**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**BÁO CÁO ĐỒ ÁN**

**THỰC HÀNH VI ĐIỀU KHIỂN**

**ĐỀ TÀI: ĐIỀU KHIỂN CÁNH TAY ROBOT THÔNG QUA SERVO BẰNG PIC16F877A**

Giảng viên hướng dẫn: **Trần Tuấn Kiệt**

**TP. HỒ CHÍ MINH – 2022**

**THÀNH VIÊN NHÓM**

|  |  |  |
| --- | --- | --- |
| STT | HỌ VÀ TÊN | MSSV |
| 1 | Nguyễn Duy Khánh | 19200341 |
| 2 | Lê Minh Phúc | 19200431 |
| 3 | Nguyễn Trần Quốc Khôi | 20200237 |
| 4 | Phan Trọng Khôi | 20200238 |
| 5 | Lê Tuấn Kiệt | 20200239 |
| 6 | Nguyễn Tuấn Kiệt | 20200240 |
| 7 | Nguyễn Ngọc Mẫn | 20200261 |
| 8 | Võ Thành Phúc | 20200311 |
| 9 | Trần Văn Trường | 20200390 |

**BẢNG THỐNG KÊ CẬP NHẬT TÀI LIỆU KỸ THUẬT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SỐ LẦN CẬP NHẬT TÀI LIỆU KỸ THUẬT | | | | | |
| Phiên bản | Mô tả | Thành viên | Ngày | Xác nhận (Trưởng nhóm) | Ngày xác nhận |
| 1.0 | Tạo tài liệu | Lê Tuấn Kiệt | 11/8/22 | Thành Phúc | 12/8/22 |
| 2.0 | Tạo/Cập nhật lần 1 | Lê Tuấn Kiệt | 25/8/22 | Thành Phúc | 26/8/22 |
| 3.0 | Tạo/Cập nhật lần 2 | Thành Phúc | 4/9/22 | Thành Phúc | 5/9/22 |
| Thực hành Vi điều khiển | | | | | |

**TỔNG QUAN ĐỀ TÀI:**

Cùng với sự phát triển không ngừng của các ngành khoa học kỹ thuật, các ngành công nghiệp cũng phát triển nhanh chóng. Việc áp dụng các máy móc hiện đại vào sản xuất là một yêu cầu không thể thiếu trong các nhà máy nhằm tăng năng suất, tăng chất lượng và giảm giá thành sản phẩm.

Song song với sự phát triển đó, công nghệ chế tạo Robot cũng phát triển nhanh chóng đặc biệt là ở các nước phát triển nhằm đáp các nhu cầu về sản xuất, sinh hoạt, quốc phòng…Robot có thể thực hiện những việc mà con người khó thực hiện và thậm chí không thực hiện được như: làm những việc đòi hỏi độ chính xác cao, làm việc trong môi trường nguy hiểm (như lò phản ứng hạt nhân, dò phá mìn trong quân sự), thám hiểm không gian vũ trụ…

Trong các họ Robot, chúng ta không thể không nhắc đến Robot công nghiệp với những đặc thù mà những Robot khác không có. Cánh tay Robot với khả năng linh hoạt được ứng dụng rộng rãi trên khắp các nước thay thể con người làm những công việc trong môi trường độc hại hay chế tạo Robot khác hoặc làm những việc có độ chính xác cao mà con người không thể làm bằng tay được.

Sau một thời gian bàn bạc, nghiên cứu, tham khảo chúng em đã chọn đề tài điều khiển cánh tay Robot thông qua servo bằng vi điều khiển PIC16F877A, bài làm còn dựa trên nhiều lý thuyết, vì vậy chúng em đang hoàn thiện và ứng dụng trong thực tế.

**LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành đến thầy Trần Tuấn Kiệt, giảng viên môn Thực Hành Vi Điều Khiển, trường ĐH Khoa Học Tự Nhiên ĐHQG TP HCM đã trang bị giúp chúng em những kỹ năng cơ bản và kiến thức cần thiết để hoàn thành được đồ án này.

Tuy nhiên, trong quá trình làm đồ án do kiến thức chuyên ngành của chúng em còn hạn chế nên không thể tránh khỏi một vài thiếu sót khi trình bày và đánh giá vấn đề. Rất mong nhận được sự góp ý, đánh giá của các thầy để đề tài của em thêm hoàn thiện hơn.

Em xin chân thành cảm ơn!

**MỤC LỤC**

**DANH MỤC HÌNH**

Hình 1.1 Microchip PIC16F877A 1

Hình 1.2 Cấu trúc của vi điều khiển. 4

Hình 1.3 Cấu hình bên trong vi điểu khiển. 5

Hình 1.4 PIC16F884/887. 7

Hình 1.5 Khung cánh tay Robot. 9

Hình 1.6 LCD 1602. 10

Hình 1.7 Sơ đồ chân LCD 16x2. 11

Hình 1.8 Bảng giá trị PIC16F877A. 13

Hình 1.9 Bảng mã ASCII hiển thị ký tự cho LCD1602. 15

Hình 2.1 LCD1602 địa chỉ Pic16F877A. 16

Hình 2.2 Động cơ servo sg90 17

Hình 2.3 Dạng sóng động cơ servo sg90 18

Hình 2.4 Mô phỏng 20

Hình 2.5 Code trên Mplab 21

Hình 2.6 Mạch thiết kế trên easyeda 30

Hình 2.7 Mạch in PCB 31

Hình 2.8 Mạch mô phỏng 3D 32

Hình 2.9 Mạch in PDF 33

Hình 3.1 Mạch thực tế 36

**DANH MỤC BẢNG**

**CHƯƠNG 1: LINH KIỆN CẦN THIẾT LÀM CÁNH TAY ROBOT BẰNG PIC16F877A**

**1. Vi điều khiển PIC16F877A**

1.1 Giới thiệu 1

1.2 Đặc điểm 2

1.3 Cấu trúc ngoại vi 2

1.4 Cấu trúc vi điều khiển 3

1.5 Cấu trúc bên trong của vi điều khiển 4

1.6 Cấu hình chân của vi điều khiển 6

**2. Khung cánh tay Robot 4 bậc tự do**

2.1 Giới thiệu 9

2.2 Thông số kỹ thuật 9

**3. LCD 1602**

3.1 Giới thiệu 10

3.2 Thông số kỹ thuật 10

3.3 Sơ đồ chân LCD 16x2 11

3.4 Địa chỉ ba vùng nhớ 12

3.5 Các lệnh điều khiển của LCD 13

3.6 Bảng mã ASCII sử dụng cho LCD 15

3.7 Bảng địa chỉ cho LCD 16

**4. Động cơ servo sg90**

4.1 Giới thiệu 16

4.2 Thông số kỹ thuật 17

4.3 Dạng sóng 17

4.4 SERVO giao tiếp PIC 18

**CHƯƠNG 2:** **THIẾT KẾ VÀ CÀI ĐẶT**

2.1 Chuẩn bị 18

2.2 Thiết kế và thực hiện phần mềm 18

2.2.a Thiết kế sơ đồ nguyên lý 18

2.2.b Mô phong trên phần mềm 19

2.2.c Lập trình code 21

2.3 Thiết kế và thực hiện phần cứng 29

2.4 Kết quả thực hiện 36

**KẾT LUẬN**

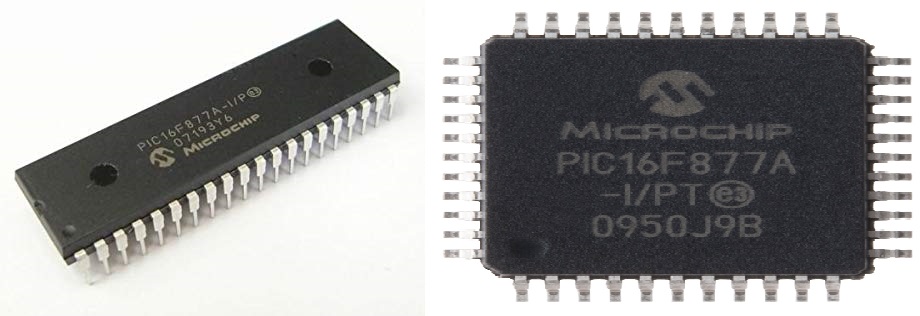
**- VIDEO ĐIỀU KHIỂN CÁNH TAY ROBOT**

**- TÀI LIỆU THAM KHẢO**

**CHƯƠNG 1: LINH KIỆN CẦN THIẾT LÀM CÁNH TAY ROBOT THÔNG QUA SERVO BẰNG PIC16F877A**

1. **VI ĐIỀU KHIỂN PIC16F877A** 
   1. **GIỚI THIỆU:**

PIC là một họ vi điều khiển RISC được sản xuất bởi công ty Microchip Technology. Dòng PIC đầu tiên là PIC1650 sau đó phát triển lên nhiều dòng khác nhau như: Pic10F, Pic12F , Pic16F , Pic18F , Pic24F , Pic32F.



Hình 1.1 Microchip PIC16F877A

Bộ vi điều khiển PIC16F877A là một trong những bộ vi điều khiển nổi tiếng nhất trong giới kỹ thuật điện tử. Bộ vi điều khiển này có nhiều ưu điểm thuận tiện cho việc sử dụng, lập trình chương trình điều khiển cũng rất dễ dàng.

Một trong các ưu điểm nổi bật là có thể ghi xóa nhiều lần, vì có công nghệ bộ nhớ Flash. Nó có tất cả 40 chân trong đó có 33 chân là I/O. PIC 16F877A được sử dụng nhiều trong các dự án có dùng vi điều khiển PIC.

Ta có thể tìm thấy PIC16F877A trên nhiều thiết bị điện tử. Được sử dụng cho các thiết bị điều khiển từ xa, thiết bị bảo mật và an ninh, các thiết bị tự động trong gia đình và một số các thiết bị trong công nghiệp.

**1.2 ĐẶC ĐIỂM:**

* Có 35 bộ lệnh
* Hoạt động với tần số lên đến 20Mhz
* Điện áp hoạt động từ 4.2V đến 5V. Nếu cấp điện áp vượt quá 5V thì sẽ bị hư.
* Không có bộ dao động bên trong giống như PIC18F46K22, PIC18F4550.
* Dòng điện tối đa trên mỗi PORT khoảng 100mA. Do vậy, dòng điện tối đa cho mỗi chân GPIO của PIC 16F877A là 10mA
* Có 4 package: PDIP 40 chân, PLCC 44 chân, TQFP 44 chân, QFN 44 chân

**1.3 CẤU TRÚC NGOẠI VI:**

- Có 35 chân I/O cho phép lựa chọn hướng độc lập:

+ Mỗi ngõ ra có thể nhận/cấp dòng lớn khoảng 25mA nên có thể trực tiếp điều khiển led

+ Có các port báo ngắt khi có thay đổi mức logic.

+ Có các port có điện trở kéo lên bên trong có thể lập trình

+ Có ngõ vào báo thức khỏi chế độ công suất cực thấp.

