TRƯỜNG ĐẠI HỌC PHENIKAA

**KHOA ĐIỆN – ĐIỆN TỬ**



**BÁO CÁO**

**KẾT THÚC HỌC PHẦN**

**ĐỀ TÀI: XÂY DỰNG VÀ LẬP TRÌNH HỆ THỐNG ĐO KHOẢNG CÁCH SỬ DỤNG CẢM BIẾN HC-SR04**

**Tên học phần: Kỹ thuật vi xử lý và vi điều khiển**

**Học kỳ: II – Năm học: 2024 – 2025**

Sinh viên thực hiện: **Hoàng Trung An**

**Nguyễn Hùng Dương**

Mã sinh viên: **22010740 - 22014374**

Lớp: **EEE703046-1-2-24(N01)**

GVHD: **ThS. Nguyễn Ngọc Thanh**

Khoa: **Điện – Điện tử**

**HÀ NỘI, NĂM 2025**

**PHIẾU ĐÁNH GIÁ**

|  |  |  |
| --- | --- | --- |
| Điểm số | Điểm chữ | Cán bộ chấm 1: |
| Cán bộ chấm 2: |

**Rubric chấm điểm:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tiêu chí đánh giá** | **Điểm**  **0-3.9** | **Điểm**  **4.0-5.4** | **Điểm**  **5.5-6.9** | **Điểm**  **7.0-8.4** | **Điểm**  **8.5-10** | **Trọng số (%)** |
| ***Xác định các vấn đề cần giải quyết*** | Không xác định được yêu cầu bài toán | Xác định được yêu cầu bài toán | Xác định và phân tích được được yêu cầu bài toán | Xác định và phân tích được được yêu cầu bài toán. Phân tích khả năng giải quyết bài toán bằng vi đều khiển | Xác định và phân tích được được toàn bộ yêu cầu bài toán. Phân tích khả năng giải quyết bài toán bằng vi đều khiển kèm theo các ràng buộc liên quan | 10 |
| ***Thiết kế mạch nguyên lý và PCB*** | Không thiết kế được sơ đồ nguyên lý và PCB layout | Thiết kế được sơ đồ nguyên lý, không thiết kế dược PCB layout | Thiết kế sơ đồ nguyên lý và PCB layout ở mức độ cơ bản | Thiết kế sơ đồ nguyên lý và có tính toán tham số các linh kiện; PCB layout thiết kế gọn gàng, đường mạch hợp lý | Thiết kế sơ đồ nguyên lý và tính toán đúng, đầy đủ tham số các linh kiện; PCB layout thiết kế gọn gàng, đường mạch hợp lý, bố trí linh kiện khoa học | 30 |
| ***Chất lượng sản phẩm*** | Sản phẩm không đáp ứng hoặc chỉ đáp ứng một phần nhỏ các yêu cầu hoặc mục tiêu của dự án. (Chỉ có mạch mô phỏng hoặc giải thuật lập trình) | Sản phẩm không đáp ứng đầy đủ các yêu cầu hoặc mục tiêu của dự án, có ít giá trị thực tế. (Có mạch mô phỏng, giải thuật, và code lập trình, mô hình chạy được) | Sản phẩm đáp ứng một số mục tiêu và yêu cầu của dự án. (Có mạch mô phỏng, giải thuật, code lập trình, mạch cắm breadboard, các mô hình mô phỏng và thử nghiệm đều chạy được) | Sản phẩm đáp ứng các mục tiêu và yêu cầu của dự án một cách đầy đủ và hiệu quả. (Có mạch mô phỏng, giải thuật, code lập trình, làm mạch in thủ công, các mô hình mô phỏng và thử nghiệm đều chạy được) | Sản phẩm đạt được tất cả các mục tiêu và yêu cầu của dự án một cách xuất sắc và có tính độc đáo, sáng tạo. (Có mạch mô phỏng, giải thuật, code lập trình, làm mạch in gọn đẹp, các mô hình mô phỏng và thử nghiệm đều chạy được) | 40 |
| ***Các ứng dụng, hạn chế và hướng phát triển*** | Không nêu được ứng dụng, hạn chế và hướng phát triển | Nêu được 1 trong 3 nội dung: ứng dụng, hạn chế và phướng phát triển của đề tài | Nêu được các ứng dụng, hạn chế và/hoặc hướng phát triển của đề tài | Phân tích đầy đủ ứng dụng, các hạn chế hoặc hướng phát triển của đề tài | Phân tích đầy đủ ứng dụng, các hạn chế và hướng phát triển của đề tài | 10 |
| ***Hoạt động nhóm*** | Không làm việc nhóm hoặc có thành viên không nắm được công việc của mình | Có kế hoạch phân chia công việc cụ thể cho các thành viên | Các thành viên tham gia thực hiện công việc của mình theo phân công | Các thành viên tham gia thực và thường xuyên trao đổi với các thành viên trong nhóm | Các thành viên tham gia và thường xuyên trao đổi, hiểu rõ công việc và nội dung của các thành viên khác trong nhóm | 10 |

**PHÂN CÔNG NHÓM**

**I) Thành viên nhóm**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và Tên** | **Thông tin cá nhân** |
| 1 | Hoàng Trung An | - Mã SV: 22010740  - Lớp: EEE703046-1-2-24(N01)  - Trường Đại học Phenikaa  - SĐT: 0359616178  - Nơi ở: Hà Nội |
| 2 | Nguyễn Hùng Dương | - Mã SV: 22014374  - Lớp: EEE703046-1-2-24(N01)  - Trường Đại học Phenikaa  - SĐT: 0983161645  - Nơi ở: Hà Nội |

**II) Phân công nhiệm vụ**

|  |  |  |
| --- | --- | --- |
| **Thành viên** | **Nhiệm vụ được giao** | **Nhóm đánh giá** |
| Hoàng Trung An | Xây dựng sơ đồ nguyên lý, vẽ pcb, lập trình | **Đạt** |
| Nguyễn Hùng Dương | Mô phỏng sơ đồ nguyên lý trên Proteus, hàn mạch, lập trình | **Đạt** |

**MỤC LỤC**

[MỞ ĐẦU 6](#_Toc170290239)

[CHƯƠNG 1: TỔNG QUAN VỀ VI XỬ LÝ VI ĐIỀU KHIỂN 7](#_Toc170290240)

[1.1. Khái niệm 7](#_Toc170290241)

[1.2. Phân loại 7](#_Toc170290242)

[1.3. Cấu trúc Vi điều khiển 7](#_Toc170290243)

[1.4. Nguyên tắc hoạt động của Vi điều khiển 7](#_Toc170290244)

[CHƯƠNG 2: THIẾT KẾ HỆ THỐNG 8](#_Toc170290245)

[2.1. Mô tả hệ thống 8](#_Toc170290246)

[2.2. Thiết kế sơ đồ khối 8](#_Toc170290247)

[2.3. Thiết kế sơ đồ nguyên lý 8](#_Toc170290248)

[2.4. Thiết kế layout và chế tạo PCB 8](#_Toc170290249)

