**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**



**ĐỒ ÁN CHUYÊN NGÀNH**

**Đề tài:**

**TÌM HIỂU VỀ INTERNET OF THINGS**

**VÀ ỨNG DỤNG**

Giảng viên hướng dẫn: ThS. Phạm Thi Vương

Sinh viên thực hiện: Nguyễn Thị Hằng 13520244

Nguyễn Văn Quang 13520675

***TP. Hồ Chí Minh, ngày 15 tháng 06 năm 2017***

**LỜI MỞ ĐẦU**

Trong thời gian gần đây chúng ta rất hay thấy một cụm từ là lạ xuất hiện trên mạng, đó là "Internet of Things". Đây là một khái niệm còn khá mới mẻ với hầu hết người dùng thông thường mặc dù nó đã được ra đời cách đây khá lâu. Nói ngắn gọi, nó là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó.

Internet of Things (IoT) là xu hướng đang được các doanh nghiệp trong lĩnh vực công nghệ quan tâm và đầu tư nghiên cứu. Cuộc đua IoT đã và đang diễn ra mạnh mẽ giữa các doanh nghiệp trên toàn thế giới.

Có thể nói, IoT mang lại rất nhiều cơ hội và song song với đó là không ít thách thức kèm theo. Nắm bắt được xu hướng, chúng em – sinh viên Khoa Công nghệ phần mềm quyết tâm tìm hiểu và bước vào sân chơi này, để có những trải nghiệm và tạo ra bước tiến mới cho sự phát triển IoT nói riêng cũng như ngành Công nghệ thông tin của nước ta.

**LỜI CẢM ƠN**

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô, gia đình và bạn bè.

Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý Thầy Cô ở Khoa Công Nghệ Phần Mềm – Trường Đại Học Công Nghệ Thông Tin đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Và đặc biệt, trong học kỳ này, Khoa đã tổ chức cho chúng em được tiếp cận với môn học mà theo em là rất hữu ích đối với sinh viên ngành Công Nghệ Phần Mềm. Đó là môn học “Đồ án chuyên ngành”.

Em xin chân thành cảm ơn ThS. Phạm Thi Vương đã tận tâm hướng dẫn chúng em qua từng buổi học. Nếu không có những lời hướng dẫn của thầy cô thì em nghĩ sản phẩm này chúng em rất khó có thể hoàn thiện được. Một lần nữa, chúng em xin chân thành cảm ơn thầy cô.

Bước vào lĩnh vực mới – IoT, kiến thức của chúng em còn hạn chế và còn nhiều bỡ ngỡ. Do vậy, không tránh khỏi những thiếu sót là điều chắc chắn, em rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy Cô và các bạn học cùng lớp để kiến thức của em trong lĩnh vực này được hoàn thiện hơn.

Xin chân thành cảm ơn!

**MỤC LỤC**

[MỤC LỤC 5](#_Toc485286915)

[PHẦN 1: INTERNET OF THINGS 7](#_Toc485286916)

[1.1. Khái niệm 7](#_Toc485286917)

[1.2. Thuật ngữ 7](#_Toc485286918)

[1.3. Lịch sử 7](#_Toc485286919)

[1.4. Khả năng định danh độc nhất 8](#_Toc485286920)

[1.5. Xu hướng và tính chất 8](#_Toc485286921)

[1.5.1. Thông minh 8](#_Toc485286922)

[1.5.2. Kiến trúc dựa trên sự kiện 9](#_Toc485286923)

[1.5.3. Là một hệ thống phức tạp 9](#_Toc485286924)

[1.5.4. Kích thước 9](#_Toc485286925)

[1.5.5. Vấn đề không gian, thời gian 9](#_Toc485286926)

[1.6. Các hệ thống phụ 10](#_Toc485286927)

[1.7. Ứng dụng 10](#_Toc485286928)

[PHẦN 2: ARDUINO 12](#_Toc485286929)

[2.1. Giới thiệu các mạch, cảm biến và module sử dụng 12](#_Toc485286930)

[2.1.1. Arduino UNO R3 12](#_Toc485286931)

[2.1.2. Arduino Motor Shield L293D 16](#_Toc485286932)

[2.1.3. Cảm biến siêu âm SRF05 17](#_Toc485286933)

[2.1.4. Module thu phát hồng ngoại HX1838 19](#_Toc485286934)

[2.2. IDE hỗ trợ 20](#_Toc485286935)

[2.2.1. Arduino IDE 21](#_Toc485286936)

[2.2.2. Processing Arduino IDE 26](#_Toc485286937)

[2.3. Thư viện sử dụng 28](#_Toc485286938)

[2.3.1. IRremote 28](#_Toc485286939)

[2.3.2. Adafruit Motor Shield V2 29](#_Toc485286940)

[2.3.3. NewPing 29](#_Toc485286941)

[2.4. Cách thực hiện 30](#_Toc485286942)

[2.4.1. Hello world 30](#_Toc485286943)

[2.4.2. Điều khiển led bằng remote 31](#_Toc485286944)

[2.4.3. Điều khiển động cơ xe bằng remote 32](#_Toc485286945)

[2.4.4. Làm radar dò vật cản 33](#_Toc485286946)

[2.4.5. Xe tự động 35](#_Toc485286947)

[PHỤ LỤC 1: SOURCE CODE 38](#_Toc485286948)

[(1) Hello World 38](#_Toc485286949)

[(2) Điều khiển led bằng remote 39](#_Toc485286950)

[(3) Điều khiển động cơ xe bằng remote 40](#_Toc485286951)

[(4) Làm radar dò vật cản 42](#_Toc485286952)

[(4.1) Code Arduino IDE 42](#_Toc485286953)

[(4.2) Code Processing 43](#_Toc485286954)

[(5) Xe tự động 46](#_Toc485286955)

[PHỤ LỤC 2: TÀI LIỆU THAM KHẢO 50](#_Toc485286956)

# **PHẦN 1: INTERNET OF THINGS**

1. **Khái niệm**

Internet of Things (viết tắt là IoT) là một kịch bản của thế giới, khi mà mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. IoT đã phát triển từ sự hội tụ của công nghệ không dây, công nghệ vi cơ điện tử và Internet.

Nói đơn giản là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó.

Một vật trong IoT có thể là một người với một trái tim cấy ghép; một động vật ở trang trại với bộ chip sinh học; một chiếc xe với bộ cảm ứng tích hợp cảnh báo tài xế khi bánh xe xẹp hoặc bất kỳ vật thể tự nhiên hay nhân tạo nào mà có thể gán được một địa chỉ IP và cung cấp khả năng truyền dữ liệu thông qua mạng lưới. Cho đến nay, IoT là những liên kết máy-đến-máy (M2M) trong ngành sản xuất, công nghiệp năng lượng, kỹ nghệ xăng dầu. Khả năng sản phẩm được tích hợp máy-đến-máy thường được xem như là thông minh.

1. **Thuật ngữ**

IoT là thuật ngữ dùng để chỉ các đối tượng có thể được nhận biết (identifiable) cũng như chỉ sự tồn tại của chúng trong một kiến trúc mang tính kết nối. Cụm từ này được đưa ra bởi Kevin Ashton vào năm 1999. Ông là một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT, nơi thiết lập các quy chuẩn toàn cầu cho RFID (một phương thức giao tiếp không dây dùng sóng radio) cũng như một số loại cảm biến khác. IoT sau đó cũng được dùng nhiều trong các ấn phẩm đến từ các hãng và nhà phân tích.

1. **Lịch sử**

Thực tế, Internet of things đã manh nha từ nhiều thập kỹ trước. Tuy nhiên mãi đến năm 1999 cụm từ IOT mới được đưa ra bởi Kevin Ashton , Ông là một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT, nơi thiết lập các quy chuẩn toàn cầu cho RFID (một phương thức giao tiếp không dây dùng sóng radio) cũng như một số loại cảm biến khác.

1. **Khả năng định danh độc nhất**

Điểm quan trọng của IoT đó là các đối tượng phải có thể được nhận biết và định dạng (identifiable). Nếu mọi đối tượng, kể cả con người, được "đánh dấu" để phân biệt bản thân đối tượng đó với những thứ xung quanh thì chúng ta có thể hoàn toàn quản lý được nó thông qua máy tính.

Việc đánh dấu (tagging) có thể được thực hiện thông qua nhiều công nghệ, chẳng hạn như RFID, NFC, mã vạch, mã QR, watermark kĩ thuật số... Việc kết nối thì có thể thực hiện qua Wi-Fi, mạng viễn thông băng rộng (3G, 4G), Bluetooth, ZigBee, hồng ngoại...

Ngoài những kĩ thuật nói trên, nếu nhìn từ thế giới web, chúng ta có thể sử dụng các địa chỉ độc nhất để xác định từng vật, chẳng hạn như địa chỉ IP. Mỗi thiết bị sẽ có một IP riêng biệt không nhầm lẫn. Sự xuất hiện của IPv6 với không gian địa chỉ cực kì rộng lớn sẽ giúp mọi thứ có thể dễ dàng kết nối vào Internet cũng như kết nối với nhau.

1. **Xu hướng và tính chất**
   1. **Thông minh**

Sự thông minh và tự động trong điều khiển thực chất không phải là một phần trong ý tưởng về IoT. Các máy móc có thể dễ dàng nhận biết và phản hồi lại môi trường xung quanh (ambient intelligence), chúng cũng có thể tự điều khiển bản thân (autonomous control) mà không cần đến kết nối mạng.

Tuy nhiên, trong thời gian gần đây người ta bắt đầu nghiên cứu kết hợp hai khái niệm IoT và autonomous control lại với nhau. Tương lai của IoT có thể là một mạng lưới các thực thể thông minh có khả năng tự tổ chức và hoạt động riêng lẻ tùy theo tình huống, môi trường, đồng thời chúng cũng có thể liên lạc với nhau để trao đổi thông tin, dữ liệu.

Việc tích hợp trí thông minh vào IoT còn có thể giúp các thiết bị, máy móc, phần mềm thu thập và phân tích các dấu vết điện tử của con người khi chúng ta tương tác với những thứ thông minh, từ đó phát hiện ra các tri thức mới liên quan tới cuộc sống, môi trường, các mối tương tác xã hội cũng như hành vi con người.

