```
/*
  Group 15
  Bharath Kumar
  Onkar Indurkar
  Intro to Robotics
  02-16-23
  LAB 5
*/
#include "SimpleRSLK.h"
uint16_t value;
uint16_t sensorVal[LS_NUM_SENSORS];
uint16_t sensorCalVal[LS_NUM_SENSORS];
uint16_t sensorMaxVal[LS_NUM_SENSORS];
uint16_t sensorMinVal[LS_NUM_SENSORS];
int cntPerRevolution = 360;
float Pi = 3.14;
float wheelDiameter = 7; /*in centimeters*/
bool isCalibrationComplete = false;
unsigned long timeBegin;
unsigned long timeEnd;

void setup() {
  Serial.begin(9600);
  setupRSLK();
  setupWaitBtn(LP_RIGHT_BTN);/* Right button on Launchpad */
  setupLed(GREEN_LED);       /* Green led in rgb led */
  setupLed(RED_LED);        /* Red led in rgb led */
  clearMinMax(sensorMinVal, sensorMaxVal);
}

void loop() {
  waitBtnPressed(LP_RIGHT_BTN);
  if (isCalibrationComplete == false) {
    simpleCalibrate();
    isCalibrationComplete = true;
  }
  waitBtnPressed(LP_RIGHT_BTN);
  blink(3);
  linefollowing();
}

/*******************
  Function name:blink
  Description: function to delay three seconds and blink Green LED at 1Hz
  Input: seconds to blink
  Return: void
*******************/
void blink(int cnt) {
  for (int i = 0; i < cnt; i++) {
    digitalWrite(GREEN_LED, HIGH);/* turn the LED on (HIGH is the voltage level)*/
    delay(500); /* wait for a second*/
    digitalWrite(GREEN_LED, LOW);/* turn the LED off by making the voltage LOW*/
    delay(500);
  }
}
```

```
/*******************
  Function name:distanceTraveled
  Description: calculate the distance traveled with the encoder pulse count
  Input: wheel diameter, pulse counts per revolution, encoder count
  Return: distance traveled in centimeters
*********************/
float distanceTraveled(float wheel_diam, float cnt_per_rev, float current_cnt) {
  float temp = (wheel_diam * Pi * current_cnt) / cnt_per_rev;
  return temp;
}

/*******************
  Function name:printing
  Description: Serial print the distance traveled and time taken
  Input: no input
  Return: void
*********************/
void printing() {
  unsigned long dist1 = distanceTraveled(7, 360, getEncoderRightCnt());
  unsigned long dist2 = distanceTraveled(7, 360, getEncoderLeftCnt());
  unsigned long dist = (dist1 + dist2) / 2;
  Serial.println("LAB Group 15");
  Serial.println("The robot travelled...");
  Serial.print(dist); /*distance*/
  Serial.println(" centimeters");
  Serial.print(dist * 0.01);
  Serial.println(" meters");
  Serial.print(dist * 0.393);
  Serial.println(" inches");
  Serial.print(dist * 0.0328);
  Serial.println(" feet\n");

  Serial.println("It took...");/*time and speed*/
  unsigned long timer = ((timeEnd - timeBegin) * 0.001);
  Serial.print(timer * 0.001);
  Serial.println(" seconds");
  Serial.println("at avg speed of...");
  Serial.print(dist / (timer * 0.001));
  Serial.println(" centimeters/second");
  Serial.print((dist * 2.237) / (timer * 0.001));
  Serial.println(" mph");
  Serial.println("\n\n");
  delay(5000);
}

/*******************
  Function name:simpleCalibrate
  Description:function to calibrate the robot before setting on the track
  Input: no input
  Return: void
*********************/
void simpleCalibrate() {
  setMotorDirection(BOTH_MOTORS, MOTOR_DIR_FORWARD); /* Set both motors direction forward */
  enableMotor(BOTH_MOTORS);/* Enable both motors */
```

```
