陀螺仪使用与云台控制

陀螺仪原理

- 传统的陀螺仪是通过高速旋转的物体的旋转轴，对于改变其方向的外力作用有趋向于垂直方向的倾向这个原理制作的机械装置，通过对比陀螺仪的的几个面，可以获得系统当前的姿态信息
- 电子陀螺仪使用电子元件替换机械元件，体积更小，可以获得更多姿态信息，常见的有三轴、六轴、九轴陀螺仪

机械式陀螺仪

三轴陀螺仪	围绕XYZ三轴的角速度	必须通过其他方式获取系统初始姿态，再通过角速度解算当前姿态
六轴陀螺仪	围绕XYZ三轴的角速度 XYZ三轴方向的加速度	可通过重力加速度与角速度解算当前俯仰姿态
九轴陀螺仪	围绕XYZ三轴的角速度 XYZ三轴方向的加速度 XYZ三轴方向的磁强计	可通过磁力计与角速度解算当前偏航姿态

三轴、六轴、九轴陀螺仪的功能

看门狗

- 在使用6050前我们介绍另一个单片机模块，由于6050和手柄一类的元器件容易收到干扰，以至出错不响应，为了防止程序卡死在某个部分，我们使用看门狗对程序进行监控
- 看门狗的原理很简单，当我们放出看门狗后，每隔一个固定的时间就要喂一次狗，否则看门狗就会强制对CPU进行重启，看门狗调用函数被封装进wDog.ino文件

看门狗初始化，设置喂狗时间为1s

```
wDogInit();
```

喂狗，每次喂狗间隔时间不能超过1s

```
feetDog();
```

陀螺仪读取方法

- 陀螺仪模块需要调用I2C协议库、设备库、MPU6050库以及MPU6050Read.ino。

```
#include "Wire.h"//I2C协议库
#include "I2Cdev.h"//I2C设备库
#include "MPU6050.h"//MPU6050库

MPU6050 accelgyro;//实例化6050

int Accel[3] = {0, 0, 0};//用于存放xyz三轴加速度的全局变量
int Gyro[3] = {0, 0, 0};//用于存放xyz三轴角速度的全局变量
int GyroOffset[3] = {0, 0, 0};//用于存放xyz三轴角速度零偏的全局变量
```

陀螺仪读取方法

- 需要对看门狗、I2C、6050进行初始化，并获取6050的角速度零偏。

```
void setup()
{
  OLED_Init();//OLED初始化
  wDogInit();//看门狗初始化（1s内必须喂狗）
  Wire.begin();//I2C初始化，用于与6050通信
  delay(200);//延时200ms确保6050上电
  feetDog();//喂狗
  accelgyro.initialize();//6050初始化
  accelgyro.initialize();//再次初始化6050，确保初始化完成
  //测试6050连接，并从串口输出测试结果
  Serial.println(accelgyro.testConnection() ? "MPU6050 successful" : "MPU6050 failed");
  feetDog();//喂狗
  delay(600);//延时600ms让系统静置，准备从6050读取角速度零偏
  getGyroOffset(GyroOffset);//读取角速度零偏
  feetDog();//喂狗
}
```