* 1. **Kiến trúc dựa trên sự kiện**

Các thực thể, máy móc trong IoT sẽ phản hồi dựa theo các sự kiện diễn ra trong lúc chúng hoạt động theo thời gian thực. Một số nhà nghiên cứu từng nói rằng một mạng lưới các sensor chính là một thành phần đơn giản của IoT.

* 1. **Là một hệ thống phức tạp**

Trong một thế giới mở, IoT sẽ mang tính chất phức tạp bởi nó bao gồm một lượng lớn các đường liên kết giữa những thiết bị, máy móc, dịch vụ với nhau, ngoài ra còn bởi khả năng thêm vào các nhân tốc mới.

* 1. **Kích thước**

Một mạng lưới IoT có thể chứa đến 50 đến 100 nghìn tỉ đối tượng được kết nối và mạng lưới này có thể theo dõi sự di chuyển của từng đối tượng. Một con người sống trong thành thị có thể bị bao bọc xung quanh bởi 1000 đến 5000 đối tượng có khả năng theo dõi.

* 1. **Vấn đề không gian, thời gian**

Trong IoT, vị trí địa lý chính xác của một vật nào đó là rất quan trọng. Hiện nay, Internet chủ yếu được sử dụng để quản lí thông tin được xử lý bởi con người. Do đó những thông tin như địa điểm, thời gian, không gian của đối tượng không mấy quan trọng bởi người xử lí thông tin có thể quyết định các thông tin này có cần thiết hay không, và nếu cần thì họ có thể bổ sung thêm.

Trong khi đó, IoT về lý thuyết sẽ thu thập rất nhiều dữ liệu, trong đó có thể có dữ liệu thừa về địa điểm, và việc xử lí dữ liệu đó được xem như không hiệu quả. Ngoài ra, việc xử lí một khối lượng lớn dữ liệu trong thời gian ngắn đủ để đáp ứng cho hoạt động của các đối tượng cũng là một thác thức hiện nay.

1. **Các hệ thống phụ**

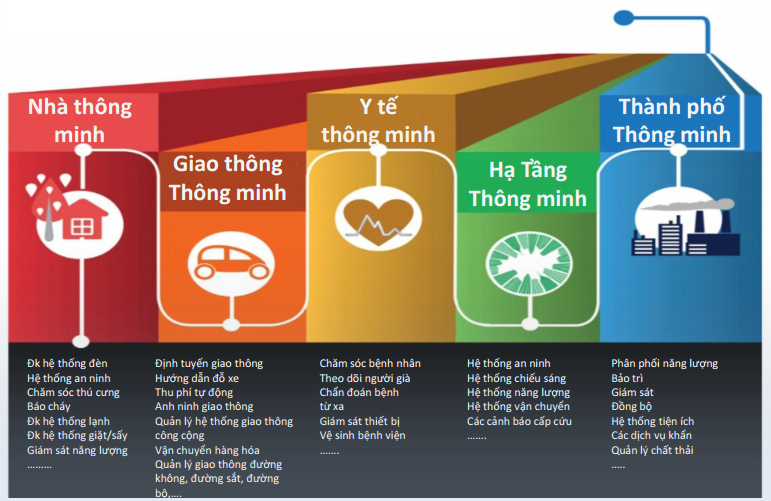
Không phải tất cả mọi thứ nằm trong IoT đều nhất thiết phải kết nối vào một mạng lưới toàn cầu, chúng ta có thể hoạt động trong từng hệ thống đơn lẻ (subsystem). Hãy tưởng tượng đến một căn nhà thông minh, trong đó các đồ điện gia dụng có thể tự chúng tương tác với nhau và hoạt động mà không cần phải vào Internet, trừ khi chúng ta cần điều khiển nó từ xa. Ngôi nhà này có thể được xem là một subsystem. Cũng giống như hiện nay chúng ta có các mạng LAN, WAN, mạng ngang hàng nội bộ chứ không kết nối trực tiếp vào Internet.

1. **Ứng dụng**

Với IoT chúng ta có thể:

* Kết nối với mọi vật.
* Giám sát mọi vật.
* Tìm kiếm mọi vật khi có nhu cầu.
* Quản lý mọi vật.
* Điều khiển mọi vật.
* Và giải trí cùng mọi vật.

Ứng dụng IoT phổ biến ở các lĩnh vực:



# **PHẦN 2: ARDUINO**

1. **Giới thiệu các mạch, cảm biến và module sử dụng**
   1. **Arduino UNO R3**

**Giới thiệu**

Arduino board có rất nhiều phiên bản với hiệu năng và mục đích sử dụng khác nhau như: Arduino Mega, Aruino LilyPad... Trong số đó, Arduino Uno R3 là một trong những phiên bản được sử dụng rộng rãi nhất bởi chi phí và tính linh động của nó.

**Hình ảnh**



**Thông số kỹ thuật**

|  |  |
| --- | --- |
| Vi điều khiển | ATmega328 họ 8bit |
| Điện áp hoạt động | 5V DC (chỉ được cấp qua cổng USB) |
| Tần số hoạt động | 16 MHz |
| Dòng tiêu thụ khoảng | 30mA |
| Điện áp vào khuyên dùng | 7-12V DC |
| Điện áp vào giới hạn | 6-20V DC |
| Số chân Digital I/O | 14 (6 chân hardware PWM) |
| Số chân Analog | 6 (độ phân giải 10bit) |
| Dòng tối đa trên mỗi chân I/O | 30 mA |
| Dòng ra tối đa (5V) | 500 mA |
| Dòng ra tối đa (3.3V) | 50 mA |
| Bộ nhớ flash | 32 KB (ATmega328) với 0.5KB dùng bởi bootloader |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |

Arduino UNO có thể sử dụng 3 vi điều khiển họ 8bit AVR là ATmega8, ATmega168, ATmega328. Bộ não này có thể xử lí những tác vụ đơn giản như điều khiển đèn LED nhấp nháy, xử lí tín hiệu cho xe điều khiển từ xa, làm một trạm đo nhiệt độ - độ ẩm và hiển thị lên màn hình LCD,…

**Năng lượng**

Arduino UNO có thể được cấp nguồn 5V thông qua cổng USB hoặc cấp nguồn ngoài với điện áp khuyên dùng là 7-12V DC và giới hạn là 6-20V. Thường thì cấp nguồn bằng pin vuông 9V là hợp lí nhất nếu bạn không có sẵn nguồn từ cổng USB. Nếu cấp nguồn vượt quá ngưỡng giới hạn trên, bạn sẽ làm hỏng Arduino UNO.

**Các chân năng lượng**

* GND (Ground): cực âm của nguồn điện cấp cho Arduino UNO. Khi bạn dùng các thiết bị sử dụng những nguồn điện riêng biệt thì những chân này phải được nối với nhau.
* 5V: cấp điện áp 5V đầu ra. Dòng tối đa cho phép ở chân này là 500mA.
* 3.3V: cấp điện áp 3.3V đầu ra. Dòng tối đa cho phép ở chân này là 50mA.
* Vin (Voltage Input): để cấp nguồn ngoài cho Arduino UNO, bạn nối cực dương của nguồn với chân này và cực âm của nguồn với chân GND.
* IOREF: điện áp hoạt động của vi điều khiển trên Arduino UNO có thể được đo ở chân này. Và dĩ nhiên nó luôn là 5V. Mặc dù vậy bạn không được lấy nguồn 5V từ chân này để sử dụng bởi chức năng của nó không phải là cấp nguồn.
* RESET: việc nhấn nút Reset trên board để reset vi điều khiển tương đương với việc chân RESET được nối với GND qua 1 điện trở 10KΩ.

**Bộ nhớ**

Vi điều khiển Atmega328 tiêu chuẩn cung cấp cho người dùng:

* **32KB bộ nhớ Flash**: những đoạn lệnh bạn lập trình sẽ được lưu trữ trong bộ nhớ Flash của vi điều khiển. Thường thì sẽ có khoảng vài KB trong số này sẽ được dùng cho bootloader nhưng đừng lo, bạn hiếm khi nào cần quá 20KB bộ nhớ này đâu.
* **2KB cho SRAM** (**S**tatic **R**andom **A**ccess **M**emory): giá trị các biến bạn khai báo khi lập trình sẽ lưu ở đây. Bạn khai báo càng nhiều biến thì càng cần nhiều bộ nhớ RAM. Tuy vậy, thực sự thì cũng hiếm khi nào bộ nhớ RAM lại trở thành thứ mà bạn phải bận tâm. Khi mất điện, dữ liệu trên SRAM sẽ bị mất.
* **1KB cho EEPROM** (**E**lectrically **E**raseble **P**rogrammable **R**ead **O**nly **M**emory): đây giống như một chiếc ổ cứng mini – nơi bạn có thể đọc và ghi dữ liệu của mình vào đây mà không phải lo bị mất khi cúp điện giống như dữ liệu trên SRAM.

**Các cổng vào ra**

Arduino UNO có 14 chân digital dùng để đọc hoặc xuất tín hiệu. Chúng chỉ có 2 mức điện áp là 0V và 5V với dòng vào/ra tối đa trên mỗi chân là 40mA. Ở mỗi chân đều có các điện trở pull-up từ được cài đặt ngay trong vi điều khiển ATmega328 (mặc định thì các điện trở này không được kết nối).

