# 北京理工大学珠海学院

**计算机学院**

# 传感器与传感网络

课程设计报告

题 目 平衡小车

组 长 吴慕凡

组 员 雷有为，朱信至

2019年6月17日

# 目录

目录

[1. 前言 1](#_Toc31587_WPSOffice_Level1)

[1.1 项目背景 1](#_Toc32064_WPSOffice_Level2)

[1.2 需求分析 1](#_Toc29339_WPSOffice_Level2)

[2. 主要模块功能设计 2](#_Toc32064_WPSOffice_Level1)

[2.1 上位机功能设计 3](#_Toc11252_WPSOffice_Level2)

[3.硬件设计 3](#_Toc29339_WPSOffice_Level1)

[5. 总结 7](#_Toc20123_WPSOffice_Level1)

[代码参考 8](#_Toc27072_WPSOffice_Level1)

# 前言

## 项目背景

  目前平衡车市场已经达到了几十亿的市场需求，其中中国就达到了每年将近6亿人民币的市场需求。如今平衡车已经衍生出了城市交通车，安防巡逻车等多种平衡车使用用途。

## 需求分析

1. 由于平衡车的特殊运动原理，使得它区别于其他多轮车的使用方式上就有所不同，可以解放双手控制，单纯的控制身体平衡以及双脚的扭转方向就可以简单的控制平衡车的运动，做到其他多轮车难以实现的效果。
2. 基于平衡车这个特殊的平台，可以在其基础上实现一些特殊的功能。例如探测车。

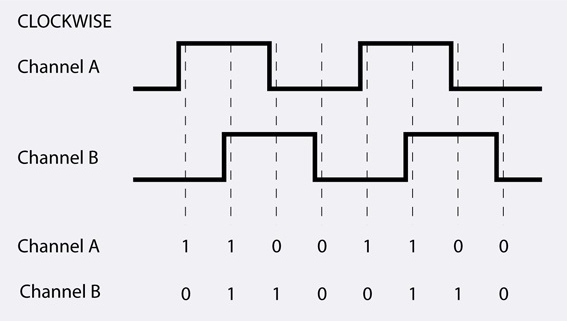
# 2. 主要模块功能设计

平衡小车的控制原理，车子往那倒，轮子就往哪边走，通过惯性将小车扶起来。

本平衡小车主要用到了两种传感器，一是用于测量直流电机运动状态的霍尔编码器，二是用于测量平衡车当前俯仰角度的陀螺仪MPU6050。通过这两者结合加上控制算法，即可达到平衡车的直立平衡的效果。

**霍尔编码器**：

* 直流电机上面的霍尔编码器有两个霍尔传感器，精度一圈是390，也就是说转一圈下来，霍尔传感器可以检测，390组跳变沿，一高一低为一组。然后，这两个霍尔编码器相差四分之一个周期，通过这个周期差，我们可以通过算法计算出当前电机的运动方向以及速度。



A相与B相错开了四分之一个周期，通过这个规律，当A相的跳变沿为下降沿时候，B相如果是高电平，那么运动方向为正向，反之如果A相下降沿时，B相为低电平，那么电机运动方向为负方向。借此还可以将编码器的精度提升两倍或者四倍

相关代码；

attachInterrupt(0, READL, CHANGE); //开启外部中断 编码器接口1

attachPinChangeInterrupt(4, READR, CHANGE); //开启外部中断 编码器接口2

（以上代码用于io口的外部中断，即io口电平发生变化时触发中断）

void READL() {

if (digitalRead(LeftA) == LOW) { //如果是下降沿触发的中断

if (digitalRead(LeftB) == LOW) leftcount--; //根据另外一相电平判定方向

else leftcount++;

