```
/*
Robotics 101.
 Two modes: automatic and manual.
 Automatic five phototransistors connected by a MUX
 Two switches: one to calibrate white, other for switching Auto Mode ON/OFF
 Nunchuck manual control available
 Bluetooth manual control and calibration available
 version 1.07
 Status: Works

 Name: Alberto Tam Yong
 Date: 04-17-2013
 */

#include <Wire.h>
#include <string.h>
#undef int
#include <stdio.h>

#define M1A 3
#define M1B 5
#define M2A 6
#define M2B 9
#define switch1 4
#define switch2 10
#define muxA 2
#define muxB 7
#define muxC 8
#define photoSensor A0
#define currentSensor A1

int sensor[5];
int sensorCalibration[5]={300,300,300,300,300};
int sensorCalibration1[5];
int sensorCalibration2[5];

int sensorAccuracy1 = 600;
int sensorAccuracy2 = 30;
int sensorAccuracyManual = 100;
int sharpTurn = 20;

int prevSwitch2 = 0;

int currentSensorVar;
float currentUnit = 66;
float analogAccuracy = 4.9;
int orientation1 = 0;
int prevOrientation1 = 0;
int orientation2 = 0;
int prevOrientation2 = 0;
int autoMode = 0;

//Manual mode
int speed0M = 50;
int speed1M = 120;
int speed2M = 180;
int speed3M = 255;

//Default Mode
int speed1 = 80;
int speed2 = 140;
int speed3 = 180;

//Wii Nunchuck
uint8_t outbuf[6];      // array to store arduino output
int cnt = 0;

int joyx = 0;
int joyy = 0;
int accelx = 0;
int accely = 0;
int accelz = 0;

int z_button = 1;
int c_button = 1;

int wii = 0;
int preWii = 0;
//--------------

/*Movement Operations*/
void orientationCheck() {
  if(orientation1 != prevOrientation1)
  {
    digitalWrite(M1A,LOW);
    digitalWrite(M1B,LOW);
    delay(20);
    prevOrientation1 = orientation1;
  }
  if(orientation2 != prevOrientation2)
  {
    digitalWrite(M2A,LOW);
    digitalWrite(M2B,LOW);
    delay(20);
    prevOrientation2 = orientation2;
  }
}

void Forward() {
  orientation1 = 0;
  orientation2 = 0;
  orientationCheck();
  Serial.println("Forward");
  analogWrite(M1A,0);
  analogWrite(M1B,speed3);
  analogWrite(M2A,0);
  analogWrite(M2B,speed3);
}

void Back() {
  orientation1 = 1;
  orientation2 = 1;
  orientationCheck();
  Serial.println("Back");
  analogWrite(M1A,speed3);
  analogWrite(M1B,0);
  analogWrite(M2A,speed3);
  analogWrite(M2B,0);
}

void Left() {
  orientation1 = 0;
  orientationCheck();
  //delay(100);
  Serial.println("Left");
  analogWrite(M2A,0);
  analogWrite(M2B,0);
  analogWrite(M1A,0);
  analogWrite(M1B,speed3);
}

void Right() {
  orientation2 = 0;
  orientationCheck();
  //delay(100);
  Serial.println("Right");
  analogWrite(M1A,0);
  analogWrite(M1B,0);
  analogWrite(M2A,0);
  analogWrite(M2B,speed3);
}

void LeftR() {
  orientation1 = 0;
  orientation2 = 1;
  orientationCheck();
  //delay(100);
  Serial.println("LeftR");
  analogWrite(M2A,speed3);
  analogWrite(M2B,0);
  analogWrite(M1A,0);
  analogWrite(M1B,speed3);
}

void RightR() {
  orientation1 = 1;
  orientation2 = 0;
  orientationCheck();
  //delay(100);
  Serial.println("RightR");
  analogWrite(M1A,speed3);
  analogWrite(M1B,0);
  analogWrite(M2A,0);
  analogWrite(M2B,speed3);
}

void LeftSoft() {
  orientation1 = 1;
  orientation2 = 1;
  orientationCheck();
  //delay(100);
  Serial.println("LeftSoft");
  analogWrite(M2A,0);
  analogWrite(M2B,speed1);
  analogWrite(M1A,0);
  analogWrite(M1B,speed3);
}

void RightSoft() {
  orientation1 = 1;
  orientation2 = 1;
  orientationCheck();
  //delay(100);
  Serial.println("RightSoft");
  analogWrite(M1A,0);
  analogWrite(M1B,speed1);
  analogWrite(M2A,0);
  analogWrite(M2B,speed3);
}

void LeftSoftBack() {
  orientation1 = 1;
  orientation2 = 1;