[2.5. Lập trình thuật toán 8](#_Toc170290250)

[CHƯƠNG 3: KẾT QUẢ THỰC HIỆN VÀ KẾT LUẬN 9](#_Toc170290251)

[3.1. Kết quả thực hiện 9](#_Toc170290252)

[3.1.1. Lập trình hệ thống đo khoảng cách 9](#_Toc170290253)

[3.1.2. Kết quả mô phỏng (nếu có) 9](#_Toc170290254)

[3.1.3. Kết quả thực nghiệm 9](#_Toc170290255)

[3.2. Kết luận và hướng phát triển 9](#_Toc170290256)

[TÀI LIỆU THAM KHẢO 10](#_Toc170290257)

MỤC LỤC HÌNH VẼ

[Hình 1.1: Hình dạng thực tế của PIC16F887. 9](#_Toc193831037)

[Hình 2.1: Sơ đồ khối hệ thống 16](#_Toc193831038)

[Hình 2.2: Sơ đồ nguyên lý mạch nguồn 17](#_Toc193831039)

[Hình 2.3: Sơ đồ khối MCU 19](#_Toc193831040)

[Hình 2.4: Sơ đồ khối HEADER 19](#_Toc193831041)

[Hình 2.5: Sơ đồ nguyên lý 20](#_Toc193831042)

[Hình 2.7: Mạch dán PCB được thiết kế bằng ứng dụng Kicad 22](#_Toc193831043)

[Hình 2.8: Sơ đồ mạch in 23](#_Toc193831044)

[Hình 2.9: Sơ đồ khối ngoại vi I2C chế độ Master của PIC16F887.[4] 23](#_Toc193831045)

[Hình 2.10: Thanh ghi SSPCON cấu hình I2C.[4] 25](#_Toc193831046)

[Hình 2.11: Thanh ghi SSPCON2 cấu hình I2C.[4] 26](#_Toc193831047)

[Hình 2.12: Thanh ghi SSPSTAT trạng thái I2C.[4] 27](#_Toc193831048)

[Hình 2.13: Lưu đồ thuật toán gửi 1 byte vào bus I2C. 28](#_Toc193831049)

[Hình 2.14: Quy trình gửi data từ Master cho Slave với địa chỉ 0x4E qua bus I2C. 30](#_Toc193831050)

[Hình 2.15: Lưu đồ thuật toán gửi 1 kí tự vào LCD. 31](#_Toc193831051)

[Hình 2.16: Dữ liệu truyền cho PCF8574AT điều khiển LCD. 32](#_Toc193831052)

[Hình 2.17: Kết nối module PCF8574T 33](#_Toc193831053)

[Hình 2.18 Sơ đồ kết nối HCSR04 33](#_Toc193831054)

[Hình 2.19. Minh họa một thanh ghi thay đổi 8 bit với các dòng đầu vào và đầu ra nối tiếp 38](#_Toc193831055)

[Hình 3.1. Sơ đồ mạch mô phỏng 45](#_Toc193831056)

[Hình 3.1.2 Kết quả đo được từ 2 chân Trig và Echo của cảm biến HCSR04 45](#_Toc193831057)

[Hình 3.1.3 Kết quả được hiện thị thông qua UART 45](#_Toc193831058)

[Hình 3.1.4.Sản phầm hoàn thiện 46](#_Toc193831059)

# MỞ ĐẦU

Giới thiệu: Vi xử lý và vi điều khiển đóng vai trò quan trọng trong các hệ thống nhúng hiện đại. Trong lĩnh vực đo lường, cảm biến siêu âm HC-SR04 là một giải pháp phổ biến để đo khoảng cách với chi phí thấp và độ chính xác tương đối cao.

Lí do chọn đề tài: Hệ thống đo khoảng cách có nhiều ứng dụng thực tiễn như robot tránh vật cản, hệ thống đo mực nước, và các ứng dụng tự động hóa. Để cải thiện độ chính xác của phép đo, thuật toán Low Pass Filter (LPF) được áp dụng để lọc nhiễu từ tín hiệu thu được.

Mục tiêu nghiên cứu:

* Nghiên cứu nguyên lý hoạt động của cảm biến HC-SR04.
* Xây dựng hệ thống đo khoảng cách sử dụng vi điều khiển.
* Ứng dụng thuật toán Low Pass Filter để cải thiện độ chính xác.

Introduction: Microprocessors and microcontrollers play an important role in modern embedded systems. In the field of measurement, the HC-SR04 ultrasonic sensor is a popular solution for measuring distance with low cost and relatively high accuracy.

Reason for choosing the topic: Distance measurement systems have many practical applications such as obstacle avoidance robots, water level measurement systems, and automation applications. To improve the accuracy of the measurement, the Low Pass Filter (LPF) algorithm is applied to filter noise from the received signal.

Research objectives:

• Study the operating principle of the HC-SR04 sensor.

• Build a distance measurement system using a microcontroller.

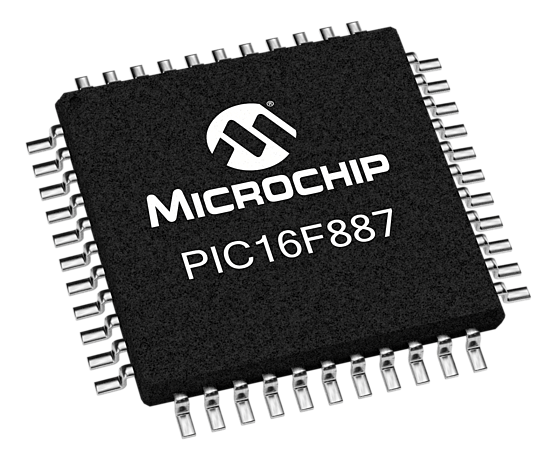
• Apply the Low Pass Filter algorithm to improve accuracy.

# CHƯƠNG 1: TỔNG QUAN VỀ VI XỬ LÝ VI ĐIỀU KHIỂN

## 1.1. TỔNG QUAN VI ĐIỀU KHIỂN

Thông thường có 4 họ vi điều khiển 8 bit chính là 6811 của Motorola, 8051 của Intel, Z8 của Xilog và Pic 16 của Microchip Technology. Mỗi một loại trên đây đều có một tập lệnh và thanh ghi riêng duy nhất, nên chúng thường không tương thích lẫn nhau. Ngoài ra cũng có những bộ vi điều khiển 16 bits và 32 bits được sản xuất bởi các hãng khác nhau.

* PIC là một họ vi điều khiển RISC được sản xuất bởi công ty Microchip Technology. Dòng PIC đầu tiên là PIC1650 được phát triển bởi Microelectronics Division thuộc General Instrument.
* PIC bắt nguồn là chữ viết tắt của "Programmable Intelligent Computer" (Máy tính khả trình thông minh) là một sản phẩm của hãng General Instrument đặt cho dòng sản phẩm đầu tiên của họ là PIC1650.
* Năm 1985 General Instrument bán bộ phận vi điện tử của họ, và chủ sở hữu mới hủy bỏ hầu hết các dự án - lúc đó đã quá lỗi thời. Tuy nhiên PIC được bổ sung EEPROM để tạo thành 1 bộ điều khiển vào ra khả trình. Ngày nay rất nhiều dòng PIC được xuất xưởng với hàng loạt các module ngoại vi tích hợp sẵn (như USART, PWM, ADC...), với bộ nhớ chương trình từ 512 Word đến 32K Word.



