ME 4405 Final Project: Balancing Square Reaction Wheel

Srijan Duggal and Kyle Heiss

May 2021

Contents

1 Introduction

There are many practical applications for stabilization of unstable systems, conceptually similar to an inverted pendulum. One system that inspired us was a self-balancing motorcycle from Honda (using their riding assist technology). Their system uses a different mechanism than a reaction wheel, but it got us interested in stabilization. We found some examples of self-balancing motorcycle projects on YouTube, where the creators change the angular velocity of a flywheel in order to induce a reaction torque. The goal of the project is to create a cube that can stay on its edge using the process described above and remain stable even after being perturbed. This a scaled down implementation of a balancing mechanism that could be used on other devices.

2 System-Level Design

The operating principle of this system is that a flywheel changes speed based on the current tilt angle of the cube. In order to achieve this, the design can have a frame with a motor on it, where the flywheel is mounted to the motor shaft. There must be some way to measure the current tilt angle of the cube, which we chose to be an Inertial Measurement Unit (accelerometer + gyroscope) because we did not want the system to be fixed on an axle. As a constraint for this course, we must use an MSP432 processor for feedback control.

The functionality of the system is as follows. It is started in a balanced upright position, and then released. The system then modulates the angular acceleration of the flywheel in order to create a moment that acts on the frame in order to balance it.

3 Dynamic Modeling

In this section we will derive the equations of motion for our system. We will split the system into two rigid bodies: the frame (including the motor) and the reaction wheel. We will start with Free Body Diagrams of each body, and the following directional convention will be positive.

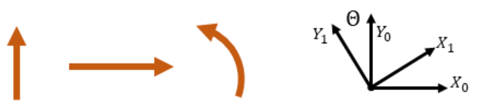


Figure 1: Positive Sign Convention & Coordinate Axes

Key points and parameters

A: point about which the frame rotates
B: center of mass of frame
C: motor shaft centerline (also center of mass of reaction wheel)
F: the fixed frame
XYZ_0: coordinate system on the fixed frame, origin at A
XYZ_1: coordinate system rigidly attached to the pendulum frame, origin at A
d: distance from point A to point B
d_{attach}: distance from point A to point C
m_{frame}: mass of the frame
m_{wheel}: mass of the reaction wheel
$I_{frame,A}$: moment of inertia about point A of the frame
$I_{wheel,C}$: moment of inertia about point C of the wheel

θ: clockwise angle between the frame and the vertical
ϕ: clockwise angle of the reaction wheel from the horizontal axis

Forces

$F_{rxn,x}$: Force between frame and reaction wheel, X_0 direction
$F_{rxn,y}$: Force between frame and reaction wheel, Y_0 direction
M_{rxn}: Moment between frame and reaction wheel, Z_0 direction
$F_{g,frame}$: Force of gravity on the frame
R_x: Reaction force at the pivot point, X_0 direction
R_y: Reaction force at the pivot point, Y_0 direction
$F_{g,frame}$: Force of gravity on the reaction wheel

3.1 FBD of Frame

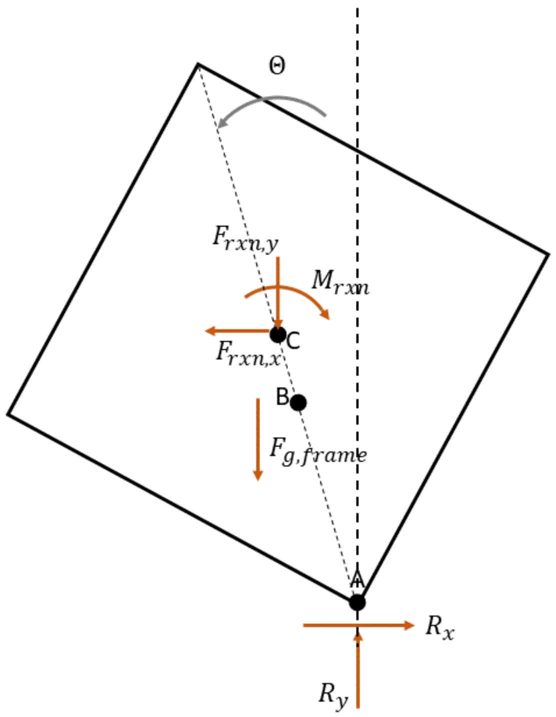


Figure 2: Free Body Diagram of Cube Frame

Summing the forces and moments on the frame, we get the following three equations:

$$\sum F_y = R_y - F_{g,frame} - F_{rxn,y} = m_{frame}\vec{a}_{C/F} \cdot \hat{e}_{Y_0} \tag{1}$$

$$\sum F_x = R_x - F_{rxn,x} = m_{frame}\vec{a}_{C/F} \cdot \hat{e}_{X_0} \tag{2}$$

$$\sum M_A = F_{rxn,x}d_{attach}cos\theta + F_{rxn,y}d_{attach}sin\theta + F_{g,frame}dsin\theta - M_{rxn} = I_{frame,A}\ddot{\theta} \tag{3}$$

3.2 FBD of Reaction Wheel

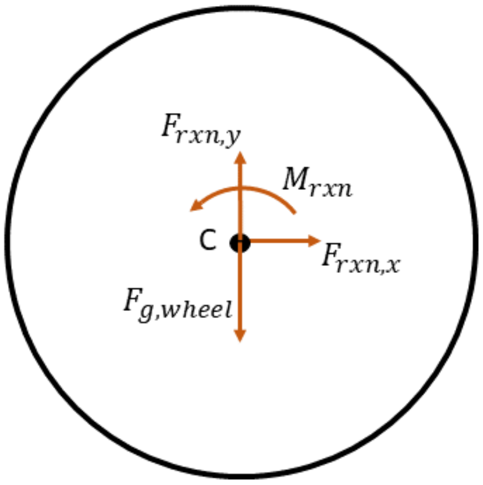


Figure 3: Free Body Diagram of Reaction Wheel

Summing the forces and moments on the reaction wheel, we get the following three equations:

$$\sum F_y = F_{rxn,y} - F_{g,wheel} = m_{wheel}\vec{a}_{C/F} \cdot \hat{e}_{Y_0} \tag{4}$$

$$\sum F_x = F_{rxn,x} = m_{wheel}\vec{a}_{C/F} \cdot \hat{e}_{X_0} \tag{5}$$

$$\sum M_c = M_{rxn} = I_{wheel,C}\ddot{\phi} \tag{6}$$

3.3 Kinematics

Now let's look at the kinematics to get the relevant acceleration: $\vec{a}_{C/F}$
Let's start with the position vector from point A to C
$\vec{r}_{AC} = d_{attach}\hat{e}_{Y_1}$

The velocity and acceleration of point A with respect to the fixed frame are both 0 since we assume a no-slip condition.
$\vec{v}_{A/F} = 0$

$\vec{a}_{A/F} = 0$

The velocity of point C is found using the relationship between two points on a rigid body.

$$\begin{aligned}\vec{v}_{C/F} &= \vec{v}_{A/F} + \dot{\theta}\hat{e}_{Z_0} \times \vec{r}_{AC} \\ &= \dot{\theta}\hat{e}_{Z_0} \times \vec{r}_{AC}\end{aligned}$$

And finally we can solve for the acceleration we are looking for.

$$\begin{aligned}\vec{a}_{C/F} &= \vec{a}_{A/F} + \ddot{\theta}\hat{e}_{Z_0} \times \vec{r}_{AC} - \dot{\theta}^2\vec{r}_{AC} \\ &= \ddot{\theta}\hat{e}_{Z_0} \times \vec{r}_{AC} - \dot{\theta}^2\vec{r}_{AC} \\ &= \ddot{\theta}\hat{e}_{Z_0} \times d_{attach}\hat{e}_{Y_1} - \dot{\theta}^2 d_{attach}\hat{e}_{Y_1}\end{aligned}$$

Looking at the coordinate system diagram in Figure 1, we can find the relationship between the XYZ_0 and XYZ_1 frames.

$$\begin{aligned}\hat{e}_{X_1} &= cos\theta\hat{e}_{X_0} + sin\theta\hat{e}_{Y_0} \\ \hat{e}_{Y_1} &= -sin\theta\hat{e}_{X_0} + cos\theta\hat{e}_{Y_0}\end{aligned}$$

