TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA ĐIỆN – ĐIỆN TỬ**

🗕🗕🗕🗕🗕🙣🕮🙡🗕🗕🗕🗕🗕



**LÝ TRUYỀN PHÁT**

**THIẾT KẾ HỆ THỐNG BÁO CHÁY**

**ĐỒ ÁN HỆ THỐNG NHÚNG**

**KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG**

**THÀNH PHỐ HỒ CHÍ MÌNH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA ĐIỆN – ĐIỆN TỬ**

🗕🗕🗕🗕🗕🙣🕮🙡🗕🗕🗕🗕🗕



**LÝ TRUYỀN PHÁT**

**THIẾT KẾ HỆ THỐNG BÁO CHÁY**

**ĐỒ ÁN HỆ THỐNG NHÚNG**

**KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG**

Người hướng dẫn

**TS Nguyễn Văn Dũng**

**THÀNH PHỐ HỒ CHÍ MÌNH, NĂM 2022**

**LỜI CẢM ƠN**

Để hoàn thành bài tiểu luận này, em xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn bộ môn - Thầy Nguyễn Văn Dũng đã giảng dạy tận tình, chi tiết để em có đủ kiến thức và vận dụng chúng vào bài tiểu luận này.

Do chưa có nhiều kinh nghiệm về kiến thức và làm đồ án nên chắc chắn không thể trành khỏi những thiếu sót trong đồ án. Em rất mong nhận được sự góp ý, nhận xét cũng như phê bình từ phía thầy. Em xin chân thành cảm ơn.

*TP. Hồ Chí Minh, Thứ 5 ngày 13 tháng 01 năm 2022*

*Tác giả*

***Lý Truyền Phát***

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin đảm bảo đây là đồ án của riêng tôi dưới sự hướng dẫn khoa học của Thầy Nguyễn Văn Dũng. Các nội dung trong đồ án và kết quả là trung thực. Những số liệu trong các bảng trong báo cáo giúp cho việc phân tích, nhận xét, đánh giá được thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo của báo cáo

**Nếu phát hiện, kiểm tra có bất kỳ sự gian lận nào trong đồ án của tôi, tôi xin hoàn toàn chịu toàn bộ trách nhiệm về nội dung luận văn trong đồ án của tôi.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, Thứ 5 ngày 13 tháng 01 năm 2022*

*Tác giả*

***Lý Truyền Phát***

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

Phần xác nhận của GV hướng dẫn:

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

*Tp. Hồ Chí Minh, ngày tháng năm*

*(kí và ghi họ tên)*

**PHIẾU THEO DÕI TIẾN ĐỘ THỰC HIỆN**

**ĐỒ ÁN HỆ THỐNG NHÚNG**

Họ tên sinh viên: **LÝ TRUYỀN PHÁT** MSSV: **41800545**

Lớp: **18040201**

GVHD: TS Nguyễn Văn Dũng

Tên đề tài: **HỆ THỐNG BÁO CHÁY BẰNG DS18B20 HIỂN THỊ RA LED**

|  |  |  |
| --- | --- | --- |
| **Tuần/Ngày** | **Nội dung đã thực hiện** | **GVHD xác nhận** |
| **1**  **(25/10 – 31/10)** | * Nhận tờ nhiệm vụ Đồ án * Hiểu rõ yêu cầu nội dung đề tài, qui định thực hiện đồ án |  |
| **2**  **(1/11 – 7/11)** | * Tìm hiểu pic 16F877A * Tìm hiểu linh kiện DS18B20 * Tìm hiểu giao tiếp 1 wire |  |
| **3**  **(8/11 – 14/11)** | * Tìm hiểu nguyên lí hoạt động * Vẽ sơ đồ khối, sơ đồ mạch * Thực hành giải thuật |  |
| **4**  **15/11 – 21/11** | * Mô phỏng mạch trên testboard * Viết báo cáo |  |
| **5**  **22/11 – 28/11** | GVHD đánh giá khối lượng hoàn thành……..%  🞎được tiếp tục 🞎không tiếp tục |  |
| **6**  **(29/11 – 5/12)** | * Mô phỏng thuật toán * Vẽ layout * Viết báo cáo |  |
| **7**  **(6/12 – 12/12)** | * In mạch * Lắp mạch cứng * Viết báo cáo |  |
| **8**  **(13/12 – 19/03)** | * Kiểm tra mạch cứng * Kiểm tra báo cáo |  |
| **9**  **20/12 – 26/12** | * Hoàn thành mạch cứng * Hoàn thành báo cáo |  |
| **10**  **(27/12 – 2/1)** | GVHD đã duyệt báo cáo đồ án, đề nghị:  🞎được bảo vệ 🞎không được bảo vệ |  |

**MỤC LỤC**

[**DANH MỤC CÁC TỪ VIẾT TẮT** IX](#_Toc93074616)

[**DANH MỤC HÌNH ẢNH** X](#_Toc93074617)

[**DANH MỤC BẢNG** XI](#_Toc93074618)

[***CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI*** 1](#_Toc93074619)

[*1.1* *TÍNH CẤP THIẾT CỦA ĐỀ TÀI* 1](#_Toc93074620)

[*1.2* *YÊU CẦU CỦA ĐỀ TÀI* 1](#_Toc93074621)

[*1.3* *NỘI DUNG NGHIÊN CỨU* 1](#_Toc93074622)

[*1.4* *PHƯƠNG PHÁP THỰC HIÊN* 1](#_Toc93074623)

[*1.5* *BỐ CỤC* 1](#_Toc93074624)

[***CHƯƠNG 2: THÀNH PHẦN LINH KIỆN*** 3](#_Toc93074625)

[*2.1* *PIC16F877A* 3](#_Toc93074626)

[*2.2* *DS18B20* 9](#_Toc93074627)

[*2.2.1* *Giới thiệu* 9](#_Toc93074628)

[*2.2.2* *Đặc tính và thông số của cảm biến DS18B20:* 10](#_Toc93074629)

[*2.2.3* *Mã ROM 64 bit:* 10](#_Toc93074630)

[*2.2.4* *Bộ nhớ* 11](#_Toc93074631)

[*2.2.5* *Kiểu kết nối với vi xử lý* 15](#_Toc93074632)

[*2.3* *LED* 15](#_Toc93074633)

[*2.4* *THẠCH ANH 20MHZ* 17](#_Toc93074634)

[***CHƯƠNG 3: NGUYÊN LÍ HOẠT ĐỘNG*** 18](#_Toc93074635)

[*3.1* *LÝ THUYẾT* 18](#_Toc93074636)

[*3.2* *GIAO TIẾP GIỮA CÁC LINH KIỆN* 18](#_Toc93074637)

[*3.2.1* *Giao tiếp One - Wire* 18](#_Toc93074638)

[*3.2.2* *Kỹ thuật quét LED* 22](#_Toc93074639)

[*3.3* *LƯU ĐỒ GIẢI THUẬT* 23](#_Toc93074640)

[*3.4* *SƠ ĐỒ KHỐI* 24](#_Toc93074641)

[***CHƯƠNG 4: THỰC HÀNH VÀ KẾT QUẢ*** 25](#_Toc93074642)

[*4.1* *SƠ ĐỒ MẠCH MÔ PHỎNG* 25](#_Toc93074643)

[*4.2* *MẠCH IN* 26](#_Toc93074644)

[*4.3* *MẠCH ĐỒNG* 27](#_Toc93074645)

[*4.4* *MẠCH HOÀN CHỈNH* 28](#_Toc93074646)

[***CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN*** 30](#_Toc93074647)

[*5.1* *ƯU ĐIỂM* 30](#_Toc93074648)

[*5.2* *NHƯỢC ĐIỂM* 30](#_Toc93074649)

[*5.3* *KẾT LUẬN* 30](#_Toc93074650)

[*5.4* *HƯỚNG PHÁT TRIỂN ĐỒ ÁN* 30](#_Toc93074651)

[***TÀI LIỆU THAM KHẢO*** 32](#_Toc93074652)

[***PHỤ LỤC A: ONE WIRE*** 1](#_Toc93074653)

[***PHỤ LỤC B: DS18B20*** 3](#_Toc93074654)

[***PHỤ LỤC C: CODE*** 5](#_Toc93074655)

# **DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| **Viết tắt** | **Tiếng anh** |
| **PIC** | Programmable Intelligent Computer |
| **LED** | Light Emitting Diode |
| **CRC** | Cyclic Redundancy Check |
| **SRAM** | Static random-access memory (RAM TĨNH) |
| **ROM** | Read – Only Memory |
| **EEPROM** | (Electrically Erasable Programmable Read-Only Memory) |
| **IC** | Integrated Circuit |
| **VAC** | Volt Alternating Current |
| **VDC** | Volt Direct Current |
| **GND** | Ground |

# **DANH MỤC HÌNH ẢNH**

[Hình 2.1 Pic 16F877A 3](#_Toc93074659)

[Hình 2.2 Thành phần của Pic 16F877A 4](#_Toc93074660)

[Hình 2.3 Sơ đồ chân Pic 16F877A 5](#_Toc93074661)

[Hình 2.4 Cảm biến nhiệt DS18B20 10](#_Toc93074662)

[Hình 2.5 Mã ROM 64 bit 11](#_Toc93074663)

[Hình 2.6 Byte 0 và Byte 1 của DS18B20 11](#_Toc93074664)

[Hình 2.7 Thanh ghi TL và TH 12](#_Toc93074665)

[Hình 2.8 Byte 4 thanh ghi cấu hình 12](#_Toc93074666)

[Hình 2.9 Bộ nhớ SRAM 13](#_Toc93074667)

