HelloCheez.sEMG肌肉电传感器

简介

Hello.sEMG是启思电子推出的一款肌电传感器模块。其能够通过检测人体表面的肌电信号（sEMG）反映肌肉和神经的活动。

本传感器模块集成了滤波、放大电路，将范围在mV甚至uV级别的微弱人体表面肌电信号进行13000倍放大，并通过差分输入、模拟滤波电路的方式对噪音（特别是工频干扰）进行有效抑制。输出信号为模拟量形式，以1.65V为基准电压，0～3.3V量程的输出。输出信号的大小取决于选定肌肉的活动量，输出信号的波形可显著指示被观察位置皮下肌肉的情况，方便做肌电信号的分析与研究，如使用Arduino作为控制器检测肌肉活动情况，如肌肉是否紧绷，强度如何，是否疲劳等。

本产品是一种主动感应传感器，能提供高质量的信号搜集，且易于使用，仅需要一些极为简单的准备工作即可。本产品使用干电极，因此具有寿命长、使用简单方便等特点，更适合普通用户。 本产品的测量具有非侵入性、无创伤、操作简单等优点，可用于人机交互等相关应用。虽然测量肌肉活动历来被用于医学研究，然而随着不断缩小但功能更强大的微控制器和集成电路的完善，肌电图电路和传感器也逐渐被应用于各种控制系统。

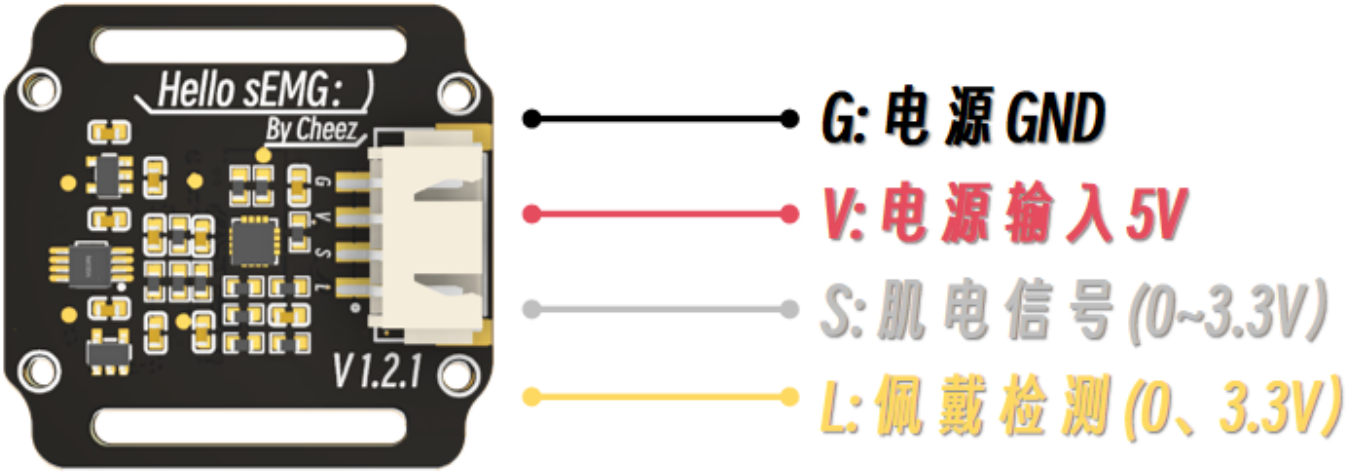
注意：

1. **电极板保持和肌肉方向一致。**
2. **皮肤干燥时，建议使用导电凝胶或湿巾擦拭电极和皮肤。**
3. **供电电压为5V，供电电流不小于20mA，纹波与其他噪音要小。**
4. **本品并非专业医疗仪器，不能作为辅助配件参与诊断和治疗。**

