

**BỘ GIÁO DỤC & ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN THỰC TẬP HỆ THỐNG NHÚNG TRONG CÔNG NGHIỆP**



BÁO CÁO MÔN HỌC

THỰC TẬP HỆ THỐNG NHÚNG TRONG CÔNG NGHIỆP

ĐỀ TÀI:

HỆ THỐNG GIỮ XE DỪNG RFID KẾT HỢP CẢM BIẾN

GVHD: TS.Nguyễn Thanh Nghĩa

SVTH: Võ Minh Thuận 21161366

Lê Quang Thương 21161367

Trần Thị Xuân Hy 21161323

Tp.HCM – 3/2024

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Thầy Nguyễn Thanh Nghĩa- người đã trực tiếp hướng dẫn chúng trong suốt quá trình thực hiện báo cáo này. Nhờ sự tận tình hướng dẫn, những góp ý quý báu của thầy, chúng em đã hoàn thành được bài báo cáo một cách tốt nhất.

Em cũng xin gửi lời cảm ơn đến các thầy cô giáo trong khoa Điện – Điện tử đã tạo điều kiện cho chúng em học tập và nghiên cứu.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp.HCM, tháng 3 năm 2024

GIÁO VIÊN HƯỚNG DẪN

MỤC LỤC

DANH MỤC HÌNH ẢNH.....	1
DANH MỤC BẢNG	3
CHƯƠNG 1: GIỚI THIỆU	4
1.1 LÝ DO CHỌN ĐỀ TÀI.....	4
1.2 MỤC TIÊU	4
1.3 GIỚI HẠN	4
1.4 PHƯƠNG PHÁP NGHIÊN CỨU	4
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	6
2.1 CÔNG NGHỆ RFID.....	6
2.2 ARDUINO.....	7
2.2.1 Giới thiệu về Arduino và Arduino Uno.....	7
2.2.2 Cấu tạo và cách sử dụng Arduino Uno.....	8
2.3 RASPBERRY	10
2.4 CHUẨN GIAO TIẾP UART.....	12
2.5 CHUẨN GIAO TIẾP I2C	15
2.6 LCD 16x2 MODE 8BIT	18
CHƯƠNG 3: TÍNH TOÁN, THIẾT KẾ VÀ THI CÔNG HỆ THỐNG.....	20
3.1 GIỚI THIỆU	20
3.2 TÍNH TOÁN VÀ THIẾT KẾ HỆ THỐNG	20
3.2.1 Thiết kế sơ đồ khối hệ thống	20
3.2.2 Tính toán và thiết kế mạch	22
3.2.3 Sơ đồ nguyên lý toàn mạch	23
3.3 THI CÔNG HỆ THỐNG.....	24

3.3.1 Thi công mạch	24
3.3.2 Lắp ráp và kiểm tra.....	24
3.4 LẬP TRÌNH HỆ THỐNG	25
3.4.1 Lập trình cho Raspberry	25
3.4.2 Lập trình cho Arduino	28
3.4.3 Lập trình cho App.....	36
CHƯƠNG 4: KẾT QUẢ, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN.....	40
4.1 KẾT QUẢ.....	40
4.2 ĐÁNH GIÁ.....	44
4.3 HƯỚNG PHÁT TRIỂN	44
TÀI LIỆU THAM KHẢO	45

DANH MỤC HÌNH ẢNH

Hình 2- 1: Ảnh mạch đọc và thẻ RFID.....	6
Hình 2- 2: Ảnh board Arduino Uno	8
Hình 2- 3: Ảnh Raspberry Pi 3	11
Hình 2- 4: Mô hình kết nối UART	12
Hình 2- 5: Hình sơ đồ dữ liệu truyền UART	14
Hình 2- 6: Hình kết nối dạng i2c	15
Hình 2- 7: Mô hình truyền dữ liệu i2c.....	16
Hình 3 - 1: Sơ đồ khối hệ thống	20
Hình 3 - 2: Hình sơ đồ nguyên lý toàn mạch	23
Hình 3 - 3: Hình thi công mạch	24
Hình 3 - 4: Hình cài đặt Sublime Text B2.1	25
Hình 3 - 5: Hình cài đặt Sublime Text B2.2.....	26
Hình 3 - 6: Hình cài đặt Sublime Text B2.3.....	26
Hình 3 - 7: hình cài đặt Sublime Text B2.4.....	26
Hình 3 - 8: Hình cài đặt Sublime Text B2.5.....	26
Hình 3 - 9: Hình cài đặt Sublime Text B2.6.....	27
Hình 3 - 10: Hình tạo dự án Sublime Text B3.1	27
Hình 3 - 11: Hình tạo dự án Sublime Text B3.2	27
Hình 3 - 12: Hình tạo dự án Sublime Text B3.3.1	27
Hình 3 - 13: hình tạo dự án Sublime Text B3.3.2	27
Hình 3 - 14: Hình tạo dự án Sublime Text B3.4	28
Hình 3 - 15: Hình cài Pi_Apps B1.1.....	28
Hình 3 - 16: Hình cài Pi-Apps B1.2	29
Hình 3 - 17: Hình cài Pi-Apps B1.3	29
Hình 3 - 18: Hình cài Arduino Ide B2.1	30
Hình 3 - 19:Hình cài Arduino Ide B2.2.....	31
Hình 3 - 20: Hình cài Arduino Ide B2.3.....	32
Hình 3 - 21: Hình cài Arduino Ide B2.4.....	32
Hình 3 - 22: Hình tạo dự án Arduino B3.1	33

Hình 3 - 23: Hình tạo dự án Arduino B3.2.1	33
Hình 3 - 24: Hình tạo dự án Arduino B3.2.2.....	34
Hình 3 - 25: Hình biên dịch chương trình Arduino B5.1	34
Hình 3 - 26: Hình biên dịch chương trình Arduino B5.2	35
Hình 3 - 27: Hình biên dịch chương trình Arduino B5.3	35
Hình 3 - 28: Hình biên dịch chương trình Arduino B5.4	36
Hình 3 - 29: Hình biên dịch App B5.1	38
Hình 3 - 30: Hình giao diện App	39
Hình 4 - 1: Hình ảnh hệ thống sau thi công.....	40
Hình 4 - 2: Hình chạy hệ thống ban đầu.....	41
Hình 4 - 3: Hình hệ thống khi có xe đi vào	42
Hình 4 - 4: Hình hệ thống khi có xe đi ra.....	42

DANH MỤC BẢNG

<i>Table 1:</i> Bảng nguồn vào, dòng vòa của các linh kiện	22
<i>Table 2:</i> Bảng số liệu thử nghiệm khi hệ thống ở chế độ vào 5 lần	43
<i>Table 3:</i> Bảng số liệu thử nghiệm khi hệ thống ở chế độ ra 5 lần	43

CHƯƠNG 1: GIỚI THIỆU

1.1 LÝ DO CHỌN ĐỀ TÀI

Giải quyết vấn đề không tìm được vị trí trống để đỗ xe khi bãi xe quá đông cũng như không tìm thấy xe mình đã đỗ ở đâu. Giảm thiểu vấn đề dung giấy gửi xe truyền thống dễ lộn, rách gây khó khăn trong việc gửi và lấy xe ra khỏi bãi. Bãi giữ xe tích hợp RFID giúp dễ dàng hơn trong việc gửi xe cụ thể như: có thể hiển thị cho người gửi biết vị trí còn trống để di chuyển nhanh hơn vào bãi, tìm kiếm chỗ mình đã đậu xe nhanh hơn qua app hiển thị, an ninh hơn trong việc gửi và lấy xe ra khỏi bãi .

Để nâng cao kiến thức về công nghệ RFID và ứng dụng của nó trong thực tế và rèn luyện kỹ năng nghiên cứu, thiết kế, và lập trình hệ thống. Tóm lại, hệ thống bãi giữ xe thông minh dùng RFID là một đề tài nghiên cứu có tính cấp thiết, khả thi, và ứng dụng cao. Việc nghiên cứu đề tài này sẽ góp phần nâng cao hiệu quả quản lý và sử dụng bãi giữ xe, đồng thời giảm thiểu tắc nghẽn giao thông và ô nhiễm môi trường. Ngoài ra cũng có thể tự tạo cơ hội thực tập, tự học hỏi và ứng dụng thực tiễn vào thực tế.

1.2 MỤC TIÊU

Hệ thống bãi giữ xe dùng RFID để quản lý chỗ đậu xe theo đúng vị trí, quản lý số lượng xe theo các khu vực đặc thù, hiển thị và giám sát từ xa.

1.3 GIỚI HẠN

- Đề tài chỉ tập trung vào nghiên cứu và thiết kế hệ thống bãi giữ xe thông minh sử dụng công nghệ RFID, không nghiên cứu các công nghệ khác có thể áp dụng cho hệ thống bãi giữ xe thông minh như: camera giám sát, hệ thống nhận diện biển số xe, v.v.

- Hệ thống chỉ được nghiên cứu và thử nghiệm trong môi trường mô phỏng có phạm vi nhỏ, có 15 thẻ quẹt ra vào và 15 chỗ trống trong bãi xe.

- Khoảng cách đo của đầu cảm biến chuyển động trong khoảng 1m

- Khoảng cách đo của đầu cảm biến vật cản trong khoảng 30cm

- Hệ thống không hoạt động song song 2 chiều (ra hoặc vào).

1.4 PHƯƠNG PHÁP NGHIÊN CỨU

- Phương pháp tham khảo: bằng cách thu thập thông tin từ sách, giáo trình và tài liệu của ngành và các môn học liên quan đến cơ sở lý thuyết của hệ thống.

- Phương pháp quan sát: khảo sát 1 số bãi xe thông minh hiện hành như ở siêu thị , trường học,... để đưa ra hướng để thiết kế ra hệ thống tốt nhất có thể trong khả năng của nhóm.

- Phương pháp thực nghiệm: từ những ý tưởng thu thập và kiến thức được học tập trên lớp kết hợp cùng giáo viên hướng dẫn, nhóm đã chọn lọc ra một số cách làm để tối ưu nhất cho hệ thống.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 CÔNG NGHỆ RFID

Công nghệ RFID hoạt động theo nguyên lý khá đơn giản, đó là: Thiết bị RFID đọc được đặt cố định ở một vị trí. Chúng sẽ phát ra sóng vô tuyến điện ở một tần số nhất định để phát hiện thiết bị phát xung quanh đó.

Khi RFID phát đi vào vùng sóng vô tuyến điện mà RFID đọc phát ra, hai bên sẽ cảm nhận được nhau. RFID phát sẽ nhận sóng điện tử, thu nhận và phát lại cho RFID đọc về mã số của mình. Nhờ vậy mà RFID đọc biết được thiết bị RFID phát nào đang nằm trong vùng hoạt động.

Bên trong thẻ chip của công nghệ RFID chứa các mã nhận dạng. Đối với thẻ 32bit có thể chứa tới 4 tỷ mã số. Khi sản xuất, mỗi một thẻ chip RFID sẽ được gán 1 mã số hoàn toàn khác nhau. Điều này sẽ giúp cho RFID đọc nhận dạng chính xác mà không bị nhầm lẫn. Chính nhờ điều này giúp cho các thiết bị đã được gán RFID mang lại độ an toàn, tính bảo mật cao.

Thiết bị RFID được cấu tạo từ 2 thành phần chính là: thiết bị phát mã đã được gắn chip và thiết bị để đọc. Thiết bị phát mã sẽ được gắn vào vật cần được nhận dạng còn thiết bị đọc sẽ được gắn anten giúp thu phát sóng điện tử. Thiết bị RFID khác nhau sẽ có mã số khác nhau và không bị trùng lặp. Khi hai thiết bị gặp nhau, tần số trùng khớp thì sẽ nhận dạng được nhau.



Hình 2- 1: Ảnh mạch đọc và thẻ RFID

Có nhiều loại thẻ RFID với các tần số và dung lượng lưu trữ khác nhau:

- Thẻ LF (Low Frequency): Tần số 125 kHz hoặc 134 kHz, dung lượng lưu trữ thấp, thường được sử dụng cho các ứng dụng như kiểm soát ra vào, quản lý thư viện.
- Thẻ HF (High Frequency): Tần số 13.56 MHz, dung lượng lưu trữ cao hơn, thường được sử dụng cho các ứng dụng như thanh toán, thẻ tín dụng.
- Thẻ UHF (Ultra High Frequency): Tần số 860 MHz - 960 MHz, tầm hoạt động xa hơn, thường được sử dụng cho các ứng dụng như theo dõi hàng hóa, quản lý kho bãi.

Đầu đọc RFID cũng có nhiều loại với các tính năng và giá cả khác nhau:

- Đầu đọc cầm tay: Dùng để đọc thẻ RFID trong phạm vi ngắn.
- Đầu đọc cố định: Dùng để lắp đặt tại các cổng ra vào hoặc khu vực cần kiểm soát.
- Đầu đọc tích hợp: Dùng để tích hợp vào các hệ thống khác như hệ thống quản lý bãi giữ xe, hệ thống quản lý kho bãi.

Công nghệ RFID có nhiều ưu điểm như: Tốc độ đọc nhanh, độ chính xác cao, tầm hoạt động xa, khả năng chống chịu môi trường tốt. Tuy nhiên, RFID cũng có một số nhược điểm như: giá thành cao hơn so với mã vạch, nguy cơ bị tấn công mạng.

Kết luận: Công nghệ RFID là một công nghệ tiên tiến có nhiều ứng dụng trong quản lý bãi giữ xe. Hệ thống RFID giúp nâng cao hiệu quả quản lý, giảm thiểu nhân lực và chi phí vận hành, đồng thời mang lại tiện lợi cho người sử dụng.

2.2 ARDUINO

2.2.1 Giới thiệu về Arduino và Arduino Uno

Arduino là một nền tảng mã nguồn mở được sử dụng để xây dựng các dự án điện tử. Arduino bao gồm cả bảng mạch lập trình (thường được gọi là vi điều khiển) và một phần mềm hoặc IDE (Môi trường phát triển tích hợp) chạy trên máy tính, được sử dụng để viết và tải mã máy tính lên bo mạch.

Nền tảng Arduino giờ đã khá phổ biến với những người mới bắt đầu với thiết bị điện tử. Không giống như hầu hết các bo mạch lập trình trước đây, Arduino không cần phần cứng riêng để tải mã mới lên bo mạch - bạn có thể chỉ cần sử dụng cáp USB. Ngoài ra, Arduino IDE sử dụng phiên bản đơn giản của C++, giúp việc học lập trình dễ dàng hơn. Arduino cung cấp một mẫu chuẩn giúp dễ tiếp cận các chức năng của bộ vi điều khiển hơn.

Uno sẽ là một lựa chọn tuyệt vời cho những ai mới làm quen với arduino. Nó có mọi thứ cần thiết để bạn bắt đầu. Nó có 14 chân đầu vào / đầu ra digital (trong đó 6 chân có

thể được sử dụng làm đầu ra PWM), 6 đầu vào analog, kết nối USB, giắc cắm nguồn, nút reset và nhiều thứ khác nữa. Nó chứa mọi thứ cần thiết để hỗ trợ vi điều khiển; chỉ cần kết nối nó với một máy tính bằng cáp USB hoặc cấp điện cho nó bằng bộ chuyển đổi dòng xoay chiều thành dòng một chiều hoặc pin.



Hình 2- 2: Ảnh board Arduino Uno

2.2.2 Cấu tạo và cách sử dụng Arduino Uno

- Nguồn (USB / Đầu cắm nguồn cái)

Mỗi bo mạch Arduino có một cách nối nguồn. Arduino UNO được cấp nguồn từ cáp USB hoặc đầu cắm nguồn cái. Cổng USB cũng hỗ trợ tải mã lên bo mạch Arduino.

LƯU Ý: KHÔNG sử dụng nguồn điện lớn hơn 20 V sẽ làm hư Arduino. Điện áp thích hợp cho hầu hết các mô hình Arduino là từ 6 đến 12 V.

- Chân (5V, 3.3V, GND, Analog, Kỹ thuật số, PWM, AREF)

Các chân trên Arduino là chỗ nối dây để xây dựng mạch (để liên kết bo mạch với dây thường có các đầu cắm bằng nhựa đen để bạn có thể cắm ngay dây vào bo mạch). Arduino có nhiều loại chân khác nhau, mỗi loại được ghi chú trên bo mạch và được sử dụng cho các chức năng khác nhau.

