AIRS 机器狗文档

1.实现机器狗偏离轨迹自动修正方向

在机器猫竞速挑战任务中，由于狗的左右两边脚在长时间走路的过程中容易产生累计误差，使狗不会沿着直线行走，测试行走过程中的误差偏差，使机器狗能够更好的完成长距离快速竞速行走。
首先我们要获取误差，之后我们才能消除误差，而获取误差的方法就是通过陀螺仪获取机器狗在水平方向上偏转的角度，消除误差的方法是通过左右差速使得机器狗转向，从而使其回到原来的方向。
所以，我们要先获取陀螺仪的数据，即 yaw，关于z轴的旋转角，然后将发生的转动进行累加，从而获得机器狗目前的朝向，之后我们要设计好前进，左转，右转的步态，根据目前的状态，调用合适的步态。

具体的代码与注释如下：

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <MPU6050_6Axis_MotionApps20.h>
#define SERVOMIN 180*4  //最小电机占空比计数
#define SERVOMAX 620*4  //最大电机占空比计数
uint8_t devStatus;
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();
MPU6050 mpu;

//前进步态
  char trF[36][8] = {
  61，68，54，61，–26，–39，–13，–26，
  66，61，58，55，–26，–39，–13，–26，
  70，54，61，49，–26，–37，–12，–25，
  73，46，65，43，–24，–35，–11，–23，
  77，39，67，36，–23，–33，–10，–21，
  80，33，70，31，–21，–30，–8，–19，
  83，26，72，25，–19，–26，–6，–15，
  85，19，74，18，–16，–20，–3，–9，
  86，16，76，14，–13，–12，–1，  0，
  87，19，76，16，–9，–11，  3，  2，
  93，24，82，21，–16，–14，–5，–2，
  96，29，85，26，–24，–17，–12，–5，
  95，34，84，30，–28，–20，–17，–7，
  92，39，81，35，–31，–22，–20，–9，
  89，44，79，39，–34，–24，–22，–11，
  84，49，74，44，–36，–25，–24，–12，
  79，54，70，48，–38，–26，–26，–13，
```

```
  72,  59,  65,  52, –39, –26, –26, –13,
  65,  64,  59,  56, –39, –26, –26, –13,
  60,  68,  54,  60, –38, –26, –26, –13,
  52,  72,  48,  62, –37, –25, –25, –12,
  45,  75,  41,  66, –35, –24, –23, –11,
  37,  79,  35,  69, –32, –22, –20,  –9,
  30,  81,  29,  71, –29, –20, –17,  –7,
  24,  83,  24,  73, –25, –17, –14,  –4,
  19,  85,  18,  75, –19, –14,  –7,  –2,
  16,  87,  14,  76, –10, –11,   1,   1,
  20,  90,  17,  79, –12, –12,   1,   0,
  25,  96,  22,  85, –15, –20,  –2,  –9,
  30,  97,  26,  85, –18, –26,  –5, –15,
  35,  94,  31,  83, –20, –30,  –8, –18,
  40,  91,  36,  80, –23, –33, –10, –21,
  45,  86,  40,  76, –24, –36, –11, –24,
  51,  81,  45,  72, –25, –37, –12, –25,
  55,  76,  49,  68, –26, –38, –13, –26,
  60,  70,  53,  62, –26, –39, –13, –26,
};

//左转步态
  char trL[29][8]  = {
 62,  69,  54,  57, –28, –40, –13, –18,
  63,  60,  59,  55, –28, –40, –13, –18,
  64,  51,  62,  53, –28, –37, –12, –18,
  65,  42,  67,  52, –28, –34, –10, –17,
  67,  33,  70,  50, –27, –30,  –8, –17,
  69,  23,  73,  49, –27, –22,  –5, –17,
  70,  18,  75,  46, –27, –14,  –2, –15,
  72,  17,  76,  45, –27,  –7,   2, –14,
  73,  24,  82,  46, –29, –13,  –5, –14,
  72,  29,  85,  48, –30, –17, –14, –14,
  72,  36,  83,  49, –30, –20, –18, –14,
  70,  42,  79,  51, –31, –22, –22, –15,
  68,  49,  74,  52, –31, –25, –24, –15,
  66,  55,  69,  53, –31, –26, –26, –15,
  65,  61,  62,  55, –31, –26, –26, –15,
  62,  67,  55,  56, –31, –26, –26, –15,
  61,  72,  48,  57, –31, –25, –25, –15,
  59,  77,  40,  58, –31, –25, –23, –15,
  57,  81,  32,  60, –31, –22, –19, –14,
  55,  84,  25,  61, –30, –19, –15, –15,
  51,  87,  18,  61, –28, –16,  –7, –14,
  51,  88,  15,  63, –25, –11,   2, –14,
  52,  96,  20,  63, –27, –19,  –1, –14,
  54,  98,  26,  64, –27, –27,  –5, –17,
  55,  97,  31,  63, –27, –32,  –8, –17,
  56,  93,  37,  61, –27, –36, –10, –17,
```

```
  58,  87,  42,  60, -28, -39, -12, -17,
  60,  79,  48,  59, -28, -40, -13, -18,
  61,  70,  53,  58, -28, -40, -13, -18,
};
//右转步态
  char trR[29][8]  = {
  61,  64,  55,  61, -26, -31, -15, -26,
  67,  62,  56,  54, -26, -31, -15, -26,
  72,  60,  57,  47, -25, -31, -15, -24,
  77,  59,  58,  39, -25, -31, -15, -22,
  81,  56,  60,  31, -22, -30, -14, -19,
  84,  54,  61,  24, -19, -30, -15, -14,
  87,  51,  61,  16, -16, -27, -14,  -5,
  88,  51,  63,  15, -11, -25, -14,   2,
  96,  52,  63,  21, -19, -27, -14,  -2,
  98,  54,  64,  26, -27, -27, -17,  -5,
  97,  55,  63,  32, -32, -27, -17,  -8,
  93,  57,  61,  38, -36, -27, -17, -10,
  87,  58,  60,  44, -39, -28, -17, -12,
  79,  60,  59,  49, -40, -28, -18, -13,
  70,  62,  58,  54, -40, -28, -18, -13,
  62,  63,  55,  59, -40, -28, -18, -13,
  52,  64,  54,  62, -37, -28, -18, -12,
  43,  65,  52,  67, -35, -28, -17, -10,
  34,  67,  51,  70, -31, -27, -17,  -8,
  26,  69,  49,  73, -25, -27, -17,  -5,
  18,  70,  46,  75, -15, -27, -15,  -2,
  17,  72,  45,  76,  -7, -27, -14,   2,
  22,  73,  46,  82, -12, -29, -14,  -5,
  28,  72,  48,  85, -16, -30, -14, -14,
  35,  72,  49,  83, -19, -30, -14, -18,
  41,  70,  50,  79, -22, -31, -14, -22,
  48,  68,  52,  74, -25, -31, -15, -24,
  54,  66,  53,  69, -26, -31, -15, -26,
  60,  65,  54,  62, -26, -31, -15, -26,
 };

//校准电机误差
char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  8,  8, 8,  -6, 0,  3};
//关节映射表
byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };
//旋转方向
int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
```

```
                                -1, 1, 1, -1
                               };
//角度转化为占空比计数时需要的偏移量
int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };

 //将角度转化为占空比计数
int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) * 7.04  * rotationDirections[i] +
angle * 7.04 * rotationDirections[i];
  return int(p);
}
volatile bool mpuInterrupt = false;     // 表示mpu interrupt 引脚是否被拉高
void dmpDataReady() {
  mpuInterrupt = true;  //表示发生了中断
}

void setup() {
  //配置串口
  Serial.begin(57600);
  //配置电机pwm
  pwm.begin();
  pwm.setPWMFreq(240);
  //初始化陀螺仪
  mpu.initialize();
    do {
    //获取dmp状态
    devStatus = mpu.dmpInitialize();
    delay(500);
    //设置陀螺仪offset
    mpu.setZAccelOffset(985);
    mpu.setXGyroOffset(39);
    mpu.setYGyroOffset(13);
    mpu.setZGyroOffset(-6);

    if (devStatus == 0) {
      mpu.setDMPEnabled(true);
      attachInterrupt(0, dmpDataReady, RISING);

    }

  } while (devStatus);//如果dmp没有成功打开，则一直尝试
}

float GR=131.0;//陀螺仪 比例系数
```

```
int16_t ax=0,ay=0,az=0,gx=0,gy=0,gz=0;//用来存储getMotion6函数读取到的数据
float sum=0;//用来累加z轴旋转角偏移量
int t=0;//标志目前机器人是否发生偏移，以及偏移方向
float gzz;

void loop() {

    //设置200作为阀值，避免过于频繁的转向
    if(sum>200){
      t=0;
      }
    else if(sum<-200){
      t=1;
      }
    else{
       t=2;
     }

     //控制部分
     if(t==2){//前进
        for(int i=0;i<=35;i++){
   for(int j=0;j<=7;j++){
     //根据步态数据设置每个关节的转角
     pwm.setPWM(pins[8+j],0,S2P(   (int(trF[i][j]))  ,8+j   ));

   }
   //每更新完一次所有关节的角度，读取一次陀螺仪数据
  mpu.getMotion6(&ax,&ay,&az,&gx,&gy,&gz);
  gzz=gz/GR;//关于z轴的旋转角，即水平的转向
  if(abs(gzz)>0.1)+//小于0.1的值被当作误差忽略掉
  {
     sum+=gzz;//每当方向发生偏移，sum的值都会增加
     Serial.print("sum: ");Serial.println(sum); //显示目前的偏移量
   }
   }
     }
    else if(t==1){//左转
       for(int i=0;i<=28;i++){
   for(int j=0;j<=7;j++){

     pwm.setPWM(pins[8+j],0,S2P(   (int(trL[i][j]))  ,8+j   ));

   }
   mpu.getMotion6(&ax,&ay,&az,&gx,&gy,&gz);
  gzz=gz/GR;
  if(abs(gzz)>0.1){
     sum+=gzz;
     Serial.print("sum: ");Serial.println(sum);
   }
```

```
    }
      }

       else if(t==0){//右转
      for(int i=0;i<=28;i++){
    for(int j=0;j<=7;j++){

      pwm.setPWM(pins[8+j],0,S2P(   (int(trR[i][j]))  ,8+j  ));
    }
    mpu.getMotion6(&ax,&ay,&az,&gx,&gy,&gz);
   gzz=gz/GR;
   if(abs(gzz)>0.1){
      sum+=gzz;
      Serial.print("sum: ");Serial.println(sum);
    }
    }
      }
}
```

2.实现机器狗动态平衡

机器狗的动态平衡就是实现无论机器狗站在任何一种倾角的地面上，机器狗都能通过调节自己步态的变化，使得自己的身体始终保持水平。
我们可以把水平状态理解为我们要达到的目标，即倾斜角为0，而任何方向的倾角都是误差，我们要做的就是发现误差，得到误差，消除误差，这就与我们上一个任务非常的相似了，而且我们获得误差的方法同样是通过陀螺仪，不过不同的是，这一次我们需要用到pid控制算法。

下面是一些关于PID的讲解。分别介绍了P,I,D在调节误差的过程中分别起到了什么作用

PID(Proportional Integral Derivative)控制是最早发展起来的控制策略之一，由于其算法简单、鲁棒性好和可靠性高，被广泛应用于工业过程控制，尤其适用于可建立精确数学模型的确定性控制系统。在工程实际中，应用最为广泛的调节器控制规律为比例、积分、微分控制，简称PID控制，又称PID调节，它实际上是一种算法。PID控制器问世至今已有近70年历史，它以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。当被控对象的结构和参数不能完全掌握，或得不到精确的数学模型时，控制理论的其它技术难以采用时，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用PID控制技术最为方便。即当我们不完全了解一个系统和被控对象，或不能通过有效的测量手段来获得系统参数时，最适合用PID控制技术。PID控制，实际中也有PI和PD控制。PID控制器就是根据系统的误差，利用比例、积分、微分计算出控制量进行控制的。

1. 比例环节

成比例地反映控制系统的偏差信号e(t)，偏差一旦产生，控制器立即产生控制作用，以减小偏差。当仅有比例控制时系统输出存在稳态误差（Steady-state error）。

P参数越小比例作用越强，动态响应越快，消除误差的能力越强。但实际系统是有惯性的，控制输出变化后，实际y(t)值变化还需等待一段时间才会缓慢变化。由于实际系统是有惯性的，比例作用不宜太强，比例作用太强会引起系统振荡不稳定。P参数的大小应在以上定量计算的基础上根据系统响应情况，现场调试决定，通常将P参数由大向小调，以能达到最快响应又无超调(或无大的超调)为最佳参数。

优点:调整系统的开环比例系数，提高系统的稳态精度，减低系统的惰性，加快响应速度。

缺点:仅用P控制器,过大的开环比例系数不仅会使系统的超调量增大，而且会使系统稳定裕度变小，甚至不稳定。

2. 积分环节

控制器的输出与输入误差信号的积分成正比关系。主要用于消除静差，提高系统的无差度。积分作用的强弱取决于积分时间常数T,T越大，积分作用越弱，反之则越强。

比例作用的输出与误差的大小成正比，误差越大，输出越大，误差越小，输出越小，误差为零，输出为零。由于没有误差时输出为零，因此比例调节不可能完全消除误差，不可能使被控的PV值达到给定值。必须存在一个稳定的误差，以维持一个稳定的输出，才能使系统的PV值保持稳定。这就是通常所说的比例作用是有差调节，是有静差的，加强比例作用只能减少静差，不能消除静差(静差：即静态误差，也称稳态误差)。

为了消除静差必须引入积分作用，积分作用可以消除静差，以使被控的y(t)值最后与给定值一致。引进积分作用的目的也就是为了消除静差，使y(t)值达到给定值，并保持一致。

积分作用消除静差的原理是，只要有误差存在，就对误差进行积分，使输出继续增大或减小，一直到误差为零，积分停止，输出不再变化，系统的PV值保持稳定，y(t)值等于u(t)值，达到无差调节的效果。

但由于实际系统是有惯性的，输出变化后，y(t)值不会马上变化，须等待一段时间才缓慢变化，因此积分的快慢必须与实际系统的惯性相匹配，惯性大、积分作用就应该弱，积分时间I就应该大些，反之而然。如果积分作用太强，积分输出变化过快，就会引起积分过头的现象，产生积分超调和振荡。通常I参数也是由大往小调，即积分作用由小往大调，观察系统响应以能达到快速消除误差，达到给定值，又不引起振荡为准。

对一个自动控制系统，如果在进入稳态后存在稳态误差，则称这个控制系统是有稳态误差的或简称有差系统（System with Steady-state Error）。为了消除稳态误差，在控制器中必须引入“积分项”。积分项对误差取决于时间的积分，随着时间的增加，积分项会增大。这样，即便误差很小，积分项也会随着时间的增加而加大，它推动控制器的输出增大使稳态误差进一步减小，直到等于零。因此，比例+积分(PI)控制器，可以使系统在进入稳态后无稳态误差。PI控制器不但保持了积分控制器消除稳态误差的“记忆功能”，而且克服了单独使用积分控制消除误差时反应不灵敏的缺点。

优点：消除稳态误差。

缺点：积分控制器的加入会影响系统的稳定性，使系统的稳定裕度减小。

3. 微分环节

反映偏差信号的变化趋势，并能在偏差信号变得太大之前，在系统中引入一个有效的早期修正信号，从而加快系统的动作速度，减少调节时间。在微分控制中，控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。

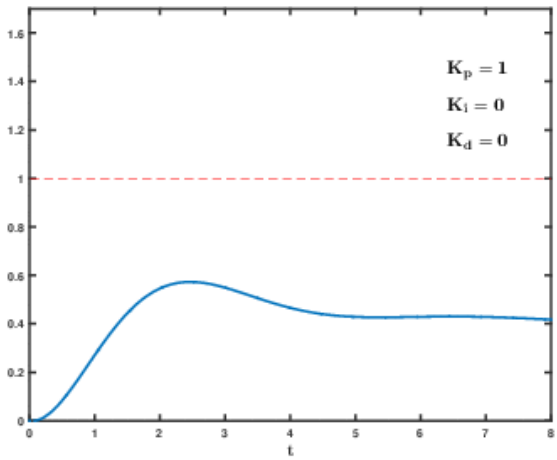
前面已经分析过，不论比例调节作用，还是积分调节作用都是建立在产生误差后才进行调节以消除误差，都是事后调节，因此这种调节对稳态来说是无差的，对动态来说肯定是有差的，因为对于负载变化或给定值变化所产生的扰动，必须等待产生误差以后，然后再来慢慢调节予以消除。

但一般的控制系统，不仅对稳定控制有要求，而且对动态指标也有要求，通常都要求负载变化或给定调整等引起扰动后，恢复到稳态的速度要快，因此光有比例和积分调节作用还不能完全满足要求，必须引入微分作用。比例作用和积分作用是事后调节(即发生误差后才进行调节)，而微分作用则是事前预防控制，即一发现y(t)有变大或变小的趋势，马上就输出一个阻止其变化的控制信号，以防止出现过冲或超调等。
D越大，微分作用越强，D越小，微分作用越弱。系统调试时通常把D从小往大调，具体参数由试验决定。

如：由于给定值调整或负载扰动引起y(t)变化，比例作用和微分作用一定等到y(t)值变化后才进行调节，并且误差小时，产生的比例和积分调节作用也小，纠正误差的能力也小，误差大时，产生的比例和积分作用才增大。因为是事后调节动态指标不会很理想。而微分作用可以在产生误差之前一发现有产生误差的趋势就开始调节，是提前控制，所以及时性更好，可以最大限度地减少动态误差，使整体效果更好。但微分作用只能作为比例和积分控制的一种补充，不能起主导作用，微分作用不能太强，太强也会引起系统不稳定，产生振荡，微分作用只能在P和I调好后再由小往大调，一点一点试着加上去。

自动控制系统在克服误差的调节过程中可能会出现振荡甚至失稳。其原因是由于存在有较大惯性组件（环节）或有滞后(delay)组件，具有抑制误差的作用，其变化总是落后于误差的变化。解决的办法是使抑制误差的作用的变化“超前”，即在误差接近零时，抑制误差的作用就应该是零。这就是说，在控制器中仅引入“比例”项往往是不够的，比例项的作用仅是放大误差的幅值，而目前需要增加的是“微分项”，它能预测误差变化的趋势。这样，具有比例+微分的控制器，就能够提前使抑制误差的控制作用等于零，甚至为负值，从而避免了被控量的严重超调。所以对有较大惯性或滞后的被控对象，比例+微分(PD)控制器能改善系统在调节过程中的动态特性。PD控制只在动态过程中才起作用，对恒定稳态情况起阻断作用。因此，微分控制在任何情况下都不能单独使用。

优点：使系统的响应速度变快，超调减小，振荡减轻，对动态过程有“预测”作用。



这张动图清晰的展示了通过调节kp,ki,kd的大小，整个误差调节的过程发生的变化。

那既然pid有这么多优点，为什么上一个偏离轨迹的任务不用pid算法呢。 因为pid算法最后得到的pid_value会被用来当作下一次的调节量，简单来说就是，误差大，调整的就多，误差小，调整的就少。pid就是找到任意时刻那个最合适的调整值，但是在上个任务中，如果发现机器狗向

左偏移，那么下一个循环执行的步态就是右转步态，无论目前偏移了多少度，执行的命令都是一致的，所以即使我们求出了pid值，接下来进行的操作仍然是一样的，所以没有必要进行无比要的运算。 那么为什么没有发生左右来回振荡的情况呢，因为我们在目标周围设置的可以容忍的误差，我们的目标是一个角度范围，而不是一个确定值，所以即使最后执行的右转步态使得机器狗稍微向左偏移，接下来程序仍然会判断目前没有偏移，执行前进指令。

那么使用pid算法需要设置哪些变量呢？ 首先P,I,D三项与其参数kp,ki,kd是必需的，其次需要误差error，和为了计算微分而保留的之前的误差previous_error，当然还有最后用来存储结果pid_value，以上变量除kp,ki,kd的初始值均设为0, kp,ki,kd要随着测试根据上文提到的规律不断调整，这里因为代码已经调试过，可直接使用下文实例代码提供的数值。

