TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO BÀI TẬP LỚN

Lập trình vi điều khiển STM32 giao tiếp với cảm biến nhiệt độ, hiển thị LCD và truyền thông số lên máy tính

NGUYỄN TIẾN ĐẠT
dat.nt173727@sis.hust.edu.vn

Ngành Kỹ thuật điều khiển và Tự động hóa
Chuyên ngành Kỹ thuật đo và Tin học công nghiệp

Giảng viên hướng dẫn: ThS. Lê Công Cường
Chữ ký của GVHD

Bộ môn: Kỹ thuật đo và Tin học công nghiệp
Viện: Điện

HÀ NỘI, 1/2022

ĐỀ TÀI

Lập trình vi điều khiển STM32 giao tiếp với cảm biến nhiệt độ, hiển thị LCD và truyền thông số lên máy tính. Cài đặt bộ lập lịch từ máy tính.

Giáo viên hướng dẫn

Ký và ghi rõ họ tên

Lời cảm ơn

Đầu tiên em xin gửi lời cảm ơn vô cùng sâu sắc và chân thành tới ThS. Lê Công Cường, người đã hướng dẫn em rất tận tình trong suốt quá trình học tập cũng như thực hiện đề tài này.

Do thời gian và kiên thức có hạn nên không thể tránh khỏi những thiếu sót nhất định. Em rất mong nhận được ý kiến đóng góp quý báu của thầy.

Em xin chân thành cảm ơn!

Tóm tắt nội dung đồ án

Mục đích của đề tài này là thiết kế mạch đo nhiệt độ hiển thị LCD và truyền thông số lên máy tính. Trong đề tài, em đã đo được nhiệt độ, sử dụng 2 phương pháp là thiết kế bộ lập lịch và sử dụng hệ điều hành FreeRTOS. Đề tài này sẽ đề cập chi tiết đến cảm biến nhiệt độ, module LCD I2C sau đó sẽ đi vào phần thiết kế mạch phần cứng. Sau cùng sẽ là thiết kế phần mềm trền vi điều khiển STM32F103C8T6.

Sinh viên thực hiện
Ký và ghi rõ họ tên

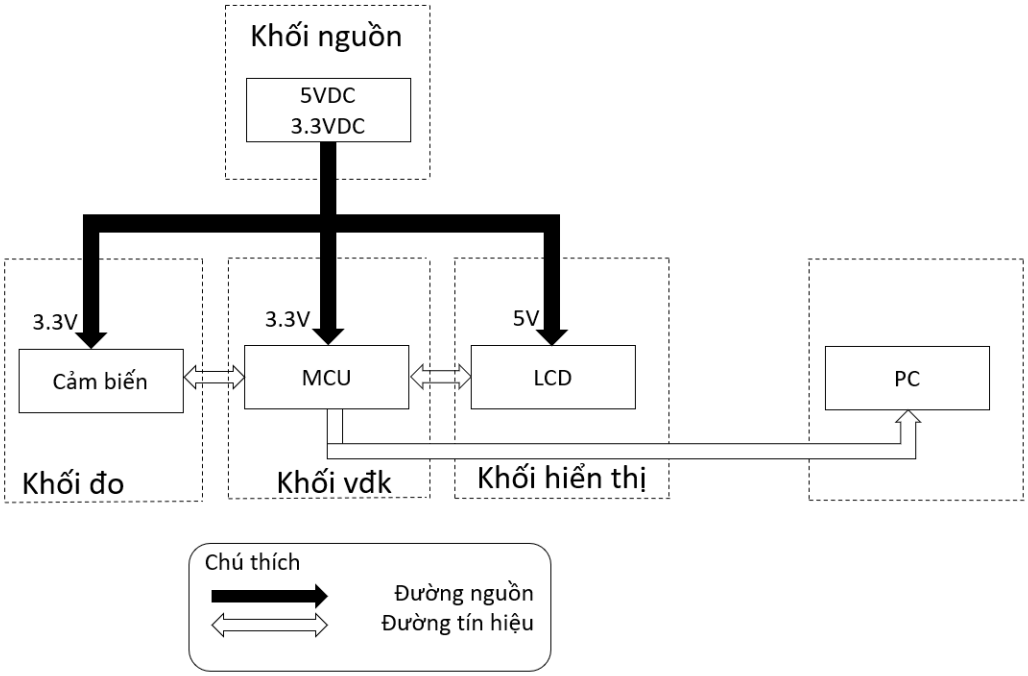
MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

CHƯƠNG 1. THIẾT KẾ HỆ THỐNG

1.1 Sơ đồ khối hệ thống



Hình 1.1 Sơ đồ khối hệ thống

Mô tả chức năng:

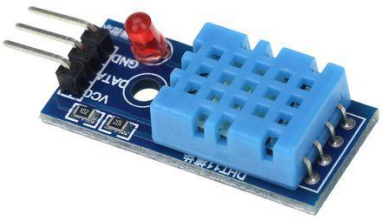
- Khối nguồn: Mạch được cấp nguồn 5V bởi Mini USB, sau đó chuyển đổi thành nguồn 3.3V để cấp nguồn cho vi điều khiển và cảm biến nhiệt độ.
- Khối đo: cảm biến nhiệt độ cung cấp tín hiệu cho vi điều khiển.
- Khối hiển thị: nhận dữ liệu từ MCU hiển thị lên LCD và PC.

Lựa chọn thiết bị

1.2 Cảm biến nhiệt độ DHT11

1.2.1 Giới thiệu chung

Cảm biến độ ẩm và nhiệt độ DHT11 Temperature Humidity Sensor là cảm biến rất thông dụng hiện nay vì chi phí rẻ và rất dễ lấy dữ liệu thông qua giao tiếp One wire. Bộ tiền xử lý tín hiệu tích hợp trong cảm biến giúp có được dữ liệu chính xác mà không phải qua bất kỳ tính toán nào.



Hình 1.2 Cảm biến nhiệt độ DHT11

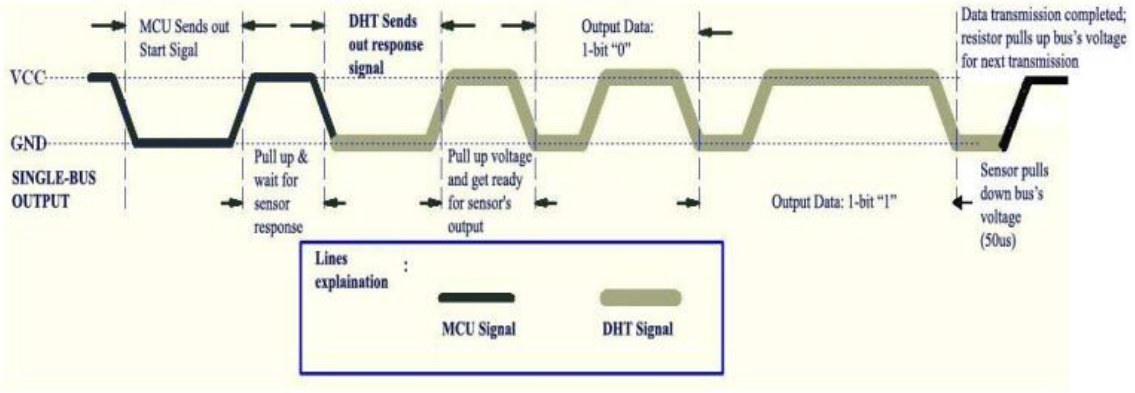
Thông số kỹ thuật:

- Nguồn: 3 -> 5 VDC.
- Dòng sử dụng: 2.5mA max (khi truyền dữ liệu).
- Đo tốt ở độ ẩm 20 to 70%RH với sai số 5%.
- Đo tốt ở nhiệt độ 0 to 50°C sai số ±2°C.
- Tần số lấy mẫu tối đa 1Hz (1 giây 1 lần).
- Kích thước 15mm x 12mm x 5.5mm.
- 4 chân, khoảng cách chân 0.1".

1.2.2 Nguyên lý hoạt động

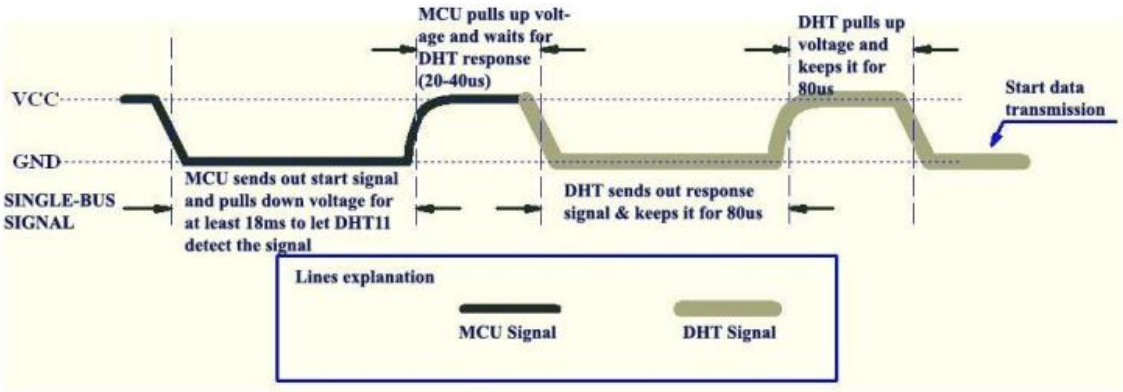
Để có thể giao tiếp với DHT11 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước:

- Gửi tin hiệu muốn đo (Start) tới DHT11, sau đó DHT11 xác nhận lại.
- Khi đã giao tiếp được với DHT11, Cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ đo được.



Hình 1.3 Quá trình giao tiếp với DHT11

- Bước 1: Gửi tín hiệu Start



Hình 1.4 Gửi tín hiệu Start tới DHT11

- MCU thiết lập chân DATA là Output, kéo chân DATA xuống 0 trong khoảng thời gian >18ms. Khi đó DHT11 sẽ hiểu MCU muốn đo giá trị nhiệt độ và độ ẩm.
- MCU đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào.
- Sau khoảng 20-40us, DHT11 sẽ kéo chân DATA xuống thấp. Nếu >40us mà chân DATA ko được kéo xuống thấp nghĩa là ko giao tiếp được với DHT11.

- Chân DATA sẽ ở mức thấp 80us sau đó nó được DHT11 kéo nên cao trong 80us. Bằng việc giám sát chân DATA, MCU có thể biết được có giao tiếp được với DHT11 không. Nếu tín hiệu đo được DHT11 lên cao, khi đó hoàn thiện quá trình giao tiếp của MCU với DHT.

➢ Bước 2: Đọc giá trị trên DHT11

DHT11 sẽ trả giá trị nhiệt độ và độ ẩm về dưới dạng 5 byte. Trong đó:

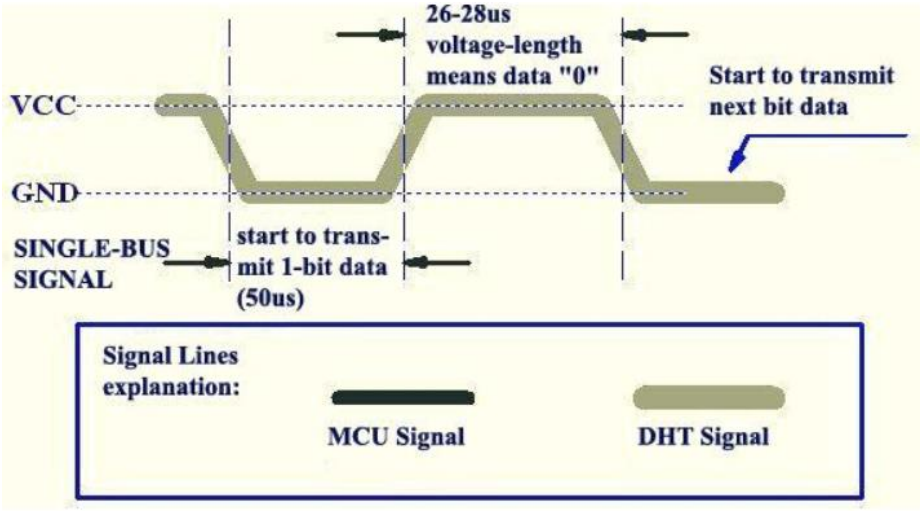
- Byte 1: giá trị phần nguyên của độ ẩm (RH%)
- Byte 2: giá trị phần thập phân của độ ẩm (RH%)
- Byte 3: giá trị phần nguyên của nhiệt độ (TC)
- Byte 4: giá trị phần thập phân của nhiệt độ (TC)
- Byte 5: kiểm tra tổng.

Nếu Byte 5 = (8 bit) (Byte1 +Byte2 +Byte3 + Byte4) thì giá trị độ ẩm và nhiệt độ là chính xác, nếu sai thì kết quả đo không có nghĩa.

Đọc dữ liệu:

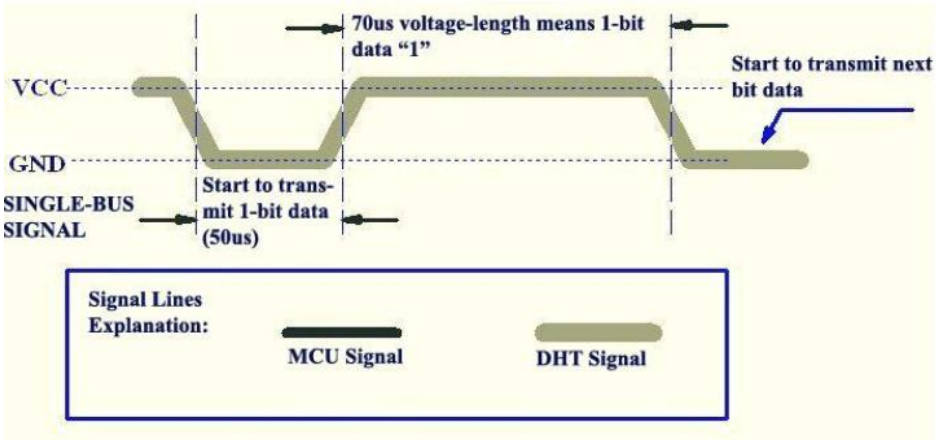
Sau khi giao tiếp được với DHT11, DHT11 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về MCU, tương ứng chia thành 5 byte kết quả của Nhiệt độ và độ ẩm.

