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.

2. 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.
3. We calculated an average speed of 0.027845 m/s during the 30cm test.
4. Ideally, the calculated pose should be (0,0,0). This would indicate that Sparki has perfectly calculated its displacement from the datum.
5. 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
6. Detecting the start line and resetting the x,y, and theta back to their original values of 0
7. Austin Albert
   Ian Brobin
   Connor Thompson
   Chandler Garthwaite
8. We took around 3 hours of lab time programming
9. 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));
}
```