产品参数

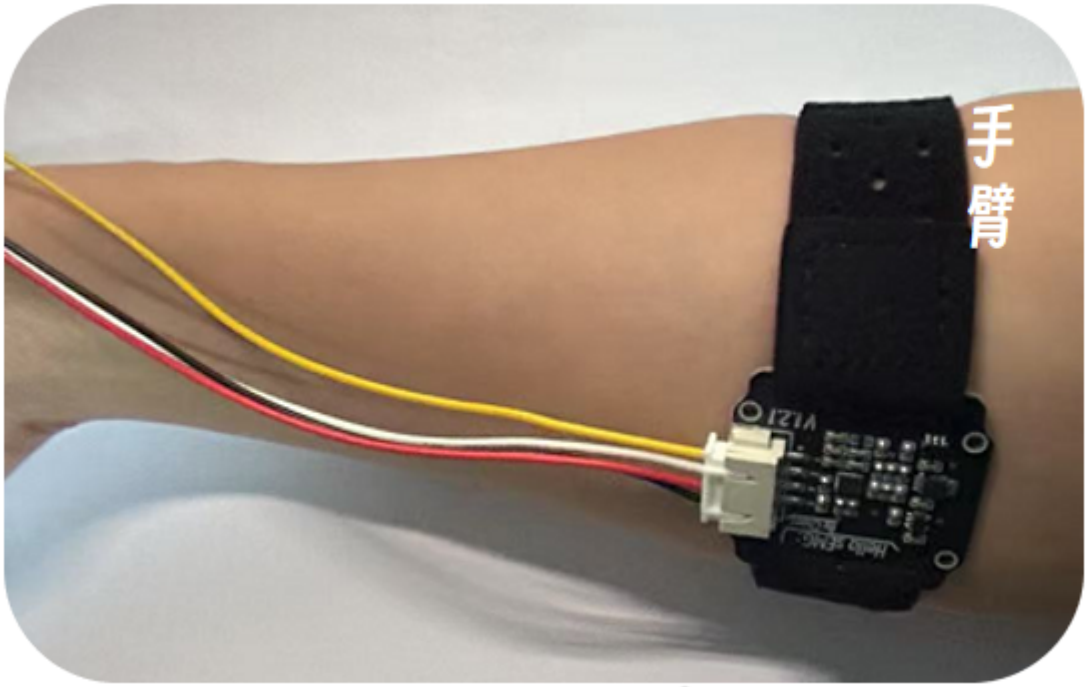
- 供电电压：5V-DC
- 输出电压：模拟量（0 - 3.3V）、 数字量（0 / 3.3V）
- 工作电流：< 10mA
- 尺寸：35mm x 33mm
- 模块接口：XH2.54-4P
- 工作温度：0～50℃
- 放大倍数：13000
- CMRR: <110dB

