Alex Crotts **ECGR 4161/5196 Lab #8 Report** **Due 04/18/22**
acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

Lab Objective:

This lab was divided into two parts: one “learning process” and one demonstration. For part one, the objective of the lab was for the robot to traverse a 3 meter course while identifying three obstacles(i.e., 30x30 cm boxes) along its path. The robot must stop and record data every 20 cm using a 180 servo motor and one ultrasonic. Once it had reached the end of the 3 meter path, it would print the arrays of distance data to the serial monitor. For part two of the lab, once the robot had traveled 3 meters, it would need to turn 180 degrees and travel back to the starting point without stopping unless an obstacle is nearby. While traveling back to the starting point, the robot must identify the object by having the servo motor point the ultrasonic sensor towards the object and stare at it for three seconds. The servo would then need to point the ultrasonic sensor forward until it has reached its next identified obstacle or has gone back to the starting point.

Lab part 1:

The left and right side arrays that were recorded by the sensor and interpreted by the microcontroller are shown below. In this case, a value of 1 means that an obstacle was detected while a value of 0 means no obstacle was present at that position.

```
Left Array:
[ 0 ]
[ 1 ]
[ 1 ]
[ 1 ]
[ 1 ]
[ 0 ]
[ 1 ]
[ 1 ]
[ 0 ]
[ 0 ]
[ 0 ]
[ 0 ]
[ 0 ]
[ 0 ]
[ 1 ]
[ 0 ]

Right Array:
[ 0 ]
[ 0 ]
[ 0 ]
[ 0 ]
[ 0 ]
[ 1 ]
[ 0 ]
[ 0 ]
[ 1 ]
[ 1 ]
[ 0 ]
[ 0 ]
[ 0 ]
[ 0 ]
[ 1 ]
[ 1 ]
```

Figure 1. Serial monitor output of the left and right distance arrays

Alex Crotts **ECGR 4161/5196 Lab #8 Report** **Due 04/18/22**
acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

Table 1. Left and right arrays recorded by the robot

Distance (in cm)	Left Array	Right Array
0	0	0
20	1	0
40	1	0
60	1	0
80	1	0
100	0	1
120	1	0
140	1	0
160	0	1
180	0	1
200	0	0
220	0	0
240	0	0
260	0	0
280	1	1
300	0	1

Commentary and Conclusion:

The lab was semi successful however it did not achieve two lab requirements due to time constraints. The robot was able to travel straight, identify the boxes, and turn in place but we did have an issue where our robot overshot the 3 meters as well as not making a complete stop when returning to the

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

starting point. The issue was found to be that the encoder count reset after each "drive straight" function was performed and so the microcontroller did not know when the robot had traveled 3 meters.

Lab Code:

Part 1 Code:

```
/*
Drive down a hallway and map objects along the wall
Alex Crotts and Luis Umana - 4/18/2022
*/

#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 pulses per 1 wheel rotation
float WheelBase = 13.335;       // In centimeters

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

void setup() {
  // Initialization
  pinMode(trigPin, OUTPUT);    // Set trigPin as an output
  pinMode(echoPin, INPUT);     // Set echoPin as an input
  servo.attach(38);
  servo.write(0);
  setupRSLK();
  resetLeftEncoderCnt();       // Reset encoder counts
  resetRightEncoderCnt();
  Serial.begin(9600);
  Serial.println("Beginning Scan");
  delay(1000);       // Delay to allow the serial monitor to settle
}

void Drive_Straight(int y) {     // This function can be called for any distance
  // Function for driving straight for X centimeters
  resetLeftEncoderCnt();
  resetRightEncoderCnt();
  enableMotor(BOTH_MOTORS);
  setMotorDirection(BOTH_MOTORS, MOTOR_DIR_FORWARD);     // Set both motors to forward
  setMotorSpeed(BOTH_MOTORS, MotorSpeed);         // Set both motors to the same speed
  int L_Pulse_Count = 0;       // Zero the left encoder pulse count
  int R_Pulse_Count = 0;       // Zero the right encoder pulse count
```

