FPT-JETKING

HIGHER DIPLOMA IN CHIP DESIGN

BÃI GIỮ XE TỰ ĐỘNG ỨNG DỤNG

Giáo viên hướng dẫn: PHAN VĂN ĐỨC

Lớp: C1.2502.E0

Tên nhóm: Nhóm 1

Tên thành viên nhóm:	Tên đầy đủ	Mã số Sinh viên
	1. Dương Minh An	C1S2412009
	2. Nguyễn Thanh Hào	C1S2412005
	3. Huỳnh Phương Nam	C1S2412003

Tp.Hồ Chí Minh, ngày 26 tháng 05 năm 2025

TÓM TẮT

Mô hình bãi giữ xe tự động ứng dụng được thiết kế và thi công dựa trên các kỹ thuật tự động hóa và yêu cầu thực tế. Hệ thống tích hợp vi điều khiển 8051, ESP32, và công nghệ xử lý hình ảnh để giám sát, quản lý xe vào/ra thông qua công nghệ nhận diện biển số (ANPR), thẻ RFID, và cảm biến hồng ngoại (IR). Các chức năng chính bao gồm: nhận diện biển số xe qua API PlateRecognizer, xác thực thẻ RFID, điều khiển cổng tự động bằng servo motor, hiển thị số lượng xe trên LCD, và giao tiếp UART giữa các thành phần. Hệ thống tận dụng nền tảng IoT (Blynk, tùy chọn) để giám sát từ xa, tối ưu hóa nhân công, giảm thời gian tìm chỗ và nâng cao trải nghiệm người dùng.

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Tên Tiếng Anh	Diễn giải
AI	Artificial Intelligence	là trí tuệ nhân tạo, một lĩnh vực khoa học máy tính tập trung vào việc phát triển các hệ thống máy tính
ANPR	Automatic Number Plate Recognition	Là công nghệ nhận diện và đọc biển số xe tự động thông qua hình ảnh hoặc video.
API	Application Programming Interface	Là một giao diện cho phép các phần mềm hoặc hệ thống khác nhau giao tiếp với nhau.
COM	Communication Port	Là một cổng thông dụng trong các máy tính dùng kết nối các thiết bị ngoại vi
IDE	Integrated Development Environment	Là một ứng dụng phần mềm hỗ trợ lập trình viên phát triển mã phần mềm một cách hiệu quả
IoT	Internet of Things	Là mạng lưới các thiết bị thông minh có khả nắng kết nối và trao đổi dữ liệu với nhau qua internet.
LCD	Liquid Crystal Display	Là viết tắt của màn hình tinh thể lỏng, một loại công nghệ hiển thị phổ biến được sử dụng trong nhiều thiết bị điện tử như tivi, máy tính, điện thoại thông minh và máy tính bảng
PWM	Pulse Width Modulation	Là một phương pháp điều chế tín hiệu được sử dụng để điều khiển công suất hoặc điện áp bằng cách thay đổi độ rộng của các xung vuông
RFID	Radio Frequency Identification	Là một mã định danh duy nhất dùng để xác định tài nguyên nghiên cứu,
UART	Universal Asynchronous Receiver-Transmitter	Là một giao thức truyền thông nối tiếp giúp trao đổi dữ liệu giữa các thiết bị điện tử dễ dàng hơn.

PHẦN 1: GIỚI THIỆU CHỦ ĐỀ

1.1 Giới thiệu tổng quan:

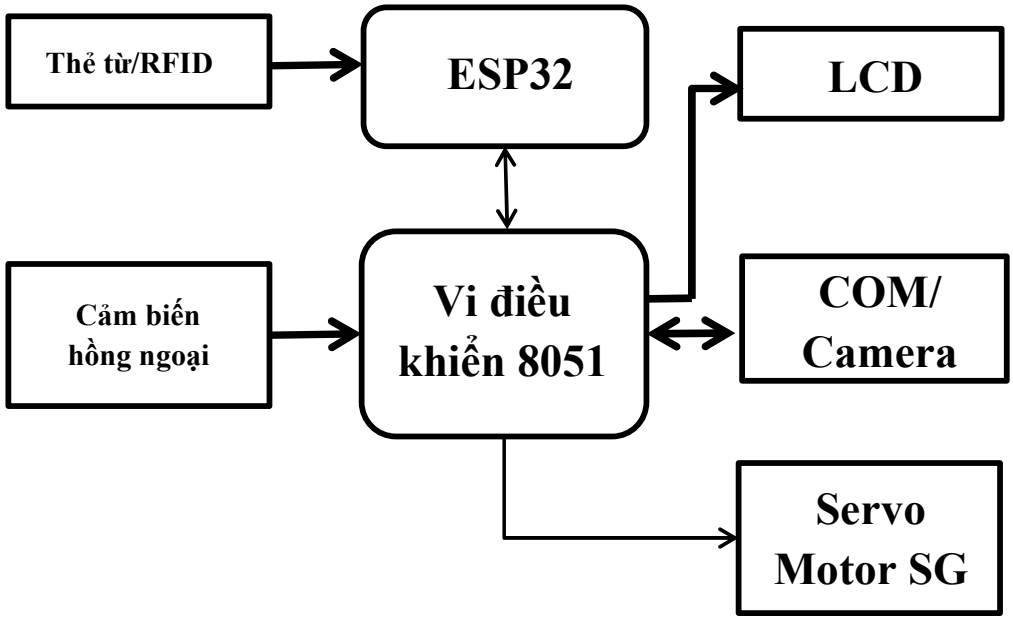
Trong bối cảnh đô thị hóa nhanh chóng tại các thành phố lớn như Hà Nội và TP. Hồ Chí Minh, nhu cầu tìm kiếm chỗ đỗ xe ngày càng tăng, gây ra ùn tắc giao thông và mất thời gian. Hệ thống bãi đỗ xe tự ứng dụng được thiết kế để giải quyết vấn đề này, tích hợp các công nghệ hiện đại như nhận diện biển số (ANPR), thẻ RFID, và IoT. Mô hình thử nghiệm sử dụng vi điều khiển 8051 để điều khiển cổng và hiển thị trạng thái, ESP32 để xác thực thẻ RFID, và phần mềm Python để nhận diện biển số qua webcam, kết hợp với API PlateRecognizer. Hệ thống có tiềm năng mở rộng để tích hợp với ứng dụng di động và nền tảng IoT như Blynk, cho phép giám sát và điều khiển từ xa.

1.2 Mục tiêu hệ thống:

- Tối ưu hóa không gian: Sử dụng cảm biến và điều khiển tự động để quản lý số lượng xe giảm diện tích cần thiết so với bãi đỗ truyền thống.
- Giảm thời gian tìm chỗ đỗ: Nhận diện biển số và thẻ RFID giúp xe vào/ra nhanh chóng, giảm ùn tắc.
- Tăng trải nghiệm người dùng: Hiển thị số lượng xe và trạng thái cổng trên LCD, đồng thời hỗ trợ giám sát từ xa qua Blynk (tùy chọn).
- Tiết kiệm năng lượng: Hoạt động bằng điện năng, không tạo khí thải, góp phần bảo vệ môi trường.
- An ninh và bảo mật: Camera và RFID đảm bảo xe được xác thực, giảm nguy cơ mất cắp hoặc phá hoại.
- Chi phí vận hành thấp: Hệ thống tự động hóa giảm nhân lực giám sát, dễ bảo trì nhờ thiết kế module.

1.3 Cấu trúc tổng quan hệ thống:

1.3.1 Sơ đồ hệ thống:



Hình 1.1 Sơ đồ hệ thống

1.3.2 Thành phần hệ thống:

- Phần mềm nhận diện (Python):
  - Chạy trên máy tính, sử dụng webcam để chụp ảnh biển số xe.
  - Gửi ảnh đến API PlateRecognizer, so sánh với danh sách biển số được phép.
  - Giao tiếp với 8051 qua UART (COM4) để gửi tín hiệu điều khiển cổng ra.
- Hạ tầng phần cứng (8051 và ESP32):
  - 8051: Điều khiển servo motor (cổng vào/ra), hiển thị số xe và trạng thái trên LCD, nhận tín hiệu từ ESP32 (RFID) và máy tính (UART).
  - ESP32: Quản lý hai module RFID MFRC522 để xác thực thẻ vào/ra, gửi tín hiệu đến 8051.
  - Cảm biến IR: Phát hiện xe tại cổng vào/ra.
  - Servo motor: Mở/đóng cổng vào/ra.

- LCD 16x2: Hiển thị số xe, trạng thái cổng, và thông báo lỗi.

- Hệ thống IoT:
  - Sử dụng Blynk để giám sát số lượng xe và trạng thái cổng qua Wi-Fi (ESP32).
  - Tích hợp ứng dụng di động để đặt chỗ và thanh toán.

1.3.3. Chức năng hoạt động:

- Nhận diện biển số (ANPR):
  - Webcam chụp ảnh biển số, gửi đến API PlateRecognizer.
  - Nếu biển số hợp lệ, gửi tín hiệu 1 qua UART đến 8051 để mở cổng ra.
- Xác thực thẻ RFID:
  - ESP32 kiểm tra UID thẻ qua module MFRC522.
  - Nếu thẻ hợp lệ, gửi tín hiệu đến 8051 để mở cổng vào.
- Điều khiển cổng tự động:
  - Servo motor (P1.5, P1.7) mở/đóng cổng dựa trên tín hiệu từ cảm biến, RFID, và UART.
  - LCD hiển thị trạng thái ("OPENING...", "WRONG ID", "FULL CAR").
- Quản lý số lượng xe:
  - Biến count được cập nhật khi xe vào/ra.
  - Hiển thị trên LCD và gửi đến Blynk.
- Giám sát từ xa (Blynk):
  - ESP32 gửi dữ liệu về số xe và trạng thái cổng đến Blynk Cloud.
  - Người dùng theo dõi qua ứng dụng Blynk (kế hoạch tương lai).

1.3.4 Ý nghĩa khoa học và thực tiễn:

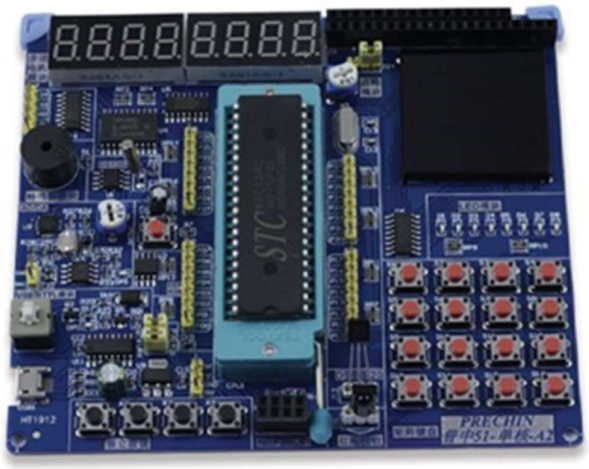
Khoa học: Tích lũy kinh nghiệm về IoT, giao tiếp UART, và tích hợp phần cứng/phần mềm (8051, ESP32, Python). Đặt nền tảng cho các nghiên cứu về hệ thống đỗ xe tự động.

Thực tiễn: Giảm thời gian tìm chỗ đỗ, tối ưu không gian, và nâng cao an ninh. Hệ thống có tiềm năng ứng dụng tại các trung tâm thương mại, tòa nhà, hoặc khu đô thị.

