**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**

**KHOA ĐIỆN-ĐIỆN TỬ**

**-----☞🕮☜-----**

****

**BÁO CÁO BÀI TẬP LỚN**

**Môn học: HỆ THỐNG NHÚNG**

**NHÓM 2**

**Đề tài: CẢM BIẾN NHIỆT ĐỘ LM35,ĐỒNG HỒ THỜI GIAN THỰC DÙNG PIC16F877A**

**Giảng viên hướng dẫn:** Võ Thiện Lĩnh

**Sinh viên thực hiện:** Ngô Huỳnh Quốc Huy

**MSSV:** 6251020057

**Lớp:** CQ.62.KS.DTTH&CN

**TP.Hồ Chí Minh, Năm 2024**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**

KHOA ĐIỆN-ĐIỆN TỬ

**-----☞🕮☜-----**

****

**BÁO CÁO BÀI TẬP LỚN**

**Môn học HỆ THỐNG NHÚNG**

**NHÓM 2**

**Đề tài: HỆ THỐNG ĐIỂM DANH BẰNG THẺ RFID 13.56MHz SỬ DỤNG VI ĐIỀU KHIỂN PIC16F877A VÀ ESP8266**

**Giảng viên hướng dẫn:** Võ Thiện Lĩnh

**Sinh viên thực hiện: Nguyễn Mạnh Tuấn**

**MSSV: 00**

**Lớp:** CQ.62.KS.DTTH&CN

**TP.Hồ Chí Minh, năm 2024**

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**TP. Hồ Chí Minh, ngày ….… tháng ….… năm ….…**

**Giảng viên hướng dẫn**

**Võ Thiện Lĩnh**

**LỜI MỞ ĐẦU**

Hệ thống nhúng là một lĩnh vực quan trọng trong công nghệ thông tin và điện tử, đóng vai trò trò chơi sau đó thúc đẩy việc phát triển các thiết bị thông minh và tự động hóa. Với sự kết hợp giữa phần cứng và phần mềm, hệ thống được nhúng được thiết kế để thực hiện các công cụ có thể trong một môi trường định nghĩa nhất, từ các thiết bị gia dụng như tủ thông minh, đến các hệ thống phức tạp hơn như ô lái xe hay thiết bị y tế. Sự phát triển nhanh chóng của công nghệ đã thúc đẩy hệ thống ứng dụng nhúng trong nhiều lĩnh vực khác nhau, giúp nâng cao hiệu suất và tiện ích trong cuộc sống hàng ngày. Trong bối cảnh đó, việc nghiên cứu và hiểu rõ về hệ thống nhúng sẽ trở nên cần thiết hơn bao giờ hết, nhắm đáp ứng nhu cầu ngày càng cao của thị trường và xã hội.

**LỜI CẢM ƠN**

Trước hết, nhóm chúng em xin gửi lời cảm ơn chân thành đến nhà trường đã tạo điều kiện tốt nhất để chúng em có một môi trường học tập thoải mái và năng động.

Tiếp theo, nhóm chúng em xin cảm ơn đến thầy giáo Võ Thiện Lĩnh – Giảng viên bộ môn đã giảng dạy tận tình, chi tiết để nhóm chúng em có đủ kiến thức hoàn thành nội dung báo cáo thảo luận này và tạo điều kiện để chúng em có cơ hội được làm việc chung với nhau.

Những ý kiến đóng góp của mọi người đã giúp nhóm em hoàn thiện bài viết của mình một cách toàn diện và phong phú hơn. Mặc dù nhóm em đã cố gắng hoàn thành bài báo cáo này nhưng do chưa có nhiều kinh nghiệm, những hạn chế về kiến thức cũng như thời gian có hạn. Nên trong bài báo cáo này chắc chắn sẽ không tránh khỏi những thiếu sót và một số chỗ nội dung chưa hợp lí. Nhóm Em rất mong nhận được sự nhận xét, góp ý từ thầy để bài báo cáo này được hoàn thiện hơn.

Lời cuối cùng, nhóm em xin gửi lời chúc sức khỏe, thành công, hạnh phúc đến tất cả các thành viên, giảng viên của trường Đại học Giao thông Vận tải – Phân hiệu tại Thành phố Hồ Chí Minh và các bạn trong lớp.

MỤC LỤC

[**PHẦN 1: Ý TƯỞNG VÀ MÔ PHỎNG** 6](#_Toc182249972)

[**1.Ý tưởng:** 6](#_Toc182249973)

[**2. Mạch nguyên lý:** 7](#_Toc182249974)

[**3.Sơ đồ mạch in** 7](#_Toc182249975)

[**4.Mạch thực tế:** 9](#_Toc182249976)

[**PHẦN 2: CÁC THÀNH PHẦN TRONG MẠCH** 9](#_Toc182249977)

[**1.GIỚI THIỆU PIC16F877A** 9](#_Toc182249978)

[**a.NGUYÊN LÝ PIC16F877A** 11](#_Toc182249979)

[**b. GIỚI THIỆU VỀ CHÂN PORT PIC16F8877A:** 12](#_Toc182249980)

[**2. IC DS1307** 13](#_Toc182249981)

[**3.IC CẢM BIẾN NHIỆT LM35** 15](#_Toc182249982)

[**4.MÀN LCD 16X2** 17](#_Toc182249983)

[**PHẦN 3: NGUYÊN LÝ HOẠT ĐỘNG** 18](#_Toc182249984)

[**1.Nguyên Lý Hoạt Động LM35** 18](#_Toc182249985)

[**2.Nguyên Lý Hoạt Động IC DS1307** 18](#_Toc182249986)

[**3.** **Nguyên Lý Hoạt Động Vi Điều Khiển PIC16F877A** 18](#_Toc182249987)

[**PHẦN 4. CHƯƠNG TRÌNH CODE PIC C Compiler** 19](#_Toc182249988)

[**1.GIỚI THIỆU VỀ PIC Compiler** 19](#_Toc182249989)

[**2.CODE PIC Compiler** 20](#_Toc182249990)

[**4.KẾT QUẢ** 25](#_Toc182249991)

[**PHẦN 5: TÀI LIỆU THAM KHẢO** 25](#_Toc182249992)

# **PHẦN 1: Ý TƯỞNG VÀ MÔ PHỎNG**

**1.Ý tưởng:**

**- Sử dụng vi điều khiển pic16f877a kết hợp nó với module RC522 bằng giao tiếp SPI và kết hợp với module Esp 8266-01 nhỏ gọn và giá thành rẻ và có thể giao tiếp UART với pic16f877a**