[Hình 2.10 Kiểu kết nối của DS18B20 15](#_Toc93074668)

[Hình 2.11 LED 7 đoạn 16](#_Toc93074669)

[Hình 2.12 Linh kiện thạnh anh 17](#_Toc93074670)

[Hình 3.1 Giao tiếp One-wire 19](#_Toc93074671)

[Hình 3.2 Cơ sở truyền nhận của DS18B20 19](#_Toc93074672)

[Hình 4.1 Sơ đồ mô phỏng 25](#_Toc93074681)

[Hình 4.2 Sơ đồ mạch in 26](#_Toc93074682)

[Hình 4.3 Mạch đồng 27](#_Toc93074683)

[Hình 4.4 Mạch báo cháy 28](#_Toc93074684)

[Hình 4.5 Hoạt động mạch báo cháy khi bình thường 29](#_Toc93074685)

[Hình 4.6 Mạch báo chạy khi cảnh báo 29](#_Toc93074686)

# **DANH MỤC BẢNG**

[Bảng 2.1 Chức năng của Pic 16F877A 6](#_Toc93074707)

[Bảng 2.2 Giá trị nhiệt độ 11](#_Toc93074708)

[Bảng 2.3 Bảng cấu hình độ phân giải và thời gian chuyển đổi tương đương 13](#_Toc93074709)

[Bảng 2.4 Các lệnh cơ bản của DS18B20 14](#_Toc93074710)

[Bảng 2.5 Mã hiển thị trên LED 7 đoạn Anode chung: 16](#_Toc93074711)

[Bảng 2.6 Mã hiển thị trên LED 7 đoạn Cathode chung: 16](#_Toc93074712)

[Bảng 3.1 Bảng giá trị thời gian 20](#_Toc93074717)

# ***CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI***

## *TÍNH CẤP THIẾT CỦA ĐỀ TÀI*

Với sự phát triển của công nghiệp hóa, hiện đại hóa cùng với sự cấp thiết về vấn đề an toàn trong gia đình. Thay vì trước kia phải đầu tư nguồn tiền lớn để có thể trang bị được những hệ thống báo cháy hiện đại. Mô hình thiết kế hệ thống báo cháy đơn giản bằng Pic 18F877A và cảm biến DS18B20 phù hợp cho những hệ thống báo cháy trong gia đình. Đề tài trình bài hệ thống báo cháy qua sử dụng các ngôn ngữ lập trình chương trình Pic C Compiler và những kiến thức cơ bản về vi điều khiển. Đề tài tìm hiểu về hệ thống báo cháy thông qua linh kiện cảm biến nhiệt độ DS18B20 được hiển thị lên LED 7 đoạn. Phương pháp thực hiện bằng phần mềm mô phỏng Proteus và phần mềm Pic C. Lí do em chọn đề tài lả vì đề tài có tính thực tế và có thể áp dụng trong thực tế tại nhà, trường học, bệnh viện,….

## *YÊU CẦU CỦA ĐỀ TÀI*

Lập trình trên vi điều khiển Pic 18F877A thiết kế mạch đo nhiệt độ môi trường, trong nhà, hiển thị nhiệt độ lên LED 7 đoạn và cảnh báo bằng chuông và đèn.

## *NỘI DUNG NGHIÊN CỨU*

* Tìm hiểu về linh kiện Pic 16F877A và cảm biến DS18B20
* Tìm hiểu về giao tiếp one wire và kỹ thuật quét LED

## *PHƯƠNG PHÁP THỰC HIÊN*

Đồ án được thực hiện dựa trên việc nhận và phân tích dữ liệu của vi điều khiển 18F877A. Đầu tiên, cảm biến nhiệt độ là DS18B20 sẽ tiếp nhận dữ liệu sau đó sẽ truyền dữ liệu vào trong PIC để phân tích. Khi nhiệt độ cảm biến từ DS18B20 cảm nhận được tăng hay giảm thì dữ liệu truyền về PIC cũng tăng hay giảm theo, tùy theo điều kiện được thiết lập cụ thể mà hệ thống sẽ thông báo bằng chuông hoặc đèn.

## *BỐ CỤC*

Với các mục tiêu, yêu cầu và nội dụng, báo cáo được phân thành các chương:

**Chương 1:** Giới thiệu đề tài

**Chương 2:** Thành phần linh kiện

**Chương 3:** Nguyên lí hoạt động

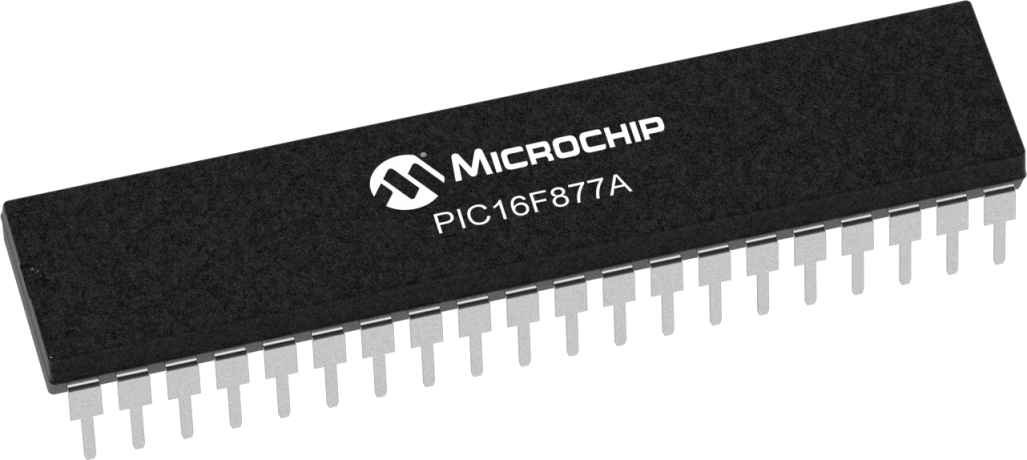
**Chương 4:** Thực hành và kết quả

**Chương 5:** Kết luận và hướng phát triển

# ***CHƯƠNG 2: THÀNH PHẦN LINH KIỆN***

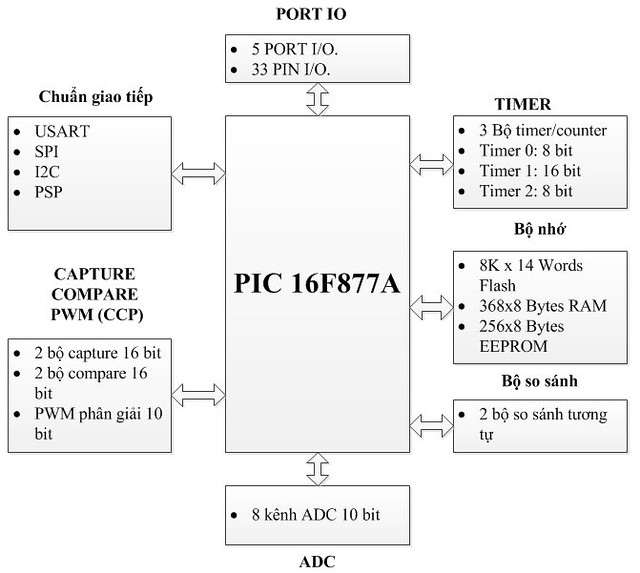
## *PIC16F877A*

Đây là một dòng vi điều khiển khá phổ biến trong ứng dụng thực tế hiện này, pic16F877A đầy đủ các chức năng, phù hợp với các ứng dụng cơ bản thích hợp để sử dụng trong các đồ án vừa và nhỏ của các môn học.



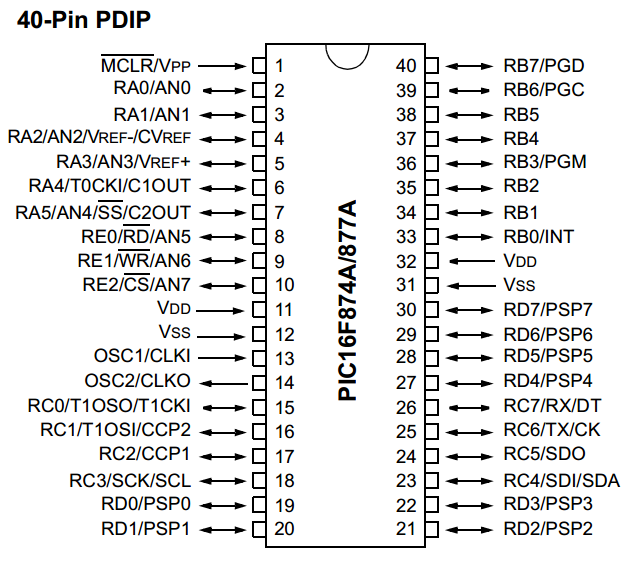
Hình 2.1 Pic 16F877A

* Một số đặc điểm của PIC16F877A:



Hình 2.2 Thành phần của Pic 16F877A

* PIC 16F877A trong mạch là loại có 40 chân,với 5 cổng xuất nhập là
  + Port A(RA0÷RA5)
  + Port B(RB0÷RB7)
  + Port C(RC0÷RC7)
  + Port D(RD0÷RD7)
  + Port E(RE0÷RE2).
* Có 3 bộ định thời là timer0,timer1,timer2 với 1 Timer 16 bits hoạt động với xung clock được cấp vào.
* Tốc độ hoạt động ở chế độ nhanh nhất có thể là 20Mhz.
* Điện áp cho phép có thể hoạt động được nằm trong khoảng từ 2(V) đến 5.5(V).
* Có 8 kênh chuyển đổi từ ANALOG sang DIGITAL
* Vi điều khiển có 8K bộ nhớ Flash nên có thể xóa và lập trình rất nhiều lần. Số lần tầm khoảng 100000 lần.
* Các chuẩn giao tiếp: USART, I2C, SPI, PSP