  setMotorSpeed(BOTH_MOTORS, 20); /* Set both motors speed 20 */

  for (int x = 0; x < 100; x++) {
    readLineSensor(sensorVal);
    setSensorMinMax(sensorVal, sensorMinVal, sensorMaxVal);
  }

  disableMotor(BOTH_MOTORS);/* Disable both motors */
}

/*********************
  Function name:linefollowing
  Description:function to run the robot on the line until it encounters a 'T'
  Input: no input
  Return: void
**********************/
void linefollowing() {
  enableMotor(BOTH_MOTORS);
  /*setting the speeds for different turns*/
  float normalSpeed = 23.19; /* ratio = 3.449 */
  uint16_t fastSpeed = 80;

  float normalSpeed2 = 35;   /* ratio = 4.5714 */
  uint16_t fastSpeed2 = 160;

  float normalSpeed3 = 10;   /* ratio = 12 */
  uint16_t fastSpeed3 = 120;

  float straightSpeed = 40.5;

  resetRightEncoderCnt();/*resetting encoders*/
  resetLeftEncoderCnt();

  uint8_t lineColor = DARK_LINE; /*DARK_LINE  if your floor is lighter than your line*/
  timeBegin = micros();/*start timer*/
  while (true) {
    readLineSensor(sensorVal);
    readCalLineSensor(sensorVal, sensorCalVal, sensorMinVal, sensorMaxVal, lineColor);

    value = 0;
    uint32_t linePos = getLinePosition(sensorCalVal, lineColor);
    for (uint8_t i = 0; i < LS_NUM_SENSORS; i++) {
      value += sensorVal[i]; /*summing the values of all sensors*/
    }
    if (value > 19000) { /* if condition to check the sum of sensor values to disable the
motors*/
      timeEnd = micros(); /* read the timer value while disabling the motor*/
      disableMotor(BOTH_MOTORS);/*disable motors*/
      digitalWrite(RED_LED, HIGH);/*turn on RED LED while disabling motor*/
      break;
    }
    if (linePos > 0 && linePos < 1200) { /* Make an extreme left turn */
      pivotTurn(normalSpeed3, fastSpeed3);
    }
    else if (linePos > 5800 && linePos < 7000) { /* Make an extreme right turn */
```

```
      pivotTurn(fastSpeed3,normalSpeed3);
    }
    else if (linePos > 1200 && linePos < 2200) {  /* Make a left turn */
      pivotTurn(normalSpeed2, fastSpeed2);
    }
    else if (linePos > 4800 && linePos < 5800) { /* Make a right turn */
      pivotTurn(fastSpeed2,normalSpeed2);
    }
    else if (linePos > 3800 && linePos < 4800) {  /* Make a slight right turn */
      pivotTurn(fastSpeed,normalSpeed);
    }
    else if (linePos > 2200 && linePos < 3200) { /* Make a slight left turn */
      pivotTurn(normalSpeed, fastSpeed);
    }
    else { /* if linepos=3500 robot needs to go straight as line is detected under the middle
sensors*/
      straight(straightSpeed,straightSpeed);
    }
  }
  while (true) printing(); /* start the printing function after the robot stops running*/
}

/*******************
  Function name: pivotTurn
  Description:function to make the robot turn with different input speeds
  Input: left and right wheel speed
  Return: void
*********************/
void pivotTurn(float leftspd, float rightspd){
  setRawMotorSpeed(LEFT_MOTOR, leftspd);
  setRawMotorSpeed(RIGHT_MOTOR, rightspd);
}

/*******************
  Function name:straight
  Description:function to make the robot go straight
  Input: left and right wheel speed
  Return: void
*********************/
void straight(float leftspd, float rightspd){
      setRawMotorSpeed(LEFT_MOTOR, leftspd);
      setRawMotorSpeed(RIGHT_MOTOR, rightspd);
}
```