Cue-B
Final Project Presentation

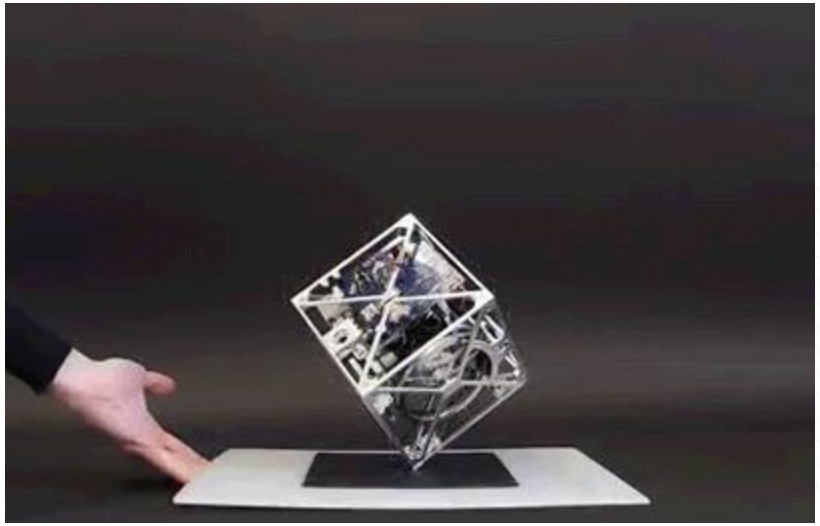
Will Compton, Domenic DiCarlo, Thalmus McDowell, Micah Morris – Group A2

ME 4012 Modeling and Control of Motion Systems

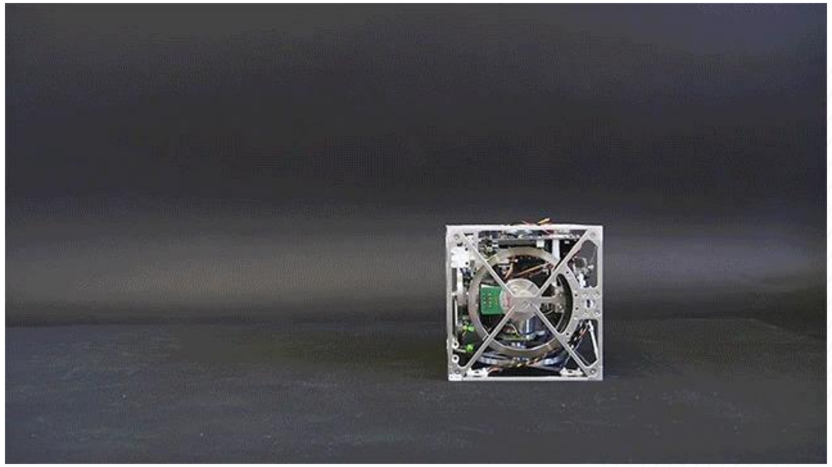
Spring 2022 Final Project Presentation

Motivation

- Reaction-wheel controlled self-balancing cube
- Inspiration taken from Cubli
  - Cube that can balance, stand, and "walk"
- System is not naturally stable (gravity)
  - Control needed for balance