}

else { //如果是上升沿触发的中断

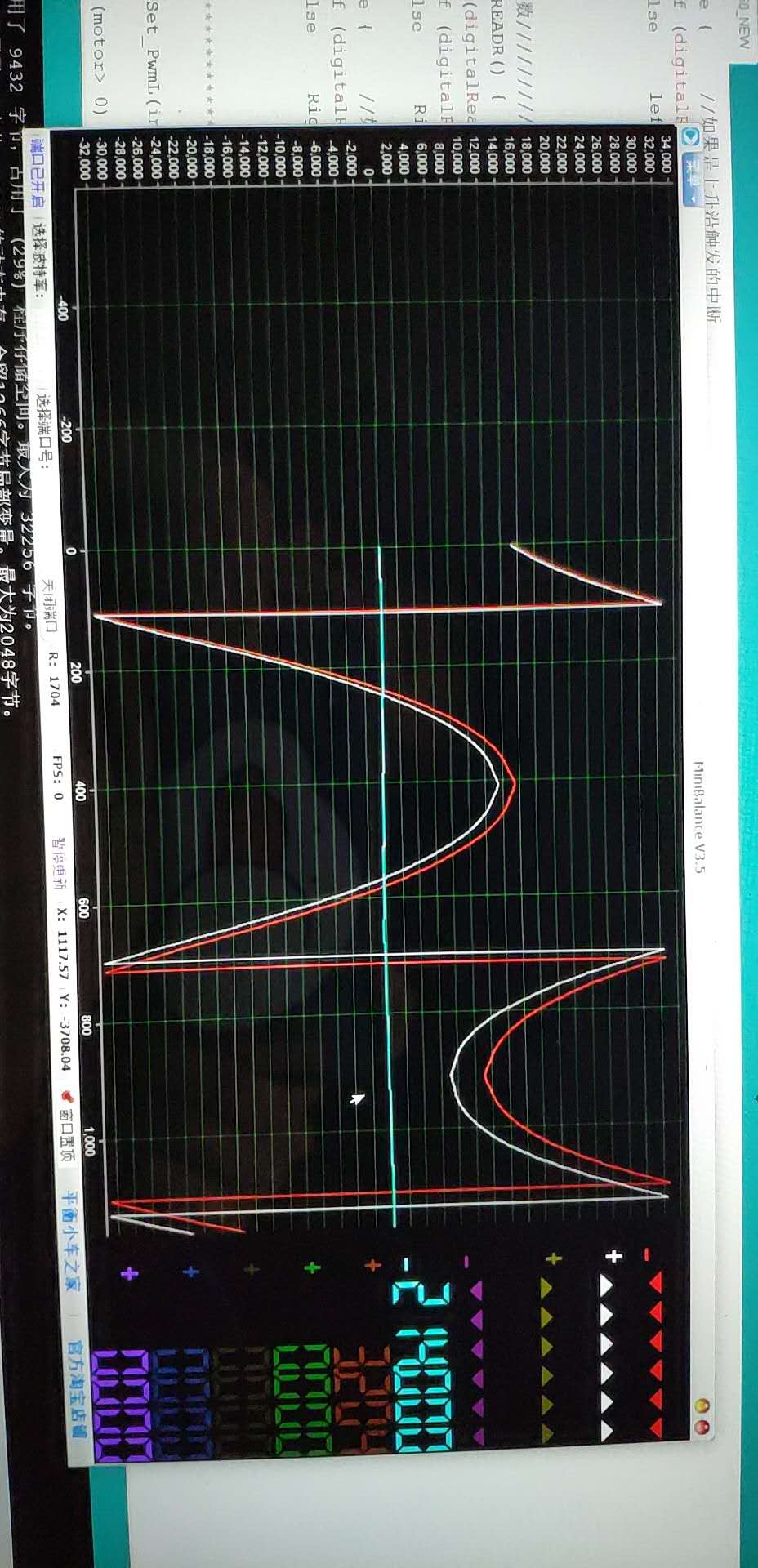
if (digitalRead(LeftB) == LOW) leftcount++; //根据另外一相电平判定方向

else leftcount--;