PHẦN 2: GIỚI THIỆU THIẾT BỊ - PHẦN MỀM

2.1 Thành phần phần cứng:

2.1.1 Kit vi điều khiển 8051:

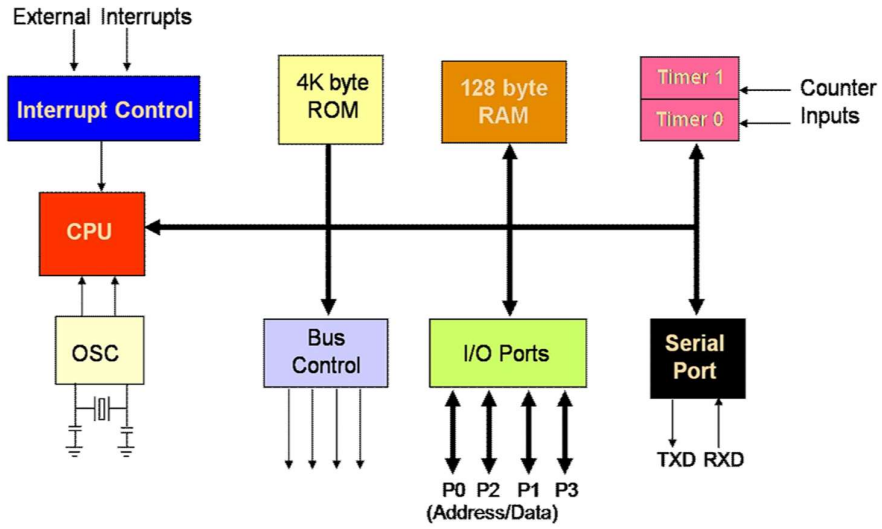


Hình 2.1: Kit vi điều khiển 8051 (Kit 8051 Pro)

- ***Các Chức Năng Chính:***
  - 8 Led 7 thanh 0.56 – Quét LED theo phương pháp sử dụng 74HC138
  - Điều khiển động cơ bước và động cơ DC sử dụng IC ULN2003 để tải công suất.
  - Còi BUZZ không có IC giúp bạn có thể tạo ta tiếng kêu tùy ý. như phát một bản nhạc chẳng hạn.
  - 8 LED đơn được kết nối trực tiếp với IO của CHIP.
  - 3 kênh ADC (1 cảm biến ánh sáng, 1 cảm biến nhiệt độ, 1 biến trở xoay trực tiếp) và 1 kênh DAC.
  - Điều khiển, giao tiếp với thạnh ghi dịch 74HC595, giúp bạn mở rộng IO cho CHIP.
  - Điều khiển RELAY 5V.
  - Giao tiếp với IC thời gian thực DS1302 để làm đồng hồ thời gian thực, lịch vạn niên …

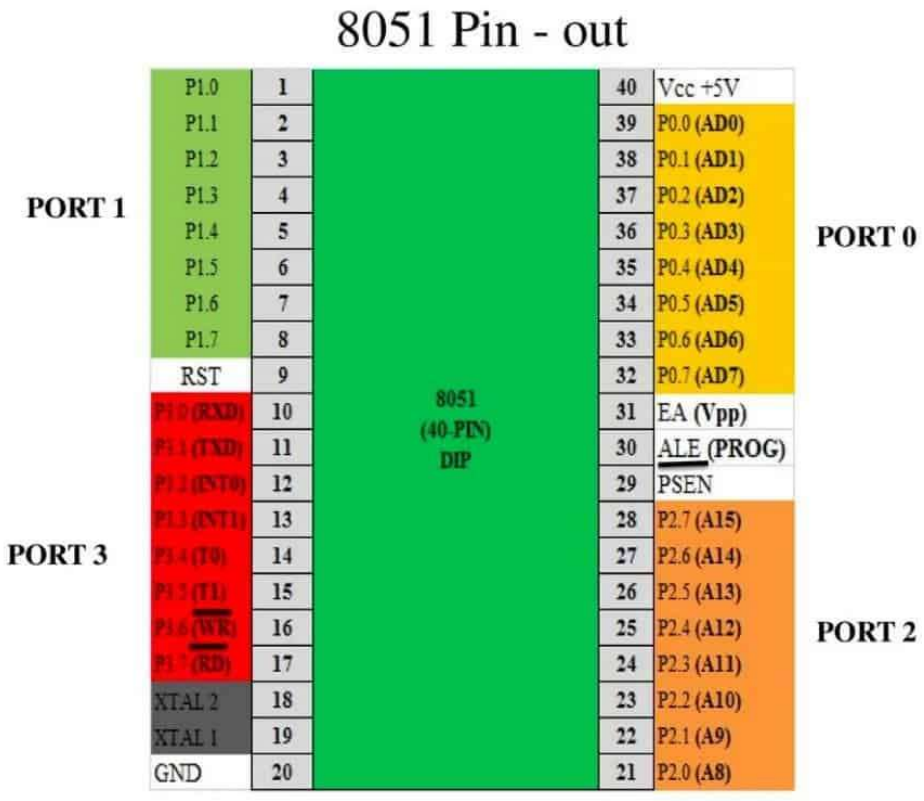
- Cổng USB để cấp nguồn, và giao tiếp USB -> COM khi bạn giao tiếp truyền thông nối tiếp RS232.
- 4 phím bấm đơn, được nối với ngắt ngoài. giúp bạn học ngắt ngoài hiệu quả.
- Giao tiếp với IC cảm biến nhiệt độ DS18B20.
- CHIP họ 8051. Có thể thay thế hoàn toàn linh động với kẹp ZIF40P.
- Chân ISP dùng để nạp chip qua chuẩn ISP. không cần nhổ CHIP.
- Giao tiếp với mắt nhận hồng ngoại. làm việc với điều khiển từ xa.
- Giao tiếp, hiển thị lên màn hinh GLCD 128×64.
- Làm việc với Ma trận phím 4×4.
- Các IO được nối ra ngoài giúp bạn giao tiếp, gỡ lỗi hiệu quả.
- Màn hình màu TFT 220×176 pixel hiển thị nhiều hơn, đẹp hơn (mua thêm).
- Giao tiếp với IC EEPROM 24C02 (IC nằm dưới màn hình).
- Giao tiếp với IC giải mã 3->8 (74LS138).
- Làm việc với IC 74HC573.
- Giao tiếp và hiển thị trên màn hình LCD 1602.

2.1.2 Vi điều khiển 8051:



Hình 2.2: Cấu tạo vi điều khiển 8051

- **Kiến trúc cơ bản bên trong 8051 bao gồm các khối chức năng sau:**
  - CPU (Central Processing Unit): đơn vị điều khiển trung tâm.
  - Bộ nhớ chương trình ROM bao gồm 4 Kbyte.
  - Bộ nhớ dữ liệu RAM bao gồm 128 byte.
  - Bốn cổng xuất nhập.
  - Hai bộ định thời/bộ đếm 16bit thực hiện chức năng định thời và đếm sự kiện.
  - Bộ giao diện nối tiếp (cổng nối tiếp).
  - Khối điều khiển ngắt với hai nguồn ngắt ngoài.
  - Bộ chia tần số.
- **Sơ đồ chân và chức năng các chân của vi điều khiển 8051:**



Hình 2.3: Sơ đồ chân vi điều khiển 8051

- ***Chân 1 đến 8***: được gọi là Cổng 1 (Port 1): Tám chân này có

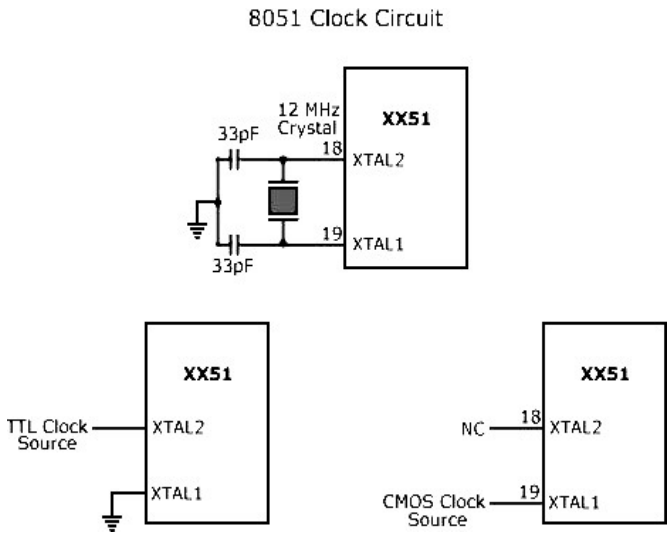
duy nhất 1 chức năng là xuất và nhập. Cổng 1 có thể xuất và nhập theo bit hoặc byte. Ta đánh tên cho mỗi chân của Port 1 là P1.X (X = 0 đến 7).

- ***Chân 9:*** là chân vào reset của 8051.
- ***Chân 10 đến 17***: được gọi là Cổng 3 (Port 3): Tám chân này ngoài chức năng là xuất và nhập như các chân ở cổng 1 (chân 1 đến 8) thì mỗi chân này còn có chức năng riêng nữa, cụ thể như sau:

Bit	Tên	Chức năng
P3.0	RxD	Chân nhận dữ liệu cho cổng nối tiếp
P3.1	TxD	Chân truyền dữ liệu cho cổng nối tiếp
P3.2	INT0	Chân ngắt bên ngoài 0
P3.3	INT1	Chân ngắt bên ngoài 1
P3.4	T0	Ngõ vào của Timer/counter 0
P3.5	T1	Ngõ vào của Timer/counter 1
P3.6	WR	Xung ghi bộ nhớ dữ liệu ngoài
P3.7	RD	Xung đọc bộ nhớ dữ liệu ngoài

Bảng 2.1: Sơ đồ chân Port3

- ***Chân 18 và 19*** (XTAL1 & XTAL2): Hai chân này được sử dụng để nối với bộ dao động ngoài. Thông thường một bộ dao động thạch anh sẽ được nối tới các chân đầu vào XTAL1 (chân 19) và XTAL2 (chân 18) cùng với hai tụ gốm giá trị khoảng 30pF. Một phía của tụ điện được nối xuống đất. Các hệ thống xây dựng trên 8051 thường có tần số thạch anh từ 10 đến 40 MHz, thông thường ta dùng thạch anh 12 Mhz



Hình 2.4: Mạch dao động cấp cho 8051

- ***Chân 20***: được nối vào chân 0V của nguồn cấp.
- ***Chân 21 đến chân 28***: được gọi là cổng 2 (Port 2)

  Tám chân của cổng 2 có 2 công dụng, ngoài chức năng là cổng xuất và nhập như cổng 1, thì cổng 2 này còn là byte cao của bus địa chỉ khi sử dụng bộ nhớ ngoài.
- ***Chân 29*** (PSEN): Chân PSEN là chân điều khiển đọc chương trình ở bộ nhớ ngoài, nó được nối với chân OE của ROM ngoài để cho phép đọc các byte mã lệnh trên ROM ngoài. PSEN ở mức thấp trong thời gian đọc mã lệnh. Khi thực hiện chương trình trong ROM nội thì PSEN được duy trì ở mức cao.
- ***Chân 30*** (ALE): Chân ALE cho phép tách các đường dữ liệu và các đường địa chỉ tại Port 0 và Port 2.
- ***Chân 31*** (EA): Tín hiệu chân EA cho phép chọn bộ nhớ chương trình là bộ nhớ trong hay ngoài vi điều khiển. Nếu chân EA được nối ở mức cao (nối nguồn Vcc), thì vi điều khiển thi hành chương trình trong ROM nội. Nếu chân EA ở mức thấp (được

nối GND) thì vi điều khiển thi hành chương trình từ bộ nhớ ngoài.

- ***Chân 32 đến 39***: Được gọi là cổng 0 (Port 0). Cổng 0 gồm 8 chân cũng có 2 công dụng, ngoài chức năng xuất nhập, cổng 0 còn là bus đa hợp dữ liệu và địa chỉ, chức năng này sẽ được sử dụng khi 8051 giao tiếp với các biết bị ngoài có kiến trúc Bus như các vi mạch nhớ...Vì cổng P0 là một máng mở khác so với các cổng P1, P2 và P3 nên các chân ở cổng 0 phải được nối với điện trở kéo khi sử dụng các chân này như chân vào/ra. Điện trở này tùy thuộc vào đặc tính ngõ vào của thành phần ghép nối với chân của port 0. Thường ta dùng điện trở kéo khoảng 4K7 đến 10K.
- ***Chân 40***: Chân nguồn của vi điều khiển, được nối vào chân Vcc của nguồn.

- **Bộ nhớ chương trình:** Bộ nhớ chương trình là bộ nhớ chỉ đọc, là nơi lưu trữ chương trình của vi điều khiển. Bộ nhớ chương trình của họ 8051 có thể thuộc một trong các loại sau ROM, EPROM, FLASH hoặc không có bộ nhớ chương trình trên chip. Với họ vi điều khiển 89xx, bộ nhớ chương trình được tích hợp sẵn trong chip có kích thước nhỏ nhất là 4kByte. Với các vi điều khiển không tích hợp sẵn bộ nhớ chương trình trên chip, buộc phải thiết kế bộ nhớ chương trình bên ngoài. Địa chỉ đầu tiên của bộ nhớ chương trình là 0000H, chính là địa chỉ reset của vi điều khiển. Ngay khi bật nguồn hoặc reset vi điều khiển, thì CPU sẽ nhảy đến thực hiện lệnh ở địa chỉ 0000H này. Khi sử dụng bộ nhớ trên chip thì chân EA phải được nối lên mức logic cao (+5V). Nếu bạn muốn mở rộng bộ nhớ chương trình thì chúng ta phải dùng bộ nhớ ngoài với dung lượng tối đa là 64Kbyte.