```
//控制前后平衡的pid参数
float error = 0, P = 0, I = 0, D = 0, PID_value = 0;
float previous_error = 0;
float Kp = 0.1, Ki = 0.00001, Kd = 0.1;
```

在设置好变量之后，编写实现算法的代码，首先跟error赋值，也就是目前陀螺仪的倾角（这里只考虑前后倾角，不考虑左右倾角），使P等于误差，I等于I加误差，D等于误差减上一步的误差，也就是error–previous_error， 分别体现了比例，积分和微分，之后PID分别乘上参数kp，ki，kd相加得到新的pid_value,在运算都执行完之后，更新previous_error的值。

```
error= RollPitchDeviation[1];//获取误差
P = error; //比例
I = I + error; //积分
D = error - previous_error; //微分
PID_value = (Kp * P) + (Ki * I) + (Kd * D); //P,I,D分别乘上比例系数得到结果
previous_error = error; //更新previous_error
```

处理左右倾角的方法和前后倾角相同，不过要使用另一组PID变量，kp,ki,kd的值也要做相应的调整，因为左右腿和前后腿之间的距离差不同，调整相同的量会对左右倾角产生更大的影响。

现在我们知道了机器狗的倾角，也知道了如何获得合适的调整量，那么我们如何实际的来调整机器狗的倾角呢。我们使用的方法很简单，改变前后腿（或者左右腿）的高度差，具体的做法是上下两个关节进行反向的旋转，下关节的旋转角度为上关节的两倍，这样机器狗足部末端就能在竖直方向上运动。

为得到对任一关节的准确控制，需要能通过关节号判断其所处的上下前后左右，从而达到批量控制，具体判断的方法如下：

bool leftQ = (i – 1) % 4 > 1 ? true : false; 判断关节左右

bool frontQ = i % 4 < 2 ? true : false; 判断关节前后

bool upperQ = i / 4 < 3 ? true : false; 判断关节上下

下面是根据上面的思路调整每一个关节的角度的方法，第一行调整左右倾角，第二行调整前后倾角。

```
currentAdjust[j] -=   radPerDeg *adaptiveCoefficient(j+8, 0) *
                      (((j+7) % 4 > 1) ? ( ((j+8) / 4 < 3) ? PID_value2 : 2*PID_value2) : ( ((j+8) / 4 < 3) ? -PID_value2 : -2*PID_value2));

currentAdjust[j] -=   radPerDeg *adaptiveCoefficient(j+8, 1) *
                      (((j+8) % 4 < 2) ? ( ((j+8) / 4 < 3) ? PID_value : -2*PID_value) : ( ((j+8) / 4 < 3) ? -PID_value : -2*PID_value));
```

另外还有一点需要注意的是需要给舵机可以调整的角度设置一个上限，否则当机器狗偏转的角度过大时，为了使陀螺仪恢复水平，舵机有可能剧烈旋转，导致整个机器失控，我设置的最大调整角度为90度，一般如果到达这个角度还不能保持平衡，那就已经超过了机器狗可以处理的倾斜角度上限了。

整体的处理思路为，当任意调整角度超过90度时，所有舵机复位成站立姿态，延时一定时间之后再重新尝试保持平衡，注意复位之后一定要跳出主循环，否则延时之后重新开始平衡时会从出现问题的下一个舵机开始进行处理，而不是完整的一组8个舵机。

完整的代码和注释如下：

```
#include <I2Cdev.h>
#include <MPU6050_6Axis_MotionApps20.h>


#define PACKET_SIZE 42
#define OVERFLOW_THRESHOLD 128
#define SERVOMIN 180*4
#define SERVOMAX 620*4
#define PWM_RANGE 620*4-180*4
#define HISTORY 2



Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();
MPU6050 mpu;


int8_t lag = 0;
float ypr[3]; // [yaw, pitch, roll]存储陀螺仪得到的关于xyz轴的旋转角
float yprLag[HISTORY][3];
float radPerDeg = M_PI / 180;
float degPerRad = 180 / M_PI;
uint8_t mpuIntStatus;   // 存储mpu中断状态
uint8_t devStatus;      // 判断进行的操作是否成功 (0 = success, !0 = error)
uint16_t packetSize;    // 期望的dmp包尺寸 (默认为 42 bytes)
uint16_t fifoCount;     //目前fifo缓存的bytes数量
uint8_t fifoBuffer[PACKET_SIZE];
float currentAdjust[8] = {};   //根据目前的姿态每个关节需要调整的角度
Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector



//处理陀螺仪数据相关函数
template <typename T> int8_t sign(T val) {
  return (T(0) < val) - (val < T(0));
}

volatile bool mpuInterrupt = false;
void dmpDataReady() {
  mpuInterrupt = true;
}


unsigned long usedTime = 0;
void getFIFO() {//get FIFO only without further processing
  while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
  mpu.getFIFOBytes(fifoBuffer, packetSize);
  fifoCount -= packetSize;
}


void getYPR() {
  if (mpuInterrupt || fifoCount >= packetSize)
  {
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();
    fifoCount = mpu.getFIFOCount();
    if ((mpuIntStatus & 0x10) || fifoCount > OVERFLOW_THRESHOLD) { //1024) {
      mpu.resetFIFO();
      lag = (lag - 1 + HISTORY) % HISTORY;
    }
    else if (mpuIntStatus & 0x02) {
      getFIFO();
      mpu.dmpGetQuaternion(&q, fifoBuffer);
      mpu.dmpGetGravity(&gravity, &q);
      mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
      for (byte g = 0; g < 3; g++)
        ypr[g] *= degPerRad;
      for (byte g =0; g < 3; g++) {
        yprLag[lag][g] = ypr[g];
        ypr[g] = yprLag[(lag - 1 + HISTORY) % HISTORY][g] ;
      }
      lag = (lag + 1) % HISTORY;
    }
  }
}

//初始站立步态
char balance[1][8]  = {
  30, 30,  30,  30, 30, 30, 30, 30,
 };

//电机校准误差
char cal[16]= {0,  0,  0,  0,  0,  0,  0,  3,  4,  -2,  0, 4,  6, 0,  0};

//关节对映表
byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };
//旋转方向
int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };

//计算占空比计数需要的偏移量
int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };

//将角度转化为占空比计数
int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) * 7.04  * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}


void setup() {
  //I2C部分
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24;
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
  #endif

  //SERIAL部分
  Serial.begin(115200);
  Serial.setTimeout(10);
  while (!Serial);
  while (Serial.available() && Serial.read());
  delay(100);


  //MPU部分
  mpu.initialize();
  delay(500);
  do{

    devStatus = mpu.dmpInitialize();
    delay(500);
    mpu.setZAccelOffset(985);
    mpu.setXGyroOffset(39);
    mpu.setYGyroOffset(13);
    mpu.setZGyroOffset(-6);
    if (devStatus == 0) {
      mpu.setDMPEnabled(true);
      attachInterrupt(0, dmpDataReady, RISING);
      mpuIntStatus = mpu.getIntStatus();
      packetSize = mpu.dmpGetFIFOPacketSize();
    }
  } while (devStatus);


  //PWM部分
  pwm.begin();
  pwm.setPWMFreq(240);

  //姿势初始化部分
  for (byte a = 0; a < 8; a++)
  currentAdjust[a] = 0;//将初始调整角度全部置零
```

```
    //将机器狗设置为站立姿势
    for(int j=0;j<=7;j++){
      pwm.setPWM(pins[8+j],0,82P((int(balance[0][j])) ,8+j));
    }
delay(500);
}


inline int8_t adaptiveCoefficient(byte idx, byte para) {
  return EEPROM.read(160 + idx * 2 + para);//从eeprom读取设置调整角度需要的参数
}


//控制前后平衡的pid参数
float error = 0, P = 0, I = 0, D = 0, PID_value = 0;
float previous_error = 0;
float Kp = 0.1, Ki = 0.00001, Kd = 0.1;

//控制左右平衡的pid参数
float error2 = 0, P2 = 0, I2 = 0, D2 = 0, PID_value2 = 0;
float previous_error2 = 0;
float Kp2 = 0.01, Ki2 = 0.00001, Kd2 = 0.0001;      //kp，ki明显小于上面的kp，ki，因为左右腿之间的距离远小于前后腿，较小的高度差就能导致很大的倾斜角


float RollPitchDeviation[2];

void loop() {

  getYPR();//获取陀螺仪数据

  for (byte i = 0; i < 2; i++) {
    RollPitchDeviation[i] = ypr[2 - i];
    RollPitchDeviation[i] = sign(ypr[2 - i]) * max(fabs(RollPitchDeviation[i]) - 2, 0);//滤除过小的角度，防止调整过于频繁
  }


  error= RollPitchDeviation[1];//获取误差
  P = error; //比例
  I = I + error; //积分
  D = error - previous_error; //微分
  PID_value = (Kp * P) + (Ki * I) + (Kd * D); //P,I,D分别乘上比例系数得到结果
  previous_error = error; //更新previous_error

  //同上
  error2= RollPitchDeviation[0];
  P2 = error2;
  I2 = I2 + error2;
  D2 = error2 - previous_error2;
  PID_value2 = (Kp2 * P2) + (Ki2 * I2) + (Kd2 * D2);
  previous_error2 = error2;

  for(int i=0;i<=0;i++){
    for(int j=0;j<=7;j++){

      //bool leftQ = (i - 1 ) % 4 > 1 ? true : false;  判断关节左右
      //bool frontQ = i % 4 < 2 ? true : false;   判断关节前后
      //bool upperQ = i / 4 < 3 ? true : false;   判断关节上下
      //当下关节旋转角度是上关节的两倍时，脚部可以刚好在竖直方向上移动
      currentAdjust[j] -=   radPerDeg *adaptiveCoefficient(j+8, 0) *(((j+7) % 4 > 1) ? ( ((j+8) / 4 < 3) ? PID_value2 : 2*PID_value2) : ( ((j+8) / 4 < 3) ? -PID_value2 : -2*PID_value2));
      currentAdjust[j] -=   radPerDeg *adaptiveCoefficient(j+8, 1) *(((j+8) % 4 < 2) ? ( ((j+8) / 4 < 3) ? PID_value : -2*PID_value) : ( ((j+8) / 4 < 3) ? -PID_value : -2*PID_value));

      if(abs(currentAdjust[j])>90)
      {
        for(int num0=0;num0<=7;num0++){
          currentAdjust[num0] =0;

        pwm.setPWM(pins[8+num0],0,82P((int(balance[i][num0]))+currentAdjust[num0],8+num0));}
        delay(3000);
        break;}
      pwm.setPWM(pins[8+j],0,82P((int(balance[i][j]))+currentAdjust[j],8+j));//切换站立的步态转角再加上处理出来的需要调整的角度