Let's plug this back into the acceleration equation:

$$\begin{aligned}\vec{a}_{C/F} &= \ddot{\theta}\hat{e}_{Z_0} \times d_{attach}\left(-sin\theta\hat{e}_{X_0} + cos\theta\hat{e}_{Y_0}\right) - \dot{\theta}^2 d_{attach}\left(-sin\theta\hat{e}_{X_0} + cos\theta\hat{e}_{Y_0}\right) \\ &= -\ddot{\theta}d_{attach}sin\theta\hat{e}_{Y_0} - \ddot{\theta}d_{attach}cos\theta\hat{e}_{X_0} + \dot{\theta}^2 d_{attach}sin\theta\hat{e}_{X_0} - \dot{\theta}^2 d_{attach}cos\theta\hat{e}_{Y_0} \\ &= -\left(\ddot{\theta}d_{attach}cos\theta - \dot{\theta}^2 d_{attach}sin\theta\right)\hat{e}_{X_0} - \left(\ddot{\theta}d_{attach}sin\theta + \dot{\theta}^2 d_{attach}cos\theta\right)\hat{e}_{Y_0}\end{aligned}$$

3.4 Equations of Motion

Let's plug the acceleration into Equations 4 and 5 (reaction wheel dynamics):

$$\begin{aligned}F_{rxn,y} &= F_{g,wheel} + m_{wheel}\vec{a}_{C/F} \cdot \hat{e}_{Y_0} \\ &= F_{g,wheel} - m_{wheel}\left(\ddot{\theta}d_{attach}sin\theta + \dot{\theta}^2 d_{attach}cos\theta\right) \\ F_{rxn,x} &= m_{wheel}\vec{a}_{C/F} \cdot \hat{e}_{X_0} \\ &= -m_{wheel}\left(\ddot{\theta}d_{attach}cos\theta - \dot{\theta}^2 d_{attach}sin\theta\right)\end{aligned}$$

And now let's substitute the reaction wheel dynamic equations into Equation 3 (sum of moments about frame), and simplify:

$$F_{rxn,x}d_{attach}cos\theta + F_{rxn,y}d_{attach}sin\theta + F_{g,frame}dsin\theta - M_{rxn} = I_{frame,A}\ddot{\theta}$$

$$-m_{wheel}\left(\ddot{\theta}d_{attach}cos\theta-\dot{\theta}^2d_{attach}sin\theta\right)d_{attach}cos\theta+$$
$$\left(F_{g,wheel}-m_{wheel}\left(\ddot{\theta}d_{attach}sin\theta+\dot{\theta}^2d_{attach}cos\theta\right)\right)d_{attach}sin\theta+F_{g,frame}dsin\theta-M_{rxn}=I_{frame,A}\ddot{\theta}$$

$$-m_{wheel}\ddot{\theta}d_{attach}^2cos\theta^2+m_{wheel}\dot{\theta}^2d_{attach}^2sin\theta cos\theta+F_{g,wheel}d_{attach}sin\theta-m_{wheel}\ddot{\theta}d_{attach}^2sin\theta^2-$$
$$m_{wheel}\dot{\theta}^2d_{attach}^2sin\theta cos\theta+F_{g,frame}dsin\theta-M_{rxn}=I_{frame,A}\ddot{\theta}$$

We can use a trigonometric identity to combine terms 1 and 4; terms 2 and 5 cancel, and we are left with the following:

$$-m_{wheel}\ddot{\theta}d_{attach}^2+F_{g,wheel}d_{attach}sin\theta+F_{g,frame}dsin\theta-M_{rxn}=I_{frame,A}\ddot{\theta}$$

Let's substitute in the equation for M_{rxn} and rearrange this to get our equation of motion for the system

$$\left(m_{wheel}d_{attach}+m_{frame}d\right)gsin\theta-I_{wheel,C}\ddot{\phi}=\left(I_{frame,A}+m_{wheel}d_{attach}^2\right)\ddot{\theta}$$

The first term represents the moment caused by the force of gravity on the frame and the reaction wheel. The second term represents the moment caused by acceleration of the reaction wheel (this is what we can control). The third term represents the kinematics of the frame + reaction wheel assembly.

4 Initial Mechanical Design

We wanted our design to be easily manufacturable at on-campus maker-spaces, so the frame and flywheel materials were chosen to be acrylic. Acrylic was readily available and is easy to laser cut relatively precisely to a CAD drawing, so we thought it would be a good fit. The frame is made from two acrylic plates and four threaded rods. The flywheel is also made of an acrylic plate with large nuts and bolts on the outer edge (this was our method of adding inertia). Additionally, the choice of nuts and bolts for the inertial elements of the flywheel allow for easily adjustable inertia parameters.

For an actuator, we wanted to use a brushed DC motor for its ease of control. The initial design did not contain mounting holes for the IMU sensor, but it was planned to have the wires from the IMU and the motor running out the back of the system to the power supply and the rest of the circuit (on a benchtop). After looking at some motors and potential mounting hubs, we got a preliminary idea of how the design would physically fit together and what components we would need. A rendering from the CAD is shown below:

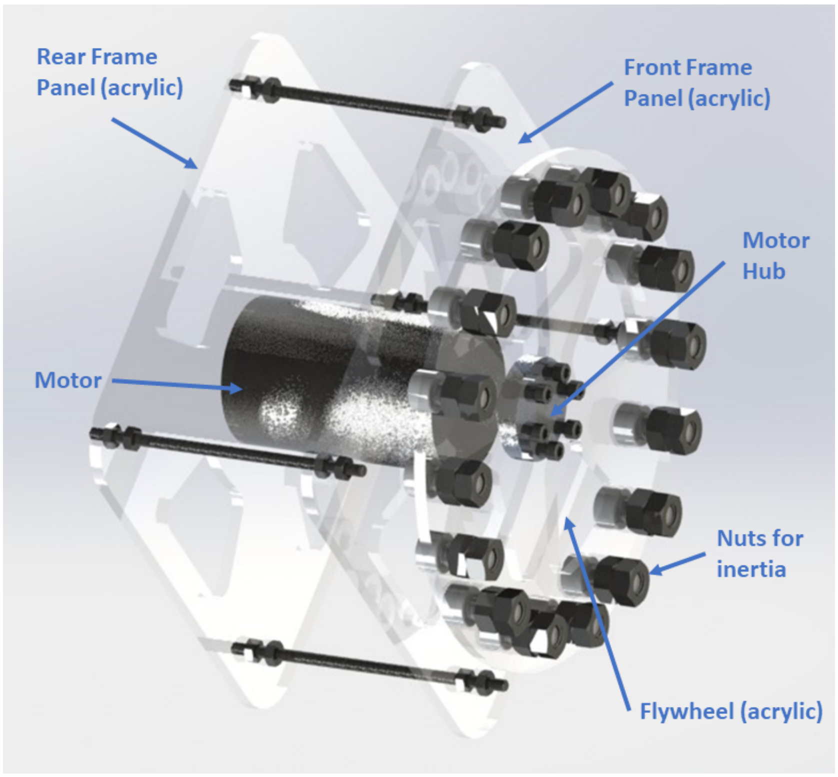


Figure 4: A 3D model of the system for parameter investigation

This model was used to get the mass and inertial properties of the frame and flywheel in order to determine specifications for the motor, shown below:

System values:

$d = 3.53[in] = 0.090[m]$
$d_{attach} = 3.53[in] = 0.090[m]$
$m_{frame} = 0.69[lbs] = 0.31[Kg]$
$m_{wheel} = 0.46[lbs] = 0.208[Kg]$
$I_{frame,B} = 1.35[lbs * in^2] = 4.0E - 4[Kg * m^2]$
$I_{wheel,C} = 1.88[lbs * in^2] = 5.5E - 4[Kg * m^2]$

5 Motor Selection

In selecting the motor, the specifications we were interested in were torque limits, speed limits, power limits, and cost. When thinking about what our system required and how to get these quantitative values, we started with the following ideas:

1. The motor is directly attached to the flywheel, so the motor torque is given by $I_{wheel,C}\ddot{\phi}$ and the motor speed is given by $\dot{\phi}$.

