

DEVELOPMENT OF A PROTOTYPE BALL-AND-PLATE BALANCING  
PLATFORM

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Mechanical Engineering

by

Scott Mangin

March 2022

© 2022  
Scott Mangin  
ALL RIGHTS RESERVED



## COMMITTEE MEMBERSHIP

TITLE: Development of a Prototype Ball-and-  
Plate Balancing Platform

AUTHOR: Scott Mangin

DATE SUBMITTED: March 2022

COMMITTEE CHAIR: William R. Murray, Ph.D.  
Professor of Mechanical Engineering

COMMITTEE MEMBER: Charlie T. Refvem, M.S.  
Lecturer of Mechanical Engineering

COMMITTEE MEMBER: John R. Ridgely, Ph.D.  
Professor of Mechanical Engineering

## ABSTRACT

### Development of a Prototype Ball-and-Plate Balancing Platform

Scott Mangin

Ball-and-plate balancing platforms have been utilized throughout academia to further understanding of nonlinearities that can occur when applying control algorithms to nonholonomic and underactuated systems. The objective of this thesis is to build upon an existing ball-and-plate balancing platform used in the Intro to Mechatronics class and create a robust platform system that can be utilized by future students to test various controller designs derived from MATLAB/Simulink<sup>®</sup>. The ball-and-plate platform design uses a myriad of sensors to track the system components in real time: a resistive touch panel is used to track the position of the ball on the plate, an inertial measurement unit is used to track the orientation of the top plate, and capacitive incremental encoders attached to the brushless-DC gimbal motors are used to both track the orientation of the motor actuation arms and commutate the motors. The gimbal motors are driven using the open-source ODrive motor driver, which receives torque commands from a separate STM32 microcontroller. The STM32 microcontroller aggregates and processes the data from the touch panel and IMU, and it acts as a “middle-man” for communication between the ODrive and MATLAB/Simulink<sup>®</sup> model running on a host PC. The platform successfully handles communications between the host PC, STM32, and ODrive at a rate of 200 Hz. The platform also incorporates a serial user interface that allows for fine position control of the motor arms for zeroing the top plate before each test.

## TABLE OF CONTENTS

	Page
LIST OF TABLES . . . . .	viii
LIST OF FIGURES . . . . .	ix
LIST OF ABBREVIATIONS . . . . .	xii
CHAPTER	
1 Background . . . . .	1
1.1 Literature Review . . . . .	1
1.1.1 Contemporary Research . . . . .	1
1.2 Types of Ball-and-Plate Platforms . . . . .	4
1.2.1 Stewart Platform . . . . .	4
1.2.2 U-Joint/Ball-Joint Platform . . . . .	5
1.3 Capturing System Orientation . . . . .	6
1.3.1 Top Plate Orientation . . . . .	6
1.3.2 Ball Position Measurement . . . . .	7
1.4 Prior Ball-and-Plate Platform Design . . . . .	8
2 System Definition and Control Structure . . . . .	10
2.1 System Definition . . . . .	10
2.2 System Functional Overview . . . . .	12
2.2.1 ODrive Controller Design . . . . .	13
3 Electronics . . . . .	15
3.1 Electronics Overview . . . . .	15
3.1.1 Microcontroller . . . . .	15
3.1.2 Touch Panel . . . . .	16

3.1.3	Motor . . . . .	18
3.1.4	Motor Encoder . . . . .	20
3.1.5	Motor Driver . . . . .	21
3.1.6	Inertial Measurement Unit . . . . .	23
3.1.7	Break Beam Sensors . . . . .	24
4	Mechanical Design . . . . .	25
4.1	Design Overview . . . . .	25
4.1.1	Top Plate . . . . .	26
4.1.2	Touch Panel Spacers . . . . .	27
4.1.3	U-Joint . . . . .	28
4.1.4	Bottom Plate . . . . .	29
4.1.5	Motor Mount Assembly . . . . .	30
4.1.6	Endstop Bracket . . . . .	31
5	Software Design . . . . .	33
5.1	Software Overview . . . . .	33
5.2	STM32 Software Overview . . . . .	33
5.2.1	Touch Panel Class . . . . .	34
5.2.2	IMU Class . . . . .	40
5.2.3	Communication Structure . . . . .	42
5.2.4	User Interface Function . . . . .	43
5.2.5	Data Collection . . . . .	45
5.3	ODrive Software . . . . .	45
5.3.1	ODrive Firmware Configuration . . . . .	46
5.3.2	ODrive Motor Control . . . . .	47
5.3.3	ODrive Motor Driver Firmware Development . . . . .	48

5.4	Host PC Firmware . . . . .	49
5.4.1	MATLAB/Simulink <sup>®</sup> . . . . .	49
5.4.2	Limitations of Simulink <sup>®</sup> Integration . . . . .	51
6	Hardware Testing . . . . .	52
6.1	Platform Requirements Overview . . . . .	52
6.2	Motor Performance Testing . . . . .	52
6.2.1	Torque Step Response . . . . .	53
6.2.2	Torque Constant Testing . . . . .	54
6.2.3	Motor Torque Testing - Gimbal Mode . . . . .	56
6.2.4	Motor Torque Testing - Current Control Mode . . . . .	58
6.3	Software Timing Characterization . . . . .	59
6.3.1	Communication Timing . . . . .	60
6.3.2	IMU Timing . . . . .	64
6.3.3	Touch Panel Timing . . . . .	65
7	Conclusions and Future Work . . . . .	66
	BIBLIOGRAPHY . . . . .	68
	APPENDICES	
A	Wiring Diagram . . . . .	69
B	Platform Part Drawings . . . . .	76
C	Torque Testing Data . . . . .	87

## LIST OF TABLES

Table		Page
2.1	System Parameters from Schematics, Previous Thesis . . . . .	12
3.1	List of Considered Motors . . . . .	19
4.1	List of Mechanical Design System Components . . . . .	26
5.1	DMA Allocation and Priority. . . . .	43
6.1	USB Response Overview and Timings. . . . .	62

## LIST OF FIGURES

Figure	Page
1.1 Platform Design from Rensselaer Polytechnic Institute [2]. . . . .	2
1.2 Platform Design from Collaboration [7]. . . . .	3
1.3 Platform Design from Inha University [3] . . . . .	4
1.4 Yuan Tseh Lee Array Stewart Platform. [1] . . . . .	5
1.5 Current Ball-and-Plate Platform Design. . . . .	9
2.1 Ball-and-Plate Balancing Platform Coordinate System. . . . .	11
2.2 Ball-and-Plate Balancing Platform 2D System - XZ plane. . . . .	11
2.3 Ball-and-Plate Balancing Platform 2D System - YZ plane. . . . .	12
2.4 Ball-and-Plate Platform Internal Block Diagram. . . . .	13
2.5 ODrive Cascaded Controller Design [5]. . . . .	14
3.1 STM32F411CE Microcontroller Development Board. . . . .	16
3.2 AMTouch Resistive Touch Panel. . . . .	17
3.3 Example of Reading Ball Position on the X Axis. . . . .	18
3.4 G60 Motor from T-Motor. . . . .	20
3.5 Different Configurations of the AMT-113 Encoder. . . . .	21
3.6 Assembled ODrive Board. . . . .	22
3.7 BNO055 Adafruit Break-out Board from Adafruit. . . . .	24
3.8 IR Break Beam Sensors from Adafruit. . . . .	24
4.1 Render of Ball-and-Plate Platform SolidWorks Assembly . . . . .	25
4.2 Exploded View of Top Plate Assembly. . . . .	27

4.3	CAD Render of touch panel Spacers. . . . .	28
4.4	Exploded View of U-Joint Design. . . . .	29
4.5	CAD Render of ODrive Mounting Bracket. . . . .	30
4.6	CAD Render of Motor Mount Assembly. . . . .	31
4.7	CAD Render of Break Away Mounting Bracket. . . . .	32
5.1	Touch Panel Class Object. . . . .	35
5.2	Touch Panel Finite State Machine. . . . .	36
5.3	Touch Panel Filter Flowchart. . . . .	37
5.4	Filtered Versus Unfiltered Touch Panel Data. . . . .	39
5.5	IMU Class Object. . . . .	40
5.6	IMU Finite State Machine. . . . .	41
5.7	UI Finite State Machine. . . . .	44
5.8	Platform Ball Balanced Using Bubble Level. . . . .	44
5.9	$I_q$ Current Measurement Noise Comparison. . . . .	48
5.10	Simulink <sup>®</sup> Scope Output of Missed CPU Ticks. . . . .	50
5.11	Execution Order of Simulink <sup>®</sup> Blocks. . . . .	51
6.1	Step Response, Torque Command in Current Control Mode. . . . .	53
6.2	Torque Test Experimental Setup. . . . .	54
6.3	Experimental Torque Constants for Axis 0 Motor. . . . .	55
6.4	Experimental Torque Constants for Axis 1 Motor. . . . .	55
6.5	Percent Error of Torque Output from Commanded Torque, Gimbal Mode, Axis 0. . . . .	56
6.6	Percent Error of Torque Output from Commanded Torque, Gimbal Mode, Axis 1. . . . .	57
6.7	Stall $I_q$ Output in Gimbal Mode, ODrive. . . . .	58



6.8	Percent Error of Torque Output from Commanded Torque, Current Control, Axis 0. . . . .	59
6.9	Percent Error of Torque Output from Commanded Torque, Current Control, Axis 1. . . . .	59
6.10	Time Duration Between Host PC USB Messages. . . . .	61
6.11	Sections of Response Upon Receiving USB Message. . . . .	62
6.12	Comparison Between Baudrates. . . . .	63
6.13	Cycle Time Between I <sup>2</sup> C Messages. . . . .	64
6.14	Example of I <sup>2</sup> C Setup Time Duration. . . . .	65

## LIST OF ABBREVIATIONS

ADC	Analog-to-Digital Converter
BLDC	Brushless Direct Current Motor
CPU	Central Processing Unit
DMA	Direct Memory Access
DOF	Degree of Freedom
GPIO	General Purpose Input/Output
HAL	Hardware Abstraction Layer
I <sup>2</sup> C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
MCU	Microcontroller Unit
PID	Proportional-Integral-Derivative Controller
PLA	Polylactic Acid (Filament)
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus

## Chapter 1

### BACKGROUND

#### 1.1 Literature Review

While not implemented in many practical applications, ball-and-plate platforms are useful to study and demonstrate the effects of control algorithms in controlling inherently unstable systems. Being able to experimentally test control algorithms on unstable systems is especially important in understanding nonlinearities that can occur when applying control algorithms to nonholonomic and underactuated systems (such as a ball-and-plate platform). The ball-and-plate platform is a topic that is heavily explored in academia, with many examples of various kinds of ball-and-plate platform construction and objectives. The subsequent sections detail a portion of the research and development on ball-on-plate platform systems.