  orientationCheck();
  //delay(100);
  Serial.println("LeftSoft");
  analogWrite(M2B,0);
  analogWrite(M2A,speed1);
  analogWrite(M1B,0);
  analogWrite(M1A,speed3);
}

void RightSoftBack() {
  orientation1 = 1;
  orientation2 = 1;
  orientationCheck();
  //delay(100);
  Serial.println("RightSoft");
  analogWrite(M1B,0);
  analogWrite(M1A,speed1);
  analogWrite(M2B,0);
  analogWrite(M2A,speed3);
}

void Stop() {
  digitalWrite(M1A,LOW);
  digitalWrite(M1B,LOW);
  digitalWrite(M2A,LOW);
  digitalWrite(M2B,LOW);
}
/*----------------*/

/*Independent Motor Control*/
void iLeftF(int pwm) {
  orientation1 = 0;
  orientationCheck();
  //delay(100);
  analogWrite(M1A,0);
  analogWrite(M1B,pwm);
}

void iRightF(int pwm) {
  orientation2 = 0;
  orientationCheck();
  //delay(100);
  analogWrite(M2A,0);
  analogWrite(M2B,pwm);
}

void iLeftB(int pwm) {
  orientation1 = 1;
  orientationCheck();
  //delay(100);
  analogWrite(M1A,pwm);
  analogWrite(M1B,0);
}

void iRightB(int pwm) {
  orientation2 = 1;
  orientationCheck();
  //delay(100);
  analogWrite(M2A,pwm);
  analogWrite(M2B,0);
}
/*-----------------*/

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

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

char nunchuk_decode_byte (char x) {
  x = (x ^ 0x17) + 0x17;
  return x;
```

```
}

void wiiGetInfo () {
  joyx = outbuf[0];
  joyy = outbuf[1];
  accelx = outbuf[2] * 2 * 2;
  accely = outbuf[3] * 2 * 2;
  accelz = outbuf[4] * 2 * 2;

  z_button = 1;
  c_button = 1;

  if ((outbuf[5] >> 0) & 1)
    z_button = 0;
  if ((outbuf[5] >> 1) & 1)
    c_button = 0;

  if ((outbuf[5] >> 2) & 1)
    accelx += 2;
  if ((outbuf[5] >> 3) & 1)
    accelx += 1;

  if ((outbuf[5] >> 4) & 1)
    accely += 2;
  if ((outbuf[5] >> 5) & 1)
    accely += 1;

  if ((outbuf[5] >> 6) & 1)
    accelz += 2;
  if ((outbuf[5] >> 7) & 1)
    accelz += 1;
}

void wiiManualControl() {
  int pwmF = map(joyy,160,235,50,255);
  int pwmB = 255 - map(joyy,27,100,10,190);
  int pwmR = map(joyx,160,230,185,255);
  int pwmL = 255 - map(joyx,25,100,0,75);

  int pwmRr = map(joyx,160,230,100,255);
  int pwmLr = 255 - map(joyx,25,100,0,155);

  if(joyx >= 100 && joyx <= 160)
  {
    //Forward
    if(joyy >= 160)
    {
      iLeftF(pwmF);
      iRightF(pwmF);
    }
    else if(joyy <= 100)
    {
      iLeftB(pwmB);
      iRightB(pwmB);
    }
    else
      Stop();
  }
  else if(joyx<100)
  {
    if(joyy>160)
    {
      iLeftF(pwmL);
      iRightF(pwmF);
    }
    else if(joyy<100)
    {
      iLeftB(pwmL);
      iRightB(pwmB);
    }
    else if(joyy>=100&&joyy<=160)
    {
      delay(20);
      iRightB(pwmLr);
      iLeftF(pwmLr);
    }
    else
      Stop();
  }
  else if(joyx>160)
  {
    if(joyy>160)
    {
      iLeftF(pwmF);
      iRightF(pwmR);
    }
    else if(joyy<100)
    {
      iLeftB(pwmB);
      iRightB(pwmR);
    }
    else if(joyy>=100&&joyy<=160)
    {
      delay(20);
      iRightF(pwmRr);
      iLeftB(pwmRr);
    }
    else
      Stop();
  }
}

void wiiLoop() {
  Wire.requestFrom (0x52, 6);   // request data from nunchuck
  while (Wire.available ())
  {
    outbuf[cnt] = nunchuk_decode_byte (Wire.read ());   // receive byte as an integer
    cnt++;
  }

  if (cnt >= 5)
  {
    wiiGetInfo();

    if(z_button == 1)
    {
      wii = 1;
      preWii = 1;
    }
    else if(z_button == 0)
    {
      if(preWii == 1)
        Stop();
      preWii = 0;
      wii = 0;
    }
  }

  cnt = 0;
  send_zero (); // send the request for next bytes
  delay (30);
}

/*-----------------*/

void ManualControlBluetooth(char control) {
  int prevSpeed1 = speed1;
