Báo Cáo Lập Trình Nhúng.

Trường đại học NTU - Lê Hoàng Nam – MSSV 61133986

Giảng viên bộ môn: Mai Cường Thọ

N T U  Khoa Công nghệ thông tin – Lớp 61CNTT.1

Mục lục

Chương I:

[Bài 1. Blinking Led 3](#_Toc85899977)

[1. Mô tả: 3](#_Toc85899978)

[2. Sơ đồ: 3](#_Toc85899979)

[3. Đặt điểm linh kiện: 3](#_Toc85899980)

[4. Mã lệnh chính: 3](#_Toc85899981)

[Bài 2. Led Button 4](#_Toc85899982)

[1. Mô tả: 4](#_Toc85899983)

[2. Sơ đồ: 4](#_Toc85899984)

[3. Đặt điểm linh kiện: 4](#_Toc85899985)

[4. Mã lệnh chính: 4](#_Toc85899986)

[Bài 3. TMP 6](#_Toc85899987)

[1. Mô tả: 6](#_Toc85899988)

[2. Sơ đồ: 6](#_Toc85899989)

[3. Đặt điểm linh kiện: 6](#_Toc85899990)

[4. Mã lệnh chính: 6](#_Toc85899991)

[Bài 4. RGB 8](#_Toc85899992)

[5. Mô tả: 8](#_Toc85899993)

[6. Sơ đồ: 8](#_Toc85899994)

[7. Đặt điểm linh kiện: 8](#_Toc85899995)

[8. Mã lệnh chính: 8](#_Toc85899996)

[Bài 5. led 7 đoạn 11](#_Toc85899997)

[1. Mô tả: 11](#_Toc85899998)

[2. Sơ đồ: 11](#_Toc85899999)

[3. Đặt điểm linh kiện: 11](#_Toc85900000)

[4. Mã lệnh chính: 11](#_Toc85900001)

[Bài 6. 8 Led 14](#_Toc85900002)

[1. Mô tả: 14](#_Toc85900003)

[2. Sơ đồ: 14](#_Toc85900004)

[3. Đặt điểm linh kiện: 14](#_Toc85900005)

[4. Mã lệnh chính: 14](#_Toc85900006)

[Bài 7. Led Matrix 16](#_Toc85900007)

[1. Mô tả: 16](#_Toc85900008)

[5. Sơ đồ: 16](#_Toc85900009)

[6. Đặt điểm linh kiện: 16](#_Toc85900010)

[7. Mã lệnh chính: 16](#_Toc85900011)

[Bài 8. Nhúng LCD (Kiểm tra) 18](#_Toc85900012)

[1. Mô tả: 18](#_Toc85900013)

[2. Sơ đồ: 18](#_Toc85900014)

[3. Đặt điểm linh kiện: 18](#_Toc85900015)

[4. Mã lệnh chính: 18](#_Toc85900016)

[Bài 9. Led Matrix (Kiểm tra) 20](#_Toc85900017)

[1. Mô tả: 20](#_Toc85900018)

[2. Sơ đồ: 20](#_Toc85900019)

[3. Đặt điểm linh kiện: 20](#_Toc85900020)

[4. Mã lệnh chính: 20](#_Toc85900021)

Chương II:

[Bài 1: Blink led 24](#_Toc85900022)

[5. Mô tả: 24](#_Toc85900023)

[6. Sơ đồ: 24](#_Toc85900024)

[7. Đặt điểm linh kiện: 24](#_Toc85900025)

[8. Mã lệnh chính: 24](#_Toc85900026)

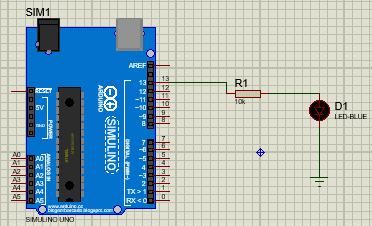
Chương I:

# Bài 1. Blinking Led

## Mô tả:

* Làm đèn sáng nhấp nháy.

## Sơ đồ:



Hình Sơ đồ Proteus Blinking Led

## Đặt điểm linh kiện:

* Led.
* Điện trở.