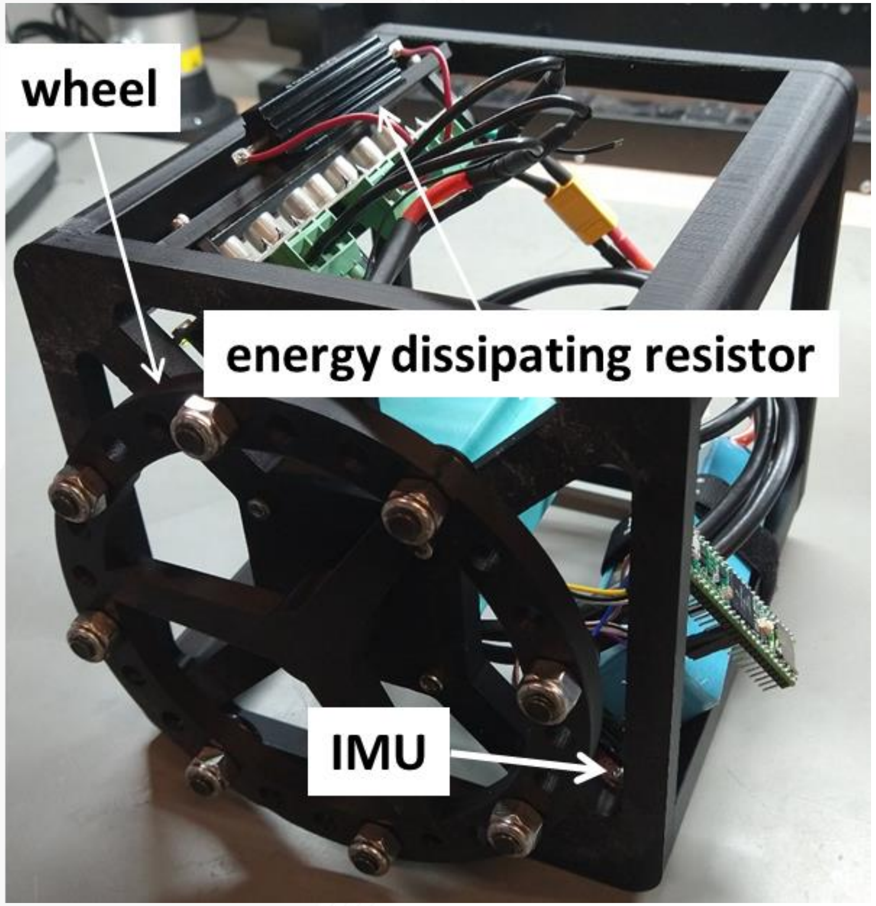


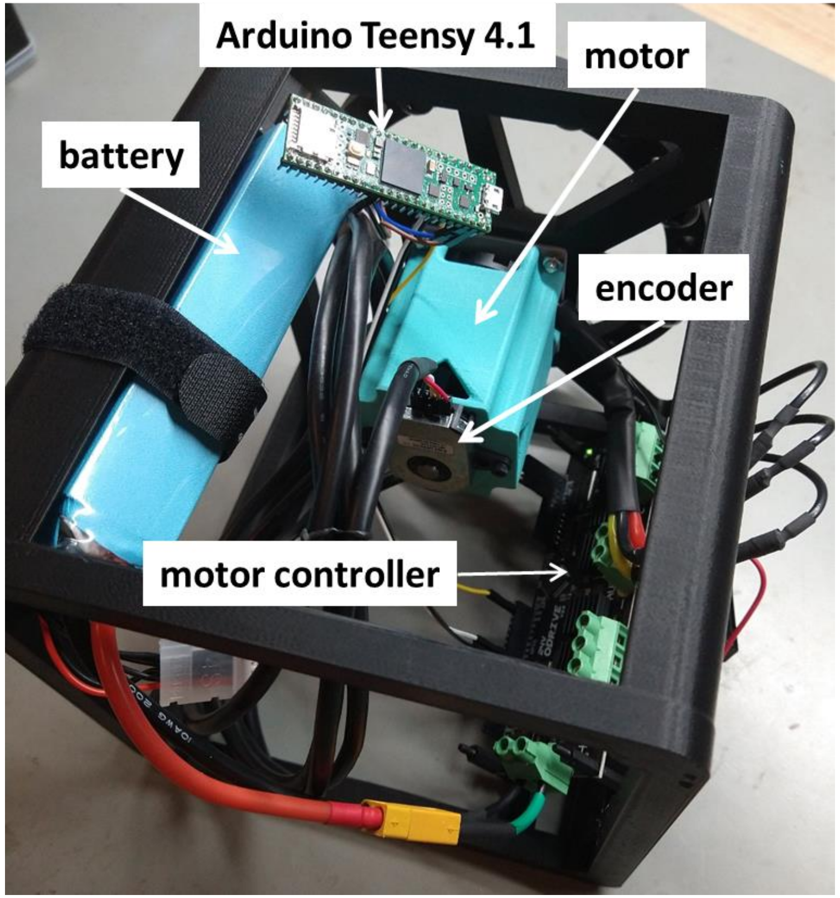
Cubli - Balancing



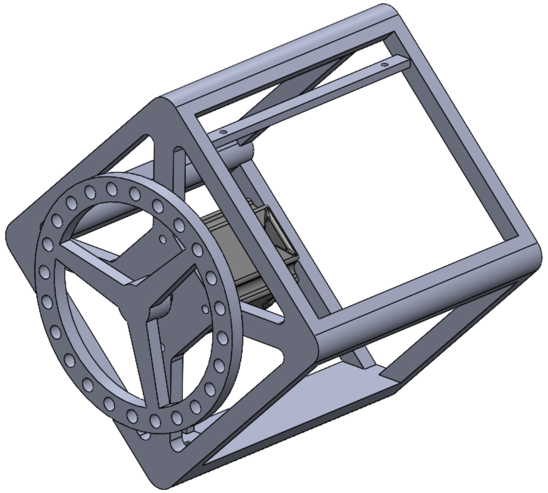
Cubli – Standing up

Design





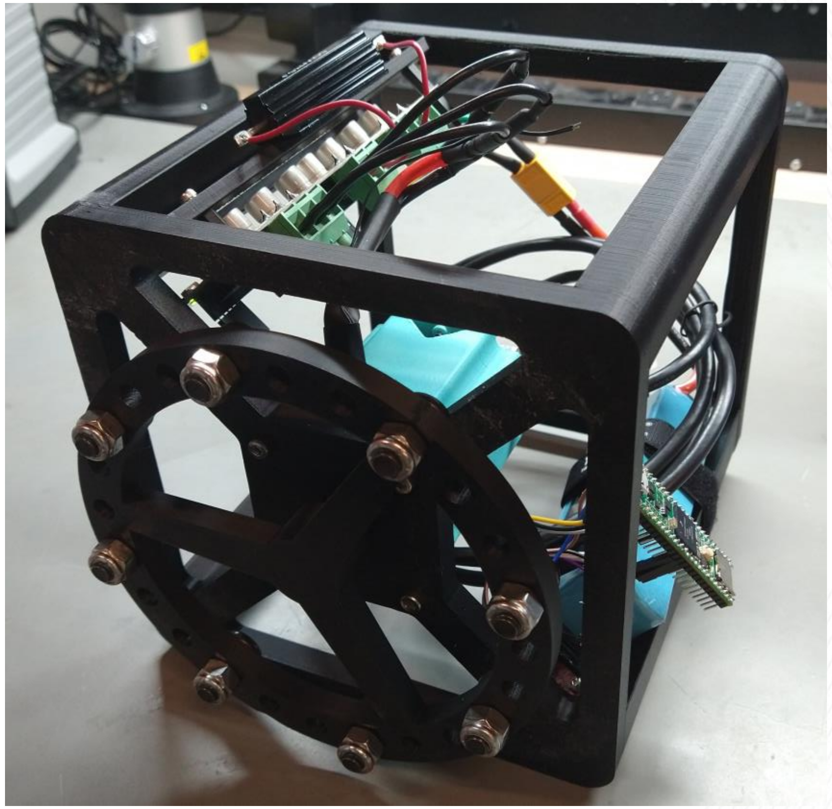
Cue-B – Component layout



CAD Model

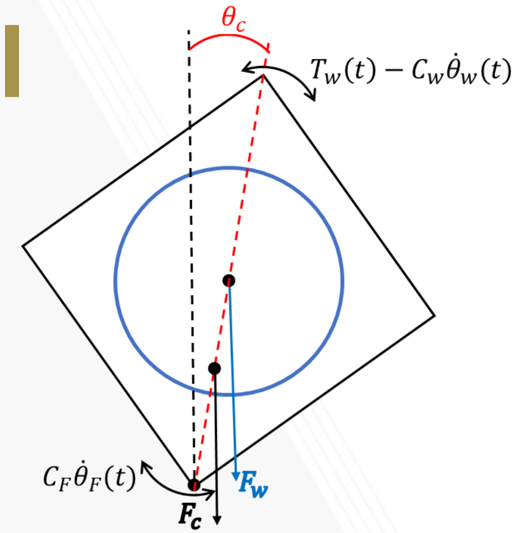
Design

- IMU senses changes in angular acceleration/velocity
- IMU positioned on pivot point
- Battery is placed above pivot point to make control easier by lowering center of mass
- Reaction wheel has mass distributed far from center: maximize inertia, minimize mass
- Motor drives reaction wheel to produce torque to achieve stability
  - ODrive motor controller has a torque mode where torque can be commanded directly

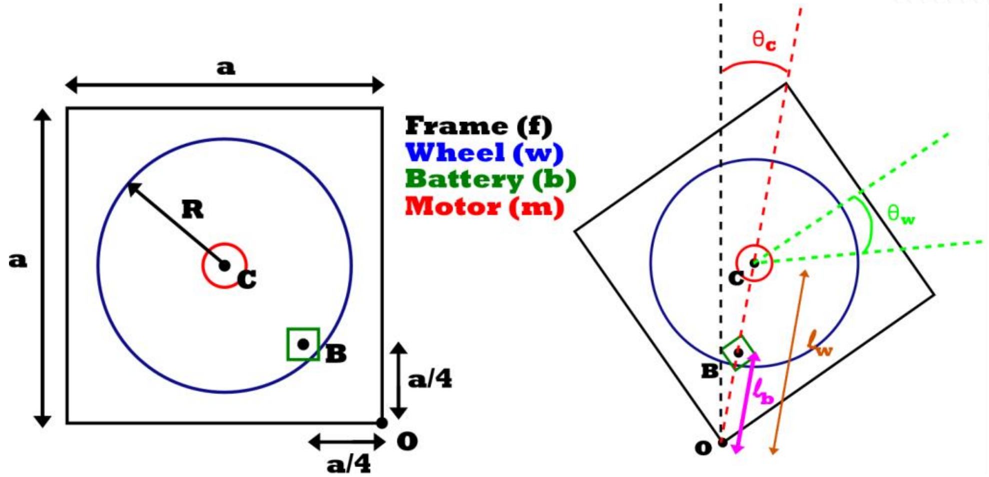


Cue-B – Assembled

Model



Variable	Meaning
θ_c	Angular displacement of COM
$\dot{\theta}_w$	Angular velocity of wheel
m_w	Mass of the Wheel
m_c	Mass of everything excluding wheel (applied at COM)
T_w	Torque applied by motor
l_w	Distance from pivot to wheel (center of frame)
l_c	Distance from pivot to COM