Một số chân digital có các chức năng đặc biệt như sau:

* **2 chân Serial**: 0 (RX) và 1 (TX): dùng để gửi (transmit – TX) và nhận (receive – RX) dữ liệu TTL Serial. Arduino Uno có thể giao tiếp với thiết bị khác thông qua 2 chân này. Kết nối bluetooth thường thấy nói nôm na chính là kết nối Serial không dây. Nếu không cần giao tiếp Serial, bạn không nên sử dụng 2 chân này nếu không cần thiết
* **Chân PWM (~): 3, 5, 6, 9, 10, và 11**: cho phép bạn xuất ra xung PWM với độ phân giải 8bit (giá trị từ 0 → 28-1 tương ứng với 0V → 5V) bằng hàm analogWrite(). Nói một cách đơn giản, bạn có thể điều chỉnh được điện áp ra ở chân này từ mức 0V đến 5V thay vì chỉ cố định ở mức 0V và 5V như những chân khác.
* **Chân giao tiếp SPI**: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ngoài các chức năng thông thường, 4 chân này còn dùng để truyền phát dữ liệu bằng giao thức SPI với các thiết bị khác.
* **LED 13**: trên Arduino UNO có 1 đèn led màu cam (kí hiệu chữ L). Khi bấm nút Reset, bạn sẽ thấy đèn này nhấp nháy để báo hiệu. Nó được nối với chân số 13. Khi chân này được người dùng sử dụng, LED sẽ sáng.

Arduino UNO có 6 chân analog (A0 → A5) cung cấp độ phân giải tín hiệu 10bit (0 → 210-1) để đọc giá trị điện áp trong khoảng 0V → 5V. Với chân **AREF** trên board, bạn có thể để đưa vào điện áp tham chiếu khi sử dụng các chân analog. Tức là nếu bạn cấp điện áp 2.5V vào chân này thì bạn có thể dùng các chân analog để đo điện áp trong khoảng từ 0V → 2.5V với độ phân giải vẫn là 10bit.

Đặc biệt, Arduino UNO có 2 chân A4 (SDA) và A5 (SCL) hỗ trợ giao tiếp I2C/TWI với các thiết bị khác.

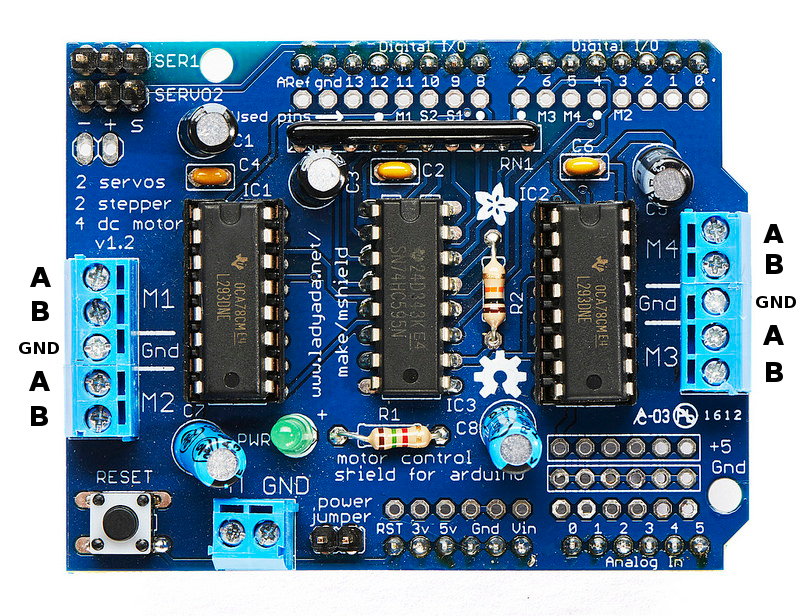
* 1. **Arduino Motor Shield L293D**

**Giới thiệu**

Motor Driver Shield L293D (Shield) là một trong các module mở rộng dành cho Arduino. Tương thích tốt với board Arduino Uno R3, Arduino Leonardo R3, Arduino Mega 2560.

Shield sử dụng 2 IC cầu H L293D và 1 IC logic 74HC595 điều khiển. Shield có thể điều khiển nhiều loại motor (servo, stepper, dc motor) khác nhau với mức áp đến 36V, dòng tối đa 600mA mỗi kênh điều khiển.

**Hình ảnh**



**Thông số kỹ thuật và khả năng điều khiển của Shield:**

* Hoạt động DC 6V-35V cấp cho động cơ.
* Dùng IC chuyên dụng L293D.
* Dòng điện 0.6A cho mỗi ngõ ra.
* 2 động cơ servo.
* 4 động cơ một chiều (M1, M2, M3, M4) độc lập thông qua 4 cầu H của L293D dòng 600mA (cực đại 1.2A).
* 2 động cơ bước loại đơn cực (unipolar) hoặc lưỡng cực (bipolar).
* Chống tự quay khi cấp nguồn (có sẵn các điện trở nối GND).
* Có sẵn nút RESET để khởi động lại board Arduino.

**Các chân trên Shield được kết nối với board Arduino như sau:**

* Servo Motor: 2 dây điều khiển 2 servo kết nối với chân số 9 và 10. Nguồn nuôi lấy trực tiếp từ board Arduino.
* Động cơ một chiều (DC):
  + Motor 1 nối với chân 11
  + Motor 2 nối với chân 3
  + Motor 3 nối với chân 5
  + Motor 4 nối với chân 6
* Động cơ bước: Chân 4, 7, 8, 12 dùng điều khiển motor thông qua IC 74HC595.

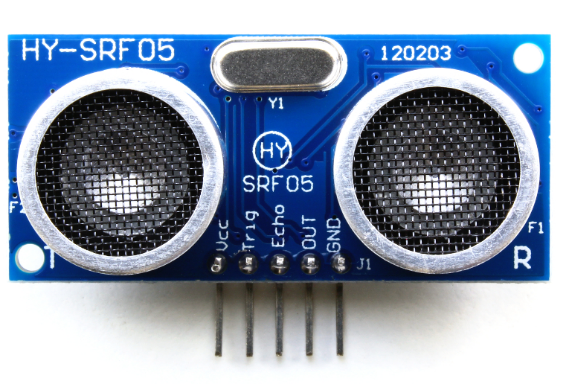
Trên shield có 1 jumper màu vàng PWR được sử dụng nếu kết nối nguồn ngoài cho board Arduino thông qua jack DC (dùng pin 9V chẳng hạn) thì nguồn nuôi motor được lấy luôn từ jack này, không cần nối nguồn với EXT\_PWR. Nếu ngắt jumper, bạn cần nối 1 nguồn riêng vào terminal EXT\_PWR để nuôi motor.

* 1. **Cảm biến siêu âm SRF05**

**Giới thiệu**

Cảm biến SRF05 là phiên bản nâng cấp của SRF04, cảm biến dùng để nhận biết khoảng cách đến vật cản nhờ sóng siêu âm phát ra từ cảm biến và nhận về.

**Hình ảnh**



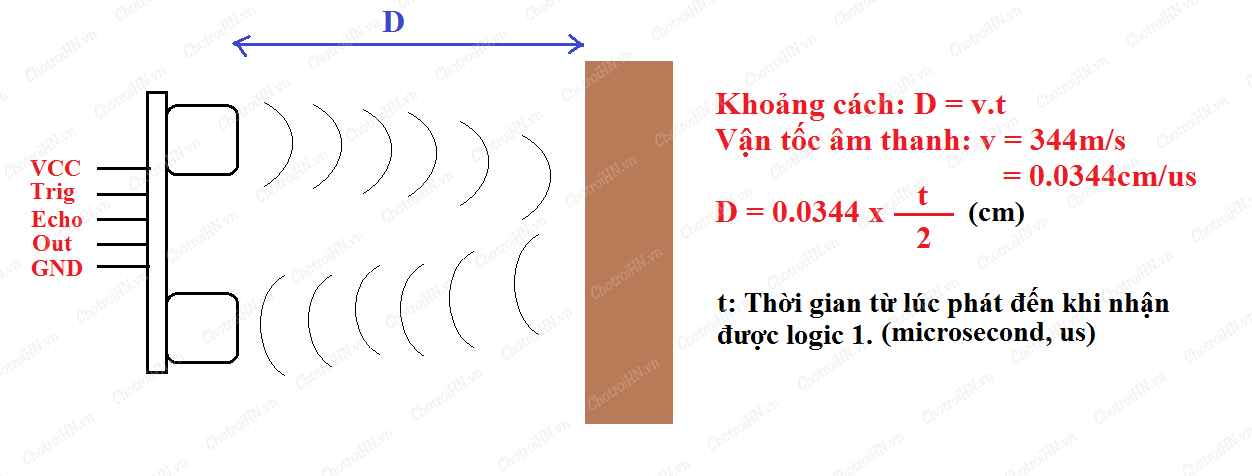
**Thông số kỹ thuật**

* SRF05 có 5 chân, bao gồm: VCC, Trigger, Echo, Out, và GND.
* Điện áp hoạt động: 5VDC
* Độ chính xác: 0.5cm

**Chức năng của các chân:**

* VCC: Chân cấp nguồn 5V.
* GND: Chân nối mass(0V).
* Trigger: Chân kích phát sóng siêu âm (ở chế độ 1). Ở chế độ 2 thì chân Trigger vừa là chân kích vừa là chân báotín hiệu phản xạ của sóng siêu âm.
* Echo: Chân báo có tín hiệu phản xạ của sóng siêu âm.
* Out: Không dùng (Not Connect).

**Nguyên lý hoạt động**



Để đo khoảng cách, ta sẽ phát 1 xung (1000 microSeconds - us) từ chân **Trig**. Sau đó, cảm biến sẽ tạo ra 1 xung HIGH ở chân **Echo** cho đến khi nhận lại được sóng phản xạ ở pin này. Chiều rộng của xung sẽ bằng với thời gian sóng siêu âm được phát từ cảm biển và quay trở lại.