## Mã lệnh chính:

void setup() {

pinMode(13, OUTPUT);

}

void loop() {

digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(13, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

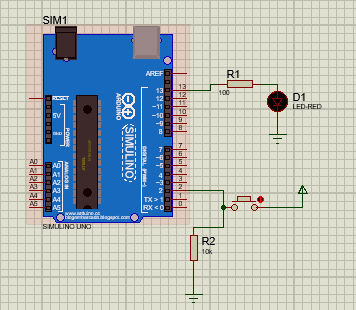
}

# Bài 2. Led Button

## Mô tả:

* Làm đèn sáng sau khi nhấn nút.

## Sơ đồ:



Hình Sơ đồ Proteus Led Button

## Đặt điểm linh kiện:

* Led.
* Điện trở.
* Nút bấm.

## Mã lệnh chính:

int x = 0;

void setup() {

pinMode(2,INPUT);

pinMode(13,OUTPUT);

}

void loop() {

x = digitalRead(2);

if (x==HIGH){

digitalWrite(13,HIGH);

}

else{

digitalWrite(13,LOW);

}

delay(1000);

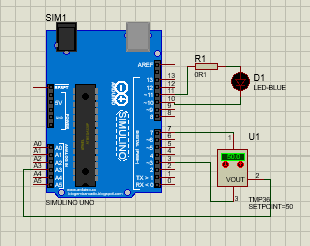
}

# Bài 3. TMP

## Mô tả:

* Làm đèn sáng sau khi đạt nhiệt độ nhất định.

## Sơ đồ:



Hình Sơ đồ Proteus TMP

## Đặt điểm linh kiện:

* Led RGB.
* Điện trở.
* TMP 36.

## Mã lệnh chính:

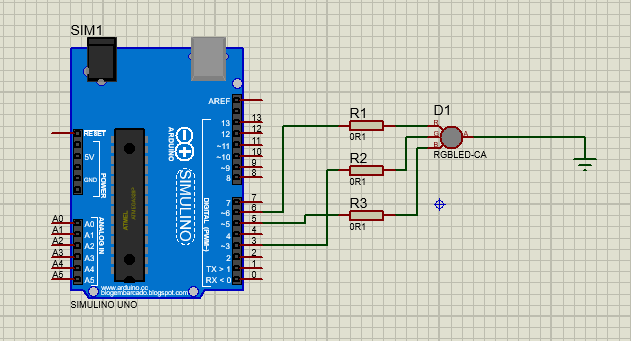
|  |
| --- |
| void setup()  {  Serial.begin(9600);  pinMode(7,1);  pinMode(3,1);  digitalWrite(7,1);  digitalWrite(3,0);  digitalWrite(10,1);  digitalWrite(11,0);  pinMode(11,1);  }  void loop()  {  Serial.println(analogRead(A3));  if(analogRead(A3)<181)digitalWrite(10,1);  if(analogRead(A3)>181)digitalWrite(10,0);} |

# Bài 4. RGB

## Mô tả:

* Làm Led RGB sáng và đổi màu liên tục.

## Sơ đồ:



Hình Sơ đồ Proteus LED RGB

## Đặt điểm linh kiện:

* Led RGB.
* Điện trở.

## Mã lệnh chính:

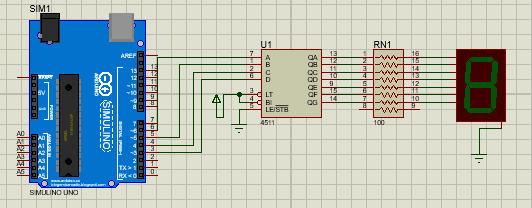
|  |
| --- |
| const int R = 3;  const int G = 5;  const int B = 6;  int Red = 255;  int Green = 0;  int Blue = 0;  void setup()  {  pinMode(R, OUTPUT);  pinMode(G, OUTPUT);  pinMode(B, OUTPUT);  analogWrite(R, Red);  analogWrite(G, Green);  analogWrite(B, Blue);  }  void loop() {  for (Blue = 0; Blue<255; Blue = Blue + 5) {  analogWrite(B, Blue);  delay(10);  }  for (Red = 255; Red>0; Red = Red - 5) {  analogWrite(R, Red);  delay(10);  }  for (Green = 0; Green<255; Green = Green + 5) {  analogWrite(G, Green);  delay(10);  }  for (Blue = 255; Blue>0; Blue = Blue - 5) {  analogWrite(B, Blue);  delay(10);  }  for (Red = 0; Red<255; Red = Red + 5) {  analogWrite(R, Red);  delay(10);  }  for (Green = 255; Green>0; Green = Green - 5) {  analogWrite(G, Green);  delay(10);  }  }  pinMode(11,1);  }  void loop()  {  Serial.println(analogRead(A3));  if(analogRead(A3)<181)digitalWrite(10,1);  if(analogRead(A3)>181)digitalWrite(10,0);} |

# Bài 5. led 7 đoạn

## Mô tả:

* Điều khiển led 7 đoạn hiển thị các số tự nhiên bằng tín hiệu nhị phân.

