Alex Crotts **ECGR 4161/5196 Lab #5 Report** **Due 02/28/22**
acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

Lab Objective

The objective of this lab was to create a program where the TI-RSLK would drive any given radius and drive in a complete 360-degree circle before returning to the starting point.

Commentary and Conclusion

One issue we encountered was that our bot would have tight turns, creating an oval shape during our initial test run. This was caused by accidentally giving more power to the left motor instead of the right motor whenever the right motor needed to be adjusted. "If" statements were added to the code to adjust the speed of the motors to maintain a calculated ratio between the left and right wheel encoder pulses. Our final results had a distance of 4.5cm away from the circle start to the stop point, and 4.5cm away from the complete demo start to the stop point.

Lab Code

```
1   /*
2    This code makes the RSLK robot drive straight, rotate 90 degrees, and drive in a circle of X cm radius
3    Luis Umana and Alex Crotts, 2022-02-23
4   */
5
6   // Include the RSLK library to control the motors and read the encoders
7   #include "SimpleRSLK.h"
8
9   // Define the parameters of the robot and the "driving course"
10  int MotorSpeed = 12;          // Slow motor speed for stability
11  float WheelDiameter = 6.985;    // In centimeters
12  float StraightDistance = 75;     // In centimeters
13  float PulsePerRev = 360;        // Number of encoder pulses the microcontroller reads per 1 wheel rotation
14  float WheelBase = 13.335;      // In centimeters
15  float TurnCCWDeg = 90;          // Degree of rotation before starting the circle
16
17  // Calculate the number of encoder pulses needed to complete each phase of the driving course
18  // Number of pulses needed to drive X centimeters straight
19  double Straight_pulses = StraightDistance/((WheelDiameter * PI)/(PulsePerRev));
20
21  // Number of pulses needed to rotate 90 degrees in place
22  double TurnCCW_pulses = (TurnCCWDeg/360)*(WheelBase * PI)/(WheelDiameter * PI) * (PulsePerRev);
23
24  // Number of pulses needed for the inner wheel to drive in an X cm radius circle
25  double Circle_Inner_Wheel_Pulses = 2*(StraightDistance - 0.5*WheelBase) * PI/(WheelDiameter * PI) * PulsePerRev;
26  // Number of pulses needed for the outer wheel to drive in an X cm radius circle
27  double Circle_Outer_Wheel_Pulses = 2*(StraightDistance + 0.5*WheelBase) * PI/(WheelDiameter * PI) * PulsePerRev;
28  // Ratio of outer wheel encoder pulses to inner wheel encoder pulses for driving in a circle
29  double SpeedRatio = Circle_Outer_Wheel_Pulses/Circle_Inner_Wheel_Pulses;
30  // Make the left motor run at a proportionally lower speed than the right motor when driving in a circle
31  double LeftMotorSpeed = MotorSpeed/SpeedRatio;
32
33
```