**1. Mục tiêu dự án:**

**-Thiết kế một mạch có khả năng quét được ID thẻ mặc định và có tần số là 13.56Mhz và có thể nhận biết đâu là thẻ có nằm trong danh sách nhân viên công ty**

**2. Các linh kiện cần thiết:**

**-Vi điều khiển PIC16F877A.**

**-Module RC522**

**-Module ESP8266 -01**

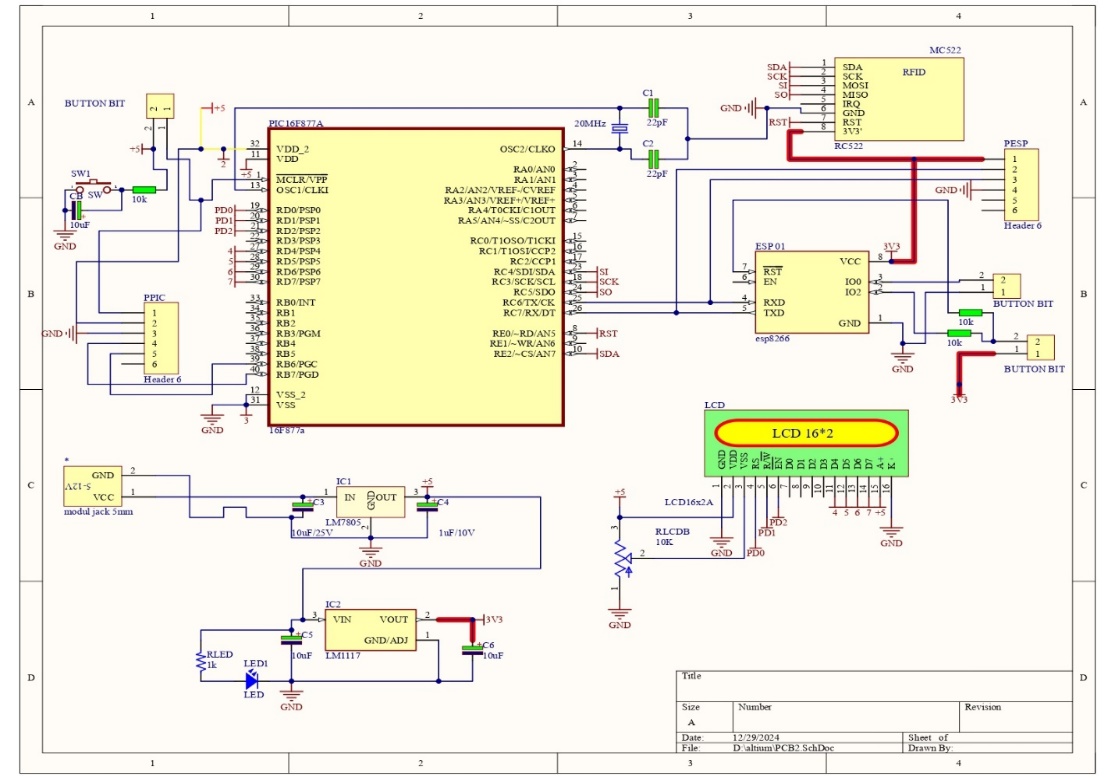
**- IC 7805 , IC LM1117 3.3V**

**-Màn LCD 16x2**

**- Capacitor, Resistor, Quazt , Swich Bit ,Jack DC 5.5mm, Button**

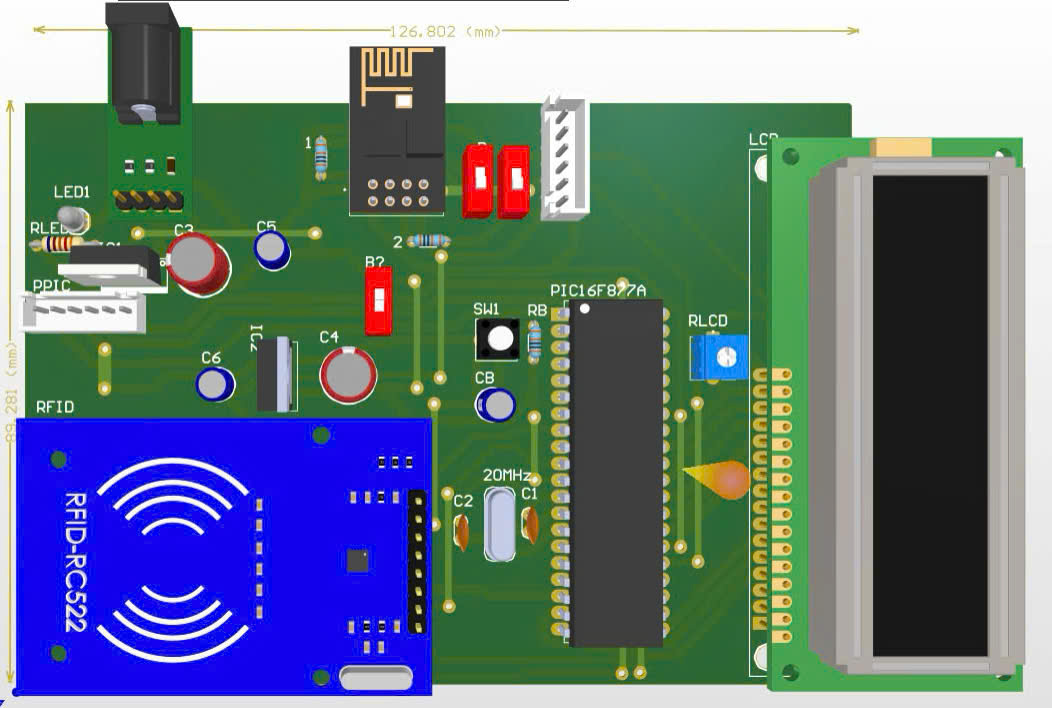
**Mô** **tả:**  Hệ thống hoạt động như sau: RC522 đọc UID của thẻ RFID và gửi dữ liệu đến PIC16F877A qua SPI. PIC kiểm tra UID với danh sách đã lưu, nếu khớp, nó hiển thị tên trên LCD và gửi thông tin qua UART đến ESP8266. ESP8266 nhận dữ liệu và dùng HTTP POST hoặc GET để gửi thông tin lên Google Sheets. Nếu thẻ không hợp lệ, LCD sẽ hiển thị "Thẻ không hợp lệ" và ESP8266 cũng truyền thông báo này lên Google Sheets.