  int prevSpeed2 = speed2;
  int prevSpeed3 = speed3;

  speed1 = speed1M;
  speed2 = speed2M;
  speed3 = speed3M;

  if(control=='F')
    Forward();
  else if(control=='B')
    Back();
  else if(control == 'R')
    RightR();
  else if(control =='L')
    LeftR();
  else if(control =='G')
    LeftSoft();
  else if(control =='I')
    RightSoft();
  else if(control =='H')
    LeftSoftBack();
  else if(control =='J')
    RightSoftBack();
  else if(control =='n')
  {
    currentReading();
    Serial.println();
  }
  else if(control =='S')
    Stop();
  else
    Stop();

  speed1 = prevSpeed1;
  speed2 = prevSpeed2;
  speed3 = prevSpeed3;
}

void muxSelectOutput(int a,int b,int c) {
  digitalWrite(muxA,a);
  digitalWrite(muxB,b);
  digitalWrite(muxC,c);
}

void muxSelect(int num) {
  muxSelectOutput(bitRead(num,0),bitRead(num,1),bitRead(num,2));
}

void refreshSensors() {
  for(int i=0;i<5;i++)
  {
    muxSelect(i);
    sensor[i] = analogRead(photoSensor);
  }
}

void adjustSensorsBlack() {
  refreshSensors();
  delay(10);
  for(int i=0;i<5;i++)
  {
    sensorCalibration1[i]=(sensor[i]-sensorAccuracy1);
  }
}

void adjustSensorsWhite() {
  refreshSensors();
  delay(10);
  for(int i=0;i<5;i++)
  {
    sensorCalibration2[i]=(sensor[i]+sensorAccuracy2);
  }
}

void adjustSensorsAvg() {
  for(int i=0;i<5;i++)
  {
    sensorCalibration[i]=(sensorCalibration1[i]+sensorCalibration2[i])/2;
  }
}

void LineFollowingProtocol() {
  //--x--
  if(sensor[0]<sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]<sensorCalibration[3]
  {
    Forward();
  }
  else if(sensor[0]<sensorCalibration[0] && sensor[1]>sensorCalibration[1] && sensor[2]<sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    LeftSoft();
  }
  else if(sensor[0]<sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]<sensorCalibration[2] && sensor[3]>sensorCalibratio
  {
    RightSoft();
  }

  //x----
  else if(sensor[0]>sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]<sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    LeftR();
  }
  else if(sensor[0]<sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]<sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    RightR();
  }

  //xx--- steeper turns
  else if(sensor[0]>sensorCalibration[0] && sensor[1]>sensorCalibration[1] && sensor[2]<sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    LeftR();
    delay(3*sharpTurn);
  }
  else if(sensor[0]<sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]<sensorCalibration[2] && sensor[3]>sensorCalibratio
  {
    RightR();
    delay(3*sharpTurn);
  }

  //-xx-- steep turns
  else if(sensor[0]<sensorCalibration[0] && sensor[1]>sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    Left();
    delay(30);
  }
  else if(sensor[0]<sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]>sensorCalibratio
  {
    Right();
    delay(3*sharpTurn);
  }

  //xxx-- 90 degree turns
  else if(sensor[0]>sensorCalibration[0] && sensor[1]>sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    LeftR();
    delay(4*sharpTurn);
  }
  else if(sensor[0]<sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]>sensorCalibratio
  {
    RightR();
    delay(4*sharpTurn);
  }

  //x-x-- 90 degree turn v-paradox
  else if(sensor[0]>sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    LeftR();
    delay(5*sharpTurn);
  }
  else if(sensor[0]<sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    RightR();
    delay(5*sharpTurn);
  }

  //xxxx- Avoid extremes
  else if(sensor[0]>sensorCalibration[0] && sensor[1]>sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]>sensorCalibratio
