**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**A blue and white logo

Description automatically generatedKhoa: Điện tử - Viễn thông**

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

*Đồ án :* **MẠCH ĐẾM SỐ BƯỚC CHÂN HIỂN THỊ LCD OLED**

**Môn học**: **Giảng viên môn học**:

Năm học:

Nhóm sinh viên thực hiện:

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và Tên** | **MSSV** |
| 1 |  |  |
| 2 |  |  |

**MỤC LỤC**

1. **TÌM HIỂU VỀ THIẾT BỊ ĐẾM SỐ BƯỚC CHÂN VÀ PHƯƠNG PHÁP ĐO SỐ BƯỚC CHÂN**

## **Định nghĩa**

Thiết bị đếm bước chân là một công cụ đo lường sự vận động của cơ thể thông qua việc đếm số bước di chuyển của người sử dụng trong một khoảng thời gian nhất định. Việc sở hữu một thiết bị đếm bước chân hoặc sử dụng các ứng dụng liên quan có nhiều lợi ích bao gồm:

* *Giữ một mức độ hoạt động vật lý hàng ngày tối thiểu*: Theo khuyến nghị của Tổ chức Y tế Thế giới (WHO), người lớn nên vận động ít nhất 75 - 150 phút mỗi tuần. Thiết bị đếm bước chân có thể giúp người dùng theo dõi số bước đi hàng ngày, giúp hộ đạt được mục tiêu tối thiểu này.
* *Kiểm soát lượng calo tiêu thụ:* Thiết bị đếm bước chân hoặc ứng dụng đi kèm có thể tính toán số calo đốt cháy dựa trên số bước đi của người dùng và mức độ hoạt động. Việc này cho phép người dùng kiểm soát được khẩu phần dinh dưỡng và lượng calo tiêu thụ hàng ngày.
* *Cải thiện sức khỏe toàn diện:* Việc tập thể dục định kỳ không chỉ giúp giảm cân và duy trì thể lực mà còn cải thiện sức khỏe tâm lý và giảm nguy cơ mắc bệnh tim mạch, bệnh tiểu đường và bệnh mỡ máu cao.
* *Theo dõi tiến trình tập luyện:* Thiết bị đếm bước chân hoặc ứng dụng có thể lưu lại lịch sử hoạt động của người dùng. Việc này giúp người dùng theo dõi tiến trình tập luyện và đưa ra các phương tiện điều chỉnh hoạt động thích hợp.

Vì vậy, việc sử dụng thiết bị đếm bước chân hay các ứng dụng đi kèm là rất hữu ích để giúp người dùng duy trì một lối sống lành mạnh và nâng cao sức khỏe toàn diện.

1. **SƠ ĐỒ KHỐI**
2. Sơ đồ khối:

OLED LCD SSD1306

**VI ĐIỀU KHIỂN (STM8S003F3P6)**

KHỐI NGUỒN

NÚT NHẤN VÀ LED TRẠNG THÁI

CẢM BIẾN GIA TỐC MPU6050

1. Chức năng các khối:

* Khối cảm biến gia tốc: ở đây nhóm em sử dụng cảm biến gia tốc MPU6050 để xác định gia tốc hướng di chuyển theo trục X và Y.
* Khối nút nhấn và LED trạng thái: nút nhấn cho phép chọn chế độ đếm bước chân hay dừng đếm và lưu kết quả, LED sáng khi mạch ở trạng thái đếm bước chân và tắt khi thoát chế độ đếm bước chân.
* Khối OLED LCD SSD1306: hiển thị số bước chân và chức năng hiện tại của mạch.
* Khối vi điều khiển: nhận tín hiệu vào từ cảm biến gia tốc, nút nhấn và xử lý để xuất kết quả lên OLED LCD.
* Khối nguồn: gồm pin Lipo 3.7V qua mạch Boost Converter DC-DC nâng điện áp lên 5V cung cấp cho toàn mạch hoạt động.

1. **Linh kiện sử dụng cho các khối:**
   1. **Khối vi điều khiển:** 
      1. **Sử dụng vi điều khiển STM8S003F3P6 của hãng ST**

* Bộ nhớ Flash 8KB, 1KB SRAM, 128 Byte EEPROM, có 3 bộ Timer là Timer 1, 2, 4. Trong đó Timer 1 và 2 là thanh ghi 16-bit, timer 4 là thanh ghi 8-bit
* Hỗ trợ giao tiếp UART, SPI, I2C, CAN
* Hỗ trợ 5 kênh ADC 10-bit
* Điện áp hoạt động 2.95V đến 5.5V
* Thạch anh nội 16Mhz

Quan trọng là hỗ trợ giao tiếp I2C để dễ dàng tiếp cận đến module MPU6050 và OLED LCD SSD1306

* + 1. **Lý do sử dụng STM8**
* Giá rẻ, có các chức năng cơ bản từ ADC, UART, I2C, SPI, …
* Hãng ST Microelectronics hỗ trợ khá tốt các thư viện lập trình, tool, có thể lập trình theo thư viện hay theo thanh ghi.
* Trình biên dịch (Compile) và phần mềm nạp mã nguồn mở.
* Vẫn là họ 8 bit hướng tiếp cận tương tự như PIC hay AVR nên dễ dàng hơn, nếu sử dụng thư viện Standard(Std) sau này có lợi cho việc tiếp cận lên họ 32-bit ARM STM32.

Sử dụng Module STM8S003F3P6 để dễ dàng sử dụng

Vì chip đóng gói dạng SSOP-20 nên nhóm em thiết kế riêng module để dễ dàng cắm lên Testboard thử nghiệm. Dưới đây là sơ đồ mạch vẽ trên EasyEDA và mạch sau khi hàn lên PCB

A diagram of a computer

Description automatically generated

Sơ đồ mạch module STM8S003F3P6

A close-up of a circuit board

Description automatically generatedA blue circuit board with gold and silver colored components

Description automatically generated with medium confidence

Module sau khi hàn trên PCB

* + 1. **Trình biên dịch, mạch nạp và phần mềm nạp cho STM8S003F3P6**

Để lập trình cho họ STM8 chúng ta có nhiều lựa chọn, một trong số đó kể đến:

+Trình biên dịch Comic C có hỗ trợ phần mềm IDE là ST Visual Develop (STVD) và phần mềm nạp ST Visual Programmer. [1]

+Trình biên dịch SDCC (Small Device C Compiler) là một trình biên dịch tiêu chuẩn C miễn phí có thể sử dụng để biên dịch mã cho nhiều loại vi điều khiển thuộc họ 8051, PIC và có cả STM8, không đi kèm phần mềm IDE và phần mềm nạp.

