TRƯỜNG ĐẠI HỌC KỸ THUẬT – CÔNG NGHỆ CẦN THƠ
KHOA ĐIỆN TỬ – VIỄN THÔNG

ĐỒ ÁN MÔN HỌC VI ĐIỀU KHIỂN

ĐỀ TÀI

ĐỌC GIÁ TRỊ NHIỆT ĐỘ ĐIỀU KHIỂN BƠM NƯỚC CỨU HỎA

Giảng viên hướng dẫn: *Đường Khánh Sơn*

Nhóm sinh viên thực hiện: *Nhóm 7*

1. *Nguyễn Văn Đình* – *MSSV: 1350353*
2. *Thi Minh Nhựt* – *MSSV: 1350366*
3. *Phạm Thanh Quý* – *MSSV: 1350222*
4. *Liên Thái Trường* – *MSSV: 1350358*
5. *Lư Anh Tuấn* – *MSSV: 1350240*

Cần Thơ
Ngày 23 tháng 4 năm 2016

Mục lục

Danh sách hình vẽ

Danh sách bảng

NỘI DUNG BÁO CÁO

Giới thiệu về đề tài

Em chân thành cảm ơn thầy Đường Khánh Sơn đã nhận xét và góp ý cho nhóm em để hoàn thiện bài báo cáo hơn. Em xin tiếp nhận ý kiến và sẽ bổ sung, hoàn thiện thêm một số tín năng nữa: thu thập số liệu, cải tiến lại việc bơm nước (tính đến thời gian động cơ bơm nước lên). Chân thành cảm ơn thầy!

- ***Tên đề tài***: *Đọc giá trị nhiệt độ điều khiển bơm nước cứu hỏa.*
- ***Nội dung đề tài***: Sử dụng *vi điều khiển PIC 16F887* đọc giá trị nhiệt độ từ *cảm biến nhiệt độ DS18B20* để *điều khiển việc bơm nước tự động* thông qua việc *kích Relay* để *đóng, mở* khởi động động cơ bơm nước. Đề tài sử dụng động cơ DC để mô phỏng cho chương trình.

Dù nhóm em đã cố gắng hoàn thành thật tốt, nhưng không tránh khỏi thiếu sót, mong thầy và các bạn nhận xét, cho ý kiến để bài báo cáo của nhóm được hoàn thiện hơn.

Nhóm sinh viên thực hiện

Nhóm 7

Chương 1

Phần cứng

1.1 Danh sách phần cứng

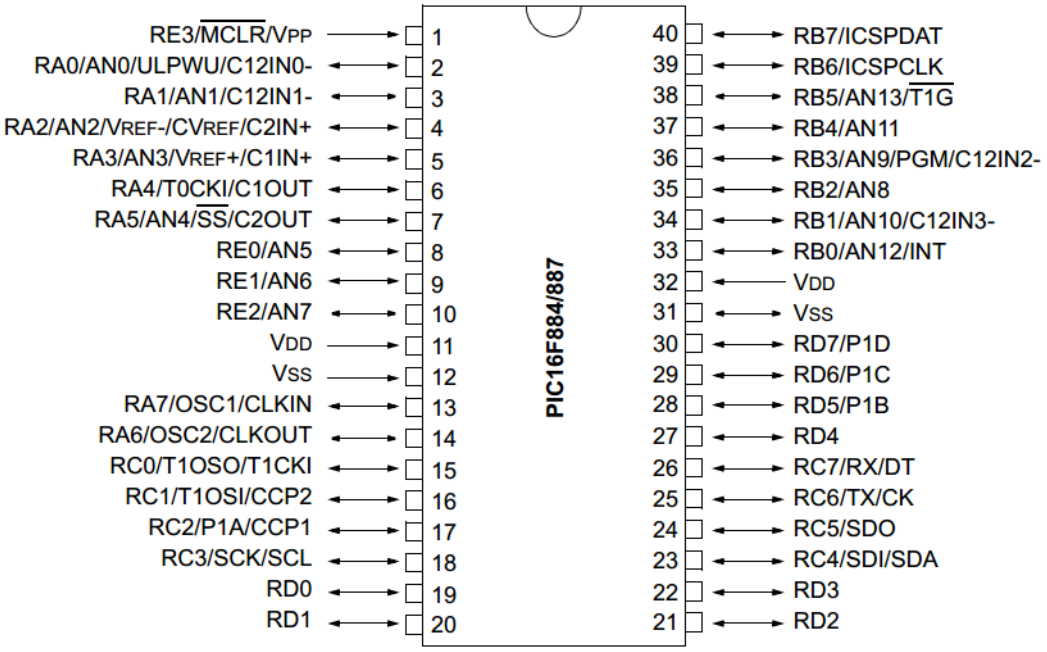
STT	Phần cứng	Mô tả
1	PIC 16F887	Vi điều khiển
2	Mạch nạp PICKit2 K150	Nạp chương trình cho PIC
3	Cảm biến DS18B20 (loại dây)	Đo nhiệt độ
4	LCD 1602	Màn hình hiển thị
5	Relay với Opto cách ly	Đóng mở motor
6	Động cơ 5VDC	Dùng để bơm nước (chạy demo)
7	Nguồn 5VDC - 2A	Cấp nguồn cho vi điều khiển
8	Nút nhấn 4 chân: 13 nút	Làm bàn phím và nút Reset
9	Điện trở $4.7k\Omega$: 14 điện trở	Tạo điện trở Pullup
10	Biến trở $10k\Omega$	Chỉnh độ tương phản của LCD
11	Thạch anh $20MHz$	Làm mạch dao động cho vi điều khiển
12	Tụ điện $15pF$: 2 tụ	Kết hợp tạo mạch dao động.
13	Domino 3 chân	Đấu nối dây cho cảm biến nhiệt độ
14	Đế IC 40 chân	Để gắn vi điều khiển PIC 16F887 lên
15	Rào cái loại đơn	Ra chân cho LCD
16	Rào đực loại đơn: 2 rào	Ra chân cua PIC 16F887
17	Dây kết nối 2 đầu cái cái	Dùng để kết nối các chân với nhau
18	Test board hàn mạch: 2 cái	Hàn linh kiện lên đây
19	Các dụng cụ khác, ...	

Bảng 1.1: Mô tả phần cứng dùng được dùng cho đề tài

Trong *phụ lục A*, nhóm có trình bày hình ảnh của các linh kiện trên trong thực tế.

1.2 Vi điều khiển PIC 16F887

1.2.1 Sơ đồ chân



Hình 1.1: Sơ đồ chân của PIC 16F887

1.2.2 Làm mạch điều khiển

a. Các thành phần cơ bản: *cấp nguồn, mạch dao động, mạch reset, sắp xếp thứ tự các chân*

- *Cấp nguồn cho vi điều khiển*: sử dụng nguồn DC – 5V.
  $*$*Lưu ý*: Sử dụng cả 4 chân: số $11, 12, 31, 32$.