- Có module so sánh tương tự:

+ Có 2 bộ so sánh điện áp tương tự

+ Có module nguồn điện áp tham chiếu có thể lập trình.

+ Có nguồn điện áp tham chiếu cố định có giá trị bằng 0,6V.

+ Có các ngõ vào và các ngõ ra của bộ so sánh điện áp.

+ Có chế độ chốt SR.

- Có bộ chuyển đổi tương tự sang số, có 14 bộ chuyển đổi tương tự với độ phân giải 10 bit

- Có timer0: 8 bit hoạt động định thời/đếm xung ngoại có bộ chia trước có thể lập trình.

- Có timer1:

+ 16 bit hoạt động định thời/đếm xung ngoại có bộ chia trước có thể lập trình.

+ Có ngõ vào cổng của timer1 để có thể điều khiển timer1 đếm từ tín hiệu bên ngoài.

+ Có bộ dao động công suất thấp có tần số 32kHz.

- Có timer2: 8 bit hoạt động định thời với thanh ghi chu kỳ, có bộ chia trước và chia sau.

- Có module capture, compare và điều chế xung PWM+ nâng cao

+ Có bộ capture 16 bit có thể đếm được xung với độ phân giải cao nhất là 12,5ns.

+ Có bộ điều chế xung PWM với số kênh ngõ ra là 1, 2 hoặc 4, có thể lập trình với tần số lớn nhất là 20kHz.

+ Có ngõ ra PWM điều khiển lái.

- Có module capture, compare và điều chế xung PWM

+ Có bộ capture 16 bit có thể đếm được xung với chu kỳ cao nhất là 12,5ns.

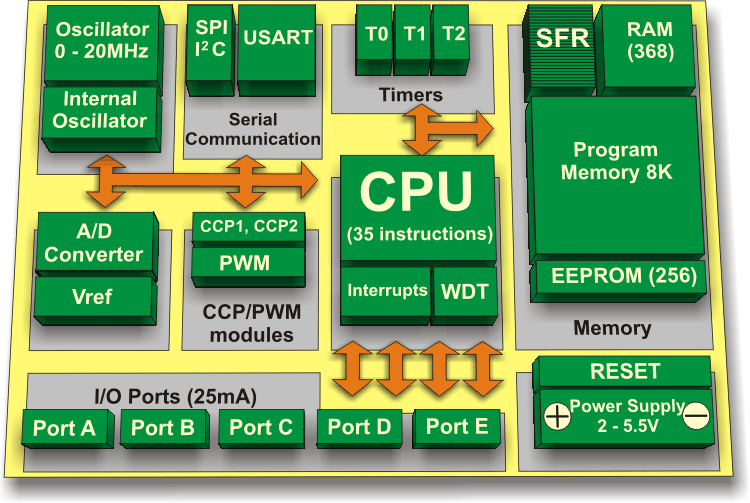
+ Có bộ so sánh 16 bit có thể so sánh xung đếm với chu kỳ lớn nhất là 200ns

+ Có bộ điều chế xung PWM có thể lập trình với tần số lớn nhất là 20kHz.

- Có thể lập trình trên bo ISP thông qua 2 chân.

- Có module truyền dữ liệu nối tiếp đồng bộ MSSP hổ trợ chuẩn truyền 3 dây SPI, chuẩn I2C ở 2 chế độ chủ và tớ.

**1.4 CẤU TRÚC CỦA VI ĐIỀU KHIỂN**



Hình 1.2 Cấu trúc của vi điều khiển

- Các khối bên trong vi điều khiển bao gồm:

+ Có khối thanh ghi định cấu hình cho vi điều khiển.

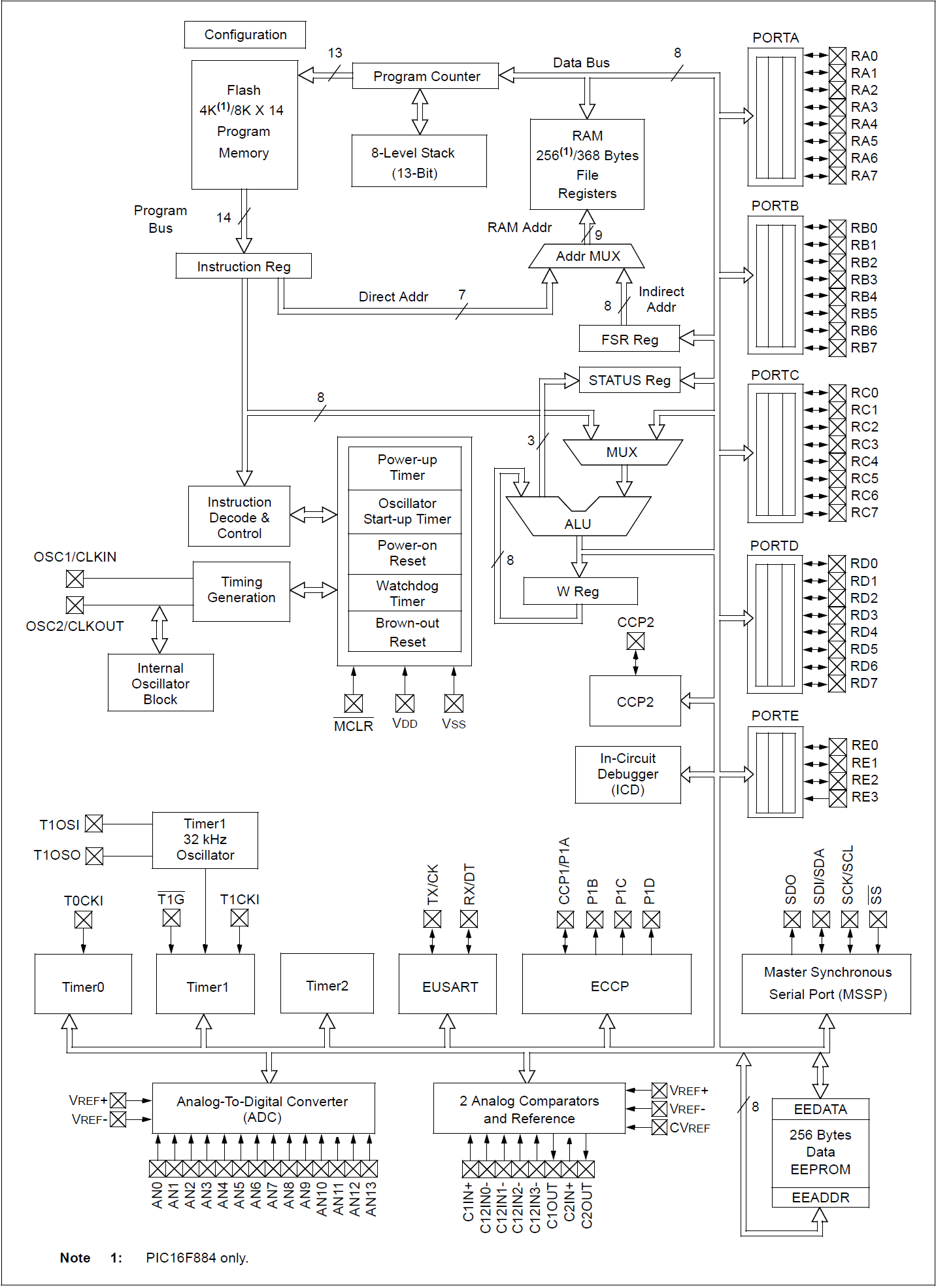
+ Có khối bộ nhớ chương trình có nhiều dung lượng cho 5 loại khác nhau.

+ Có khối bộ nhớ ngăn xếp 8 cấp (8 level stack).

+ Có khối bộ nhớ Ram cùng với thanh ghi FSR để tính toán tạo địa chỉ cho 2 cách truy xuất gián tiếp và trực tiếp.

+ Có thanh ghi lệnh (Instruction register) dùng để lưu mã lệnh nhận về từ bộ nhớ chương trình.

**1.5 CẤU HÌNH BÊN TRONG CỦA VI ĐIỀU KHIỂN:**



Hình 1.3 Cấu hình bên trong vi điểu khiển

- Có thanh ghi trạng thái (status register) cho biết trạng thái sau khi tính toán của khối ALU.

- Có thanh ghi FSR.

- Có khối ALU cùng với thanh ghi working hay thanh ghi A để xử lý dữ liệu.

- Có khối giải mã lệnh và điều khiển (Instruction Decode and Control).

- Có khối dao động nội (Internal Oscillator Block).

- Có khối dao động kết nối với 2 ngõ vào OSC1 và OSC2 để tạo dao động.

- Có khối các bộ định thời khi cấp điện PUT, có bộ định thời chờ dao động ổn định, có mạch reset khi có điện, có bộ định thời giám sát watchdog, có mạch reset khi phát hiện sụt giảm nguồn.

- Có khối bộ dao động cho timer1 có tần số 32kHz kết nối với 2 ngõ vào T1OSI và T1OSO.

- Có khối CCP2 và ECCP.

- Có khối mạch gỡ rối (In-Circuit Debugger IDC).

- Có khối timer0 với ngõ vào xung đếm từ bên ngoài là T0CKI.

- Có khối truyền dữ liệu đồng bộ/bất đồng bộ nâng cao.

- Có khối truyền dữ liệu đồng bộ MSSP cho SPI và I2C.

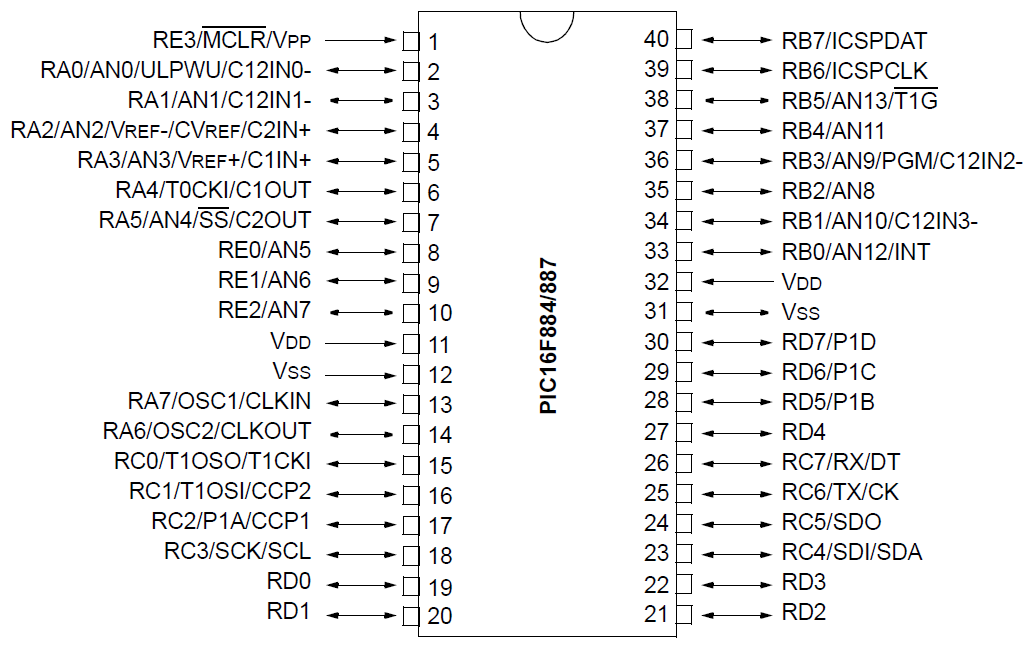
- Có khối bộ nhớ Eeprom 256 byte và thanh ghi quản lý địa chỉ EEADDR và thanh ghi dữ liệu EEDATA.

