ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

THIẾT KẾ TRUYỀN THÔNG THEO CHUẨN I2C

Họ và tên: Lý Bảo Khánh
MSV: 21020920

1. NỘI DUNG THÍ NGHIỆM

Thiết lập truyền thông theo chuẩn I2C

Linh kiện sử dụng:

- STM32F401RET6
- LCD16x2
- PCF8574T

2. NỘI DUNG LÝ THUYẾT

2.1. I2C là gì?

I2C (Inter-Integrated Circuit) là một giao thức giao tiếp được phát triển bởi Philips Semiconductors để truyền dữ liệu giữa một bộ xử lý trung tâm với nhiều IC trên cùng một board mạch chỉ sử dụng hai đường truyền tín hiệu

Đây là loại giao thức nối tiếp đồng bộ. Mỗi byte dữ liệu có độ dài 8 bit, bit MSB được truyền đầu tiên, số lượng byte trong 1 lần truyền không hạn chế. Theo sau mỗi byte là 1 bit ACK.

Khi không nhận được đúng địa chỉ hoặc khi muốn kết thúc giao tiếp, thiết bị nhận sẽ gửi xung NOT – ACK (SDA mức thấp) báo cho master biết. Khi đó master sẽ tạo xung stop để kết thúc hoặc tạo start để bắt đầu quá trình mới.

2.2. Những đặc điểm của giao thức giao tiếp I2C

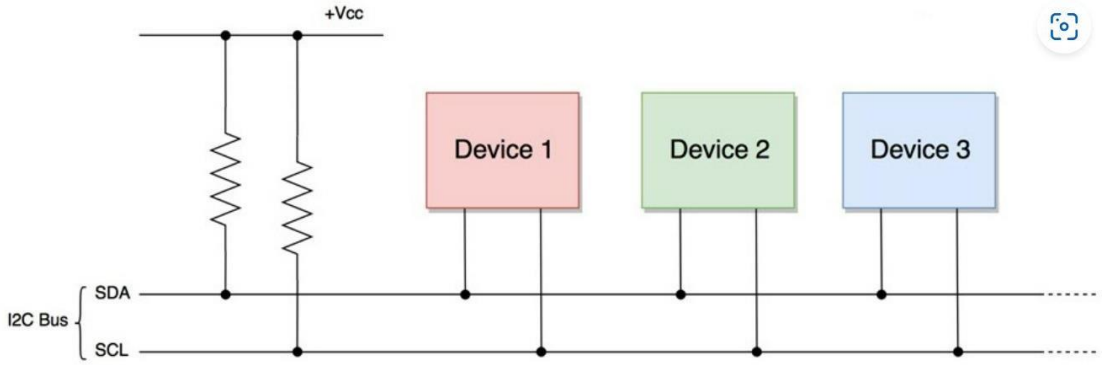
Dưới đây là những đặc điểm quan trọng của giao thức giao tiếp I2C:

- Chỉ cần có 2 đường bus (dây) chung để điều khiển bất kỳ thiết bị / IC nào trên mạng I2C
- Không cần thoả thuận trước về tốc độ truyền dữ liệu như trong giao tiếp UART. Vì vậy tốc độ truyền dữ liệu có thể được điều chỉnh bất cứ khi nào cần thiết
- Cơ chế đơn giản để xác thực dữ liệu được truyền
- Sử dụng hệ thống địa chỉ 7 bit để xác định một thiết bị / IC cụ thể trên bus I2C
- Các mạng I2C dễ dàng mở rộng. Các thiết bị mới có thể được kết nối đơn giản với hai đường bus chung I2C

2.3. Phần cứng

a. Bus vật lý I2C

Bus I2C (dây giao tiếp) chỉ gồm 2 dây và đặt tên là Serial Clock Line (SCL) và Serial Data Line (SDA). Dữ liệu được truyền đi được gửi qua dây SDA và được đồng bộ với tín hiệu đồng hồ (clock) từ SCL. Tất cả các thiết bị / IC trên mạng I2C được kết nối với cùng đường SCL và SDA như sau:



Cả hai đường bus I2C (SDA, SCL) đều hoạt động như các bộ lái cực máng hở (open drain). Nó có nghĩa là bất kỳ thiết bị / IC trên mạng I2C có thể lái SDA và SCL xuống mức thấp, nhưng không thể lái chúng lên mức cao. Vì vậy, một điện trở kéo lên (khoảng 1 kΩ đến 4,7 kΩ) được

sử dụng cho mỗi đường bus, để giữ cho chúng ở mức cao (ở điện áp dương) theo mặc định.