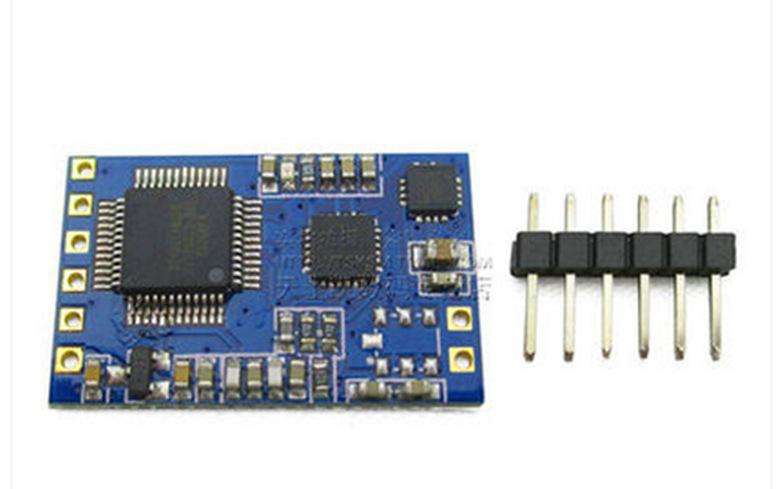
}

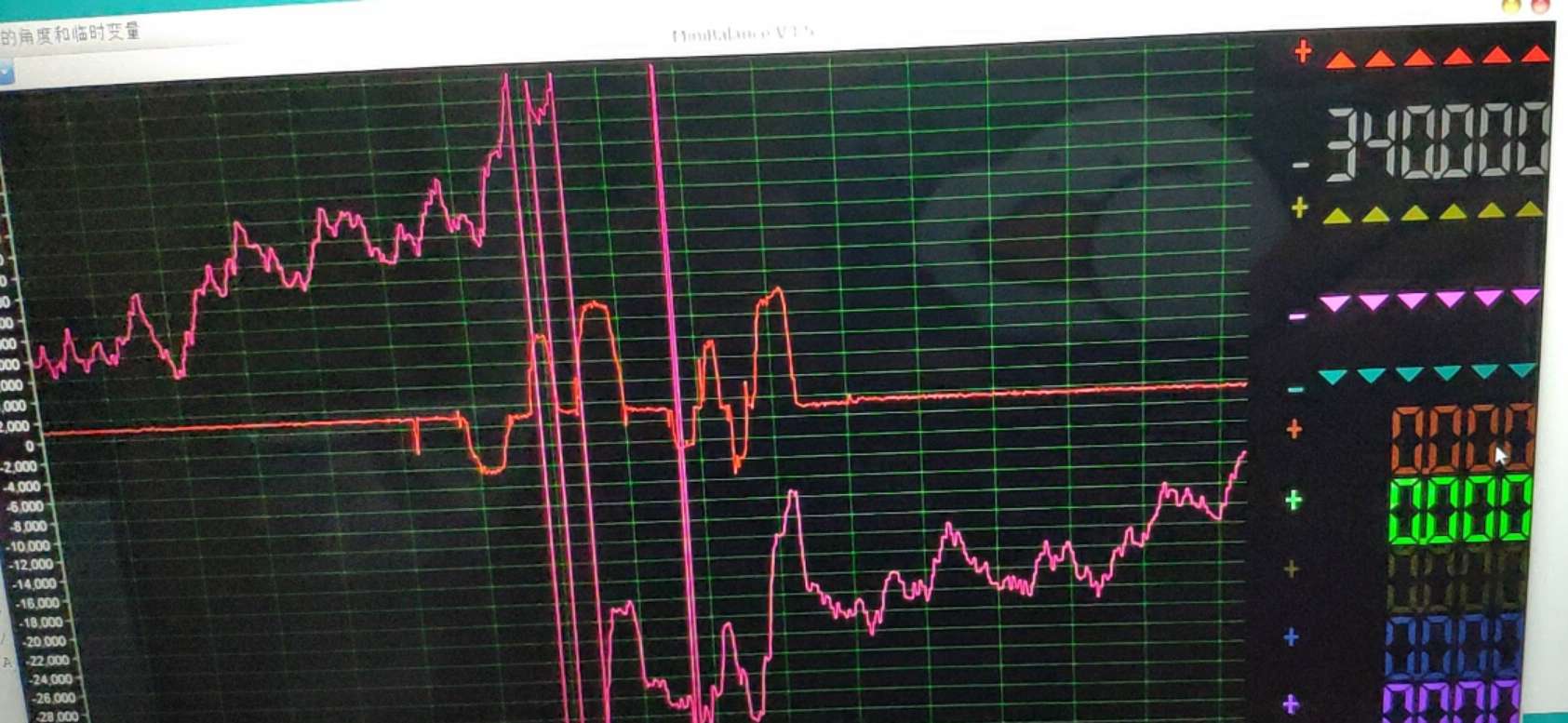
}

（以上代码用于霍尔编码器的计数，触发一次中断就进入计数一次）

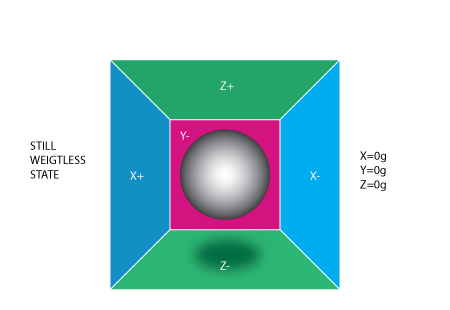


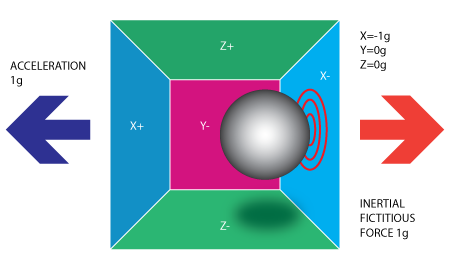
通过上位机将数据转化为波形，可以看到在不同转速下，霍尔编码器件读取的数据进行加减（pwm在-255到255之间均匀的变化）

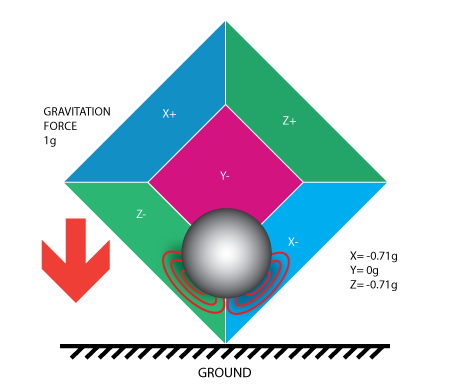
**陀螺仪MPU6050:** ****

* 这个陀螺仪通过I2c与arduino进行通信，通过读取寄存器数据，可以读取到xyz三个方向的中立加速度，以及xyz三个轴向的角速度，通过这六个数据，我们加以处理，通过窗口滤波，一节互补滤波，可以求出当前陀螺仪的当前角度位置。
* ****

