Capstone Project

Assistive Self-Stabilizing Robot

Omar Escareno, Khem Holden, Andrew Lei, Jeffrey Ng, Jornell Quiambao, Diego Reyes

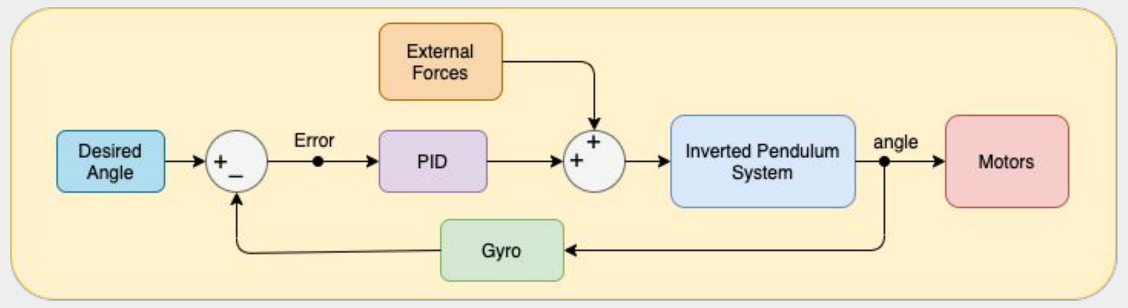
Baskin Engineering UC Santa Cruz

Objective

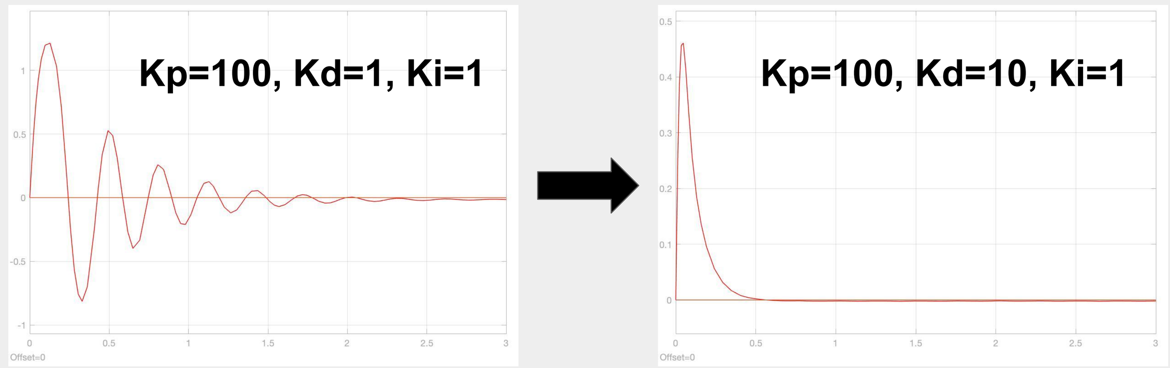
Elderly and disabled persons have difficulty carrying personal items. To alleviate this issue, our goal is to design an assistive robot with autonomous tracking and self-stabilizing capabilities. The design features include:

- Two-wheeled rover for easy mobility with a turn radius of 0 degrees
- Self-balancing implemented using a PID controller
- Tracking the user using computer vision with a Jetson Nano
- Raspberry Pi to control the main state machine for robot movement and to communicate with sensors through GPIO pins
- Simulation of the design provided by ROS and MATLAB

Self-Balancing Controller



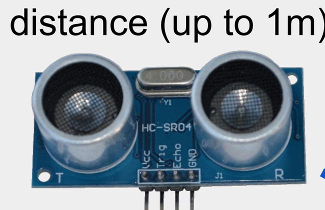
- A PID controller is used to force the system into a desired angle offset of 0 degrees
- We want to tune the controller to reduce overshoot and settling time
- MATLAB plots below show the step response of a single push to offset the angle of a simulated system using different PID control values

