智能仓储系统的开发研究

先进计算与机器人研究所

2023 年 7 月 4 日

目录

1 第二章: 等待校准的秤

从上一章 1.3 节的图像可以看出, 称重结果有些不准, 离真实数据偏差较大, 我们需要进一步校准。从图像看出, 用直线拟合效果也能接接受。在这一章我们将给出一个可以进行一次校准的秤, 还是在 1.3 节 Calibrate 类上进行修改, 加入 kb 在 arduino 硬盘上读写的操作, 以及将校准后的结果打印在串口。

秤校准的步骤和 1.3 节相比基本类似:

- 1. 获得不准的数据数组;
- 2. 计算拟合系数;
- 3. 输出校准后的结果

```
class Calibrate {
union coeffience {
  unsigned long value_out;
  byte value_in[4];
};

private:
  int pin_SCK, pin_DT, range;
  float GapValue;
  unsigned long k, b;
  unsigned long x[13];
  int n;

public:
  Calibrate();

  void Get_x_array(unsigned long ADS);
  void Get_kb();
  unsigned long Output_CalibratedWeight(unsigned long weight);
  void kb_Initialize();

  void setpin_SCKDT(int p1, int p2);
  void set_range(int r);
  unsigned long Output_Weight(unsigned long ADS);
  unsigned long HX711_Read();
};
```

相比 1.3 节添加了两个函数, 修改了一个函数。

- Get_kb 中加入了写入 arduino 硬盘的操作;
- Output_CalibratedWeight 输出校准后的读数;
- kb_Initialize 是从 arduino 硬盘上读取存的 k,b 值。

```
void Calibrate::Get_kb(){
  unsigned long y[n] = {5, 15, 35, 55, 105, 574}, sum_x = 0, sum_y = 0;
  for (unsigned long& yi:y)
      sum_y+=yi;
  for (unsigned long& x_i:x)
    sum_x+=x_i;
  unsigned long mean_x = sum_x/n, mean_y = sum_y/n;
  unsigned long k1=0, k2=0;
  for (int i=0; i<n; ++i) {
    k1+=x[i]*y[i];
    k2+=x[i]*x[i];
  }
  k = (k1-n*mean_x*mean_y)/(k2-n*mean_x*mean_x);
```

```
  b = mean_y - mean_x * k;

  coeffience k_out;
  coeffience b_out;

  k_out.value_out = k;
  b_out.value_out = b;

  for(int i=4; i<8; i++) //判断是否与上次储存的k相同
    if (!(EEPROM.read(i)==k_out.value_in[i-4])) {
      Serial.print("newk:");
      Serial.println(k);
      for (int j=i; j<8; j++)
        EEPROM.write(j, k_out.value_in[j-4]);
      break;
  }

  for(int i=8; i<12; i++) //判断是否与上次储存的b相同
    if (!(EEPROM.read(i)==b_out.value_in[i-8])) {
      Serial.print("newb:");
      Serial.println(b);
      for (int j=8; j<12; j++)
        EEPROM.write(j, b_out.value_in[j-8]);
      break;
    }
};
```

在得到 k,b 后, 用共用体 k_out, b_out 存储 k, b 值, 通过 for 循环与 arduino 上对应地址存的 k(4 7), b(8 11) 值比较, 如果不一样则从当前位置写入。

```
coeffience k_out;
coeffience b_out;

k_out.value_out = k;
b_out.value_out = b;

for(int i=4; i<8; i++) //判断是否与上次储存的k相同
  if (!(EEPROM.read(i)==k_out.value_in[i-4])) {
    Serial.print("newk:");
    Serial.println(k);
    for (int j=i; j<8; j++)
      EEPROM.write(j, k_out.value_in[j-4]);
    break;
}

for(int i=8; i<12; i++) //判断是否与上次储存的b相同
  if (!(EEPROM.read(i)==b_out.value_in[i-8])) {
    Serial.print("newb:");
    Serial.println(b);
    for (int j=8; j<12; j++)
      EEPROM.write(j, b_out.value_in[j-8]);
    break;
  }
```

