# Slope Stabilizing Robot

**C7: Sara Chung and Raymond Shen**
18-500 Capstone Design, Spring 2025
Electrical and Computer Engineering Department
Carnegie Mellon University

## Product Pitch

The Slope Stable Robot is a self-balancing robot to carry water without spilling, even while moving up slopes. Using real-time analysis powered by an Arduino/FPGA, the robot dynamically adjusts its platform through linear actuators to maintain a level surface.

Our critical requirements for the robot can climb up 30º slopes and be able to carry a 90% (~14 oz) filled 16 oz cup without spilling, and has adjustable speed to get up slopes and to prevent spilling. During testing, the robot was able to go up 15º slopes with 90% (~14oz) filled cup of water.

This improves the quality of deliveries and reduces the risk of injuries for chemical transports. These might who might otherwise struggle with heavy or unstable loads on inclined surfaces. Our design aims to make chemical transport easier and safer for workers.

## System Architecture

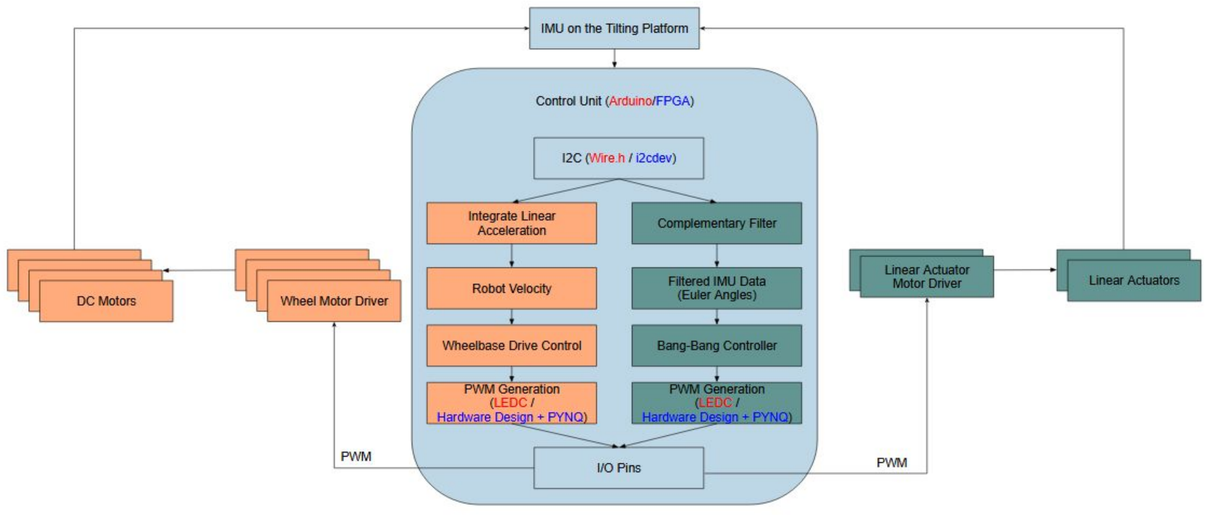



**FIG 1: System Architecture**

We started implementing the system with an Ultra96 FPGA, successfully driving the motors with the PWM signals and reading from the IMU, but throughout the process there were many setbacks, so we ultimately transitioned to using an Arduino Nano ESP32 to complete the whole system.

We control our entire system with the data from the IMU, using the roll angle detected as well as the linear velocity. For the Arduino, we interface with the IMU using the Wire.h library. This library allows easy IMU communication through simplified I2C functions that abstract away low-level protocol details, enabling quick register access with minimal code. We leveraged the MPU6050 library which handles all data filtering operations, implementing a complementary filter that combines gyroscope and accelerometer data to provide stable orientation estimates. In our initial FPGA implementation using PYNQ, we utilized the I2Cdev library to access the IMU registers, and implemented the complementary filter in the Python layer of PYNQ rather than in hardware. This approach maintained software flexibility while leveraging the FPGA for I/O operations.

For motor control, we generated PWM signals using the ESP32's LEDC peripheral on the Arduino side, which provided precise timing and multiple channels. In our FPGA implementation, we designed custom PWM modules in the Zynq Ultrascale+ hardware using the low-speed GPIO on the Ultra96 board. By integrating this hardware design into PYNQ, we could dynamically control PWM parameters through Python while maintaining hardware-level timing precision. Both approaches successfully interfaced with BTS7960 motor drivers to control our 24V DC motors and 12V linear actuators, with the Arduino solution ultimately proving more practical for rapid iteration and system completion.

## Conclusions & Additional Information

https://course.ece.cmu.edu/~ece500/projects/s25-teamc7/

Our overall assessment is that we accomplished a platform and tilting system and adjustable speed, that is able to meet the goal of not spilling water while going up a ramp. Our system is just too slow and is not a viable solution to the use case due to its speed.

