LOAD CELL HIEN THI DU LIEU LEN WEB

1. TÌM HIỂU VỀ MẠCH ĐO CÂN NẶNG DỰA TRÊN CẢM BIẾN LOADCELL
2. SƠ ĐỒ KHỐI
3. Sơ đồ khối và chức năng các khối

**NodeMCU ESP8266**

**STM8S003F3P6 MCU**

**KHỐI NGUỒN**

**MẠCH TIỀN KHUẾCH ĐẠI**

**CẢM BIẾN LOADCELL**

Khối cảm biến cân nặng LoadCell: bản chất là cầu Wheastones, sự thay đổi cân nặng dẫn đến sự thay đổi trở kháng ngõ ra của LoadCell. Từ đó nhận biết cân nặng

Mạch tiền khuếch đại: khuếch đại điện áp vi sai từ ngõ ra LoadCell, giúp cho vi điều khiển có thể đọc được điện áp dễ hơn

MCU STM8S003F3P6: nhận điện áp từ mạch tiền khuếch đại và tính toán ra cân nặng và gửi dữ liệu cân nặng qua UART đến ESP8266

NodeMCU ESP8266: nhận dữ liệu qua UART từ MCU truyền đến và tải dữ liệu đó lên trang web qua chế độ **Access Point**

Khối nguồn: cung cấp điện áp cho toàn mạch hoạt động.

1. Linh kiện sử dụng cho các khối
   1. A black text and black text

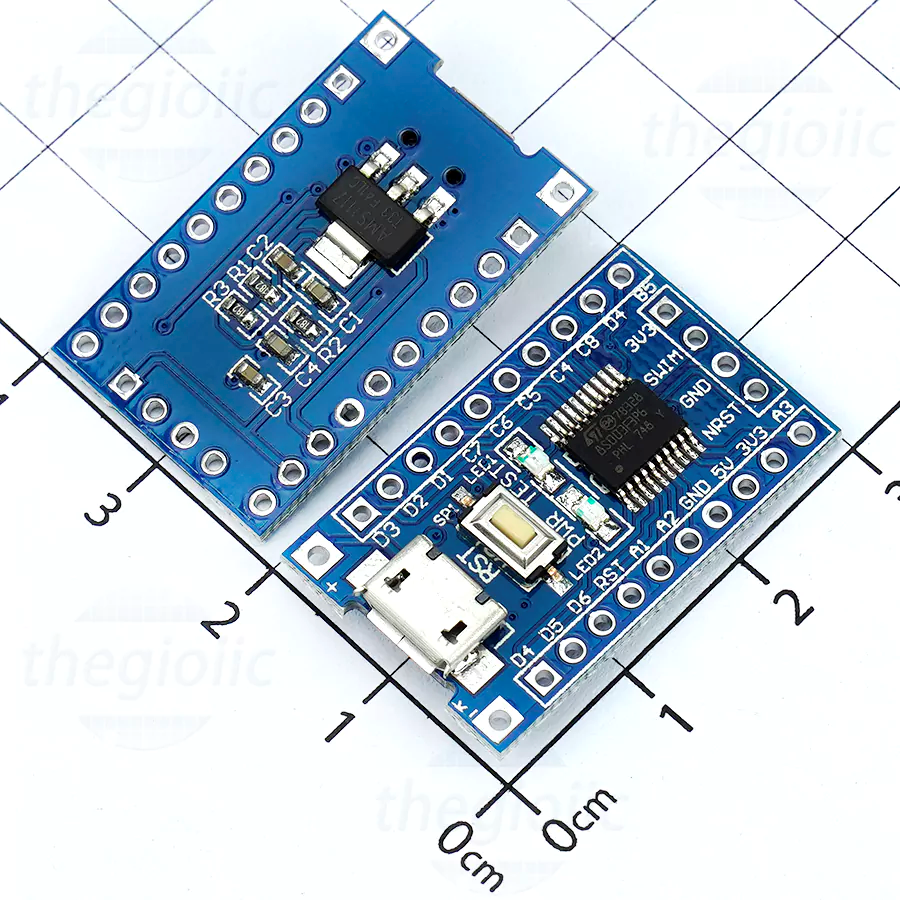
      Description automatically generatedMicro LoadCell 0-20Kg-CZL635
   2. Module STM8S003F3P6