2. If the frame is falling, then the change in flywheel speed needs to react a torque larger than the torque due to gravity, in order to accelerate the frame in the opposite direction of its fall. In magnitude:

   $T = I_{wheel,C}\ddot{\phi} > (m_{wheel}d_{attach} + m_{frame}d)\, gsin\theta$

In practice, we will not want the frame to fall at a constant speed - we will want to apply a greater torque than the disturbance so that it slows the speed of the fall and causes the frame to rise. Knowing that the torque must be higher than this value we have chosen a safety factor of 2 for our design. This means that the motor's operating torque is 2x larger than required to stabilize the system from ± 5°. We can plug in the

$$T_{specification} = 2\left(m_{wheel}d_{attach} + m_{frame}d\right) gsin\theta$$

To determine the motor specifications, we used our settling time specification ($T_S = 2$ seconds), our set angle bounds (± 5°), and the above inequality for torque. If we assume that the maximum torque value is applied for the entire settling time in order to respond to a deviation of 5° , we can achieve a conservative estimate for our required motor speed.

$\ddot{\phi} * T_s = \dot{\phi}$ and $T = I_{wheel,C}\ddot{\phi}$. Rearranging, we can get the target flywheel angular velocity ($\dot{\phi}$ or ω)

$$\omega_{specification} = T * T_S / I_{wheel,C}$$

Now that we found the relationships for calculating our estimated required torque limit and required angular velocity limit, we started looking for motors. We first filtered by cost and voltage (up to 24V because we had access to a 24V power supply). Then we got a feel for what torque and speed ratings were available to us. Then, using the 3D model and the equations mentioned previously, we wanted to see the effect of changing our system parameters (particularly flywheel mass and inertia) on the torque and speed requirements. To guide us in selecting a mass and inertia of the flywheel, we made these graphs, which represent the representing the effect of increasing the mass and inertia of the flywheel:

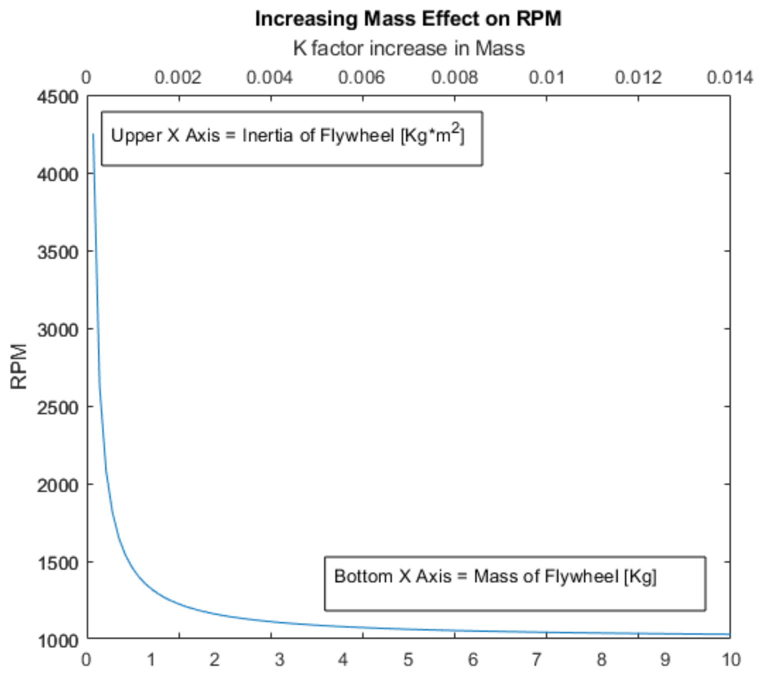


Figure 5: The effect of increasing flywheel mass and inertia on the RPM of the system

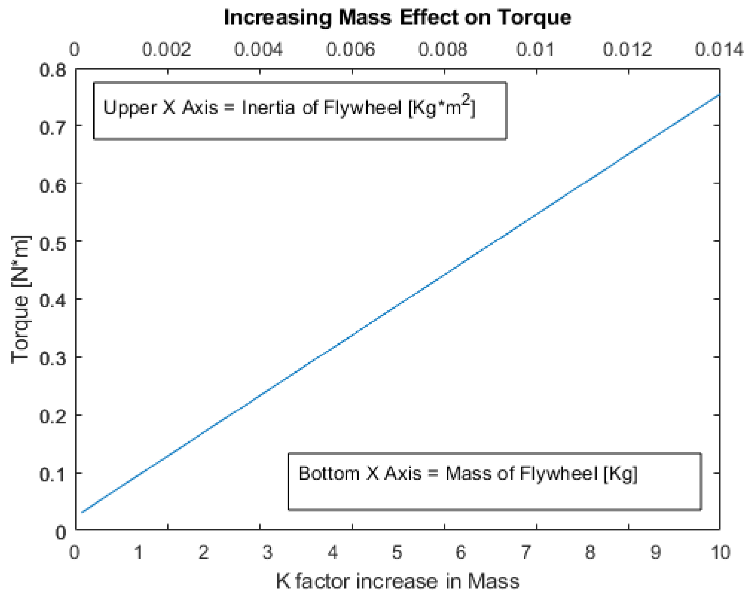


Figure 6: The effect of increasing flywheel mass and inertia on the torque required to stabilize the system

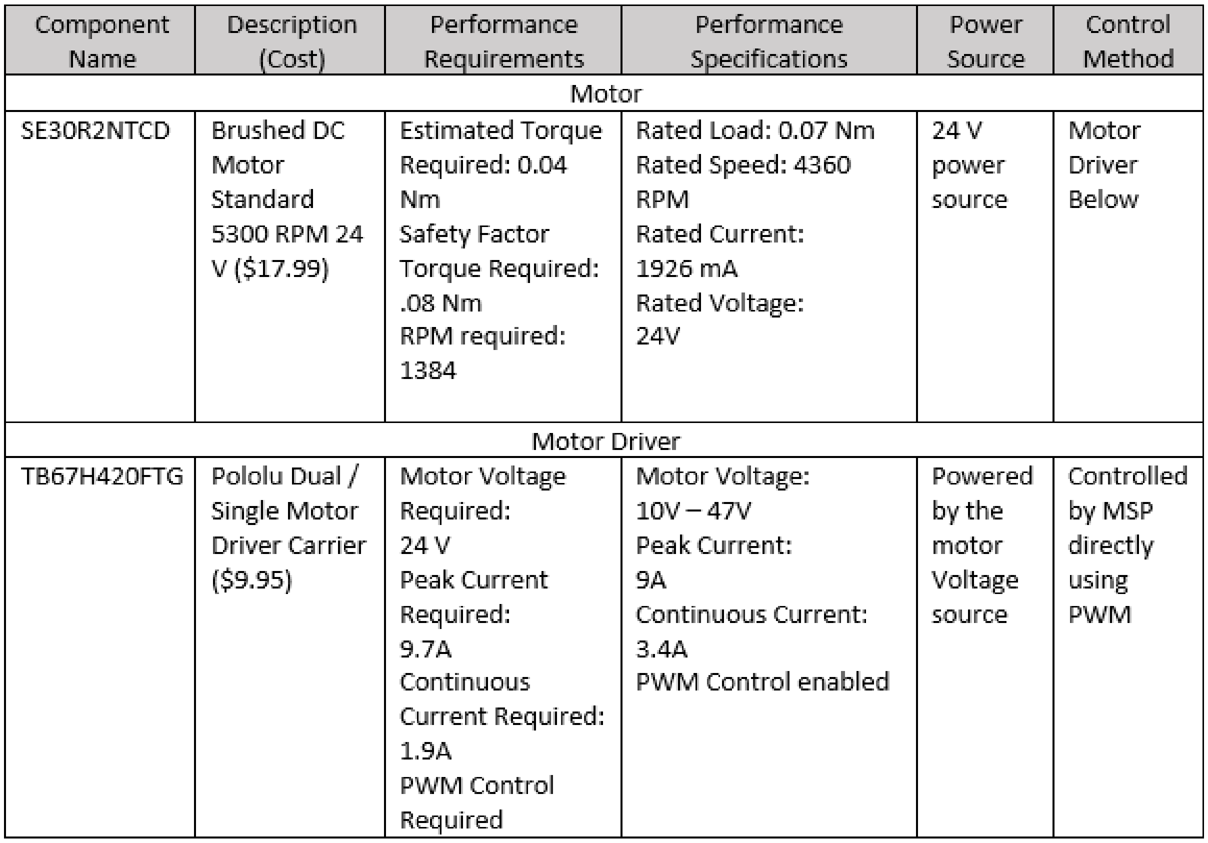
These graphs show that as flywheel inertia is increased, the maximum speed required of the motor significantly decreases. Thus, there is a great advantage to increasing the flywheel, as the cost in terms of torque only increases linearly. These graphs helped us select a flywheel mass and inertia that drove the torque and speed requirements to values that were available in our price range.