Việc sử dụng một hệ thống cực máng hở (open drain) là để không xảy ra hiện tượng ngắn mạch, điều này có thể xảy ra khi một thiết bị cố gắng kéo đường dây lên cao và một số thiết bị khác cố gắng kéo đường dây xuống thấp.

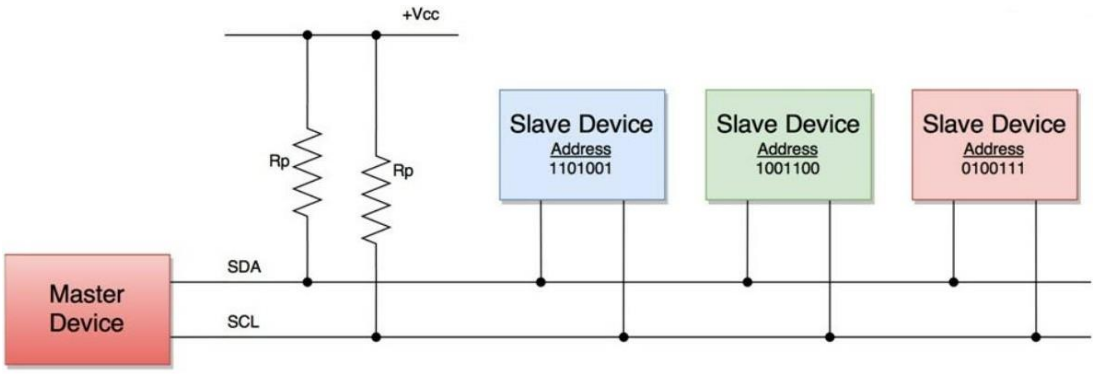
b. Thiết bị Master và Slave

Các thiết bị kết nối với bus I2C được phân loại hoặc là thiết bị Master hoặc là thiết bị Slave. Ở bất cứ thời điểm nào thì chỉ có duy nhất một thiết bị Master ở trang thái hoạt động trên bus I2C. Nó điều khiển đường tín hiệu đồng hồ SCL và quyết định hoạt động nào sẽ được thực hiện trên đường dữ liệu SDA.

Tất cả các thiết bị đáp ứng các hướng dẫn từ thiết bị Master này đều là Slave. Để phân biệt giữa nhiều thiết bị Slave được kết nối với cùng một bus I2C, mỗi thiết bị Slave được gán một địa chỉ vật lý 7-bit cố định.

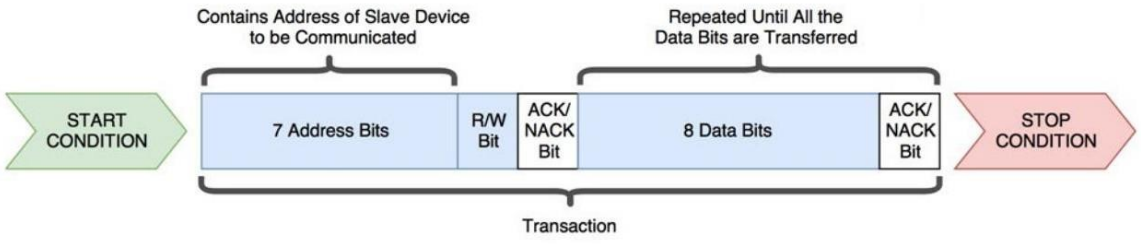
Khi một thiết bị Master muốn truyền dữ liệu đến hoặc nhận dữ liệu từ một thiết bị Slave, nó xác định địa chỉ thiết bị Slave cụ thể này trên đường SDA và sau đó tiến hành truyền dữ liệu. Vì vậy, giao tiếp có hiệu quả diễn ra giữa thiết bị Master và một thiết bị Slave cụ thể.

