```
/*
 * Project 2 Challenge 2
 * GE 1501 Section 22 (9:15)
 * Nick DePatie
 * Table Cleaning Robot
 * This Robot will cover a whole
table by detecting the edges using
an ultrasonic distance sensor facing
downwards
 *
 * Pseudocode:
 *  loop:
 *  define variables and name pins
 *  setup pin modes
 *  record distance gotten from
distance sensor
 *  if the switch is turned off, the
robot can proceed with the program
 *  if the button is pressed, the
robot will turn on and enter mode1
 *
 *  mode1:
 *  if the table is not detected,
then
 *    the robot will backup and turn
90 degrees
 *    the distance will be recorded
 *       if the table is not
detected, then end the program
 *       if the table is detected,
turn another 90 degrees then restart
mode1
 *   if the table is detected, then
the robot will go forward
 *
 *   movement functions are used in
order to make coding the main
function easier
 *     i.e. backward: both motors
rotate backwards
 *
 *   distance function is used to
```

```
retrieve dsitance from dsitance
sensor:
 *      send a pulse
 *      receive pulse
 *      record this value as distance
 */



//the right motor will be controlled
by the motor A pins on the motor
driver

const int AIN1 = 13;
//control pin 1 on the motor driver
for the right motor
const int AIN2 = 12;
//control pin 2 on the motor driver
for the right motor
const int PWMA = 11;
//speed control pin on the motor
driver for the right motor
```

```
//the left motor will be controlled
by the motor B pins on the motor
driver

const int PWMB = 10;
//speed control pin on the motor
driver for the left motor
const int BIN2 = 9;
//control pin 2 on the motor driver
for the left motor
const int BIN1 = 8;
//control pin 1 on the motor driver
for the left motor

//button inputs
const int MODE1=4;

//distance variables
const int trigPin = 6;
const int echoPin = 5;
const int switchPin = 7;
//switch to turn the robot on and off
const int intpin = 3;

float distance = 0;
//variable to store the distance
measured by the distance sensor

//robot behaviour variables
int turnTime = 450;
//amount that the robot will turn
(90 degrees)

int z=0;
/*z will be used to determine which
way the robot should turn in order
to have a zig-zag movement:

   if z=0, then it will turn right,
and if z=1, then the robot will turn
```

```
left*/
/************************************
*************************************
*******/
void setup()
{
  pinMode(trigPin,
OUTPUT);                 //this pin
will send ultrasonic pulses out from
the distance sensor
  pinMode(echoPin,
INPUT);                  //this pin
will sense when the pulses reflect
back to the distance sensor

  pinMode(switchPin,
INPUT_PULLUP);         //set this as
a pullup to sense whether the switch
is flipped

  //these three pins are the button
```

```
pins that use a pullup resistor to
detect the press
  pinMode(MODE1,
INPUT_PULLUP);

  //set the motor control pins as
outputs
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);

  pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
  pinMode(PWMB, OUTPUT);

  Serial.
begin(9600);
//begin serial communication with
the computer
  Serial.print("To infinity and
beyond!");  //test the serial
```

```
connection
}

/************************************
*************************************
*******/


void loop()
{
  //DETECT THE DISTANCE READ BY THE
DISTANCE SENSOR
  distance = getDistance();


  z=0;     //by default, the robot
will turn right on the first pass



 //Printing Distance Value and
```

```
Potentiometer value
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println("
in");                  // print the
units
  Serial.println(digitalRead(MODE1));

  if (digitalRead(switchPin) == LOW)
{    //if the on switch is flipped

    if (digitalRead(MODE1) == LOW)
{      // looks for button press oto
activate mode
      Serial.print("ACTIVATED");
      mode1
();                           //see
"void mode 1()" for contents of mode
    }
  }
```

```
}

/************************************
*************************************
******************/
/************************************
*************************************
******************/

//THIS IS THE ACTUAL FUNCTION FOR
THE MODE

void mode1 ()
{
  //actual script for the actions of
mode 1
 distance=getDistance();

  if (distance > 6)
{
//if table is not detected backup
```

```
    stopMove();
    delay (400);
    backward (400);
    delay (400);
      if (z == 0)
{
//if the robot is suppsoed to turn
right, the robot will turn right 90
degrees then test the distance
        rightTurn ();
        distance=getDistance();
        delay (400);
          if (distance < 6)
{                         //if the
table is detected, the robot will
turn right another 90 degrees then
return to the beginning of the mode
            forward
();
//moves robot a bit forward unless
the edge of the table is detected
```

```
            rightTurn();

z++;
     //this turns the robot into the
left turning mode
            mode1();
          }
          else
{
  //if the table is not detected,
then the robot will stop and wait
for another button press (***the
task has been finished and the table
is clean***)
            return loop ();
          }
      } else if (z==1)
{                             //if
the robot is suppsoed to turn left,
the robot will turn left 90 degrees
then test the distance
```

```
        leftTurn ();
        distance=getDistance();
        delay (400);
          if (distance < 6)
{                        //if the
table is detected, the robot will
turn left another 90 degrees then
return to the beginning of the mode
            forward
();
//moves the robot a bit forward
unless the edge of the table is
detected
            leftTurn();

z--;
    //this turns the robot into the
right turning mode
            mode1();
          }
          else
```

```
{
 //if the table is no detected, then
the robot will stop and wait for
another button press (***the task
has been finished and the table is
clean***)
            return loop ();
      }
      }
    delay (50);
  }
      else if (distance < 6)
{                       // if table
is detected, keep going forward
        rightMotor(50);
        leftMotor(50);
      }

      delay
(50);
    // delay at the end to avoid
```

```
bugs and record distance
      return mode1 ();
}




/************************************
*************************************
****************/
/************************************
*************************************
****************/


void rightMotor(int
motorSpeed)
//function for driving the right
motor
{
  if (motorSpeed >
0)
```

```
//if the motor should drive forward
(positive speed)
  {
    digitalWrite(AIN1,
HIGH);                         //set
pin 1 to high
    digitalWrite(AIN2,
LOW);                          //set
pin 2 to low
  }
  else if (motorSpeed <
0)                            //if
the motor should drive backward
(negative speed)
  {
    digitalWrite(AIN1,
LOW);                          //set
pin 1 to low
    digitalWrite(AIN2,
HIGH);                         //set
pin 2 to high
```

```
  }

else
               //if the motor should
stop
  {
    digitalWrite(AIN1,
LOW);                          //set
pin 1 to low
    digitalWrite(AIN2,
LOW);                          //set
pin 2 to low
  }
  analogWrite(PWMA,
abs(motorSpeed));
//now that the motor direction is
set, drive it at the entered speed
}
```

```
/************************************
*************************************
*******/
void leftMotor(int
motorSpeed)
//function for driving the left motor
{
  if (motorSpeed >
0)
//if the motor should drive forward
(positive speed)
  {
    digitalWrite(BIN1,
HIGH);                         //set
pin 1 to high
    digitalWrite(BIN2,
LOW);                          //set
pin 2 to low
  }
  else if (motorSpeed <
0)                            //if
```

```
the motor should drive backward
(negative speed)
  {
    digitalWrite(BIN1,
LOW);                          //set
pin 1 to low
    digitalWrite(BIN2,
HIGH);                         //set
pin 2 to high
  }

else
               //if the motor should
stop
  {
    digitalWrite(BIN1,
LOW);                          //set
pin 1 to low
    digitalWrite(BIN2,
LOW);                          //set
pin 2 to low
```

```
  }
  analogWrite(PWMB,
abs(motorSpeed));
//now that the motor direction is
set, drive it at the entered speed
}




/************************************
*************************************
*******/
//RETURNS THE DISTANCE MEASURED BY
THE HC-SR04 DISTANCE SENSOR

float getDistance()
{

  // **this function takes the
average of three distance readings
```

```
to average out and hopefully
eliminate anomalies in distance
sensing**

  float echoTime1;
//variable to store the time it
takes for a ping to bounce off an
object
  float calculatedDistance1;
//variable to store the distance
calculated from the echo time
  float echoTime2;
//variable to store the time it
takes for a ping to bounce off an
object
  float calculatedDistance2;
  float echoTime3;
  float
calculatedDistance3;

  float calculatedDistance;
  //send out an ultrasonic pulse
that's 10ms long
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  echoTime1 = pulseIn(echoPin,
HIGH);      //use the pulsein
command to see how long it takes for
the

     //pulse to bounce back to the
sensor

  calculatedDistance1 = echoTime1 /
148.0;  //calculate the distance of
the object that reflected the pulse
(half the bounce time multiplied by
the speed of sound)
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  echoTime2 = pulseIn(echoPin,
HIGH);




  calculatedDistance2 = echoTime2 /
148.0;
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  echoTime3 = pulseIn(echoPin,
HIGH);




  calculatedDistance3 = echoTime3 /
148.0;
```

```
  calculatedDistance=
(calculatedDistance1+calculatedDistan
ce2+calculatedDistance3)/3;
//averaging out all 4 calculated
distances
  return
calculatedDistance;
//send back the distance that was
calculated
}




/************************************
*************************************
*******/
//MOVEMENT FUNCTIONS USED TO MAKE
```

```
MOVEMENT EASIER TO FOLLOW

void stopMove () {      //stops
robot (this program was used after
every movement statement to preserve
motors and add delay between
movements)
  rightMotor(0);
  leftMotor(0);
  delay(100);
}

void forward () {       //moves
robot forward for 750 milliseconds
unless the table is not detected
  for (int x=0; x<=15; x++) {
    distance=getDistance();
      if (distance > 6) {
        return loop ();
      }
      else {
        rightMotor(50);
        leftMotor(50);
      }
      delay (50);
  }
  stopMove ();
}

void rightTurn () {    //turns the
robot right 90 degrees
  rightMotor(-75);
  leftMotor(75);
  delay(turnTime);
  stopMove ();
}

void leftTurn () {     //turns the
robot left 90 degrees
  rightMotor(75);
  leftMotor(-75);
  delay(turnTime);
  stopMove ();
}

void backward (int backuptime) {
//backs the robot up for however
long is entered
  rightMotor(-75);
  leftMotor(-75);
  delay (backuptime);
  stopMove ();
}
```