Hình 2.3 Sơ đồ chân Pic 16F877A

* Các cổng vào ra của pic:
* Port A: có 6 bit với 6 chân RA0÷RA5. Port A sẽ là cổng xuất nhập 2 chiều. Thanh ghi xác đinh chiều port A là thanh ghi TrisA. Các bit ở thanh ghi TrisA bằng 1 là nhập ngược lại sẽ là xuất.
* Port B: có 8 bit RB0-RB7 là một cổng xuất nhập 2 chiều. Thanh ghi qui đinh chiều của cổng B là thanh ghi Tris B.
* Port C: có 8 bit RC0-RC7 là một cổng xuất nhập 2 chiều, thanh ghi qui định chiều của cổng là thanh ghi TrisC. Các chân RC3, RC4 dùng để kết nối truyền nhận thông tin với các thiết bị ngoại vi, còn RC6, RC7 dùng để kết nối các thiết bị như module sim,...
* Port D: có 8 bit RD0-RD7, nó có thể là cổng nhập hoặc cổng xuất. Port D có thể được cấu hình như một cổng vi xử lý 8 bit.
* Port E: có 3 bit RE0-RE2, được cấu hình là xuất hoặc nhập. Port E được dùng làm đầu vào điều khiển I/O.
* Pic16F877A có 2 chân VCC và 2 chân GND ta phải cấp nguồn cho tất cả các chân này. Ngoài nguồn cấp chúng ta cần phải dùng một thạch anh 20MHz để cấp xung dao động cho Pic16F877A và được cấp thông qua 2 chân 13 và 14 của pic16F877A.
* chức năng của các chân pic 16F877A

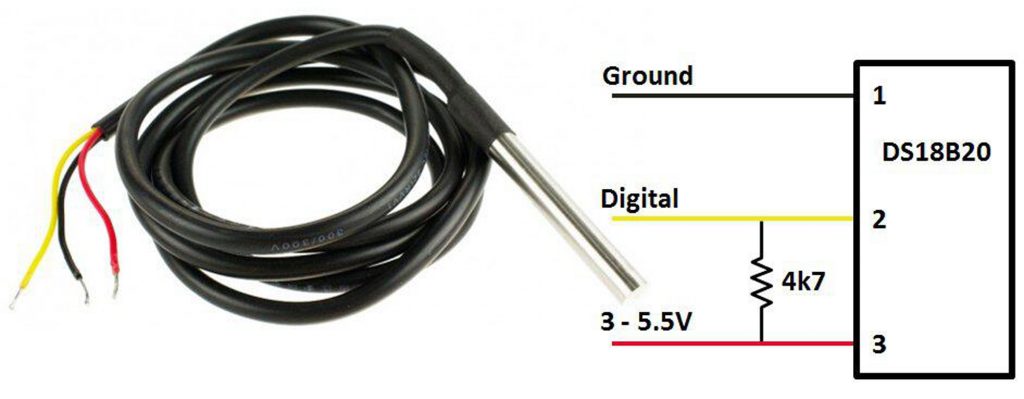
Bảng 2.1 Chức năng của Pic 16F877A

|  |  |  |
| --- | --- | --- |
| **Số chân** | **Kí hiệu** | **Tính năng** |
| 1 | MCLR/VPP | – MCLR: Chân Reset tích cực mức thấp – VPP : Điện áp ngõ vào |
| 2 | RA0/AN0 | – RA0 : I/O  – AN0 : Ngõ vào tương tự kệnh AN0 |
| 3 | RA1/AN1 | – RA1 : I/O – AN1 : Ngõ vào |
| 4 | RA2/AN2/VREF-/CVREF | – RA2 : I/O – AN2 : Ngõ vào – VREF -: Ngõ vào điện áp chuẩn mức thấp của bộ A/D |
| 5 | RA3/AN3/VREF+ | – RA3 : I/O – AN3 : Ngõ vào – VREF+ : Ngõ vào điện áp chuẩn mức cáo của bộ A/D |
| 6 | RA4/TOCKI/C1OUT | – RA4 : I/O – TOCKI : Ngõ vào xung clock cho timer0 – C1 OUT : Ngõ ra bộ so sánh thứ 1 |
| 7 | RA5/AN4//SS /C2OUT | – RA5 : I/O – AN4 : Ngõ vào – SS : Ngõ vào lựa chọn SPI phụ – C2 OUT : Ngõ ra bộ so sánh thứ 2 |
| 8 | RE0//RD/AN5 | – RE0 : I/O – RD : Điều khiển đọc port song song – AN5 : Ngõ vào |
| 9 | RE1//WR/AN6 | – RE1 :I/O – WR : Điều khiển ghi port song song – AN6 : Ngõ vào |
| 10 | RE2//CS/AN7 | – RE2 : I/O – CS : Chip chọn điều khiển port song song – AN7 : Ngõ vào |
| 11 | VDD | Chân nguồn VDD |
| 12 | VSS | Chân đất GND |
| 13 | OSC1/CLKI | – OSC1 : Ngõ vào dao động thạch anh hoặc xung clock bên ngoài. Ngõ vào Schmit trigger ở chế độ RC – CLKI : Ngõ vào xung bên ngoài. Được kết hợp với OSC1 |
| 14 | OSC2/CLKO | – OSC2 : Ngõ ra dao động thạch anh.  – CLKO : ở chế độ RC, ngõ ra của OSC2 bằng tần số của OSC1. |
| 15 | RC0/T1 OCO/T1CKI | – RC0 : I/O – T1OCO : Ngõ vào bộ dao động Timer 1 – T1CKI : Ngõ vào xung clock ngoài Timer 1 |
| 16 | RC1/T1OSI/CCP2 | – RC1 : I/O – T1OSI : Ngõ vào Timer 1 – CCP2 : Ngõ vào Capture 2, ngõ ra compare 2 và PWM2 |
| 17 | RC2/CCP1 | – RC2 : I/O – CCP1 : Ngõ vào Capture 1, ngõ ra compare 1 và PWM1 |
| 18 | RC3/SCK/SCL | – RC3 : I/O – SCK : Ngõ vào xung clock nối tiếp của chế độ SPI – SCL : Ngõ vào xung clock nối tiếp của chế độ I2C |
| 19 | RD0/PSP0 | – RD0 : I/O – PSP0 : Dữ liệu port nhánh song song |
| 20 | RD1/PSP1 | – RD1 : I/O – PSP1 : Dữ liệu port nhánh song song |
| 21 | RD2/PSP2 | – RD2 : I/O – PSP2 : Dữ liệu port nhánh song song |
| 22 | RD3/PSP3 | – RD3: I/O – PSP3 : Dữ liệu port nhánh song song |
| 23 | RC4/SDI/SDA | – RC4 : I/O – SDI : Dữ liệu vào SPI – SDA : Dữ liệu xuất/nhập vào I2C |
| 24 | RC5/SDO | – RC5 : I/O – SDO : Dữ liệu ra SPI |
| 25 | RC6/TX/CK | – RC6 : I/O – TX : Truyền dữ liệu bất đồng bộ USART – CK : Xung đồng bộ USART |
| 26 | RC7/RX/DT | – RC7 : I/O – RX : Nhận dữ liệu bất đồng bộ USART – DT : Dữ liệu đồng bộ USART |
| 27 | RD4/PSP | – RD4: Xuất/nhập số – PSP4 : Dữ liệu port nhánh song song |
| 28 | RD5/PSP5 | – RD5: I/O xuất/nhập – PSP5 : Dữ liệu port nhánh song song |
| 29 | RD6/PSP6 | – RD6: I/O xuất/nhập – PSP6 : Dữ liệu port nhánh song song |
| 30 | RD7/PSP7 | – RD7: I/O – PSP7 : Dữ liệu port nhánh song song |
| 31 | VSS | Chân nối GND |
| 32 | VDD | Chân nguồn VDD |
| 33 | RB0/INT | – RB0 : I/O – INT : Ngắt ngoài |
| 34 | RB1 | – I/O |
| 35 | RB2 | –I/O |
| 36 | RB3 | – RB3 : I/O – Chân cho phép lập trình điện áp thấp ICPS |
| 37 | RB4 | – I/O – Ngắt Port B |
| 38 | RB5 | – I/O – Ngắt Port B |
| 39 | RB6/PGC | – RB6 : I/O – PGC : Xung clock lập trình ICSP – Ngắt Port B |
| 40 | RB7/PGD | – RB7 : I/O – PGD :Dữ liệu lập trình ICSP – Ngắt Port B |

## *DS18B20*

### *Giới thiệu*

Cảm biến nhiệt độ DS18B20 là một loại cảm biến tương tự  được ứng dụng trong các ứng dụng đo nhiệt độ thực với độ phân giải cao ( 12bit ). Vì DS18B20 có kích thước nhỏ gọn giá thành rẻ, IC sử dụng giao tiếp 1 dây rất gọn gàng, dễ lập trình và hoạt động khá chính xác có sai số khá nhỏ. Đặc biệt, IC còn có chức năng cảnh báo nhiệt độ khi vượt ngưỡng và đặc biệt hơn là có thể cấp nguồn từ chân data ( parasite power ).