$T_{specification} = 0.08[N * m]$

$\omega_{specification} = 144.94[rad/s]$

$RPM_{specification} = 1384[RPM]$

With the values of motor torque ($0.08[N*m]$) and motor speed ($1384[RPM]$), we are able to choose a motor sufficient for our use case. The motor we selected is the SE30R2NTCD Brushed DC Motor from Digikey (`https://www.digikey.com/en/products/detail/nmb-technologies-corporation/SE30R2NTCD/6021451`). Additionally, we chose the following motor driver to help us with the implementation (`https://www.pololu.com/product/2999`). We also chose a motor mounting hub from Pololu to attach the flywheel to the motor shaft.



Component Name	Description (Cost)	Performance Requirements	Performance Specifications	Power Source	Control Method
Motor					
SE30R2NTCD	Brushed DC Motor Standard 5300 RPM 24 V ($17.99)	Estimated Torque Required: 0.04 Nm Safety Factor Torque Required: .08 Nm RPM required: 1384	Rated Load: 0.07 Nm Rated Speed: 4360 RPM Rated Current: 1926 mA Rated Voltage: 24V	24 V power source	Motor Driver Below
Motor Driver					
TB67H420FTG	Pololu Dual / Single Motor Driver Carrier ($9.95)	Motor Voltage Required: 24 V Peak Current Required: 9.7A Continuous Current Required: 1.9A PWM Control Required	Motor Voltage: 10V – 47V Peak Current: 9A Continuous Current: 3.4A PWM Control enabled	Powered by the motor Voltage source	Controlled by MSP directly using PWM

Table 1: Specifications for selected motor and driver

6 Sensor Selection

We wanted to use a gyroscope to control our system, as our variable of interest is θ which can be calculated be a numerical integral from the gyroscopes ω reading. The specifications that we were interested in were:

1. Range
2. Resolution
3. Bandwidth
4. Drift
5. Sampling Rate

6.1 Range

We started with the equation of motion from earlier:

$$(m_{wheel}d_{attach} + m_{frame}d)\, gsin\theta - I_{wheel,C}\ddot{\phi} = \left(I_{frame,A} + m_{wheel}d_{attach}^2\right)\ddot{\theta}$$

The maximum torque from the motor: $-I_{wheel,C}\ddot{\phi}$ was specified earlier to be 0.08 [Nm]. The maximum acceleration would occur if the left hand size of this equation is maximized. So we will consider the case where the first term $(m_{wheel}d_{attach} + m_{frame}d)\, gsin\theta$ is zero and the second term $-I_{wheel,C}\ddot{\phi}$ is maximized. This gives us the following relationship:

$$0.08[Nm] = \left(I_{frame,A} + m_{wheel}d_{attach}^2\right)\ddot{\theta}$$

Solving for $\ddot{\theta}$, we get the following:

$$\ddot{\theta} = \frac{0.08[Nm]}{I_{frame,A} + m_{wheel}d_{attach}^2} = \frac{0.08[Nm]}{I_{frame,B} + m_{frame}d^2 + m_{wheel}d_{attach}^2}$$

We can plug in the approximate physical properties from earlier:

$$\ddot{\theta} = \frac{0.08[Nm]}{4.0E-4[kgm^2] + 0.31[kg](0.090[m])^2 + 0.208[kg](0.090[m])^2} = 17,41[rad/s^2]$$

Using our settling time (2 seconds), we can integrate this value to determine the max angular velocity the system will experience. Our range spec needed to be higher than this value.

$$\omega = \dot{\theta} = 34.8[rad/s] = 1994[deg/s]$$

6.2 Resolution

For the resolution, we looked for 0.1% of the range.

6.3 Bandwidth

We did not have a quantitative requirement for bandwidth. We had an intuition that in a situation close to what our performance goals, the system would not be moving at any frequencies higher than 100Hz.

If we had to come up with a quantitative requirement, the process we had in mind was the following. Find the transfer function from the angular velocity of the frame to the reference angle. Then make a bode plot of this transfer function and see where the cutoff frequency is. The sensor bandwidth should be significantly higher than this frequency. Another aspect that plays into bandwidth is the sampling rate. The bandwidth should be much higher than the sampling rate as well.

6.4 Drift

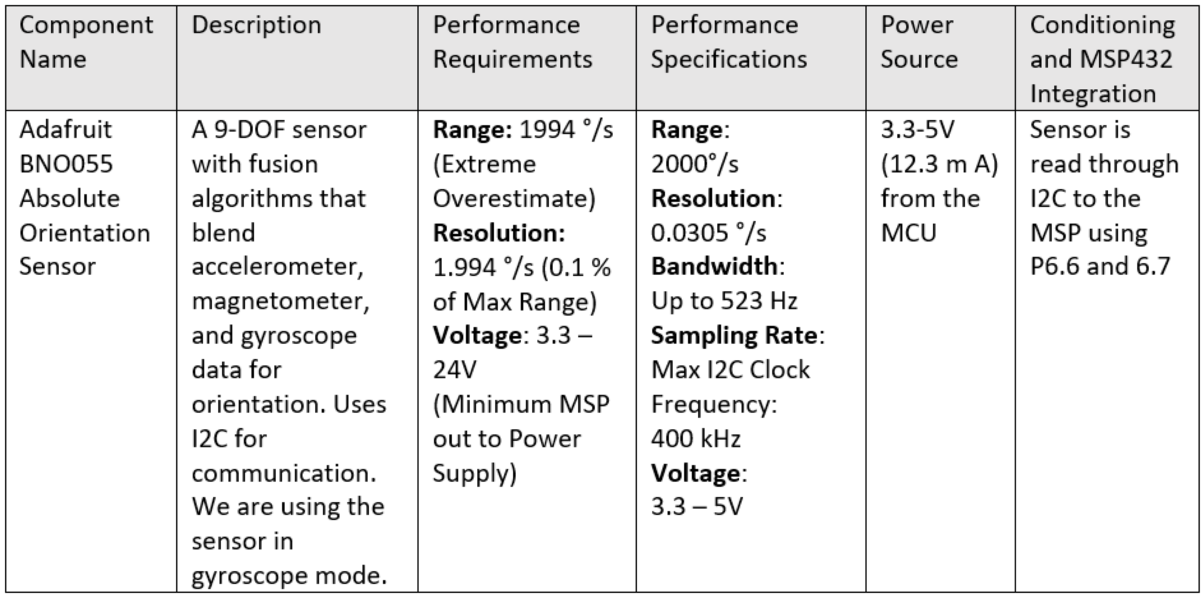
We also did not have a quantitative requirement for drift. However, if we would have been interested in one, the acceptable drift would have been driven by the maximum time that we needed to balance for.

6.5 Sampling Rate

We also did not have a quantitative requirement for sampling rate, as this would be driven by the controller. There would be a minimum sampling rate required in order to have a stable system with the digital implementation of our continuous-time controller.

6.6 Selection

With all of these specifications in mind, we chose the BNO055 Absolute Orientation Sensor from Adafruit (`https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/arduino-code#`). The specifications are shown below. It satisfies all of our requrements, has onboard signal processing, and can communicate via UART or I2C. The maximum possible sampling rate could be back-calculated by the max I2C clock frequency and the register size for the data we need.



Component Name	Description	Performance Requirements	Performance Specifications	Power Source	Conditioning and MSP432 Integration
Adafruit BNO055 Absolute Orientation Sensor	A 9-DOF sensor with fusion algorithms that blend accelerometer, magnetometer, and gyroscope data for orientation. Uses I2C for communication. We are using the sensor in gyroscope mode.	**Range:** 1994 °/s (Extreme Overestimate) **Resolution:** 1.994 °/s (0.1 % of Max Range) **Voltage**: 3.3 – 24V (Minimum MSP out to Power Supply)	**Range**: 2000°/s **Resolution**: 0.0305 °/s **Bandwidth**: Up to 523 Hz **Sampling Rate**: Max I2C Clock Frequency: 400 kHz **Voltage**: 3.3 – 5V	3.3-5V (12.3 m A) from the MCU	Sensor is read through I2C to the MSP using P6.6 and 6.7

Table 2: Specifications for selected IMU

During initial testing of reading this sensor from the MSP432 microcontroller, we were unable to reliably read the gyroscope and acceleration registers. As a result, we chose to use an Arduino to read this sensor and then communicate the readings to the MSP432. This was desirable because their was an existing library for Arduino to read this sensor.