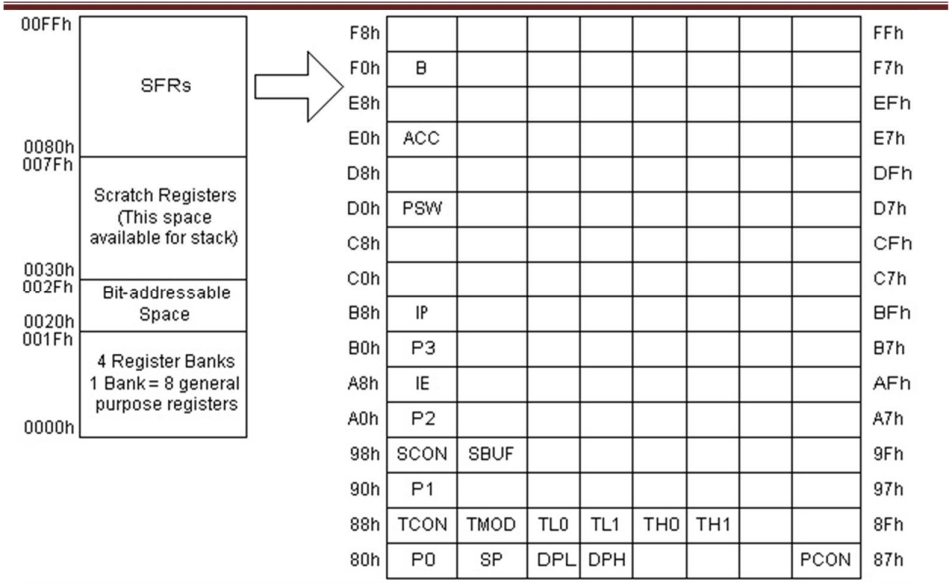
- **Bộ nhớ dữ liệu:** Bộ nhớ dữ liệu tồn tại độc lập so với bộ nhớ chương trình. Họ vi điều khiển 8051 có bộ nhớ dữ liệu tích hợp trên chip nhỏ nhất là 128byte và có thể mở rộng với bộ nhớ dữ liệu ngoài lên tới 64kByte. Bộ nhớ dữ liệu được phân chia như sau:

Các băng thanh ghi có địa chỉ từ 00H đến 1FH: 32 byte thấp của bộ nhớ nội được dùng cho các băng thanh ghi (dãy thanh ghi). Bộ lệnh 8051 hỗ trợ 8 thanh ghi R0 đến R7 và theo mặc định sau khi reset hệ thống, các thanh ghi này có các địa chỉ từ 00H đến 07H. Do có 4 băng thanh ghi nên tại một thời điểm chỉ có duy nhất 1 băng thanh ghi được truy suất bởi các thanh ghi R0 - R7, để thay đổi các băng thanh ghi thì ta thay đổi các bit chọn băng trong thanh ghi trạng thái PSW.

Vùng RAM địa chỉ hóa từng bit có địa chỉ từ 20h đến 2Fh: 8051 chứa 210 vị trí bit được định địa chỉ trong đó 128 bit chứa trong các byte ở địa chỉ từ 20H đến 2FH (16 byte x 8 bit = 128 bit) và phần còn lại chứa trong các thanh ghi đặc biệt. Ngoài ra 8051 còn có các cổng xuất/nhập có thể định địa chỉ từng bit, điều này làm đơn giản việc giao tiếp bằng phần mềm với các thiết bị xuất/nhập đơn bit.

Vùng RAM đa dụng có địa chỉ từ 30h đến 7Fh: Bất kỳ vị trí nhớ nào trong vùng RAM đa mục đích đều có thể được truy xuất tự do bằng cách sử dụng các kiểu định địa chỉ trực tiếp hoặc gián tiếp

Các thanh ghi chức năng đặc biệt có địa chỉ từ 80h đến FFh: Cũng như các thanh ghi từ R0 đến R7, ta có 21 thanh ghi chức năng đặc biệt SFR chiếm phần trên của Ram nội từ địa chỉ 80H đến FFH. Cần lưu ý là không phải tất cả 128 địa chỉ từ 80H đến FFH đều được định nghĩa mà chỉ có 21 địa chỉ được định nghĩa.



Hình 2.5: Cấu trúc bộ nhớ dữ liệu của 8051

Các thanh ghi chức năng đặc biệt (SFR): Thanh ghi của 8051 được dùng để lưu trữ tạm thời dữ liệu hoặc địa chỉ. Các thanh ghi này chủ yếu có kích thước 8 bit, 8 bit của các thanh ghi được sắp xếp như hình dưới trong đó bit D7 là bit có trọng số cao nhất, còn bit D0 là bit có trọng số thấp nhất.

D7	D6	D5	D4	D3	D2	D1	D0

Bảng 2.2: Thanh ghi 8bit

Phần dưới đây, chúng ta sẽ nghiên cứu khái quát về các thanh ghi cơ bản của 8051

- **Thanh ghi chính A:** Là thanh ghi đặc biệt của 8051 dùng để thực hiện các phép toán của CPU, thường kí hiệu là A (Accumulator).
- **Thanh ghi phụ B:** Là thanh ghi tính toán phụ của vi điều khiển 8051, ở địa chỉ F0H được dùng chung với thanh ghi chính A trong các phép toán nhân, chia. Thanh ghi B cũng được sử dụng như một thanh ghi trung gian
- **Thanh ghi cổng P0 - P3:** Các port xuất/nhập của 8051 bao gồm

Port 0 tại địa chỉ 80H, Port 1 tại địa chỉ 90H, Port 2 tại địa chỉ A0H và Port 3 tại địa chỉ B0H. Tất cả các port này đều có thể truy suất theo bit hoặc theo byte.

- **Thanh ghi trạng thái chương trình PSW:** Thanh ghi trạng thái chương trình PSW (địa chỉ: D0H) là thanh ghi mô tả toàn bộ trạng thái chương trình đang hoạt động của hệ thống.

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	AC	F0	RS1	RS0	OV	-	P

Bảng 2.3: Thanh ghi trạng thái chương trình PSW

- **Con trỏ ngăn xếp SP (Stack Point):** Ngăn xếp chính là vùng bộ nhớ RAM được CPU sử dụng để lưu thông tin tạm thời. Thông tin này có thể là dữ liệu hoặc địa chỉ. CPU cần các thanh ghi này vì số các thanh ghi bị hạn chế.

  Như vậy, để có thể truy cập vào vùng nhớ ngăn xếp thì cần phải có thanh ghi trong CPU trỏ đến. Thanh ghi SP này sẽ được dùng để trỏ đến ngăn xếp, nên được gọi là thanh ghi con trỏ ngăn xếp. Thanh ghi này có độ rộng là 8 bit, tức là chỉ có thể trỏ được các địa chỉ từ 00h đến FFh.
- **Con trỏ dữ liệu DPTR (Data pointer):** Con trỏ dữ liệu được dùng để truy xuất bộ nhớ chương trình ngoài hoặc bộ nhớ dữ liệu ngoài. Con trỏ dữ liệu là một thanh ghi 16bit ở địa chỉ 82H (DPL - byte thấp) và 83H (DPH byte cao).
- **Thanh ghi bộ đệm truyền thông nối tiếp SBUF (Serial Data Buffer):** Bộ đệm truyền thông được chia thành hai bộ đệm, bộ đệm truyền dữ liệu và bộ đệm nhận dữ liệu. Khi dữ liệu được chuyển vào thanh ghi SBUF, dữ liệu sẽ được chuyển vào bộ đệm truyền dữ

liệu và sẽ được lưu giữ ở đó cho đến khi quá trình truyền dữ liệu qua truyền thông nối tiếp kết thúc. Khi thực hiện việc chuyển dữ liệu từ SBUF ra ngoài, dữ liệu sẽ được lấy từ bộ đệm nhận dữ liệu của truyền thông nối tiếp.

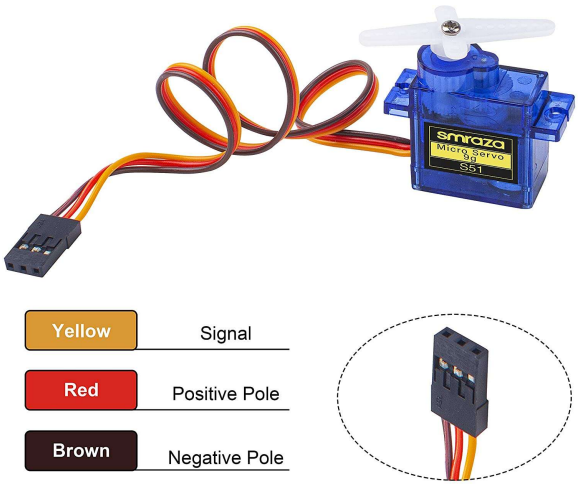
- **Thanh ghi của bộ định thời/bộ đếm:** 8051 có 2 bộ đếm/định thời (counter/timer) 16 bit để định các khoảng thời gian hoặc để đếm các sự kiện. Các cặp thanh ghi (TH0, TL0) và (TH1, TL1) là các thanh ghi của bộ đếm thời gian. Bộ định thời 0 có địa chỉ 8AH (TL0, byte thấp) và 8CH (TH0, byte cao). Bộ định thời 1 có địa chỉ 8BH (TL1, byte thấp) và 8DH (TH1, byte cao).

  Hoạt động của bộ định thời được thiết lập bởi thanh ghi chế độ định thời TMOD (Timer Mode Register) ở địa chỉ 88H. Chỉ có TCON được định địa chỉ từng bit.

- **Thanh ghi ngắt (Interrupt register):** Ngắt (Interrupt) - như tên của nó, là một số sự kiện khẩn cấp bên trong hoặc bên ngoài bộ vi điều khiển xảy ra, buộc vi điều khiển tạm dừng thực hiện chương trình hiện tại, phục vụ ngay lập tức nhiệm vụ mà ngắt yêu cầu – nhiệm vụ này gọi là trình phục vụ ngắt (ISR: Interrupt Service Routine).

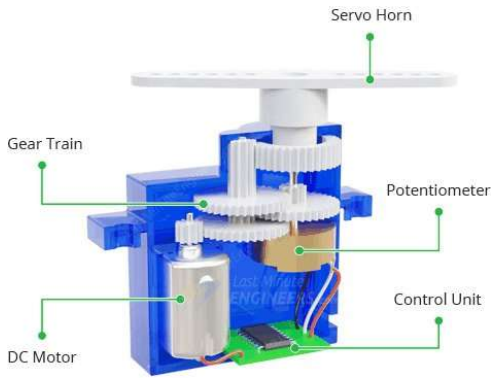
  Để thiết lập ngắt, ta sẽ thiết lập các giá trị trong thanh ghi cho phép ngắt IE ở địa chỉ A8H, thứ tự ưu tiên ngắt được đặt bằng cách set các bit ở thanh ghi ưu tiên ngắt IP ở địa chỉ B8h.

