Learning to Balance

A Reaction-Wheel Unicycle Robot

Project #2411 **Sponsor: Project Lab**

Kyle Mackenzie, Simon Ghyselincks, Tristan Lee, Jackson Fraser, Julian Lapenna

Background and Motivation

Problem Statement

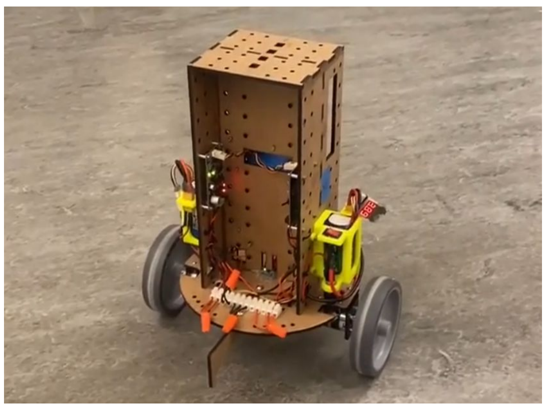
Robotics projects often have complex dynamics with properties that are difficult to model.

Classical controllers often rely on simplifications of true physical dynamics to make real-time decision-making feasible, which can limit the realm of possible controls strategies.

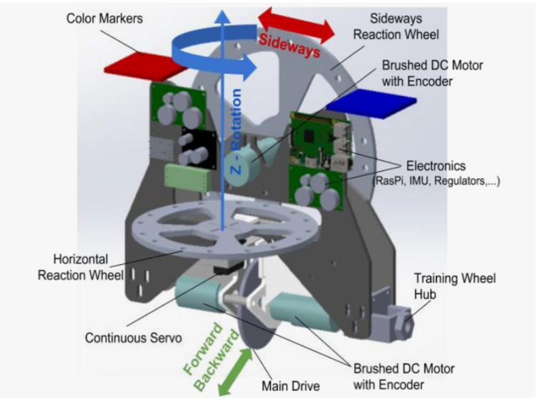
Reinforcement learning (RL) has the potential to unlock the full complexity of the system and fully explore the system dynamics, compared to a traditional model with approximations.

Main Goal

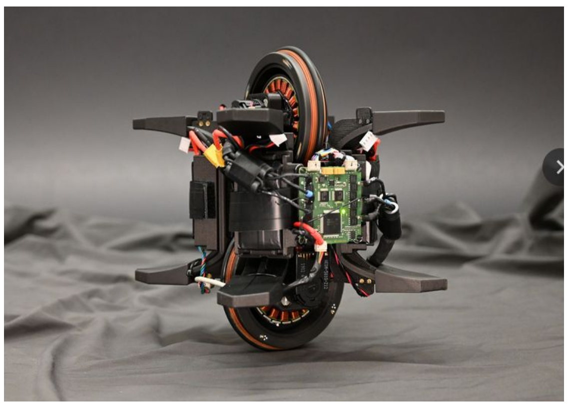
The goal of our project is to approximate the dynamics of our system, model them, and design a controller using traditional controls methods. We'll then compare that with an RL controller and see how it stacks up.