##### 1.1.1 Contemporary Research

In 2002, professors and mechatronics students at Rensselaer Polytechnic Institute in Troy, NY constructed a ball-on-plate balancing system intended to be “better, cheaper, quicker” than other contemporary designs [2]. The overall purpose of the project was to illustrate the mechatronics principles of balancing mathematical modeling and experimental validation to the students within their mechatronics courses. Figure 1.1 showcases their ball-and-plate platform design. The platform controlled the orientation of the plate through the use of a gimbal ring and two DC motors with arm linkages attached close to the point of rotation. The platform captured the orientation of the top plate through optical encoders attached to each motor and the

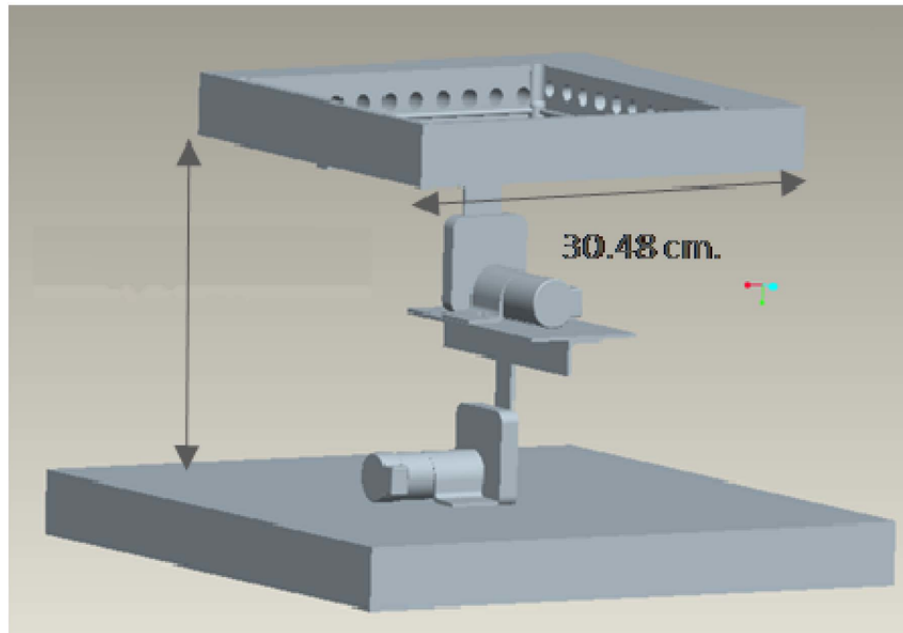
position of the ball through the use of a resistive touch panel. The system utilized a cascaded controller design; the inner loop of the controller featured PID position control of the motors to achieve servo position control, while the outer control loop controlled the ball position through a lead controller that utilized a transfer function constructed through root-locus techniques. The results of platform experiments are not displayed in the paper, but the authors proclaim the platform as a successful venture that was able to balance the ball at any desired position [2].



**Figure 1.1: Platform Design from Rensselaer Polytechnic Institute [2].**

In 2012, a joint collaboration between professors from E&ME, the National University of Sciences and Technology Rawalpindi, and Department of Mechatronics Air University was conducted to create a ball-on-plate balancing system to test and verify

real-world aspects of control systems, such as non-linearities and compensator design [7]. This platform utilized a unique pivot mechanism, where the motors were linked in series to drive the angle of the top platform directly from the center of the top plate, as shown in Figure 1.2. This platform was also unique in that it utilized an 11x11 grid of phototransistors to monitor the position of the ball instead of a more typical touch panel. This design succeeded in balancing the ball at any desired position on the top plate using PID control.



**Figure 1.2: Platform Design from Collaboration [7].**

In 2018, Heeseung Bang and Young Sam Lee from Inha University in South Korea developed a cost-effective implementation of a ball-and-plate balancing platform to be utilized as an educational tool [3]. Unlike the other projects listed, this platform was developed using the Stewart platform framework, shown in Figure 1.3, as well as sliding mode control implemented through a Nucleo-32 microcontroller board. The paper describes the kinematics and inverse kinematics necessary to both simulate the Stewart platform and develop the sliding mode control model.

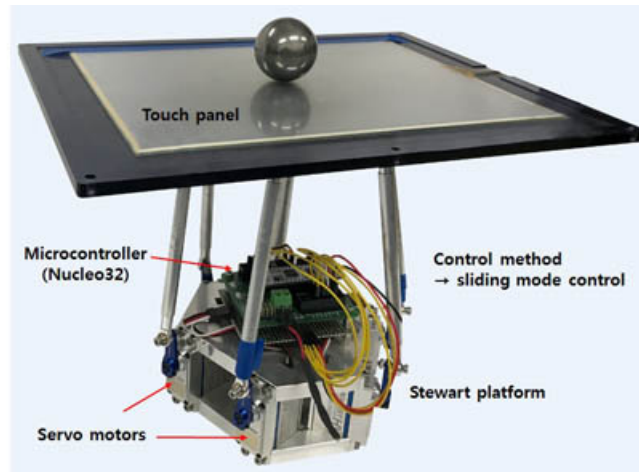


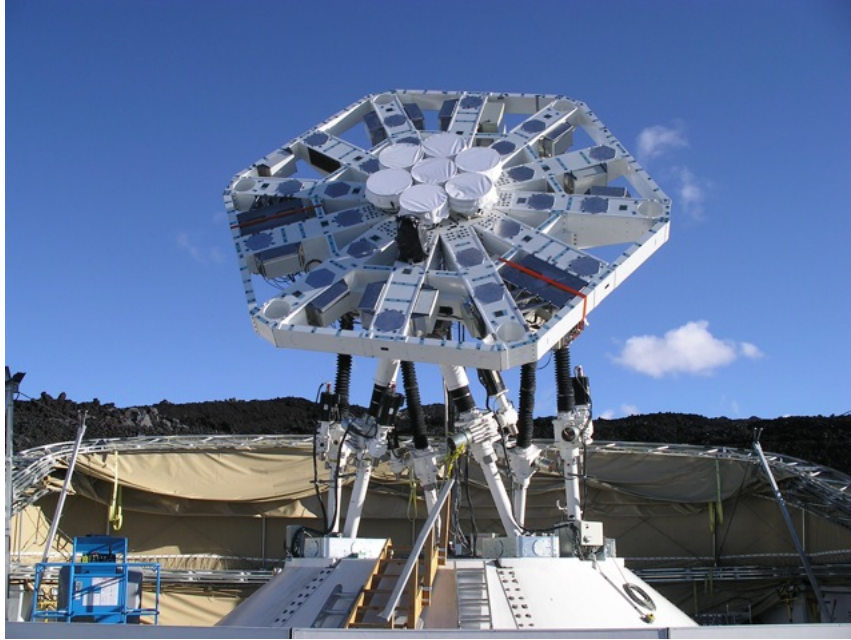
Figure 1.3: Platform Design from Inha University [3]

## 1.2 Types of Ball-and-Plate Platforms

A ball-and-plate platform operates through actuating a top platform that can pivot its orientation to actuate a ball located on the top platform. There are two common methods used to actuate the top plate platform for ball-and-plate balancing platforms: the Stewart platform, and the universal-joint/ball-joint platform.

### 1.2.1 Stewart Platform

The Stewart platform is a type of parallel manipulator useful for many different applications. It utilizes six prismatic actuators (that can be hydraulic or electric) attached to a bottom plate to position a flat top plate at a specified orientation. Figure 1.4 shows the Stewart platform design being used for positioning the Yuan Tseh Lee Array radio telescope.



**Figure 1.4: Yuan Tseh Lee Array Stewart Platform. [1]**

Many ball-and-plate balancing platforms utilize this design because the Stewart mechanism is purpose-built for these types of applications. The Stewart platform is a parallel manipulator that provides superior stiffness and actuation to move the platform in limited but precise orientations. One major aspect of the Stewart platform is the degrees of freedom, or DOF, required to define the system. Unlike the 2 or 3-DOF found in the u-joint and ball-joint category of platforms respectively, the Stewart platform is a 6-DOF system, requiring more information regarding the orientation of the system than the ball-joint/u-joint category of platforms.

### **1.2.2 U-Joint/Ball-Joint Platform**

Relevant to the ball-and-plate platform design utilized in this thesis, the u-joint/ball-joint platform is a variation of the ball-and-plate platform which utilizes a universal or ball joint supporting structure to constrain the top plate. A u-joint is a two-part assembly that allows for the top platform to be constrained to the bottom platform

while allowing for shallow angled movement for control over the orientation of the top plate. Changes in orientation are typically accomplished through motors with arm linkages attached to the top plate. Ball joint platforms are typically provided with position control to continuously set the orientation of the top platform to balance the ball at any position on the top plate. However, while they are similar in functionality and design, ball-joint and u-joint platform have different degrees of freedom; a u-joint design will have 2-DOF, while a ball-joint support structure will have 3-DOF.

### **1.3 Capturing System Orientation**

There are two key properties of a ball-and-plate platform system that must be continually measured for proper feedback control. These are: the orientation of the top plate, and the position of the ball on the plate.

#### **1.3.1 Top Plate Orientation**

When it comes to capturing the orientation of the top plate of a ball-and-plate platform system, there are two common methods: using an encoder connected to each motor, and attaching an inertial measurement unit, or IMU, to the top plate.

The first method, an encoder, utilizes the geometry of the system to capture plate orientation. An encoder is an electro-mechanical device that allows the rotational orientation of an attached shaft to be measured. An encoder can track the orientation of the shaft through a variety of methods, such as through magnetic hall sensors, optical disks, or capacitance.

The second method, an IMU, can calculate the orientation of the plate through the use of sensor fusion, utilizing data readings from the on-board accelerometers, gyro-



scopes, and/or magnetometers. Since successful real-time sensor fusion algorithms are difficult to implement and would take significant processing capacity from many lower-powered microcontrollers, many sophisticated IMU packages come with breakout boards with included processors to off-load the sensor fusion calculations. Instead, communication protocols such as I<sup>2</sup>C or TTL serial from a UART chip allow for configuration and periodic polling of the orientation data from the IMU.

However, the encoder and IMU can be used in tandem; a system can be initialized to a baseline state using the IMU, then through the use of a system model, one can utilize the angle of the actuator motors to calculate the orientation of the platform. This allows for the orientation of the top plate to be processed at a faster discrete time than with the IMU alone.

### **1.3.2 Ball Position Measurement**

A ball-and-plate platform must also be able to measure the position of the ball in the plane of the plate. One of the most popular methods to measure the position of the ball on a ball-and-plate platform is through the use of a touch panel. There are many types of touch panels, such as capacitive, resistive, infrared, and optical. The most common type of touch panel used for ball-and-plate platforms is a resistive touch panel. A resistive touch panel utilizes two panels of electrically-conductive polyethylene terephthalate (PET) separated by air through spacer dots [4]. A sensing pin is connected to the top panel, using an analog-to-digital converter (ADC) to capture the analog voltage as an unsigned integer. Several voltage output pins are connected to the bottom panel, capturing the voltage differential. When the top screen is pressed down, the top and bottom panels make contact, creating a voltage differential output that is read through the sense pin.

Another method to measure the position of the ball on the top plate is to utilize a computer vision system. Through the use of a top-mounted camera and image processing, the ball position can be measured from the reference point of the camera. The x-y position found through the camera system must then be converted to the local coordinate system. To have quick and consistent image processing, a more powerful processor, such as a higher clock speed processor found on a Raspberry Pi, would be desirable.

#### **1.4 Prior Ball-and-Plate Platform Design**

The basis for the design of the platform for this thesis project comes from the platform design used in the ME 305 and ME 405 mechatronics courses taught by Charlie Refvem at California Polytechnic University, San Luis Obispo. This ball-and-plate platform design utilizes a top plate and bottom plate constructed from stacked layers of PCB fiber boards connected through a 3-D printed u-joint. A picture of the platform design is shown in Figure 1.5. This design is the basis for the platform design described in this thesis report, as this project is a continuation of the work done by Zachary Richter in developing a system model for this ball-and-plate balancing platform layout.

