```
#include <Wire.h>
#include <Streaming.h>
#include <PID_v1.h>


double previousLoop = 0;
double dt = 0;
double accelAngle = 0;
double angle = 0;
double gyroAngle = 0;
int motorOut = 0;

double ArrayTest[] = {0,0,0,0,0,0,0,0};

//pid
double Setpoint, Input, Output;
PID myPID(&Input, &Output, &Setpoint, 7 , 0 , .05 , DIRECT);



//WM+ stuff
byte data[6]; //six data bytes
int yaw, pitch, roll; //three axes
int yaw0, pitch0, roll0; //calibration zeroes

//nunchuck stuff
uint8_t outbuf[6]; // array to store arduino output
int cnt = 0;
int joy_x_axis, joy_y_axis, accel_x_axis, accel_y_axis, accel_z_axis;
boolean z_button, c_button;



void setup(){
  Serial.begin(115200);
  Serial << "WM+ and Nunchuck tester" << endl;
  pinMode(5, OUTPUT);
  pinMode(4, OUTPUT);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);
  Wire.begin();
  switchtowmp();
  wmpOn();
  calibrateZeroes();
  switchtonunchuck();
  nunchuck_init();

  pwmSetup();
  myPID.SetMode(AUTOMATIC);
  myPID.SetOutputLimits(-255, 255);
  //how often you want the pid to update in ms
  myPID.SetSampleTime(10); //changing this will require pid to be retuned
}




void loop(){

  //yaw0:7996 Pitch0:8042 Roll0:7696
```

```
  yaw0 = 7996;
  pitch0 = 8042;
  roll0 = 7696 ;




  wait(); //will try to keep every loop running at 100hz or 10ms
  switchtowmp();
  receiveData();
  switchtonunchuck();
  receive_nunchuck_data();


  //This is not the way to calculate angle (brain fart on my part)... Redo it
  accelAngle = (asin((accel_y_axis-305)/450.)*180)*100;
  storeValue(accelAngle);
  accelAngle = averageArray();
  dt = millis() - previousLoop; //move this to right before its used to keep it more accurate
   //gyroAngle = gyroAngle + (roll)/300.*dt;

  angle = .95*(angle+roll*dt/3000.) + .05*((int)accelAngle)/100.;

  //gyroAngle = angle;


  Input = angle;
  Setpoint = -11.0 + 30*analogRead(A0)/1023.; //remake this
  myPID.Compute();



  //storeValue(Output);


  motorOut =  Output;
 /*
 if(abs(angle - Setpoint) < 5){
   myPID.SetOutputLimits(-150, 150);
 }else{
   myPID.SetOutputLimits(-255, 255);
 }
*/



  motorMove(-motorOut,-motorOut);

  /*
  motorMove(-100,0);
  delay(2000);
  motorMove(0,0);
  delay(2000);
  motorMove(100,0);
  delay(2000);
  motorMove(0,0);
  delay(2000);
*/
```

```
  Serial << "roll," << roll/100. << ", \tangle: ," << angle << ",  \taccel,"<< (int)accelAngle/100. << ",\tset point: ," << Setpoint << endl;
  previousLoop = millis();
}




void wait(){
  while(millis() - previousLoop  < 10){
    //delayMicroseconds(10);
   }

    //Serial.print(" dt:");
    //Serial.println(dt);
}


void motorMove(int lefts, int rights){

  if(rights < 0){
   //left side of H bridge
   digitalWrite(10, LOW);
   digitalWrite(12, HIGH);

  }
  if(rights > 0){
   //left side of H bridge
   digitalWrite(10, HIGH);
   digitalWrite(12, LOW);
  }

  if(lefts < 0){
   //right side of H bridge
   digitalWrite(8, LOW);
   digitalWrite(9, HIGH);
  }
  if(lefts > 0){
   //right side of H bridge
   digitalWrite(8, HIGH);
   digitalWrite(9, LOW);
  }
  if(rights < 0){
    OCR2A = -rights;
  }else{
    OCR2A = rights;
  }

  if(lefts < 0){
    OCR2B = -lefts;
  }else{
    OCR2B = lefts;
  }


  // OCR2B = rights;



 // left.write(90 + lefts);
 // right.write(90 - rights);
```

```
  return;

}


void storeValue(double newValue){ //shifts all the values down and the value sent goes into the 0'th element. lastelement deleted
  int arrayIndex = (sizeof ArrayTest/sizeof(int))/2; //find the number of entries in array
  for(int i = (arrayIndex-1); i >= 0 ; i--){ //-1 since for loop..