## Sơ đồ:



Hình Sơ đồ Proteus Led 7 đoạn

## Đặt điểm linh kiện:

* Led 7 đoạn.
* Điện trở.
* 4511 IC.

## Mã lệnh chính:

int a=6, b=5, c=4, d=3;

void setup() {

pinMode(a, OUTPUT);

pinMode(b, OUTPUT);

pinMode(c, OUTPUT);

pinMode(d, OUTPUT); }

void khong(){

digitalWrite(a, LOW);

digitalWrite(b, LOW);

digitalWrite(c, LOW);

digitalWrite(d, LOW); }

void mot(){

digitalWrite(a, HIGH);

digitalWrite(b, LOW);

digitalWrite(c, LOW);

digitalWrite(d, LOW);

}

void hai(){

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, LOW);

digitalWrite(d, LOW);

}

void ba(){

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, LOW);

digitalWrite(d, LOW);

}

void bon(){

digitalWrite(a, LOW);

digitalWrite(b, LOW);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

}

void nam(){

digitalWrite(a, HIGH);

digitalWrite(b, LOW);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

}

void sau(){

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

}

void bay(){

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

}

void tam(){

digitalWrite(a, LOW);

digitalWrite(b, LOW);

digitalWrite(c, LOW);

digitalWrite(d, HIGH);

}

void chin(){

digitalWrite(a, HIGH);

digitalWrite(b, LOW);

digitalWrite(c, LOW);

digitalWrite(d, HIGH);

}

void loop() {

khong(); delay(1000);

mot(); delay(1000);

hai(); delay(1000);

ba(); delay(1000);

bon(); delay(1000);

nam(); delay(1000);

sau(); delay(1000);

bay(); delay(1000);

tam(); delay(1000);

chin(); delay(1000);

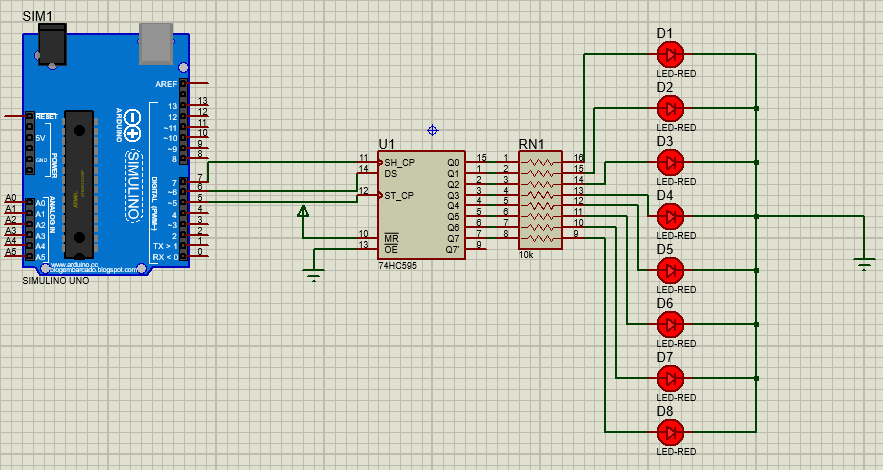
}

# Bài 6. 8 Led

## Mô tả:

* Điều khiển led 7 đoạn hiển thị các số tự nhiên bằng tín hiệu nhị phân.

## Sơ đồ:



Hình 6 Sơ đồ Proteus 8 Led

## Đặt điểm linh kiện:

* Led.
* Điện trở.
* 74HC595 IC.

