ECGR 4161/5196 Lab 4 Report **Due 06/22/22**

Name 1: Joshua Ayers

Student email ID: jayers18

Name 2: Ayman Alhadainy

Student email ID: aalhadai

Lab Objective:

During this lab, the robot was programmed to drive in two shapes; an equilateral triangle of side length 1.2 meters and a hexagon of side length 0.6 meters respectively. The left button was programmed to initiate the triangle pathing sequence and the right button was programmed to initiate the hexagon pathing sequence. In order to make sure that both wheels are driving at the same speed, the left and right motor speed were increment or decremented in response to the encoder counts. Furthermore, the angle for each side of both shapes was calculated in order to make the robot do the turns at the edges of the triangle and hexagon correctly. Finally, both motors in the robot were set to low speed to make sure the robot does not lose track while moving in a straight line or while making turns at edges of both shapes. The previous steps were done to make sure the robot accomplishes both tasks successively.

Commentary and Conclusion:

During this lab, the only sensor used was the motor encoder to determine the positional data and rotational data. No issue occurred while making the robot drive straight, however, some issues occurred while making the robot turn at both shape edges. The robot was making an extra 360 degree turn at the beginning at each of the edges. To have this issue solved, the circle direction was inversed and then multiplied with the ratio of the degree each team needed to turn divided by 360. After solving this issue, no other occurred.