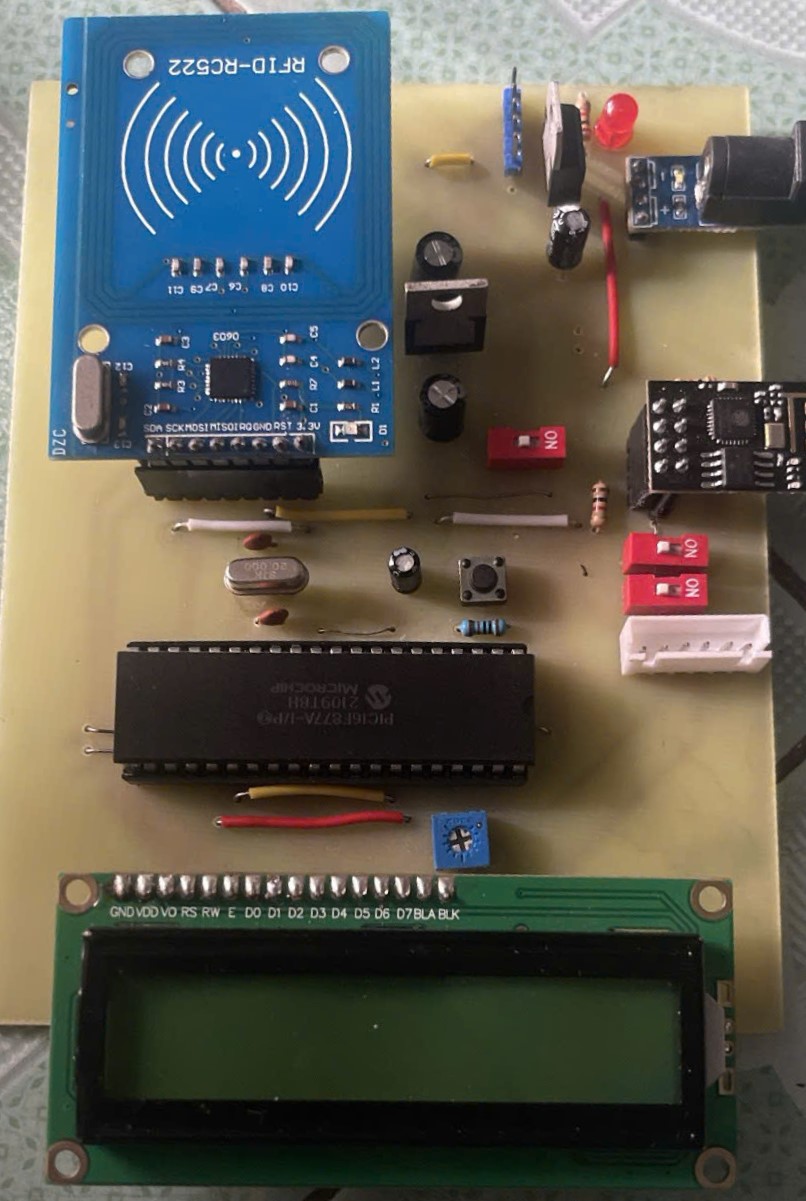
## 2. Mạch nguyên lý:

****

**2.Bản xem PCB**

**3.Bản xem 3D**



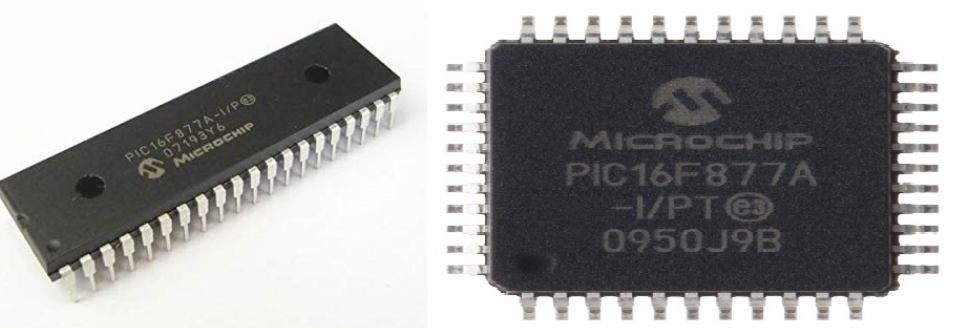
**4.Mạch thực tế: **

# **PHẦN 2: CÁC THÀNH PHẦN TRONG MẠCH**

## **1.GIỚI THIỆU PIC16F877A**

**PIC16F877A** là một vi điều khiển 8-bit của Microchip, thuộc dòng PIC16, được sử dụng rộng rãi trong các ứng dụng nhúng nhờ tính linh hoạt, hiệu năng ổn định và giá thành hợp lý. Đây là một vi điều khiển phổ biến trong lĩnh vực giáo dục, nghiên cứu, và các dự án công nghiệp nhỏ.

**-** PIC16F877A là một Vi điều khiển PIC 40 chân và được sử dụng hầu hết trong các dự án và ứng dụng nhúng. Nó hỗ trợ nhiều giao thức giao tiếp như giao thức nối tiếp, giao thức song song, giao thức I2C. PIC16F877A hỗ trợ cả ngắt chân phần cứng và ngắt bộ định thời.

****

*Hình 2.1. Pic16F8877A.*

- Vi xử lý có rất nhiều loại bắt đầu từ 4 bit cho đến 32 bit, vi xử lý 4 bit hiện nay không còn nhưng vi xử lý 8 bit vẫn còn mặc dù đã có vi xử lý 64 bit. Lý do sự tồn tại của vi xử lý 8 bit là phù hợp với một số yêu cầu điều khiển trong công nghiệp. Các vi xử lý 32 bit, 64 bit thường sử dụng cho các máy tính vì khối lượng dữ liệu của máy tính rất lớn nên cần các vi xử lý càng mạnh càng tốt.

Cấu trúc tổng quát của PIC16F877A như sau :

- 8K Flash Rom.

- 368 bytes Ram.

- 256 bytes EFPROM.

- 5 port vào ra với tín hiệu điều khiển độc lập.

- 2 bộ định thời Timer0 và Timer2 8 bit.

- 1 bộ định thời Timer1 16 bit có thể hoạt động ở cả chế độ tiết kiệm năng lượng

với nguồn xung clock ngồi.

- 2 bộ Capture/ Compare/ PWM.

- 1 bộ biến đổi Analog -> Digital 10 bit, 8 ngõ vào.

- 2 bộ so sánh tương tự.