陀螺仪读取方法

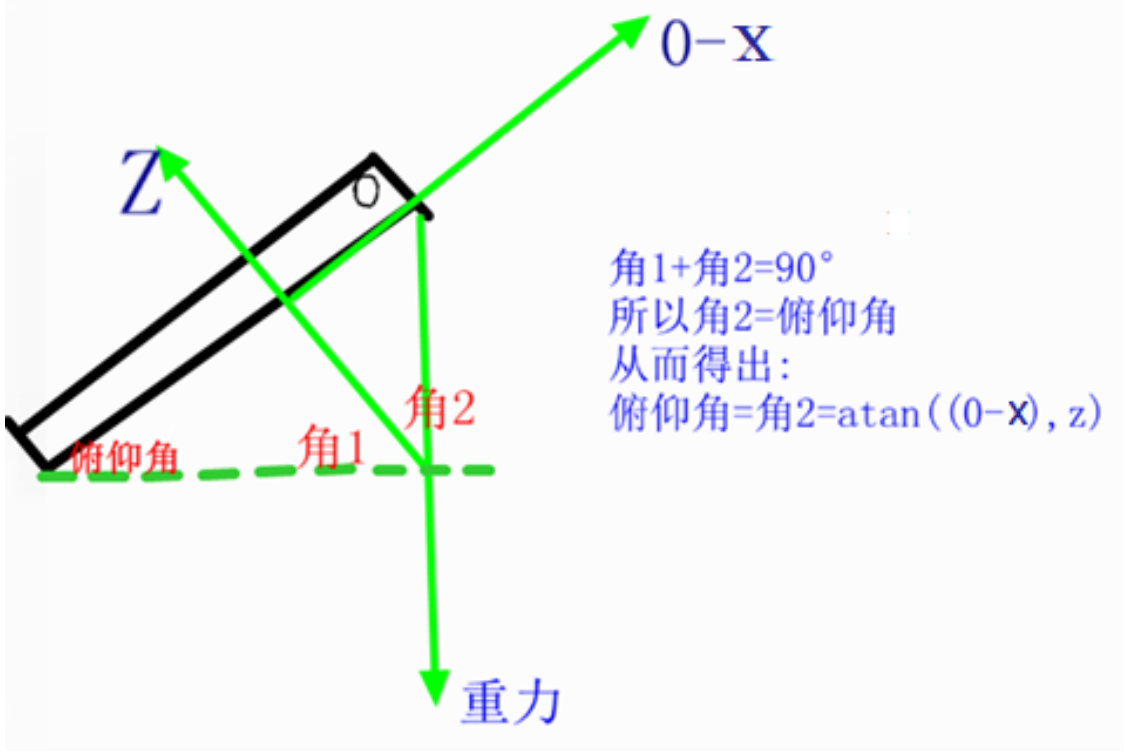
- 在loop()中读取6050的六轴原始值，并显示

```
void loop()
{
  read6050(Accel, Gyro);//读取6050的三轴加速度与角速度的原始数据
  AccelAngle = accelAngle(Accel[0], Accel[2]);//根据加速度计算倾角
  for (int i = 0; i < 3; i++)//循环3次
    Gyro[i] -= GyroOffset[i];//三轴角速度去零偏
  feetDog();//喂狗
  //显示
  char disStr[20];//存放显示信息的字符串
  OLED12864_ShowStr(0,0,"Accel:");//OLED显示
  sprintf(disStr,"x:%d,y:%d,z:%d ",Accel[0],Accel[1],Accel[2]);//拼接显示字符串
  OLED12864_ShowStr(0,1,disStr);//OLED显示
  OLED12864_ShowStr(0,2,"Gyro:");//OLED显示
  sprintf(disStr,"x:%d,y:%d,z:%d ",Gyro[0],Gro[1],Gyro[2]);//拼接显示字符串
  OLED12864_ShowStr(0,3,disStr);//OLED显示
}
```