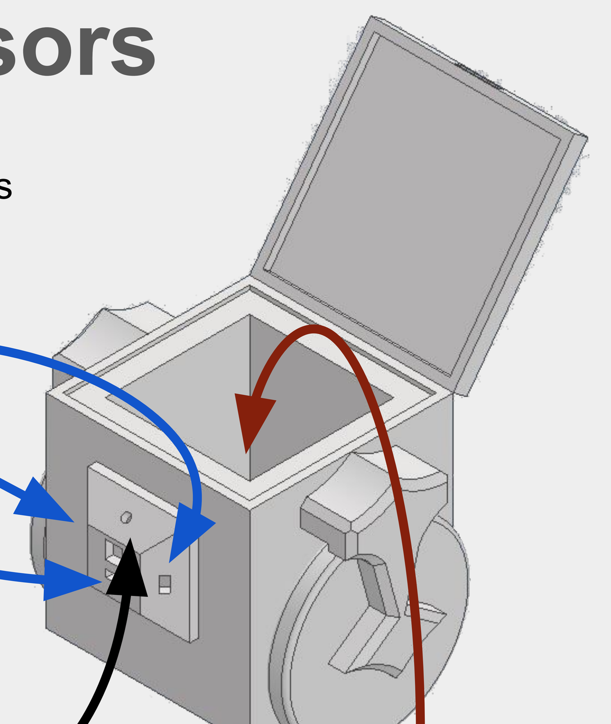


Sensors

HC-SR04 Ultrasonic Sensors

- Object Detection for turning corners
- Proximity Detection to maintain following distance (up to 1m)



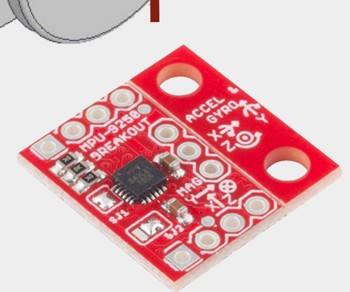


Logitech C270 Webcam

- 720p/30 FPS camera
- Used for computer vision

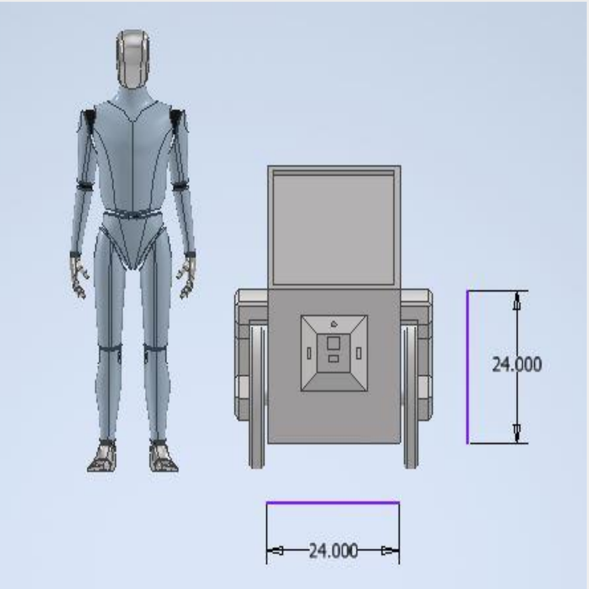
MPU-9250 IMU

- Gyro sensor for detecting angle offset for self balancing a desired angle offset of 0 degrees
- Located in the center axis base of the robot