Alex Crotts **ECGR 4161/5196 Lab #5 Report** **Due 02/28/22**
acrotts2@uncc.edu
Luis Umana
lumana@uncc.edu

```
34  void setup() {
35    // Setup the RSLK to perform necessary functions and initialize the encoders
36    setupRSLK();
37    resetLeftEncoderCnt();     // Initialize the left encoder
38    resetRightEncoderCnt();    // Initialize the right encoder
39    Serial.begin(9600);        // Begin the serial monitor
40  }
41
42  void Drive_Straight() {
43    // Function for driving straight for X centimeters
44    resetLeftEncoderCnt();
45    resetRightEncoderCnt();
46    enableMotor(BOTH_MOTORS);
47    setMotorDirection(BOTH_MOTORS, MOTOR_DIR_FORWARD);     // Set both motors to drive forward
48    setMotorSpeed(BOTH_MOTORS, MotorSpeed);                // Set both motors to the same speed
49    int L_Pulse_Count = 0;     // Zero the left encoder pulse count
50    int R_Pulse_Count = 0;     // Zero the right encoder pulse count
51
52    while((L_Pulse_Count < Straight_pulses) || (R_Pulse_Count < Straight_pulses)) {
53      // Run this loop until the number of pulses read by the microcontroller reaches the calculated pulses above
54      L_Pulse_Count = getEncoderLeftCnt();     // Read the left encoder value
55      R_Pulse_Count = getEncoderRightCnt();    // Read the right encoder value
56
57      if((L_Pulse_Count + 1 < R_Pulse_Count)){
58        // If the left motor is driving slower than the right, speed up the left motor and slow down the right
59        setMotorSpeed(LEFT_MOTOR, ++MotorSpeed);     // Speed up the left motor in increments of 1
60        setMotorSpeed(RIGHT_MOTOR, --MotorSpeed);    // Slow down the right motor in increments of 1
61      }
62
63      if((R_Pulse_Count + 1 < L_Pulse_Count)){
64        // If the right motor is driving slower than the left, speed up the right motor and slow down the left
65        setMotorSpeed(RIGHT_MOTOR, ++MotorSpeed);    // Speed up the right motor in incremements of 1
66        setMotorSpeed(LEFT_MOTOR, --MotorSpeed);     // Slow down the left motor in increments of 1
67      }
68
69      if(L_Pulse_Count >= Straight_pulses){
70        // If the number of pulses reaches the calculated value, turn off the motors and stop running the function
71        disableMotor(LEFT_MOTOR);    // Turn off the left motor
72        disableMotor(RIGHT_MOTOR);   // Turn off the right motor
73        }
74
75        // Print encoder counts to the serial monitor for debugging
76        Serial.print("Driving Straight Now");
77        Serial.print("\t");
78        Serial.print("Left Encoder: ");
79        Serial.print(L_Pulse_Count);
80        Serial.print("\t");
81        Serial.print("Right Encoder: ");
82        Serial.println(R_Pulse_Count);
```

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

```
83      delay(100);
84    }
85  }
86
87  void Rotate_CCW() {
88    // Function for rotating 90 degrees CCW in place
89    resetLeftEncoderCnt();
90    resetRightEncoderCnt();
91    enableMotor(BOTH_MOTORS);
92    setMotorDirection(LEFT_MOTOR, MOTOR_DIR_BACKWARD);     // Set the left motor to drive backward
93    setMotorDirection(RIGHT_MOTOR, MOTOR_DIR_FORWARD);     // Set the right motor to drive forward
94    setMotorSpeed(BOTH_MOTORS, MotorSpeed);              // Set the motor to the same speed
95    int L_CCW_Pulse_Count = 0;     // Zero the encoder count
96    int R_CCW_Pulse_Count = 0;     // Zero the encoder count
97
98    while((L_CCW_Pulse_Count < TurnCCW_pulses - 10) || (R_CCW_Pulse_Count < TurnCCW_pulses - 10)) {
99      // Run this loop until the number of pulses read by the microcontroller reaches the calculated value above
100     L_CCW_Pulse_Count = getEncoderLeftCnt();     // Read the left encoder value
101     R_CCW_Pulse_Count = getEncoderRightCnt();    // Read the right encoder value
102
103     if(L_CCW_Pulse_Count >= TurnCCW_pulses -10 || R_CCW_Pulse_Count >= TurnCCW_pulses - 10) {
104       // If the number of pulses reaches the calculated value, turn off the motors and stop running the function
105       disableMotor(LEFT_MOTOR);      // Turn off the left motor
106       disableMotor(RIGHT_MOTOR);     // Turn off the right motor
107       delay(1000);
108     }
109
110     // Print encoder counts to the serial monitor for debugging
111     Serial.print("Turning CCW Now");
112     Serial.print("\t");
113     Serial.print("Left Encoder CCW Turn: ");
114     Serial.print(L_CCW_Pulse_Count);
115     Serial.print("\t");
116     Serial.print("Right Encoder CCW Turn: ");
117     Serial.println(R_CCW_Pulse_Count);
118     delay(100);
119   }
120 }
121
122 void Drive_Circle() {
123   // Function for driving in a circle of radius X centimeters
124   resetLeftEncoderCnt();
125   resetRightEncoderCnt();
126   enableMotor(BOTH_MOTORS);
127   setMotorDirection(BOTH_MOTORS, MOTOR_DIR_FORWARD);     // Set both motors to drive forward
128   setMotorSpeed(LEFT_MOTOR, LeftMotorSpeed);        // Set the left motor to the MotorSpeed value defined above
129   setMotorSpeed(RIGHT_MOTOR, MotorSpeed);   // Right motor drives at the proportionally higher speed calculated above
130   int L_Circle_Pulse_Count = 0;     // Zero the encoder count
131   int R_Circle_Pulse_Count = 0;     // Zero the encoder count
```