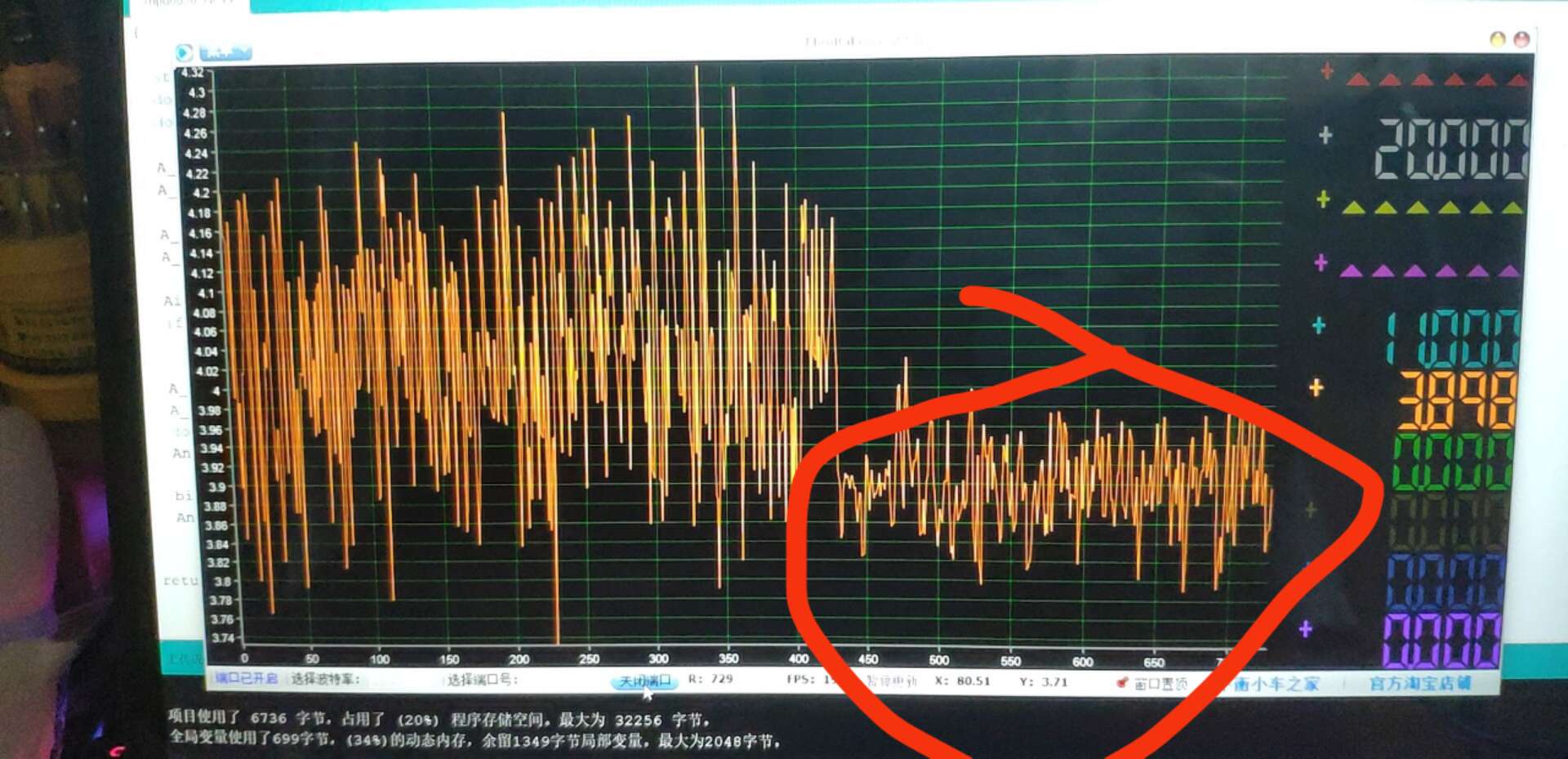
陀螺仪xyz三个方向的重力加速度的相关数据校准是需要通过人工来测得的。

从图片可以看到，陀螺仪的xzy三个方向的重力加速度，就好比是一个盒子的六面壁，当小球压在其中一个面上的时候，那个面就会感受到压力，从而读出数据，通过慢动作旋转可以近似的找出小球xzy三面加速度的范围。（这里旋转的时候一定要慢和轻，因为晃动和抖动都会增大小球在那个面的压力，从而使找出来的中期加速度取值范围不准）





通过这张图，可以看出在这种情况下，我们利用arctan的计算公式，将两面的重力加速带入，从而求出当前的角度，下图的左边是直接用了两面加速度的数值来计算，可以看到噪音很大，容易受到抖动晃动的影响。



通过滤波，可以将算出来的当前角度抖动减缓（上图的红圈部分），但这还不够，在抖动大的时候还是有比较明显的峰值。

这个时候就用到了陀螺仪测出来的xzy轴的三个角速度，通过那个旋转角的角速度，来做积分并且使用了一阶互补滤波，可以是角度的变化变得更加平滑。



* **关键算法**
* bias\_A=Angle\_A\*0.02 + (bias\_A+0.005\*g\_y)\*0.98; //一阶互补

将得出的角度数据给到PID算法中，（注意轮子转动方向一定是，角度倾斜的方向）

## 

## 2.1 上位机功能设计

这里用的是minibalance的上位机，可以清晰的将读取到的数据用波形显示出来 ，并且可以同时显示多个数据用作比较。

**相关代码：**

void DataScope(void)

{

int i;

data.DataScope\_Get\_Channel\_Data(leftcount, 1);

data.DataScope\_Get\_Channel\_Data(Rightcount, 2);

data.DataScope\_Get\_Channel\_Data(Motor, 3);

data.DataScope\_Get\_Channel\_Data(A\_X\_, 4);

data.DataScope\_Get\_Channel\_Data(bias\_A, 5);

data.DataScope\_Get\_Channel\_Data(angle, 6);

Send\_Count = data.DataScope\_Data\_Generate(6);

for ( i = 0 ; i < Send\_Count; i++)

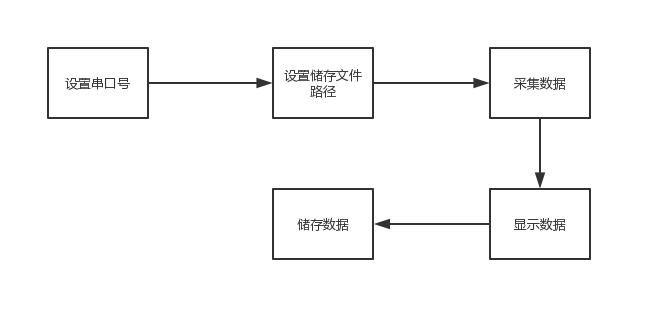
{

Serial.write(DataScope\_OutPut\_Buffer[i]);