7 Electrical Design

Once the sensor and motor were selected, we could design the electrical schematic for our system. The components were an MSP432 microcontroller, an Arduino, a DC motor, a DC motor driver, a power supply, and an IMU. The Arduino and MSP were powered by a laptop, and the DC Motor was powered by a bench power supply. The IMU used I2C to communicate to the Arduino, and received power from the Arduino. The MSP432 used UART to communicate with the Arduino, and sent PWM commands to the motor driver to modulate the speed.

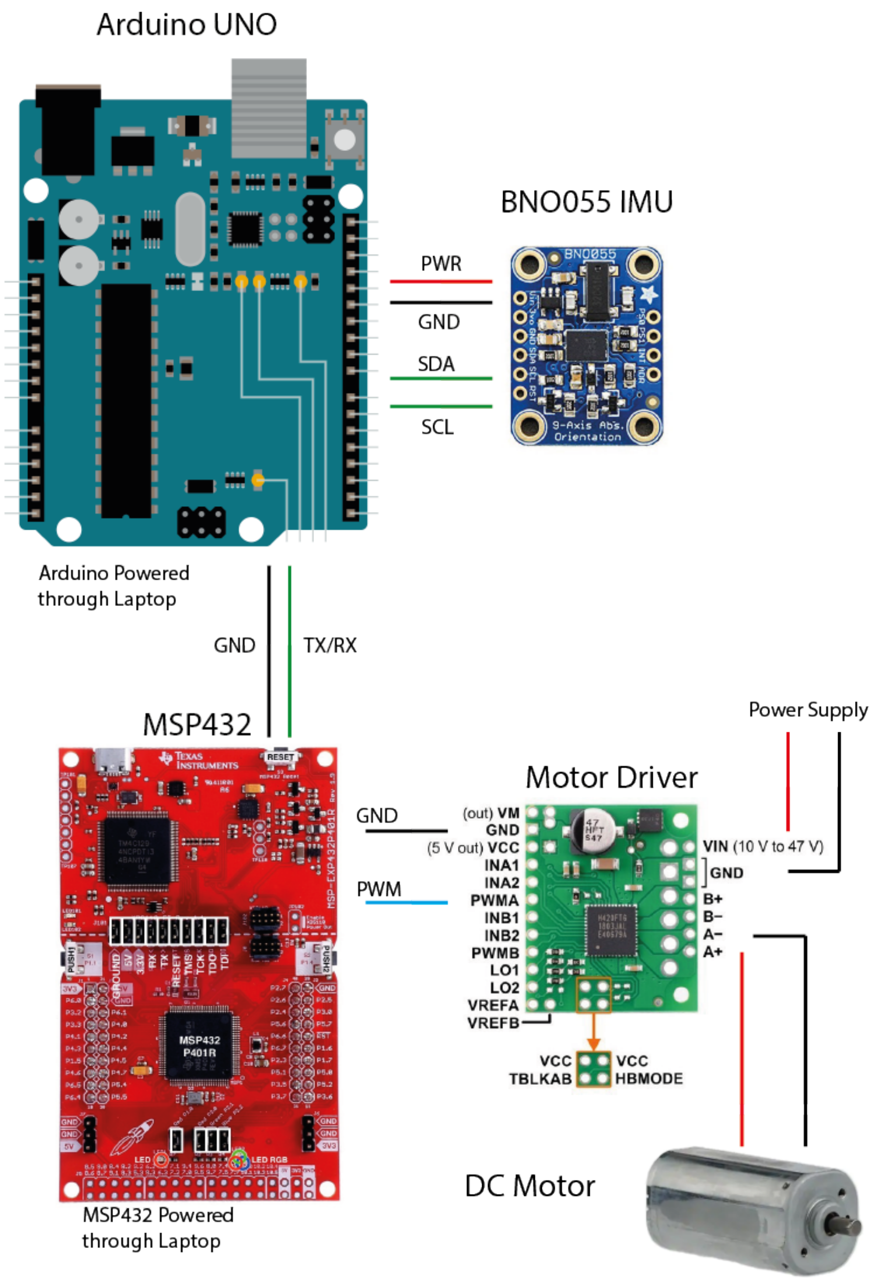


Figure 7: System Wiring Diagram

8 Final Design and Prototype

After deciding on the motor and the sensor, we were able to finalize the CAD model. We added mounting holes for IMU to the back acrylic panel of the cube frame, for nuts and bolts. We added a mounting hole pattern on the front acrylic frame and a press-fit cut to the back acrylic frame for the motor. We added the proper mounting holes for the hub to the flywheel. We also changed the flywheel to wood instead of acrylic - we had a little mishap. During one of our tests, we accidentally flipped the wiring for the duty cycle sent to the motor, so rather than 10% of full speed, the motor was spinning at 90% of full speed. After a few seconds the flywheel actually shattered into many pieces, so we decided to make it out of wood to prevent that type of failure again.

