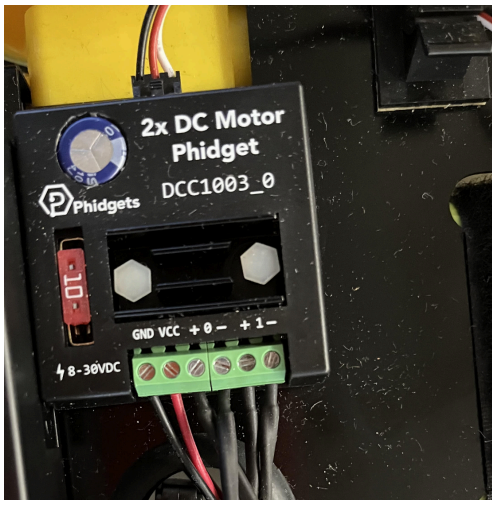
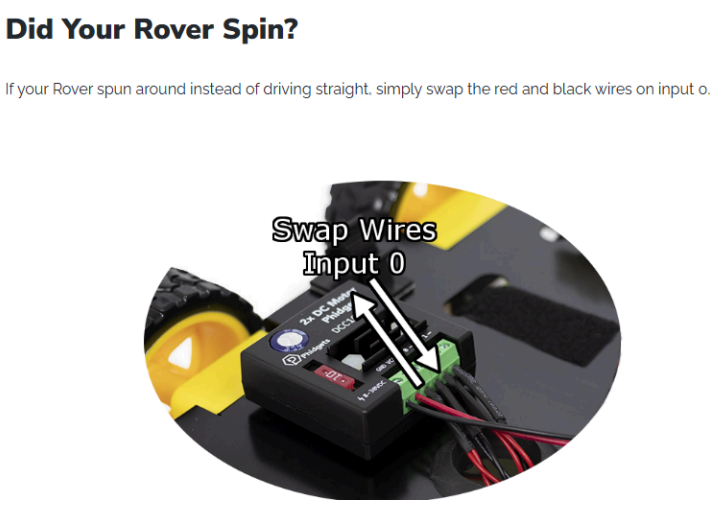
Credit Name: CSE3910 - Project D
Assignment Name: Phidgets

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

The first issue I faced when I started Phidgets was that my rover was wired incorrectly and that the wires were loose.





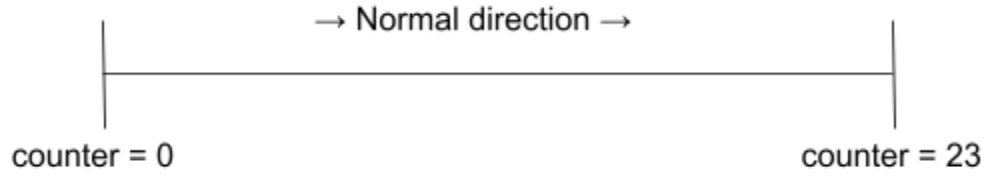
The above image is the correct wiring but originally, the red and black wires were switched on input “0”. When I tightened the screws and ran the code the rover would spin around. I fixed this by swapping the wires. The rest of the rover was already set up correctly.

The lessons on moving linearly, turning, and avoiding obstacles were straightforward. For all of them I copied the given code and modified it to see what would happen. The hardest part of Phidgets was the Square challenge which emphasised how the rover only works the way you need it to work sometimes.

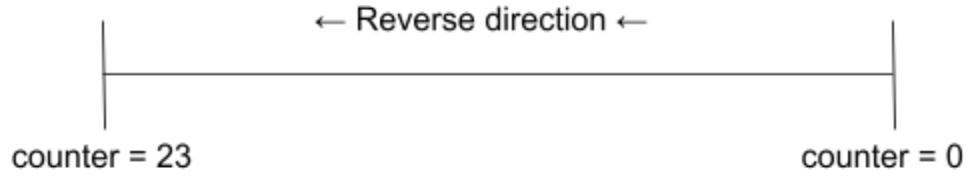
Originally, I had planned to just use a while loop and some conditionals. This would’ve been easy to code if the challenge was to just go in a square shape. Since we needed to turn 180 degrees and follow the square in the reverse direction if an object was in the way, my plan would’ve become a bunch of nested conditional statements within the while loop. This would’ve been harder to debug and read the code. Instead, I opted to use methods to make a much simpler while loop, this also made it a lot easier to debug.

```
 92⊕     private static boolean normDirection(DCMotor lM, DCMotor rM, DistanceSensor s) throws Exception {
128
129⊕     private static boolean revDirection(DCMotor lM, DCMotor rM, DistanceSensor s) throws Exception {
162
163⊕     private static void turnAround(DCMotor lM, DCMotor rM) throws Exception {
194
195⊕     private static void oneSecStop(DCMotor lM, DCMotor rM) throws Exception{
203
204 }
```

Most of my methods worked and were easy to make. I used a counter variable to keep track of the distance travelled so that it could follow the same square outline even after it spun around.