Tất cả các thiết bị Slave khác không phản hồi trừ khi địa chỉ của chúng được chỉ định bởi thiết bị Master trên dòng SDA.



c. Giao thức truyền dữ liệu

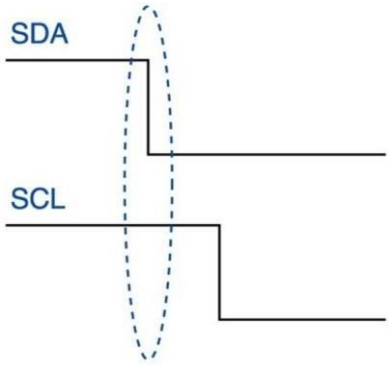
Dữ liệu được truyền giữa thiết bị Master và các thiết bị Slave thông qua một đường dữ liệu SDA duy nhất, thông qua các chuỗi có cấu trúc gồm các số 0 và 1 (bit). Mỗi chuỗi số 0 và 1 được gọi là giao dịch (transaction) và dữ liệu trong mỗi giao dịch có cấu trúc như sau:



- **Điều kiện bắt đầu (Start Condition)**

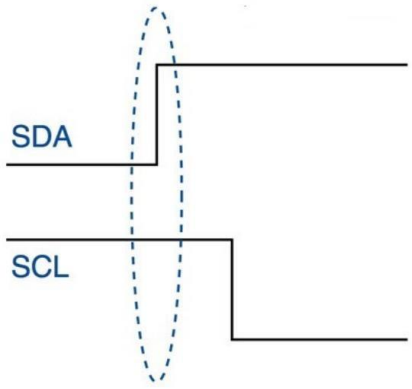
  Bất cứ khi nào một thiết bị chủ / IC quyết định bắt đầu một giao dịch, nó sẽ chuyển mạch SDA từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ cao xuống thấp.

  Khi điều kiện bắt đầu được gửi bởi thiết bị Master, tất cả các thiết bị Slave đều hoạt động ngay cả khi chúng ở chế độ ngủ (sleep mode) và đợi bit địa chỉ.



Start Condition

- **Khối địa chỉ**
  Nó bao gồm 7 bit và được lấp đầy với địa chỉ của thiết bị Slave đến / từ đó thiết bị Master cần gửi / nhận dữ liệu. Tất cả các thiết bị Slave trên bus I2C so sánh các bit địa chỉ này với địa chỉ của chúng.
- **Bit Read/Write**
  Bit này xác định hướng truyền dữ liệu. Nếu thiết bị Master / IC cần gửi dữ liệu đến thiết bị Slave, bit này được thiết lập là '0'. Nếu IC Master cần nhận dữ liệu từ thiết bị Slave, bit này được thiết lập là '1'.
- **Bit ACK/ NACK**
  ACK / NACK là viết tắt của Acknowledged/Not-Acknowledged. Nếu địa chỉ vật lý của bất kỳ thiết bị Slave nào trùng với địa chỉ được thiết bị Master phát, giá trị của bit này được set là '0' bởi thiết bị Slave. Ngược lại, nó vẫn ở mức logic '1' (mặc định).
- **Khối dữ liệu**
  Nó bao gồm 8 bit và chúng được thiết lập bởi bên gửi, với các bit dữ liệu cần truyền tới bên nhận. Khối này được theo sau bởi một bit ACK / NACK và được set thành '0' bởi bên nhận nếu nó nhận thành công dữ liệu. Ngược lại, nó vẫn ở mức logic '1'.
  Sự kết hợp của khối dữ liệu theo sau bởi bit ACK / NACK được lặp lại cho đến quá trình truyền dữ liệu được hoàn tất.
- **Điều kiện kết thúc (Stop conditon)**
  Sau khi các khung dữ liệu cần thiết được truyền qua đường SDA, thiết bị Master chuyển đường SDA từ mức điện áp thấp sang mức điện áp cao trước khi đường SCL chuyển từ cao xuống thấp.



Stop Condition

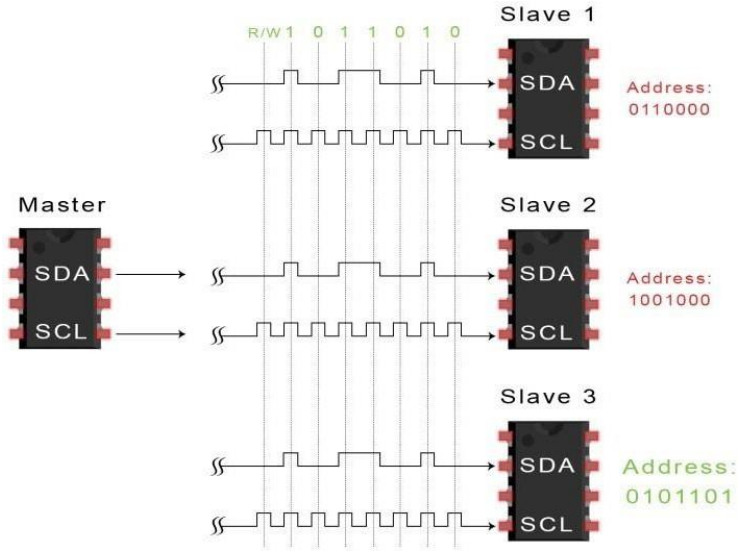
d. Nguyên lý hoạt động của giao thức giao tiếp I2C

Giao tiếp I2C được bắt đầu bởi thiết bị Master hoặc để gửi dữ liệu đến thiết bị Slave hoặc nhận dữ liệu từ thiết bị đó. Chúng ta hãy tìm hiểu về cách làm việc của cả hai kịch bản một cách chi tiết.