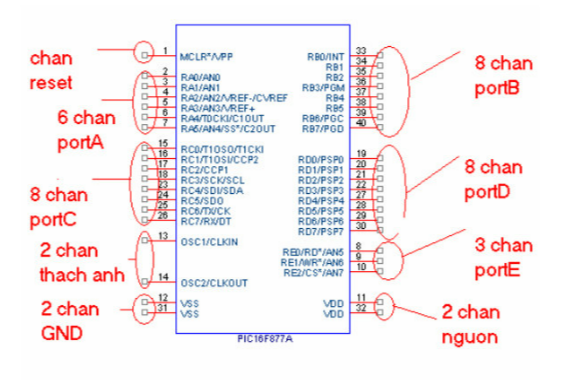
- 1 bộ định thời giám sát (Watch Dog Timer).

- 1 cống song song 8 bit với các tín hiệu điều khiển.

- 1 cổng nối tiếp.

- 15 nguồn ngắt.

### **a.NGUYÊN LÝ PIC16F877A**



*Hình 2.2. Sơ đồ nguyên lý Pic16F8877A.*

- PIC16F877A có tất cả 40 chân.

- 40 chân trên được chia thành 5 PORT, 2 chân cấp nguồn, 2 chân GND, 2 chân thạch anh và một chân dùng để RESET vi điều khiển.

- 5 port cӫa PIC16F877A bao gồm :

+ PORTB : 8 chân

+ PORTD : 8 chân

+ PORTC : 8 chân

+ PORTA : 6 chân

+ PORT E : 3 chân

### **b. GIỚI THIỆU VỀ CHÂN PORT PIC16F8877A:**

**\* Port A:**

- Gồm có 6 chân. Các chân của PortA, ta lập trình để có thể được chức năng "hai chiều”: xuất dữ liệu từ vi điều khiển ra ngoại vi và nhập dữ liệu từ ngoại vi vào vi điều khiển.

- Ngoài ra, PORTA còn có các chức năng quan trọng sau:

+ Ngõ vào Analog của bộ ADC: thực hiện chức năng chuyển từ Analog sang

Digital.

+ Ngõ vào điện thế so sánh.

+ Ngõ vào xung Clock của Timer0 trong kiến trúc phần cứng: thực hiện các nhiệm vụ đếm xung thông qua Timer0 ...

+ Ngõ vào của bộ giao tiếp MSSP (Master Synchronous Serial Port).

**\*Port B:**

- Có 8 chân. Cũng như PORTA, các chân PORTB cũng thực hiện được 2 chức năng: input và output. Hai chức năng trên được điều khiển bới thanh ghi TRISB. Khi muốn chân nào của PORTB là input thì ta set bit tương ứng trong thanh ghi TRISB, ngược lại muốn chân nào là output thì ta clear bit tương ứng trong TRISB.

- Thanh ghi TRISB còn được tích hợp bộ điện trở kéo lên có thể điều khiển được bằng chương trình.

**\*Port C:**

- Có 8 chân và cung thực hiện được 2 chức nang input và output dưới sự điều khiển của thanh ghi TRISC tương tự như hai thanh ghi trên.

- Ngoài ra PORTC còn có các chức năng quan trọng sau:

+ Ngõ vào xung clock cho Timer1 trong kiến trúc phần cứng.

+ Bộ PWM thực hiện chức năng điều xung lập trình được tần số, duty cycle: sử

dụng trong điều khiển tốc độ và vị trí của động cơ…

+ Tích hợp các bộ giao tiếp nối tiếp I2C, SPI, SSP, USART.

**\*Port D:**

- Có 8 chân. Thanh ghi TRIS D điều khiển 2 chức năng input và output của PORT D tương tự như trên. PORT D cũng là cổng xuất dữ liệu của chuẩn giao tiếp song song PSP (Parallel Slave Port).

**\*Port E:**

- Có 3 chân. Thanh ghi điều khiển xuất nhập tương ứng là TRISE. Các chân củaPORT E có ngõ vào analog. Bên cạnh đó PORT E còn là các chân điều khiển của chuẩn giao tiếp PSP.

## **2. ESP 8266-01**

### Giới thiệu

**ESP8266-01** là một module Wi-Fi giá rẻ, nhỏ gọn, được phát triển bởi Espressif Systems. Nó nổi bật nhờ khả năng kết nối không dây, tích hợp vi điều khiển và hỗ trợ giao tiếp dễ dàng với các thiết bị khác. Module này rất phổ biến trong các dự án IoT (Internet of Things) nhờ tính linh hoạt và chi phí thấp.

### **Thông số kỹ thuật:**

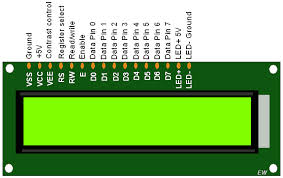
1. **Vi xử lý:**
   * Chip ESP8266EX với lõi Tensilica L106 32-bit RISC.
   * Xung nhịp: 80 MHz (có thể nâng lên 160 MHz).
2. **Bộ nhớ:**
   * Flash: 512 KB (cho firmware và lưu trữ dữ liệu).
3. **Giao diện:**
   * 8 chân GPIO (trong đó chỉ 2 chân GPIO0 và GPIO2 khả dụng trên ESP8266-01).
   * UART hỗ trợ giao tiếp nối tiếp với vi điều khiển hoặc máy tính.
4. **Kết nối:**
   * Hỗ trợ chuẩn Wi-Fi 802.11 b/g/n.
   * Chế độ hoạt động: Station, Access Point (AP), hoặc cả hai (Station + AP).
5. **Nguồn:**
   * Điện áp hoạt động: 3.3V.
   * Dòng tiêu thụ: Khoảng 70-200mA (yêu cầu nguồn cung cấp ổn định).



## **3.MÀN LCD 16X2**

### Giới thiệu

Màn hình LCD 16x2 là một loại màn hình tinh tế (LCD) có thể được sử dụng phổ biến trong nhiều ứng dụng điện tử, đặc biệt là trong các điều khiển dự án. Màn hình này có khả năng hiển thị 16 ký tự trên mỗi dòng và có tổng cộng 2 dòng, có tên là "16x2".



-LCD 16×2 có 16 chân trong đó 8 chân dữ liệu (D0 – D7) và 3 chân điều khiển (RS, RW, EN).

-5 chân còn lại dùng để cấp nguồn và đèn nền cho LCD 16×2.

-Các chân điều khiển giúp ta dễ dàng cấu hình LCD ở chế độ lệnh hoặc chế độ dữ liệu.

-Chúng còn giúp ta cấu hình ở chế độ đọc hoặc ghi.

-LCD 16×2 có thể sử dụng ở chế độ 4 bit hoặc 8 bit tùy theo ứng dụng ta đang làm.