- Có khối chuyển đổi tín hiệu tương tự sang số ADC.

- Có khối 2 bộ so sánh với nhiều ngõ vào ra và điện áp tham chiếu.

- Có khối các port A, B, C, E và D

**1.6 CẤU HÌNH CHÂN CỦA VI ĐIỀU KHIỂN**



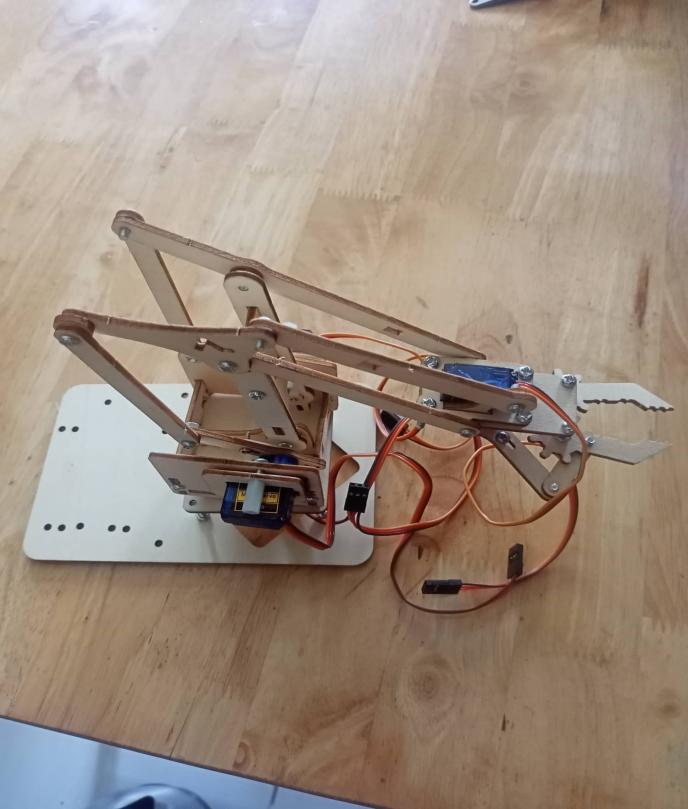
Hình 1.4 PIC16F884/887

* Chân 1: MCLR là chân clear của mạch này. Nó sẽ khởi động lại vi điều khiển và được kích hoạt bởi mức logic thấp, có nghĩa là chân này phải được cấp liên tục một điện áp 5V và nếu cấp điện áp 0V thì PIC16F877A sẽ bị đặt lại.
* Một nút nhấn và một điện trở được kết nối đến chân này. Chân MCLR này luôn được cấp điện áp 5V. Khi muốn khởi động lại mạch. Bạn chỉ cần nhấn vào nút nhấn thì chân MCLR sẽ được đưa về 0 và mạch được đặt lại.
* Chân 2 RA0/AN0: PORT A có 6 chân, từ chân số 2 đến chân số 7. Tất cả đều là các chân xuất, nhập dữ liệu hai chiều. Chân số 2 là chân đầu tiên của PORT A. Chân này có thể được sử dụng như một chân tương tự (analog) chân AN0. Nó được tích hợp bộ chuyển đổi analog sang digital.
* Chân 3 RA1/AN1: Đầu vào tín hiệu analog 1
* Chân 4 RA2/AN2/Vref-: Có thể hoạt động như đầu vào analog thứ 2 hoặc chân điện áp tham chiếu âm.
* Chân 5 RA3/AN3/Vref+: Có thể hoạt động như đầu vào analog thứ 3 hoặc chân điện áp tham chiếu dương.
* Chân 6 RA0/T0CKI: Với timer 0, chân này hoạt động được như một đầu vào xung clock và đầu ra open drain.
* Chân 7 RA5/SS/AN4: Có thể hoạt động như một đầu vào analog thứ 4. Có cổng nối tiếp đồng bộ và là chân SS cho cổng này.
* Chân 8 RE0/RD/AN5: PORT E bắt đầu từ chân số 8 đến chân số 10 và là cổng I/O hai chiều. Nó còn là cổng analog thứ 5 hoặc là chân RD (tích cực mức logic thấp) cho cổng slave giao tiếp song song
* Chân 9 RE1/WR/AN6: Là đầu vào analog thứ 6 và là chân WR (tích cực mức logic thấp) cho cổng slave giao tiếp song song.
* Chân 10 RE2/CS/A7: Là đầu vào analog 7 và là chân CS cho cổng slave song song.
* Chân 11 và 32 VDD: Đây là hai chân cấp nguồn 5V.
* Chân 12 và 31 VSS: Các chân tham chiếu nối đất cho I/O và các chân logic. Chúng nên được nối với 0V hoặc mắc GND.
* Chân 13 OSC1/CLKIN: Là đầu vào bộ dao động hoặc chân đầu vào xung nhịp bên ngoài.
* Chân 14 OSC2/CLKOUT: Đây là chân đầu ra của bộ dao động. Một bộ dao động thạch anh được nối vào giữa hai chân 13 và 14 để cấp xung nhịp bên ngoài cho bộ vi điều khiển. ¼ tần số của OSC1 được OSC2 xuất ra trong chế độ RC. Điều này xác định tốc độ chu kỳ xử lý lệnh.
* X`Chân 15 RC0/T1OCO/T1CKI: PORT C có 8 chân. Là cổng I/O hai chiều. Trong số đó, chân 15 là chân đầu tiên. Nó có thể là đầu vào xung nhịp của bộ định thời 1 hoặc đầu ra bộ dao động của bộ định thời 2.
* Chân 16 RC11/T1OSI/CCP2: Là đầu vào dao động của bộ định thời 1 hoặc đầu vào capture 2 / đầu ra so sánh 2 / đầu ra PWM 2.
* Chân 17 RC2/CCP1: Đầu vào capture 1/ đầu ra so sánh 1/ đầu ra PWM1
* Chân 18 RC3/SCK/SCL: Đầu ra của chế độ SPI hoặc I2C và có thể là I/O cho bộ dao động nối tiếp đồng bộ.
* Chân 23 RC4/SDI/SDA: Chân dữ liệu trong chế độ SPI hoặc là chân xuất nhập dữ liệu chế độ I2C.
* Chân 24 RC5/SDO: Là chân xuất dữ liệu chế độ SPI.
* Chân 25 RC6/TX/CK: Có thể là chân xung clock đồng bộ hoặc chân truyền không đồng bộ UART.
* Chân 26 RC7/RX/DT: Là chân dữ liệu đồng bộ hoặc chân nhận tín hiệu UART.
* Các chân 19, 20, 21, 22, 27, 28, 29, 30: Tất cả các chân này đều thuộc PORT D, đây là một cổng I/O hai chiều. Khi bus vi xử lý được kết nối, nó có thể hoạt động ncho cổng slave giao tiếp dữ liệu song song.
* Chân 33-40 PORT B: Hai chân này đều thuộc PORTB. Trong đó RB0 có thể được sử dụng làm chân ngắt ngoài và RB6 và RB7 có thể được sử dụng làm chân debugger.

**2. KHUNG CÁNH TAY ROBOT 4 BẬC TỰ DO**

**2.1 GIỚI THIỆU :**

Cánh Tay Robot 4 Bậc Tự Do là thiết bị cơ khí tự động được áp dụng rộng rãi trong lĩnh vực công nghệ hiện nay, bạn có thể nhìn thấy nó trong sản xuất công nghiệp, điều trị y tế, giảng dạy và học tập…



Hình 1.5 Khung cánh tay robot

**2.2. THÔNG SỐ KỸ THUẬT**

- Servo tương thích: MG90S, SG90

- Điện áp dụng làm việc: 4.8 → 6V

- Dòng không tải: 100mA

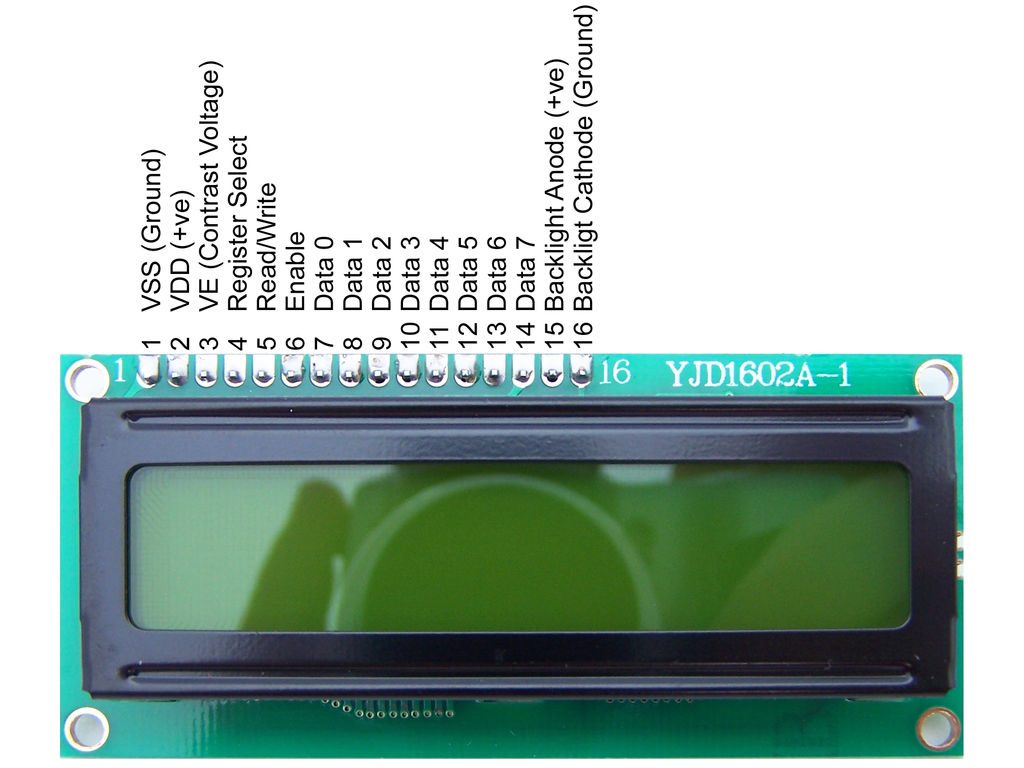
- Góc quay: 180 độ

- Vật liệu: Gỗ

**3. LCD 1602**

**3.1 GIỚI THIỆU:**

Màn hình text LCD1602 xanh lá sử dụng driver HD44780, có khả năng hiển thị 2 dòng với mỗi dòng 16 ký tự, màn hình có độ bền cao, rất phổ biến, nhiều code mẫu và dễ sử dụng thích hợp cho những người mới học và làm dự án.