+Trình biên dịch IAR Embedded Workbench for STM8 có hỗ trợ IDE và nạp trực tiếp trên IDE. Tuy nhiên IAR là phần mềm tính phí chỉ có thể sử dụng bản dùng thử trong 14 ngày sau đó cần mua LisenceKey để tiếp tục sử dụng.

A hand holding a small blue and green device

Description automatically generatedThế nên trong dự án này nhóm em sử dụng trình biên dịch SDCC, sử dụng mạch nạp và phần mềm nạp của một Admin hỗ trợ trên FanPage Lập trình vi điều khiển để nạp chương trình cho STM8S003F3P6:

Mạch nạp

A pink usb flash drive with black and pink wires

Description automatically generated with medium confidenceNgoài ra, nếu muốn sử dụng mạch nạp và phần mềm đến từ hãng ST là ST Visual Develop (STVD) và phần mềm nạp ST Visual Programmer thì chúng ta sử dụng mạch nạp ST-Link V2, ngoài nạp cho STM8 ra còn hỗ trợ nạp và debug cho STM32.

Mạch nạp ST-Link V2

* 1. **Cảm biến gia tốc MPU6050/ GY-521:**

Là một cảm biến được sử dụng để đo 6 thông số: 3 trục góc quay (Gyro), 3 trục gia tốc hướng ( Accelerometer), ngoài ra cảm biến này còn có thể đo nhiệt độ. Là loại cảm biến gia tốc phổ biến nhất trên thị trường hiện nay.

* Phạm vi đo tốc độ góc là ±250, ±500, ±1000, and ±2000°/s
* Phạm vi đo gia tốc góc là ±2g, ±4g, ±8g and ±16g
* Điện áp: 3-5V

MPU 65050/GY-521 dùng để đo gia tốc góc, vận tốc góc, góc quay với độ chính xác cao, đo gia tốc, vận tốc chuyển động tịnh tiến.

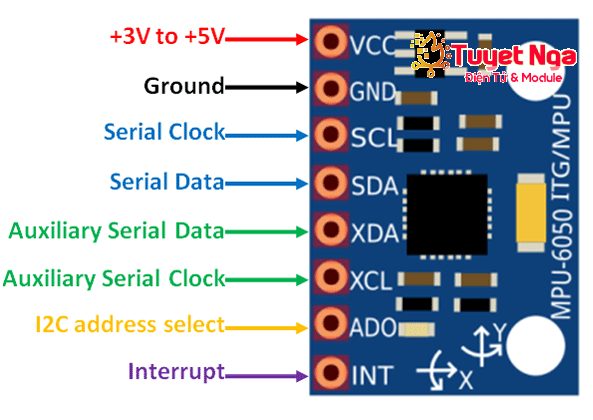
Module cảm biến MPU6050 có thể kết nối với vi điều khiển qua 1 trong 2 giao thức là SPI hoặc I2C. Trong dự án này nhóm em sử dụng I2C để giao tiếp với module.

Các cảm biến bên trong MPU-6050 sử dụng bộ chuyển đổi tương tự – số (Anolog to Digital Converter – ADC) 16-bit cho ra kết quả chi tiết về góc quay, tọa độ… Với 16-bit bạn sẽ có 2^16 = 65536 giá trị cho 1 cảm biến.

Lý do chọn Module MPU6050 vì giao tiếp đơn giản (I2C) có bộ chuyển đổi tương tự - số (ADC) và thời gian chuyển đổi cấu hình được.

* + 1. **Sơ đồ chân và schematic của module MPU6050:**

A blue circuit board with black and silver pins

Description automatically generated

Chi tiết các chân trên Module MPU6050

Ảnh có chứa biểu đồ, sơ đồ

Mô tả được tạo tự động

Sơ đồ mạch chi tiết Module MPU6050

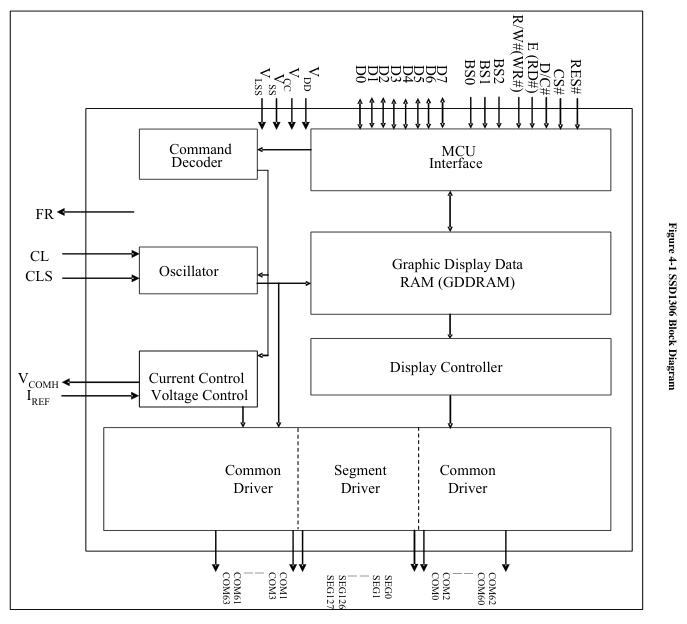
* 1. **OLED LCD SSD1306 0.91 inch**

SSD1306 là chip điều khiển họ CMOS với bộ điều khiển dành cho hiển thị ma trận điểm, với LCD 0.91 inch chip sẽ điều khiển 128 đoạn và 64 điểm chung. Chip này được thiết kế cho tấm nền OLED loại Cathode chung.

SSD1306 được tích hợp nhièu thứ như RAM, bộ điều khiển và bộ dao động, giúp giảm số lượng linh kiện bên ngoài và mức tiêu thụ điện năng.