Hình 1.1: Hình dạng thực tế của PIC16F887.

## 1.2. PHÂN LOẠI VI ĐIỀU KHIỂN

### Phân loại theo độ dài thanh ghi

### Dựa vào độ dài của các thanh ghi và các lệnh của VĐK mà người ta chia ra các loại vi điều khiển 8 bit, 16 bit hay 32 bit …

### Các loại VĐK 16 bit do có độ dài lệnh lớn hơn nên các tập lệnh cũng nhiều hơn, phong phú hơn. Tuy nhiên bất cứ chương trình nào viết băng VĐK 16 bit chúng ta đều có thể viết trên vi điều khiển 8 bit với chương trình thích hợp.

### Phân loại theo kiến trúc CISC và RISC

### Vi điều khiển CISC là vi điều khiển có tập lệnh phức tạp. Các VĐK này có một số lượng lớn các lệnh nên giúp cho người lập trình có thể linh hoạt và dễ dàng hơn khi viết chương trình.

### Vi điều khiển RISC là vi điều khiển có tập lệnh đơn gian. Chúng có một số lượng nhỏ các lệnh đơn giản. Do đó, chúng đòi hỏi phần cứng ít hơn, giá thành thấp hơn, và nhanh hơn so với CISC. Tuy nhiên nó đòi hỏi người lập trình phải viết các chương trình phức tạp hơn, nhiều lệnh hơn.

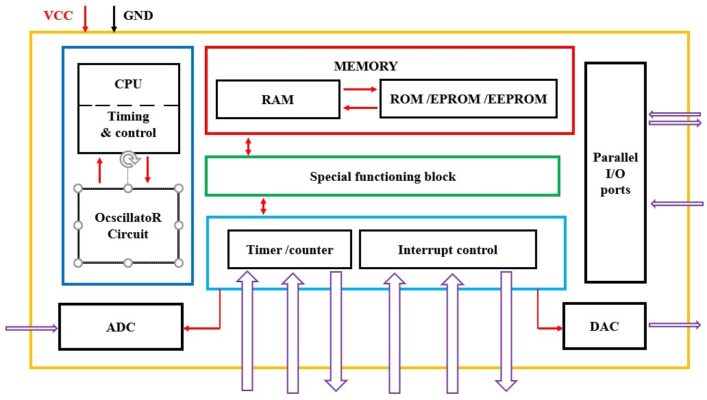
### bo xu ly risc

### Kiến trúc Harvard và kiến trúc Von-Neumann

### Kiến trúc Harvard sử dụng bộ nhớ riêng biệt cho chương trình và dữ liệu. Bus địa chỉ và bus dữ liệu độc lập với nhau nên quá trình truyền nhận dữ liệu đơn giản hơn. Kiến trúc Von-Neumann sử dụng chung bộ nhớ cho chương trình và dữ liệu. Điều này làm cho VĐK gọn nhẹ hơn, giá thành rẻ hơn.

**1.3. Cấu trúc vi điều khiển**

**Cấu trúc tổng quan của vi điều khiển**

****

**CPU hay Vi xử lý**

CPU (Center Programing Unit) hay bộ xử lý trung tâm là bộ não của vi điều khiển. CPU chịu trách nhiệm nạp lệnh, giải mã và thực thi. Tất cả những hành vi của vi điều khiển đều là do CPU điều khiển.

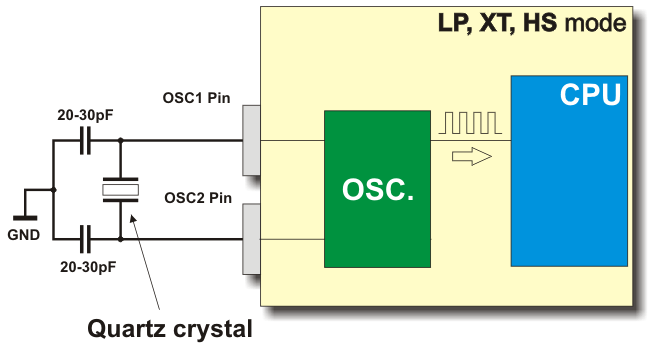
Chúng giao tiếp với các phần khác trong vi điều khiển thông qua hệ thống Bus.



**Ocscillator Circuit**

Nếu CPU là bộ não thì Ocscillator Circuit hay còn gọi là Clock được coi là trái tim của vi điều khiển. Để mọi thứ có thể hoạt động, bắt buộc chúng ta phải cấp xung, trái tim hoạt động mới có thể bơm máu cho toàn bộ cơ thể hoạt động được.

Chúng ta thường nghe quảng cáo dòng vi xư lý có tốc độ bao nhiêu Ghz gì gì đó, chính là tốc độ Clock mà vi xử lý đó có thể đáp ứng được, tốc độ xung càng cao thì tốc độ xử lý của CPU cũng tăng lên. Đương nhiên mọi thứ đều có giới hạn của nó.



**Memory – Bộ nhớ**

Bộ nhớ có thể coi là một phần không thể thiếu, chúng là nơi lưu trữ chương trình nạp lên hoặc dùng làm nơi chứa các thông tin tức thời mà CPU cần dùng tới. Có 2 kiểu bộ nhớ cơ bản:

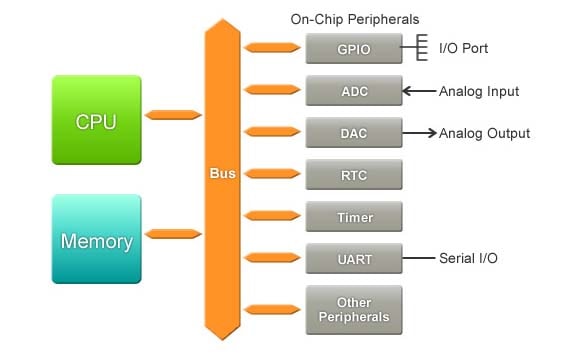
* RAM (Random access memory) là bộ nhớ lưu các dữ liệu mà CPU cần dùng để tính toán, đưa ra quyết định, chúng sẽ bị xóa khi mất điện
* ROM/EPROM/EEPROM hoặc Flash: là bộ nhớ lưu trữ chương trình hay trí khôn của vi điều khiển, chúng được ghi khi chúng ta nạp chương trình vào vi điều khiển, không bị mất khi tắt điện hoặc reset.



**Timer/counter**

Một vi điều khiển có thể có nhiều bộ đếm thời gian và bộ đếm. Bộ đếm thời gian và bộ đếm có chức năng đếm thời gian tạo ra các sự kiện để vi điều khiển hoạt động đúng thời điểm.

