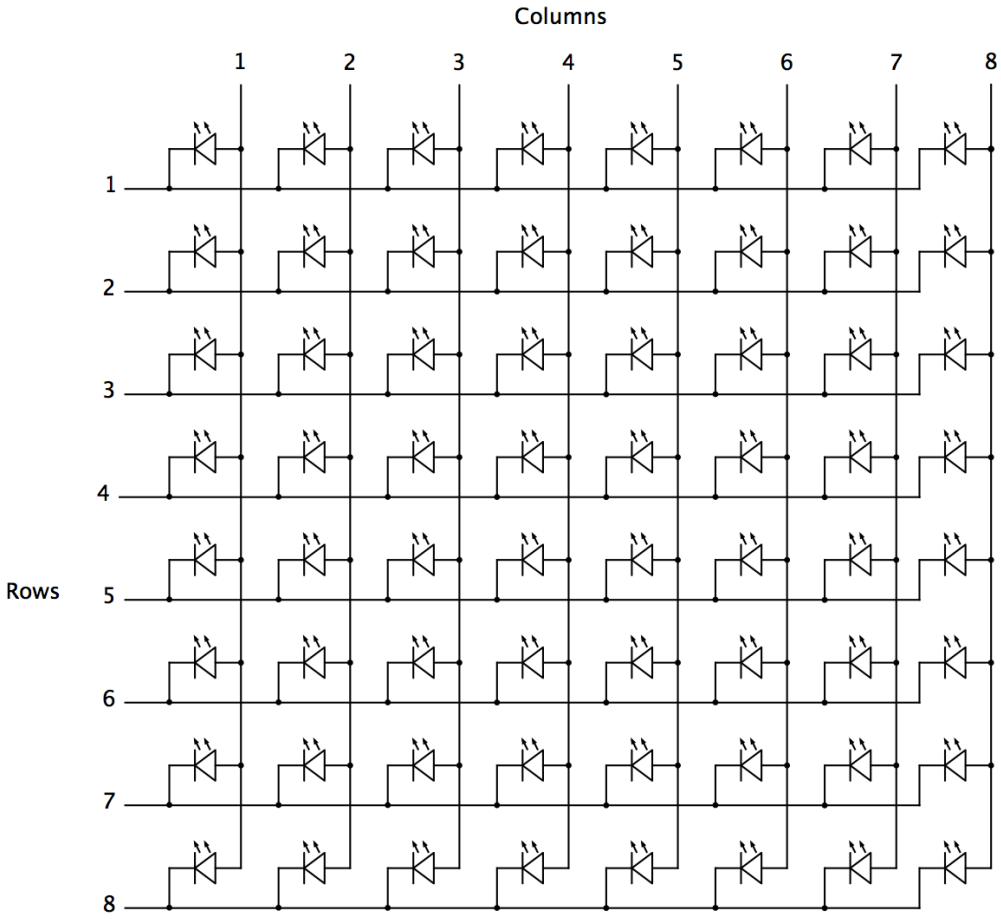
Bài 4: Lập rình LED Matrix với STM32F1

Phần 1: Hôm nay chúng ta tiếp tục tìm hiểu về bài toán LED Matrix. Về cơ bản quá trình điều khiển LED Matrix tuân theo nguyên tắc điều khiển LED 7 thanh nhưng dữ liệu hiển thị cho mỗi lần là một cột. Cần hiển thị nhiều lần để có thể hiển thị trên toàn bộ một màn hình LED Matrix

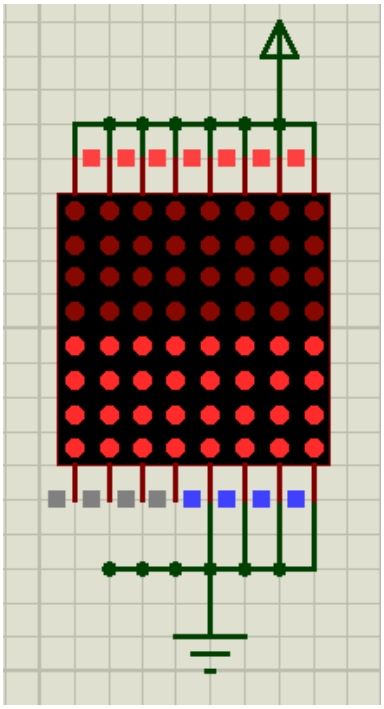
Cấu tạo LED Martrix có dạng như hình dưới đây. Về cấu tạo rất giống với một hệ nhiều LED 7 thanh, ví dụ gồm 8 LED 7 thanh ghép với nhau