****

Hình 2.4 Cảm biến nhiệt DS18B20

* Chân 1: Chân GND
* Chân 2: Đầu ra output
* Chân 3: chân VCC (3 – 5.5V)

### *Đặc tính và thông số của cảm biến DS18B20:*

* Điện áp đầu vào từ 3V đến 5.5V
* Dải đo nhiệt độ từ -55 oC đến 125 oC
* Sai số oC khi đo ở dải -10 – 85 oC
* Có cảnh báo nhiệt khi vượt ngưỡng cho phép và có thể cấp nguồn từ chân data.
* Độ phân giải: 9 – 12 bits
* Mỗi cảm biến đều có một mã định danh riêng nên có thể giao tiếp nhiều DS18B20 cùng một lúc.

### *Mã ROM 64 bit:*

* Mỗi linh kiện cảm biến DS18B20 đều có một mã định danh 64 bit duy nhất được lưu trong ROM và phân phối như nhau:
* 8 bit đầu tiên chứa đựng mã quy ước. DS18B20 với mã là: 28h.
* 48 bit tiếp theo là serial number của thiết bị.
* 8 bit cuối cùng là mã kiểm tra và phát hiện lỗi CRC cho 56 bit trước.

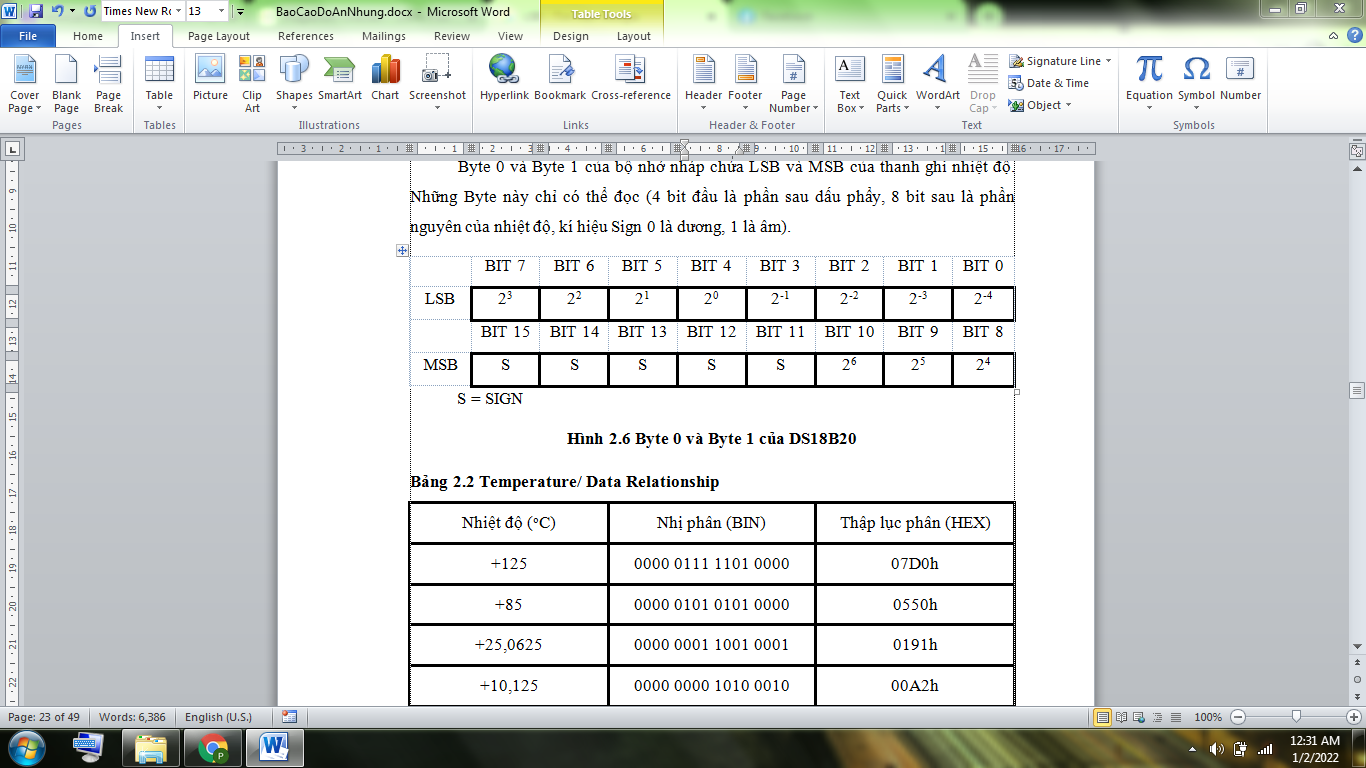


Hình 2.5 Mã ROM 64 bit

### *Bộ nhớ*

Bộ nhớ của DS18B20 bao gồm bộ nhớ nháp (SRAM scratchpad), thanh ghi lưu trữ kích hoạt cảnh báo cao (TH), thấp (TL) và thanh ghi cấu hình. Hai thanh ghi này đều trang bị bộ nhớ EEPROM. Thanh ghi TH và TL có thể được sử dụng như bộ nhớ đa mục đích nếu chức năng cảnh báo không được sử dụng.

Byte 0 và Byte 1 của bộ nhớ nháp chứa LSB và MSB của thanh ghi nhiệt độ. Những Byte này chỉ có thể đọc (4 bit đầu là phần sau dấu phẩy, 8 bit sau là phần nguyên của nhiệt độ, kí hiệu Sign 0 là dương, 1 là âm).

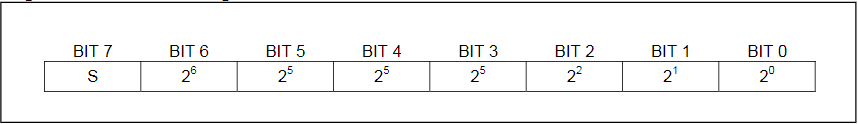


Hình 2.6 Byte 0 và Byte 1 của DS18B20

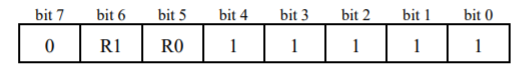
Bảng 2.2 Giá trị nhiệt độ

|  |  |  |
| --- | --- | --- |
| Nhiệt độ (oC) | Nhị phân (BIN) | Thập lục phân (HEX) |
| +125 | 0000 0111 1101 0000 | 07D0h |
| +85 | 0000 0101 0101 0000 | 0550h |
| +25,0625 | 0000 0001 1001 0001 | 0191h |
| +10,125 | 0000 0000 1010 0010 | 00A2h |
| +0,5 | 0000 0000 0000 1000 | 0008h |
| 0 | 0000 0000 0000 0000 | 0000h |
| -0,5 | 1111 1111 1111 1000 | FFF8h |
| -10,125 | 1111 1111 0101 1110 | FF5Eh |
| -25,0625 | 1111 1110 0110 1111 | FE6Fh |
| -55 | 1111 1100 1001 0000 | Fc90h |

Byte 2 và 3 chứa thanh ghi TH và TL để cảnh báo alarm. Bit 4 đến bit 11 của byte 0, 1 được dùng để so sánh với thanh ghi TH vs TL. Nếu nhiệt độ nhỏ hơn hoặc bằng TL hoặc lớn hơn hoặc bằng TH, Alarm sẽ được thiết lập bên cạnh DS18B20



Hình 2.7 Thanh ghi TL và TH



Hình 2.8 Byte 4 thanh ghi cấu hình

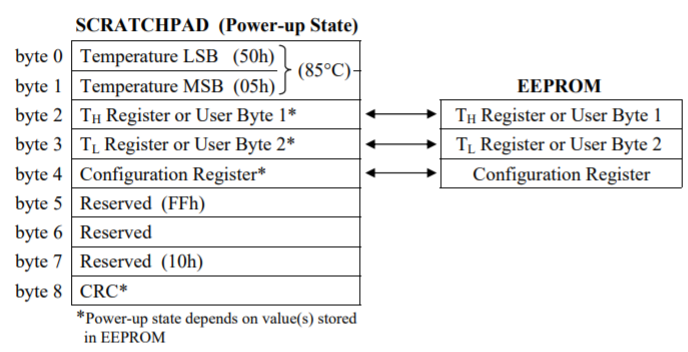
Byte 4 thanh ghi cấu hình. Có thể được dùng để thiết lập độ phân giải của DS18B20 sử dụng bit 5 (R0) và bit 6 (R1). Khi cấp nguồn R0, R1 mặc định giá trị 1 (độ phân giải 12 bit). Bit 0 đến 4 và bit 7 được để dành riêng cho thiết bị và không thể ghi đè

Bảng 2.3 Bảng cấu hình độ phân giải và thời gian chuyển đổi tương đương

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **R1** | **R0** | **Độ phân giải (bit)** | **Thời gian chuyển đổi tối đa** | |
| 0 | 0 | 9 bit | 93.75 ms | (tCONV/8) |
| 0 | 1 | 10 bit | 187.5 ms | (tCONV/4) |
| 1 | 0 | 11 bit | 375 ms | (tCONV/2) |
| 1 | 1 | 12 bit | 750 ms | (tCONV) |

Byte 5, 6 và 7 được dành riêng để các thiết bị sử dụng, nên ta không thể ghi lên các byte này

Byte 8 chứa đựng mã CRC của byte 0 đến 7 của bộ nhớ nháp.