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

```
44
45    while((L_Pulse_Count < y) || (R_Pulse_Count < y)) {
46      // Run until the number of pulses reaches the value stated in the void loop
47      L_Pulse_Count = getEncoderLeftCnt();      // Read the left encoder value
48      R_Pulse_Count = getEncoderRightCnt();     // Read the right encoder value
49
50      // Drive the robot forward until it runs into something
51      if(digitalRead(BP_SW_PIN_0) == 0)
52        break;
53
54      if(digitalRead(BP_SW_PIN_1) == 0)
55        break;
56
57      if(digitalRead(BP_SW_PIN_2) == 0)
58        break;
59
60      if(digitalRead(BP_SW_PIN_3) == 0)
61        break;
62
63      if(digitalRead(BP_SW_PIN_4) == 0)
64        break;
65
66      if(digitalRead(BP_SW_PIN_5) == 0)
67        break;
68
69      if((L_Pulse_Count + 1 < R_Pulse_Count)){
70        // If left motor is slower than right, speed up left motor and slow down right
71        setMotorSpeed(LEFT_MOTOR, ++MotorSpeed);      // Speed up the left motor by 1
72        setMotorSpeed(RIGHT_MOTOR, --MotorSpeed);     // Slow down the right motor by 1
73      }
74
75      if((R_Pulse_Count + 1 < L_Pulse_Count)){
76        // If right motor is slower than left, speed up right motor and slow down left
77        setMotorSpeed(RIGHT_MOTOR, ++MotorSpeed);     // Speed up the right motor by 1
78        setMotorSpeed(LEFT_MOTOR, --MotorSpeed);      // Slow down the left motor by 1
79      }
80
81      if(L_Pulse_Count >= y){
82        // If the number of pulses reaches the specified value, turn off the motors
83        disableMotor(LEFT_MOTOR);     // Turn off the left motor
84        disableMotor(RIGHT_MOTOR);    // Turn off the right motor
85        }
86    }
87  }
88
89  void Rotate(int z, int L_Motor_Dir, int R_Motor_Dir) {
90    // The function can be called for any rotation direction and degree
91    // Function for rotating the RSLK robot in place
92    resetLeftEncoderCnt();
93    resetRightEncoderCnt();
94    enableMotor(BOTH_MOTORS);
```

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

```
95    // Set the left motor to drive in the specified direction
96    setMotorDirection(LEFT_MOTOR, L_Motor_Dir);
97    // Set the right motor to drive in the specified direction
98    setMotorDirection(RIGHT_MOTOR, R_Motor_Dir);
99    setMotorSpeed(BOTH_MOTORS, MotorSpeed);         // Set the motor to the same speed
100   int L_CCW_Pulse_Count = 0;      // Zero the encoder count
101   int R_CCW_Pulse_Count = 0;      // Zero the encoder count
102
103     while(R_CCW_Pulse_Count < z) {
104     // Run until the number of pulses reaches the specified value
105     L_CCW_Pulse_Count = getEncoderLeftCnt();      // Read the left encoder value
106     R_CCW_Pulse_Count = getEncoderRightCnt();     // Read the right encoder value
107
108     if(R_CCW_Pulse_Count >= z) {
109       // If the number of pulses reaches the specified value, turn off the motors
110       disableMotor(LEFT_MOTOR);       // Turn off the left motor
111       disableMotor(RIGHT_MOTOR);      // Turn off the right motor
112       delay(1000);
113     }
114   }
115 }
116
117 long Read_Distance() {
118   // This function reads the distance from the ultrasonic sensor
119   byte Readings[7];   // Declare an array of readings
120   int x = 0;          // Array indexed at zero
121   long pulseLength;   // Length of the ultrasonic pulse
122   long centimeters;   // Calculated distance
123   long total = 0;     // Initially zero the total for averaging the array
124   long average;       // Calculated average of the array
125
126   // Sending the pulse to the ultrasonic sensor
127   digitalWrite(trigPin, LOW);
128   delayMicroseconds(10);
129   digitalWrite(trigPin, HIGH);
130   delayMicroseconds(10);
131   digitalWrite(trigPin, LOW);
132   delayMicroseconds(10);
133
134   // Calculating the distance from the pulse length
135   pulseLength = pulseIn(echoPin, HIGH);
136   centimeters = pulseLength / 58;
137
138   // Set up the loop to store values in an array
139   for(Readings[x]; x < 7; x++) {
140     // Read from the sensor:
141     Readings[x] = centimeters;
142     // Add the reading to the total:
143     total = total + Readings[x];
144
145     // If we're at the end of the array...
```