## **4.Module RC522**

**RC522** là một module RFID giá rẻ, được thiết kế để đọc và ghi dữ liệu từ các thẻ RFID (Radio Frequency Identification). Nó sử dụng công nghệ tần số vô tuyến để giao tiếp với thẻ và được sử dụng phổ biến trong các ứng dụng nhận diện, kiểm soát truy cập, và theo dõi.

### **Thông số kỹ thuật:**

1. **Chip xử lý:**
   * MFRC522 do NXP sản xuất.
2. **Tần số hoạt động:**
   * 13.56 MHz (chuẩn ISO/IEC 14443A).
3. **Giao diện:**
   * SPI (Serial Peripheral Interface) là giao thức chính.
   * Ngoài ra, hỗ trợ UART và I2C (ít sử dụng trên module RC522).
4. **Phạm vi đọc:**
   * Khoảng 2-5 cm (tùy thuộc vào thẻ RFID).
5. **Nguồn:**
   * Điện áp hoạt động: 2.5V - 3.3V (có thể cấp 5V qua module).
6. **Dòng tiêu thụ:**
   * Khoảng 13-26 mA ở chế độ hoạt động.
7. **Kích thước:**
   * Module nhỏ gọn, dễ tích hợp (60mm x 40mm).

### **Chức năng chính:**

1. **Đọc dữ liệu từ thẻ RFID:**
   * Module có thể đọc UID (Unique Identifier) hoặc các dữ liệu lưu trữ trong bộ nhớ của thẻ RFID.
2. **Ghi dữ liệu vào thẻ RFID:**
   * Ngoài việc đọc, RC522 cũng hỗ trợ ghi dữ liệu vào thẻ hỗ trợ chuẩn ISO/IEC 14443A.
3. **Kiểm soát truy cập:**
   * RC522 có thể xác thực thẻ thông qua UID hoặc mã hóa dữ liệu để đảm bảo an toàn.
4. **Giao tiếp với vi điều khiển:**
   * Module giao tiếp với vi điều khiển qua SPI để truyền nhận dữ liệu nhanh chóng.

# **PHẦN 3: NGUYÊN LÝ HOẠT ĐỘNG**

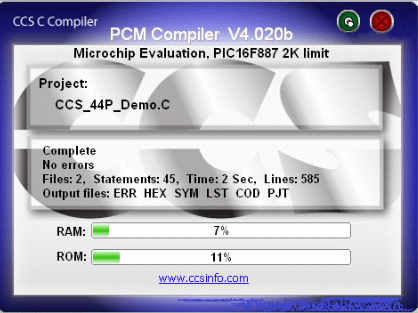
## **Nguyên lý hoạt động của mạch:**

1. **Nguồn cấp và ổn áp:**
   * **IC 7805**: Ổn định điện áp từ 7-12V xuống 5V để cấp nguồn cho PIC16F877A và ESP8266.
   * **IC LM1117-3.3V**: Chuyển đổi từ 5V (đầu ra của 7805) xuống 3.3V, cung cấp nguồn cho module RC522 và ESP8266 (cần 3.3V cho hoạt động ổn định).
2. **Giao tiếp giữa các thành phần:**
   * **PIC16F877A và RC522**: Giao tiếp qua giao thức **SPI**. PIC đọc UID từ RC522 và kiểm tra xem UID có khớp với danh sách trong bộ nhớ.
   * **PIC16F877A và LCD**: Hiển thị thông tin tên khi thẻ hợp lệ.
   * **PIC16F877A và ESP8266**: Gửi dữ liệu qua UART (TX của PIC nối với RX của ESP8266) để đẩy UID hoặc thông tin tên lên Google Sheets thông qua Internet.
3. **RC522**: Phát tín hiệu RF và đọc UID của thẻ RFID khi được đưa vào phạm vi nhận diện.
4. **ESP8266**: Nhận dữ liệu từ PIC và gửi thông qua Wi-Fi tới Google Sheets bằng giao thức HTTP.

**PHẦN 4. CHƯƠNG TRÌNH CODE PIC C Compiler**

## **1.GIỚI THIỆU VỀ PIC Compiler**

- PIC C compiler là ngôn ngữ lập trình cấp cao cho PIC được viết trên nền C. chương trình viết trên PIC C tuân thủ theo cấu trúc của ngôn ngữ lập trình C. Trình biên dịch của PIC C compiler sẽ chuyển chương trình theo chuẩn của C thành dạng chương trình theo mã Hexa (file.hex) để nạp vào bộ nhớ của PIC.



*Hình 1.2. Phần mềm Pic C.*

- PIC C compiler gồm có 3 phần riêng biệt là PCB, PCM và PCH. PCB dùng cho họ MCU với bộ lệnh 12 bit, PCM dùng cho họ MCU với bộ lệnh 14 bit và PCH dùng cho họ MCU với bộ lệnh 16 và 18 bit. Mỗi phần khác nhau trong PIC C compiler chỉ dùng được cho họ MCU tương ứng mà không cho phép dùng chung.

**\*Đặc điểm:**

**- Ngôn ngữ lập trình C:** Cho phép viết mã nguồn dễ hiểu và dễ bảo trì hơn so với Assembly.

**- Hỗ trợ vi điều khiển PIC**:Biên dịch được mã nguồn cho nhiều dòng vi điều khiển PIC của Microchip như PIC12, PIC16, PIC18, ...

**- Thư viện phong phú**: Cung cấp nhiều thư viện tích hợp sẵn cho các chức năng thường dùng như điều khiển I/O, UART, SPI, I2C, PWM, ADC, và nhiều giao tiếp khác.

**- Hỗ trợ tối ưu hóa mã:** PIC C Compiler tối ưu mã để giảm kích thước chương trình và tối ưu hóa tốc độ, rất hữu ích cho các ứng dụng yêu cầu hiệu suất cao.