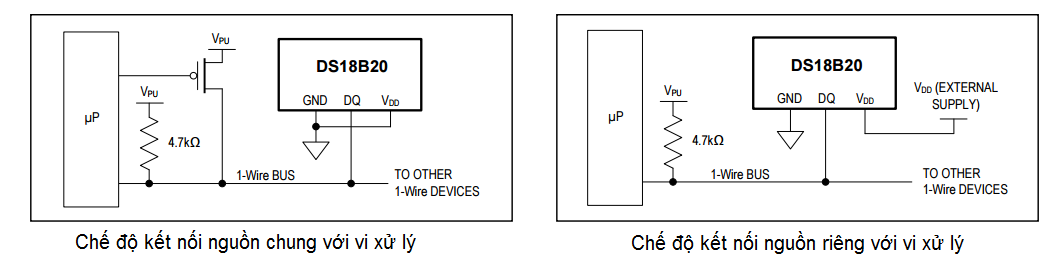
Hình 2.9 Bộ nhớ SRAM

Khi bị ngắt nguồn cấp, dữ liệu được lưu trong các thanh ghi EEPROM sẽ không bị mất đi, khi có nguồn cấp lại dữ liệu trong các thanh ghi sẽ được nạp vào bộ nhớ nháp theo vị trí byte tương ứng. Dữ liệu này được nạp lại từ EEPROM bằng lệnh E2B8h

Bảng 2.4 Các lệnh cơ bản của DS18B20

|  |  |  |
| --- | --- | --- |
| **Các lệnh chức năng ROM** | **Mã HEX** | **Chức năng** |
| **Read ROM** | 0x33 | Lệnh đọc 64 bit mã ROM của DS18B20 và chỉ được sử dụng khi có duy nhất 1 cảm biến DS18B20. |
| **Match ROM** | 0x55 | Lệnh tìm tìm địa chỉ ROM chính xác của 1 DS18B20 khi có nhiều DS18B20. DS18B20 có 64 bit phù hợp với chuỗi 64 bit vừa được gửi tới thì phản hồi lại các lệnh về bộ nhớ tiếp theo. Ngược lại, sẽ tiếp tục chờ một xung reset. |
| **Skip ROM** | 0xCC | Lệnh này cho phép Master gửi lệnh đến tất cả các DS18B20 mà không cần định danh. |
| **Search ROM** | 0xF0 | Lệnh dùng để xác định số DS18B20 trên bus và xác định mã ROM tương ứng. |
| **Alarm Search** | 0xEC | Cảnh báo khi nhiệt độ vượt ngưỡng điều kiện cảnh báo |
| **Write Scratchpad** | 0x4E | Ghi dữ liệu vào bộ nhớ nháp |
| **Read Scratpad** | 0xBE | Đọc dữ liệu nhiệt độ trong bộ nhớ nháp |
| **Convert T** | 0x44 | Lệnh này thực hiện quá trình đo và chuyển đổi giá trị nhiệt độ thành số nhị phân lưu trữ trên thanh ghi nhiệt độ 2 byte trong bộ nhớ nháp. Trong thời gian đang chuyển đổi, giá trị sẽ trả về 0 nếu thực hiện lệnh đọc |

### ***Kiểu kết nối với vi xử lý***



Hình 2.10 Kiểu kết nối của DS18B20

Chúng ta cần một điện trở treo 4.7kOhm lên nguồn Vcc với chân Data của DS18b20 như hình 2.10.

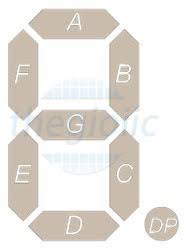
## *LED*

LED (Light Emitting Diode) là linh kiện được sử dụng rộng rãi trong các ứng dụng của Vi điều khiển và trong thực tế. LED có 7 đoạn được quy định: a, b, c, d, e, f, g và điểm chấm DP với mỗi đoạn là một đoạn LED.

LED 7 đoạn có 2 loại: Anode chung và Cathode chung:

* Anode chung: Điện áp bằng 0 sẽ kích hoạt đèn sáng
* Cathode chung: Điên áp bằng 1 sẽ kích hoạt đèn sáng

Vì LED 7 đoạn chứa các led đơn, do đó khi kết nối để bảo vệ led thì phải đảm bảo dòng qua mỗi led đơn trong khoảng 10mA-20mA.



Hình 2.11 LED 7 đoạn

Bảng 2.5 Mã hiển thị trên LED 7 đoạn Anode chung:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Số hiển thị** | **DP** | **G** | **F** | **E** | **D** | **C** | **B** | **A** | **HEX** |
| **0** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | **0x40** |
| **1** | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | **0x79** |
| **2** | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | **0x24** |
| **3** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | **0x30** |
| **4** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | **0x19** |
| **5** | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | **0x12** |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | **0x02** |
| **7** | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | **0x78** |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0x00** |
| **9** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | **0x10** |

Bảng 2.6 Mã hiển thị trên LED 7 đoạn Cathode chung:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Số hiển thị** | **DP** | **G** | **F** | **E** | **D** | **C** | **B** | **A** | **HEX** |
| **0** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | **0x3F** |
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | **0x06** |
| **2** | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | **0x5B** |
| **3** | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | **0x4F** |
| **4** | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | **0x66** |
| **5** | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | **0x6D** |
| **6** | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | **0x7D** |
| **7** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | **0x07** |
| **8** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **0x7F** |
| **9** | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | **0x6F** |

## *THẠCH ANH 20MHZ*

Vi điều khiển sử dụng linh kiện thạch anh 20Mhz để tạo xung dao động cho PIC 16F877A hoạt động, thạch anh được nối với 2 tụ gốm 33pF. Hai chân ra của thạch anh được nối trực tiếp vào chân 13(OSC1) và chân 14(OSC2) để tạo xung truyền vào cho PIC16F877A làm việc.

****

Hình 2.12 Linh kiện thạnh anh

# ***CHƯƠNG 3: NGUYÊN LÍ HOẠT ĐỘNG***

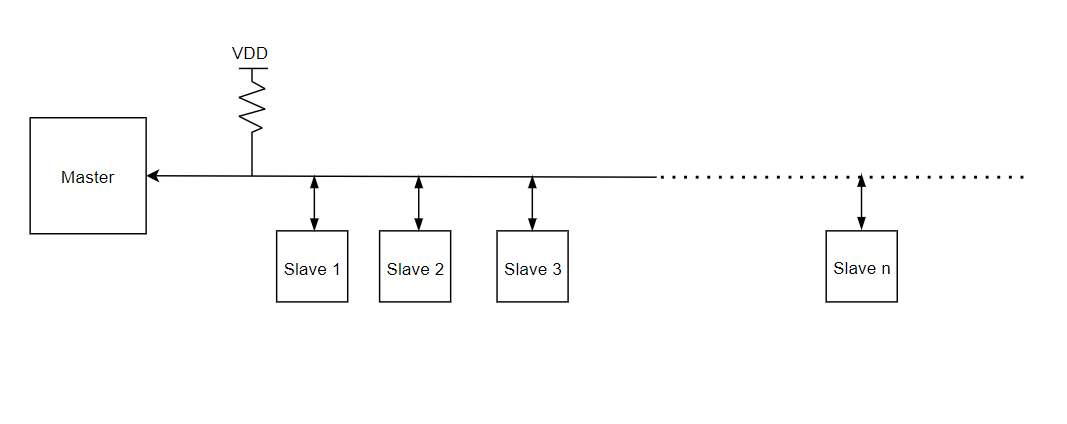
## *LÝ THUYẾT*

* Mạch sử dụng vi điều khiển 16F877A để thực hiện làm bộ xử lý trung tâm cho mạch. Thiết bị cảm biến nhiệt độ DS18B20 sẽ thu thập dữ liệu nhiệt độ từ môi trường sau đó sẽ gửi dữ liệu về cho bộ xử lý trung tâm đó là PIC16F877A bằng giao tiếp 1 WIRE của DS18B20. Sau khi nhận được, dữ liệu sẽ được phân tích chuyển đổi và xuất lên LED 7 đoạn.
* Trong trường hợp nếu như nhiệt độ đo được lớn hơn so với ngưỡng an toàn được thiết lập trong khi lập trình cho vi điều khiển thì sẽ kích hoạt báo cháy bằng chuông và led.

## *GIAO TIẾP GIỮA CÁC LINH KIỆN*

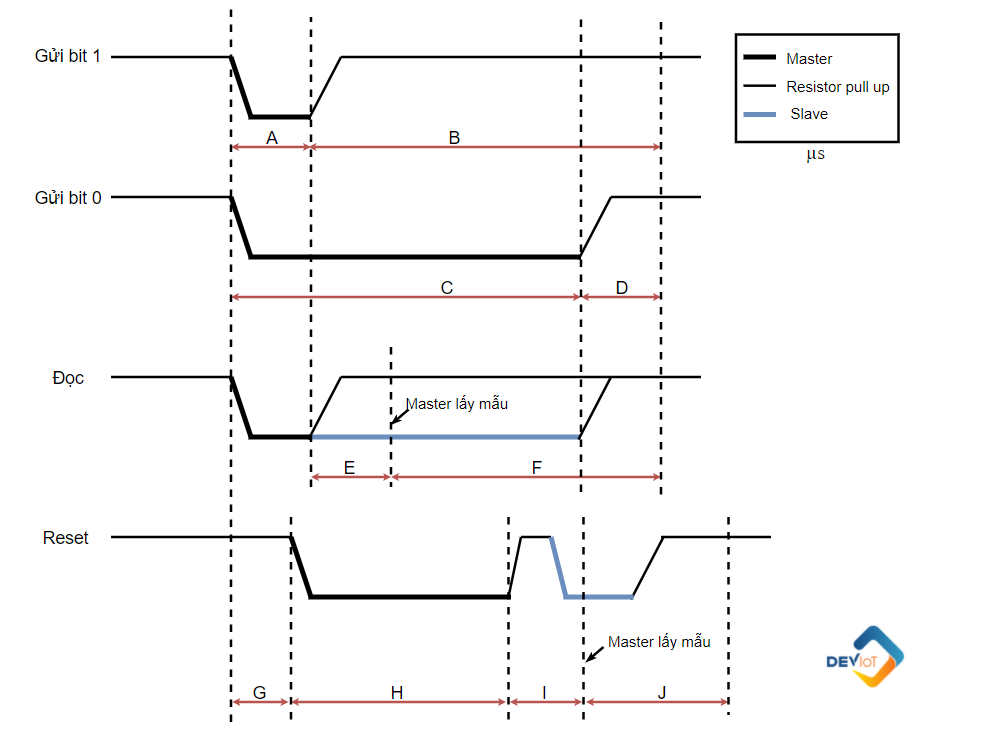
### *Giao tiếp One - Wire*

* **Giới thiêu:**
* One-wire là hệ thống bus giao tiếp được thiết kế bởi Dallas Semiconductor Corp và chỉ sử dụng 1 dây để truyền nhận dữ liệu.
* One-wire được dùng để giao tiếp với các thiết bị nhỏ, nhiệt độ,… các việc không yêu cầu tốc độ cao vì one-wire có tốc độ truyền thấp nhưng dữ liệu lại truyền được khoảng cách xa hơn do chỉ sử dụng 1 dây.
* One-wire cho phép truyền nhận dữ liệu với nhiều Slave trên đường truyền nhưng chỉ có thể có 1 Master.