Nguồn DC – 5V	**PIC 16F887**
5V	*Chân VDD*: số 11 và số 32
0V	*Chân VSS*: số 12 và số 31

Bảng 1.2: Cách cấp nguồn cho vi điều khiển PIC 16F887

- *Làm mạch dao động cho vi điểu khiển*: lựa cho giá trị tụ điện và tần số thạch anh thích hợp.
  - *Thạch anh:* Để thực hiện các lệnh trong vi điều khiển, cần phải tạo ra xung nhịp, tần số xung nhịp phụ thuộc vào thạch anh gắn vào vi điều khiển. Nhiệm vụ chính của thạch anh là tạo ra dao động ổn định.

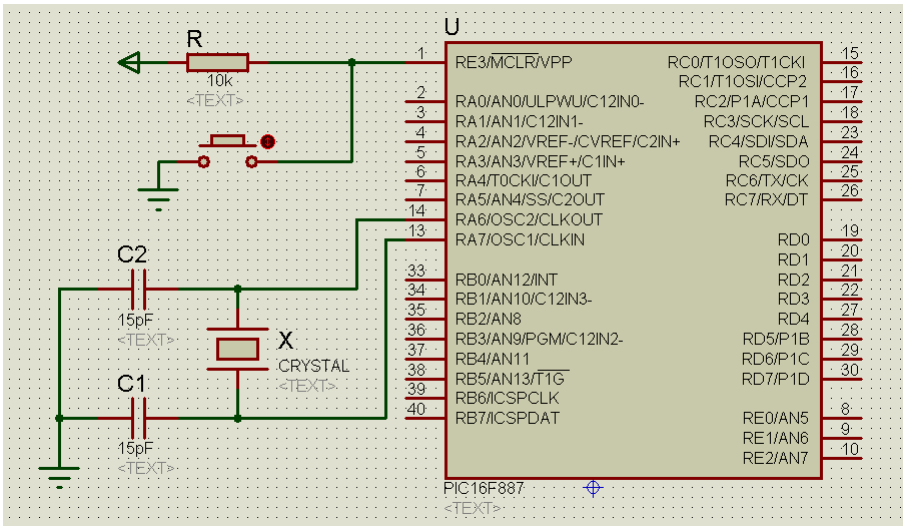
– *Tụ điện*: sử dụng hai tụ điện mắc vào để tăng tính ổn định tần số (tụ bù nhiệt ổn tần).
– Sử dụng mạch tạo dao động với thạch anh ổn định hơn mạch dao động *RC*.
– Cách kiểm tra thạch anh: Sử dụng *VOM* ở giai *đo tần số*, đo hai đầu của thạch anh, nếu *tần số bằng tần số ghi trên thạch anh* thì *thạch anh còn tốt*, còn ngược lại thì thạch anh bị hỏng.

Thạch anh	**PIC 16F887**
2 chân $20MHz$ và 2 tụ $15pF$	*Chân OSC1/CLKI và OSC2/CLKO*: chân số 13 và số 14

Bảng 1.3: Cách mắc thạch anh cho vi điều khiển PIC 16F887

∗ *Trong mạch thiết kế của đề tài: sử dụng thạch anh* $20MHz$ *và hai tụ điện có giá trị* $15pF$ *để tạo mạch dao động cho vi điều khiển.*

- *Tạo mạch Reset cho vi điều khiển:* sử dụng chân $MCLR/V_{PP}$. Bình thường chân này ở mức 1 (mức cao), để reset vi điều khiển, đưa chân $MCLR/V_{PP}$ xuống mức 0 (mức thấp). Xem hình 1.2
- *Sơ đồ nguyên lý*: được vẽ bằng phần mềm *Protues*.



Hình 1.2: Cách mắc mạch dao động bằng thạnh anh cho PIC 16F887

- *Có thể tạo thêm LED báo nguồn và LED báo Reset cho vi điều khiển để cho mạch có tính rõ ràng.*

b. Kiểm tra mạch đã hàn: Sau khi hàn mạch xong, sử dụng VOM *kiểm tra xem có ngắn mạch* giữa các chân với nhau không, phải đảm bảo là không có ngắn mạch giữa các chân thì sử dụng mạch thiết kế, nếu không sẽ gây hỏng PIC 16F887 hoặc mạch chạy không theo đúng chức năng đã lập trình.

1.3 Màn hình hiển thị LCD

1.3.1 Sơ đồ chân và chức năng của các chân trong LCD

Module LCD (xem hình A.1d trong phụ lục A): có 16 chân kết nối được đánh số theo thứ tự $1-16$ và được ghi trên LCD. Chức năng của các chân được mô tả trong bảng 1.4:

STT	Ký hiệu	Mô tả	Giá trị
1	VSS	GND	$0V$
2	VCC		$5V$
3	VEE	Tùy chỉnh độ tương phản	
4	RS	Lựa chọn thanh ghi	RS = 0: ghi lệnh RS = 1: ghi dữ liệu
5	R/W	Chọn thanh ghi đọc/viết dữ liệu	R/W = 0: viết dữ liệu R/W = 1: đọc dữ liệu
6	E	Enable	
7	DB0	Chân chuyền dữ liệu	8 bit từ $DB0 \rightarrow DB7$
8	DB1		
9	DB2		
10	DB3		
11	DB4		
12	DB5		
13	DB6		
14	DB7		
15	A	Cực dương của LED nền	$0-5V$
16	K	Cực âm của LED nền	$0V$

Bảng 1.4: Sơ đồ chân và chứa năng các chân trong LCD

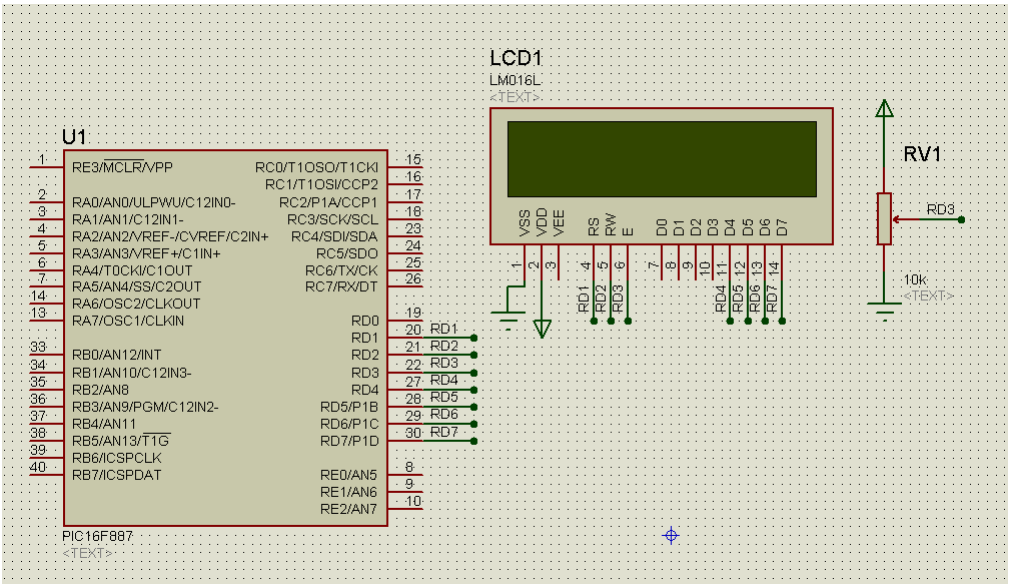
1.3.2 Kết nối LCD với vi điều khiển PIC 16F887

- Có 2 cách truyền dữ liệu: truyền cả 8 bit dữ liệu (từ $DB0 \rightarrow DB7$) hoặc truyền 4 bit dữ liệu (từ $DB4 \rightarrow DB7$). Chọn cách truyền *4 bit dữ liệu*.
- *Cách kết nối*: xem bảng 1.5.