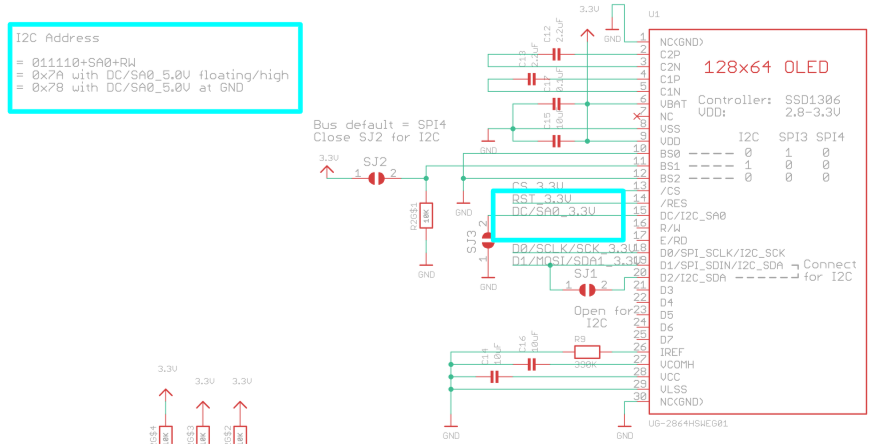
Cung cấp giao tiếp I2C hoặc giao tiếp nối tiếp, phù hợp cho nhiều ứng dụng di động nhỏ gọn, chẳng hạn như màn hình phụ của điện thoại di động, máy nghe nhạc MP3 và máy tính, v.v.

* Kích thước hiển thị: 29 x 11.2mm
* Độ phân giải: 128 x 32 dot matrix panel
* Segment maximum source current: 100uA
* Common maximum sink current: 15mA

Sơ đồ khối bên trong chip SSD1306:

**A close-up of a blue rectangular object

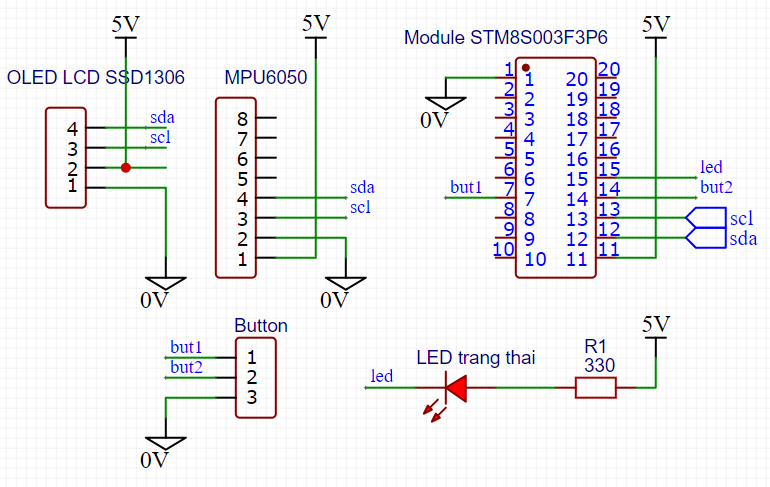
Description automatically generated**

A text on a white background

Description automatically generatedĐể cấu hình cho OLED LCD và hiển thị dữ liệu thì chúng ta cần chú ý đến các thanh ghi và địa chỉ I2C của chip SSD1306. Theo như datasheet thì địa chỉ của màn hình OLED SSD1306 loại 0.91 inch là 0x78 (trong schematic chân SA0 được nối mass)

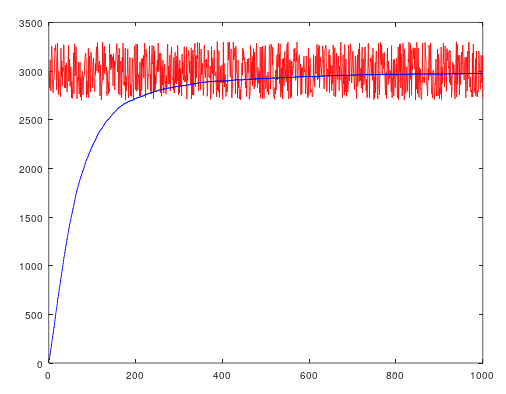
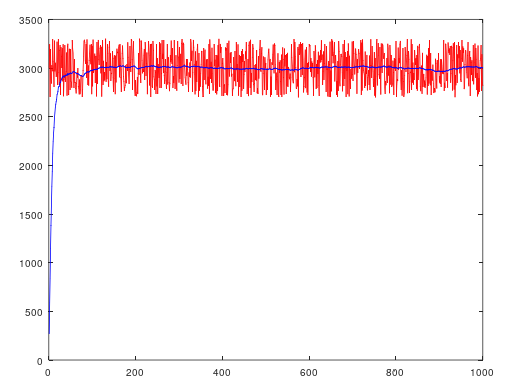
A diagram of a diagram

Description automatically generated

Sơ đồ mạch:

Đọc dữ liệu từ MPU6050 đó là dữ liệu thô, dễ bị nhiễu vì vậy cần thông qua lọc để có thể tính toán giá trị.

Nói sơ qua về việc đọc dữ liệu từ MPU6050, sử dụng giao tiếp I2C để ghi và đọc dữ liệu cho cảm biến gia tốc. Đầu tiên để sử dụng

Bộ lọc Kalman:

Tang measure\_noise = 100

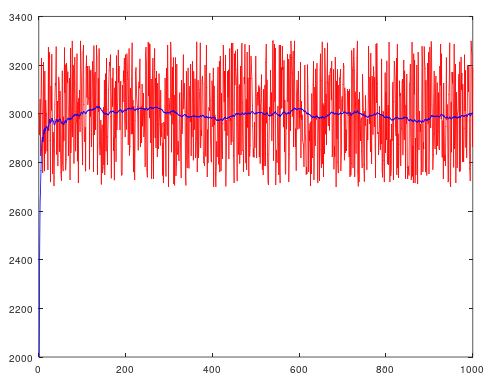
\_err\_measure = 10; #\_r , nhieu do dac, neu nhieu do dac lon thi dau ra bi delay

# va xa gia tri dung

\_err\_estimate(1:loop) = 1; #\_p, day la nhieu uoc luong, neu nhieu uoc luong lon thi do gon song

# cua gia tri uoc luong tang

\_q = 0.001;



Tang process\_noise = 20

* Lưu đồ giải thuật

BẮT ĐẦU

Set output, input cho các chân GPIOA, GPIOB, GPIOC

Khởi tạo biến dem, full

Khởi tạo giao tiếp I2C, khởi tạo OLED LCD

Hiển thị lời chào trên màn hình OLED LCD, khởi tạo module MPU6050

Giao tiếp I2C lấy dữ liệu gia tốc trục X, Y

S

Giá trị mới > Giá trị cũ

Đ

Cộng biến Steps lên 1

Delay 400ms

Tính tổng bình phương dữ liệu thu được

Lấy 20 mẫu gia tốc trục X, Y. Kèm theo lọc giá trị bằng bộ lọc Kalman

* Code

Chi tiết về SDCC vì không sử dụng IDE nên để biên dịch chương trình cần thao tác qua lệnh, mở cmd trên window và trỏ đến thư mục chứa project và thao tác lệnh để dịch chương trình