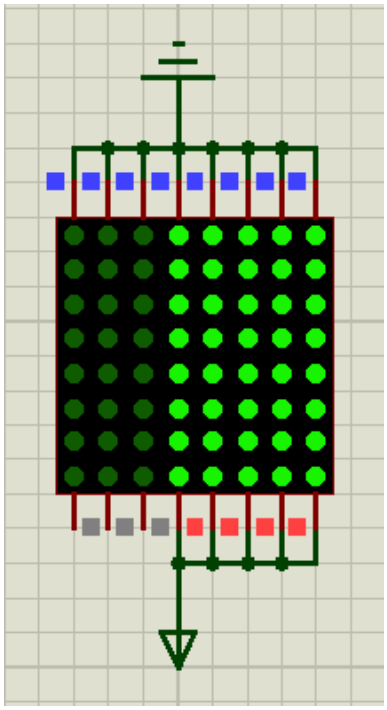


Để chuẩn bị cho bài toán về LED matrix các em chuẩn bị một mạch điện mô phỏng đơn giản theo hình sau. Đánh vào từ khóa “Matrix”, kéo xuống chọn loại 8x8 (màu xanh, đỏ tùy các em).

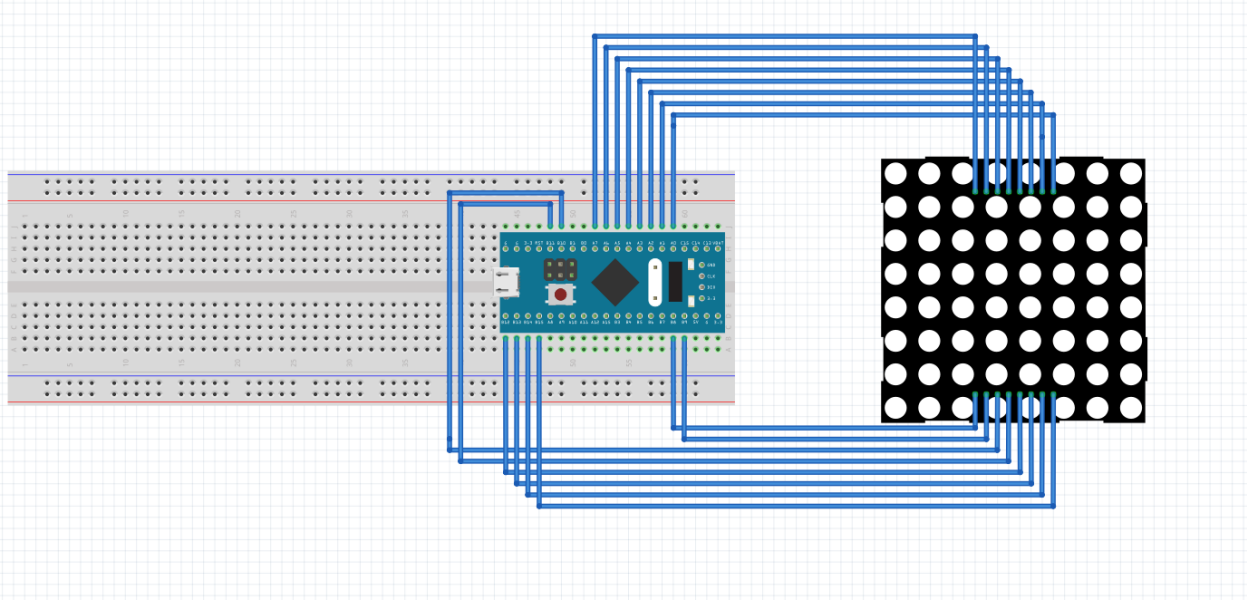
Ở đây PORT2 dùng để điều khiển 8 hàng LED, PORT3 dùng để điều khiển 8 cột LED. LED sẽ sáng nếu thỏa mãn 2 điều kiện là Port3 xuất ra 1 và Port2 xuất ra 0. Port 2 cấp giá trị theo chân, PORT3 được dùng để xuất dữ liệu ra LED. Để đèn sáng thì chân tương ứng ở PORT3 phải có giá trị bằng 0. Chi tiết xem hình dưới.

Hãy chọn Led và thử nghiệm theo hình dưới. **Chú ý, led đỏ hình dưới đã được quay 180 độ so với mặc định để cho giống với hình vẽ đầu tiên!** Nếu để mặc định, Led (xanh) sẽ sáng theo hình dưới (hàng cột đổi chỗ), và khi đó đoạn code sau sẽ không chạy theo ý muốn. Đoạn code và ma trận được xây dựng cho Led đã được quay 180 độ.