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

```
146     if (x >= 7) {
147       // ...wrap around to the beginning:
148       x = 0;
149       }
150   }
151
152   // Calculate the average of the array:
153   average = total / 7;
154   // send it to the computer as ASCII digits
155   Serial.print("Average Distance: ");
156   Serial.println(average);
157   delay(100);        // delay in between reads for stability
158   return(average);
159     }
160
161 void loop() {
162   int RightArray[16];  //Store placement of boxes on the right side
163   int LeftArray[16];   //Store placement of boxes on the left side
164   long RightDistance;  //Distance on the right side
165   long LeftDistance;   //Distance on the left side
166
167   for (int i = 0 ; i < 16; i++){
168     servo.write(180);  //Turn servo to the left
169     delay(800);
170     LeftDistance = Read_Distance();  //Read distance
171
172     if (LeftDistance < 110) {  //If there is a box, store a 1
173       LeftArray[i] = 1;
174     }
175     else {  // If there is no box, store a 0
176       LeftArray[i] = 0;
177     }
178
179     servo.write(0);  //Turn servo to the right
180     delay(800);
181     RightDistance = Read_Distance();  //Read distance
182
183     if (RightDistance < 110) {  //If there is a box, store a 1
184       RightArray[i] = 1;
185     }
186     else {  //If there is no box, store a 0
187       RightArray[i] = 0;
188     }
189     delay(1000);
190     //Drive forward 20cm and begin the scan again
191     Drive_Straight(20*PulsePerRev/(WheelDiameter*PI));
192   }
193
194   //Print the left side array
195   Serial.println("Left Array:");
196   for (int j = 0; j < 16; j++) {
```

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

```
    Serial.print("[ ");
    Serial.print(LeftArray[j]);
    Serial.println(" ]");
  }

  //Print the right side array
  Serial.println();
  Serial.println("Right Array:");
  for (int k = 0; k < 16; k++) {
    Serial.print("[ ");
    Serial.print(RightArray[k]);
    Serial.println(" ]");
  }

    Serial.println();
    delay(10000);  //Wait 10 seconds before repeating
}
```

Part 2 Code:

```
/*
Drive down a hallway and identify where objects are along the wall
Alex Crotts and Luis Umana - 4/18/2022
*/

#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 per 1 wheel rotation
float WheelBase = 13.335;       // In centimeters

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

void setup() {
  // Initialization
  pinMode(trigPin, OUTPUT);   // Set trigPin as an output
  pinMode(echoPin, INPUT);    // Set echoPin as an input
  servo.attach(38);
  servo.write(0);
  setupRSLK();
  resetLeftEncoderCnt();      // Reset encoder counts
  resetRightEncoderCnt();
```

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

```
30    Serial.begin(9600);
31    Serial.println("Beginning Scan");
32    delay(1000);       // Delay to allow the serial monitor to settle
33  }
34
35  void Drive_Straight(int y) {    // This function can be called for any distance
36    // Function for driving straight for X centimeters
37    resetLeftEncoderCnt();
38    resetRightEncoderCnt();
39    enableMotor(BOTH_MOTORS);
40    setMotorDirection(BOTH_MOTORS, MOTOR_DIR_FORWARD);  // Set both motors to forward
41    setMotorSpeed(BOTH_MOTORS, MotorSpeed);       // Set both motors to the same speed
42    int L_Pulse_Count = 0;       // Zero the left encoder pulse count
43    int R_Pulse_Count = 0;       // Zero the right encoder pulse count
44
45    while((L_Pulse_Count < y) || (R_Pulse_Count < y)) {
46      // Run until the number of pulses reaches the value stated in the void loop
47      L_Pulse_Count = getEncoderLeftCnt();       // Read the left encoder value
48      R_Pulse_Count = getEncoderRightCnt();      // Read the right encoder value
49
50      // Drive the robot forward until it runs into something
51      if(digitalRead(BP_SW_PIN_0) == 0)
52        break;
53
54      if(digitalRead(BP_SW_PIN_1) == 0)
55        break;
56
57      if(digitalRead(BP_SW_PIN_2) == 0)
58        break;
59
60      if(digitalRead(BP_SW_PIN_3) == 0)
61        break;
62
63      if(digitalRead(BP_SW_PIN_4) == 0)
64        break;
65
66      if(digitalRead(BP_SW_PIN_5) == 0)
67        break;
68
69      if((L_Pulse_Count < R_Pulse_Count)){
70        // If left motor is slower than right, speed up left motor and slow down right
71        setMotorSpeed(LEFT_MOTOR, ++MotorSpeed);     // Speed up the left motor by 1
72        setMotorSpeed(RIGHT_MOTOR, --MotorSpeed);    // Slow down the right motor by 1
73      }
74
75      if((R_Pulse_Count < L_Pulse_Count)){
76        // If right motor is slower than left, speed up right motor and slow down left
77        setMotorSpeed(RIGHT_MOTOR, ++MotorSpeed);     // Speed up the right motor by 1
78        setMotorSpeed(LEFT_MOTOR, --MotorSpeed);      // Slow down the left motor by 1
79      }
80
```