    }}


}
```

3.设计机器狗步态使其跳舞

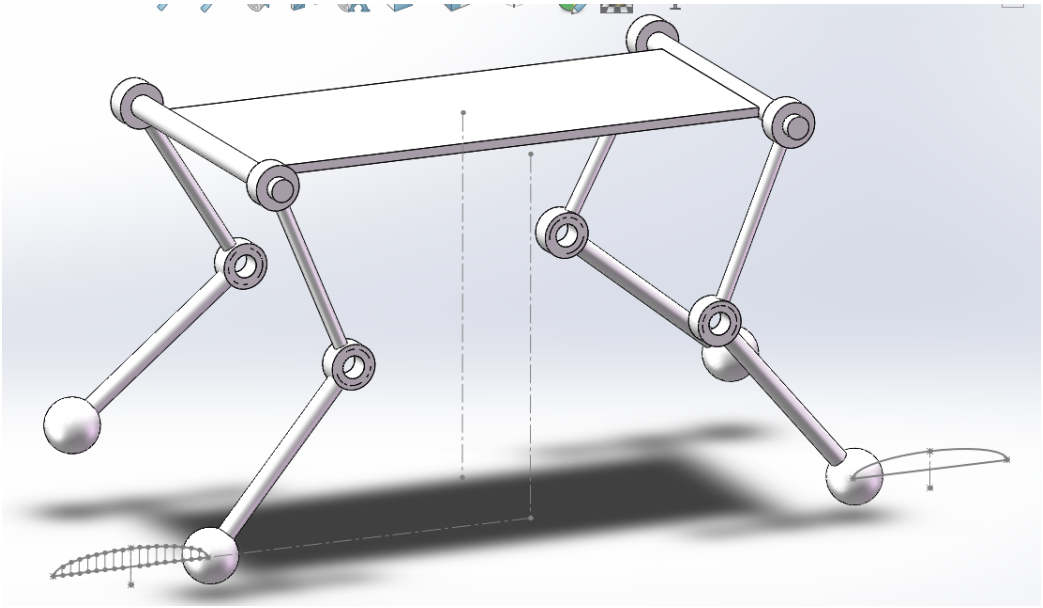
正运动学，就是通过我们驱动的舵机角度，求取小猫机器人的足端坐标。主要用于确定小猫足端的运动范围。

逆运动学，已知小猫的足端坐标，反过来求取关节舵机的转角。在多数情况下，我们通常用运动学逆解，首先将足端的运动轨迹曲线规划好，然后将曲线离散化处理，通常就是在上面均匀的找出一组点，来代替曲线，然后找到每个点的坐标，根据坐标最后算出舵机转角。

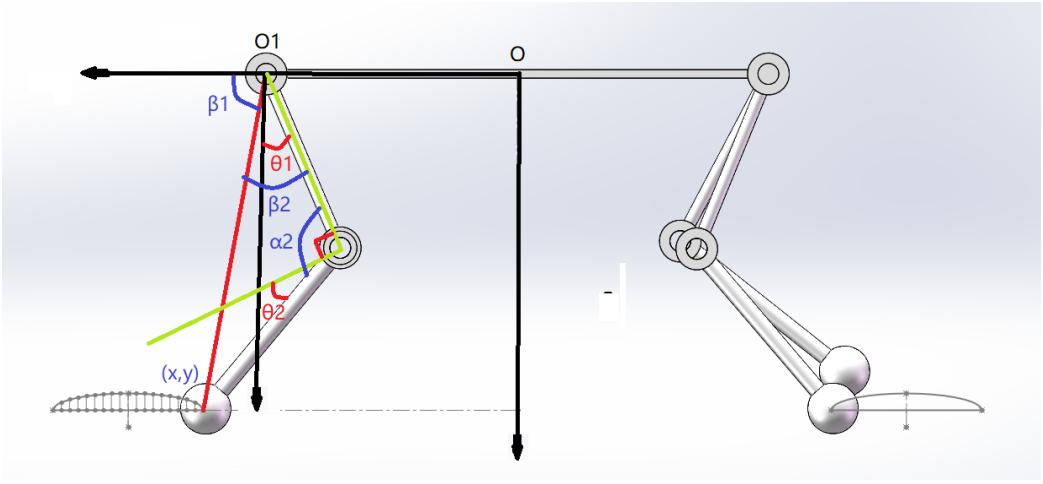
为了得到逆解的准确舵机角度数据，我将这只小猫的运动3维简图做了出来，方便我们学习。

可以看到小猫的足端中心的轨迹已经用草图画了出来–下方的那两个灰色闭环曲线。

下图是简化了的运动简图：

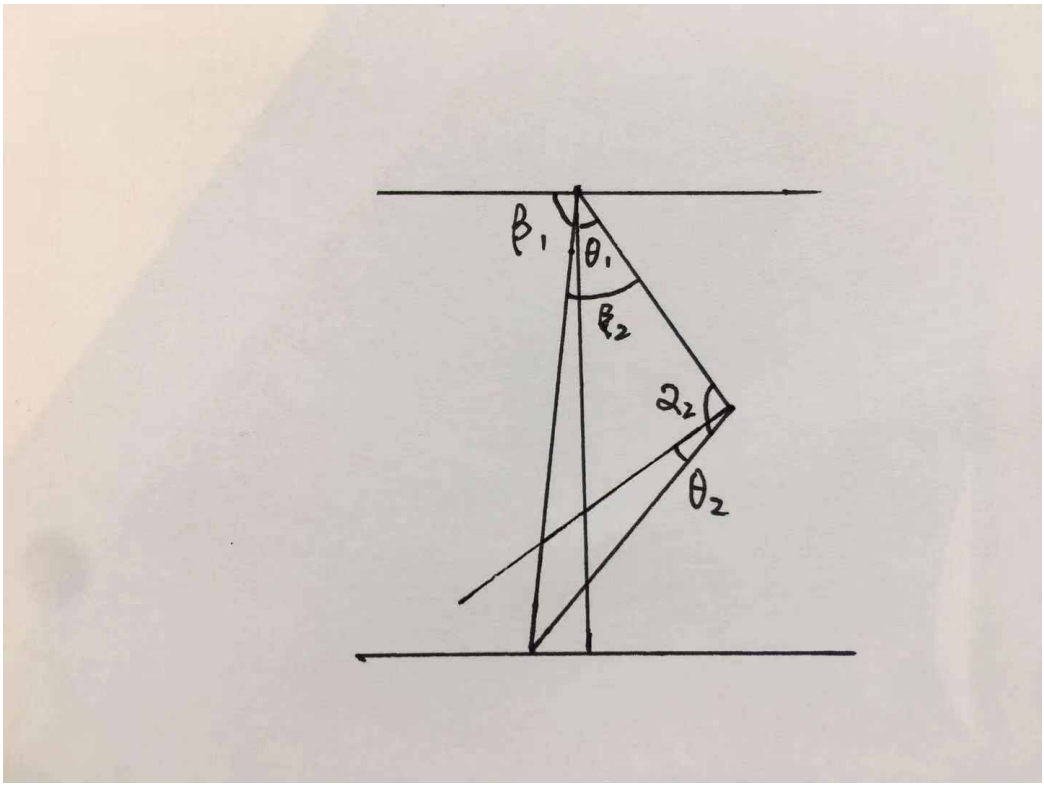


我们拿一条腿（左前腿）举例，



可以看到x轴坐标在大于零和小于零时，不同角之间的关系会发生变化，所以在设计程序的时候，要使用两种不同的计算公式。

x大于零的情况：



以下是θ1的计算方法（当我们把角度通过反三角函数计算出来以后，会发现是弧度值，因此需要将进行转换）：

$$\beta_1 = \arctan(y/x)$$

$$\beta_1 = \frac{\beta_1 \times 180}{\pi}$$

$$\beta_2 = \arccos\left(\frac{dt^2 + x^2 + y^2 - xt^2}{2 \times dt \times \sqrt{x^2 + y^2}}\right)$$

$$\beta_2 = \frac{\beta_2 \times 180}{\pi}$$

$$\theta_1 = \beta_1 + \beta_2 - 90$$

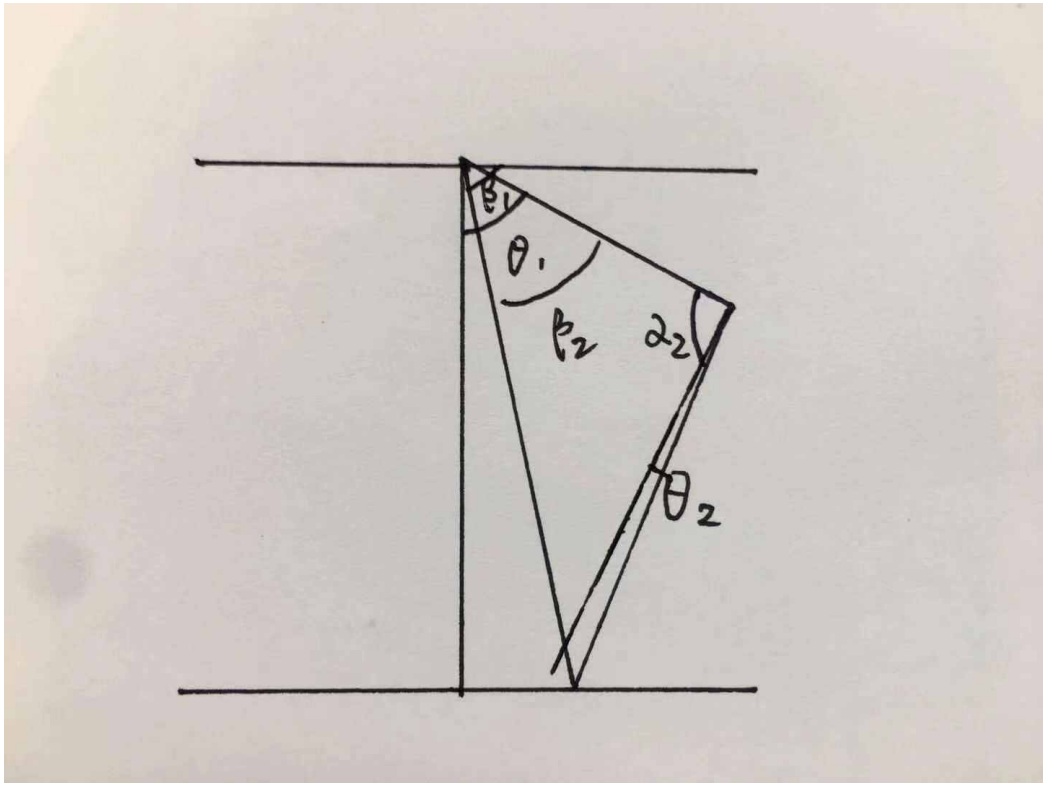
以下是θ2的计算方法：

$$\alpha_2 = \arccos\left(\frac{dt^2 + xt^2 - (x^2 + y^2)}{2 \times dt \times xt}\right)$$

$$\alpha_2 = \frac{\alpha_2 \times 180}{\pi}$$

$$\theta_2 = \alpha_2 - 90$$

x小于零的情况：



以下是θ1的计算方法（当我们把角度通过反三角函数计算出来以后，会发现是弧度值，因此需要将进行转换）：

$$\beta_1 = \arctan(y/x)$$

$$\beta_1 = \frac{\beta_1 \times 180}{\pi}$$

$$\beta_2 = \arccos\left(\frac{dt^2 + x^2 + y^2 - xt^2}{2 \times dt \times \sqrt{x^2 + y^2}}\right)$$

$$\beta_2 = \frac{\beta_2 \times 180}{\pi}$$

$$\theta_1 = 90 + \beta_2 - \beta_1$$

以下是θ2的计算方法：

$$\alpha_2 = \arccos\left(\frac{dt^2 + xt^2 - (x^2 + y^2)}{2 \times dt \times xt}\right)$$

$$\alpha_2 = \frac{\alpha_2 \times 180}{\pi}$$

$$\theta_2 = \alpha_2 - 90$$

以下是运算实现的具体代码：

```
#define dt 45.0
#define xt 50.0

#define pi 3.1415

float angle[40][2]={

  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
```

```
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
  0.0,0.0,
};

float xy[40][2]={
  //1
  0.0,60.0,
  0.0,58.5,
  0.0,57.0,
  0.0,55.5,
  0.0,54.0,
  0.0,52.5,
  0.0,51.0,
  0.0,49.5,
  0.0,48.0,
  0.0,46.5,
  0.0,45.0,
  0.0,43.5,
  0.0,42.0,
  0.0,40.5,
  0.0,39.0,
  0.0,37.5,
  0.0,36.0,
  0.0,34.5,
  0.0,33.0,
  0.0,31.5,
  0.0,30.0,
  //2
  0.0,31.5,
  0.0,33.0,
  0.0,34.5,
  0.0,36.0,
  0.0,37.5,
  0.0,39.0,
  0.0,40.5,
  0.0,42.0,
  0.0,43.5,
  0.0,45.0,
  0.0,46.5,
  0.0,48.0,
  0.0,49.5,
  0.0,51.0,
  0.0,52.5,
  0.0,54.0,
  0.0,55.5,
```

```
  0.0,57.0,
  0.0,58.5,

};

void sf(){
  float xb2=0.0, a=0.0, a1=0.0, a2=0.0;
  float b=0.0;
 for(int i=0;i<=39;i++)
  {
    xy[i][1]+=30;

  }
  for(int i=0;i<=39;i++)
  {
    if(xy[i][0]>=0){
    xb2=xy[i][0]*xy[i][0]+xy[i][1]*xy[i][1];
    a1=acos((dt*dt+xb2-xt*xt)/(2*sqrt(xb2)*dt))*180/pi;
    a2=atan(xy[i][1]/xy[i][0])*180/pi;
    a=a1+a2-90.0;
    b=acos((dt*dt+xt*xt-xb2)/(2*dt*xt))*180/pi;
    b=b-90.0;
    delay(3);
    angle[i][0]=round(a);
    angle[i][1]=round(b);
  }
  else{
    xy[i][0]=-xy[i][0];
    xb2=xy[i][0]*xy[i][0]+xy[i][1]*xy[i][1];
    a1=acos((dt*dt+xb2-xt*xt)/(2*sqrt(xb2)*dt))*180/pi;
    a2=atan(xy[i][1]/xy[i][0])*180/pi;
    a=90.0-a2+a1;
    b=acos((dt*dt+xt*xt-xb2)/(2*dt*xt))*180/pi;
    b=b-90.0;
    delay(3);
    angle[i][0]=round(a);
    angle[i][1]=round(b);
  }
}
}
void sc(){
  for(int i=0;i<=39;i++){
    Serial.print(int(angle[i][0]),DEC);
    Serial.print(",");
    Serial.print(int(angle[i][0]),DEC);
    Serial.print(",");
    Serial.print(int(angle[i][0]),DEC);
    Serial.print(",");
    Serial.print(int(angle[i][0]),DEC);
```

```
    Serial.print(",");
    Serial.print(int(angle[i][1]),DEC);
    Serial.print(",");
    Serial.print(int(angle[i][1]),DEC);
    Serial.print(",");
    Serial.print(int(angle[i][1]),DEC);
    Serial.print(",");
    Serial.print(int(angle[i][1]),DEC);
    Serial.print(",");