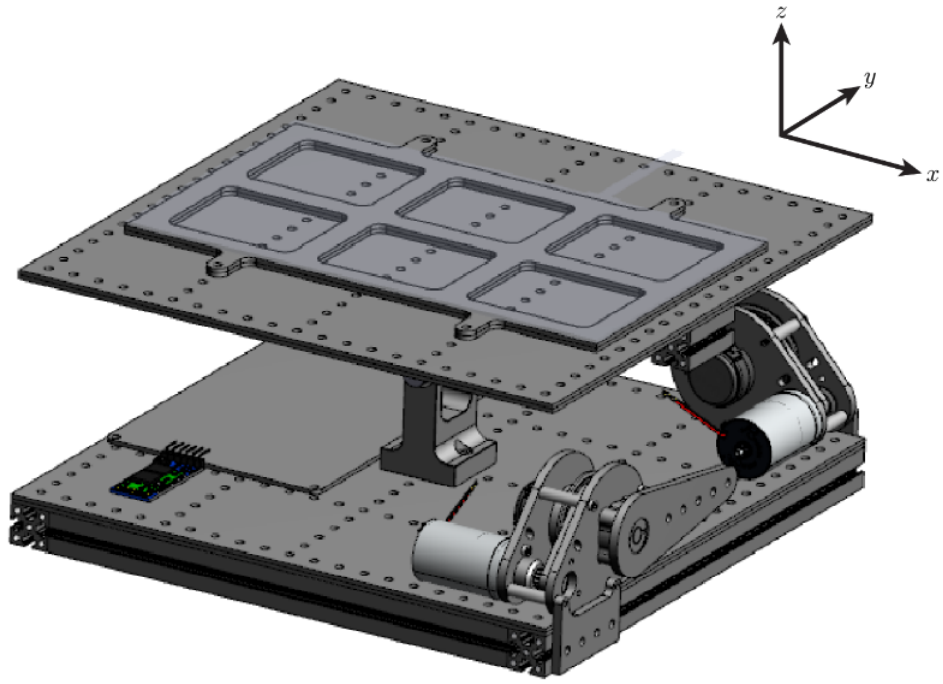


Figure 1.5: Current Ball-and-Plate Platform Design.

## Chapter 2

### SYSTEM DEFINITION AND CONTROL STRUCTURE

#### 2.1 System Definition

The ball-and-plate platform design that was chosen for this project utilizes the u-joint design covered previously in the Background Research chapter. The finalized SolidWorks model is shown in Figure 4.1. The decision to implement the u-joint ball-and-plate platform design was driven primarily to expand the previous thesis work done by Zachary Richter, titled “Nonlinear Model Development and Validation for Ball and Plate Control System” [6]. Richter’s thesis describes modeling a ball-and-plate system through kinematic and Lagrangian methods. This chapter summarizes important system parameters for the purpose of this thesis project to develop the hardware prototype for future controller design.

First, the system must be fully defined for the purposes of axes assignment in the hardware system. Figure 2.1 diagrams the system relative to the body frame, or the axis frame relative to the plate transposed from the u-joint. Figures 2.2 and 2.3 show the important system parameters relative to the x-z and y-z planes of the body coordinate frame, O. The dynamic parameters that need to be captured for controlling the ball-and-plate platform include the position of the ball in x and y relative to the body coordinate frame, as well as the definition of the positive axis rotation in y ( $\beta$ ) and x ( $\gamma$ ). An important aspect of the system to note is that the x and y coordinates are defined as outward relative to their respective motor. All system parameters are defined in Table 2.1.

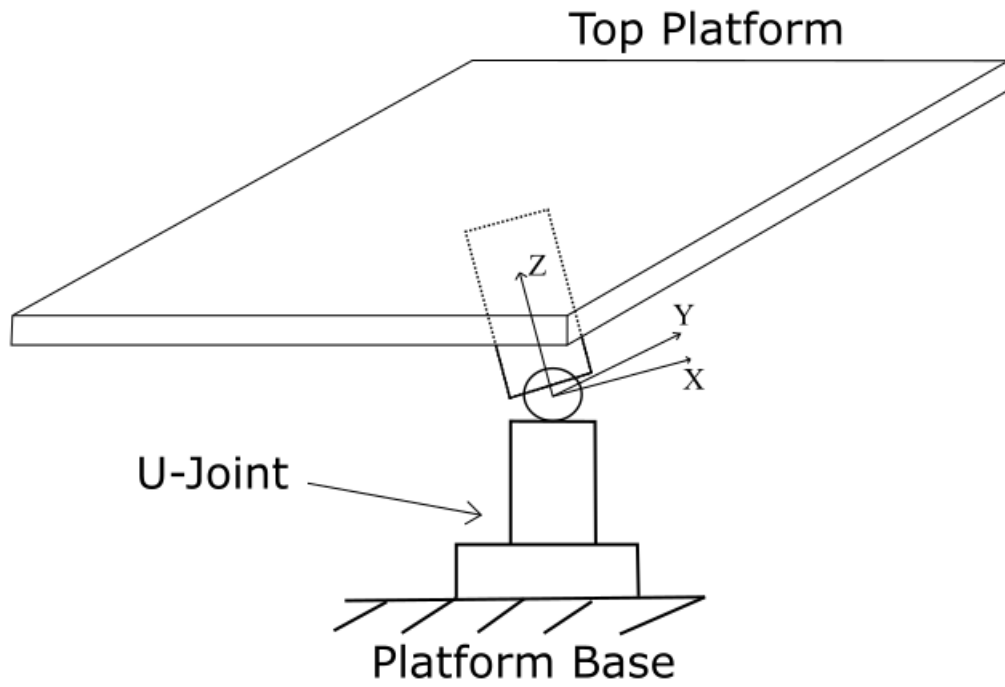


Figure 2.1: Ball-and-Plate Balancing Platform Coordinate System.

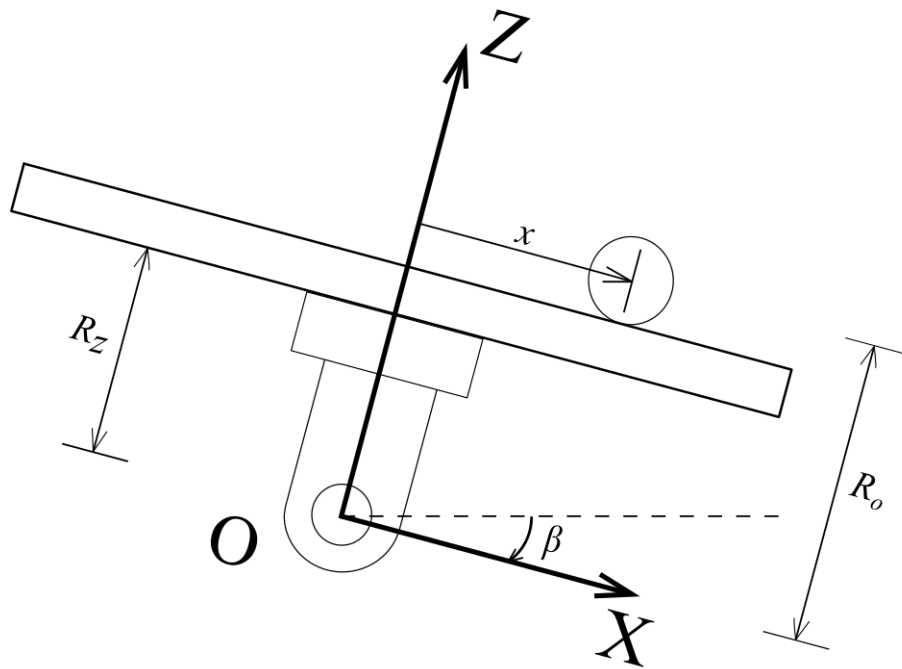


Figure 2.2: Ball-and-Plate Balancing Platform 2D System - XZ plane.

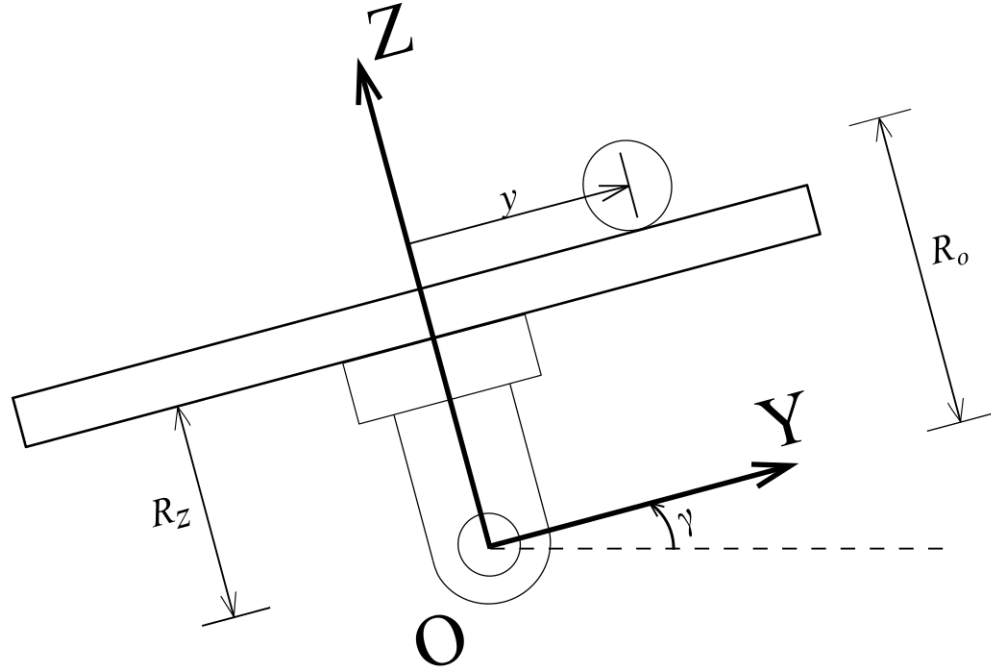


Figure 2.3: Ball-and-Plate Balancing Platform 2D System - YZ plane.

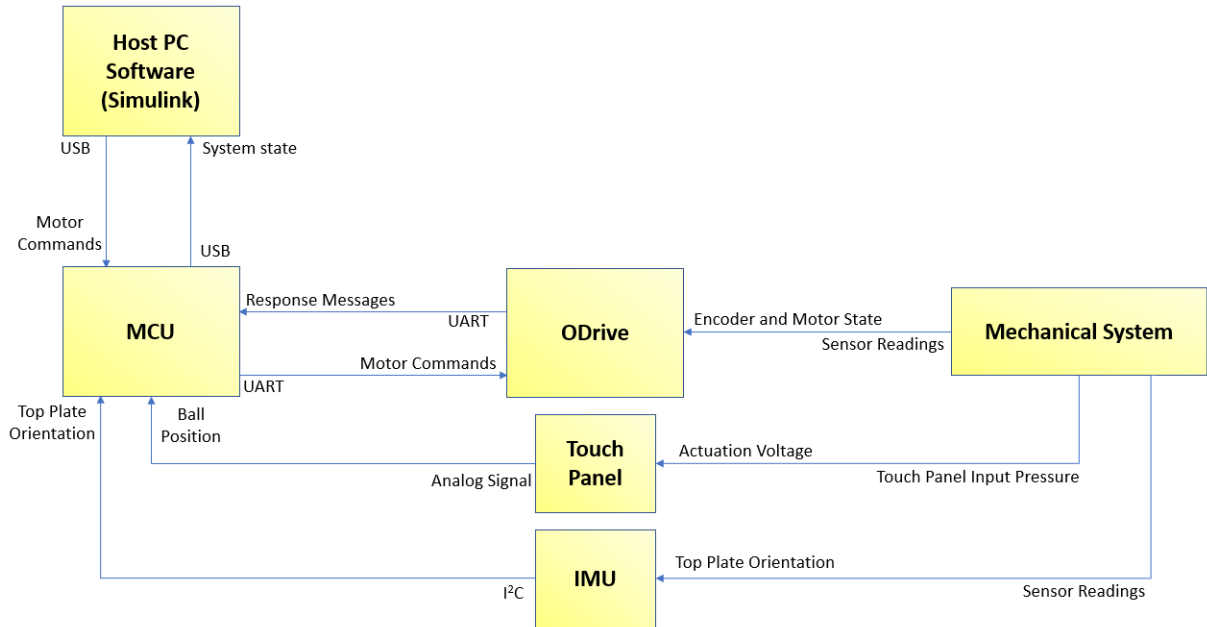
Table 2.1: System Parameters from Schematics, Previous Thesis

Variable	Description
O	Origin of body coordinate frame
X,Y,Z	Coordinate directions of body frame
$x$	Ball position in X body coordinate direction
$y$	Ball position in Y body coordinate direction
$R_o$	Z-position of center-of-gravity of ball
$R_z$	Z-position of center-of-gravity of plate
$\gamma$	Angular rotation about the positive X-axis
$\beta$	Angular rotation about the positive Y-axis

## 2.2 System Functional Overview

Since this platform incorporates separate processors communicating with each other, aspects of the control system are distributed between them. The internal block diagram of the platform system is shown in Figure 2.4. In all configurations, an ODrive motor driver carries out the current control inner loop to regulate the torque output

for the two platform motors. The Simulink<sup>®</sup> system provides the controller design to analyze the state of the system and command a specified torque to each motor of the ODrive system. The communication between the ODrive and Simulink<sup>®</sup> systems is handled through a microcontroller, specifically the STM32F411CE “Black Pill” breakout board. The STM32 microcontroller, or MCU, also collects and processes the sensor data from the IMU and touch panel for use by the Simulink<sup>®</sup> controller. The communication between the host PC, microcontroller, and ODrive components is facilitated through the TTL serial protocol using an ASCII format.



