1. The robot continues its last movement command for the specified time. For example:

sparki.moveForward();  
delay(100);

will move the robot forward for .1 seconds then continue code execution.

1. If the loop call takes longer than 100 ms, then the calculated distance will drift over time. We are using a precalculated speed to determine Sparki’s position over time. If the time is not perfectly accurate then the calculated location will not be accurate.
2. We calculated an average speed of 0.027845 m/s during the 30cm test.
3. Ideally, the calculated pose should be (0,0,0). This would indicate that Sparki has perfectly calculated its displacement from the datum.
4. Note: Our Axis are different with the positive x axis pointing forward from the start and the positive y axis points to the left of the start line.

First lap; x: 0m y: 0m theta: 351

Second Lap; x: 0m y: 0m theta: 705

Third Lap; x: 0m y: 0m theta: 1058

1. Detecting the start line and resetting the x,y, and theta back to their original values of 0
2. Austin Albert  
   Ian Brobin  
   Connor Thompson  
   Chandler Garthwaite
3. We took around 3 hours of lab time programming
4. Yes it does we did run into problems resetting the odometry at the start line and had a 15 degrees error. But resolved the problem in the end.

Code:

#include <Sparki.h>

#define CYCLE\_TIME .100 // seconds

#define FORWARD 0

#define RIGHT 1

#define LEFT 2

#define FORWARD\_VELOCITY 0.027845

#define ROTATIONAL\_VELOCITY 0.66

// Program States

#define CONTROLLER\_FOLLOW\_LINE 1

#define CONTROLLER\_DISTANCE\_MEASURE 2

int current\_state = CONTROLLER\_FOLLOW\_LINE; // Change this variable to determine which controller to run

const int threshold = 700;

int line\_left = 1000;

int line\_center = 1000;

int line\_right = 1000;

float pose\_x = 0., pose\_y = 0., pose\_theta = 0.;

int robot\_motion=FORWARD;

void resetOdometry(){

pose\_x = 0;

pose\_y = 0;

pose\_theta = 0;

}

void setup() {

pose\_x = 0.;

pose\_y = 0.;

pose\_theta = 0.;

}

void readSensors() {

line\_left = sparki.lineLeft();

line\_right = sparki.lineRight();

line\_center = sparki.lineCenter();

// distance = sparki.ping();

}

void moveRight(){

sparki.moveRight();

robot\_motion=RIGHT;

}

void moveLeft(){

sparki.moveLeft();

robot\_motion=LEFT;

}

void moveForward(){

sparki.moveForward();

robot\_motion=FORWARD;

}

void followLine() {

readSensors();

//start line

if ( (line\_center < threshold) && (line\_left < threshold) && (line\_right < threshold) )

{

resetOdometry();

moveForward(); // move forward

}

else if ( line\_left < threshold ) // if line is below left line sensor

{

moveLeft(); // turn left

}

else if ( line\_right < threshold ) // if line is below right line sensor

{

moveRight(); // turn right

}

// if the center line sensor is the only one reading a line

else if ( (line\_center < threshold) && (line\_left > threshold) && (line\_right > threshold) )

{

moveForward(); // move forward

}

}

void measure\_30cm\_speed() {

unsigned long startTime = millis();

sparki.moveForward(30);

unsigned long endTime = millis();

sparki.clearLCD();

sparki.println("Time Taken:");

sparki.println(endTime - startTime);

sparki.updateLCD();

}

void updateOdometry() {

float deltaX = 0;

float deltaY = 0;

float deltaTheta = 0;

switch(robot\_motion){

case FORWARD:

deltaX = CYCLE\_TIME \* FORWARD\_VELOCITY \* cos(pose\_theta);

deltaY = CYCLE\_TIME \* FORWARD\_VELOCITY \* sin(pose\_theta);

break;

case LEFT:

deltaTheta = CYCLE\_TIME \* ROTATIONAL\_VELOCITY;

break;

case RIGHT:

deltaTheta = -1 \* CYCLE\_TIME \* ROTATIONAL\_VELOCITY;

break;

}

pose\_x += deltaX;

pose\_y += deltaY;

pose\_theta += deltaTheta;

if (pose\_theta > (2 \* M\_PI)){

pose\_theta -= (2 \* M\_PI);

}

if (pose\_theta < (-2 \* M\_PI)){

pose\_theta += (2 \* M\_PI);

}

}

void displayOdometry() {

sparki.clearLCD(); // wipe the screen

sparki.print("pose\_x: "); // show left line sensor on screen

sparki.print(pose\_x);

sparki.println(" m");

sparki.print("pose\_y: "); // show center line sensor on screen

sparki.print(pose\_y);

sparki.println(" m");

sparki.print("pose\_theta: "); // show right line sensor on screen

sparki.print(pose\_theta \* (180 / M\_PI));

sparki.println(" d");

sparki.updateLCD(); // display all of the information written to the screen

}

void loop() {

// TODO: Insert loop timing/initialization code here

unsigned long startTime = millis();

switch (current\_state) {

case CONTROLLER\_FOLLOW\_LINE: {

followLine();

break;

}

case CONTROLLER\_DISTANCE\_MEASURE: {

measure\_30cm\_speed();

current\_state++;

break;

}

}

updateOdometry();

displayOdometry();

unsigned long endTime = millis();

// Ensure loop lasts 100ms every loop

delay(1000\*CYCLE\_TIME - (endTime - startTime));

}