## **2.CODE PIC Compiler**

#include <16F877A.h>

#device ADC=10

#FUSES NOWDT, NOBROWNOUT, NOLVP

#use delay(crystal=20MHz)

#use rs232(baud=9600, xmit=PIN\_C6, rcv=PIN\_C7)

#use rtos(timer=0, minor\_cycle=10ms)

#define LCD\_ENABLE\_PIN PIN\_D2

#define LCD\_RS\_PIN PIN\_D0

#define LCD\_RW\_PIN PIN\_D1

#define LCD\_DATA4 PIN\_D4

#define LCD\_DATA5 PIN\_D5

#define LCD\_DATA6 PIN\_D6

#define LCD\_DATA7 PIN\_D7

#include <lcd.c>

#define MFRC522\_CS PIN\_E2

#define MFRC522\_SCK PIN\_C3

#define MFRC522\_SI PIN\_C4

#define MFRC522\_SO PIN\_C5

#define MFRC522\_RST PIN\_E0

#include <Built\_in.h>

**char** employeeUID[**3**][**6**] = {

{**0x0E**, **0xFE**, **0x30**, **0x02**, **0xC2**},

{**0x0E**, **0xFE**, **0x30**, **0x02**, **0x62**},

{**0xA3**, **0x2B**, **0x14**, **0x2D**, **0xB1**}

};

**char** employeeNames[**3**][**20**] = {

"NGUYEN MANH TUAN-04",

"BUI TRUNG CHINH-99",

"VO THANH SANG-0324"

};

**char** UID[**6**];

**unsigned** **int** TagType;

**int** found = **0**;

**int** sentFlag = **0**;

**int** timeout\_counter = **0**;

int1 **check\_UID**(**char** Data[], **char** UID[]) {

**for** (**int** i = **0**; i < **5**; i++) {

**if** (UID[i] != Data[i]) {

**return** false;

}

}

**return** true;

}

#task(rate=100ms, max=10ms)

**void** **read\_rfid**() {

**if** (MFRC522\_isCard(&TagType)) {

**if** (MFRC522\_ReadCardSerial(&UID)) {

found = **0**;

**for** (**int** j = **0**; j < **3**; j++) {

**if** (check\_UID(employeeUID[j], UID)) {

found = **1**;

**if** (sentFlag == **0**) {

printf("%s**\n**", employeeNames[j]);

sentFlag = **1**;

}

**break**;

}

}

**if** (found == **0** && sentFlag == **0**) {

**for** (**int** i = **0**; i < **5**; i++) {

printf("%X", UID[i]);

}

sentFlag = **1**;

}

MFRC522\_Halt();

timeout\_counter = **0**; // Ð?t l?i bi?n ð?m th?i gian ch?

}

} **else** {

sentFlag = **0**;

**if** (timeout\_counter < **50**) { // 50 x 100ms = 5 giây

timeout\_counter++;

}

}

}

#task(rate=200ms, max=10ms)

**void** **update\_lcd**() {

**if** (UID[**0**] != **0**) { // Ki?m tra UID có giá tr? h?p l?

**if** (found) {

**for** (**int** j = **0**; j < **3**; j++) {

**if** (check\_UID(employeeUID[j], UID)) {

printf(LCD\_PUTC, "**\f**THANH CONG!!!!");

printf(LCD\_PUTC, "**\n**%s", employeeNames[j]);

**break**;

}

}

} **else** {

printf(LCD\_PUTC, "**\f**KHONG PHAI NV");

printf(LCD\_PUTC, "**\n**ID: ");

**for** (**int** i = **0**; i < **5**; i++) {

printf(LCD\_PUTC, "%X", UID[i]);

}

}

}

**if** (timeout\_counter >= **50**) { // N?u không có quét th? trong 5 giây

printf(LCD\_PUTC, "**\f**XIN CHAO!!!!");

memset(UID, **0**, **sizeof**(UID));

printf(LCD\_PUTC, "**\n**VUILONG QUETTHE");// Ð?t l?i UID

}

}

**void** **main**() {

lcd\_init();

printf(LCD\_PUTC, "XIN CHAO!!!!!");

delay\_ms(**1000**);

printf(LCD\_PUTC, "**\n**VUILONG QUETTHE");

MFRC522\_Init();

// Kh?i t?o các bi?n

memset(UID, **0**, **sizeof**(UID));

TagType = **0**;

found = **0**;

sentFlag = **0**;

timeout\_counter = **0**;

rtos\_run();

}

## **3.CODE ARDUINO IDE dành cho ESP8266**

#include <ESP8266WiFi.h> // Thu vien ho tro ket noi WiFi cho ESP8266

#include "HTTPSRedirect.h" // Thu vien ho tro HTTPS va giao tiep qua HTTPS

#include <WiFiClientSecure.h> // Thu vien ho tro ket noi an toan qua SSL/TLS

#include <Wire.h> // Thu vien giao tiep I2C (khong can thiet trong chuong trinh nay, co the xoa)

**const** **char**\* ssid = "iPhone"; // Ten mang WiFi can ket noi

**const** **char**\* password = "tuantuan"; // Mat khau mang WiFi

**const** **char**\* host = "script.google.com"; // Dia chi server Google Apps Script

**const** **char**\* GScriptId = "AKfycbzWmnf0WaHeXdrMMmnKFKQXZihB1gtHCkam7WS\_tSdd\_fapcMp3LlIR4WwpJKGBO5cC";

// ID cua Google Apps Script de gui du lieu len Google Sheets

String payload\_base = "{**\"**command**\"**: **\"**insert\_row**\"**, **\"**sheet\_name**\"**: **\"**Sheet1**\"**, **\"**values**\"**: ";

// Du lieu co ban (payload) de gui len server Google Apps Script

**const** **int** httpsPort = **443**; // Cong HTTPS mac dinh (443) de giao tiep an toan

HTTPSRedirect\* client = nullptr; // Tao doi tuong client de giao tiep qua HTTPS

String value = ""; // Bien luu tru gia tri nhap vao tu UART

**bool** newData = false; // Co danh dau khi co du lieu moi can gui len server

**void** **setup**() {

Serial.begin(**9600**); // Khoi tao giao tiep UART voi baud rate 9600

Serial.flush(); // Xoa bo nho dem cua UART

Serial.println();

Serial.print("Dang ket noi WiFi: ");

Serial.println(ssid); // In ra ten mang WiFi dang ket noi

Serial.flush();

WiFi.begin(ssid, password); // Ket noi den mang WiFi

**int** attempts = **0**; // Dem so lan thu ket noi

**while** (WiFi.status() != WL\_CONNECTED && attempts < **30**) { // Gioi han so lan ket noi la 30

delay(**500**); // Doi 500ms truoc khi thu lai

Serial.print(".");

attempts++;

}

**if** (WiFi.status() == WL\_CONNECTED) { // Neu ket noi thanh cong

Serial.println("**\n**WiFi da ket noi");

} **else** {

Serial.println("**\n**Khong ket noi duoc WiFi!");

**return**; // Dung chuong trinh neu khong ket noi WiFi

}

client = new HTTPSRedirect(httpsPort); // Khoi tao doi tuong client

client->setInsecure(); // Bo qua xac minh SSL (khong nen dung trong san pham thuc te)

client->setPrintResponseBody(false); // Tat viec in noi dung phan hoi de giam tai

client->setContentTypeHeader("application/json"); // Dinh nghia kieu du lieu gui la JSON