LCD 16x2	**PIC 16F887**	
VSS: *số 1*	VSS: *số 12*	
VCC: *số 2*	VDD: *số 11*	
VEE: *số 3*		Biến trở $10k\Omega$
RS: *số 4*	RD1: *số 20*	
R/W: *số 5*	RD2: *số 21*	
E: *số 6*	RD3: số 22	
DB4: *số 11*	RD4: *số 23*	
DB5: *số 12*	RD5: *số 28*	
DB6: *số 13*	RD6: *số 29*	
DB7: *số 14*	DB7: *số 30*	
A: *số 15*	VDD: *số 11*	
K: *số 16*	VSS: *số 12*	

Bảng 1.5: Cách kết nối LCD với vi điều khiển PIC 16F887

- *Sơ đồ nguyên lý*: trên hình 1.3 được vẽ bằng phần mềm Protues.



Hình 1.3: Cách kết nối LCD với vi điều khiển PIC 16F887

1.3.3 Tập lệnh của LCD

Chúng ta có thể tìm hiểu các hướng dẫn trên các trang Internet hoặc trong tài liệu tham khảo `Systronix_20x4_lcd_brief_data.pdf` để hiểu rõ các lệnh của LCD.

1.3.4 Điều khiển LCD

- Ta quan tâm đến việc hiển thị lên LCD. Tức ghi giá trị và cho hiển thị lên LCD. Dùng lệnh: `Output_low(LCD_RW);` $\longrightarrow$ đưa chân `R/W = 0` để thực hiện chế độ ghi (xem bảng 1.4) và các chân được định nghĩa trong file `DEF_PIN.H` (xem mục B.2 của phụ lục B).
- Sử dụng các hàm được định nghĩa trong file `LCD_LIB_4BIT.C` của mục B.3 trong phụ lục B:
  - Hiển thị ký tự, chuỗi hoặc số lên LCD: dùng `LCD_PutChar(unsigned int cX);`
    - Ký tự hoặc chuỗi: *ví dụ*: hiển thị chuỗi `"HELLO LCD"` thì ta dùng dòng lệnh sau:

```
printf(LCD_PutChar,"HELLO LCD");
```

    - Số: số thực hoặc số nguyên, thì ta định dạng: `%d` (số nguyên) hoặc `%lu` (với số nguyên dài) hoặc `%f` (với số thực), *ví dụ*: hiện thị số `123` hoặc `1.23` thì ta dùng lệnh sau:

```
//Hien thi so nguyen
printf(LCD_PutChar,"%d",123);
//Hien thi voi 2 chu so thap phan sau day phay
printf(LCD_PutChar,"%.2f",1.23);
```

  - Vị trí con trỏ: dùng hàm `LCD_SetPosition(unsigned int cX);`
    - Chuyển đến vị trí đầu tiên của dòng 1: `LCD_SetPosition(0x00);`, tăng giá trị lên để chuyển đến những vị trí tiếp theo dòng 1.
    - Chuyển đến vị trí đầu tiên của dòng 2: `LCD_SetPosition(0x40;)`, tăng giá trị lên để chuyển đến những vị trí tiếp theo dòng 2.
  - Xóa màn hình: dùng lệnh `LCD_PutCmd(0x01);`
  - Các lệnh trên là các lệnh được sử dụng để thao tác với LCD trong phạm vi đề tài.
- Để sử dụng được thư viện `LCD_LIB_4BIT.C` cần chép chung các file `DEF_887.H; DEF_PIN.H; LCD_LIB_4BIT.C` và một số khai báo cơ bản khi viết chương trình bằng ngôn ngữ CCS.

1.4 Cảm biến nhiệt độ DS18B20

1.4.1 Thông số kỹ thuật

- Là IC cảm biến nhiệt độ có 3 chân: VCC, GND và DATA.
- Giao tiếp thông qua *giao thức một dây* với vi xử lý.
- Đặc điểm chính:
  - Cung cấp nhiệt độ với độ phân giải 12 *bit*.

- Khoảng nhiệt độ rộng: $-10 \div 125^0C$.
- Sai số cho phép: $\pm 0.5^0C$.
- Có chức năng cảnh báo nhiệt khi nhiệt độ vượt ngưỡng cho phép. Bộ nhớ nhiệt độ cảnh báo không bị mất khi mất nguồn.
- Có mã nhận diện lên đến 64-bit, vì vậy chúng ta có thể kiểm tra nhiệt độ với nhiều IC DS18B20 mà chỉ dùng 1 dây dẫn duy nhất để giao tiếp với các IC này.

- Để biết nhiều thông số hơn, xem thông tin Datasheet của IC với tên là `DS18B20.pdf`[1].

1.4.2 Sơ đồ chân và các kết nối với vi điều khiển PIC16F887

- IC DS18B20 có 3 chân: VCC, GND và DATA. Khi được tích hợp thành dây đo nhiệt độ (xem trong hình A.1c trong phần phụ lục A): *Dây màu đỏ* – VCC; *Dây màu đen* – GND; *Dây màu vàng* hoặc *Dây màu trắng* – DATA.