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

```
81      if(L_Pulse_Count >= y){
82        // If the number of pulses reaches the specified value, turn off the motors
83        disableMotor(LEFT_MOTOR);     // Turn off the left motor
84        disableMotor(RIGHT_MOTOR);    // Turn off the right motor
85        }
86    }
87    resetLeftEncoderCnt();
88    resetRightEncoderCnt();
89  }
90
91  void Rotate(int z, int L_Motor_Dir, int R_Motor_Dir) {
92    // The function can be called for any rotation direction and degree
93    // Function for rotating the RSLK robot in place
94    resetLeftEncoderCnt();
95    resetRightEncoderCnt();
96    enableMotor(BOTH_MOTORS);
97    // Set the left motor to drive in the specified direction
98    setMotorDirection(LEFT_MOTOR, L_Motor_Dir);
99    // Set the right motor to drive in the specified direction
100   setMotorDirection(RIGHT_MOTOR, R_Motor_Dir);
101   setMotorSpeed(BOTH_MOTORS, MotorSpeed/1.2);    // Set the motors to the same speed
102   int L_CCW_Pulse_Count = 0;      // Zero the encoder count
103   int R_CCW_Pulse_Count = 0;      // Zero the encoder count
104
105     while(R_CCW_Pulse_Count < z) {
106     // Run until the number of pulses read reaches the specified value
107     L_CCW_Pulse_Count = getEncoderLeftCnt();      // Read the left encoder value
108     R_CCW_Pulse_Count = getEncoderRightCnt();     // Read the right encoder value
109
110     if(R_CCW_Pulse_Count >= z) {
111       // If the number of pulses reaches the specified value, turn off the motors
112       disableMotor(LEFT_MOTOR);       // Turn off the left motor
113       disableMotor(RIGHT_MOTOR);      // Turn off the right motor
114       delay(1000);
115     }
116   }
117 }
118
119 long Read_Distance() {
120   // This function reads the distance from the ultrasonic sensor
121   byte Readings[7];   // Declare an array of readings
122   int x = 0;          // Array indexed at zero
123   long pulseLength;   // Length of the ultrasonic pulse
124   long centimeters;   // Calculated distance
125   long total = 0;     // Initially zero the total for averaging the array
126   long average;       // Calculated average of the array
127
128   // Sending the pulse to the ultrasonic sensor
129   digitalWrite(trigPin, LOW);
130   delayMicroseconds(10);
131   digitalWrite(trigPin, HIGH);
```

