**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**

**----oOo----**

**Text, logo

Description automatically generated**

**VI ĐIỀU KHIỂN VÀ ỨNG DỤNG**

**BÀI BÁO CÁO 10**

**NHÓM 5**

Giảng viên: **Phạm Quang Trí**

Sinh viên:

Nguyễn Ngọc Huy\_21115981

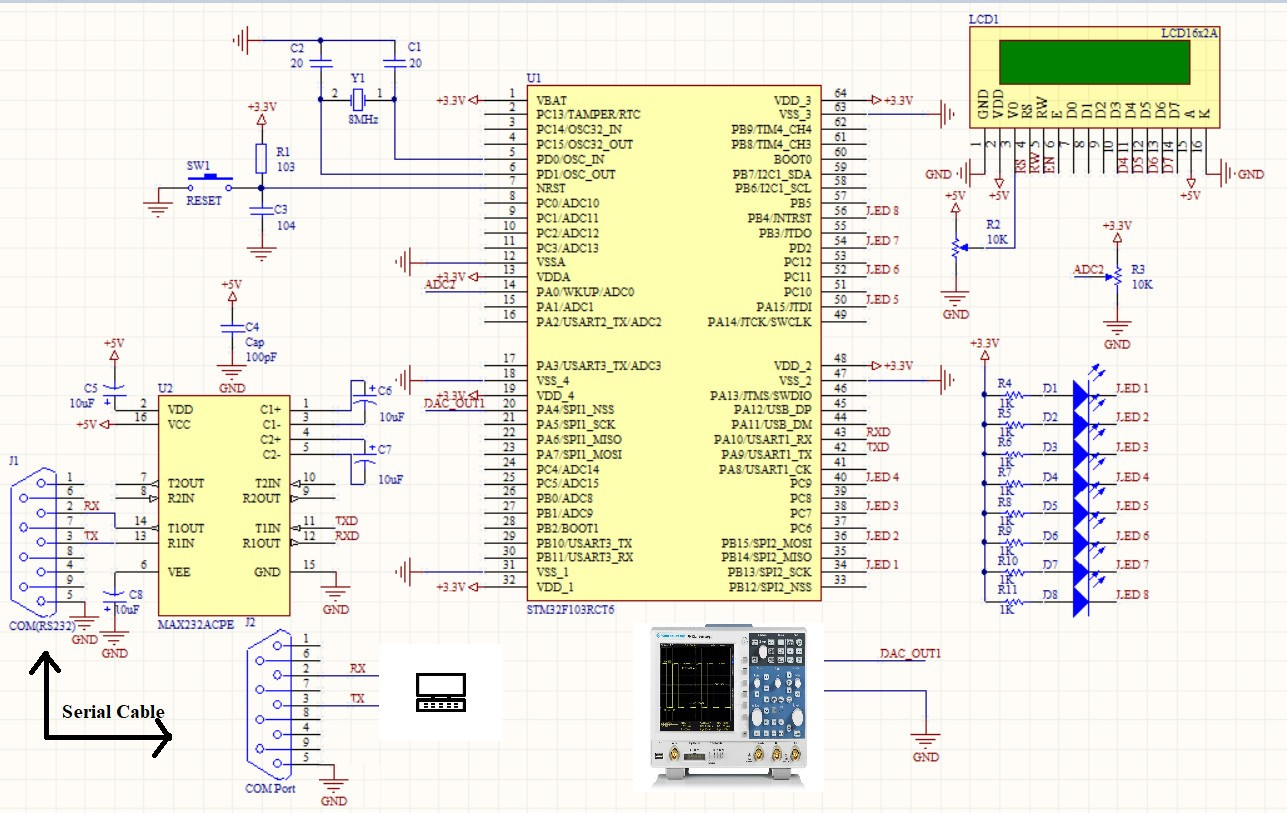
Nguyễn Thị Như Huyền\_21126481

Hồ Bửu Sơn\_21119571

**TP.HCM 2023**

**Yêu cầu 1**: **Trong quyển báo cáo, dựa vào yêu cầu bài tập đã chọn, phải vẽ sơ đồ kết nối phần cứng của toàn bộ hệ thống sử dụng vi điều khiển STM32.**

* Sơ đồ kết nối phần cứng



**Yêu cầu 2: Trong quyển báo cáo, trình bày rõ ràng lưu đồ giải thuật và mã nguồn của chương trình điều khiển**

* Lưu đồ giải thuật

lcd\_send\_cmd

lcd\_write\_nibble

Gửi 4 bit của lệnh (4 bit cao)