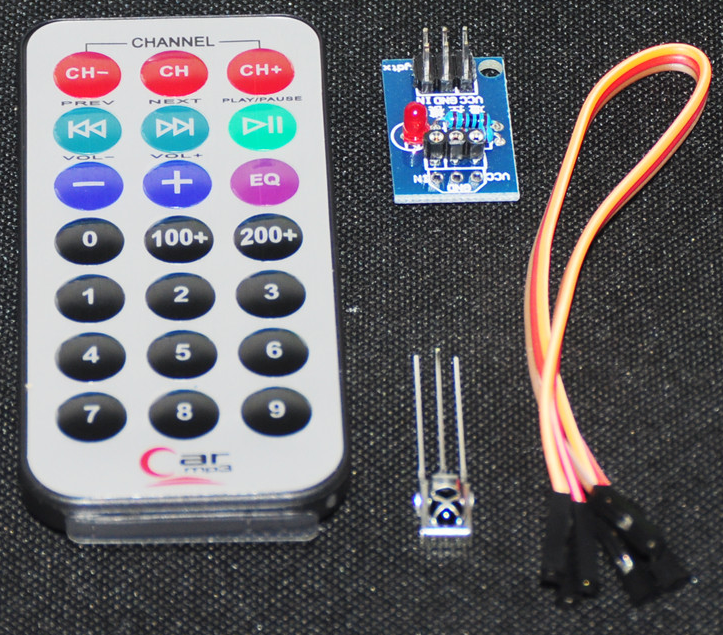
Tốc độ của âm thanh trong không khí khoảng 344 m/s = 0.0344 cm/us. Khi đã tính được thời gian, ta sẽ tính khoảng cách D theo công thức bên trên.

* 1. **Module thu phát hồng ngoại HX1838**

**Giới thiệu**

Remote điều khiển hồng ngoại được thiết kế nhỏ gọn và dễ dang kết nối với vi điều khiển. Khoảng cách hoạt động khoảng 5-8 mét, nhưng các ứng dụng thực tế phụ thuộc vào các yếu tố môi trường xung quanh (chẳng hạn như một màng ngăn, kính, hoặc chướng ngại vật khác bị chặn), bộ thu sử dụng HX1838, có thể nhận được bất kỳ mã kiểm soát 38Khz tần số dữ liệu từ xa.

**Hình ảnh**



**Thông số kỹ thuật**

* Cảm biến thu hồng ngoại: HX1838
* Điện áp hoạt động: DC 3.3 ~ 5V
* Tín hiệu ngõ ra: Digital
* Khoảng cách thu tín hiệu: 5 ~ 8m
* Tích hợp 1 Remote theo module
* Kích thước module: 0.94 in x 0.67 in x 0.47 in (2.4 cm x 1.7 cm x 1.2 cm)
* Khối lượng: 0.18 oz (5g)

**Chức năng các chân**

* Vcc -5V
* GND - GND
* D0 - Ngõ ra digital

1. **IDE hỗ trợ**
   1. **Arduino IDE**

Để lập trình được cho các board Arduino, các bạn cần phải có một công cụ gọi là Intergrated Development Environment (IDE). Công cụ này được đội ngũ kĩ sư của Arduino phát triển và có thể chạy trên Windows , MAC OS X và Linux.

Dưới đây công cụ được cài trên Windows.

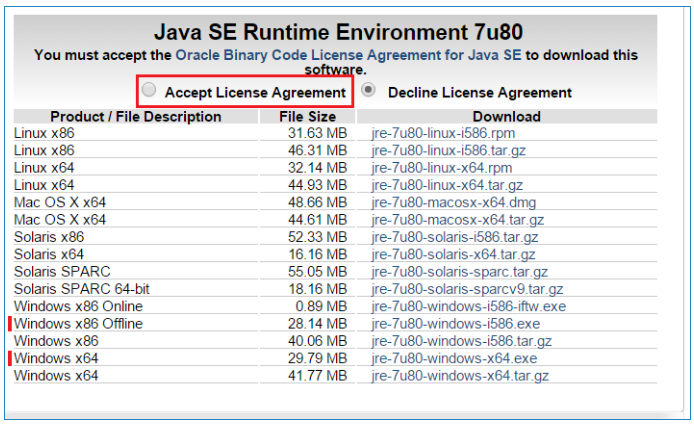
* + 1. **Cài đặt Java Runtime Environment (JRE)**

Vì Arduino IDE được viết trên Java nên bạn cần phải cài đặt JRE trước Arduino IDE.

Link tải:

<http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html>

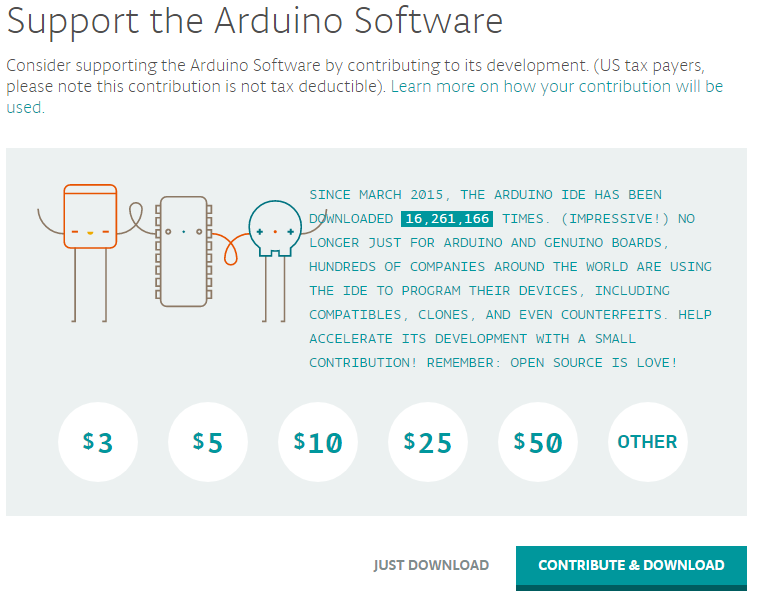
2 bản JRE phổ biến nhất là bản dành cho Windows 32bit (x86) và Windows 64bit (x64) như đánh dấu trong hình. Nhớ chọn "Accept License Agreement".



* + 1. **Cài đặt Arduino IDE**

**Bước 1**: Truy cập địa chỉ <https://www.arduino.cc/en/Main/Software/>, đây là nơi lưu trữ cũng như cập nhật các bản IDE của Arduino. Bấm vào mục “*Windows ZIP file for non admin install*” như hình minh họa.

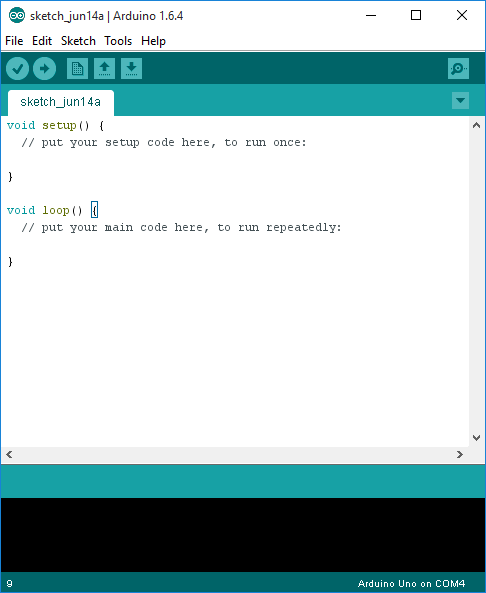




**Bước 2**: Sau khi download, giải nén file arduino-1.6.4-windows.zip.

**Bước 3:** Copy thư mục arduino-1.6.4 vừa giải nén đến nơi lưu trữ.

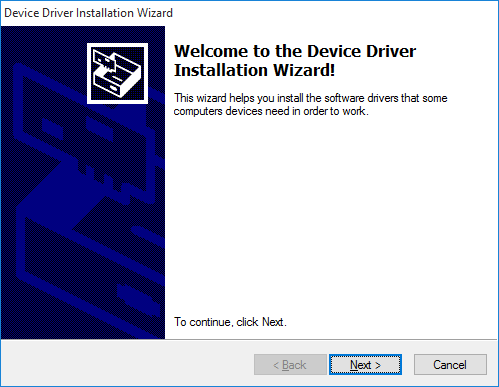
**Bước 4**: Chạy file arduino.exe trong thư mục arduino-1.6.4\ để khởi động Arduino IDE

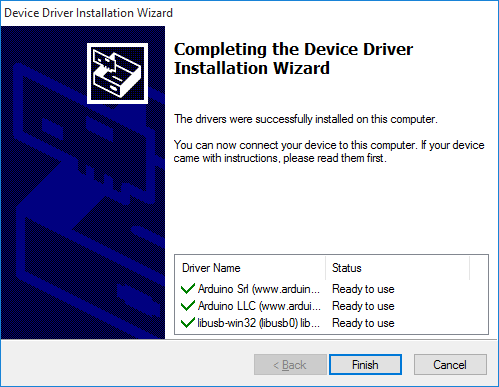


* + 1. **Cài đặt Driver**

Để máy tính của bạn và board Arduino giao tiếp được với nhau, chúng ta cần phải cài đặt driver trước tiên.

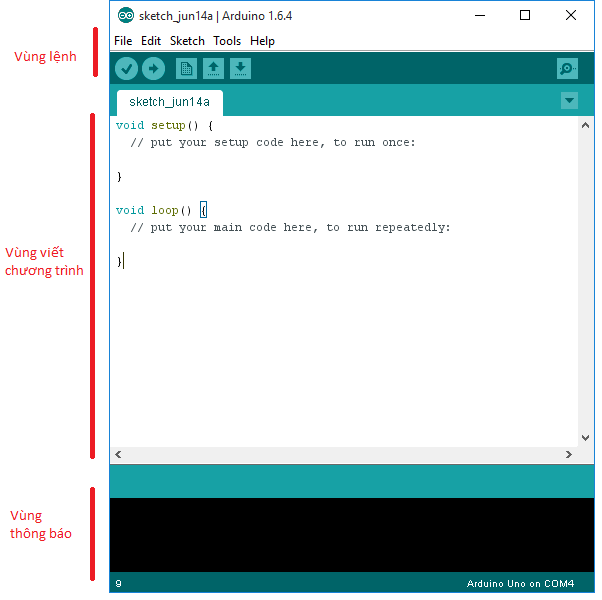
Đầu tiên, các bạn chạy file arduino-1.6.4\drivers\dpinst-x86.exe (Windows x86) hoặc arduino-1.6.4\drivers\dpinst-amd64.exe (Windows x64). Cửa sổ “Device Driver Installation Wizard” hiện ra, các bạn chọn Next để tiếp tục.





