```
/*
This is the final copy of the code 5/31/2011 using the ping))) ultrasonic sensor

todo:
   -combine the left and right color detection method
   -read both analogs at the same time for color detection

   -Localize variables into their methods
   -use correct variable types for everything

   -make both doors use 1 method

   -clean it up if we go to canada. *possibly start from scratch and rewrite the program into a new structure since we know exactly what needs to happen
*/


//pid setup
#include <PID_Beta6.h>
//Define Variables we'll be connecting to
double Setpoint, Input, Output;

//Specify the links and initial tuning parameters
PID myPID(&Input, &Output, &Setpoint,130,90,0); //P I D constants
//end pid setup


//servo setup
#include <Servo.h>
Servo myservo;  // myservo is the ping sensor servo
Servo servoLeft;  //left wheel cont. rotation servo
Servo servoRight; //right wheel cont. rotation servo
Servo DoorServoRight; //right door
Servo DoorServoLeft; //left door


//stages and quick tuning paramaters
int stage = 1;
int stopAngle = 93;//this is the stopped angle when the wheels will not rotate (servoLeft and servoRight)
int normalAngle = 170; //this points the ping 90 degrees (myservo)
int forwardAngle = 65; //this points the ping straight ahead of the robot (myservo)
double dollTolerance = 6;//how off can readings be before they're considered incorrect. Dont let this get too small...or the pid wont correct it. keep it 5 - 10
double DollRodDist = 48.0; //pid setpoint distance maintained from wall. Used for stage 1 and 3.
double sideDist = 22; //pid setpoint distance maintained from wall stage 2
int stage3ForwardtillTurn = 33; //start of stage 3 when it points the servo forward and drives up to the wall
int Stage3TurnTime = 1600; //this is how long it turns before backing up into wall

//ping stuff
double PingCurrent = 0; //this is the sharp running average value that's calculated once per loop
double PingUpTimeCheck = 0; //this keeps the micro from flooding the ping with requests...
int pingDelay = 45; //this is how often the ping will take measurements (it works at 50)


//servo doors
double rightDoorInitialTime = 0;
int rightDoorStage = 0;
double LeftDoorInitialTime = 0;
int LeftDoorStage = 0;


void setup()
{
  Serial.begin(115200);
```

```
 //wheels and sensors
  myservo.attach(2); //this is sharp sensor // attaches the servo on pin 2 to the servo object
  servoLeft.attach(4);
  servoRight.attach(3);
  //doors
   DoorServoRight.attach(5);  // attaches the servo on pin to the servo object
   DoorServoRight.write(150);
   DoorServoLeft.attach(6);
   DoorServoLeft.write(35);


  pidSetup();

 //reflection sensors
  pinMode(8, INPUT);//for the line detector
  pinMode(7, INPUT);//right door sensor
  pinMode(9, INPUT);//left door sensor

  //color LEDs
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(13, OUTPUT);


  //this will be replaced for a endless wait for a button to be pressed to start the robot
  delay(1000);
}



void loop(){

 /*
    myservo.write(normalAngle);
    while(1){

    PingCurrent = DistanceMeasurePing(12);
    Serial.print("ping:");
    Serial.println(PingCurrent);
    delay(100);
    }

*/

    if(stage == 1){
     stage1();
    }

    if(stage == 2){
     stage2();
    }

    if(stage == 3){
     stage3();
    }


}



void stage1(){
```

```
  //point the ping)) to the side and give servo time to respond
  myservo.write(normalAngle);
  delay(300);


  while(1){ //breaks when line is detected
    //move the doors if needed
    doorCallLeft();
    doorCallRight();

    //check the sides for a doll rod and set the flag variable if needed
    //number being sent is the analog pin that the photo resistor is on
    rightColordetect(0);
    leftColordetect(1);

    //update the motors and try to maintain distance. 1 means it's doing error checking on the ping readings
    PidUpdate(1, DollRodDist );


   //look to see if it ran into the tape at the end of stage1. If it did, then set program to go into stage2, stop motors, and exit from the while(1)
   if(digitalRead(8) == HIGH){
    Serial.println("line detected");
    stage = 2;

     servoLeft.write( stopAngle );
     servoRight.write( stopAngle );

    return;
  }


 }//while end

}//method end



void stage2(){ //this distance should be 28.64cm i think

   double Stage2InitialTime = millis();


   //these shouldn't be needed
   servoLeft.write( stopAngle );
   servoRight.write( stopAngle );

   //go ahead and close both doors since they won't be collecting anything.
   DoorServoRight.write(23); //storing  make sure to change this when you get final servos!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   DoorServoLeft.write(153);

  //go ahead and turn for approximately the correct amount.
  servoLeft.write(stopAngle + 15);
  servoRight.write(stopAngle + 2);
  delay(2000);

  //stop and make sure servo is pointing normal
  servoLeft.write( stopAngle );
  servoRight.write( stopAngle );
  myservo.write(normalAngle);
  delay(500); //give servo some time to point forward
```

```
  while(1){ //breaks when line is detected

   //update the motors and try to maintain distance. 0 means it's not doing error checking on the ping readings since there are no doll rods to interfere
   PidUpdate(0, sideDist);

   //if it sees the tape and 5 seconds have elapsed from stage2 start uptime. 5 seconds because sometimes when it turned it would see the same tape at the end
of stage 1.
   if(digitalRead(8) == HIGH & (Stage2InitialTime + 5000) < millis()){
    Serial.println("line detected");
    stage = 3;
    servoLeft.write( stopAngle );
    servoRight.write( stopAngle );

    return;
  }


 }//while end


}//stage2 end


void stage3(){

     //wheels should be stopped already... point ping))) forward and give some time for it to point forward
     servoLeft.write( stopAngle );
     servoRight.write( stopAngle );
     myservo.write(forwardAngle);
     delay(800); //servo delay to point forward

     //go ahead and update the measurement. I don't think this is needed
     PingCurrent = DistanceMeasurePing(12);

     while(DistanceMeasurePing(12) > stage3ForwardtillTurn){//stage3ForwardtillTurn is how close it will get to the wall before turning and reversing
     //now try to make it go somewhat straight
     servoRight.write( stopAngle - 10);
     servoLeft.write( stopAngle + 10);
     }

     servoLeft.write( stopAngle );
     servoRight.write( stopAngle );


    //turn the car to face the final stretch
    servoLeft.write( stopAngle + 3 );
    servoRight.write( stopAngle + 10 );
    delay(Stage3TurnTime); //how long it turns

    //these aren't really needed
    servoRight.write( stopAngle );
    servoLeft.write( stopAngle );

    //point ping))) normal to the robot (towards wall)
    myservo.write(normalAngle);//this one doesnt need a delay


    //now it should be at the final stretch


    //backup to the wall
```

```
    servoRight.write( stopAngle + 10 );
    servoLeft.write( stopAngle - 10);
    delay(3000);
    servoRight.write( stopAngle );
    servoLeft.write( stopAngle );



  while(1){ //breaks when line is detected


    //update the motors and try to maintain distance. 1 means it's doing error checking on the ping readings
    PidUpdate(1, DollRodDist);

    //move the doors if needed
    doorCallLeft();
    doorCallRight();

    //check the sides for a doll rod and set the flag variable if needed
    //number being sent is the analog pin that the photo resistor is on
    rightColordetect(0);
    leftColordetect(1);


   //A2 is the switch on the front. When the final wall is hit at the end it folds up the robot.
   if(analogRead(A2) > 600){
     //close door
    DoorServoLeft.write(153);
    DoorServoRight.write(23);
    //reverse
    servoRight.write( stopAngle + 10 );
    servoLeft.write( stopAngle - 10);
    delay(200);
    myservo.write(0); //fold in ping
    delay(300);
    //go forward
    servoRight.write( stopAngle - 10 );
    servoLeft.write( stopAngle + 10);
    delay(1500);
    servoRight.write(stopAngle);
    servoLeft.write(stopAngle);

    while(1){} //endless loop at the end to keep the robot from doing anything

   }



  }//while end


}//stage3 end



void pidSetup(){
   //pid
     myPID.SetOutputLimits(-10,10);
     PingCurrent = DistanceMeasurePing(12); //initialize the pid
     Input = PingCurrent;
```

```
    //turn the PID on
    myPID.SetMode(AUTO);
    myPID.SetSampleTime(50); //in ms, so 100ms is .1sec. note that if you change this, pid constants will be different
    delay(100);

}


//DistanceMeasurePing(12)
double DistanceMeasurePing(int pingPin){

  //if not enough time has passed for ping sensor, then just return whatever value was last read
  if((PingUpTimeCheck + pingDelay) > millis()){

  return PingCurrent;
  }
  PingUpTimeCheck = millis();



  // establish variables for duration of the ping,
  // and the distance result in inches and centimeters:
  double duration, cm;

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  // The same pin is used to read the signal from the PING))): a HIGH
  // pulse whose duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return duration / 29. / 2.;
}



void doorCallRight(){//when the color sensor detects a red peg, it will set rightDoorStage to 1. a value of 0 makes the doorOpenRight() do nothing


/*
right
wide open: 150
knocking: 55
collecting:43
storing:23
DoorServoRight.write(23);
*/

  int openDoor = 150;
  int knockingOver = 55;
```

```
  int collecting = 43;
  int storing = 15;

  int timeCollect = 1400; //this is the time the door waits to go to the collecting phase
  int delayToStore = 600; // how long it waits after the light sensor is triggered
  int TimeToKeepDoorClosed = 600;


  if(rightDoorStage == 0){
    DoorServoRight.write(openDoor);
    return;
  }

  //At the end of the door function it will reset initialTime to zero to signal when the door is in the completely open high angle state
  if(rightDoorStage == 1){
   rightDoorInitialTime = millis();
   DoorServoRight.write(knockingOver);
   rightDoorStage = 2;
  }


  if(rightDoorStage == 2 && (rightDoorInitialTime + timeCollect) < millis() ){
    DoorServoRight.write(collecting);
    rightDoorStage = 3;
  }

  if(rightDoorStage == 3 && digitalRead(7) == LOW){
   rightDoorInitialTime = millis();
   rightDoorStage = 4;
  }


  if(rightDoorStage == 4 && (rightDoorInitialTime + delayToStore) < millis() ){

     DoorServoRight.write(storing);
     rightDoorInitialTime = millis();
     rightDoorStage = 5;
     //delay(1000);
    }

  if(rightDoorStage == 5 && (rightDoorInitialTime + TimeToKeepDoorClosed) < millis() ){
    DoorServoRight.write(openDoor);
    rightDoorStage = 0;
    rightDoorInitialTime = 0;
  }


 return;
}


void doorCallLeft(){//when the color sensor detects a red peg, it will set rightDoorStage to 1. a value of 0 makes the doorOpenRight() do nothing

/*
double LeftDoorInitialTime = 0;
int LeftDoorStage = 0;

left
wide open: 35
knocking: 115
collecting:128
storing:153
```

```
*/

  int openDoor = 35;
  int knockingOver = 115;
  int collecting = 128;
  int storing = 165;

  int timeCollect = 1400; //this is the time the door waits to go to the collecting phase
  int delayToStore = 600; // how long it waits after the light sensor is triggered
  int TimeToKeepDoorClosed = 600;


  if(LeftDoorStage == 0){
    DoorServoLeft.write(openDoor);
    return;
  }

  //At the end of the door function it will reset initialTime to zero to signal when the door is in the completely open high angle state
  if(LeftDoorStage == 1){
   LeftDoorInitialTime = millis();
   DoorServoLeft.write(knockingOver);
   LeftDoorStage = 2;
  }


  if(LeftDoorStage == 2 && (LeftDoorInitialTime + timeCollect) < millis() ){
    DoorServoLeft.write(collecting);
    LeftDoorStage = 3;
  }

  if(LeftDoorStage == 3 && digitalRead(9) == LOW){
   LeftDoorInitialTime = millis();
   LeftDoorStage = 4;
  }


  if(LeftDoorStage == 4 && (LeftDoorInitialTime + delayToStore) < millis() ){

     DoorServoLeft.write(storing);
     LeftDoorInitialTime = millis();
     LeftDoorStage = 5;
     //delay(1000);
    }

  if(LeftDoorStage == 5 && (LeftDoorInitialTime + TimeToKeepDoorClosed) < millis() ){
    DoorServoLeft.write(openDoor);
    LeftDoorStage = 0;
    LeftDoorInitialTime = 0;
  }


 return;
}



void PidUpdate(boolean ToleranceEnable, double NewSetpoint){

    Setpoint = NewSetpoint;
    PingCurrent = DistanceMeasurePing(12);
```

```
    if(ToleranceEnable == 1){
    //if it sees a distance that's not correct, ignore it and use one that's approx correct
      if(PingCurrent < DollRodDist - dollTolerance || PingCurrent > DollRodDist + dollTolerance){
        PingCurrent = DollRodDist;
      }
    }
    Input = PingCurrent;
    myPID.Compute();


   int speedm = 10;
   int newOut = Output;
   int leftSpeed = (stopAngle + speedm + newOut);
   int rightSpeed = stopAngle -(speedm - newOut);
   servoLeft.write( leftSpeed );
   servoRight.write( rightSpeed );


}


void rightColordetect(int AnalogPin){
  //delayTime*4 is about the total time this function takes. (will be greatly shorter when Serial commands removed).
  int delayTime = 4; //at 2ms, it only works about an inch away. 5ms was somewhere between 1-2 inches.
  int ambient, red, blue, green;
  double ambientr, redr, bluer, greenr;
  //if none of the ratios are past this, it will say it's looking at nothing.
  double toleranceRBG = 1.08;//1.05;  //(lower will need less LED light, but less reliable)

  if(rightDoorStage != 0){
    return;
  }


  //ambient
  //Serial.print("ambient ");
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(13, HIGH);
//  delay(200);//remove this
  delay(delayTime);
  ambient = analogRead(AnalogPin);

  //red
  //Serial.print("red ");
  digitalWrite(10, LOW);
  digitalWrite(11, HIGH);
  digitalWrite(13, HIGH);
  delay(delayTime);
  red = analogRead(AnalogPin);



  //green
  //Serial.print("green ");
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(13, LOW);
  delay(delayTime);
  green = analogRead(AnalogPin);
```

```
  //blue
  // Serial.print("blue ");
  digitalWrite(10, HIGH);
  digitalWrite(11, LOW);
  digitalWrite(13, HIGH);
  delay(delayTime);
  blue = analogRead(AnalogPin);


  ambientr = (double)ambient/(ambient);
  greenr = (double)green/(ambient);
  bluer = (double)blue/(ambient);
  redr = (double)red/(ambient);

  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(13, HIGH);


  Serial.print("ambient:");
  Serial.print(ambientr,3);

  Serial.print("  red:");
  Serial.print(redr,3);

  Serial.print("  blue:");
  Serial.print(bluer,3);

  Serial.print("  green:");
  Serial.println(greenr,3);

  if(redr < toleranceRBG && greenr < toleranceRBG && bluer < toleranceRBG){
    Serial.print("\ndo nothing\n");
    return;
  }


  if(redr > bluer && redr > greenr){
    Serial.print("\nRED!!!!!!!!!!!!!!!!!!!!!\n");// <-(flag variable gets set to 1 here)
    rightDoorStage = 1;
    return;
  }


  Serial.print("\ndo nothing\n");
  return;
}//end color detection


void leftColordetect(int AnalogPin){
  //delayTime*4 is about the total time this function takes. (will be greatly shorter when Serial commands removed).
  int delayTime = 4; //at 2ms, it only works about an inch away. 5ms was somewhere between 1-2 inches.
  int ambient, red, blue, green;
  double ambientr, redr, bluer, greenr;
  //if none of the ratios are past this, it will say it's looking at nothing.
  double toleranceRBG = 1.08;//1.05;  //(lower will need less LED light, but less reliable)

  if(LeftDoorStage != 0){
    return;
  }
```

```
  //ambient
  //Serial.print("ambient ");
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(13, HIGH);
//  delay(200);//remove this
  delay(delayTime);
  ambient = analogRead(AnalogPin);

  //red
  //Serial.print("red ");
  digitalWrite(10, LOW);
  digitalWrite(11, HIGH);
  digitalWrite(13, HIGH);
  delay(delayTime);
  red = analogRead(AnalogPin);



  //green
  //Serial.print("green ");
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(13, LOW);
  delay(delayTime);
  green = analogRead(AnalogPin);


  //blue
 // Serial.print("blue ");
  digitalWrite(10, HIGH);
  digitalWrite(11, LOW);
  digitalWrite(13, HIGH);
  delay(delayTime);
  blue = analogRead(AnalogPin);


  ambientr = (double)ambient/(ambient);
  greenr = (double)green/(ambient);
  bluer = (double)blue/(ambient);
  redr = (double)red/(ambient);

  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(13, HIGH);


  Serial.print("ambient:");
  Serial.print(ambientr,3);

  Serial.print("  red:");
  Serial.print(redr,3);

  Serial.print("  blue:");
  Serial.print(bluer,3);

  Serial.print("  green:");
  Serial.println(greenr,3);

  if(redr < toleranceRBG && greenr < toleranceRBG && bluer < toleranceRBG){
    Serial.print("\ndo nothing\n");
```

```
    return;
  }


  if(redr > bluer && redr > greenr){
    Serial.print("\nRED!!!!!!!!!!!!!!!!!!!!!!\n");// <-(flag variable gets set to 1 here)
    LeftDoorStage = 1;
    return;
  }


  Serial.print("\ndo nothing\n");
  return;
}//end color detection
```