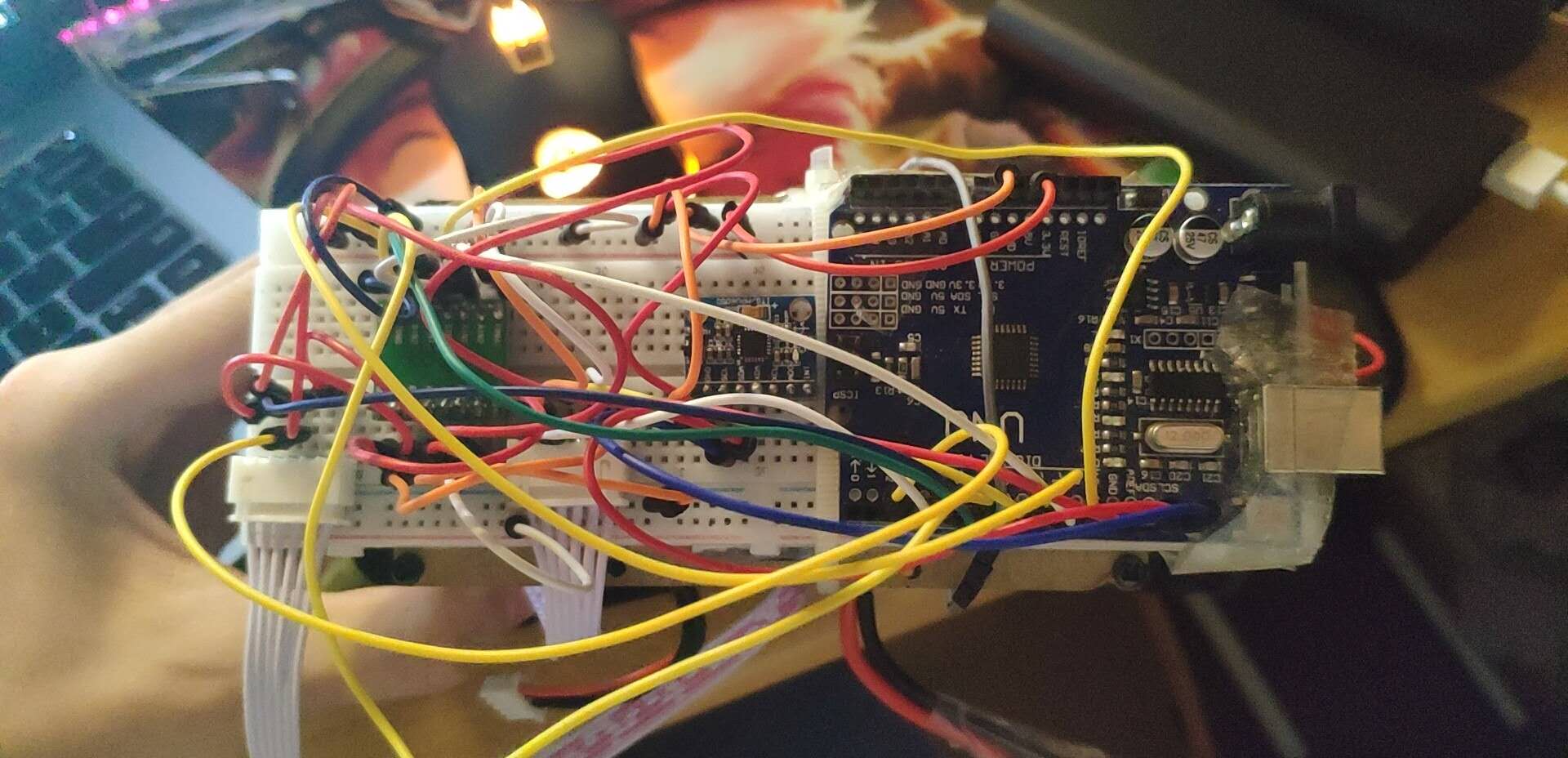
}

delay(50);