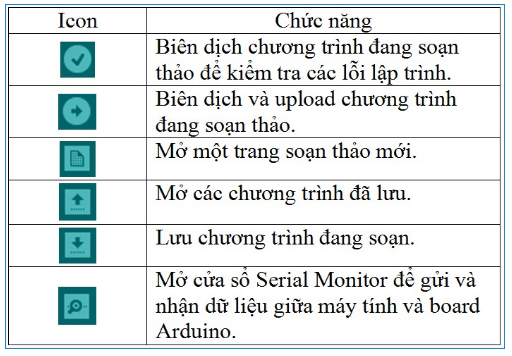
* + 1. **Cách sử dụng Arduino IDE**

**Giao diện**

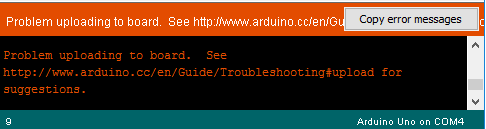
****

**Vùng lệnh**

Bao gồm các nút lệnh menu (File, Edit, Sketch, Tools, Help). Phía dưới là các icon cho phép sử dụng nhanh các chức năng thường dùng của IDE được miêu tả như sau:



**Vùng thông báo**



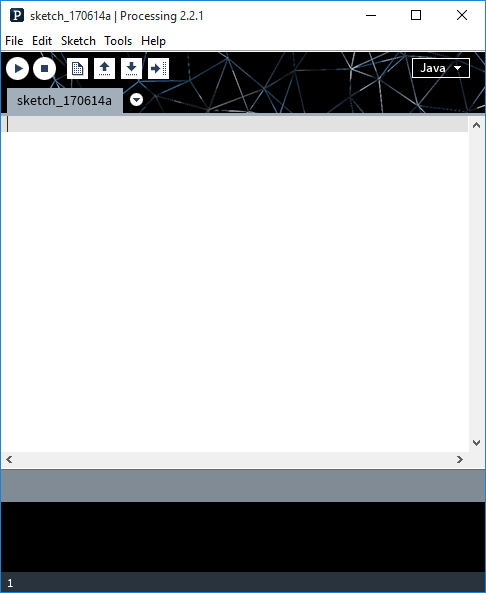
Những thông báo từ IDE sẽ được hiển thị tại đây. Để ý rằng góc dưới cùng bên phải hiển thị loại board Arduino và cổng COM được sử dụng. Luôn chú ý tới mục này bởi nếu chọn sai loại board hoặc cổng COM, bạn sẽ không thể upload được code của mình.

* 1. **Processing Arduino IDE**
     1. **Công dụng**
* Dùng để điều khiển các mạch Arduino qua giao thức Serial bằng máy tính.
* Ngôn ngữ Processing có rất nhiều sự tương đồng với Arduino, Arduino được dựa trên C để viết thêm thư viện Arduino, còn Processing được dựa trên Java để viết nên thư viện Processing.
  + 1. **Cài đặt**

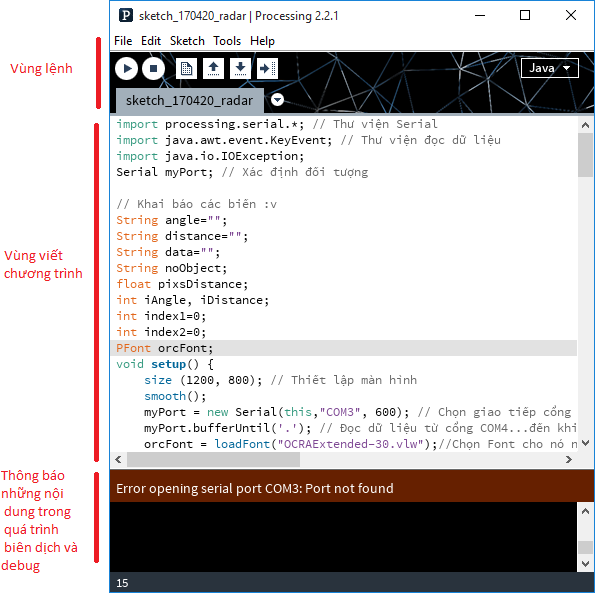
**Bước 1:** Tải Processing theo link sau:

[**https://processing.org/download/**](https://processing.org/download/)

**Bước 2:** Giải nén file vừa tải và chạy file processing.exe



* + 1. **Cách sử dụng Processing**

****

1. **Thư viện sử dụng**
   * 1. **IRremote**

**Mục đích sử dụng:**

Sử dụng tín hiệu thu được từ module thu hồng ngoại và phân tích thành mã HEX. Dựa vào mã HEX để điều khiển theo ý muốn của lập trình viên.

**Keywords & phương thức:**

* IRrecv irrecv(receiverPin);: tạo đối tượng IRrecv mới có tên là irrecv sử dụng tham số là receiverPin.
* decode\_results results; : lưu kết quả giải mã được.
* irrecv.enableIRIn(); : bắt đầu giải mã tín hiệu IR.
* irrecv.decode(&results);: trả về true nếu có tín hiệu đến.
* irrecv.resume; : đợi tín hiệu IR tiếp theo.

**Download tại:**

<https://github.com/z3t0/Arduino-IRremote>

* + 1. **Adafruit Motor Shield V2**

**Mục đích sử dụng:**

Điều khiển các loại motor: motor DC, servo, step motor.

**Download tại:**

<https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library>

* + 1. **NewPing**

**Mục đích sử dụng:**

Tính khoảng cách từ vật thể đến cảm biến. Thư viện NewPing này cho phép ping cảm biến luôn và đáng tin cậy ở tốc độ 30 lần / giây.

**Phương thức:**

[max\_cm\_distance] là khoảng cách tối đa mới.

* sonar.ping([max\_cm\_distance]) - Gửi một ping và nhận được echo thời gian (trong micro giây) dưới dạng một kết quả.
* sonar.ping\_in([max\_cm\_distance]) – Tương tự phương thức trên nhưng kết quả trả về là in.
* sonar.ping\_cm([max\_cm\_distance]) - Tương tự phương thức trên nhưng kết quả trả về là cm.
* sonar.ping\_median(iterations [, max\_cm\_distance]) - Thực hiện nhiều lệnh ping (mặc định = 5), loại bỏ ping phạm vi và trung bình trở lại trong micro giây.
* sonar.convert\_in(echoTime) - Convert echoTime từ micro giây sang in.
* sonar.convert\_cm(echoTime) - Convert echoTime từ micro giây sang cm.
* sonar.ping\_timer(function [, max\_cm\_distance]) - Gửi hàm ping và call để kiểm tra nếu ping đã hoàn thành.
* sonar.check\_timer() - Kiểm tra xem ping đã trở lại trong giới hạn khoảng cách thiết lập.
* NewPing::timer\_ms(frequency, function) - Gọi hàm liên tục với tần số micro giây.
* NewPing::timer\_stop() – Dừng timer.

**Download tại:**

<https://github.com/PaulStoffregen/NewPing>

1. **Cách thực hiện**
   1. **Hello world**

**Mục đích:**

Tìm hiểu và lập trình trên arduino bài đầu tiên. Làm đèn led trên Arduino UNO R3 nhấp nháy theo chu kỳ 1 giây sáng – 1 giây tắt.

**Phần cứng dụng:**

* Mạch Arduino UNO R3.
* Cáp USB type C.

**Cách thực hiện:**

Gắn cáp kết nối Arduino UNO R3 với máy tính qua cổng USB type C.

**Code:**

Phụ lục (1) trang 39

* 1. **Điều khiển led bằng remote**

**Mục đích:**

Phát triển bài code phía trên, nhưng lúc này bật tắt đèn led theo ý muốn. Khi bấm vào điều khiển phím 1 thì đèn sẽ bật, khi bấm vào phím 2 thì đèn sẽ tắt.

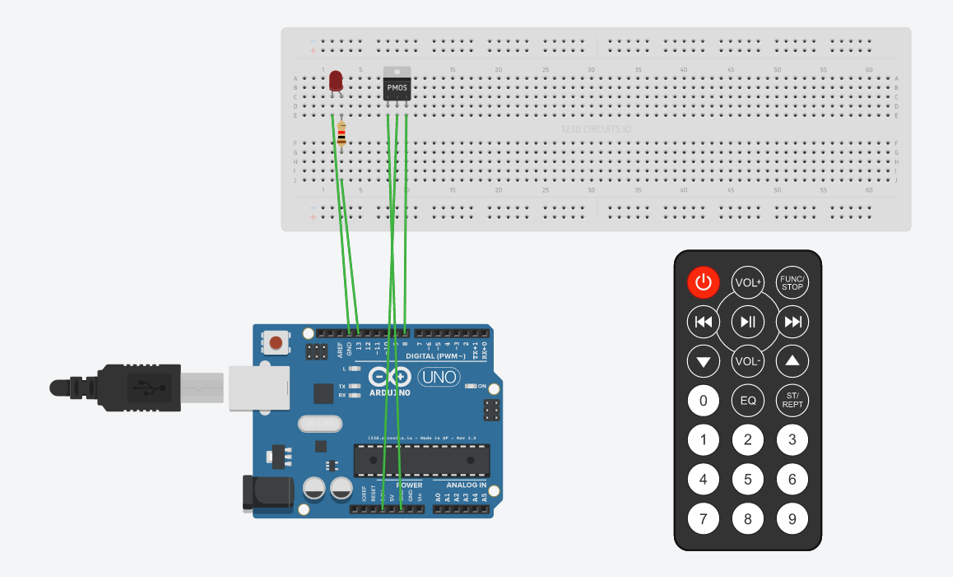
**Phần cứng dụng:**

* Mạch Arduino UNO R3.
* Cáp USB type C.
* Đèn led, điện trở.
* Remote, module thu hồng ngoại HX1838.
* Breadboard (nếu có).

**Cách thực hiện:**

* Cắm chân GRD của cảm biến hồng ngoại với chân GND của Arduino.
* Chân VCC nối với output 3,3V của Arduino.
* Chân OUT gắn chân số 8 của Arduino.
* Led gắn thêm điện trở 220ohm và đi qua chân 13 của Arduino. Chân còn lại cắm vào GND.

