1. **Components Used:**
    - Teensy 4.0
    - Large Pololu Frame
    - 65mm TT Motor Wheels x4
    - TT Motor with Encoder x4
    - Feather Board Motor Driver
    - Battery 18650 x 4
    - Laptop - running Windows 10 (to remain connected to Teensy during run)

2. **Code For Task 1/1. (maneuver robot and compare dead reckoning)**

```
// Authors: Keanan Gabertan, Vincent Vuong
// CPE476 - Midterm
// This code will have the rover perform a
// set of maneuvers and calculate its dead reckoning
// based on its odometry

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include <DeadReckoner.h>

// ENCODER PINS
// TWO for each motor
// A - records every time there is a rising signal
// B - records forward or reverse
// ENCODER A is orange on motor encoder
#define FR_ENCODER_A 14
//#define FR_ENCODER_B 15
#define FL_ENCODER_A 11
//#define FL_ENCODER_B 10
#define BR_ENCODER_A 22
//#define BR_ENCODER_B 21
#define BL_ENCODER_A 2
//#define BL_ENCODER_B 3



// MEASUREMENTS
// The units for all measurements must be consistent.
```

```
// You can use any length unit as desired.
#define RADIUS 40 // wheel radius in mm
#define LENGTH 150 // wheel base length in mm

// Ticks per rev based on its 1:45 gear ratio and 3 magnetometers
#define TICKS_PER_REV 135

// TIME INTERVALS
#define POSITION_COMPUTE_INTERVAL 50 // milliseconds
#define SEND_INTERVAL 100            // milliseconds


// Number of left and right tick counts on the encoder.
volatile unsigned long leftTicks, rightTicks;

// Previous times for computing elapsed time.
unsigned long prevPositionComputeTime = 0, prevSendTime = 0;

// Previous x and y coordinate.
double prevX = 0, prevY = 0;

// Instantiate motorshield object and motors
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

Adafruit_DCMotor *front_left = AFMS.getMotor(1);
Adafruit_DCMotor *front_right = AFMS.getMotor(2);
Adafruit_DCMotor *back_left = AFMS.getMotor(3);
Adafruit_DCMotor *back_right = AFMS.getMotor(4);

// Instantiate deadReckoner object
DeadReckoner deadReckoner(&leftTicks, &rightTicks, TICKS_PER_REV,
RADIUS, LENGTH);

// Manuever functions
void goForward()
{
    int i=0;
```

```
    back_left->run(FORWARD);
    front_right->run(FORWARD);
    front_left->run(FORWARD);
    back_right->run(FORWARD);

    for (i = 0; i < 255; i++)
    {
        front_right->setSpeed(i);
        front_left->setSpeed(i);
        back_left->setSpeed(i);
        back_right->setSpeed(i);
        delay(10);
    }
}

void turnLeft()
{
  int i=0;
  front_right->run(FORWARD);
  back_right->run(FORWARD);
  //front_left->run(BACKWARD);
  //back_left->run(BACKWARD);
  front_left->setSpeed(15);
  back_left->setSpeed(10);

  for (i=0; i<255; i++) {
      front_right->setSpeed(i);
      back_right->setSpeed(i);

      delay(10);
  }
}

void turnRight()
{
  int i=0;
  //front_right->run(BACKWARD);
```

```
  //back_right->run(BACKWARD);
  front_left->run(FORWARD);
  back_left->run(FORWARD);
  front_right->setSpeed(15);
  back_right->setSpeed(10);

  for (i=0; i<255; i++) {
      front_left->setSpeed(i);
      back_left->setSpeed(i);
      delay(10);
  }
}

void calculateDR() {
  if (millis() - prevPositionComputeTime > POSITION_COMPUTE_INTERVAL) {
    // Computes the new angular velocities and uses that to compute the
new position.
    // The accuracy of the position estimate will increase with smaller
time interval until a certain point.
    deadReckoner.computePosition();
    prevPositionComputeTime = millis();
  }

  if (millis() - prevSendTime > SEND_INTERVAL) {
    // Cartesian coordinate of latest location estimate.
    // Length unit correspond to the one specified under MEASUREMENTS.
    double x = deadReckoner.getX();
    double y = deadReckoner.getY();

    // Left and right angular velocities.
    double wl = deadReckoner.getWl();
    double wr = deadReckoner.getWr();

    // getTheta method returns the robot position angle in radians
measured from the x axis to the center of the robot.
    // This angle is set initially at zero before the robot starts
moving.
```