****

Hình 3.1 Giao tiếp One-wire

* **Cơ sở truyền nhận**
* Đường dây dữ liệu luôn được giữ ở mức cao (High).
* Các thao tác hoạt động cơ bản của bus sẽ được quy định bởi thời gian kéo đường truyền xuống mức thấp (Low). Có 4 thao tác cơ bản như hình vẽ dưới.



Hình 3.2 Cơ sở truyền nhận của DS18B20

* **Giải thích ý nghĩa :**
* Gửi bit 1: Thiết bị Master sẽ kéo bus xuống mức 0 trong một khoảng thời gian A (µs) và trở về mức 1 trong khoảng B (µs).
* Gửi bit 0: Thiết bị Master kéo bus xuống mức 0 trong một khoảng thời gian C (µs) và trở về mức 1 trong khoảng D (µs).
* Đọc bit: Thiết bị Master kéo bus xuống khoảng A (µs). Trong khoảng thời gian E (µs), nếu bus ở mức 1, thiết bị master sẽ đọc bit 1 và master sẽ đọc được bit 0 nếu bus ở mức 0.
* Reset:  Thiết bị Master kéo bus xuống khoảng H (µs) và sau đó về mức 1. Trong khoảng thời gian I (µs) kế tiếp, nếu thiết bị Slave gắn với bus gửi về tín hiệu 0, Master sẽ hiểu rằng Slave vẫn có mặt và tiếp tục quá trình trao đổi dữ liệu. Ngược lại, tín hiệu 1 thì Master hiểu rằng không có thiết bị Slave tồn tại và dừng quá trình.
* **Đặc tính**
* Chuẩn One-Wire cần chính xác nhất là thời gian. Vì vậy cần một bộ định thời để delay chính xác nhất để tối ưu đường truyền.
* Bus dữ liệu khi ở trạng thái chờ phải ở mức cao do vậy bus dữ liệu phải được kéo lên nguồn thông qua một điện trở.
* Thông thường sẽ sử dụng mode Standard .

Bảng 3.1 Bảng giá trị thời gian

|  |  |  |
| --- | --- | --- |
| Thước đo | Tốc độ | Thời gian đề xuất (µs) |
| A | Standard | 6 |
| Overdrive | 1 |
| B | Standard | 64 |
| Overdrive | 7.5 |
| C | Standard | 60 |
| Overdrive | 7.5 |
| D | Standard | 10 |
| Overdrive | 2.5 |
| E | Standard | 9 |
| Overdrive | 1 |
| F | Standard | 55 |
| Overdrive | 7 |
| G | Standard | 0 |
| Overdrive | 2.5 |
| H | Standard | 480 |
| Overdrive | 70 |
| I | Standard | 70 |
| Overdrive | 8.5 |
| J | Standard | 410 |
| Overdrive | 40 |

* **Chế độ hoạt động:**
* Chế độ Standard - Chế độ tiêu chuẩn:
  + 15.4 Kb/s
  + 65 µs bit
* Chế độ Overdrive – Chế độ tốc độ nhanh:
  + 125 Kb/s
  + 8 µs bit

### *Kỹ thuật quét LED*

Kĩ thuật quét LED là loại bỏ những hạn chế khi nhìn hình ảnh qua mắt người. Qua cái nhìn của mắt, chuỗi hình ảnh sẽ liên tục nếu nó xảy ra với tốc độ 24 khung hình/s. Kĩ thuật này nối các chân dữ liệu của nhiều LED 7 đoạn lại với nhau và thực hiện sáng tắt tại một thời điểm chỉ có duy nhất 1 LED sáng. Khoảng thời gian sáng tắt LED rất ngắn nên qua mắt người đèn sáng sẽ cùng lúc và liên tục. Thời gian sáng tắt phải nhỏ hơn 1/24 giây, nghĩa là 1 giây sẽ có nhiều hơn 24 lần chuyển giữa các LED.

## *LƯU ĐỒ GIẢI THUẬT*

Khởi tạo giao tiếp 1 Wire

Yêu cầu dữ liệu từ cảm biến   
DS18B20

Hiển thị nhiệt độ đo được lên LED

SAI

Kiểm tra điều kiện nhiệt độ yêu cầu

ĐÚNG

LED hiển thị nhiệt độ

LED, BUZZ báo hiệu

## *SƠ ĐỒ KHỐI*

**Khối nguồn**

Nguồn 5V

**Khối hiển thị**

LED 7 ĐOẠN

**Khối cảm biến**

DS18B20

**Khối điều khiển**

PIC16F877A

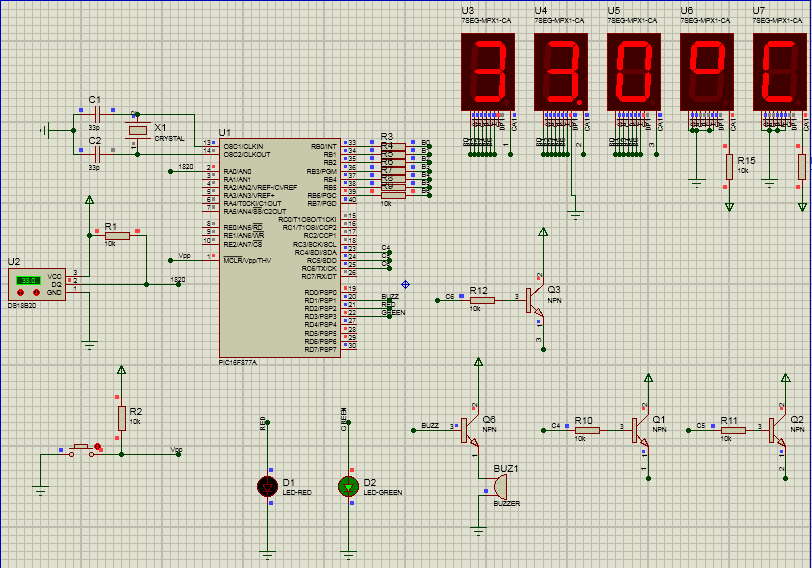
**Khối hiển thị**

LED, BUZZ

* **Khối nguồn:** Cấp nguồn 5V vào khối điều khiển.
* **Khối cảm biến:** Nhận dữ liệu từ môi trường và truyền về khối điều khiển.
* **Khối điều khiển:** Giao tiếp với khối cảm biến để phân tích dữ liệu từ khối cảm biến.
* **Khối hiển thị:** Nhận dữ liệu từ khối điều khiển và xuất lên dữ liệu theo yêu cầu.