State Equations

$$\begin{bmatrix}\dot{\theta}_c\\ \ddot{\theta}_c\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \left(\dfrac{1}{I_{oc}}\right)(m_w l_w + m_c l_c)g & 0 \end{bmatrix}\begin{bmatrix}\theta_c\\ \dot{\theta}_c\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{1}{I_{oc}}\end{bmatrix} T_w$$

$$\mathbb{Y} = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta_c\\ \dot{\theta}_c\end{bmatrix}$$

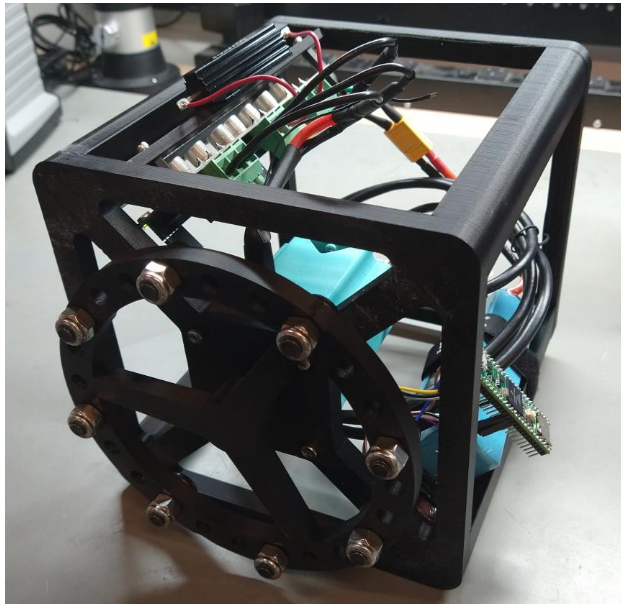
$$\begin{bmatrix}\dot{\theta}_c\\ \ddot{\theta}_c\end{bmatrix} = \begin{bmatrix}0 & 1\\ a & 0\end{bmatrix}\begin{bmatrix}\theta_c\\ \dot{\theta}_c\end{bmatrix} + \begin{bmatrix}0\\ b\end{bmatrix} T_w$$

$$\mathbb{Y} = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta_c\\ \dot{\theta}_c\end{bmatrix}$$

Georgia Tech
CREATING THE NEXT

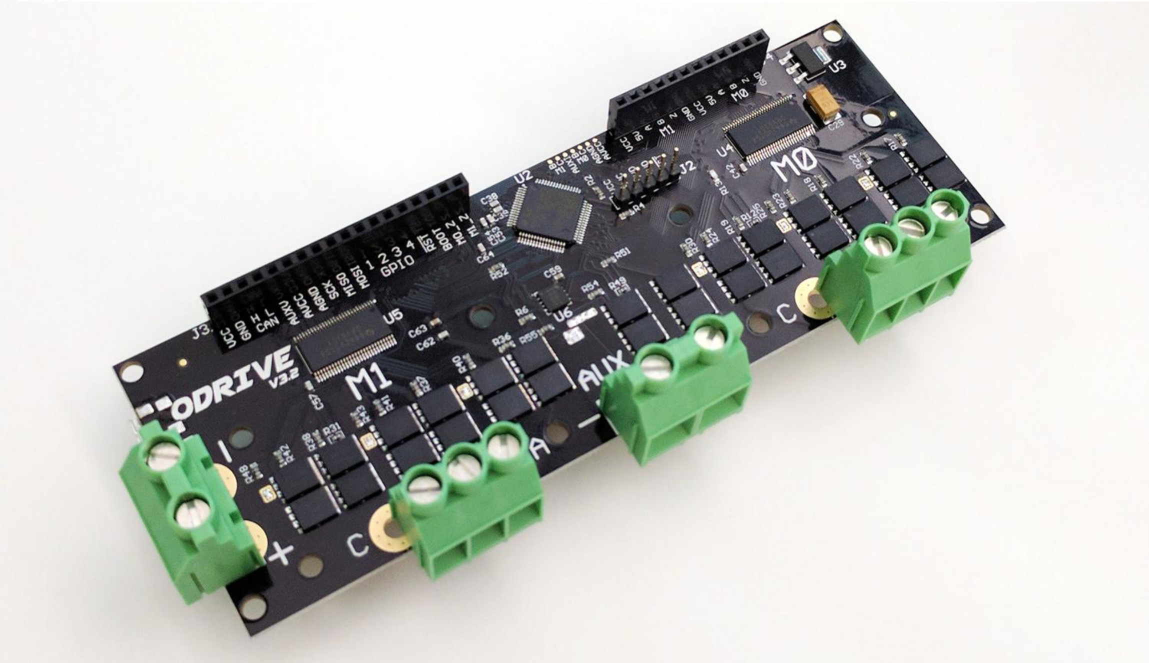
Modeling Assumptions

- Negligible mechanical damping (motor, joint with ground, air resistance)
  - Justification: these sources of damping are small, and typically aid stability rather than degrade it
  - Reason: decouples the interaction of the wheel and the frame (damping would couple through velocity related effects)
    - This allows the system to be approximated as second order, greatly simplifying analysis and controller design.



Modeling Assumptions

- The ODrive torque controller is fast enough that it’s dynamics are negligible.
  - Justification: The ODrive is a highly capable BLDC motor controller, paired with a 20480 CPR encoder, should enable precise and fast torque control
  - Reason: Allows model to treat torque commands as taking place instantaneously, without having to model the dynamics of the motor controller.



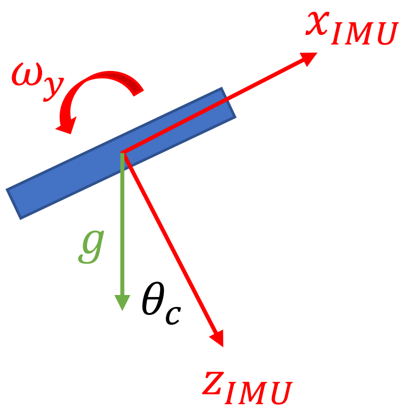
Controller Design: State Estimation

- The angle of the center of mass of the cube, θ_c, cannot be measured directly
  - An Infinite Impulse Response observer, the complementary filter, is used to estimate θ_c.