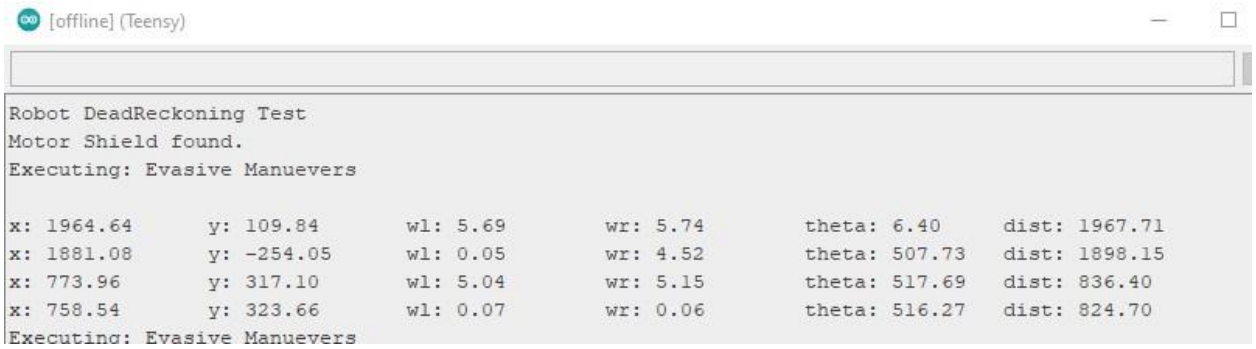
```
    double theta = deadReckoner.getTheta();

    // Total distance robot has troubled.
    double distance = sqrt(x * x + y * y);

    Serial.print("x: "); Serial.print(x);
    Serial.print("\ty: "); Serial.print(y);
    Serial.print("\twl: "); Serial.print(wl);
    Serial.print("\twr: "); Serial.print(wr);
    Serial.print("\ttheta: "); Serial.print(theta*RAD_TO_DEG); // theta
converted to degrees.
    Serial.print("\tdist: "); Serial.println(distance);

    prevSendTime = millis();
  }
}

// If motor encoders interrupt, increase tick count
void pulseLeft() { leftTicks++; }
void pulseRight() { rightTicks++; }

void attachInterrupts() {
  attachInterrupt(digitalPinToInterrupt(FL_ENCODER_A), pulseLeft,
RISING);
  attachInterrupt(digitalPinToInterrupt(FR_ENCODER_A), pulseRight,
RISING);
}

void setup() {
  //attachInterrupts();
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Robot DeadReckoning Test");

  if (!AFMS.begin()) {         // create with the default frequency
1.6KHz
    Serial.println("Could not find Motor Shield. Check wiring.");
    while (1);
```

```
  }
  Serial.println("Motor Shield found.");

  // Set the speed to start, from 0 (off) to 255 (max speed)
  front_right->setSpeed(10);
  front_left->setSpeed(10);
  back_right->setSpeed(10);
  back_left->setSpeed(10);

  // turn on motor
  front_right->run(RELEASE);
  front_left->run(RELEASE);
  back_right->run(RELEASE);
  back_left->run(RELEASE);
  attachInterrupts();
}

void loop() {
  uint8_t i=0;

  Serial.print("Executing: Evasive Manuevers\n\n");
  calculateDR();

  if(i==0) {
    i++;
    // Time step: 0 seconds -------------------------
    // Run Forward for 5 sec ------------------------
    goForward();
    delay(5000);
    calculateDR();
    // Time step: 5 seconds -------------------------
    // Turn Left for 4.5 sec ------------------------
    turnLeft();
    delay(4500);
    calculateDR();
    // Time step: 9.5 seconds -----------------------
    // Run Forward for 3 sec ------------------------
```

```
    goForward();
    delay(3000);
    calculateDR();
    // Time step: 12.5 seconds ----------------------
    // Turn Right for 3.5 seconds --------------------
    turnRight();
    delay(3500);
    calculateDR();
    // Time step: 16 seconds -------------------------

    front_right->run(RELEASE);
    front_left->run(RELEASE);
    back_right->run(RELEASE);
    back_left->run(RELEASE);
  }
}
```

