CSE3910 Project D - Phidgets Rover: Reflection Log

Zephram Gilson

---

move.java

Originally, when I copied the code provided on the website into my IDE, the motors' velocities were both set to 1.

```java
public class move {
    public static void main(String[] args) throws Exception {

        // Connect to wireless rover
        Net.addServer("", "192.168.100.1", 5661, "", 0);

        // Create
        DCMotor leftMotors = new DCMotor();
        DCMotor rightMotors = new DCMotor();

        // Address
        leftMotors.setChannel(0);
        rightMotors.setChannel(1);

        // Open
        leftMotors.open(5000);
        rightMotors.open(5000);

        // Move forward at full speed
        leftMotors.setTargetVelocity(1);
        rightMotors.setTargetVelocity(1);

        // Wait for 1 second
        Thread.sleep(1000);

        // Stop motors
        leftMotors.setTargetVelocity(0);
        rightMotors.setTargetVelocity(0);
    }
}
```

This resulted in the rover moving backwards. To fix this, I changed the target velocities to -1, making the motors run the opposite direction, effectively making the rover move forward.

From this point, I experimented with moving at different speeds, opposite directions, and with modified `sleep` time.

turn.java

```java
public class turn {
    public static void main(String[] args) throws Exception {

        // Connect to wireless rover
        Net.addServer("", "192.168.100.1", 5661, "", 0);

        // Create
        DCMotor leftMotors = new DCMotor();
        DCMotor rightMotors = new DCMotor();

        // Address
        leftMotors.setChannel(0);
        rightMotors.setChannel(1);

        // Open
        leftMotors.open(5000);
        rightMotors.open(5000);

        // Turn in one direction
        leftMotors.setTargetVelocity(1);
        rightMotors.setTargetVelocity(-1);

        //Wait for 2 second
        Thread.sleep(2000);

        // Stop motors
        leftMotors.setTargetVelocity(0);
        rightMotors.setTargetVelocity(0);
    }
}
```

The code provided on the website made the rover spin 540 degrees (1.5 rotations), so I experimented with the `Thread.sleep(2000)` (the duration which the motors are in motion), which resulted in the rover turning for more time (more rotations) or less time (less rotations).

Similarly, I experimented with the motors' target velocities which resulted in the rover being able to turn faster, making more rotations within the time (sleep) period, and the rover being able to turn slower, making less rotations within the sleep period.

```java
    // Open
    leftMotors.open(5000);
    rightMotors.open(5000);

    // Turn in one direction
    leftMotors.setTargetVelocity(.5);
    rightMotors.setTargetVelocity(-.5);

    //Wait for 2 second
    Thread.sleep(4000);

    // Stop motors
    leftMotors.setTargetVelocity(0);
    rightMotors.setTargetVelocity(0);
}
```

(Example of experimentation, the motors move at half the original speed, but for twice as long, theoretically causing a result of the rover undergoing the same amount of rotations as the original code.)

With the `turn` program, I also experimented with turning in opposite directions, as well as moving + turning.

avoidObstacles.java

```
    // Address
    leftMotors.setChannel(0);
    rightMotors.setChannel(1);

    // Open
    leftMotors.open(5000);
    rightMotors.open(5000);
    sonar.open(5000);

    while (true) {

        System.out.println("Distance: " + sonar.getDistance() + " mm");

        if (sonar.getDistance() < 200) {
            // Object detected! Stop motors
            leftMotors.setTargetVelocity(0);
            rightMotors.setTargetVelocity(0);
        } else {
            // Move forward slowly (25% max speed)
            leftMotors.setTargetVelocity(-0.25);
            rightMotors.setTargetVelocity(-0.25);
        }

        // Wait for 250milliseconds
        Thread.sleep(250);
    }
}
```

With the code provided on the website, the rover moves at 25% of the max speed, allowing it some reaction time for when it detects that it is within 200mm of an obstacle (detected by the sonar)

I experimented with lowering the distance in the `if` statement, however if made too low, the rover won't be able to fully stop its motors before it hits the obstacle.

Similarly, I experimented with adjusting the speed of the motors, which caused the same result; the rover can't slow down its motors all the way before crashing into the obstacle.

Both of these experiments brought me to the conclusion that there is, theoretically, an optimal combination of motor speed and sonar distance to result in the rover stopping just before the obstacle, however it is safer for the rover to not move too fast, allowing some buffer stopping time rather than stopping unnecessarily far away.

With this program, I also experimented with turning away to move around the obstacle rather than just stopping, as well as modifying the sonar Phidget's data interval so it updates every 100 milliseconds (10 times a second) instead of every 250 milliseconds (4 times a second).

challenge.java

The challenge of this project was to make the rover travel 1m, then turn 90 degrees, and repeat such that it travels in a square and ends in the same place.

Some factors that made this challenge difficult included:

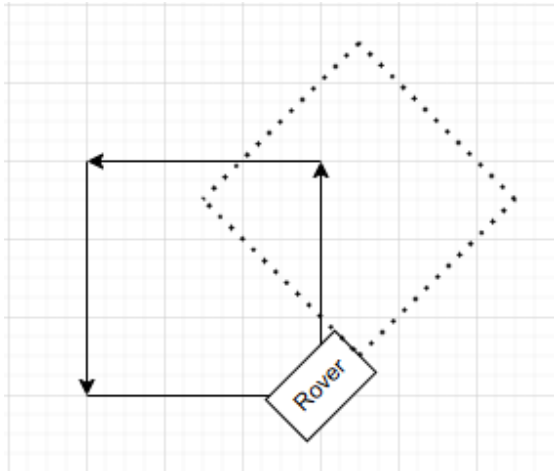
1. Needing precise values for how long the rover should travel
2. Slight variations throughout every test run due to physical factors such as dust on the tires/motors
3. Needing to account for how much the rover moves forward while turning 90 degrees (must be taken into consideration so that the value of time for how long the motors move the rover forward can be adjusted accordingly)

To combat these challenges,

1. Many trials were needed to get as accurate values as possible
2. I cleaned the floor in between trials as well as wiping dust off of the tires of the rover to try to maintain consistency
3. I needed to make an adjustment to my `for` loop;
    a. Originally, it would simply move forward and turn, and repeat this process 4 times.