## Mã lệnh chính:

int latchPin = 5;

int clockPin = 7;

int dataPin = 6;

byte ledStatus;

void setup() {

pinMode(latchPin, OUTPUT);

pinMode(clockPin, OUTPUT);

pinMode(dataPin, OUTPUT);

}

void loop() {

//Sáng tuần tự

ledStatus = 0;//mặc định là không có đèn nào sáng hết (0 = 0b00000000)

digitalWrite(latchPin, LOW ledStatus = 0b11111111;

//ShiftOut ra IC

shiftOut(dataPin, clockPin, MSBFIRST, ledStatus);

digitalWrite(latchPin, HIGH);//các đèn LED sẽ sáng với trạng thái vừa được cập nhập

delay(1000);

digitalWrite(latchPin, LOW);

ledStatus = 0;

//ShiftOut ra IC

shiftOut(dataPin, clockPin, MSBFIRST, ledStatus);

digitalWrite(latchPin, HIGH);//các đèn LED sẽ sáng với trạng thái vừa được cập nhập

delay(250);

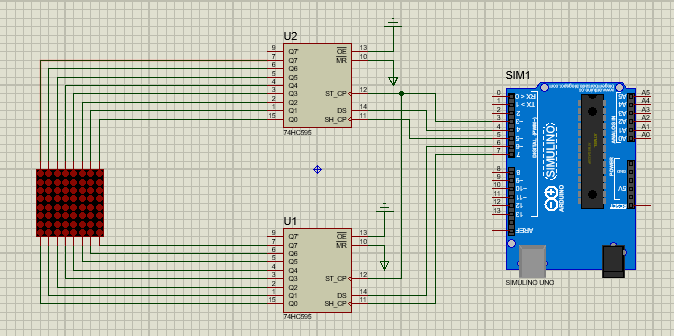
}

# Bài 7. Led Matrix

## Mô tả:

* Điều khiển led matrix hiển thị chữ A.

## Sơ đồ:



Hình 7 Sơ đồ Proteus Led Matrix

## Đặt điểm linh kiện:

* Led Matrix.
* Điện trở.
* 74HC595 IC.

## Mã lệnh chính:

#define \_data\_hang 4

#define \_clock\_hang 5

#define \_data\_cot 6

#define \_clock\_cot 7

#define \_latch 3

byte chu[][8] = {

{0xFF,0xC0,0x80,0xB7,0xB7,0x80,0xC0,0xFF}, //A

};

byte hang = 0b10000000;

void setup() {

Serial.begin(9600);

pinMode(\_latch,OUTPUT);//RCLK

pinMode(\_data\_hang,OUTPUT);//SER hang

pinMode(\_clock\_hang,OUTPUT);//SRCLK hang

pinMode(\_clock\_cot,OUTPUT);//SRCLK cot

pinMode(\_data\_cot,OUTPUT);//SER cot

}

void loop() {

for(int i = 0;i<10;i++){

digitalWrite(\_latch,LOW);

shiftOut(\_data\_cot,\_clock\_cot,LSBFIRST,chu[0][i]);// hang 8 - hang 1

shiftOut(\_data\_hang,\_clock\_hang,LSBFIRST,hang >> i);// cot 8 - cot 1

digitalWrite(\_latch,HIGH);

delay(1);

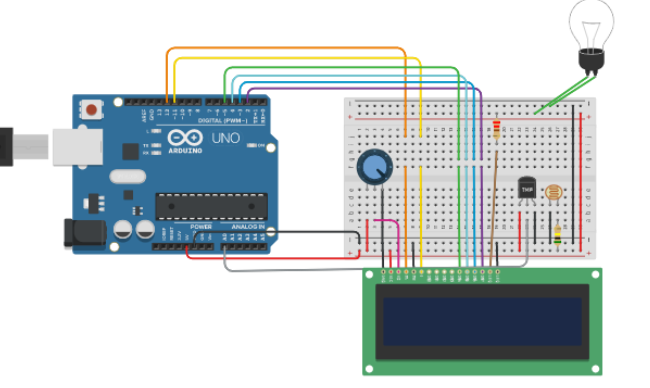
}

# Bài 8. Nhúng LCD (Kiểm tra)

## Mô tả:

* Liên tục đọc giá trị độ sáng và hiển thị ra màn hình LCD.
* Nếu giá trị <50% thì bật đèn sáng và ngược lại.

## Sơ đồ:



Hình 8 Sơ đồ Proteus LCD

## Đặt điểm linh kiện:

* Led LCD.
* Điện trở.
* Light bult.

## Mã lệnh chính:

#include <LiquidCrystal.h>

#define SENSOR\_PIN A0

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int LIGHT = 0;

int val =0;

void setup() {

// set up the LCD's number of columns and rows:

pinMode(9,OUTPUT);

digitalWrite(9,LOW);

lcd.begin(16, 2);

// Print a message to the LCD.