3. Output from Serial Monitor

Run 1 - Error



Run 2 - More accurate

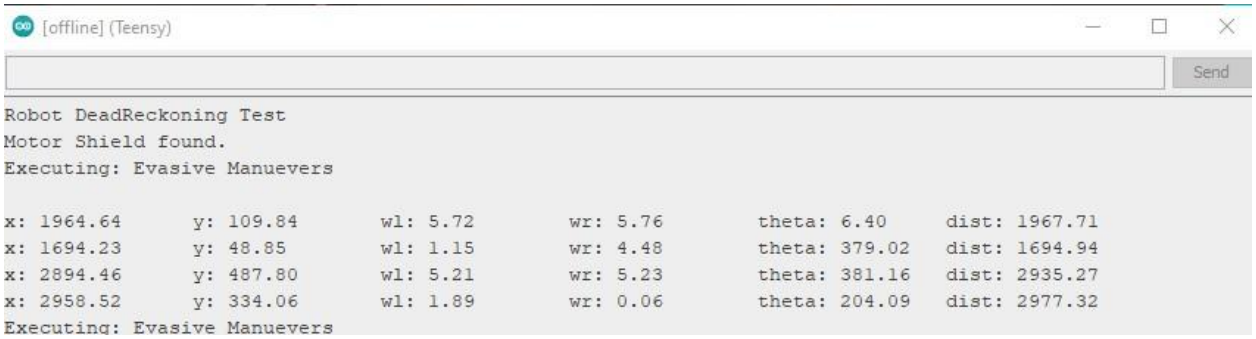
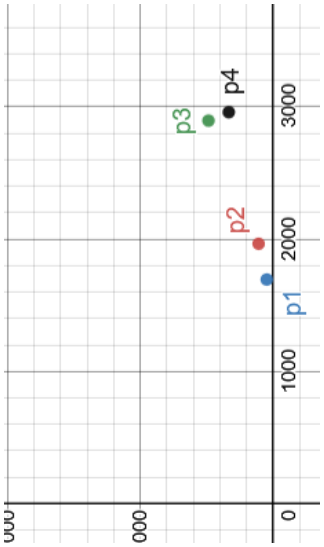