**Figure 2.4: Ball-and-Plate Platform Internal Block Diagram.**

### 2.2.1 ODrive Controller Design

The controller design of the ODrive utilizes a cascaded controller design, where, depending on the control mode selected, different portions of the controller are bypassed. Figure 2.5 is a diagram of the cascaded controller provided by ODrive [5]. Each stage of the controller shown (position, velocity, current controllers) utilizes

either a P or PI controller. However, in current command mode as utilized in this project, both the position and velocity controller sections of the cascaded controller are bypassed. Instead, the current command is directly applied through conversion of the torque command to current by utilizing the torque constant of each motor. This cascaded controller design is used when the platform is being zeroed by the user, as proportional-controlled position control of the motor arms allow for small adjustments to the top plate orientation for serving as a sensor baseline.

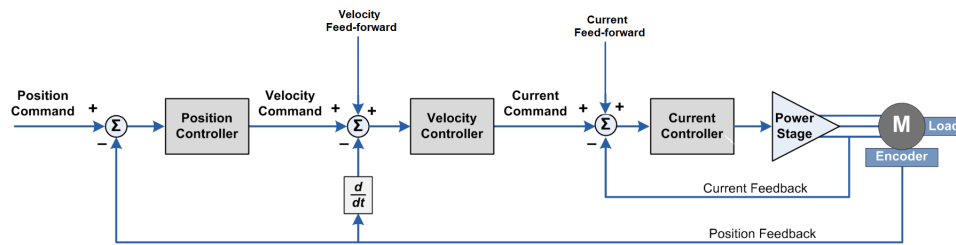


Figure 2.5: ODrive Cascaded Controller Design [5].



## Chapter 3

### ELECTRONICS

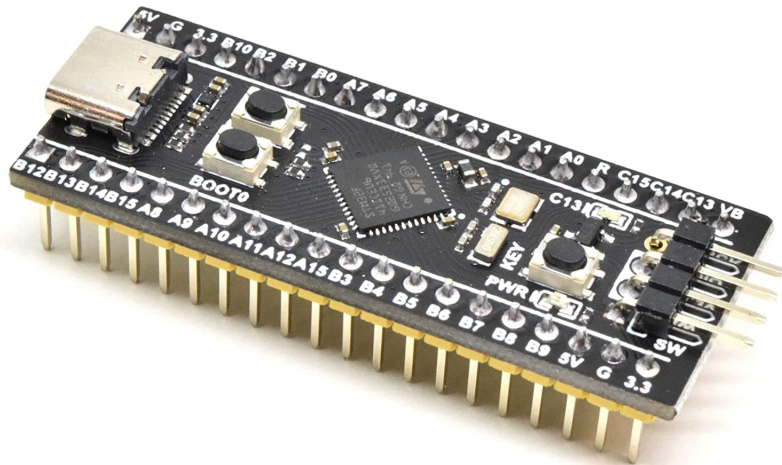
#### 3.1 Electronics Overview

The electronic system of the ball-and-plate balancing platform prototype developed for this thesis incorporates a suite of sensors and peripherals to allow for a variety of design choices for future controller experimentation. A wiring diagram for the platform prototype can be seen in Appendix A.

##### 3.1.1 Microcontroller

The chosen microcontroller unit, or MCU, to operate the ball-and-plate platform was the STM32F411CE microcontroller from ST Microelectronics, shown in Figure 3.1. This microcontroller provides a CPU clockspeed of 100 MHz and ample pinouts necessary for the various components needed for the function of the platform. Compared to other microcontrollers, the STM Nucleo-32 line are able to fit diverse applications and programming languages. The STM32 line of microcontrollers supports MicroPython, portable C++ using the Arduino framework, and C/C++ with the Hardware Abstraction Layer library, also known as the HAL Library. The HAL library provides users ready access to communication protocols, GPIO pins, timer channels, and many other peripherals provided by the STM32 MCU. The electronics setup uses the TTL serial protocol to connect to the ODrive motor driver, I<sup>2</sup>C to communicate with the IMU, as well as various available pins for ADC readouts of the touchscreen. The necessary I/O requirements of the microcontroller were relatively small, as many

necessary I/O considerations were shared between the chosen MCU and the ODrive Motor Driver board. The microcontroller software is covered in the Software section.



**Figure 3.1: STM32F411CE Microcontroller Development Board.**

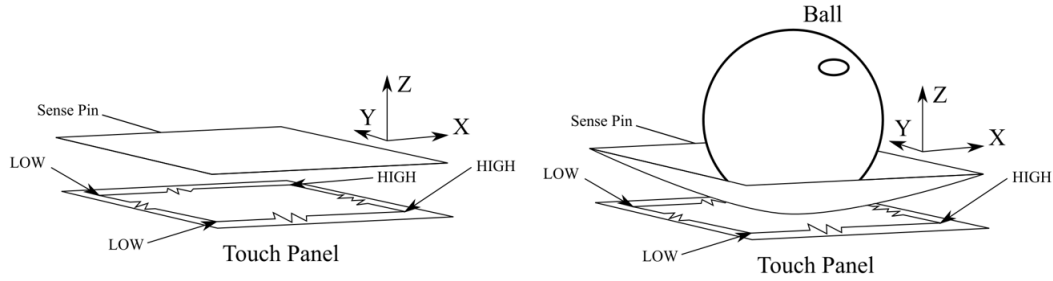
### 3.1.2 Touch Panel

The touch panel that was selected for this ball-and-plate balancing platform prototype was the AMTouch 02511-00 5-pin resistive touch panel, shown in Figure 3.2. This touch panel has outer dimensions of 404.20 x 330.00 [mm], and the active area (the area of the touch panel that registers touch actuations) has dimensions of 378.50 x 303.00 [mm]. This touch panel was chosen because of the combination of size and close-to-square aspect ratio (4:3), which allows for more options in regards to trajectories mirrored about the center as compared to the more common wider aspect ratios such as 16:9. The touch panel was provided by AMTouch for free to be used for this project.



**Figure 3.2: AMTouch Resistive Touch Panel.**

The 5-pin touch panel configuration functions by placing HIGH and LOW voltages for specified corners on each side of the bottom conductive layer of the touch panel; each of these corners is connected to its adjacent corners using resistors. For example, if measuring the x-position, the touchscreen should be configured to have the left two corners of the touch panel held LOW (ground) and the right two corners of the touch panel held HIGH (+5V). The fifth pin of the 5-pin panel, labeled as the sensing pin, is used as an analog output of the voltage that occurs when depressing the top conductive layer of the touch panel onto the bottom conductive layer. An example of this situation is shown in Figure 3.3; the left diagram of the figure shows the un-deflected touch panel, the right side of the figure shows the touch panel being actuated by a ball. The analog voltage of the contact between the top and bottom conductive layers is read by the ADC on the STM32F411 chip. The touchscreen then can be calibrated to correlate the voltage read from the 12-bit analog-to-digital converter (ADC) to the position of the ball on the touch panel through the use of known calibration points. The position signal must then be filtered to reduce the amount of noise of the sensor reading. See the Software section for calibration and filtering functionality.



**Figure 3.3: Example of Reading Ball Position on the X Axis.**

While a 5-pin touch panel is more prone to signal debouncing issues than a 4-pin touch panel, the AMTouch 02511-00 resistive touch panel was the touch panel chosen for this project as it met the project requirements for size, durability, and functionality.

### 3.1.3 Motor

Motor selection depended on several main criteria:

1. The motor shall provide adequate torque to control the system according to simulation results.
2. The motor shall be under \$200, as the build cost for this prototype should not exceed \$1000.
3. The motor shall be compatible with the operations of a ball-and-plate platform: high torque, and low RPM.

Utilizing the simulation work done by Zachary Richter for this prototype design, the desired motor torque necessary for an aggressive simulated response of the aluminum top plate of the platform was approximately  $430 [mN\cdot m]$  [6]. A list of motors considered for the prototype is included in Table 3.1.

**Table 3.1: List of Considered Motors**

Motor Model	Motor Manufacturer	Nominal Torque [N-m]	Stall Torque [N-m]	Stall Current [A]	No Load Speed [RPM]	Price
<b>G60, Kv 25</b>	<b>T-Motor</b>	<b>0.6</b>	<b>1.75</b>	<b>4.2</b>	<b>560</b>	<b>\$108.90</b>
EC 90 Flat, 607950	Maxon	0.95	7.80	111	2340	\$213.68
EC 60 Flat, 614949	Maxon	0.54	4.30	81.9	4300	\$134.90
IG42	Shayang Ye Industrial	1.96	4.20	1.44	47	\$61.95

The motor chosen for the ball-and-plate platform prototype is the G60 KV-25 motor from T-Motor, shown in Figure 3.4. The G60 gimbal motor was chosen for its excellent price-to-performance ratio. Also, since it is a gimbal motor, it is the type of brushless DC motor, or BLDC motor, best suited for this application due to providing smooth operation at low speeds and high torque, perfectly describing the low angle travel but high torque application of a ball-and-plate platform. The G60 motor was also chosen for its power efficiency over the other considered motors. The additional torque overhead provided by the G60 motor is also beneficial, as real-world conditions that can occur for the platform, such as increased friction within the u-joint bearings and general hardware degradation, can be overcome with the additional torque. The additional torque overhead from the G60 motor also allows for more aggressive controller implementations to be accommodated. While other DC-gearred motors could produce a greater torque output at a smaller power requirement, potential effects on the dynamics of the platform from gear backlash and their low no-load speed disqualified their use for this project.



**Figure 3.4: G60 Motor from T-Motor.**

#### **3.1.4 Motor Encoder**

The encoder chosen for the ball-and-plate platform prototype is the ENC-AMT-113SV capacitive encoder shown in Figure 3.5. This encoder was chosen for its programmable configuration, variation in bore sleeve sizing for prototyping, as well as its proven compatibility with the ODrive motor driver through their list of recommended encoders. The encoder configuration for this project utilizes an incremental resolution of 8192 counts per revolution, or CPR. While an IMU might be able to serve as the sole necessary sensor to fully capture the orientation of the top plate, an encoder is still required for this project, as the commutation of the BLDC motors allows for current to be delivered in phase with the windings of the motors. Also, since an encoder signal can be read much faster than the sensor fusion on an IMU can take place, 100 [Hz], the encoder can be used instead of the IMU for controller designs that require

faster data update rates. The encoder also possesses a collection of shaft connectors for interfacing with a variety of shaft diameters, allowing for better out-of-the-box prototyping capabilities than other encoder options.