Ở bài toán này chúng ta tập trung tìm cách hiển thị một hình ảnh chúng ta mong muốn lên LED matrix 8×8 này. Việc chúng ta cần là xây dựng bảng dữ liệu cho hình ảnh mà chúng ta muốn hiển thị và lưu lại như lưu lại bảng dữ liệu của LED 7 thanh ở bài trước. Ở đây chúng ta sẽ lập bảng dữ liệu cho chữ cái H. Sau khi có bảng dữ liệu chúng ta thực hiện lập trình xuất mã tương tự LED 7 thanh. Ý tưởng là sử dụng thuật toán quét Led để mỗi vòng lặp, hiển thị led sáng trên một cột.



Chúng ta có chương trình như sau:

//hien thi chu H len LED matrix

```
/*

*/
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#define RCC_GPIO_ROW              RCC_APB2Periph_GPIOA
```

```
#define RCC_GPIO_COL            RCC_APB2Periph_GPIOB
#define GPIO_ROW                GPIOA
#define GPIO_COL                GPIOB
#define GPIO_PIN_ROW_1  GPIO_Pin_0 // khai bao hang 1
#define GPIO_PIN_ROW_2  GPIO_Pin_1 // khai bao hang 2
#define GPIO_PIN_ROW_3  GPIO_Pin_2 // khai bao hang 3
#define GPIO_PIN_ROW_4  GPIO_Pin_3 // khai bao hang 4
#define GPIO_PIN_ROW_5  GPIO_Pin_4 // khai bao hang 5
#define GPIO_PIN_ROW_6  GPIO_Pin_5 // khai bao hang 6
#define GPIO_PIN_ROW_7  GPIO_Pin_6 // khai bao hang 7
#define GPIO_PIN_ROW_8  GPIO_Pin_7 // khai bao hang 8
#define GPIO_PIN_COL_1   GPIO_Pin_8 // khai bao cot 1
#define GPIO_PIN_COL_2   GPIO_Pin_9 // khai bao cot 2
#define GPIO_PIN_COL_3   GPIO_Pin_10 // khai bao cot 3
#define GPIO_PIN_COL_4   GPIO_Pin_11 // khai bao cot 4
#define GPIO_PIN_COL_5   GPIO_Pin_12 // khai bao cot 5
#define GPIO_PIN_COL_6   GPIO_Pin_13 // khai bao cot 6
#define GPIO_PIN_COL_7   GPIO_Pin_14 // khai bao cot 7
#define GPIO_PIN_COL_8   GPIO_Pin_15 // khai bao cot 8



 Unsigned int chuH[]={0x3C00,0x3C00,0x3C00,0x0000,0x0000,0x3C00,0x3C00,0x3C00};
//unsigned int chuE[]={0x8000, 0x8000,0xFC00,0x8000,0x8000,0xFC00,0x8000 ,0x8000};


void GPIO_Config()
{
        GPIO_InitTypeDef GPIO;
```

```
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); //cap clock cho Port A
	GPIO.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO.GPIO_Pin  = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7; //cau hinh cac chan I/O su dung cho ROW
	GPIO.GPIO_Speed = GPIO_Speed_2MHz;

	GPIO_Init(GPIOA, &GPIO);

	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); //cap clock cho Port B
	GPIO.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO.GPIO_Pin  = GPIO_Pin_8 |GPIO_Pin_9|GPIO_Pin_10 |GPIO_Pin_11 |GPIO_Pin_12 |GPIO_Pin_13 |GPIO_Pin_14 |GPIO_Pin_15;//cau hinh cac chan I/O su dung cho COL
	GPIO.GPIO_Speed = GPIO_Speed_2MHz;

	GPIO_Init(GPIOB, &GPIO);
}
void Delay_ms(uint16_t time)

{
	// tang bien dem len 12000 lan
	uint32_t time_n=time*12000;
	// cho den khi biem time_n giam =0 thi thoat
	while(time_n!=0){time_n--;}


}
int main()
```

```
{
        unsigned char i;
        GPIO_Config();//goi chuong trinh con da khai bao
        while(1)
        {
                for(i=0;i<8;i++)
                {
                        GPIOA->ODR=0x01<<i;
                        GPIOB->ODR=chuH[i];
    Delay_ms(1);
          }
  }
}
```

Còn lại chương trình tuân theo các bước của quy trình "*Quét LED*" đã nhắc ở bài thực hành số 3 trước.

Sau khi hoàn thiện chương trình các bạn thực hiện Build, các bạn đưa file hex vào mạch điện chúng ta đã có và chạy mô phỏng. Các bạn sẽ thấy có kết quả như sau:

Trên đây chúng ta đã thực hiện điều khiển được 01 LED martrix. Về phần điều khiển nhiều LED matrix cũng tương tự như thế nhưng với điều kiện khi số LED martrix tăng lên yêu cầu số chân tăng lên rất nhanh. Chúng ta có thể sử dụng một số IC mở rộng như 74HC595 để thực hiện mở rộng để có thể điều khiển được nhiều LED matrix hơn.