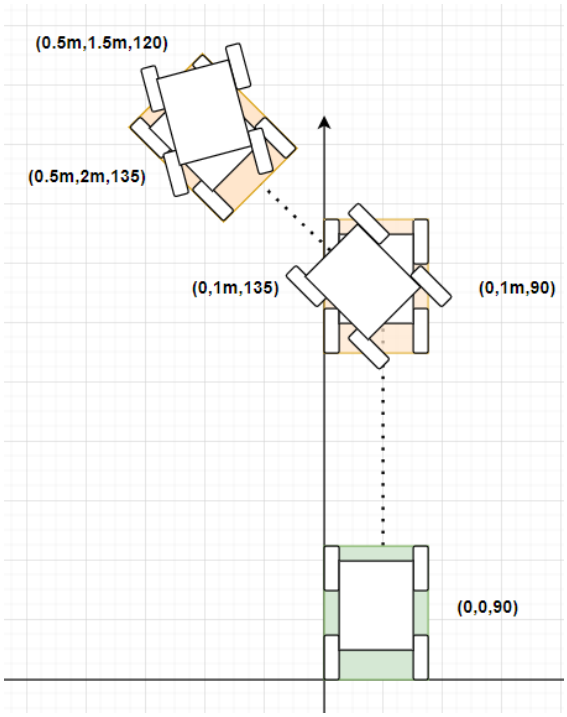
Figure [4]: Experimental Results

After transposing data:
After going straight:
(x1, y1, theta1) = (0.12m,1.96m,96.4)
After left turn:
(x2,y2, theta2) = (0.05m,1.69m,109.02)
After going straight:
(x3,y3, theta3) = (0.49m,2.89m,111.16)
After right turn:
(x4,y4, theta4) = (0.334m,2.96m,294.09)



<- Dead reckoning data (x,y) imposed onto graph

4. **Theoretical Dead Reckoning Calculations:**



Hand Calculated Theoretical Values Using Forward Kinematics

Equal (constant) forward speed for both wheels

$v_L = v_R = v$

Also for interval-wise changes in the common velocity

$x(t) = x_0 + vt\cos(\theta)$
$y(t) = y_0 + vt\sin(\theta)$
$\theta(t) = \theta_0$

The robot moves along a straight trajectory

<u>Time step 0:</u>
x(0) = 0
y(0) = 0

<u>Time step 1:</u>
Given `theta = w(t),
R = 40mm (wheel) radius, w = angular velocity
V = r x w = 40 * 90 = .36m/s
x(5) = 0 + v(5)cos(90) = 0
y(5) = 0 + v(5)sin(90) = 1.8m
t(1) = (0,1.8m,90)

<u>Time step 2</u>:
Given theta at t_2 = 135
V = r x w = 40 * 135 = .54m/s
x(9.5) = 0 + v(4.5)cos(135) = -0.38

y(9.5) = 1.8 + v(4.5)sin(135) = 5.4?
t(2) = (-0.38,5.4m,135)

Time step 3 :
Given theta at t_3 = 135
V = r x w = 40 * 135 = .54m/s
x(9.5) = -0.38 + v(3)cos(135) = -1.5m
y(9.5) = 5.4? + v(3)sin(135) = 6.5m?
t(3) = (-1.5,6.5,135)

Time step 4 :
Given theta at t_4 = 120
V = r x w = 40 * 135 = .48m/s
x(12.5) = -1.5m + v(3.5)cos(120) = -2.34m
y(12.5) = 6.5m? + v(3.5)sin(120) = 7.9m?
t(4) = (-2.34,7.9,120)