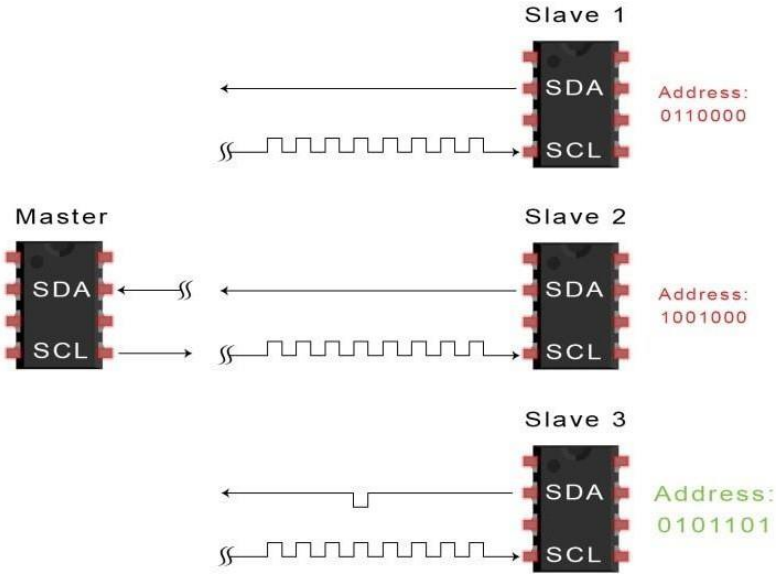
Gửi dữ liệu đến thiết bị Slave

Trình tự hoạt động sau đây diễn ra khi một thiết bị Master gửi dữ liệu đến một thiết bị Slave cụ thể thông qua bus I2C:

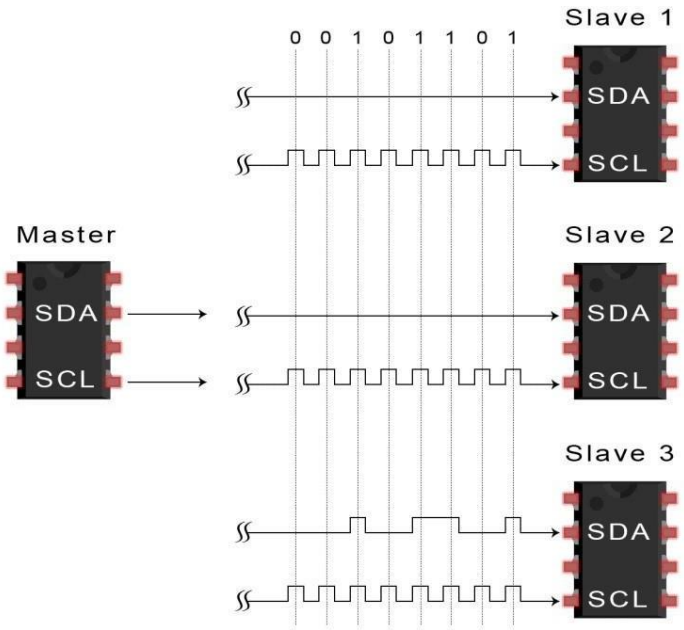
- Thiết bị Master gửi điều kiện bắt đầu đến tất cả các thiết bị Slave
- Thiết bị Master gửi 7 bit địa chỉ của thiết bị Slave mà thiết bị Master muốn giao tiếp cùng với bit Read/Write