TWIP - Single-Axis Reinforcement Learning
Team 2153, 2022



AIUR - 3-axis Classical Controls
Team 1868, 2019



Wheelbot - 2-actuator, Classical Controls
Max Planck Institute

System Dynamics

Orthogonal Axes Reduce Coupling

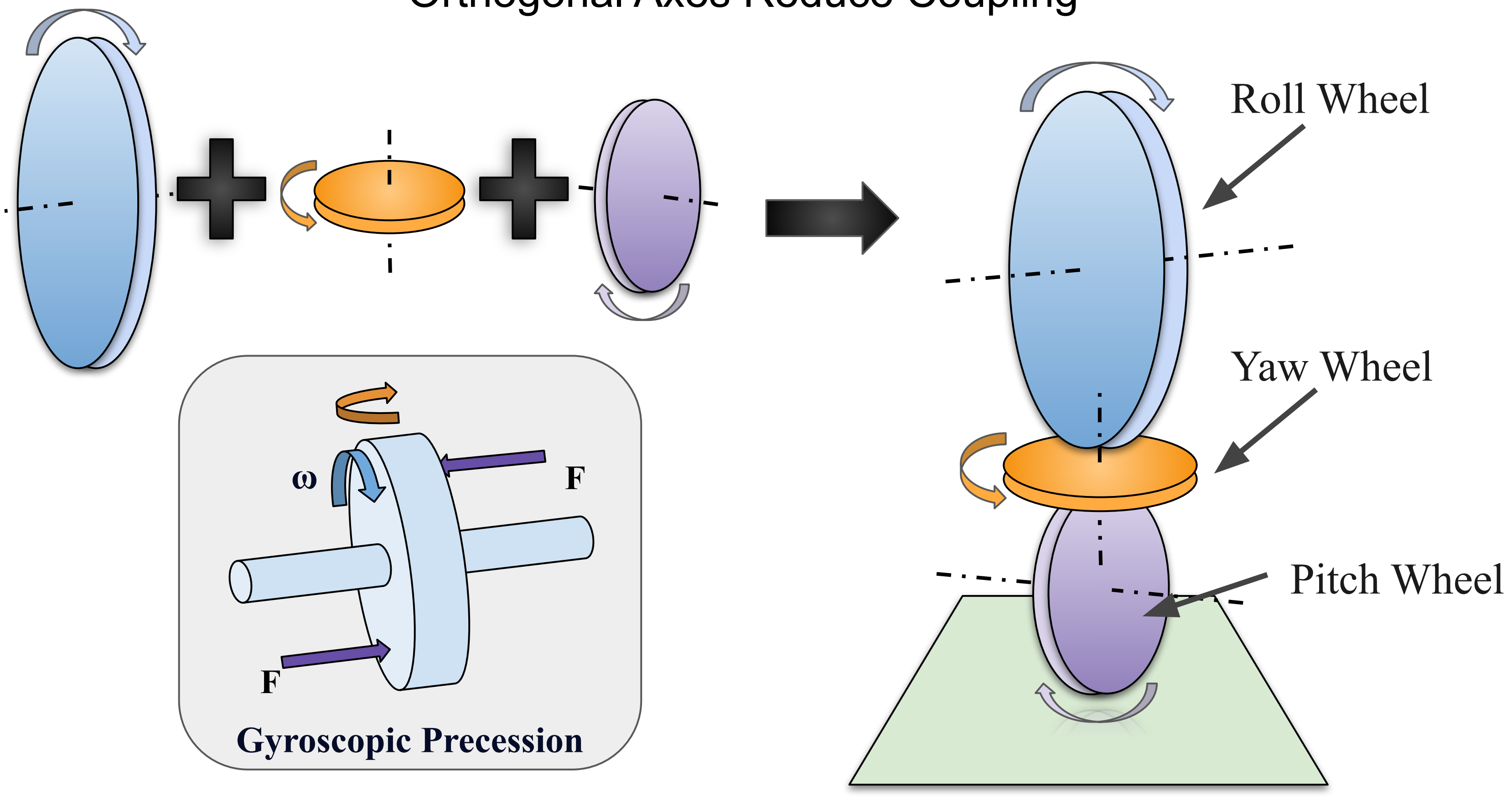


Fig. 1 Orthogonal reaction wheel coupling and gyroscopic effects

Physical Design