- GND: Viết tắt của 'Ground'. Có một số chân GND trên Arduino, có thể sử dụng bất kỳ chân nào để nối đất cho mạch.

- 5V & 3.3V: Chân 5V cấp nguồn 5V, và chân 3.3V cấp nguồn 3,3V. Hầu hết các linh kiện đơn giản sử dụng với Arduino chạy ổn định ở 3.3-5V.

- Vin: Chân cấp nguồn vào với điện áp đầu vào từ 5V đến 9V. Nên cấp điện áp vào là 6-7V.

- Analog: Khu vực các chân có ký hiệu 'Analog In' (A0 đến A5 trên UNO) là các chân nhận tín hiệu đầu vào. Các chân này có thể đọc tín hiệu từ một cảm biến tương tự (như cảm biến nhiệt độ) và chuyển đổi nó thành một giá trị số mà chúng ta có thể đọc.

- Digital: Qua khu vực các chân analog là tới các chân digital (0 đến 13 trên UNO). Các chân này sử dụng cho cả đầu vào digital (ví dụ như cho biết nút nào được nhấn) và đầu ra digital (như cấp năng lượng cho đèn LED).

- PWM: Bạn có thể thấy dấu ngã (~) bên cạnh một số chân số (3, 5, 6, 9, 10 và 11 trên UNO). Các chân này hoạt động như các chân digital thông thường, ngoài ra có thể sử dụng cho điều chế độ rộng xung (PWM).

- AREF: Là viết tắt của tham chiếu analog. Chân này thường ít được sử dụng. Thông thường nó được dùng để thiết lập điện áp tham chiếu bên ngoài (giữa 0V và 5V) làm giới hạn trên cho các chân analog đầu vào

- Nút reset

Cũng giống như Nintendo gốc, Arduino có nút reset (10). Nếu nhấn nút này sẽ tạm thời kết nối chân reset với đất và khởi động lại bất kỳ mã nào được nạp trên Arduino. Nó rất hữu dụng nếu mã của bạn không lặp lại và bạn muốn kiểm tra nó nhiều lần.

- Đèn LED báo nguồn

Ngay bên dưới và bên phải của từ “UNO” trên bảng mạch có một đèn LED nhỏ bên cạnh chữ ‘ON’. Đèn LED này sẽ sáng lên khi cắm Arduino vào nguồn điện.

- Đèn LED RX - TX

TX là viết tắt của truyền, RX là viết tắt của nhận. Những ký hiệu này xuất hiện khá nhiều trong các thiết bị điện tử để chỉ ra các chân chịu trách nhiệm về giao tiếp nối tiếp. Trong trường hợp bo mạch ở trên, có hai vị trí trên UNO Arduino nơi TX và RX xuất hiện - vị trí thứ nhất là chỗ các chân số 0 và 1, và vị trí thứ hai bên cạnh đèn LED báo TX và RX. Những đèn LED này sẽ cung cấp chỉ dẫn trực quan bất cứ khi nào Arduino nhận hoặc truyền dữ liệu.

- Mạch tích hợp - IC

IC hay mạch tích hợp có màu đen với các chân kim loại. Bạn có thể xem nó như là bộ não của Arduino. IC trên Arduino ở các bo mạch khác nhau có sự khác nhau, nhưng thường là dòng IC ATmega từ công ty ATMEL. Điều này rất quan trọng, vì bạn cần phải biết loại IC (cùng với loại bo mạch) trước khi tải lên một chương trình. Thông tin này thường được viết ở phía trên cùng của IC. Nếu bạn muốn biết thêm về sự khác biệt giữa các IC khác nhau thì có thể đọc datasheet của nó.

- Điều chỉnh điện áp

Bộ điều chỉnh điện áp là thứ bạn không có tương tác với Arduino. Nhưng nó điều chỉnh lượng điện áp được đưa vào bo mạch Arduino. Giống như người gác cổng, nó sẽ xử lý điện áp phụ có thể gây hại cho mạch. Tất nhiên, nó có giới hạn của nó, do đó, không cấp điện cho Arduino lớn hơn 20V.

- Về phần mềm lập trình

Phần mềm lập trình Arduino, hay còn gọi là Arduino IDE (Integrated Development Environment), là một môi trường phát triển tích hợp được sử dụng để viết, biên dịch và tải mã lên bo mạch Arduino. IDE Arduino là phần mềm miễn phí, mã nguồn mở và có sẵn cho các hệ điều hành Windows, macOS và Linux.

- Cách sử dụng:

- Cài đặt IDE Arduino: Tải xuống IDE Arduino từ trang web chính thức của Arduino: <https://www.arduino.cc/>.

- Kết nối bo mạch Arduino: Kết nối bo mạch Arduino với máy tính bằng cáp USB.

- Viết mã: Viết mã Arduino trong trình soạn thảo mã của IDE Arduino.

- Biên dịch mã: Nhấp vào nút "Biên dịch" để biên dịch mã.

- Tải mã: Nhấp vào nút "Tải lên" để tải mã lên bo mạch Arduino.

Giám sát: Mở Serial Monitor để giám sát hoạt động của bo mạch Arduino.

2.3 RASPBERRY

Nguyên lý hoạt động của Raspberry Pi

- Khởi động:

- Khi chúng ta cắm nguồn cho Raspberry Pi, bộ vi xử lý ARM trên bo mạch sẽ bắt đầu khởi động.

- BIOS sẽ tìm kiếm hệ điều hành trên thẻ nhớ microSD được gắn vào bo mạch.

- Nếu tìm thấy hệ điều hành, BIOS sẽ khởi động nó.

- Về hệ điều hành:

- Raspberry Pi sử dụng hệ điều hành dựa trên Linux, phổ biến nhất là Raspbian.
- Hệ điều hành này cung cấp giao diện người dùng đồ họa (GUI) và các công cụ để bạn sử dụng Raspberry Pi.

• Chúng ta có thể cài đặt thêm phần mềm để mở rộng chức năng của Raspberry Pi.

- Về chương trình:

- Có thể viết chương trình bằng Python, C, C++ hoặc các ngôn ngữ lập trình khác để điều khiển Raspberry Pi.

• Các chương trình này có thể thực hiện nhiều nhiệm vụ khác nhau, như:

- + Điều khiển các thiết bị điện tử.
- + Thu thập dữ liệu từ các cảm biến.
- + Gửi dữ liệu lên Internet.
- + Xử lý hình ảnh và video.

- Cổng kết nối:

• Raspberry Pi có nhiều cổng kết nối khác nhau, bao gồm:

- + Cổng USB để kết nối với bàn phím, chuột, ổ đĩa flash, v.v.
- + Cổng HDMI để kết nối với màn hình.
- + Cổng Ethernet để kết nối mạng.
- + Cổng GPIO để kết nối với các thiết bị điện tử khác.



Hình 2- 3: Ảnh Raspberry Pi 3

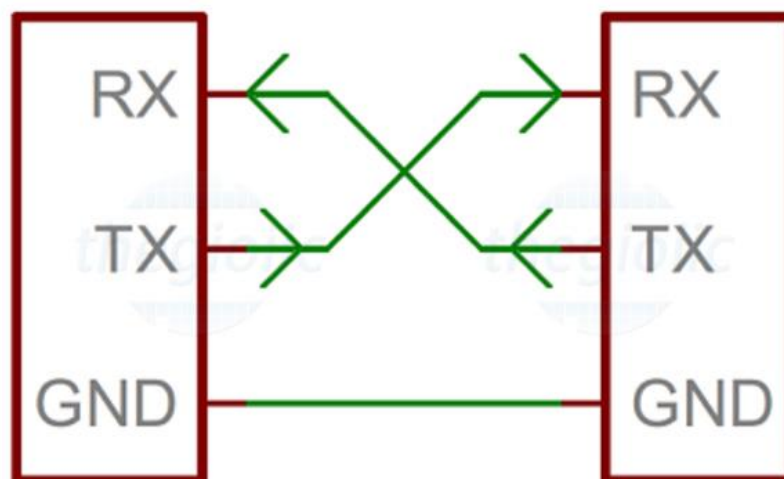
Nguyên lý hoạt động của Raspberry Pi giao tiếp với Arduino:

Có hai cách chính để Raspberry Pi giao tiếp với Arduino: là giao tiếp qua cổng USB và giao tiếp qua cổng GPIO. Nhưng ở đề tài lần này chúng ta sẽ chọn giao tiếp qua cổng USB vì nó kết nối đơn giản và phổ biến hơn. Cả hai thiết bị đều có cổng USB, vì vậy bạn chỉ cần sử dụng cáp USB để kết nối chúng với nhau. Sau khi kết nối, Raspberry Pi sẽ nhận diện Arduino như một thiết bị ngoại vi. Chúng ta có thể sử dụng các chương trình trên Raspberry Pi để gửi và nhận dữ liệu từ Arduino. Ở đây chúng ta sẽ sử dụng Raspberry Pi để đọc dữ liệu từ các cảm biến được kết nối với Arduino (cảm biến vật cản và cảm biến chuyển động) và đọc dữ liệu từ RFID.

2.4 CHUẨN GIAO TIẾP UART

UART (Universal Asynchronous Receiver-Transmitter) bộ truyền nhận dữ liệu nối tiếp bất đồng bộ, đây là một trong những giao thức truyền thông giữa thiết bị với thiết bị được sử dụng nhiều nhất. Giao tiếp UART được sử dụng nhiều trong các ứng dụng để giao tiếp với các module như: Wifi, Bluetooth, Xbee, module đầu đọc thẻ RFID với Raspberry Pi, Arduino hoặc vi điều khiển khác. Đây cũng là chuẩn giao tiếp thông dụng và phổ biến trong công nghiệp.

Trong giao tiếp UART, hai UART giao tiếp trực tiếp với nhau. UART truyền chuyển đổi dữ liệu song song từ một thiết bị điều khiển như CPU thành dạng nối tiếp, truyền nó nối tiếp đến UART nhận, sau đó chuyển đổi dữ liệu nối tiếp trở lại thành dữ liệu song song cho thiết bị nhận.



Hình 2- 4: Mô hình kết nối UART

- Hai đường dây mà mỗi thiết bị UART sử dụng để truyền dữ liệu đó là:
 - + Transmitter (Tx)
 - + Receiver (Rx)

UART truyền dữ liệu không đồng bộ, có nghĩa là không có tín hiệu đồng hồ để đồng bộ hóa đầu ra của các bit từ UART truyền đến việc lấy mẫu các bit bởi UART nhận. Thay vì tín hiệu đồng hồ, UART truyền thêm các bit start và stop vào gói dữ liệu được chuyển. Các bit này xác định điểm bắt đầu và điểm kết thúc của gói dữ liệu để UART nhận biết khi nào bắt đầu đọc các bit.

Khi UART nhận phát hiện một bit start, nó bắt đầu đọc các bit đến ở một tần số cụ thể được gọi là tốc độ truyền (baud rate). Tốc độ truyền là thước đo tốc độ truyền dữ liệu, được biểu thị bằng bit trên giây (bps – bit per second), có nhiều tốc độ truyền khác nhau từ 9600 bps -> 115200 bps. Cả hai UART đều phải hoạt động ở cùng một tốc độ truyền. Tốc độ truyền giữa UART truyền và nhận chỉ có thể chênh lệch khoảng 10% trước khi thời gian của các bit bị lệch quá xa.

Cả hai UART cũng phải được cấu hình để truyền và nhận cùng một cấu trúc gói dữ liệu.

- Cách thức hoạt động của giao tiếp UART:

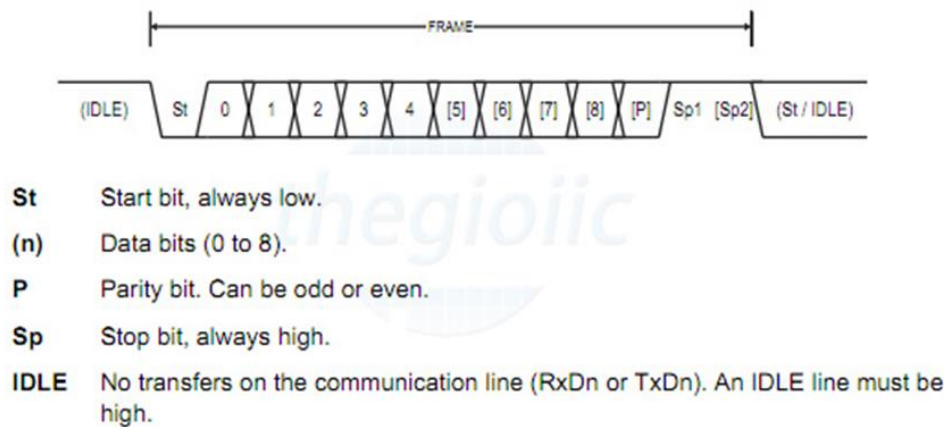
+ UART sẽ truyền dữ liệu nhận được từ một bus dữ liệu (Data Bus). Bus dữ liệu được sử dụng để gửi dữ liệu đến UART bởi một thiết bị khác như CPU, bộ nhớ hoặc vi điều khiển. Dữ liệu được chuyển từ bus dữ liệu đến UART truyền ở dạng song song. Sau khi UART truyền nhận dữ liệu song song từ bus dữ liệu, nó sẽ thêm một bit start, một bit chẵn lẻ và một bit stop, tạo ra gói dữ liệu. Tiếp theo, gói dữ liệu được xuất ra nối tiếp từng bit tại chân Tx. UART nhận đọc gói dữ liệu từng bit tại chân Rx của nó. UART nhận sau đó chuyển đổi dữ liệu trở lại dạng song song và loại bỏ bit start, bit chẵn lẻ và bit stop. Cuối cùng, UART nhận chuyển gói dữ liệu song song với bus dữ liệu ở đầu nhận.

+ UART truyền dữ liệu nối tiếp, theo một trong ba chế độ:

*Full duplex: Giao tiếp đồng thời đến và đi từ mỗi master và slave.

*Half duplex: Dữ liệu đi theo một hướng tại một thời điểm.

*Simplex: Chỉ giao tiếp một chiều.



Hình 2- 5: Hình sơ đồ dữ liệu truyền UART

Dữ liệu truyền qua UART được tổ chức thành các gói. Mỗi gói chứa 1 bit bắt đầu, 5 đến 9 bit dữ liệu (tùy thuộc vào UART), một bit chẵn lẻ tùy chọn và 1 hoặc 2 bit dừng.

- Bit khởi đầu: Đường truyền dữ liệu trong giao tiếp UART thường được giữ ở mức điện áp cao khi nó không truyền dữ liệu. Để bắt đầu truyền dữ liệu, UART truyền sẽ kéo đường truyền từ mức cao xuống mức thấp trong một chu kỳ đồng hồ. Khi UART 2 phát hiện sự chuyển đổi điện áp cao xuống thấp, nó bắt đầu đọc các bit trong khung dữ liệu ở tần số của tốc độ truyền (Baud rate).

- Khung dữ liệu: Khung dữ liệu chứa dữ liệu thực tế đang được truyền. Nó có thể dài từ 5 bit đến 8 bit nếu sử dụng bit Parity (bit chẵn lẻ). Nếu không sử dụng bit Parity, khung dữ liệu có thể dài 9 bit. Trong hầu hết các trường hợp, dữ liệu được gửi với bit LSB (bit có trọng số thấp nhất) trước tiên.

- Bit chẵn lẻ: Bit chẵn lẻ là một cách để UART nhận cho biết liệu có bất kỳ dữ liệu nào đã thay đổi trong quá trình truyền hay không. Bit có thể bị thay đổi bởi bức xạ điện từ, tốc độ truyền không khớp hoặc truyền dữ liệu khoảng cách xa. 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.

- Bit dừng: Để báo hiệu sự kết thúc của gói dữ liệu, UART gửi sẽ điều khiển đường truyền dữ liệu từ điện áp thấp đến điện áp cao trong ít nhất hai khoảng thời gian bit.

* Ưu điểm

- Chỉ sử dụng hai dây.
- Không cần tín hiệu clock.
- Có một bit chẵn lẻ để cho phép kiểm tra lỗi.
- Cấu trúc của gói dữ liệu có thể được thay đổi miễn là cả hai bên đều được thiết lập cho nó.