Chọn chế độ lệnh hoặc dữ liệu cho LCD

Gửi 4 bit của lệnh (4 bit thấp)

Cho phép đọc thông tin từ LCD

Delay 2ms

Cho phép LCD trao đổi thông tin

Xuất 4 bit dữ liệu ra các chân kết nối D4-D7 của LCD

lcd\_send\_data

Gửi 4 bit của lệnh (4 bit cao)

Delay 1us

Gửi 4 bit của lệnh (4 bit thấp)

Không cho phép LCD trao đổi thông tin

Delay 2ms

Delay 1us

lcd\_display

Tăng con trỏ dữ liệu

lcd\_init

Delay 20ms

Đúng

Chưa hết chuỗi dữ liệu

Gửi dữ liệu hiển thị

Đặt chế độ LCD 8 bit dữ liệu

Sai

Delay 5ms

Đặt chế độ LCD 8 bit dữ liệu

Delay 1ms

lcd\_gotoxy

Đặt chế độ LCD 4 bit   
Hiển thị 2 dòng thông tin  
Kiểu ký tự 5x8

Mã tọa độ bằng 0

Biểu thị thông tin   
Tắt hiển thị con trỏ

Đúng

Hàng trên ?   
(row=0)

Mã tọa độ = 0x80+col

Xóa màn hình LCD

Sai

Hàng dưới?   
(row=1)

Đúng

Mã tọa độ = 0xC0+col

Chế độ tăng con trỏ tự động và không dịch chuyển màn hình hiển thị

Sai

Gửi lệnh điều khiển (mã tọa độ)

StartTask1\_led

get\_value

i = 0

Đảo trạng thái LED 1

Tính toán giá trị và gán vào mảng value

Đảo trạng thái LED 2

Đảo trạng thái LED 3

tăng i

Đảo trạng thái LED 4

Đúng

i < NS ?

Đảo trạng thái LED 5 5

Sai

Đảo trạng thái LED 6

Đảo trạng thái LED 7

Đảo trạng thái LED 8

Delay 1s

StartTask03\_dac

Khởi động bộ đếm thời gian timer 2

StartTask02\_nhietdo

Bắt đầu chuyển đổi ADC kênh 1

Gọi hàm get\_value

Chờ cho quá trình chuyển đổi ADC kênh 1 hoàn tất

Cho phép kênh 1 DAC ở chế độ DMA hoạt động và xuất giá trị DAC ra ngoài

Lấy giá trị ADC kênh 1 gán vào biến 'adc\_value1'

Delay 1ms

Tính toán giá trị chuyển đổi ADC và gán vào biến 'v1'

StartTask04\_lcd

Tính toán giá trị nhiệt độ và gán vào biến 'nhietdo'

Khởi động LCD

B

Định dạng chuỗi dữ liệu

Xóa màn hình LCD

Bắt đầu chuyển đổi ADC kênh 2

Định dạng chuỗi dữ liệu gửi qua UART

Chờ cho quá trình chuyển đổi ADC kênh 2 hoàn tất

Gửi chuỗi dữ liệu qua UART

Lấy giá trị ADC kênh 2 gán vào biến 'adc\_value2'

Delay 1s

Tính toán giá trị chuyển đổi ADC và gán vào biến 'v2'

A

A

Định dạng chuỗi dữ liệu gửi lên LCD

Định vị tọa độ "0,0"

Hiển thị chuỗi dữ liệu là giá trị điện áp lên LCD

Delay 500ms

B

* Mã nguồn chương trình

#include "main.h"

#include "cmsis\_os.h"

#define NS 100

#define PI 3.14159254

ADC\_HandleTypeDef hadc1;

ADC\_HandleTypeDef hadc2;

DAC\_HandleTypeDef hdac;

DMA\_HandleTypeDef hdma\_dac\_ch1;

TIM\_HandleTypeDef htim2;

UART\_HandleTypeDef huart1;

osThreadId myTask01Handle;

osThreadId myTask02Handle;

osThreadId myTask03Handle;

osThreadId myTask04Handle;

uint8\_t data\_send[25];

uint16\_t data\_send\_lcd[16];

uint32\_t value[NS];

float adc\_value1;

float adc\_value2;

float v1;

float v2;

float nhietdo;