Serial.begin(9600);

}

void loop() {

val = analogRead(LIGHT);

if (val < 100)

{

digitalWrite (9, HIGH);

delay(1000);

}

else

{

digitalWrite (9, LOW);

delay(1000);

}

// set the cursor to column 0, line 1

// (note: line 1 is the second row, since counting begins with 0):

lcd.setCursor(0, 0);

// print the number of seconds since reset:

lcd.print("do sang:");

lcd.setCursor(0, 1);

lcd.print(val);

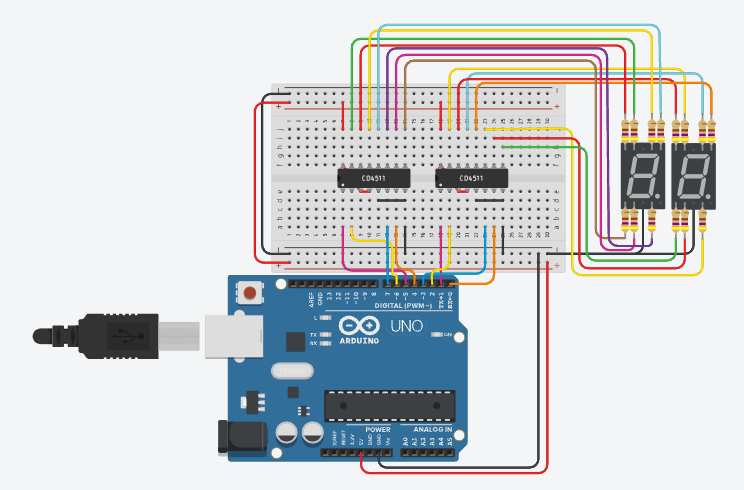
}

# Bài 9. Led Matrix (Kiểm tra)

## Mô tả:

* Hiển thị các số từ 00 đến 99 bằng Led 7 đoạn.

## Sơ đồ:



Hình 9 Sơ đồ Proteus 8 Led

## Đặt điểm linh kiện:

* Led 7 đoạn.
* Điện trở.
* 4511 IC.

## Mã lệnh chính:

void setup()

{

pinMode(0, OUTPUT);

pinMode(1, OUTPUT);

pinMode(2, OUTPUT);

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

pinMode(5, OUTPUT);

pinMode(6, OUTPUT);

pinMode(7, OUTPUT);

}

void show\_so(int so, int led = 0)

{

if (so == 0)

{

digitalWrite(0 + led, LOW);

digitalWrite(1 + led, LOW);

digitalWrite(2 + led, LOW);

digitalWrite(3 + led, LOW);

}

else if (so == 1)

{

digitalWrite(0 + led, HIGH);

digitalWrite(1 + led, LOW);

digitalWrite(2 + led, LOW);

digitalWrite(3 + led, LOW);

}

else if (so == 2)

{

digitalWrite(0 + led, LOW);

digitalWrite(1 + led, HIGH);

digitalWrite(2 + led, LOW);

digitalWrite(3 + led, LOW);

}

else if (so == 3)

{

digitalWrite(0 + led, HIGH);

digitalWrite(1 + led, HIGH);

digitalWrite(2 + led, LOW);

digitalWrite(3 + led, LOW);

}

else if (so == 4)

{

digitalWrite(0 + led, LOW);

digitalWrite(1 + led, LOW);

digitalWrite(2 + led, HIGH);

digitalWrite(3 + led, LOW);

}

else if (so == 5)

{

digitalWrite(0 + led, HIGH);

digitalWrite(1 + led, LOW);

digitalWrite(2 + led, HIGH);

digitalWrite(3 + led, LOW);

}

else if (so == 6)

{

digitalWrite(0 + led, LOW);

digitalWrite(1 + led, HIGH);

digitalWrite(2 + led, HIGH);

digitalWrite(3 + led, LOW);

}

else if (so == 7)

{

digitalWrite(0 + led, HIGH);

digitalWrite(1 + led, HIGH);

digitalWrite(2 + led, HIGH);

digitalWrite(3 + led, LOW);

}

else if (so == 8)

{

digitalWrite(0 + led, LOW);

digitalWrite(1 + led, LOW);

digitalWrite(2 + led, LOW);

digitalWrite(3 + led, HIGH);

}

else if (so == 9)

{

digitalWrite(0 + led, HIGH);

digitalWrite(1 + led, LOW);

digitalWrite(2 + led, LOW);

digitalWrite(3 + led, HIGH);