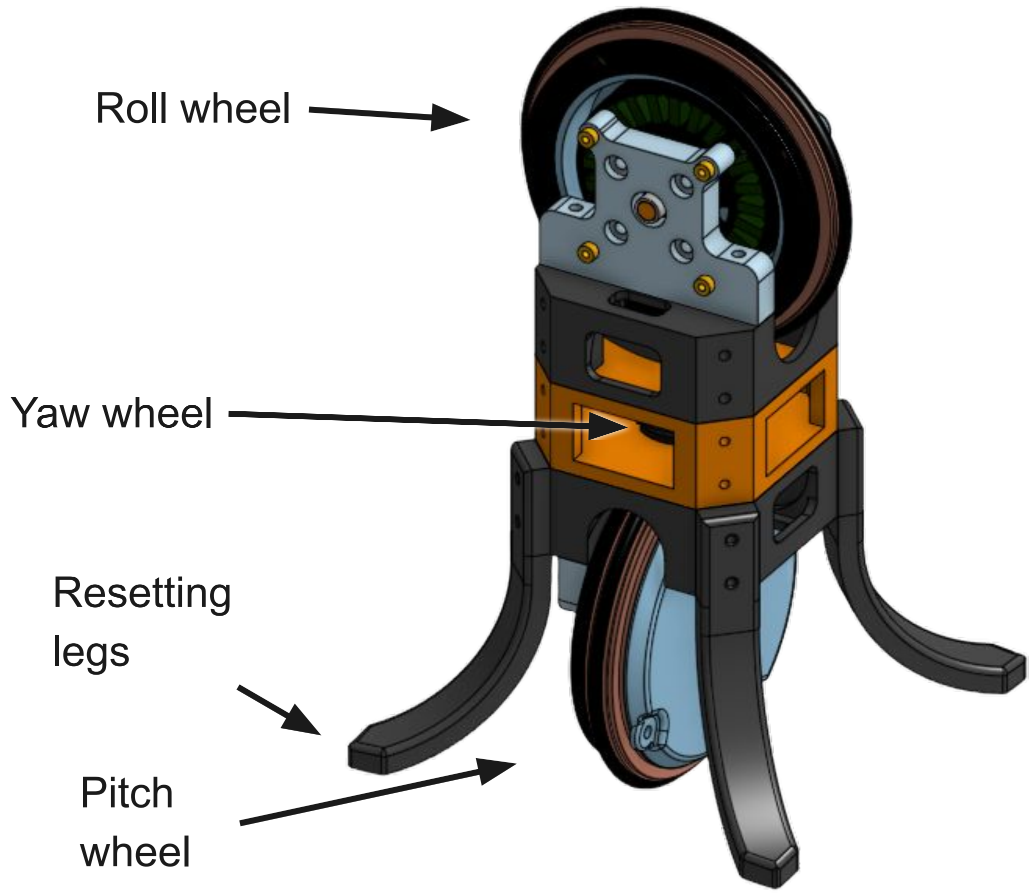


Fig 2. Full Robot CAD

Full Robot

- Low center of mass (CoM) for self-righting capabilities
- CoM placed above pitch wheel contact point

Actuators

- Directly driven by a brushless DC motor for high torque to mass ratio
- Motor mount holds BLDC driver equipped with magnetic encoder
- Steel rings add moment of inertia