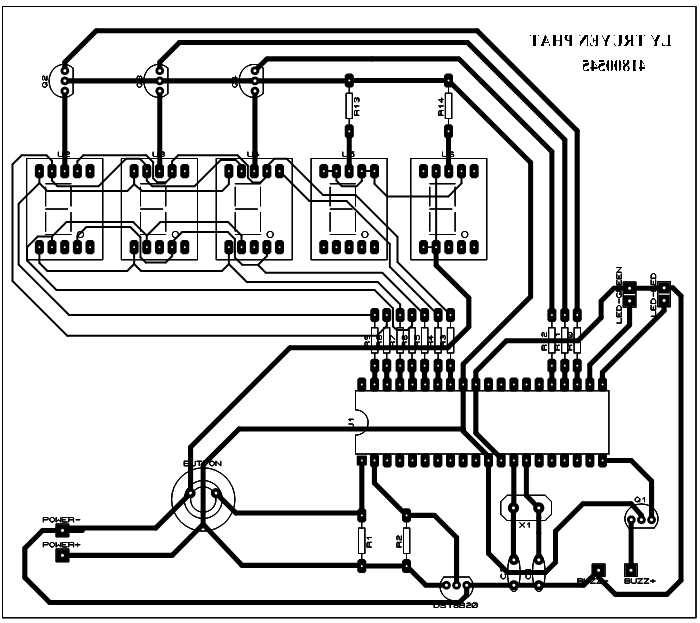
# ***CHƯƠNG 4: THỰC HÀNH VÀ KẾT QUẢ***

## *SƠ ĐỒ MẠCH MÔ PHỎNG*



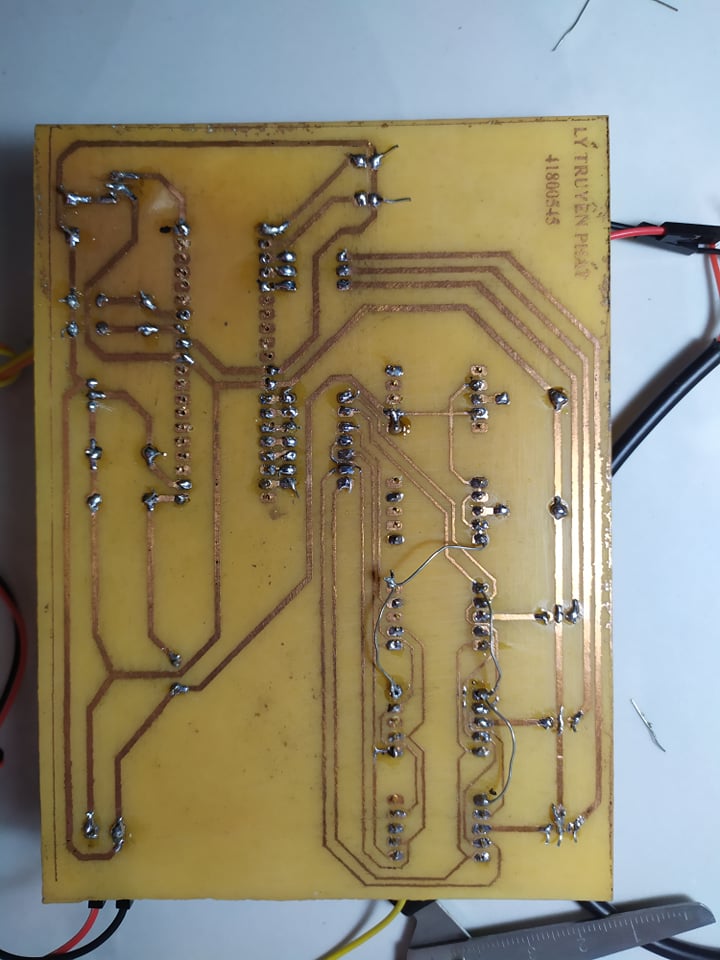
Hình 4.1 Sơ đồ mô phỏng

## *MẠCH IN*



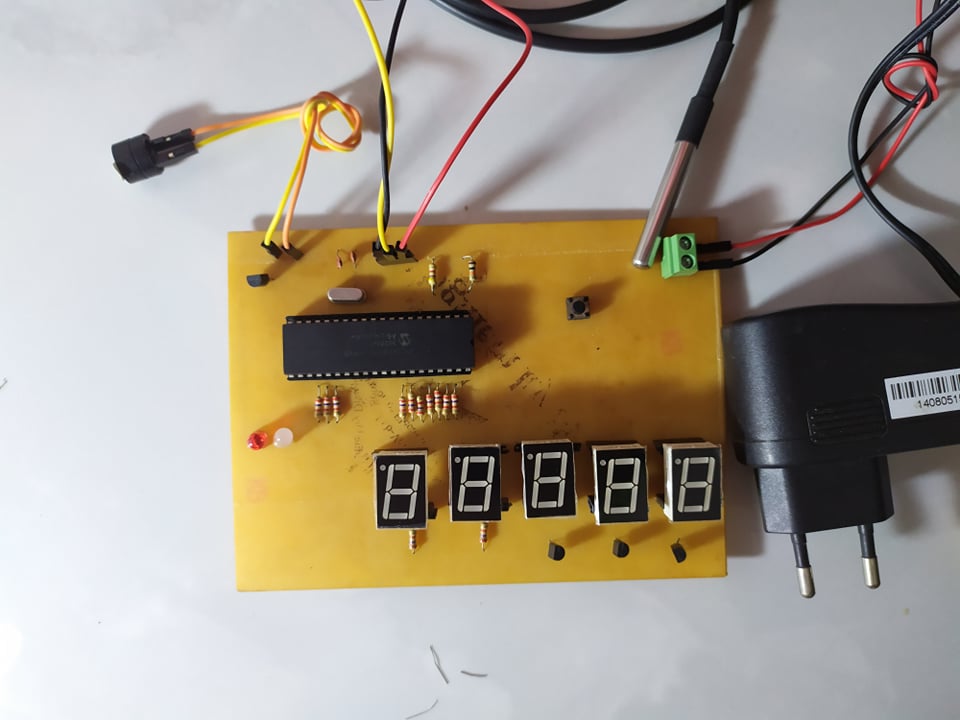
Hình 4.2 Sơ đồ mạch in

## *MẠCH ĐỒNG*



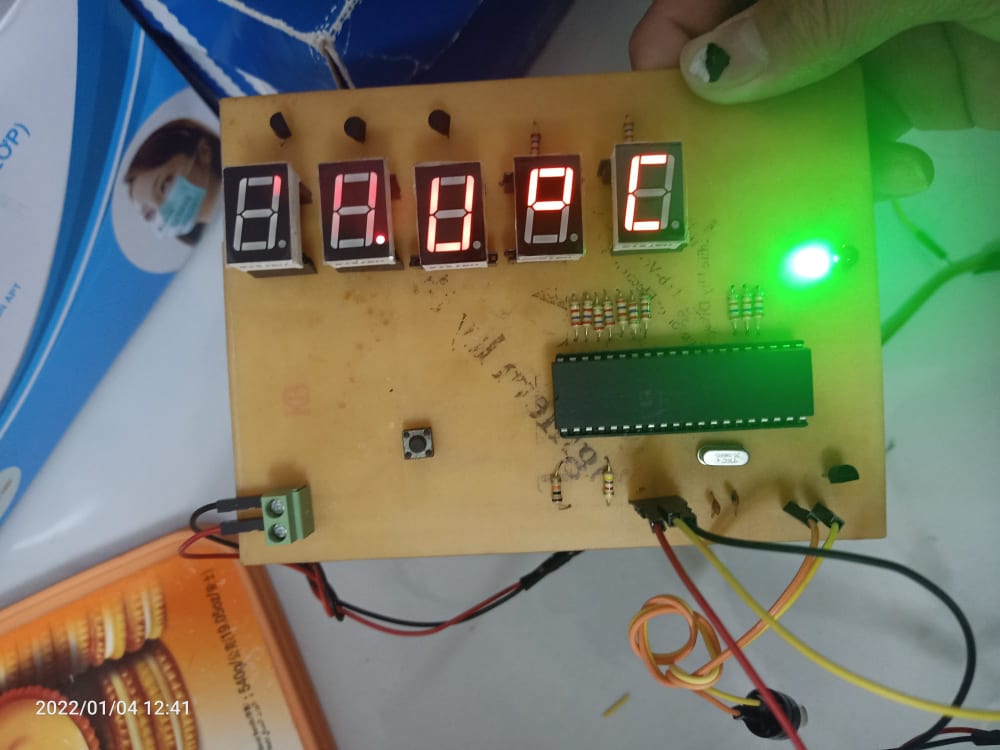
Hình 4.3 Mạch đồng

## *MẠCH HOÀN CHỈNH*

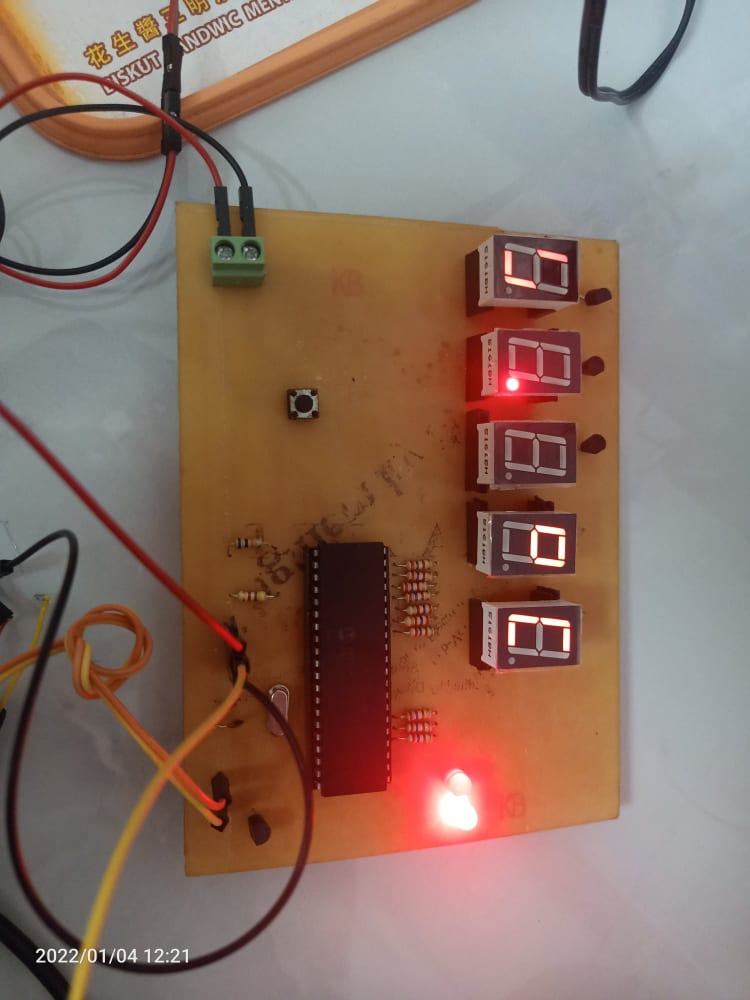
****

Hình 4.4 Mạch báo cháy

1. ***KẾT QUẢ THỰC HIỆN***

******

Hình 4.5 Hoạt động mạch báo cháy khi bình thường



Hình 4.6 Mạch báo chạy khi cảnh báo

# ***CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN***

## *ƯU ĐIỂM*

Mạch chạy tương đối ổn định, thiết kế mạch đơn giản dễ hiểu và cảm biến được sử dụng có độ chính xác cao, LED hiển thị rõ chính xác nhiệt độ đo được từ môi trường. Cảnh báo nhiệt độ chính xác, chuông và đèn cảnh báo hoạt động tốt đảm bảo an toàn cho ngôi nhà.