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_DMA\_Init(void);

static void MX\_ADC1\_Init(void);

static void MX\_ADC2\_Init(void);

static void MX\_DAC\_Init(void);

static void MX\_TIM2\_Init(void);

static void MX\_USART1\_UART\_Init(void);

void StartTask1\_led(void const \* argument);

void StartTask02\_nhietdo(void const \* argument);

void StartTask03\_dac(void const \* argument);

void StartTask04\_lcd(void const \* argument);

void get\_value()

{

for(int i=0; i<100;i++)

{

value[i] = 780\*(sin(i\*2\*PI/NS))+10070;

}

}

void lcd\_write\_nibble(uint8\_t rs, uint8\_t data){

HAL\_GPIO\_WritePin(RS\_GPIO\_Port, RS\_Pin,rs);

HAL\_GPIO\_WritePin(RW\_GPIO\_Port, RW\_Pin,0);

HAL\_GPIO\_WritePin(EN\_GPIO\_Port, EN\_Pin,1);

HAL\_GPIO\_WritePin(D4\_GPIO\_Port, D4\_Pin, (data>>0) & 0x01);

HAL\_GPIO\_WritePin(D5\_GPIO\_Port, D5\_Pin, (data>>1) & 0x01);

HAL\_GPIO\_WritePin(D6\_GPIO\_Port, D6\_Pin, (data>>2) & 0x01);

HAL\_GPIO\_WritePin(D7\_GPIO\_Port, D7\_Pin, (data>>3) & 0x01);

for(uint8\_t i=0; i<72; i++) asm("NOP");

HAL\_GPIO\_WritePin(EN\_GPIO\_Port, EN\_Pin,0);

for(uint8\_t i=0; i<72; i++) asm("NOP");

}

void lcd\_send\_cmd(uint8\_t cmd){

lcd\_write\_nibble(0, (cmd>>4)&0x0F);

lcd\_write\_nibble(0, cmd&0x0F);

HAL\_Delay(2);

}

void lcd\_send\_data(uint8\_t data){

lcd\_write\_nibble(1, (data>>4)&0x0F);

lcd\_write\_nibble(1, data&0x0F);

HAL\_Delay(2);

}

void lcd\_init(void){

HAL\_Delay(20);

lcd\_write\_nibble(0, 0x03); HAL\_Delay(5);

lcd\_write\_nibble(0, 0x03); HAL\_Delay(1);

lcd\_write\_nibble(0, 0x03); HAL\_Delay(1);

lcd\_write\_nibble(0, 0x02); HAL\_Delay(1);

lcd\_send\_cmd(0x28);

lcd\_send\_cmd(0x0C);

lcd\_send\_cmd(0x01);

lcd\_send\_cmd(0x06);

}

void lcd\_display(uint8\_t \*data, uint8\_t len)

{

for (uint8\_t j = 0; j < len; j++)

{

lcd\_send\_data(data[j]);

}

}

void lcd\_gotoxy(uint8\_t row, uint8\_t col){

uint8\_t coordinates = 0;

switch(row){

case 0:

coordinates = 0x80 | col;

break;

case 1:

coordinates = 0xC0 | col;

break;

}

lcd\_send\_cmd(coordinates);

}

int main(void)

{

HAL\_Init();

SystemClock\_Config();

MX\_GPIO\_Init();

MX\_DMA\_Init();

MX\_ADC1\_Init();

MX\_ADC2\_Init();

MX\_DAC\_Init();

MX\_TIM2\_Init();

MX\_USART1\_UART\_Init();

osThreadDef(myTask01, StartTask1\_led, osPriorityNormal, 0, 300);

myTask01Handle = osThreadCreate(osThread(myTask01), NULL);

osThreadDef(myTask02, StartTask02\_nhietdo, osPriorityRealtime, 0, 300);

myTask02Handle = osThreadCreate(osThread(myTask02), NULL);

osThreadDef(myTask03, StartTask03\_dac, osPriorityNormal, 0, 4500);

myTask03Handle = osThreadCreate(osThread(myTask03), NULL);

osThreadDef(myTask04, StartTask04\_lcd, osPriorityRealtime, 0, 2500);

myTask04Handle = osThreadCreate(osThread(myTask04), NULL);

osKernelStart();

while (1)

{

}

}

void StartTask1\_led(void const \* argument)

{

for(;;)

{

HAL\_GPIO\_TogglePin(LED1\_GPIO\_Port, LED1\_Pin);

HAL\_GPIO\_TogglePin(LED2\_GPIO\_Port, LED2\_Pin);

HAL\_GPIO\_TogglePin(LED3\_GPIO\_Port, LED3\_Pin);

HAL\_GPIO\_TogglePin(LED4\_GPIO\_Port, LED4\_Pin);