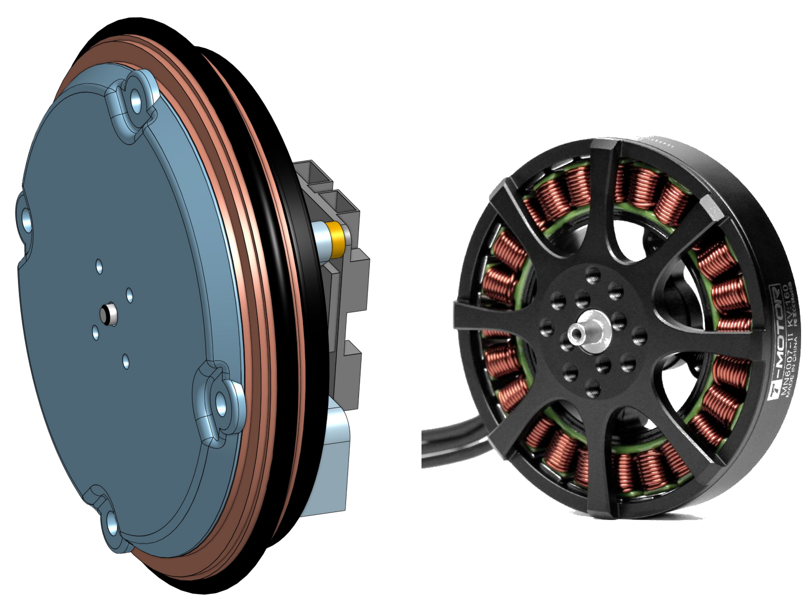


Fig 3. Flywheel CAD and BLDC

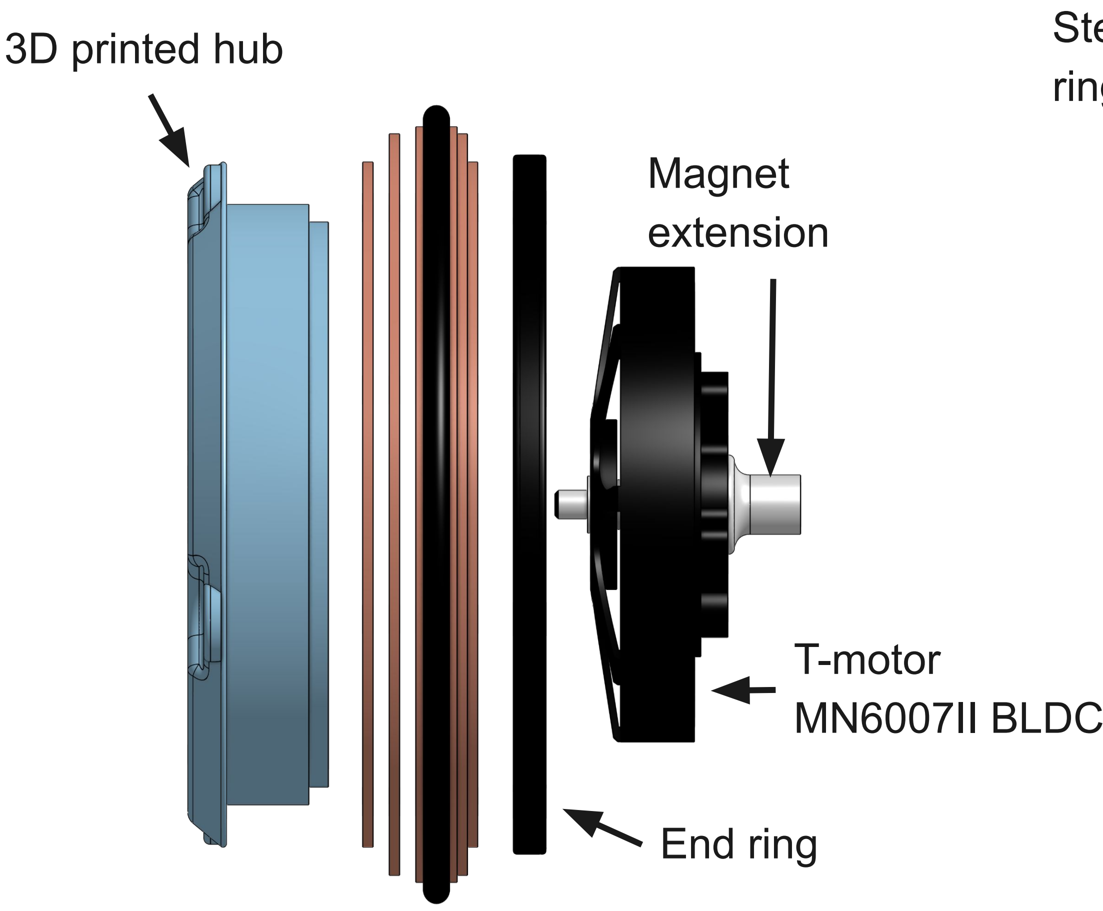


Fig 4. Flywheel exploded view

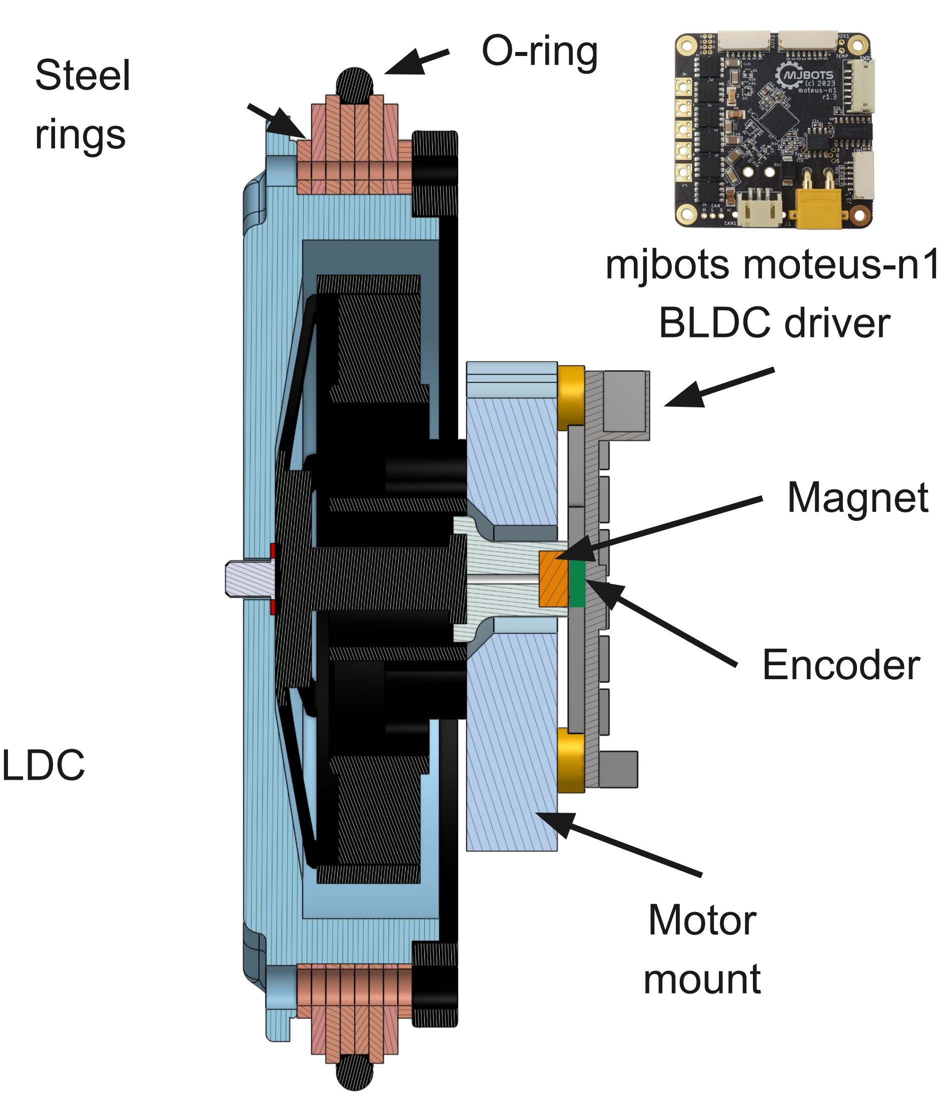


Fig 5. Flywheel cross-section

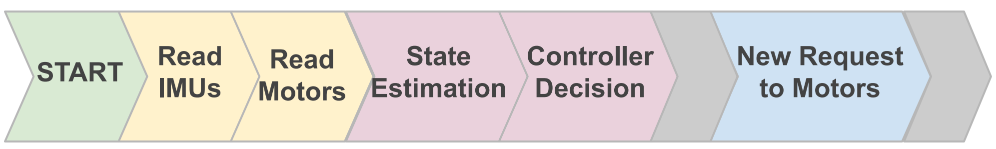
Computer and Software Systems

Nvidia Jetson Nano

A fast and versatile microcontroller with a GPU enhanced processor capable of running multiple neural networks in parallel for Reinforcement Learning and Computer Vision applications. Our solution uses a custom compiled RT Linux kernel and Python framework.

A Central Controller

The Jetson orchestrates the entire control loop cycle by coordinating a distributed system of sensors and motor drivers, updating at 100 times per second. The microcontroller also hosts the central control agent that makes decisions based on the robot's position and movement through the environment.