Ví dụ một số lệnh như sau :

sdcc main.c

Câu lệnh trên sẽ dịch file main.c tạo ra mã assembly, dịch tiếp mã assembly tạo ra nhiều file trong đó file main.ihx là file định dạng hex để nạp vào vi điều khiển tuy nhiên mình sẽ không sử dụng file này để nạp vì SDCC tạo ra file ihx có thể chứa các dòng với độ dài và địa chỉ không được sắp xếp đúng nên sử dụng một lệnh nữa để chuyển sang file hex

packihx main.ihx > main.hex

Với project của mình thì không chỉ có một file main.c mà gồm nhiều file thư viện khác nữa. SDCC chỉ có thể biên dịch 1 file tại 1 thời điểm nên để biên dịch tất cả các file thì phải sử dụng đến makefile, nó là một công cụ giúp mình thực thi nhiều câu lệnh làm mình tối ưu được phần biên dịch code.

#include "stm8s.h"

#include "stdio.h"

#include "stdint.h"

#include "lib.h"

#include "math.h"

#define LED\_PORT    GPIOD

#define LED\_PIN     (1<<3)

#define BUT\_PORT    GPIOD

#define BUT\_PIN     (1<<4)

// uint32\_t clk=0;

float AccX, AccY, AccZ;

float AccX\_Kalman, AccY\_Kalman, AccZ\_Kalman;

int16\_t AccXLSB, AccXMSB, AccYLSB, AccYMSB, AccZLSB, AccZMSB;

float AngleRoll, AnglePitch;

float Sensor;

float Sensor1;

float Sensor2;

float Sensor3;

/\*==========================================================

-----------------OLED LCD SSD1306------------

=============================================================\*/

#define OLED\_I2C\_ADDR   0x78

#define OLED\_CMD\_MODE   0x00    // set command mode

#define OLED\_DAT\_MODE   0x40    // set data mode

const uint8\_t OLED\_INIT\_CMD[] = {

    0xA8, 0x1F,       // set multiplex (HEIGHT-1): 0x1F for 128x32, 0x3F for 128x64

    0x22, 0x00, 0x03, // set min and max page

    0x20, 0x00,       // set horizontal memory addressing mode

    0xDA, 0x02,       // set COM pins hardware configuration to sequential

    0x8D, 0x14,       // enable charge pump

    0xAF,             // switch on OLED

    0xA1, 0xC8        // flip the screen

};