校准结果是在第一章中 Output_Weight 基础上, 加入了系数 k,b 的作用, 返回校准后的质量。

```
unsigned long Calibrate::Output_CalibratedWeight(unsigned long ADS){
  return k*Output_Weight(ADS)+b;
};
```

如果之前校准过, 可以通过 kb_Initialize() 读取 arduino 上存的 k,b 值:

```
void PressureSensor::kb_Initialize() {
```

```
  coeffience k_arduino;
  coeffience b_arduino;

  for(int i=4; i<8; i++) {
    k_arduino.value_in[i-4] = EEPROM.read(i);
    b_arduino.value_in[i-4] = EEPROM.read(i+4);
  }

  k = k_arduino.value_out;
  b = b_arduino.value_out;
  Serial.print("k:␣");
  Serial.println(k);
  Serial.print("b:␣");
  Serial.println(b);
};
```

arduino 上 k 对应的地址是 4 到 7, b 对应的地址是 8 到 11。首先创建两个共用体 k_arduino, b_arduino。

```
  coeffience k_arduino;
  coeffience b_arduino;
```

然后用一个 for 循环读对应地址上信息存放到对应共用体中。

```
  for(int i=4; i<8; i++) {
    k_arduino.value_in[i-4] = EEPROM.read(i);
    b_arduino.value_in[i-4] = EEPROM.read(i+4);
  }
```

最后是更新变量 k,b, 并在串口打印。

```
  k = k_arduino.value_out;
  b = b_arduino.value_out;
  Serial.print("k:␣");
  Serial.println(k);
  Serial.print("b:␣");
  Serial.println(b);
```

1.1 Calibrate.ino

```
#include "Surface.h"
#include "Calibrate.h"

Surface YL_Surface;
Calibrate YL_Calibrated;

bool flag=1;

void setup() {
  Serial.begin(9600);
  YL_Calibrated.setpin_SCKDT(4, 5);
  YL_Calibrated.set_range(20);
  YL_Calibrated.kb_Initialize();
}

void loop() {
  if (flag) {
    YL_Calibrated.Get_x_array(YL_Surface.Get_Surface());
    YL_Calibrated.Get_kb();
    flag=0;
  }
```

```
22
23    unsigned long CalibratedWeight = YL_Calibrated.Output_CalibratedWeight(YL_Surface.
          Get_Surface());
24    Serial.println(CalibratedWeight);
25    delay(1000);
26  }
```

主程序分为三部分:

第一部分: 导入头文件 Calibrate.h, Surface.h, 实例化 Surface 类. Oled 类, flag=1 方便主循环调用一次校准;

```
1   #include "Surface.h"
2   #include "Calibrate.h"
3
4   Surface YL_Surface;
5   Calibrate YL_Calibrated;
6
7   bool flag=1;
```

第二部分: 在 setup 函数里初始化波特率 9600, 与 Arduino 串口通信的波特率保持一致;

```
1   Serial.begin(9600);
```

设置与 SCK 和 DT 相连的 arduino 引脚 4 和 5, 设置当前压力传感器的量程 20. kb_Initialize() 读取之前存的 k,b 值:

```
1   YL_Calibrated.setpin_SCKDT(4, 5);
2   YL_Calibrated.set_range(20);
3   YL_Calibrated.kb_Initialize();
```

第三部分: flag 为 1 时进行一次校准, 并在校准后 flag 赋值为 0, 避免重复校准。先通过 Get_x_array 获得不准的读数数组, 再通过 Get_kb 获得拟合系数, 最后由 Output_CalibratedWeight 计算校准后的结果。

```
1   if (flag) {
2     YL_Calibrated.Get_x_array(YL_Surface.Get_Surface());
3     YL_Calibrated.Get_kb();
4     flag=0;
5   }
6
7   unsigned long CalibratedWeight = YL_Calibrated.Output_CalibratedWeight(YL_Surface.
        Get_Surface());
8   Serial.println(CalibratedWeight);
9   delay(1000);
10  }
```