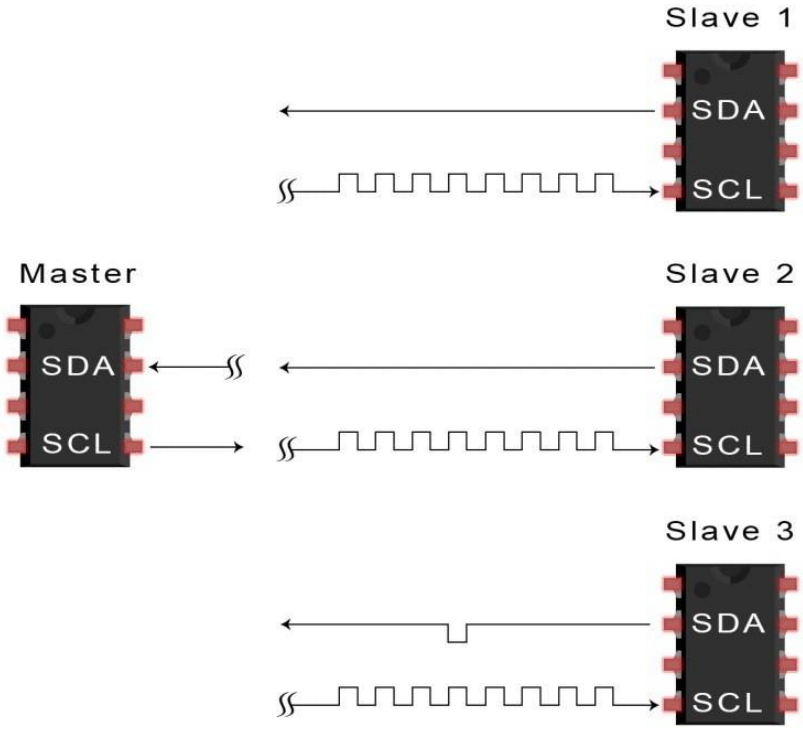
- Mỗi thiết bị Slave so sánh địa chỉ được gửi từ thiết bị Master đến địa chỉ riêng của nó. Nếu địa chỉ trùng khớp, thiết bị Slave gửi về một bit ACK bằng cách kéo đường SDA xuống thấp và bit ACK / NACK được thiết lập là ‘0’,. Nếu địa chỉ từ thiết bị Master không khớp với địa chỉ riêng của thiết bị Slave thì đường SDA ở mức cao và bit ACK / NACK sẽ ở mức ‘1’ (mặc định).



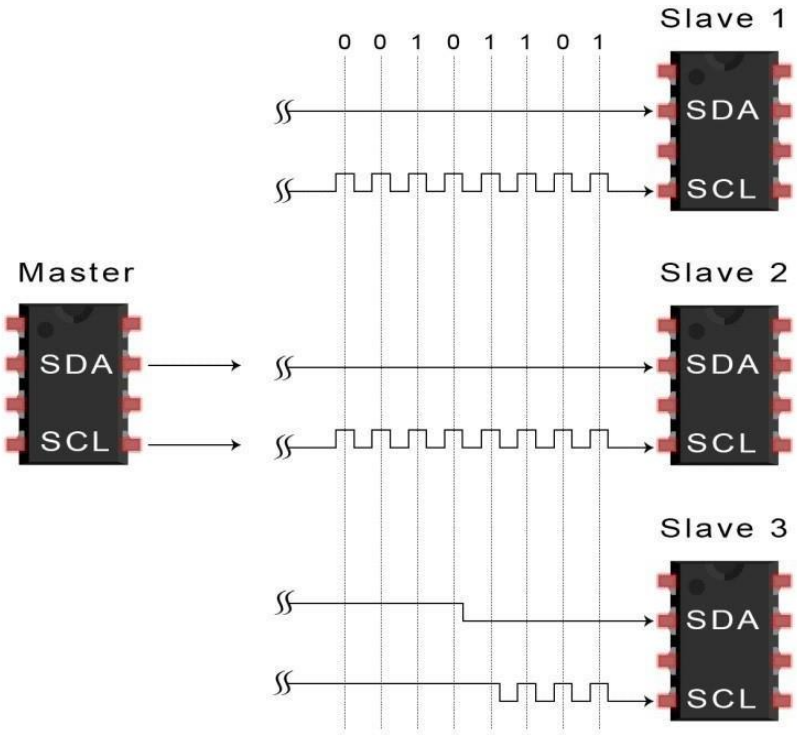
- Thiết bị Master gửi hoặc nhận khung dữ liệu. Nếu thiết bị Master muốn gửi dữ liệu đến thiết bị Slave, bit Read / Write là mức điện áp thấp. Nếu thiết bị Master đang nhận dữ liệu từ thiết bị Slave, bit này là mức điện áp cao.