* 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

* **Lý do sử dụng STM8S003F3P6**
* 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

Vì chip đóng gói dạng SSOP-20 nên nhóm em sử dụng Module STM8S003F3P6 để dễ dàng sử dụng

**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.

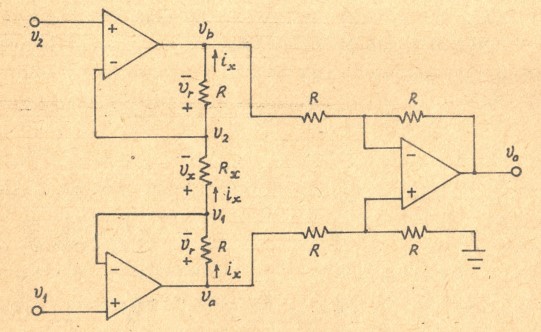
A pink usb flash drive with black and pink wires

Description automatically generated with medium confidenceThế nên trong dự án này nhóm em sử dụng trình biên dịch Comic C, sử dụng mạch nạp ST-Link V2 và phần mềm nạp ST Visual Programmer

Mạch nạp ST-Link V2

1. THIẾT KẾ VÀ SƠ ĐỒ MẠCH
2. Thiết kế mạch tiền khuếch đại

* Sử dụng mạch khuếch đại vi sai dùng opamp LM324N

Mạch khuếch đại vi sai tổng quát:

A diagram of a circuit

Description automatically generatedTriển khai trên LM324

A math equation with a plus and rx

Description automatically generatedNhư vậy giá trị Từ đó tính được độ lợi mạch thông qua công thức sau:

Theo đó tín hiệu đi vào mạch khuếch đại vi sai sẽ được so sánh và khuếch đại lên 101 lần, với mạch trên thì là , cụm điện trở R là trở từ

Sau mạch khuếch đại vi sai, em thiết kế thêm mạch khuếch đại không đảo để bộ ADC của vi điều khiển dễ xử lý. Độ lợi mạch KĐ không đảo là 10, trở quyết định độ lợi. Điều chỉnh để có độ là 10.

1. A diagram of a circuit

   Description automatically generatedSơ đồ toàn mạch

Nguồn 5V được lấy từ cổng microUSB của ESP8266 cấp cho toàn mạch hoạt động, MCU STM8S003F3P6 nhận tín hiệu ADC từ mạch khuếch đại thông qua chân PD3, chân PD3 là chân ADC, giữa MCU và ESP8266 được giao tiếp qua giao thức UART, chân PD5 của MCU là chân TX, PD6 là chân RX. Ngoài ra có một nút nhấn nối đến chân PD2 có chức năng cân bằng giá trị cân nặng về 0

1. LƯU ĐỒ GIẢI THUẬT VÀ CODE TRÊN PHẦN MỀM ST VISUAL DEVELOP
2. Lưu đồ giải thuật

Lấy 20 mẫu ADC sau đó tính trung bình và lưu vào biến **ADC\_Offset** để làm giá trị cân bằng

Lấy tiếp 1 mẫu ADC và lưu vào biến **ADC\_current**

Khởi tạo kênh ADC1, UART1, Timer4 và set chân PD2 làm GPIO INPUT

Gửi một chuỗi chứa dữ liệu về số đo cân nặng đã tính được qua UART, chuỗi bắt đầu bằng ký tự ‘S’ sau đó là số cân nặng