Bit 0:



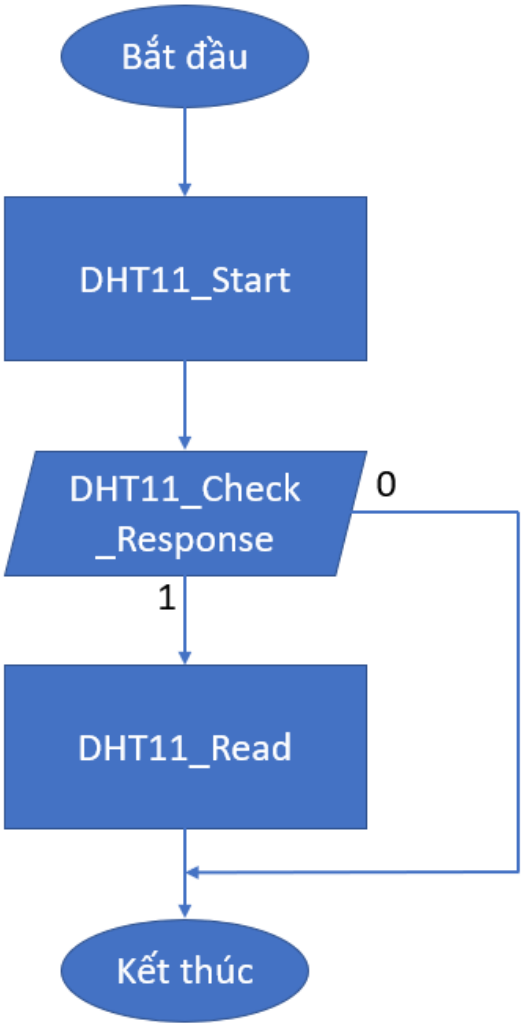
Hình 1.5 Bit 0 của DHT11

Bit 1:



Hình 1.6 Bit 1 của DHT11

1.2.3 Lưu đồ thuật toán giao tiếp cảm biến DHT11



Hình 1.7 Lưu đồ thuật toán giao tiếp cảm biến DHT11

1.3 Cảm biến nhiệt độ DS18B20

1.3.1 Giới thiệu chung

Cảm biến nhiệt độ DS18B20 là cảm biến (loại digital) đo nhiệt độ mới của hãng MAXIM với độ phân giải cao (12bit). IC sử dụng giao tiếp 1 dây rất gọn gàng, dễ lập trình. 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 1.8 Cảm biến nhiệt độ DS18B20

Thông số kỹ thuật:

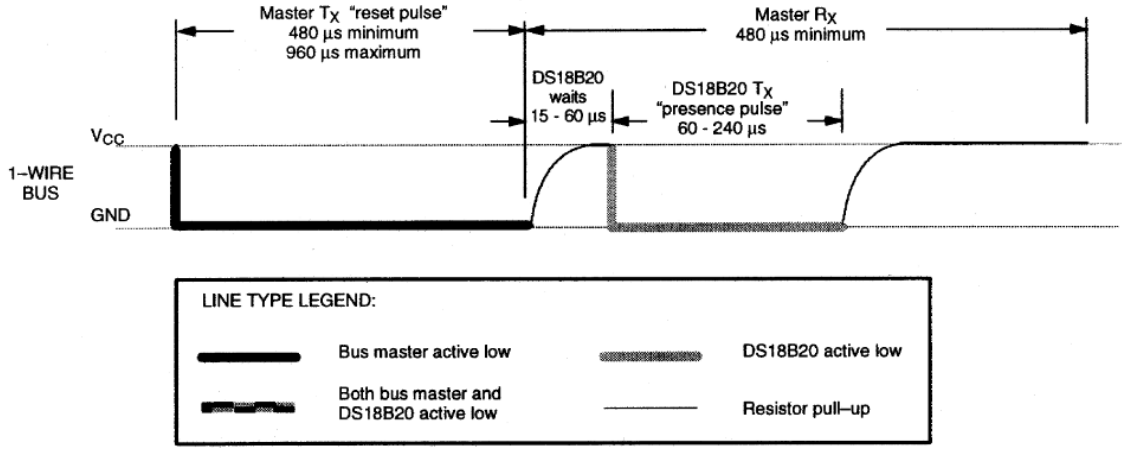
- Nguồn: 3 – 5.5V.
- Dải đo nhiệt độ: -55 đến 125 độ C (-67 đến 257 độ F).
- Sai số: +- 0.5 độ C khi đo ở dải -10 – 85 độ C.
- Độ phân giải: người dùng có thể chọn từ 9 – 12 bits.
- Chuẩn giao tiếp: 1-Wire (1 dây).
- Có cảnh báo nhiệt khi vượt ngưỡng cho phép và cấp nguồn từ chân data.
- Thời gian chuyển đổi nhiệt độ tối đa: 750ms (khi chọn độ phân giải 12 bit)
- Mỗi IC có một mã riêng (lưu trên EEPROM của IC) nên có thể giao tiếp nhiều DS18B20 trên cùng 1 dây

1.3.2 Nguyên lý hoạt động

Để có thể giao tiếp với DS18B20 theo chuẩn 1 chân vi xử lý thực hiện theo 3 bước:

- Gửi tin hiệu muốn đo (Start) tới DS18B20, sau đó DS18B20 xác nhận lại.
- Khi đã giao tiếp được với DS18B20, gửi cấu hình đến DS18B20.
- Đọc giá trị nhiệt độ ở ROM của DS18B20.

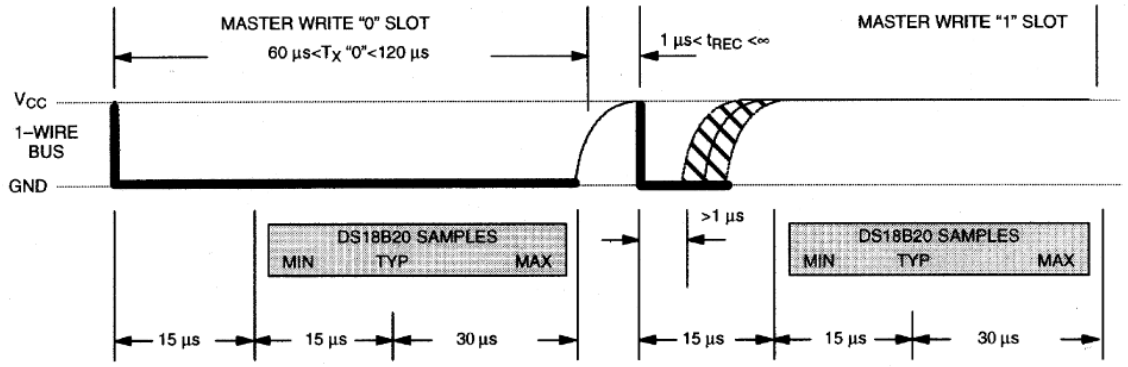
➢ Bước 1: Gửi tín hiệu Start



Hình 1.9 Gửi tín hiệu Start tới DS18B20

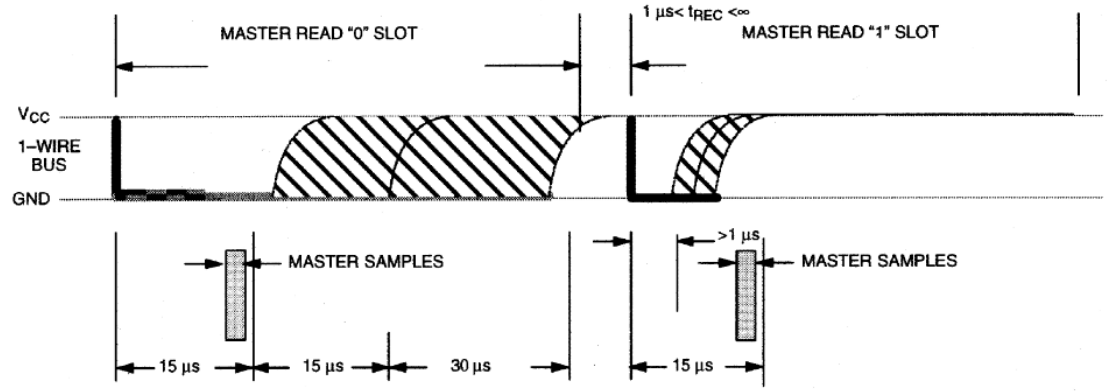
- MCU thiết lập chân DATA là Output, kéo chân DATA xuống 0 trong khoảng thời gian 480us-960us. Khi đó DS18B20 sẽ hiểu MCU muốn đo giá trị nhiệt độ.
- MCU đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào.
- Sau khoảng 15-60us, DS18B20 sẽ kéo chân DATA xuống thấp. Nếu >60us mà chân DATA ko được kéo xuống thấp nghĩa là ko giao tiếp được với DHT11.
- Chân DATA sẽ ở mức thấp 60us-240us sau đó nó được DS18B20 kéo nên cao. Bằng việc giám sát chân DATA, MCU có thể biết được có giao tiếp được với DS18B20 không. Nếu tín hiệu đo được DS18B20 lên cao, khi đó hoàn thiện quá trình giao tiếp của MCU với DS18B20.

➢ Bước 2: Gửi cấu hình đến DS18B20



Hình 1.10 Gửi bit 0 và 1 tới DS18B20

Bước 3: Đọc giá trị nhiệt độ ở ROM của DS18B20



Hình 1.11 Đọc bit 0 và 1 từ DS18B20

1.3.3 Các lệnh ROM của DS18B20

- READ ROM (33h)

Cho phép đọc ra 8 byte mã đã khắc bằng laser trên ROM, bao gồm: 8 bit mã định tên linh kiện (10h), 48 bit số xuất xưởng, 8 bit kiểm tra CRC. Lệnh này chỉ dùng khi trên bus có 1 cảm biến DS1820, nếu không sẽ xảy ra xung đột trên bus do tất cả các thiết bị tớ cùng đáp ứng.

- MATCH ROM (55h)

Lệnh này được gửi đi cùng với 64 bit ROM tiếp theo, cho phép bộ điều khiển bus chọn ra chỉ một cảm biến DS1820 cụ thể khi trên bus có nhiều cảm biến DS1820 cùng nối vào. Chỉ có DS1820 nào có 64 bit trên ROM trung khớp với chuỗi 64 bit vừa được gửi tới mới đáp ứng lại các lệnh về bộ nhớ tiếp theo. Còn các cảm biến DS1820 có 64 bit ROM không trùng khớp sẽ tiếp tục chờ một xung reset. Lệnh này được sử dụng cả trong trường hợp có một cảm biến một dây, cả trong trường hợp có nhiều cảm biến một dây.

- SKIP ROM (CCh)

Lệnh này cho phép thiết bị điều khiển truy nhập thẳng đến các lệnh bộ nhớ của DS1820 mà không cần gửi chuỗi mã 64 bit ROM. Như vậy sẽ tiết kiệm được thời gian chờ đợi nhưng chỉ mang hiệu quả khi chỉ có một cảm biến.

- SEARCH ROM (F0h)

Lệnh này cho phép bộ điều khiển bus có thể dò tìm được số lượng thành viên tớ đang được đấu vào bus và các giá trị cụ thể trong 64 bit ROM của chúng bằng một chu trình dò tìm.

- ALARM SEARCH (ECh)

Tiến trình của lệnh này giống hệt như lệnh Search ROM, nhưng cảm biến DS1820 chỉ đáp ứng lệnh này khi xuất hiện điều kiện cảnh báo trong phép đo nhiệt độ cuối cùng. Điều kiện cảnh báo ở đây được định nghĩa là giá trị nhiệt độ đo được lớn hơn giá trị TH và nhỏ hơn giá trị TL là hai giá trị nhiệt độ cao nhất và nhiệt độ thấp nhất đã được đặt trên thanh ghi trong bộ nhớ của cảm biến. Sau khi thiết bị chủ (thường là một vi điều khiển) sử dụng các lệnh ROM để định địa chỉ cho các cảm biến một dây đang được đấu vào bus, thiết bị chủ sẽ đưa ra các lệnh chức năng DS1820. Bằng các lệnh chức năng thiết bị chủ có thể đọc ra và ghi vào bộ nhớ nháp (scratchpath) của cảm biến DS1820. khởi tạo quá trình chuyển đổi giá trị nhiệt độ đo được và xác định chế độ cung cấp điện áp nguồn. Các lệnh chức năng có thể được mô tả ngắn gọn như sau:

- WRITE SCRATCHPAD (4Eh)

Lệnh này cho phép ghi 2 byte dữ liệu vào bộ nhớ nháp của DS1820. Byte đầu tiên được ghi vào thanh ghi TH (byte 2 của bộ nhớ nháp) còn byte thứ hai được ghi vào thanh ghi TL (byte 3 của bộ nhớ nháp). Dữ liệu truyền theo trình tự đầu tiên là bit có ý nghĩa nhất và kế tiếp là những bit có ý nghĩa giảm dần. Cả hai byte này phải được ghi trước khi thiết bị chủ xuất ra một xung reset hoặc khi có dữ liệu khác xuất hiện.

- READ SCRATCHPAD (BEh)

Lệnh này cho phép thiết bị chủ đọc nội dung bộ nhớ nháp. Quá trình đọc bắt đầu từ bit có ý nghĩa nhấy của byte 0 và tiếp tục cho đến byte rhứ 9 (byte 8 – CRC). Thiết bị chủ có thể xuất ra một xung reset để làm dừng quá trình đọc bất kỳ lúc nào nếu như chỉ có một phần của dữ liệu trên bộ nhớ nháp cần được đọc.

- COPYSCRATCHPAD (48h)

Lệnh này copy nội dung của hai thanh ghi TH và TL (byte 2 và byte 3) vào bộ nhớ EEPROM. Nếu cảm biến được sử dụng trong chế dộ cấp nguồn l bắt đầu việc đo.