- Phương pháp có nhiều tài liệu và được sử dụng rộng rãi.

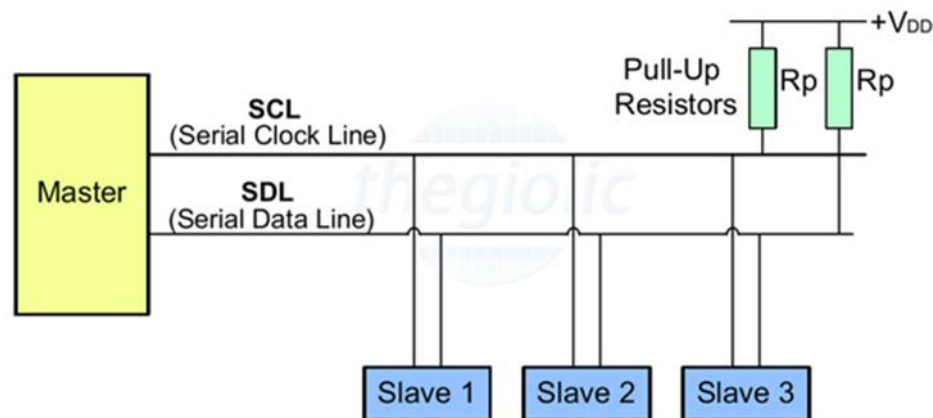
* Nhược điểm

- Kích thước của khung dữ liệu được giới hạn tối đa là 9 bit.
- Không hỗ trợ nhiều hệ thống slave hoặc nhiều hệ thống master.
- Tốc độ truyền của mỗi UART phải nằm trong khoảng 10% của nhau.

2.5 CHUẨN GIAO TIẾP I2C

I2C hay IIC (Inter – Integrated Circuit) là 1 giao thức giao tiếp nối tiếp đồng bộ được phát triển bởi Philips Semiconductors, sử dụng để truyền nhận dữ liệu giữa các IC với nhau chỉ sử dụng hai đường truyền tín hiệu.

I2C kết hợp các tính năng tốt nhất của SPI và UART. I2C có thể kết nối nhiều slave với một master duy nhất (như SPI) và có thể có nhiều master điều khiển một hoặc nhiều slave. Điều này thực sự cần thiết khi muốn có nhiều hơn một vi điều khiển ghi dữ liệu vào một thẻ nhớ duy nhất hoặc hiển thị văn bản trên một màn hình LCD.



Hình 2- 6: Hình kết nối dạng i2c

Giống như giao tiếp UART, I2C chỉ sử dụng hai dây để truyền dữ liệu giữa các thiết bị:

- SDA (Serial Data) - đường truyền cho master và slave để gửi và nhận dữ liệu.
- SCL (Serial Clock) - đường mang tín hiệu xung nhịp.

Các bit dữ liệu sẽ được truyền từng bit một dọc theo một đường duy nhất (SDA) theo các khoảng thời gian đều đặn được thiết lập bởi 1 tín hiệu đồng hồ (SCL).

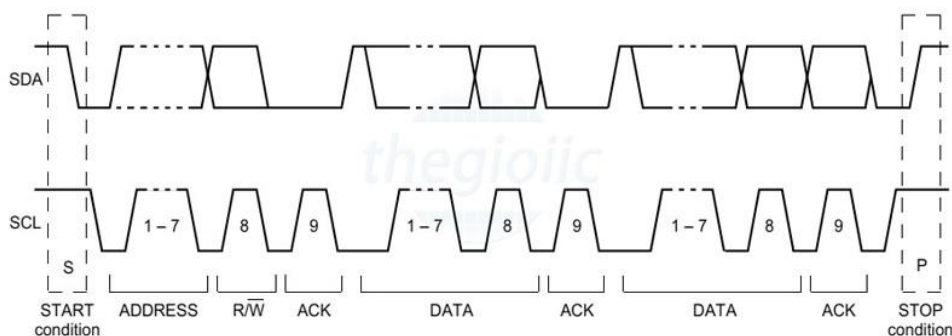
- Cách hoạt động của I2C

Giao tiếp I2C bao gồm quá trình truyền nhận dữ liệu giữa các thiết bị chủ tớ, hay Master - Slave.

Thiết bị Master là 1 vi điều khiển, nó có nhiệm vụ điều khiển đường tín hiệu SCL và gửi nhận dữ liệu hay lệnh thông qua đường SDA đến các thiết bị khác.

Các thiết bị nhận các dữ liệu lệnh và tín hiệu từ thiết bị Master được gọi là các thiết bị Slave. Các thiết bị Slave thường là các IC, hoặc thậm chí là vi điều khiển.

Master và Slave được kết nối với nhau bằng hai đường bus SCL và SDA đều hoạt động ở chế độ Open Drain, nghĩa là bất cứ thiết bị nào kết nối với mạng I2C này cũng chỉ có thể kéo 2 đường bus này xuống mức thấp (LOW), nhưng lại không thể kéo được lên mức cao. Vì để tránh trường hợp bus vừa bị 1 thiết bị kéo lên mức cao vừa bị 1 thiết bị khác kéo xuống mức thấp gây hiện tượng ngắn mạch. Do đó cần có 1 điện trở (từ 1 – 4,7 k Ω) để giữ mặc định ở mức cao.



Hình 2- 7: Mô hình truyền dữ liệu i2c

Với I2C, dữ liệu được truyền trong các tin nhắn. Tin nhắn được chia thành các khung dữ liệu. Mỗi tin nhắn có một khung địa chỉ chứa địa chỉ nhị phân của địa chỉ slave và một hoặc nhiều khung dữ liệu chứa dữ liệu đang được truyền. Thông điệp cũng bao gồm điều kiện khởi động và điều kiện dừng, các bit đọc / ghi và các bit ACK / NACK giữa mỗi khung dữ liệu:

- Điều kiện khởi động: Đường SDA chuyển từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ mức cao xuống mức thấp.
- Điều kiện dừng: Đường SDA chuyển từ mức điện áp thấp sang mức điện áp cao sau khi đường SCL chuyển từ mức thấp lên mức cao.
- Bit địa chỉ: Thông thường quá trình truyền nhận sẽ diễn ra với rất nhiều thiết bị, IC với nhau. Do đó để phân biệt các thiết bị này, chúng sẽ được gán 1 địa chỉ vật lý 7 bit cố định.

- Bit đọc / ghi: Bit này dùng để xác định quá trình là truyền hay nhận dữ liệu từ thiết bị Master. Nếu Master gửi dữ liệu đi thì ứng với bit này bằng '0', và ngược lại, nhận dữ liệu khi bit này bằng '1'.

- Bit ACK / NACK: Viết tắt của Acknowledged / Not Acknowledged. Dùng để so sánh bit địa chỉ vật lý của thiết bị so với địa chỉ được gửi tới. Nếu trùng thì Slave sẽ được đặt bằng '0' và ngược lại, nếu không thì mặc định bằng '1'.

- Bit dữ liệu: Gồm 8 bit và được thiết lập bởi thiết bị gửi truyền đến thiết bị nhận. Sau khi các bit này được gửi đi, lập tức 1 bit ACK/NACK được gửi ngay theo sau để xác nhận rằng thiết bị nhận đã nhận được dữ liệu thành công hay chưa. Nếu nhận thành công thì bit ACK/NACK được set bằng '0' và ngược lại.

- Quá trình truyền nhận:

- Khi bắt đầu Master sẽ gửi đi 1 xung Start bằng cách kéo lần lượt các đường SDA, SCL từ mức 1 xuống 0.

- Tiếp theo đó, Master gửi đi 7 bit địa chỉ tới các Slave cùng với bit Read/Write.

- Slave sẽ so sánh địa chỉ vừa được gửi tới. Nếu trùng khớp, Slave sẽ xác nhận bằng cách kéo đường SDA xuống 0 và set bit ACK/NACK bằng '0'. Nếu không trùng khớp thì SDA và bit ACK/NACK đều mặc định bằng '1'.

- Thiết bị Master sẽ gửi hoặc nhận khung bit dữ liệu. Nếu Master gửi đến Slave thì bit Read/Write ở mức 0. Ngược lại nếu nhận thì bit này ở mức 1.

- Nếu như khung dữ liệu đã được truyền đi thành công, bit ACK/NACK được set thành mức 0 để báo hiệu cho Master tiếp tục.

- Sau khi tất cả dữ liệu đã được gửi đến Slave thành công, Master sẽ phát 1 tín hiệu Stop để báo cho các Slave biết quá trình truyền đã kết thúc bằng các chuyển lần lượt SCL, SDA từ mức 0 lên mức 1.

- Các chế độ hoạt động của I2C:

- Chế độ chuẩn (standard mode) với tốc độ 100 kBit/s.

- Chế độ tốc độ thấp (low speed mode) với tốc độ 10 kBit/s.

Khác với giao tiếp SPI chỉ có thể có 1 Master, giao tiếp I2C cho phép chế độ truyền nhận dữ liệu giữa nhiều thiết bị Master khác nhau với thiết bị Slave. Tuy nhiên quá trình này có hơi phức tạp vì thiết bị Slave có thể nhận 1 lúc nhiều khung dữ liệu từ các thiết bị Master khác nhau, điều đó đôi khi dẫn đến xung đột hoặc sai sót dữ liệu nhận được.

Để tránh điều đó, khi làm việc ở chế độ này, mỗi thiết bị Master cần phát hiện xem đường SDA đang ở trạng thái nào. Nếu SDA ở mức 0, nghĩa là đang có 1 thiết bị Master khác đang có quyền điều khiển và phải chờ đến khi truyền xong. Ngược lại nếu SDA ở mức 1, nghĩa là đường truyền SDA đã an toàn và có sử dụng.

- Ưu điểm:

- Chỉ sử dụng hai dây.
- Hỗ trợ nhiều master và nhiều slave.
- Bit ACK / NACK xác nhận mỗi khung được chuyển thành công.
- Phần cứng ít phức tạp hơn so với UART.
- Giao thức nổi tiếng và được sử dụng rộng rãi.

- Nhược điểm:

- Tốc độ truyền dữ liệu chậm hơn SPI.
- Kích thước của khung dữ liệu bị giới hạn ở 8 bit.
- Cần phần cứng phức tạp hơn để triển khai so với SPI.

2.6 LCD 16x2 MODE 8BIT

Chế độ 8 bit trên LCD 16x02 cho phép bạn truyền dữ liệu đến màn hình LCD bằng cách sử dụng 8 bit dữ liệu cùng một lúc. Điều này giúp tăng tốc độ truyền dữ liệu và đơn giản hóa việc lập trình so với chế độ 4 bit.

Cấu hình chân:

- Chân D0 đến D7: Dữ liệu (8 bit).
- Chân RS: Chọn thanh ghi (Register Select).
- Chân RW: Đọc/Ghi (Read/Write).
- Chân E: Kích hoạt (Enable).

Gửi dữ liệu:

- Dữ liệu: Gửi 8 bit dữ liệu muốn hiển thị đến các chân D0-D7.
 - Chọn thanh ghi: Gửi mức 0 đến chân RS để chọn thanh ghi lệnh (command register) hoặc mức 1 để chọn thanh ghi dữ liệu (data register).
 - Đọc/Ghi: Gửi mức 0 đến chân RW để ghi dữ liệu vào LCD hoặc mức 1 để đọc dữ liệu từ LCD.
 - Kích hoạt: Gửi mức xung cao (pulse) đến chân E để kích hoạt việc ghi dữ liệu.
- Ví dụ: Để hiển thị ký tự "A" trên màn hình LCD, bạn cần thực hiện các bước sau:
- Gửi mã ASCII của ký tự "A" (0x41) đến các chân D0-D7.

- Gửi mức 1 đến chân RS để chọn thanh ghi dữ liệu.
- Gửi mức 0 đến chân RW để ghi dữ liệu.
- Gửi mức xung cao đến chân E.

Ưu điểm: Tốc độ truyền dữ liệu nhanh hơn chế độ 4 bit và lập trình đơn giản hơn.

Nhược điểm: Sử dụng nhiều chân I/O hơn chế độ 4 bit.

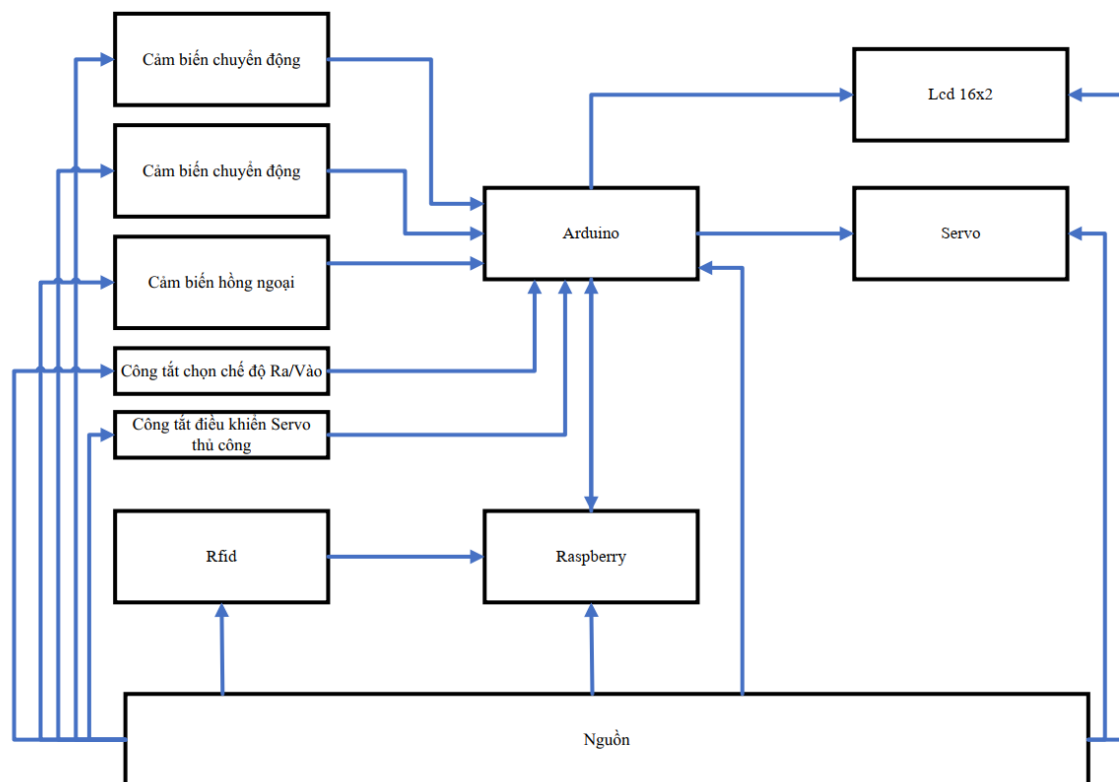
CHƯƠNG 3: TÍNH TOÁN, THIẾT KẾ VÀ THI CÔNG HỆ THỐNG

3.1 GIỚI THIỆU

Hệ thống giữ xe dùng RFID kết hợp với các cảm biến cần lựa chọn và tính toán nguồn sử dụng, độ xa của cảm biến, thiết bị để hiển thị thông tin chủ thẻ, công tắc lựa chọn chế độ, đèn báo hiệu.

3.2 TÍNH TOÁN VÀ THIẾT KẾ HỆ THỐNG

3.2.1 Thiết kế sơ đồ khối hệ thống



Hình 3 - 1: Sơ đồ khối hệ thống

- Chức năng của các khối:

- Khối Nguồn

+ Cung cấp năng lượng cho toàn bộ hệ thống

- Khối Arduino

+ Thu thập dữ liệu của các cảm biến (cảm biến chuyển động, cảm biến hồng ngoại, công tắc chọn chế độ ra vào, công tắc điều khiển Servo thủ công) và thực hiện hiển thị thông tin lên LCD và điều khiển Servo để đóng/mở Baga(cổng).

+ Giao tiếp với Raspberry để truyền và nhận dữ liệu thu thập hoặc nhận lệnh điều khiển.

- Khối Raspberry

+ Khối xử lý trung tâm, thực hiện đọc dữ liệu từ Rfid để kiểm tra id, nhận dữ liệu hoặc truyền dữ liệu qua cổng USB với Arduino, thực hiện việc tính toán logic dữ liệu và gửi dữ liệu lên firebase để lưu trữ và giám sát.