Bắt đầu

Kiểm tra trạng thái chân PD2

Lấy 20 mẫu ADC, để tính lại giá trị cân bằng và lưu vào biến **ADC\_Offset**

LOW

HIGHHIGH

Tính toán cân nặng dựa trên 2 dữ liệu là **ADC\_Offset** và **ADC\_current**

1. Code trên phần mềm ST Visual Develop

Đầu tiên là phần khai báo thư viện ở trong file main.h, đặt macro cho chân PD2 là BUT\_PORT và BUT\_PIN để tiện cấu hình. Theo sau là một số biến phục vụ cho việc tính toán, biến adc\_value để lưu giá trị ADC đọc được, biến này 16-bits do ADC của STM8S003F3P6 là 10-bits. Các biến còn lại để lưu giá trị cân nặng

#include "main.h"

//===========================khai bao cac chan vao ra======================//

#define BUT\_PORT    GPIOD

#define BUT\_PIN     GPIO\_PIN\_2//khai bao BUTTON o chan PD2

//============================khai bao bien va hang==========================//

uint16\_t adc\_value = 0;

float ADC\_Offset;

float ADC\_current;

float Weight;

unsigned char i;

//============================khai bao ham==================================//

void Clock\_setup(void); //thiet lap xung clk 16Mhz cho MCU

void GPIO\_setup(void);  //thiet lap GPIO

void uart\_init(void);       //thiet lap UART1

void uart1\_number(int number);

void uart1\_string(const char \*cy);

void uart1\_nline(void);

Nội dung trong file main.h, mục đích khai báo các thư viện cần thiết như delay, hàm cho vi điều khiển stm8s.h, và hàm phục vụ ngắt stm8s\_it.h

#ifndef \_\_MAIN\_H

#define \_\_MAIN\_H

#include <stdlib.h>

#include "stm8s.h"

#include "stm8s\_delay.h"

#include "stm8s\_it.h"

#include "stm8s\_exti.h"

#endif /\* \_\_MAIN\_H\*/

Đi sâu vào hàm main trong chương trình, nơi sẽ khai báo các tính năng phục vụ xuyên suốt chương trình chính. Đầu tiên là cấu hình xung clock cho MCU hoạt động, ở đây sử dụng thạch anh nội 16MHz.

Tiếp theo sẽ clear các PORT cùa MCU về giá trị ban đầu, sau đó set cho chân PD2 làm chế độ INPUT. Hàm **delay\_config()** là hàm sử dụng TIMER4 của STM8 để tạo độ trễ nhằm phục vụ một số mục đích liên quan đến thời gian

Sau đó sẽ là hàm khởi tạo UART1 và khởi tạo ADC, hàm bật ADC1 của MCU hoạt động là hàm **ADC1\_Cmd(ENABLE)**

//============================HAM MAIN=======================================//

int main()

{

    Clock\_setup();//goi lai ham

    GPIO\_setup();//goi lai ham

    delay\_config();

    uart\_init();

    ADC1\_DeInit();

    ADC1\_Init(ADC1\_CONVERSIONMODE\_SINGLE, ADC1\_CHANNEL\_4, ADC1\_PRESSEL\_FCPU\_D18,

        ADC1\_EXTTRIG\_TIM, DISABLE, ADC1\_ALIGN\_RIGHT, ADC1\_SCHMITTTRIG\_CHANNEL4, DISABLE);

    ADC1\_Cmd(ENABLE);

Sau khi khởi tạo ADC, lấy 20 mẫu ADC tại chân PD3 để tìm giá trị cân bằng khi loadCell chưa có vật nặng, được lưu lại ở biến **ADC\_Offect** sẽ được sử dụng để tính toán khối lượng sau này

    for (i=0; i<20; ++i)

    {

        ADC1\_StartConversion();

        while(ADC1\_GetFlagStatus(ADC1\_FLAG\_EOC)==0){}

        ADC1\_ClearFlag(ADC1\_FLAG\_EOC);

        adc\_value = ADC1\_GetConversionValue();

        ADC\_Offset += adc\_value;

