Credit Name: CSE3910 - Project D
Assignment Name: Box

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

At first, the program was planned to control a wireless rover to move in a square path using the Phidgets library. The initial design involved creating motor objects for the left and right motors, assigning appropriate channels, and using timed movements to simulate straight-line motion and 90° turns. The program was supposed to repeat this sequence four times to form a square.Also during testing, I noticed that the dust and debris on the ground affected the rover's ability to turn accurately. The wheels would occasionally slip, causing the rover to turn too much or too little, which disrupted the square path. This happened because the dust reduced friction between the wheels and the ground, making it harder for the motors to control movement precisely.To resolve this issue, I cleaned the ground using paper and water to make the surface as smooth as possible. By increasing traction, the wheels could grip the surface more effectively, resulting in more accurate and consistent 90° turns. This small adjustment improved the overall precision of the rover's movements and ensured it followed the intended square path correctly.

1. Incorrect Motor Channels

Problem: The motor channels were incorrectly assigned, which caused the rover to move erratically or not respond at all.

Fix: I ensured the left motor was assigned to channel 0 and the right motor to channel 1, following the Phidgets documentation.

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


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

2. Timing Adjustments for Turns

Problem: The rover did not turn precisely 90°, either turning too far or too little. This was due to an incorrect delay in the Thread.sleep() method.

Fix: I fine-tuned the delay to 500 milliseconds to achieve a more accurate 90° turn.

```
//Before
leftMotors.setTargetVelocity(-1);
rightMotors.setTargetVelocity(1);
Thread.sleep(400);


//After
leftMotors.setTargetVelocity(-1);
rightMotors.setTargetVelocity(1);
Thread.sleep(500);
```

3. Stopping Motors Smoothly

Problem: The rover did not stop properly after completing turns, causing instability in movements.

Fix: I added a short pause after each turn to stop the motors and stabilize their state before proceeding.

```
//Before
leftMotors.setTargetVelocity(-1);
rightMotors.setTargetVelocity(1);
Thread.sleep(500);


//After
leftMotors.setTargetVelocity(-1);
rightMotors.setTargetVelocity(1);
Thread.sleep(500);

leftMotors.setTargetVelocity(0);
rightMotors.setTargetVelocity(0);
Thread.sleep(1000);
```

4. Repetitive Code Structure

Problem: The code had repetitive blocks for moving forward and turning, which made it harder to maintain and modify.

Fix: I implemented a for loop to reduce redundancy and make the code cleaner.

```
//Before

leftMotors.setTargetVelocity(1);
rightMotors.setTargetVelocity(1);
Thread.sleep(1500);

leftMotors.setTargetVelocity(-1);
rightMotors.setTargetVelocity(1);
Thread.sleep(500);
```

```
//After
for (int i = 0; i < 4; i++) {

    leftMotors.setTargetVelocity(1);
    rightMotors.setTargetVelocity(1);
    Thread.sleep(1500);

    leftMotors.setTargetVelocity(-1);
    rightMotors.setTargetVelocity(1);
    Thread.sleep(500);

    leftMotors.setTargetVelocity(0);
    rightMotors.setTargetVelocity(0);
    Thread.sleep(1000);
}
```

Now, the program works as intended. The rover moves forward and performs precise 90° turns to complete a square path. Adjustments such as channel corrections, fine-tuned delays, and smoother motor stops improved the program's accuracy and stability. Implementing a loop also made the code more concise and easier to maintain. These changes ensured the program met its objective effectively.