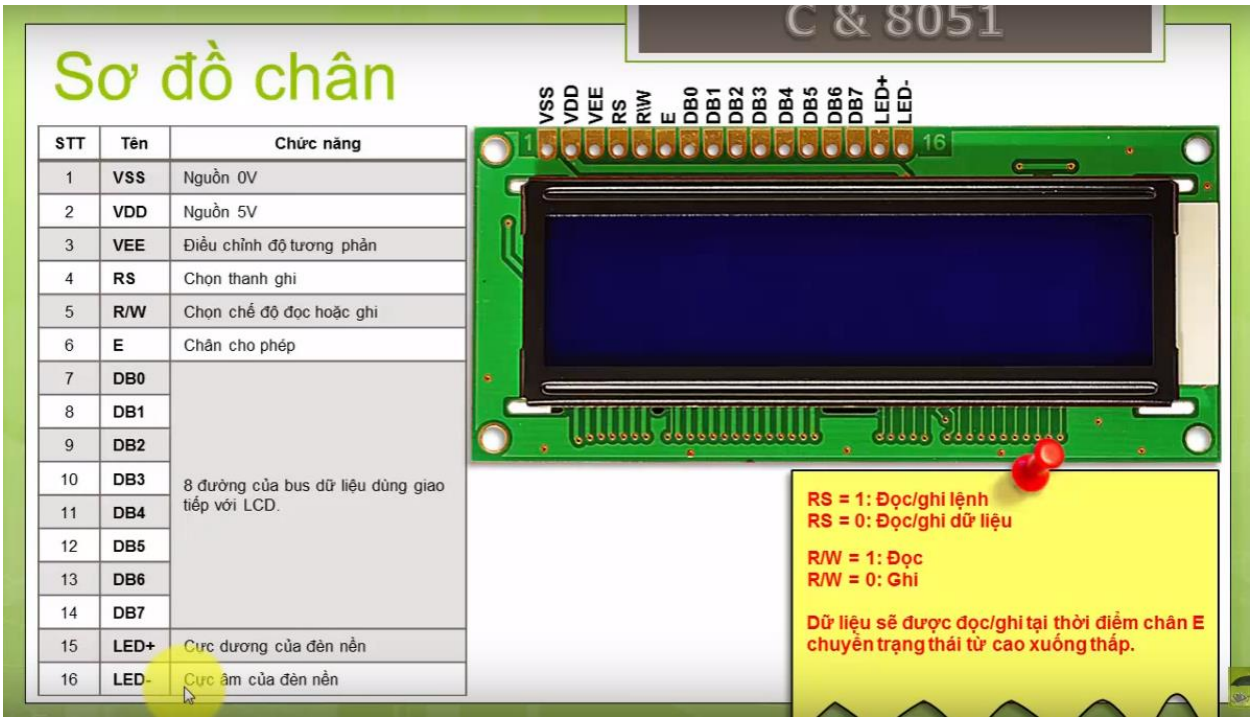
Bài tập1:

Hiển thị lần lượt các chữ cái trong tên của bạn! Khi trình bày theo nhóm thì hiển thị lần lượt các kí tự "NHOM MOT, NHOM HAI…).

Good luck and Have fun! (Không fun => cố chịu ^^)

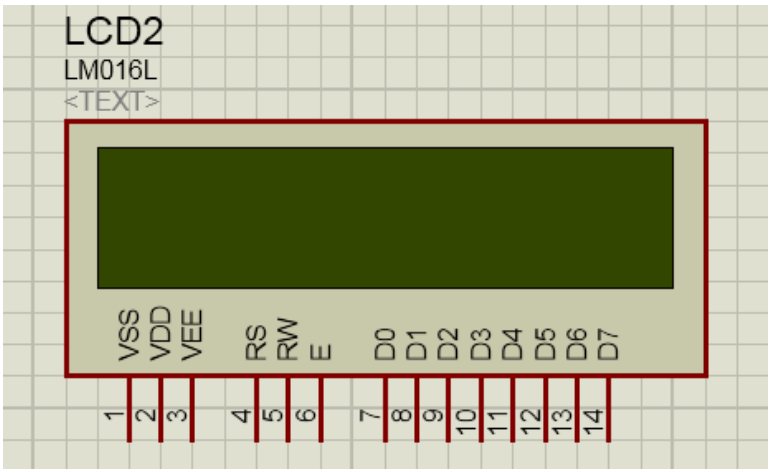
Phần 2: Chúng ta đến với một phần rất quan trọng: Hiển thị thông tin ra LCD.