- Khối Rfid

+ Cảm biến rfid dùng để quét thẻ nhận giá trị là mã định danh, họ và tên người chủ thẻ để cho phép đóng mở thẻ

- Khối Cảm biến chuyển động

+ Cảm biến sẽ đọc dữ liệu chuyển động và chuyển đổi thành điện áp 1 là có chuyển động, 0 là không có chuyển động

+ Cảm biến 1 là nhận diện người đi vào

+ Cảm biến 2 là nhận diện người đi ra

- Khối công tắc

+ Công tắc ra/vào: chọn chế độ ra vào

+ Công tắc điều khiển Servo thủ công là điều khiển Servo mà không cần quét thẻ rfid

- Khối cảm biến hồng ngoại

+ Cảm biến giúp xác định người đã đi vào hoặc đi ra hay chưa

- Khối lcd 16x2

+ Lcd hiển thị những thông tin cho người vào hoặc đi ra biết trạng thái đậu xe là vị trí đậu xe ở đâu và trạng thái đi ra là biết mình đã được xác nhận đi ra hay chưa.

- Thông số vào ra của các khối:

- Khối nguồn

+ Nguồn vào: 12v

+ Nguồn ra: 5v

- Khối Arduino

+ Nguồn vào: 5v

+ Nguồn ra: 5v

- Khối Cảm biến chuyển động

- + Nguồn vào 5v
- + Nguồn ra 3.3v-5v
- Khối Cảm biến hồng ngoại
 - + Nguồn ngõ vào 5v
 - + Nguồn ngõ ra 3.5v-5v
- Khối Servo
 - + Nguồn vào 5v

3.2.2 Tính toán và thiết kế mạch

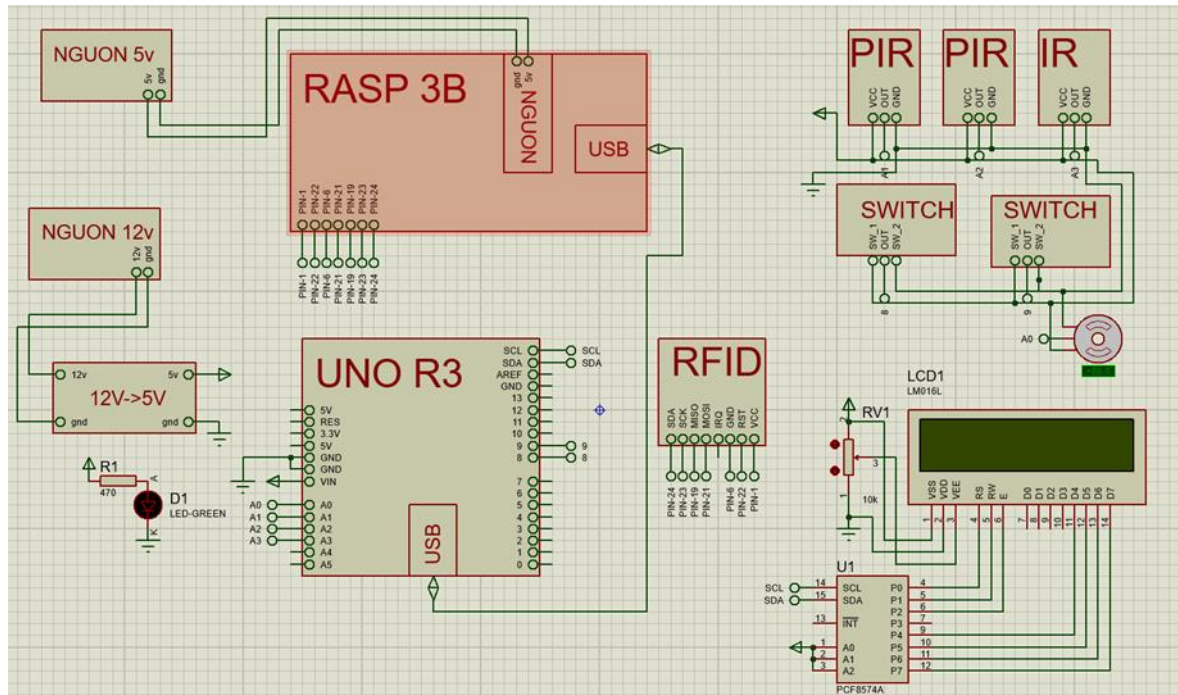
Table 1: Bảng nguồn vào, dòng vào của các linh kiện

Tên	Nguồn vào	Dòng vào
Arduino Uno R3	5v	$30\text{mA} \times 8(\text{gpios}) = 240\text{mA}$
Cảm biến hồng ngoại	5v	$\sim 400\mu\text{A}$
Cảm biến chuyển động	5v	$50\mu\text{A}$
Servo	5v	Hiện tại (không hoạt động) 10mA Dòng điện (điễn hình khi di chuyển) $100\text{-}250\text{mA}$ Hiện tại (ngưng) 360mA
Lcd	5v	120mA
Led – báo nguồn	5v	18mA
Tổng	5V	$\sim 740\text{mA}$

Bảng trên cho ta thấy được dòng vào của từng linh kiện từ đó ta có thể tính được nguồn vào như thế nào là phù hợp.

- Ta dùng nguồn vào 5V và tổng dòng được tính bằng tổng dòng vào của các linh kiện:
 $\text{Dòng tổng} = 240\text{mA} + 400\mu\text{A} + 50\mu\text{A} + 360\text{mA} + 120\text{mA} + 18\text{mA} = 740\text{mA}.$
- Từ đó ta chọn được nguồn vào với điện áp là 5V và dòng tối thiểu là 740mA nên chọn nguồn có dòng từ 1A trở nên cho hệ thống hoạt động ổn định.

3.2.3 Sơ đồ nguyên lý toàn mạch

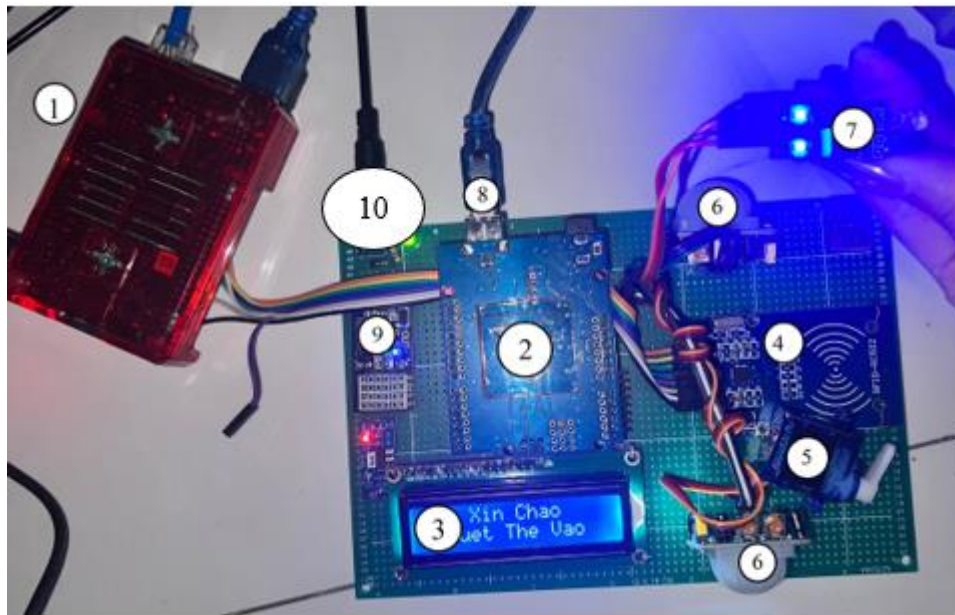


Hình 3 - 2: Hình sơ đồ nguyên lý toàn mạch

- Khối nguồn 5v và 12v cấp nguồn cho các thiết bị
- Sử dụng 7 chân của Raspberry để kết nối với RFID
- Kết nối Raspberry với Arduino Uno qua cổng USB .
- Arduino Uno sử dụng 2 chân SDA và SCL để giao tiếp với LCD 16x2 thông qua chuẩn giao tiếp I2C nên chỉ giao tiếp qua 2 chân.
- LCD 16x2 kết nối với modul I2C qua 7 chân để truyền dữ liệu hiển thị lên LCD và LCD chạy với chế độ 4bit.
- 2 switch kết nối với Arduino để có thể điều chỉnh lựa chọn chế độ ra hay vào của hệ thống (1 switch để chọn chế độ vào/ra, 1 cái điều khiển Servo thủ công).
- PIR (cảm biến chuyển động), IR (cảm biến vật cản) và Servo kết nối với Arduino Uno qua chân Out.

3.3 THI CÔNG HỆ THỐNG

3.3.1 Thi công mạch



Hình 3 - 3: Hình thi công mạch

- Sơ đồ bố trí các linh kiện

1. Raspberry Pi3
2. Arduino Uno R3
3. LCD 16x2
4. RFID RC522
5. Servo
6. Cảm biến vật cản
7. Cảm biến chuyển động
8. Cổng USB
9. MP2482 mạch ổn áp 5V
10. Nguồn cung cấp 12V

3.3.2 Lắp ráp và kiểm tra

Lắp ráp theo qui trình sau đây:

1. Hàn jack cắm nguồn, hàn khối nguồn chuyển đổi 12V sang 5V, có gắn diot bảo vệ đầu vào của khối nguồn 12V sang 5V. Kiểm tra nguồn ở đầu ra là 5V hay không, nếu đúng hàn tiếp. Ở ngõ ra 5V nối với trở 470 ôm và led báo có nguồn.

2. Hàn hàng rào cấm cho Arduino và nối chân nguồn của Arduino với đầu ra 5V của khối nguồn, kiểm tra thông mạch của các chân cấm xem có bị chạm hay không, khi không bị chạm ta cấp nguồn vào jack nguồn và đo điện áp ở 2 chân được hàn cấm với Arduino xem đúng 5V hay chưa, sau đó cấm Arduino vào hàng rào đã được hàn. Kiểm tra xem led nguồn của Arduino có sáng hay không

3. Hàn hàng rào để kết nối cảm biến, công tắc và servo, hàn các dây nguồn, dây tín hiệu của cảm biến với các GPIO của Arduino. Sau đó kiểm tra xem có thông mạch xem có bị chạm chân hay không. Tiếp theo hàn màn hình LCD vào board mạch, hàn các chân tín hiệu với các chân tín hiệu của Arduino và kiểm tra thông mạch.

4. Kiểm tra lại các chân kết nối hàn đúng với các chân tín hiệu của Arduino đã được khai báo, định nghĩa trong chương trình code.

5. Gắn tất cả các linh kiện vào board mạch và kiểm tra đánh giá các tín hiệu của cảm biến, dữ liệu hiển thị trên LCD

3.4 LẬP TRÌNH HỆ THỐNG

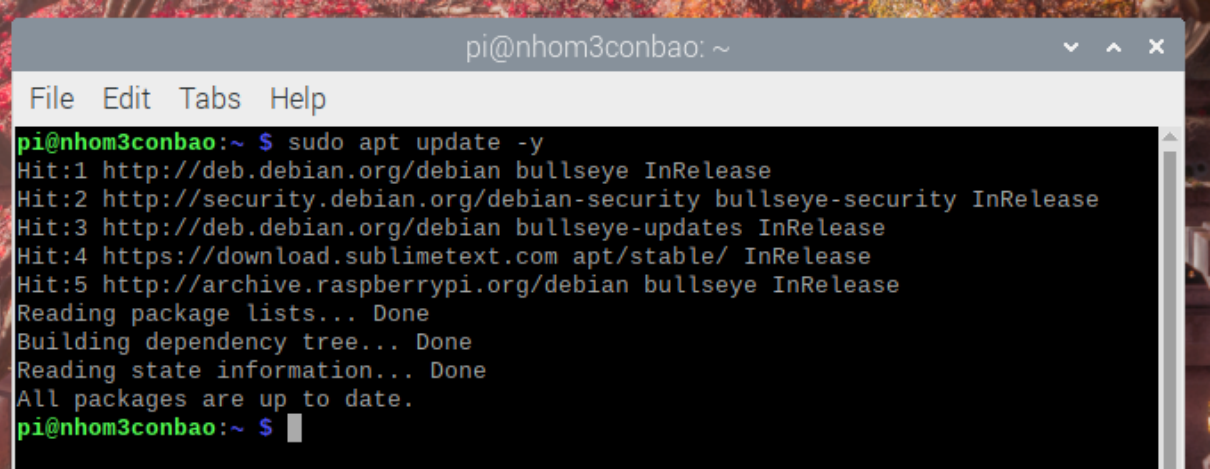
3.4.1 Lập trình cho Raspberry

B1. Cài hệ điều hành cho Rasp

Bạn cài đặt theo đúng hướng dẫn của mình theo link sau: https://github.com/VM-Thuan-2003/TT_HTN_SPK/blob/main/LESSON_WEEK_1/REPORT_WEEK_1_GROUP_4.pdf

B2. Cài Sublime Text cho Rasp

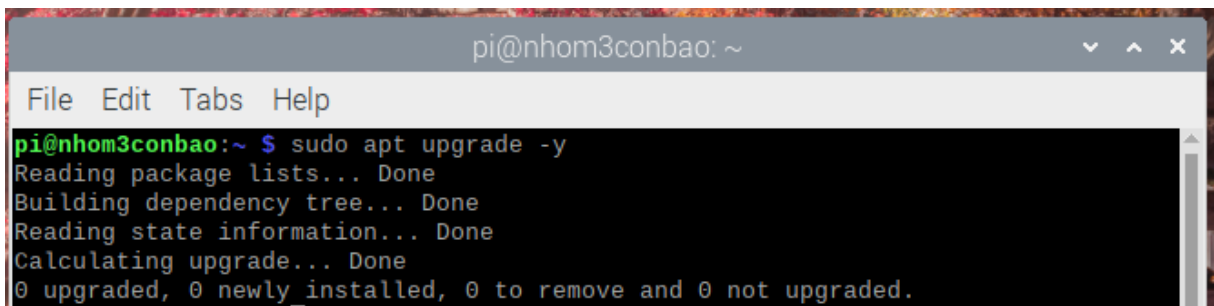
- B21. Nhập lệnh `sudo apt update -y`



```
pi@nhom3conbao: ~  
File Edit Tabs Help  
pi@nhom3conbao:~ $ sudo apt update -y  
Hit:1 http://deb.debian.org/debian bullseye InRelease  
Hit:2 http://security.debian.org/debian-security bullseye-security InRelease  
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease  
Hit:4 https://download.sublimetext.com apt/stable/ InRelease  
Hit:5 http://archive.raspberrypi.org/debian bullseye InRelease  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
All packages are up to date.  
pi@nhom3conbao:~ $
```

Hình 3 - 4: Hình cài đặt Sublime Text B2.1

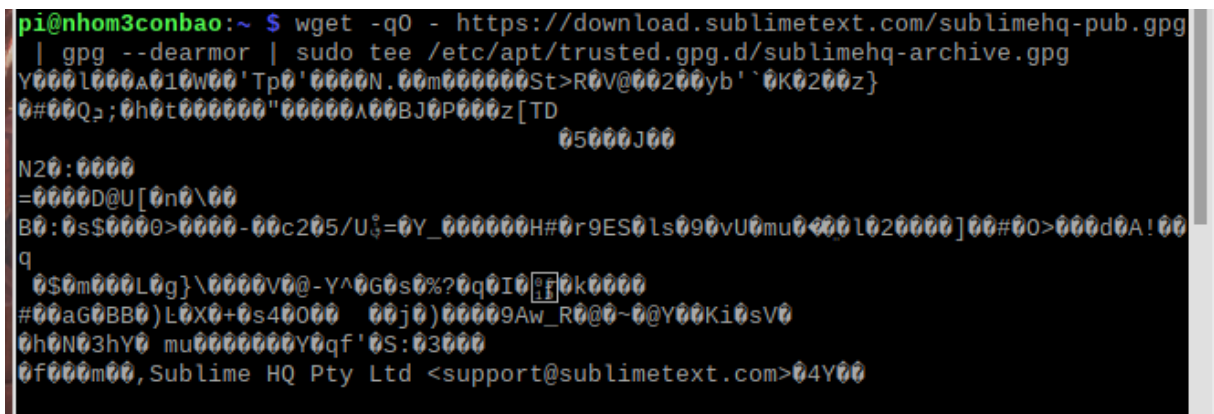
- B22. Nhập lệnh `sudo apt upgrade -y`



```
pi@nhom3conbao: ~  
File Edit Tabs Help  
pi@nhom3conbao:~ $ sudo apt upgrade -y  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Calculating upgrade... Done  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Hình 3 - 5: Hình cài đặt Sublime Text B2.2

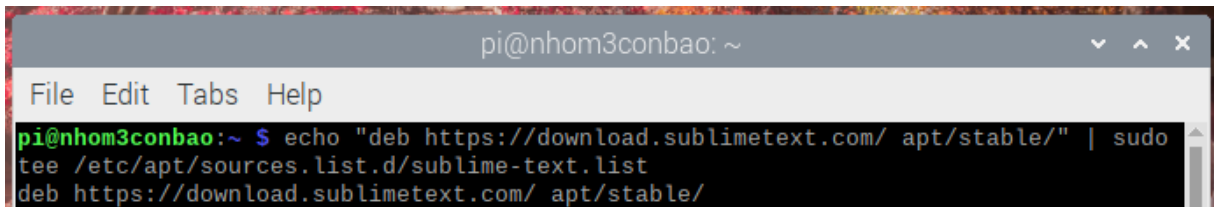
- B23. Nhập lệnh `wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/sublimehq-archive.gpg`