- Nếu khung dữ liệu được thiết bị Slave nhận được thành công, nó sẽ thiết lập bit ACK / NACK thành ‘0’, báo hiệu cho thiết bị Master tiếp tục



- Sau khi tất cả dữ liệu được gửi đến thiết bị Slave, thiết bị Master gửi điều kiện dừng để báo hiệu cho tất cả các thiết bị Slave biết rằng việc truyền dữ liệu đã kết thúc.

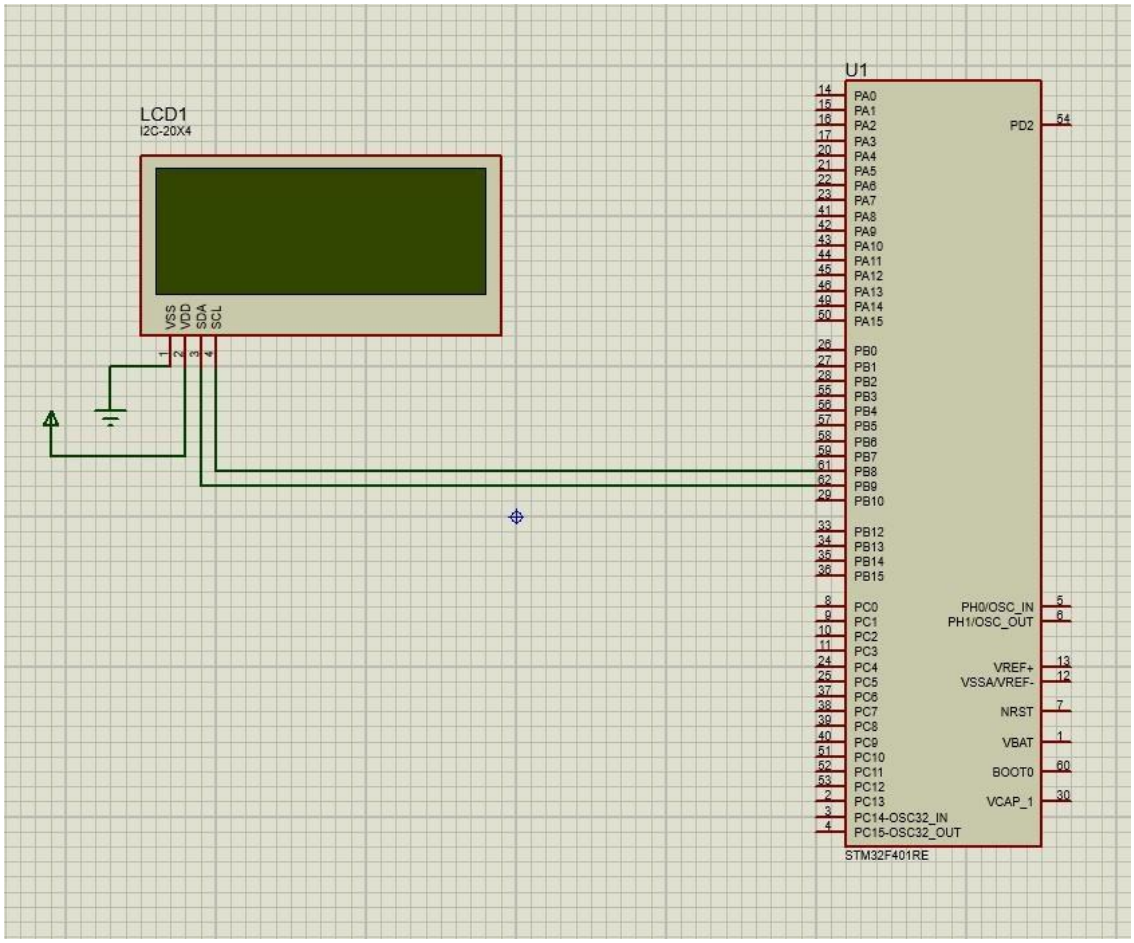

Slave 1
0 0 1 0 1 1 0 1
SDA
SCL
Master
SDA
SCL
Slave 2
SDA
SCL
Slave 3
SDA
SCL

3. NỘI DUNG THỰC NGHIỆM.

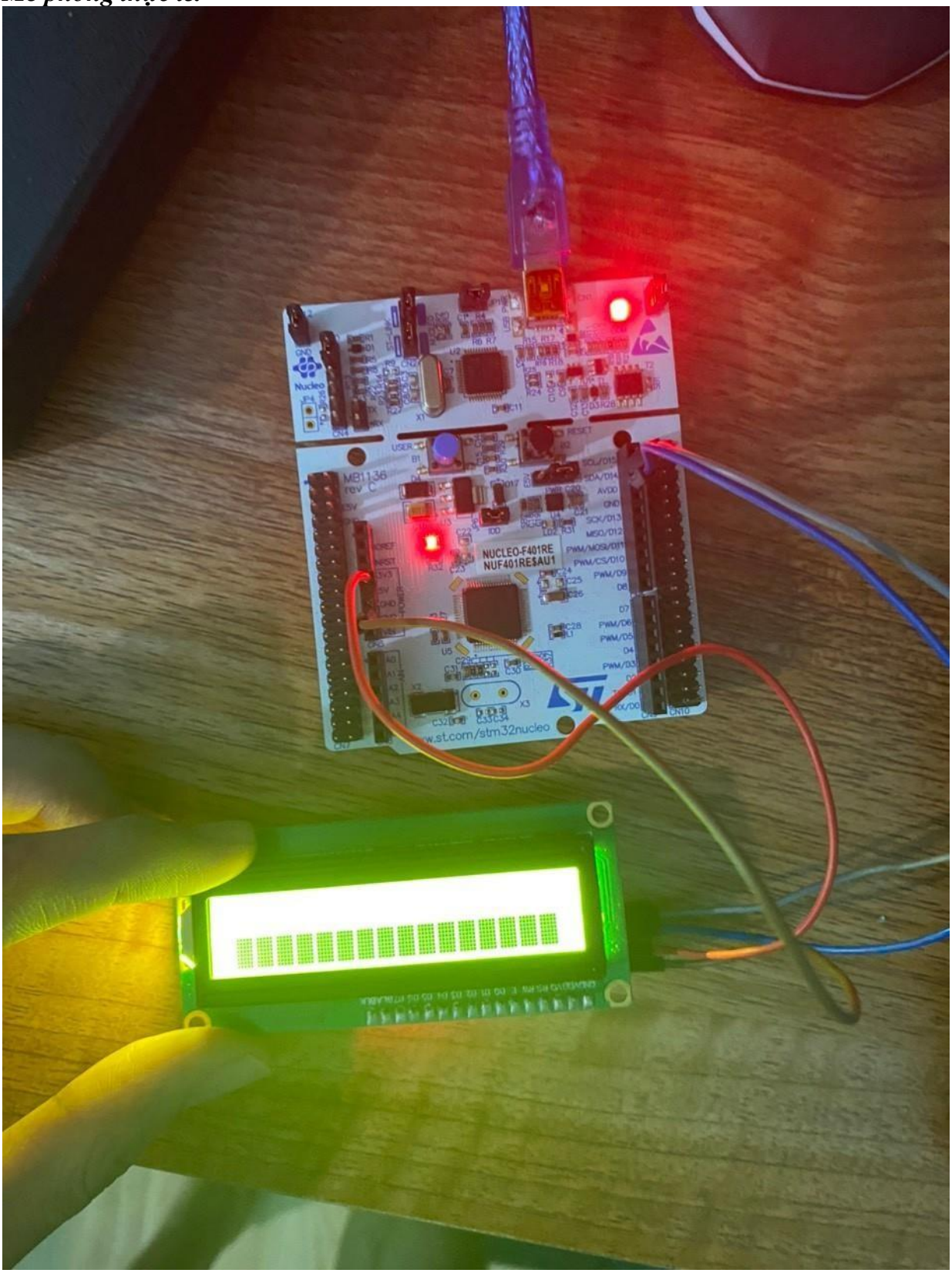
- Sử dụng truyền thông I2C giữa vi điều khiển và PCF8457T, rồi từ PCF8457T để điều khiển màn LCD16x2.
- Có thể sử dụng truyền thông I2C giữa 2 vi điều khiển, lưu ý thiết lập đúng địa chỉ Slave.

3.1. Mô phỏng trên proteus.

- **Mạch mô phỏng**



3.2. Mô phỏng thực tế.



Giải thích về cách hoạt động của mạch:

- Nối các chân PB8 (SCL) và PB9 (SDA) lần lượt với SCL và SDA trên PCF8547T, đồng thời cấp nguồn cho cả 2.
- Trên vi điều khiển, cấu hình các chân I2C để giao tiếp, ở đây dùng vi điều khiển như một Master để gửi dữ liệu đi.
- Lập trình cho vi điều khiển để gửi tín hiệu điều khiển qua PCF8547T đến LCD16x2 (ở đây gửi ký tự 'T' để hiển thị trên LCD).