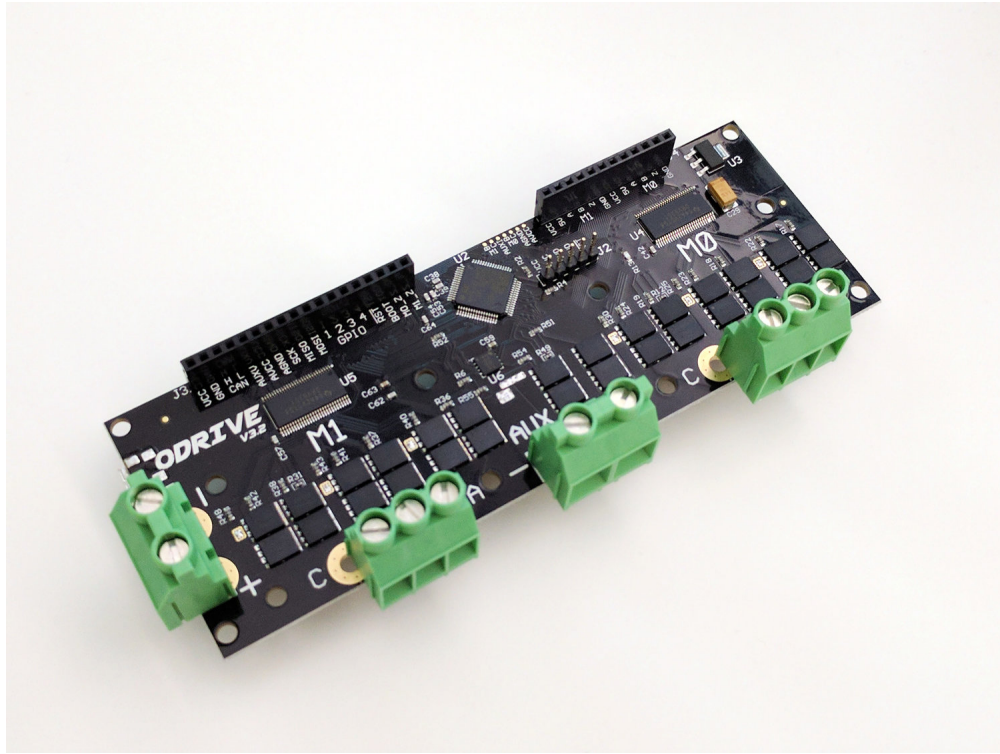


**Figure 3.5: Different Configurations of the AMT-113 Encoder.**

### **3.1.5 Motor Driver**

The motor driver chosen for the ball-and-plate platform prototype is the ODrive 24V motor driver from ODrive Robotics, pictured in Figure 3.6. This motor driver was chosen because of its performance record in similar projects (such as in Stanford’s Doggo project), as well as its ability to provide various control types (position, velocity, torque, voltage) for two BLDC motors through one cost-effective package. Its unique position as an open-source project compared to other considered motor drivers was also beneficial. The ODrive community has active forum posts and ample documentation, with the developers actively engaging in the forums to troubleshoot issues and work on updating features. Its open-source nature also allows for analyzing

and making changes to the on-board firmware to tailor the ODrive to any particular project.



**Figure 3.6: Assembled ODrive Board.**

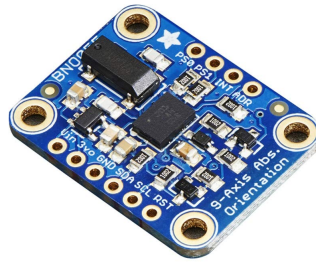
The motor control provided by the ODrive is facilitated through the use of Onsemi MOSFETs (part number NTMFS4935NT1G) and the DRV8301 motor driver. The ODrive motor drive configuration is suited for high current hobbyist motors, so much of the current control capabilities are not very suitable for the gimbal motors chosen for this project. The ODrive can natively support gimbal motors through the use of voltage control by re-purposing the functionality of many variables to support voltage control. However, after replacing the current-sensing shunt resistors on each motor axis, current control of the G60 motor outperformed the gimbal mode configuration in both torque output accuracy and precision. Therefore, gimbal mode was not used for the final prototype, but the gimbal mode configuration is still available for use in future platform iterations.



For this project, only current/torque control is utilized, and the on-board current controller utilizes a simple PI torque inner control loop with configuration parameters adjustable through serial communication to the on-board STM32F4 microcontroller or through the on-board micro-usb port. This approach allows for granularity in adjusting the motor performance to best fit the platform configuration, as well as off-loading CPU processing for improved performance of the microcontroller running the platform. The firmware loaded to the ODrive is a custom version of the official 0.5.4.0 version. A custom version was required to change the value of the shunt resistance for current measurements. The ODrive firmware was required to be updated periodically throughout the project. Since the ODrive project is open-source and in constant development, many software updates are posted throughout the year for key features that were necessary for the success of the project, such as changing the state of the motor driver through the TTL serial interface.

### **3.1.6 Inertial Measurement Unit**

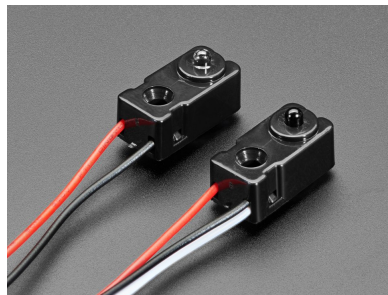
The IMU chosen for the ball-and-plate platform prototype is the Adafruit BNO055 Absolute Orientation Sensor breakout board shown in Figure 3.7. This inertial measurement unit, or IMU, was chosen over other options due to its diverse set of features and ready availability as compared to other IMUs. The BNO055 features on-board sensor fusion algorithms to offload CPU computations to a breakout board, calculating many useful orientation parameters such as angular position and velocity. The IMU sensor data that was pertinent to this project was the absolute orientation of the IMU in Euler angles, as well as the angular velocity that the IMU is experiencing. These parameters allow for the orientation of the top plate to be fully captured. The IMU can report Euler angles to 1/16th of a degree at an update rate of 100 [Hz].



**Figure 3.7: BNO055 Adafruit Break-out Board from Adafruit.**

### **3.1.7 Break Beam Sensors**

One of the safety mechanisms utilized on the platform are the IR break beam sensors, shown in Figure 3.8. These break beam sensors, provided from Adafruit, utilize an emitting sensor and receiving sensor, where an IR beam is continuously emitted to the receiver. When this beam connection is broken, the sensor, as configured through a pull-down resistor located on the ODrive board, will signal to the ODrive that the endstop has been tripped, shutting off the motor on that axis. This soft-stop safety measure allows for experimental control systems to be run on the platform while ensuring that potential harm to the platform and user is minimized.



**Figure 3.8: IR Break Beam Sensors from Adafruit.**

## Chapter 4

### MECHANICAL DESIGN

#### 4.1 Design Overview

The ball-and-plate platform prototype is designed for stiffness of the top plate, so many components are designed for this consideration. This design choice was to ensure that the top plate was a robust surface to prevent bending that could influence sensor readings from the IMU and touch panel sensors. The overall platform design is shown in Figure 4.1, with key system subsections listed in Table 4.1. The part drawings for all parts used in the platform can be found in Appendix B.

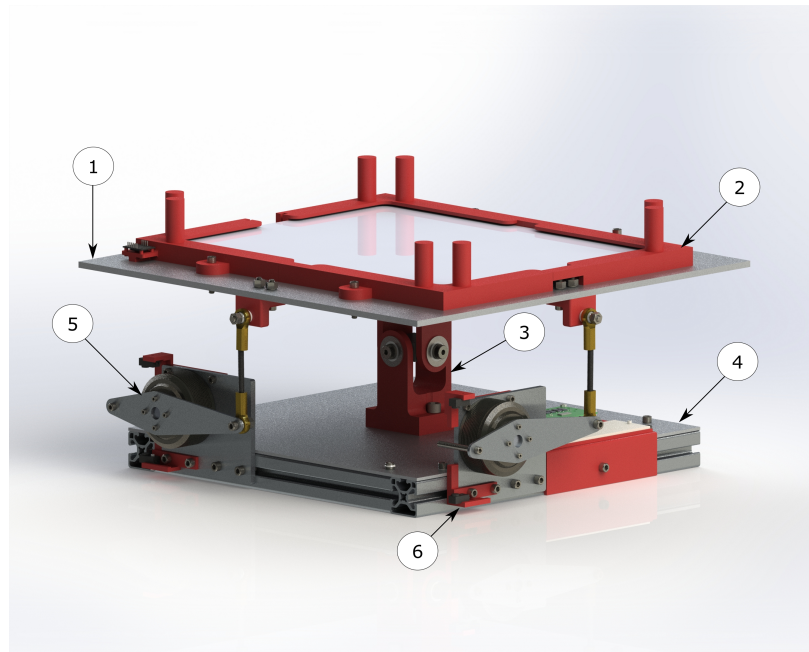


Figure 4.1: Render of Ball-and-Plate Platform SolidWorks Assembly

**Table 4.1: List of Mechanical Design System Components**

Item	Component
1	Top Plate
2	Touch Panel Spacers
3	U-Joint Assembly
4	Bottom Plate
5	Motor Plate Assembly
6	Endstop Brackets

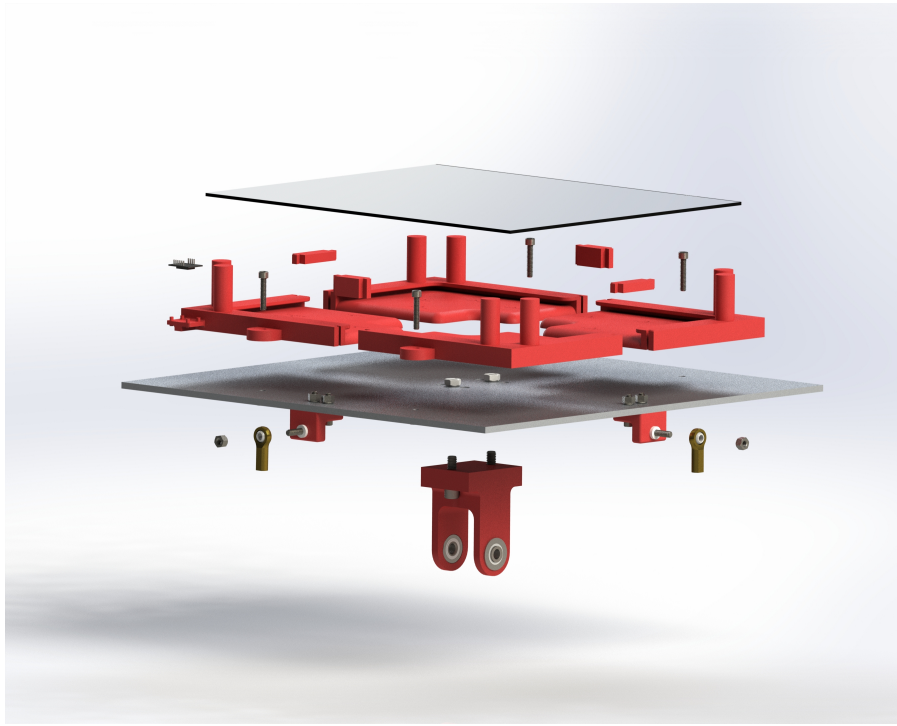
The mechanical design of the platform system consists of several subsections: the top plate, touch panel spacers, the u-joint assembly, the bottom plate, the motor mount assemblies, and the endstop brackets. The top plate, bottom plate, and the motor mounts are made out of aluminum 6061-T651, while the remaining components are 3D printed out of PLA.

The platform measures 19 x 19 x 9.13 [*in*] and weighs 25.4 [*lbf*]. These parameters make the platform reasonably portable, while also minimizing wayward motions during operation and providing the ability to support many different potential configurations. Compared to the ball-and-plate platform design developed by Charlie Refvem, this platform is 237.5% larger by touch panel display size to accommodate increased effects from non-linearities for more rigorous controller experimentation.