俯仰姿态的解算

- 我们可以通过6050获得三轴角速度和三轴加速度，但仍需经过计算获得俯仰姿态

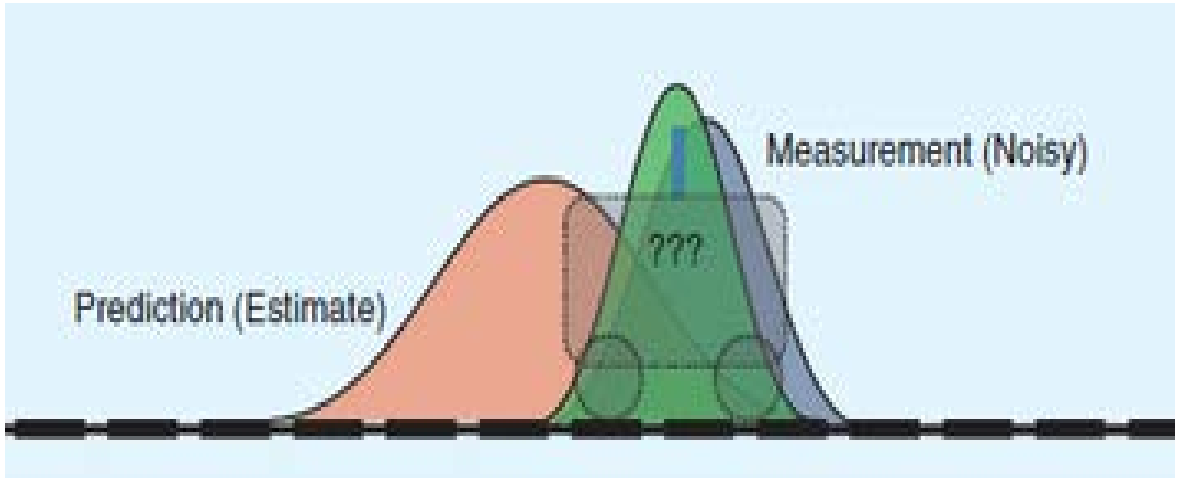
通过加速度计计算俯仰角 ▼



当前角度=上一时刻角度+ y轴角速度*时间

▲ 通过角速度计计算俯仰角

▼ 通过卡曼滤波进行姿态融合



定时器

- 由于通过角速度计计算俯仰角需要乘以时间，所以我们需要一个定时器来给一个确定的时间。在这里，我们设置每5ms进行一次俯仰角的计算。
- 在达到设定时间时，程序会自动跳转到定时器中断服务子程序，运行完中断服务后再跳转回原来跳出的位置继续运行。

```
#include <MsTimer2.h>//定时器库

void setup()
{
  //设置中断，每5ms进入一次中断服务程序 onTimer()
  MsTimer2::set(5, onTimer);
  MsTimer2::start(); //开始计时
}

void loop()
{

}
//中断服务程序，每5ms进入一次
void onTimer()
{
  sei(); //全局中断开，保证下次可继续进入中断
  //。。。。。。
}
```

姿态解算与融合

- Accel与Gyro数组、AccelAngle与Angle变量由于需要在loop与onTimer中使用，所以选择全局变量，全局变量在函数外定义

```
int Accel[3] = {0, 0, 0};//用于存放xyz三轴加速度的全局变量
int Gyro[3] = {0, 0, 0};//用于存放xyz三轴角速度的全局变量
int GyroOffset[3] = {0, 0, 0};//用于存放xyz三轴角速度零偏的全局变量
float AccelAngle, Angle; //存放加速度计算出的角度和滤波后角度的全局变量
```

```
//中断服务程序，每5ms进入一次
void onTimer()
{
  sei(); //全局中断开，保证下次可继续进入中断
  read6050(Accel, Gyro);//读取6050的三轴加速度与角速度的原始数据
  AccelAngle = accelAngle(Accel[0], Accel[2]);//根据加速度计算倾角
  for (int i = 0; i < 3; i++)//循环3次
    Gyro[i] -= GyroOffset[i];//三轴角速度去零偏
  //根据加速度倾角与角速度通过滤波计算真实倾角
  Angle = Kalman_Filter(AccelAngle, -Gyro[1]);
}
```

使用x与z轴加速度计算俯仰角，该函数位于Read6050内

加速度计算的倾角和y轴角速度进行融合计算，该函数位于filter内

一维云台控制

- 通过融合出的俯仰角度，控制云台，使云台在一个维度上保持水平

```
void onTimer()
{
  sei(); //全局中断开，保证下次可继续进入中断
  read6050(Accel, Gyro);//读取6050的三轴加速度与角速度的原始数据
  AccelAngle = accelAngle(Accel[0], Accel[2]);//根据加速度计算倾角
  for (int i = 0; i < 3; i++)//循环3次
    Gyro[i] -= GyroOffset[i];//三轴角速度去零偏
  Angle = Kalman_Filter(AccelAngle, -Gyro[1]);//根据加速度倾角与角速度通过滤波计算真实倾角
  int servoAngle;//舵机角度
  servoAngle = MIDDLE_ANGLE + Angle; // 根据当前倾角计算舵机角度
  myservo.write(servoAngle); // 指定舵机转向的角度
}
```

任务

- 补充代码，完成一维云台平衡控制程序