Hình 1.6 LCD1602

**3.2 THÔNG SỐ KỸ THUẬT:**

- Điện áp hoạt động là 5 V.

- Kích thước: 80 x 36 x 12.5 mm

- Chữ đen, nền xanh lá

- Khoảng cách giữa hai chân kết nối là 0.1 inch tiện dụng khi kết nối với Breadboard.

- Tên các chân được ghi ở mặt sau của màn hình LCD hổ trợ việc kết nối, đi dây điện.

- Có đèn led nền, có thể dùng biến trở hoặc PWM điều chình độ sáng để sử dụng ít điện năng hơn.

- Có thể được điều khiển với 6 dây tín hiệu

**3.3 SƠ ĐỒ CHÂN LCD 16x2:**

**Table

Description automatically generated**

Hình 1.7 Sơ đồ chân LCD 16x2

Trong 16 chân của LCD được chia ra làm 3 dạng tín hiệu như sau:

- Các chân cấp nguồn: Chân số 1 là chân nối mass (0V), chân thứ 2 là Vdd nối với nguồn+5V. Chân thứ 3 dùng để chỉnh contrast thường nối với biến trở.

- Các chân điều khiển: Chân số 4 là chân RS dùng để điều khiển lựa chọn thanh ghi. ChânR/W dùng để điều khiển quá trình đọc và ghi. Chân E là chân cho phép dạng xung chốt.

- Các chân dữ liệu D7÷D0: Chân số 7 đến chân số 14 là 8 chân dùng để trao đổi dữ liệu giữa thiết bị điều khiển và LCD.

**3.4 ĐỊA CHỈ BA VÙNG NHỚ:**

Bộ điều khiển LCD có ba vùng nhớ nội, mỗi vùng có chức năng riêng. Bộ điều khiển phải khởi động trước khi truy cập bất kỳ vùng nhớ nào.

**a. Bộ nhớ DDRAM**

Bộ nhớ chứa dữ liệu để hiển thị (Display Data RAM: DDRAM) lưu trữ những mã ký tự để hiển thị lên màn hình. Mã ký tự lưu trữ trong vùng DDRAM sẽ tham chiếu với từng bitmap kí tự được lưu trữ trong CGROM đã được định nghĩa trước hoặc đặt trong vùng do người sử dụng định nghĩa.

**b. Bộ phát kí tự ROM – CGROM**

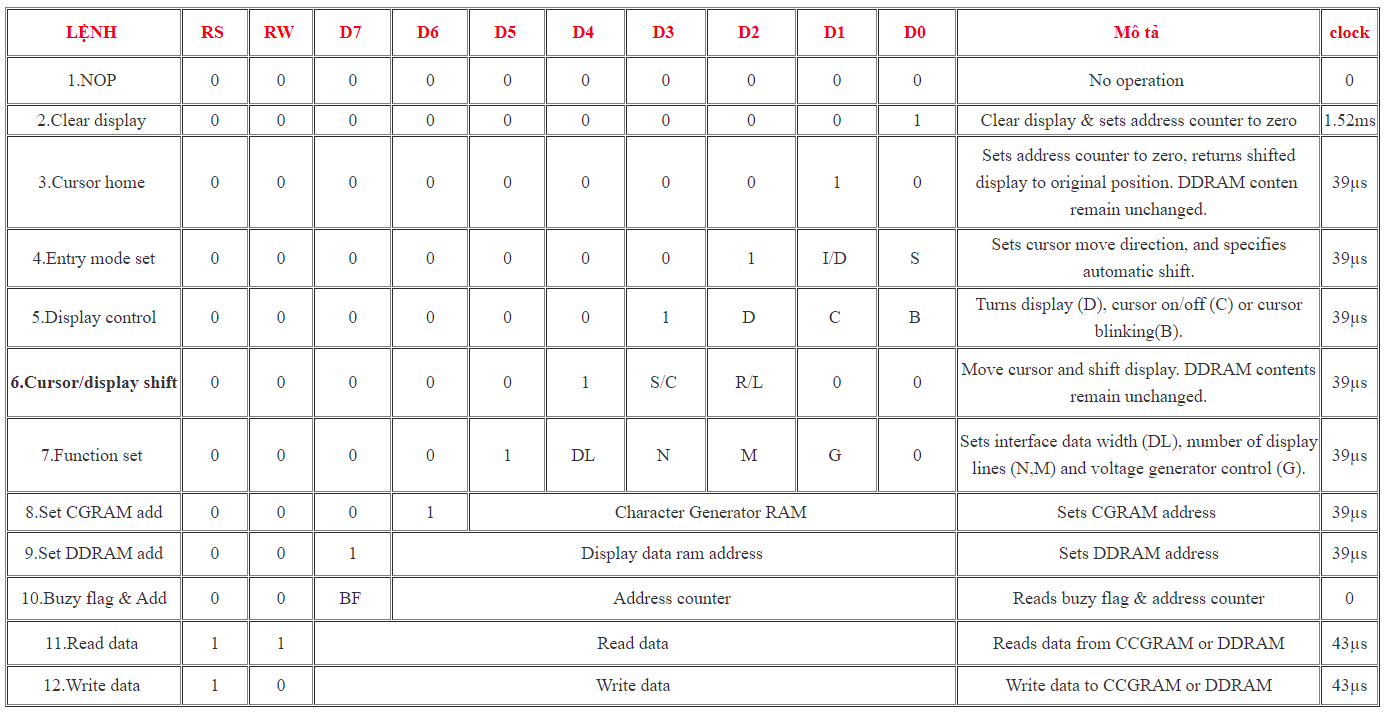
Bộ phát kí tự ROM (Character Generator ROM: CGROM) chứa các kiểu bitmap cho mỗi kí tự được định nghĩa trước mà LCD có thể hiển thị, như được trình bày bảng mã ASCII. Mã kí tự lưu trong DDRAM cho mỗi vùng kí tự sẽ được tham chiếu đến một vị trí trong CGROM.

Ví dụ: mã kí tự số hex 0x53 lưu trong DDRAM được chuyển sang dạng nhị phân 4 bit cao là DB[7:4] = “0101” và 4 bit thấp là DB[3:0] = “0011” chính là kí tự chữ ‘S’ sẽ hiển thị trên màn hình LCD.

**c. Bộ phát kí tự RAM – CGRAM**

Bộ phát kí tự RAM (Character Generator RAM: CG RAM) cung cấp vùng nhớ để tạo ra 8 kí tự tùy ý. Mỗi kí tự gồm 5 cột và 8 hàng.

**3.5 CÁC LỆNH ĐIỀU KHIỂN CỦA LCD:**



Hình 1.8 Bảng giá trị PIC16F877A

- **Lệnh thiết lập chức năng giao tiếp “Function set”:**

+ Bit DL (data length) = 1 thì cho phép giao tiếp 8 đường data D7 ÷ D0, nếu bằng 0 thì cho phép giao tiếp 4 đường D7 ÷ D4.

+ Bit N (number of line) = 1 thì cho phép hiển thị 2 hàng, nếu bằng 0 thì cho phép hiển thị 1 hàng.

+ Bit F (font) = 1 thì cho phép hiển thị với ma trận 5×8, nếu bằng 0 thì cho phép hiển thị với ma trận 5×11.

* Các bit cao còn lại là hằng số không đổi.
* Lenh xóa màn hình **“ clear display ”** khi thực hiện lệnh này thì LCD sẽ bị xóa và bộ đệm chỉ được xóa về 0.
* Lệnh di chuyển con trỏ về đầu màn hình **“Cursor Home”**:khi thực hiện lệnh này thì bộ đếm địa chỉ được xoá về 0, phần hiển thị trở về vị trí gốc đã bị dịch trước đó. Nội dung bộ nhớ RAM hiển thị DDRAM không bị thay đổi.
* Lệnh thiết lập lối vào **“Entry mode set”:** lệnh này dùng để thiết lập lối vào cho các kí tự hiển thị,

+ Bit I/D = 1 thì con trỏ tự động tăng lên 1 mỗi khi có 1 byte dữ liệu ghi vào bộ hiển thị, khi I/D = 0 thì con trỏ sẽ tự động giảm đi 1 mỗi khi có 1 byte dữ liệu ghi vào bộ hiển thị.

+ Bit S = 1 thì cho phép dịch chuyển dữ liệu mỗi khi nhận 1 byte hiển thị.

* **Lệnh điều khiển con trỏ hiện thị “display control”:**

+ Bit D: cho phép LCD hiển thị thì D = 1, không cho hiển thị thì bit D = 0.

+ Bit C: cho phép con trỏ hiển thị thì C= 1, không cho hiển thị con trỏ thì bit C = 0.

+ Bit B: cho phép con trỏ nhấp nháy thì B= 1, không cho con trỏ nhấp nháy thì bit B = 0.

+ Với các bit như trên thì để hiển thị phải cho D = 1, 2 bit còn lại thì tùy chọn, trong thư viện thì cho 2 bit đều bằng 0, không cho phép mở con trỏ và nhấp nháy, nếu bạn không thích thì hiệu chỉnh lại.

* **Lệnh di chuyển con trỏ “Cursor /Display Shift”:**

+ Lệnh này dùng để điều khiển di chuyển con trỏ hiển thị dịch chuyển

+ Bit SC: SC = 1 cho phép dịch chuyển, SC = 0 thì không cho phép.

+ Bit RL xác định hướng dịch chuyển: RL = 1 thì dịch phải, RL = 0 thì dịch trái. Nội dung bộ nhớ DDRAM vẫn không đổi.

+ Vậy khi cho phép dịch thì có 2 tùy chọn: dịch trái và dịch phải.

+ Lệnh thiết lập địa chỉ cho bộ nhớ RAM phát kí tự “Set CGRAM Addr”: lệnh này dùng để thiết lập địa chỉ cho bộ nhớ RAM phát kí tự.

+ Lệnh thiết lập địa chỉ cho bộ nhớ RAM hiển thị “Set DDRAM Addr”: lệnh này dùng để thiết lập địa chỉ cho bộ nhớ RAM lưu trữ các dữ liệu hiển thị.