LCD là một màn hình nhiều điểm ảnh, có thể coi là một Led ma trận dạng lớn, tuy nhiên chúng ta không cần đi vào điều khiển từng Pixel hay là từng chấm nhỏ như trong Phần 1 mà chúng ta sẽ điều khiển qua lệnh, con trỏ… để hiển thị thông tin một cách dễ dàng hơn. Có nhiều loại LCD, trong bài này chúng ta dùng loại đơn giản 16x2. Trước tiên chúng ta tìm hiểu về cấu tạo của nó.

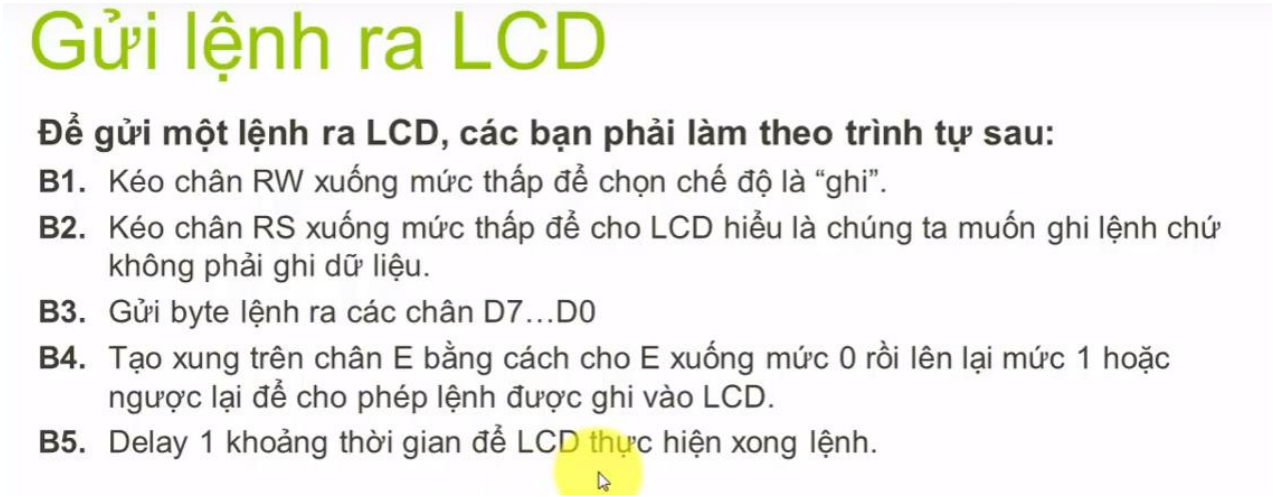


STT	Tên	Chức năng
1	VSS	Nguồn 0V
2	VDD	Nguồn 5V
3	VEE	Điều chỉnh độ tương phản
4	RS	Chọn thanh ghi
5	R/W	Chọn chế độ đọc hoặc ghi
6	E	Chân cho phép
7	DB0	8 đường của bus dữ liệu dùng giao tiếp với LCD.
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED+	Cực dương của đèn nền
16	LED-	Cực âm của đèn nền

Dưới đây là hình mô phỏng, các em tìm nó với từ khóa “lcd 16x2”.



Tiếp theo chúng ta học về cách kết nối, lập trình để sử dụng LCD. Trước tiên chúng ta phải gửi lệnh vào cho LCD biết nó phải làm gì. Cách gửi lệnh như sau:



Các em có thể xem một số lệnh sau đây. Những lệnh này gọi là lệnh khởi tạo LCD.

Tập lệnh của LCD

Chú ý: Ở đây Dâng chỉ liệt kê các lệnh thường dùng, chi tiết hơn về tập lệnh các bạn xem thêm trong tài liệu tham khảo.

Mã lệnh	Chức năng	T_{exe}
0x01	Xoá toàn bộ nội dung đang hiển thị trên màn hình.	1.52ms
0x02	Di chuyển con trỏ về vị trí đầu màn hình.	1.52ms
0x06	Tự động di chuyển con trỏ đến vị trí tiếp theo mỗi khi xuất ra LCD 1 ký tự.	37us
0x0C	Bật hiển thị và tắt con trỏ	37us
0x0E	Bật hiển thị và bật con trỏ	37us
0x80	Di chuyển con trỏ về đầu dòng 1	37us
0xC0	Di chuyển con trỏ về đầu dòng 2	37us
0x38	Giao tiếp 8 bit, hiển thị 2 dòng, kích thước font 5x7	37us
0x28	Giao tiếp 4 bit, hiển thị 2 dòng, kích thước font 5x7	37us

Sau khi gửi các lệnh vào LCD (qua các chân D0 – D7) để nó biết sẽ phải làm gì (bật màn hình, hiển thị hay tắt con trỏ, di chuyển con trỏ …), chúng ta sẽ phải gửi lệnh chứa các kí tự mà chúng ta muốn hiển thị. Cách gửi dữ liệu ra cũng tương tự như cách gửi lệnh, các em có thể xem trong hình sau:

Gửi dữ liệu ra LCD

Để gửi ký tự ra LCD, các bạn phải làm theo trình tự sau:

B1. Kéo chân RW xuống mức thấp để chọn chế độ là "ghi".