    Serial.println();
  }
}

void setup() {
  Serial.begin(57600);
  sf();
  sc();
}
void loop() {
}
```

之后就可以利用机器狗足部末端的xy坐标设计连贯的步态，而不用具体去考虑每一个舵机的转角，当我们设计好能够顺利运行的多组步态，就可以串联多组步态，让机器狗实现舞蹈。

通过循环语句可以实现步态数组的复用，使用不同的组合方式，可以让用相同的步态执行不同的舞蹈。

以下是我们使用的步态数据：

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <MPU6050_6Axis_MotionApps20.h>
#include <avr/pgmspace.h>


#define SERVOMIN 180*4
#define SERVOMAX 620*4
#define PWM_RANGE 620*4-180*4
#define DOF 16
uint8_t devStatus;


Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();
MPU6050 mpu;


//step
  const  char stp[40][8] PROGMEM = {
40,40,40,40,14,14,14,14,
42,42,42,42,11,11,11,11,
43,43,43,43,8,8,8,8,
45,45,45,45,6,6,6,6,
46,46,46,46,3,3,3,3,
48,48,48,48,0,0,0,0,
49,49,49,49,-2,-2,-2,-2,
51,51,51,51,-5,-5,-5,-5,
52,52,52,52,-7,-7,-7,-7,
53,53,53,53,-9,-9,-9,-9,
55,55,55,55,-12,-12,-12,-12,
56,56,56,56,-14,-14,-14,-14,
57,57,57,57,-16,-16,-16,-16,
59,59,59,59,-19,-19,-19,-19,
60,60,60,60,-21,-21,-21,-21,
```

```
61,61,61,61,-23,-23,-23,-23,
62,62,62,62,-25,-25,-25,-25,
64,64,64,64,-27,-27,-27,-27,
65,65,65,65,-30,-30,-30,-30,
66,66,66,66,-32,-32,-32,-32,
68,68,68,68,-34,-34,-34,-34,
66,66,66,66,-32,-32,-32,-32,
65,65,65,65,-30,-30,-30,-30,
64,64,64,64,-27,-27,-27,-27,
62,62,62,62,-25,-25,-25,-25,
61,61,61,61,-23,-23,-23,-23,
60,60,60,60,-21,-21,-21,-21,
59,59,59,59,-19,-19,-19,-19,
57,57,57,57,-16,-16,-16,-16,
56,56,56,56,-14,-14,-14,-14,
55,55,55,55,-12,-12,-12,-12,
53,53,53,53,-9,-9,-9,-9,
52,52,52,52,-7,-7,-7,-7,
51,51,51,51,-5,-5,-5,-5,
49,49,49,49,-2,-2,-2,-2,
48,48,48,48,0,0,0,0,
46,46,46,46,3,3,3,3,
45,45,45,45,6,6,6,6,
43,43,43,43,8,8,8,8,
42,42,42,42,11,11,11,11,

};


//front
const char stpF[40][8] PROGMEM= {
40,40,40,40,14,14,14,14,
42,42,40,40,11,11,14,14,
43,43,40,40,8,8,14,14,
45,45,40,40,6,6,14,14,
46,46,40,40,3,3,14,14,
48,48,40,40,0,0,14,14,
49,49,40,40,-2,-2,14,14,
51,51,40,40,-5,-5,14,14,
52,52,40,40,-7,-7,14,14,
53,53,40,40,-9,-9,14,14,
55,55,40,40,-12,-12,14,14,
56,56,40,40,-14,-14,14,14,
57,57,40,40,-16,-16,14,14,
59,59,40,40,-19,-19,14,14,
60,60,40,40,-21,-21,14,14,
61,61,40,40,-23,-23,14,14,
62,62,40,40,-25,-25,14,14,
64,64,40,40,-27,-27,14,14,
65,65,40,40,-30,-30,14,14,
66,66,40,40,-32,-32,14,14,
68,68,40,40,-34,-34,14,14,
66,66,40,40,-32,-32,14,14,
65,65,40,40,-30,-30,14,14,
64,64,40,40,-27,-27,14,14,
62,62,40,40,-25,-25,14,14,
61,61,40,40,-23,-23,14,14,
60,60,40,40,-21,-21,14,14,
59,59,40,40,-19,-19,14,14,
57,57,40,40,-16,-16,14,14,
56,56,40,40,-14,-14,14,14,
55,55,40,40,-12,-12,14,14,
53,53,40,40,-9,-9,14,14,
52,52,40,40,-7,-7,14,14,
51,51,40,40,-5,-5,14,14,
49,49,40,40,-2,-2,14,14,
48,48,40,40,0,0,14,14,
46,46,40,40,3,3,14,14,
45,45,40,40,6,6,14,14,
43,43,40,40,8,8,14,14,
42,42,40,40,11,11,14,14,

};

//back
const char stpB[40][8] PROGMEM= {
40,40,40,40,14,14,14,14,
40,40,42,42,14,14,11,11,
40,40,43,43,14,14,8,8,
40,40,45,45,14,14,6,6,
40,40,46,46,14,14,3,3,
40,40,48,48,14,14,0,0,
40,40,49,49,14,14,-2,-2,
```

```
40,40,51,51,14,14,-5,-5,
40,40,52,52,14,14,-7,-7,
40,40,53,53,14,14,-9,-9,
40,40,55,55,14,14,-12,-12,
40,40,56,56,14,14,-14,-14,
40,40,57,57,14,14,-16,-16,
40,40,59,59,14,14,-19,-19,
40,40,60,60,14,14,-21,-21,
40,40,61,61,14,14,-23,-23,
40,40,62,62,14,14,-25,-25,
40,40,64,64,14,14,-27,-27,
40,40,65,65,14,14,-30,-30,
40,40,66,66,14,14,-32,-32,
40,40,68,68,14,14,-34,-34,
40,40,66,66,14,14,-32,-32,
40,40,65,65,14,14,-30,-30,
40,40,64,64,14,14,-27,-27,
40,40,62,62,14,14,-25,-25,
40,40,61,61,14,14,-23,-23,
40,40,60,60,14,14,-21,-21,
40,40,59,59,14,14,-19,-19,
40,40,57,57,14,14,-16,-16,
40,40,56,56,14,14,-14,-14,
40,40,55,55,14,14,-12,-12,
40,40,53,53,14,14,-9,-9,
40,40,52,52,14,14,-7,-7,
40,40,51,51,14,14,-5,-5,
40,40,49,49,14,14,-2,-2,
40,40,48,48,14,14,0,0,
40,40,46,46,14,14,3,3,
40,40,45,45,14,14,6,6,
40,40,43,43,14,14,8,8,
40,40,42,42,14,14,11,11,

};



//left
  const char stpL[40][8] PROGMEM= {
60,60,60,60,-21,-21,-21,-21,
60,60,60,60,-22,-21,-21,-22,
61,60,60,61,-23,-21,-21,-23,
61,60,60,61,-24,-21,-21,-24,
62,60,60,62,-24,-21,-21,-24,
62,60,60,62,-25,-21,-21,-25,
63,60,60,63,-26,-21,-21,-26,
63,60,60,63,-27,-21,-21,-27,
64,60,60,64,-28,-21,-21,-28,
64,60,60,64,-29,-21,-21,-29,
65,60,60,65,-30,-21,-21,-30,
65,60,60,65,-30,-21,-21,-30,
66,60,60,66,-31,-21,-21,-31,
66,60,60,66,-32,-21,-21,-32,
67,60,60,67,-33,-21,-21,-33,
68,60,60,68,-34,-21,-21,-34,
68,60,60,68,-35,-21,-21,-35,
69,60,60,69,-35,-21,-21,-35,
69,60,60,69,-36,-21,-21,-36,
70,60,60,70,-37,-21,-21,-37,
70,60,60,70,-38,-21,-21,-38,
70,60,60,70,-37,-21,-21,-37,
69,60,60,69,-36,-21,-21,-36,
69,60,60,69,-35,-21,-21,-35,
68,60,60,68,-35,-21,-21,-35,
68,60,60,68,-34,-21,-21,-34,
67,60,60,67,-33,-21,-21,-33,
66,60,60,66,-32,-21,-21,-32,
66,60,60,66,-31,-21,-21,-31,
65,60,60,65,-30,-21,-21,-30,
65,60,60,65,-30,-21,-21,-30,
64,60,60,64,-29,-21,-21,-29,
64,60,60,64,-28,-21,-21,-28,
63,60,60,63,-27,-21,-21,-27,
63,60,60,63,-26,-21,-21,-26,
62,60,60,62,-25,-21,-21,-25,
62,60,60,62,-24,-21,-21,-24,
61,60,60,61,-24,-21,-21,-24,
61,60,60,61,-23,-21,-21,-23,
60,60,60,60,-22,-21,-21,-22,
};
```

```
//right
  const char stpR[40][8] PROGMEM= {
60,60,60,60,-21,-21,-21,-21,
60,60,60,60,-21,-22,-22,-21,
60,61,61,60,-21,-23,-23,-21,
60,61,61,60,-21,-24,-24,-21,
60,62,62,60,-21,-24,-24,-21,
60,62,62,60,-21,-25,-25,-21,
60,63,63,60,-21,-26,-26,-21,
60,63,63,60,-21,-27,-27,-21,
60,64,64,60,-21,-28,-28,-21,
60,64,64,60,-21,-29,-29,-21,
60,65,65,60,-21,-30,-30,-21,
60,65,65,60,-21,-30,-30,-21,
60,66,66,60,-21,-31,-31,-21,
60,66,66,60,-21,-32,-32,-21,
60,67,67,60,-21,-33,-33,-21,
60,68,68,60,-21,-34,-34,-21,
60,68,68,60,-21,-35,-35,-21,
60,69,69,60,-21,-35,-35,-21,
60,69,69,60,-21,-36,-36,-21,
60,70,70,60,-21,-37,-37,-21,
60,70,70,60,-21,-38,-38,-21,
60,70,70,60,-21,-37,-37,-21,
60,69,69,60,-21,-36,-36,-21,
60,69,69,60,-21,-35,-35,-21,
60,68,68,60,-21,-35,-35,-21,
60,68,68,60,-21,-34,-34,-21,
60,67,67,60,-21,-33,-33,-21,
60,66,66,60,-21,-32,-32,-21,
60,66,66,60,-21,-31,-31,-21,
60,65,65,60,-21,-30,-30,-21,
60,65,65,60,-21,-30,-30,-21,
60,64,64,60,-21,-29,-29,-21,
60,64,64,60,-21,-28,-28,-21,
60,63,63,60,-21,-27,-27,-21,
60,63,63,60,-21,-26,-26,-21,
60,62,62,60,-21,-25,-25,-21,
60,62,62,60,-21,-24,-24,-21,
60,61,61,60,-21,-24,-24,-21,
60,61,61,60,-21,-23,-23,-21,
60,60,60,60,-21,-22,-22,-21,

};


//turn left
const char turnL[29][8] PROGMEM = {
 62,  69,  54,  57, -28, -40, -13, -18,
  63,  60,  59,  55, -28, -40, -13, -18,
  64,  51,  62,  53, -28, -37, -12, -18,
  65,  42,  67,  52, -28, -34, -10, -17,
  67,  33,  70,  50, -27, -30,  -8, -17,
  69,  23,  73,  49, -27, -22,  -5, -17,
  70,  18,  75,  46, -27, -14,  -2, -15,
  72,  17,  76,  45, -27,  -7,   2, -14,
  73,  24,  82,  46, -29, -13,  -5, -14,
  72,  29,  85,  48, -30, -17, -14, -14,
  72,  36,  83,  49, -30, -20, -18, -14,
  70,  42,  79,  51, -31, -22, -22, -15,
  68,  49,  74,  52, -31, -25, -24, -15,
  66,  55,  69,  53, -31, -26, -26, -15,
  65,  61,  62,  55, -31, -26, -26, -15,
  62,  67,  55,  56, -31, -26, -26, -15,
  61,  72,  48,  57, -31, -25, -25, -15,
  59,  77,  40,  58, -31, -25, -23, -15,
  57,  81,  32,  60, -31, -22, -19, -14,
  55,  84,  25,  61, -30, -19, -15, -15,
  51,  87,  18,  61, -28, -16,  -7, -14,
  51,  88,  15,  63, -25, -11,   2, -14,
  52,  96,  20,  63, -27, -19,  -1, -14,
  54,  98,  26,  64, -27, -27,  -5, -17,
  55,  97,  31,  63, -27, -32,  -8, -17,
  56,  93,  37,  61, -27, -36, -10, -17,
  58,  87,  42,  60, -28, -39, -12, -17,
  60,  79,  48,  59, -28, -40, -13, -18,
  61,  70,  53,  58, -28, -40, -13, -18,
};


//turn right
const char turnR[29][8] PROGMEM = {
  61,  64,  55,  61, -26, -31, -15, -26,
```

```
  67,  62,  56,  54, -26, -31, -15, -26,
  72,  60,  57,  47, -25, -31, -15, -24,
  77,  59,  58,  39, -25, -31, -15, -22,
  81,  56,  60,  31, -22, -30, -14, -19,
  84,  54,  61,  24, -19, -30, -15, -14,
  87,  51,  61,  16, -16, -27, -14,  -5,
  88,  51,  63,  15, -11, -25, -14,   2,
  96,  52,  63,  21, -19, -27, -14,  -2,
  98,  54,  64,  26, -27, -27, -17,  -5,
  97,  55,  63,  32, -32, -27, -17,  -8,
  93,  57,  61,  38, -36, -27, -17, -10,
  87,  58,  60,  44, -39, -28, -17, -12,
  79,  60,  59,  49, -40, -28, -18, -13,
  70,  62,  58,  54, -40, -28, -18, -13,
  62,  63,  55,  59, -40, -28, -18, -13,
  52,  64,  54,  62, -37, -28, -18, -12,
  43,  65,  52,  67, -35, -28, -17, -10,
  34,  67,  51,  70, -31, -27, -17,  -8,
  26,  69,  49,  73, -25, -27, -17,  -5,
  18,  70,  46,  75, -15, -27, -15,  -2,
  17,  72,  45,  76,  -7, -27, -14,   2,
  22,  73,  46,  82, -12, -29, -14,  -5,
  28,  72,  48,  85, -16, -30, -14, -14,
  35,  72,  49,  83, -19, -30, -14, -18,
  41,  70,  50,  79, -22, -31, -14, -22,
  48,  68,  52,  74, -25, -31, -15, -24,
  54,  66,  53,  69, -26, -31, -15, -26,
  60,  65,  54,  62, -26, -31, -15, -26,
};

//lay down back
const char lay[2][8] PROGMEM = {
35,35,35,35,25,25,25,25,
35,35,97,97,25,25,38,38,

 };


//push up
const char push[40][8] PROGMEM = {
55,55,97,97,-12,-12,38,38,
56,56,97,97,-14,-14,38,38,
57,57,97,97,-16,-16,38,38,
59,59,97,97,-19,-19,38,38,
60,60,97,97,-21,-21,38,38,
61,61,97,97,-23,-23,38,38,
62,62,97,97,-25,-25,38,38,
64,64,97,97,-27,-27,38,38,
65,65,97,97,-30,-30,38,38,
66,66,97,97,-32,-32,38,38,
68,68,97,97,-34,-34,38,38,
69,69,97,97,-36,-36,38,38,
70,70,97,97,-38,-38,38,38,
71,71,97,97,-40,-40,38,38,
73,73,97,97,-42,-42,38,38,
74,74,97,97,-44,-44,38,38,
75,75,97,97,-46,-46,38,38,
77,77,97,97,-48,-48,38,38,
78,78,97,97,-50,-50,38,38,
79,79,97,97,-52,-52,38,38,
81,81,97,97,-54,-54,38,38,
79,79,97,97,-52,-52,38,38,
78,78,97,97,-50,-50,38,38,
77,77,97,97,-48,-48,38,38,
75,75,97,97,-46,-46,38,38,
74,74,97,97,-44,-44,38,38,
73,73,97,97,-42,-42,38,38,
71,71,97,97,-40,-40,38,38,
70,70,97,97,-38,-38,38,38,
69,69,97,97,-36,-36,38,38,
68,68,97,97,-34,-34,38,38,
66,66,97,97,-32,-32,38,38,
65,65,97,97,-30,-30,38,38,
64,64,97,97,-27,-27,38,38,
62,62,97,97,-25,-25,38,38,
61,61,97,97,-23,-23,38,38,
60,60,97,97,-21,-21,38,38,
59,59,97,97,-19,-19,38,38,
57,57,97,97,-16,-16,38,38,
56,56,97,97,-14,-14,38,38,
 };
```

```
char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  10,  8, 8,  -3, 8,  8};

byte pins[] = {12, 11, 3, 4,
```

下面是控制运动的代码部分，重复的部分很多，基本就是之前都写过的使用步态数据使机器狗运动的代码，不做重复的注释了。唯一需要注意的点是，为了解决动态内存不足的问题需要使用PROGMEM, 具体在变量声明和使用时如何操作我写了一段示例程序用来说明。

```
#include <avr/pgmspace.h>


char displayInt;
const char charSet [8] PROGMEM =
    {0,1,2,3,0,1,2,-3,};
// 读取

void setup() {
  // put your setup code here, to run once:
 Serial.begin(57600);
}

void loop() {
  // put your main code here, to run repeatedly:
for (int i = 0; i < 3; i++)
  {
     displayInt = int(pgm_read_byte(charSet + 7));    // 第一种方法
    //displayInt = pgm_read_byte(&charSet[7]);    // 第二种方法
    Serial.println(int(displayInt));
  }

}
```

下面是舞蹈程序的后续部分：

```
char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  10,  8, 8,  -3, 8,  8};

byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };

int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };



int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) *
  7.04  * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}


volatile bool mpuInterrupt = false;
// indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
  mpuInterrupt = true;
}


void setup() {
  Serial.begin(57600);
  pwm.begin();
  pwm.setPWMFreq(240);


  mpu.initialize();
```

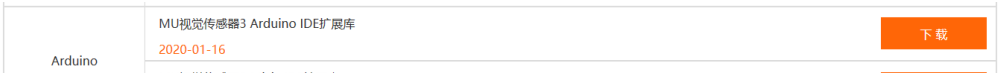
```
    do {
    devStatus = mpu.dmpInitialize();
    delay(500);
    mpu.setZAccelOffset(985);
    mpu.setXGyroOffset(39);
    mpu.setYGyroOffset(13);
    mpu.setZGyroOffset(-6);

    if (devStatus == 0) {
      mpu.setDMPEnabled(true);
      attachInterrupt(0, dmpDataReady, RISING);

    }

  } while (devStatus);
}


int i;
int j;
int k;
int l;
int m;
char temp0;
char temp1;


void loop() {

  //step 1
  for (m=0;m<=3;m++){
  for( l=0;l<=3;l++){
  for( i=0;i<=38;i++){
    for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&stpL[i][j]);
        Serial.println(temp0);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
    }
   for (  k=1;k<=2;k++){
    for( j=0;j<=7;j++){
          temp0=pgm_read_byte(&stpL[i][j]);
          temp1=pgm_read_byte(&stpL[i+1][j]);
          pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
    }
        }
  }
  }
  for( l=0;l<=3;l++){
  for( i=0;i<=38;i++){
    for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&stpR[i][j]);
        Serial.println(temp0);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
    }
   for ( k=1;k<=2;k++){
    for( j=0;j<=7;j++){
          temp0=pgm_read_byte(&stpR[i][j]);
          temp1=pgm_read_byte(&stpR[i+1][j]);
          pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
    }
        }
  }
  }
  }


    //step 2
  for( l=0;l<=10;l++){
  for( i=0;i<=38;i++){
    for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&stp[i][j]);
        Serial.println(temp0);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
    }
   for ( k=1;k<=2;k++){
    for( j=0;j<=7;j++){
          temp0=pgm_read_byte(&stp[i][j]);
          temp1=pgm_read_byte(&stp[i+1][j]);
          pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
    }
        }
  }
  }

    //step 3
   for (m=0;m<=3;m++){
  for( l=0;l<=3;l++){
  for( i=0;i<=38;i++){
    for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&stpF[i][j]);
        Serial.println(temp0);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
```

```
  }
 for (  k=1;k<=2;k++){
  for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&stpF[i][j]);
        temp1=pgm_read_byte(&stpF[i+1][j]);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
  }
      }
}
}

for( l=0;l<=3;l++){
for( i=0;i<=38;i++){
  for( j=0;j<=7;j++){
      temp0=pgm_read_byte(&stpB[i][j]);
      Serial.println(temp0);
      pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
  }
 for ( k=1;k<=2;k++){
  for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&stpB[i][j]);
        temp1=pgm_read_byte(&stpB[i+1][j]);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
  }
      }
}
}
}

//step 4
for( l=0;l<=10;l++){
for( i=0;i<=38;i++){
  for( j=0;j<=7;j++){
      temp0=pgm_read_byte(&stp[i][j]);
      Serial.println(temp0);
      pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
  }
 for ( k=1;k<=2;k++){
  for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&stp[i][j]);
        temp1=pgm_read_byte(&stp[i+1][j]);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
  }
      }
}
}

  //step 5
for( l=0;l<=20;l++){
for( i=0;i<=28;i++){
  for( j=0;j<=7;j++){
      temp0=pgm_read_byte(&turnL[i][j]);
      Serial.println(temp0);
      pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
  }
 for ( k=1;k<=2;k++){
  for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&turnL[i][j]);
        temp1=pgm_read_byte(&turnL[i+1][j]);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
  }
      }
}
}

  //step 6
for( l=0;l<=20;l++){
for( i=0;i<=28;i++){
  for( j=0;j<=7;j++){
      temp0=pgm_read_byte(&turnR[i][j]);
      Serial.println(temp0);
      pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
  }
 for ( k=1;k<=2;k++){
  for( j=0;j<=7;j++){
        temp0=pgm_read_byte(&turnR[i][j]);
        temp1=pgm_read_byte(&turnR[i+1][j]);
        pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
  }
      }
}
}

//step 7
for( l=0;l<=10;l++){
for( i=0;i<=38;i++){
  for( j=0;j<=7;j++){
      temp0=pgm_read_byte(&stp[i][j]);
      Serial.println(temp0);
      pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
  }
 for ( k=1;k<=2;k++){
```

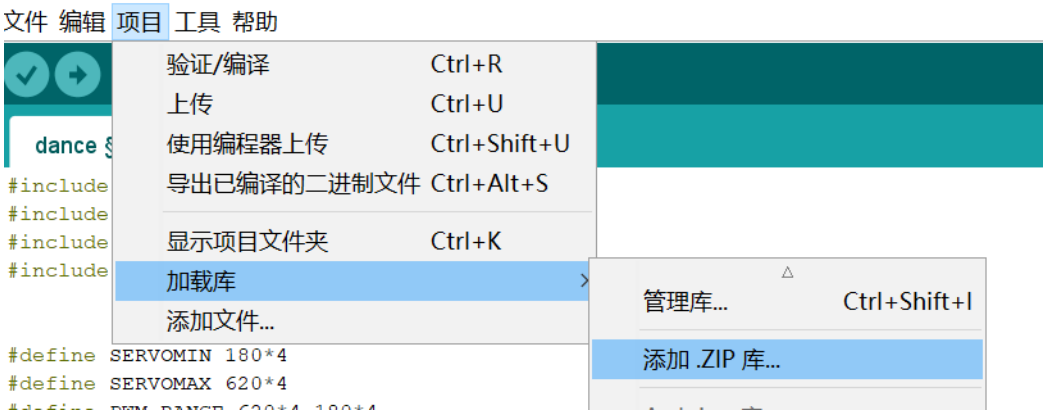
```
 for( j=0;j<=7;j++){
       temp0=pgm_read_byte(&stp[i][j]);
       temp1=pgm_read_byte(&stp[i+1][j]);
       pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
  }
      }
}
}

  //step 8
for( i=0;i<=0;i++){
  for( j=0;j<=7;j++){
      temp0=pgm_read_byte(&lay[i][j]);
      Serial.println(temp0);
      pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
  }
 for ( k=1;k<=79;k++){
  for( j=0;j<=7;j++){
       temp0=pgm_read_byte(&lay[i][j]);
       temp1=pgm_read_byte(&lay[i+1][j]);
       pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/80*(int(temp0)-int(temp1)),8+j));
  }
      }
}

    //step 9
   for( l=0;l<=10;l++){
for( i=0;i<=38;i++){
  for( j=0;j<=7;j++){
      temp0=pgm_read_byte(&push[i][j]);
      Serial.println(temp0);
      pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
  }
 for ( k=1;k<=9;k++){
  for( j=0;j<=7;j++){
       temp0=pgm_read_byte(&push[i][j]);
       temp1=pgm_read_byte(&push[i+1][j]);
       pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/10*(int(temp0)-int(temp1)),8+j));
  }
      } }
}

//step 10
 for( l=0;l<=10;l++){
for( i=0;i<=38;i++){
  for( j=0;j<=7;j++){
      temp0=pgm_read_byte(&stp[i][j]);
      Serial.println(temp0);
      pwm.setPWM(pins[8+j],0,S2P( int(temp0),8+j));
  }
 for ( k=1;k<=2;k++){
  for( j=0;j<=7;j++){
       temp0=pgm_read_byte(&stp[i][j]);
       temp1=pgm_read_byte(&stp[i+1][j]);
       pwm.setPWM(pins[8+j],0,S2P( int(temp0)+i/3*(int(temp0)-int(temp1)),8+j));
  }
      }
}
}

//stop
   delay(100000000000);
}
```