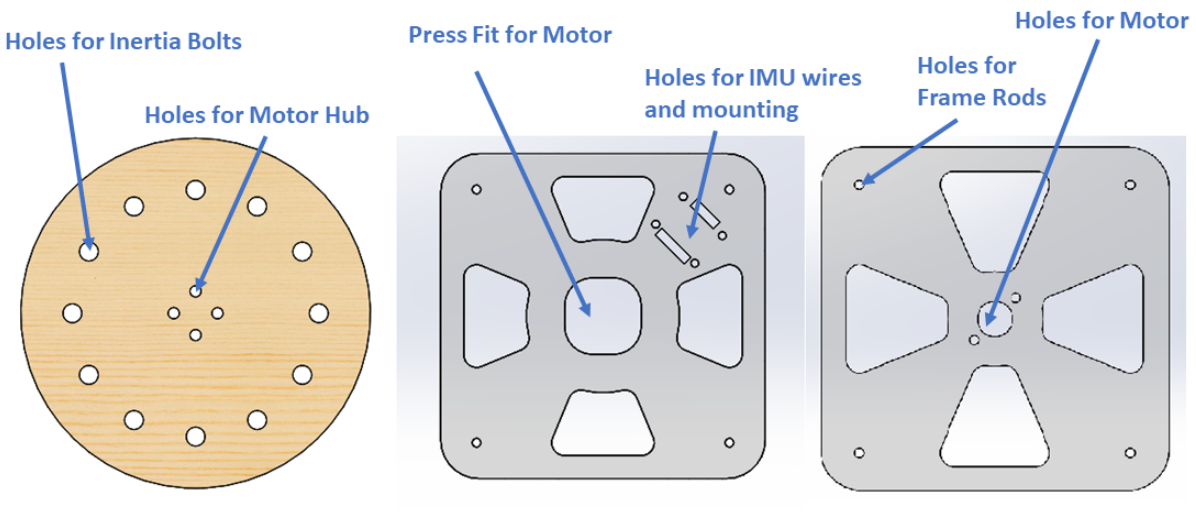


Figure 8: Design of Panels

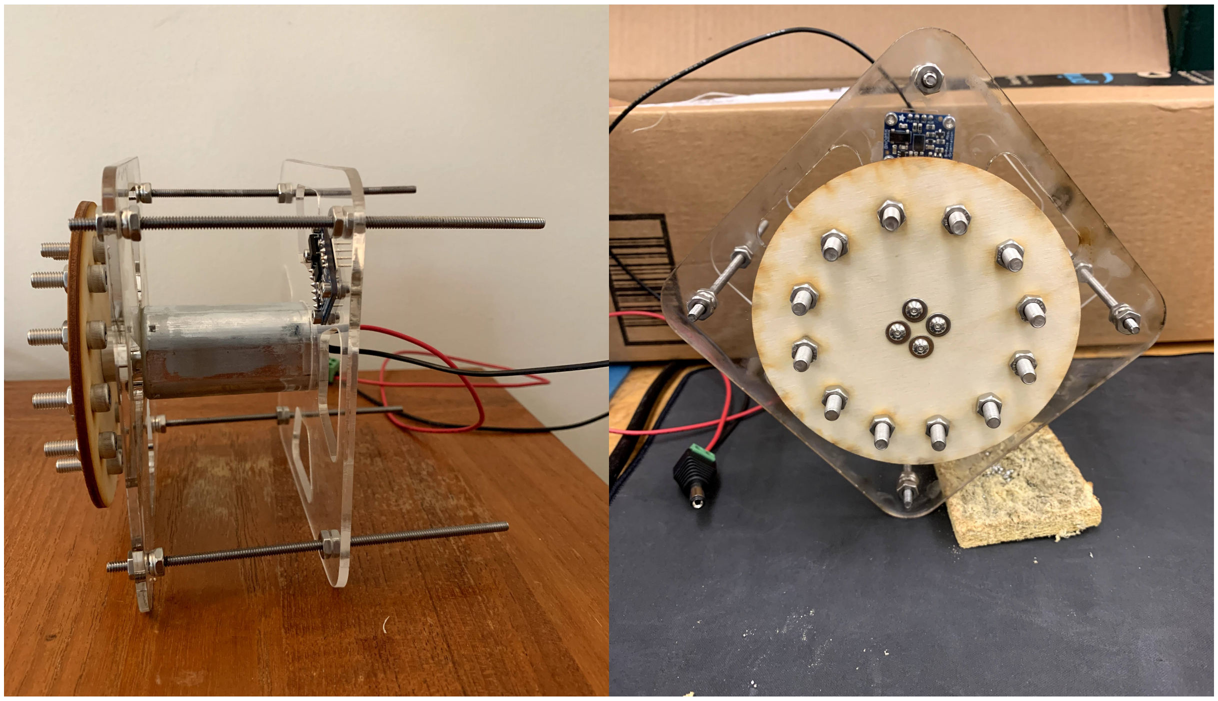


Figure 9: System Prototype

9 Controller Design

9.1 Controller Specifications

- Settling time of 2 seconds for a 5 degree disturbance
- Reject step disturbances up to 5 degrees

9.2 System Transfer Function

The goal here is to find $\frac{\Theta(s)}{M_{rxn}(s)}$ Let's start with the equation of motion.

$$(m_{wheel}d_{attach} + m_{frame}d)\, gsin\theta - M_{rxn} = \left(I_{frame,A} + m_{wheel}d_{attach}^2\right)\ddot{\theta}$$

Now let's convert to the Laplace Domain. We will assume θ can only take on small values, so $\sin\theta \approx \theta$.

$$(m_{wheel}d_{attach} + m_{frame}d)\, g\Theta(s) - M_{rxn}(s) = \left(I_{frame,A} + m_{wheel}d_{attach}^2\right)s^2\Theta(s)$$

$$\frac{\Theta(s)}{M_{rxn}(s)} = -\frac{1}{(I_{frame,A} + m_{wheel}d_{attach}^2)\,s^2 - (m_{wheel}d_{attach} + m_{frame}d)\,g}$$

9.3 Closed Loop Block Diagram

Our overall closed loop system diagram will look like this. We have our plant: $\frac{\Theta(s)}{M_{rxn}(s)}$ from before shown in the dashed lines, and we will have a unity gain feedback control loop.

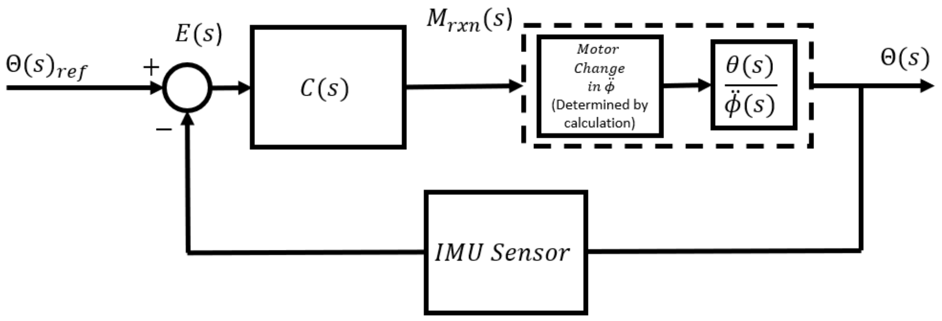

$\Theta(s)_{ref}$
$+$
$-$
$E(s)$
$C(s)$
$M_{rxn}(s)$
Motor Change in $\ddot{\phi}$ (Determined by calculation)
$\frac{\theta(s)}{\ddot{\phi}(s)}$
$\Theta(s)$
IMU Sensor

Figure 10: Block Diagram of Closed Loop System

9.4 Design of $C(s)$

To design C(s), we will use a step input for $\Theta_{ref}(s)$ rather than modeling a disturbance directly. The idea here is that before time $t = 0$, the system will be tilted (-2°) and then at time $t = 0$ the commanded input will be 0°.

The transfer function $\frac{\Theta(s)}{\Theta_{ref}(s)}$ is given by the following, where $G(s)$ represents the system transfer function $\frac{\Theta(s)}{M_{rxn}(s)}$.

$$\frac{\Theta(s)}{\Theta_{ref}(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)}$$

The closed loop poles, given by $1 + C(s)G(s)$, are the same regardless of whether the input is coming from $\Theta_{ref}(s)$ or from a disturbance at another point in the loop. The closed loop poles control the steady state response of the system and the closed loop zeros govern the transient response. If we are interested primarily in the stability, then we can design a controller for the $\Theta_{ref}(s)$ and the stability performance would improve regardless of the indirect representation of our disturbance input.

9.4.1 Response Requirements

The response to a step input of 5°should have the following characteristics:

- Settling time of less than 2 seconds
- Maximum overshoot of 50%

9.5 PID Controller Requirements

The system we are using is inherently unstable and equivalent to and inverted pendulum system in many ways. Examining the root locus of the plant function $\theta(s)/M_{rxn}(s)$ shows us that the system cannot be brought to anything other than marginally stable (the two poles lie on the imaginary axis).

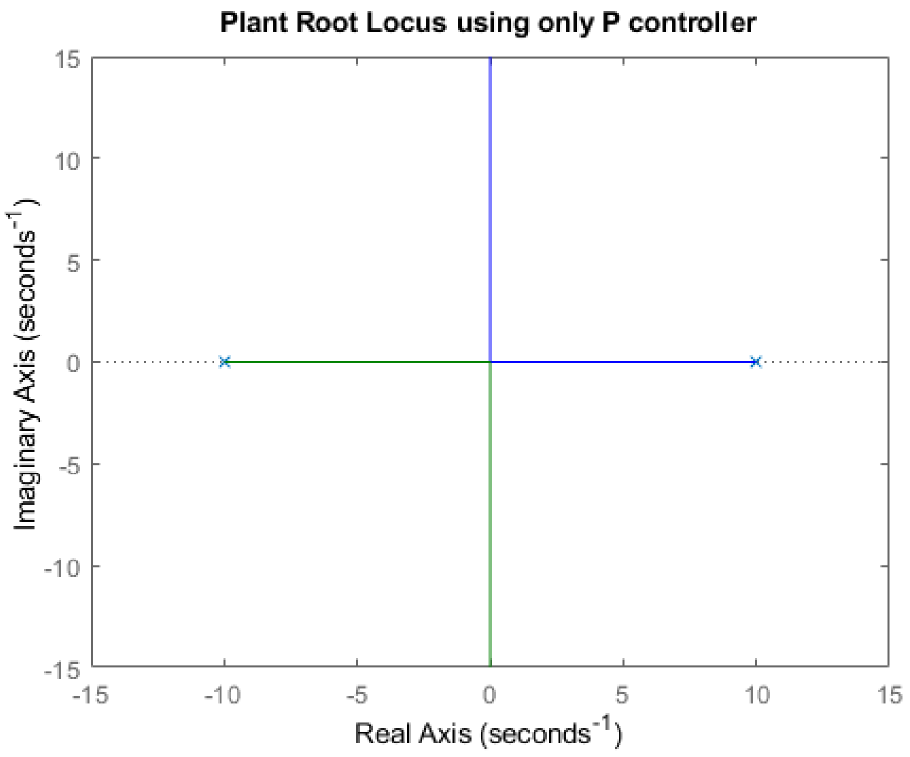


Figure 11: System Root Locus using Only P controller

This is expected as classical inverted pendulums are not theoretically stable if a P controller is used. Next we examined the system's root locus while using a PD controller. The plot shows that with a sufficiently high K value, the system is stable (no right hand poles).