Lab Code:

```
#include "Energia.h"
#include "SimpleRSLK.h"

void setup()
{
  setupRSLK();
  pinMode(LP_S1_PIN,INPUT_PULLUP); // right buttion
  pinMode(74,INPUT_PULLUP); // left bution
```

```
}

void DRV_STR(int distance,int speed) // takes arbitrary distance in CM
{
    // this section will determine the number of encoder pulses for the
    int internal_ratio = 360; // number of encoder pulses per wheel rotation
    const float pi = 3.1415926572; // aproximation of pi
    int wheeldiam = 7; // aproximation of wheel diamiter
    float wheelCurcumfrence = 7*pi;
    float numRotations = distance/wheelCurcumfrence; // this number represents the
wheel rotations that
    int finalPulses = numRotations*internal_ratio; // this represent the number of
pulses

    int LMS = speed; // left motor speed
    int RMS = speed; // Right motor speed
    // motors setup
    resetLeftEncoderCnt();
    resetRightEncoderCnt();
    enableMotor(BOTH_MOTORS);
    setMotorDirection(LEFT_MOTOR,MOTOR_DIR_FORWARD);
    setMotorDirection(RIGHT_MOTOR,MOTOR_DIR_FORWARD);
    //acivate motor with ramp up to aleveate jolt
    int encoderL = getEncoderLeftCnt();
    int encoderR = getEncoderRightCnt();
    setMotorSpeed(LEFT_MOTOR,1);
    setMotorSpeed(RIGHT_MOTOR,1);
    delay(15);

    while((((encoderL = getEncoderLeftCnt()) + (encoderR =
getEncoderRightCnt()))/2)<finalPulses)// compaire the average of the encoder
values on each wheel with the required number of pulses
    {

     if(encoderL<(encoderR+3)) // comao
      {
        setMotorSpeed(LEFT_MOTOR,LMS+1);
        setMotorSpeed(RIGHT_MOTOR,RMS-1);
      }
     else if(encoderL>(encoderR+3))
      {
        setMotorSpeed(LEFT_MOTOR,LMS-1);
        setMotorSpeed(RIGHT_MOTOR,RMS+1);
      }
    }
    disableMotor(BOTH_MOTORS);
    resetLeftEncoderCnt();
    resetRightEncoderCnt();
}

void PVT_TURN_DEG(int TFLG,int DFLG,int deg, int speed) // REMEMBER IT IS THE
EXTERNAL ANGLE NOT INTERNAL ANGLE
```

```
{
   // TFLAG indicates which turn direction relative to the robot the user wishes
to pivot 0=L 1=R (defalts = L)
   // DFLAG indicates which direction the robot is driving the user wishes to
pivot 0=F 1=B (defalts = F)
   // DEG indicates the number of degrees that the ro

   // BELOW IS BASIC SETUP
   disableMotor(BOTH_MOTORS); // just makes sure that the robot is stoped
   delay(10); // stop alows robot to come to a halt
   resetLeftEncoderCnt();
   resetRightEncoderCnt();
   enableMotor(BOTH_MOTORS);

  int currentDeg = 0;

  switch (DFLG){  // SWITCH STATEMENT FOR SETTING THE MOTOR DIRECTION
  case 1:

    setMotorDirection(LEFT_MOTOR,MOTOR_DIR_FORWARD);
    setMotorDirection(RIGHT_MOTOR,MOTOR_DIR_FORWARD);
    break;
  case 0:

    setMotorDirection(LEFT_MOTOR,MOTOR_DIR_BACKWARD);
    setMotorDirection(RIGHT_MOTOR,MOTOR_DIR_BACKWARD);
    break;
  default:

    setMotorDirection(LEFT_MOTOR,MOTOR_DIR_FORWARD);
    setMotorDirection(RIGHT_MOTOR,MOTOR_DIR_FORWARD);
    break;
  }

  switch (TFLG){  // SWITCH STATEMENT FOR SETTING THE MOTOR TO BEING TURNING
  case 1:

    setMotorSpeed(RIGHT_MOTOR,speed);
    currentDeg = getEncoderRightCnt()/3;
    break;
  case 0:

   setMotorSpeed(LEFT_MOTOR,speed);
   currentDeg = getEncoderLeftCnt()/3;
    break;
  default:
    setMotorSpeed(RIGHT_MOTOR,speed);
    currentDeg = getEncoderRightCnt()/3;
    break;
  }

  //the encoder pulses 360 times for a 90 degree turn
```

```
113   //Thus we will need to have 4*360 pulses or 1440
114   while(currentDeg<deg) // LOOP POLLS MOTOR ENCODERS (NEED TO LEARN HOW TO TRIGGER
115 INTERUPS)
116   {
117   switch (TFLG){
118     case 1:
119       currentDeg = getEncoderRightCnt()/3;
120       break;
121     case 0:
122       currentDeg = getEncoderLeftCnt()/3;
123       break;
124     default:
125       currentDeg = getEncoderRightCnt()/3;
126       break;
127     }
128   }
129   // DISSABLES ENCODERS AND MOTORS FOR SETUP FOR NEXT ACTION
130   disableMotor(BOTH_MOTORS);
131   resetLeftEncoderCnt();
132   resetRightEncoderCnt();
133   enableMotor(BOTH_MOTORS);
134 }
135
136 void SATIC_TURN_DEG(int TFLG,int deg, int speed) // REMEMBER IT IS THE EXTERNAL
137 ANGLE NOT INTERNAL ANGLE
138 {
139    // TFLAG indicates which turn direction relative to the robot the user wishes
140 to pivot 0=L 1=R (defalts = L)
141    // DFLAG indicates which direction the robot is driving the user wishes to
142 pivot 0=F 1=B (defalts = F)
143    // DEG indicates the number of degrees that the robot turns
144
145    // BELOW IS BASIC SETUP
146    disableMotor(BOTH_MOTORS); // just makes sure that the robot is stoped
147    delay(50); // stop alows robot to come to a halt
148    resetLeftEncoderCnt();
149    resetRightEncoderCnt();
150    enableMotor(BOTH_MOTORS);
151
152   int currentDeg = 0;
153
154   switch (TFLG){  // SWITCH STATEMENT FOR SETTING THE MOTOR TO BEING TURNING
155   case 1:
156     setMotorDirection(LEFT_MOTOR,MOTOR_DIR_FORWARD);
157     setMotorDirection(RIGHT_MOTOR,MOTOR_DIR_BACKWARD);
158     setMotorSpeed(BOTH_MOTORS,speed);
159     currentDeg = (getEncoderLeftCnt()/6)+(getEncoderRightCnt()/6)/2;
160     break;
161   case 0:
162    setMotorDirection(LEFT_MOTOR,MOTOR_DIR_BACKWARD);
163    setMotorDirection(RIGHT_MOTOR,MOTOR_DIR_FORWARD);
164    setMotorSpeed(BOTH_MOTORS,speed);
```

```
165    currentDeg = (getEncoderLeftCnt()/6)+(getEncoderRightCnt()/6)/2;
166     break;
167   default:
168     setMotorSpeed(BOTH_MOTORS,speed);
169     currentDeg = getEncoderRightCnt()/3;
170     break;
171   }
172
173   //the encoder pulses 360 times for a 90 degree turn
174   //Thus we will need to have 4*360 pulses or 1440
175   while(currentDeg<deg) // LOOP POLLS MOTOR ENCODERS (NEED TO LEARN HOW TO TRIGGER
176 INTERUPS)
177   {
178       currentDeg = (getEncoderLeftCnt()/6)+(getEncoderRightCnt()/6)/2;
179   }
180   // DISSABLES ENCODERS AND MOTORS FOR SETUP FOR NEXT ACTION
181   disableMotor(BOTH_MOTORS);
182   resetLeftEncoderCnt();
183   resetRightEncoderCnt();
184   enableMotor(BOTH_MOTORS);
185 }
186
187 void loop()
188 {
189      if(digitalRead(74)==0)
190      {
191       delay(2500);
192       for(int i=0;i<3;i++)
193         {
194           delay(500);
195          DRV_STR(120,22);
196          SATIC_TURN_DEG(0,57.5,22);
197         }
198      }
199      else if(digitalRead(LP_S1_PIN)==0)
200      {
201       delay(2500);
202       for(int i=0;i<6;i++)
203         {
204           delay(500);
205           DRV_STR(56,22);
206           SATIC_TURN_DEG(1,29,22);
207         }
208       }
209 }
210
```