}

}

void loop()

{

static int num1 = 0;

static int num2 = 0;

show\_so(num1 % 10, 4);

show\_so(num2 % 10);

num2++;

if (num2 % 10 == 0)

{

num2 = 0;

num1++;

}

delay(1000);

}

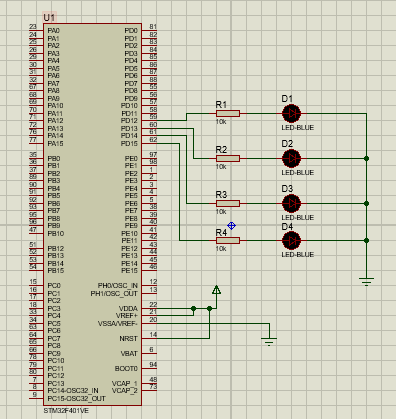
Chương II:

# Bài 8: Blink led

## Mô tả:

* Làm led nhấp nháy với STM32.

## Sơ đồ:



Hình 10 Sơ đồ Proteus 8 Led

## Đặt điểm linh kiện:

* Led 7 đoạn.
* Điện trở.
* 4511 IC.

## Mã lệnh chính:

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under BSD 3-Clause license,

\* the "License"; You may not use this file except in compliance with the

\* License. You may obtain a copy of the License at:

\* opensource.org/licenses/BSD-3-Clause

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "stdio.h"

#include "stm32f4xx\_hal.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE2);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 25;

RCC\_OscInitStruct.PLL.PLLN = 168;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14|GPIO\_PIN\_15, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : PD12 PD13 PD14 PD15 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14|GPIO\_PIN\_15;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_HIGH;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

# Bài 9: chip STM32F401VE sáng led trái tim (22led)

## 1.Mô tả:

* Điều khiển nháy sáng đèn led hình trái tim 22 led cạnh nhau và các hiệu ứng của nó qua mạch RES16DIPIS kết nối chip STM32F401VE.