4.摄像头的使用

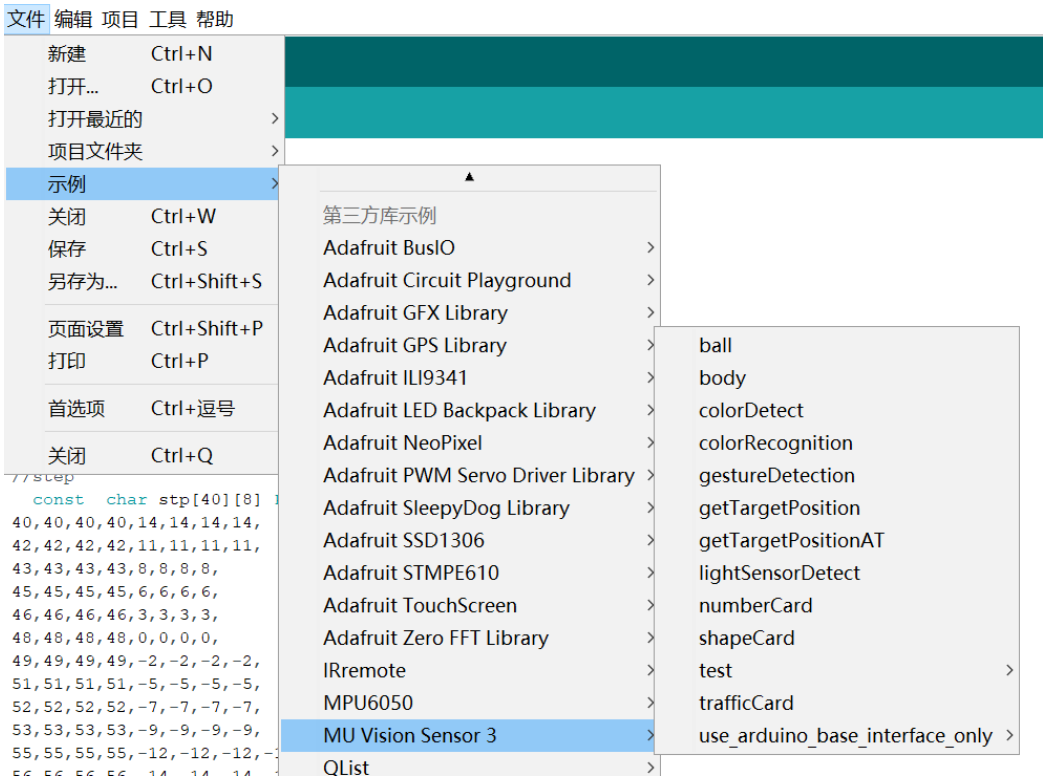
使用arduino编写有关摄像头的程序，需要先下载摄像头厂家提供的拓展库，因为arduino的运算能力不足以支持我们在处理通讯，控制舵机的同时，还要进行图像处理这种需要大量运算的操作。我们使用的摄像头是mu视觉传感器3，可以到官网这个链接：http://mai.morpx.com/page.php?a=sensor-support，下载arduino的拓展库，选择这一个即可。



之后依次点项目，加载库，添加.ZIP库，找到下载好的压缩包解压出来的文件夹里唯一可选的文件，点击之后即可完成安装。



之后我们可以在示例里找到演示使用摄像头各个功能的代码，我们可以利用这些代码编写出符合自己需求的程序。



下面是我编写的利用摄像头识别交通卡片，用卡片来控制机器狗直行，左转，右转，停止的程序，思路很简单，在摄像头识别到某一卡片时，执行相关步态即可，代码与注释如下:

```
#include <Arduino.h>
#include <MuVisionSensor.h>


#define SERVOMIN 180*4  //最小电机占空比计数
#define SERVOMAX 620*4  //最大电机占空比计数
#define MU_ADDRESS    0x60


uint8_t devStatus;
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();
MPU6050 mpu;
MuVisionSensor Mu(MU_ADDRESS);


//前进步态
  char trF[36][8] = {
  61,  68,  54,  61, -26, -39, -13, -26,
  66,  61,  58,  55, -26, -39, -13, -26,
  70,  54,  61,  49, -26, -37, -12, -25,
  73,  46,  65,  43, -24, -35, -11, -23,
  77,  39,  67,  36, -23, -33, -10, -21,
  80,  33,  70,  31, -21, -30,  -8, -19,
  83,  26,  72,  25, -19, -26,  -6, -15,
  85,  19,  74,  18, -16, -20,  -3,  -9,
  86,  16,  76,  14, -13, -12,  -1,   0,
  87,  19,  76,  16,  -9, -11,   3,   2,
  93,  24,  82,  21, -16, -14,  -5,  -2,
  96,  29,  85,  26, -24, -17, -12,  -5,
  95,  34,  84,  30, -28, -20, -17,  -7,
  92,  39,  81,  35, -31, -22, -20,  -9,
  89,  44,  79,  39, -34, -24, -22, -11,
  84,  49,  74,  44, -36, -25, -24, -12,
  79,  54,  70,  48, -38, -26, -26, -13,
  72,  59,  65,  52, -39, -26, -26, -13,
  65,  64,  59,  56, -39, -26, -26, -13,
  60,  68,  54,  60, -38, -26, -26, -13,
  52,  72,  48,  62, -37, -25, -25, -12,
  45,  75,  41,  66, -35, -24, -23, -11,
  37,  79,  35,  69, -32, -22, -20,  -9,
  30,  81,  29,  71, -29, -20, -17,  -7,
  24,  83,  24,  73, -25, -17, -14,  -4,
  19,  85,  18,  75, -19, -14,  -7,  -2,
  16,  87,  14,  76, -10, -11,   1,   1,
  20,  90,  17,  79, -12, -12,   1,   0,
  25,  96,  22,  85, -15, -20,  -2,  -9,
  30,  97,  26,  85, -18, -26,  -5, -15,
  35,  94,  31,  83, -20, -30,  -8, -18,
  40,  91,  36,  80, -23, -33, -10, -21,
  45,  86,  40,  76, -24, -36, -11, -24,
  51,  81,  45,  72, -25, -37, -12, -25,
  55,  76,  49,  68, -26, -38, -13, -26,
```

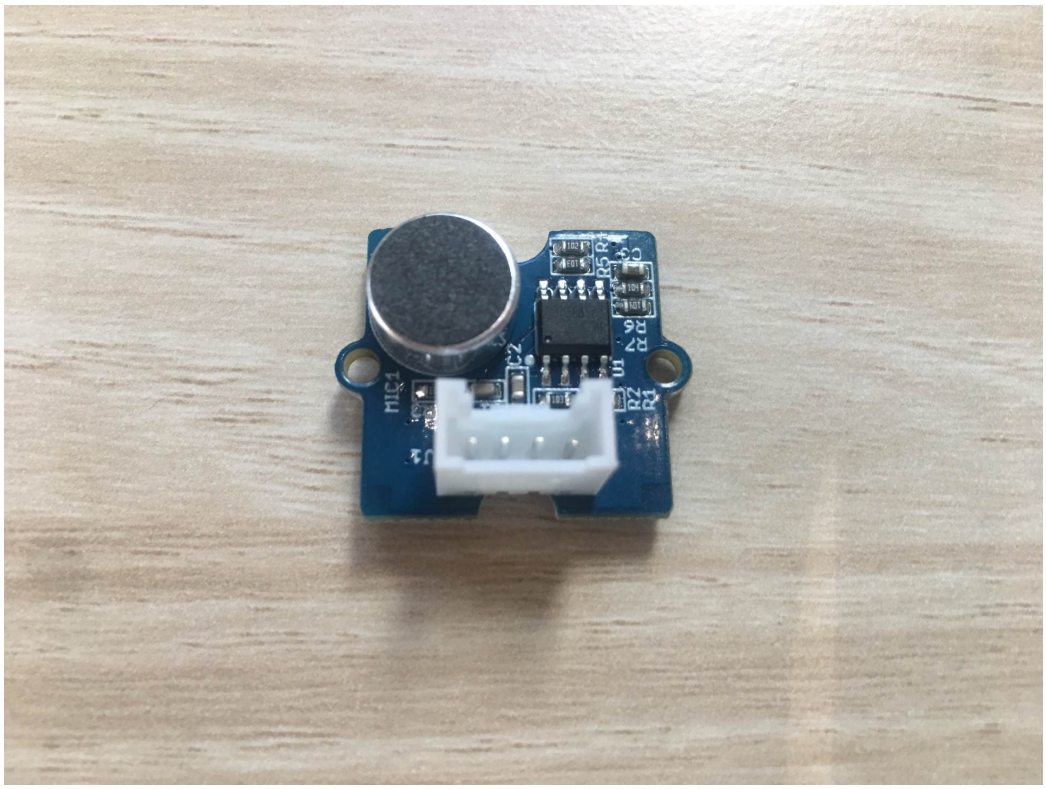
```
  60,  70,  53,  62, -26, -39, -13, -26,
};

//左转步态
  char trL[29][8]  = {
 62,  69,  54,  57, -28, -40, -13, -18,
  63,  60,  59,  55, -28, -40, -13, -18,
  64,  51,  62,  53, -28, -37, -12, -18,
  65,  42,  67,  52, -28, -34, -10, -17,
  67,  33,  70,  50, -27, -30,  -8, -17,
  69,  23,  73,  49, -27, -22,  -5, -17,
  70,  18,  75,  46, -27, -14,  -2, -15,
  72,  17,  76,  45, -27,  -7,   2, -14,
  73,  24,  82,  46, -29, -13,  -5, -14,
  72,  29,  85,  48, -30, -17, -14, -14,
  72,  36,  83,  49, -30, -20, -18, -14,
  70,  42,  79,  51, -31, -22, -22, -15,
  68,  49,  74,  52, -31, -25, -24, -15,
  66,  55,  69,  53, -31, -26, -26, -15,
  65,  61,  62,  55, -31, -26, -26, -15,
  62,  67,  55,  56, -31, -26, -26, -15,
  61,  72,  48,  57, -31, -25, -25, -15,
  59,  77,  40,  58, -31, -25, -23, -15,
  57,  81,  32,  60, -31, -22, -19, -14,
  55,  84,  25,  61, -30, -19, -15, -15,
  51,  87,  18,  61, -28, -16,  -7, -14,
  51,  88,  15,  63, -25, -11,   2, -14,
  52,  96,  20,  63, -27, -19,  -1, -14,
  54,  98,  26,  64, -27, -27,  -5, -17,
  55,  97,  31,  63, -27, -32,  -8, -17,
  56,  93,  37,  61, -27, -36, -10, -17,
  58,  87,  42,  60, -28, -39, -12, -17,
  60,  79,  48,  59, -28, -40, -13, -18,
  61,  70,  53,  58, -28, -40, -13, -18,
};

//右转步态
  char trR[29][8]  = {
  61,  64,  55,  61, -26, -31, -15, -26,
  67,  62,  56,  54, -26, -31, -15, -26,
  72,  60,  57,  47, -25, -31, -15, -24,
  77,  59,  58,  39, -25, -31, -15, -22,
  81,  56,  60,  31, -22, -30, -14, -19,
  84,  54,  61,  24, -19, -30, -15, -14,
  87,  51,  61,  16, -16, -27, -14,  -5,
  88,  51,  63,  15, -11, -25, -14,   2,
  96,  52,  63,  21, -19, -27, -14,  -2,
  98,  54,  64,  26, -27, -27, -17,  -5,
  97,  55,  63,  32, -32, -27, -17,  -8,
  93,  57,  61,  38, -36, -27, -17, -10,
  87,  58,  60,  44, -39, -28, -17, -12,
  79,  60,  59,  49, -40, -28, -18, -13,
  70,  62,  58,  54, -40, -28, -18, -13,
  62,  63,  55,  59, -40, -28, -18, -13,
  52,  64,  54,  62, -37, -28, -18, -12,
  43,  65,  52,  67, -35, -28, -17, -10,
  34,  67,  51,  70, -31, -27, -17,  -8,
  26,  69,  49,  73, -25, -27, -17,  -5,
  18,  70,  46,  75, -15, -27, -15,  -2,
  17,  72,  45,  76,  -7, -27, -14,   2,
  22,  73,  46,  82, -12, -29, -14,  -5,
  28,  72,  48,  85, -16, -30, -14, -14,
  35,  72,  49,  83, -19, -30, -14, -18,
  41,  70,  50,  79, -22, -31, -14, -22,
  48,  68,  52,  74, -25, -31, -15, -24,
  54,  66,  53,  69, -26, -31, -15, -26,
  60,  65,  54,  62, -26, -31, -15, -26,
 };

   char stp[1][8]  = {
  30,30,30,30,30,30,30,30,
 };


//校准电机误差
char cal[16]= {0,  0,  0,  0,  0,  0,  0,  3,  3,  8,  8, 8,  -6, 0,  3};

//关节映射表
byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

//旋转方向
int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                              -1, 1, 1, -1
                             };
//角度转化为占空比计数时需要的偏移量
int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };


 //将角度转化为占空比计数
int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) *
  7.04  * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}



volatile bool mpuInterrupt = false;     // 表示mpu interrupt 引脚是否被拉高

void dmpDataReady() {
  mpuInterrupt = true;  //表示发生了中断
}


void setup() {
  //配置串口
  Serial.begin(57600);
  //配置电机pwm
  pwm.begin();
  pwm.setPWMFreq(240);

   uint8_t err = 0;
   Wire.begin();
  // initialized MU on the I2C port
  err = Mu.begin(&Wire);

  if (err == MU_OK) {
    Serial.println("MU initialized.");
  } else {
    do {
      Serial.println("fail to initialize MU! Please check protocol "
                     "version or make sure MU is working on the "
                     "correct port with correct mode.");
      delay(5000);
    } while (1);
  }
  // enable vision: traffic card
  Mu.VisionBegin(VISION_TRAFFIC_CARD_DETECT);
}



int t=5;
```

```
void loop() {

   // put your main code here, to run repeatedly:
  long time_start = millis();

  // read result
  if (Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kStatus)) {                   // update vision result and get status, 0: undetected, other: detected
    Serial.println("vision traffic card detected:");
    Serial.print("x = ");
    Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kXValue));       // get vision result: x axes value
    Serial.print("y = ");
    Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kYValue));       // get vision result: y axes value
    Serial.print("width = ");
    Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kWidthValue));   // get vision result: width value
    Serial.print("height = ");
    Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kHeightValue));  // get vision result: height value
    Serial.print("label = ");
    switch (Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kLabel)) {              // get vision result: label value
      case MU_TRAFFIC_CARD_FORWARD:
        Serial.println("forward");
        t=0;
        break;
      case MU_TRAFFIC_CARD_LEFT:
        Serial.println("left");
        t=1;
        break;
      case MU_TRAFFIC_CARD_RIGHT:
        Serial.println("right");
        t=2;
        break;
      case MU_TRAFFIC_CARD_TURN_AROUND:
        Serial.println("turn around");
        t=3;
        break;
      case MU_TRAFFIC_CARD_PARK:
        Serial.println("park");
        t=4;
        break;
      default:
        Serial.print("unknow card type: ");
        Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kLabel));
        break;
    }
  } else {
    Serial.println("vision shape card undetected.");
  }
  Serial.print("fps = ");
  Serial.println(1000/(millis()-time_start));
  Serial.println();



      //控制部分
      if(t==0){//前进
    for(int i=0;i<=35;i++){
    for(int j=0;j<=7;j++){
      //根据步态数据设置每个关节的转角
      pwm.setPWM(pins[8+j],0,S2P(   (int(trF[i][j]))  ,8+j  ));
    }
    }
      }
     else if(t==1){//左转
       for(int i=0;i<=28;i++){
    for(int j=0;j<=7;j++){
      pwm.setPWM(pins[8+j],0,S2P(   (int(trL[i][j]))  ,8+j  ));
    }
    }
      }
     else if(t==2){//右转
      for(int i=0;i<=28;i++){
    for(int j=0;j<=7;j++){
      pwm.setPWM(pins[8+j],0,S2P(   (int(trR[i][j]))  ,8+j  ));
    }
    }
      }
      else if(t==4){//停下
     for(int i=0;i<=0;i++){
    for(int j=0;j<=7;j++){
      pwm.setPWM(pins[8+j],0,S2P(   (int(stp[i][j]))  ,8+j  ));
    }
    }
      }
}
```

5.各种传感器的使用

bittle传感器拓展包提供了6种不同的外设，包括5种传感器和一块oled显示屏，下面我们依次介绍如何使用各个设备。

1）声音传感器



使用非常简单，直接读取模拟数据，串口输出即可，得到的数值反应了音量的大小。

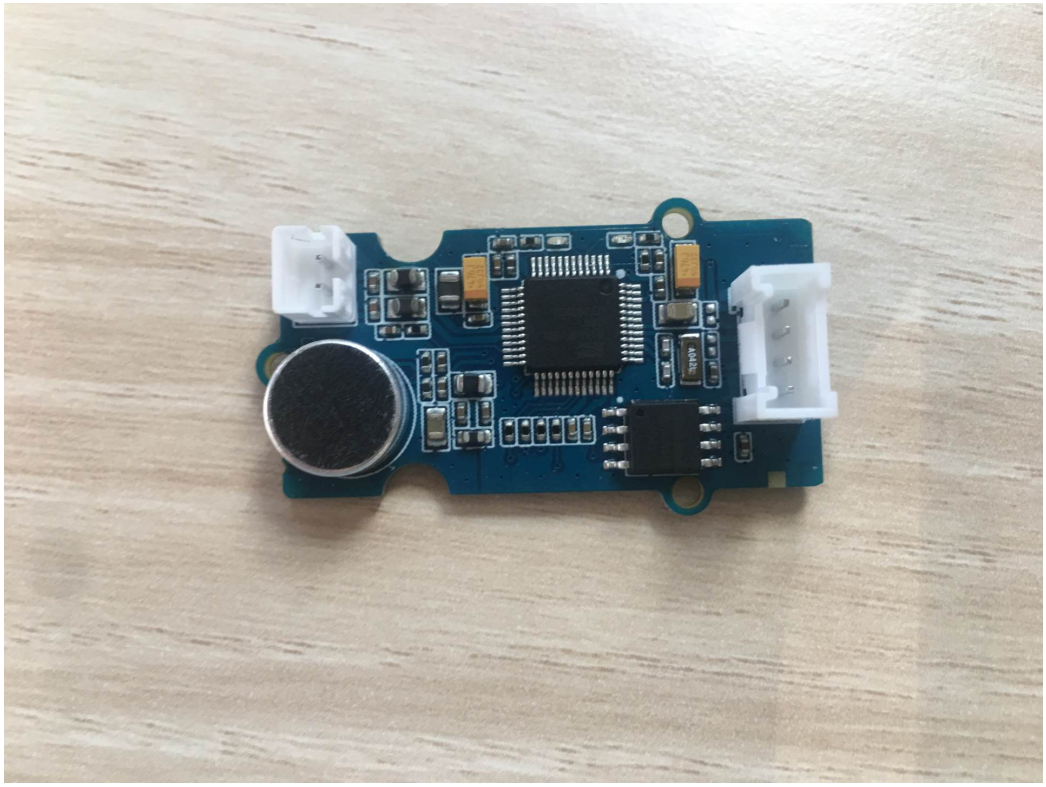
```
int val;

void setup() {
   Serial.begin(9600);
}

void loop() {
    val = analogRead(A2);
    Serial.println(val);

}
```