    }

    ADC\_Offset = ADC\_Offset / 20.0;

Trong hàm lặp while(1), chúng ta sẽ kiểm tra chân PD2 là chân được nối với nút nhấn xem có được nhấn không, nếu được nhấn thì trạng thái chân sẽ ở mức thấp. Nếu chân PD2 ở mức thấp thì sẽ thực hiện lấy 20 mẫu ADC và cập nhật lại biến **ADC\_Offset**.

while (1)

    {

        if(GPIO\_ReadInputPin(BUT\_PORT, BUT\_PIN) == 0)

        {

            delay\_ms(200);

            while(GPIO\_ReadInputPin(BUT\_PORT, BUT\_PIN) == 0){}

            ADC\_Offset = 0;

            for (i=0; i<20; ++i)

            {

                ADC1\_StartConversion();

                while(ADC1\_GetFlagStatus(ADC1\_FLAG\_EOC)==0){}

                ADC1\_ClearFlag(ADC1\_FLAG\_EOC);

                adc\_value = ADC1\_GetConversionValue();

                ADC\_Offset += adc\_value;

            }

            ADC\_Offset = ADC\_Offset / 20.0;

        }

Sau đó sẽ thực hiện lấy 1 mẫu ADC và lưu vào biến **ADC\_current.**

Lúc này ta có 2 thông số là **ADC\_current,ADC\_Offset** là giá trị ADC đọc được ở 2 thời điểm khác nhau, nếu có vật nặng thì giá trị ADC\_current sẽ lớn hơn giá trị offset

Từ giá trị ADC đó ta suy ra giá trị điện áp tương ứng bằng cách chia 1024 và nhân với điện áp cung cấp cho khối ADC là 3.3V. Lúc này ta đã tìm được cân nặng theo hệ đơn vị kg, để dễ gửi dữ liệu đi ta chuyển đơn vị qua gam bằng cách nhân với 1000.

Khi gửi dữ liệu đi qua UART nếu gửi liên tục sẽ không phân biệt được dữ liệu giữa các lần gửi vì vậy cần đặt ra một chuỗi cố định. Ở đây sẽ gửi ký tự ‘S’ trước tiên sau đó mới gửi dữ liệu về số cân nặng theo đơn vị gam, kết thúc bằng ký tự xuống dòng.

Và thời gian giữa các lần lấy mẫu là 500ms.

ADC1\_StartConversion();

        while(ADC1\_GetFlagStatus(ADC1\_FLAG\_EOC)==0){}

        ADC1\_ClearFlag(ADC1\_FLAG\_EOC);

        adc\_value = ADC1\_GetConversionValue();

        ADC\_current = adc\_value;

        Weight = (( abs((int) (ADC\_current - ADC\_Offset) )/1024.0) \* 3.3) \* 1000.0;

        uart1\_string("S");

        uart1\_number((int)Weight);

        uart1\_nline();

        delay\_ms(500);

Về phần khởi tạo UART, ở đây lập trình theo thanh ghi, tốc độ baud là 115200 để truyền dữ liệu lên ESP8266 được nhanh chóng, độ dài Data 8 bits, 1 bit stop

void uart\_init(void){

    uint32\_t BaudRate\_Mantissa = 0, BaudRate\_Mantissa100 = 0;

    UART1->CR1 &= (uint8\_t)(~UART1\_CR1\_M);  /\* Clear the word length bit \*/

    UART1->CR1 |= (uint8\_t)0x00; /\* Set the word length bit, 8 Data bits \*/

    UART1->CR3 &= (uint8\_t)(~UART1\_CR3\_STOP);  /\* Clear the STOP bits \*/

    UART1->CR3 |= (uint8\_t)0x00; /\*\*< One stop bit is transmitted at the end of frame\*/

    UART1->CR1 &= (uint8\_t)(~(UART1\_CR1\_PCEN | UART1\_CR1\_PS  ));  /\* Clear the Parity Control bit \*/