2.1.3 Servo motor:



Hình 2.6: Servo motor SG90

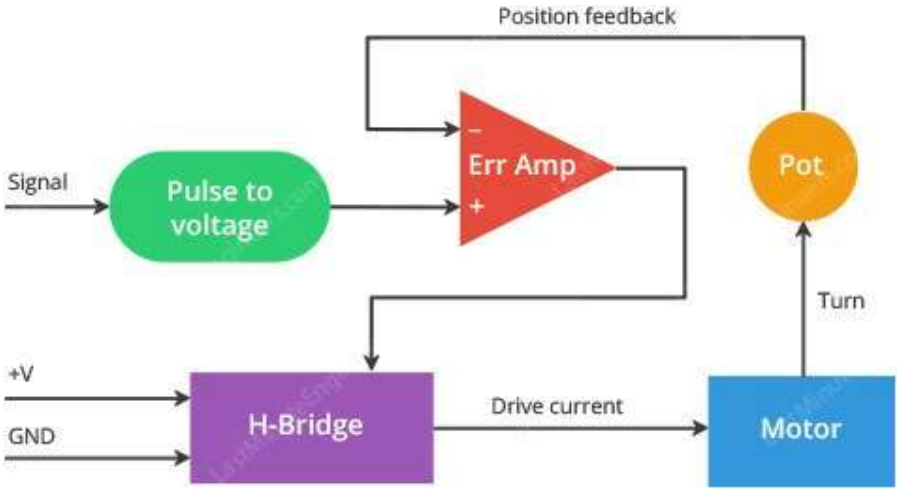
- ***Giới thiệu:*** Động cơ Servo SG90 là một loại động cơ nhỏ gọn, giá thành thấp, được sử dụng phổ biến trong lập trình Arduino. Nó thường xuất hiện trong các dự án như robot dò line, cánh tay robot, v.v.
- ***Cấu tạo động cơ Servo SG90:***



Hình 2.7: Cấu tạo bên trong động cơ Servo SG90

  - Động cơ DC: Tạo ra chuyển động quay của trục đầu ra.
  - Hệ thống điều khiển: Nhận tín hiệu từ Arduino, điều chỉnh vị trí động cơ.
  - Hệ thống giảm tốc: Tăng lực xoắn, giúp điều chỉnh vị trí chính xác hơn.

- ***Nguyên lý hoạt động:***



Hình 2.8: Nguyên lý hoạt động động cơ Servo SG90

- Nhận tín hiệu điều khiển PWM từ Arduino.
- Bộ điều khiển so sánh tín hiệu với vị trí hiện tại.
- Mạch phản hồi cung cấp thông tin vị trí động cơ.
- Điều chỉnh liên tục để duy trì độ chính xác.

- ***Sơ đồ chân động cơ Servo SG90 Arduino:***

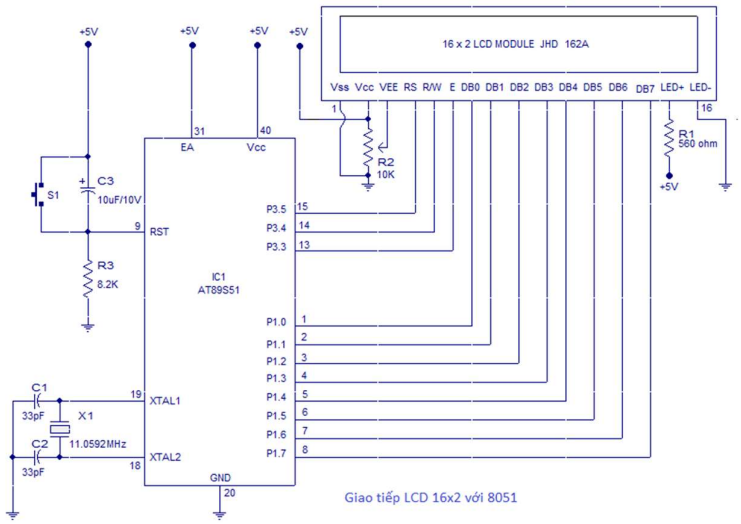
Pin	Màu	Mô tả
5V	Đỏ	Cấp nguồn
GND	Nâu	Nối đất
Control	Vàng	Nhận tín hiệu PWM điều khiển vị trí

Bảng 2.4: Sơ đồ chân động cơ Servo SG90

2.1.4 Thiết bị hiển thị LCD 16x2:

Hình 2.9: Thiết bị hiển thị LCD 16x2

- **Sơ đồ nguyên lý:** cách giao tiếp mô-đun LCD 16 × 2 với vi điều khiển AT89S1. Tụ điện C3, điện trở R3 và công tắc nút nhấn S1 tạo thành mạch reset. Tụ gồm C1, C2 và tinh thể X1 tạo ra tần số xung nhịp cho vi điều khiển. Các chân từ P1.0 đến P1.7 của vi điều khiển được kết nối với các chân DB0 đến DB7 của mô-đun tương ứng và qua đó, dữ liệu đi đến mô-đun LCD. P3.3, P3.4 và P3.5 được nối với các chân E, R / W, RS của vi điều khiển và tín hiệu điều khiển được truyền đến mô-đun LCD. Biến trở R2 được sử dụng để điều chỉnh độ tương phản của màn hình. Chương trình giao tiếp LCD với vi điều khiển 8051 được hiển thị bên dưới.



Hình 2.10: Giao tiếp LCD với 8051

- **Mô-đun LCD 16x2:** là loại mô-đun LCD rất phổ biến được sử dụng trong các dự án nhúng dựa trên 8051. Nó bao gồm 16 hàng và 2 cột 5 × 7 hoặc 5 × 8 ma trận điểm LCD. Các mô-đun đang nói về ở đây là loại JHD162A, một loại rất phổ biến. Nó có sẵn trong một gói 16 chân với ánh sáng nền , chức năng điều chỉnh độ tương phản và mỗi ma trận điểm có độ phân giải 5 × 8 chấm. Số chân, tên của chúng và các chức năng tương ứng được hiển thị trong bảng bên dưới:

Chân số	Tên chân	Chức năng
1	Vss	Chân này phải được nối GND
2	Vcc	Chân nối nguồn cấp (5V)
3	Vee	Chỉnh độ tương phản
4	RS	Chọn thanh ghi
5	R/W	Đọc hoặc ghi
6	E	Cho phép mô-đun
7	DB0	Chân dữ liệu
8	DB1	Chân dữ liệu
9	DB2	Chân dữ liệu
10	DB3	Chân dữ liệu
11	DB4	Chân dữ liệu
12	DB5	Chân dữ liệu
13	DB6	Chân dữ liệu
14	DB7	Chân dữ liệu
15	LED+	Anode của led
16	LED-	Cathode của led

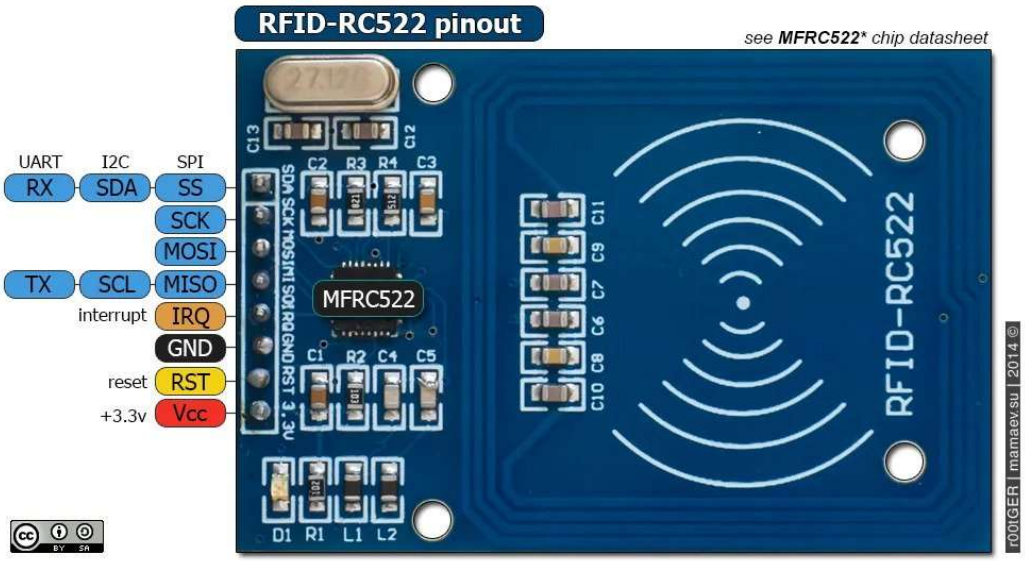
Bảng 2.5: Sơ đồ chân LCD

- **Các lệnh cho mô-đun LCD 16x2:** Mỗi lệnh sẽ làm cho mô-đun thực hiện một nhiệm vụ cụ thể. Các lệnh thường được sử dụng và chức năng của chúng được đưa ra trong bảng dưới đây:

Lệnh	Chức năng
0F	LCD bật, con trỏ bật, con trỏ nhấp nháy bật
01	Xoá toàn màn hình
02	Quay về màn hình chính
04	Giảm con trỏ
06	Tăng con trỏ
0E	Màn hình bật, con trỏ nhấp nháy tắt
80	Bắt con trỏ trở về vị trí đầu tiên của hàng 1
C0	Bắt con trỏ trở về vị trí đầu tiên của hàng 2
38	Sử dụng 2 hàng và ma trận 5x7
83	Con trỏ hàng 1 vị trí 3
3C	Kích hoạt dòng 2
08	Tắt màn hình hiển thị và con trỏ
C1	Nhảy đến dòng 2 vị trí 1
OC	Bật màn hình hiển thị, tắt con trỏ
C2	Nhảy đến hàng 2, vị trí 2

Bảng 2.6: Lệnh mô-đun LCD

2.1.5 Mạch đọc thẻ RFID:



Hình 2.11: RFID

- ***Các đặc tính RC522***
  - *RFID RC522 sử dụng cảm ứng điện từ để kích hoạt thẻ và có tần số 13,56MHz để truyền dữ liệu.*
  - *Thẻ RFID có thể sử dụng được với cả hai mặt của module khoảng cách tối đa 5cm.*
  - *Điện áp 3.3V được yêu cầu để hoạt động.*
  - *Chế độ ngủ tự động giúp module tiêu thụ ít điện năng hơn.*
  - *Module có ba loại giao tiếp (UART, SPI, I2C). Do đó, có thể sử dụng được với hầu hết mọi vi điều khiển.*
  - *Có thể truyền dữ liệu lên đến 10Mb/s.*

Chân	Mô tả chi tiết
VCC	Chân nguồn VCC. Trong một số phiên bản của RC522, chân này được ký hiệu là 3V3 thay vì VCC.
RST	Là chân reset được sử dụng để đặt lại giá trị trong trường hợp xảy ra lỗi khi thiết bị không bất kỳ phản hồi.

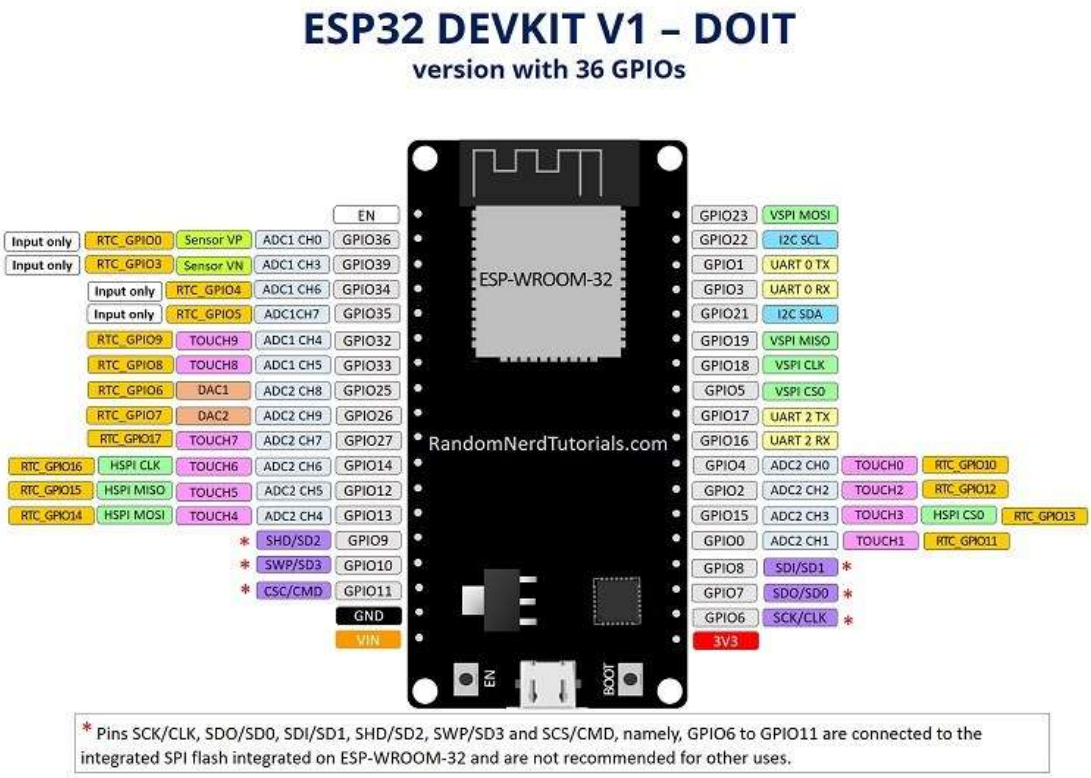
Chân	Mô tả chi tiết
Ground	Chân nối đất giúp tạo mass chung với các thiết bị bên ngoài (ví dụ bộ nguồn, bi điều khiển hoặc arduino).
IRQ	Linh kiện có thể chuyển sang chế độ ngủ để tiết kiệm điện năng và chân IRQ sẽ khởi động lại nó
MISO, SCL, TX	Chân này kết nối với Arduino / Vi điều khiển để giao tiếp SPI. Truyền dữ liệu từ module sang Arduino. MISO cũng có thể sử dụng cho các chức năng khác thay vì SPI. Cũng có thể giao tiếp I2C và UART Serial để giao tiếp dữ liệu với module.
MOSI	MOSI là chân đầu vào dữ liệu module RFID khi giao tiếp SPI
SCK	Các chân SCK gửi xung clock khi giao tiếp SPI
SS, SDA, RX	Chân SS là chân kích hoạt chip giao tiếp SPI. Nhận tín hiệu khi Master (Arduino) giao tiếp SPI. Chân SS của RFID có thể được sử dụng như một chân thứ hai (SDA) của giao tiếp I2C. Cũng là chân nhận dữ liệu trong quá trình giao tiếp UART.