```
pi@nhom3conbao:~ $ wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg  
| gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/sublimehq-archive.gpg  
Y000l000A010W00'Tp0'0000N.00m000000St>R0V000200yb'`0K0200z}  
0#00Q;0h0t000000"00000A00BJ0P000z[TD  
05000J00  
N20:0000  
=0000D@U[0n0\00  
B0:0s$0000>0000-00c205/U0=0Y_000000H#0r9ES0ls090vU0mu000l020000]00#00>000d0A!00  
q  
0$0m000L0g}\0000V00~-Y^0G0s0%?0q0I00f0k0000  
#00aG0BB0)L0X0+0s40000 00j0)00009Aw_R0@0~0@Y00Ki0sV0  
0h0N03hY0 mu0000000Y0qf'0S:03000  
0f000m00,Sublime HQ Pty Ltd <support@sublimetext.com>04V00
```

Hình 3 - 6: Hình cài đặt Sublime Text B2.3

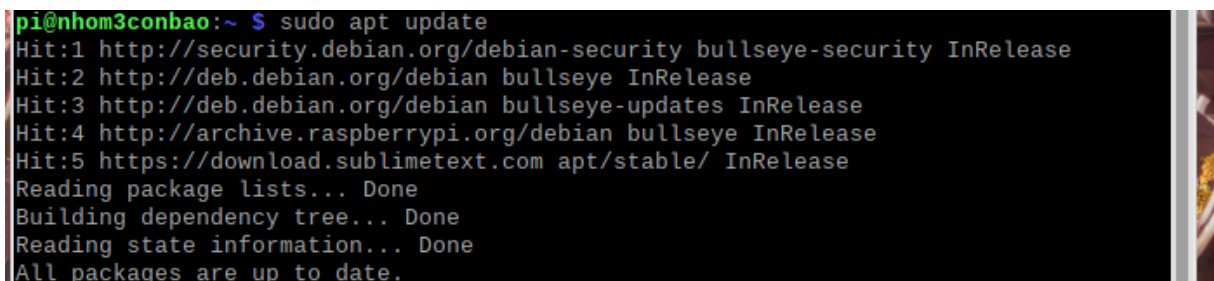
- B24. Nhập lệnh `echo "deb https://download.sublimetext.com/ apt/stable/" | sudo tee /etc/apt/sources.list.d/sublime-text.list`



```
pi@nhom3conbao: ~  
File Edit Tabs Help  
pi@nhom3conbao:~ $ echo "deb https://download.sublimetext.com/ apt/stable/" | sudo  
tee /etc/apt/sources.list.d/sublime-text.list  
deb https://download.sublimetext.com/ apt/stable/
```

Hình 3 - 7: hình cài đặt Sublime Text B2.4

- B25. Nhập lệnh `sudo apt update`



```
pi@nhom3conbao:~ $ sudo apt update  
Hit:1 http://security.debian.org/debian-security bullseye-security InRelease  
Hit:2 http://deb.debian.org/debian bullseye InRelease  
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease  
Hit:4 http://archive.raspberrypi.org/debian bullseye InRelease  
Hit:5 https://download.sublimetext.com apt/stable/ InRelease  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
All packages are up to date.
```

Hình 3 - 8: Hình cài đặt Sublime Text B2.5

- B26. Nhập lệnh `sudo apt install sublime-text -y`


```

pi@nhom3conbao:~ $ sudo apt install sublime-text -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
sublime-text is already the newest version (4169).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

```

Hình 3 - 9: Hình cài đặt Sublime Text B2.6

B3. Tạo dự án

- B31. Vào terminal nhập lệnh cd path với path là đường dẫn lưu dự án

```

pi@nhom3conbao: ~/Desktop/TT_HTN_SPK/PROJECT_CK/CODE
File Edit Tabs Help
pi@nhom3conbao:~ $ cd Desktop/TT_HTN_SPK/PROJECT_CK/CODE/
pi@nhom3conbao:~/Desktop/TT_HTN_SPK/PROJECT_CK/CODE $

```

Hình 3 - 10: Hình tạo dự án Sublime Text B3.1

- B32. Nhập lệnh mkdir Tên dự án (không dấu) để tạo folder để thực hiện dự án

```

pi@nhom3conbao: ~/Desktop/TT_HTN_SPK/PROJECT_CK/CODE
File Edit Tabs Help
pi@nhom3conbao:~ $ cd Desktop/TT_HTN_SPK/PROJECT_CK/CODE/
pi@nhom3conbao:~/Desktop/TT_HTN_SPK/PROJECT_CK/CODE $ mkdir CODE_PY

```

Hình 3 - 11: Hình tạo dự án Sublime Text B3.2

- B33. Nhập lệnh cd Tên dự án và nhập tiếp lệnh subl để mở folder trong sublime Text để thực hiện viết chương trình

```

pi@nhom3conbao: ~/Desktop/TT_H...SPK/PROJECT_CK/CODE/CODE_PY
File Edit Tabs Help
pi@nhom3conbao:~ $ cd Desktop/TT_HTN_SPK/PROJECT_CK/CODE/
pi@nhom3conbao:~/Desktop/TT_HTN_SPK/PROJECT_CK/CODE $ mkdir CODE_PY
pi@nhom3conbao:~/Desktop/TT_HTN_SPK/PROJECT_CK/CODE $ cd CODE_PY
pi@nhom3conbao:~/Desktop/TT_HTN_SPK/PROJECT_CK/CODE/CODE_PY $ subl

```

Hình 3 - 12: Hình tạo dự án Sublime Text B3.3.1

```

untitled • (CODE_PY) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

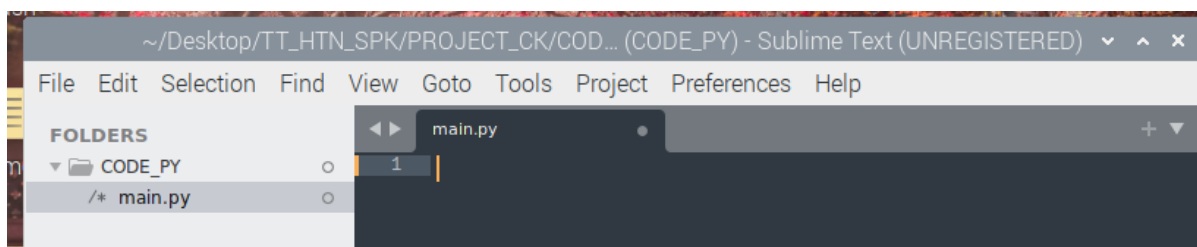
FOLDERS
▼ CODE_PY

1

```

Hình 3 - 13: hình tạo dự án Sublime Text B3.3.2

- B34. Ở thanh làm việc bên trái ta nhấn chuột phải chọn new file vào tạo file main.py là file code chính ở đó.



Hình 3 - 14: Hình tạo dự án Sublime Text B3.4

B4. Viết chương trình code

[https://github.com/VM-Thuan-](https://github.com/VM-Thuan-2003/TT_HTN_SPK/blob/main/PROJECT_CK/CODE/main.py)

[2003/TT_HTN_SPK/blob/main/PROJECT_CK/CODE/main.py](https://github.com/VM-Thuan-2003/TT_HTN_SPK/blob/main/PROJECT_CK/CODE/main.py)

B5. Biên dịch chương trình

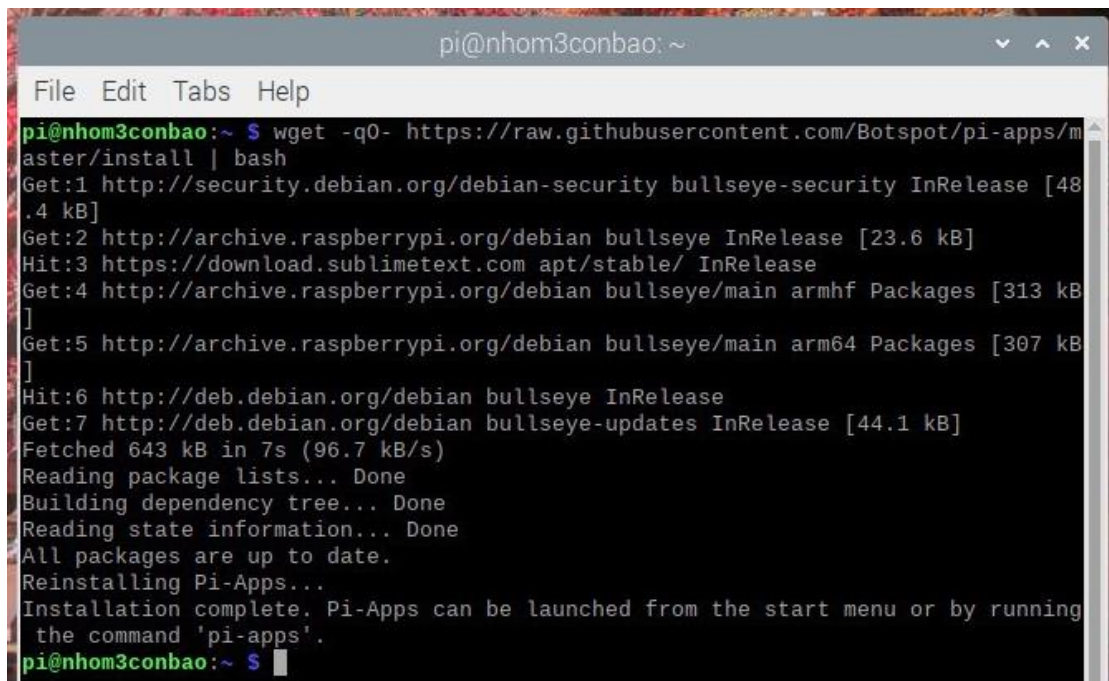
- Ta nhấn tổ hợp phím ctrl + B để chạy chương trình trong sublime Text.
- Ta có thể chạy chương trình ở ngoài terminal được trở vào dự án hiện tại bằng lệnh `python main.py`

3.4.2 Lập trình cho Arduino

B1. Cài Pi_Apps

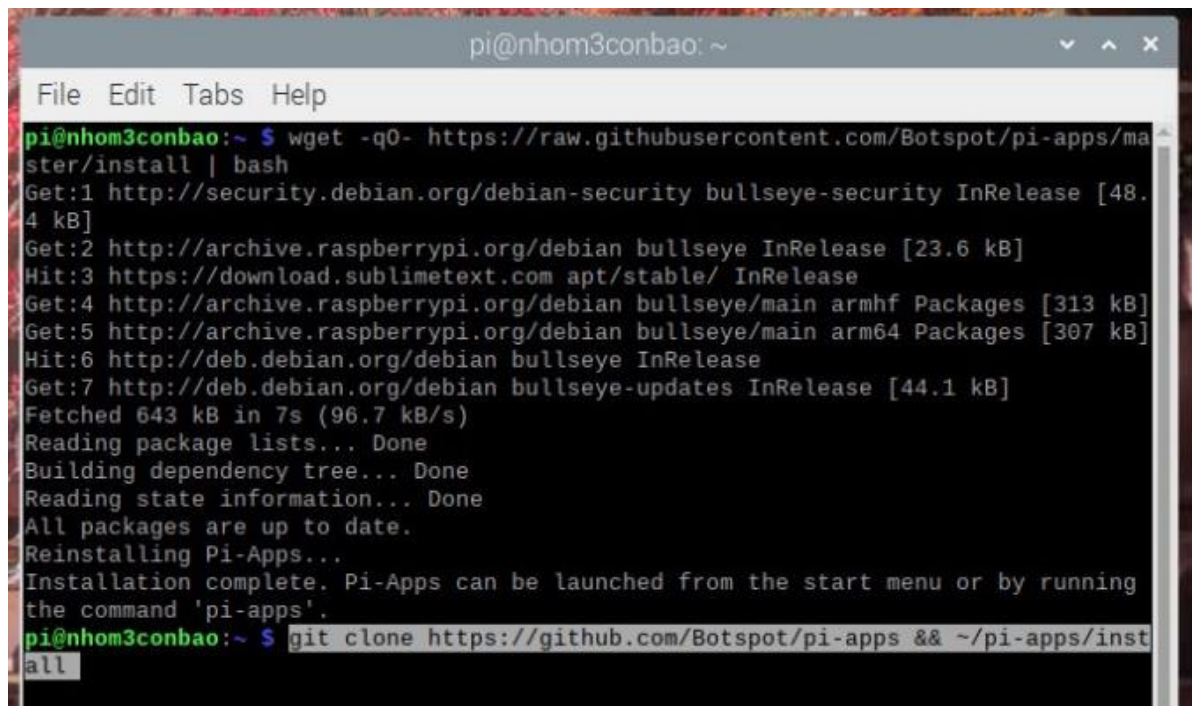
- B11. Vào terminal và nhập lệnh

`wget -qO- https://raw.githubusercontent.com/Botspot/pi-apps/master/install | bash`



Hình 3 - 15: Hình cài Pi_Apps B1.1

- B12. Sau đó nhập lệnh `git clone https://github.com/Botspot/pi-apps && ~/pi-apps/install`



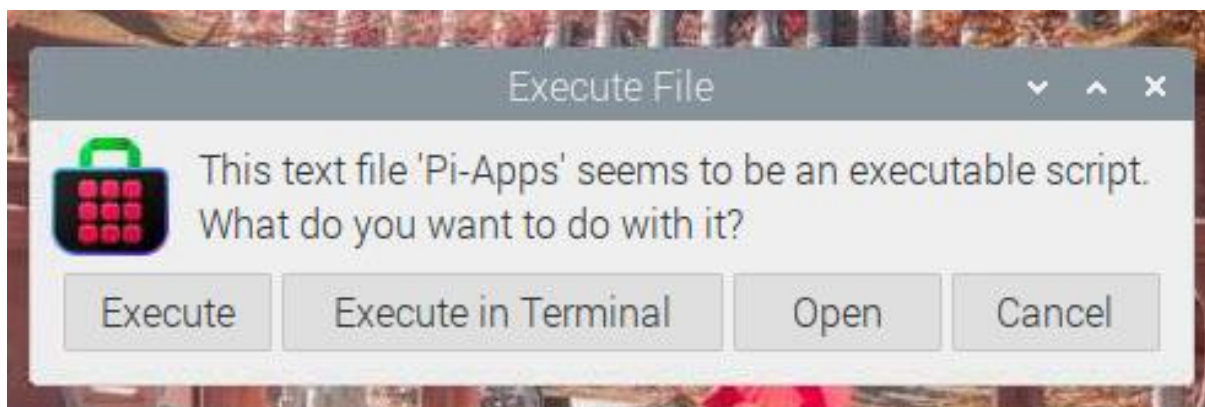
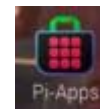
```