Các ngoại vi của vi điều khiển



*Ngoại vi của vi điều khiển*

**I/O Ports – Input/ouput**

Có thể coi I/O Port là tay chân của vi điều khiển, chúng giúp cho vi điều khiển tương tác với các thành phần khác ngoài môi trường.

Cổng đầu vào / đầu ra được sử dụng chủ yếu điều khiển hoặc giao tiếp các thiết bị như màn hình LCD, đèn LED, máy in, …cho vi điều khiển.

**Các chuẩn giao tiếp**

Giống như miệng và tai vậy. Vi điều khiển sẽ sử dụng các chuẩn giao tiếp khác nhau để liên lạc với nhau hoặc liên lạc với các phâng tử khác trên mạch. Có thể kể đến như I2C, SPI, UART, USB, ….

**Bộ chuyển đổi analog sang digital (ADC)**

Bộ chuyển đổi ADC được sử dụng để chuyển đổi tín hiệu analog sang dạng digital. Tín hiệu đầu vào trong bộ chuyển đổi này phải ở dạng analog (ví dụ: đầu ra cảm biến) và đầu ra từ thiết bị này ở dạng digital. Đầu ra digital có thể được sử dụng cho các ứng dụng kỹ thuật số (ví dụ: các thiết bị đo lường).

**Bộ chuyển đổi Digital sang Analog (DAC)**

Hoạt động của DAC là đảo ngược của ADC. DAC chuyển đổi tín hiệu digital thành định dạng analog. Nó thường được sử dụng để điều khiển các thiết bị analog như động cơ DC, các ổ đĩa…

**Interrupt control hay quản lý sự kiện**

Ngoài việc thực thi chương trình, vi điều khiển còn phải tương tác với các tác nhân bên trong và bên ngoài. Các tác nhân này sẽ tạo ra các sự kiện gọi là Ngắt, để quản lý nó cần có một khối quản lý ngắt ( Interrupt control)

# 1.4. Nguyên lý hoạt động

# Nguyên lý hoạt động của vi điều khiển có thể được chia thành các bước chính sau:

# Chu trình lấy lệnh, giải mã và thực thi

# Lấy lệnh (Fetch): Vi điều khiển đọc lệnh từ bộ nhớ chương trình (ROM/Flash).

# Giải mã (Decode): Bộ điều khiển giải mã lệnh để xác định cần thực hiện thao tác gì.

# Thực thi (Execute): Vi điều khiển thực hiện lệnh bằng cách truy cập bộ nhớ, điều khiển các thiết bị ngoại vi hoặc tính toán.

# Xử lý tín hiệu vào/ra (I/O)

# Vi điều khiển nhận tín hiệu từ các cảm biến, công tắc, bàn phím, v.v.

# Sau đó, nó xử lý tín hiệu và gửi lệnh điều khiển ra các thiết bị như LED, động cơ, màn hình hiển thị.

# Hoạt động của bộ nhớ

# Bộ nhớ chương trình (ROM/Flash): Lưu trữ chương trình điều khiển.

# Bộ nhớ dữ liệu (RAM): Lưu trữ tạm thời dữ liệu trong quá trình xử lý.

# Bộ nhớ EEPROM (nếu có): Lưu trữ dữ liệu vĩnh viễn, ngay cả khi mất nguồn.

# Quản lý ngắt (Interrupt)

# Khi có sự kiện xảy ra (như tín hiệu từ cảm biến), vi điều khiển tạm dừng chương trình chính để xử lý sự kiện, sau đó quay lại tiếp tục chương trình.

# Giao tiếp ngoại vi

# Vi điều khiển có thể giao tiếp với các thiết bị khác thông qua các giao thức như UART, SPI, I2C để truyền và nhận dữ liệu.

# 

# CHƯƠNG 2: THIẾT KẾ HỆ THỐNG

## 2.1. Mô tả hệ thống

**2.1.1. Giới thiệu**

Trong chương này, nhóm sẽ trình bày thiết kế thước cảm biến áp dụng thuật toán Low pass filter sử dụng cảm biến HC-SR04. Hệ thống bao gồm phần cứng và phần mềm tích hợp để đo khoảng cách bằng thuật toán Low pass filter với tín hiệu thu từ cảm biến HC-SR04

## 2.2. Thiết kế sơ đồ khối

Sơ đồ khối hệ thống bao gồm:

Khối nguồn: Cung cấp nguồn cho toàn bộ các khối trong hệ thống.

Khối xử lý trung tâm: Vi điều khiển PIC16F887.

Khối giao tiếp: Giao tiếp qua LCD bằng giao thức I2C.

Khối Programmer: Nạp chương trình điều khiển vào trong vi điều khiển.

Khối nút nhấn: Thực hiện yêu cầu các chức năng người dùng.

Khối hiển thị: Hiển thị nội dung qua LCD.

Khối điều khiển: Điều khiển chế độ qua nút bấm.

A diagram of a software company

AI-generated content may be incorrect.

Hình 2.1: Sơ đồ khối hệ thống

**2.3.1. Sơ đồ khối mạch nguồn**

Khối nguồn là khối quan trọng vì cung cấp nguồn cho toàn bộ hoạt động hệ thống bao gồm: khối xử lý trung tâm, khối cảm biến, khối nút bấm, khối hiển thị.

Nguồi nuôi thường duy trì ổn định ở mức +5V. Do yêu cầu cao của hệ thống các nguồn nuôi thường được chế tạo một cách đặc biệt nhằm đem lại hiệu quả, và tính ổn định cao.

Thông thường có 2 kiểu nguồn chính: Dùng pin hoặc ắc quy cho điện áp tương đối ổn định, mặc dù trên thị trường không có loại pin hoặc ắc quy chuẩn 5V cho nên nếu dùng nó thì phải qua một bộ biến đổi điện áp để đưa điện áp về dạng chuẩn hơn nữa trong quá trình sử dụng, năng lượng trong pin, ắc quy hết đi hệ thống sẽ bị gián đoạn.

Để cung cấp nguồn nuôi cho cả hệ thống chúng tôi sử dụng nguồn 7-9V DC hạ áp xuống 5V DC. Mạch hạ áp buck được sử dụng thay vì hạ áp tuyến tính để tăng hiệu năng hạ áp lên đến 3A và giảm thiểu hiệu suất nguồn tiêu tán thông qua nhiệt năng vốn là đặc điểm của mạch hạ áp tuyến tính. Sơ đồ mạch như hình 2.2:

**A diagram of a circuit

AI-generated content may be incorrect.**

Hình 2.2: Sơ đồ nguyên lý mạch nguồn

**Nguyên lý hoạt động của mạch buck hạ áp:**[5]

1. Nguồn vào (VIN):

- Chân 7 (VIN):Điện áp đầu vào 7-9V được cung cấp tới chân VIN của IC. Chân này cung cấp điện áp cho tất cả các mạch điều khiển bên trong IC.

2. Chân EN (Enable Input):

- Chân 2 (EN): Điều khiển bật/tắt IC. Khi chân EN được kéo lên cao qua điện trở R5 (100kΩ) và R6 (30kΩ), IC sẽ hoạt động. Nếu kéo chân này xuống dưới mức ngưỡng quy định, IC sẽ tắt.