Bảng 2.7: Chân RFID

- ***Ứng dụng module RFID RC522***
  - *RFID sử dụng như một thiết bị bảo mật.*
  - *Ở một số công ty, các thiết bị được sử dụng trong các mặt hàng mua sắm.*
  - *Một số sân bay cũng bắt đầu sử dụng RFID để xác định và kiểm tra túi xách và các vật dụng khác.*
  - *Hệ thống chấm công hoặc bãi đậu xe cũng sử dụng RFID.*
- ***Cách sử dụng RFID RC522:*** *Việc sử dụng RC 522 vừa đơn giản vừa phức tạp. Ngay cả thư viện hỗ trợ của nó cũng có quá nhiều phức tạp để sử dụng. Trước tiên, hãy hiểu rằng các ví dụ và thư*

viện hỗ trợ đều cho giao thức SPI nhưng cùng một thư viện có thể sử dụng được cho các giao tiếp UART và I2C Serial khác.

2.1.6 ESP32:



Hình 2.12: Vi điều khiển ESP32

- ***Vi điều khiển ESP32 có:***
  - **Chân Input Only:** GPIO từ 34 đến 39 là GPI – chân chỉ đầu vào. Các chân này không có điện trở kéo lên hoặc kéo xuống bên trong. Chúng không thể được sử dụng làm đầu ra, vì vậy chỉ sử dụng các chân này làm đầu vào:
    - GPIO 34
    - GPIO 35
    - GPIO 36
    - GPIO 39
  - **Chân tích hợp Flash trên ESP32:** GPIO 6 đến GPIO 11 dùng để kết nối Flash SPI, không khuyến khích sử dụng trong các ứng dụng khác

- GPIO 6 (SCK/CLK)
- GPIO 7 (SDO/SD0)
- GPIO 8 (SDI/SD1)
- GPIO 9 (SHD/SD2)
- GPIO 10 (SWP/SD3)
- GPIO 11 (CSC/CMD)

- **Chân cám biến điện dung:** Các chân ESP32 này có chức năng như 1 nút nhấn cảm ứng, có thể phát hiện sự thay đổi về điện áp cảm ứng trên chân. Các cảm biến cảm ứng bên trong đó được kết nối với các GPIO sau:
  - T0 (GPIO 4)
  - T1 (GPIO 0)
  - T2 (GPIO 2)
  - T3 (GPIO 15)
  - T4 (GPIO 13)
  - T5 (GPIO 12)
  - T6 (GPIO 14)
  - T7 (GPIO 27)
  - T8 (GPIO 33)
  - T9 (GPIO 32)
- **Analog to Digital Converter (ADC):** ESP32 có các kênh đầu vào ADC 18 x 12 bit (trong khi ESP8266 chỉ có ADC 1x 10 bit). Đây là các GPIO có thể được sử dụng làm ADC và các kênh tương ứng:
  - ADC1_CH0 (GPIO 36)
  - ADC1_CH1 (GPIO 37)
  - ADC1_CH2 (GPIO 38)
  - ADC1_CH3 (GPIO 39)
  - ADC1_CH4 (GPIO 32)
  - ADC1_CH5 (GPIO 33)

- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

Các kênh đầu vào ADC có độ phân giải 12 bit. Điều này có nghĩa là bạn có thể nhận được các số đọc tương tự từ 0 đến 4095, trong đó 0 tương ứng với 0V và 4095 đến 3,3V. Bạn cũng có thể lập trình độ phân giải của các kênh của mình trên code.

- **Digital to Analog Converter (DAC):** Có các kênh DAC 2 x 8 bit trên ESP32 để chuyển đổi tín hiệu kỹ thuật số thành đầu ra tín hiệu điện áp tương tự. Các kênh này chỉ có độ phân giải 8 bit, nghĩa là có giá trị từ 0 – 255 tương ứng với 0 – 3.3V, đây là các kênh DAC:
  - DAC1 (GPIO25)
  - DAC2 (GPIO26)
- **Các chân thời gian thực RTC:** Các chân này có tác dụng đánh thức ESP32 khi trong chế độ Low Power Mode. Sử dụng như 1 chân ngắt ngoài. Các chân RTC:
  - RTC_GPIO0 (GPIO36)
  - RTC_GPIO3 (GPIO39)
  - RTC_GPIO4 (GPIO34)
  - RTC_GPIO5 (GPIO35)

- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)
- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)
- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

o **Chân PWM:** ESP32 LED PWM có 16 kênh độc lập có thể được định cấu hình để tạo tín hiệu PWM với các thuộc tính khác nhau. Tất cả các chân có thể hoạt động như đầu ra đều có thể được sử dụng làm chân PWM (GPIO từ 34 đến 39 không thể tạo PWM). Để xuất PWM, bạn cần xác định các thông số này trong code:

- Frequency – tần số
- Duty cycle
- Kênh PWM
- Chân GPIO nơi bạn muốn xuất tín hiệu

o **Chân I2C:** ESP32 có hai kênh I2C và bất kỳ chân nào cũng có thể được đặt làm SDA hoặc SCL. Khi sử dụng ESP32 với Arduino IDE, các chân I2C mặc định là:

- GPIO 21 (SDA)
- GPIO 22 (SCL)

o **Chân Ngắt Ngoài:** Tất cả các chân ESP32 đều có thể sử dụng ngắt ngoài

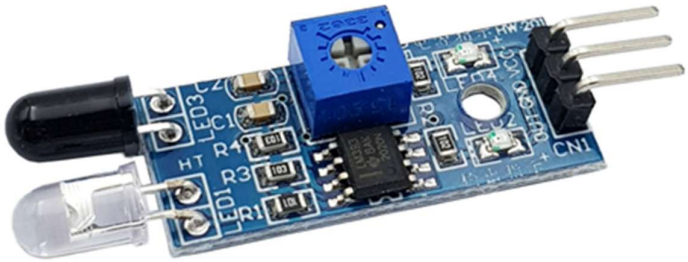
- **Dưới đây là bảng tra cứu khả năng sử dụng của các chân**:

GPIO Pin	Input	Output	Chức năng đặc biệt
0	pulled up	OK	outputs PWM signal at boot
1	TX pin	OK	debug output at boot
2	OK	OK	connected to on-board LED
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot
6	X	X	connected to the integrated SPI flash
7	X	X	connected to the integrated SPI flash
8	X	X	connected to the integrated SPI flash
9	X	X	connected to the integrated SPI flash
10	X	X	connected to the integrated SPI flash
11	X	X	connected to the integrated SPI flash
12	OK	OK	boot fail if pulled high
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only

GPIO Pin	Input	Output	Chức năng đặc biệt
39	OK		input only

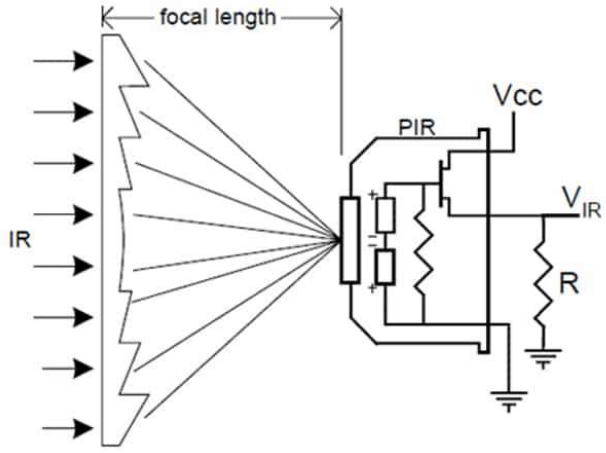
Bảng 2.8: Bảng tra cứu khả năng sử dụng của các chân ESP32

2.1.7 Cảm biến hồng ngoại (IR Sensor):



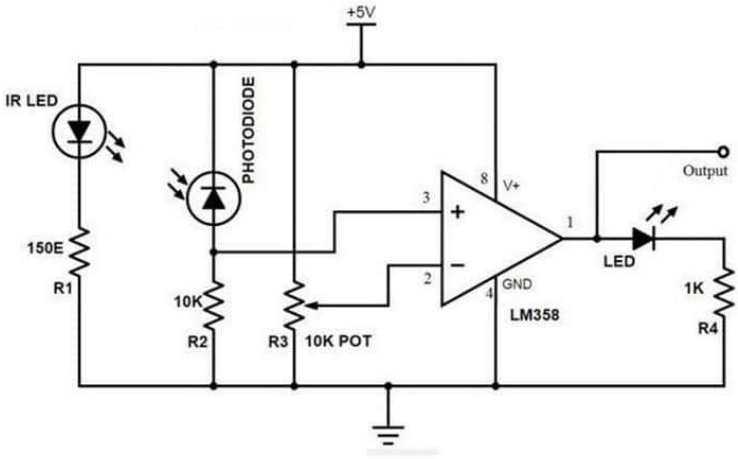
Hình 2.13: Cảm biến hồng ngoại (IR Sensor)

- **Cảm biến hồng ngoại hay còn gọi là IR sensor (Infrared sensor)**: Là một thiết bị điện tử có khả năng phát hoặc nhận bức xạ hồng ngoại trong môi trường xung quanh. Các vật thể phát ra nhiệt độ trên 35 độ C thì sẽ phát ra bức xạ hồng ngoại mà con người không thể nhìn thấy được. Bước sóng của cảm biến hồng ngoại dài hơn ánh sáng khả kiến, nên cảm biến hồng ngoại có thể phát ra các tia vô hình đối với mắt người cũng như nhận vào các tia hồng ngoại.



Hình 2.14: Cấu tạo cảm biến hồng ngoại

- **Phân loại cảm biến hồng ngoại:** Có hai loại cảm biến hồng ngoại chính là cảm biến hồng ngoại chủ động và cảm biến hồng ngoại thụ động.
    - ***Cảm biến hồng ngoại chủ động*** thường có cấu tạo gồm 2 phần là: diode phát sáng (LED) và bộ thu. Khi một vật thể đến gần cảm biến thì ánh sáng hồng ngoại từ LED phát sẽ phản xạ khỏi vật thể. Sau đó chúng sẽ được bộ thu phát hiện.
    - ***Cảm biến hồng ngoại thụ động*** chỉ nhận được các tia hồng ngoại phát ra từ các vật thể như người, động vật hoặc một nguồn nhiệt bất kỳ. Bản thân cảm biến hồng ngoại thụ động không thể tự phát ra tia hồng ngoại nào cả. Bộ phận cảm biến sẽ phân tích để xác định điều kiện báo động sau khi nhận biết được nguồn nhiệt. Chính vì thế người ta gọi đó là thụ động.
- **Cấu tạo của cảm biến hồng ngoại như sau:**
    - ***Đèn led hồng ngoại:*** Đây là thiết bị được phát ra từ nguồn sáng hồng ngoại
    - ***Máy dò hồng ngoại:*** Là thiết bị nhận tín hiệu và phát hiện ra bức xạ hồng ngoại phản xạ trở lại
    - ***Điện trở:*** Là thiết bị có tác dụng đi cường độ dòng điện quá lớn chạy quá đèn led làm cho hệ thống chập cháy
    - ***Dây điện:*** Tác dụng chính là kết nối các chi tiết để tạo nên cảm biến hoạt động ổn định
- **Sơ đồ mạch IR Sensor:** Led phát hồng ngoại luôn luôn phát ra sóng ánh sáng có bước sóng hồng ngoại, led thu bình thường có nội trở lớn, khi led thu nhận tia hồng ngoại chiếu vào đủ lớn thì nội trở giảm xuống.



