College of Engineering

Robotic Walker

SAN DIEGO STATE UNIVERSITY

Introduction

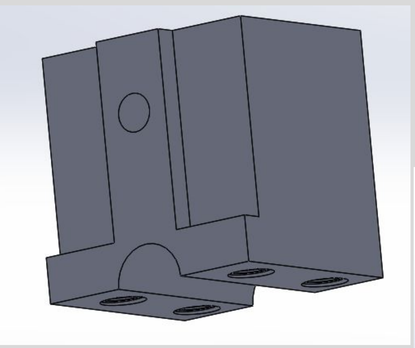
Presented with an existing design of a 6-wheel walker, the task assigned to the team was to transition the predecessor design into a 2-wheel design while implementing a self-balancing system. The design was made to consider the elderly consumers of the market. The design will provide stability for the user when walking from Point A to Point B. If the user leans to the point where they might fall, the internal code will recognize it and balance itself out to prevent the fall from happening. Our team built the foundation of this project with hope that it will be further advanced by future senior design groups.

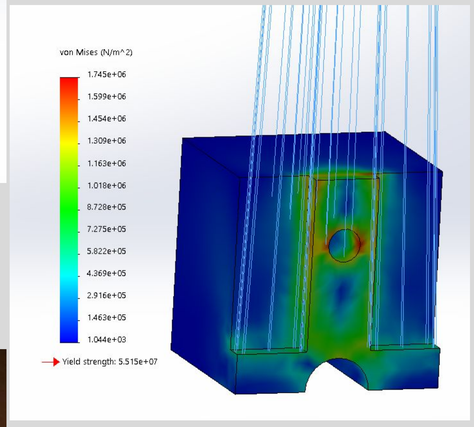
Objective

To design a robotic walker that is in the format of a two-wheel segway with internal self-balancing mechanism.

Design

Due to COVID-19, the design was mostly bought pre-made as suggested by our sponsor. There was one part that was manufactured which was the custom wheel mount for the side of our dolly. The reason for doing this was to mount the hoverboard wheels onto the outside of the dolly rather than inside based on the wheel axles. The top storage compartment was considered for users to put any personal belongings inside it. The bottom storage compartment contained all of the electrical components.





IMU/PID Test

A BNO055 Inertial Measurement Unit (IMU) was used. The tests performed were to understand how the accelerometer and gyroscope behaved based on the tilt angle using its raw data. Everything was done on the Elegoo Mega microcontroller which was coded using Arduino. The graphs are measured angles with respect to time.

The data from the accelerometer had no offset, great response time, but was susceptible to a lot of noise. The data from the gyroscope also showed great response time. In comparison to the accelerometer, it had no noise, but had an offset. From this, the complementary filter would be used to combine the two data results in order to find a tilt angle that would have a great response, no offset, and no noise. Then, the complementar filter would be compared when:

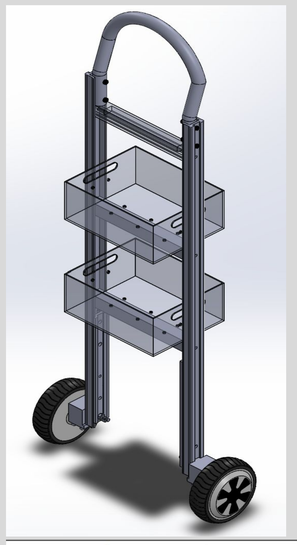
CF = 0.95 vs 0.05

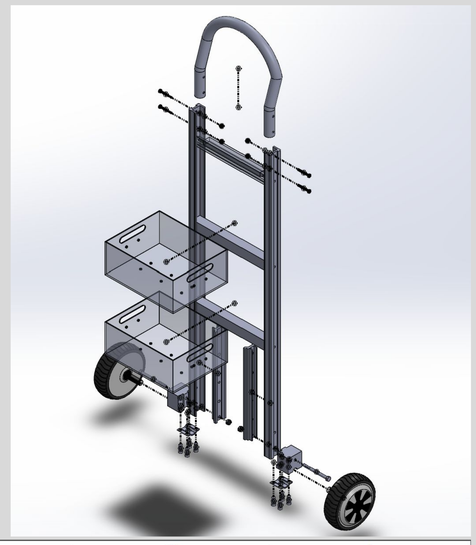
The CF variable is a percentage number based on how much the gyroscope is being used in the following line of code::