Geometry of Gravitational Acceleration

$$\theta^i_{c,accel} = \tan^{-1}\frac{a_x}{a_z}$$

Tracks low-frequency movements well
No Drift
Inaccurate with high frequency



Integration of Gyroscope

$$\theta^i_{c,gyro} = \theta^{i-1}_c + \omega\Delta t$$

Tracks high-frequency movements well
Prone to drift - inaccurate with low frequency

$$\theta^i_c = \alpha\theta^i_{c,gyro} + (1-\alpha)\theta^i_{c,accel}$$

$$0 \ll \alpha < 1$$

Controller Design: State Space

- System model is approximated as linear second order
  - State space control selected to place poles
- Controllability

$$C_M = [B \quad AB] = \begin{bmatrix} 0 & b \\ b & 0 \end{bmatrix} \rightarrow Full\ Rank, Controllable$$

- Time Specifications:
  - Settling time, $t_s = 1s$
  - Maximum Overshoot, $M_p = 0.05 \;\; (5\%)$

$$M_p = 0.05 \rightarrow \zeta = \frac{\ln(M_p)}{\sqrt{\pi^2 + \ln^2 M_p}} = 0.69 \qquad t_s = 1 \rightarrow \omega_n = \frac{4}{t_s \zeta} = 5.79$$

$$CLCE: \; s^2 + 2\zeta\omega_n s + w_n^2 = s^2 + 8s + 33.5960 = 0$$

Georgia Tech
CREATING THE NEXT

Controller Design: State Space

- Closed Loop Characteristic of State Space System

$$CLCE\!: \det\left(sI - (A - BK)\right) = 0$$

$$\left|\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \left(\begin{bmatrix} 0 & 1 \\ a & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ b \end{bmatrix}\begin{bmatrix} k_1 & k_2 \end{bmatrix}\right)\right| = 0$$

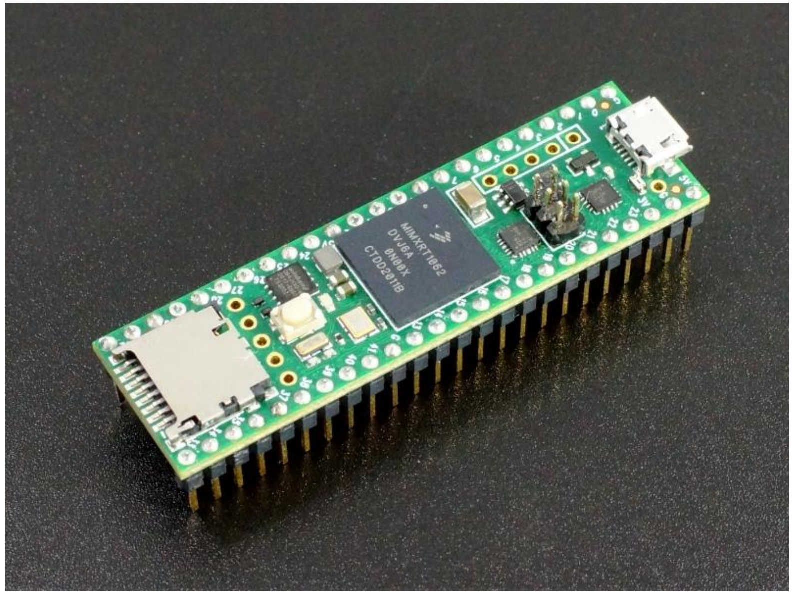
$$\begin{vmatrix} s & -1 \\ -a + k_1 b & s + k_2 b \end{vmatrix} = 0$$

$$s^2 + k_2 bs + a - k_1 b = s^2 + 8s + 35.596 = 0$$

$$k_1 = 2.29 \quad k_2 = 0.201$$

Results: Implementation

- Teensy 4.1 is a fast and capable microcontroller
  - Communication with IMU is performed over SPI at 7MHz
  - Both complementary filter and control loop are implemented in the same callback function
    - This callback is tied to a hardware timer at 200Hz
    - This speed is sufficiently fast to ignore the difference between the discrete and continuous implementations of the controller.