Hình 2.15: Sơ đồ mạch cảm biến hồng ngoại

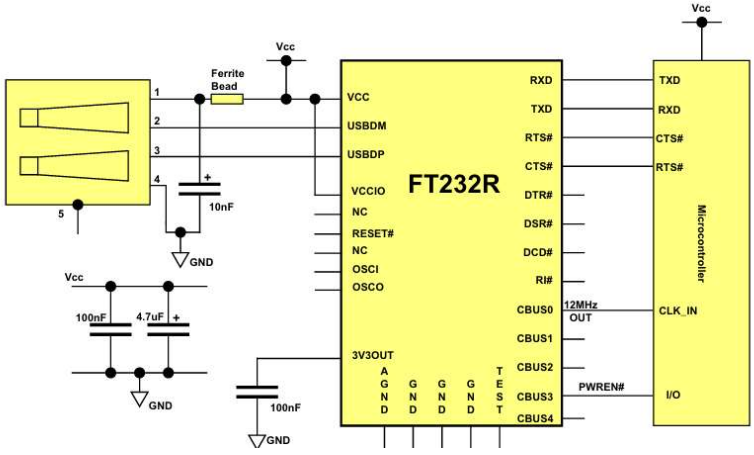
Trường hợp khi có vật cản phía trước thì những chùm tia hồng ngoại đập vào vật cản và phản xạ lại led thu làm cho nó thay đổi giá trị nội trở và dẫn đến thay đổi mức điện áp ở phái đầu vào của op amp. Khi khoảng cách càng gần thì sự thay đổi sẽ càng lớn. Và khi đó điện áp đầu vào không đảo được đánh giá với giá trị điện áp sẽ không đổi ghim trên biến trở R3.

Nếu như giá trị điện áp đầu vào không đảo lớn hơn đầu vào đảo, op amp xuất ra mức 1, nếu như áp đầu vào không đảo nhỏ hơn đầu vào đảo, op amp sẽ xuất ra mức 0. Khi đấy điện trở R1, R2, R4 sẽ được sử dụng để đảm bảo dòng điện tối thiểu mA đi qua các thiết bị LED IR như Photodiode, đèn Led thông thường. Biến trở R3 sẽ dùng để điều chỉnh độ nhạy của mạch.

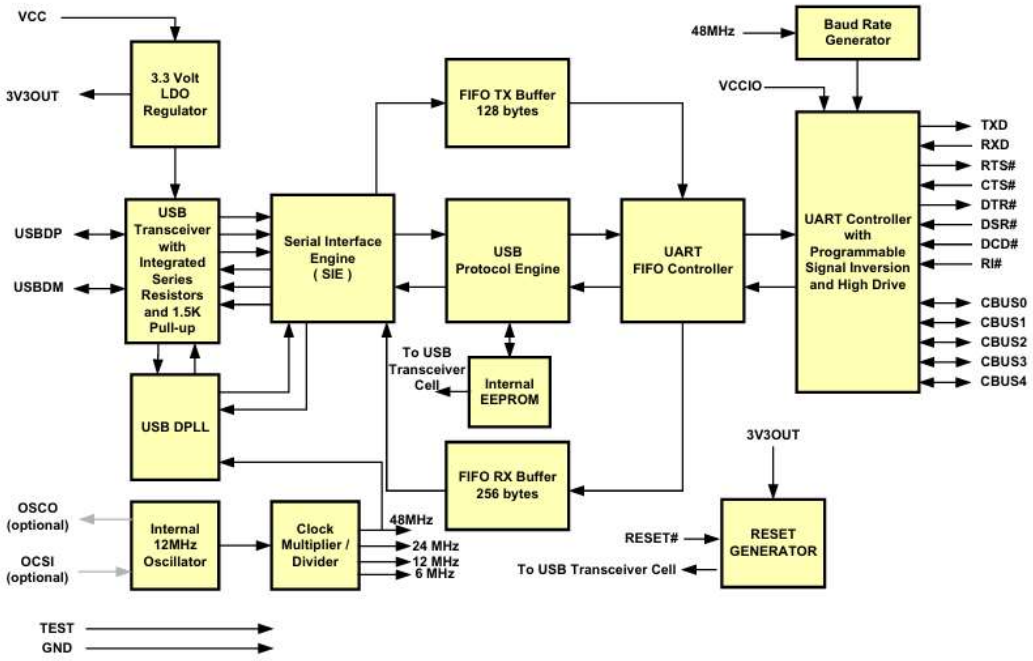
2.1.8 Mạch USB-to-UART (CH340):

Hình 2.16: Mạch USB-to-UART

- ***Thông số kỹ thuật của Mạch chuyển USB UART TTL FT232RL***
  - *IC chính: FT232RL chính hãng FTDI*
  - *Nguồn cấp: 5V từ cổng USB (cổng micro USB hoặc USB Type-C)*
  - *Có ngõ ra nguồn có thể điều chỉnh 3V3 hoặc 5VDC*
  - *Chuyển giao tiếp từ USB sang UART TTL*
  - *Drive hỗ trợ Windows Mac, Linux*
  - *Có cầu chì tự phục hồi: 500mA*
  - *Tốc độ Baudrate: tùy chỉnh*
  - *Kích thước PCB: 36 x 18.5mm*
  - *Trọng lượng: 3g*
  - *Kết nối máy tính với 8051 qua cổng COM4, baud rate 9600.*



Hình 2.17: USB to MCU UART Interface



Hình 2.18: Block Diagram

- ***Sơ đồ chân chân ra của Mạch chuyển USB UART TTL FT232RL***

Tên	Mô tả
DTR	Data terminal ready control output / Handshake signal. (có thể reset arduino khi nạp chương trình)
RXD	Receive asynchronous data Input – nhận tín hiệu
TXD	Transmit asynchronous data output – truyền tín hiệu
VCC	Chân nguồn cấp, có thể chọn 5V hoặc 3.3VDC thông qua Jumper
CTS	Clear to send control input / handshake signal (không sử dụng)

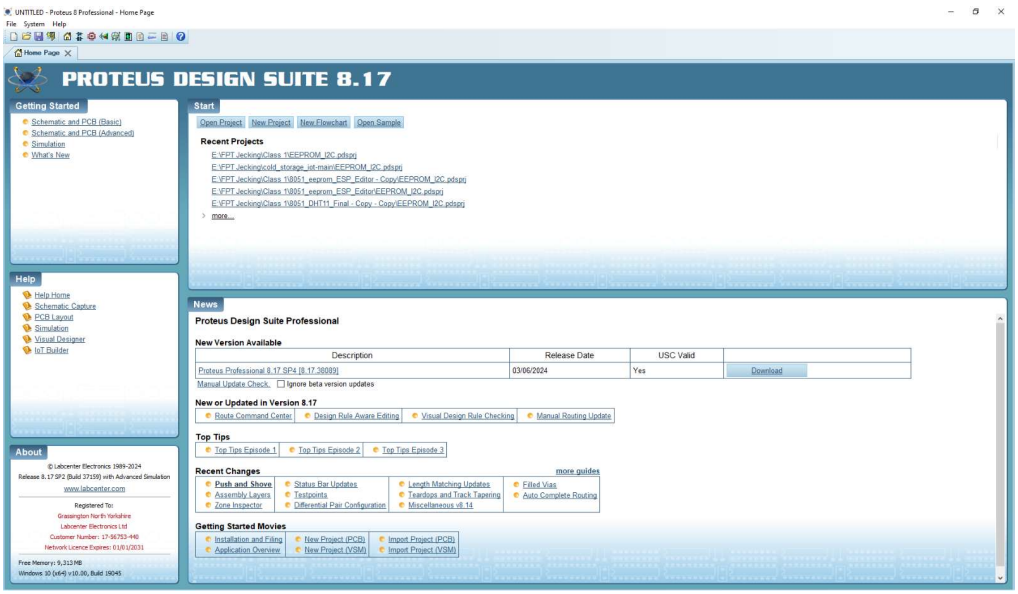
Bảng 2.9: Sơ đồ chân chân ra của Mạch chuyển USB UART TTL FT232RL

2.1.9 Camera giám sát xe ra:

- Tận dụng camera hiện hữu trên máy tính xách tay để duyệt biển số cho xe ra.

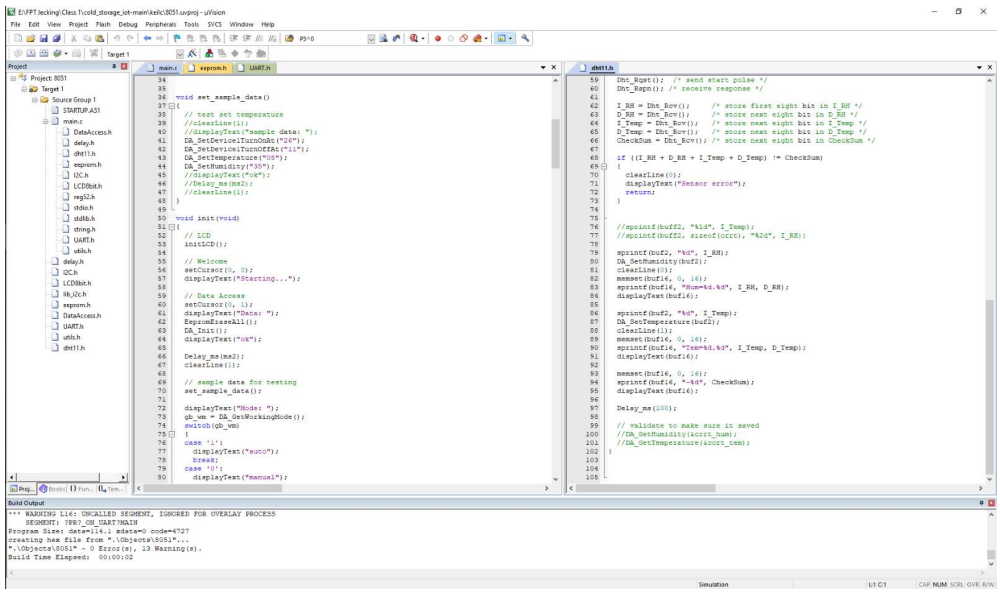
2.2 Phần mềm và công cụ:

2.2.1 Proteus để mô phỏng và chạy thử:



Hình 2.19: Hình ảnh phần mềm mô phỏng Proteus

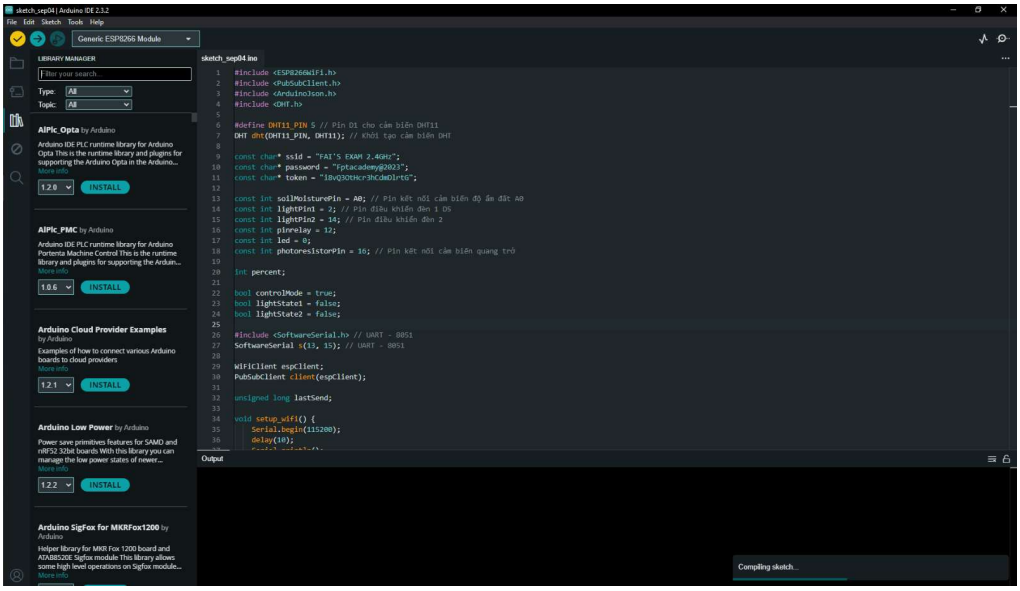
2.2.2 Keil C:



Hình 2.20: Hình ảnh ngôn ngữ lập trình C trên Keil C

- Điều khiển servo, LCD, và nhận tín hiệu UART/RFID.
- Sử dụng ngắt UART để nhận dữ liệu liên tục.