3. Chân SW (Switch Node):

- Chân 1 (SW): Nút chuyển đổi, đầu ra từ bộ chuyển mạch high-side. Kết nối với cuộn cảm L1 (15μH) và diode D2 (SS34). Diode này cần được đặt gần chân SW để giảm thiểu nhiễu chuyển mạch.

4. Chân FB (Feedback):

- Chân 4 (FB): Đầu vào phản hồi. Điện áp đầu ra được điều chỉnh qua phân áp điện trở R8 (51kΩ) và R9 (10kΩ), và điện áp này được đưa vào chân FB để IC điều chỉnh điện áp đầu ra về mức mong muốn. Điện áp chuẩn nội bộ của IC là 0.8V.

5. Chân COMP (Compensation):

- Chân 3 (COMP): Đầu ra của bộ khuếch đại lỗi. Điện áp này được điều chỉnh qua mạng điện trở R10 (100kΩ) và tụ điện C11 (150pF) để ổn định hệ thống phản hồi.

6. Chân FREQ (Switching Frequency Program Input):

- Chân 6 (FREQ): Điều chỉnh tần số chuyển mạch của IC. Điện trở R7 (100kΩ) kết nối từ chân này đến đất để thiết lập tần số hoạt động của IC.

7. Chân GND (Ground):

- Chân 5 (GND):Kết nối với đất. Chân này cần được nối gần tụ điện đầu ra để giảm thiểu trở kháng và nhiễu.

8. Chân BST (Bootstrap):

- Chân 8 (BST):Nguồn dương cho driver MOSFET bên trong. Một tụ điện bootstrap (C10, 100nF) kết nối giữa chân BST và chân SW để cung cấp năng lượng cho MOSFET high-side.

9. Tụ lọc đầu vào và đầu ra:

- Tụ C9 (100nF): Tụ lọc đầu vào giúp ổn định điện áp vào.

- Tụ C12 (22μF): Tụ lọc đầu ra giúp ổn định điện áp đầu ra.

**2.3.3. Khối MCU**

A diagram of a computer

AI-generated content may be incorrect.

Hình 2.3: Sơ đồ khối MCU

**2.3.4. Sơ đồ khối Header**

A diagram of a circuit board

AI-generated content may be incorrect.

Hình 2.4: Sơ đồ khối HEADER

## 2.3. Thiết kế sơ đồ nguyên lý

**2.3.1. Sơ đồ nguyên lý tổng quan**

A white board with black text

AI-generated content may be incorrect.

Hình 2.5: Sơ đồ nguyên lý

*A table with numbers and letters

AI-generated content may be incorrect.*

*Bảng 2.6: Bảng linh kiện và tham số linh kiện sử dụng trong mạch*

## 2.4. Thiết kế layout và chế tạo PCB

A computer circuit board with many wires

AI-generated content may be incorrect.

Hình 2.7: Mạch dán PCB được thiết kế bằng ứng dụng Kicad

**2.4.1. Sơ đồ mạch in**

A black and white drawing of a computer chip

AI-generated content may be incorrect.

Hình 2.8: Sơ đồ mạch in

## 2.5. Thiết kế giải thuật

### 2.5.1. Thuật toán ngoại vi I2C

A diagram of a computer system

Description automatically generated

Hình 2.9: Sơ đồ khối ngoại vi I2C chế độ Master của PIC16F887.[4]

Để sử dụng I2C ở chế độ master trên vi điều khiển PIC, các thanh ghi sau đây cần được cấu hình và sử dụng. Dưới đây quy trình khởi động ngoại vi I2C với baud rate 100KHz cùng với ghi chú thích về ý nghĩa và vai trò của từng thanh ghi:

A screen shot of a computer program

AI-generated content may be incorrect.

A close-up of a register

Description automatically generated A screenshot of a computer program

Description automatically generated

Hình 2.10: Thanh ghi SSPCON cấu hình I2C.[4]

A white paper with black text

Description automatically generated A white text on a black background

Description automatically generated

Hình 2.11: Thanh ghi SSPCON2 cấu hình I2C.[4]

A close up of a document

Description automatically generated

A screenshot of a computer program

Description automatically generated

Hình 2.12: Thanh ghi SSPSTAT trạng thái I2C.[4]

**SSPADD = ((\_XTAL\_FREQ / 4) / I2C\_BaudRate) - 1;**

Thiết lập tốc độ baud rate cho giao tiếp I2C. Công thức tính giá trị SSPADD dựa trên tần số của bộ dao động \_XTAL\_FREQ và tốc độ I2C mong muốn (ở đây I2C\_BaudRate=100kHz). A diagram of a data flow

AI-generated content may be incorrect.

Hình 2.13: Lưu đồ thuật toán gửi 1 byte vào bus I2C.

A computer screen shot of a program

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.14: Quy trình gửi data từ Master cho Slave với địa chỉ 0x4E qua bus I2C.

A diagram of a flowchart

Description automatically generated

Hình 2.15: Lưu đồ thuật toán gửi 1 kí tự vào LCD.

A screenshot of a computer program

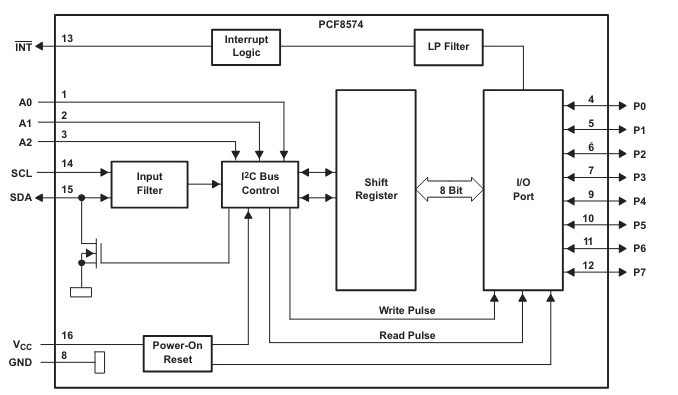
AI-generated content may be incorrect.

A screenshot of a black screen

AI-generated content may be incorrect.

Hình 2.16: Dữ liệu truyền cho PCF8574AT điều khiển LCD.

PCF8574AT là module điều khiển mở rộng IO thông qua bus I2C nên điều khiển LCD vẫn sử dụng phương thức 4 bit.[7]



Hình 2.17: Kết nối module PCF8574T

### 2.5.2. Thuật toán chuyển từ thời gian sang khoảng cách với HCSR04

Cảm biến khoảng cách siêu âm HC-SR04 là một giải pháp phổ biến và chi phí thấp cho chức năng đo khoảng cách không tiếp xúc. Nó có thể đo khoảng cách từ 2cm đến 400cm với độ chính xác khoảng 3 mm. Mô -đun này bao gồm máy phát siêu âm, máy thu siêu âm và mạch điều khiển của nó.

A diagram of a machine

AI-generated content may be incorrect.