**if** (!client->connected()) { // Kiem tra ket noi den server

client->connect(host, httpsPort); // Ket noi den server Google Apps Script

}

}

**void** **loop**() {

**if** (Serial.available() > **0**) { // Neu co du lieu nhap tu UART

value = Serial.readStringUntil('\n'); // Doc chuoi den khi gap ky tu xuong dong

value.trim(); // Loai bo ky tu trang o dau va cuoi chuoi

**if** (value.length() > **0**) { // Neu chuoi khong rong

newData = true; // Danh dau co du lieu moi can gui

} **else** {

Serial.println("Vui long nhap chuoi hop le."); // Thong bao khi chuoi nhap vao rong

}

}

**if** (newData) { // Neu co du lieu moi can gui

Serial.print("Dang gui du lieu '");

Serial.print(value);

Serial.println("' len Google Sheets...");

sendData(value); // Goi ham gui du lieu len Google Sheets

newData = false; // Reset co sau khi gui xong

}

delay(**100**); // Giam tai CPU voi delay nho

}

**void** **sendData**(String value) {

String url = String("/macros/s/") + GScriptId + "/exec"; // Tao URL de gui du lieu den Apps Script

String payload = payload\_base + "**\"**" + value + "**\"**}"; // Tao chuoi JSON chua du lieu can gui

Serial.println("Dang xuat ban du lieu..."); // Thong bao qua UART

**if** (client->POST(url, host, payload)) { // Gui HTTP POST request den server

Serial.println("Du lieu da duoc xuat ban thanh cong!"); // Neu thanh cong

} **else** {

Serial.println("Loi khi ket noi!"); // Neu that bai

}

delay(**100**); // Delay nho giua cac lan gui

}

**4.CODE dành cho GOOGLE SHEET**

// Enter Spreadsheet ID here

var SS = SpreadsheetApp.openById('**1**Ym-FEebN278x11pYIVIGxMfQpEifI7\_E3qPAuNwCc8Q');// thay Id nay thành Id trang tinh của bạn

var str = "";

function **doPost**(e) {

var parsedData;

var result = {};

try {

parsedData = JSON.parse(e.postData.contents);

}

catch(f){

**return** ContentService.createTextOutput("Error in parsing request body: " + f.message);

}

**if** (parsedData !== undefined){

var flag = parsedData.format;

**if** (flag === undefined){

flag = **0**;

}

var sheet = SS.getSheetByName(parsedData.sheet\_name); // sheet name to publish data to is specified in Arduino code

var dataArr = parsedData.values.split(","); // creates an array of the values to publish

var date\_now = Utilities.formatDate(new Date(), "Asia/Ho\_Chi\_Minh", "yyyy/MM/dd"); // Lấy ngày hiện tại

var time\_now = Utilities.formatDate(new Date(), "Asia/Ho\_Chi\_Minh", "hh:mm:ss a"); // Lấy giờ hiện tại

var value0 = dataArr [**0**]; // value0 from Arduino code

var value1 = dataArr [**1**]; // value1 from Arduino code

var value2 = dataArr [**2**]; // value2 from Arduino code

// read and execute command from the "payload\_base" string specified in Arduino code

**switch** (parsedData.command) {

**case** "insert\_row":

sheet.insertRows(**2**); // insert full row directly below header text

//var range = sheet.getRange("A2:D2"); // use this to insert cells just above the existing data instead of inserting an entire row

//range.insertCells(SpreadsheetApp.Dimension.ROWS); // use this to insert cells just above the existing data instead of inserting an entire row

sheet.getRange('A2').setValue(date\_now); // publish current date to cell A2

sheet.getRange('B2').setValue(time\_now); // publish current time to cell B2

sheet.getRange('C2').setValue(value0); // publish value0 from Arduino code to cell C2

sheet.getRange('D2').setValue(value1); // publish value1 from Arduino code to cell D2

sheet.getRange('E2').setValue(value2); // publish value2 from Arduino code to cell E2

str = "Success"; // string to return back to Arduino serial console

SpreadsheetApp.flush();

**break**;

**case** "append\_row":

var publish\_array = new Array(); // create a new array

publish\_array [**0**] = date\_now; // add current date to position 0 in publish\_array

publish\_array [**1**] = time\_now; // add current time to position 1 in publish\_array

publish\_array [**2**] = value0; // add value0 from Arduino code to position 2 in publish\_array

publish\_array [**3**] = value1; // add value1 from Arduino code to position 3 in publish\_array

publish\_array [**4**] = value2; // add value2 from Arduino code to position 4 in publish\_array

sheet.appendRow(publish\_array); // publish data in publish\_array after the last row of data in the sheet

str = "Success"; // string to return back to Arduino serial console

SpreadsheetApp.flush();

**break**;

}

**return** ContentService.createTextOutput(str);

} // endif (parsedData !== undefined)