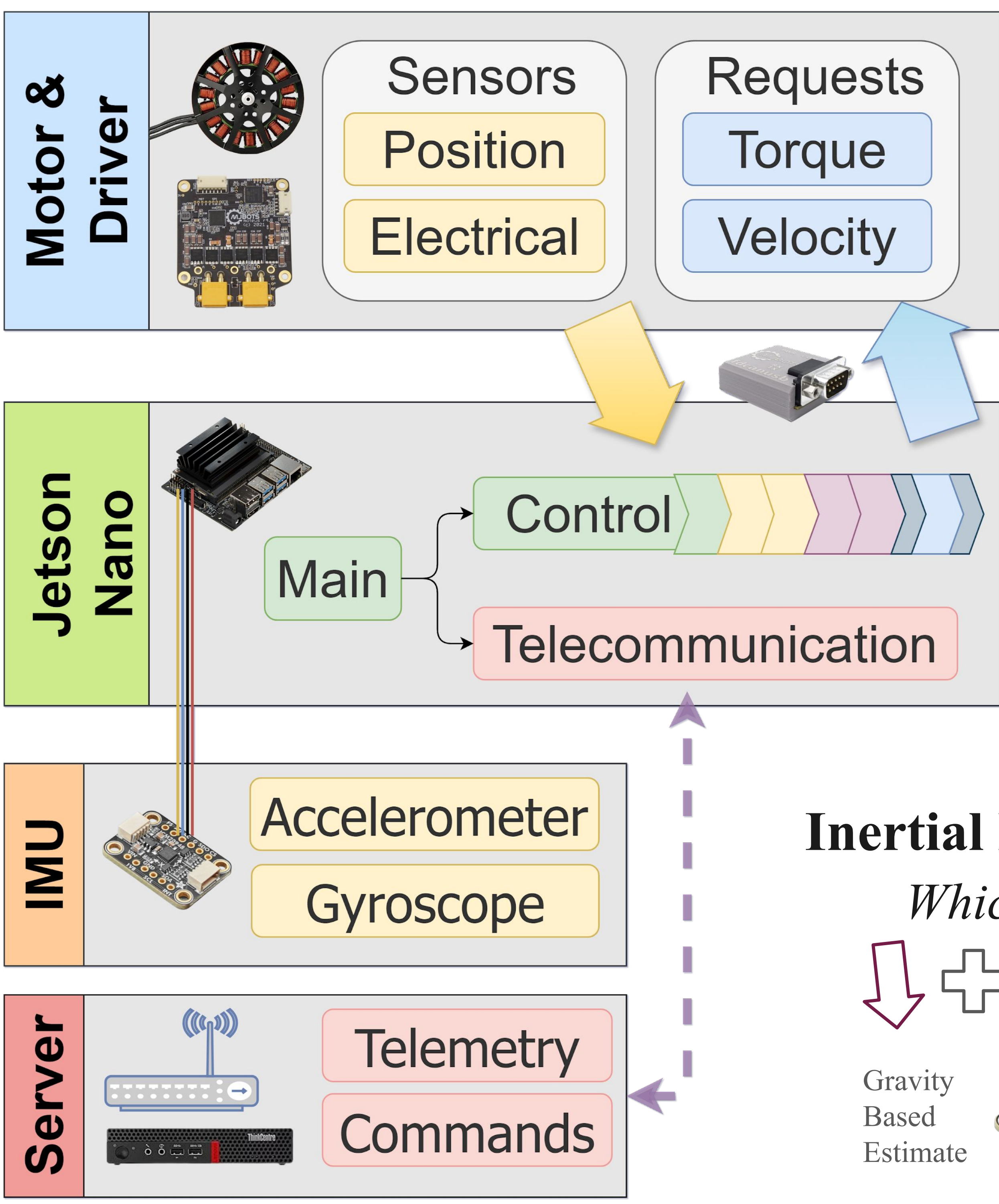


Fig 6. Software system communication

Moteus-n1 Drivers

The motor drivers run their own low level control loop using an onboard microcontroller. They are field oriented control based drivers that use a magnetic encoder and the current in the coils to sense the motor positions, speeds, and torques.

This independent control loop abstracts away the smaller details and allows the jetson to make direct torque requests.

Inertial Measurement Unit

Which Way is Down?