- Cách kết nối:

DS18B20	**PIC16F887**
Dây đỏ – VCC	VDD
Dây đen – GND	VSS
Dây vàng hoặc dây trắng – DATA	RC1
Nối điện trở $4.7k\Omega$ giữa chân VCC và chân DATA Để bit dữ liệu được kéo lên nguồn	

Bảng 1.6: Kết nối DS18B20 với PIC16F887

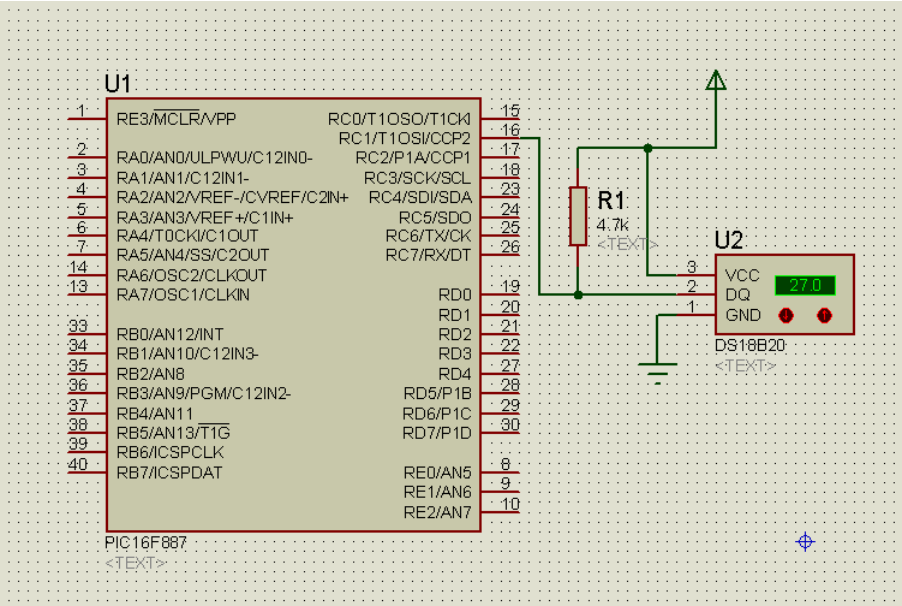
- *Sơ đồ nguyên lý*: trên hình 1.4 được vẽ bằng phần mềm Protues.

1.4.3 Giao thức giao tiếp một dây

- *Chuẩn giao tiếp một dây – 1 WIRE*: là đường dẫn tín hiệu và đường dẫn điện áp nguồn nuôi có thể dùng chung trên một dây dẫn, nhiều cảm biến có thể dùng chung trên một đường dẫn (rất thích hợp với các ứng dụng đo lường đa điểm).

- Các thiết bị *Slave* kết nối *cùng một bus* được phân biệt bởi *64 bit địa chỉ*: Bắt đầu từ LBS:
  - Byte đầu: mã của họ thiết bị có độ lớn 8 bit *(8-bit family codes)*.
  - 6 byte tiếp theo: địa chỉ riêng của thiết bị.
  - Byte cuối: kiểm tra tính toàn vẹn của dữ liệu *cyclic redundancy check – CRC* có giá trị ứng với 7 byte đầu tiên.

[1]http://www.alldatasheet.com/view.jsp?Searchword=DS18B20



Hình 1.4: Cách kết nối cảm biến nhiệt độ DS18B20 với vi điều khiển PIC16F887

- Cách thức hoạt động:
  - Bốn thao tác chính là: *reset*, *gửi bit 1*, *gửi bit 0* và *đọc bit*.
  - Mô tả 4 thao tác trên: trong bảng 1.7.

Hoạt động	**Mô tả**	**Thực thi**
Gửi bit 1	Gửi bit 1 đến thiết bị Slave	Kéo bus xuống mưc thấp: $6\mu s$, nhả ra: $64\mu s$
Gửi bit 0	Gửi bit 1 đến thiết bị Slave	Kéo bus xuống mưc thấp: $60\mu s$, nhả ra: $10\mu s$
Đọc bit	Đọc 1 bit	Kéo bus xuống mưc thấp: $6\mu s$, nhả ra: $9\mu s \longrightarrow$ đọc bit rồi delay $55\mu s$
Reset	reset thiết bị Slave và chuẩn bị nhận lệnh	Kéo bus xuống mức thấp trong $480\mu s$

Bảng 1.7: Các thức hoạt động của chuẩn giao tiếp ONE WIRE

- File code `ONEWIRE.C` của chuẩn giao tiếp một dây được trình bày trong mục B.4 của phụ lục B.

1.4.4 Đọc nhiệt độ từ cảm biến DS18B20

- Các lệnh ROM của DS18B20: bảng 1.8.

Tên lệnh	DATA	Chức năng
READ ROM	33H	Đọc 64 bit ROM của DS18B20
MATCH ROM	33H	khi dùng nhiều DS18B20 thì cần phải dùng lệnh này xác nhận 64 bit ROM của một cảm biến
SKIP ROM	CCH	Truy cập đến bộ nhớ, bỏ qua xác nhận 64 bit ROM
SEARCH ROM	F0H	Tìm số DS18B20 được kết nối vào bus
ALARMSEARCH	ECH	Phản hồi khi nhiệt độ thấp hơn hoặc cao hơn ngưỡng cảnh báo đã cài đặt

Bảng 1.8: Các lệnh trên ROM của DS18B20

- Các lệnh thực thi của DS18B200: bảng 1.9.

Tên lệnh	DATA	Chức năng
WRITE SCRCHPAD	4EH	Gửi 3 byte dữ liệu vào bộ nhớ nháp: Ngưỡng cảnh báo trên, ngưỡng cảnh báo dưới, cấu hình độ phâ giải cho phép đo
READ SCRCHPAD	BEH	Đọc nội dung của SCRCHPAD gồm 9 byte
COPY SCRCHPAD	48H	Copy 2 byte, 3 byte, 4 byte từ SCRCHPAD vào ROM của DS18B20
CONVERT T	44H	Bắt đầu chuyển đổi nhiệt độ, kết quả lưu vào byte 0, byte 1 của SCRCHPAD. Thời gian chuyển đổi $\leq 200ms$
READ POWERSUPPLY	B4H	Một lệnh đọc sau lệnh này sẽ cho biết DS18B20 sử dụng chế độ cấp nguồn nào. Phản hồi về: 1 nếu là VDD; 0 nếu DATA

Bảng 1.9: Các lệnh thực thi của DS18B20

- Sử dụng hàm `ds18b20_read();` trong file `DS18B20.C` để đọc giá trị nhiệt độ ở dạng ${}^0C$ (xem trong mục B.5 của phụ lục B).

1.5 Điều khiển tắt mở động cơ kết hợp với Module Relay

1.5.1 Cách kết nối với PIC16F887

- Mô tả: Sử dụng tín hiệu (0 hoặc 1) từ một chân của vi điều khiển để kích cho Relay đóng mở để kín mạch hoặc hở mạch động cơ.
- Cách kết nối:

Relay	**PIC 16F887**
DC –	0V
DC+	5V
IN	RC6

Bảng 1.10: Cách kết nối Relay với vi điều khiển PIC16F887

- Đầu ra của Relay có 3 chân: một chân chung - *COM* và hai tiếp điểm: thường đóng *NC* và thường mở *NO*.
- Tùy theo ta chọn việc kích ở mức 1 hoặc ở mức 0 mà ta chọn cách đấu nối thích hợp: thường đóng hoặc thường mở.
  - Nếu kích ở mức 1: $NO \longrightarrow NC$ và ngược lại: $NC \longrightarrow NO$.
  - Nếu kích ở mức 0: $NC \longrightarrow NO$ và ngược lại: $NO \longrightarrow NC$.
- Chọn kích ở mức 1. Đấu tải vào đầu ra của Relay: Một đầu tải nối vào chân COM của relay, đầu còn lại của tải nối vào một đầu của nguồn cấp; đầu còn lại của nguồn cấp đấu vào tiếp điểm NO của Relay.