Hình 2.18 Sơ đồ kết nối HCSR04

1. Cung cấp tín hiệu kích hoạt, xung mức độ cao (5V) cao nhất (5V).
2. Mô -đun sẽ tự động truyền tám lần siêu âm 40kHz.
3. Nếu có một trở ngại ở phía trước của mô-đun, nó sẽ phản ánh sự bùng nổ siêu âm.
4. Nếu tín hiệu trở lại, đầu ra echo của cảm biến sẽ ở trạng thái cao (5V) trong một khoảng thời gian để gửi và nhận siêu âm. Chiều rộng xung dao động từ khoảng 150 μs đến 25ms và nếu không phát hiện ra chướng ngại vật, chiều rộng xung Echo sẽ là khoảng 38ms.

Dùng timer1 module để ghi lại thời gian xung sườn lên của chân echo cho đến khi gặp sườn xuống.

Mô -đun Timer1 có thể được sử dụng làm bộ đếm hoặc bộ đếm thời gian 16 bit. Nó bao gồm hai thanh ghi 8 bit TMR1H và TMR1L có thể đọc được và có thể ghi. Cặp thanh ghi, TMR1H: TMR1L tăng từ 0000H đến FFFFH và cuộn qua 0000h. Nếu ngắt dòng Timer1 được bật được tạo trong quá trình cuộn qua 0000h. Ở đây sẽ sử dụng mô -đun này như một bộ đếm thời gian 16 bit.

A diagram of a device

AI-generated content may be incorrect.

Vì chúng ta đang sử dụng mô-đun Timer1 làm Timer, chúng ta nên sử dụng đồng hồ bên trong (Fosc/4), tức là TMR1CS = 0. Bộ chia trước được sử dụng để chia đồng hồ bên trong (Fosc/4). Ở đây chúng ta có thể đặt Bộ chia trước thành 2, tức là T1CKPS1 = 0 & T1CKPS0 = 1. Bit T1SYNC bị bỏ qua khi TMR1CS = 0. Vì chúng ta đang sử dụng đồng hồ bên trong (Fosc/4), chúng ta có thể vô hiệu hóa bộ dao động, tức là T1OSEN = 0. Bit TMR1ON có thể được sử dụng để BẬT hoặc TẮT bộ hẹn giờ theo yêu cầu của chúng ta.

Dùng ngắt ngoài cho chân Echo của cảm biến để khởi tạo sự khiến bắt đầu đếm cho Timer 1, IRQ là viết tắt của các yêu cầu ngắt. Thực tế có một số chân vật lý phần cứng chuyên dụng có khả năng nhận được các yêu cầu ngắt bên ngoài. Số lượng các chân IRQ có sẵn thay đổi từ chip vi điều khiển sang chip khác. Nó rất hiếm khi tìm thấy một con chip vi điều khiển không có ghim IRQ.

Một chốt IRQ có thể bắn tín hiệu ngắt vào CPU để đình chỉ thói quen chính được thực hiện. Việc bắn gián đoạn xảy ra trên một số sự kiện trên pin IRQ. Nó có thể là một trong những sự kiện sau đây:

* Thấp (kích hoạt cấp độ)
* Cao (kích hoạt cấp độ)
* Rising Edge (kích hoạt cạnh)
* Cạnh rơi (kích hoạt cạnh)
* IOC (ngắt thay đổi)

**A screenshot of a computer program

AI-generated content may be incorrect.**

Tính toán khoảng thời gian vừa đếm được từ Timer 1 sang khoảng cách

Nguyên tắc vật lý:

1. Tốc độ âm thanh: Sóng siêu âm di chuyển ở khoảng 343 mét mỗi giây (34.300 cm/s) trong không khí ở nhiệt độ phòng.
2. Chuyến đi khứ hồi: Bộ đếm thời gian đo tổng thời gian để âm thanh di chuyển đến vật thể và quay lại.
3. Công thức cơ bản: khoảng cách = (thời gian × tốc độ của âm thanh) / 2.

Timer 1 được khởi tạo với Prescaler 1:1

Tinh thể 12MHz → Tần số hẹn giờ 3MHz (FOSC/4)

Mỗi lần hẹn giờ = 0,33333 microseconds (1/3000000)

Không áp dụng PRESCALER (1: 1)

Chuyển đổi thời gian: hẹn giờ đánh dấu vào micro giây

Thời gian (μs) = Timer\_Ticks × 0,333 ...

Công thức khoảng cách:

Khoảng cách (cm) = (thời gian (μs) × 34.300 cm / s) / 2 / 1.000.000

Khoảng cách (cm) = (Timer\_Ticks × 0,333 ... × 34.300) / 2.000.000

Khoảng cách (cm) = Timer\_Ticks × 0,00572 ... (Đơn giản hóa)

Khoảng cách (cm) = Timer\_Ticks / 174.8 ... (đảo ngược để phân chia)

Đơn giản hóa cuối cùng:

Mã sử ​​dụng 174 làm xấp xỉ số nguyên

**A screenshot of a computer program

AI-generated content may be incorrect.**

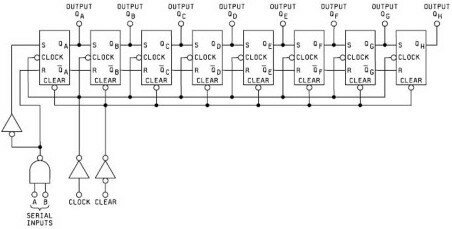
Kết quả được lấy giá trị trung bình sau 20 lần đo A screenshot of a computer program

AI-generated content may be incorrect.

### 2.5.2. Thuật toán ngoại vi UART

Các giao thức giao tiếp nối tiếp khác nhau tồn tại, mỗi giao thức có chi tiết hoạt động độc đáo. Tuy nhiên, tất cả họ đều chia sẻ một cốt lõi chung: đăng ký thay đổi. Các thanh ghi này là nền tảng trong việc triển khai phần cứng của các giao thức giao tiếp nối tiếp, chuyển dữ liệu từng bit từng chu kỳ.

Cách thức hoạt động:

Các thanh ghi dịch chuyển bao gồm các flip-flop được kết nối huyết thanh chia sẻ cùng một dòng đồng hồ. Đầu vào dữ liệu được chuyển từ chân đầu vào sang đầu đầu ra, chuyển một bit cho mỗi chu kỳ đồng hồ. Điều này có nghĩa là một thanh ghi thay đổi 8 bit mất 8 chu kỳ đồng hồ để chuyển byte. 

Hình 2.19. Minh họa một thanh ghi thay đổi 8 bit với các dòng đầu vào và đầu ra nối tiếp

Giao tiếp nối tiếp là cơ bản trong nhiều ứng dụng, chẳng hạn như giao tiếp thiết bị bên ngoài, tải xuống chương trình cơ sở, I/O điều khiển, truyền dữ liệu và giao diện gỡ lỗi. Thành thạo các giao thức giao tiếp nối tiếp khác nhau là điều cần thiết cho các kỹ sư hệ thống nhúng do sử dụng rộng rãi của họ. Các giao thức phổ biến bao gồm USB, CAN, I2C, I2S, LIN, SPI, Ethernet, 1 dây và UART/USART. Hướng dẫn này tập trung vào UART để giải thích các nguyên tắc cơ bản và cơ học của nó, dẫn đến truyền dữ liệu giữa MCU nhúng.