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

```
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(10);

  // Calculating the distance from the pulse length
  pulseLength = pulseIn(echoPin, HIGH);
  centimeters = pulseLength / 58;

  // Set up the loop to store values in an array
  for(Readings[x]; x < 7; x++) {
    // Read from the sensor:
    Readings[x] = centimeters;
    // Add the reading to the total:
    total = total + Readings[x];

    // If we're at the end of the array...
    if (x >= 7) {
      // ...wrap around to the beginning:
      x = 0;
      }
  }

  // Calculate the average of the array:
  average = total / 7;
  // send it to the computer as ASCII digits
  Serial.print("Average Distance: ");
  Serial.println(average);
  delay(100);        // delay in between reads for stability
  return(average);
    }

void loop() {
  long RightArray[16];  //Store placement of boxes on the right side
  long LeftArray[16];   //Store placement of boxes on the left side
  long EncoderArray[16];  //Store encoder values at each stop
  long RightDistance; //Right side distances
  long LeftDistance;  //Left side distances
  int TotalPulses = 300*PulsePerDegree/(WheelDiameter*PI);  //Pulses for driving 3m
  int L_EncoderCnt;   //Left encoder value
  int R_EncoderCnt;   //Right encoder value

  for (int i = 0 ; i < 16; i++){
    EncoderArray[i] = 20*i*PulsePerRev/(WheelDiameter*PI);  //Store encoder value
    servo.write(180);   //Turn servo to the left
    LeftDistance = Read_Distance();   //Read distance
    delay(500);

    if (LeftDistance < 110) {
      LeftArray[i] = 1;   //If there is a box, store a 1
    }
    else {
```

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

```
183       LeftArray[i] = 0;   //If there is no box, store a 0
184     }
185
186     servo.write(0);   //Turn servo to the right
187     RightDistance = Read_Distance();  //Read distance
188     delay(500);
189
190     if (RightDistance < 110) {
191       RightArray[i] = 1;    //If there is a box, store a 1
192     }
193     else {
194       RightArray[i] = 0;    //If there is no box, store a 0
195     }
196
197     delay(500);
198     //Drive forward 20 cm and begin the scan again
199     Drive_Straight(20*PulsePerRev/(WheelDiameter*PI));
200   }
201
202   delay(500);
203   servo.write(90);
204   //After scanning the hallway, turn 185 degrees for compensation
205   Rotate(185*PulsePerDegree, MOTOR_DIR_FORWARD, MOTOR_DIR_BACKWARD);
206   delay(500);
207
208   for (int j = 16; j > 0; j--) {
209     //Check the left array for values of 1
210     if (LeftArray[j] == 1) {   //If a value of 1 is found, drive forward
211       //Drive until the pulses in the encoder array are reached
212       Drive_Straight(EncoderArray[16-j]);
213       L_EncoderCnt = getEncoderLeftCnt();   //Record encoder count
214       servo.write(180);   //Turn servo to face the object
215       disableMotor(BOTH_MOTORS);  //Stop moving
216       delay(3000);    //Wait 3 seconds
217       servo.write(90);    //Turn servo back to middle
218     }
219
220     //Check the right array for values of 1
221     if (RightArray[j] == 1) {  //If a value of 1 is found, drive forward
222       //Drive until the pulses in the encoder array are reached
223       Drive_Straight(EncoderArray[16-j]);
224       L_EncoderCnt = getEncoderLeftCnt();   //Record encoder count
225       servo.write(0);   //Turn servo to face the object
226       disableMotor(BOTH_MOTORS);  //Stop moving
227       delay(3000);    //Wait 3 seconds
228       servo.write(90);    //Turn servo back to middle
229     }
230
231     if(LeftArray[j] == 0 && RightArray[j] == 0){
232       //If no box is detected, don't move
233       servo.write(90);  //Keep servo in the middle
```

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

```
234       L_EncoderCnt = getEncoderLeftCnt();   //Record encoder count
235       disableMotor(BOTH_MOTORS);    //Don't move
236     }
237
238     if(L_EncoderCnt = TotalPulses){
239       //If the encoder count reaches 3 meters, stop moving
240       disableMotor(BOTH_MOTORS);
241     }
242   }
243     //Print the left side array
244     Serial.println("Left Array:");
245     for (int k = 0; k < 16; k++) {
246       Serial.print("[ ");
247       Serial.print(LeftArray[k]);
248       Serial.println(" ]");
249     }
250
251     //Print the right side array
252     Serial.println();
253     Serial.println("Right Array:");
254     for (int m = 0; m < 16; m++) {
255       Serial.print("[ ");
256       Serial.print(RightArray[m]);
257       Serial.println(" ]");
258     }
259
260       Serial.println();
261       delay(10000);   //Wait 10 seconds before repeating
262   }
```