Chi tiết mạch như hình dưới:



**Code:**

Phụ lục (2) trang 40

* 1. **Điều khiển động cơ xe bằng remote**

Mục đích:

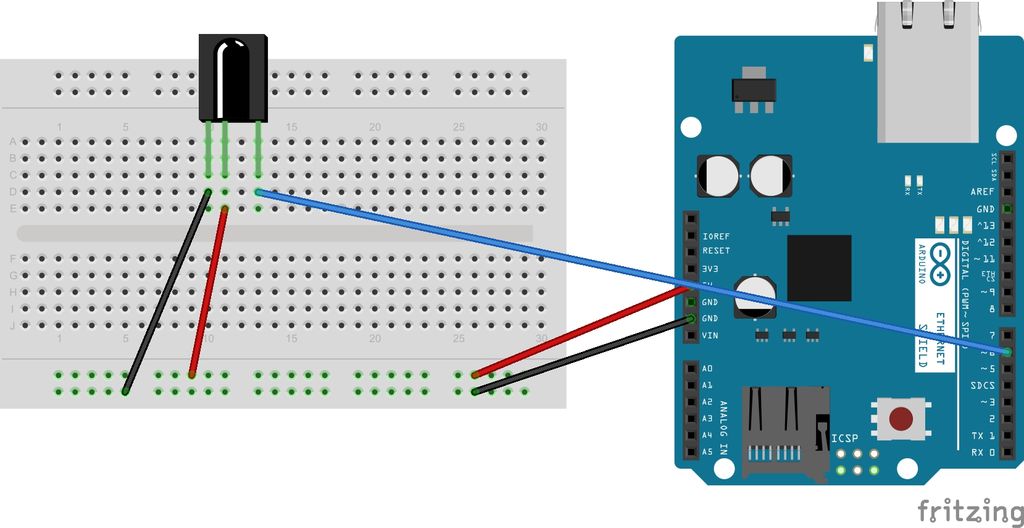
Tạo mô hình xe và điều khiển bằng remote.

**Phần cứng dụng:**

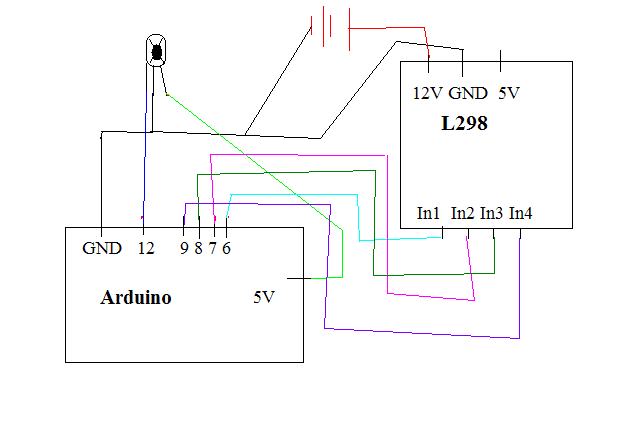
* Mạch Arduino UNO R3.
* Cáp USB type C.
* Remote, module thu hồng ngoại HX1838.
* Motor Shield L293D.
* 4 hộp số motor DC (loại vàng) cùng với 4 gá motor DC + 2 tấm khung.
* Nguồn cấp cho Arduino và Motor Shield: Pin Samsung 2230mAh (3 pin).

**Cách thực hiện:**

Sơ đồ kết nối remote:



Tiếp tục lắp ráp theo sơ đồ sau:



**Code:**

Phụ lục (3) trang 41

* 1. **Làm radar dò vật cản**

**Mục đích:**

Tạo Robot của mình có thể quét địa hình, phát hiện chướng ngại vật, hỗ trợ Robot di chuyển một cách đơn giản hơn khi gặp vật cản.

**Phần cứng dụng:**

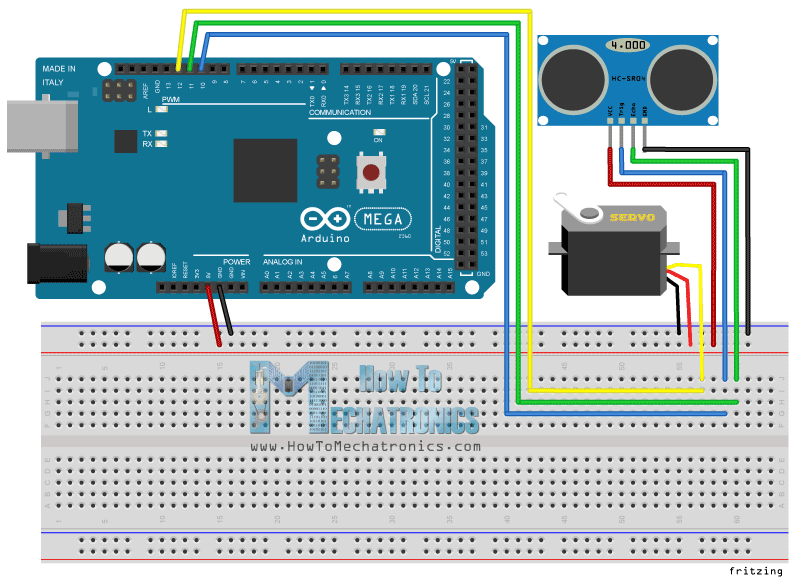
* Mạch Arduino UNO R3.
* Cáp USB type C.
* Motor Servo Tower Pro SG90.
* Cảm biến siêu âm SRF05.

**Cách thực hiện:**

Phần cứng kết nối như sau:

|  |  |  |
| --- | --- | --- |
| Chân Siêu âm | Chân Servo | Chân Arduino UNO |
| Vcc | Vcc | 5V |
| GND | GND | GND |
|  | Data | 12 |
| Trig |  | 10 |
| Echo |  | 11 |

Sơ đồ kết nối



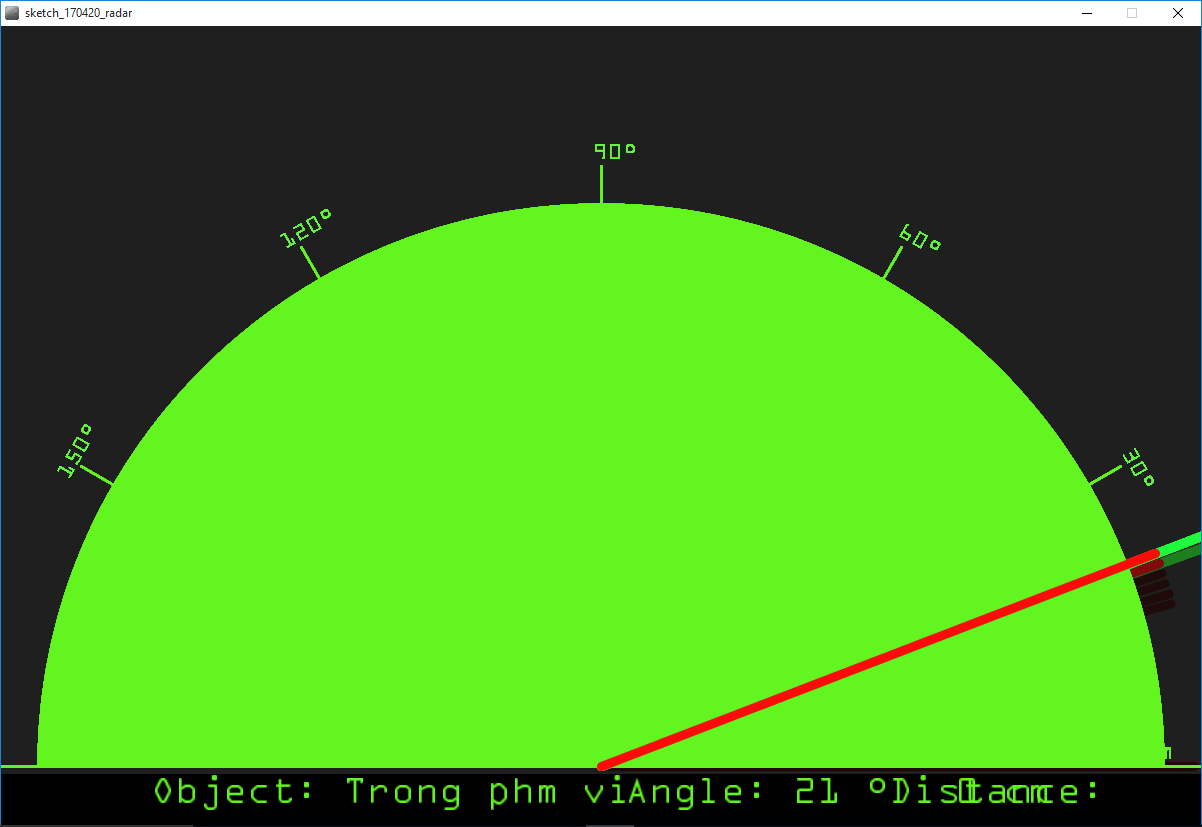
**Code Arduino:**

Phụ lục (4.1) trang 43

**Code Processing:**

Phụ lục (4.2) trang 44

**Kết quả:** (ở giao diện debug của Processing)



* 1. **Xe tự động**

**Mục đích:**

Tạo xe sẽ chạy tự động theo đường thẳng, khi gặp vật cản sẽ lùi lại và quét đường đi, sau đó chọn hướng nào có đường đi xa nhất.

**Phần cứng dụng:**

* Mạch Arduino UNO R3.
* Cáp USB type C.
* Motor Servo Tower Pro SG90.
* Cảm biến siêu âm SRF05.
* Motor Shield L293D.
* 4 hộp số motor DC (loại vàng) cùng với 4 gá motor DC + 2 tấm khung.
* Nguồn cấp cho Arduino và Motor Shield: Pin Samsung 2230mAh (3 pin).