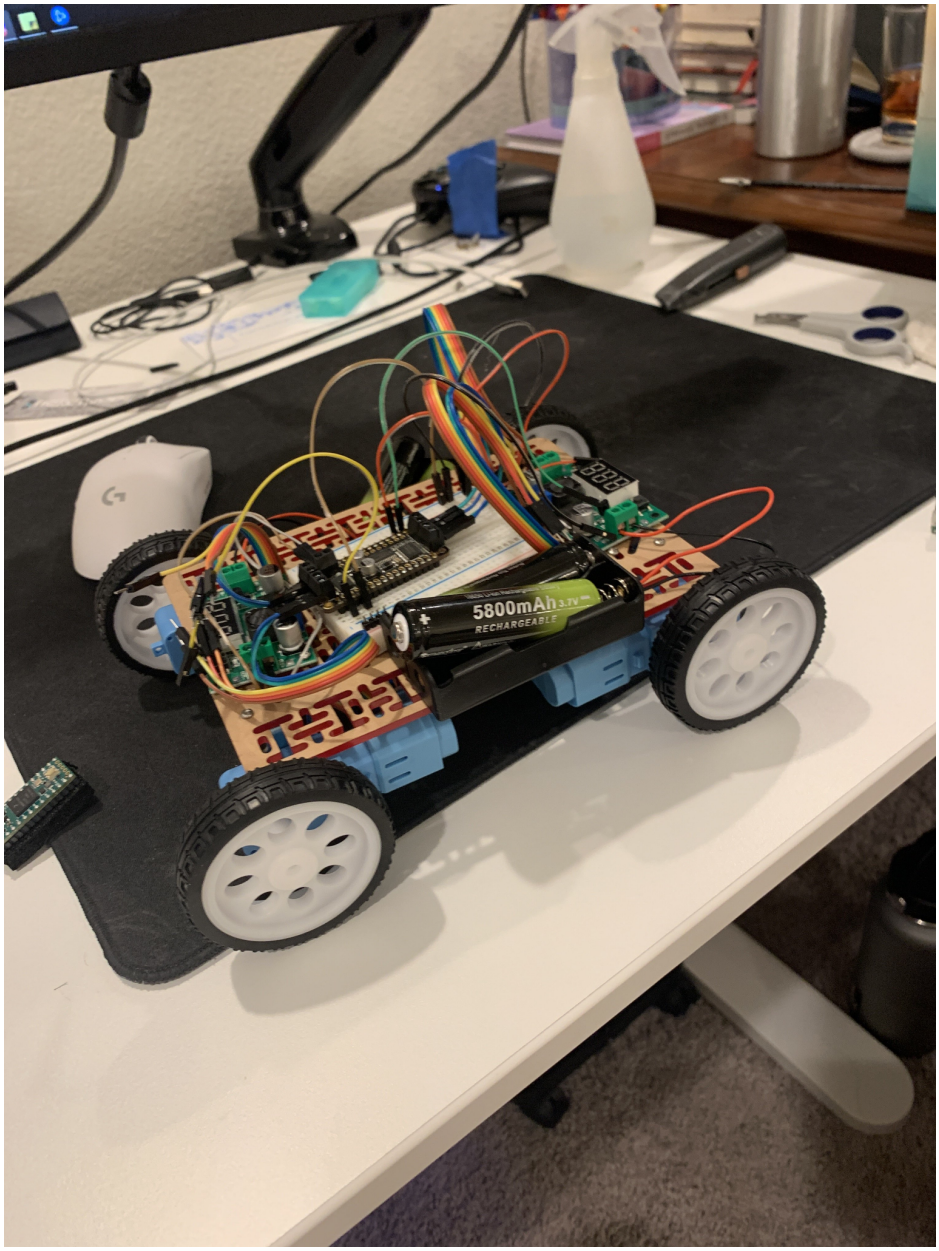
5. Error Calculation

(expected value - actual value observed)

Error between diagram and recorded values	Error between hand calculations and recorded values
Time step 1 error: X : 0-0.12 = 0.12 Y : 1-1.96 = 0.96 Theta : 90 - 96.4 = 6.4 Time step 2 error: X : 0-0.05 = 0.05 Y : 1-1.69 = 0.69 Theta : 135 -109.-3 = 26.3 Time step 3 error: X : 0.5-0.49 = 0.44 Y : 2-2.89 = 0.89 Theta : 135 - 111.16 = 23.84 Time step 4 error: X : 0.5-0.334 = 0.166 Y : 1.5-2.96 = 1.19 Theta : 120 - 294.09 = 174.09	Time step 1 error: X : 0-0.12 = 0.12 Y : 1.8-1.96 = 0.16 Theta : 90 - 96.4 = 6.4 Time step 2 error: X : 0.38-0.05 = 0.33 Y : 5.4-1.69 = 3.71 Theta : 135 -109.-3 = 26.3 Time step 3 error: X : 1.5-0.49 = 1.01 Y : 6.5-2.89 = 3.61 Theta : 135 - 111.16 = 23.84 Time step 4 error: X : 2.34-0.334 = 2.1 Y : 7.9-2.96 = 4.49 Theta : 120 - 294.09 = 174.09

6. **Board Setup**

***** Additionally, we had a windows laptop connected to the rover while testing to monitor the serial monitor data



7. **Video Link of Demo**
   https://youtu.be/cNK7KlY-8Co
8. **Github link to the shared repository**
   https://github.com/EyeEsquire/cpe476.git