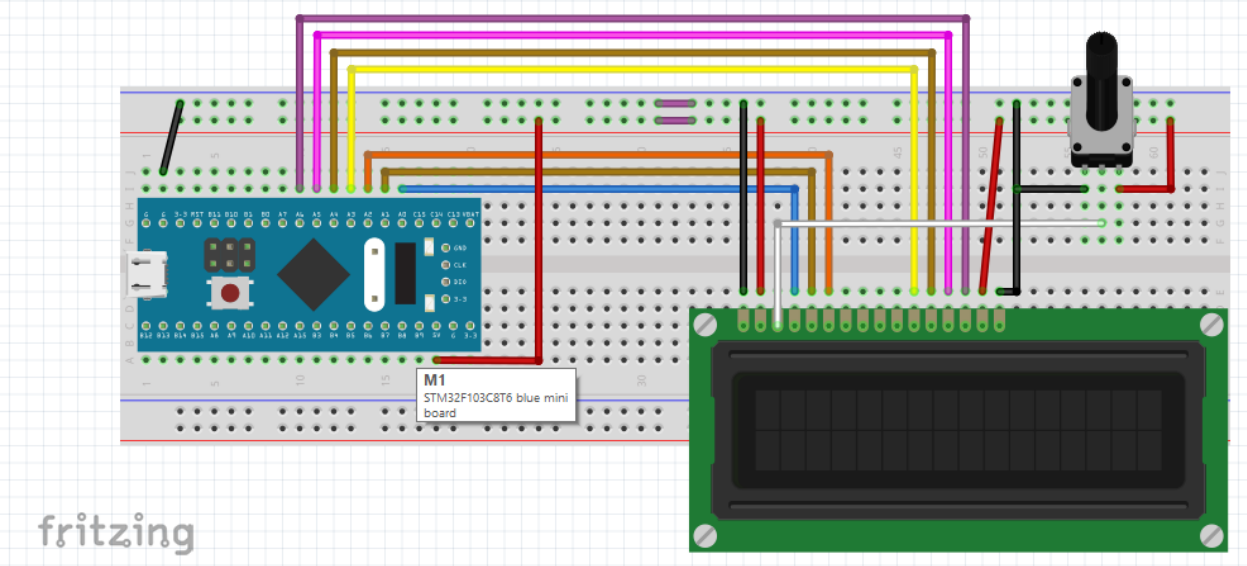
B2. Kéo chân RS lên mức cao để cho LCD hiểu là chúng ta muốn ghi dữ liệu chứ không phải ghi lệnh.

B3. Gửi mã của ký tự (ASCII) ra các chân D7…D0.

B4. Tạo xung trên chân E bằng cách cho E xuống mức 0 rồi lên lại mức 1 hoặc ngược lại để cho dữ liệu được ghi vào LCD.

B5. Delay 1 khoảng thời gian (37us) để LCD hiển thị ký tự

Bước 1 thường là mặc định. Các bước sau tương tự như việc gửi lệnh vào LCD. Tại sao B1 lại là mặc định, vì chúng ta thường dùng LCD để hiển thị chứ không để nhận dữ liệu vào. Hình sau nối chân RW với Ground:



Cách nối chân:

LCD_RS GPIO_Pin_0

LCD_RW GPIO_Pin_1

LCD_EN GPIO_Pin_2

LCD_D4 GPIO_Pin_3

LCD_D5 GPIO_Pin_4

LCD_D6 GPIO_Pin_5

LCD_D7 GPIO_Pin_6

***Lưu ý: Các em add thư viện cần thiết vào project như delay.h, lcd.h**

File main.c:

```
#include "stm32f10x.h"
#include "LCD.h"
```

```
#include "delay.h"

void delay(uint32_t time)
{
	while(time--);
}

int main(void)
{
	SysTick_Init();
	GPIO_LCD_Config();
	LCD_Init();
	delay_ms(200);

	while(1)
	{
		LCD_Gotoxy(0,0);
		LCD_Puts(“KY THUAT VI XU LY-PTIT");
		delay_ms(10);
	}
}
```