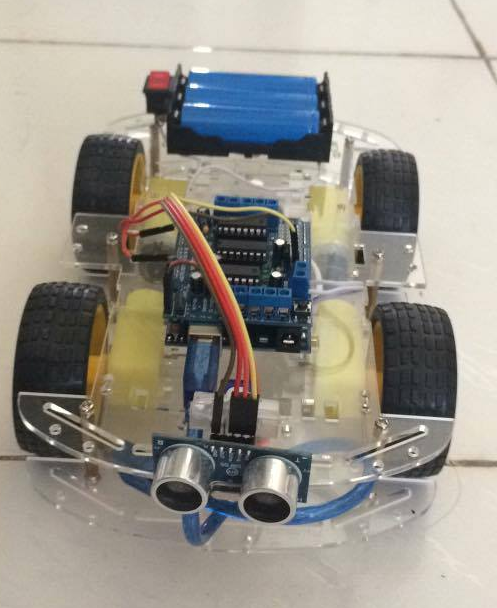
**Cách thực hiện:**

* Thư viện Adafruit Motor Shield: dùng để điều khiển motor và servo.
* Thư viện NewPing: dùng để nhận tín hiệu và đo khoảng cách từ vật đến cảm biến.
* Ghép Motor Shield L293D vào Arduino UNO R3.
* Gắn 2 motor vào vị trí M3, M4 của Motor Shield L293D.
* Gắn Servo vào vị trí Serv1 của Motor Shield L293D.
* SRF05: chân TRIG hàn vào chân 11, chân ECHO hàn vào chân 3 của Motor Shield L293D.
* Nguồn cắm vào cổng nguồn của Motor Shield L293D.

**Code:**

Phụ lục (5) trang 47

**Hình ảnh sản phẩm:**



# **PHỤ LỤC 1: SOURCE CODE**

* + - * 1. **Hello World** / trang 30

int led = 13; // chân digital 13 cần được kết nối với đèn LED

void setup()// Hàm setup chạy một lần duy nhất khi khởi động chương trình

{

pinMode(led, OUTPUT);

}

void loop() // Hàm loop chạy mãi mãi sau khi kết thúc hàm setup()

{

digitalWrite(led, HIGH); // bật đèn led sáng

delay(1000); // dừng trong 1 giây => thây đèn sáng được 1 giây

digitalWrite(led, LOW); // tắt đèn led

delay(1000); // dừng trong 1 giây => thấy đèn tối được 1 giây

}

* + - * 1. **Điều khiển led bằng remote** / trang 31

#include <IRremote.h> // thư viện hỗ trợ IR remote

const int receiverPin = 8; // chân digital 8 dùng để đọc tín hiệu

IRrecv irrecv(receiverPin); // tạo đối tượng IRrecv mới

decode\_results results;// lưu giữ kết quả giải mã tín hiệu

const int LED1 = 13;

void setup()

{

Serial.begin(9600);// serial

irrecv.enableIRIn(); // start the IR receiver

pinMode(LED1, OUTPUT); //Tất cả các chân LED là OUTPUT

digitalWrite(LED1, LOW); //Mặc định các đèn LED tắt

}

// translate IR signals

void translateIR()

{

switch (results.value)

{

case 0xFF6897://phim so 1

digitalWrite(LED1, HIGH);

break;

case 0xFF9867://phim so 2

digitalWrite(LED1, LOW);

break;

}

}

void loop()

{

if (irrecv.decode(&results)) // nếu nhận được tín hiệu

{

translateIR();

Serial.println(results.value, HEX);

delay(200);

irrecv.resume(); // nhận giá trị tiếp theo

}

}

* + - * 1. **Điều khiển động cơ xe bằng remote** / trang 32

#define ir 12

#define inA1 6

#define inA2 7

#define inB1 8

#define inB2 9

#include <IRremote.h>

IRrecv irrecv(ir);

decode\_results results;

void setup()

{

irrecv.enableIRIn();

pinMode(inA1,OUTPUT);

pinMode(inA2,OUTPUT);

pinMode(inB1,OUTPUT);

pinMode(inB2,OUTPUT);

}

void translateIR()

{

switch(results.value)

{

case 0xC03E:

robotMove(1);

break;

case 0xC05E:

robotMove(0);

break;

case 0xC0DE:

robotMove(2);

break;

case 0xC05A:

robotMove(3);

break;

case 0xC0DA:

robotMove(4);

break;

case 0xC01E:

robotMove(6);

break;

case 0xC038:

robotMove(5);

break;

}

}

void robotMove(int chedo)

{

switch(chedo)

{

case 0: //Dung

motorMove(inA1,inA2,0);

motorMove(inB1,inB2,0);

break;

case 1: //Di thang

motorMove(inA1,inA2,1);

motorMove(inB1,inB2,1);

break;

case 2: //Lui

motorMove(inA1,inA2,2);

motorMove(inB1,inB2,2);

break;

case 3: //Re phai

motorMove(inA1,inA2,0);

motorMove(inB1,inB2,1);

break;

case 4: //Re trai

motorMove(inA1,inA2,1);

motorMove(inB1,inB2,0);

break;

case 5: //Xoay phai

motorMove(inA1,inA2,2);

motorMove(inB1,inB2,1);

break;

case 6: //Xoay trai

motorMove(inA1,inA2,1);

motorMove(inB1,inB2,2);

break;

}

}

void motorMove(int in1,int in2, byte action)

{

switch(action)

{

case 0: //Dung

digitalWrite(in1,0);

digitalWrite(in2,0);

break;

case 1: //Chieu thu 1

digitalWrite(in1,1);

digitalWrite(in2,0);

break;

case 2: //Chieu thu 2

digitalWrite(in1,0);

digitalWrite(in2,1);

break;

}

}

void loop()

{

if (irrecv.decode(&results))

{

translateIR();

delay(200);

irrecv.resume();

}

}

* + - * 1. **Làm radar dò vật cản** / trang 33

### **(4.1) Code Arduino IDE**

// Sử dụng thư viện Servo

#include <Servo.h>

// Khai báo chân Trig và Echo

const int trigPin = 10;

const int echoPin = 11;

// Khai báo biến thời gian và khoảng cách

long duration;

int distance;

Servo myServo; // Tạo đối tượng Servo để điều khiển Servo

void setup() {

pinMode(trigPin, OUTPUT); // Set chân trig là chân OUTPUT

pinMode(echoPin, INPUT); // Set chân echo INPUT

Serial.begin(9600);

myServo.attach(12); // Chân Servo là chân 12

}

void loop() {

// Quay Servo từ 15 độ - 165 độ

for(int i=15;i<=165;i++){

myServo.write(i);

delay(30);

distance = calculateDistance();// Gọi hàm tính khoảng cách

Serial.print(i); // Gửi giá trị i ( góc quay của Servo) đến Serial Port

Serial.print(","); // Gửi ","

Serial.print(distance); // Tiếp theo gửi các giá trị khoảng cách tới Serial Port

Serial.print("."); // Gửi dấu "."

}

// Sau khi quay xong Servo từ 15 - 165 độ....quay ngược lại từ 165 độ về 15 độ, sau đó gửi tương tự các giá trị như bên trên

for(int i=165;i>15;i--){

myServo.write(i);

delay(30);

distance = calculateDistance();

Serial.print(i);

Serial.print(",");

Serial.print(distance);

Serial.print(".");

}

}

// Hàm tính khoảng cách bằng siêu âm

int calculateDistance(){

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Phát xung siêu âm với thời gian là 10 Micro giây

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);// Ngưng phát xung

duration = pulseIn(echoPin, HIGH); // Tính thời gian xung siêu âm phát ra đập vào vật cản rồi dội lại là bao nhiêu???

distance= duration\*0.034/2;// Tính khoảng cách từ siêu âm đến vật cản ( vì vận tốc của xung siêu âm là 344m/s...và sau khi đập vào vật cản rồi dội lại, nên /2) ==>> khoảng cách = thời gian\*vận tốc/2

return distance;

}

### **(4.2) Code Processing**

import processing.serial.\*; // Thư viện Serial

import java.awt.event.KeyEvent; // Thư viện đọc dữ liệu

import java.io.IOException;

Serial myPort; // Xác định đối tượng

// Khai báo các biến

String angle="";

String distance="";

String data="";

String noObject;

float pixsDistance;

int iAngle, iDistance;

int index1=0;

int index2=0;

PFont orcFont;

void setup() {

size (1920, 1080); // Thiết lập màn hình

smooth();

myPort = new Serial(this,"COM4", 9600); // Chọn giao tiếp cổng COM4, tốc độ 9600

myPort.bufferUntil('.'); // Đọc dữ liệu từ cổng COM4...đến khi nhận đc dấu '.' thỳ dừng lại....!!!! ( Processing sẽ nhận đc giá trị như phần Code Arduino đã gửi)

orcFont = loadFont("OCRAExtended-30.vlw");//Chọn Font cho nó nguy hiểm tý, các bạn có thể chọn Font khác theo ý mình

}

void draw() {

fill(98,245,31);

textFont(orcFont);

// Cái này mình cũng không hiểu lắm...hình như nó mô phỏng dòng chuyển động...Mà thôi,các bạn cứ coppy cho nhanh

noStroke();

fill(0,4);

rect(0, 0, width, height-height\*0.065);

fill(98,245,31); // green color

// Gọi các hàm vẽ Radar

drawRadar();

drawLine();

drawObject();

drawText();

}

void serialEvent (Serial myPort) { // Đọc dữ liệu từ cổng COM

data = myPort.readStringUntil('.');// Đọc dữ liệu đến khi phát hiện dấu "." rồi gán vào biến "data"

data = data.substring(0,data.length()-1);

index1 = data.indexOf(","); // Tìm ký tự "," rồi gán vào "index1"

angle= data.substring(0, index1); // Đọc dữ liệu từ ký tự đầu tiên...đến khi nhận dấu "," rồi gán vào biến "angle"

distance= data.substring(index1+1, data.length()); // Đọc dữ liệu từ ký tự sau dấu "," đến dấu "." ( đến hết đó )

// Chuyển đổi các chuỗi thành số nguyên

iAngle = int(angle);

iDistance = int(distance);