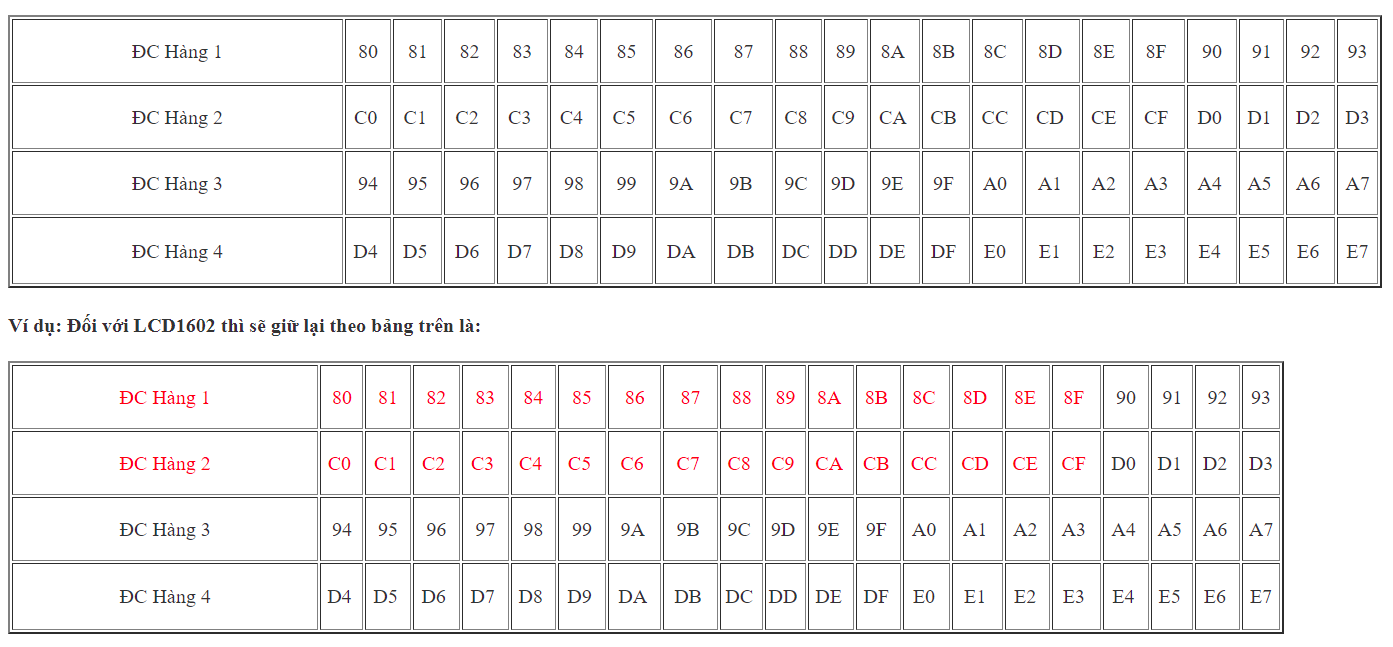
+ Hai lệnh cuối cùng là lệnh đọc và lệnh ghi dữ liệu LCD.

**3.6 BẢNG MÃ ASCII SỬ DỤNG CHO LCD:**



Hình 1.9 Bảng mã ASCII hiển thị ký tự cho LCD1602

**3.7 BẢNG ĐỊA CHỈ CHO LCD:**



Hình 2.1 LCD1602 địa chỉ Pic16F877A

**4. ĐỘNG CƠ SERVO SG90**

**4.1 GIỚI THIỆU:**

[Động cơ servo SG90](https://nshopvn.com/product/dong-co-servo-sg90-tower-pro/" \t "https://nshopvn.com/product/dong-co-servo-sg90-tower-pro/_blank) có kích thước nhỏ, là loại được sử dụng nhiều nhất để làm các mô hình nhỏ hoặc các cơ cấu kéo không cần đến lực nặng.

Động cơ servo SG90 180 độ có tốc độ phản ứng nhanh, các bánh răng được làm bằng nhựa nên cần lưu ý khi nâng tải nặng vì có thể làm hư bánh răng, động cơ RC Servo 9G có tích hợp sẵn Driver điều khiển động cơ bên trong nên có thể dễ dàng điều khiển góc quay bằng phương pháp điều độ rộng xung PWM.

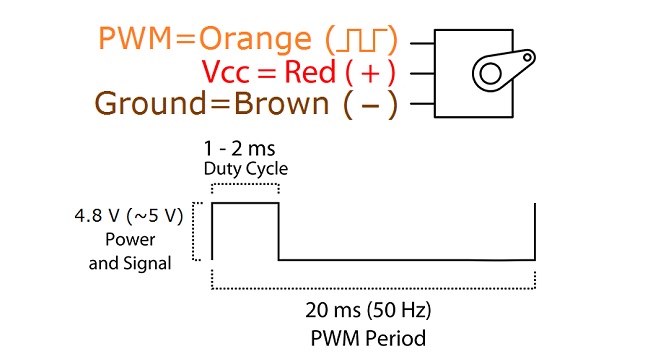


Hình 2.2 Động cơ servo sg90

**4.2. THÔNG SỐ KỸ THUẬT:**

* -Điện áp hoạt động: 4.8-5VDC
* Tốc độ: 0.12 sec/ 60 deg (4.8VDC)
* Lực kéo: 1.6 Kg.cm
* Kích thước: 21x12x22mm
* Trọng lượng: 9g.

**4.3. DẠNG SÓNG:**



Hình 2.3Dạng sóng động cơ servo sg90

**4.4. SERVO GIAO TIẾP VỚI PIC**

- Kết nối dây màu đỏ với 5V, dây màu nâu với mass, dây màu cam với chân phát xung của vi điều khiển. Ở chân xung cấp một xung từ 1ms-2ms theo để điều khiển góc quay theo ý muốn.

**Chương 2: THIẾT KẾ VÀ CÀI ĐẶT**

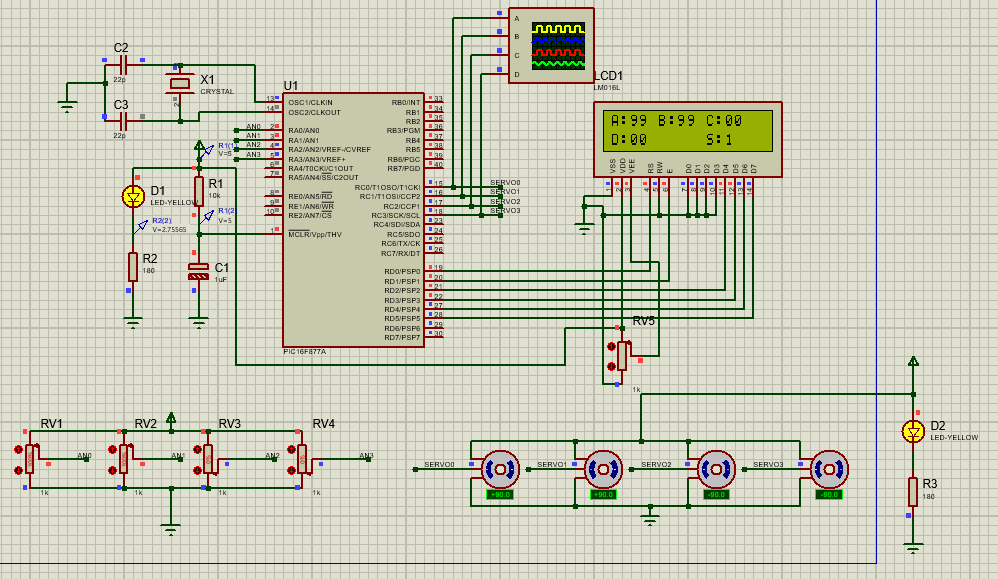
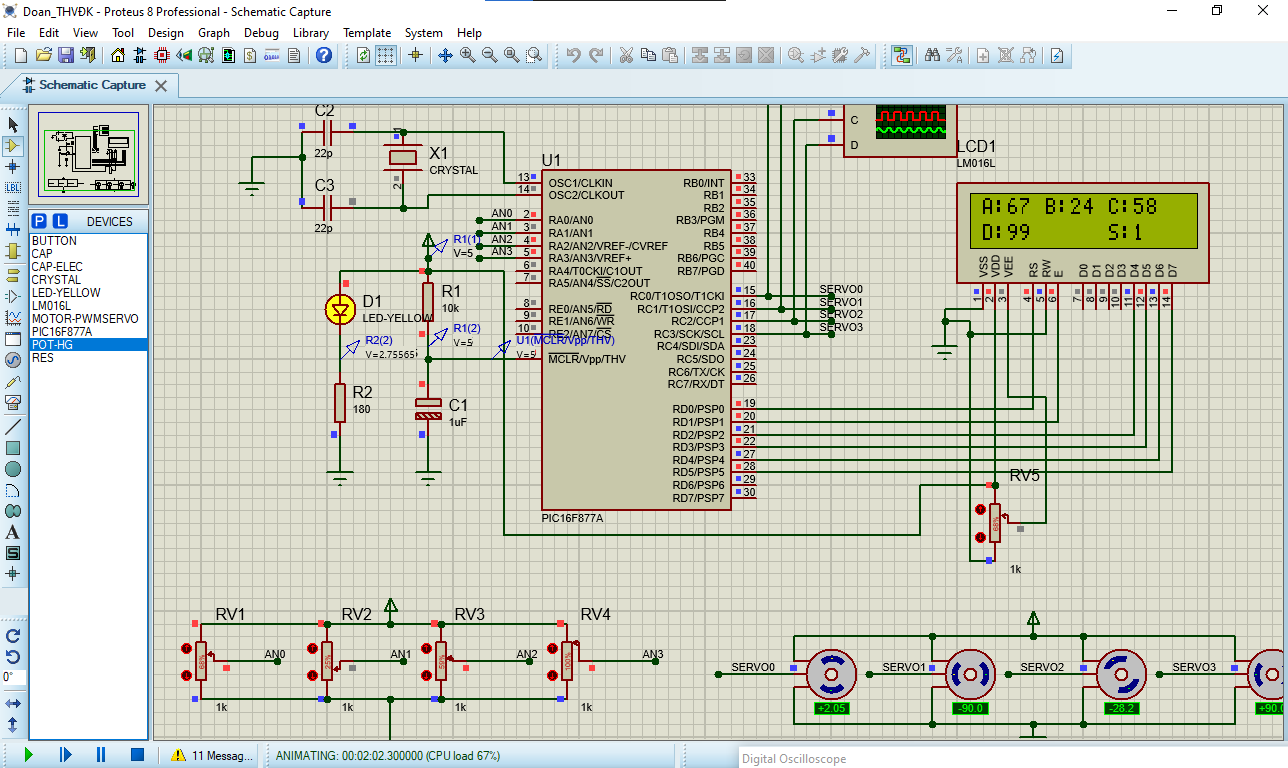
**2.1 Chuẩn bị:**