File delay.c

```
#include "delay.h"
#define SYSCLK_MHZ 72
```

```
void SysTick_Init()
{
  //SysTick->VAL  = 0;                                  // Load the SysTick Counter
Value
  SysTick->LOAD = SysTick_LOAD_RELOAD_Msk;
  SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
                  SysTick_CTRL_TICKINT_Msk   |
                  SysTick_CTRL_ENABLE_Msk;                 // Enable SysTick IRQ
and SysTick Timer
}
uint32_t volatile SysTickCounter = 1;

void SysTick_Handler()
{
  ++SysTickCounter;
}



uint64_t SysTick64()
{

  return (((uint64_t)SysTickCounter) << 24) - SysTick->VAL;
}

uint32_t SysTick32()
{
  return (SysTickCounter << 24) - SysTick->VAL;
```

```
}

uint32_t SysTick24()
{
  return ~(SysTick->VAL);
}

uint64_t SysTick_Millis()
{
  return SysTick64() / (SYSCLK_MHZ *1000);
}

uint64_t SysTick_Micros()
{
  return SysTick64() / SYSCLK_MHZ;
}

void delay_us(unsigned long us)
{
  uint32_t finish = SysTick32() + (us * SYSCLK_MHZ);
  while (((int32_t)(finish - SysTick32())) > 0);
}

void delay_ms(unsigned long ms)
{
  for (; ms; ms--)  // while (ms--)
  {
```

```
    delay_us(1000);
  }
}
```

File delay.h

```
#ifndef __DELAY_H
#define __DELAY_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f10x.h"


void SysTick_Init();
void SysTick_Handler();
uint64_t SysTick64();
uint32_t SysTick32();
uint32_t SysTick24();
uint64_t SysTick_Millis();
uint64_t SysTick_Micros();
void delay_us(unsigned long us);
void delay_ms(unsigned long ms);
```

```
#ifdef __cplusplus
}
#endif

#endif /* __MISC_H */
```

File LCD.c

```
#include "LCD.h"
#define LCD_RS      GPIO_Pin_0
#define LCD_RW      GPIO_Pin_1
#define LCD_EN      GPIO_Pin_2
#define LCD_D4      GPIO_Pin_3
#define LCD_D5      GPIO_Pin_4
#define LCD_D6      GPIO_Pin_5
#define LCD_D7      GPIO_Pin_6


void GPIO_LCD_Config(void)
{
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  GPIO_InitTypeDef        GPIO_LCD_InitStruction;
  GPIO_LCD_InitStruction.GPIO_Mode    = GPIO_Mode_Out_PP;
  GPIO_LCD_InitStruction.GPIO_Pin     = LCD_D4|LCD_D5|LCD_D6|LCD_D7|
                                LCD_EN|LCD_RS|LCD_RW;
  GPIO_LCD_InitStruction.GPIO_Speed   = GPIO_Speed_10MHz;
```

```
  GPIO_Init(GPIOA, &GPIO_LCD_InitStruction);
}
void LCD_Enable(void)
{
  GPIO_SetBits(GPIOA, LCD_EN);
  delay_us(3);
  GPIO_ResetBits(GPIOA, LCD_EN);
  delay_us(50);
}

void LCD_Send4Bit(unsigned char Data)
{
  GPIO_WriteBit(GPIOA, LCD_D4, Data & 0x01);
  GPIO_WriteBit(GPIOA, LCD_D5, (Data>>1)&1);
  GPIO_WriteBit(GPIOA, LCD_D6, (Data>>2)&1);
  GPIO_WriteBit(GPIOA, LCD_D7, (Data>>3)&1);
}

void LCD_SendCommand(unsigned char command)
{
  LCD_Send4Bit(command >> 4);
  LCD_Enable();
  LCD_Send4Bit(command);
  LCD_Enable();

}
```

```
void LCD_Clear()
{
  LCD_SendCommand(0x01);
  delay_us(10);
}

void LCD_Init()
{
  LCD_Send4Bit(0x00);
  delay_ms(20);
  GPIOA->BRR = LCD_RS;
  GPIOA->BRR = LCD_RW;
  LCD_Send4Bit(0x03);
  LCD_Enable();
  delay_ms(5);
  LCD_Enable();
  delay_us(100);
  LCD_Enable();
  LCD_Send4Bit(0x02);
  LCD_Enable();
  LCD_SendCommand( 0x28 ); // giao thuc 4 bit, hien thi 2 hang, ki tu 5x8
  LCD_SendCommand( 0x0c); // cho phep hien thi man hinh
  LCD_SendCommand( 0x06 ); // tang ID, khong dich khung hinh
  LCD_SendCommand(0x01); // xoa toan bo khung hinh
}

void LCD_Gotoxy(unsigned char x, unsigned char y)
```

```
{
  unsigned char address;
  if(y == 0)address=(0x80+x);
  else if(y == 1) address=(0xc0+x);
  delay_ms(1);
  LCD_SendCommand(address);
  delay_us(50);
}

void LCD_PutChar(unsigned char Data)
{
  GPIO_SetBits(GPIOA, LCD_RS);
  LCD_SendCommand(Data);
  GPIO_ResetBits(GPIOA, LCD_RS);
}

void LCD_Puts(char *s)
{
  while (*s)
  {
    LCD_PutChar(*s);
    s++;
  }
}

void TempShow(unsigned char z)
{
```

```
  LCD_PutChar((z/100)+48); //Tram
  LCD_PutChar((z%100/10)+48); //Chuc
  LCD_PutChar((z%10)+48); //Don vi
}
File LCD.h
#ifndef __LCD_H
#define __LCD_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f10x.h"
#include "delay.h"

void GPIO_LCD_Config(void);
void LCD_Enable(void);
void LCD_Send4Bit(unsigned char Data);
void LCD_SendCommand(unsigned char command);
void LCD_Clear();
void LCD_Init();
void LCD_Gotoxy(unsigned char x, unsigned char y);
void LCD_PutChar(unsigned char Data);
void LCD_Puts(char *s);
```

```
#ifdef __cplusplus
}
#endif

#endif /* __MISC_H */
```