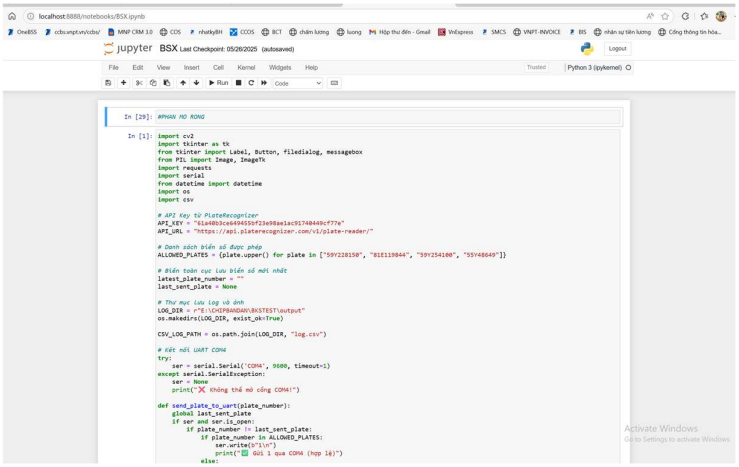
2.2.3 Arduino IDE:



Hình 2.21: Hình ảnh lập trình Arduino IDE cho ESP32

- Điều khiển module RFID MFRC522, kiểm tra UID thẻ.
- Gửi tín hiệu đến 8051 qua GPIO 4, 2.

2.2.4 Python:



Hình 2.22: Hình ảnh lập trình Python

- Sử dụng OpenCV để chụp ảnh webcam, Tkinter để tạo giao diện.

- Gửi ảnh đến API PlateRecognizer, so sánh với danh sách biển số được phép.
- Gửi tín hiệu UART (1 hoặc 0) đến 8051 trong 5 giây.

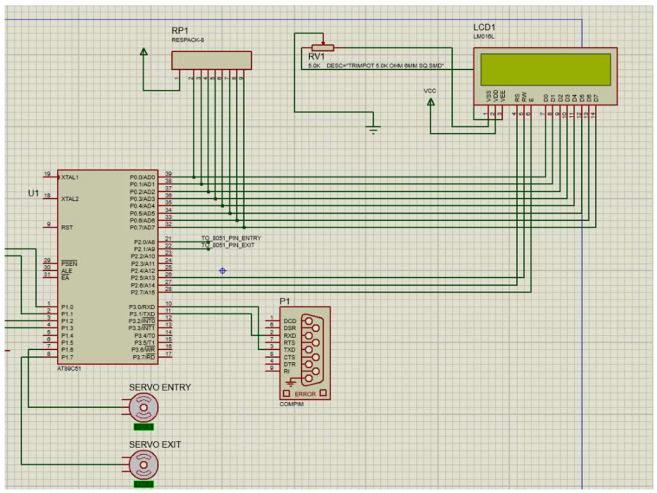
2.2.5 Blynk:

Hình 2.23: Hình ảnh phần mềm BLynk

- Kế hoạch tích hợp với ESP32 để giám sát số xe và trạng thái cổng qua ứng dụng di động.

PHẦN 3: QUÁ TRÌNH THIẾT KẾ

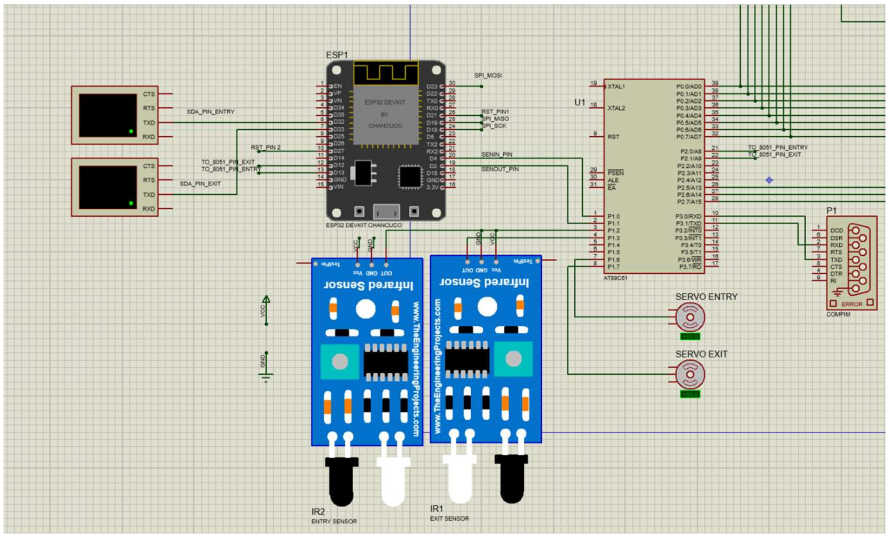
3.1 Sơ đồ kết nối:



Hình 3.1: Sơ đồ kết nối LCD

- Chân kết nối 8051:

Pin	Mô tả
P0.0–P0.7	D0–D7 của LCD 16x2
P1.0 (senin), P1.1 (senout)	Tín hiệu RFID từ ESP32 (Senin, Senout)
P1.2 (signin), P1.3 (signout)	Cảm biến IR phát hiện xe qua cổng
P1.4 (camdata)	Tín hiệu UART từ máy tính
P1.6, P1.7	Servo motor cổng vào/ra
P2.0, P2.1	Tín hiệu RFID từ ESP32 (ESP32_Signal, ESP32_Signal1)
P3.0 (RxD), P3.1 (TxD)	UART kết nối với máy tính qua CH340



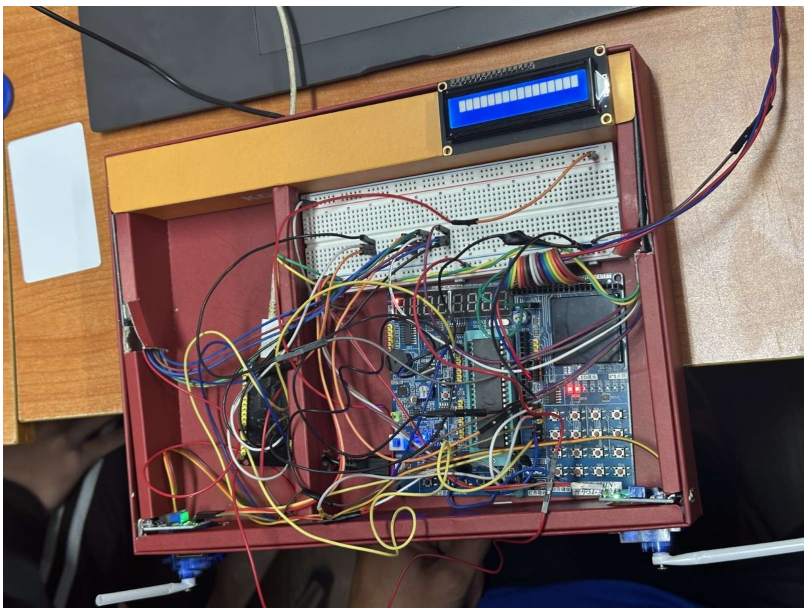
Hình 3.2: Sơ đồ kết nối thiết bị ngoại vi

- Chân kết nối ESP32:

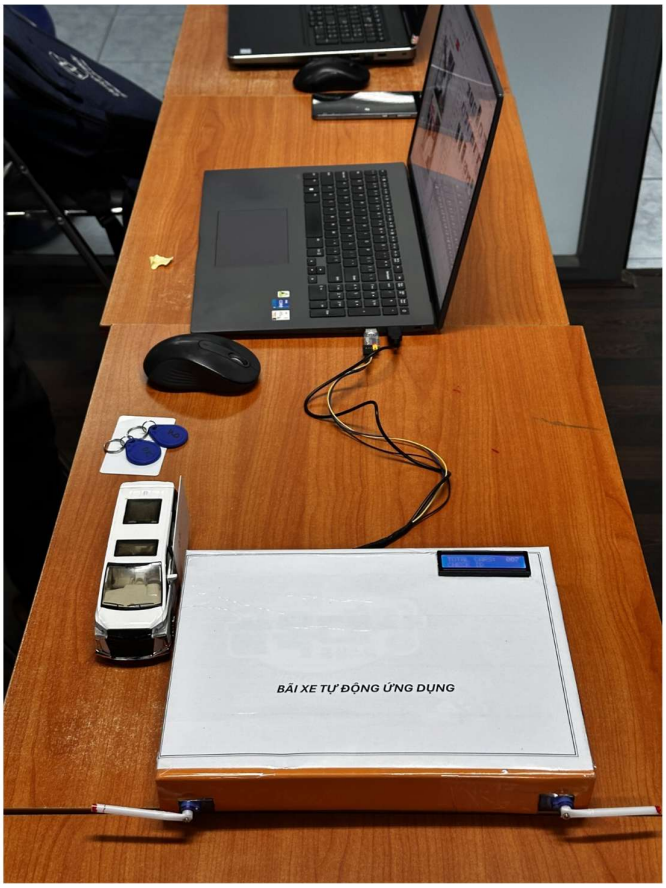
PIN	Chức năng
GPIO 32, 33	SS của module RFID MFRC522 (cổng vào/ra)
GPIO 12, 4, 13, 2	Gửi tín hiệu RFID đến 8051
Wi-Fi	Kết nối với Blynk Cloud

3.2 Lắp đặt mô hình:

- Kit 8051 Pro gắn LCD 16x2, hai servo motor, và hai cảm biến IR.
- ESP32 kết nối với hai module RFID MFRC522 tại cổng vào/ra.
- Mạch USB-to-UART CH340 nối máy tính với 8051 qua COM4.
- Nguồn 5V cấp cho toàn bộ hệ thống.