下面是两个具体使用声音传感器来完成一些简单任务的示例代码，第一个是如果检测到过大的声音就原地踏步，第二个是记录1秒内被敲击的次数（即声音过大的次数），根据次数的不同做出不同反应，代码基本是之前做过的东西的组合，程序内有注释说明，不再做过多的讲解。

```
SoftwareSerial softSerial(SOFTSERIAL_RX_PIN,SOFTSERIAL_TX_PIN);

const char *voiceBuffer[] =
{
    "Turn on the light",
    "Turn off the light",
    "Play music",
    "Pause",
    "Next",
    "Previous",
    "Up",
    "Down",
    "Turn on the TV",
    "Turn off the TV",
    "Increase temperature",
    "Decrease temperature",
    "What's the time",
    "Open the door",
    "Close the door",
    "Left",
    "Right",
    "Stop",
    "Start",
    "Mode 1",
    "Mode 2",
    "Go",
};

void setup()
{
    Serial.begin(9600);
    softSerial.begin(9600);
    softSerial.listen();
}

void loop()
{
    char cmd;

    if(softSerial.available())
    {
        cmd = softSerial.read();
        Serial.println(voiceBuffer[cmd - 1]);
    }
}
```

```
#define SERVOMIN 180*4
#define SERVOMAX 620*4
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();

#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel pixels_10 = Adafruit_NeoPixel(7, 10, NEO_GRB + NEO_KHZ800);



char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  10,  8, 8,  -3, 8,  8};

byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };

int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };



int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) * 7.04  * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}


int val;
int i;
int j;



void setup() {
  Serial.begin(57600);
  pwm.begin();
  pwm.setPWMFreq(240);

}

int times =0;
  void loop() {
    for(int t=0;t<1000;t++){//判断1000次
    val = analogRead(A2);//读取数据
    delay(1);//延时1ms
    Serial.println(val);
    if(val>800)
    times++;//如果音量超过800，次数增加1
    }


    if(times>5)//次数超过5次，站立并led亮起红灯
    {
    for( j=0;j<=7;j++){
        pwm.setPWM(pins[8+j],0,S2P( 30,8+j));
    }
     pixels_10.setBrightness(255);
  pixels_10.begin();
    for (int count = 0; count < 7; count++) {
      pixels_10.setPixelColor(count, pixels_10.Color(0xff, 0x00, 0x00));
  }
  pixels_10.show();
    }
    //否则坐下，并led亮起绿灯
    else{
        for( j=0;j<=7;j++){
        pwm.setPWM(pins[8+j],0,S2P(0,8+j));
    }
     pixels_10.setBrightness(255);
  pixels_10.begin();
    for (int count = 0; count < 7; count++) {
      pixels_10.setPixelColor(count, pixels_10.Color(0x00, 0xff, 0x00));
  }
  pixels_10.show();
    }

    delay(10000);
  }
```

2）语音识别模块



该模块可以识别一些设定好的语句，具体的实现已经封装好在该模块自带的芯片中，我们只要读取其输出的数字信号即可，每一个传入的数字代表一个语句，可以在对应数组中找出。

```
SoftwareSerial softSerial(SOFTSERIAL_RX_PIN,SOFTSERIAL_TX_PIN);

const char *voiceBuffer[] =
{
    "Turn on the light",
    "Turn off the light",
    "Play music",
    "Pause",
    "Next",
    "Previous",
    "Up",
    "Down",
    "Turn on the TV",
    "Turn off the TV",
    "Increase temperature",
    "Decrease temperature",
    "What's the time",
    "Open the door",
    "Close the door",
    "Left",
    "Right",
    "Stop",
    "Start",
    "Mode 1",
    "Mode 2",
    "Go",
};

void setup()
{
    Serial.begin(9600);
    softSerial.begin(9600);
    softSerial.listen();
}

void loop()
{
    char cmd;

    if(softSerial.available())
    {
        cmd = softSerial.read();
        Serial.println(voiceBuffer[cmd - 1]);
    }
}
```

下面是具体使用语音识别模块来完成一些简单任务的示例代码，当听到start时，机器狗会站起来，当听到stop时，机器狗会坐下去，代码基本是之前做过的东西的组合，程序内有注释说明，不再做过多的讲解。

```
#define SERVOMIN 180*4
#define SERVOMAX 620*4
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();

SoftwareSerial softSerial(SOFTSERIAL_RX_PIN,SOFTSERIAL_TX_PIN);

const char *voiceBuffer[] =
{
    "Turn on the light",
    "Turn off the light",
    "Play music",
    "Pause",
    "Next",
    "Previous",
    "Up",
    "Down",
    "Turn on the TV",
    "Turn off the TV",
    "Increase temperature",
    "Decrease temperature",
```

```
    "What's the time",
    "Open the door",
    "Close the door",
    "Left",
    "Right",
    "Stop",
    "Start",
    "Mode 1",
    "Mode 2",
    "Go",
};

char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  10,  8, 8,  -3, 8,  8};

byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };

int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };



int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) * 7.04
      * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}


void setup()
{
    Serial.begin(9600);
    softSerial.begin(9600);
    softSerial.listen();
      pwm.begin();
  pwm.setPWMFreq(240);
}

int j;

void loop()
{
    char cmd;

    if(softSerial.available())
    {
        cmd = softSerial.read();
        Serial.println(voiceBuffer[cmd - 1]);
    }

    if(cmd==19)//19对应start，听到start，机器狗站立
    {
    for( j=0;j<=7;j++){
        pwm.setPWM(pins[8+j],0,S2P(30,8+j));
    }
    delay(10000);
    }

    if(cmd==18)//18对应stop，听到stop，机器狗坐下
    {
    for( j=0;j<=7;j++){
        pwm.setPWM(pins[8+j],0,S2P(0,8+j));
    }
    delay(10000);
    }
}
```

3）超声波模块

由于超声波指向性强，能量消耗缓慢，在介质中传播的距离较远，因而经常用于距离的测量。超声波测距的原理是利用超声波发射器向某一方向发射超声波，超声波在空气中传播，遇到障碍物就立即被反射回来，接收器收到反射波后，根据发射和反射的时长t，就能计算出障碍物的距离，即：s=340× t ／ 2 。

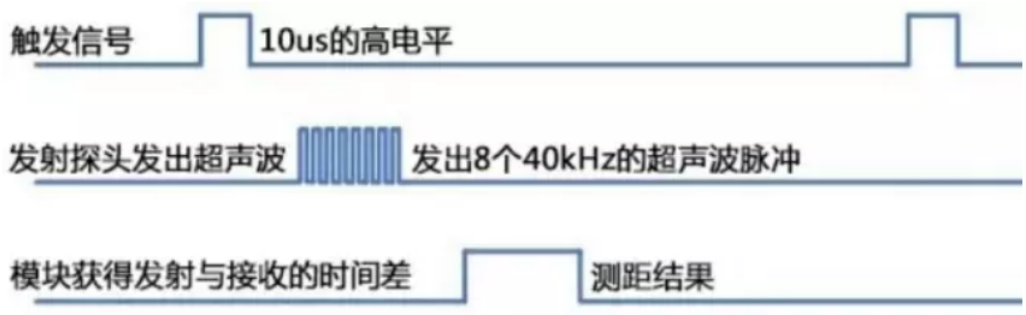
声波在空气中的传播速度为340m/s，测距精度为厘米级，若要准确测量，还需要根据环境温度进行修正。

模块工作原理

(1)TRIP引脚内部经10K电阻上拉，测量时将TRIP引脚拉低，然后给一个10us以上的脉冲信号。

(2)模块自动发送8个40khz的方波，自动检测是否有信号返回；

(3)有信号返回时，通过Echo引脚输出高电平，高电平的持续的时间就是超声波从发射到返回的时间。



因此我们只需要给Trig一个10us的高电平，然后检测echo引脚的高电平持续时间即可。对持续时间做相应的数据处理即可得到障碍物距离。

代码与注释如下：

```
    public:
    Ultrasonic(int pin);
    void DistanceMeasure(void);
    double microsecondsToCentimeters(void);
    double microsecondsToInches(void);
    private:
    int this_pin;//pin number of Arduino that is connected with SIG pin of Ultrasonic Ranger.
    long duration;// the Pulse time received;
};
Ultrasonic::Ultrasonic(int pin)
{
    this_pin = pin;
}
/*Begin the detection and get the pulse back signal*/
void Ultrasonic::DistanceMeasure(void)
{
    pinMode(this_pin, OUTPUT);
    digitalWrite(this_pin, LOW);
    delayMicroseconds(2);
    digitalWrite(this_pin, HIGH);
    delayMicroseconds(5);
    digitalWrite(this_pin,LOW);
    pinMode(this_pin,INPUT);
    duration = pulseIn(this_pin,HIGH);
}
/*The measured distance from the range 0 to 400 Centimeters*/
double Ultrasonic::microsecondsToCentimeters(void)
{
    return duration/29.0/2.0;
}
/*The measured distance from the range 0 to 157 Inches*/
double Ultrasonic::microsecondsToInches(void)
{
    return duration/74.0/2.0;
}

Ultrasonic ultrasonic(6);
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    double RangeInInches;
    double RangeInCentimeters;
    ultrasonic.DistanceMeasure();// get the current signal time;
    RangeInInches = ultrasonic.microsecondsToInches();//convert the time to inches;
    RangeInCentimeters = ultrasonic.microsecondsToCentimeters();//convert the time to centimeters
    Serial.println("The distance to obstacles in front is: ");
    Serial.print(RangeInInches);//0~157 inches
    Serial.println(" inch");
    Serial.print(RangeInCentimeters);//0~400cm
    Serial.println(" cm");
    delay(1000);
}
```