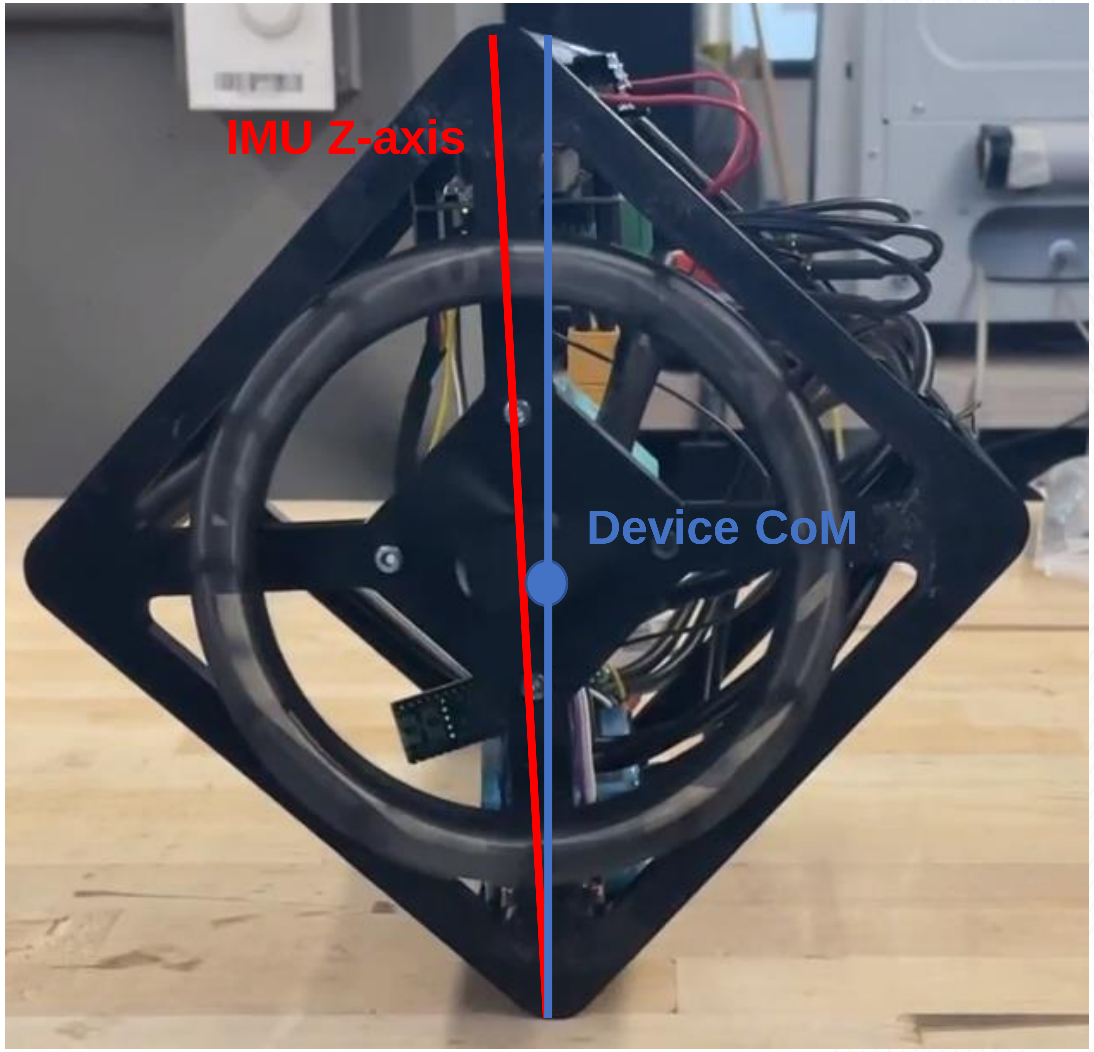
Georgia Tech
CREATING THE NEXT

```
void rateCallback(){
    // Read data from the IMU
    IMUGetData(imuData);
    // Compute an angle estimate using the acceleration values
    accelY = (1 - LOW_PASS_ALPHA) * imuData[0] / ACCEL_SENSITIVITY_2 + LOW_PASS_ALPHA * accelY;
    accelZ = (1 - LOW_PASS_ALPHA) * imuData[1] / ACCEL_SENSITIVITY_2 + LOW_PASS_ALPHA * accelZ;
    thetaAccel = - atan2f(accelY, accelZ);
    // Compute an angle estimate using the gyroscope values
    thetaDotX = imuData[2] / GYRO_SENSITIVITY_250 * DEG2RAD;
    thetaGyro = thetaX + thetaDotX * DT;
    // Complemnetary filter
    thetaX = COMP_FILT_ALPHA * thetaGyro + (1 - COMP_FILT_ALPHA) * thetaAccel;

    // Check that the device is still in a reasonable position, otherwise shut the motor down.
    if (abs(thetaX) > SHUTDOWN_THRESHOLD) {
      controlActive = false;
    }
    if (controlActive) {
        // Compute the control input using Kp, Kd gains
        controlInput = Kp * (thetaX - THETA_OFFSET) + Kd * thetaDotX;
        // Convert the control input from Nm to A
        controlInput = controlInput * TORQUE_CONSTANT;
        controlInput = max(min(controlInput, MAX_CURRENT), -MAX_CURRENT);  // Don't command outside motor operating range
        // Set the torque
        odrive.SetCurrent(0, controlInput);
    } else {
        // Control is not active, set torque to zero.
        controlInput = 0;
        odrive.SetCurrent(0, 0);
    }
}
```

```
void rateCallback(){
    // Read data from the IMU
    IMUGetData(imuData);
    // Compute an angle estimate using the acceleration values
    accelY = (1 - LOW_PASS_ALPHA) * imuData[0] / ACCEL_SENSITIVITY_2 + LOW_PASS_ALPHA * accelY;
    accelZ = (1 - LOW_PASS_ALPHA) * imuData[1] / ACCEL_SENSITIVITY_2 + LOW_PASS_ALPHA * accelZ;
    thetaAccel = - atan2f(accelY, accelZ);
    // Compute an angle estimate using the gyroscope values
    thetaDotX = imuData[2] / GYRO_SENSITIVITY_250 * DEG2RAD;
    thetaGyro = thetaX + thetaDotX * DT;
    // Complemnetary filter
    thetaX = COMP_FILT_ALPHA * thetaGyro + (1 - COMP_FILT_ALPHA) * thetaAccel;

    // Check that the device is still in a reasonable position, otherwise shut the motor down.
    if (abs(thetaX) > SHUTDOWN_THRESHOLD) {
      controlActive = false;
    }
    if (controlActive) {
        // Compute the control input using Kp, Kd gains
        controlInput = Kp * (thetaX - THETA_OFFSET) + Kd * thetaDotX;
        // Convert the control input from Nm to A
        controlInput = controlInput * TORQUE_CONSTANT;
        controlInput = max(min(controlInput, MAX_CURRENT), -MAX_CURRENT);  // Don't command outside motor operating range
        // Set the torque
        odrive.SetCurrent(0, controlInput);
    } else {
        // Control is not active, set torque to zero.
        controlInput = 0;
        odrive.SetCurrent(0, 0);
    }
}
```

Results: Implementation

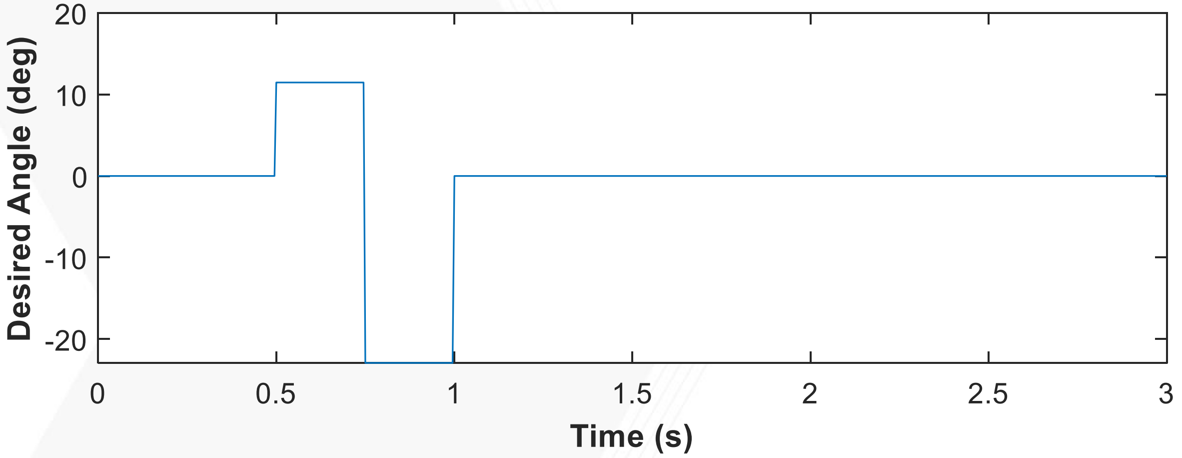
- The center of mass of the entire device is not necessarily aligned with the z-axis of the IMU
  - An offset angle, θ_{offset}, is estimated by trying to manually balance the device with no control
  - This offset angle is subtracted from the measured angle when determining the error