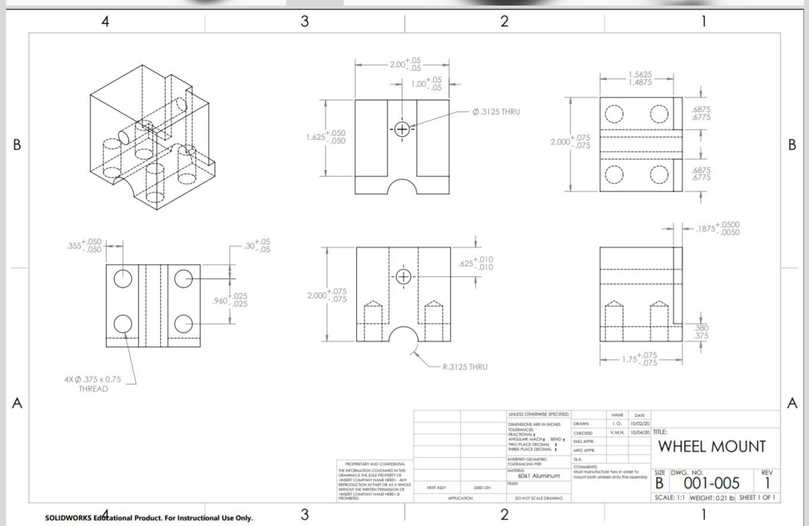
Tilt = (tilt + gyr.y()*dt)*CF + accAnglePitch*(1-CF);

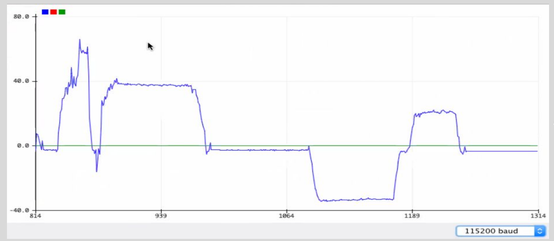
PID will be calculated using the following line of code:

pid = Kp*tiltError + Ki*tiltErrorArea + Kd*tiltErrorSlope;

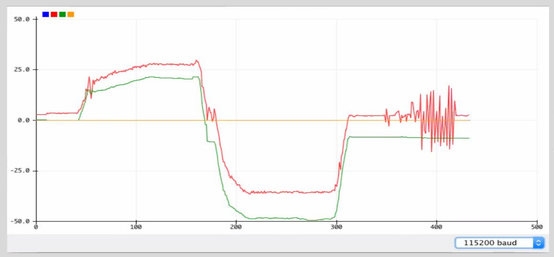




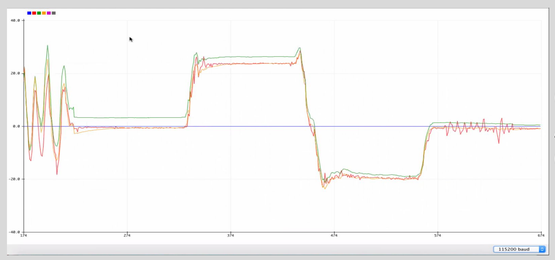




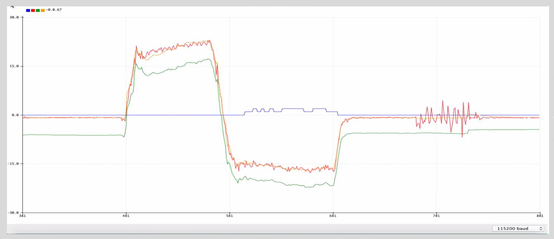
Accel only



Gyro (green)
Accel (red)



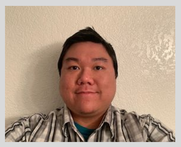
Gyro (green)
Accel (red)
CF @ 0.95 (orange)



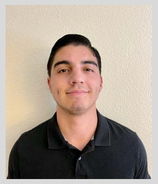
Gyro (green)
Accel (red)
CF @ 0.5 (orange)

Meet the Team

Minh Hoang
(Team/Code Leader)



Isaiah Oropeza
(Design Leader)



Sam Kearns
(Code Team)

Victor Hernandez
(Design Team)

Acknowledgement

Sponor/Advisor: Dr. Kee Moon
San Diego State University, College of Engineering

ME490 Fall 2020