#### **4.1.1 Top Plate**

The purpose of the top plate is to provide a level surface to attach the touch panel, the u-joint, and the motor arms. The top plate was machined through waterjetting, creating hole patterns for the various necessary attachment interfaces. The top plate is designed as a 1/4" aluminum plate to prevent any potential deflection of the top

plate, which could affect readings from the IMU and touch panel. A CAD render of an exploded view of the top plate sub-assembly is shown in Figure 4.2.



**Figure 4.2: Exploded View of Top Plate Assembly.**

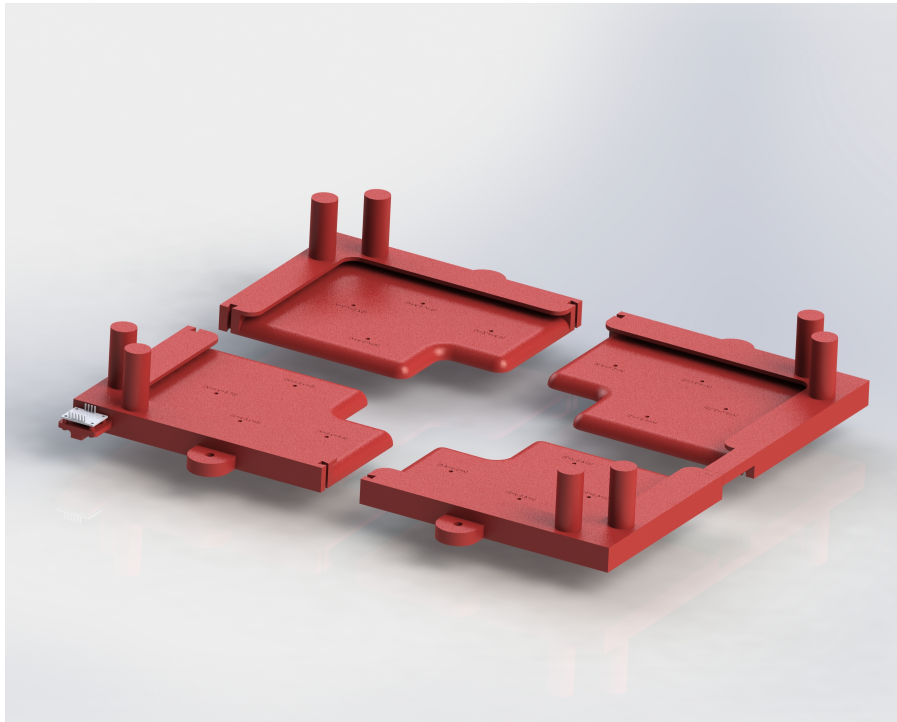
#### **4.1.2 Touch Panel Spacers**

To mount the touch panel to the top platform, a spacer was necessary to create room for the lock nuts necessary for secure u-joint attachment to the top plate. The touch panel spacers are shown in Figure 4.3. The touch panel spacer also performs other functions:

1. The touch panel spacer incorporates fencing around the touch panel to help keep the ball within the active sensing area of the touch panel.
2. The touch panel spacer provides an attachment point for the IMU to the top platform.

3. The touch panel spacer provides embedded calibration points to use for touch panel calibration.

The spacer is printed in four separate pieces that are attached to the top of the platform through specific holes and attached to each other through 3D printed connector pieces.

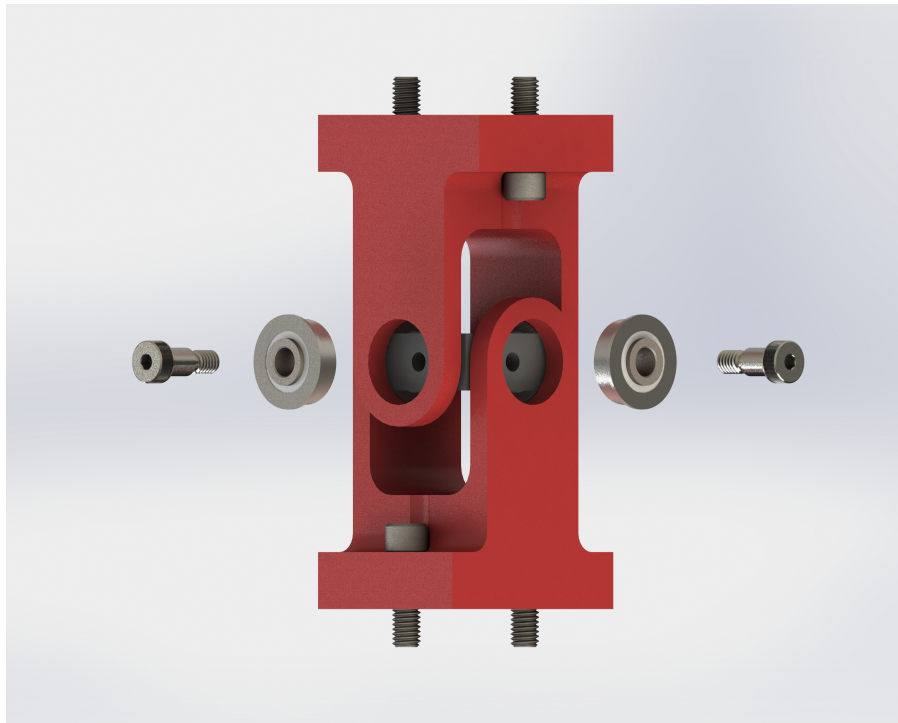


**Figure 4.3: CAD Render of touch panel Spacers.**

#### **4.1.3 U-Joint**

The u-joint design was inspired by the current ball-and-plate platform developed by Charlie Refvem. The u-joint is constructed of five parts: the two PLA pillow blocks that connect to the bottom and top platforms (3D printed as two pieces to minimize errors in the printing process) and the crossbar that is located between the two pillow blocks. An exploded view of the assembly is shown in Figure 4.4. The crossbar

is constructed of Delrin and allows for the axes of rotation connecting the top and bottom pillow blocks to be through the same origin point. The pillow blocks are secured to the crossbar through shoulder screws supported by shielded ball-bearings. One important consideration for the u-joint is in ensuring that it allows for the desired amount of rotation necessary on each axis for adequate function of the top plate of the platform. From experimentation in SolidWorks, the amount of rotation that the top platform can accommodate is roughly  $\pm 12$  degrees in both the x and y body coordinate axes.



**Figure 4.4: Exploded View of U-Joint Design.**

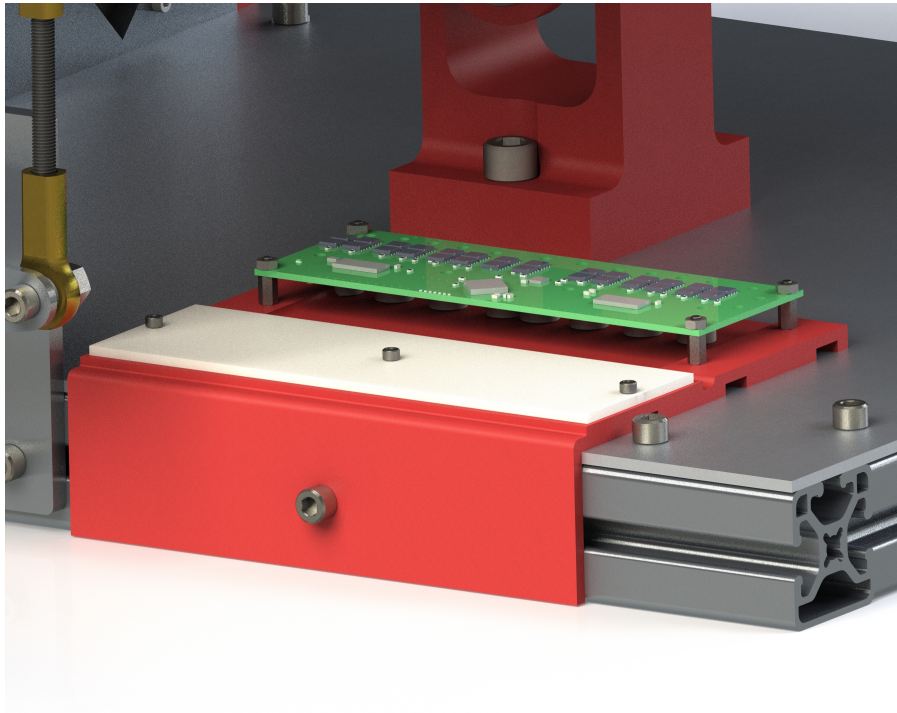
#### **4.1.4 Bottom Plate**

The bottom plate is designed to provide support for the top plate and the various system components. The bottom plate subsystem is composed of an aluminum extrusion frame that supports the bottom plate; an additional support provides reinforcement



of the platform to prevent deflection in the center where the u-joint is located. The use of aluminum extrusions was chosen because of the versatility that is provided.

The bottom plate subsystem also contains electronic hardware for the platform, including the O-Drive motor driver and the STM32F411 microcontroller. The electronics are currently mounted to the platform via a 3-D printed mounting bracket, shown in Figure 4.5. The bracket was designed to facilitate a perma-proto breadboard located close to the O-Drive motor driver breakout board to minimize wire length and provide a more professional aesthetic.



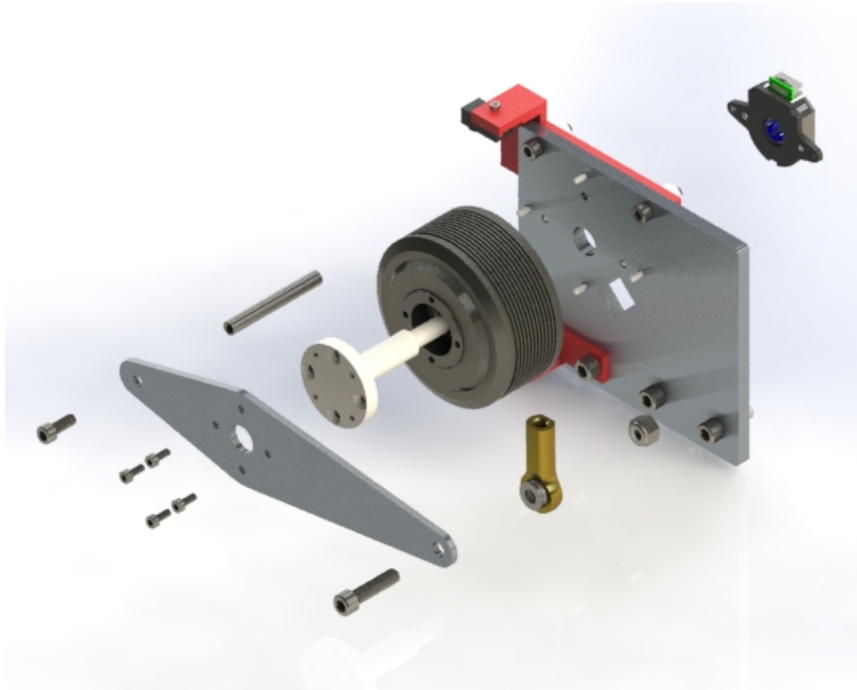
**Figure 4.5: CAD Render of ODrive Mounting Bracket.**

#### **4.1.5 Motor Mount Assembly**

The motor mount assembly consists of the motor plate, the selected T-Motor G60 motor, the AMT-113SV encoder, the 3mm break-away sensor suite, and the motor arm that connects the output of the gimbal motor to the top platform and the endstop



safety system. The motor mount is made from 1/4" aluminum stock; the hole patterns required for motor mounting can be cut via plunge-milling or waterjet cutting; both processes were used in the creation of the hole patterns on the motor mounts for this project. An exploded view of this sub-assembly can be seen in Figure 4.6.