const uint8\_t OLED\_FONT[]= {

    0x00, 0x00, 0x00, 0x00, 0x00, //   0

    0x00, 0x00, 0x2f, 0x00, 0x00, // ! 1

    0x00, 0x07, 0x00, 0x07, 0x00, // " 2

    0x14, 0x7f, 0x14, 0x7f, 0x14, // # 3

    0x24, 0x2a, 0x7f, 0x2a, 0x12, // $ 4

    0x62, 0x64, 0x08, 0x13, 0x23, // % 5

    0x36, 0x49, 0x55, 0x22, 0x50, // & 6

    0x00, 0x05, 0x03, 0x00, 0x00, // ' 7

    0x00, 0x1c, 0x22, 0x41, 0x00, // ( 8

    0x00, 0x41, 0x22, 0x1c, 0x00, // ) 9

    0x14, 0x08, 0x3E, 0x08, 0x14, // \* 10

    0x08, 0x08, 0x3E, 0x08, 0x08, // + 11

    0x00, 0x00, 0xA0, 0x60, 0x00, // , 12

    0x08, 0x08, 0x08, 0x08, 0x08, // - 13

    0x00, 0x60, 0x60, 0x00, 0x00, // . 14

    0x20, 0x10, 0x08, 0x04, 0x02, // / 15

    0x3E, 0x51, 0x49, 0x45, 0x3E, // 0 16

    0x00, 0x42, 0x7F, 0x40, 0x00, // 1 17

    0x42, 0x61, 0x51, 0x49, 0x46, // 2 18

    0x21, 0x41, 0x45, 0x4B, 0x31, // 3 19

    0x18, 0x14, 0x12, 0x7F, 0x10, // 4 20

    0x27, 0x45, 0x45, 0x45, 0x39, // 5 21

    0x3C, 0x4A, 0x49, 0x49, 0x30, // 6 22

    0x01, 0x71, 0x09, 0x05, 0x03, // 7 23

    0x36, 0x49, 0x49, 0x49, 0x36, // 8 24

    0x06, 0x49, 0x49, 0x29, 0x1E, // 9 25

    0x00, 0x36, 0x36, 0x00, 0x00, // : 26

    0x00, 0x56, 0x36, 0x00, 0x00, // ; 27

    0x08, 0x14, 0x22, 0x41, 0x00, // < 28

    0x14, 0x14, 0x14, 0x14, 0x14, // = 29

    0x00, 0x41, 0x22, 0x14, 0x08, // > 30

    0x02, 0x01, 0x51, 0x09, 0x06, // ? 31

    0x32, 0x49, 0x59, 0x51, 0x3E, // @ 32

    0x7C, 0x12, 0x11, 0x12, 0x7C, // A 33

    0x7F, 0x49, 0x49, 0x49, 0x36, // B 34

    0x3E, 0x41, 0x41, 0x41, 0x22, // C 35

    0x7F, 0x41, 0x41, 0x22, 0x1C, // D 36

    0x7F, 0x49, 0x49, 0x49, 0x41, // E 37

    0x7F, 0x09, 0x09, 0x09, 0x01, // F 38

    0x3E, 0x41, 0x49, 0x49, 0x7A, // G 39

    0x7F, 0x08, 0x08, 0x08, 0x7F, // H 40

    0x00, 0x41, 0x7F, 0x41, 0x00, // I 41

    0x20, 0x40, 0x41, 0x3F, 0x01, // J 42

    0x7F, 0x08, 0x14, 0x22, 0x41, // K 43

    0x7F, 0x40, 0x40, 0x40, 0x40, // L 44

    0x7F, 0x02, 0x0C, 0x02, 0x7F, // M 45

    0x7F, 0x04, 0x08, 0x10, 0x7F, // N 46

    0x3E, 0x41, 0x41, 0x41, 0x3E, // O 47

    0x7F, 0x09, 0x09, 0x09, 0x06, // P 48

    0x3E, 0x41, 0x51, 0x21, 0x5E, // Q 49

    0x7F, 0x09, 0x19, 0x29, 0x46, // R 50

    0x46, 0x49, 0x49, 0x49, 0x31, // S 51

    0x01, 0x01, 0x7F, 0x01, 0x01, // T 52

    0x3F, 0x40, 0x40, 0x40, 0x3F, // U 53

    0x1F, 0x20, 0x40, 0x20, 0x1F, // V 54

    0x3F, 0x40, 0x38, 0x40, 0x3F, // W 55

    0x63, 0x14, 0x08, 0x14, 0x63, // X 56

    0x07, 0x08, 0x70, 0x08, 0x07, // Y 57

    0x61, 0x51, 0x49, 0x45, 0x43, // Z 58

    0x00, 0x7F, 0x41, 0x41, 0x00, // [ 59

    0x02, 0x04, 0x08, 0x10, 0x20, // \ 60

    0x00, 0x41, 0x41, 0x7F, 0x00, // ] 61

    0x04, 0x02, 0x01, 0x02, 0x04, // ^ 62

    0x40, 0x40, 0x40, 0x40, 0x40,  // \_ 63

    0x00, 0x60, 0x60, 0x00, 0x00 // . 64

};

void OLED\_init(void);

void OLED\_printP(const char \*p);

void OLED\_printNum(uint16\_t num);

void OLED\_printOneNumber(uint8\_t num);

void OLED\_printNumber(float ch);

void OLED\_printFloat(float num);

void OLED\_cursor(uint8\_t xpos, uint8\_t ypos);

void OLED\_clear(void);

void initMPU6050(void);

void getMPU6050(void);

float \_err\_measure;

float \_err\_estimate;

float \_q;

float \_current\_estimate = 0;

float \_last\_estimate = 0;

float \_kalman\_gain = 0;

void SimpleKalmanFilter(float mea\_e, float est\_e, float q)

{

  \_err\_measure=mea\_e;

  \_err\_estimate=est\_e;

  \_q = q;

}

/\*loc 1\*/

float updateEstimate(float mea)

{

  \_kalman\_gain = \_err\_estimate/(\_err\_estimate + \_err\_measure);

  \_current\_estimate = \_last\_estimate + \_kalman\_gain \* (mea - \_last\_estimate);

  \_err\_estimate =  (1.0 - \_kalman\_gain)\*\_err\_estimate + fabsf(\_last\_estimate-\_current\_estimate)\*\_q;

  \_last\_estimate=\_current\_estimate;

  return \_current\_estimate;

}

/\*loc 2\*/

float kalman\_single(float z, float measure\_noise, float process\_noise){

    const float R = measure\_noise \* measure\_noise;

    const float Q = process\_noise \* process\_noise;

    static float x\_hat, P;

    float P\_, K;

    /\*\*\*\*\*\*\*\*\*\*\*\*Noi suy Kalman\*\*\*\*\*\*\*\*8\*/

        P\_ = P + Q;         // P\_ = A\*P\*A' + Q;

        K = P\_ / (P\_ + R);  // K = P\_ \* H' \* inv(H \* P\_ \* H' + R)

        x\_hat = x\_hat + K \* (z - x\_hat);    //x\_hat = x\_hat + K \* (z - H \* x\_hat)

        P = (1 - K) \* P\_;                   //P = (1 - K \* H) \* P\_

        return x\_hat;

}

/\*==========================================================

-----------------MAIN FUNCTION------------

=============================================================\*/

float vectorprevious;

float vector;

float totalvector;

int Steps = 0;

const char Message1[] = "DO AN";

const char Message2[] = "SO BUOC=";

float tong;

// float zaccl;

float threshold = 6.0;

uint8\_t flag = 0;

void main (void)

{

    /\*Using HSI clock 16Mhz\*/

    CLK->CKDIVR &= (uint8\_t)(~CLK\_CKDIVR\_HSIDIV);

    CLK->CKDIVR |= (uint8\_t)0x00;

    GPIOB->ODR = GPIO\_ODR\_RESET\_VALUE; /\* Reset Output Data Register \*/

    GPIOB->DDR = GPIO\_DDR\_RESET\_VALUE; /\* Reset Data Direction Register \*/

    GPIOB->CR1 = GPIO\_CR1\_RESET\_VALUE; /\* Reset Control Register 1 \*/

    GPIOB->CR2 = GPIO\_CR2\_RESET\_VALUE; /\* Reset Control Register 2 \*/

    GPIOA->ODR = GPIO\_ODR\_RESET\_VALUE; /\* Reset Output Data Register \*/

    GPIOA->DDR = GPIO\_DDR\_RESET\_VALUE; /\* Reset Data Direction Register \*/

    GPIOA->CR1 = GPIO\_CR1\_RESET\_VALUE; /\* Reset Control Register 1 \*/

    GPIOA->CR2 = GPIO\_CR2\_RESET\_VALUE; /\* Reset Control Register 2 \*/

    GPIOC->ODR = GPIO\_ODR\_RESET\_VALUE; /\* Reset Output Data Register \*/

    GPIOC->DDR = GPIO\_DDR\_RESET\_VALUE; /\* Reset Data Direction Register \*/

    GPIOC->CR1 = GPIO\_CR1\_RESET\_VALUE; /\* Reset Control Register 1 \*/

    GPIOC->CR2 = GPIO\_CR2\_RESET\_VALUE; /\* Reset Control Register 2 \*/

    /\*Pin B4 SCL\*/

    GPIOB->DDR |= (uint8\_t)(1<<4);

    GPIOB->ODR |= (uint8\_t)(1<<4);

    GPIOB->CR1 &= ~(uint8\_t)(1<<4);     /\*OPEN DRAIN\*/

    GPIOB->CR2 &= ~(uint8\_t)(1<<4);     /\*Output Speed Up to 2Mhz\*/

    /\*Pin B5 SDA\*/

    GPIOB->DDR |= (uint8\_t)(1<<5);

    GPIOB->ODR |= (uint8\_t)(1<<5);

    GPIOB->CR1 &= ~(uint8\_t)(1<<5);     /\*OPEN DRAIN\*/

    GPIOB->CR2 &= ~(uint8\_t)(1<<5);     /\*Output Speed Up to 2Mhz\*/

    /\*Pin A3 button\*/

    GPIOA->DDR &= ~(uint8\_t)(1<<3);