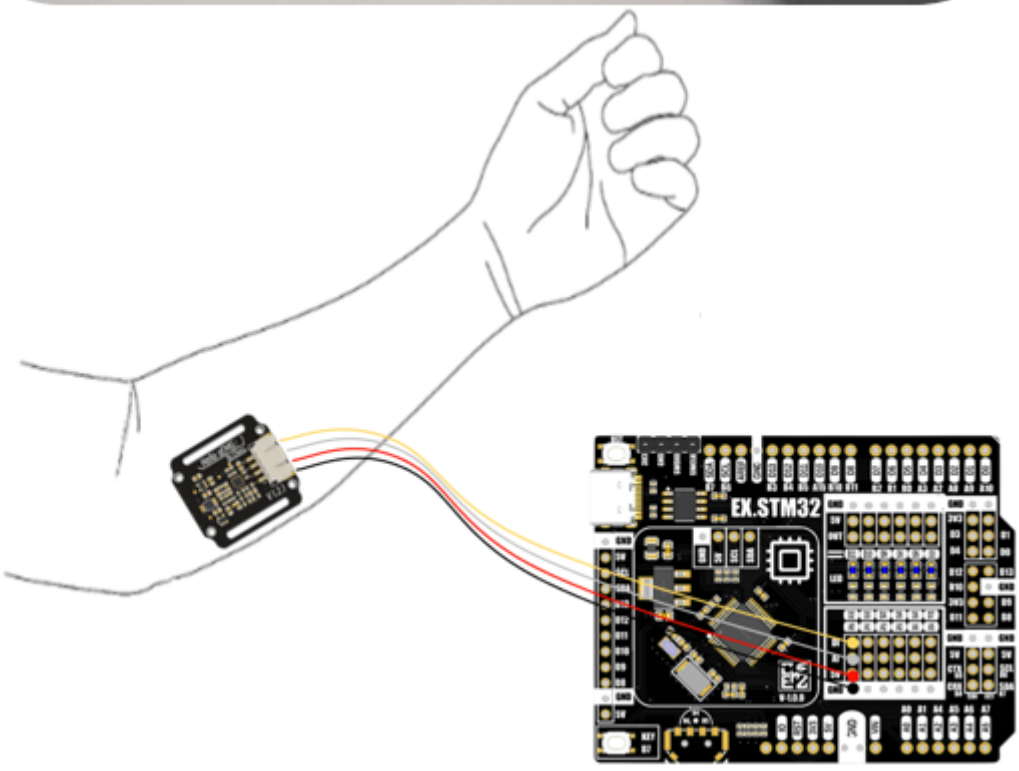
引脚说明


Hello sEMG:)
By Cheez
V1.2.1
G: 电源 GND
V: 电源输入 5V
S: 肌电信号 (0~3.3V)
L: 佩戴检测 (0、3.3V)

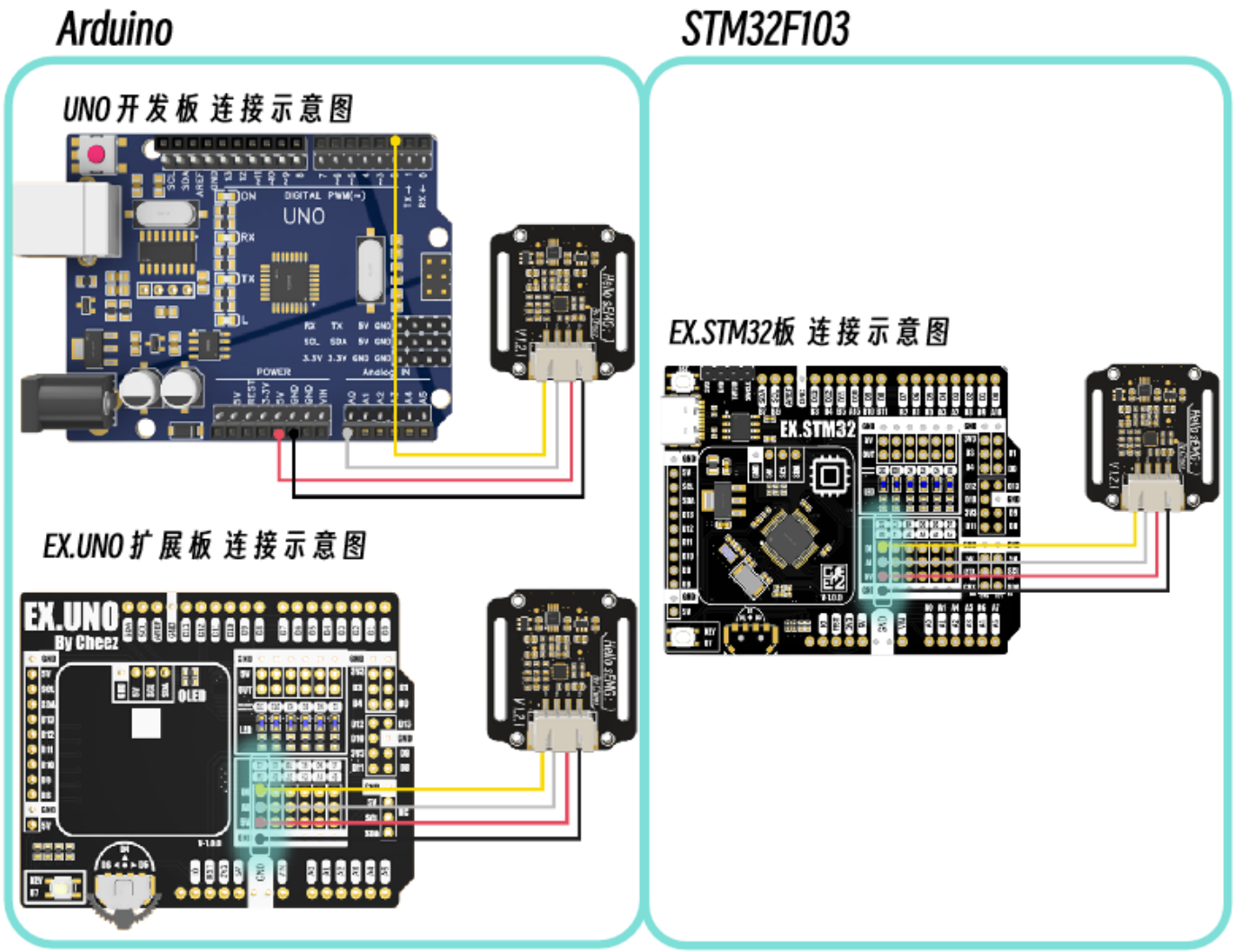
使用教程

安装方式





接线图



教程1

使用Arduino板测试肌电变化，输出肌肉电原始数据、佩戴检测数据。

准备

- **硬件**
  - Arduino UNO开发板 x1
  - EX.UNO控制板x1（可选）
  - Hello.sEMG肌电传感器 x 1
  - XH2.54端子 转 4Pin杜邦线 x1
  - 腕带 x1
- 软件
  - Arduino IDE 2.3