1.5.2 Thực hiện kích Relay đóng mở động cơ

- Để kích đóng: dùng lệnh `output_high(RC6);` $\longrightarrow$ đưa chân RC6 lên mức 1. Tiếp điểm NO đóng lại, động cơ hoạt động.
- Để kích ngắt: dùng lệnh `output_low(RC6);` $\longrightarrow$ đưa chân RC6 xuống mức 0. Nếu tiếp điểm NO đang đóng thì sẽ mở ra, còn đang mở thì vẫn duy trạng thái mở.

Chương 2

Phần mềm

2.1 Chương trình chính

Nội dung của file MAIN.C

```
#include "INCLUDES.H"

float temp_float;
float nhietdo_setup;

void main(){
  int temp1, temp2,temp3; // Bien duoc do ra tu EEPROM
  int keypad[10],i, position;
  int B_ENTER, B_EXIT, E0;

  temp1 = read_eeprom(add_setup_temp_1);
  temp2 = read_eeprom(add_setup_temp_2);
  temp3 = read_eeprom(add_setup_temp_3);
  nhietdo_setup = temp1*10 + temp2 + temp3*0.1; //Gia tri nhiet do setup

  Output_low(LCD_RW);
  LCD_Init();

  TRISB = 0xFF;
  TRISE = 0xFF;
  LCD_PutCmd(0x01);
  LCD_SetPosition(0x00); //Chuyen vi tri con tro sang dong 1

  printf(lcd_putchar,"Setup: %.1f",nhietdo_setup);
  lcd_putchar(223);
  printf(lcd_putchar,"C");

  while (true){
    temp_float = ds18b20_read(); //Doc nhiet do tu cam bien DS18B20
    LCD_SetPosition(0x40);   //Chuyen vi tri con tro sang dong 2
    printf(lcd_putchar,"Measure: %.1f",temp_float);
    lcd_putchar(223);
    printf(lcd_putchar,"C");
```

```
34      OF_RELAY(temp_float, nhietdo_setup);
35      E0 = input(SETUP_EXIT);
36      if (E0 == 0){
37        LCD_PutCmd(0x01);
38        LCD_SetPosition(0x00);
39        printf(lcd_putchar,"SETUP TEMP: ");
40        delay_ms(1000);
41        i = 0;
42        position = 64;
43        B_EXIT = input(SETUP_EXIT); //Khi chua nhan nut EXIT hay ENTER
44        B_ENTER = input(ENTER);
45        while ((B_ENTER==1) | (B_EXIT == 1)){
46          int value;
47          B_EXIT = input(SETUP_EXIT);
48          B_ENTER = input(ENTER);
49          value = READ_BUTTON();
50          //Chi cho nhap va hien thi 3 so
51          if ((value >= 0) && (value < 10) && (i<=2)){
52            LCD_SetPosition(position);
53            printf(lcd_putchar,"%d",value);
54            delay_ms(1000);
55            keypad[i] = value;
56            i++;
57            position++;
58          }
59          else
60            //Khi da nhap du 3 so va nhan nut ENTER
61            if ((value == 100) && (i>=3)){
62              write_eeprom(add_setup_temp_1,keypad[0]);
63              write_eeprom(add_setup_temp_2,keypad[1]);
64              write_eeprom(add_setup_temp_3,keypad[2]);
65
66              temp1 = read_eeprom(add_setup_temp_1);
67              temp2 = read_eeprom(add_setup_temp_2);
68              temp3 = read_eeprom(add_setup_temp_3);
69              //Gia tri nhiet do setup
70              nhietdo_setup = temp1*10 + temp2 + temp3*0.1;
71
72              LCD_PutCmd(0x01);
73              LCD_SetPosition(0x00); //Chuyen vi tri con tro sang dong 1
74
75              printf(lcd_putchar,"Setup: %.1f",nhietdo_setup);
76              lcd_putchar(223);
77              printf(lcd_putchar,"C");
78
79              temp_float = ds18b20_read();
80
81              LCD_SetPosition(0x40); //Chuyen vi tri con tro sang dong 2
82              printf(lcd_putchar,"Measure: %.1f",temp_float);
83              lcd_putchar(223);
84              printf(lcd_putchar,"C");
```

```
85              break;
86            }
87            else
88              if (value == 50){  //Khi da nhan EXIT thoat khoi vong lap
89                LCD_PutCmd(0x01);
90                LCD_SetPosition(0x00); //Chuyen vi tri con tro sang dong 1
91
92                printf(lcd_putchar,"Setup: %.1f",nhietdo_setup);
93                lcd_putchar(223);
94                printf(lcd_putchar,"C");
95
96                temp_float = ds18b20_read();
97
98                LCD_SetPosition(0x40); //Chuyen vi tri con tro sang dong 2
99                printf(lcd_putchar,"Measure: %.1f",temp_float);
100               lcd_putchar(223);
101               printf(lcd_putchar,"C");
102               break;
103             }
104       }
105     }
106   }
107 }
```

2.2 Các chương trình con

2.2.1 File khai báo

Nội dung của file INCLUDES.H

```
1  #include <16f887.h> //Ten chip
2  #include "def_887.h" //Dinh nghia cac dia chi cac chan cua chip
3  #FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT,NOLVP, NOCPD,
       NOWRT
4  //Tan so thach anh 20MHz
5  #use delay(clock=20000000)
6  //Khai bao thu vien de su dung cac ham tinh toan
7  #include <MATH.H>
8
9  //Dinh nghia cac chan duoc su dung trong chuong trinh
10 #include "DEF_PIN.H"
11
12 //Khai bao dia chi EEPROM de luu du lieu
13 #include "ADDRESS_EEPROM.H"
14
15 //Thu vien LCD
16 #include "LCD_LIB_4BIT.C"
17
18 //Khai bao cac ham cua cam bien DS18B20
19 #include "ONEWIRE.C" //Giao tiep 1 day
```

```
//Doc nhiet do tu cam bien DS18B20
#include "DS18B20.C"

//Doc gia tri tu ban phim
#include "BUTTON.C"
#include "RELAY.C"
```

2.2.2 Các file đã được trình bày ở các phần trước

Nội dung của các file DEF_887.H; DEF_PIN.H; LCD_LIB_4BIT.C; ONEWIRE.C; DS18B20.C (mục B.1 , mục B.2, mục B.3, mục B.4, mục B.5 của phụ lục B) đã được đề cập trước đó.

2.2.3 File khai báo địa chỉ ô nhớ EEPROM

Nội dung của file ADDRESS_EEPROM.H

```
//Dia chi o nho luu gia tri nhiet do da cai dat vao
#define add_setup_temp_1 0xFD //Luu chu so hang chuc
#define add_setup_temp_2 0xFE //Luu chu so hang don vi
#define add_setup_temp_3 0xFF //Luu chu so thap phan
```

2.2.4 File chương trình kích Relay đóng mở động cơ

Nội dung của file RELAY.C

```
/*   Nhan vao gia tri nhiet do
     --> Dieu khien tat mo motor */
void OF_RELAY(float temperature, float temp_setup);

void OF_RELAY(float temperature, float temp_setup){
  if (temperature >= temp_setup){
    output_high(RELAY_SIG);
    delay_ms(1000);
  }
  else{
    output_low(RELAY_SIG);
    delay_ms(1000);
  }
}
```

2.2.5 File chương trình đọc giá trị ghi trên nút nhấn

Nội dung của file BUTTON.C

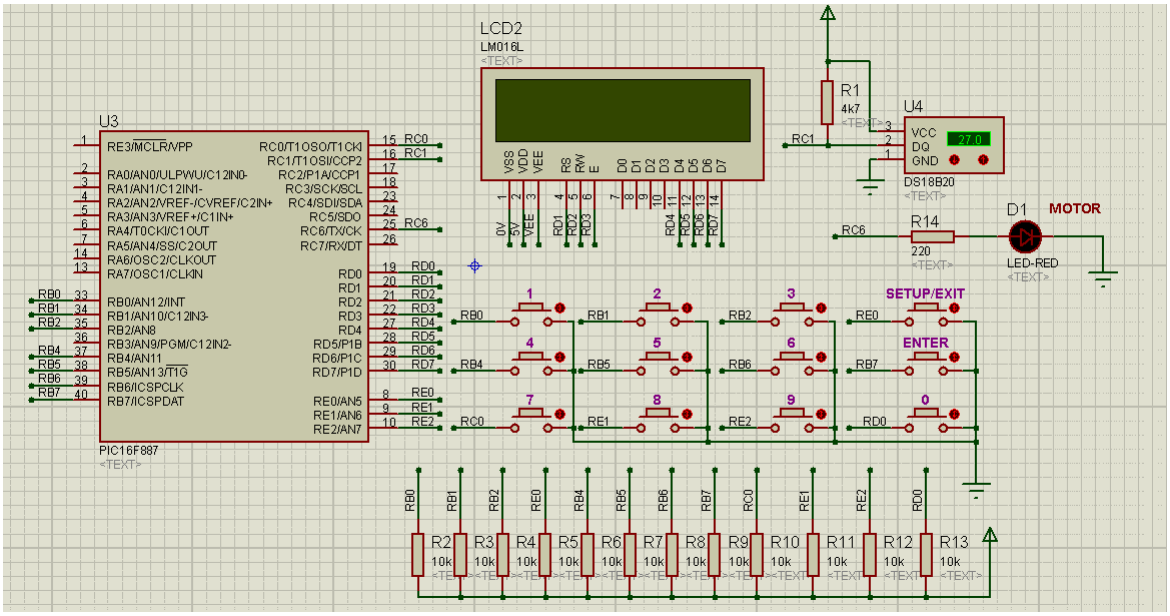
```
int READ_BUTTON();

```

```
int READ_BUTTON(){
  if (input(ONE) == 0){
    return 1;
  }
  else
    if (input(TWO) == 0){
      return 2;
    }
    else
      if (input(THREE) == 0){
        return 3;
      }
      else
        if (input(FOUR) == 0){
          return 4;
        }
        else
          if (input(FIVE) == 0){
            return 5;
          }
          else
            if (input(SIX) == 0){
              return 6;
            }
            else
              if (input(SEVEN) == 0){
                return 7;
              }
              else
                if (input(EIGHT) == 0){
                  return 8;
                }
                else
                  if (input(NINE) == 0){
                    return 9;
                  }
                  else
                    if (input(ZERO) == 0){
                      return 0;
                    }
                    else
                      if (input(ENTER) == 0){
                        return 100;
                      }
                      else
                        if (input(SETUP_EXIT)==0){
                          return 50;
                        }
                        else
                          return -1;
}
```

Chương 3

Mô hình – Kết quả

3.1 Mạch mô phỏng bằng Protues



Hình 3.1: Mạch mô phỏng của đề tài với Protues

3.2 Kết quả làm mạch thực tế

Hình 3.2: Mạch làm thực tế của đề tài

PHỤ LỤC

Phụ lục A

Các phần cứng và sơ đồ được dùng trong đề tài

Các phần cứng dùng điều khiển chính

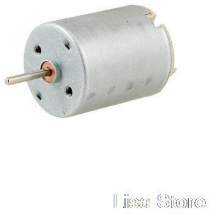
(a) PIC 16F887

(b) Mạch nạp PICKIT2 K150

(c) Cảm biến nhiệt độ DS18B20

(d) LCD 16x2

(e) Relay 5VDC với Opto cách ly



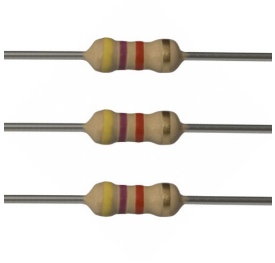
(f) Động cơ 5VDC

(g) Nút nhấn loại 4 chân

(h) Biến trở $10k\Omega$

Hình A.1: Các phần cứng chính dùng điều khiển cho mô hình

Các linh kiện hỗ trợ và phụ kiện kết nối



(a) Điện trở $4.7k\Omega$

(b) Thạch anh tần số $20MHz$

(c) Tụ điện $15pF$

(d) Domino đấu nối

(e) Đế IC 40 chân

(f) Rào cái loại đơn

(g) Rào cái loại đơn

(h) Dây kết nối 2 đầu cái - cái

(i) Testboard hàn mạch

Các phụ kiện khác

– *Bộ nguồn cấp cho vi điều khiển*: nguồn $5VDC - 2A$.

– Các dụng cụ thực hành điện tử: VOM, mỏ hàn, chì hàn, kiềm, dây kết nối khi hàn, ...

Phụ lục B

Nội dung của các file thư viện được sử dụng trong chương trình

B.1 Định nghĩa PIC16F887 - file DEF_887.H

- Nguồn tham khảo: *Tài liệu thực tập vi điều khiển* của thầy Đường Khánh Sơn.
- Nội dung của file `DEF_887.H`:

```
// register definitions

#define W 0
#define F 1

// register files
#byte INDF        =0x00
#byte TMR0        =0x01
#byte PCL         =0x02
#byte STATUS      =0x03
#byte FSR         =0x04
#byte PORTA       =0x05
#byte PORTB       =0x06
#byte PORTC       =0x07
#byte PORTD       =0x08
#byte PORTE       =0x09

#byte EEDATA      =0x10C
#byte EEADR       =0x10D
#byte EEDATH      =0x10E
#byte EEADRH       =0x10F
#byte ADCON0    =0x1F
#byte ADCON1    =0x9F
#byte ADRESH    =0x9F
#byte ADSESL    =0x9F

```

```
#byte PCLATH      =0x0a
#byte INTCON      =0x0b
#byte PIR1   =0x0c
#byte PIR2   =0x0d
#byte PIE1   =0x8c
#byte PIE2   =0x8d

#byte OPTION_REG  =0x81
#byte TRISA       =0x85
#byte TRISB       =0x86
#byte TRISC       =0x87
#byte TRISD       =0x88
#byte TRISE       =0x89

#byte EECON1      =0x18C
#byte EECON2      =0x18D

//DINH NGHIA BIT
#bit ra5  =0x05.5
#bit ra4  =0x05.4
#bit ra3  =0x05.3
#bit ra2  =0x05.2
#bit ra1  =0x05.1
#bit ra0  =0x05.0

#bit rb7  =0x06.7
#bit rb6  =0x06.6
#bit rb5  =0x06.5
#bit rb4  =0x06.4
#bit rb3  =0x06.3
#bit rb2  =0x06.2
#bit rb1  =0x06.1
#bit rb0  =0x06.0

#bit rC7  =0x07.7
#bit rC6  =0x07.6
#bit rC5  =0x07.5
#bit rC4  =0x07.4
#bit rC3  =0x07.3
#bit rC2  =0x07.2
#bit rC1  =0x07.1
#bit rC0  =0x07.0

#bit rD7  =0x08.7
#bit rD6  =0x08.6
#bit rD5  =0x08.5
#bit rD4  =0x08.4
#bit rD3  =0x08.3
#bit rD2  =0x08.2
#bit rD1  =0x08.1
#bit rD0  =0x08.0
```

```
78
79  #bit rE2  =0x09.2
80  #bit rE1  =0x09.1
81  #bit rE0  =0x09.0
82
83
84  #bit trisa5  =0x85.5
85  #bit trisa4  =0x85.4
86  #bit trisa3  =0x85.3
87  #bit trisa2  =0x85.2
88  #bit trisa1  =0x85.1
89  #bit trisa0  =0x85.0
90
91  #bit trisb7  =0x86.7
92  #bit trisb6  =0x86.6
93  #bit trisb5  =0x86.5
94  #bit trisb4  =0x86.4
95  #bit trisb3  =0x86.3
96  #bit trisb2  =0x86.2
97  #bit trisb1  =0x86.1
98  #bit trisb0  =0x86.0
99
100 #bit trisc7  =0x87.7
101 #bit trisc6  =0x87.6
102 #bit trisc5  =0x87.5
103 #bit trisc4  =0x87.4
104 #bit trisc3  =0x87.3
105 #bit trisc2  =0x87.2
106 #bit trisc1  =0x87.1
107 #bit trisc0  =0x87.0
108
109 #bit trisd7  =0x88.7
110 #bit trisd6  =0x88.6
111 #bit trisd5  =0x88.5
112 #bit trisd4  =0x88.4
113 #bit trisd3  =0x88.3
114 #bit trisd2  =0x88.2
115 #bit trisd1  =0x88.1
116 #bit trisd0  =0x88.0
117
118 #bit trise2  =0x89.2
119 #bit trise1  =0x89.1
120 #bit trise0  =0x89.0
121
122 // INTCON Bits for C
123 #bit gie   = 0x0b.7
124 #bit peie = 0x0b.6
125 #bit tmr0ie = 0x0b.5
126 #bit int0ie = 0x0b.4
127 #bit rbie  = 0x0b.3
128 #bit tmr0if   = 0x0b.2
```

```
#bit int0if   = 0x0b.1
#bit rbif   = 0x0b.0

// PIR1 for C
#bit pspif = 0x0c.7
#bit adif   = 0x0c.6
#bit rcif   = 0x0c.5
#bit txif   = 0x0c.4
#bit sspif   = 0x0c.3
#bit ccp1if   = 0x0c.2
#bit tmr2if   = 0x0c.1
#bit tmr1if   = 0x0c.0

//PIR2 for C
#bit cmif   = 0x0d.6
#bit eeif   = 0x0d.4
#bit bclif = 0x0d.3
#bit ccp2if   = 0x0d.0

// PIE1 for C
#bit adie   = 0x8c.6
#bit rcie   = 0x8c.5
#bit txie   = 0x8c.4
#bit sspie   = 0x8c.3
#bit ccp1ie   = 0x8c.2
#bit tmr2ie   = 0x8c.1
#bit tmr1ie   = 0x8c.0

//PIE2 for C
#bit osfie   = 0x8d.7
#bit cmie   = 0x8d.6
#bit eeie  = 0x8d.4

// OPTION Bits
#bit not_rbpu  = 0x81.7
#bit intedg    = 0x81.6
#bit t0cs      = 0x81.5
#bit t0se      = 0x81.4
#bit psa       = 0x81.3
#bit ps2       = 0x81.2
#bit ps1       = 0x81.1
#bit ps0       = 0x81.0

// EECON1 Bits
#bit eepgd  = 0x18c.7
#bit free     = 0x18C.4
#bit wrerr    = 0x18C.3
#bit wren      = 0x18C.2
#bit wr        = 0x18C.1
#bit rd        = 0x18C.0

```

```
180 //ADCON0
181 #bit CHS0   =0x1F.3
182 #bit CHS1   =0x1F.4
183 #bit CHS2   =0x1F.5
```

B.2 Định nghĩa các chân được sử dụng trong đề tài - file DEF_PIN.H

- Nội dung của file `DEF_PIN.H`:

```
1  //Dinh nghia cac chan cua LCD
2  #use standard_io(C)
3  #use standard_io(D)
4  #define LCD_RS PIN_D1
5  #define LCD_RW PIN_D2
6  #define LCD_EN PIN_D3
7
8  #define LCD_D4 PIN_D4
9  #define LCD_D5 PIN_D5
10 #define LCD_D6 PIN_D6
11 #define LCD_D7 PIN_D7
12
13
14 //Dinh nghia chan tin hieu cua cam bien nhiet do
15 #define ONE_WIRE_PIN PIN_C1
16
17 //Dinh nghia chan tin hieu cua Relay
18 #define RELAY_SIG PIN_C6
19
20 //Dinh nghia cac chan cua ban phim
21 #define ONE PIN_B0
22 #define TWO PIN_B1
23 #define THREE PIN_B2
24 #define SETUP_EXIT PIN_E0
25 #define FOUR PIN_B4
26 #define FIVE PIN_B5
27 #define SIX PIN_B6
28 #define ENTER PIN_B7
29 #define SEVEN PIN_C0
30 #define EIGHT PIN_E1
31 #define NINE PIN_E2
32 #define ZERO PIN_D0
```

B.3 Thư viện LCD - file LCD_LIB_4BIT.C

- Nguồn tham khảo: *Tài liệu thực tập vi điều khiển* của thầy Đường Khánh Sơn.

- Nội dung: sử dụng một số hàm truyền 4 bit dữ liệu.
  - `LCD_Init();` – Khởi tạo LCD.
  - `LCD_SetPosition(unsigned int cX);` – Cài đặt vị trí con trỏ của LCD.
  - `LCD_PutChar(unsigned int cX);` – Viết ký tự hoặc chuỗi lên LCD.
  - `LCD_PutCmd(unsigned int cX);` – Gửi lệnh lên LCD.
  - `LCD_PulseEnable();` – Xung kích hoạt.
  - `LCD_setData(unsigned int cX);` – Đặt dữ liệu lên chân Data.
- Nội dung của file `LCD_LIB_4BIT.C`:

```
#include <stddef.h>

#define Line_1 0x80
#define Line_2 0xC0
#define Clear_Scr = 0x01


#separate void LCD_Init(void);
#separate void LCD_SetPosition(unsigned int cX);
#separate void LCD_PutChar(unsigned int cX);
#separate void LCD_PutCmd(unsigned int cX);
#separate void LCD_PulseEnable(void);
#separate void LCD_setData(unsigned int cX);

#separate void LCD_Init(void){
  LCD_SetData(0x00);
  delay_ms(200);
  output_low(LCD_RS);
  LCD_SetData(0x03);
  LCD_PulseEnable();
  LCD_PulseEnable();
  LCD_PulseEnable();
  LCD_SetData(0x02);
  LCD_PulseEnable();
  LCD_PutCmd(0x2C);
  LCD_PutCmd(0x0C);
  LCD_PutCmd(0x06);
  LCD_PutCmd(0x01);
}


#separate void LCD_SetPosition(unsigned int cX){
  LCD_setData(swap(cX)|0x08);
  LCD_PulseEnable();
  LCD_setData(swap(cX));
  LCD_PulseEnable();
}

#separate void LCD_PutChar(unsigned int cX){
```

```
  output_high(LCD_RS);
  LCD_PutCmd(cX);
  output_low(LCD_RS);
}

#separate void LCD_PutCmd(unsigned int cX){
  LCD_SetData(swap(cX));
  LCD_PulseEnable();
  LCD_SetData(swap(cX));
  LCD_PulseEnable();
}


#separate void LCD_PulseEnable(void){
  output_high(LCD_EN);
  delay_us(3);
  output_low(LCD_EN);
  delay_ms(3);

}

#separate void LCD_setData(unsigned int cX){
  output_bit(LCD_D4,cX & 0x01);
  output_bit(LCD_D5,cX & 0x02);
  output_bit(LCD_D6,cX & 0x04);
  output_bit(LCD_D7,cX & 0x08);
}
```

B.4 Chuẩn giao tiếp một dây - file ONEWIRE.C

- Nguồn tham khảo: `ytuonghay.vn`
- Nội dung của file `ONEWIRE.C`:

```
#ifndef ONE_WIRE_C
#define ONE_WIRE_C

/*
 * One wire (1-wire) driver for CCS C compiler.
 * Suitable for use with devices
 * such as the DS18B20 1-wire digital temperature sensor.
 */

/*
 * onewire_reset()
 * Description: Initiates the one wire bus.
 */
// OK if just using a single permanently connected device
void onewire_reset() {
  output_low(ONE_WIRE_PIN); // pull the bus low for reset
```

```
17    delay_us(500);
18    output_float(ONE_WIRE_PIN); // float the bus high
19    // wait-out remaining initialisation window
20    delay_us(500);
21    output_float(ONE_WIRE_PIN);
22  }
23
24
25  /*
26   * onewire_write(int8 data)
27   * Arguments: a byte of data.
28   * Description: writes a byte of data to the device.
29   */
30  void onewire_write(int8 data) {
31    int8 count;
32
33    for(count = 0; count < 8; ++count) {
34      output_low(ONE_WIRE_PIN);
35      // pull 1-wire low to initiate write time-slot.
36      delay_us(2);
37      // set output bit on 1-wire
38      output_bit(ONE_WIRE_PIN, shift_right(&data, 1, 0));
39      delay_us(60);// wait until end of write slot.
40      output_float(ONE_WIRE_PIN); // set 1-wire high again,
41      delay_us(2); // for more than 1us minimum.
42    }
43  }
44
45  /*
46   * onewire_read()
47   * Description:
48   * reads and returns a byte of data from the device.
49   */
50  int onewire_read() {
51    int count, data;
52
53    for(count = 0; count < 8; ++count) {
54      output_low(ONE_WIRE_PIN);
55      // pull 1-wire low to initiate read time-slot.
56      delay_us(2);
57      // now let 1-wire float high,
58      output_float(ONE_WIRE_PIN);
59      delay_us(8);// let device state stabilise,
60      // and load result.
61      shift_right(&data, 1, input(ONE_WIRE_PIN));
62      delay_us(120);// wait until end of read slot.
63    }
64    return data;
65  }
66
67  #endif /*ONE_WIRE_C*/
```

B.5 Đọc nhiệt độ từ cảm biến DS18B20 - File DS18B20.C

- Nguồn tham khảo: `mcu.banlinhkien.vn`
- Nội dung của file `DS18B20.C`:

```
#ifndef DS18B20_C
#define DS18B20_C

float ds18b20_read();
void ds18b20_configure(int8 TH, int8 TL, int8 config);

/*
 * ds1820_read()
 * Description:
 *reads the ds18x20 device on the 1-wire bus and returns
 *             the temperature
 */

float ds18b20_read() {
  int8 busy=0, temp1, temp2;
  signed int16 temp3;
  float result;

  //ds1820_configure(0x00, 0x00, 0x00);
  //9 bit resolution

  onewire_reset();
  onewire_write(0xCC);//Skip ROM, address all devices
  onewire_write(0x44);//Start temperature conversion

  while(busy == 0) //Wait while busy (bus is low)
    busy = onewire_read();

  onewire_reset();
  onewire_write(0xCC);//Skip ROM, address all devices
  onewire_write(0xBE);//Read scratchpad
  temp1 = onewire_read();
  temp2 = onewire_read();
  temp3 = make16(temp2, temp1);

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

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

  delay_ms(200);    // ??????
  return(result);
}
```

```
45
46 /*
47  * ds1820_configure(int8 TH, int8 LH, int8 config)
48  * Description:
49  * writes configuration data to the DS18x20 device
50  * Arguments:
51  * alarm trigger high, alarm trigger low, configuration
52  */
53
54 void ds18b20_configure(int8 TH, int8 TL, int8 config) {
55   onewire_reset();
56   onewire_write(0xCC);//Skip ROM, address all devices
57   onewire_write(0x4E);//Write to scratchpad
58   onewire_write(TH);
59   onewire_write(TL);
60   onewire_write(config);
61 }
62
63 #endif /*DS1820_C*/
```