    GPIOA->IDR |= (uint8\_t)(1<<3);

    GPIOA->CR1 |= (uint8\_t)(1<<3);      /\*PULL UP\*/

    GPIOA->CR2 &= ~(uint8\_t)(1<<3);     /\*External interrupt disabled\*/

    /\*Pin C3 button\*/

    GPIOC->DDR &= ~(uint8\_t)(1<<3);

    GPIOC->IDR |= (uint8\_t)(1<<3);

    GPIOC->CR1 |= (uint8\_t)(1<<3);      /\*PULL UP\*/

    GPIOC->CR2 &= ~(uint8\_t)(1<<3);     /\*External interrupt disabled\*/

    /\*Pin C4 LED\*/

    GPIOC->DDR |= (uint8\_t)(1<<4);

    GPIOC->ODR |= (uint8\_t)(1<<4);

    GPIOC->CR1 |= (uint8\_t)(1<<4);      /\*PULL PUSH\*/

    GPIOC->CR2 &= ~(uint8\_t)(1<<4);     /\*Output Speed Up to 2Mhz\*/

    Timer4Config(); /\*cho hien thi man hinh OLED truoc, neu khoi tao

    cam bien truoc thi man hinh oled se khong sang tu do nguoi dung

    hieu lam la stm8 dang khong chay\*/

    I2C\_init();

    OLED\_init();

    OLED\_clear();

    OLED\_cursor(0,1);

    OLED\_printP("DO AN MON DO NANG CAO");

    Timer4DelayMs(1000);

    /\*hieu ung quet man hinh\*/

    OLED\_cursor(0,0);

    I2C\_start();

    I2C\_address(OLED\_I2C\_ADDR, I2C\_DIRECTION\_TX);

    I2C\_write(OLED\_DAT\_MODE);

    for (uint16\_t i=0;i<512;i++)

    {

        I2C\_write(0xff);

        Timer4DelayMs(2);

    }

    I2C\_stop();

    OLED\_clear();

    uart\_init();

    initMPU6050();

    // for(uint8\_t i=0; i<100;i++){

    //  getMPU6050();

    //  xval[i] = (float) AccXLSB/4096.0 - 0.04;

    //  yval[i] = (float) AccYLSB/4096.0 + 0.01;

    //  zval[i] = (float) AccZLSB/4096.0 - 0.02;

    //  sum = xval[i]+sum;

    //  sum1 = yval[i]+sum1;

    //  sum2 = zval[i]+sum2;

    //  Timer4DelayMs(10);

    // }

    // xavg = sum/100.0;

    // yavg = sum1/100.0;

    // zavg = sum2/100.0;

    // uart1\_number((int)(xavg\*100)%100);

    // uart1\_number((int)(yavg\*100)%100);

    // uart1\_number((int)(zavg\*100)%100);

    // uart1\_number((int)100);

    Timer4DelayMs(10);

    // SimpleKalmanFilter(2, 2, 0.05);

    while (1)

    {

        if ( (GPIOA->IDR & (uint8\_t)(1<<3)) == RESET ){

            OLED\_cursor(0,2);

            OLED\_printP("STEP = ");

            GPIOC->ODR &= ~(uint8\_t)(1<<4);

        }

        if ( (GPIOC->IDR & (uint8\_t)(1<<3)) == RESET ){

            OLED\_cursor(0,2);

            OLED\_printP("COUNTER = ");

            GPIOC->ODR |= (uint8\_t)(1<<4);

        }

        // Sensor2 = sqrtf( (AccX\_Kalman\*AccX\_Kalman) + (AccY\_Kalman\*AccY\_Kalman) );

        // Sensor1 = sqrtf( (AccX\*AccX) + (AccY\*AccY) );

        // Sensor3 = sqrtf( (yaccl\*yaccl) + (zaccl\*zaccl) );

        // for (int a = 0; a < 10; a++)

        // {

        //  getMPU6050();

        //  Sensor1 = sqrtf( (AccX\*AccX) + (AccY\*AccY) );

        //  tong = tong + Sensor1;

        // }

        // if (tong > threshold )

        // {

        //  flag = 1;

        // }

        // else {

        //  flag = 0;

        // }

        // tong = 0;

        // Timer4DelayMs(10);

        // for (int a = 0; a < 10; a++)

        // {

        //  getMPU6050();

        //  Sensor1 = sqrtf( (AccX\*AccX) + (AccY\*AccY) );

        //  tong = tong + Sensor1;

        // }

        // if (tong > threshold && flag == 1)

        // {

        //  Steps = Steps;

        //  // flag = 0;

        // }

        // else if (tong < threshold && flag == 1) {

        //  Steps++;

        //  // flag = 0;

        // }

        // tong = 0;

        // if (Steps<0){

        //  Steps = 0;

        // }

        // OLED\_cursor(0,2);

        // OLED\_printP("SO BUOC = ");

        // OLED\_cursor(96,2);

        // OLED\_printNum(Steps);

        // Timer4DelayMs(200);

        // OLED\_clear();

    }

}

// vectorprevious = vector;

// for (int a=0; a<20; a++){

// uart1\_number((int)(tong)%100);

// uart1\_string(",");

// uart1\_number((int)(Sensor2\*100)%100);

// uart1\_nline();

// }

// Sensor = sqrtf( (AccX\*AccX) + (AccY\*AccY) + (AccZ\*AccZ) );

// AngleRoll = atanf(AccY/sqrtf(AccX\*AccX + AccZ\*AccZ))\*1/(3.142/180);

// AnglePitch = atanf(AccX/sqrtf(AccY\*AccY + AccZ\*AccZ))\*1/(3.142/180);

void OLED\_init(void) {

    I2C\_start();

    I2C\_address(OLED\_I2C\_ADDR, I2C\_DIRECTION\_TX);

    I2C\_write(OLED\_CMD\_MODE);

    for (uint8\_t i = 0;i < 12;i++)

    {

        I2C\_write(OLED\_INIT\_CMD[i]);

    }

    I2C\_stop();

}

void OLED\_printP(const char \*p)

{

    I2C\_start();

    I2C\_address(OLED\_I2C\_ADDR,I2C\_DIRECTION\_TX);

    I2C\_write(OLED\_DAT\_MODE);

    while (\*p)

    {

        uint16\_t offset = \*p - 32;

        offset += offset << 2;