下面是两个具体使用超声波传感器来完成一些简单任务的示例代码，两个程序都是进行避障，第一个处理的方法很简单，遇到障碍物就左转(右转也是一样的），直到前方没有障碍物为止，第二个程序稍微完善一些，遇到障碍物会停下来，左右扭头判断左右障碍物的距离，然后会选择距离较远的一边转向，代码基本是之前做过的东西的组合，程序内有注释说明，不再做过多的讲解。

```
#include <MPU6050_6Axis_MotionApps20.h>

#define SERVOMIN 180*4  //最小电机占空比计数
#define SERVOMAX 620*4  //最大电机占空比计数

uint8_t devStatus;
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();
MPU6050 mpu;


//前进步态
  char trF[36][8] = {
  61,  68,  54,  61, -26, -39, -13, -26,
  66,  61,  58,  55, -26, -39, -13, -26,
  70,  54,  61,  49, -26, -37, -12, -25,
  73,  46,  65,  43, -24, -35, -11, -23,
  77,  39,  67,  36, -23, -33, -10, -21,
  80,  33,  70,  31, -21, -30,  -8, -19,
  83,  26,  72,  25, -19, -26,  -6, -15,
  85,  19,  74,  18, -16, -20,  -3,  -9,
  86,  16,  76,  14, -13, -12,  -1,   0,
  87,  19,  76,  16,  -9, -11,   3,   2,
  93,  24,  82,  21, -16, -14,  -5,  -2,
  96,  29,  85,  26, -24, -17, -12,  -5,
  95,  34,  84,  30, -28, -20, -17,  -7,
  92,  39,  81,  35, -31, -22, -20,  -9,
  89,  44,  79,  39, -34, -24, -22, -11,
  84,  49,  74,  44, -36, -25, -24, -12,
  79,  54,  70,  48, -38, -26, -26, -13,
  72,  59,  65,  52, -39, -26, -26, -13,
  65,  64,  59,  56, -39, -26, -26, -13,
  60,  68,  54,  60, -38, -26, -26, -13,
  52,  72,  48,  62, -37, -25, -25, -12,
  45,  75,  41,  66, -35, -24, -23, -11,
  37,  79,  35,  69, -32, -22, -20,  -9,
  30,  81,  29,  71, -29, -20, -17,  -7,
  24,  83,  24,  73, -25, -17, -14,  -4,
  19,  85,  18,  75, -19, -14,  -7,  -2,
  16,  87,  14,  76, -10, -11,   1,   1,
  20,  90,  17,  79, -12, -12,   1,   0,
  25,  96,  22,  85, -15, -20,  -2,  -9,
  30,  97,  26,  85, -18, -26,  -5, -15,
  35,  94,  31,  83, -20, -30,  -8, -18,
  40,  91,  36,  80, -23, -33, -10, -21,
  45,  86,  40,  76, -24, -36, -11, -24,
  51,  81,  45,  72, -25, -37, -12, -25,
  55,  76,  49,  68, -26, -38, -13, -26,
  60,  70,  53,  62, -26, -39, -13, -26,
};

//左转步态
  char trL[29][8]  = {
 62,  69,  54,  57, -28, -40, -13, -18,
  63,  60,  59,  55, -28, -40, -13, -18,
```

```
  64,  51,  62,  53, -28, -37, -12, -18,
  65,  42,  67,  52, -28, -34, -10, -17,
  67,  33,  70,  50, -27, -30,  -8, -17,
  69,  23,  73,  49, -27, -22,  -5, -17,
  70,  18,  75,  46, -27, -14,  -2, -15,
  72,  17,  76,  45, -27,  -7,   2, -14,
  73,  24,  82,  46, -29, -13,  -5, -14,
  72,  29,  85,  48, -30, -17, -14, -14,
  72,  36,  83,  49, -30, -20, -18, -14,
  70,  42,  79,  51, -31, -22, -22, -15,
  68,  49,  74,  52, -31, -25, -24, -15,
  66,  55,  69,  53, -31, -26, -26, -15,
  65,  61,  62,  55, -31, -26, -26, -15,
  62,  67,  55,  56, -31, -26, -26, -15,
  61,  72,  48,  57, -31, -25, -25, -15,
  59,  77,  40,  58, -31, -25, -23, -15,
  57,  81,  32,  60, -31, -22, -19, -14,
  55,  84,  25,  61, -30, -19, -15, -15,
  51,  87,  18,  61, -28, -16,  -7, -14,
  51,  88,  15,  63, -25, -11,   2, -14,
  52,  96,  20,  63, -27, -19,  -1, -14,
  54,  98,  26,  64, -27, -27,  -5, -17,
  55,  97,  31,  63, -27, -32,  -8, -17,
  56,  93,  37,  61, -27, -36, -10, -17,
  58,  87,  42,  60, -28, -39, -12, -17,
  60,  79,  48,  59, -28, -40, -13, -18,
  61,  70,  53,  58, -28, -40, -13, -18,
};

//右转步态
  char trR[29][8]  = {
  61,  64,  55,  61, -26, -31, -15, -26,
  67,  62,  56,  54, -26, -31, -15, -26,
  72,  60,  57,  47, -25, -31, -15, -24,
  77,  59,  58,  39, -25, -31, -15, -22,
  81,  56,  60,  31, -22, -30, -14, -19,
  84,  54,  61,  24, -19, -30, -15, -14,
  87,  51,  61,  16, -16, -27, -14,  -5,
  88,  51,  63,  15, -11, -25, -14,   2,
  96,  52,  63,  21, -19, -27, -14,  -2,
  98,  54,  64,  26, -27, -27, -17,  -5,
  97,  55,  63,  32, -32, -27, -17,  -8,
  93,  57,  61,  38, -36, -27, -17, -10,
  87,  58,  60,  44, -39, -28, -17, -12,
  79,  60,  59,  49, -40, -28, -18, -13,
  70,  62,  58,  54, -40, -28, -18, -13,
  62,  63,  55,  59, -40, -28, -18, -13,
  52,  64,  54,  62, -37, -28, -18, -12,
  43,  65,  52,  67, -35, -28, -17, -10,
  34,  67,  51,  70, -31, -27, -17,  -8,
  26,  69,  49,  73, -25, -27, -17,  -5,
  18,  70,  46,  75, -15, -27, -15,  -2,
  17,  72,  45,  76,  -7, -27, -14,   2,
  22,  73,  46,  82, -12, -29, -14,  -5,
  28,  72,  48,  85, -16, -30, -14, -14,
  35,  72,  49,  83, -19, -30, -14, -18,
  41,  70,  50,  79, -22, -31, -14, -22,
  48,  68,  52,  74, -25, -31, -15, -24,
  54,  66,  53,  69, -26, -31, -15, -26,
  60,  65,  54,  62, -26, -31, -15, -26,
 };


//校准电机误差
char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  8,  8, 8,  -6, 0,  3};

//关节映射表
byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

//旋转方向
int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };

//角度转化为占空比计数时需要的偏移量
int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };


 //将角度转化为占空比计数
int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) * 7.04  * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}

class Ultrasonic
{
    public:
    Ultrasonic(int pin);
    void DistanceMeasure(void);
    double microsecondsToCentimeters(void);
    double microsecondsToInches(void);
    private:
    int this_pin;//pin number of Arduino that is connected with SIG pin of Ultrasonic Ranger.
    long duration;// the Pulse time received;
};
Ultrasonic::Ultrasonic(int pin)
{
    this_pin = pin;
}
/*Begin the detection and get the pulse back signal*/
void Ultrasonic::DistanceMeasure(void)
{
    pinMode(this_pin, OUTPUT);
    digitalWrite(this_pin, LOW);
    delayMicroseconds(2);
    digitalWrite(this_pin, HIGH);
    delayMicroseconds(5);
    digitalWrite(this_pin,LOW);
    pinMode(this_pin,INPUT);
    duration = pulseIn(this_pin,HIGH);
}
/*The measured distance from the range 0 to 400 Centimeters*/
double Ultrasonic::microsecondsToCentimeters(void)
{
    return duration/29.0/2.0;
}
/*The measured distance from the range 0 to 157 Inches*/
double Ultrasonic::microsecondsToInches(void)
{
    return duration/74.0/2.0;
}


Ultrasonic ultrasonic(6);
void setup()
{
    Serial.begin(9600);
      pwm.begin();
     pwm.setPWMFreq(240);
```

```
}



void loop()
{
    double RangeInCentimeters;
    ultrasonic.DistanceMeasure();// get the current signal time;
    RangeInCentimeters = ultrasonic.microsecondsToCentimeters();//转换为厘米

      if(RangeInCentimeters>20)//障碍物距离大于20时，保持直行
      {
    for(int i=0;i<=35;i++){
    for(int j=0;j<=7;j++){
      //根据步态数据设置每个关节的转角
      pwm.setPWM(pins[8+j],0,S2P(    (int(trF[i][j]))   ,8+j   ));
    }
    }
      }
      else//当超声波检测到障碍物距离过近时，左转，知道距离再次大于20
      {
         for(int i=0;i<=35;i++){
    for(int j=0;j<=7;j++){
      //根据步态数据设置每个关节的转角
      pwm.setPWM(pins[8+j],0,S2P(    (int(trL[i][j]))   ,8+j   ));
    }
    }
      }
    Serial.println("The distance to obstacles in front is: ");
    Serial.print(RangeInCentimeters);//0~400cm
    Serial.println(" cm");

}



#include <MPU6050_6Axis_MotionApps20.h>

#define SERVOMIN 180*4  //最小电机占空比计数
#define SERVOMAX 620*4  //最大电机占空比计数

uint8_t devStatus;
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();
MPU6050 mpu;


//前进步态
  char trF[36][8] = {
  61,  68,  54,  61, -26, -39, -13, -26,
  66,  61,  58,  55, -26, -39, -13, -26,
  70,  54,  61,  49, -26, -37, -12, -25,
  73,  46,  65,  43, -24, -35, -11, -23,
  77,  39,  67,  36, -23, -33, -10, -21,
  80,  33,  70,  31, -21, -30,  -8, -19,
  83,  26,  72,  25, -19, -26,  -6, -15,
  85,  19,  74,  18, -16, -20,  -3,  -9,
  86,  16,  76,  14, -13, -12,  -1,   0,
  87,  19,  76,  16,  -9, -11,   3,   2,
  93,  24,  82,  21, -16, -14,  -5,  -2,
  96,  29,  85,  26, -24, -17, -12,  -5,
  95,  34,  84,  30, -28, -20, -17,  -7,
  92,  39,  81,  35, -31, -22, -20,  -9,
  89,  44,  79,  39, -34, -24, -22, -11,
  84,  49,  74,  44, -36, -25, -24, -12,
  79,  54,  70,  48, -38, -26, -26, -13,
  72,  59,  65,  52, -39, -26, -26, -13,
  65,  64,  59,  56, -39, -26, -26, -13,
  60,  68,  54,  60, -38, -26, -26, -13,
  52,  72,  48,  62, -37, -25, -25, -12,
  45,  75,  41,  66, -35, -24, -23, -11,
  37,  79,  35,  69, -32, -22, -20,  -9,
  30,  81,  29,  71, -29, -20, -17,  -7,
  24,  83,  24,  73, -25, -17, -14,  -4,
  19,  85,  18,  75, -19, -14,  -7,  -2,
  16,  87,  14,  76, -10, -11,   1,   1,
  20,  90,  17,  79, -12, -12,   1,   0,
  25,  96,  22,  85, -15, -20,  -2,  -9,
  30,  97,  26,  85, -18, -26,  -5, -15,
  35,  94,  31,  83, -20, -30,  -8, -18,
  40,  91,  36,  80, -23, -33, -10, -21,
  45,  86,  40,  76, -24, -36, -11, -24,
  51,  81,  45,  72, -25, -37, -12, -25,
  55,  76,  49,  68, -26, -38, -13, -26,
  60,  70,  53,  62, -26, -39, -13, -26,
};

//左转步态
  char trL[29][8]  = {
 62,  69,  54,  57, -28, -40, -13, -18,
  63,  60,  59,  55, -28, -40, -13, -18,
  64,  51,  62,  53, -28, -37, -12, -18,
  65,  42,  67,  52, -28, -34, -10, -17,
  67,  33,  70,  50, -27, -30,  -8, -17,
  69,  23,  73,  49, -27, -22,  -5, -17,
  70,  18,  75,  46, -27, -14,  -2, -15,
  72,  17,  76,  45, -27,  -7,   2, -14,
  73,  24,  82,  46, -29, -13,  -5, -14,
  72,  29,  85,  48, -30, -17, -14, -14,
  72,  36,  83,  49, -30, -20, -18, -14,
  70,  42,  79,  51, -31, -22, -22, -15,
  68,  49,  74,  52, -31, -25, -24, -15,
  66,  55,  69,  53, -31, -26, -26, -15,
  65,  61,  62,  55, -31, -26, -26, -15,
  62,  67,  55,  56, -31, -26, -26, -15,
  61,  72,  48,  57, -31, -25, -25, -15,
  59,  77,  40,  58, -31, -25, -23, -15,
  57,  81,  32,  60, -31, -22, -19, -14,
  55,  84,  25,  61, -30, -19, -15, -15,
  51,  87,  18,  61, -28, -16,  -7, -14,
  51,  88,  15,  63, -25, -11,   2, -14,
  52,  96,  20,  63, -27, -19,  -1, -14,
  54,  98,  26,  64, -27, -27,  -5, -17,
  55,  97,  31,  63, -27, -32,  -8, -17,
  56,  93,  37,  61, -27, -36, -10, -17,
  58,  87,  42,  60, -28, -39, -12, -17,
  60,  79,  48,  59, -28, -40, -13, -18,
  61,  70,  53,  58, -28, -40, -13, -18,
};

//右转步态
  char trR[29][8]  = {
  61,  64,  55,  61, -26, -31, -15, -26,
  67,  62,  56,  54, -26, -31, -15, -26,
  72,  60,  57,  47, -25, -31, -15, -24,
  77,  59,  58,  39, -25, -31, -15, -22,
  81,  56,  60,  31, -22, -30, -14, -19,
  84,  54,  61,  24, -19, -30, -15, -14,
  87,  51,  61,  16, -16, -27, -14,  -5,
  88,  51,  63,  15, -11, -25, -14,   2,
  96,  52,  63,  21, -19, -27, -14,  -2,
  98,  54,  64,  26, -27, -27, -17,  -5,
  97,  55,  63,  32, -32, -27, -17,  -8,
```

```
  93,  57,  61,  38, -36, -27, -17, -10,
  87,  58,  60,  44, -39, -28, -17, -12,
  79,  60,  59,  49, -40, -28, -18, -13,
  70,  62,  58,  54, -40, -28, -18, -13,
  62,  63,  55,  59, -40, -28, -18, -13,
  52,  64,  54,  62, -37, -28, -18, -12,
  43,  65,  52,  67, -35, -28, -17, -10,
  34,  67,  51,  70, -31, -27, -17,  -8,
  26,  69,  49,  73, -25, -27, -17,  -5,
  18,  70,  46,  75, -15, -27, -15,  -2,
  17,  72,  45,  76,  -7, -27, -14,   2,
  22,  73,  46,  82, -12, -29, -14,  -5,
  28,  72,  48,  85, -16, -30, -14, -14,
  35,  72,  49,  83, -19, -30, -14, -18,
  41,  70,  50,  79, -22, -31, -14, -22,
  48,  68,  52,  74, -25, -31, -15, -24,
  54,  66,  53,  69, -26, -31, -15, -26,
  60,  65,  54,  62, -26, -31, -15, -26,
};


//校准电机误差
char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  8,  8, 8,  -6, 0,  3};

//关节映射表
byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

//旋转方向
int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };
//角度转化为占空比计数时需要的偏移量
int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };


 //将角度转化为占空比计数
int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) * 7.04  * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}

class Ultrasonic
{
    public:
    Ultrasonic(int pin);
    void DistanceMeasure(void);
    double microsecondsToCentimeters(void);
    double microsecondsToInches(void);
    private:
    int this_pin;//pin number of Arduino that is connected with SIG pin of Ultrasonic Ranger.
    long duration;// the Pulse time received;
};
Ultrasonic::Ultrasonic(int pin)
{
    this_pin = pin;
}
/*Begin the detection and get the pulse back signal*/
void Ultrasonic::DistanceMeasure(void)
{
    pinMode(this_pin, OUTPUT);
    digitalWrite(this_pin, LOW);
    delayMicroseconds(2);
    digitalWrite(this_pin, HIGH);
    delayMicroseconds(5);
    digitalWrite(this_pin,LOW);
    pinMode(this_pin,INPUT);
    duration = pulseIn(this_pin,HIGH);
}
/*The measured distance from the range 0 to 400 Centimeters*/
double Ultrasonic::microsecondsToCentimeters(void)
{
    return duration/29.0/2.0;
}
/*The measured distance from the range 0 to 157 Inches*/
double Ultrasonic::microsecondsToInches(void)
{
    return duration/74.0/2.0;
}


Ultrasonic ultrasonic(6);
void setup()
{
    Serial.begin(9600);
      pwm.begin();
     pwm.setPWMFreq(240);
}


double left=0;
double right=0;


void loop()
{

    double RangeInCentimeters;
    ultrasonic.DistanceMeasure();// get the current signal time;
    RangeInCentimeters = ultrasonic.microsecondsToCentimeters();//convert the time to centimeters


     pwm.setPWM(pins[0],0,S2P(0,0)); //头转回中间部位


      if(RangeInCentimeters>20)//如果目前测得的障碍物距离大于20，保持直行
      {
    for(int k=0;k<=10;k++){
    for(int i=0;i<=35;i++){
    for(int j=0;j<=7;j++){
      //根据步态数据设置每个关节的转角
      pwm.setPWM(pins[8+j],0,S2P(    (int(trF[i][j]))   ,8+j   ));
    }
    }}
      }
      else//否则，停下来，左右转头
      {
          pwm.setPWM(pins[0],0,S2P(90,0)); //头左转
           delay(1000);
           ultrasonic.DistanceMeasure(); //测距
           left = ultrasonic.microsecondsToCentimeters(); //转换成厘米

          pwm.setPWM(pins[0],0,S2P(-90,0)); //头右转
          delay(1000);
```

```
            ultrasonic.DistanceMeasure(); //测距
            right = ultrasonic.microsecondsToCentimeters(); //转换成厘米


        if(left>right)//左侧障碍物离得更远则左转
        {
          for(int k=0;k<=10;k++){
         for(int i=0;i<=35;i++){
          for(int j=0;j<=7;j++){
      //根据步态数据设置每个关节的转角
      pwm.setPWM(pins[8+j],0,S2P(    (int(trL[i][j]))   ,8+j   ));
    }}
    }}
    else//右侧障碍物离得更远则右转
    {
      for(int k=0;k<=10;k++){
      for(int i=0;i<=35;i++){
         for(int j=0;j<=7;j++){
      //根据步态数据设置每个关节的转角
      pwm.setPWM(pins[8+j],0,S2P(    (int(trR[i][j]))   ,8+j   ));
    }}
    }
    }
      }

    Serial.println("The distance to obstacles in front is: ");
    Serial.print(RangeInCentimeters);//0~400cm
    Serial.println(" cm");
}
```

4）PIR人体红外检测模块

该模块利用红外线来判断周围是否有人接近，使用方式也非常简单，直接读取数字信号即可，0表示没有，1表示有。

```
int pirPin = 9;

int pirValue;
int sec = 0;

void setup()
{
    pinMode(pirPin, INPUT);
    Serial.begin(9600);
}

void loop()
{
    pirValue = digitalRead(pirPin);

   //  用于显示传感器输出状态
     sec += 1;
     Serial.print("Second: ");
     Serial.print(sec);
     Serial.print("PIR value: ");
     Serial.print(pirValue);
     Serial.print('\n');
     delay(1000);
}
```

下面是两个具体使用PIR人体红外传感器来完成一些简单任务的示例代码，第一个是如果检测人体就坐下，没有检测到就站起来，第二个是只有在检测到人体后才会去读取摄像头的数据，因为PIR人体红外传感器的能耗要比摄像头小很多，这样能增加机器狗的运行时间，代码基本是之前做过的东西的组合，程序内有注释说明，不再做过多的讲解。

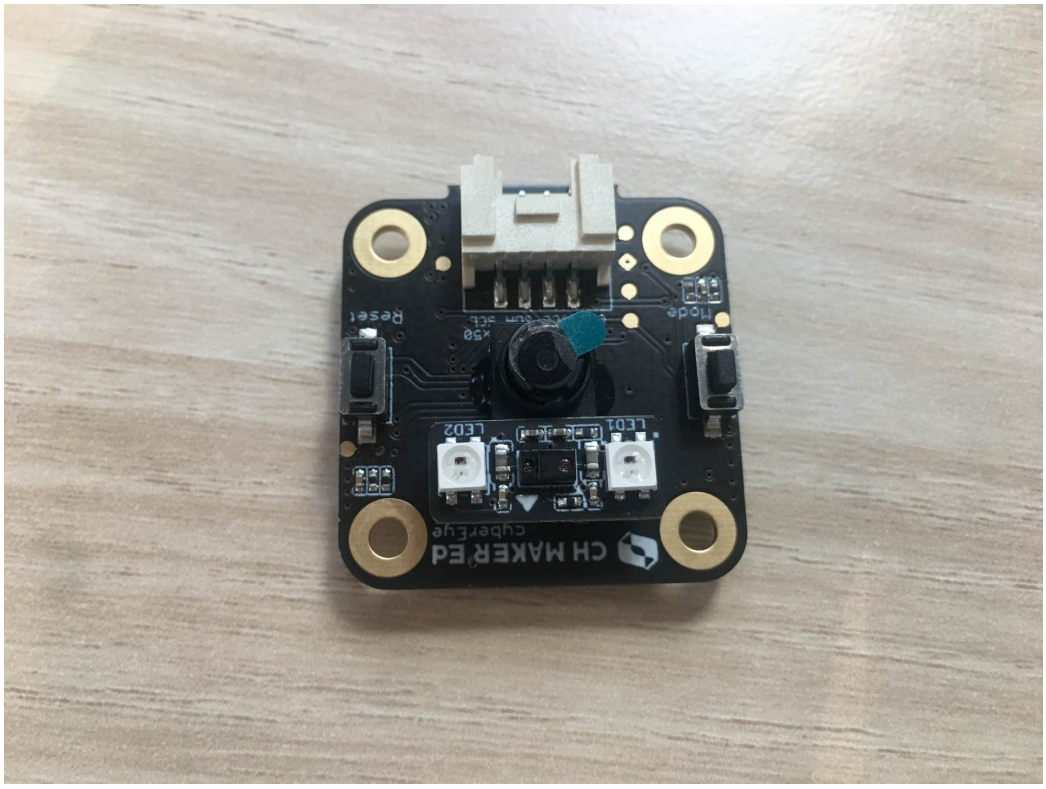
```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <avr/pgmspace.h>

#define SERVOMIN 180*4
#define SERVOMAX 620*4
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();

char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  10,  8, 8,  -3, 8,  8};

byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };

int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };
int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) * 7.04
      * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}


int pirPin = 7;

int pirValue;
int sec = 0;

void setup()
{
    pinMode(pirPin, INPUT);
    Serial.begin(9600);
    pwm.begin();
    pwm.setPWMFreq(240);
}

void loop()
{
    pirValue = digitalRead(pirPin);

   //  以下注释可以观察传感器输出状态
     sec += 1;
     Serial.print("Second: ");
     Serial.print(sec);
     Serial.print("PIR value: ");
     Serial.print(pirValue);
     Serial.print('\n');
     delay(1000);


     if(pirValue==1) //检测到人体，坐下
     {
    for(int j=0;j<=7;j++){
        pwm.setPWM(pins[8+j],0,S2P(0,8+j));
    }
    delay(5000);
    }

    if(pirValue==0)  //没检测到人体，站起来
    {
    for(int j=0;j<=7;j++){
        pwm.setPWM(pins[8+j],0,S2P(30,8+j));
    }
    delay(5000);
    }
}
```

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <avr/pgmspace.h>

#define SERVOMIN 180*4
#define SERVOMAX 620*4
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();
#define I2C_MODE
#define MU_ADDRESS    0x60
#include <Arduino.h>
#include <MuVisionSensor.h>
#ifdef I2C_MODE
#include <Wire.h>
#endif
#ifdef SERIAL_MODE
#include <SoftwareSerial.h>
#define TX_PIN 2
#define RX_PIN 3
SoftwareSerial mySerial(RX_PIN, TX_PIN);
#endif
MuVisionSensor Mu(MU_ADDRESS);

char trF[36][8] = {
  61,  68,  54,  61, -26, -39, -13, -26,
  66,  61,  58,  55, -26, -39, -13, -26,
  70,  54,  61,  49, -26, -37, -12, -25,
```

