Testing

RPLidar A1M8 sensor

The RPLidar sensor's measurement was tested directly on the Raspberry Pi 5.

Before creating the nodes, I ensured the accuracy of the sensor measurements which were acquired serially in a script that processed the data by converting the hexadecimal values into decimal values, separated the distance and angle bytes and calculated the x and y points. The points were later added into a Point Cloud Data (.pcd) file and visualized in Rviz.

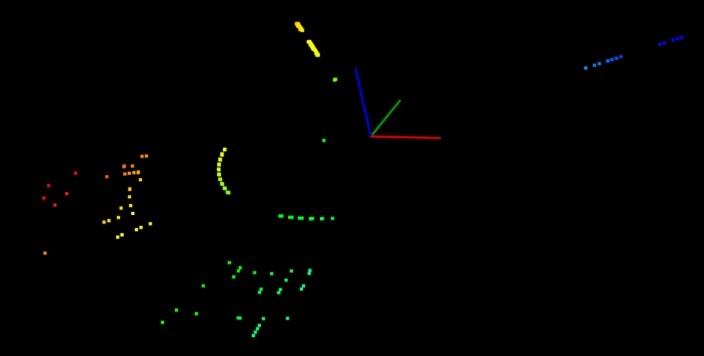


Figure 1-First Point Cloud obtained from the Lidar measurement

I validated the measurement data by verifying whether the resulting point cloud mirrored the reality, which it did, meaning that the Lidar measurement is accurate.

MPU6050 Inertial Measurement Unit

I tested the IMU measurements directly in the ROS node by logging them and visually verifying whether the data is correctly modified in real time, according to the acceleration or rotation that the car's movement made.

In order to test the quality of the calibrated measurements, I plotted the calibrated and raw data using a Python script along the three combinations of axes, as well as 3D, and visually compared the two values. For demonstration purposes, I plotted only a quarter of the values acquired using the IMU. It can be seen that the calibrated values are closer to the value of 1 along the axes representations, meaning that the calibration was successful.

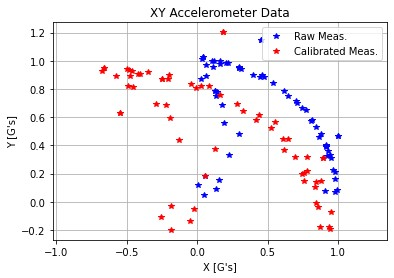


Figure 2-The raw and calibrated values plotted along the X and Y axes

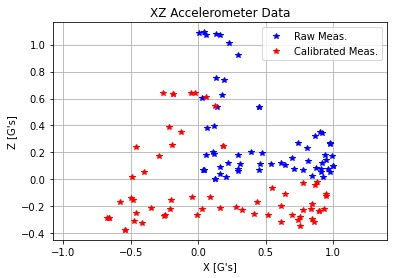


Figure 3-The raw and calibrated values plotted along the X and Z axes

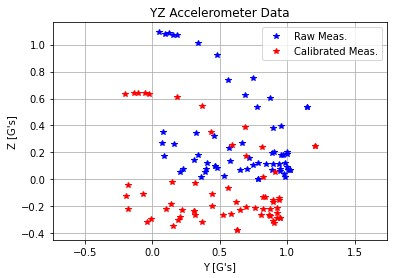


Figure 4-The raw and calibrated values plotted along the Y and Z axes

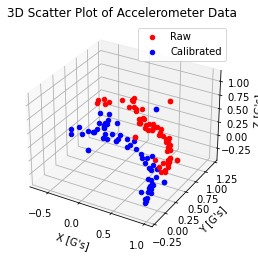


Figure 5-The raw and calibrated values plotted along the Y and Z axes

The metrics resulted from the calibration can be seen in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **X axis** | **Y axis** | **Z axis** |
| Mean Difference (Bias) | 0.3330 | 0.1280 | 0.3581 |
| Standard Deviation of Differences | 0.2079 | 0.0726 | 0.0328 |
| RMSE | 0.3917 | 0.1468 | 0.3595 |
| Maximum Difference | 0.7073 | 0.2678 | 0.4553 |
| Minimum Difference | 0.0295 | -0.0582 | 0.2889 |
| Mean Absolute Error | 0.3330 | 0.1317 | 0.3581 |
| Percentage Improvement | -10.55% | 54.50% | -12.56% |
| Correlation Coefficient | 0.9695 | 0.9979 | 0.9967 |

Wheel Encoder

I tested the functionality of the encoder in the ROS node by logging the measured speed and checking whether its modification mirrors the movement of the robot (meaning a lower speed when the robot moves slowly, faster one when it moves faster and a speed of 0 when there is no motion).