HAL\_GPIO\_TogglePin(LED5\_GPIO\_Port, LED5\_Pin);

HAL\_GPIO\_TogglePin(LED6\_GPIO\_Port, LED6\_Pin);

HAL\_GPIO\_TogglePin(LED7\_GPIO\_Port, LED7\_Pin);

HAL\_GPIO\_TogglePin(LED8\_GPIO\_Port, LED8\_Pin);

osDelay(1);

}

}

void StartTask02\_nhietdo(void const \* argument)

{

for(;;)

{

HAL\_ADC\_Start(&hadc1);

for(int i = 0; i < 100; i++) asm("NOP");

adc\_value1 = HAL\_ADC\_GetValue(&hadc1);

v1 = adc\_value1 \* 3.3 / 4095;

nhietdo = (1.43 - v1) / (4.3/1000) + 25;

sprintf(&data\_send[0],"Nhiet do: %.2f C \r\n", nhietdo);

HAL\_UART\_Transmit(&huart1,&data\_send[0],sizeof(data\_send),10);

osDelay(1000);

}

}

void StartTask03\_dac(void const \* argument)

{

HAL\_DAC\_Start(&hdac, DAC1\_CHANNEL\_1);

HAL\_TIM\_Base\_Start(&htim2);

for(;;)

{

get\_value();

HAL\_DAC\_Start\_DMA(&hdac,DAC\_CHANNEL\_1,value,100,DAC\_ALI GN\_12B\_R);

osDelay(1);

}

}

void StartTask04\_lcd(void const \* argument)

{

lcd\_init();

for(;;)

{

lcd\_send\_cmd(0x01);

HAL\_ADC\_Start(&hadc2);

for(int i = 0; i < 10; i++) asm("NOP");

adc\_value2 = HAL\_ADC\_GetValue(&hadc2);

v2 = adc\_value2 \* 3.3 / 4095;

sprintf(&data\_send\_lcd[0]," DIEN AP: %.2f V ", v2);

lcd\_gotoxy(0, 0);

lcd\_display(&data\_send\_lcd,16);

osDelay(500);

}

}

**\* Công thức tính toán**

Dạng sin 8HZ; biên độ 0.7V; offset 1.5V

DAC(output) = Vref x DOR/4095 (1)

-Biên độ 0.7V thế vào công thức (1) ta được:

0.7 = 3.3 x X/4095 => X

-Offset 1.5V thế vào công thức (1) ta được:

1.5 = 3.3 x X/4095 => X

\* **Công thức tính tần số sóng sin**

fREQ= fCPU/((Pre+1)×(Counter period+1)×x)(2)

Chọn số lần lấy mẫu x= 100, fCPU = 72MHz

Prescaler =179, Counter period=499

Thế vào công thức (2) ta được:

1/(8Hz\*100)= ((179+1)\*(Counter period+1))/(72.10^6)

Chọn Pre = 179, Counter period = 499

\* **Công thức tính nhiệt độ STM32F103RCT6:**

Điện áp tham chiếu: 1.43(V)

Giá trị điện áp kênh 3: V[2]

Độ nhạy cảm biến trong STM32: 4.3(mV/°C)

Nhiệt độ phòng: 25°C

\* **Tính toán công thức thời gian chờ chuyển đổi ADC:**

Để tính giá trị thời gian tương đương khoảng 4~7 chu kỳ lệnh nhóm em chọn 6 vì sau khi tính nhận được kết quả ra tròn.

Tần số ADC: 12MHz

Với Rank1 của ADC1 là 239.5 Cyc

🡪 Chọn 100 vòng lặp

Với Rank1 của ADC2 là: 7.5 Cyc

🡪 Chọn 10 vòng lặp

\* **Công thức chuyển đổi ADC:**

Giá trị ADC sau khi chuyển đổi:

Giá trị điện áp sau khi chuyển đổi:

=3.3V

=0V

Độ phân giải: n=12

=4095

**Yêu cầu 3: Nạp chương trình vào kit thí nghiệm STM32 và thực hiện cho chạy thử trên phần cứng trong phòng thực hành. Quay video clip minh chứng kết quả thực hiện, tải lên Youtube duy nhất 1 video clip ( trong trường hợp minh chứng có nhiều video clip nhỏ thì sinh viên phải tự ghép lại thành 1 video clip tổng hợp ) và ghi liên kết vào báo cáo.**

Liên kết video: <https://youtu.be/1aWyirYKwBs?si=CUDJwWSwPGx1wdkS>