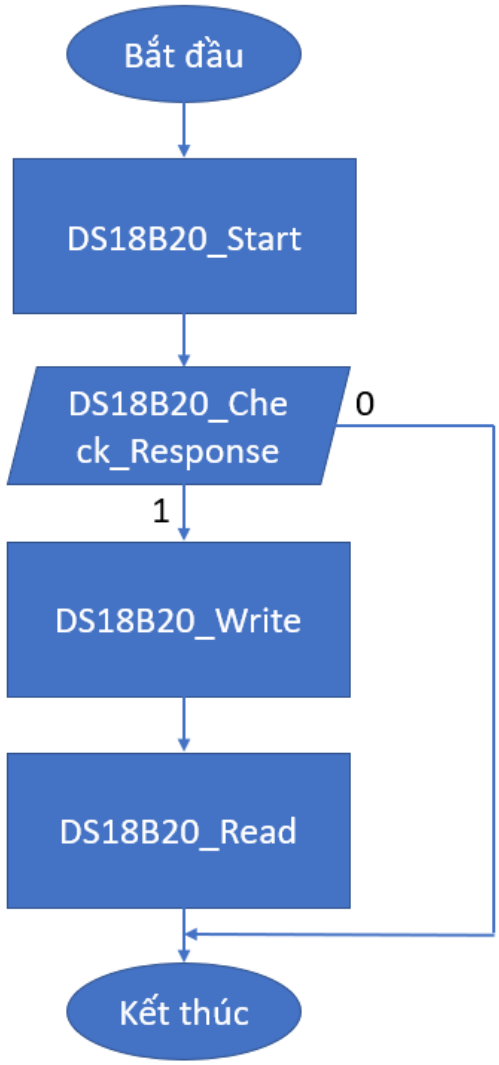
- CONVERT T (44h)

Lệnh này khởi động một quá trình đo và chuyển đổi giá trị nhiệt độ thành số (nhị phân). Sau khi chuyển đổi giá trị kết quả đo nhiệt độ được lưu trữ trên thanh ghi nhiệt độ 2byte trong bộ nhớ nháp Thời gian chuyển đổi không quá 200 ms, trong thời gian đang chuyển đổi nếu thực hiện lệnh đọc thì các giá trị đọc ra đều bằng 0.

- READ POWER SUPPLY (B4h)

Một lệnh đọc tiếp sau lệnh này sẽ cho biết DS1820 đang sử dụng chế độ cấp nguồn như thế nào, giá trị đọc được bằng 0 nếu cấp nguồn bằng chính đường dẫn dữ liệu và bằng 1 nếu cấp nguồn qua một đường dẫn riêng.

1.3.4 Lưu đồ thuật toán giao tiếp cảm biến DS18B20

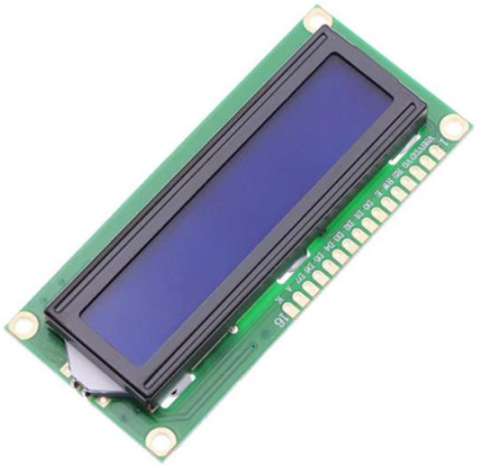


Hình 1.12 Lưu đồ thuật toán giao tiếp cảm biến DS18B20

1.4 Màn hình LCD 1602 và module LCD I2C

1.4.1 Giới thiệu chung

Màn hình LCD1602 xanh lá sử dụng Driver HD44780 có khả năng hiển thị 2 dòng, mỗi dòng 16 ký tự.



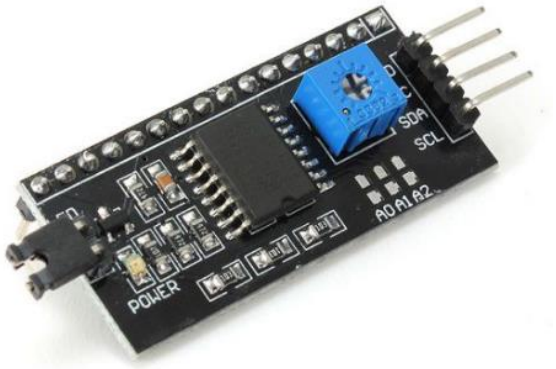
Hình 1.13 LCD 1602

Thông Số Kỹ Thuật Màn hình LCD1602 xanh lá:

- Điện áp hoạt động là 5 V.
- Kích thước: 80 x 36 x 12.5 mm.
- Chữ đen, nền xanh lá.
- Khoảng cách giữa hai chân kết nối là 0.1 inch.
- Tiện dụng khi kết nối với Breadboard.

Để sử dụng các loại LCD có driver là HD44780 (LCD 1602, LCD 2004,…) cần có ít nhất 6 chân của MCU kết nối với các chân RS, EN, D7, D6, D5 và D4 để có thể giao tiếp với LCD.

Nhưng với mạch chuyển đổi giao tiếp I2C cho LCD, chỉ cần 2 chân (SDA và SCL) của MCU kết nối với 2 chân (SDA và SCL) của module là đã có thể hiển thị thông tin lên LCD. Ngoài ra có thể điều chỉnh được độ tương phản bởi biến trở gắn trên module.



Hình 1.14 Module PCF8574

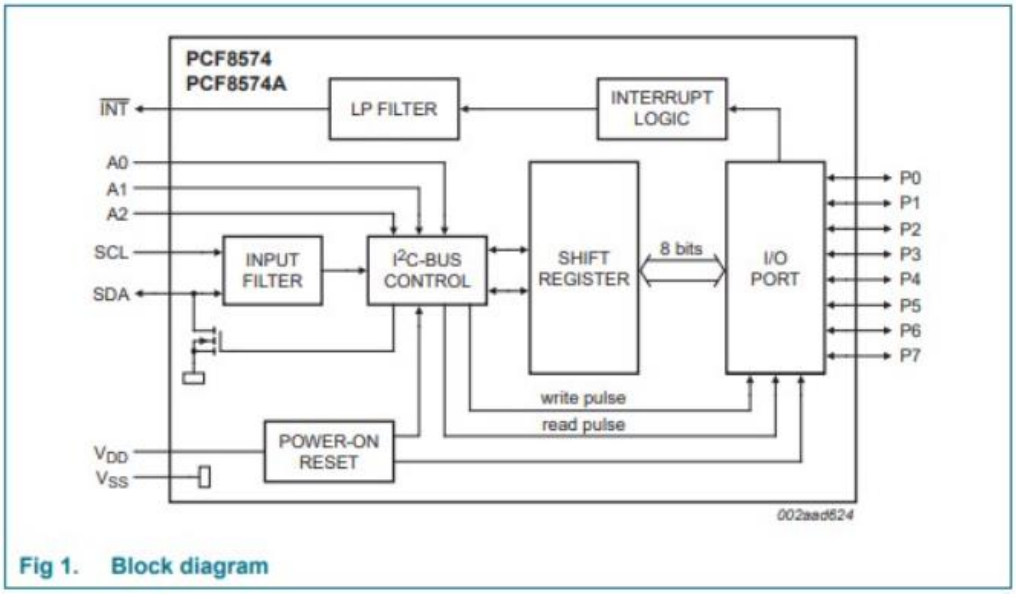
THÔNG SỐ MẠCH CHUYỂN ĐỔI GIAO TIẾP I2C

- Kích thước: 41.5mm(L)X19mm(W)X15.3MM(H).
- Trọng lượng: 5g.
- Điện áp hoạt động: 2.5v-6v.
- Jump chốt: Cung cấp đèn cho LCD hoặc ngắt.
- Biến trở xoay độ tương phản cho LCD.

1.4.2 Nguyên lý hoạt động

1.4.2.1. Module PCF8574

Với module PCF8574 chúng ta sẽ điều khiển tương tự như LCD chế độ 4 bit, nhưng thay vì phải xuất tín hiệu ra 8 chân RS, RW, EN, CS và D4-D7. Chúng ta chỉ cần xuất tín hiệu ra 2 chân SDA và SCL là được.



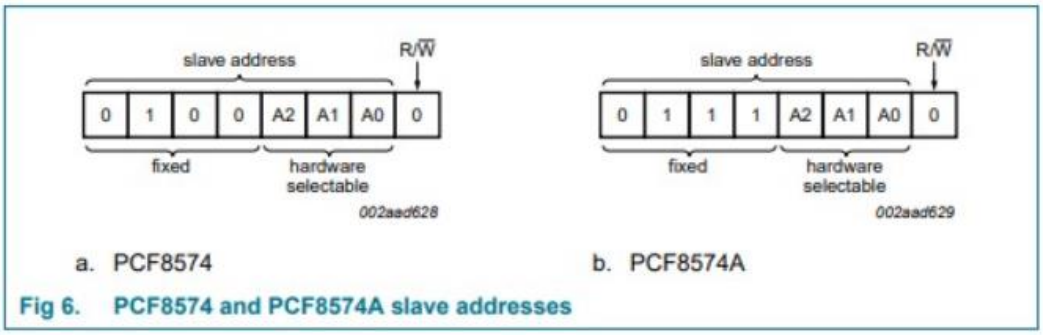
Hình 1.15 Sơ đồ khối module PCF8574

Nhìn vào sơ đồ khối của PCF8574 chúng ta thấy, đầu ra của nó gồm 8 chân P0-P7 Tương ứng với 4 bit RS, RW, BL (Back Light), EN và 4 bit Data D4 – D7 trên LCD.

Đầu vào sẽ gồm 2 chân SDA và SCL giao tiếp với vi điều khiển, 3 chân A0, A1, A2 để thay đổi địa chỉ cho PCF8574. Nghĩa là bạn có thể mắc nối tiếp 8 thiết bị PCF8574 trên cùng 1 bus I2C.

- Cách cấu hình địa chỉ cho Module LCD I2C PCF8574

Khi truyền nhận I2C, byte đầu tiên chúng ta luôn phải truyền vào địa chỉ của chip I2C cần giao tiếp, Byte đầu tiên trong PCF8574 được tổ chức như sau:



Hình 1.16 Địa chỉ module PCF8574

Các bit 4-7 được fix sẵn, chỉ có các bit 1 2 3 tương ứng với A2,A1,A0 có thể được sửa đổi, khi hàn các chân trên Board (Mặc định là 1 1 1). Để tính toán địa chỉ chúng ta dựa vào bảng sau:

Table 4. PCF8574 address map

Pin connectivity			Address of PCF8574								Address byte value		7-bit hexadecimal address without R/W
A2	A1	A0	A6	A5	A4	A3	A2	A1	A0	R/W	Write	Read	
V_{SS}	V_{SS}	V_{SS}	0	1	0	0	0	0	0	-	40h	41h	20h
V_{SS}	V_{SS}	V_{DD}	0	1	0	0	0	0	1	-	42h	43h	21h
V_{SS}	V_{DD}	V_{SS}	0	1	0	0	0	1	0	-	44h	45h	22h
V_{SS}	V_{DD}	V_{DD}	0	1	0	0	0	1	1	-	46h	47h	23h
V_{DD}	V_{SS}	V_{SS}	0	1	0	0	1	0	0	-	48h	49h	24h
V_{DD}	V_{SS}	V_{DD}	0	1	0	0	1	0	1	-	4Ah	4Bh	25h
V_{DD}	V_{DD}	V_{SS}	0	1	0	0	1	1	0	-	4Ch	4Dh	26h
V_{DD}	V_{DD}	V_{DD}	0	1	0	0	1	1	1	-	4Eh	4Fh	27h

Table 5. PCF8574A address map

Pin connectivity			Address of PCF8574A								Address byte value		7-bit hexadecimal address without R/W
A2	A1	A0	A6	A5	A4	A3	A2	A1	A0	R/W	Write	Read	
V_{SS}	V_{SS}	V_{SS}	0	1	1	1	0	0	0	-	70h	71h	38h
V_{SS}	V_{SS}	V_{DD}	0	1	1	1	0	0	1	-	72h	73h	39h
V_{SS}	V_{DD}	V_{SS}	0	1	1	1	0	1	0	-	74h	75h	3Ah
V_{SS}	V_{DD}	V_{DD}	0	1	1	1	0	1	1	-	76h	77h	3Bh
V_{DD}	V_{SS}	V_{SS}	0	1	1	1	1	0	0	-	78h	79h	3Ch
V_{DD}	V_{SS}	V_{DD}	0	1	1	1	1	0	1	-	7Ah	7Bh	3Dh
V_{DD}	V_{DD}	V_{SS}	0	1	1	1	1	1	0	-	7Ch	7Dh	3Eh
V_{DD}	V_{DD}	V_{DD}	0	1	1	1	1	1	1	-	7Eh	7Fh	3Fh

Hình 1.17 Bảng địa chỉ PCF8574

- Cách truyền dữ liệu cho module LCD I2C PCF8574

Lệnh ghi lên LCD I2C

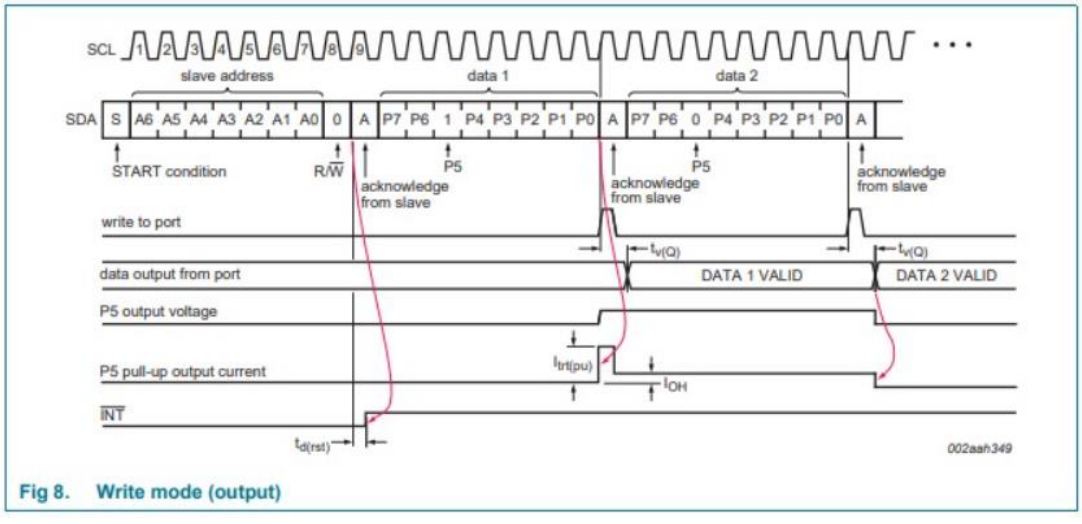


Fig 8. Write mode (output)

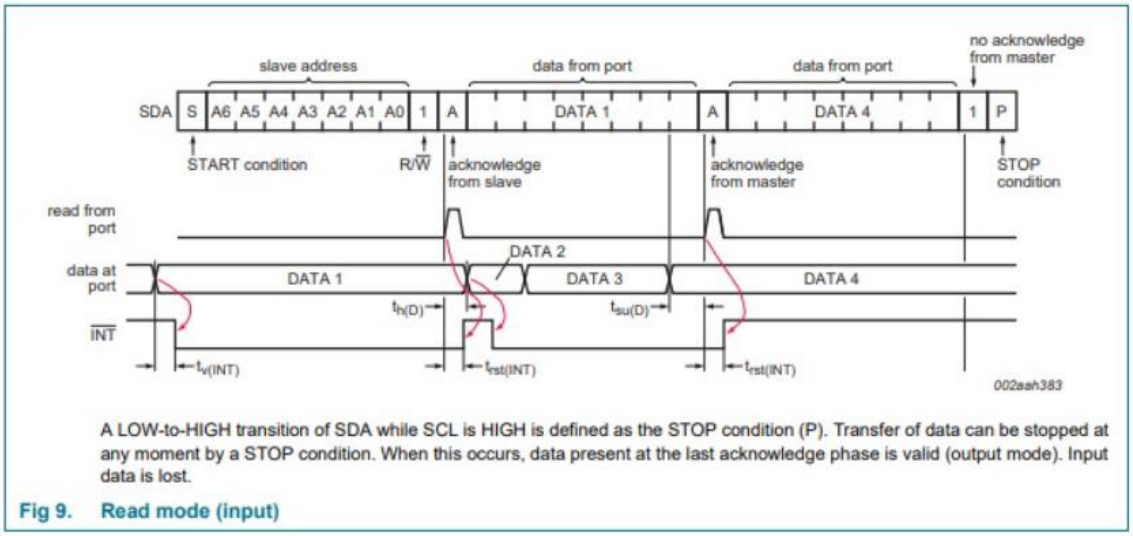
Hình 1.18 Ghi dữ liệu vào PCF8574

Để ghi lên LCD I2C chúng ta sẽ làm theo các bước:

<S> <slave address + write> <ACK> <data out> <ACK> <data out> <ACK> … … <data out> <ACK> <P>

Khi bắt đầu truyền dữ liệu, MCU sẽ truyền địa chỉ vào mạn I2C, nếu Module nào có cùng địa chỉ, chúng sẽ gửi ACK, sau đó MCU sẽ gửi các data tương ứng với Command và Paragram truyền vào LCD theo chế độ 4 BIT.

Lệnh đọc LCD I2C



A LOW-to-HIGH transition of SDA while SCL is HIGH is defined as the STOP condition (P). Transfer of data can be stopped at any moment by a STOP condition. When this occurs, data present at the last acknowledge phase is valid (output mode). Input data is lost.

Fig 9. Read mode (input)

Hình 1.19 Đọc dữ liệu từ PCF8574

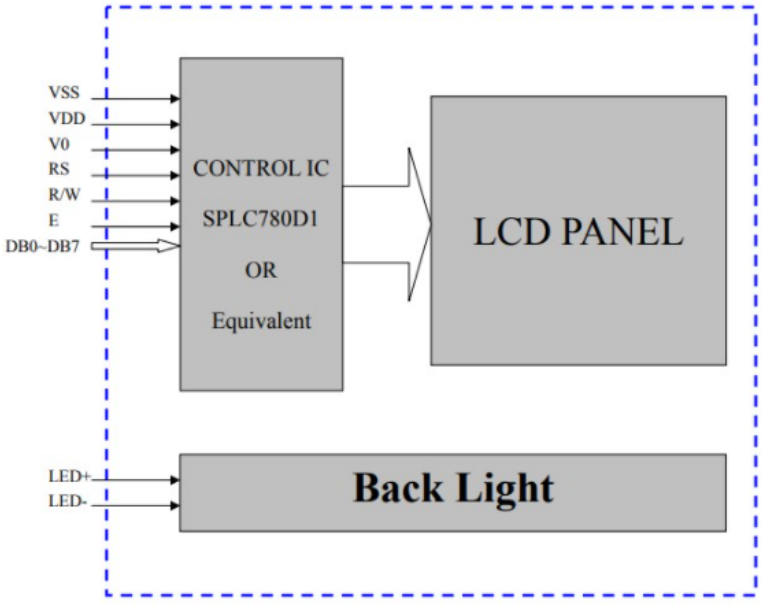
Để đọc từ LCD I2C chúng ta sẽ làm theo các bước:

<S> <slave address + read> <ACK> <data in> <ACK> <data in> <ACK> … … <data in> <ACK> <P>

1.4.2.2. LCD1602

Màn hình LCD1602 cấu tạo từ 3 phần:

- Chip driver: Điều khiển LCD, giao tiếp với vi điều khiển theo interface LCD
- LCD Panel: Hiển thị ra bên ngoài
- Back Light: Đèn nền LCD



Hình 1.20 Sơ đồ khối LCD1602

- DISPLAY DATA RAM (DD RAM): Bộ nhớ hiển thị dữ liệu

Điều khiển LCD1602 chính là thay đổi giá trị của DD RAM, mỗi ô trên DD RAM tương ứng với một vị trí của màn hình.

Với LCD1602 chúng có 2 line

Line 1: từ 0x80 tới 0x8F

Line 2: từ 0xC0 tới 0xCF

Ví dụ: Nếu bạn muốn hiển thị ở Line 1 ô đầu tiên, Chúng ta sẽ thay đổi giá trị của ô nhớ địa chỉ 0x80. Giá trị được ghi sẽ so sánh với bảng mã trong CG ROM, từ đó hiển thị ra đúng kí tự được lưu trên đó.

Lệnh để nhảy giữa các ô nhớ là Set cursor (con trỏ)

- Character Generator ROM (CG ROM): Bộ nhớ kí tự chỉ đọc

Đây là bộ nhớ đươc ghi sẵn của LCD, trong đó chứa các kí tự mà lcd hỗ trợ. Có hai mẫu Character mà LCD1602 hỗ trợ đó là 5×8 và 5×10

- Character Generator RAM (CG RAM): Bộ nhớ kí tự có thể lập trình

Đây là bộ nhớ để người sử dụng có thể tự tạo ra các font chữ riêng trên LCD của mình.

2 LINES X 16 CHARACTERS PER LINE

Char.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Hình 1.21 Sơ đồ display Ram LCD1602

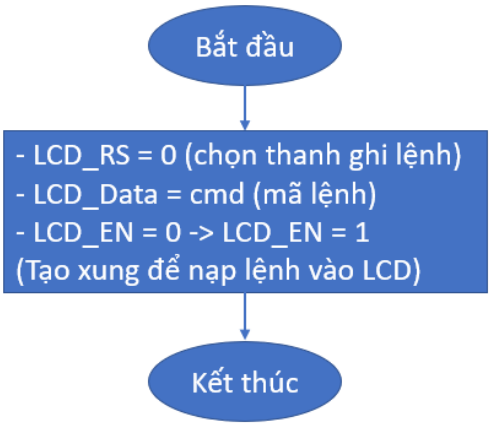
Quy trình ghi vào LCD1602 như sau:

- Chân RS kéo xuống 0 nếu gửi Lệnh (Command) điều khiển LCD, lên 1 gửi data (ghi vào DD RAM)
- Chân R/W: Kéo xuống 0 ghi dữ liệu
- Các chân D0 – D7: Khi ghi dữ liệu, các chân D0-D7 của LCD sẽ ở chế độ Input, chân MCU kết nối vào sẽ ở chế độ OutPut
- Chân EN sẽ được kéo lên 1 để chốt dữ liệu vào LCD, sau đó lại nhả về 0

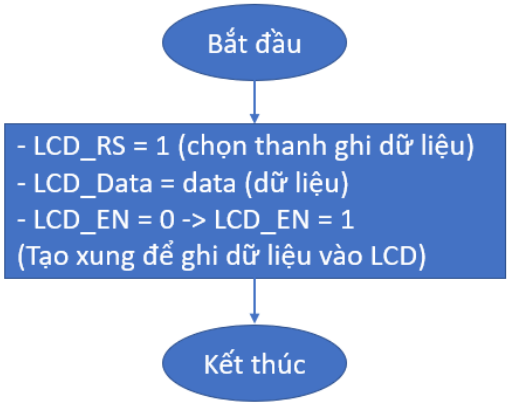
Quy trình đọc LCD1602:

- Chân RS kéo xuống 0 nếu gửi Lệnh (Command) điều khiển LCD, lên1 gửi data (ghi vào DD RAM)
- Chân R/W: Kéo lên 1 để đọc dữ liệu
- Các chân D0 – D7: Khi đọc dữ liệu, các chân D0-D7 của LCD sẽ ở chế độ Output, chân MCU kết nối vào sẽ ở chế độ Input
- Chân EN sẽ được kéo lên 1 để chốt dữ liệu vào LCD, sau đó lại nhả về 0

1.4.3 Lưu đồ thuật toán giao tiếp LCD I2C



Hình 1.22 Lưu đồ ghi lệnh trên LCD



Hình 1.23 Lưu đồ ghi dữ liệu trên LCD

1.5 Vi điều khiển STM32F103C8T6

1.5.1 Giới thiệu chung

Vi điều khiển STM32 hiện nay là 1 dòng vi điều khiển rất phổ biến ngoài thị trường. Nó là một loại vi điều khiển 32bit với rất nhiều ưu điểm vượt trội hơn so với các dòng vi điều khiển 8bit, 16bit khác. Trong đó có thể kể đến các đặc điểm nổi bật như:

- Hoạt động ở tần số cao 72 MHz
- Hỗ trợ RTC
- Tích hợp ADC độ phân giải cao
- Hỗ trợ điều khiển hoạt động sử dụng hệ điều hành thời gian thực (RTOS)

Sau quá trình xem xét về số cổng giao tiếp, số chân vào ra cần sử dụng, em chọn vi điều khiển STM32F103C8T6 – LQFP48 pinout.

Hình 1.24 STM32F103C8T6

Bảng 1.1 Thông số về điện áp cấp cho vi điều khiển

Thông số	Mô tả
Dải điện áp nguồn cấp	-0.3 – 4.0 V
Điện áp vào trên các chân chịu áp	VSS -0.3 – VDD + 4.0 V
Điện áp vào trên các chân khác	VSS -0.3 – 4.0 V
Điện áp chân dự phòng V_{BAT}	1.8 – 3.6 V
Dòng điện nguồn cấp cực đại	50 mA
Tần số hoạt động tối đa	72 Mhz

Bảng 1.2 Các ngoại vi hỗ trợ bởi vi điều khiển

Thông số	STM32F103C8T6
USART	3
SPI	2
I2C	2
ADC	2 bộ ADC 12 bit, mỗi bộ 9 kênh
DAC	2 kênh DAC 12 bit
Timer	7
Bộ nhớ Flash	64KB

1.5.2 Các ngoại vi của STM32 cần sử dụng

1.5.2.1. RCC

Khác với nhiều dòng vi điều khiển 8-bit cũ, nhiều dòng vi điều khiển hiện đại cần phải config clock trước khi sử dụng các ngoại vi. Mặc định, tất cả clock của các thiết bị ngoại vi sẽ bị vô hiệu hóa để tiết kiệm năng lượng.

Đối với VĐK STM32, clock được cấu hình qua thanh ghi RCC

Vi điều khiển STM32F103 có ba nguồn cấp xung clock chính đó là:

- Nguồn Clock dao động nội tốc độ cao (HSI – High Speed Internal): lấy từ bộ dao động RC nội, mặc định là 8MHz.
- Nguồn Clock dao động ngoại tốc độ cao (HSE – High Speed External): lấy từ nguồn dao động thạch anh.
- Nguồn Clock PLL (PLL – Phase Lock Loop).

Các ngoại vi được cấp xung clock thông qua ba đường bus sau:

AHB (Advanced High Speed Buses): Đây là Bus kết nối hệ thống.

APB1, APB2 (Advanced Peripheral Buses 1,2): Đây là các Bus kết nối với thiết bị ngoại vi và kết nối với hệ thống thông qua AHB.

Một số thanh ghi cấu hình Clock

- Clock control register (RCC_CR)
- Clock configuration register (RCC_CFGR)
- Clock interrupt register (RCC_CIR)
- APB2 peripheral reset register (RCC_APB2RSTR)
- APB1 peripheral reset register (RCC_APB1RSTR)
- AHB Peripheral Clock enable register (RCC_AHBENR)
- APB2 peripheral clock enable register (RCC_APB2ENR)
- APB1 peripheral clock enable register (RCC_APB1ENR)
- Backup domain control register (RCC_BDCR)
- Control/status register (RCC_CSR)
- AHB peripheral clock reset register (RCC_AHBRSTR)
- Clock configuration register2 (RCC_CFGR2)

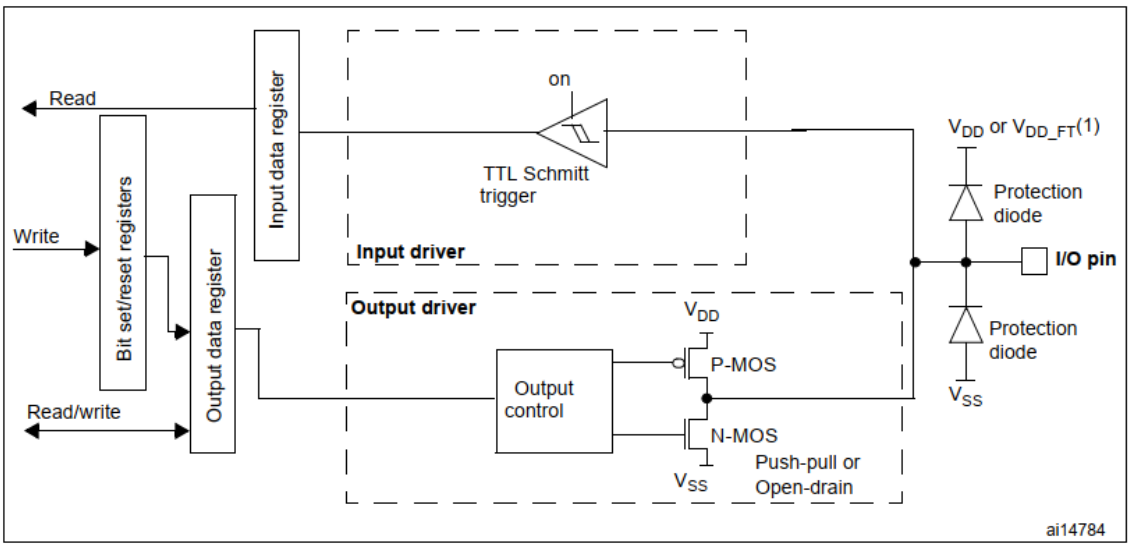
1.5.2.2. GPIO

GPIO (General Purpose Input/Output) chính là các chân Đầu ra hoặc đầu vào dùng chung.

Tại sao lại là dùng chung?

Bởi vì mỗi chân GPIO ngoài các chức năng Input/Ouput chúng còn có thêm các chức năng khác ứng với các ngoại vi khác. Ví dụ: I2C, SPI, UART, ADC… Tất cả các chân này đều có thể sử dụng như 1 cổng I/O để đọc hoặc xuất dữ liệu theo Bit.

STM32 GPIO bao gồm nhiều Port, mỗi Port có tối đa 16 chân.



Hình 1.25 Cấu hình Input/Output

STM32 GPIO bao gồm 2 khối cơ bản:

Input Driver: Bao gồm thanh ghi Input Data (IDR), và 1 trigger. Tín hiệu Input ngoài việc được ghi vào IDR còn theo các đường Analog để vào bộ ADC, hoặc theo đường Alternate function input vào các ngoại vi khác.

Output Drive: Bao gồm thanh ghi Output Data (ODR), một khối output control để chọn tín hiệu ra là từ ODR hay từ các ngoại vi khác. Tiếp đến điều khiển 2 mosfet cho điện áp ra ở I/O pin

Chức năng của STM32 GPIO bao gồm:

Input:

- Input pull up: Đầu vào có trở kéo lên (điện áp mặc định trên chân là Vcc).
- Input pull down: Đầu vào có trở kéo xuống (điện áp mặc định trên chân là 0V).
- Input floating: Đầu vào thả nổi, điện áp không cố định dao động từ 0V tới Vcc.
- Analog: Đầu vào tương tự, dùng để đo ADC.

Output:

- Ouput Push Pull: Đầu ra dạng đẩy kéo, tín hiệu sẽ chỉ có Vcc hoặc 0V tương ứng với Bit 1 và 0 ghi vào chân đó
- Ouput Opendrain: Đầu ra dạng cực máng hở. Chỉ có thể kéo về 0V bằng cách ghi bit 0, khi ghi bit 1, chân IO sẽ có điện áp tương ứng với nguồn nối vào IO đó
- Alternate function Push Pull: Đầu ra kểu đẩy kéo sử dụng trong các ngoại vi
- Alternate function Open Drain: Đầu ra dạng cực máng hở, sử dụng trong các ngoại vi (thường gặp trong I2C)

Một số thanh ghi cấu hình GPIO

Port configuration register low (GPIOx_CRL) (x=A..G)
Port configuration register high (GPIOx_CRH) (x=A..G)
Port input data register (GPIOx_IDR) (x=A..G)
Port output data register (GPIOx_ODR) (x=A..G)
Port bit set/reset register (GPIOx_BSRR) (x=A..G)
Port bit reset register (GPIOx_BRR) (x=A..G)
Port configuration lock register (GPIOx_LCKR) (x=A..G)

Hình 1.26 Thanh ghi cấu hình GPIO

1.5.2.3. Timer

Các chức năng chính của Timer STM32 bao gồm:

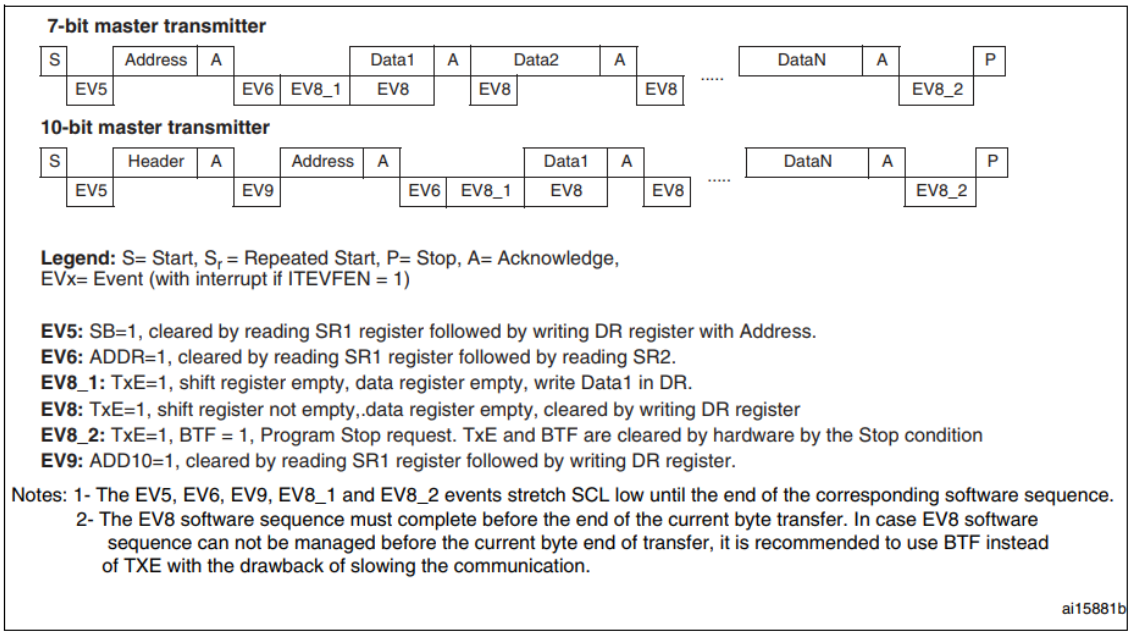
- Thanh ghi 16 bit đếm lên, xuống, lên/xuống tự nạp lại.
- 16 bit bộ chia tần số để chia tần số từ APB(giá trị dao động từ 1 – 65536).
- 4 Kênh độc lập mỗi Timer cho các chức năng: Input Capture, Output Compare, One Pulse.
- Đồng bộ hóa với các mạch tạo tín hiệu bên ngoài để kết hợp nhiều bộ Timer với nhau
- Ngắt/DMA.
- Hỗ trợ điều khiển Encoder và Hall-sensor.

1.5.2.4. I2C

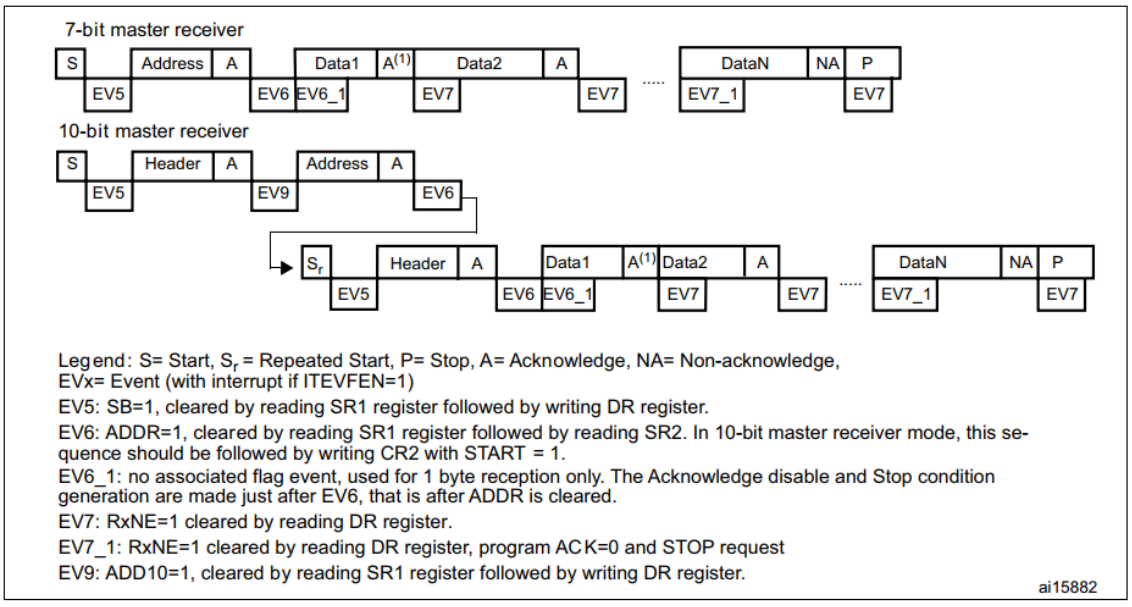
Cách truyền dữ liệu của giao thức I2C

Dữ liệu được truyền đi trên dây SDA được thực hiện như sau:

- Master thực hiện điều kiện bắt đầu I2C (Start Condition).
- Gửi địa chỉ 7 bit + 1bit Đọc/Ghi (R/W) để giao tiếp muốn đọc hoặc ghi dữ liệu tại Slave có địa chỉ trên.
- Nhận phải hồi từ Bus, nếu có một bit ACK (Kéo SDA xuống thấp) Master sẽ gửi dữ liệu.
- Nếu là đọc dữ liệu R/W bit = 1, chân SDA của master sẽ là input, đọc dữ liệu từ Slave gửi về. Nếu là ghi dữ liệu R/W = 0, chân SDA sẽ là output ghi dữ liệu vào Slave.
- Truyền điều khiện kết thúc (Stop Condition).



Hình 1.27 Master gửi dữ liệu



Hình 1.28 Master nhận dữ liệu

1.5.2.5. USART

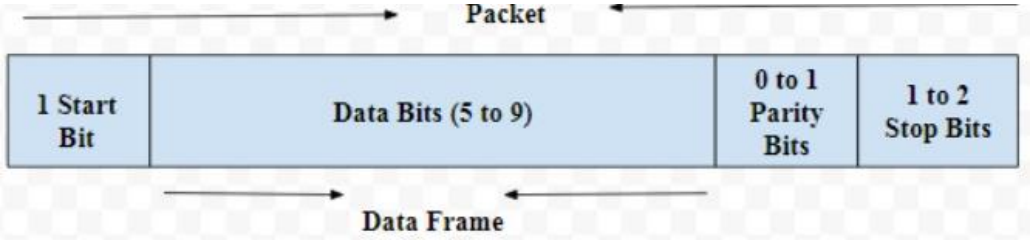
UART là giao thức truyền thông không đồng bộ, nghĩa là không có xung Clock, các thiết bị có thể hiểu được nhau nếu các Setting giống nhau

UART là truyền thông song công(Full duplex) nghĩa là tại một thời điểm có thể truyền và nhận đồng thời.

Trong đó quan trọng nhất là Baund rate (tốc độ Baund) là khoảng thời gian dành cho 1bit được truyền. Phải được cài đặt giống nhau ở gửi và nhận.

Sau đó là định dạng gói tin.

Định dạng gói tin như sau:



Hình 1.29 Định dạng gói tin UART

Start – Bit

Start-bit còn được gọi là bit đồng bộ hóa được đặt trước dữ liệu thực tế. Nói chung, một đường truyền dữ liệu không hoạt động được điều khiển ở mức điện áp cao. Để bắt đầu truyền dữ liệu, truyền UART kéo đường dữ liệu từ mức điện áp cao (1) xuống mức điện áp thấp (0). UART thu được thông báo sự chuyển đổi này từ mức cao sang mức thấp qua đường dữ liệu cũng như bắt đầu hiểu dữ liệu thực. Nói chung, chỉ có một start-bit.

Stop – Bit

Bit dừng được đặt ở phần cuối của gói dữ liệu. Thông thường, bit này dài 2 bit nhưng thường chỉ sử dụng 1 bit. Để dừng sóng, UART giữ đường dữ liệu ở mức điện áp cao.

Partity Bit

Bit chẵn lẻ cho phép người nhận đảm bảo liệu dữ liệu được thu thập có đúng hay không. Đây là một hệ thống kiểm tra lỗi cấp thấp & bit chẵn lẻ có sẵn trong hai phạm vi như Chẵn lẻ – chẵn lẻ cũng như Chẵn lẻ – lẻ. Trên thực tế, bit này không được sử dụng rộng rãi nên không bắt buộc.

Data frame

Các bit dữ liệu bao gồm dữ liệu thực được truyền từ người gửi đến người nhận. Độ dài khung dữ liệu có thể nằm trong khoảng 5 & 8. Nếu bit chẵn lẻ không được sử dụng thì chiều dài khung dữ liệu có thể dài 9 bit. Nói chung, LSB của dữ liệu được truyền trước tiên sau đó nó rất hữu ích cho việc truyền.

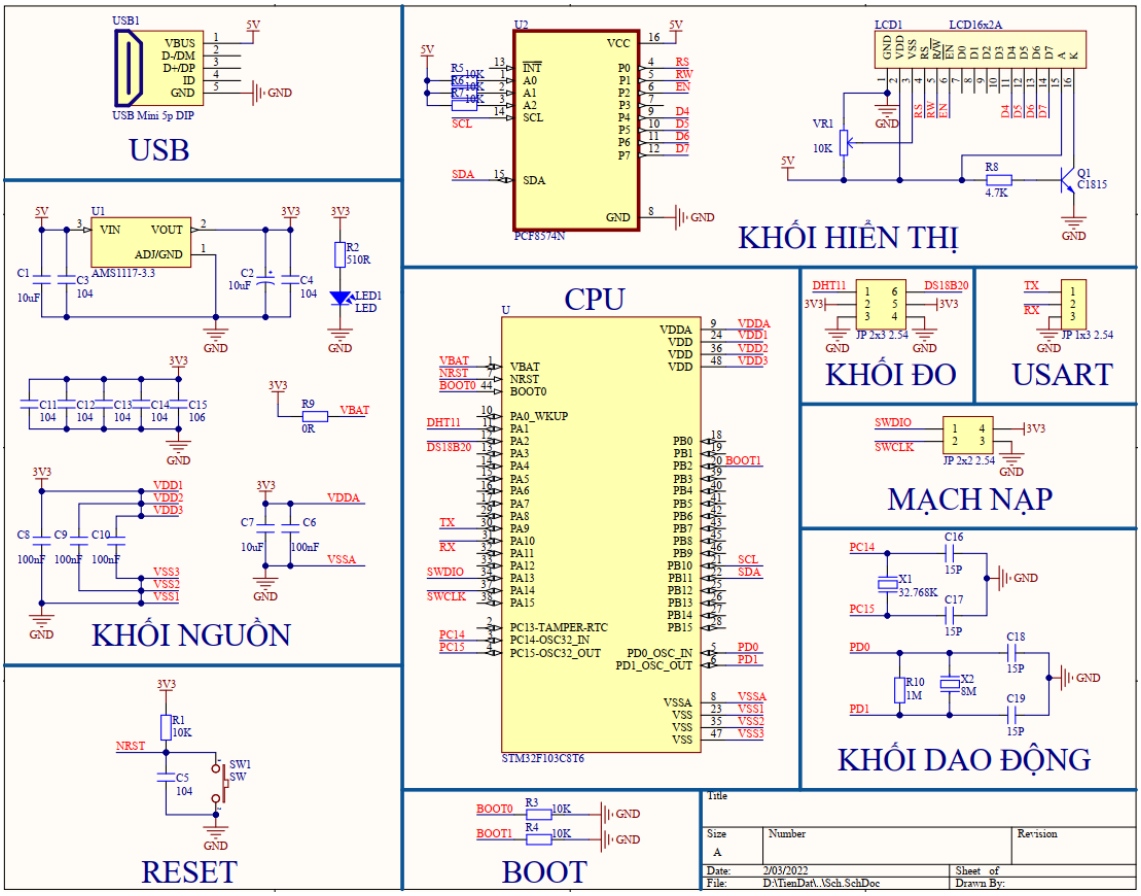
Một số thanh ghi cấu hình USART

Status register (USART_SR)
Data register (USART_DR)
Baud rate register (USART_BRR)
Control register 1 (USART_CR1)
Control register 2 (USART_CR2)
Control register 3 (USART_CR3)
Guard time and prescaler register (USART_GTPR)

Hình 1.30 Thanh ghi cấu hình USART

CHƯƠNG 2. THIẾT KẾ CHI TIẾT PHẦN CỨNG HỆ THỐNG

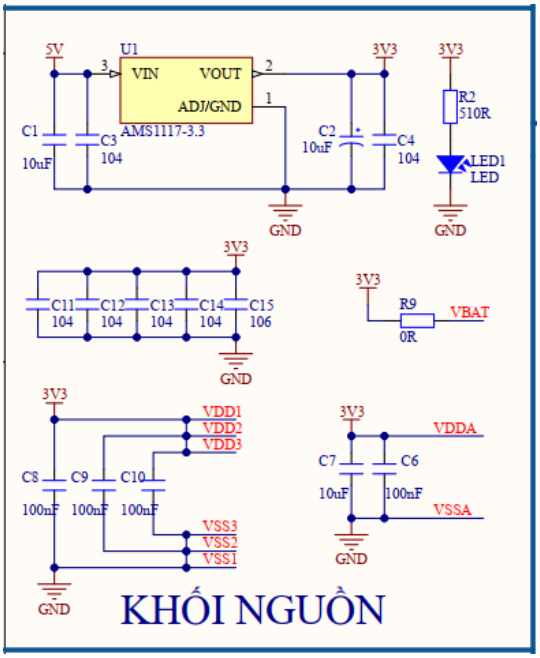
2.1 Sơ đồ nguyên lý hệ thống



Hình 2.1 Sơ đồ nguyên lý hệ thống

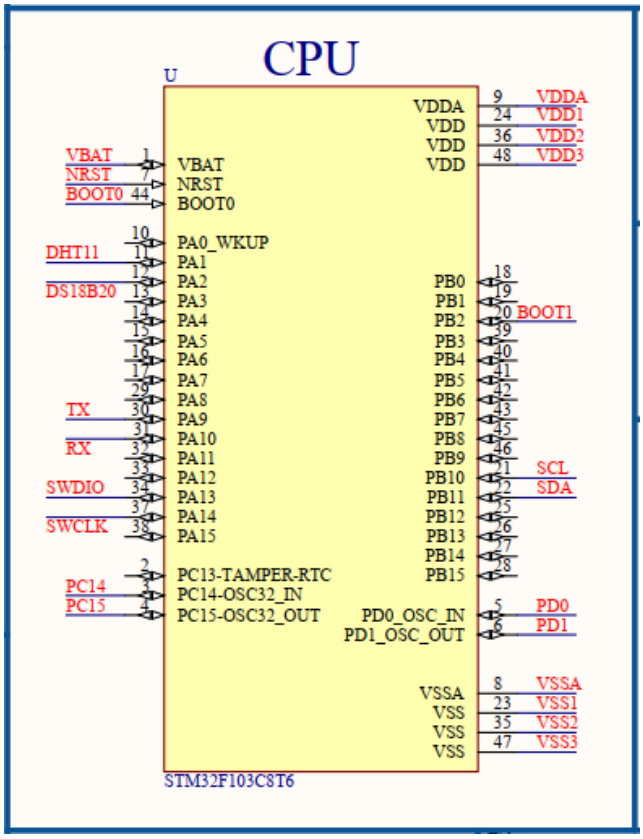
2.2 Khối nguồn

Nguồn 5VDC được cấp từ USB, để chuyển đổi từ nguồn 5VDC thành 3.3VDC, sử dụng IC LM1117-3.3V.



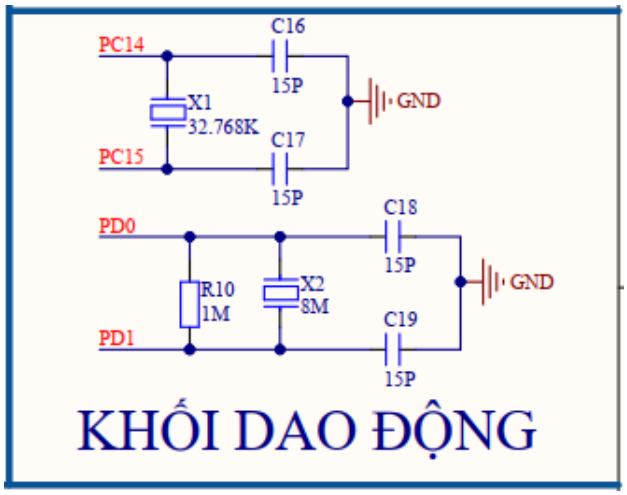
Hình 2.2 Khối nguồn

2.3 MCU



Hình 2.3 STM32F103C8T6

2.3.1 Mạch tạo dao động

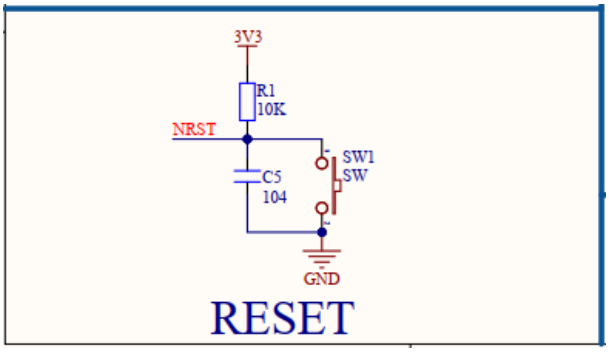


Hình 2.4 Sơ đồ khối tạo dao động

Bên trong vi điều khiển STM32F100C8T6 đã tích hợp High-speed Internal Clock và Low-speed Internal Clock. Tuy nhiên để tần số chính xác hơn thì ta nên sử dụng các Clock ngoài. Cụ thể là mạch để tạo High-speed External Clock (HSE) và Low-speed External Clock (LSI) cho vi điều khiển.

Đối với mạch tạo HSE, sử dụng một thạch anh ngoài 8 MHz. Đối với tụ C18 và C19 thì nhà sản xuất khuyến nghị sử dụng tụ gốm bên ngoài chất lượng cao trong phạm vi từ 5pF đến 25pF. Vì vậy sử dụng 2 tụ gốm với điện dung là 15pF.

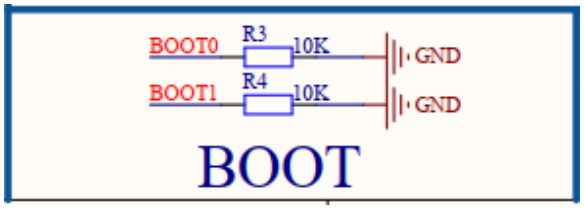
2.3.2 Mạch Reset



Hình 2.5 Mạch Reset

Mạch bao gồm 1 nút ấn với chân đầu ra sử dụng điện trở Pull Up R1 = 10k để giữ mức điện áp cao trên chân NRST của vi điều khiển. Tụ gốm C5 dùng để chống dội cho nút ấn, giá trị tụ là 100nF (104) dựa theo khuyến nghị của nhà sản xuất.

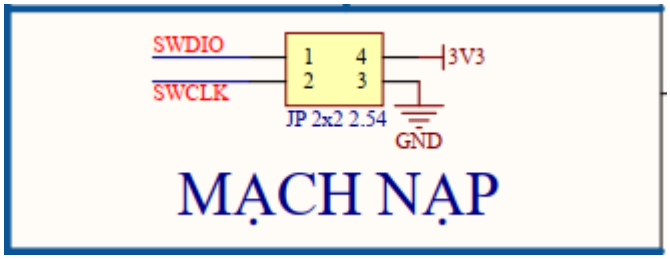
2.3.3 Mạch Boot



Hình 2.6 Mạch Boot

2 chân BOOT0 và BOOT1 được nối xuống đất để chương trình sau khi nạp sẽ được thực thi ở FLASH.

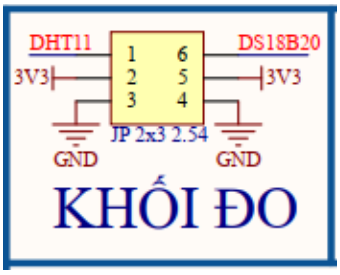
2.3.4 Mạch nạp



Hình 2.7 Mạch nạp

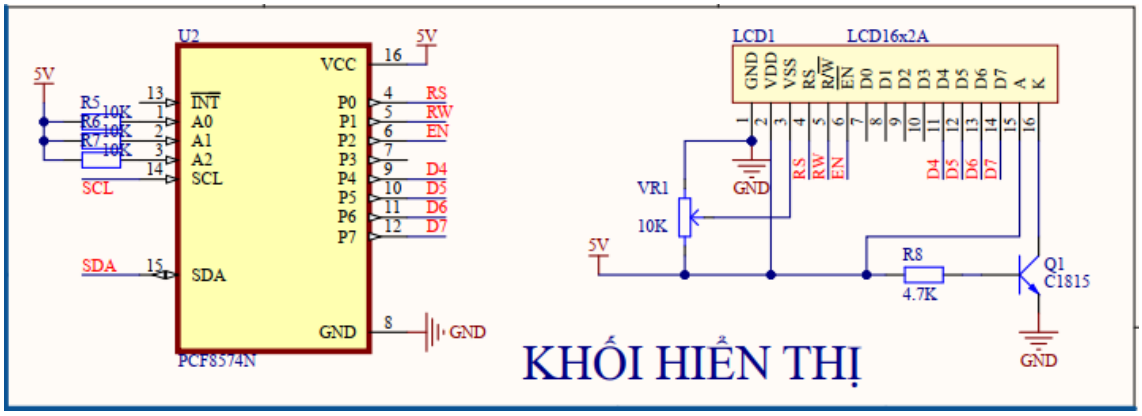
Đối với vi điều khiển STM32F103C8T6, có thể sử dụng mạch nạp ST-Link V2 để nạp code cho vi điều khiển chạy qua các chân SWCLK và SWDIO.

2.4 Khối đo



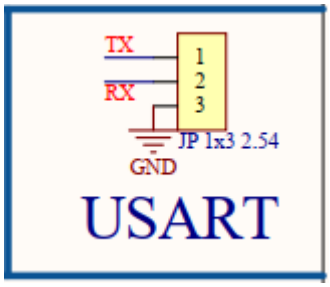
Hình 2.8 Khối đo

2.5 Khối hiển thị



Hình 2.9 Khối hiển thị

2.6 Khối truyền thông với máy tính



Hình 2.10 Khối truyền thông với máy tính

CHƯƠNG 3. THIẾT KẾ PHẦN MỀM CHO HỆ THỐNG

3.1 Thiết kế bộ lập lịch với chu kỳ lặp lại khác nhau

Chương trình hệ thống sẽ bao gồm 4 Task với chu kỳ lặp lại khác nhau được cho trong bảng sau:

Bảng 3.1 Chu kỳ lặp lại các Task

τ_i	T_i(s)	D_i(s)	C_i(s)	Tên Task
τ_1	12	4	3	Task_GetData_DS18B20
τ_2	10	6	1	Task_GetData_DHT11
τ_3	6	6	2	Task_Display_LCD
τ_4	15	9	2	Task_Send_To_PC

Trong đó:

- τ_i: các Task cần thực hiện.
- T_i: chu kỳ lặp lại của Task.
- D_i: khoảng thời gian kết thúc Task.
- C_i: thời gian thực thi Task.

Xác định chu kỳ P và độ dài Frame:

- 1 Task được thực thi 1 lần trong frame:

$$f \leq T_i \ \forall \ \tau_i \qquad PT\ 3.1$$

- P là bội của f.
- P là bội của T_i.
- Các task phải thực thi xong trong 1 frame:

$$f \geq C_i \ \forall \ \tau_i \qquad PT\ 3.2$$

- Giữa thời gian bắt đầu và kết thúc của 1 Task phải có ít nhất 1 frame:

$$2f - \gcd(T_i, f) \leq D_i \qquad PT\ 3.3$$

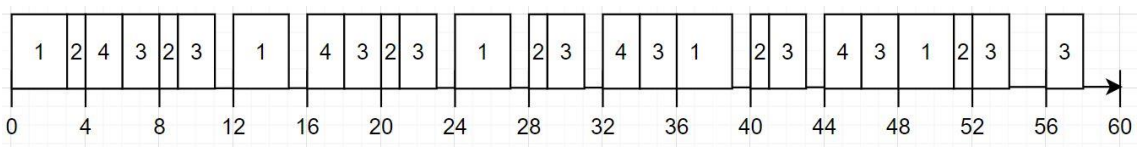
Từ các phương trình trên ta có: P = 60, f = 4.

Xác định các mốc thời gian bắt đầu ($r_{i,j}$) và kết thúc ($d_{i,j}$) của Task:

$$r_{i,j} = \emptyset_i + (j - 1)\, T_i \qquad PT\ 3.4$$

$$d_{i,j} = \emptyset_i + (j - 1)\, T_i + D_i \qquad PT\ 3.5$$

Từ đó ta có bộ lập lịch như sau:



Hình 3.1 Bộ lập lịch

3.2 Xây dựng chương trình theo bộ lập lịch

Main:

- Xác định bộ lập lịch: (t(k), T(k)), với k = 0, 1, … , n – 1;
- Xác định chu kỳ P;
- Khởi tạo i = k = 0, khởi tạo ngắt timer tại thời điểm t(0);
- While(true) sleep();

Ngắt timer:

- k_old := k;
- i := i + 1, k := i % n;
- Đặt ngắt timer tại thời điểm [i / n] * P + t(k);
- Thực hiện Task T(k_old);

Từ bộ lập lịch ta có bảng sau:

Bảng 3.2 Bộ lập lịch

k	t(k)	T(k)
0	0	T1
1	3	T2
2	4	T4
3	6	T3
4	8	T2
5	9	T3
6	12	T1
7	16	T4
8	18	T3
9	20	T2
10	21	T3
11	24	T1
12	28	T2
13	29	T3
14	32	T4
15	34	T3
16	36	T1
17	40	T2
18	41	T3
19	44	T4
20	46	T3
21	48	T1
22	51	T2
23	52	T3
24	56	T3

Từ thuật toán trên ta viết chương trình như sau:

Sau khi khởi tạo các cấu hình ban đầu, hệ thống sẽ đi vào trạng thái Sleep, mỗi khi có ngắt sẽ thức dậy và thực hiện các Task theo bộ lập lịch.

```
int main(void)
{
  Project_Init();
  while(1)
  {
    __WFI();
  }
}
```

Hình 3.2 Hàm main

```
uint8_t k = 0;
uint8_t k_old = 0;
const uint8_t n = 25;
const uint32_t P = 600000;
uint32_t i = 0;
/* Private function prototypes -----------------------------------------------*/
void (*sptt_task_point_array[]) (void) = {Task_GetData_DS18B20, Task_GetData_DHT11, Task_Send_To_PC,
Task_Display_LCD, Task_GetData_DHT11, Task_Display_LCD, Task_GetData_DS18B20, Task_Send_To_PC, Task_Display_LCD,
Task_GetData_DHT11, Task_Display_LCD, Task_GetData_DS18B20, Task_GetData_DHT11, Task_Display_LCD, Task_Send_To_PC,
Task_Display_LCD, Task_GetData_DS18B20, Task_GetData_DHT11, Task_Display_LCD, Task_Send_To_PC, Task_Display_LCD,
Task_GetData_DS18B20, Task_GetData_DHT11, Task_Display_LCD, Task_Display_LCD};
const uint32_t t[25] = {0,30000,40000,60000,80000,90000,120000,160000,180000,200000,210000,240000,280000,290000,
320000,340000,360000,400000,410000,440000,460000,480000,510000,520000,560000};
```

Hình 3.3 Khởi tạo các giá trị ban đầu cho bộ lập lịch

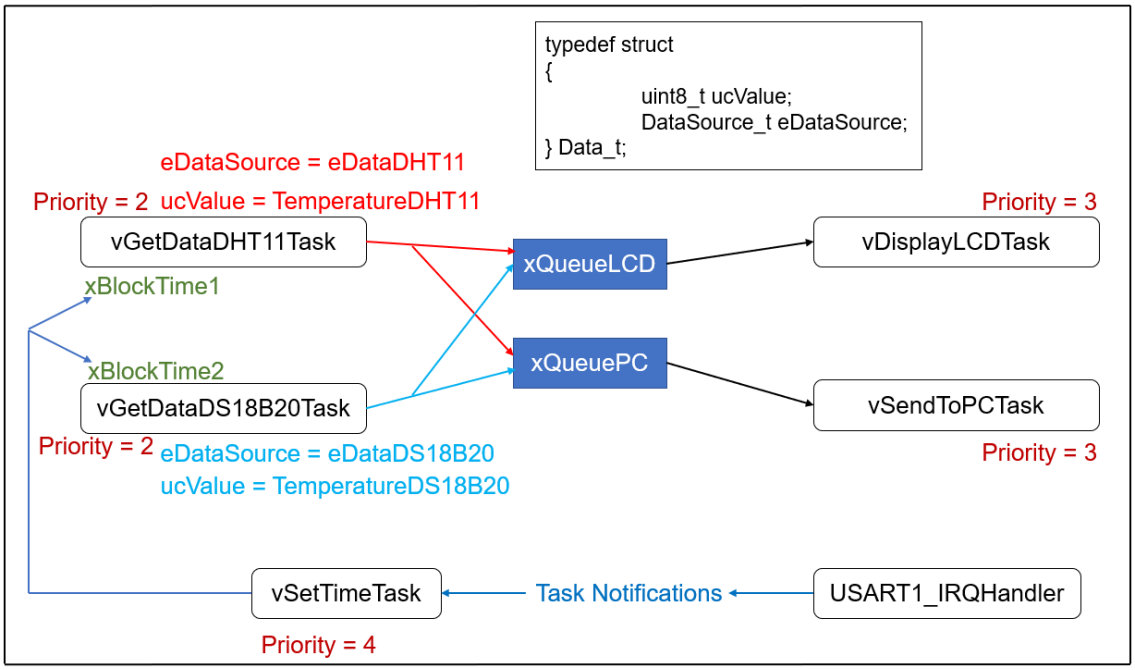
Chương trình ngắt sẽ cập nhật các thời điểm ngắt tiếp theo và thực hiện các Task.

```
void TIM3_IRQHandler(void)
{
  if (TIM_GetITStatus(TIM3, TIM_IT_CC1) != RESET)
  {
    TIM_ClearITPendingBit(TIM3, TIM_IT_CC1);
    k_old = k;
    i = i + 1;
    k = i % n;
    TIM_SetCompare1(TIM3, (i/n)*P + t[k]);
    sptt_task_point_array[k_old]();
  }
}
```

Hình 3.4 Chương trình ngắt

3.3 Xây dựng chương trình theo hệ điều hành FreeRTOS

Chương trình sẽ bao gồm 5 Task và sử dụng 2 hàng đợi để đồng bộ dữ liệu với nhau, mối quan hệ được thể hiện như hình dưới đây:



Hình 3.5 Trao đổi dữ liệu giữa các Task

vSetTimeTask có mức ưu tiên cao nhất và chỉ được chạy khi có dữ liệu từ UART gửi xuống, dùng để thay đổi chu kỳ lặp lại của vGetDataDHT11Task và vGetDataDS18B20Task.

vDisplayLCDTask và vSendToPCTask có cùng mức ưu tiên, và luôn trong trạng thái chờ dữ liệu từ hàng đợi, bất cứ khi nào có dữ liệu thì chúng sẽ được thực thi.

vGetDataDHT11Task và vGetDataDS18B20 được thi thi liên tục với chu kỳ lặp lại là xBlockTime1 và xBlockTime2.

Xây dựng chương trình:

Hàm main được dùng để khởi tạo các cấu hình cần thiết của hệ thống, sau đó khởi tạo hàng đợi và các Task.

```
int main( void )
{
  Project_Init();
  xQueueLCD = xQueueCreate( 1, sizeof( Data_t ) );
  xQueuePC = xQueueCreate( 1, sizeof( Data_t ) );
  if( (xQueueLCD != NULL) && (xQueuePC != NULL) )
  {
    xTaskCreate( vGetDataDHT11Task, "vGetDataDHT11Task", 500, NULL, 2, &xGetDataDHT11Task );
    xTaskCreate( vGetDataDS18B20Task, "vGetDataDS18B20Task", 500, NULL, 2, &xGetDataDS18B20Task );
    xTaskCreate( vDisplayLCDTask, "vDisplayLCDTask", 500, NULL, 3, &xDisplayLCDTask );
    xTaskCreate( vSendToPCTask, "vSendToPCTask", 500, NULL, 3, &xSendToPCTask );
    xTaskCreate( vSetTimeTask, "vSetTimeTask", 500, NULL, 4, &xSetTimeTask );
    /* Start the scheduler so the created tasks start executing. */
    vTaskStartScheduler();
  }
  else
  {
    /*The queue could not be created*/
  }
  for( ;; );
}
```

Hình 3.6 Hàm main

Chương trình ngắt dùng để nhận yêu cầu từ máy tính thông qua UART.

```
void USART1_IRQHandler(void)
{
  BaseType_t xHigherPriorityTaskWoken = pdFALSE;
  if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
  {
    /* Read one byte from the receive data register */
    RxBuffer[RxCounter] = USART_ReceiveData(USART1);
    RxCounter++;
    xTaskNotifyFromISR(xSetTimeTask,0,eNoAction,&xHigherPriorityTaskWoken);
    portYIELD_FROM_ISR( xHigherPriorityTaskWoken );
  }
}
```

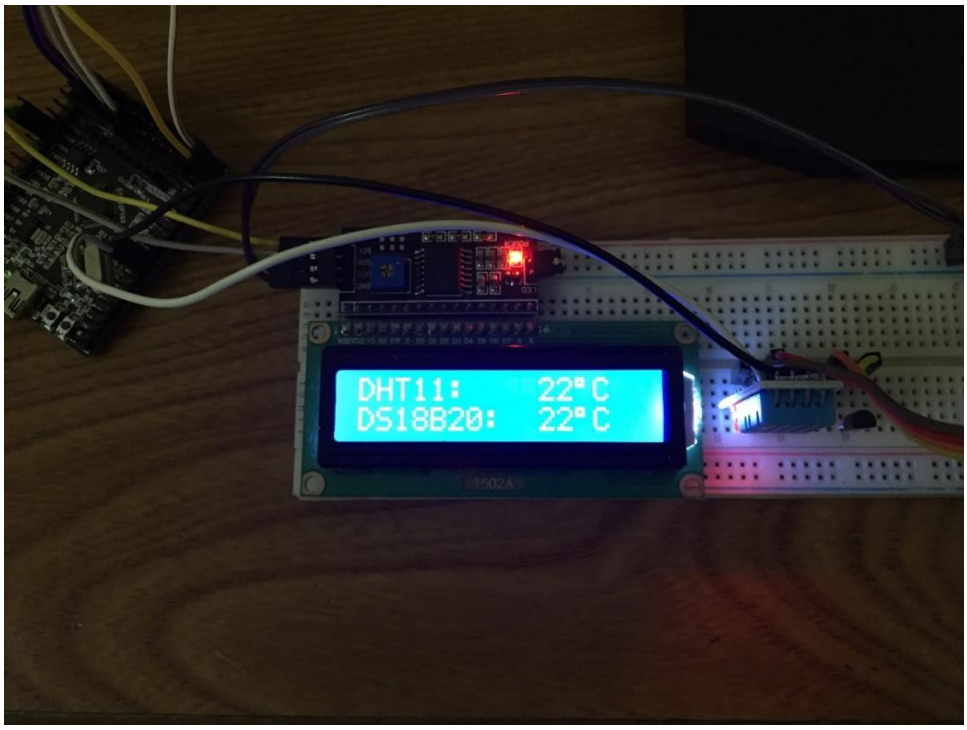
Hình 3.7 Ngắt nhận UART

Chi tiết chương trình thực hiện các Task khác được trình bày trong thư mục lập trình.