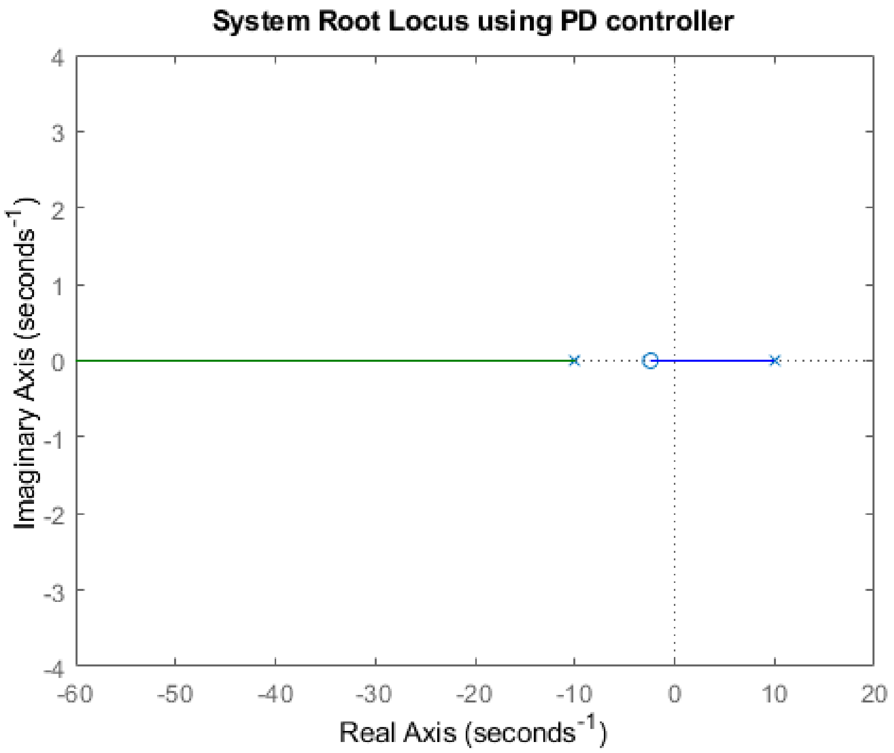


Figure 12: System Root Locus using PD controller

The figure below shows the system step response. It is important to note that the system does converge to a final value, but there is steady state error.

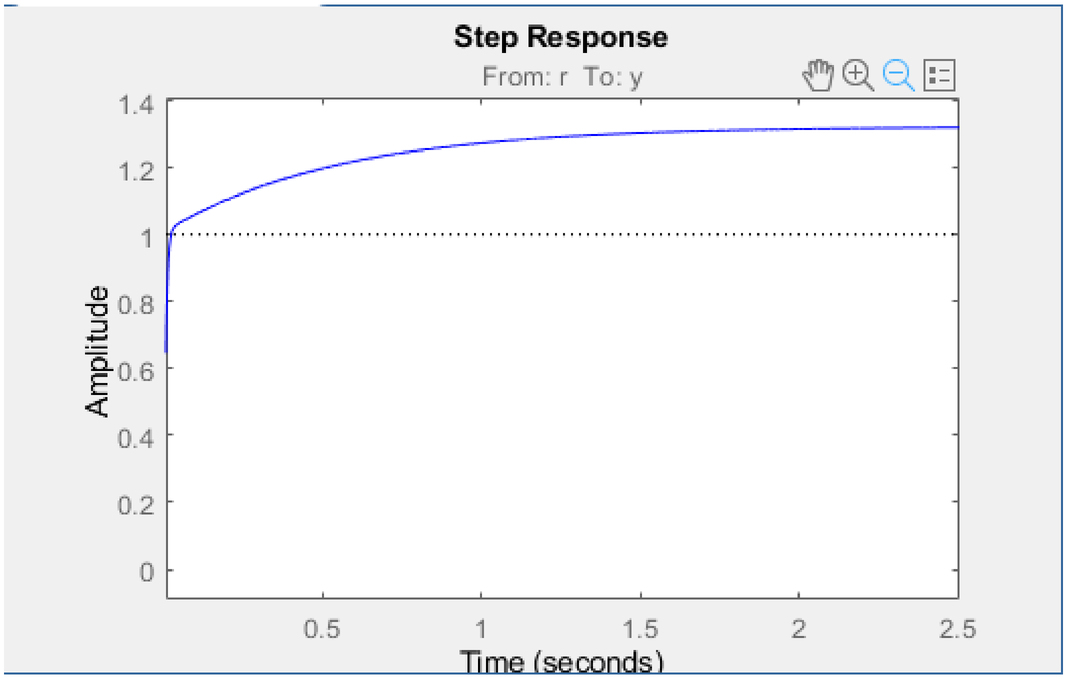


Figure 13: System Step Response using PD controller

However, when taking into account system implementation considerations, the realized system needs more than a PD controller. The way our system generates torque is by changing flywheel RPM. If our motor was ideal, it would have no RPM limit and therefore could keep the system stable even with steady-state error. If there was steady-state error, however, the system would need to generate a constant torque. In order to do this, the system would have to constantly accelerate the flywheel. Since our motor does have a maximum speed, we need a integral controller to reduce steady state error and keep our commanded RPM change from saturating our motor speed limit. As shown in the figure below, the system is still able to be stabilized with a sufficiently high K value.

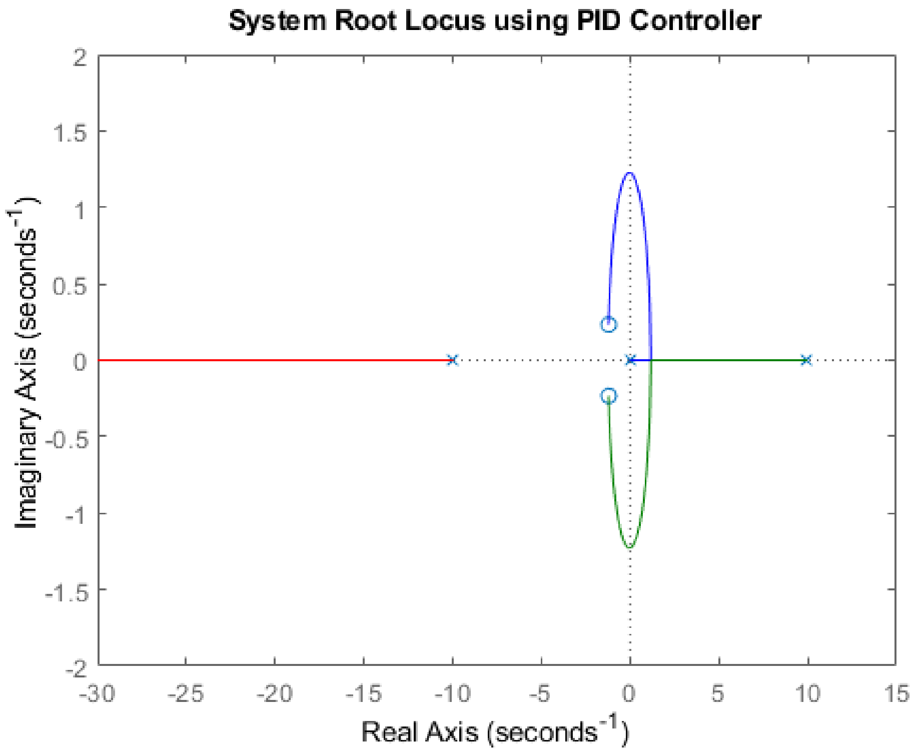


Figure 14: System Root Locus using PID controller

The figure below shows the system step response when using a PID controller. The system has zero steady state error but shorter rise time and larger overshoot caused by the addition of integral control.