Bản thân tích lũy được nhiều kinh nghiệm về lập trình, thông tin các linh kiện cách thức hoạt động cũng như kinh nghiệm về cách xây dựng phần cứng trên board. Ngoài ra mạch có chi phí lắp đặt thấp lại dễ sử dụng.

## *NHƯỢC ĐIỂM*

Tuy hoạt động khá chính xác nhưng chỉ có thể đo được duy nhất nhiệt độ môi trường. LED lại hiển thị hạn chế nên có thể thay thế bằng LCD và thêm cảm biến. Độ bền của các linh kiện lại không cao. Vì sử dụng 4 LED nên phần đi dây khá phức tạp rất dễ nhầm và phải đi dây ngoài.

## *KẾT LUẬN*

Sau khi tiến hành mô phỏng và thực nghiệm trên mạch thực tế, có thể thấy mạch hiển thị được nhiệt độ trên LED 7 đoạn giống với mô phỏng và có thể thực hiện cảnh báo nhiệt độ giới hạn bằng chuông và đèn. Từ đó cho thấy mạch chạy khá ổn định không gặp sự cố nào quá lớn và nhiệt độ hiển thị cũng như cảnh chính xác so với thực tế đo được và giống với mô phỏng thực hiện trên ứng dụng lập trình.

## *HƯỚNG PHÁT TRIỂN ĐỒ ÁN*

Xét về tính ứng dụng của mạch vào thực tế thì chưa thực sự thuận tiện vì hiện thị qua LED chỉ hiện thị được nhiệt độ và phải thi công nhiều LED đi dậy khá phức tạp dẫn đến dễ sai sót trong cách làm thủ công. Vì thế để tiếp tục phát triển đề tài hơn nữa có thể dùng dụng thêm một vài linh kiện khác như cảm biến khói, cảm biến khí gas và hiển thị nhiệt độ lên LCD để có thể hiển thị nhiều thông tin hơn, thiết kế thêm kích hoạt hệ thống phun nước tự động hoặc có thể gửi tin nhắn cảnh báo thông qua Module SMS. Người sử dụng có thể yên tâm hơn khi có cảnh báo từ xa những trường hợp không có ai ở tại nhà từ đó giảm được tối đa thiệt hại như vậy sẽ giúp cho đời sống con cải thiện hơn.

# ***TÀI LIỆU THAM KHẢO***

Các nguồn sử dụng cho bài báo cáo:

Nguyễn Đình Phú, Trường Ngọc Anh. “Giáo trình vi xử lý”, Nhà xuất bản Đại học Sư  
phạm kỹ thuật, 2016: http://thuvienso.hcmute.edu.vn/doc/giao-trinh-vi-xu-ly-  
143260.html

PIC16F877A:

[www.alldatasheet.com/datasheet-pdf/pdf/82338/MICROCHIP/PIC16F877A.html](http://www.alldatasheet.com/datasheet-pdf/pdf/82338/MICROCHIP/PIC16F877A.html)

DS18B20:

[pdf1.alldatasheet.com/datasheet-pdf/view/230838/DALLAS/DS18B20.html](https://pdf1.alldatasheet.com/datasheet-pdf/view/230838/DALLAS/DS18B20.html)

[hqdt.vn/baiviet/co-ban-ve-ds18b20-1498273365.html](https://hqdt.vn/baiviet/co-ban-ve-ds18b20-1498273365.html)

LED:

[univtechnews.wordpress.com/2017/03/29/hien-thi-led-7-doan-su-dung-pic16f877a/](https://univtechnews.wordpress.com/2017/03/29/hien-thi-led-7-doan-su-dung-pic16f877a/)

Giao tiếp one-wire:

[cypress.com/file/127331/download](https://www.cypress.com/file/127331/download)

[deviot.vn/blog/giao-tiep-onewire.72581713](https://deviot.vn/blog/giao-tiep-onewire.72581713)

# ***PHỤ LỤC A: ONE WIRE***

#ifndef ONE\_WIRE\_C

#define ONE\_WIRE\_C

#define ONE\_WIRE\_PIN PIN\_A0

void onewire\_reset() *// Hàm dùng để khởi động one wire // OK if just using a single permanently connected device*

{

output\_low(ONE\_WIRE\_PIN); *// kéo bus xuống mức thấp để reset*

delay\_us(500);

output\_float(ONE\_WIRE\_PIN); *// đẩy bus lên mức cao*

delay\_us(500); *// chờ khởi động*

output\_float(ONE\_WIRE\_PIN);

}

void onewire\_write(int8 data) // Viết 1 byte của dữ liệu vào thiết bị

{

int8 count;

for(count = 0; count < 8; ++count)

{

output\_low(ONE\_WIRE\_PIN);

delay\_us(2); *// Kéo one wire xuống thấp để khởi động khe ghi thời gian*

output\_bit(ONE\_WIRE\_PIN, shift\_right(&data, 1, 0)); *// Đặt bit đầu ra cho one* wire // Dịch phải 8 lần để ghi 8 bit dữ liệu

delay\_us(60); *// Chờ đến khi kết thúc khe ghi*

output\_float(ONE\_WIRE\_PIN); *// Đặt one wire ở mức cao*

delay\_us(2);

}

}

int onewire\_read() // Đọc và trả dữ liệu từ thiết bị

{

int count, data;

for(count = 0; count < 8; ++count)

{

output\_low(ONE\_WIRE\_PIN);

delay\_us(2); *// Kéo one wire xuống thấp để khởi động khe đọc thời gian*

output\_float(ONE\_WIRE\_PIN); *// Đặt one wire ở mức cao*

delay\_us(8); *// Để trạng thái thiết bị ổn định*

shift\_right(&data, 1, input(ONE\_WIRE\_PIN)); *// Dịch phải ghi kết quả*

delay\_us(120); *// Chờ đến khi kết thúc khe đọc*

}

return data;

}

# ***PHỤ LỤC B: DS18B20***

#define DS18B20\_C

#include "1\_day.c"

float ds18b20\_read();

float ds18b20\_read() // Đọc dữ liệu của DS18B20 trên bus one – wire và trả kết quả nhiệt độ

{

int8 busy=0, temp1, temp2;

signed int16 temp3;

float result;

onewire\_reset();

onewire\_write(0xCC); // Skip ROM, định danh địa chỉ tất cả thiết bị

onewire\_write(0x44); // Bắt đầu chuyển đổi nhiệt độ

while(busy == 0) // Nếu busy ở mức thấp quá trình tiếp tục

{

busy = onewire\_read();

onewire\_reset();

onewire\_write(0xCC); //Skip ROM, định danh địa chỉ tất cả thiết bị

onewire\_write(0xBE); //Đọc scratchpad

temp1 = onewire\_read(); //Phần nguyên nhiệt độ

temp2 = onewire\_read(); //Phân thập phân nhiệt độ

temp3 = make16(temp2, temp1);

//result = (float) temp3 / 2.0; //Calculation for DS18S20 with 0.5 deg C resolution

result = (float) temp3 / 16.0; //Calculation for DS18B20 with 0.1 deg C resolution

delay\_ms(200);

return(result);

}

}

void ds18b20\_configure(int8 TH, int8 TL, int8 config) // Ghi cấu hình cho DS18B20

{

onewire\_reset();

onewire\_write(0xCC); //Skip ROM, định danh tất cả thiết bị

onewire\_write(0x4E); //Ghi scratchpad

onewire\_write(TH); //Ghi cảnh báo mức cao

onewire\_write(TL); // Ghi cảnh báo mức thấp

onewire\_write(config); // Ghi cấu hình DS18B20

}

# ***PHỤ LỤC C: CODE***

#include <16F877A.h>

#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT

#use delay(clock=20000000)

#include "1\_day.c" // Khai báo thư viện one wire

#include "DS18B20.c" // Khai báo thư viện DS18B20

const char led7[10]={0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10};

float temperature = 0.0;

int nguyen = 0;

double du = 0;

void main()

{

set\_tris\_B(0x00);

set\_tris\_D(0x00);

set\_tris\_A(0xFF);

while(true)

{

temperature=ds18b20\_read(); //////// đọc giá trị nhiệt độ từ ds18b20

nguyen = temperature;

du = temperature - nguyen;

output\_B(led7[nguyen/10]); //////// đọc giá trị hàng chục

output\_high(pin\_C4);

delay\_ms(2);

output\_low(pin\_C4);

output\_B(led7[nguyen%10]); //////// đọc giá trị hàng đơn vị

output\_high(pin\_C5);

delay\_ms(2);

output\_low(pin\_C5);

output\_B(led7[du]); //////// đọc giá trị thập phân

output\_high(pin\_C6);

delay\_ms(2);

output\_low(pin\_C6);

if( nguyen>=40 ) //////// nhiệt độ giới hạn

{

output\_high(pin\_D2);

output\_high(pin\_D1);

output\_low(pin\_D3);

}

else

{

output\_high(pin\_D3);

output\_low(pin\_D2);

output\_low(pin\_D1);

}

delay\_ms(2);

}//while

}//void

//The End==========================================================