    UART1->CR1 |= (uint8\_t)0x00;  /\* Set No Parity Control bit \*/

    UART1->BRR1 &= (uint8\_t)(~UART1\_BRR1\_DIVM);  /\* Clear the LSB mantissa of UART1DIV  \*/

    UART1->BRR2 &= (uint8\_t)(~UART1\_BRR2\_DIVM); /\* Clear the MSB mantissa of UART1DIV  \*/

    UART1->BRR2 &= (uint8\_t)(~UART1\_BRR2\_DIVF);  /\* Clear the Fraction bits of UART1DIV \*/

    /\*Set baudrate = 115200, BRR1 = 0x08, BRR2 = 0x0B in datasheet\*/

    UART1->BRR2 = (uint8\_t)0x0B;

    UART1->BRR1 = (uint8\_t)0x08;

    UART1->CR2 |= (uint8\_t)UART1\_CR2\_TEN; /\*Transmit Enable and Receive Enable\*/

    UART1->CR2 |= (uint8\_t)UART1\_CR2\_REN;

    UART1->CR3 &= (uint8\_t)(~UART1\_CR3\_CKEN); /\*Sync mode Disable, SLK pin Disable\*/

    UART1->CR1 &= (uint8\_t)(~UART1\_CR1\_UARTD); /\* UART1 Enable \*/

}

1. Code trên Arduino cho ESP8266

Đầu tiên liệt kê các thư viện cần thiết, sau đó mình khởi tạo 1 đối tượng server hoạt động ở cổng 80, khai báo các biến sử dụng cho việc biến đổi dữ liệu cân nặng

#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#include <ESP8266mDNS.h>

#include <WiFiClient.h>

ESP8266WebServer server(80);

String Current\_str;

bool Flag = 0;

String Weight\_val;

int Gam\_Value;

float KGam\_Value;

Sau đó nhúng đoạn code HTML vào biến **TrangChuWebPage** , nội dung bao gồm cặp thẻ <html> </html> là bắt buộc, tiếp đến cặp thẻ <head> <\head> nơi chứa những khai báo đầu của trang

Thẻ <title> </title> là nội dung hiển thị trên tab trình duyệt web

Cặp thẻ <body> </body> nơi chứa nội dung của trang web

Tiếp đến

<meta http-equiv=”Content-Type” content=”text/html; charset=utf-8″> khai báo định dạng text – kiểu utf-8  
<title>CAN NANG LOADCEL</title> Khai báo phần title trên tab của trình duyệt web  
<meta name=”viewport” content=”width=device-width, initial-scale=1″> đây là đoạn mã để trình duyệt web tự động căn chỉnh trang cho vừa nhìn với màn hình (để vừa đẹp với cả điện thoại lẫn máy tính)  
<style>  
.b{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#4caf50;border-radius: 10px;}  
.t{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#f44336;border-radius: 10px;}  
</style>  
Trong cặp thẻ style mình khai báo 2 class “b” và “t” và viết các thuộc tính như chiều dài, chiều rộng, màu sắc cho nút nhấn **Value**

A green button with black text

Description automatically generatedVà ở phần thẻ body chính là nội dung chính của trang web với các button **Value**  và dòng chữ **Hiển thị giá trị cân nặng từ LoadCell**

String TrangChuWebPage =

{

  "<html>"

  "<head>"

  "   <meta http-equiv='Content-Type' content='text/html; charset=utf-8'>"

  "  <title>CAN NANG LOADCELL</title>"

  "<meta name='viewport' content='width=device-width, initial-scale=1'>"

  "<style>"

  " .b{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#4caf50;border-radius: 10px;}"

  " .t{width: 100px;height: 40px;font-size: 21px;color: #FFF;background-color:#f44336;border-radius: 10px;}"

  "</style>"

  "</head>"

  "<body>"

  " <div style='width: 330px;height: auto;margin: 0 auto;margin-top: 70px'>"