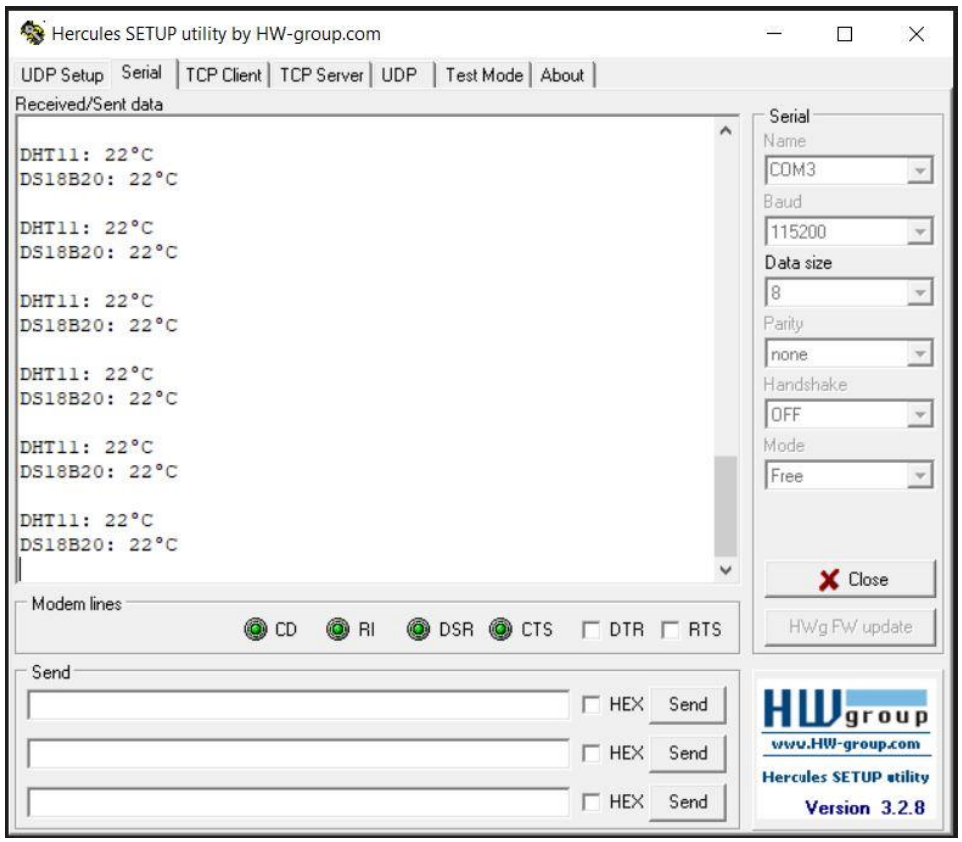
CHƯƠNG 4. KẾT LUẬN

4.1 Kết quả đạt được

4.1.1 Xây dựng chương trình theo bộ lập lịch

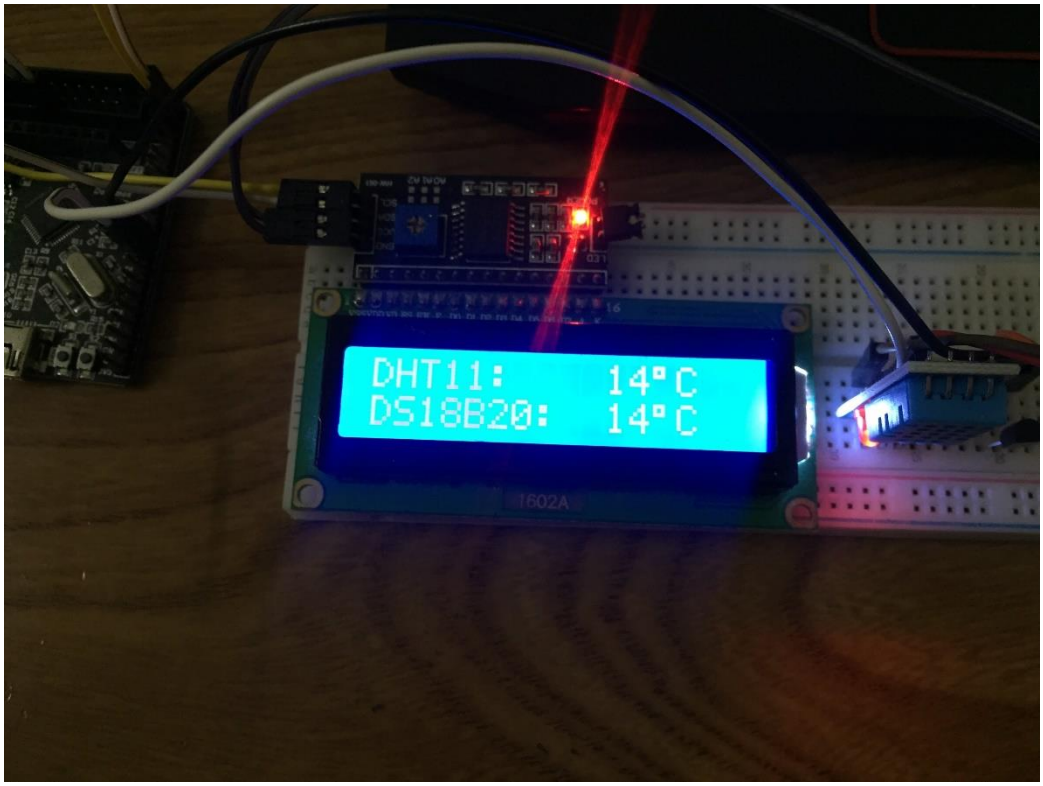


Hình 4.1 Hiển thị LCD (1:08 AM 18/2/2022)

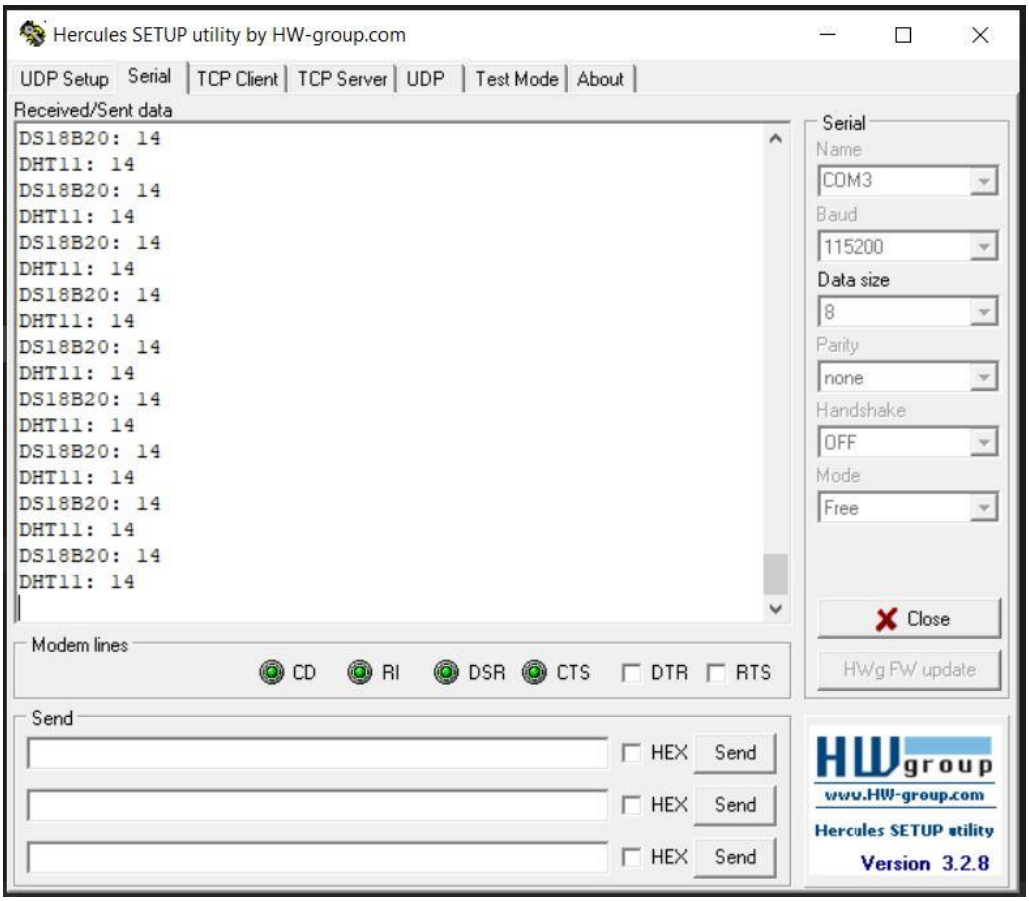


Hình 4.2 Truyền thông số lên máy tính (1:08 AM 18/2/2022)

4.1.2 Xây dựng trương trình theo hệ điều hành FreeRTOS

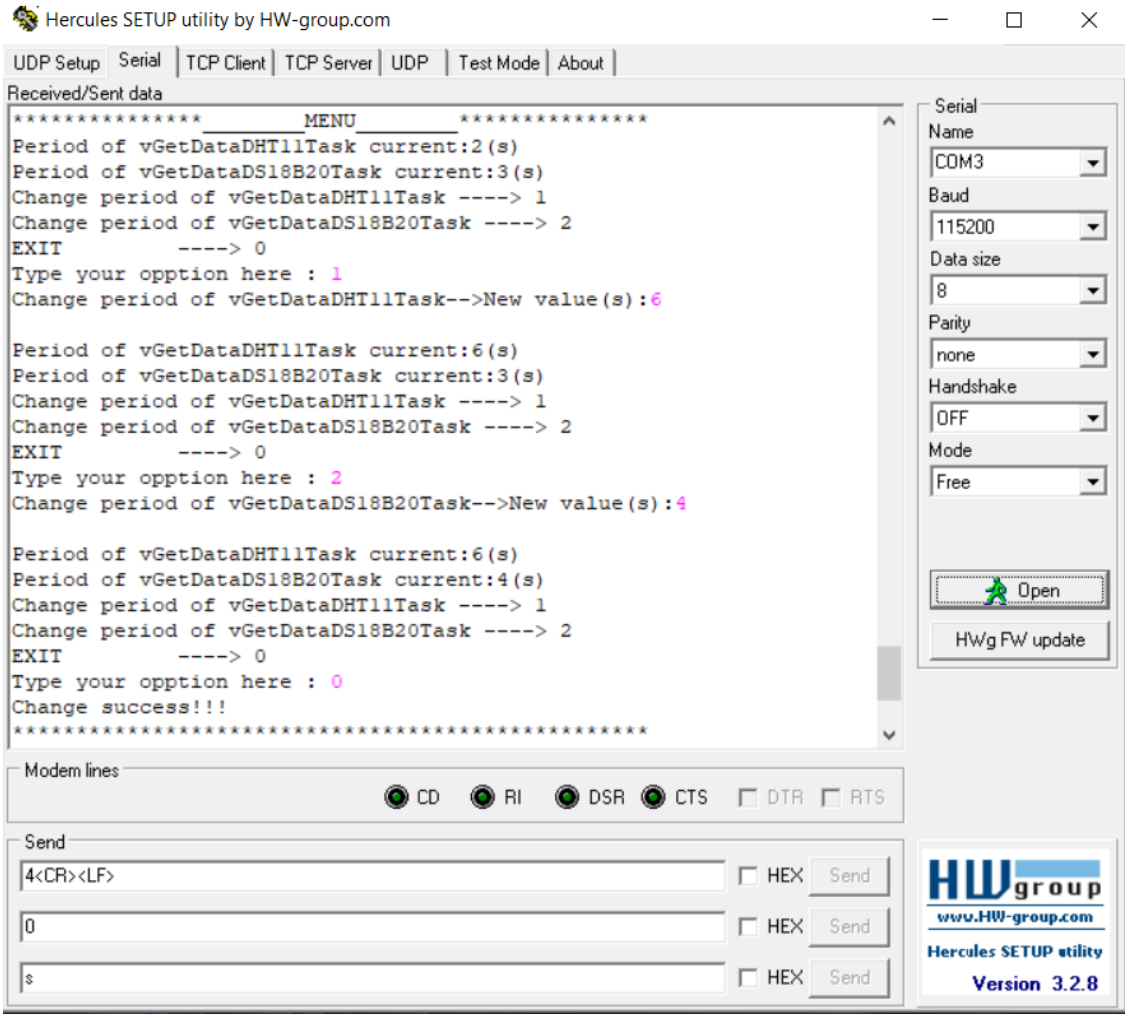


Hình 4.3 Hiển thị LCD (16:50 21/02/2020)



Hình 4.4 Truyền thông số lên máy tính (16:50 21/02/2020)

4.1.3 Cài đặt bộ lập lịch từ máy tính



Hình 4.5 Cài đặt bộ lập lịch từ máy tính

4.2 Kết luận và hướng phát triển của đồ án trong tương lai

Hệ thống đã đo được nhiệt độ tương đối chính xác.

Việc hoàn thành bài tập lớn đã giúp em nắm được cách thức cơ bản để có thể thiết kệ được bộ lập lịch và có thể sử dụng hệ điều hành FreeRTOS.

Tuy nhiên, còn một số nhược điểm cần khắc phục trong tương lai:

- Code size chưa được tối ưu.

4.3 Tự đánh giá kết quả

8/10

4.4 Link share thư mục lập trình

https://husteduvn-my.sharepoint.com/:f:/g/personal/dat_nt173727_sis_hust_edu_vn/ElMMghB6ArFNhlp1p7kPqEYBtN4irpBcBw0uauUH9oP4mQ?e=GhTOzy

TÀI LIỆU THAM KHẢO

[1] DS18B20 Datasheet, "ALLDATASHEET," [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html.

[2] DHT11 Humidity & Temperature Sensor Datasheet, "Mouser," [Online]. Available: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf.

[3] HD44780U Datasheet, "ALLDATASHEET," [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/63663/HITACHI/HD44780U.html.

[4] PCF8574 Datasheet, "ALLDATASHEET," [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/18212/PHILIPS/PCF8574.html.

[5] STM32F103C8 Datasheet, "ST," [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html#documentation.

[6] RM0008 Reference Manuals, "ST," [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html#.

[7] Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide, "FreeRTOS," [Online]. Available: https://www.freertos.org/Documentation/RTOS_book.html.

[8] FreeRTOS_Reference_Manual_V10.0.0, "FreeRTOS," [Online]. Available: https://www.freertos.org/Documentation/RTOS_book.html.

[9] Lê Công Cường, Programming Paradigms, Trường Đại học Bách Khoa Hà Nội.