    ArrayTest[i] = ArrayTest[i-1];
  }
  ArrayTest[0] = newValue;
  return;
}
  double averageArray(){
  int arrayIndex = (sizeof ArrayTest/sizeof(int))/2; //find the number of entries in array
  double sum = 0;
  for(int i = 0; i <= (arrayIndex-1); i++){ //-1 since for loop..
    sum += ArrayTest[i];
  }
  return (sum/arrayIndex);
}




void pwmSetup(){//just run once at setup

   pinMode(3, OUTPUT); //OCR2B  3 and 11 are pwm channels. //3 is the left motor
   pinMode(11, OUTPUT); //OCR2A

   TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM20); //phase correct pwm 31250hz

   TCCR2B = _BV(CS20);//change this as datasheet says to get different pwm frequencies

   OCR2A = 0;
   OCR2B = 0;

   //left side of H bridge
   pinMode(8, OUTPUT); //input1 lm293D
   pinMode(9, OUTPUT); //input2 lm293D
   digitalWrite(8, LOW);
   digitalWrite(9, LOW);
   //right side of H bridge
   pinMode(10, OUTPUT); //input4 lm293D
   pinMode(12, OUTPUT);  //input3 lm293D
   digitalWrite(10, LOW);
   digitalWrite(12, LOW);

}



void nunchuck_init ()
{
  Wire.beginTransmission (0x52);// transmit to device 0x52
  Wire.send (0x40); // sends memory address
  Wire.send (0x00); // sends sent a zero.
  Wire.endTransmission (); // stop transmitting
}

void send_zero ()
```

```
{
  Wire.beginTransmission (0x52);// transmit to device 0x52
  Wire.send (0x00); // sends one byte
  Wire.endTransmission (); // stop transmitting
}

void receive_nunchuck_data(){
  Wire.requestFrom(0x52, 6);
  for (int i=0;i<6;i++){
    data[i]=Wire.receive();
  }
  make_nunchuck_data();
  send_zero();
}

void make_nunchuck_data ()
{
  for(int i=0;i<6;i++){
    outbuf[i]=nunchuck_decode_byte(data[i]);
  }
  joy_x_axis = outbuf[0];
  joy_y_axis = outbuf[1];
  accel_x_axis = outbuf[2] * 2 * 2;
  accel_y_axis = outbuf[3] * 2 * 2;
  accel_z_axis = outbuf[4] * 2 * 2;
  z_button = 0;
  c_button = 0;

// byte outbuf[5] contains bits for z and c buttons
// it also contains the least significant bits for the accelerometer data
// so we have to check each bit of byte outbuf[5]
  if ((outbuf[5] >> 0) & 1)
  {
    z_button = 1;
  }
  if ((outbuf[5] >> 1) & 1)
  {
    c_button = 1;
  }

  if ((outbuf[5] >> 2) & 1)
  {
    accel_x_axis += 2;
  }
  if ((outbuf[5] >> 3) & 1)
  {
    accel_x_axis += 1;
  }

  if ((outbuf[5] >> 4) & 1)
  {
    accel_y_axis += 2;
  }
  if ((outbuf[5] >> 5) & 1)
  {
    accel_y_axis += 1;
  }

  if ((outbuf[5] >> 6) & 1)
  {
    accel_z_axis += 2;
  }
```

```
  if ((outbuf[5] >> 7) & 1)
  {
    accel_z_axis += 1;
  }
}

char nunchuck_decode_byte (char x)
{
  x = (x ^ 0x17) + 0x17;
  return x;
}

void wmpOn(){
  Wire.beginTransmission(0x53);//WM+ starts out deactivated at address 0x53
  Wire.send(0xfe); //send 0x04 to address 0xFE to activate WM+
  Wire.send(0x04);
  Wire.endTransmission(); //WM+ jumps to address 0x52 and is now active
}

void wmpOff(){
  Wire.beginTransmission(82);
  Wire.send(0xf0);//address then
  Wire.send(0x55);//command
//Wire.send(0x00);
//Wire.send(0xfb);
  Wire.endTransmission();
}

void wmpSendZero(){
  Wire.beginTransmission(0x52);//now at address 0x52
  Wire.send(0x00); //send zero to signal we want info
  Wire.endTransmission();
}

void calibrateZeroes(){
  for (int i=0;i<10;i++){
    wmpSendZero();
    Wire.requestFrom(0x52,6);
    for (int i=0;i<6;i++){
      data[i]=Wire.receive();
    }
    yaw0+=(((data[3] >> 2) << 8)+data[0])/10;//average 10 readings
    pitch0+=(((data[4] >> 2) << 8)+data[1])/10;// for each zero
    roll0+=(((data[5] >> 2) << 8)+data[2])/10;
  }
  Serial.print("Yaw0:");
  Serial.print(yaw0);
  Serial.print(" Pitch0:");
  Serial.print(pitch0);
  Serial.print(" Roll0:");
  Serial.println(roll0);
}

void receiveData(){
  wmpSendZero(); //send zero before each request (same as nunchuck)
  Wire.requestFrom(0x52,6); //request the six bytes from the WM+
  for (int i=0;i<6;i++){
    data[i]=Wire.receive();
  }
  //see http://wiibrew.org/wiki/Wiimote/Extension_Controllers#Wii_Motion_Plus
  //for info on what each byte represents
  yaw=((data[3] >> 2) << 8)+data[0]-yaw0;
```

```
  pitch=((data[4] >> 2) << 8)+data[1]-pitch0;
  roll=((data[5] >> 2) << 8)+data[2]-roll0;
}

void receiveRaw(){
  wmpSendZero(); //send zero before each request (same as nunchuck)
  Wire.requestFrom(0x52,6);//request the six bytes from the WM+
  for (int i=0;i<6;i++){
    data[i]=Wire.receive();
  }
  yaw=((data[3] >> 2) << 8)+data[0];
  pitch=((data[4] >> 2) << 8)+data[1];
  roll=((data[5] >> 2) << 8)+data[2];
}

void switchtonunchuck(){
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
  digitalWrite(4, HIGH);
}

void switchtowmp(){
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
}
```