  {
    LeftR();
    delay(8*sharpTurn);
  }
  else if(sensor[0]<sensorCalibration[0] && sensor[1]>sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]>sensorCalibratio
  {
    RightR();
    delay(8*sharpTurn);
  }

  //-----
  else if(sensor[0]<sensorCalibration[0] && sensor[1]<sensorCalibration[1] && sensor[2]<sensorCalibration[2] && sensor[3]<sensorCalibratio
  {
    Stop();
  }

  //xxxxx
  else if(sensor[0]>sensorCalibration[0] && sensor[1]>sensorCalibration[1] && sensor[2]>sensorCalibration[2] && sensor[3]>sensorCalibratio
  {
    Stop();
  }
}

void currentReading() {
  float currentCalc;
  currentSensorVar = analogRead(currentSensor);
  currentCalc = ((float(currentSensorVar-512)*analogAccuracy)/currentUnit);
  Serial.print(currentCalc,1);
  Serial.print("\t");
}

void controlOptions() {
  wiiLoop();
  if(autoMode == 1 && wii == 0)
  {
    refreshSensors();

    for(int i=0;i<5;i++)
    {
      Serial.print(sensor[i]);
      Serial.print("\t");
    }
    currentReading();
    Serial.println();

    LineFollowingProtocol();
    delay(5);
  }
  else if(autoMode == 0 && wii == 1)
  {
    wiiManualControl();
  }

  if(digitalRead(switch1)==LOW && digitalRead(switch2)==HIGH)
  {
    autoMode=0;
    refreshSensors();
    delay(10);
    for(int i=0;i<5;i++)
    {
      sensorCalibration[i]=(sensor[i] + sensorAccuracyManual);
      Serial.print(sensorCalibration[i]);
      Serial.print("\t");
    }
    Serial.println();
  }
  else if(digitalRead(switch1)==HIGH && digitalRead(switch2)==LOW)
  {
    autoMode = 1;
    prevSwitch2 = 1;
  }
  else if(digitalRead(switch2)==HIGH && prevSwitch2 == 1)
  {
    autoMode = 0;
    Stop();
    prevSwitch2 = 0;
  }

  if(Serial.available() > 0)
  {
    char command = Serial.read();
    if(command == 'X')
      autoMode = 1;
    else if(command == 'x')
    {
      autoMode = 0;
      Stop();
    }
    else if(command == 'V')
    {
      adjustSensorsBlack();
      for(int i=0;i<5;i++)
      {
        Serial.print(sensor[i]);
        Serial.print("\t");
      }
      Serial.println();
      for(int i=0;i<5;i++)
      {
        Serial.print(sensorCalibration[i]);
        Serial.print("\t");
      }
      Serial.println();
    }
    else if(command == 'v')
    {
      adjustSensorsWhite();
      adjustSensorsAvg();
      for(int i=0;i<5;i++)
      {
        Serial.print(sensor[i]);
        Serial.print("\t");
      }
      Serial.println();
      for(int i=0;i<5;i++)
      {
        Serial.print(sensorCalibration[i]);
        Serial.print("\t");
      }
      Serial.println();
    }
    else if(command == 't')
    {
      refreshSensors();

      for(int i=0;i<5;i++)
      {
        Serial.print(sensor[i]);
        Serial.print("\t");
      }
      currentReading();
      Serial.println();
    }
    else
    {
      autoMode = 0;
      wii = 0;
      ManualControlBluetooth(command);
    }
  }
}

void setup()
{
  Serial.begin(115200);
  Serial.println("Booting system...");

  pinMode(switch1,INPUT_PULLUP);
  pinMode(switch2,INPUT_PULLUP);

  /*WiiNunchuck*/
  pinMode(A2,OUTPUT);
  pinMode(A3,OUTPUT);

  digitalWrite(A2,LOW);
  digitalWrite(A3,HIGH);
  delay(50);

  Wire.begin ();                // join i2c bus with address 0x52
  nunchuck_init (); // send the initilization handshake
  /*--------*/
  pinMode(muxA, OUTPUT);
  pinMode(muxB, OUTPUT);
  pinMode(muxC, OUTPUT);

  digitalWrite(muxA,LOW);
  digitalWrite(muxB,LOW);
  digitalWrite(muxC,LOW);

  refreshSensors();

  Stop();

  delay(100);
  Serial.println("Ready");
}

void loop()
{
  controlOptions();
}
```