pi@nhom3conbao:~ $ wget -qO- https://raw.githubusercontent.com/Botspot/pi-apps/master/install | bash
Get:1 http://security.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:2 http://archive.raspberrypi.org/debian bullseye InRelease [23.6 kB]
Hit:3 https://download.sublimetext.com apt/stable/ InRelease
Get:4 http://archive.raspberrypi.org/debian bullseye/main armhf Packages [313 kB]
Get:5 http://archive.raspberrypi.org/debian bullseye/main arm64 Packages [307 kB]
Hit:6 http://deb.debian.org/debian bullseye InRelease
Get:7 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Fetched 643 kB in 7s (96.7 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reinstalling Pi-Apps...
Installation complete. Pi-Apps can be launched from the start menu or by running the command 'pi-apps'.
pi@nhom3conbao:~ $ git clone https://github.com/Botspot/pi-apps && ~/pi-apps/install

```

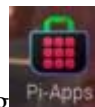
Hình 3 - 16: Hình cài Pi-Apps B1.2

- B13. Thành công thì màn hình Desktop hiển thị biểu tượng

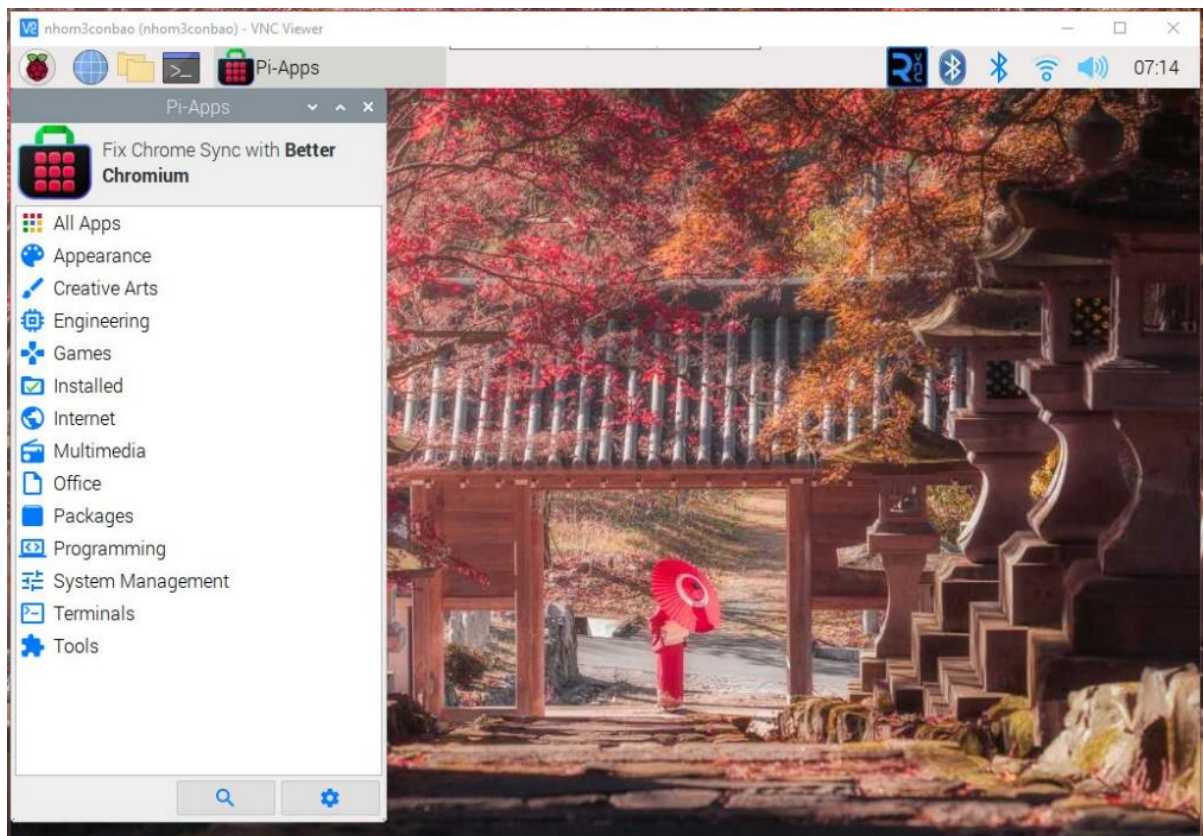


Hình 3 - 17: Hình cài Pi-Apps B1.3

B2. Cài Arduino Ide

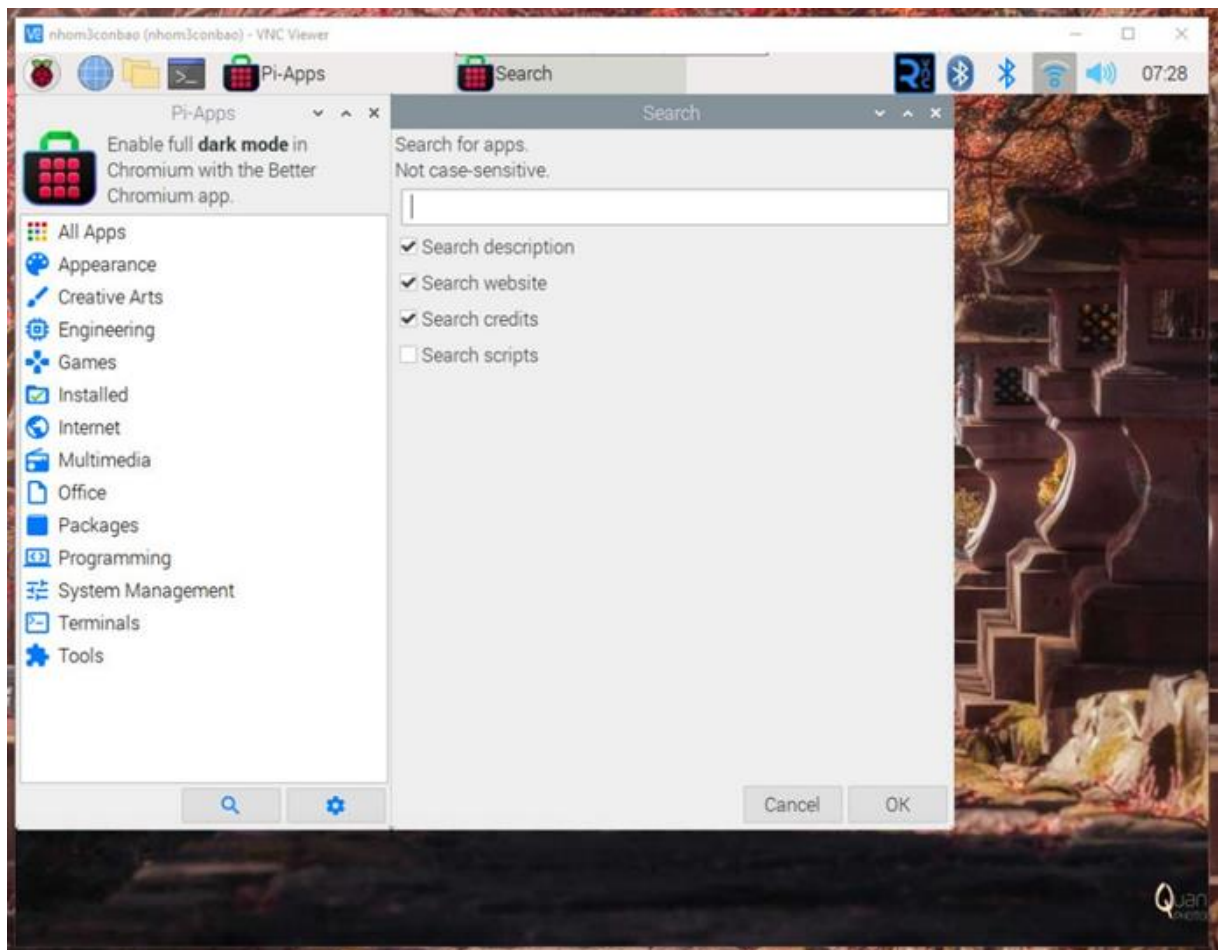


- B21. Ta nhấp đúp chuột vào biểu tượng và chọn execute để mở pi-Apps



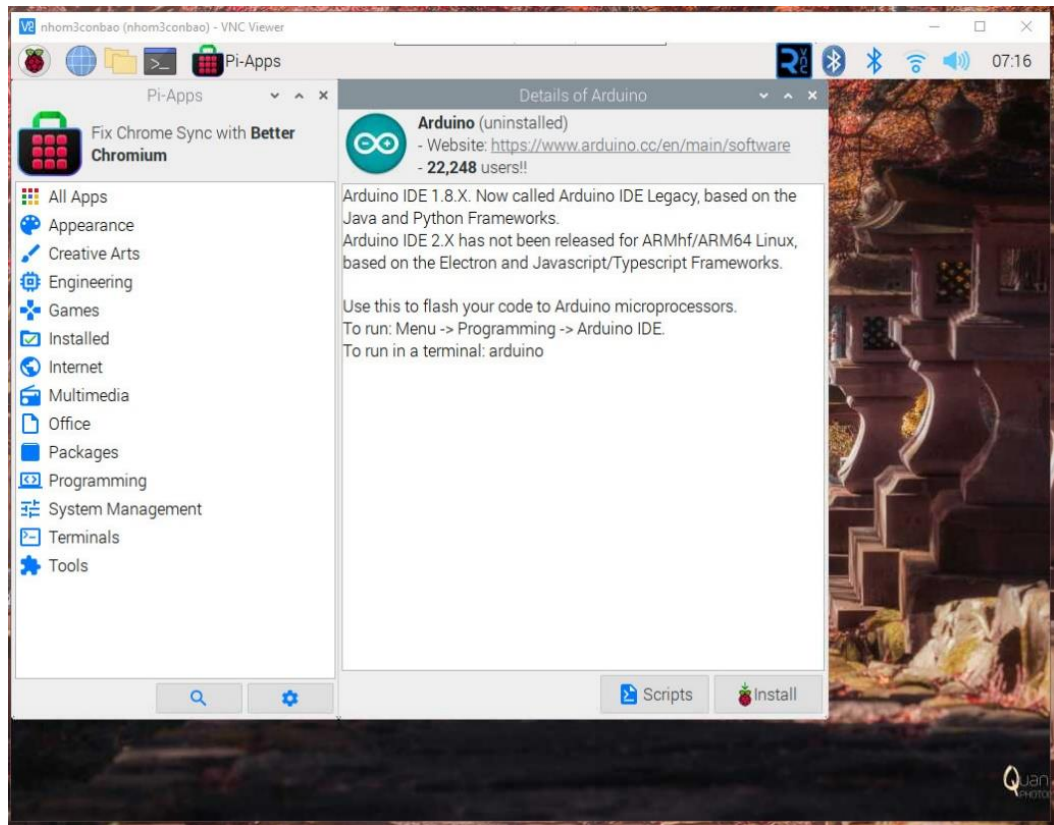
Hình 3 - 18: Hình cài Arduino Ide B2.1

- B22. Nhấn vào ô tìm kiếm nhập Arduino và nhấn tìm kiếm



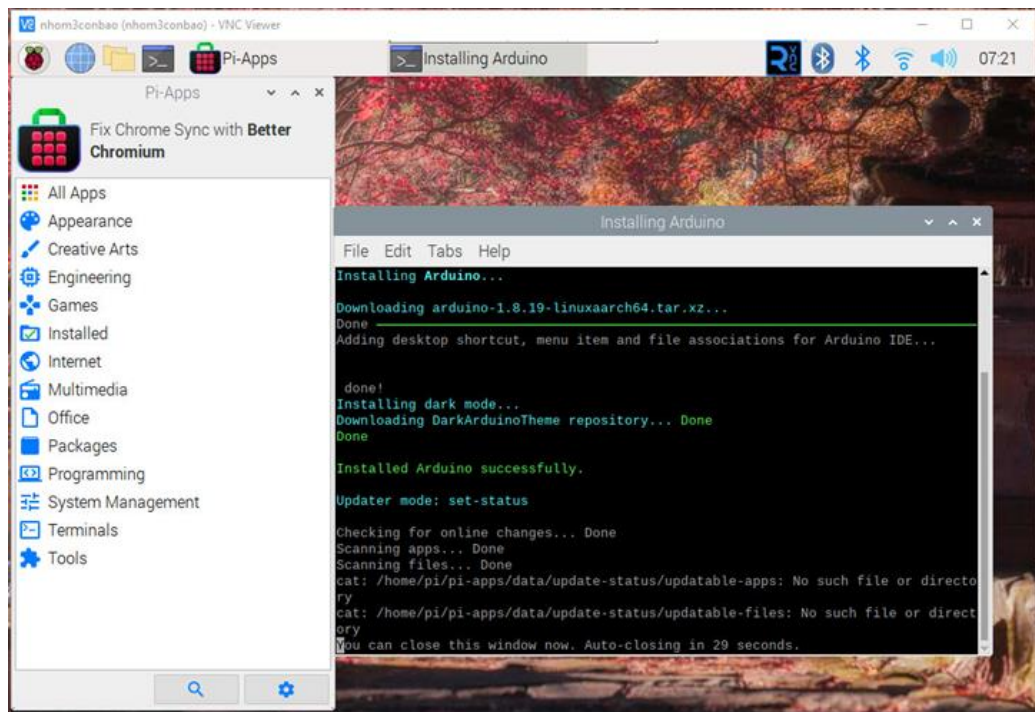
Hình 3 - 19:Hình cài Arduino Ide B2.2

- B23. Nhấn vào biểu tượng Arduino và chọn Install để cài đặt Arduino cho Rasp



Hình 3 - 20: Hình cài Arduino Ide B2.3

- B24. Khi cài thành công thì biểu tượng Arduino Ide sẽ xuất hiện trong programming của raspberry.



Hình 3 - 21: Hình cài Arduino Ide B2.4

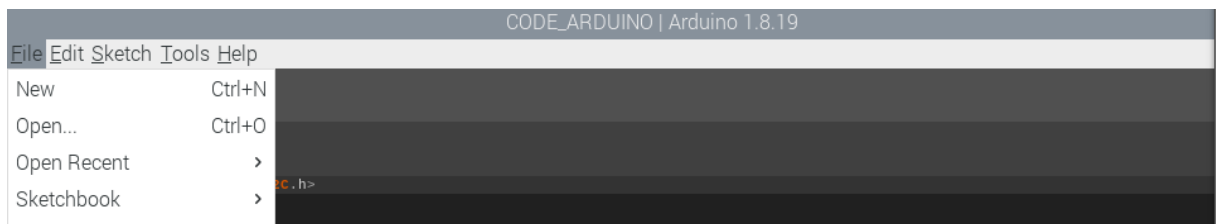
B3. Tạo dự án

- B31. Ta vào programming và nhấn đúp chuột vào biểu tượng Arduino Ide để mở Arduino Ide để làm việc

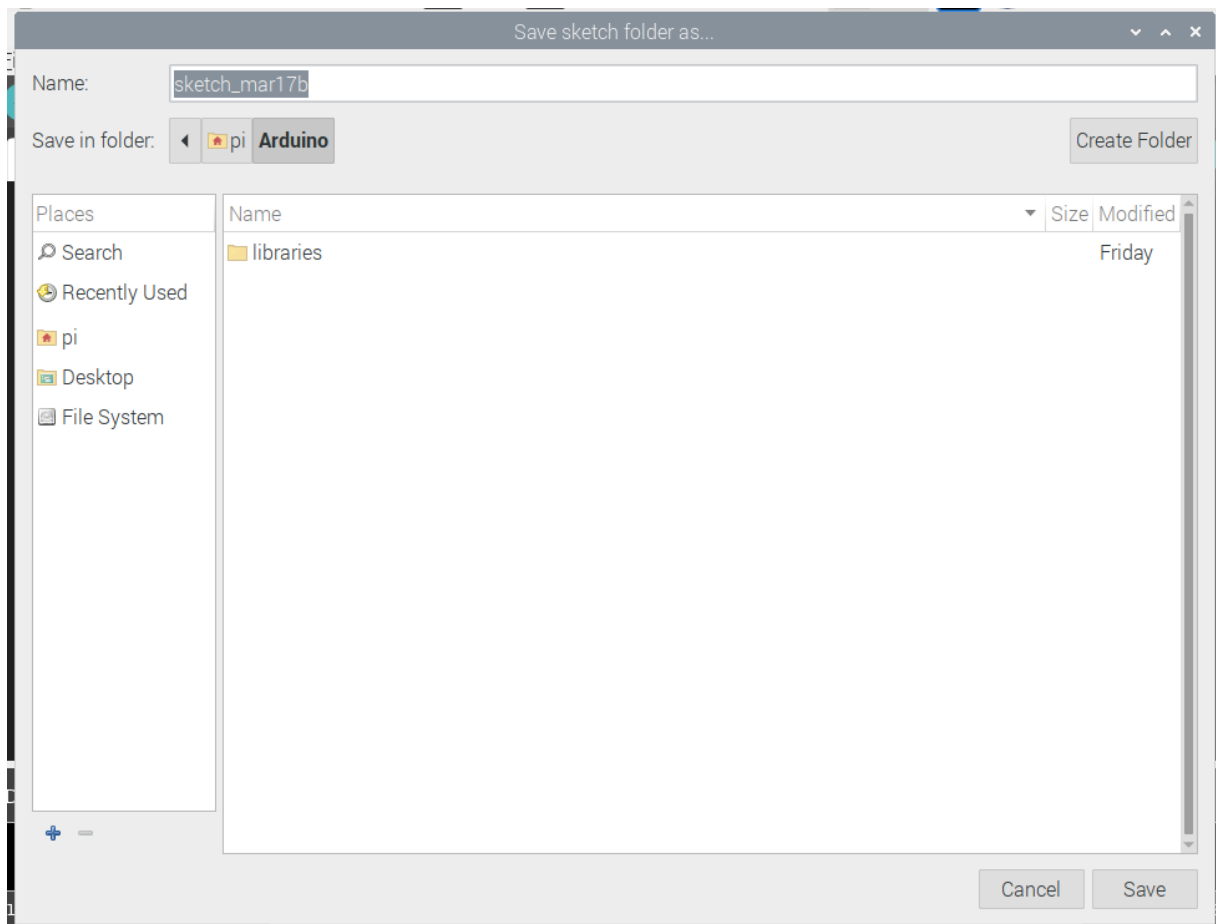


Hình 3 - 22: Hình tạo dự án Arduino B3.1

- B32. Ta chọn file->new để tạo dự án mới sau đó nhấn ctrl + S để lưu vào nơi mà ta muốn.



Hình 3 - 23: Hình tạo dự án Arduino B3.2.1



Hình 3 - 24: Hình tạo dự án Arduino B3.2.2

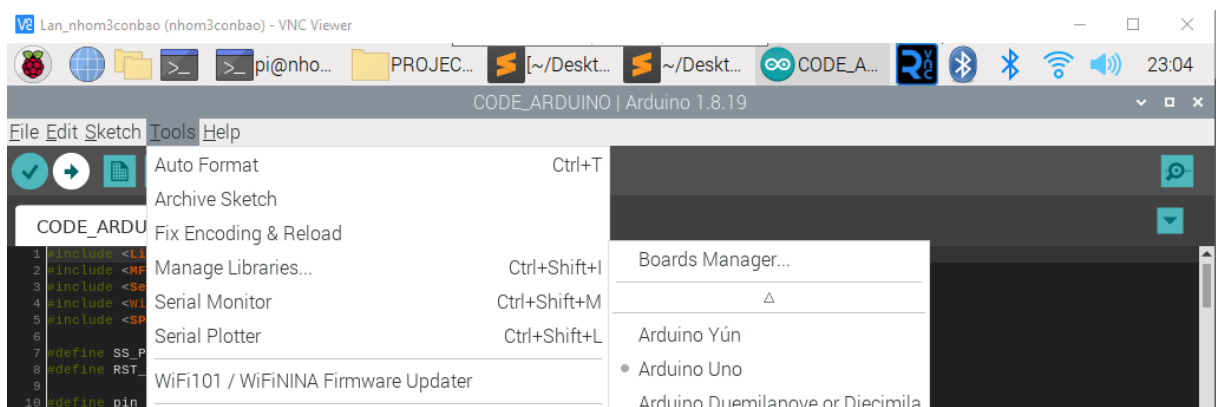
B4. Viết chương trình

[https://github.com/VM-Thuan-](https://github.com/VM-Thuan-2003/TT_HTN_SPK/blob/main/PROJECT_CK/CODE/CODE_ARDUINO/CODE_ARDUINO.ino)

[2003/TT_HTN_SPK/blob/main/PROJECT_CK/CODE/CODE_ARDUINO/CODE_ARDUINO.ino](https://github.com/VM-Thuan-2003/TT_HTN_SPK/blob/main/PROJECT_CK/CODE/CODE_ARDUINO/CODE_ARDUINO.ino)

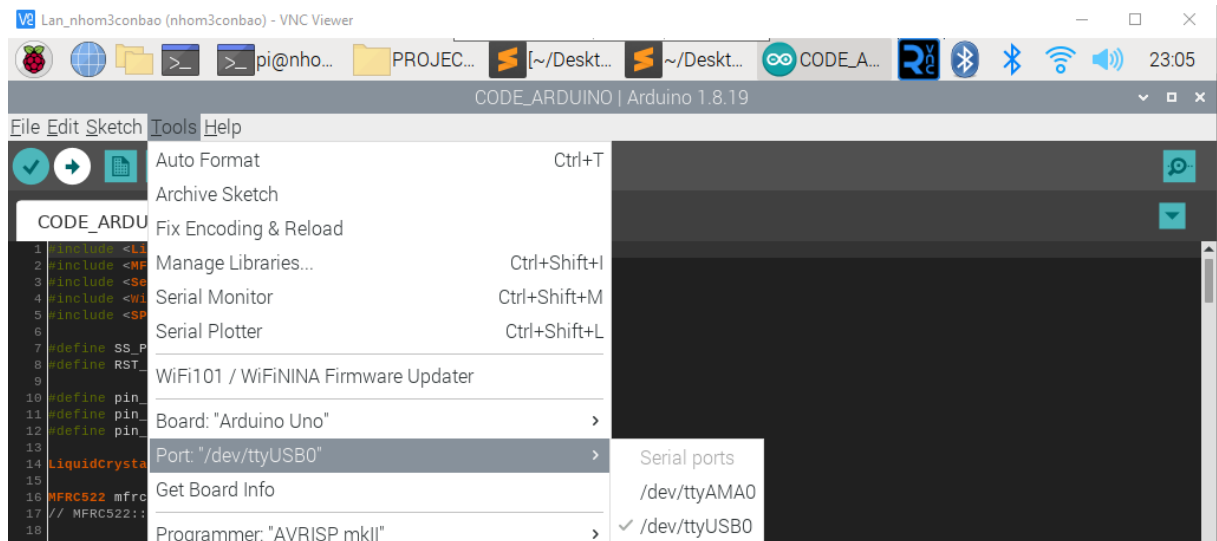
B5. Biên dịch chương trình

- B5.1. Ta biên dịch lên Arduino Uno Nên ta chọn board là Arduino Uno ở phần tool->board.



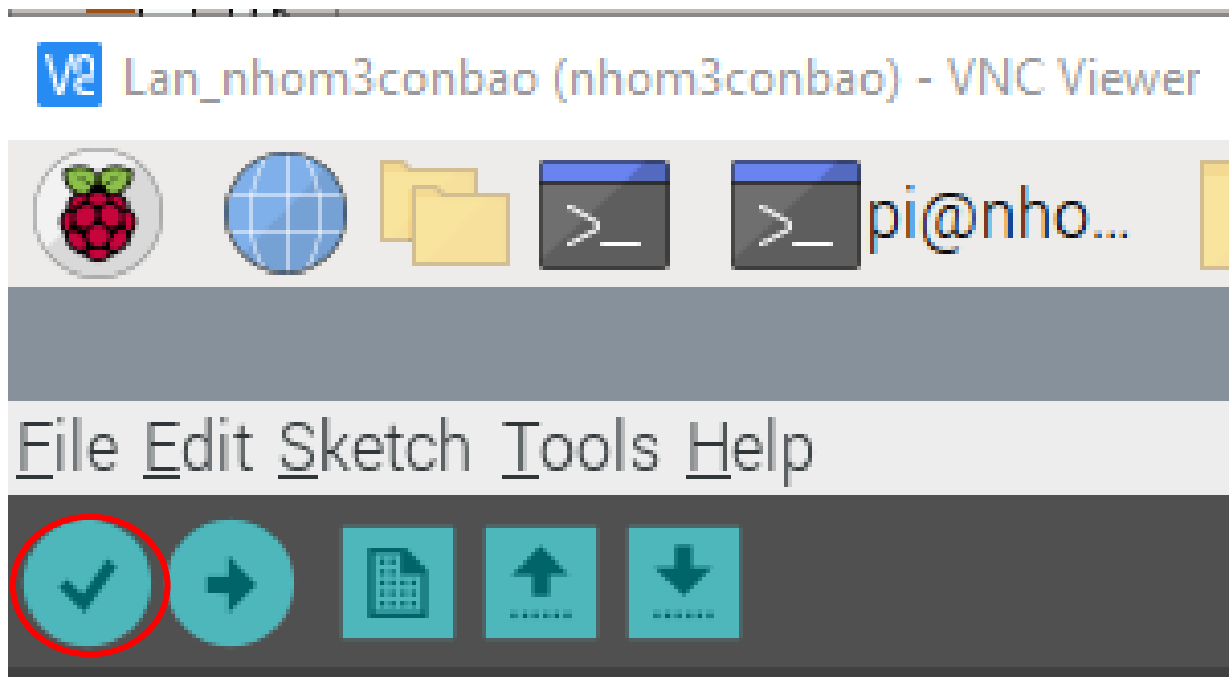
Hình 3 - 25: Hình biên dịch chương trình Arduino B5.1

- B52. Chọn Cổng giao tiếp giữa Rasp với Uno ở tool -> port



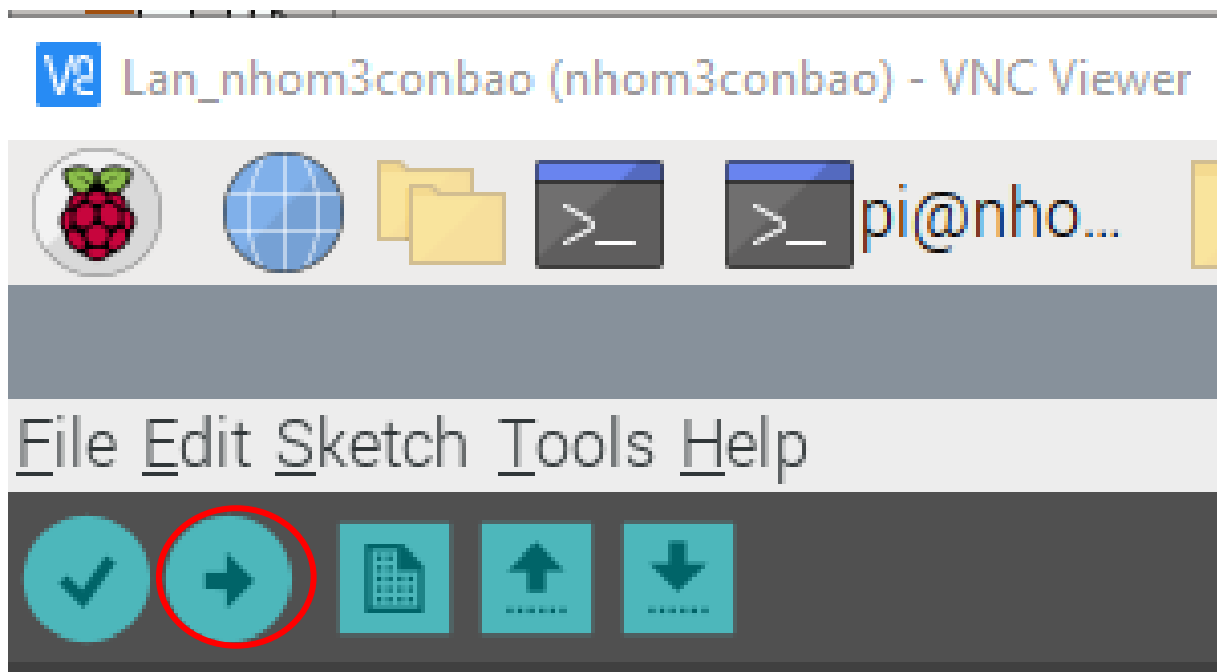
Hình 3 - 26: Hình biên dịch chương trình Arduino B5.2