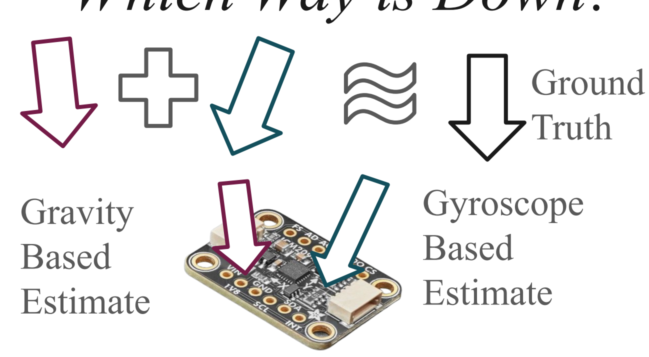


Fig 7. IMU state estimation

Telemetry Server

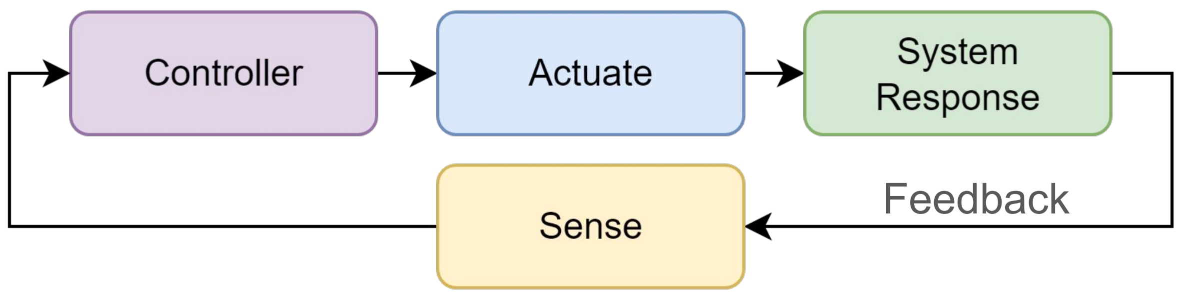
- Wi-fi transmission through the internet using MQTT Protocol
- Telemetry viewing via Grafana
- Test databasing with Influxdb
- Command UI with NodeRED

Madgewick Sensor Fusion

A complementary filter that fuses information from the rate of rotation and the direction of acceleration coming from the IMU. The combined sensor readings eliminate the angular drift that is inherent to the gyroscope alone, providing self-correcting readings.

Control

The controller is the brain. It reads from sensors computes a decision and sends it to the motors to execute. Classical control will serve as a benchmark for RL control.



Approach 1: Traditional Controls

PID

- Independent controllers for each axis/actuator pair
- Apply torque based on where it is now, where it came from and where it is going

$$\tau(t) = K_p\theta(t) + K_i\int_0^t \theta(t)dt + K_d\frac{d}{dt}\theta(t)$$

Approach 2: Reinforcement Learning

Proximal Policy Optimization

- Controller learns to balance through trial and error
- A reward function guides the models behaviour
- Domain Randomization (DR) is important for effective Sim2Real transfer

$$reward = 1 - |tanh(4\theta)| - 0.01\,|\omega_{wheel}| - 0.1\,|\alpha_{wheel}|$$

Current progress

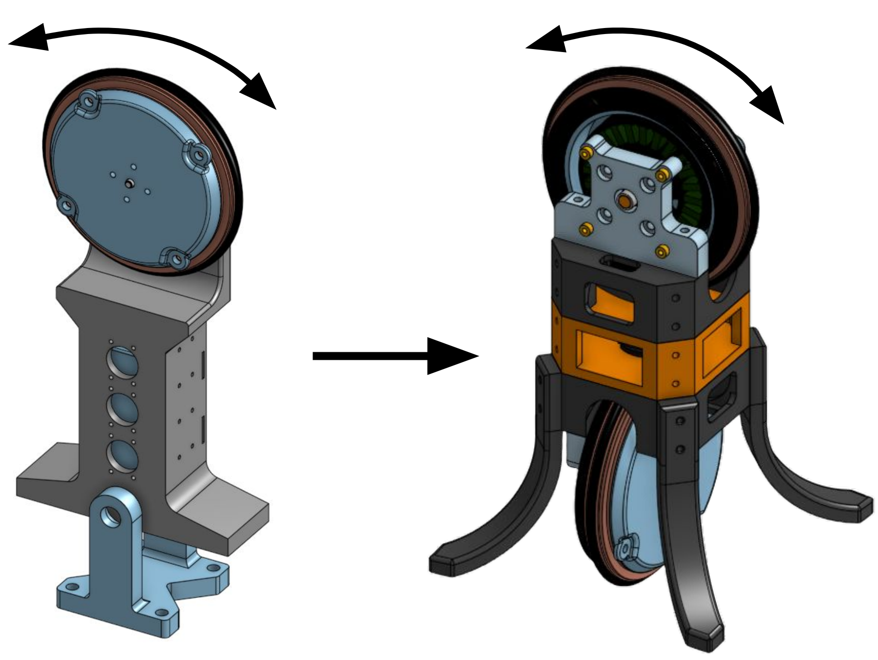


Fig 8. RWIP CAD vs full CAD

- PID model in progress
- Reinforcement learning controller balances for 10+ minutes with DR
- Exposing RL to more randomness and more effective reward functions leads to better balance