OUTPUT);  pinMode(ledPin2, OUTPUT);  pinMode(interruptPin, INPUT\_PULLUP);  attachInterrupt(digitalPinToInterrupt(interruptPin), handleInterrupt, LOW);  […]  }  void loop() {  digitalWrite(ledPin, state);  […]  } |

# 5. Kết quả

## 5.1. Kết quả đạt được:

Xe điều khiển có tổng cộng 4 mode điều khiển phụ thuộc vào cảm biến gia tốc gắn trên thiết bị điều khiển.Người dùng có thể chuyển mode để điều khiển chức năng mong muốn qua việc ấn nút ở trên thiết bị điều khiển. Mode điều khiển tương ứng sẽ được hiển thị bởi hai đèn led. Cụ thể:

+ Mode = 0: Điều khiển chuyển động của xe.

+ Mode = 1: Điều khiển chuyển động gắp thả vật và chuyển động quay của cánh tay.

+ Mode = 2: Điều khiển chuyển động nâng lên hạ xuống và chuyển động quay của cánh tay.

+ Mode = 3: Điều khiển chuyển động vươn ra hoặc thu vào và chuyển động quay của cánh tay.

Đồng thời, khi điều khiển xe và cánh tay, tùy vào độ nghiêng các hướng nhiều hay ít của cảm biến gia tốc mà tốc độ của xe và cánh tay nhanh hay chậm.

## 5.2. Hạn chế

Sản phẩm vẫn còn tồn tại một số hạn chế, cụ thể:

+ Hiện tượng nảy phím: nút bấm điều khiển chuyển chế độ cần được bấm giữ đủ lâu để chuyển được chế độ điều khiển, nếu bấm quá nhanh thì không nhận được tín hiệu chuyển chế độ điều khiển.

+ Sự phân hóa tốc độ xe: Tốc độ nhanh chậm của xe phân hóa chưa rõ ràng.

+ Sự không ổn định của cánh tay: Cánh tay gắp thả vật của sản phẩm còn hơi giật/ rung lắc nhẹ trong quá trình vận hành và điều khiển. Đồng thời, nếu không nhận được tín hiệu chuyển chế độ do hiện tượng nảy phím, cánh tay có tần suất rung lắc khá lớn.

# 6. Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | Arduino.cc, [Online]. Available: https://www.arduino.cc/. |
| [2] | Shubham, "Blog Suresh," [Online]. |