- B53. Nhấn vào biểu tượng dấu tích để kiểm tra chương trình có đúng cú pháp hay không



Hình 3 - 27: Hình biên dịch chương trình Arduino B5.3

- B54. Nhấn vào biểu tượng mũi tên sang phải để nạp chương trình xuống Board Arduino Uno



Hình 3 - 28: Hình biên dịch chương trình Arduino B5.4

3.4.3 Lập trình cho App

B1. Cài nvm cho window để quản lí các phiên bản Node.js

- B11. Truy cập vào trang web sau: <https://github.com/coreybutler/nvm-windows>
- B12. Kéo xuống đọc phần README và chọn phần Download Now! Và vào phần Assets chọn download nvm-setup.exe để tải nvm về cho Window.

- B13. Khi đã tải file nvm-setup.exe về máy tính thì nhấn đúp vào file đã tải để cài đặt phần mềm

- B14. Khi đã nhấn đúp chuột để cài phần mềm thì chọn I accept the agreement để tiếp tục, tiếp theo chọn nơi để lưu cho file của bạn (khuyến khích lưu vào ổ C của máy để ít xảy ra lỗi), sau đó chọn nơi lưu Symlink thì để mặc định và nhấn Next cuối cùng chọn Install để thực hiện cài đặt.

B2. Cài phiên bản Node.js để thực hiện viết chương trình

- B21. Vào terminal của máy tính
- B22. Nhập lệnh `nvm install 20.11.0` là lệnh để tải môi trường Node.js phiên bản 20.11.0.

- B23. Khi lệnh chạy xong thì ta nhập lệnh `nvm use 20.11.0` để chọn phiên bản hiện tại ta sẽ sử dụng là 20.11.0

- B24. Để kiểm tra lại ta đã được chọn phiên bản hiện tại là 20.11.0 chưa thì ta nhập lệnh `nvm list` để kiểm tra lại phiên bản hiện tại của mình đã được trở vào 20.11.0 hay chưa, thành công thì con trỏ sẽ trở vào 20.11.0.

B3. Tạo dự án để viết app android và chạy chương trình trên môi trường Node.js

- B31. Ta vào Terminal để thực hiện tạo dự án
- B32. Ta dùng lệnh `cd path` với path là đường dẫn trở vào nơi bạn muốn lưu dự án
- B33. Sau đó ta nhập lệnh `npx create-expo-app android-app` để thực hiện tạo dự án sử dụng react-native thuộc môi trường Node.js
- B34. Khi tạo dự án thành công thì ta nhập lệnh `cd android-app` để vào dự án, nếu bạn code trên visual studio code thì ta thêm lệnh `code .` để mở folder dự án đang làm việc lên visual studio code để làm việc.
- B35. Để chạy chương trình dự án để mô phỏng giao diện giúp việc tạo app được thích nghi hơn thì ta nhập lệnh `npm start` để chạy chương trình.
- B36. Sau khi chạy xong có cho ta 1 đường link với 1 mã QR để kết nối. nếu ta lấy điện thoại thực để mô phỏng giao diện thì trên thiết bị thực ta cài phần expo go trên android và ios đều có.
- B37. Sau khi cài phần mềm xong:
 - + Đối với điện thoại android thì ta vào điện thoại chọn quét QR hoặc nhập link thủ công đều được
 - + Đối với điện thoại ios thì ta vào máy ảnh của máy quét mã QR thì sẽ được tự động chuyển qua phần mềm expo go với dự án hiện tại được gửi lên.

B4. Viết chương trình cá nhân hóa vào dự án android của chúng ta

<https://github.com/VM-Thuan->

[2003/TT_HTN_SPK/tree/main/PROJECT_CK/CODE/APP/app-android](https://github.com/VM-Thuan-2003/TT_HTN_SPK/tree/main/PROJECT_CK/CODE/APP/app-android)

B5. Biên dịch chương trình tạo file android-app.apk

- B51. Tạo file eas.json và nội dung trong file eas.json