}

// Đến phần xây dựng giao diện Radar...Đây là vẽ hình á...các bạn cứ coppy

void drawRadar() {

pushMatrix();

translate(width/2,height-height\*0.074); // di chuyển tọa độ đến vị trí mới!!!

strokeWeight(2);

stroke(98,245,31);

// Vẽ các vệt line sao ý!!! keke

arc(0,0,(width-width\*0.0625),(width-width\*0.0625),PI,TWO\_PI);

arc(0,0,(width-width\*0.27),(width-width\*0.27),PI,TWO\_PI);

arc(0,0,(width-width\*0.479),(width-width\*0.479),PI,TWO\_PI);

arc(0,0,(width-width\*0.687),(width-width\*0.687),PI,TWO\_PI);

// Vẽ các đường góc

line(-width/2,0,width/2,0);

line(0,0,(-width/2)\*cos(radians(30)),(-width/2)\*sin(radians(30)));

line(0,0,(-width/2)\*cos(radians(60)),(-width/2)\*sin(radians(60)));

line(0,0,(-width/2)\*cos(radians(90)),(-width/2)\*sin(radians(90)));

line(0,0,(-width/2)\*cos(radians(120)),(-width/2)\*sin(radians(120)));

line(0,0,(-width/2)\*cos(radians(150)),(-width/2)\*sin(radians(150)));

line((-width/2)\*cos(radians(30)),0,width/2,0);

popMatrix();

}

void drawObject() {

pushMatrix();

translate(width/2,height-height\*0.074); // Di chuyển con trỏ đến tọa độ khác...rồi vẽ tiếp

strokeWeight(9);

stroke(255,10,10); // Màu đỏ

pixsDistance = iDistance\*((height-height\*0.1666)\*0.025); // Khoảng cách..từ cm đổi thành pixcel ( theo công thức )

// Nếu khoảng cách < 40cm

if(iDistance<40){

// Vẽ các đối tượng góc và khoảng cách...!!!!

line(pixsDistance\*cos(radians(iAngle)),-pixsDistance\*sin(radians(iAngle)),(width-width\*0.505)\*cos(radians(iAngle)),-(width-width\*0.505)\*sin(radians(iAngle)));

}

popMatrix();

}

void drawLine() {

pushMatrix();

strokeWeight(9);

stroke(30,250,60);

translate(width/2,height-height\*0.074); // Lại di chuyển con trỏ đến vị trí khác

line(0,0,(height-height\*0.12)\*cos(radians(iAngle)),-(height-height\*0.12)\*sin(radians(iAngle)));

popMatrix();

}

void drawText() { // Viết các dòng chữ trên màn hình

pushMatrix();

if(iDistance>40) {

noObject = "Ra khỏi phạm vi";

}

else {

noObject = "Trong phạm vi";

}

fill(0,0,0);

noStroke();

rect(0, height-height\*0.0648, width, height);

fill(98,245,31);

textSize(25);

text("10cm",width-width\*0.3854,height-height\*0.0833);

text("20cm",width-width\*0.281,height-height\*0.0833);

text("30cm",width-width\*0.177,height-height\*0.0833);

text("40cm",width-width\*0.0729,height-height\*0.0833);

textSize(40);

text("Object: " + noObject, width-width\*0.875, height-height\*0.0277);

text("Angle: " + iAngle +" °", width-width\*0.48, height-height\*0.0277);

text("Distance: ", width-width\*0.26, height-height\*0.0277);

if(iDistance<40) {

text(" " + iDistance +" cm", width-width\*0.225, height-height\*0.0277);

}

textSize(25);

fill(98,245,60);

translate((width-width\*0.4994)+width/2\*cos(radians(30)),(height-height\*0.0907)-width/2\*sin(radians(30)));

rotate(-radians(-60));

text("30°",0,0);

resetMatrix();

translate((width-width\*0.503)+width/2\*cos(radians(60)),(height-height\*0.0888)-width/2\*sin(radians(60)));

rotate(-radians(-30));

text("60°",0,0);

resetMatrix();

translate((width-width\*0.507)+width/2\*cos(radians(90)),(height-height\*0.0833)-width/2\*sin(radians(90)));

rotate(radians(0));

text("90°",0,0);

resetMatrix();

translate(width-width\*0.513+width/2\*cos(radians(120)),(height-height\*0.07129)-width/2\*sin(radians(120)));

rotate(radians(-30));

text("120°",0,0);

resetMatrix();

translate((width-width\*0.5104)+width/2\*cos(radians(150)),(height-height\*0.0574)-width/2\*sin(radians(150)));

rotate(radians(-60));

text("150°",0,0);

popMatrix();

}

* + - * 1. **Xe tự động** / trang 36

#include <AFMotor.h>

#include <Servo.h>

#include <NewPing.h>

#define TRIGGER\_PIN 11

#define ECHO\_PIN 3

#define SERVO\_PIN 10

//CONST VALUE

#define MAX\_DISTANCE 100

#define ROTATE\_DEFAULT 90

#define ROTATE\_MIN 36

#define ROTATE\_MAX 144

#define ROTATE\_OFFSET 2

#define MAX\_SPEED 120

#define TURN\_SPEED 210

#define COLL\_DIST 20

NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE);

Servo myServo;

AF\_DCMotor leftMotor(3, MOTOR12\_8KHZ);

AF\_DCMotor rightMotor(4, MOTOR12\_8KHZ);

int pos = 0;

int maxLeft = 0;

int maxRight = 0;

int maxFront = 0;

int curDist = 0;

void setup() {

Serial.begin(9600);

myServo.attach(SERVO\_PIN);

myServo.write(ROTATE\_DEFAULT);

delay(2000);

moveForward();

}

void loop() {

curDist = getDistance();

Serial.println(curDist);

if (curDist < COLL\_DIST && curDist > 0) {

moveBackward();

delay(800);

moveStop();

selectPath();

moveForward();

}

}

void moveForward() {

leftMotor.run(FORWARD);

rightMotor.run(FORWARD);

leftMotor.setSpeed(MAX\_SPEED);

rightMotor.setSpeed(MAX\_SPEED);

}

void moveBackward() {

leftMotor.run(BACKWARD);

rightMotor.run(BACKWARD);

leftMotor.setSpeed(MAX\_SPEED);

rightMotor.setSpeed(MAX\_SPEED);

}

void moveStop() {

leftMotor.run(RELEASE);

rightMotor.run(RELEASE);

}

void turnRight() {

leftMotor.run(FORWARD);

rightMotor.run(BACKWARD);

leftMotor.setSpeed(TURN\_SPEED);

rightMotor.setSpeed(TURN\_SPEED);

delay(1000);

}

void turnLeft() {

leftMotor.run(BACKWARD);

rightMotor.run(FORWARD);

leftMotor.setSpeed(TURN\_SPEED);

rightMotor.setSpeed(TURN\_SPEED);

delay(1000);

}

void checkPath() {

for (pos = ROTATE\_DEFAULT; pos >= ROTATE\_MIN; pos -= ROTATE\_OFFSET) {

myServo.write(pos);

caculateValue();

delay(10);

}

for (pos = ROTATE\_MIN; pos <= ROTATE\_MAX; pos += ROTATE\_OFFSET) {

myServo.write(pos);

caculateValue();

delay(10);

}

for (pos = ROTATE\_MAX; pos >= ROTATE\_DEFAULT; pos -= ROTATE\_OFFSET) {

myServo.write(pos);

caculateValue();

delay(10);

}

}

void selectPath() {

maxLeft = 0;

maxRight = 0;

maxFront = 0;

checkPath();

if (maxLeft == 0) {

turnLeft();

return;

}

if (maxRight == 0) {

turnRight();

return;

}

if (maxLeft >= maxRight)

turnLeft();

else

turnRight();

}

int getDistance() {

return sonar.ping\_cm();

}

void caculateValue() {

int distance = getDistance();

//Left

if (typeOfPos() == 1) {

if (distance > 0 && distance > maxLeft) {

maxLeft = distance;

}

}

//Front

if (typeOfPos() == 2) {

if (distance > 0 && distance > maxFront) {

maxFront = distance;

}

}

//Right

if (typeOfPos() == 3) {

if (distance > 0 && distance > maxRight) {

maxRight = distance;

}

}

}

//Return value: Left 1, Front 2, Right 3, Default 0

int typeOfPos() {

if (pos <= ROTATE\_MAX && pos >= ROTATE\_MAX - (ROTATE\_MAX - ROTATE\_MIN) / 3) {

return 1;

}

if (pos < ROTATE\_MAX - (ROTATE\_MAX - ROTATE\_MIN) / 3 && pos > ROTATE\_MIN + (ROTATE\_MAX - ROTATE\_MIN) / 3) {

return 2;

}

if (pos <= ROTATE\_MIN + (ROTATE\_MAX - ROTATE\_MIN) / 3 && pos >= ROTATE\_MIN) {

return 3;

}

return 0;

}

# **PHỤ LỤC 2: TÀI LIỆU THAM KHẢO**

1. Diễn đàn

<https://www.arduino.cc/en/Tutorial/HomePage>

1. Cộng đồng Arduino VN

<http://arduino.vn/>

1. Hướng dẫn đối với Remote

<https://techmaster.vn/posts/33772/dien-tu-co-ban-lam-remote-hong-ngoai>

<http://arduino.vn/bai-viet/288-infrared-remote-control-dieu-khien-bang-hong-ngoai-voi-arduino>

https://daynhauhoc.com/t/huong-dan-lam-mot-remote-hong-ngoai-ir-remote/21046

1. Cài đặt

<http://arduino.vn/bai-viet/68-cai-dat-driver-va-arduino-ide>

1. Làm radar

<http://arduino.vn/bai-viet/904-che-tao-radar-voi-arduino>

1. Tìm hiểu về processing

<http://arduino.vn/bai-viet/tai-ve-cac-tai-nguyen#procesing>

1. Làm xe điều khiển

<http://arduino.vn/bai-viet/429-lam-du-xe-voi-module-dieu-khien-dong-co-l293d>

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

1. Một số clip trên youtube

<https://www.youtube.com/watch?v=igD1569fsoA>

<https://www.youtube.com/watch?v=G5D7LI2vvts>