If the rover detected an object at counter = 10, then it'd spin around and the counter would be = 23 - counter(10).




```
private static void turnAround(DCMc
{

    counter = 23 - counter;

    lM.setTargetVelocity(-1);
    rM.setTargetVelocity(1);

    //Thread.sleep(1200); 1200 only
    Thread.sleep(710);

    turnCounter ++;
```

I used a turn counter to keep track of which direction to go after the rover turned around. Initially, the variables counter and turnCounter weren't static variables used by the whole class but rather local to the main method. I changed this because then it would mean I'd need to make a counter variable local to each required method and I'd also need to return the counter each time the method ran then set the main method counter equal to the counter from the other methods. Since this was too much complicated work I just made the variables static.

```
public class Square {

    public static int counter = 0;
    public static int turnCounter = 0;

    public static void main(String[] args) throws Exception

    private static boolean normDirection(DCMotor lM, DCMotor rM, DistanceSensor s) throws Exception

    private static boolean revDirection(DCMotor lM, DCMotor rM, DistanceSensor s) throws Exception

    private static void turnAround(DCMotor lM, DCMotor rM) throws Exception

    private static void oneSecStop(DCMotor lM, DCMotor rM) throws Exception

}
```

Another piece of my code that changed was the format of the main method's while loop and the turnAround method. The turnAround method used to only run if the sonar detected a distance

less than 200 mm (Shown below in the comments). I did this because the normal and reverse direction method (the method for going one side of a square followed by a 90 degree turn) would stop if the distance detected by the sonar was less than 200mm. The idea was that when the rover stopped moving, the code for it to turn around would execute, getting rid of the need for more conditional statements. I changed this since the rover would take a while to turn around when it detected an object. It would remain stationary for a while before moving. Ultimately, this change did nothing to my code since the rover always took a while to process multiple lines of code that involved the motors and sonar sensor. I didn't change it back because I felt there was no point in doing so.

```
78      private static boolean normDirection(DCMotor lM, DCMotor rM, DistanceSensor s) throws Exception
79      {
80          boolean completion = false;
81
82          while(counter != 23) // abt a meter
83          {
84              lM.setTargetVelocity(0.45);
85              rM.setTargetVelocity(0.45);
86              counter++;
87              Thread.sleep(100);
88
89              if (s.getDistance() < 200)
90              {
91                  break;
92              }
93          }
94
95          if(counter == 23)
96          {
97              //Turn in one direction
98              lM.setTargetVelocity(-0.9);
99              rM.setTargetVelocity(0.8);
100
101             Thread.sleep(600); // 90 degree turn roughly // was 810 before
102             counter = 0;
103             completion = true;
104
105         }
106
107         lM.setTargetVelocity(0);
108         rM.setTargetVelocity(0);
109         Thread.sleep(1000);
110
111         return completion;
112
113     }
114
```

```
private static void turnAround(DCMotor lM, DCMotor rM) throws Exception
{

    counter = 23 - counter;

    lM.setTargetVelocity(-1);
    rM.setTargetVelocity(1);

    //Thread.sleep(1200); 1200 only good if stop and rotate
    Thread.sleep(710);

    turnCounter ++;

    /*
    if (s.getDistance() < 200)
    {
        counter = 23 - counter;

        lM.setTargetVelocity(-1);
        rM.setTargetVelocity(1);

        Thread.sleep(1200);

        turnCounter ++;
    }
    else
    {
        return;
    }
    */
}
```

Another portion of my methods that I changed was setting the normal and reverse direction methods to return boolean values based on the status of their completion of the going the full side length of the square. I planned to use this in a conditional statement to signal for the turnAround() method to run. In the end I didn't end up doing this since it was already working the way it was and I didn't want to mess it up.

```
private static boolean revDirection(DCMotor lM, DCMotor rM, DistanceSensor s) throws Exception
{
    boolean completion = false;

    while(counter != 23) // About a metre
    {
        lM.setTargetVelocity(0.45);
        rM.setTargetVelocity(0.45);
        counter++;
        Thread.sleep(100);

        if (s.getDistance() < 200)
        {
            break;
        }
    }

    if(counter == 23)
    {
        //Turn in one direction
        lM.setTargetVelocity(0.8);
        rM.setTargetVelocity(-0.9);

        Thread.sleep(600); // Roughly a 90 degree turn

        counter = 0;
        completion = true;

    }

    return completion;

}
```

The biggest challenge I faced was not in the coding section of square but rather the values to use for Thread.sleep() and values for setTargetVelocity(). This was because depending on the power left in the battery, the motors were sometimes faster or slower and were often not spinning at the same rate. The wheels of the rover were wobbly and both the wheels and floor had to be visibly clean in order to have traction. Because of this, only sometimes I'd get a perfect 90 degree turn but sometimes it would be off by 10-20 degrees, most of the time it was consistently between 85 - 95 degrees, at most a difference of 10 degrees. Another annoying issue I faced was that 180 degree turns would work with certain values stationary but when I used the same motor speed and thread.sleep values after it had moved and detected an object, the rover would often over rotate by 90 degrees. This could've been due to the wheels previously accelerating which meant that the wheels would spin faster. Either way, I had to change thread.sleep values a lot.

```
private static void turnAround(DCMotor lM, DCMotor rM) throws E
{

    counter = 23 - counter;

    lM.setTargetVelocity(-1);
    rM.setTargetVelocity(1);

    //Thread.sleep(1200); 1200 only good if stop and rotate
    Thread.sleep(710);
```

```
{
    //Turn in one direction
    lM.setTargetVelocity(0.8);
    rM.setTargetVelocity(-0.9);

    Thread.sleep(600); // 90 degree turn roughly // was 810 before

    counter = 0;
    completion = true;

}

return completion;
```