样例代码

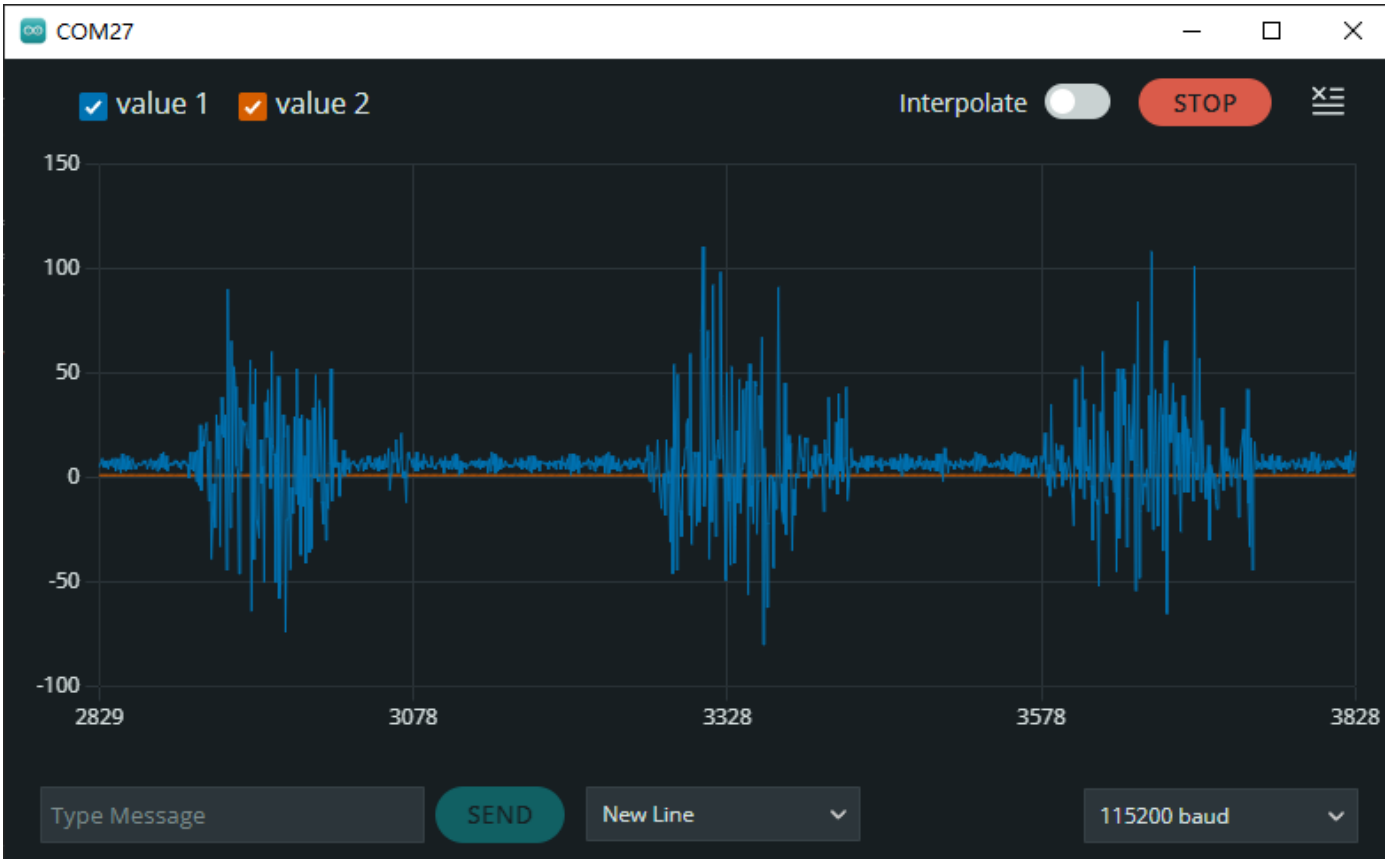
```
#define BAUD_RATE 115200                    // 串口波特率
```

```
#define INPUT_PIN A0                    // 肌电信号输入
#define DETECT_PIN 2                    // 佩戴检测输入
#define Vref (1.65f * 1024.0f / 5.0f)  // 抬升电压(V)

void setup() {
  Serial.begin(BAUD_RATE);
}

void loop() {
  int sensor_value = analogRead(INPUT_PIN);
  int detect_value = digitalRead(DETECT_PIN);
  int emgRaw = sensor_value - Vref;

  Serial.println(String(emgRaw) + "," + String(detect_value));
  delay(10);
}
```