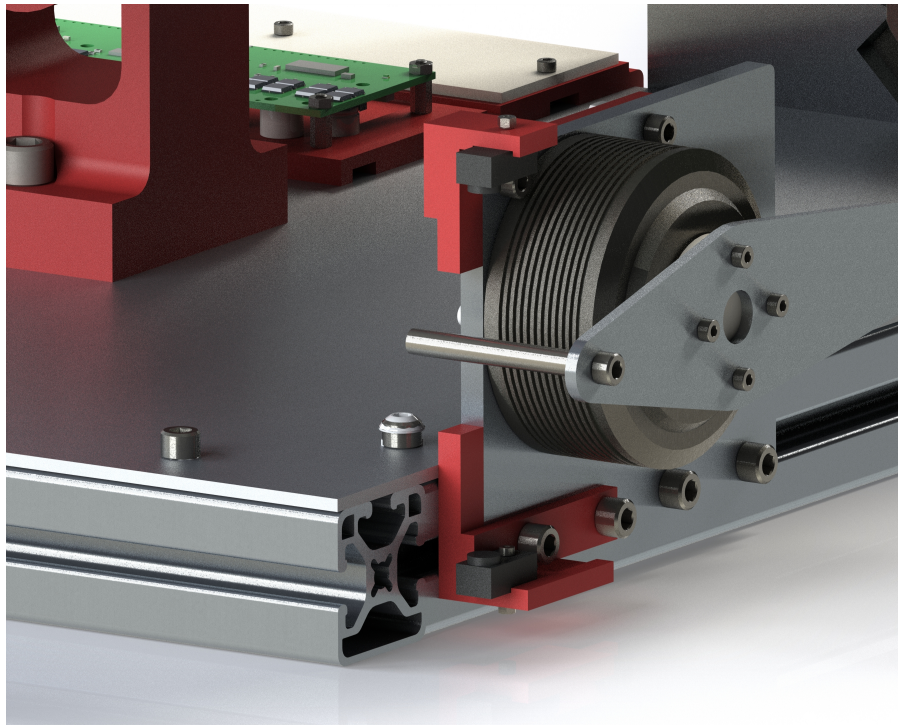


**Figure 4.6: CAD Render of Motor Mount Assembly.**

#### **4.1.6 Endstop Bracket**

The endstop safety system, shown in Figure 4.7, ensures that the motor actuates within a restricted angle of travel found through analysis of the angle of u-joint travel using the SolidWorks model. The endstop is constructed from two 3D-printed brackets that attach to the motor mount, housing break-beam sensors and featuring an extended hard-stop surface from the motor mount. The standoff attached to the motor arm interacts with the endstop bracket system; if the motor arm rotates to interrupt the IR beam of the break-beam sensors, the break-beam sensors will trigger, deactivating the motor on that axis. This allows for the break-away sensors to act

as a soft-stop and the hard-stops to engage with the motor arm if the break-away sensors cannot trigger the flag to stop the motors in time. The hard-stops are for system redundancy; in practice, the soft-stops will be able to deactivate the motor before the hard-stops are engaged.



**Figure 4.7: CAD Render of Break Away Mounting Bracket.**

## Chapter 5

### SOFTWARE DESIGN

#### 5.1 Software Overview

The software for the platform can be divided into two sections: the software implemented on the STM32 microcontroller and the software implemented on the Host PC.

#### 5.2 STM32 Software Overview

The ball-and-plate platform utilizes the chosen STM32 MCU to perform several functions:

1. The STM32 microcontroller actuates the top plate through torque commands to the ODrive motor driver powering the two G60 BLDC motors.
2. The STM32 microcontroller facilitates communication with both the ODrive motor driver and MATLAB/Simulink<sup>®</sup> through serial communication.
3. The STM32 microcontroller accommodates a user interface to assist users in setting up the platform for zeroing and operation.

The STM32F411CE was chosen as the microcontroller used for this project due to the flexibility of programming languages that could be used to build its firmware. The STM32F4 family of MCUs can run Python or C/C++ programs; programming in Python offers easier access to complex functions and standardized libraries at the

cost of CPU overhead. Programming in C/C++ offers more efficient software at the cost of coding complexity. When choosing a programming language to code the MCU, the runtime efficiency of C/C++ was desirable over the simplicity of coding in Python.

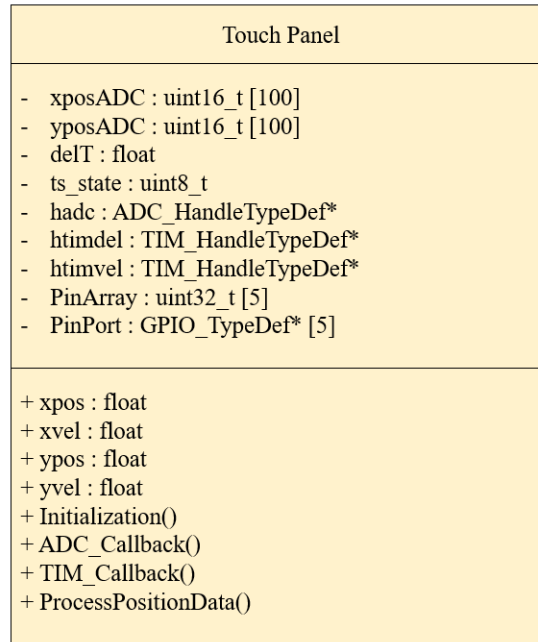
The STM32CubeIDE provides its own standardized library for their microcontrollers called the HAL, or Hardware Abstraction Layer. This library provides APIs that abstract many lower-level functionalities of STM microcontrollers through user-friendly commands and functions. This abstraction allows for easy accessibility to important hardware functionality, such as communicating with peripheral devices using communication protocols like TTL serial and I<sup>2</sup>C, converting analog signals to digital values using on-board ADC, and handling digital I/O functionality. The HAL library also allows for the unification of all STM MCUs; HAL programs are portable between many STM32 MCUs. Further sections detail the specific software functionality used for each electronic system on the platform.

The final iteration of the STM32 firmware utilizes timer channels and interrupt flags to regulate the execution order and timing of functions, along with several interrupt service routines, to allow for continuous sampling of the touchscreen as well as to keep communication packets on a standardized cycle time that is determined by the host PC.

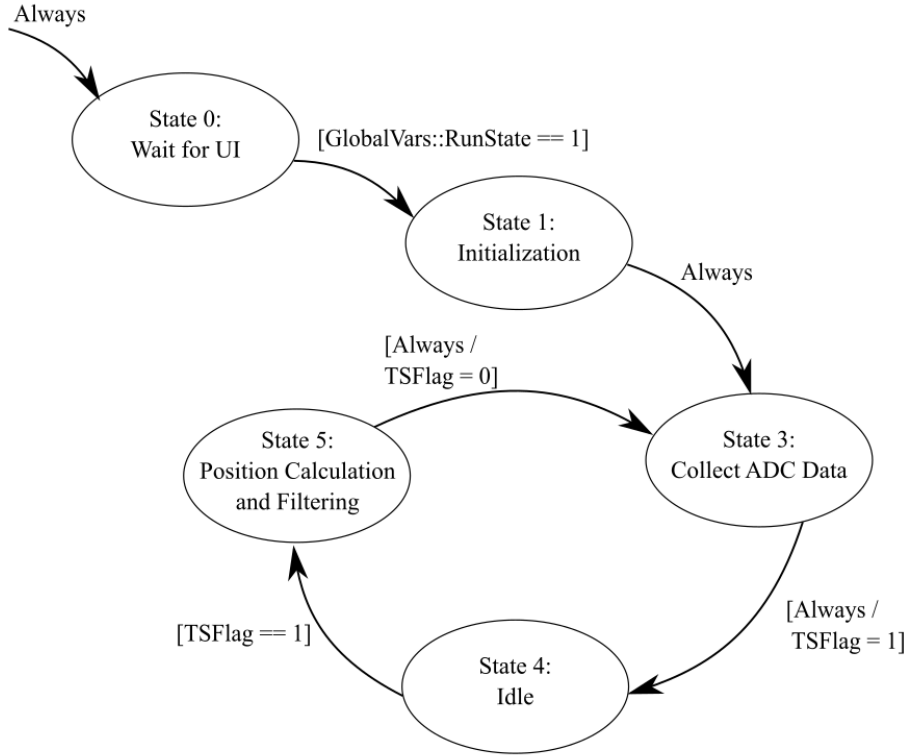
### **5.2.1 Touch Panel Class**

The MCU utilizes the on-board ADC to convert the analog voltage signal from the sense pin of the 5-pin touch panel into a ball position and velocity in the body reference frame. The ball position and velocity are utilized by the host PC for use in the Simulink<sup>®</sup> controller. The class object diagram is shown in Figure 5.1 and the

finite state machine is shown in Figure 5.2. The touch panel class was implemented as a finite state machine to allow the touch panel to run cooperatively with other important functions such as the IMU data collection and processing, as well as the communication response function when the STM32 receives messages over USB from the host PC.



**Figure 5.1: Touch Panel Class Object.**

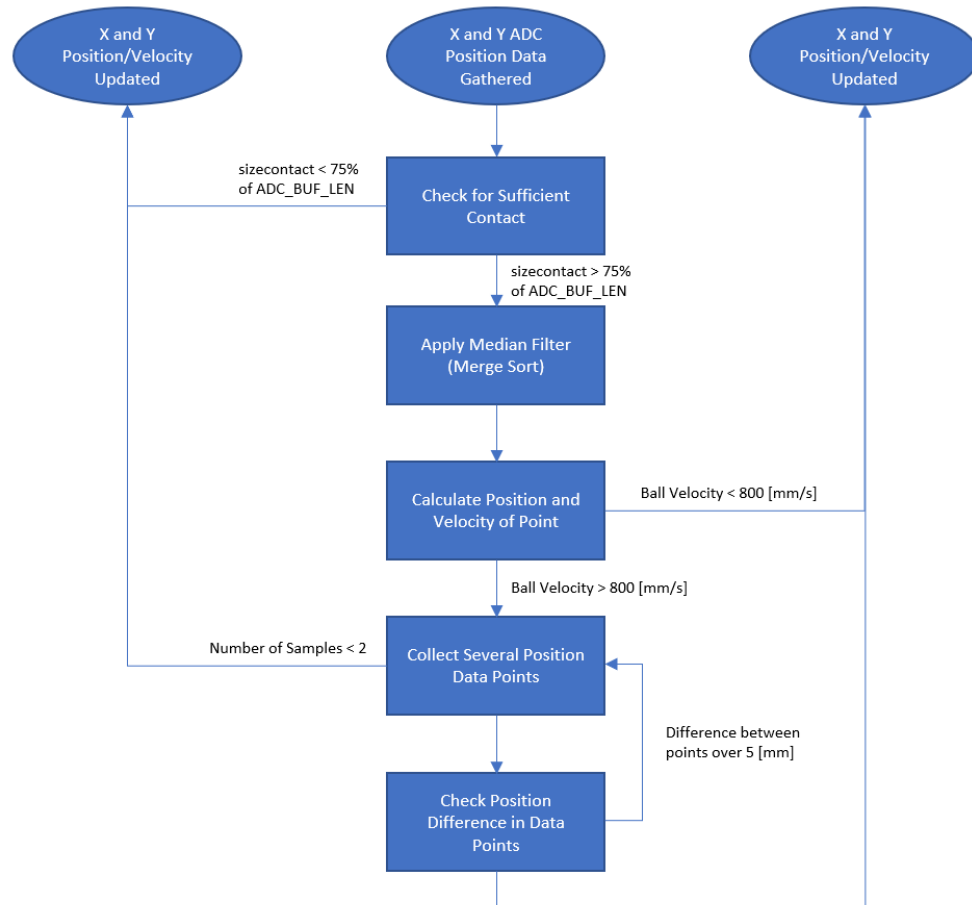


**Figure 5.2: Touch Panel Finite State Machine.**

To utilize the touch panel, several key functionalities are necessary. To accurately convert the 12-bit voltage readout from the touch panel to a position on the touch-screen, calibration is necessary. This functionality is relegated to an external file from the main script. The function utilizes the `math.h` external library to apply a least squares linear regression algorithm to an incoming set of calibration data, finding the first-order equation that correlates the voltage read by the ADC to the position of the ball in  $[mm]$  on the touch panel. Standardized calibration points on the touch panel spacers spaced evenly from the center of the u-joint allow for this process to be fast and efficient. After calibration is complete, the equation coefficients are stored as variables that can be changed in future iterations of the platform.