**else** {

**return** ContentService.createTextOutput("Error! Request body empty or in incorrect format.");

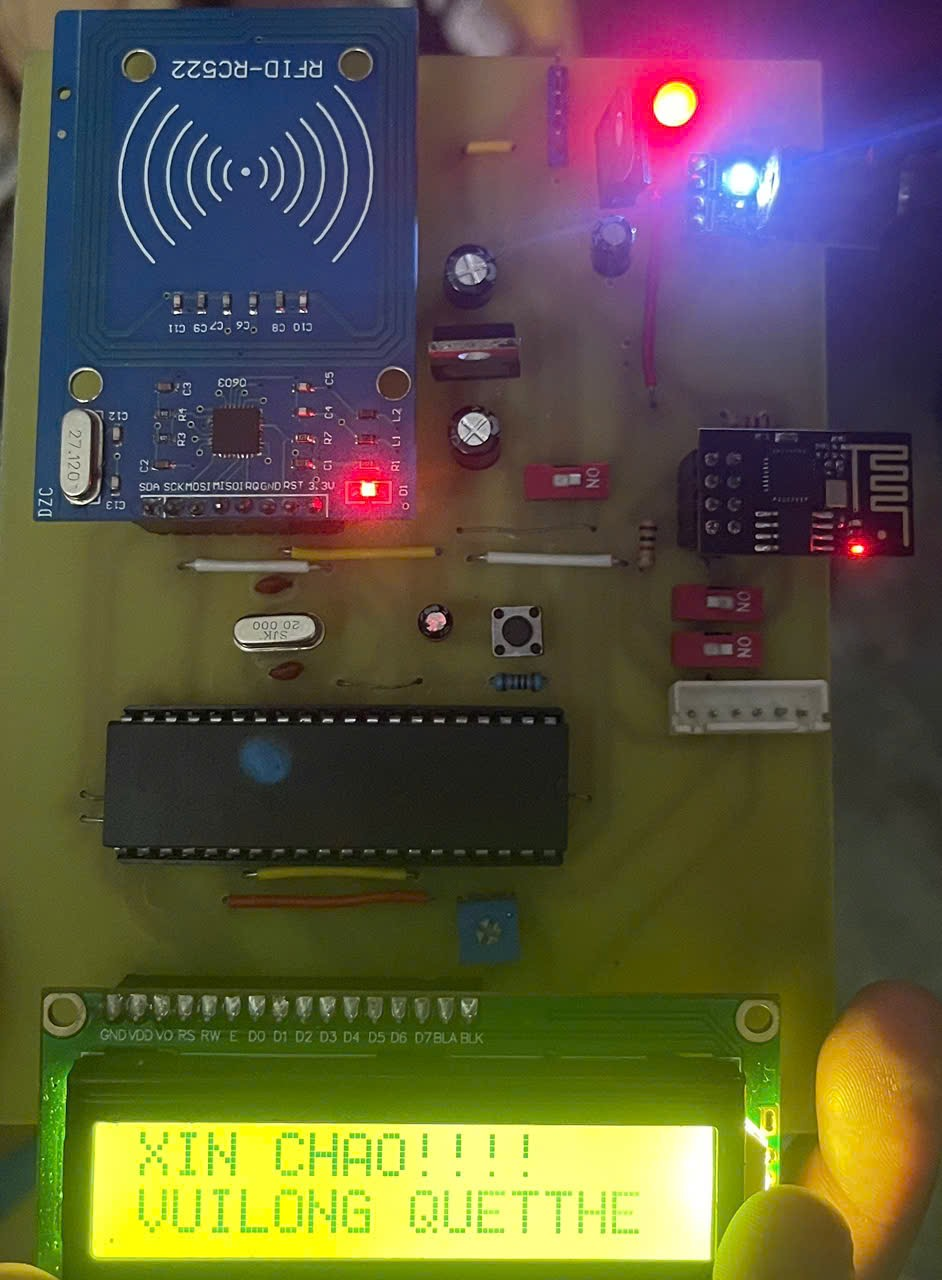
}

}

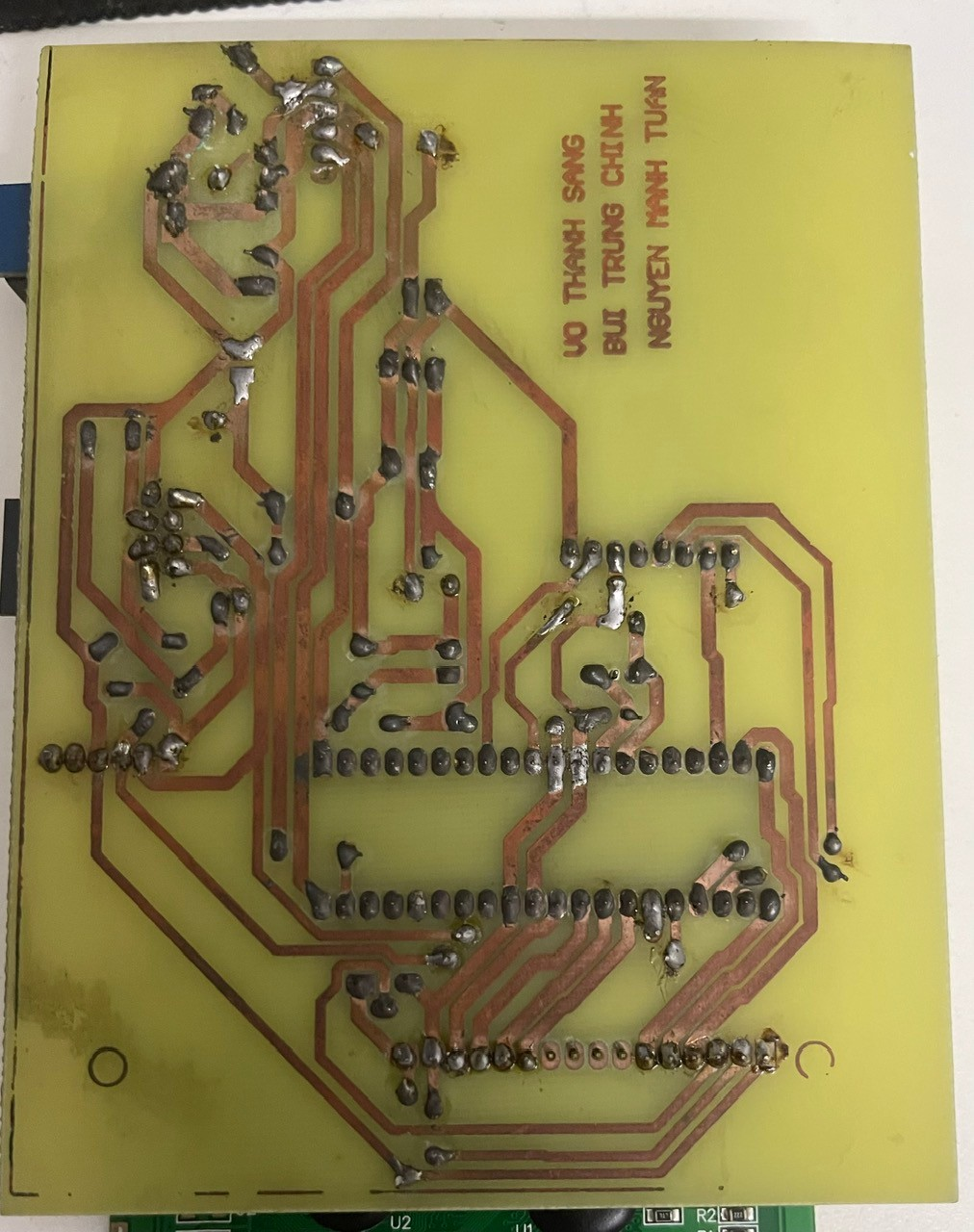
//var SS = SpreadsheetApp.openById('1I-oRgU0WjP9JSqIk3NwmRglQrz3UcQMLm3zyBQqX2yA');

**4.KẾT QUẢ**

Mặt Trước



Mặt Sau

****

# **PHẦN 5: TÀI LIỆU THAM KHẢO**

<https://chatgpt.com/>