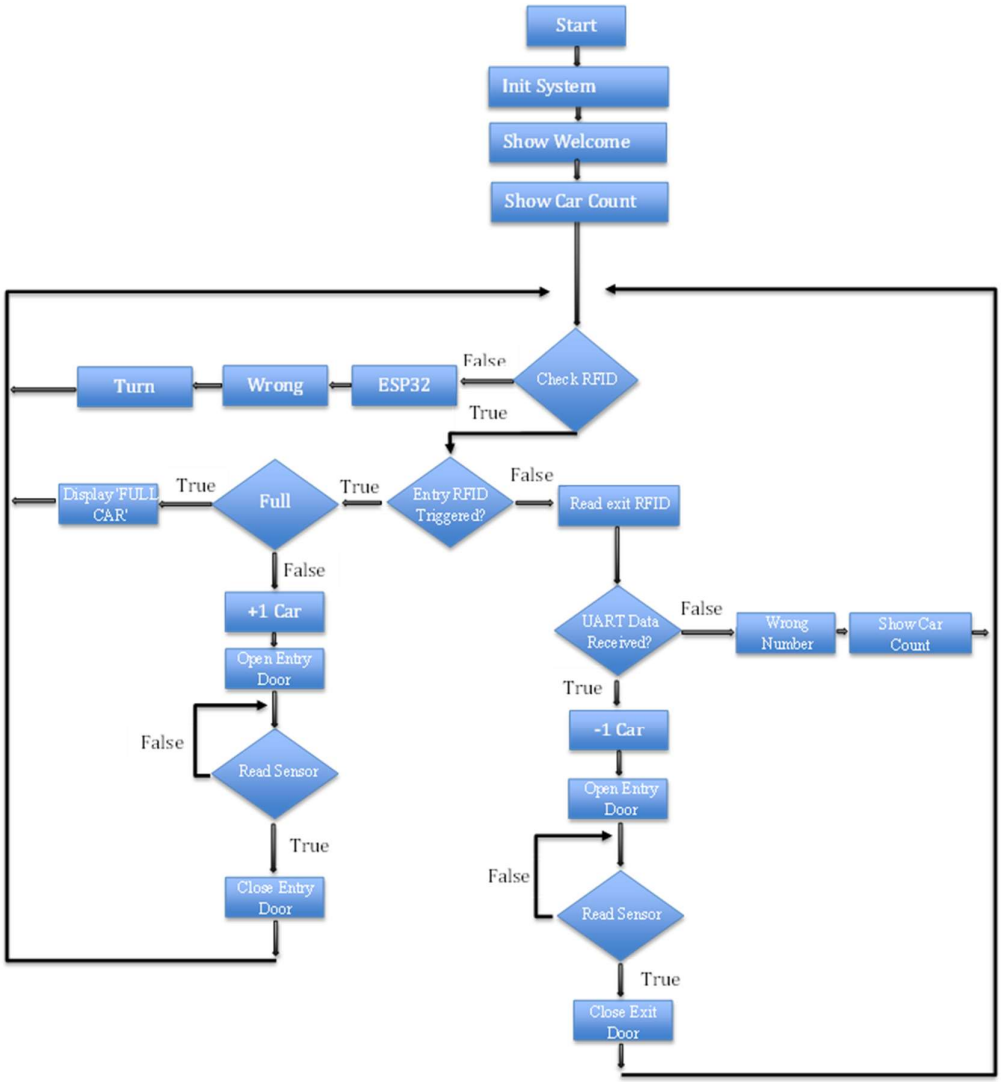
Hình 3.3: Quá trình lắp đặt mô hình



Hình 3.4: Mô hình hoàn thiện

3.3 Flowchart và code:

3.3.1 Flowchart:



Hình 3.5: Flowchart

- Xe vào:
  - Cảm biến IR (senin, signin) phát hiện xe.
  - ESP32 kiểm tra thẻ RFID (MFRC522). Nếu hợp lệ, gửi tín hiệu đến 8051 (ESP32_Signal = 0).
  - 8051 kiểm tra count < 900, mở cổng vào (servo P1.6), tăng count, hiển thị trên LCD.

- Xe ra:
    - Cảm biến IR (senout, signout) phát hiện xe.
    - Python nhận diện biển số qua webcam, gửi 1/0 qua UART (COM4).
    - 8051 nhận tín hiệu UART (camdata). Nếu camdata = 1, mở cổng ra (servo P1.7), giảm count.
    - Thông báo lỗi:
    - Nếu RFID không hợp lệ: LCD hiển thị "WRONG ID".
    - Nếu biển số không hợp lệ: LCD hiển thị "WRONG NUMBER".
    - Nếu count >= 900: LCD hiển thị "FULL CAR".

3.3.2 Mã nguồn:

- Tải code ***Python*** .
- Tải code ***8051*** .
- Tải code ***ESP32*** .

3.3.3 Phân tích mã nguồn:

- ***Python:***
    - Chức năng: Chụp ảnh webcam, gửi đến API PlateRecognizer, hiển thị kết quả trên Tkinter, gửi tín hiệu UART (1/0) trong 5 giây.
    - Đặc điểm: Sử dụng thread để gửi UART liên tục và nhận phản hồi "OK" từ 8051.
    - Thư viện: OpenCV, Tkinter, requests, pyserial.
- ***Arduino (ESP32):***
    - Chức năng: Quản lý hai module RFID MFRC522, kiểm tra UID thẻ, gửi tín hiệu đến 8051 qua GPIO 12,4,13, 2.
    - Đặc điểm: Hỗ trợ danh sách UID cố định, có thể mở rộng sang EEPROM.
- ***Keil C (8051):***
    - Chức năng: Điều khiển servo (P1.6, P1.7), LCD, nhận tín hiệu UART (P3.0, P3.1) và RFID (P2.0, P2.1).

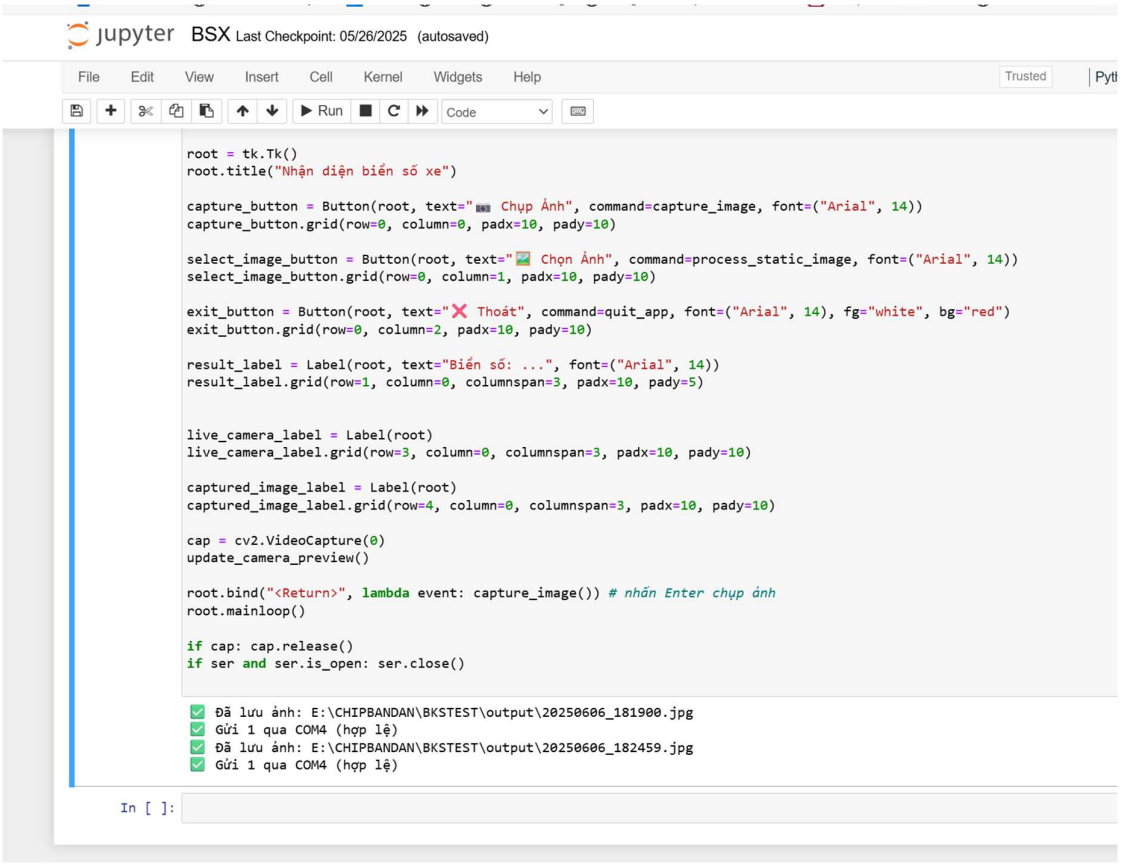
- Đặc điểm: Sử dụng ngắt UART và bộ đệm để nhận dữ liệu liên tục, hiển thị số xe (count) và trạng thái.

3.3.4 Kết quả thực nghiệm:

Hình 3.6: Thực nghiệm

- Nhận diện biển số:
  - API PlateRecognizer đạt độ chính xác 90% (18/20 biển số) trong ánh sáng tốt, thời gian xử lý 2 giây/ảnh.
- RFID:
  - Module MFRC522 xác thực thẻ trong 0.5 giây, tỷ lệ thành công 94% (47/50 lần quét).
- Điều khiển cổng:
  - Servo motor mở/đóng cổng trong 3 giây, ổn định qua 100 chu kỳ.

- UART:
  - Tín hiệu 1/0 gửi lặp trong 5 giây, tỷ lệ nhận thành công 98% sau khi dùng ngắt UART.
- LCD:
  - Hiển thị chính xác số xe (count), trạng thái cổng, và thông báo lỗi ("WRONG ID", "FULL CAR").
- Hạn chế:
  - Nhận diện biển số kém trong ánh sáng yếu (<50 lux).
  - Cảm biến IR bị nhiễu bởi ánh sáng môi trường, cần bộ lọc tín hiệu.
  - Tín hiệu UART ban đầu chỉ gửi một lần, đã khắc phục bằng cơ chế gửi lặp.



Hình 3.7: Lưu ảnh vào thư mục của Python

PHẦN 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết luận:

- Nắm bắt kiến trúc IoT, giao tiếp UART, và tích hợp phần cứng/phần mềm.
- Hoàn thành mô hình bãi đỗ xe thông minh với các chức năng: nhận diện biển số, xác thực RFID, điều khiển cổng, và hiển thị trạng thái.
- Giải quyết các vấn đề kỹ thuật như:
- Lỗi UART chỉ gửi một lần:
- Sử dụng gửi lặp (Python) và ngắt UART (8051).
- Nhiễu cảm biến IR:
- Điều chỉnh vị trí và thêm bộ lọc.
- Tương thích điện áp:
- Sử dụng level shifter giữa ESP32 (3.3V) và 8051 (5V).
- Hệ thống đáp ứng các mục tiêu cơ bản nhưng còn hạn chế về bảo mật và khả năng ứng dụng thực tế.

4.2 Đánh giá bảo mật và an toàn:

- Bảo mật RFID: Danh sách UID lưu cứng trong ESP32, dễ bị truy cập. Đề xuất lưu vào EEPROM hoặc mã hóa UID.
- Bảo mật UART: Tín hiệu 1/0 không mã hóa, có nguy cơ can thiệp. Đề xuất thêm checksum để kiểm tra lỗi.
- An toàn phần cứng: Servo và cảm biến dùng nguồn 5V (LM7805), cần cầu chì để bảo vệ quá tải.
- Dữ liệu camera: Hình ảnh biển số cần lưu trữ an toàn trên Firebase với xác thực người dùng.

4.3 Hướng phát triển:

- Cải thiện lý thuyết IoT: Nghiên cứu sâu hơn về giao thức MQTT hoặc CoAP cho kết nối IoT.
- Tích hợp Blynk: Hoàn thiện giám sát từ xa qua ESP32, hiển thị số xe và trạng thái cổng.

- Nâng cao ANPR: Sử dụng AI (deep learning) để nhận diện biển số trong điều kiện ánh sáng yếu.
- Cảm biến siêu âm: Thêm HC-SR04 để xác định vị trí xe trong ô đỗ.
- Ứng dụng di động: Xây dựng app thực tế để đặt chỗ và thanh toán qua ví điện tử (Momo, ZaloPay).
- Đồng bộ đa bãi: Sử dụng Firebase để quản lý nhiều bãi đỗ, đồng bộ dữ liệu vào/ra.

PHẦN 6: PHÂN CÔNG CÔNG VIỆC NHÓM

Họ và tên	Nhiệm vụ phụ trách
Dương Minh An	Xây dựng hệ thống nhận diện xử lý hình ảnh bằng python, xây dựng mô hình.
Nguyễn Thanh Hào	ESP32, RFID, kết nối UART và thử nghiệm. Thiết kế phần cứng, servo và cảm biến.
Huỳnh Phương Nam	Tổng hợp tài liệu, phân tích hệ thống và viết báo cáo.

PHẦN 7: TÀI LIỆU THAM KHẢO

[1] Atmel Corporation. AT89C51 Datasheet

[2] PlateRecognizer API Documentation. https://app.platerecognizer.com

[3] Espressif Systems. ESP32 Technical Reference Manual

[4] Proteus 8 Professional – Labcenter Electronics

[5] Blynk IoT Platform. https://docs.blynk.io

[6] Python Official Documentation. https://docs.python.org

[7] Arduino MFRC522 Library. https://github.com/miguelbalboa/rfid