## 2.Sơ đồ thiết kế

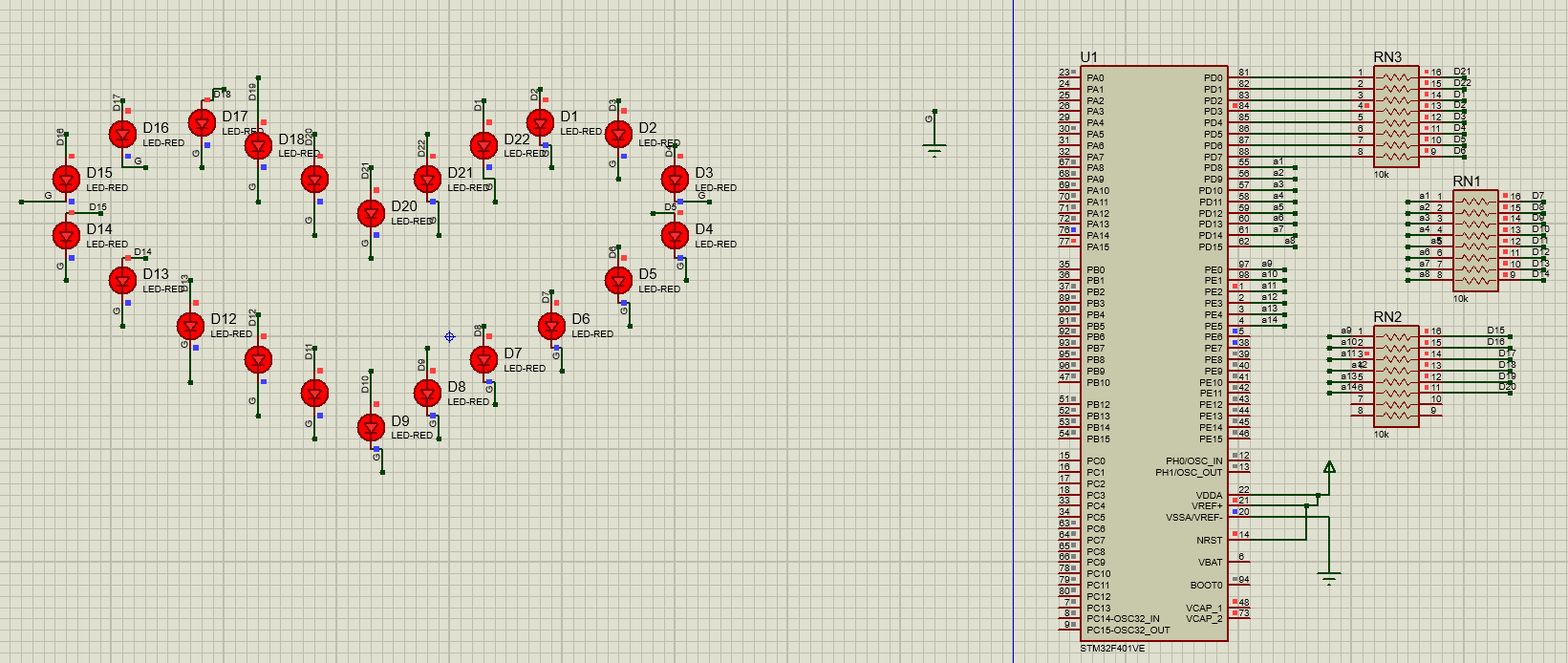


Figure 9 Sơ đồ thiết kế

## 3.Đặc điểm linh kiện:

* 22 Led-Red
* 3 mạch RES16DIPIS
* Chip STM32F401VE

## 4.Code:

#include "main.h"

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

void SANGLED();

void TATLED();

void NHAPNHAY();

void NHAPNHAY3();

void NHAPNHA();

int main(void)

{

HAL\_Init();

SystemClock\_Config();

MX\_GPIO\_Init();

while (1)

{

SANGLED();

HAL\_Delay(1000);

TATLED();

HAL\_Delay(1000);

NHAPNHAY();

HAL\_Delay(500);

NHAPNHAY();

NHAPNHAY3();

NHAPNHA();

}

}

void TATLED()

{

SANGLED();

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_11, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

void SANGLED()

{

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_8, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_9, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_10, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_11, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_5, GPIO\_PIN\_SET);

}

void NHAPNHAY()

{

TATLED();

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_9, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_11, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(100);

}

void NHAPNHAY3()

{

TATLED();

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_Delay(1000);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_Delay(1000);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_Delay(1000);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_Delay(1000);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_Delay(1000);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_Delay(1000);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_8, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_9, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_11, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(100);

}

void NHAPNHA()

{

TATLED();

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_8, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_8, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_9, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_9, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_10, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_10, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_11, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_11, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_11, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_5, GPIO\_PIN\_SET);

SANGLED();

HAL\_Delay(100);

TATLED();

HAL\_Delay(500);

SANGLED ();

HAL\_Delay(500);

TATLED();

HAL\_Delay(500);

SANGLED ();

HAL\_Delay(500);

TATLED();

HAL\_Delay(500);

SANGLED ();

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE2);

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOE, GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5

|GPIO\_PIN\_6|GPIO\_PIN\_7|GPIO\_PIN\_0|GPIO\_PIN\_1, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_8|GPIO\_PIN\_9|GPIO\_PIN\_10|GPIO\_PIN\_11

|GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14|GPIO\_PIN\_15

|GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3

|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6|GPIO\_PIN\_7, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : PE2 PE3 PE4 PE5

PE6 PE7 PE0 PE1 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_2|GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5

|GPIO\_PIN\_6|GPIO\_PIN\_7|GPIO\_PIN\_0|GPIO\_PIN\_1;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : PD8 PD9 PD10 PD11

PD12 PD13 PD14 PD15

PD0 PD1 PD2 PD3

PD4 PD5 PD6 PD7 \*/

GPIO\_InitStruct.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

|GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3

|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6|GPIO\_PIN\_7;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

}

void Error\_Handler(void)

{

\_\_disable\_irq();

while (1)

{}

}

#ifdef USE\_FULL\_ASSERT

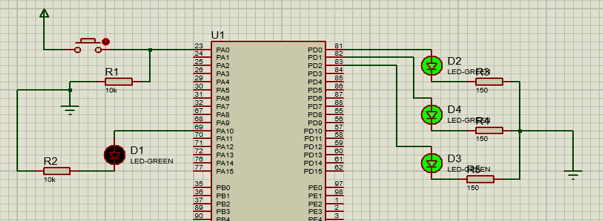
#endif

# Bài 10: Ngắt ngoài và ngắt ưu tiên trên STM32F4

## Mô tả:

* Ngắt ngoài bằng nút bấm điều khiển đèn led và mạch chính 3 đèn led vẫn hoat đông song hành đươc kết nối chim STM32F401VE.