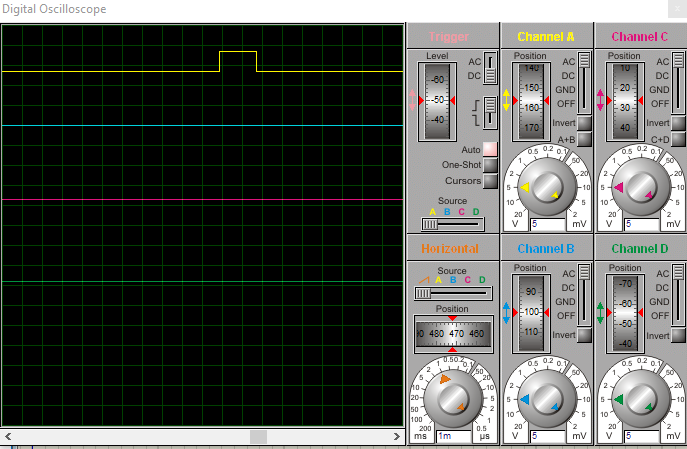
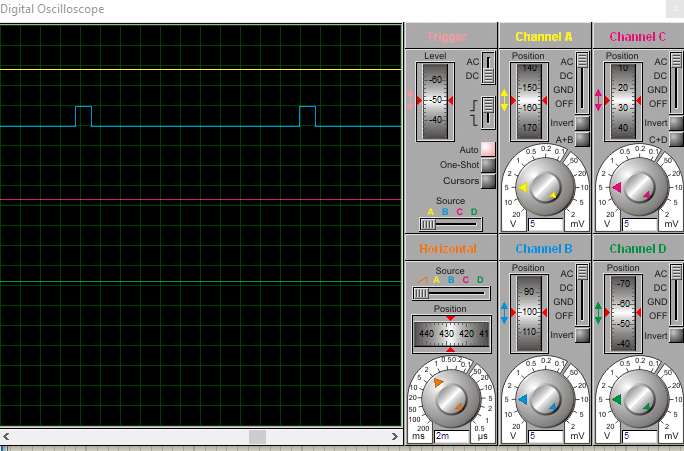
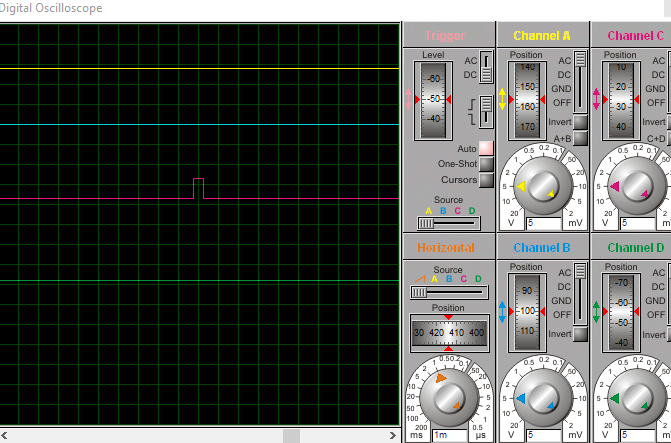
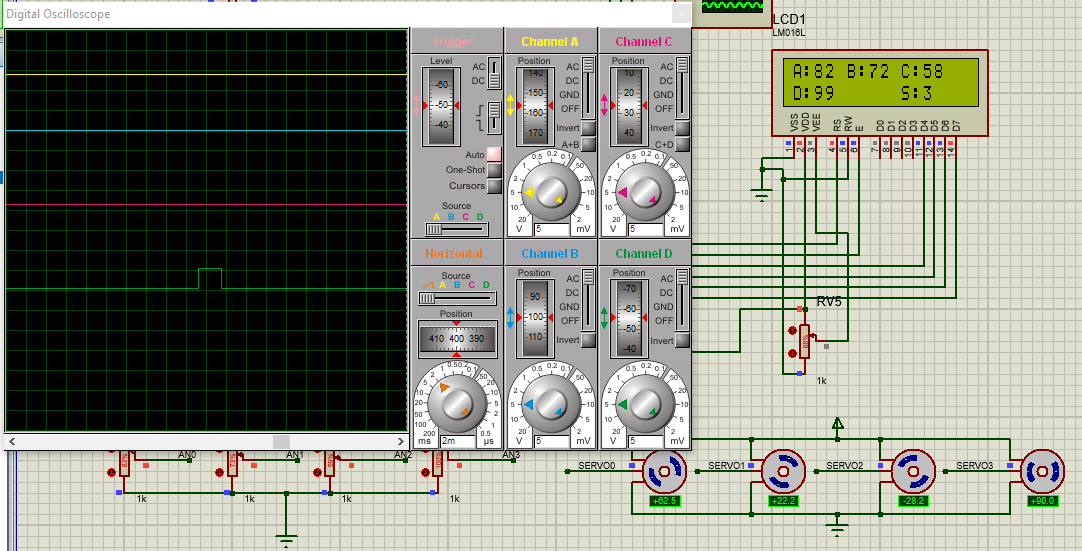
PIC16F877A , Khung cánh tay robot , LCD 1602 , Động cơ servo sg90 , Biến trở , Tụ gốm , Tụ hóa , LED xanh lá , Trở

**2.2 Thiết kế và thực hiện phần mềm :**

**a. Thiết kế sơ đồ nguyên lý**

**b. Mô phỏng trên phần mềm proteus**

****



Hình 2.4 Mô phỏng Protues

**c. Lập trình code trên Mplab**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

Hình 2.5 Code trên Mplab

**+ Code thư viện lcd.h:** Vì Mplab không có sẵn thư viện cho Lcd nên chúng em đã tự viết thêm vào

void Lcd\_SetBit(char data\_bit) //Based on the Hex value Set the Bits of the Data Lines

{

    if(data\_bit& 1)

        D4 = 1;

    else

        D4 = 0;

    if(data\_bit& 2)

        D5 = 1;

    else

        D5 = 0;

    if(data\_bit& 4)

        D6 = 1;

    else

        D6 = 0;

    if(data\_bit& 8)

        D7 = 1;

    else

        D7 = 0;

}

void Lcd\_Cmd(char a)

{

    RS = 0;

    Lcd\_SetBit(a); //Incoming Hex value

    EN  = 1;

        \_\_delay\_ms(4);

        EN  = 0;

}

Lcd\_Clear()

{

    Lcd\_Cmd(0); //Clear the LCD

    Lcd\_Cmd(1); //Move the curser to first position

}

void Lcd\_Set\_Cursor(char a, char b)

{

    char temp,z,y;

    if(a== 1)

    {

      temp = 0x80 + b - 1; //80H is used to move the curser

        z = temp>>4; //Lower 8-bits

        y = temp & 0x0F; //Upper 8-bits

        Lcd\_Cmd(z); //Set Row

        Lcd\_Cmd(y); //Set Column

    }

    else if(a== 2)

    {

        temp = 0xC0 + b - 1;

        z = temp>>4; //Lower 8-bits

        y = temp & 0x0F; //Upper 8-bits

        Lcd\_Cmd(z); //Set Row

        Lcd\_Cmd(y); //Set Column

    }

}

void Lcd\_Start()

{

  Lcd\_SetBit(0x00);

  for(int i=1065244; i<=0; i--)  NOP();

  Lcd\_Cmd(0x03);

    \_\_delay\_ms(5);

  Lcd\_Cmd(0x03);

    \_\_delay\_ms(11);

  Lcd\_Cmd(0x03);

  Lcd\_Cmd(0x02); //02H is used for Return home -> Clears the RAM and initializes the LCD

  Lcd\_Cmd(0x02); //02H is used for Return home -> Clears the RAM and initializes the LCD

  Lcd\_Cmd(0x08); //Select Row 1

  Lcd\_Cmd(0x00); //Clear Row 1 Display

  Lcd\_Cmd(0x0C); //Select Row 2

  Lcd\_Cmd(0x00); //Clear Row 2 Display

  Lcd\_Cmd(0x06);

}

void Lcd\_Print\_Char(char data)  //Send 8-bits through 4-bit mode

{

   char Lower\_Nibble,Upper\_Nibble;

   Lower\_Nibble = data&0x0F;

   Upper\_Nibble = data&0xF0;

   RS = 1;             // => RS = 1

   Lcd\_SetBit(Upper\_Nibble>>4);             //Send upper half by shifting by 4

   EN = 1;

   for(int i=2130483; i<=0; i--)  NOP();

   EN = 0;

   Lcd\_SetBit(Lower\_Nibble); //Send Lower half

   EN = 1;

   for(int i=2130483; i<=0; i--)  NOP();

   EN = 0;

}

void Lcd\_Print\_String(char \*a)

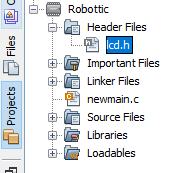
{

    int i;

    for(i=0;a[i]!='\0';i++)

       Lcd\_Print\_Char(a[i]);  //Split the string using pointers and call the Char function

}

Thêm thư viện vào Header Files

**+ Code chương trình chính:**

#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)

#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = ON       // Power-up Timer Enable bit (PWRT enabled)

#pragma config BOREN = ON       // Brown-out Reset Enable bit (BOR enabled)

#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial

Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)

#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)

#define RS RD0

#define EN RD1

#define D4 RD2

#define D5 RD3

#define D6 RD4

#define D7 RD5

#define \_XTAL\_FREQ 20000000 //Crystak Freq is 20MHz

#define PWM\_Frequency 0.05 // in KHz (50Hz)

#include <xc.h>

#include "lcd.h"

int POT\_val; //variable to store value from ADC

int count; //timer variable

// T\_TOTAL = (1/PWM\_Frequency); //calculate Total Time from frequency (in milli sec)) //20msec

int Duty\_cycle;

int T\_ON[4];

char servo;

void ADC\_Initialize() {

  ADCON0 = 0b01000001;

  ADCON1 = 0b11000000;

}

unsigned int ADC\_Read(unsigned char channel) {

  ADCON0 &= 0x11000101;

  ADCON0 |= channel<<3;

  \_\_delay\_ms(2);

  GO\_nDONE = 1;

  while(GO\_nDONE);

  return ((ADRESH<<8) + ADRESL);

}

void \_\_interrupt () timer\_isr() {

    // Overflow for every 0.05ms

    if(TMR0IF == 1){

        TMR0 = 248;

        TMR0IF = 0;

        count++;

    }

    int servo\_code[] = {0b00000001, 0b00000010, 0b00000100, 0b00001000};

    if (count >= 400)

        count = 0;

     if (count <= (T\_ON[servo]) )

         PORTC = PORTC | servo\_code[servo];

     else

         PORTC = PORTC & ~(servo\_code[servo]);

}

void main() {

    /\*\*\*\*\*Port Configuration for Timer \*\*\*\*\*\*/

    OPTION\_REG = 0b00000100;  // Timer0 with external freq and 32 as prescalar

    TMR0 = 248;       // Load the time value for 0.0001s; delayValue can be between 0-256 only

    TMR0IE = 1;       //Enable timer interrupt bit in PIE1 register

    GIE = 1;          //Enable Global Interrupt

    PEIE = 1;         //Enable the Peripheral Interrupt

    /\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*/

    /\*\*\*\*\*Port Configuration for I/O \*\*\*\*\*\*/

    TRISC = 0x00;

    PORTC = 0x00;

    TRISD = 0x00;

    PORTD = 0x00;

    /\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*/

    Lcd\_Start(); //Initialize LCD module

    ADC\_Initialize();

    while(1) {

        for (int pot\_num = 0; pot\_num <= 3; pot\_num++) {

            int prev\_ton = T\_ON[pot\_num];

            POT\_val = (ADC\_Read(pot\_num));

            Duty\_cycle = (POT\_val \* 0.0976); //Map 0 to 1024 to 0 to 100

            T\_ON[pot\_num] = Duty\_cycle \* 0.4;

            if (T\_ON[pot\_num] != prev\_ton)

                servo = pot\_num;

                Lcd\_Set\_Cursor(2,11); Lcd\_Print\_String("S:");Lcd\_Print\_Char(servo+'0');

        if (pot\_num==0)

        {Lcd\_Set\_Cursor(1,1); Lcd\_Print\_String("A:");}

        else if (pot\_num==1)

        {Lcd\_Set\_Cursor(1,6); Lcd\_Print\_String("B:");}

        else if (pot\_num==2)

        {Lcd\_Set\_Cursor(1,11); Lcd\_Print\_String("C:");}

        else if (pot\_num==3)

        {Lcd\_Set\_Cursor(2,1); Lcd\_Print\_String("D:");}

        else if (pot\_num==4)

        {Lcd\_Set\_Cursor(2,6); Lcd\_Print\_String("E:");}

        char d2 = (Duty\_cycle) %10;

        char d1 = (Duty\_cycle/10) %10;

        Lcd\_Print\_Char(d1+'0');Lcd\_Print\_Char(d2+'0');

        }

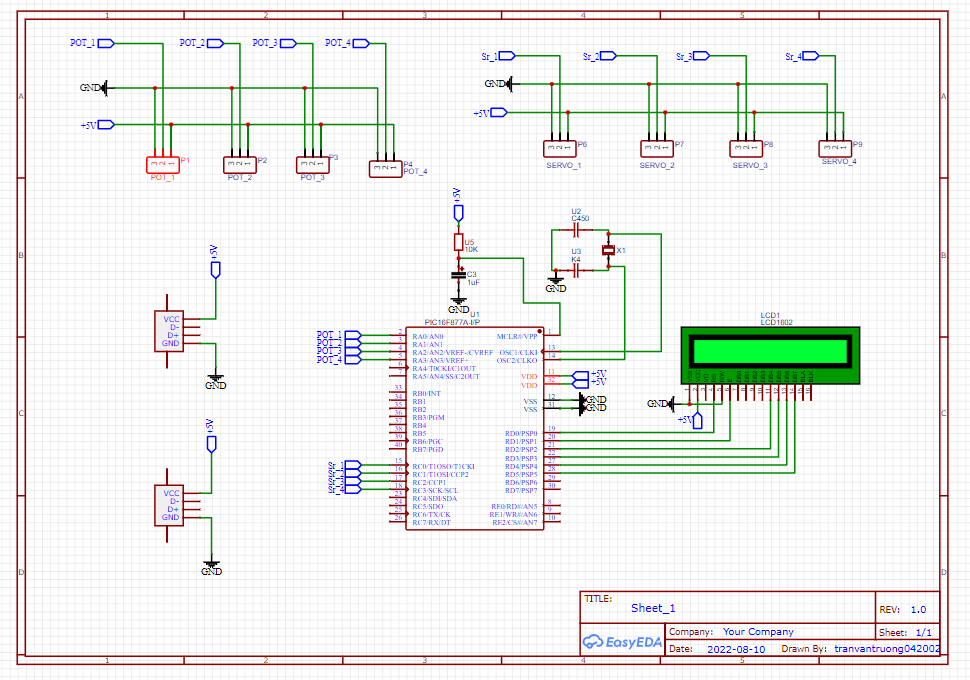
    }

}

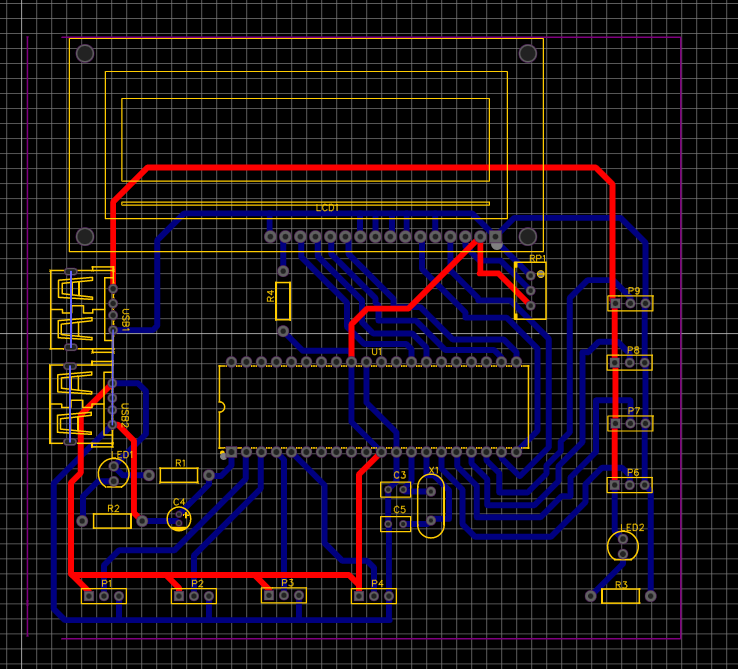
**2.3 Thiết kế và thực hiện phần cứng :**

**- Vẽ sơ đồ mạch trên websie easyeda.com:**

+ Thiết kế schemactic

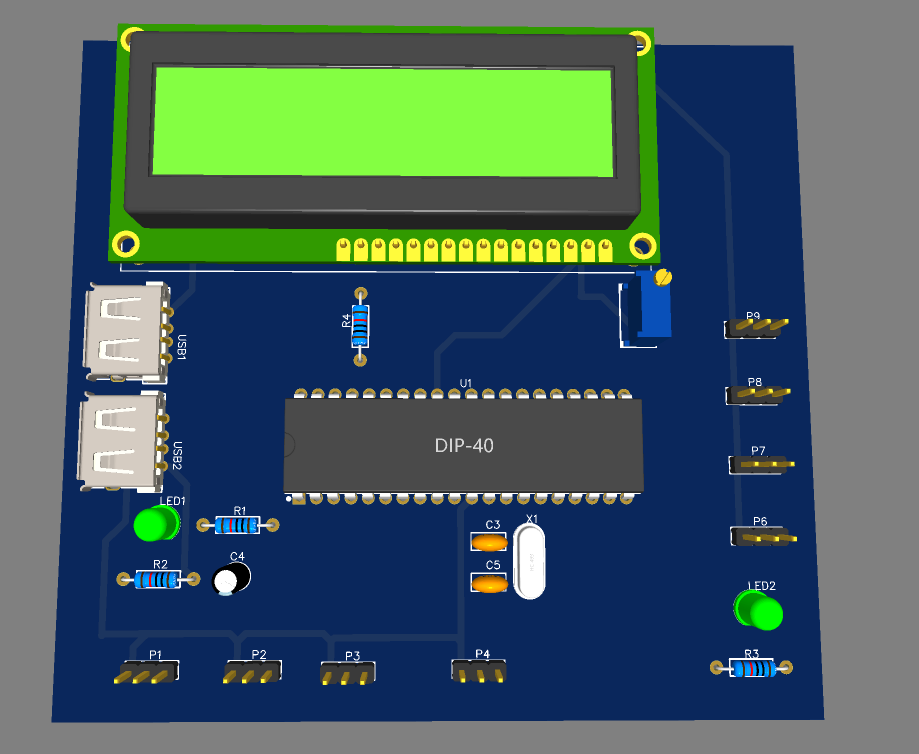
Hình 2.6 Mạch thiết kế trên easyeda

+ Mạch PCB:



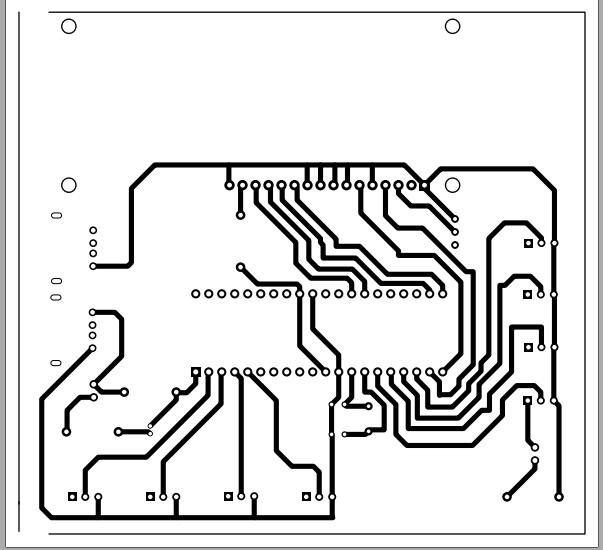
Hình 2.7 Mạch in PCB

+Mô phỏng mạch 3D



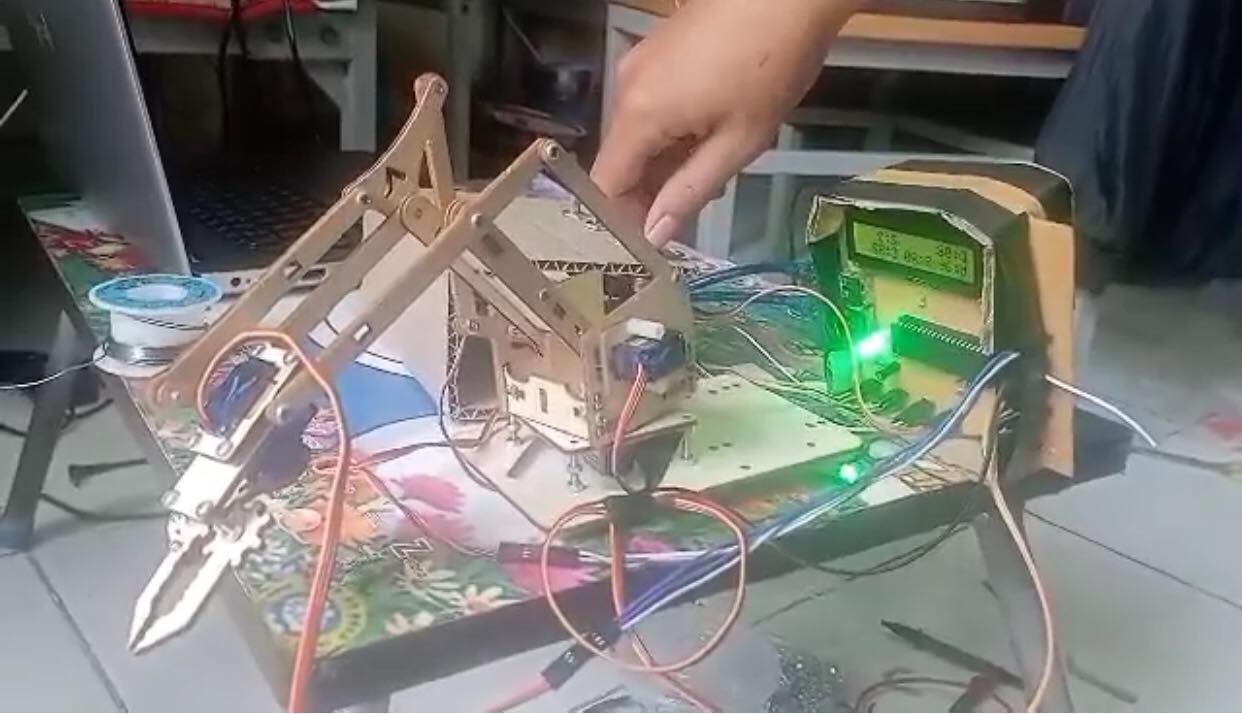
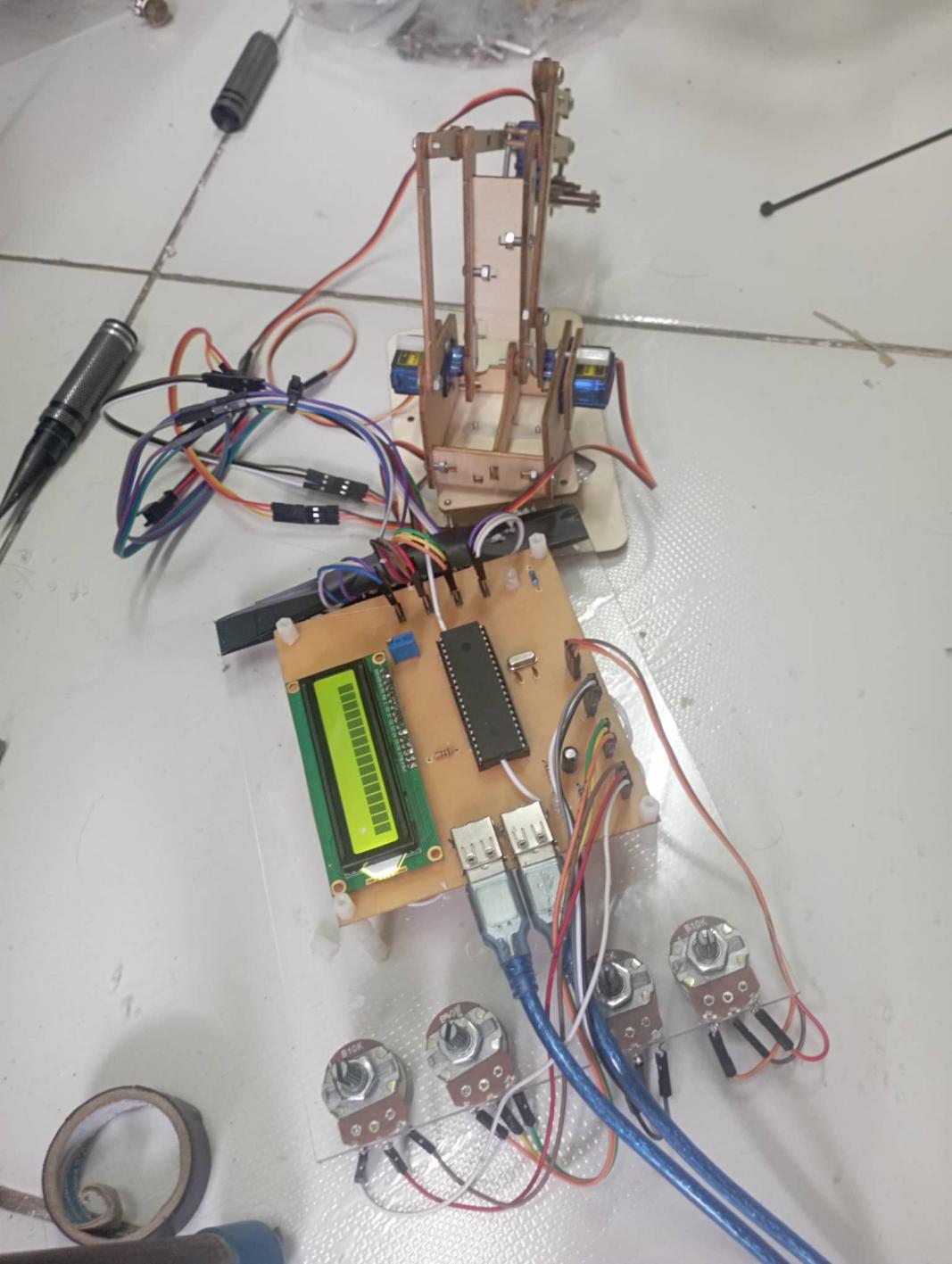
Hình 2.8 Mạch mô phỏng 3D

+ Xuất PDF

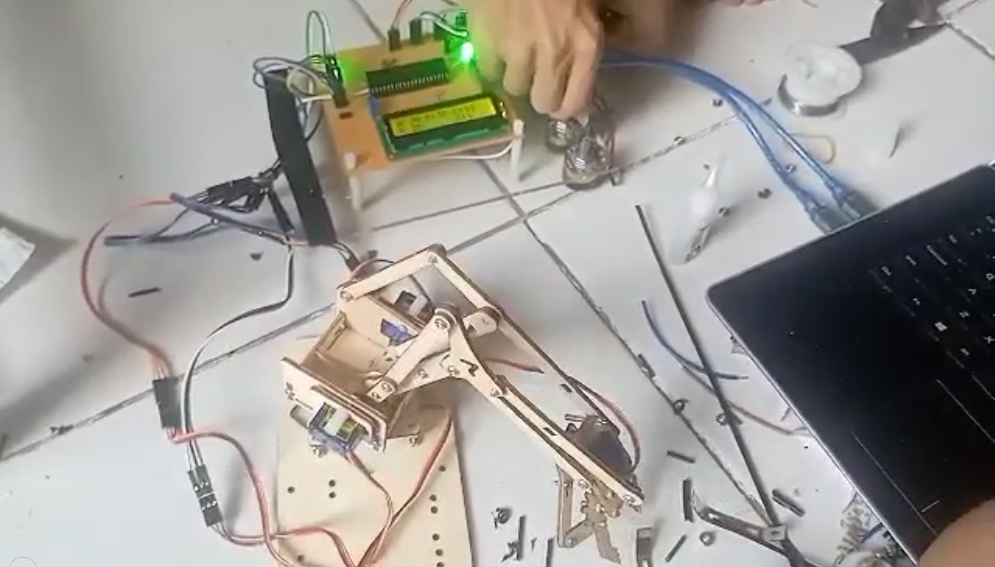
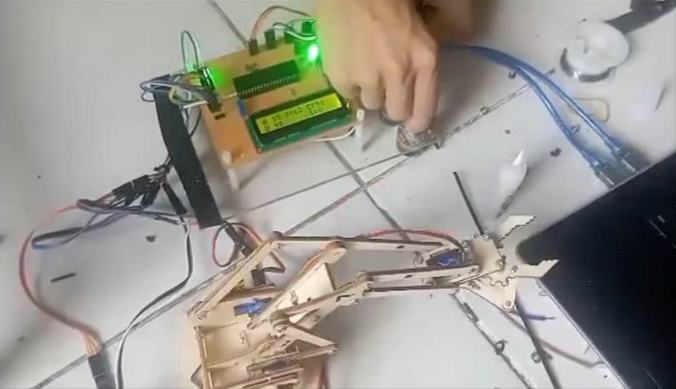


Hình 2.9 Mạch in PDF

Làm mạch thực tế

Nâng cấp và hoàn thiện mạch

**2.4 Kết quả thực hiện:**

****

Hình 3.1 Mạch thực tế

**Video demo, điều khiển 4 cấp:**

<https://youtu.be/v-_PKSSI-SU>

**Video demo, gắp đồ vật:**

<https://youtu.be/_3EYKAntifc>

**TÀI LIỆU THAM KHẢO**

-<https://advancecad.edu.vn/dieu-khien-canh-tay-robot-bang-vi-dieu-khien-pic/?fbclid=IwAR3an6c-WzcaqYH8r_BdsJsNTjHFAlPrvcjMOXJv-HhIfy274P5N4I12OuE>

-<https://kienthuctudonghoa.com/dieu-khien-servo-bang-pwm-arduino/?fbclid=IwAR1gZEoBE-iuoL0EX2zoGCRF9gJq1gQlzMTMMCk-A9OlApV8-JfMCHDqywA>

-<https://www.academia.edu/35289814/Canh_Tay_robot_arm_robot_>

**BÁO CÁO KỸ NĂNG**

**1. Kỹ năng làm việc nhóm và quản lý dự án:**

|  |  |  |
| --- | --- | --- |
| **Nhiệm vụ** | **Giao cho** | **Tình trạng** |
| Lên ý tưởng | Cả nhóm | Hoàn thành |
| Họp mặt | Cả nhóm | Hoàn thành |
| Tài chính | Phan Trọng Khôi | Hoàn thành |
| Yêu cầu phần mềm | Minh Phúc, Ngọc Mẫn | Hoàn thành |
| Yêu cầu phần cứng | Quốc Khôi, Văn Trường, Thành Phúc | Hoàn thành |
| Nhân sự | Cả nhóm | Hoàn thành |
| Thuyết trình | Duy Khánh | Hoàn Thành |
| Báo cáo | Lê Tuấn Kiệt | Hoàn thành |
| Slides thuyết trình | Nguyễn Tuấn Kiệt | Hoàn thành |
| Chạy thử 1 | Thành Phúc | Hoàn thành |
| Chạy thử 2 | Thành Phúc | Hoàn thành |
| Chạy thử lần cuối | Thành Phúc | Hoàn thành |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bảng đánh giá từng thành viên** | | | | | |
| **Tên** | **Mức độ tham gia** | **Đóng góp** | **Đánh giá** | **Hoàn thành công việc** |
| Nguyễn Duy Khánh |  |  |  | Hoàn thành |
| Lê Minh Phúc |  |  |  | Hoàn thành |
| Nguyễn Trần Quốc Khôi |  |  |  | Hoàn thành |
| Phan Trọng Khôi |  |  |  | Hoàn thành |
| Lê Tuấn Kiệt |  |  |  | Hoàn thành |
| Nguyễn Tuấn Kiệt |  |  |  | Hoàn thành |
| Nguyễn Ngọc Mẫn |  |  |  | Hoàn thành |
| Võ Thành Phúc |  |  |  | Hoàn thành |
| Trần Văn Trường |  |  |  | Hoàn thành |

**2. Tự đánh giá các kỹ năng/kiến thức:**

Nhóm tự đánh giá tổng hợp các kỹ năng, kiến thức đạt được qua đồ án môn học

|  |  |  |
| --- | --- | --- |
| **STT** | **Các kỹ năng** | **Đánh giá(\*)** |
| 1 | Kỹ năng làm việc nhóm, giao tiếp, hành xử chuyên  nghiệp, khả năng lãnh đạo và làm việc độc lập |  |
| 2 | Kỹ năng tư duy phản biện |  |
| 3 | Kỹ năng thuyêt trình |  |
| 4 | Giao tiếp kỹ thuật(viết báo cáo kỹ thuật) |  |
| 5 | Kỹ năng tư duy sáng tạo |  |
| 6 | Kỹ năng quản lý dự án/thời gian thực hiện dự án |  |
| 7 | Hình thành nội dung, xác định vấn đề và kỹ năng giải quyết vấn đề |  |
| 8 | Kiến thức, thực nghiệm qua đồ án môn học |  |

**(\*)Ghi chú:** Đánh giá theo mức A/B/C/D (A: Rất tốt, B: Tốt, C: Trung bình, D: Chưa tốt)