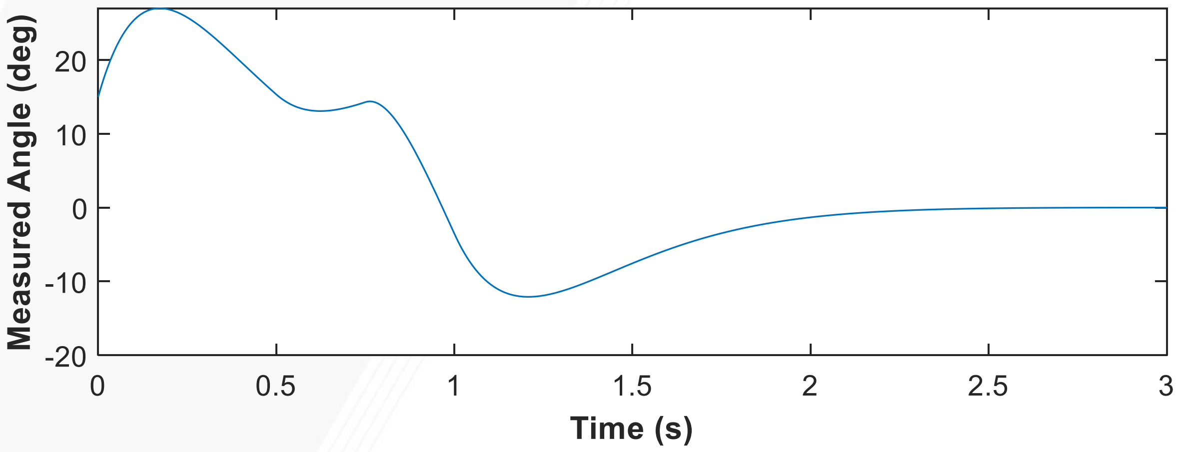


IMU Z-axis vs COM

Results: Simulation

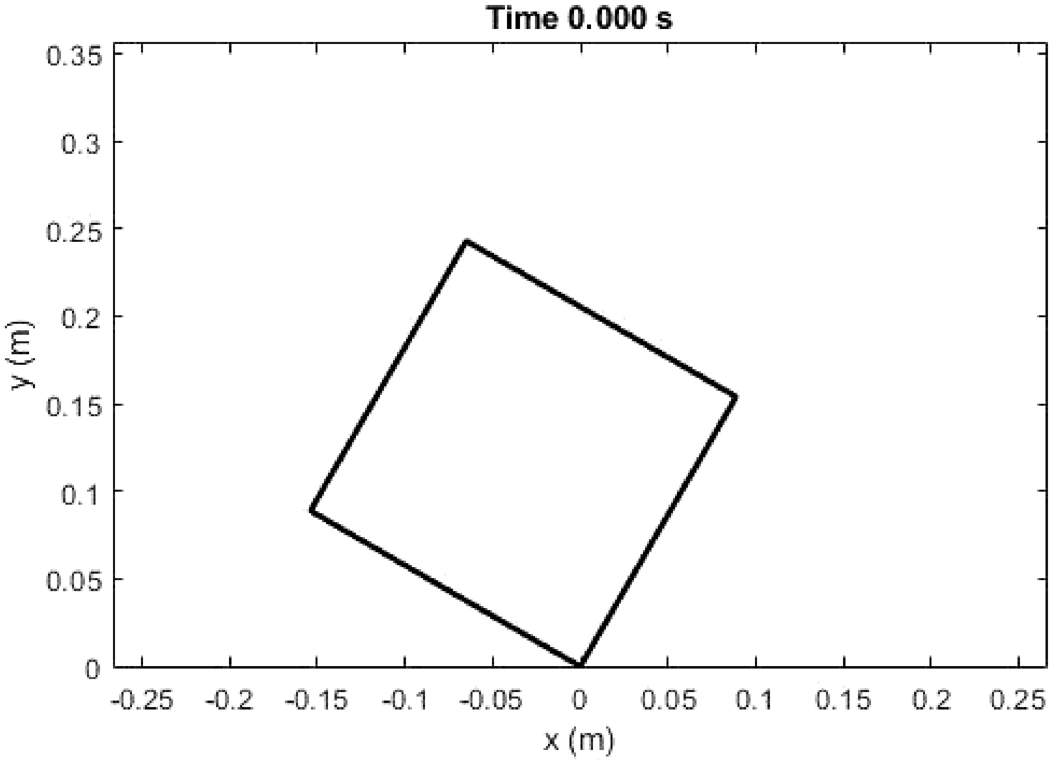
Closed Loop Step Response Simulating Perturbations from Equilibrium