```

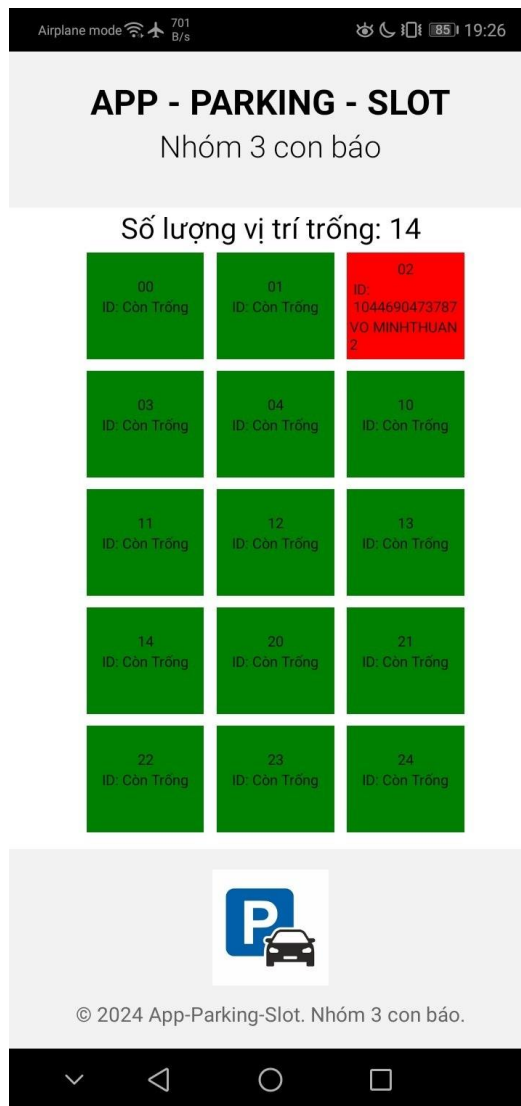
1  {
2    "build": {
3      "preview": {
4        "android": {
5          "buildType": "apk"
6        }
7      },
8      "preview2": {
9        "android": {
10         "gradleCommand": ":app:assembleRelease"
11       }
12     },
13     "preview3": {
14       "developmentClient": true
15     },
16     "production": {}
17   }
18 }

```

Hình 3 - 29: Hình biên dịch App B5.1

- B52. Mở terminal trong dự án hiện tại và nhập lệnh `npm install -g eas-cli` để sử dụng dịch vụ EAS cho terminal của bạn.
 - B53. Sau đó nhập lệnh `eas login` để đăng nhập vào eas (đăng ký tài khoản trên expo.dev)
 - B54. Sau đó nhập lệnh `eas build -p android` để tạo 1 profile lên Android Play Store build của expo
- B55. Sau đó nhập lệnh `eas build -p android --profile preview` để tạo file apk.

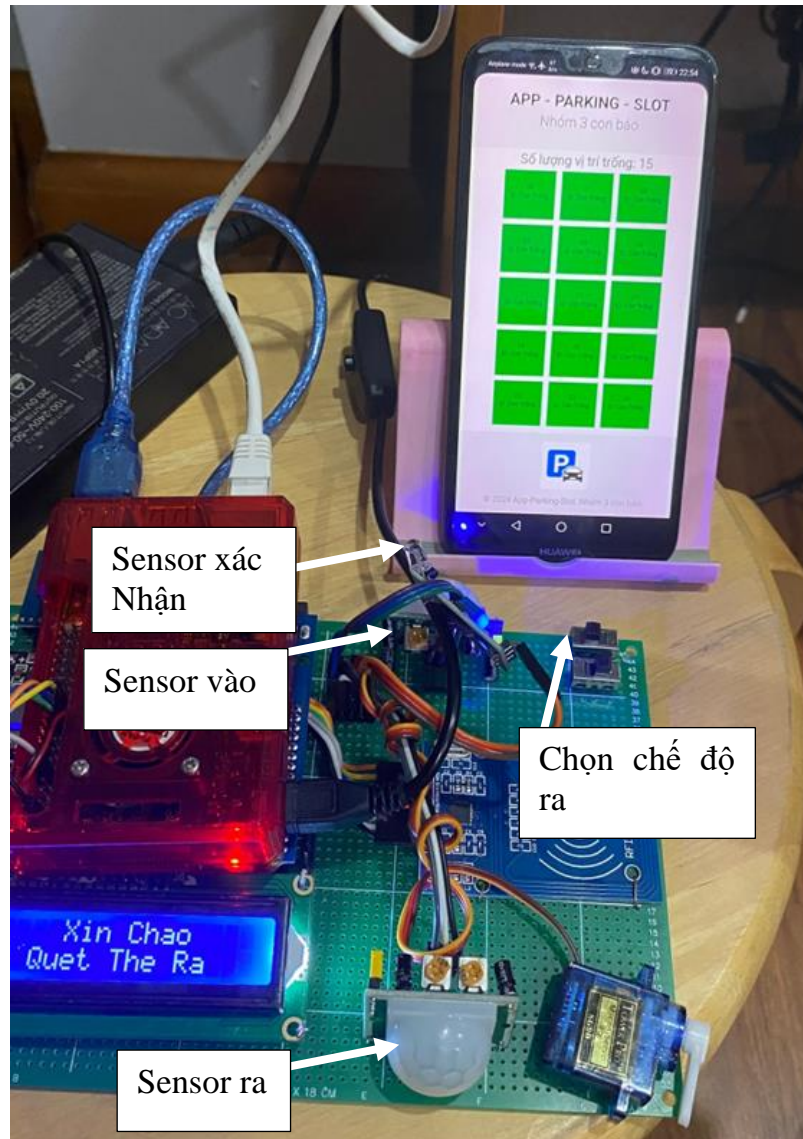
* Giao diện App



Hình 3 - 30: Hình giao diện App

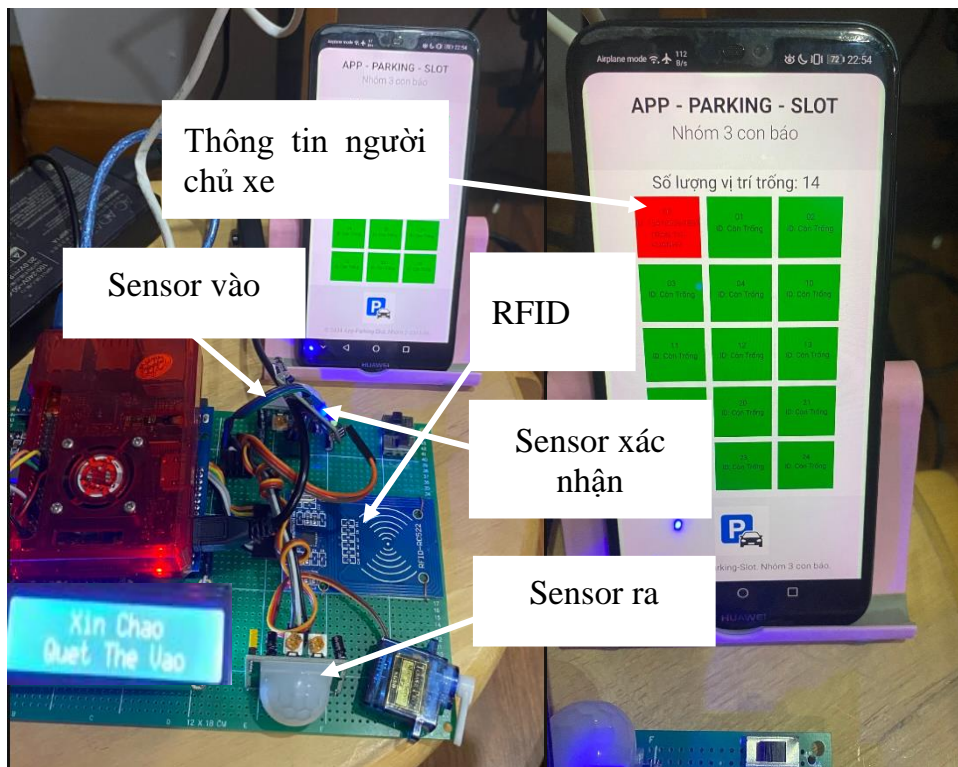
Sau khi thẻ RFID được xác nhận đúng thì dữ liệu sẽ được đưa lên firebase và hiển thị lên app, trên đầu giao diện sẽ cho người dung biết hiện tại còn bao nhiêu vị trí trống, ở vị trí nào đã có xe vào thì ô hiển thị đại diện cho vị trí đó sẽ đổi thành màu đỏ và hiển thị tên của chủ thẻ, ở vị trí nào còn trống thì sẽ hiển thị màu xanh.

Chạy thử hệ thống



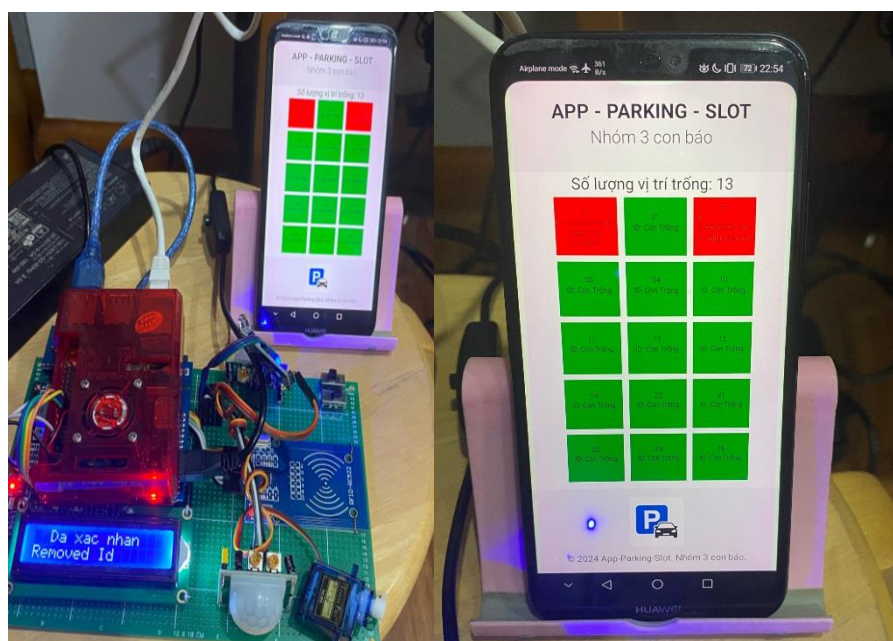
Hình 4 - 2: Hình chạy hệ thống ban đầu

Khi chưa có thẻ được quét, giao diện app hiển thị các vị trí còn trống là những ô vuông màu xanh, màn hình LCD hiển thị dòng chữ “Xin chào” và “Quẹt thẻ ra” nếu ở chế độ ra và “Quẹt thẻ vào” nếu hệ thống đang ở chế độ vào. Và hình trên thì công tắc chọn chế độ đang được gạt sáng trái là chế độ ra nên màn hình Lcd hiển thị là Quẹt thẻ ra.



Hình 4 - 3: Hình hệ thống khi có xe đi vào

Khi Sensor vào có giá trị lên 1 (true) là có xe đi vào và màn hình Lcd hiển thị Xin chào Quét Thẻ vào thì ta quét rfid nếu id chưa có trong bãi xe thì sẽ được thêm vào bãi xe và hiển thị thông tin trả về là trạng thái được thêm vào xe và thông tin người chủ xe sẽ được thêm vào firebase và hiển thị lên App và Servo được mở. Khi xe đi qua cảm biến xác nhận thì servo sẽ đóng lại.



Hình 4 - 4: Hình hệ thống khi có xe đi ra

Khi đã có thẻ RFID được xác nhận màn hình LCD sẽ hiển thị “đã xác nhận”, giao diện app lúc này sẽ hiển thị những vị trí nào còn trống sẽ là màu xanh và những vị trí nào đã có người sẽ là màu đỏ cùng với thông tin của chủ thẻ đậu ở vị trí đó. Ở đầu giao diện sẽ hiển thị cho người dung biết rằng trong bãi còn bao nhiêu vị trí gửi xe còn trống.

Thử nghiệm chạy thử sản phẩm để kiểm tra tính ổn định và chính xác:

- Khi hệ thống được điều chỉnh ở chế độ vào

Table 2: Bảng số liệu thử nghiệm khi hệ thống ở chế độ vào 5 lần

Lần	Thời gian nhận diện thẻ RFID	Thời gian servo xoay	Thời gian hiển thị lên app	Thời gian hiển thị lên LCD
1	0,4s	0,3s	7,8s	0,15s
2	0,32s	0,28s	7,9s	0,14s
3	0,37s	0,26s	7,6s	0,15s
4	0,35s	0,26s	7,9s	0,15s
5	0,4s	0,29s	7,7s	0,14s

Sau khi chạy thử 5 lần nhận thấy thời gian hệ thống nhận diện được thẻ RFID khá nhanh từ 0,32s -> 0,4s, cùng với đó thời gian mà servo đáp ứng cũng gần như là ngay lập tức. Thời gian để dữ liệu thông tin chủ thẻ hiển thị lên LCD khá nhanh tầm 0,15s, thời gian hiển thị lên app rơi vào khoảng từ 7,6s đến 7,9s.

- Khi hệ thống được điều chỉnh ở chế độ ra:

Table 3: Bảng số liệu thử nghiệm khi hệ thống ở chế độ ra 5 lần

Lần	Thời gian nhận diện thẻ RFID	Thời gian servo xoay	Thời gian hiển thị lên app	Thời gian hiển thị lên LCD
1	0,37s	0,26s	7s	0,15s
2	0,35s	0,26s	7,5s	0,14s
3	0,35s	0,25s	7,6s	0,15s
4	0,36s	0,26s	7,5s	0,14s
5	0,36s	0,26s	7,4s	0,15s

Sau khi chạy thử 5 lần nhận thấy thời gian hệ thống nhận diện được thẻ RFID khá nhanh từ 0,35s -> 0,37s, cùng với đó thời gian mà servo đáp ứng cũng gần như là ngay lập tức. Thời gian để dữ liệu thông tin chủ thẻ hiển thị lên LCD khá nhanh tầm 0,15s, thời gian hiển thị lên app rơi vào khoảng từ 7s đến 7,6s.

4.2 ĐÁNH GIÁ

Hệ thống đáp ứng được các yếu tố sau:

- Nhận diện được đúng thẻ RFID.
- Thời gian đáp ứng khá nhanh.
- Hiện thị được thông tin chủ thẻ.
- Có app hiển thị tình trạng của bãi xe.
- Khi đã dùng thẻ chế độ vào thì trường hợp làm rơi thẻ người khác cũng không thể sử dụng thẻ RFID để vào lại vì hệ thống sẽ không cho phép.
- Xác định được vị trí đậu của xe.

Tuy nhiên bên cạnh đó hệ thống cũng còn những hạn chế sau đây:

- Thời gian dữ liệu gửi để hiển thị lên app cho người dùng kiểm soát vẫn còn khá chậm (tầm 8s) bởi vì thời gian đọc dữ liệu cứ 5s 1 lần và còn tùy thuộc vào yếu tố mạng
- Sử dụng cảm biến chuyển động nên những chuyển động của môi trường cũng như yếu tố khác cũng làm cho hệ thống bị tác động.

4.3 HƯỚNG PHÁT TRIỂN

- Cải thiện thời gian đọc dữ liệu để xử lý gửi xe nhanh hơn.
- Tăng tính thẩm mỹ cho hệ thống.
- Tăng số lượng vị trí của bãi giữ xe.
- Tích hợp thêm khả năng tính tiền tự động.
- Tích hợp thêm camera để biết được khuôn mặt cũng như biển số xe để tăng tính bảo mật.
- Tích hợp diện rộng nhiều parking slot được kết nối lên điện thoại giúp đỗ xe tiện lợi hơn.

TÀI LIỆU THAM KHẢO

- [1] Nguyễn Đình Phú, “Giáo trình vi xử lý II”, NXB ĐH Quốc Gia Tp.HCM, 2007.
- [2] Aswinth Raj, “16x2 LCD Display Module” truy xuất từ: [16x2 LCD Display Module - Pinout & Datasheet \(circuitdigest.com\)](#).
- [3] Scott Campbell, “BASICS OF THE I2C COMMUNICATION PROTOCOL”, truy xuất từ: [Basics of the I2C Communication Protocol \(circuitbasics.com\)](#).
- [4] Nguyễn Văn Hiệp, “Công nghệ nhận dạng vô tuyến RFID”, Đại học Sư Phạm Kỹ Thuật Tp.HCM.