However, on every rotation, I would only set my rover to turn about 45 degrees, as it would physically turn the next 45 degrees while moving forward on the next respective loop of the process (discovered based on experimentation).

This meant that at the very end, the rover would end at a ~45 degree angle, rather than in a straight orientation which would allow the user to run the program again, and the rover would move in the same exact way rather than needing to be adjusted.



This is a visual representation of the trajectory of the rover. In this diagram, the rover is in its final position after one full rotation (depicted by solid arrows), and the dotted lines depict the next predicted trajectory.

Ideally, the rover should end in an orientation that would allow the trajectory to remain the same (or as close as possible) no matter how many times the program is run.

Here is what my code looked like with this error:

```
// Define the time to drive forward 1m (adjust based on rover speed)
int driveTime = 936; // Approximate time for 1 meter at full speed
int turnTime = 330; // Approximate time for a 90° turn

// Repeat for four sides of the square
for (int i = 0; i < 4; i++) {
    // Move forward for 1 meter
    leftMotors.setTargetVelocity(-1);
    rightMotors.setTargetVelocity(-1);
    Thread.sleep(driveTime);

    // Stop motors briefly
    leftMotors.setTargetVelocity(0);
    rightMotors.setTargetVelocity(0);
    Thread.sleep(500);

    // Turn 90 degrees
    leftMotors.setTargetVelocity(1);
    rightMotors.setTargetVelocity(-1);
    Thread.sleep(turnTime);

    // Stop motors briefly
    leftMotors.setTargetVelocity(0);
    rightMotors.setTargetVelocity(0);
    Thread.sleep(500);
}

// Stop and close motors after completing the square
leftMotors.setTargetVelocity(0);
```

To fix this, I adjusted the code so that the last iteration of the loop is modified and includes a final rotation of the rover to correct its orientation:

```
// Repeat for four sides of the square
for (int i = 0; i < 4; i++) {
    // Move forward for 1 meter
    leftMotors.setTargetVelocity(-1);
    rightMotors.setTargetVelocity(-1);
    Thread.sleep(driveTime);

    // Stop motors briefly
    leftMotors.setTargetVelocity(0);
    rightMotors.setTargetVelocity(0);
    Thread.sleep(500);

    // Turn 90 degrees (adjust last turn slightly)
    if (i == 3) {
        // Last turn includes slight adjustment
        leftMotors.setTargetVelocity(1);
        rightMotors.setTargetVelocity(-1);
        Thread.sleep(turnTime + finalTurnAdjustment);
    } else {
        // Standard 90° turn
        leftMotors.setTargetVelocity(1);
        rightMotors.setTargetVelocity(-1);
        Thread.sleep(turnTime);
    }

    // Stop motors briefly
    leftMotors.setTargetVelocity(0);
    rightMotors.setTargetVelocity(0);
    Thread.sleep(500);
}
```

After making all the slight adjustments to the times (based on many trials):

```
// Define the time to drive forward 1m (adjust based on rover speed)
int driveTime = 936; // Approximate time for 1 meter at full speed
int turnTime = 330; // Approximate time for a 90° turn
int finalTurnAdjustment = 72; // Additional time to straighten out at the end
```

The rover effectively moved in a square and ended in the same spot.

Lastly, I incorporated the condition that If the rover detects an object, it should spin 180° and move in the opposite direction:

```
// Define the time to drive forward 1m (adjust based on rover speed)
int driveTime = 936; // Approximate time for 1 meter at full speed
int turnTime = 330; // Approximate time for a 90° turn
int spin180Time = 660; // Approximate time for a 180° turn
int finalTurnAdjustment = 72; // Additional time to straighten out at the end

// Repeat for four sides of the square
for (int i = 0; i < 4; i++) {
    // Move forward for 1 meter while checking for obstacles
    long startTime = System.currentTimeMillis();
    long endTime = startTime + driveTime;

    while (System.currentTimeMillis() < endTime) {
        // Check distance sensor for obstacles
        if (distanceSensor.getDistance() < 500) { // Adjust distance threshold as needed
            // Object detected: spin 180° and move in opposite direction
            leftMotors.setTargetVelocity(1);
            rightMotors.setTargetVelocity(-1);
            Thread.sleep(spin180Time);

            // Stop briefly
            leftMotors.setTargetVelocity(0);
            rightMotors.setTargetVelocity(0);
            Thread.sleep(500);

            // Reset the drive timer for moving opposite direction
            startTime = System.currentTimeMillis();
            endTime = startTime + driveTime;
        } else {
            // Continue moving forward
            leftMotors.setTargetVelocity(-1);
            rightMotors.setTargetVelocity(-1);
        }
    }

    // Stop motors briefly after moving forward
    leftMotors.setTargetVelocity(0);
    rightMotors.setTargetVelocity(0);
    Thread.sleep(500);
```

To test it further, I tried running the code twice without adjusting it in between, and it still worked as intended.

Seeing my peers' approaches, I noticed I could have made the rover move slower, possibly making it more simple for more accuracy with turns and a `driveTime` value. However, I decided to keep my program the way it was as it still worked as desired.

These are all the changes I made and challenges I encountered in my coding process.