Credit Name: CSE3910 - Project D
Assignment Name: AvoidObstacles

How has your program changed from planning to coding to now? Please explain?

At first, the program was planned to allow a wireless rover to detect obstacles using a distance sensor and react accordingly by backing up and turning to avoid collisions. The core functionality was simple: monitor the distance sensor continuously and adjust motor velocities to back up and turn when the rover detected an object too close.

1. Sensor Sensitivity and Distance Threshold

Problem: Initially, the program used a static threshold of 200 mm to detect obstacles. However, during testing, the rover sometimes reacted too late or ignored smaller obstacles.

Fix: I fine-tuned the threshold value based on testing conditions, adjusting it between 150 mm and 200 mm for optimal responsiveness.

```
//Before
if (distance < 200) {

//After
if (distance < 180) {
```

2. Turning Duration Adjustment

Problem: The rover's turn duration was initially set to 500 ms, but this often caused the rover to turn more than necessary, veering off course.

Fix: I reduced the turn duration slightly to 470 ms, which allowed for a smoother and more controlled turn.

```
//Before
Thread.sleep(500);

//After
Thread.sleep(470);
```

3. Ground Surface Impact

Problem: Dust and uneven surfaces affected the rover's traction, causing it to slip or fail to turn consistently.

Fix: I tested the rover on a smoother surface after cleaning it to reduce slipping. This significantly improved the rover's ability to turn accurately when avoiding obstacles.

4. Motor Speeds for Backing Up

Problem: The initial reverse motor speed (-0.5) was too slow, causing delays when backing up from obstacles.

Fix: I slightly increased the reverse speed to make the rover back up faster and avoid obstacles more efficiently.

```
//Before
leftMotors.setTargetVelocity(-0.5);
rightMotors.setTargetVelocity(-0.5);

//After
leftMotors.setTargetVelocity(-0.7);
rightMotors.setTargetVelocity(-0.7);
```

5. Infinite Loop Stability

Problem: The continuous loop sometimes caused the program to run indefinitely without proper termination options during testing.

Fix: I added a simple keyboard interrupt (optional feature) to safely stop the program during testing.

```
if (System.in.available() > 0) {
    leftMotors.setTargetVelocity(0);
    rightMotors.setTargetVelocity(0);
    break;
}
```

Now, the program works as intended. The rover uses the distance sensor to detect obstacles and reacts efficiently by backing up and turning to avoid collisions. Adjustments to the sensor threshold, turning duration, and motor speeds improved the rover's responsiveness and accuracy. Testing on smoother ground further enhanced performance by minimizing slipping.

This process taught me the importance of real-world testing, fine-tuning motor controls, and accounting for environmental conditions like surface traction when developing robotics programs.