Another method with which the wheel encoder was tested was using an oscilloscope, checking the captured pulses. It can be seen on the oscilloscope capture that when the signal is low for a longer period, the robot is stationary and when there are multiple consecutive pulses, the robot is in motion.

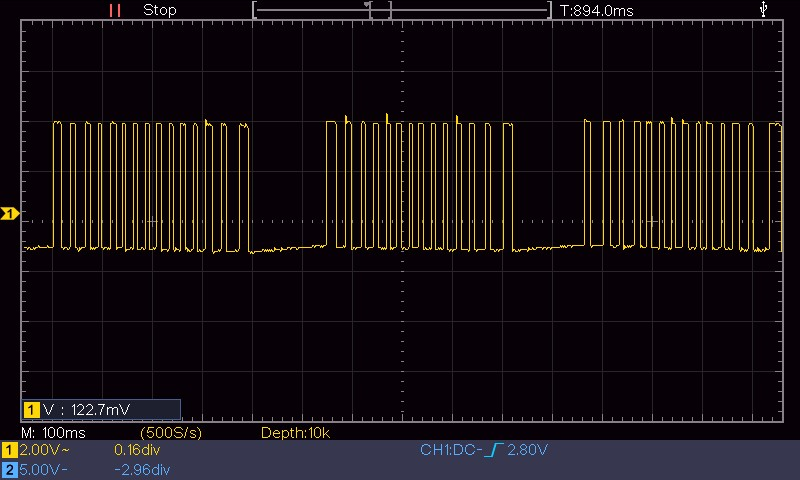


Figure 6-The modification of pulses captured the oscilloscope during the testing of the whee encoderl

FlySky FS-i6 Radio Controller

In order to test the functionality of the radio controller and the motor control module, I used both visual cues and an oscilloscope.

I checked the period of the PWM signal on the oscilloscope, ensuring that in the [1ms, 1.5ms) range the robot movement is backwards or the servo motor steers left, depending on the connected channel, in the (1.5ms, 2ms] the robot movement is forward or the servo motor steers right, and at the period of 1.5ms the robot stays neutral.

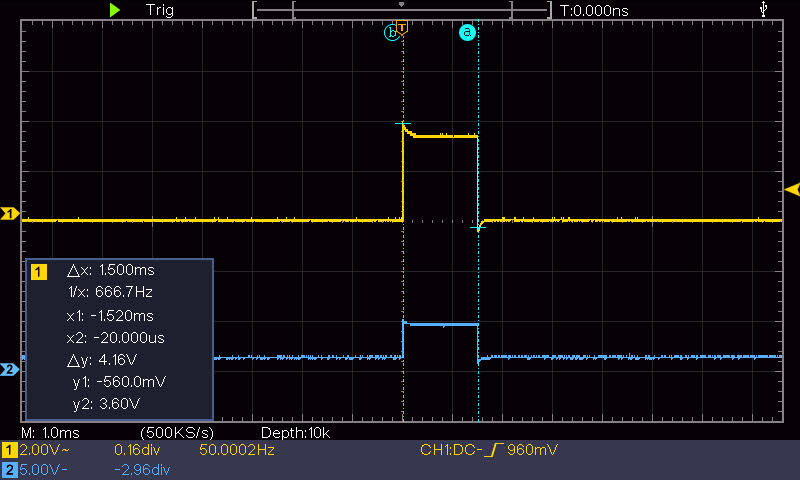


Figure 7-The period captured the oscilloscope when the robot is in neutral state

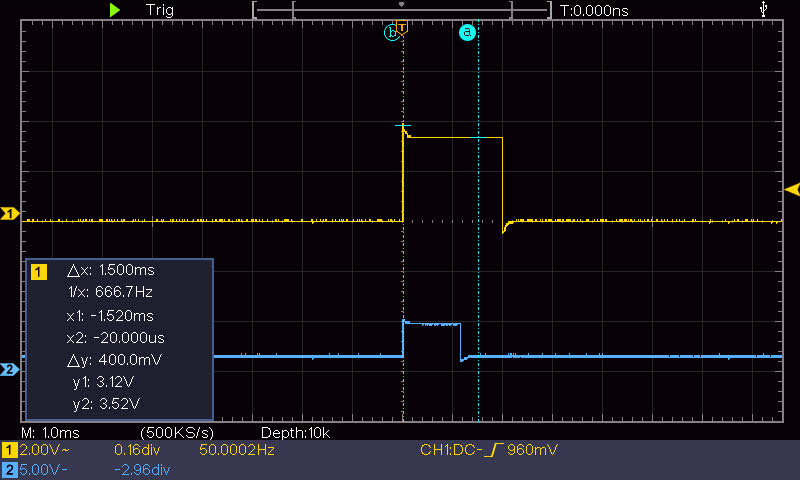


Figure 8-The period captured the oscilloscope when the robot moving forward and is steering left