We identified one improvement is to avoid a top-heavy design by using a motor instead of a linear actuators. The linear actuators are too long, so it will fall backwards on steeper slopes. The solution would be to use a motor to adjust the angle of the platform. This would also allow the platform to adjust faster since the speed of our linear actuators is slow. With the faster adjustment, a PID controller would be optimal for the best accuracy. Additional sensors could also be used to sense the ramp, and adjust before hitting the ramp to reduce spills.

Electrical & Computer ENGINEERING
Carnegie Mellon

## System Description

### Drive Control

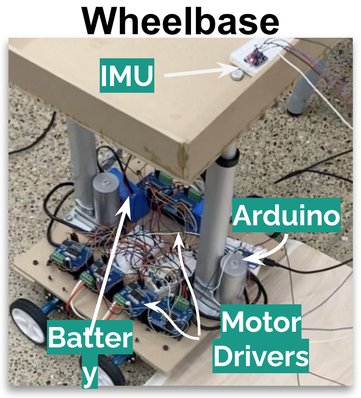
The robot's drive control system detects when the front wheel is stopped by a ramp. Upon detection, the system incrementally increases the motor duty cycle until the robot begins moving forward again. As the robot continues, it reduces the duty cycle until the rear wheel contacts the ramp, at which point it increments the duty cycle once more. This method ensures smooth traversal of inclines while allowing the platform control system to adjust dynamically to changes in the slope angle.

### Platform Control

Platform stabilization is managed using a bang-bang feedback controller based on the IMU's angle readings. When the platform tilts beyond a ±0.5º deadband, both linear actuators activate to correct and re-stabilize the platform.

**FIG. 2: Platform & Linear Actuator**


**FIG. 3: Wheelbase**




## System Evaluation

### Unit Tests

The two key components of our system that required unit tests were the PWM signal generation and the IMU data collection. This was especially important to realize the capabilities of the Ultra96 FPGA, since we have no previous embedded experience with this device. At the same time, we also began development with the Arduino as a backup, so we tested both systems at the same time.

### Control System Tests

We conducted many runs, iterating on our control system. Our first control system was contant wheel speed, while adjusting the platform based on the IMU. Our second control system was a variable wheel speed, even stopping, while adjusting the platform based on the IMU. Our final control system is a variable wheel speed, that stops just before the ramp, and then accelerates again. We also filter our some outliers in the IMU data, to help with consistency of adjusting our platform. We did this because the water was spilling when hitting the ramp and there would be outliers in IMU data that would confuse our control system.

**FIG. 4: IMU Data on Random Test Run**

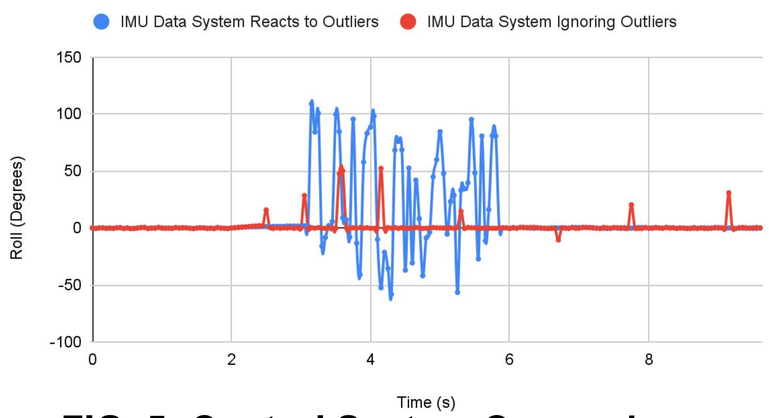



**FIG. 5: Control System Comparisons**

| Condition | Success Rate (%) | Spill at Startup | Spill at Ramp | Spill on Ramp |
|---|---|---|---|---|
| **Constant Wheel Speed** | 40 | Yes | Yes | Yes |
| **Variable Wheel Speed** | 50 | No | Yes | No |
| **Stop at Ramp (filter IMU Outliers)** | 90 | No | No | No |

### Use-Case Requirements:

| Metric | Target | Actual |
|---|---|---|
| Accommodate Varying Slope | Drive up ramps from 0° to 30° | Can only drive from 0° to 15° |
| Accommodate Varying Loads | Place 1-4 cups of water on the platform without spilling | No spillage for 16 oz cups with 11 oz (~70%) of water and driving up ramp |
| No spillage | Fill 16 oz cups with 14 oz (~90%) of water and drive up ramp | No spillage for 16 oz cups with 11 oz (~70%) of water and driving up ramp |