        I2C\_write(0x00);

        for(uint8\_t i = 5; i;i--){ I2C\_write(OLED\_FONT[offset++]); }

        p++;

    }

    I2C\_stop();

}

void OLED\_printNum(uint16\_t num){

    uint8\_t mang[] = {0,0,0,0,0};

    uint8\_t count = 0;

    if(num == 0)

    {

        mang[0] = 0;

        count = 1;

    }

    while(num != 0){

        mang[count] = num%10;

        num = num /10;

        ++count;

    }

    I2C\_start();

    I2C\_address(OLED\_I2C\_ADDR,I2C\_DIRECTION\_TX);

    I2C\_write(OLED\_DAT\_MODE);

    while(count != 0){

        uint16\_t offset = mang[count-1] + 16;

        offset += offset << 2;

        I2C\_write(0x00);

        for(uint8\_t i = 5; i;i--){ I2C\_write(OLED\_FONT[offset++]); }

        --count;

    }

    I2C\_stop();

}

void OLED\_printFloat(float num){

    uint16\_t num2 = num \* 100;

    uint8\_t mang[] = {0,0,0,0,0};

    uint8\_t count = 0;

    while(count<5){

        mang[count] = (uint8\_t)num2%10;

        num2 = num2 /10;

        ++count;

        if(count==2) {mang[2]= 48; ++count;}

    }

    I2C\_start();

    I2C\_address(OLED\_I2C\_ADDR,I2C\_DIRECTION\_TX);

    I2C\_write(OLED\_DAT\_MODE);

    while(count != 0){

        uint16\_t offset = mang[count-1] + 16;

        offset += offset << 2;

        I2C\_write(0x00);

        for(uint8\_t i = 5; i;i--){ I2C\_write(OLED\_FONT[offset++]); }

        --count;

    }

    I2C\_stop();

}

void OLED\_printOneNumber(uint8\_t num)

{

    uint16\_t offset = num + 16;

    offset += offset << 2;

    I2C\_start();

    I2C\_address(OLED\_I2C\_ADDR, OLED\_DAT\_MODE);

    I2C\_write(0x00);

    for(uint8\_t i = 5; i;i--){ I2C\_write(OLED\_FONT[offset++]); }

    I2C\_stop();

}

void OLED\_cursor(uint8\_t xpos, uint8\_t ypos)

{

    I2C\_start();

    I2C\_address(OLED\_I2C\_ADDR, I2C\_DIRECTION\_TX);

    I2C\_write(OLED\_CMD\_MODE);

    I2C\_write(xpos & 0x0F);

    I2C\_write(0x10 | (xpos >> 4));

    I2C\_write(0xB0 | (ypos & 0x07));

    I2C\_stop();

}

void OLED\_clear(void)

{

    OLED\_cursor(0,0);

    I2C\_start();

    I2C\_address(OLED\_I2C\_ADDR, I2C\_DIRECTION\_TX);

    I2C\_write(OLED\_DAT\_MODE);

    for (uint16\_t i=0;i<512;i++)

    {

        I2C\_write(0x00);

    }

    I2C\_stop();

}

void initMPU6050(void){

    I2C\_start();

    I2C\_address((uint8\_t)0x68<<1, I2C\_DIRECTION\_TX);

    I2C\_write(0x6B);

    I2C\_write(0x00);

    I2C\_stop();

    Timer4DelayMs(100);

    I2C\_start();

    I2C\_address ((uint8\_t)0x68<<1, I2C\_DIRECTION\_TX);

    I2C\_write(0x1A);

    I2C\_write(0x05);

    I2C\_stop();

    Timer4DelayMs(100);

    I2C\_start();

    I2C\_address ((uint8\_t)0x68<<1, I2C\_DIRECTION\_TX);

    I2C\_write(0x1B);

    I2C\_write(0x00); //250 do/s

    I2C\_stop();

    Timer4DelayMs(100);

    I2C\_start();

    I2C\_address ((uint8\_t)0x68<<1, I2C\_DIRECTION\_TX);

    I2C\_write(0x1C);

    I2C\_write(0x10); //+-8g

    I2C\_stop();

}

void getMPU6050(void){

    uint8\_t I2C\_Slave\_Address = (uint8\_t)0x68<<1;   /\*MPU6050 Address\*/

    I2C\_start();

    I2C\_address(I2C\_Slave\_Address, I2C\_DIRECTION\_TX);

    I2C\_write(0x3B);

    I2C->CR2 |= I2C\_CR2\_START| I2C\_CR2\_ACK; /\* Generate a START condition and Acknowledge Enable \*/

    while((I2C->SR1 & I2C\_SR1\_SB) == 0);    /\*Check --EV5 \*/

    I2C\_Slave\_Address &= (uint8\_t)0xFE;     /\* Clear bit0 (direction) just in case \*/

    I2C->DR = (uint8\_t)(I2C\_Slave\_Address | (uint8\_t)I2C\_DIRECTION\_RX); /\* Send the Address + Direction \*/

    I2C->SR1;   I2C->SR3;

    while((I2C->SR1 & I2C\_SR1\_ADDR)==0);

    while((I2C->SR3 & (uint8\_t)0x02)==SET); /\*check busy flag \*/

    while((I2C->SR1 & I2C\_SR1\_RXNE)==0);

        AccXLSB = (uint8\_t)I2C->DR;

    I2C->CR2 |= (uint8\_t)(I2C\_CR2\_ACK);     /\* Enable the acknowledgement \*/

    while((I2C->SR1 & I2C\_SR1\_RXNE)==0);

        AccXMSB = (uint8\_t)I2C->DR;

    I2C->CR2 |= (uint8\_t)(I2C\_CR2\_ACK);     /\* Enable the acknowledgement \*/

    while((I2C->SR1 & I2C\_SR1\_RXNE)==0);

        AccYLSB = (uint8\_t)I2C->DR;

    I2C->CR2 |= (uint8\_t)(I2C\_CR2\_ACK);     /\* Enable the acknowledgement \*/

    while((I2C->SR1 & I2C\_SR1\_RXNE)==0);

        AccYMSB = (uint8\_t)I2C->DR;

    I2C->CR2 |= (uint8\_t)(I2C\_CR2\_ACK);     /\* Enable the acknowledgement \*/

    while((I2C->SR1 & I2C\_SR1\_RXNE)==0);

        AccZLSB = (uint8\_t)I2C->DR;

    I2C->CR2 |= (uint8\_t)(I2C\_CR2\_ACK);     /\* Enable the acknowledgement \*/

    while((I2C->SR1 & I2C\_SR1\_RXNE)==0);