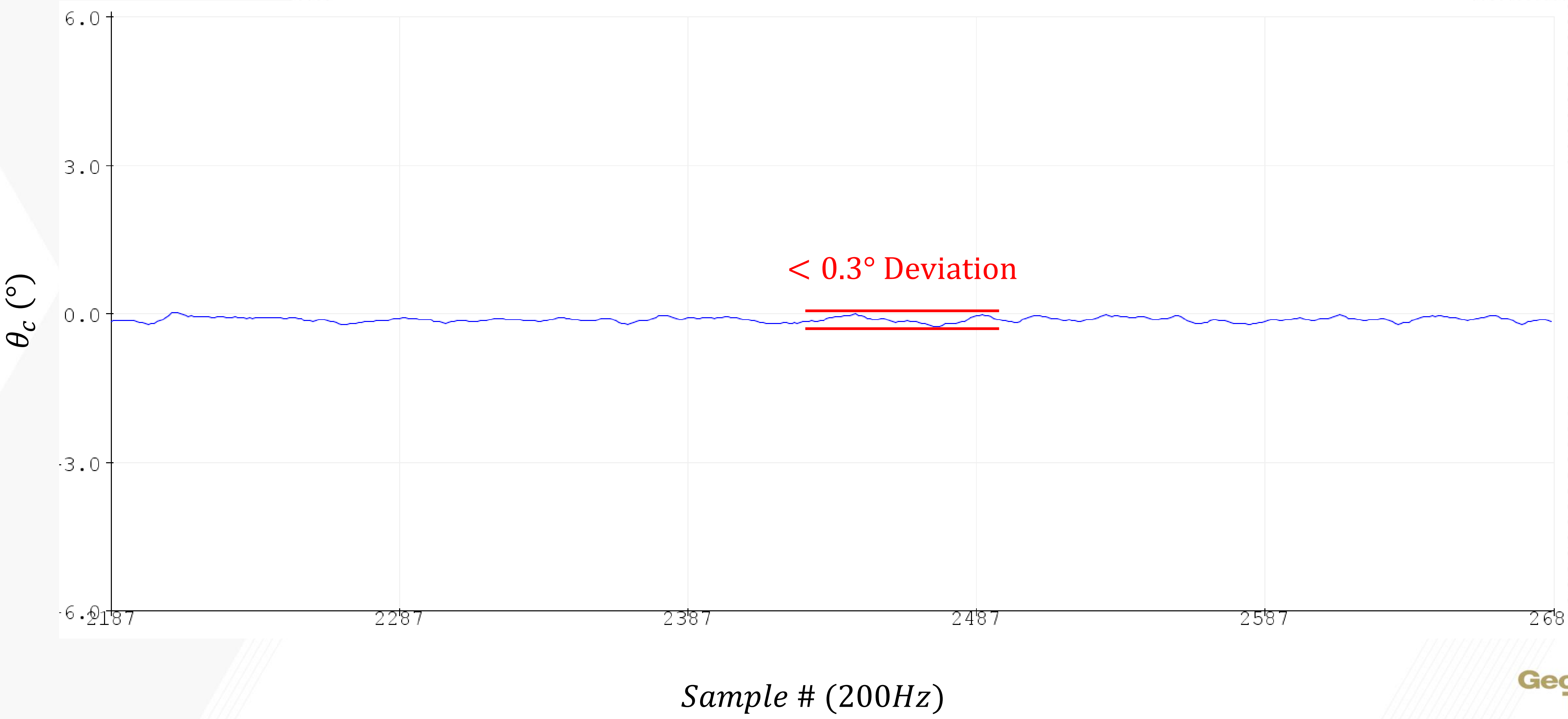
Closed Loop Step Response Simulation

- Initial Conditions: $\theta_c = 15° \; and \; \dot{\theta}_c = 3\,\frac{rad}{s}$
- K Gains: $[2.29 \; 0.201]$