  "   <h1 align='center'>Hiển thị giá trị cân nặng từ LoadCell</h1>"

  "   <table align='center'>"

  "     <tr>"

  "       <td><a href='/pot'><button class='b'>Value</button></a><td>"

  "     <tr>"

  "   </table>"

  " </div>"

  "</body>"

  "</html>"

};

Ở đầu và cuối mỗi dòng HTML đều thêm 1 dấu nháy đôi “ và tất cả các dấu nháy đôi phía trong phải sửa thành dấu nháy đơn ‘

Trong hàm **setup** chúng ta khởi tạo cổng serial với tốc độ baud 115200 và với STM8S003F3P6 cũng khởi tạo UART với tốc độ như vậy, sau đó cho ESP8266 phát ra wifi với tên ESP8266 WiFI và password 12345678

Hàm **server.on(“/”, TrangChu);** yêu cầu ESP tạo ra 1 callback tới hàm tên là **TrangChu** khi có thiết bị truy cập tới, và ở hàm **TrangChu** , mình sẽ cho esp8266 trả lời lại mã html với code reponse là 200 (http code 200 báo truy vấn thành công )

void TrangChu()

{

  server.send(200, "text/html", TrangChuWebPage);

}

void setup()

{

  Serial.begin(115200);

  while (WiFi.softAP("ESP8266 WiFI", "12345678") == false)

   {

     Serial.print(".");

     delay(300);

   }

  IPAddress myIP = WiFi.softAPIP();

  server.on("/", TrangChu);

  server.on("/pot", pot);

  server.begin();

}

Hàm **server.on("/pot", pot);** sẽ yêu cầu ESP tạo ra 1 callback tới hàm tên là **pot()**

Trong hàm này sẽ đảm nhận việc nhận dữ liệu từ UART và xử lý dữ liệu đó để hiển thị lên web, chuỗi gửi lên sẽ có dạng “Sxxxx“. Ví dụ : S789, S98, S51, S1200. Lúc đó dấu hiệu nhận biết là ký tự “S” đầu chuỗi, nếu nhận được ký tự đó là báo hiệu sắp phải nhận giá trị cân nặng nên set cho biến **Flag = 1**.

Sau đó nhận liên tục các giá trị cân nặng cho đến khi hết dữ liệu thì xoá biến **Flag**, chuỗi cân nặng được lưu vào biến **Current\_str**

void pot()

{

  char c;

  if(Serial.available())

  {

    c = Serial.read();

    if(c == 'S'){

      Flag = 1;

      }

  }

  // S234P

  while(Flag == 1)

  {

    Current\_str = "";

    while(Serial.available()){

      c = Serial.read();

      if(c)

      {

        Current\_str += c;

      }

    }

    Flag = 0;

  }

Chuỗi cân nặng nhận được là đơn vị gam nên muốn hiển thị đơn vị là KiloGam thì ta chuyển chuỗi sang số nguyên và chia cho 1000. Vì muốn khi không có vật đặt lên LoadCell thì sẽ hiển thị 0g – 0Kg nên với các giá trị từ 0-50g đều cho hiển thị là 0 vì sai số của mạch là 50 đến 100g

Tiếp tục tạo một chuỗi dạng HTML và nhúng chuỗi đã xử lý vào.

"<html><head><meta http-equiv=\"refresh\" content=\"0.6\"></head> khai báo định dạng trang refresh với thời gian làm mới là 0.6s

<body bgcolor=\"#ffff00\"> <div style='width: 500px;height: auto;margin: 0 auto;margin-top: 70px'> <h1 align='center'>Weight Value:" + **Weight\_val** + "</h1></div></body></html>"; Hiển thị lên trang web dòng chữ **“Weight Value “** cùng với chuỗi cân nặng đã xử lý

Gam\_Value = Current\_str.toInt();

  if(Gam\_Value > 0 &&  Gam\_Value < 50 )

  {

    Gam\_Value = 0;