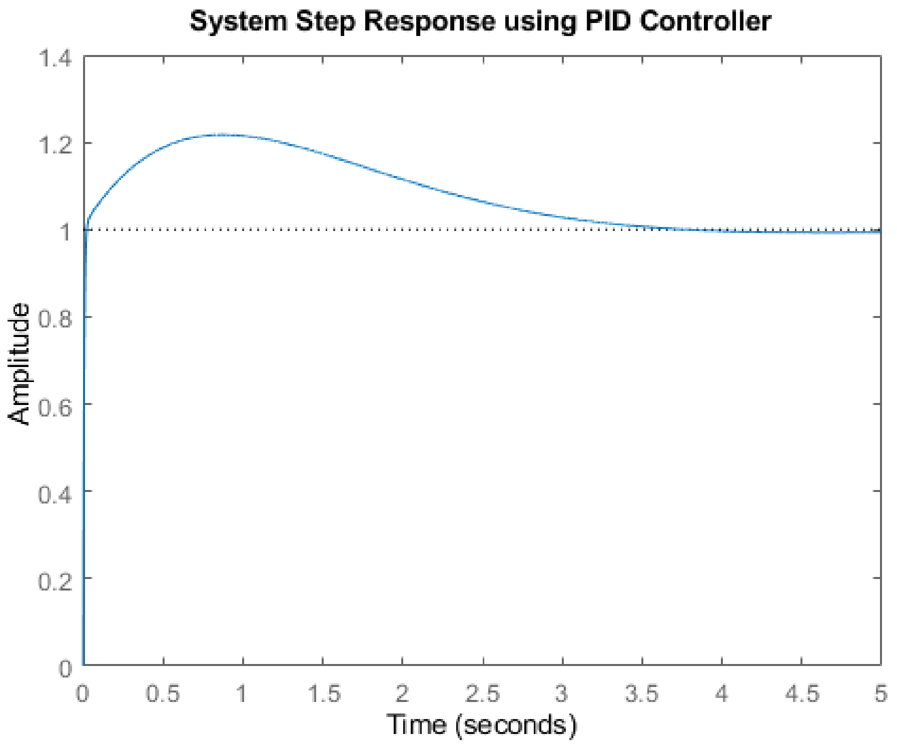


Figure 15: System Step Response using PID controller

9.6 Design using MATLAB SISO tool

Knowing that our system requires the creation of a PID controller, we used MATLAB's SISO tool to help us choose our gains for the system.

Here is the controller we designed:

$$C(s) = \frac{0.776(s^2 + 2.413 * s + 1.51)}{s} \tag{7}$$

10 Digital Implementation of Controller

Our implementation of the controller involved separating it out into the corresponding P, I, and D gains, and then applying those separately before summing the controller output together. The error in angle was calculated by taking the difference between the current angle estimate (from the IMU) and the target angle (0°). The integral in error was calculated by numerically integrating the error. The control loop rate was used to get the time multiplier for this numerical integral. The derivative in error was calculated by taking the numerical derivative of the error, using the same time multiplier as the integral. The integral term was clamped such that if the integral component of the controller was saturated, the integration would not continue. This prevents the integral component for causing overshoot for an unnecesarily extended period of time.

In order to determine the control loop rate, we used trial and error. If we had time for a more thorough approach, our thought process was to do a Bilinear transform of the controller to bring it to the z-domain. Then we could simulate different control loop rates in MATLAB in order to find a stable one.

11 Controller and System Testing

Once the controller was fully implemented, we started trying it and tuning the PID gains. We recorded the outputs of the controller and the angle estimate over time so we could gain more insight into what was occurring each time.

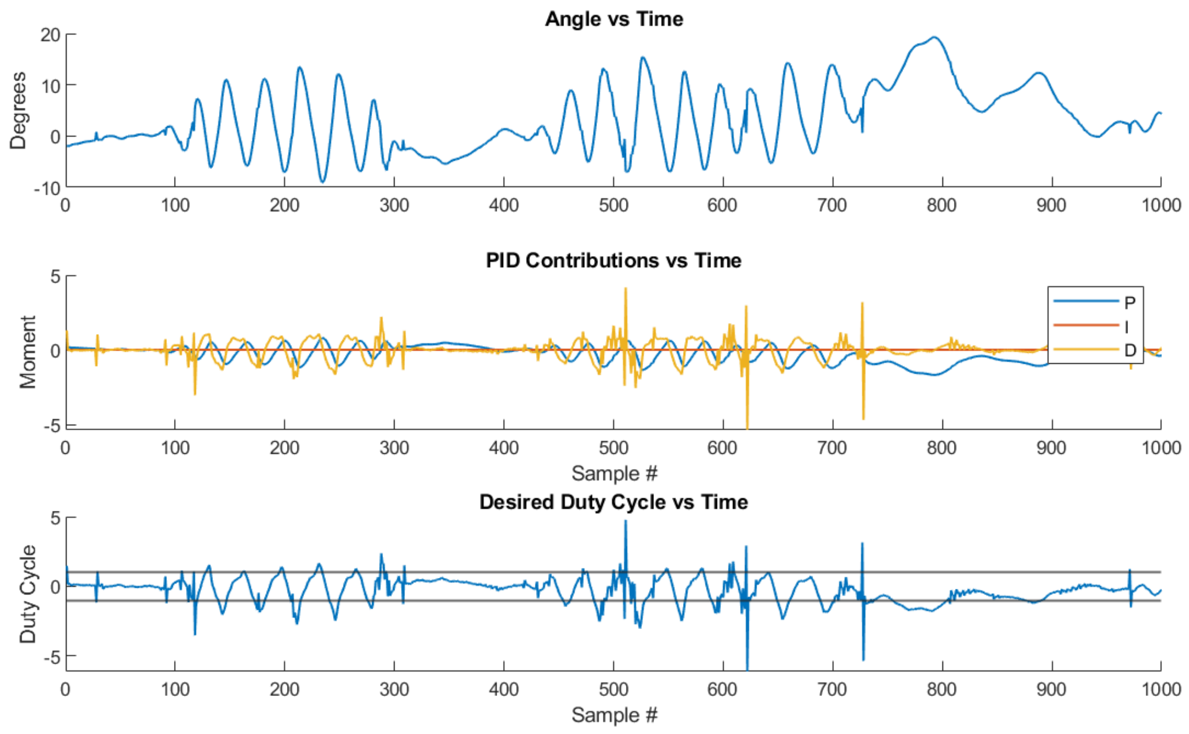


Figure 16: Data Collection while Tuning

The above figure shows the estimated angle over time, as well as the P, I, and D contributions over time. The bottom graph shows the commanded duty cycle (indirectly the commanded speed), where the horizontal black lines are 1 and -1, corresponding to the maximum limits. Positive numbers indicate one direction and negative numbers indicate the other direction. As you can see, the angle vs time graph shows some sudden blips. The derivative gain contribution is also fairly noisy, and as a result the commanded speed is also fairly noisy.

A video of us testing the system is shown here. We were unable to get the controller working by the time the project was due, but we were really excited with how much we learned and were able to accomplish!

12 Potential Improvements

12.1 IMU Reading

Since the IMU was read by the Arduino first and then communicated to the MSP432, the process for obtaining an angle reading was not ideal. If we were to continue working on this, we would try to figure out how to read the IMU registers directly on the MSP432. This was an issue because it took a lot of troubleshooting to figure out the implementation of communicating the angle reading as ASCII characters through UART, which still was not reliable. Those blips in the angle over time graph are related to errors in the transmission from the Arduino to the MSP432.

12.2 Angle Estimation

The angle estimation we ended up using did not involve the actual gyroscope reading, and instead used the accelerometer. We used the direction of the gravity vector to output an angle. However, this is not ideal because the accelerometer is not only responding to the force of gravity, which is constant. There is also the higher frequency accelerations from the motion of the frame. When trying to estimate the angle using a trigonometric function of the acceleration in the vertical and horizontal direction, these higher frequency accelerations contribute to noise in the angle reading. Ideally if we were using an IMU, we would use some sort of fusion approach with the gyroscope and accelerometer, like a complementary filter or others.

12.3 Derivative Control

The derivative control formula we used was really noisy, as numerical derivatives often are. Ideally we would use the gyroscope data from the IMU, as the derivative of the angle could be directly measured. The reason we did not do this is that we ran out of time. It took us a while to get just the accelerometer data in a place where we could communicate to the MSP432, and after that we didn't have time to also get the gyroscope data.

12.4 Controller Tuning

If we made all the above improvements, we think the system would be easier to control and more stable. From the video of our progress, you can see that the system was oscillating due to the PD controller. With more stable angle estimation and more smooth controller outputs, we think it would be much easier to tune the controller to work.