**Observations:**

**Surfaces Tested:**

* Hardwood floor
* Uneven Carpet
* Roof Deck
* Parking Garage

**Straight Line Driving:**

Driving in a straight line was challenging for the MBOT on all the surfaces for both the time and encoder scripts. Both motors were given a 25% duty cycle. This means the pulse to the motors remains high for 25% of the time and low for 75% of the time. The two motors spun at different rates when given the same Pulse Width Modulation (PWM) signal. This caused the MBOT to track off course in the direction of the motor that was traveling more slowly. This flaw is primarily due to manufacturing differences between the motor as no two motors are identical. Both the time and encoder driving scripts tracked off course equally (roughly a 10-15 degrees from the initial angle). The encoder values did not control the PWM signal sent to the motors. All the algorithm did was check to see if MBOT had traveled one meter based on the average distance covered by the encoders.

Determining the distance traveled based on the encoders was robust to varied surface resistance. The timed script greatly varied based on the surface resistance. It required manual manipulation of the time to travel forward to get the MBOT to drive forward roughly one meter.

**Turning:**

When driving the MBOT, on multiple surfaces the time it took to turn greatly varied. This is because the resistance on the wheels changed. This caused the timed script to require adjustment depending on the surface. The carpet and parking garage exhibited the highest resistance with the tile and hardwood flooring having the least resistance. The time script turns were often greater than 90 degrees to compensate for the MBOT tracking off course. The encoder script turns were very accurate and often resulted in a close to 90-degree turn. Although the MBOT had already tracked off of its given route

**Improvements to be made:**

Overall the encoder script did a much better job at ensuring the robot traveled one meter and turned 90 degrees. I believe the encoders can be used to update measurements reporting on the distance the robot has traveled. This method is called odometry and is used to update a robot’s position over increments of time. I would also use Proportional Integral Derivative (PID) controller to regulate the motor’s duty cycle based on a target duty cycle and keeping the encoder difference to a minimum (while driving in a straight line). To increase accuracy, I would also fuse the Inertial Measurement Unit (IMU) data to reconstruct the MBOT’s position. Additionally, I would use the LIDAR unit to map the surroundings and keep track of the MBOTs location with a method such as Simultaneous Localization and Mapping.