3.3. Code.

```
#include "stm32f4xx.h"

//Địa chỉ slave của LCD cùng các biến quan trọng
#define LCD_ADDR (0x27 << 1)
#define LCD_EN 0x04
#define LCD_RS 0x01

//Cần delay cho khoảng gửi các lệnh tới LCD
void delay(uint32_t val)
{
		for (int i = 0; i < val; i++);
}

//Cấu hình các chân PB8 và PB9 của vi điều khiển để hoạt động như các chân I2C
void I2C1_Init(void)
{
   RCC->AHB1ENR |= RCC_AHB1ENR_GPIOBEN;
   RCC->APB1ENR |= RCC_APB1ENR_I2C1EN;

   GPIOB->MODER |= GPIO_MODER_MODER8_1 | GPIO_MODER_MODER9_1;
   GPIOB->OTYPER |= GPIO_OTYPER_OT_8 | GPIO_OTYPER_OT_9;
   GPIOB->OSPEEDR |= GPIO_OSPEEDER_OSPEEDR8 | GPIO_OSPEEDER_OSPEEDR9;
   GPIOB->PUPDR |= GPIO_PUPDR_PUPDR8_0 | GPIO_PUPDR_PUPDR9_0;
   GPIOB->AFR[1] |= (4 << (4*0)) | (4 << (4*1));

   I2C1->CR1 |= (1<<15);
   I2C1->CR1 &= ~(1<<15);

   I2C1->CR2 |= (45<<0);
   I2C1->CCR = 225<<0;
   I2C1->TRISE = 46;
   I2C1->CR1 |= I2C_CR1_PE;
}



//Gửi data sang Slave (lưu ý phải đúng địa chỉ Slave của I2C)
```

```
void I2C1_SendData(uint8_t address, uint8_t data)
{
                I2C1->CR1 |= I2C_CR1_ACK;
   I2C1->CR1 |= I2C_CR1_START;

   while(!(I2C1->SR1 & I2C_SR1_SB));
   I2C1->DR = address;
   while(!(I2C1->SR1 & I2C_SR1_ADDR));
   (void)I2C1->SR1;
   (void)I2C1->SR2;

                while (!(I2C1->SR1 & I2C_SR1_TXE));
   I2C1->DR = data;
   while(!(I2C1->SR1 & I2C_SR1_BTF));

   I2C1->CR1 |= I2C_CR1_STOP;
}
```

//Gửi lệnh tới PCF8547T để điều khiển LCD

```
void LCD_SendCmd(uint8_t cmd) {
   uint8_t data_u, data_l;
   uint8_t data_t[4];
   data_u = (cmd&0xf0);
   data_l = ((cmd<<4)&0xf0);
   data_t[0] = data_u|0x0C;
   data_t[1] = data_u|0x08;
   data_t[2] = data_l|0x0C;
   data_t[3] = data_l|0x08;
   for(int i=0; i<4; i++)
   {
      I2C1_SendData(LCD_ADDR, data_t[i]);
      delay(2);
   }
}
```

//Gửi lệnh tới PCF8547T để điều khiển LCD

```
void LCD_SendData(uint8_t data) {
   uint8_t data_u, data_l;
   uint8_t data_t[4];
   data_u = (data&0xf0);
   data_l = ((data<<4)&0xf0);
   data_t[0] = data_u|0x0D;
   data_t[1] = data_u|0x09;
   data_t[2] = data_l|0x0D;
   data_t[3] = data_l|0x09;
   for(int i=0; i<4; i++)
```

```
    {
        I2C1_SendData(LCD_ADDR, data_t[i]);
        delay(2);
    }
}
```

//Cấu hình LCD

```
void LCD_Init(void) {
    LCD_SendCmd(0x33);
                delay(100);
    LCD_SendCmd(0x32);
                delay(20);
    LCD_SendCmd(0x06);
                delay(20);
    LCD_SendCmd(0x0C);
                delay(20);
    LCD_SendCmd(0x28);
                delay(20);
    LCD_SendCmd(0x01);
    delay(20);
}
```

//Hiển thị ký tự lên màn LCD

```
void LCD_PutStr(char *str) {
    while (*str) {
        LCD_SendData((uint8_t)(*str));
        str++;
    }
}
```

//Thực hiện gửi ký tự ‘ T’

```
int main(void)
{
    I2C1_Init();
    LCD_Init();

    while(1)
    {
        LCD_PutStr("T");
        delay(1000);
        LCD_SendCmd(0x01);
        delay(1000);
    }
}
```