  }

  KGam\_Value = Gam\_Value/1000.0;

  Weight\_val = String(KGam\_Value) + " kg" + "-";

  Weight\_val += String(Gam\_Value) + " g" ;

  Weight\_val = "<html><head><meta http-equiv=\"refresh\" content=\"0.6\"> </head> <body bgcolor=\"#ffff00\"> <div style='width: 500px;height: auto;margin: 0 auto;margin-top: 70px'> <h1 align='center'>Weight Value:"

       + Weight\_val + "</h1></div></body></html>";

  server.send(200, "text/html", Weight\_val);

}

1. PCB và chạy thử mạch
2. PCB

Chúng ta có 2 PCB, 1 cho mạch tiền khuếch đại và PCB còn lại là PCB chính kết nối các module lại với nhau

A computer screen shot of a computer program

Description automatically generatedLayout cho mạch tiền khuếch đại

A hand holding a circuit board

Description automatically generated

A blue and black computer program

Description automatically generated with medium confidenceLayout cho mạch PCB chính

Các linh kiện sử dụng làm mạch

|  |  |  |  |
| --- | --- | --- | --- |
| Tên linh kiện | Số lượng | Tên linh kiện | Số lượng |
| LM324N | 1 | Hàng rào 1x40-2.54mm Male | 2 |
| Phíp đồng 1 mặt 10x20cm | 1 | Hàng rào 1x20-2.54mm Female | 6 |
| Giấy In mạch | 1 | Mũi khoan 0.8mm | 1 |
| Bột sắt | 1 |  |  |
| Tụ điện gốm 100n | 1 |  |  |
| Tụ hoá 220uF | 1 |  |  |
| Điện trở 2k, 100k, 1k | 8 |  |  |
| Biến trở tinh chỉnh 10K | 1 |  |  |
| Module STM8S003F3P6 | 1 |  |  |

Quá trình làm PCB

|  |  |
| --- | --- |
| Thời gian |  |
| 7-1-2024 | Vẽ Schematic mạch khuếch đại vi sai LM324N |
| 8-1-2024 | Lắp mạch tiền khuếch đại lên testboard chạy thử |
| 10-1-2024 | In PCB mạch tiền khuếch đại LM324N |
| 13-1-2024 | Ủi mạch, làm PCB mạch tiền khuếch đại LM324N |
| 17-1-2024 | Vẽ Schematic mạch PCB chính, lắp mạch lên testboard chạy thử  In PCB mạch chính, làm PCB mạch chính |

1. A hand holding a red wire

   Description automatically generatedChạy thử mạch

Ảnh bên là đo điện áp sau khi qua mạch khuếch đại vi sai, điện áp vi sai tại ngõ ra cảm biến LoadCell chỉ từ 1mV – 5mV vì vậy qua mạch khuếch đại vi sai với độ lợi 101 thì điện áp được tăng lên 101 lần. Như hình bên ta thấy điện áp khi không có vật nặng là 148mV, vậy thì điện áp vi sai tại ngõ ra LoadCell sẽ lấy gọi đó là

A hand holding a red pen and measuring device

Description automatically generatedẢnh bên là đo điện áp tại ngõ ra mạch khuếch đại không đảo với độ lợi 10, thì điện áp lúc nãy được tăng lên 1.47V, và chính điện áp này sẽ được đưa vào chân ADC của MCU xử lý