Add hết các file cần thiết vào và chạy chương trình

Lệnh Set DDRAM address

Lệnh này dùng để di chuyển con trỏ đến vị trí bất kỳ trong vùng nhớ DDRAM. Do đó, chúng ta có thể hiển thị ký tự hoặc chuỗi tại một vị trí bất kỳ.

Mã lệnh:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	1	[AD]	[AD]	[AD]	[AD]	[AD]	[AD]	[AD]

Ví dụ: Chúng ta muốn xuất ký tự A ra LCD tại vị trí dòng 1, cột 5. Như vậy, chúng ta cần di chuyển con trỏ đến ô nhớ có địa chỉ là 0x04.

Mã lệnh:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
	1	0	0	0	0	1	0	0	**0x84**

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
				A												
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50

```
Lcd_Cmd(0x84);
Lcd_Chr_Cp('A');
```

Sau khi hiển thị một kí tự tại một vị trí có sẵn, chúng ta có thể tiến đến bước xa hơn, dịch chuyển kí tự mà chúng ta muốn (dịch trái, dịch phải) với câu lệnh như hình dưới.

Lệnh dịch nội dung hiển thị

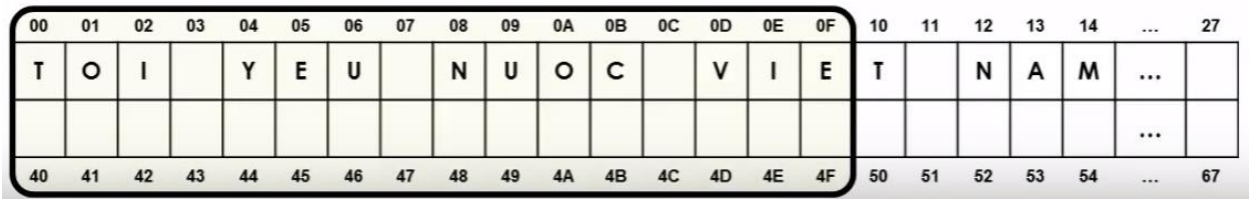
- Lệnh 0x18: Dịch toàn bộ nội dung đang hiển thị sang trái
- Lệnh 0x1C: Dịch toàn bộ nội dung đang hiển thị sang phải

01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	...	27
O	I		Y	E	U		N	U	O	C		V	I	E	T		N	A	M	...	
																				...	
41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	...	67

Cuối cùng, khi dòng chữ đang chạy, các em muốn dừng không cho chạy nữa, trở lại vị trí như ban đầu, có thể dùng lệnh sau:

Lệnh return home

- **Mã lệnh:** 0x02 hoặc 0x03
- **Chức năng:**
  Trả lại kiểu hiển thị gốc nếu nó bị thay đổi.
  Nội dung của DDRAM không thay đổi.
  Con trỏ trở về vị trí đầu dòng đầu tiên.



Bài tập 2: Hiển thị 2 dòng chữ - Số thứ tự nhóm và Họ và tên một bạn trong nhóm

Bài tập 3: Hiển thị Tên 2 bạn ở vị trí chính giữa màn hình, dòng 1 và dòng 2

Bài tập 4: Dịch phải và dịch trái một đoạn text dài hơn 16 kí tự.