Thế nào là UART(Universal Asynchronous Receiver/transmitter):

UART, viết tắt cho máy thu/máy phát không đồng bộ phổ quát, bao gồm mạch phần cứng cần thiết cho giao tiếp nối tiếp, có sẵn dưới dạng IC độc lập hoặc các mô -đun nội bộ trong vi điều khiển. Phương pháp giao tiếp này dựa trên các chân I/O chuyên dụng: RX (kết thúc nhận) và TX (kết thúc truyền).

Các chế độ giao tiếp UART:

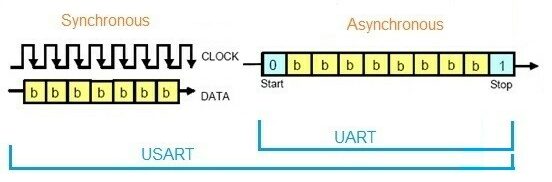
* Simplex: Cho phép giao tiếp một chiều từ máy phát đến máy thu.
* Một nửa song công: liên quan đến việc truyền tải và tiếp nhận xen kẽ.
* Toàn bộ song công: Cho phép truyền và tiếp nhận đồng thời.
* Cấu trúc gói dữ liệu UART:

Bắt đầu bit: báo hiệu sự khởi đầu của một gói mới.

* Khung dữ liệu: Chứa các bit dữ liệu thực tế, thường dao động từ 5 đến 9 bit.
* Bit Parity: Tùy chọn để kiểm tra lỗi.
* Bit dừng: Cho biết phần cuối của gói dữ liệu.
* Tỷ lệ baud: Tốc độ Baud biểu thị tốc độ truyền dữ liệu tính bằng bit-per-second (bps). Tốc độ baud tiêu chuẩn phổ biến bao gồm 9600, 1200, 2400, 4800, 19200, 38400, 57600 và 115200 bps.

Sự khác nhau giữa UART và USART:

UART và USART là hai thành phần thiết yếu trong giao tiếp nối tiếp, mỗi thành phần có đặc điểm riêng: UART: là viết tắt của máy thu/máy phát không đồng bộ phổ quát. Nó hoạt động như một thực thể độc lập, phù hợp cho các ứng dụng tốc độ thấp.



UART sử dụng một chiếc đồng hồ được tạo ra cục bộ, làm cho nó tiết kiệm năng lượng. USART: biểu thị máy thu/máy phát đồng bộ/không đồng bộ toàn cầu. Không giống như UART, USART hỗ trợ các giao thức khác nhau, cho phép giao tiếp tốc độ cao. Nó sử dụng đồng hồ do máy phát tạo ra nhưng có xu hướng tiêu thụ nhiều năng lượng hơn so với các thiết lập UART.

UART đối với vi điều khiển PIC:

PIC16F877 UART là một thành phần thiết yếu để giao tiếp nối tiếp, truyền và nhận dữ liệu hiệu quả trong khi ưu tiên LSB (bit ít có ý nghĩa nhất) trước tiên. Hệ thống độc lập phần cứng này đảm bảo hoạt động liền mạch bằng cách tuân thủ các định dạng dữ liệu nhất quán và tốc độ baud. Tại trung tâm của chức năng UART PIC16F877 là trình tạo tốc độ baud, có thể định cấu hình cho tỷ lệ dịch chuyển bit x16 hoặc x64 dựa trên bit BRGH. Mặc dù hỗ trợ phần cứng cho chẵn lẻ là không có, việc triển khai phần mềm bit dữ liệu thứ chín bù cho giới hạn này. Đáng chú ý, chế độ không đồng bộ ngừng hoạt động trong chế độ ngủ, đảm bảo hiệu quả năng lượng.

Mô -đun không đồng bộ UART bao gồm các thành phần quan trọng:

* Trình tạo tốc độ baud
* Mạch lấy mẫu
* Máy phát không đồng bộ
* Máy thu không đồng bộ

Chức năng cốt lõi của bộ phát, xoay quanh thanh ghi dịch chuyển (TSR) của bộ truyền (TSR), thu được dữ liệu từ bộ đệm truyền/ghi (TXREG). Can thiệp phần mềm là cần thiết để tải dữ liệu vào thanh ghi TXREG, đảm bảo quá trình truyền dữ liệu trơn tru. Sau khi hoàn thành bit dừng truyền dẫn trước đó, thanh ghi TSR tải khung dữ liệu mới từ TXREG, tạo điều kiện truyền dữ liệu liên tục. Đáng chú ý, bit cờ TXIF biểu thị một điều kiện ngắt, có thể điều chỉnh thông qua bit TXIE, trong khi bit trạng thái TRMT biểu thị độ trống của thanh ghi TSR, cần phải bỏ phiếu thủ công.

A diagram of a computer server

AI-generated content may be incorrect.

Sự khởi đầu truyền dẫn đòi hỏi phải đặt bit kích hoạt TXEN, được kết hợp với tải dữ liệu vào thanh ghi TXREG và tạo đồng hồ bằng trình tạo tốc độ baud. Tính linh hoạt tồn tại để bắt đầu truyền bằng cách tải TXREG đầu tiên hoặc bằng cách tải TxREG đồng thời và cho phép TXEN. Vô hiệu hóa TXEN Mid Transmission phá thai quá trình, đặt lại máy phát và hoàn nguyên chân TX/RC6 về trạng thái biến đổi cao.

Các bước cấu hình máy phát UART:

* Định cấu hình tốc độ baud: Khởi tạo thanh ghi SPBRG để đạt được tốc độ baud mong muốn, sử dụng bit BRGH cho hoạt động tốc độ cao.
* Kích hoạt cổng nối tiếp không đồng bộ: Kích hoạt cổng nối tiếp không đồng bộ bằng cách định cấu hình các bit đồng bộ và spen.
* Đặt hướng dữ liệu pin: Xác định hướng dữ liệu cho các chân RX và TX (RC6/TX/CK và RC7/RX/DT) cho hoạt động UART.
* Bật truyền UART: Kích hoạt truyền UART bằng cách đặt bit TXEN.

A screenshot of a computer program

AI-generated content may be incorrect.

* Tải dữ liệu vào TXREG: Tải dữ liệu vào thanh ghi TXREG để truyền.

A computer screen shot of a program

AI-generated content may be incorrect.