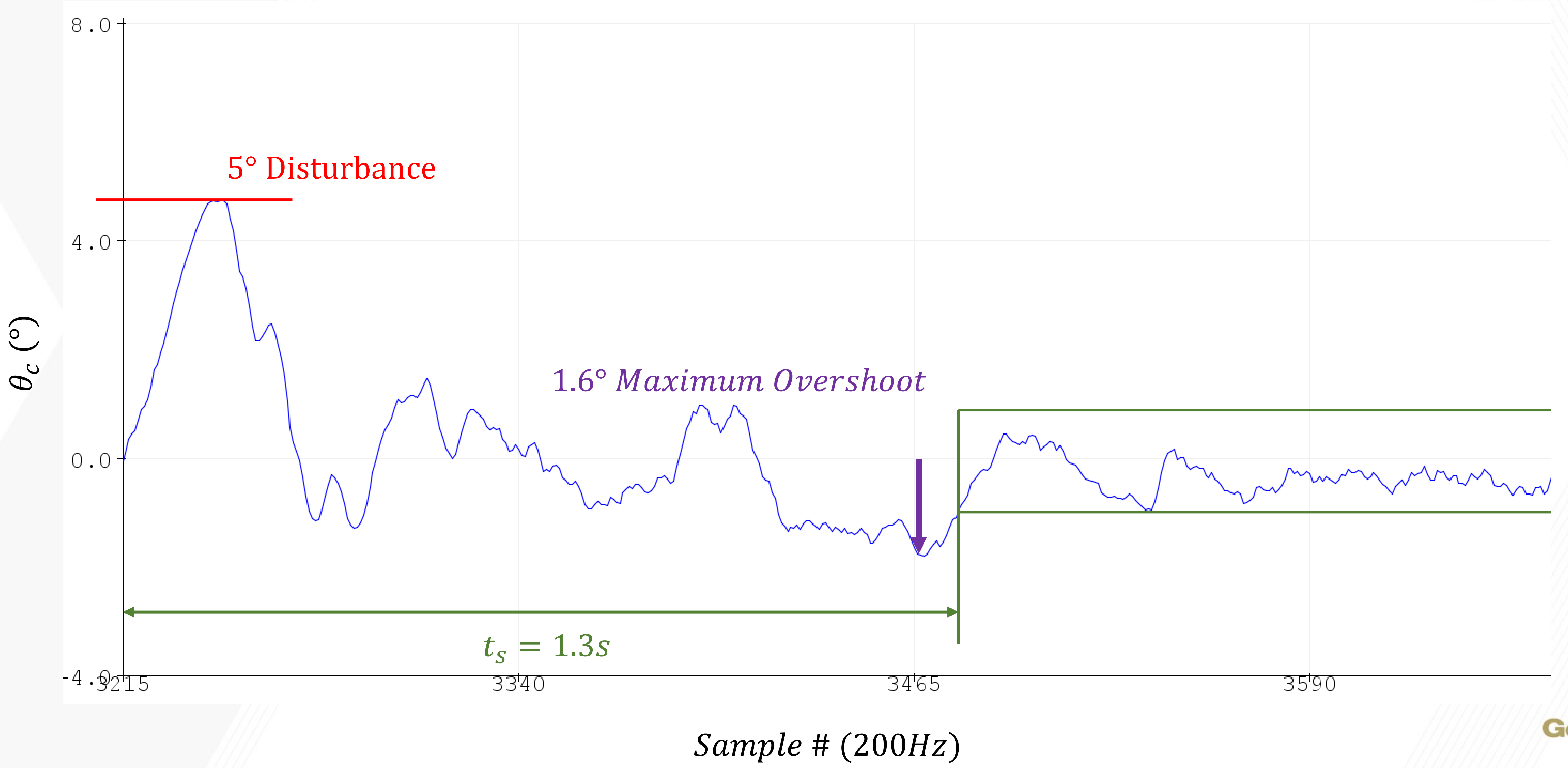


Georgia Tech
CREATING THE NEXT

Results: Steady State Balancing on Device

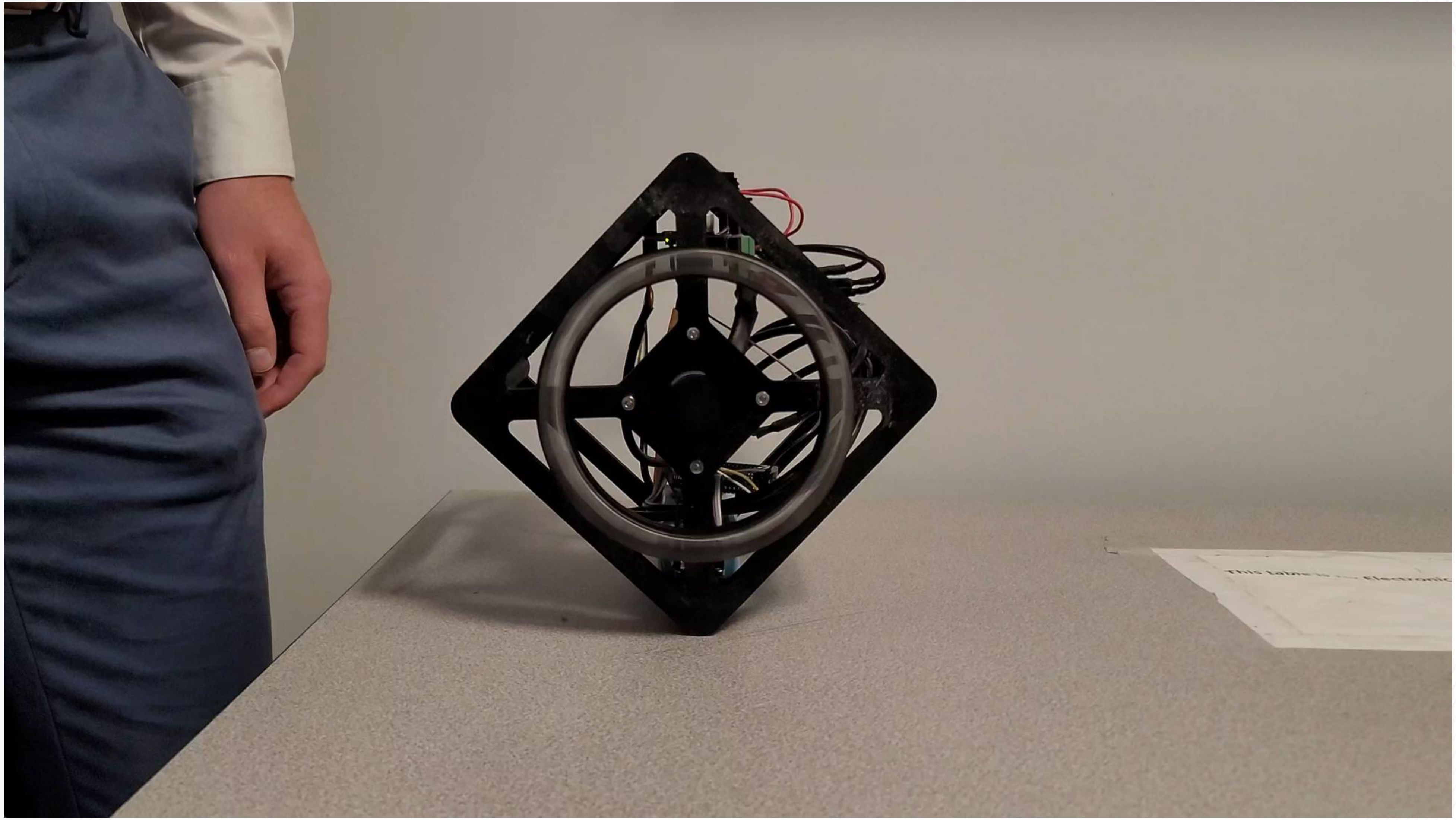


Results: Disturbance Rejection on Device



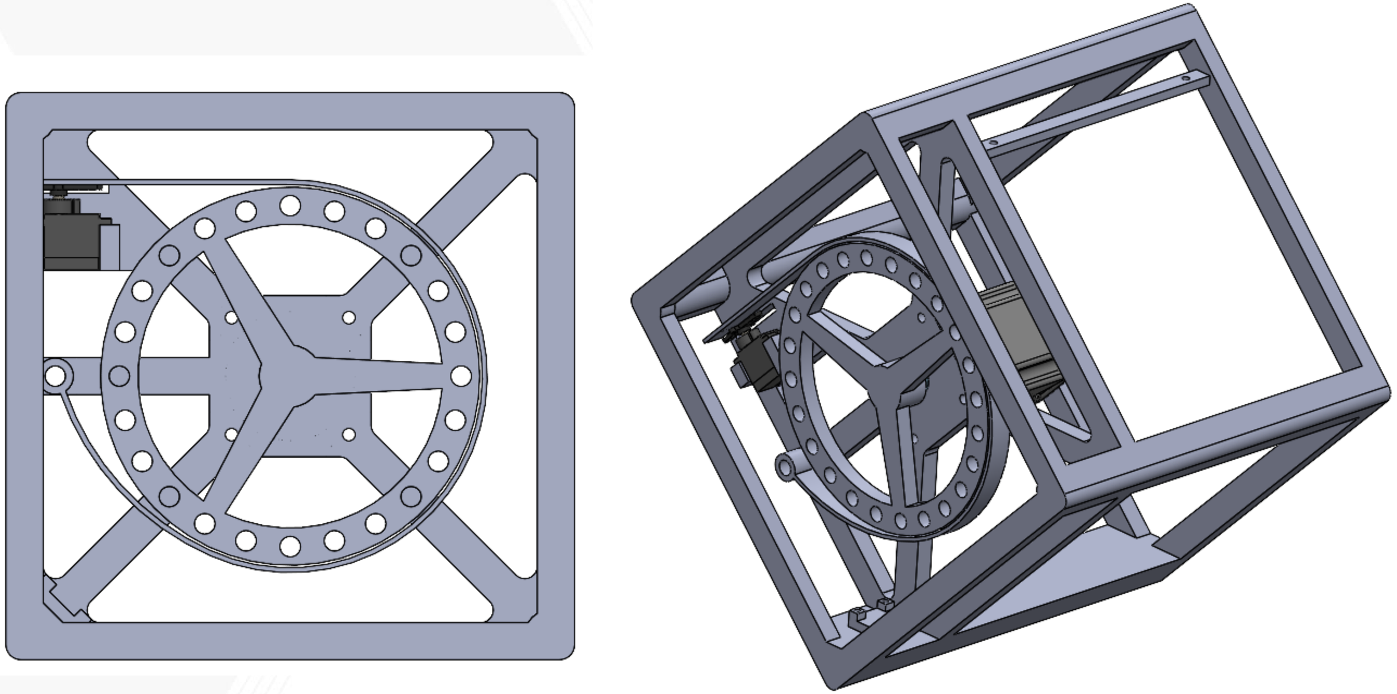
Georgia Tech
CREATING THE NEXT

Video

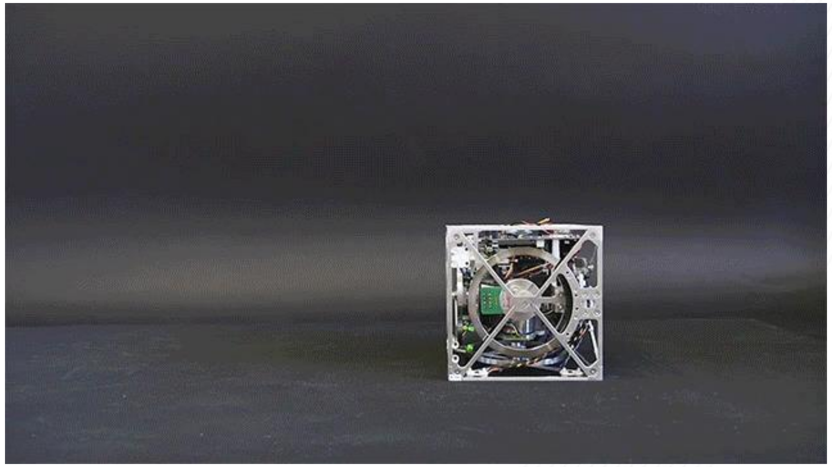


Conclusion

- Project Successful
- Controlling with state space works well
  - Cue-B is stable over a fairly wide range of control gains.
- Next step is to implement a brake to allow Cue-B to "stand" from rest



Cue-B_v2 – CAD model



Cubli – Standing and Balancing

Questions?