实验结果

可通过ArduinoIDE上的 "串口绘图仪"

来查看肌肉电波形，如下👇



教程2

使用Arduino板测试肌电变化，采样率500Hz，输出肌肉电原始数据、包络数据。

准备

- **硬件**
  - Arduino UNO开发板 x1
  - EX.UNO控制板x1（可选）
  - Hello.sEMG肌电传感器 x 1
  - XH2.54端子 转 4Pin杜邦线 x1
  - 腕带 x1
- 软件
  - Arduino IDE 2.3

```
#define SAMPLE_RATE 500         // 采样率
#define BAUD_RATE 115200        // 串口波特率
#define INPUT_PIN A0            // 信号输入
#define DETECT_PIN 2            // 检测输入
#define Vref (1.65 / 5 * 1024) // 抬升电压

// 窗口
#define BUFFER_SIZE 128           // 窗口大小
int circular_buffer[BUFFER_SIZE]; // 环形数组
int data_index, sum;              // 数据索引

void setup ()
{
  Serial.begin (BAUD_RATE);
  pinMode (DETECT_PIN, INPUT); // 设置DETECT_PIN为输入模式
}

void loop ()
{
  // 计算经过的时间
  static unsigned long past = 0;
  unsigned long present = micros ();
  unsigned long interval = present - past;
  past = present;

  static long timer = 0;
  timer -= interval;

  if (timer < 0)
    {
      timer += 1000000 / SAMPLE_RATE;
      int sensor_value = analogRead (INPUT_PIN);
      int signal = Filter (sensor_value);
      int envelop = getEnvelop (abs (signal));

      Serial.println (String (signal) + "," +
```

```
                String (envelop)
                );
    }
}

// 包络检测: BUFFER_SIZE个数据均值
int getEnvelop (int abs_emg)
{
  sum -= circular_buffer[data_index];
  sum += abs_emg;
  circular_buffer[data_index] = abs_emg;
  data_index = (data_index + 1) % BUFFER_SIZE;
  return (sum / BUFFER_SIZE) * 2;
}

/***************************滤波***********************************/

// >>> Butterworth IIR Digital Filter: bandpass
//  Sampling Rate:500.0 Hz ,Frequency:[70.0, 110.0] Hz
//  Order: 4.0 ,implemented as second-order sections (biquads)
float Filter (float input)
{
  float output = input;
  {
    static float z1, z2; // filter section state
    float x = output - (-0.55195385 * z1) - (0.60461714 * z2);
    output = 0.00223489 * x + (0.00446978 * z1) + (0.00223489 * z2);
    z2 = z1;
    z1 = x;
  }

  {
    static float z1, z2; // filter section state
    float x = output - (-0.86036562 * z1) - (0.63511954 * z2);
    output = 1.00000000 * x + (2.00000000 * z1) + (1.00000000 * z2);
    z2 = z1;
    z1 = x;
  }

  {
    static float z1, z2; // filter section state
    float x = output - (-0.37367240 * z1) - (0.81248708 * z2);
    output = 1.00000000 * x + (-2.00000000 * z1) + (1.00000000 * z2);
    z2 = z1;
    z1 = x;
  }

  {
    static float z1, z2; // filter section state
    float x = output - (-1.15601175 * z1) - (0.84761589 * z2);
    output = 1.00000000 * x + (-2.00000000 * z1) + (1.00000000 * z2);
    z2 = z1;
    z1 = x;
```

```
  }

  return output;
}
```

实验结果

蓝色为滤波处理后的肌肉电数据，橙色为包络数据。