Giao tiếp với máy tính thông qua CH340C

**A computer circuit board with many wires

AI-generated content may be incorrect.**

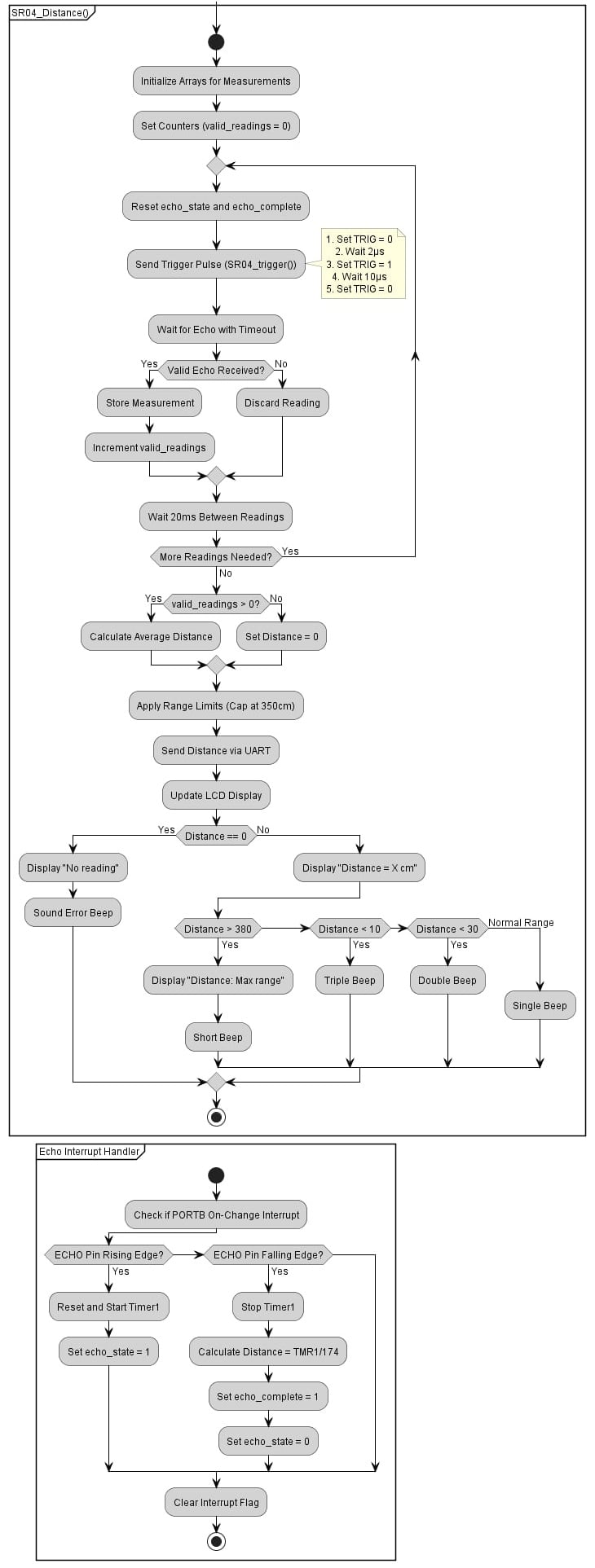
# CHƯƠNG 3: KẾT QUẢ THỰC HIỆN VÀ KẾT LUẬN

## 3.1. Kết quả thực hiện

### 3.1.1. Lập trình hệ thống

A diagram of a flowchart

AI-generated content may be incorrect.



### 3.1.2. Kết quả mô phỏng

Hệ thống được mô phỏng trên phần mềm Proteus mạch bao gồm:

1. Pic 16F887
2. Cảm biến siêu âm HCSR04
3. Màn hình LCD 20x4 IC2
4. Nút nhấn

A computer screen shot of a computer

AI-generated content may be incorrect.

Hình 3.1. Sơ đồ mạch mô phỏng

**A screen shot of a computer screen

AI-generated content may be incorrect.**

Hình 3.1.2 Kết quả đo được từ 2 chân Trig và Echo của cảm biến HCSR04

**A screen shot of a computer

AI-generated content may be incorrect.**

Hình 3.1.3 Kết quả được hiện thị thông qua UART

### Kết quả thực nghiệm



Hình 3.1.4.Sản phầm hoàn thiện

|  |  |
| --- | --- |
| Thông số cấu hình | |
| Điện áp hoạt động | 6V-28V |
| Khoảng cách đo | 2cm-400cm |
| Ngoại vi | LCD, nút nhấn |

## 3.2. Kết luận và hướng phát triển

3.2.1. Ưu điểm

1. Độ chính xác cao:

Cảm biến HCSR04 cho phép khoảng cách đo từ 2cm đến 400cm với sai lệnh 3mm, giúp tăng độ tin cậy về thông số khoảng cách.

2. Độ phức tạp thấp

4. Chi phí hợp lý:

Các thành phần như cảm biến HCSR04 và vi điều khiển PIC16F887 có giá thành không cao, phù hợp với nhiều dự án từ nghiên cứu hoạc thuật đến ứng dụng thực tế.

3.3.2. Nhược điểm

1. Phụ thuộc vào độ chính xác của cảm biến:

Nếu cảm biến HCSR04 được dùng trong môi trường khắc nghiệt dễ ngây hỏng hóc và bị ảnh hưởng bởi nhiễu do môi trường.

2. Giới hạn của vi điều khiển:

PIC16F887 là vi điều khiển khá cũ và có giới hạn về hiệu suất và bộ nhớ.

3.Khả năng mở rộng hạn chế:

Do giới hạn của vi điều khiển Pic16f887, hệ thống có thể gặp khó khăn khi cần mở rộng hoặc tích hợp với các hệ thống khác.

### 3.3.3. Kết luận

1. Mạch phần cứng:

Đã được thiết kế và lắp ráp ổn định, đảm bảo tính chính xác và độ bền cao.

1. Vận hành:

Hệ thống vận hành trơn tru, đáp ứng nhanh với các thay đổi và duy trì ổn định.

3.4. Hướng phát triển

1. Có thể tích hợp thêm các ngoại vi như động cơ, đèn cảnh báo,…

2. Nâng cấp phần cứng:

- Sử dụng các cảm biến tiên tiến hơn và vi điều khiển mạnh hơn để nâng cao khả năng và mở rộng ứng dụng.

- Tối ưu hoá thiết kế mạch để giảm kích thước và tăng tính linh hoạt trong việc tích hợp và các hệ thống khác.

3. Phát triển phần mềm hỗ trợ:

- Xây dựng giao diện cho người dùng thận thiện để người dùng có thể dễ dàng điều chỉnh và giám sát hệ thống.

# TÀI LIỆU THAM KHẢO

[1] A. De Marcellis, G. Ferri, and P. Mantenuto, “Resistive Sensor Interfacing,” in *Giant Magnetoresistance (GMR) Sensors: From Basis to State-of-the-Art Applications*, C. Reig, S. Cardoso, and S. C. Mukhopadhyay, Eds. Berlin, Heidelberg: Springer, 2013, pp. 71–102. doi: 10.1007/978-3-642-37172-1\_4.

[2] S. Shirotori, A. Kikitsu, Y. Higashi, Y. Kurosaki, and H. Iwasaki, “Symmetric Response Magnetoresistance Sensor With Low 1/f Noise by Using an Antiphase AC Modulation Bridge,” *IEEE Trans. Magn.*, vol. 57, no. 2, pp. 1–5, Feb. 2021, doi: 10.1109/TMAG.2020.3012655.

Link Github chứa tư liệu, thiết kế, demo project:

A qr code on a white background

AI-generated content may be incorrect.