A hand holding a red wire with a black digital device on a table

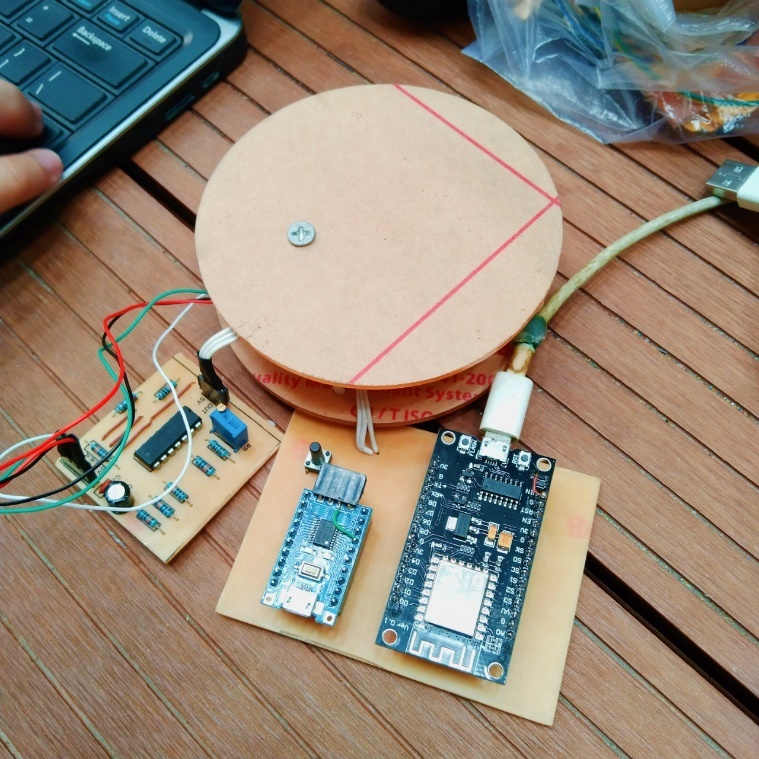
Description automatically generated

Ảnh bên là thử đặt một vật nặng là chai nước 500ml xấp xỉ 500g thì đo điện áp ngõ ra được 2.03V

Để tính được giá trị cân nặng ta lấy:

Vì vậy cân nặng tương đương 0.56Kg. Vì rated output(hay tỉ lệ ngõ ra) của LoadCell là 1.0±0.15mv/V.

Vì vậy sự thay đổi 1 mV tại LoadCell sẽ ứng với sự thay đổi 1V. Như ví dụ trên điện áp chênh lệch là 0.56V thì ứng với chênh lệch 0.56mV tại LoadCell, lúc đó cân nặng là 0.56Kg

1. Mạch thực tế
2. ĐÁNH GIÁ SẢN PHẨM
3. Ưu điểm và nhược điểm

+Ưu điểm: Đo được cân nặng từ 0-2kg. Sử dụng mạch khuếch đại vi sai nên chỉ cần đọc giá trị điện áp là có thể tính toán được giá trị cân nặng, không cần phải giao tiếp I2C module HX711

Số liệu được gửi lên web tiện theo dõi.

+Nhược điểm: khoảng đo đạc còn khá thấp, cần MCU có bộ ADC. Số liệu gửi lên web còn bị nhiễu chưa ổn định

Chưa thể xem số liệu ở mọi nơi, cần phải kết nối đến Wifi của thiết bị mới có thể xem số liệu

Trang web hiển thị số liệu còn sơ sài

1. Định hướng phát triển sản phẩm

Vì bộ ADC của STM8S003F3P6 dùng điện áp 3.3V mà ngõ ra của mạch tiền khuếch đại có thể đạt hơn 4V nếu vật nặng hơn 4Kg nên cần có cầu phân áp ở ngõ ra của mạch tiền khuếch đại, ADC sẽ lấy điện áp tại điểm giữa cầu phân áp. Từ đó tăng khoảng đo cân nặng

Sử dụng HTTP để có thể xây dựng 1 trang web thu thập dữ liệu phức tạp hơn, bao gồm vẽ biểu đồ cân nặng, sắp xếp cân nặng hay thu thập dữ liệu các lần đo,..

Sử dụng NAT PORT để có thể tuy cập vào trang web bất kỳ đâu mà vẫn có thể xem được dữ liệu cân nặng

Sử dụng lọc Kalman cho dữ liệu đọc từ ADC để chống nhiễu, đó là lọc trên phần mềm.