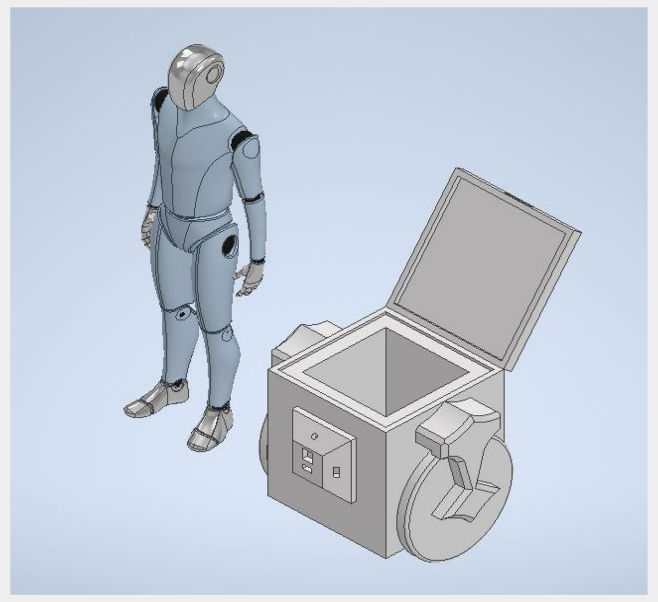


Robot Model and Simulation

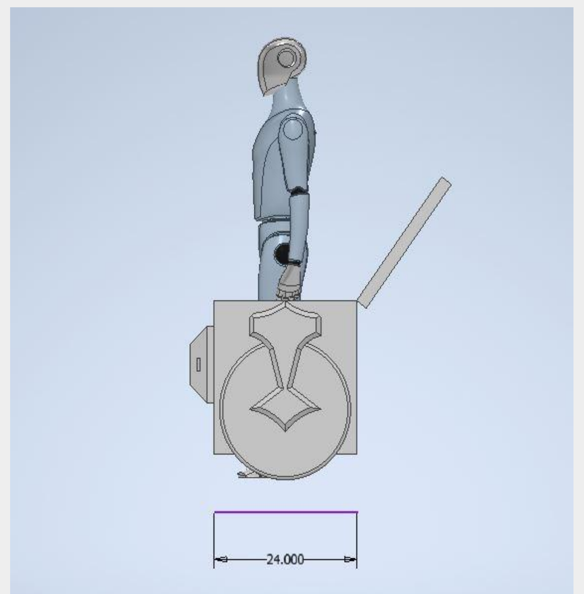
Front view



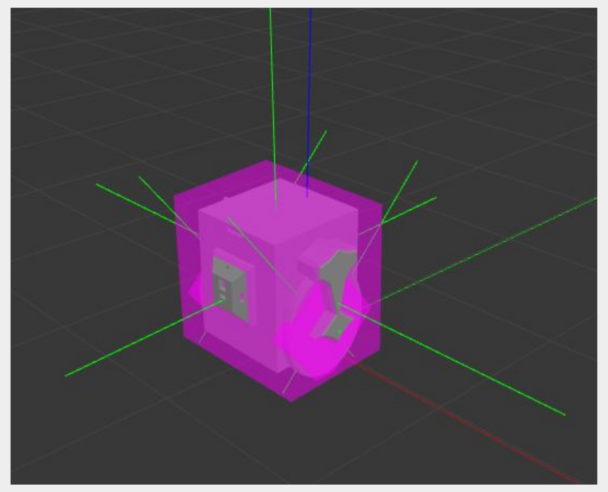
Isometric view



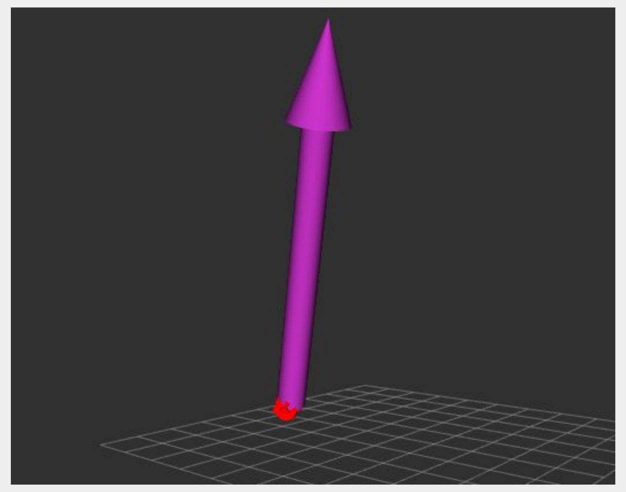
Side view



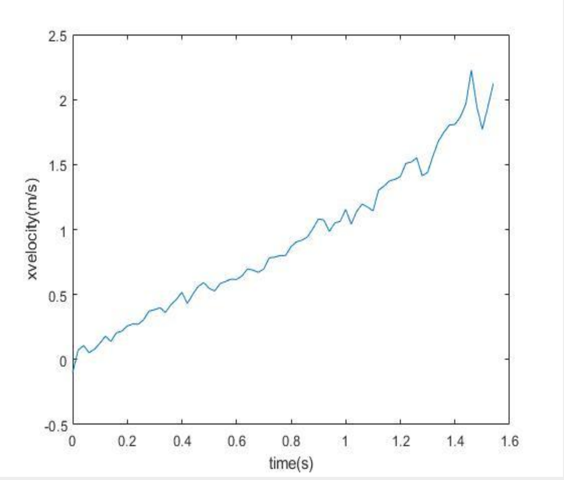
Robot frame dimensions are 24in x 24in x 24in and is shown next to scaled human model standing at 5 feet



Gazebo Visualization
Gazebo allows for visualization of the robot performance as well as the introduction of noise to sensors and wind to the environment

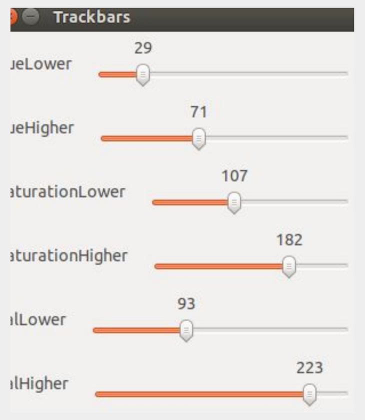