## Sơ đồ:



Hình 10 Sơ đồ Proteus 8 Led

## Đặt điểm linh kiện:

* Led 7 đoạn.
* Điện trở.
* 4511 IC.

## Mã lệnh chính:

#include "main.h"

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

int main(void){

HAL\_Init();

SystemClock\_Config();

MX\_GPIO\_Init();

while (1)

{

HAL\_GPIO\_TogglePin(GPIOD, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2);

HAL\_Delay(1000);

}

}

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)

{

if (GPIO\_Pin == GPIO\_PIN\_0)

{

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_10);

}

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE2);

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2, GPIO\_PIN\_RESET);

GPIO\_InitStruct.Pin = GPIO\_PIN\_0;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pin : PA10 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_10;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pins : PD0 PD1 PD2 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\* EXTI interrupt init\*/

HAL\_NVIC\_SetPriority(EXTI0\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(EXTI0\_IRQn);

}

void Error\_Handler(void)

{

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

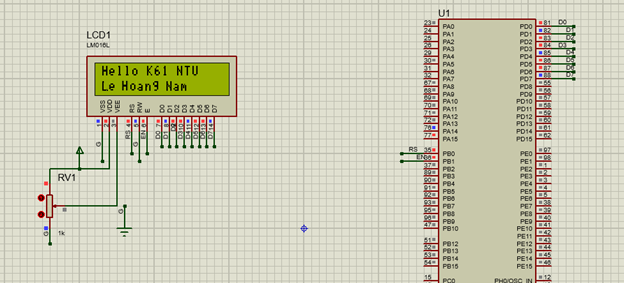
#ifdef USE\_FULL\_ASSERT

# Bài 12: STM32 LCD

## Mô tả:

* Bài này ta thực hiện việc lập trình điều khiển qua lệnh , con trỏ để hiển thị thông tin trên màn hình LCD .
* LCD là một màn hình nhiều điểm ảnh, có thể coi là một Led ma trận dạng lớn

## Sơ đồ:



Hình 10 Sơ đồ Proteus 8 Led

## Đặt điểm linh kiện:

* LCD
* STM32F401VE IC.

## Mã lệnh chính:

#include "main.h"

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

//HAM GUI LENH

void send8bit(char D)

{

int b0,b1,b2,b3,b4,b5,b6,b7;

if ((D &1)==0) b0=0; //2^0

else b0=1;

if ((D &2)==0) b1=0;

else b1=1;

if ((D &4)==0) b2=0;

else b2=1;

if ((D &8)==0) b3=0;

else b3=1;

if ((D &16)==0) b4=0;

else b4=1;

if ((D &32)==0) b5=0;

else b5=1;

if ((D &64)==0) b6=0;

else b6=1;

if ((D &128)==0) b7=0;

else b7=1;

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_0,b0);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_1,b1);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_2,b2);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_3,b3);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_4,b4);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_5,b5);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_6,b6);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_7,b7);

}

void sendlcd(char cmd)

{

// Rw dat muc thap

//b2 dat chan rs=0,de noi rang cmd la lenh

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

//B3 GUI 8 BIET VAO 8 PIN

send8bit(cmd);

//b4 enable cho kytu -- < lcd

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_Delay(10);

}

// ham gui ky tu

void sendchar(char kytu)

{

// Rw dat muc thap

//b2 dat chan rs=1

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);

//B3 GUI 8 BIET VAO 8 PIN

send8bit(kytu);

//b4 enable cho kytu -- < lcd

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_Delay(10);

}

void sendstring(char \*str)

{

for (int i =0;str[i] !='\0';i++)

sendchar(str[i]);

}

int main(void)

{

HAL\_Init();

SystemClock\_Config();

MX\_GPIO\_Init();

//xoa noi dung

sendlcd(0x01);

// bat hien thi man hinh, tat con tro

sendlcd(0x0C);

//hello

sendstring("Hello K61 NTU");

//xuong dong

sendlcd(0x38);//che do 2 dong

sendlcd(0xC0);

sendstring("Le Hoang Nam");

while (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE2);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3

|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6|GPIO\_PIN\_7, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : PB0 PB1 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_1;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

/\*Configure GPIO pins : PD0 PD1 PD2 PD3

PD4 PD5 PD6 PD7 \*/

GPIO\_InitStruct.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;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT