Alex Crotts **ECGR 4161/5196 Lab #7 Report** **Due 04/04/22**
acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

Lab Objective:

The objective of the lab was for the robot to sweep its entire surrounding, identify the closest object near it, and drive as close as possible to the identified object. Once the robot has been powered, the servo would need to move 180 degrees while the robot itself would remain stationary. After the servo shifts 180 degrees, the robot would turn 180 degrees in place and the servo would repeat the sweep. The robot would then turn towards the closest object and drive until the robot has touched the object.

Commentary and Conclusion:

The robot initially had problems rotating to the correct position after performing the sweep. It seemed to rotate to random positions that did not match the direction of the closest object. The problem was resolved by fixing an error in the Rotate function in which the angle was not converted into encoder pulses for the microcontroller to read. Also, the direction of rotation needed to be reversed in order to reach the correct final position. Once these were corrected, the robot successfully identified and rotated towards the closest object. To stop at the closest object, the bumb switches were used. Initially, the introduction of the bump switch code stopped the robot from even turning towards the correct object. After moving the bump switch code into the Drive_Straight function and placing the disableMotor function at the end of the void loop, the robot performed the sweep correctly and stopped when the bump switch was pressed.

Lab Code:

```
/*
Use an ultrasonic sensor to find objects around the robot
Alex Crotts and Luis Umana - 4/5/2022
*/

#include <Servo.h>
#include <SimpleRSLK.h>

Servo servo;

const int trigPin = 32;//This is Port Pin 3.5 on the MSP432 Launchpad
const int echoPin = 33; //This is Port Pin 5.1 on the MSP432 Launchpad

int MotorSpeed = 10;
float WheelDiameter = 6.985;     // In centimeters
float PulsePerRev = 360;          // Number of encoder pulses the
microcontroller reads per 1 wheel rotation
float WheelBase = 13.335;       // In centimeters

// Number of encoder pulses per 1 degree of rotation
double PulsePerDegree = WheelBase/WheelDiameter;

void setup() {
  // Initialization
```

```
25    pinMode(trigPin, OUTPUT);   // Set trigPin as an output
26    pinMode(echoPin, INPUT);    // Set echoPin as an input
27    servo.attach(38);
28    servo.write(0);
29    setupRSLK();
30    resetLeftEncoderCnt();      // Reset encoder counts
31    resetRightEncoderCnt();
32    Serial.begin(9600);
33    Serial.println("Beginning Scan");
34    delay(1000);      // Delay to allow the serial monitor to settle
35  }
36
37  void Drive_Straight(int y) {
38    // Integer y allows for this function to be called for any distance
39    // Function for driving straight for X centimeters
40    resetLeftEncoderCnt();
41    resetRightEncoderCnt();
42    enableMotor(BOTH_MOTORS);
43    // Set both motors to drive forward
44    setMotorDirection(BOTH_MOTORS, MOTOR_DIR_FORWARD);
45    setMotorSpeed(BOTH_MOTORS, MotorSpeed);  // Set both motors to the same speed
46    int L_Pulse_Count = 0;      // Zero the left encoder pulse count
47    int R_Pulse_Count = 0;      // Zero the right encoder pulse count
48
49    while((L_Pulse_Count < y) || (R_Pulse_Count < y)) {
50      // Run this loop until the number of pulses reaches the specified value
51      L_Pulse_Count = getEncoderLeftCnt();      // Read the left encoder value
52      R_Pulse_Count = getEncoderRightCnt();     // Read the right encoder value
53
54      // Drive the robot forward until it runs into something
55      if(digitalRead(BP_SW_PIN_0) == 0)
56        break;
57
58      if(digitalRead(BP_SW_PIN_1) == 0)
59        break;
60
61      if(digitalRead(BP_SW_PIN_2) == 0)
62        break;
63
64      if(digitalRead(BP_SW_PIN_3) == 0)
65        break;
66
67      if(digitalRead(BP_SW_PIN_4) == 0)
68        break;
69
70      if(digitalRead(BP_SW_PIN_5) == 0)
71        break;
72
73      if((L_Pulse_Count + 1 < R_Pulse_Count)){
74        // If the left is slower than the right, speed up left and slow down right
75        setMotorSpeed(LEFT_MOTOR, ++MotorSpeed);      // Speed up the left motor
```

Alex Crotts **ECGR 4161/5196 Lab #7 Report** **Due 04/04/22**
acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

```
76        setMotorSpeed(RIGHT_MOTOR, --MotorSpeed);     // Slow down the right motor
77      }
78
79      if((R_Pulse_Count + 1 < L_Pulse_Count)){
80        // If the right is slower than the left, speed up right and slow down left
81        setMotorSpeed(RIGHT_MOTOR, ++MotorSpeed);     // Speed up the right motor
82        setMotorSpeed(LEFT_MOTOR, --MotorSpeed);      // Slow down the left motor
83      }
84
85      if(L_Pulse_Count >= y){
86        // If the number of pulses reaches aspecified value, turn off motors
87        disableMotor(LEFT_MOTOR);     // Turn off the left motor
88        disableMotor(RIGHT_MOTOR);    // Turn off the right motor
89        }
90
91        // Print encoder counts to the serial monitor for debugging
92        Serial.print("Driving Straight Now");
93        Serial.print("\t");
94        Serial.print("Left Encoder: ");
95        Serial.print(L_Pulse_Count);
96        Serial.print("\t");
97        Serial.print("Right Encoder: ");
98        Serial.println(R_Pulse_Count);
99        delay(100);
100     }
101   }
102
103   void Rotate(int z, int L_Motor_Dir, int R_Motor_Dir) {
104     // Integers allow for any rotation direction and degree
105     // Function for rotating the RSLK robot in place
106     resetLeftEncoderCnt();
107     resetRightEncoderCnt();
108     enableMotor(BOTH_MOTORS);
109     // Set the left and right motors to drive in the specified directions
110     setMotorDirection(LEFT_MOTOR, L_Motor_Dir);
111     setMotorDirection(RIGHT_MOTOR, R_Motor_Dir);
112     setMotorSpeed(BOTH_MOTORS, MotorSpeed);    // Set the motors to the same speed
113     int L_CCW_Pulse_Count = 0;      // Zero the encoder count
114     int R_CCW_Pulse_Count = 0;      // Zero the encoder count
115
116       while(R_CCW_Pulse_Count < z) {
117       // Run this loop until the number of pulses reaches the specified value
118       L_CCW_Pulse_Count = getEncoderLeftCnt();      // Read left encoder value
119       R_CCW_Pulse_Count = getEncoderRightCnt();     // Read right encoder value
120
121       if(R_CCW_Pulse_Count >= z) {
122         // If the number of pulses reaches the specified value, turn off motors
123         disableMotor(LEFT_MOTOR);       // Turn off the left motor
124         disableMotor(RIGHT_MOTOR);      // Turn off the right motor
125         delay(1000);
126       }
```

acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

```
127
128       //Print encoder counts to the serial monitor for debugging
129       Serial.print("Turning CCW Now");
130       Serial.print("\t");
131       Serial.print("Left Encoder CCW Turn: ");
132       Serial.print(L_CCW_Pulse_Count);
133       Serial.print("\t");
134       Serial.print("Right Encoder CCW Turn: ");
135       Serial.println(R_CCW_Pulse_Count);
136       delay(100);
137     }
138   }
139
140   long Read_Distance() {
141     // This function reads the distance from the ultrasonic sensor
142     byte Readings[7];   // Declare an array of readings
143     int x = 0;          // Array indexed at zero
144     long pulseLength;   // Length of the ultrasonic pulse
145     long centimeters;   // Calculated distance
146     long total = 0;     // Initially zero the total for averaging the array
147     long average;       // Calculated average of the array
148
149     // Sending the pulse to the ultrasonic sensor
150     digitalWrite(trigPin, LOW);
151     delayMicroseconds(10);
152     digitalWrite(trigPin, HIGH);
153     delayMicroseconds(10);
154     digitalWrite(trigPin, LOW);
155     delayMicroseconds(10);
156
157     // Calculating the distance from the pulse length
158     pulseLength = pulseIn(echoPin, HIGH);
159     centimeters = pulseLength / 58;
160
161     // Set up the loop to store values in an array
162     for(Readings[x]; x < 7; x++) {
163       // Read from the sensor:
164       Readings[x] = centimeters;
165       // Add the reading to the total:
166       total = total + Readings[x];
167
168       // If we're at the end of the array...
169       if (x >= 7) {
170         // ...wrap around to the beginning:
171         x = 0;
172         }
173     }
174
175     // Calculate the average of the array:
176     average = total / 7;
177     // send it to the computer as ASCII digits
```

acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

```
  Serial.print("Average Distance: ");
  Serial.println(average);
  delay(100);        // delay in between reads for stability
  return(average);
    }

void loop() {
  int pos = 0;    // variable to store the servo position
  byte Sweep_1_Array[18];   // Declare the first array
  digitalWrite(77, HIGH);
  long Min1_value = Read_Distance();  // Store a reading as the min value
  int Min1_position = pos;    // Store the angle at which the min was found

  for(pos = 0; pos < 180; pos += 10) { // Moves the servo from 0 to 180 degrees
    servo.write(pos);     // tell servo to go to position in variable 'pos'
    delay(15);     // waits 15ms for the servo to reach the position
    Sweep_1_Array[pos/10] = Read_Distance();  // Read distance at this position
    delay(500);

    if(Sweep_1_Array[pos/10] < Min1_value) {
      // If the new distance is less than the min value, store the new distance
      Min1_value = Sweep_1_Array[pos/10];
      Min1_position = pos;    // Store the position the new min was found
    }
  }

  // Print the minimum distance and angle from this array
  Serial.print("Min1 distance: ");
  Serial.println(Min1_value);
  Serial.print("Min1 position: ");
  Serial.println(Min1_position);
  delay(1000);

  // Rotate the robot 180 degrees
  Rotate(180*PulsePerDegree, MOTOR_DIR_FORWARD, MOTOR_DIR_BACKWARD);

  byte Sweep_2_Array[18];   // Declare the second array
  digitalWrite(77, HIGH);
  long Min2_value = Read_Distance();  // Store a reading as the min value
  int Min2_position = pos;    // Store the angle at which the min was found

  for(pos = 180; pos>=1; pos -= 10)  {   // goes from 180 to 0 degrees
    servo.write(pos);    // tell servo to go to position in variable 'pos'
    delay(15);     // waits 15ms for the servo to reach the position
    Sweep_2_Array[pos/10] = Read_Distance();  // Read distance at this position
    delay(500);

    if(Sweep_2_Array[pos/10] < Min2_value) {
      // If the new distance is less than the min value, store the new distance
      Min2_value = Sweep_2_Array[pos/10];
      Min2_position = pos;    // Store the position the new min was found
```

acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

```
    }
  }

  // Print the minimum distance and angle from this array
  Serial.print("Min2 distance: ");
  Serial.println(Min1_value);
  Serial.print("Min2 position: ");
  Serial.println(Min1_position);
  delay(1000);

  // Compare the minimum distances from the two arrays
  if(Min1_value <= Min2_value){ // If first array contained the smaller distance
    // Turn 180 degrees minus the angle of the servo at that position
    Rotate(180*PulsePerDegree - Min1_position*PulsePerDegree, MOTOR_DIR_FORWARD,
    MOTOR_DIR_BACKWARD);
    Serial.print("Min 1 value is less than Min 2 value");
  }

  else { //If the second array contained the smallest distance
   // Turn the robot the same number of degrees as that servo position
    Rotate(Min2_position*PulsePerDegree, MOTOR_DIR_BACKWARD, MOTOR_DIR_FORWARD);
  }

  servo.write(0);   // Bring the servo back to forward
  delay(1000);

  // Drive straight 80 cm or until a bumb sensor is pressed
  Drive_Straight(80/((WheelDiameter * PI)/(PulsePerRev)));

  // Disable the motors once the bump switch is pressed
  Serial.println("Collision detected");
  disableMotor(BOTH_MOTORS);
}
```