}



# 硬件设计



电路用了一块arduino，6612电机驱动芯片，mpu6050陀螺仪芯片，

以及两个带编码器的直流电机。下面一层有两块电池，一块12v航模电池，用来给tb6612芯片供电，一块7v的电池给arduino供电。

需要注意的点，陀螺仪的位置尽量靠中间，这样方便调车。

# 总结

本次实验比较具有综合性，之前学过的相关知识都有运用到，通过arduino主控将其结合运用起来，我觉得最主要的一个难点在于是陀螺仪调试，测量。

花费了大量经验。在整个实验过程中出现了大大小小的bug，包裹硬件软件，这次实验过后大大增加了我们debug的经验，以及debug的能力，这非常的重要。最后几天把平衡车做出来真的成就感很足，因为不是用了网上店家的代码，而是是自己一点点测出来试出来的代码，为了做到这些学了很多知识，收获颇丰。

# 代码附录

#include <DATASCOPE.h> //这是PC端上位机的库文件

DATASCOPE data;

unsigned char Send\_Count; //上位机相关变量

#include <KalmanFilter.h> //卡尔曼滤波

#include <MsTimer2.h> //定时中断

#include <PinChangeInt.h> //外部中断

#include "I2Cdev.h"

#include "MPU6050\_6Axis\_MotionApps20.h"//MPU6050库文件

#include "Wire.h"

MPU6050 Mpu6050; //实例化一个 MPU6050 对象，对象名称为 Mpu6050

KalmanFilter KalFilter;//实例化一个卡尔曼滤波器对象，对象名称为 KalFilter

int16\_t ax, ay, az, gx, gy, gz; //MPU6050的三轴加速度和三轴陀螺仪数据

double angle, Gryo; //用于显示的角度和临时变量

//左电机

int AIN2=13;

int AIN1=12;

int PWMA=10;

//左电机编码器读取

int LeftA=2;

int LeftB=5;

//左电机编码器转化

int La,Lb,La1,Lb1;

int leftcount=10000;//脉冲计数

int model;//记录标志位zuo

//右电机

int BIN2=7;

int BIN1=6;

int PWMB=9;

//右电机编码器读取

int RightA=4;

int RightB=8;

//右电机编码器转化

int Ra,Rb,Ra1,Rb1;

int Rightcount=10000;//脉冲计数

int modeR;//记录标志位zuo

/\*\*\*\*\*\*\*\*\*\*\*\*\*陀螺仪滤波参数\*\*\*\*\*\*\*\*\*\*\*/

int16\_t A\_X\_sz[20] = {0},A\_Z\_sz[20] = {0};

double A\_X\_, A\_Z\_;

int32\_t A\_X\_Count = 0, A\_Z\_Count = 0;

int8\_t Ai = 0;

double bias\_A;

//窗口滤波

float Angle\_Cale(int16\_t a\_x, int16\_t a\_z, int16\_t g\_y)

{

double Angle\_A;

double Angle;

//////////均值滤波///

A\_X\_Count = A\_X\_Count + a\_x - A\_X\_sz[Ai];

A\_X\_sz[Ai] = a\_x;

A\_Z\_Count = A\_Z\_Count + a\_z - A\_Z\_sz[Ai];

A\_Z\_sz[Ai] = a\_z;

//////////////

Ai = Ai+1;

if(Ai >= 20)

{Ai = 0;}

A\_X\_ = 1.0 \* A\_X\_Count / 20;

A\_Z\_ = 1.0 \* A\_Z\_Count / 20;

double j = atan(A\_X\_/A\_Z\_);

Angle\_A = -(j\*180/3.14);// 算出角度

g\_y=g\_y/135;

bias\_A=Angle\_A\*0.02 + (bias\_A+0.005\*g\_y)\*0.98; //一阶互补

Angle= bias\_A;

return Angle;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//PID

float P\_KP=200,P\_KI=0.1,P\_KD=0; //PID系数

float Motor;

float Target\_angle=-3.6;//目标位置

int Position\_PID (int Encoder,int Target)

{

static float Pwm,Integral\_bias,Last\_Bias;

float Bias;

Bias=Encoder-Target; //计算偏差

Integral\_bias+=Bias; //求出偏差的积分

Pwm=P\_KP\*Bias/28+P\_KI\*Integral\_bias/2800+P\_KD\*(Bias-Last\_Bias)/28; //位置式PID控制器

Last\_Bias=Bias; //保存上一次偏差

return Pwm; //增量输出

}

int Balance\_pwm(double T\_ang,double angle,int g\_y )