        AccZMSB |= (uint8\_t)I2C->DR;

    I2C->CR2 &= (uint8\_t)(~I2C\_CR2\_ACK);    /\* Disable the acknowledgement \*/

    I2C\_stop();

    AccXLSB = (AccXLSB << 8)|AccXMSB;

    AccYLSB = (AccYLSB << 8)|AccYMSB;

    // AccZLSB = (AccZLSB << 8)|AccZMSB;

    // AccZLSB -= 4096;

    // AccX\_Kalman = kalman\_single((float)AccXLSB,100,10);

    // AccY\_Kalman = kalman\_single((float)AccYLSB,100,10);

    // AccZ\_Kalman = kalman\_single((float)AccZLSB,100,10);

    AccX = (float) AccXLSB/4096.0 - 0.04;

    AccY = (float) AccYLSB/4096.0 + 0.01;

    // AccZ = (float) AccZLSB/4096.0 - 0.02;

    // AccX\_Kalman = updateEstimate((float)AccX);

    // AccY\_Kalman = updateEstimate((float)AccY);

    // AccZ\_Kalman = updateEstimate((float)AccZ);

    Timer4DelayMs(1);

}

* PCB

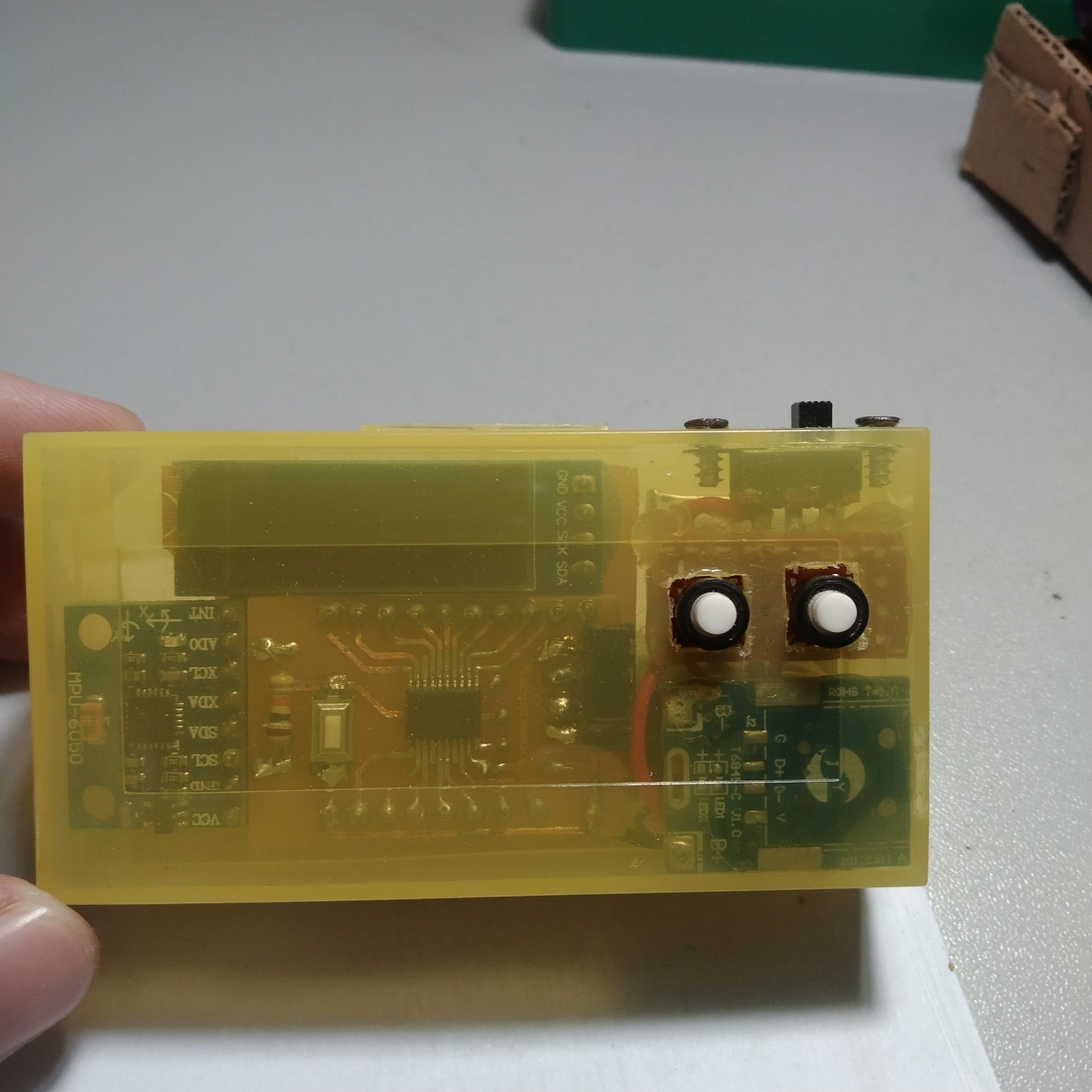
A computer screen shot of a blue and yellow circuit board

Description automatically generated

Demo sản phẩm

A yellow plastic box with a small electronic device

Description automatically generated



Hướng phát triển:

Làm mạch PCB và thiết kế sản phẩm nhỏ gọn hơn

Tích hợp thêm mạch bluetooth HC-05 để truyền dữ liệu lên điện thoại hoặc sử dụng module nRF24L01 như một thiết bị BLE để truyền dữ liệu.

Hoặc mở rộng hơn là sử dụng ESP8266 để thu thập dữ liệu lên WEB để dễ quản lý và truy cập từ xa.

Hạn chế:

Độ chính xác và độ nhạy của cảm biến chưa cao: tùy thuộc vào vận tốc, hướng di chuyển và điều kiện môi trường

Mỹ quang của sản phẩm chưa được đẹp

Kích thước còn khá lớn

Tốc độ đo còn chậm

Tham khảo tại: [STVD-STM8 - ST Visual develop IDE for developing ST7 and STM8 applications - STMicroelectronics](https://www.st.com/en/development-tools/stvd-stm8.html)

Tham khảo tại: [SDCC - Small Device C Compiler (sourceforge.net)](https://sdcc.sourceforge.net/)

Tham khảo tại: [IAR Embedded Workbench for STM8 | IAR](https://www.iar.com/products/architectures/st/iar-embedded-workbench-for-stm8/#containerblock_3932)