Sensor Visualization
Using Rviz, the vectors created by sensors, like the accelerometer, can be viewed in order to ensure proper simulation

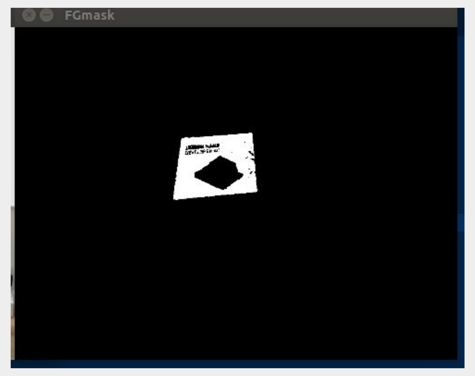


Linear Velocity
Using ROS, plotting the linear velocity of the motors show the robot's attempt at self-balancing; the current iteration of the PID controller in the Gazebo simulation has a linear velocity drift

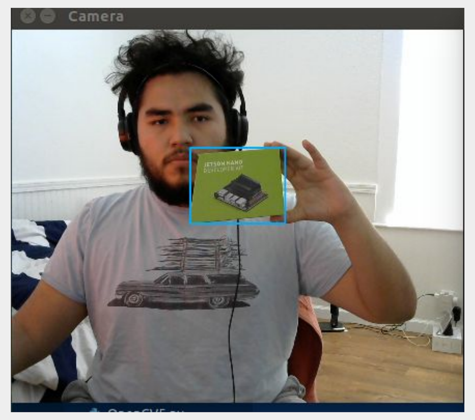
Computer Vision



Track Bars
Using a configurable color mask with multiple aspects, we can find a specific color to track