```
  73,  46,  65,  43, -24, -35, -11, -23,
  77,  39,  67,  36, -23, -33, -10, -21,
  80,  33,  70,  31, -21, -30,  -8, -19,
  83,  26,  72,  25, -19, -26,  -6, -15,
  85,  19,  74,  18, -16, -20,  -3,  -9,
  86,  16,  76,  14, -13, -12,  -1,   0,
  87,  19,  76,  16,  -9, -11,   3,   2,
  93,  24,  82,  21, -16, -14,  -5,  -2,
  96,  29,  85,  26, -24, -17, -12,  -5,
  95,  34,  84,  30, -28, -20, -17,  -7,
  92,  39,  81,  35, -31, -22, -20,  -9,
  89,  44,  79,  39, -34, -24, -22, -11,
  84,  49,  74,  44, -36, -25, -24, -12,
  79,  54,  70,  48, -38, -26, -26, -13,
  72,  59,  65,  52, -39, -26, -26, -13,
  65,  64,  59,  56, -39, -26, -26, -13,
  60,  68,  54,  60, -38, -26, -26, -13,
  52,  72,  48,  62, -37, -25, -25, -12,
  45,  75,  41,  66, -35, -24, -23, -11,
  37,  79,  35,  69, -32, -22, -20,  -9,
  30,  81,  29,  71, -29, -20, -17,  -7,
  24,  83,  24,  73, -25, -17, -14,  -4,
  19,  85,  18,  75, -19, -14,  -7,  -2,
  16,  87,  14,  76, -10, -11,   1,   1,
  20,  90,  17,  79, -12, -12,   1,   0,
  25,  96,  22,  85, -15, -20,  -2,  -9,
  30,  97,  26,  85, -18, -26,  -5, -15,
  35,  94,  31,  83, -20, -30,  -8, -18,
  40,  91,  36,  80, -23, -33, -10, -21,
  45,  86,  40,  76, -24, -36, -11, -24,
  51,  81,  45,  72, -25, -37, -12, -25,
  55,  76,  49,  68, -26, -38, -13, -26,
  60,  70,  53,  62, -26, -39, -13, -26,
};


char cal[16]= {0,  0,  0,  0,  0,  0,  0,  3,  3,  10,  8, 8,  -3, 8,  8};

byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };

int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };
int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i]) * 7.04  * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}


int pirPin = 7;

int pirValue;
int sec = 0;

void setup()
{
    pinMode(pirPin, INPUT);
    Serial.begin(9600);
    pwm.begin();
    pwm.setPWMFreq(240);
 uint8_t err = 0;
    #ifdef I2C_MODE
  Wire.begin();
  // initialized MU on the I2C port
  err = Mu.begin(&Wire);
#elif defined SERIAL_MODE
  mySerial.begin(9600);
  // initialized MU on the soft serial port
  err = Mu.begin(&mySerial);
#endif
  if (err == MU_OK) {
    Serial.println("MU initialized.");
  } else {
    do {
      Serial.println("fail to initialize MU! Please check protocol "
                     "version or make sure MU is working on the "
                     "correct port with correct mode.");
      delay(5000);
    } while (1);
  }
  // enable vision: traffic card
  Mu.VisionBegin(VISION_TRAFFIC_CARD_DETECT);
}

void loop()
{
    long time_start = millis();
    pirValue = digitalRead(pirPin);

   //  以下注释可以观察传感器输出状态
      sec += 1;
      Serial.print("Second: ");
      Serial.print(sec);
      Serial.print("PIR value: ");
      Serial.print(pirValue);
      Serial.print('\n');
      delay(1000);


      if(pirValue==1)//只有当PIR人体检测模块检测到人体时，才会去读取和处理摄像头数据，从而达到节省能耗的效果
      {
    if (Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kStatus)) {                  // update vision result and get status, 0: undetected, other: detected
    Serial.println("vision traffic card detected:");
    Serial.print("x = ");
    Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kXValue));       // get vision result: x axes value
    Serial.print("y = ");
    Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kYValue));       // get vision result: y axes value
    Serial.print("width = ");
    Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kWidthValue));   // get vision result: width value
    Serial.print("height = ");
    Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kHeightValue));  // get vision result: height value
    Serial.print("label = ");
    switch (Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kLabel)) {              // get vision result: label value
      case MU_TRAFFIC_CARD_FORWARD:
        Serial.println("forward");
        break;
      case MU_TRAFFIC_CARD_LEFT:
        Serial.println("left");
        break;
      case MU_TRAFFIC_CARD_RIGHT:
        Serial.println("right");
        break;
      case MU_TRAFFIC_CARD_TURN_AROUND:
        Serial.println("turn around");
        break;
```

```
        break;
      case MU_TRAFFIC_CARD_PARK:
        Serial.println("park");
        break;
      default:
        Serial.print("unknow card type: ");
        Serial.println(Mu.GetValue(VISION_TRAFFIC_CARD_DETECT, kLabel));
        break;
    }
  } else {
    Serial.println("vision shape card undetected.");
  }



    for(int i=0;i<=35;i++){
    for(int j=0;j<=7;j++){
      pwm.setPWM(pins[8+j],0,S2P(    (int(trF[i][j]))   ,8+j   ));

    }
    delay(5000);
    }

    if(pirValue==0)//没有检测到人体，不使用摄像头，保持站立姿势不动
    {
    for(int j=0;j<=7;j++){
        pwm.setPWM(pins[8+j],0,S2P(30,8+j));
    }
    delay(5000);
    }
}}
```

5）摄像头



上个部分已经比较详细的介绍了摄像头如何使用，这里单独介绍一下光线检测和接近检测两个功能。核心思路就是调用拓展库提供的函数读取我们需要的数据。

```
#ifdef I2C_MODE
#include <Wire.h>
#endif
#ifdef SERIAL_MODE
#include <SoftwareSerial.h>
#define TX_PIN 2
#define RX_PIN 3
SoftwareSerial mySerial(RX_PIN, TX_PIN);
#endif
MuVisionSensor Mu(MU_ADDRESS);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  uint8_t err = 0;
#ifdef I2C_MODE
  Wire.begin();
  // initialized MU on the I2C port
  err = Mu.begin(&Wire);
#elif defined SERIAL_MODE
  mySerial.begin(9600);
  // initialized MU on the soft serial port
  err = Mu.begin(&mySerial);
```

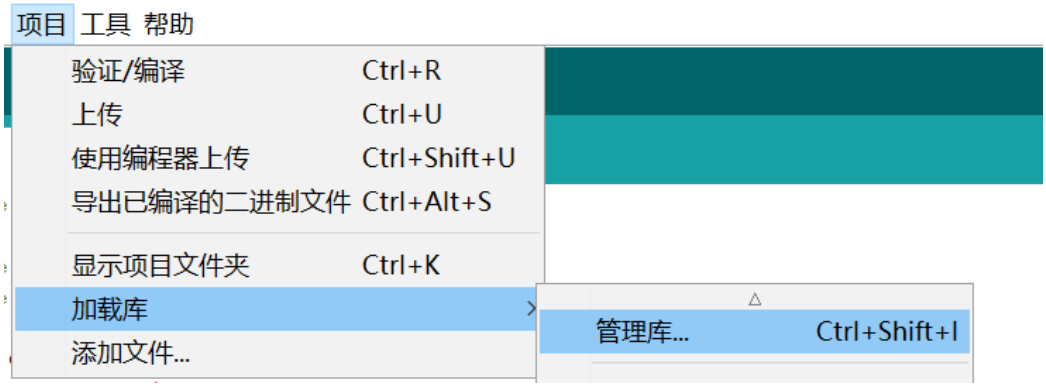
```
#endif
  if (err == MU_OK) {
    Serial.println("MU initialized.");
  } else {
    do {
      Serial.print("fail to initialize MU! Please check device ID is: ");
      Serial.print(MU_DEVICE_ID);
      Serial.println(" or make sure MU is working on the correct port with correct mode.");
      delay(5000);
    } while (1);
  }
  // enable proximity/color/ambient light detection
  Mu.LsBegin(LS_PROXIMITY_ENABLE | LS_COLOR_ENABLE | LS_AMBIENT_LIGHT_ENABLE);
  // light set sensitivity, default value is kSensitivity2
  Mu.LsSetSensitivity(kSensitivity2);
  // enable white balance for color detection
  Mu.LsWhiteBalanceEnable();
}

void loop() {
  // 利用拓展库读取数据
  uint8_t r = Mu.LsReadColor(kLsColorRed);
  uint8_t g = Mu.LsReadColor(kLsColorGreen);
  uint8_t b = Mu.LsReadColor(kLsColorBlue);
  uint16_t h = Mu.LsReadColor(kLsColorHue);
  uint8_t s = Mu.LsReadColor(kLsColorSaturation);
  uint8_t v = Mu.LsReadColor(kLsColorValue);
  uint8_t proximity = Mu.LsReadProximity();
  uint8_t color_label = Mu.LsReadColor(kLsColorLabel);
  uint16_t als = Mu.LsReadAmbientLight();
  //颜色检测
  Serial.print("color=");
  switch(color_label) {
    case MU_COLOR_BLACK:
      Serial.print("BLACK");
      break;
    case MU_COLOR_WHITE:
      Serial.print("WHITE");
      break;
    case MU_COLOR_RED:
      Serial.print("RED");
      break;
    case MU_COLOR_YELLOW:
      Serial.print("YELLOW");
      break;
    case MU_COLOR_GREEN:
      Serial.print("GREEN");
      break;
    case MU_COLOR_CYAN:
      Serial.print("CYAN");
      break;
    case MU_COLOR_BLUE:
      Serial.print("BLUE");
      break;
    case MU_COLOR_PURPLE:
      Serial.print("PURPLE");
      break;
    default:
      Serial.print("unknow color[");
      Serial.print(color_label);
      Serial.print(']');
      break;
  }

  Serial.print("proximity=");
  Serial.println(proximity);      //接近检测
  Serial.print("ambient light=");
  Serial.println(als);  //光线检测
}
```

还有一个利用检测人体的功能做的例子有，当检测检测到人体小狗踏步，否则停下来。

```
#ifdef I2C_MODE
#include <Wire.h>
#endif
#ifdef SERIAL_MODE
#include <SoftwareSerial.h>
#define TX_PIN 2
#define RX_PIN 3
SoftwareSerial mySerial(RX_PIN, TX_PIN);
#endif
MuVisionSensor Mu(MU_ADDRESS);

#include <SoftwareSerial.h>
#define SOFTSERIAL_RX_PIN  6
#define SOFTSERIAL_TX_PIN  7
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <avr/pgmspace.h>

#define SERVOMIN 180*4
#define SERVOMAX 620*4
Adafruit_PWMServoDriver pwm=Adafruit_PWMServoDriver();


char vt[21][8] = {
  57,  43,  60,  47, -18,   7, -17,   7,
  50,  43,  53,  47,  -5,   7,  -5,   7,
  43,  43,  47,  47,   7,   7,   7,   7,
  43,  43,  47,  47,   7,   7,   7,   7,
  43,  43,  47,  47,   7,   7,   7,   7,
  43,  47,  47,  50,   7,   0,   7,   0,
  43,  54,  47,  58,   7, -13,   7, -13,
  43,  60,  47,  65,   7, -25,   7, -25,
  43,  66,  47,  71,   7, -35,   7, -35,
  43,  63,  47,  67,   7, -30,   7, -29,
  43,  57,  47,  60,   7, -18,   7, -17,
  43,  50,  47,  53,   7,  -5,   7,  -5,
  43,  43,  47,  47,   7,   7,   7,   7,
  43,  43,  47,  47,   7,   7,   7,   7,
  43,  43,  47,  47,   7,   7,   7,   7,
  43,  43,  47,  47,   7,   7,   7,   7,
  47,  43,  50,  47,   0,   7,   0,   7,
  54,  43,  58,  47, -13,   7, -13,   7,
  60,  43,  65,  47, -25,   7, -25,   7,
  66,  43,  71,  47, -35,   7, -35,   7,
  63,  43,  67,  47, -30,   7, -29,   7,
};

char cal[16]= {0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  10,  8, 8,  -3, 8,  8};

byte pins[] = {12, 11, 3, 4,
               13, 10, 5, 2,
               14, 9, 6, 1,
               15, 8, 7, 0
              };

int8_t rotationDirections[] = {1, -1, 1, 1,
                               1, -1, 1, -1,
                               1, -1, -1, 1,
                               -1, 1, 1, -1
                              };

int8_t middleShifts[] = {0, 15, 0, 0,
                         -45, -45, -45, -45,
                         45, 45, -45, -45,
                         -75, -75, -75, -75
                        };
int S2P(int angle,int i){
  float p=0.0;
  p = SERVOMIN + 880 + float(middleShifts[i] + cal[i])
      * 7.04  * rotationDirections[i] + angle * 7.04 * rotationDirections[i];
  return int(p);
}



void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  uint8_t err = 0;
#ifdef I2C_MODE
  Wire.begin();
  err = Mu.begin(&Wire);                                          // initialized MU on I2C port
#elif defined SERIAL_MODE
  mySerial.begin(9600);
  err = Mu.begin(&mySerial);                                      // initialized MU on soft serial port
#endif
  if (err == MU_OK) {
    Serial.println("MU initialized.");
  } else {
    do {
      Serial.println("fail to initialize MU! Please check protocol "
                     "version or make sure MU is working on the "
                     "correct port with correct mode.");
      delay(5000);
    } while (1);
  }
  // enable vision: body detect
  Mu.VisionBegin(VISION_BODY_DETECT);                         // enable vision body
}

void loop() {
  // put your main code here, to run repeatedly:
  long time_start = millis();

  // read result
  if (Mu.GetValue(VISION_BODY_DETECT, kStatus)) {                 // update vision result and get status, 0: undetected, other: detected
    Serial.println("vision body detected:");
    Serial.print("x = ");
    Serial.println(Mu.GetValue(VISION_BODY_DETECT, kXValue));       // get vision result: x axes value
    Serial.print("y = ");
    Serial.println(Mu.GetValue(VISION_BODY_DETECT, kYValue));       // get vision result: y axes value
    Serial.print("width = ");
    Serial.println(Mu.GetValue(VISION_BODY_DETECT, kWidthValue));   // get vision result: width value
    Serial.print("height = ");
    Serial.println(Mu.GetValue(VISION_BODY_DETECT, kHeightValue));  // get vision result: height value
    //如果检测到人体，则机器狗进行踏步
   for(int i=0;i<=35;i++){
    for(int j=0;j<=7;j++){
      pwm.setPWM(pins[8+j],0,S2P(    (int(vt[i][j]))   ,8+j  ));

    }
    delay(5000);

  }} else {
    Serial.println("vision body undetected.");
  }
  Serial.print("fps = ");
  Serial.println(1000/(millis()-time_start));
  Serial.println();
}
```

6)OLED显示屏

使用OLED显示屏需要先安装一个库，具体方法如下，依次选择项目，加载库，管理库，



搜索该库并安装，

使用方法主要为利用库函数绘制我们需要的图像，或者输出需要的文字，之后使display.display();进行最后的显示。

代码与注释如下：

```
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2

#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH  16



void setup()         //初始化
{
  Serial.begin(9600);
  delay(500);

  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3C (for the 128x64) , 0x3C为I2C协议通讯地址，需根据实际情况更改
}
void loop()
{
  test_SSD1306();     //调用测试函数
}

void test_SSD1306(void)    //测试函数
{
/*-----------------点亮全屏检测屏幕是否有不正常点亮现象-----------------------------*/

  display.fillScreen(WHITE);
  display.display();
  delay(2000);

/*--------------------------------画点 点坐标(64,32)--------------------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.drawPixel(64, 32, WHITE);
  display.display();
  delay(2000);

  /*--------------------------- 画线 从(0,0)到(128,64)----------------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.drawLine(0, 0,128,64, WHITE);
  display.display();
  delay(2000);

/*---------------.画空心矩形  左上角坐标(x0,y0)  右下角坐标(x1,y1)-----------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.drawRect(0,0,128,64,WHITE);
  display.display();
  delay(2000);

/*-----------------------实心矩形----------------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.fillRect(0,0,128,64,WHITE);
  display.display();
  delay(2000);

/*-------------------------画空心圆-------------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.drawCircle(64,32,20,WHITE);
  display.display();
  delay(2000);

/*----------------------画实心圆----------------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.fillCircle(128,64,20,WHITE);
  display.display();
  delay(2000);

/*---------------------画空心三角形-------------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.drawTriangle(64,0,0,63,128,63,WHITE);
  display.display();
  delay(2000);

/*-------------------------画实心三角形-----------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.fillTriangle(64,0,0,63,128,63,WHITE);
  display.display();
  delay(2000);

/*-----------------------空心圆角矩形-------------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.drawRoundRect(0,0,128,64,5,WHITE);
  display.display();
  delay(2000);

/*----------------------画实心圆角矩形-----------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.fillRoundRect(0,0,128,64,5,WHITE);
  display.display();
  delay(2000);

/*-------------------------显示英文 数字---------------------*/
  display.clearDisplay();   // clears the screen and buffer
  display.setTextSize(1); //选择字号
  display.setTextColor(WHITE);  //字体颜色
  display.setCursor(0,0);   //起点坐标
  display.println("Hello, Arduino!");
  display.setTextColor(BLACK, WHITE); // 'inverted' text
  display.println(3.141592);
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.print("0x"); display.println(0xDEADBEEF, HEX);
  display.display();
  delay(2000);
  }
```

利用OLED显示屏我们可以将我们需要的数据直接展示出来，而不需要每一次都去读取串口数据，例如下面的例子里，我们利用OLED显示屏来显示从陀螺仪读取到的转角。

```
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2

#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH  16


#include <MPU6050_6Axis_MotionApps20.h>
uint8_t devStatus;
MPU6050 mpu;

volatile bool mpuInterrupt = false;     // 表示mpu interrupt 引脚是否被拉高

void dmpDataReady() {
  mpuInterrupt = true;  //表示发生了中断
}


void setup()        //初始化
{
  Serial.begin(9600);
  delay(500);
  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3C (for the 128x64) , 0x3C为I2C协议通讯地址，需根据实际情况更改
}
void loop()
{
  test_SSD1306();     //调用测试函数
}



float GR=131.0;//陀螺仪 比例系数
int16_t ax=0,ay=0,az=0,gx=0,gy=0,gz=0;//用来存储getMotion6函数读取到的数据
float sum=0;//用来累加z轴旋转角偏移量
float x,y,z;



void test_SSD1306(void)    //测试函数
{
/*-----------------点亮全屏检测屏幕是否有不正常点亮现象-----------------------------*/

  display.fillScreen(WHITE);
  display.display();
  delay(500);


 mpu.getMotion6(&ax,&ay,&az,&gx,&gy,&gz);//获取陀螺仪数据
 x=gx/GR;//关于x轴的旋转角，即水平的转向
 y=gy/GR;//关于y轴的旋转角，即水平的转向
 z=gz/GR;//关于z轴的旋转角，即水平的转向

  display.clearDisplay();   // clears the screen and buffer
  display.setTextSize(1); //选择字号
  display.setTextColor(WHITE);  //字体颜色
  display.setCursor(0,0);   //起点坐标
  display.print("x: ");   display.println(x);
  display.print("y: ");   display.println(y);
  display.print("z: ");   display.println(z);

  display.display();//显示编辑好的文本
  delay(500);
  }
```