{ int B\_pwm;

double ang;

ang=angle-T\_ang;

g\_y=g\_y/135;

B\_pwm= 40\*ang+0.6\*g\_y;

return B\_pwm;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void control()

{

sei();//全局中断开启

Mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); //获取MPU6050陀螺仪和加速度计的数据

//偏差

ax-=775;

az+=1600;

gy+=123;

angle = Angle\_Cale(ax,az,gy);

Motor=Balance\_pwm(Target\_angle,angle,gy);

Set\_PwmL(Motor);//正死区 19 负 —18

Set\_PwmR(Motor);//正死区 16 负 —18

}

void DataScope(void)

{

int i;

data.DataScope\_Get\_Channel\_Data(leftcount, 1);

data.DataScope\_Get\_Channel\_Data(Rightcount, 2);

data.DataScope\_Get\_Channel\_Data(Motor, 3);

data.DataScope\_Get\_Channel\_Data(A\_X\_, 4);

data.DataScope\_Get\_Channel\_Data(bias\_A, 5);

data.DataScope\_Get\_Channel\_Data(angle, 6);

Send\_Count = data.DataScope\_Data\_Generate(6);

for ( i = 0 ; i < Send\_Count; i++)

{

Serial.write(DataScope\_OutPut\_Buffer[i]);

}

delay(50);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup() {

pinMode(AIN1,OUTPUT);

pinMode(AIN2,OUTPUT);

pinMode(PWMA,OUTPUT);

pinMode(LeftA,INPUT);

pinMode(LeftB,INPUT);

pinMode(BIN1,OUTPUT);

pinMode(BIN2,OUTPUT);

pinMode(PWMB,OUTPUT);

pinMode(RightA,INPUT);

pinMode(RightB,INPUT);

digitalWrite(AIN1, 0);

digitalWrite(AIN2, 0);

digitalWrite(PWMA, 0);

digitalWrite(BIN1, 0);

digitalWrite(BIN2, 0);

digitalWrite(PWMB, 0);

Wire.begin(); //加入 IIC 总线

Serial.begin(115200); //开启串口，设置波特率为 9600

delay(1000); //延时等待初始化完成

Mpu6050.initialize(); //初始化MPU6050

delay(20);

MsTimer2::set(5, control); //使用Timer2设置5ms定时中断

MsTimer2::start(); //使用中断使能

attachInterrupt(0, READL, CHANGE); //开启外部中断 编码器接口1

attachPinChangeInterrupt(4, READR, CHANGE); //开启外部中断 编码器接口2

}

void loop() {

Serial.begin(115200), DataScope(); //延时等待初始化完成

//延时等待初始化完成

}

//左计数///////////////////////////////////////////////////////////////////////

void READL() {

if (digitalRead(LeftA) == LOW) { //如果是下降沿触发的中断

if (digitalRead(LeftB) == LOW) leftcount--; //根据另外一相电平判定方向

else leftcount++;

}

else { //如果是上升沿触发的中断

if (digitalRead(LeftB) == LOW) leftcount++; //根据另外一相电平判定方向

else leftcount--;

}

}

//右计数///////////////////////////////////////////////////////////////////////

void READR() {

if (digitalRead(RightA) == LOW) { //如果是下降沿触发的中断

if (digitalRead(RightB) == LOW) Rightcount--; //根据另外一相电平判定方向

else Rightcount++;

}

else { //如果是上升沿触发的中断

if (digitalRead(RightB) == LOW) Rightcount++; //根据另外一相电平判定方向

else Rightcount--;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PWM电机控制\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Set\_PwmL(int motor){

if (motor> 0) digitalWrite(AIN1, HIGH), digitalWrite(AIN2, LOW); //TB6612的电平控制

else digitalWrite(AIN1, LOW), digitalWrite(AIN2, HIGH); //TB6612的电平控制

if(motor>0) motor+=16;

if(motor<0) motor-=9;

analogWrite(PWMA, abs(motor));

}

void Set\_PwmR(int motor){

if (motor> 0) digitalWrite(BIN1, HIGH), digitalWrite(BIN2, LOW); //TB6612的电平控制

else digitalWrite(BIN1, LOW), digitalWrite(BIN2, HIGH); //TB6612的电平控制

if(motor>0) motor+=13;

if(motor<0) motor-=9;

analogWrite(PWMB, abs(motor));

}