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

```
132
133   while((R_Circle_Pulse_Count < Circle_Outer_Wheel_Pulses + 25)) {
134     // Run this loop until the number of pulses read by the microcontroller reaches the calculated value above
135     L_Circle_Pulse_Count = getEncoderLeftCnt();     // Read the left encoder value
136     R_Circle_Pulse_Count = getEncoderRightCnt();    // Read the right encoder value
137
138     if((R_Circle_Pulse_Count + 10 < L_Circle_Pulse_Count * SpeedRatio)) {
139       // If the right motor is driving significantly slower than the SpeedRatio has specified, then speed up the right motor
140       setMotorSpeed(RIGHT_MOTOR, MotorSpeed + 1);
141     }
142
143     if((R_Circle_Pulse_Count + 10 > L_Circle_Pulse_Count * SpeedRatio)) {
144       // If the right motor is driving significantly faster than the SpeedRatio has specified, then slow down the right motor
145       setMotorSpeed(RIGHT_MOTOR, MotorSpeed - 1);
146     }
147
148     if((R_Circle_Pulse_Count >= Circle_Outer_Wheel_Pulses + 25)) {
149       // If the number of pulses reaches the calculated value, turn off the motors and stop running the function
150       disableMotor(LEFT_MOTOR);      // Turn off the left motor
151       disableMotor(RIGHT_MOTOR);     // Turn off the right motor
152       delay(1000);
153     }
154
155     // Print encoder counts to the serial monitor for debugging
156     Serial.print("Driving in a Circle Now");
157     Serial.print("\t");
158     Serial.print("Left Encoder Circle: ");
159     Serial.print(L_Circle_Pulse_Count);
160     Serial.print("\t");
161     Serial.print("Right Encoder Circle: ");
162     Serial.println(R_Circle_Pulse_Count);
163     delay(100);
164   }
165 }
166
167 void loop() {
168   // Make the robot perform the following functions sequentially with a one second delay between each phase
169   Serial.print("Straight Pulses: \t");
170   Serial.println(Straight_pulses);
171   Serial.print("TurnCCW Pulses: \t");
172   Serial.println(TurnCCW_pulses);
173   Serial.print("Circle Inner Wheel Pulses: \t");
174   Serial.println(Circle_Inner_Wheel_Pulses);
175   Serial.print("Circle Outer Wheel Pulses: \t");
176   Serial.println(Circle_Outer_Wheel_Pulses);
177   Serial.print("Speed Ratio: \t");
178   Serial.println(SpeedRatio);
179   Serial.print("Left Motor Speed: \t");
180   Serial.println(LeftMotorSpeed);
```

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

```
181
182   delay(3000);       // Wait 3 seconds for setting the robot in position
183   Drive_Straight();    // Perform the function for driving straight X centimeters
184   delay(1000);
185   Rotate_CCW();       // Perform the function for rotating 90 degrees CCW
186   delay(1000);
187   Drive_Circle();      // Perform the function for driving in a circle of radius X cm
188   delay(1000);
189   Rotate_CCW();       // Perform the function for rotating 90 degrees CCW
190   delay(1000);
191   Drive_Straight();    // Perform the function for driving straight back to the original start position
192  }
```