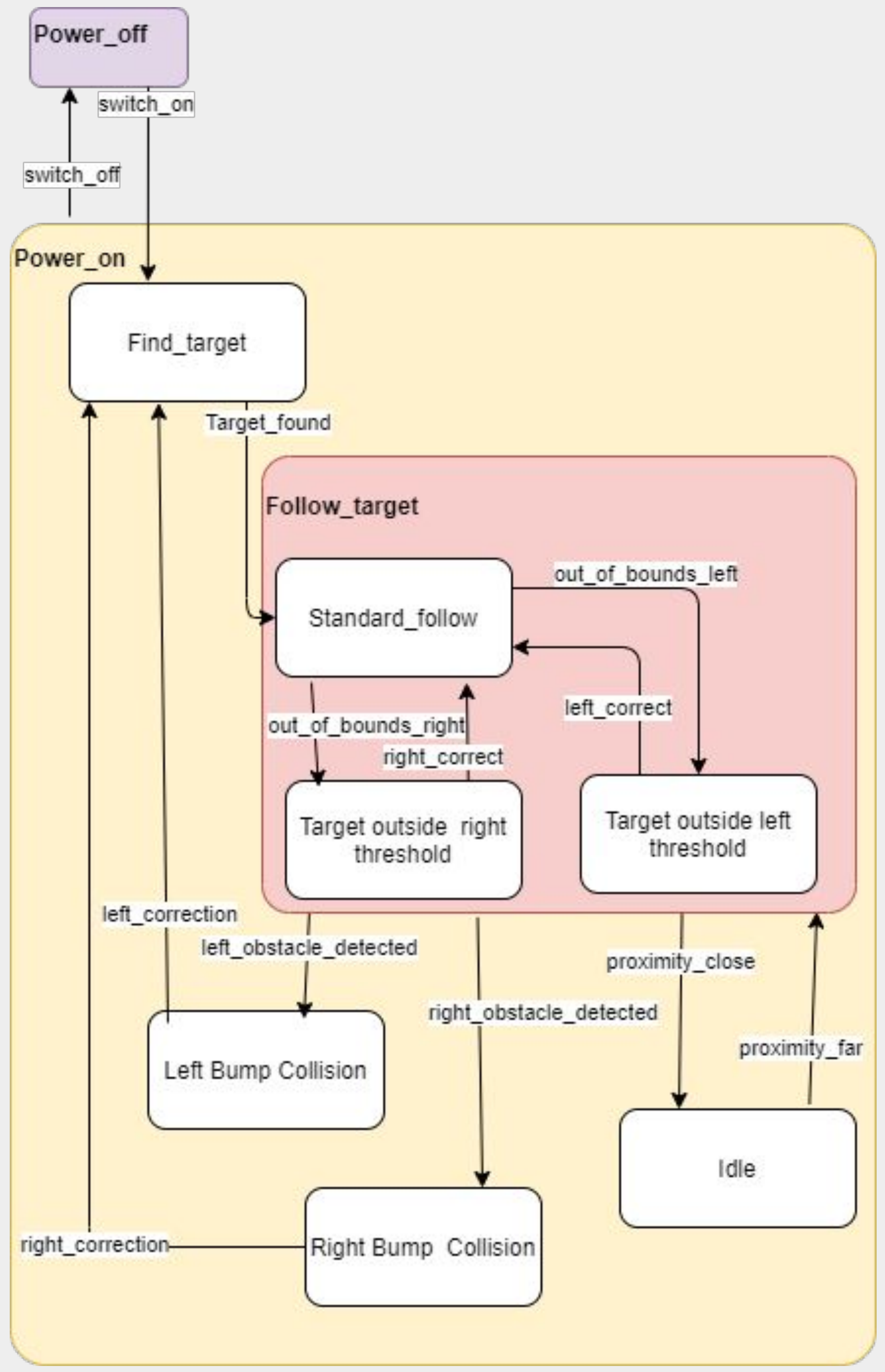


Foreground Mask
Once the specific color is found, OpenCV can create a mask on the frame to highlight what lies within the mask



Region of Interest
After creating the mask on the frame, the highlighted object can be represented with a ROI box

State Machine



- Hierarchical state diagram outlining AI of the robot
- State machine controls the general movement of the robot–tracking and collision maneuvering
- This program operates in parallel to self-balancing and computer vision

Conclusion

The design concept of the assistive self-stabilizing robot is made possible through the implementation of multiple key factors and accomplishments:

- Computer vision tracking for objects of interest with 99.5% accuracy
- Self-balancing PID controller that stabilizes the system with a steady state error of 1% and an overshoot < 0.5 degrees
- CAD design that visualizes the robot's dimensions, its cargo space/weight and its sensor placements
- Fully-functioning hierarchical state machine implemented using an Events and Services Framework
- Fully-functioning simulation of self-balancing controller of the physical system in MATLAB and ROS with the CAD design