On top of requiring calibration, the touchscreen signal also requires filtering to ensure that there is minimal noise in the position signal. This noise issue is further exacer-

bated by sometimes insufficient contact pressure to adequately read the ball position. To filter this incoming signal, oversampling, sorting, averaging, and a logic filter are used to minimize the effect of noise on the data sampling; a flowchart of the filtering process is shown in Figure 5.3.



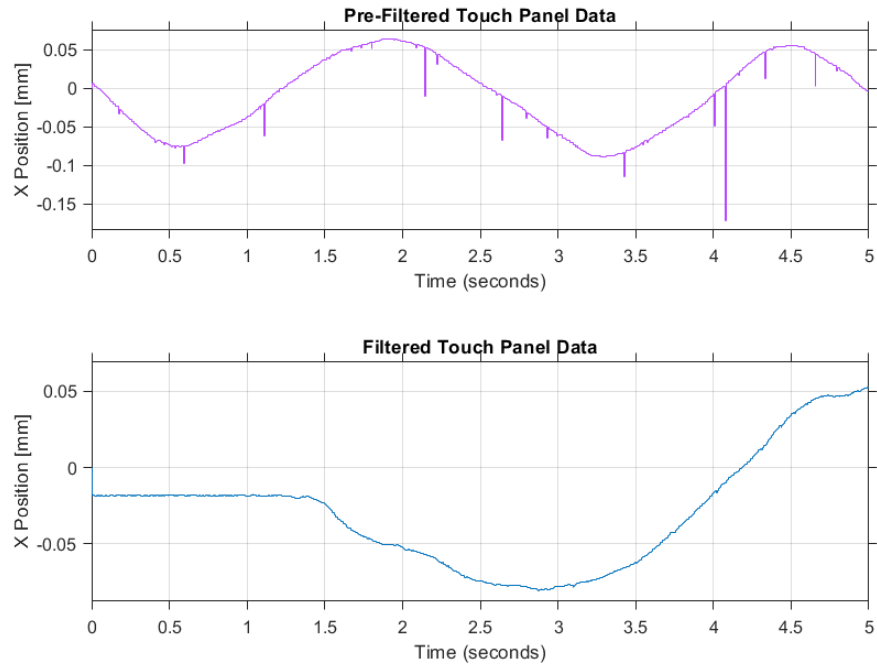
**Figure 5.3: Touch Panel Filter Flowchart.**

Touch panel position data is taken and recorded into two arrays of one-hundred 12-bit unsigned integers (one for x position readings and one for y position readings, denoted as ADC\_BUF\_LEN). Next, there is a check to ensure that there is sufficient contact at each data point between the plate and ball through a minimum reading of the ADC voltage. This minimum voltage was found through applying force at the absolute minimum coordinate location on the platform and reading the ADC value,

with a given threshold of 50 ADC value to account for any anomalies. Once the arrays are populated with valid x and y position values, a merge sort algorithm is applied to order the arrays from lowest to highest value. A merge sort was utilized instead of other common sorting algorithms (insertion sort, bubble sort, quick sort) due to its ease of implementation through available algorithms and processing efficiency. Utilizing a sorting algorithm allows for the use of median filtering to get a final 12-bit ADC reading for x and y. This value is then converted to a position through the calibration constants to the position and velocity of the ball in  $[mm]$  and  $[mm/s]$ , respectively. A final logic filter is implemented to ensure that the ball velocity is within 800  $[mm/s]$ ; this top speed was determined through applying conservation of energy when the ball is positioned at the center with the platform actuated to 10 degrees, rolling towards the edge of the active area. If the ball position changes by over 800  $[mm/s]$  the change in position could be due to insufficient contact pressure by the ball on the plate. In this case, the last known valid position is held constant while new sample data are taken. If several data points are within 5  $[mm]$  of each other, the position is now treated as valid, and the position and velocity data for the ball are updated. The ball position and velocity values are then transferred to the host PC. Since the touch panel is continuously taking samples and updating the position and velocity of the ball, there can be several updates to the ball parameters before they are communicated to the host PC. A comparison between the ball touch panel position data filtered through the logic filter versus position data that was unfiltered can be seen in Figure 5.4. The top figure shows a data set collected when there was no filter applied, showing the disturbances that occur from the top plate losing contact with the bottom plate while the ball is rolling. The bottom figure shows a separate data set collected where the logic filter was applied, showing no disturbances in the ball position. A logic filter was chosen instead of other filter designs to ensure that there is no phase lag in the measurement of the ball position. While debouncing



can still occur in the filtered position signal, the occurrence of noise is lower than the occurrence of noise in the unfiltered reading and can be accommodated as a disturbance in Simulink<sup>®</sup> controller designs.



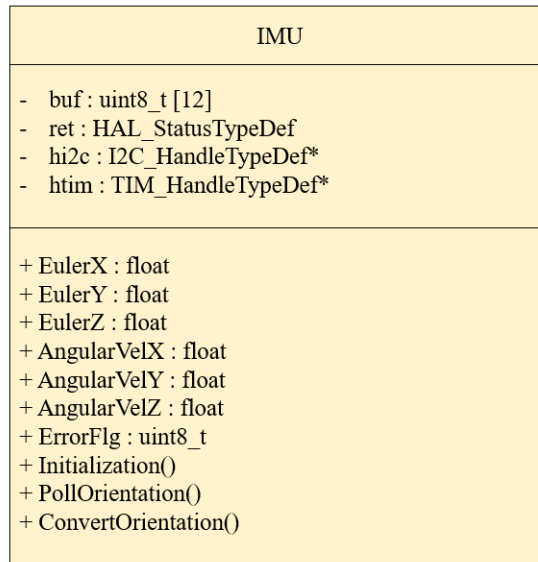
**Figure 5.4: Filtered Versus Unfiltered Touch Panel Data.**

Another important aspect of implementing a 5-pin touch panel is allowing the touch panel electrical system to settle before transitioning between taking x and y measurements. While using the native HAL command to delay reading the ADC by milliseconds would work for most projects, it is too slow to be able to have enough time for adequate filtering and other functions, such as handling important communication flows. Therefore, the use of DMA and interrupts was necessary to both accommodate the settling time of the touch panel while also allowing for non-blocking cooperative multitasking for necessary system functionality. The optimal length of time for the system to transition to steady state between switching from an accurate ADC reading of the X axis to the Y axis was quantified through characterization testing and found to be roughly 150 microseconds. This delay was implemented into the class through

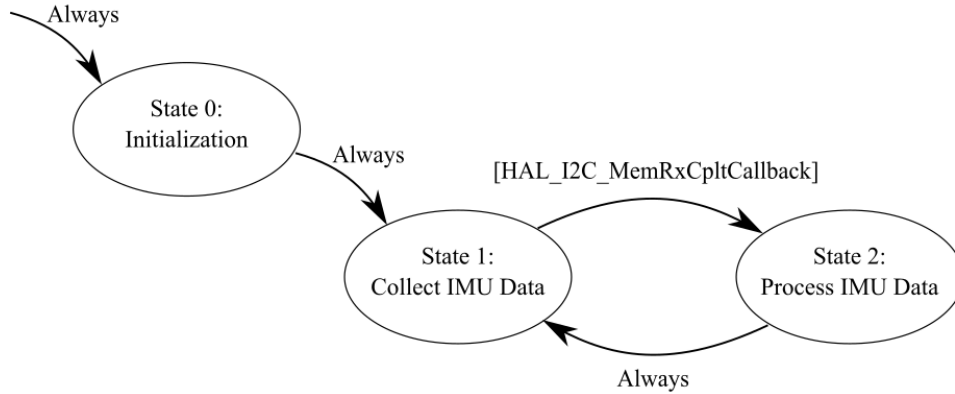
the use of a timer channel configured to increment in microseconds. Every time the ADC triggers a completed transfer flag, a timer channel is configured to trigger an interrupt at 150 [ $\mu s$ ], alternating between measuring the X and Y location of the ball.

### 5.2.2 IMU Class

The IMU class consists of utilizing the I<sup>2</sup>C connection between the STM32 and the BNO055 breakout board for periodic updating of the platform top plate Euler angles. The class object diagram is shown in Figure 5.5 and the finite state machine diagram is shown in Figure 5.6.



**Figure 5.5: IMU Class Object.**



**Figure 5.6: IMU Finite State Machine.**

This task implements the I<sup>2</sup>C HAL library commands to request the Euler angles and angular velocities at the IMU’s refresh frequency (100 Hz). This class/task also allows for configuration of the IMU using the registers and procedures detailed in the BNO055 datasheet procured from the Adafruit website. Re-configuring the IMU is necessary to adjust the orientation of the axes and units of the angular positions and velocities reported by the IMU. The IMU is sent a read command through one of the DMA controllers using the HAL command library, allowing for non-blocking reading of the desired IMU memory registers for Euler angles and gyroscopic angular velocities. The I<sup>2</sup>C communication occurs in the background as the other tasks are accomplished, such as touch panel sampling and TTL/USB serial communication. Once the binary response is fully stored in buffer, an interrupt flag is generated, triggering a flag to convert the buffer into the corresponding float values for the angular orientation and velocity.

Similar to the touch panel sensor, the IMU also requires calibration to correctly function. Unlike the touch panel, the IMU calibrates internally, automatically calibrating over time. The IMU is considered fully calibrated to the platform after all bits of the calibration stat register on the IMU are written to as '1'. After calibration is

complete, the calibration values are stored as hard-coded variables on the STM32 and written to the IMU offset and radius registers each time the STM32 is initialized. While writing to the internal flash of the STM32 would be ideal, the process to reliably write and read data to the internal STM32 flash was not able to be implemented in time.

One of the quirks of the BNO055 IMU is that it has issues with determining the true zero orientation of the top plate. To counteract this issue, once the platform is zeroed using the user interface procedure, the current orientation measurements of the IMU are written into the serial interface to be used as orientation offsets in Simulink<sup>®</sup> controller algorithms.

### **5.2.3 Communication Structure**

The STM32 acts as an intermediary between the ODrive motor driver and the MATLAB/Simulink<sup>®</sup> interface, and therefore needs to facilitate two forms of communication. To communicate with the host PC, the USB Device middleware supported by STM32CubeIDE was used. Of the various options available for communication with the ODrive, TTL serial using the DMA controller was chosen for communication between the STM32 and ODrive. Table 5.1 lists all the peripherals that use DMA and their priority on each DMA controller. While in operation, timing of the overall system communication is dictated by the MATLAB/Simulink<sup>®</sup> model, with the STM32 executing a reply message and a forwarded command message to the host PC and ODrive respectively upon receiving a message from the host PC. This allows for flexibility of implementing Simulink<sup>®</sup> controller designs with various timings, as the ODrive can facilitate various styles of control such as position, velocity, and torque control.

**Table 5.1: DMA Allocation and Priority.**

DMA Controller	Stream	Peripheral	Priority
DMA1	Stream 0	I2C1_RX	Low
DMA2	Stream 7	USART1_TX	Very High
	Stream 2	USART1_RX	Medium
	Stream 0	ADC1	Low

A designated packet period of 5 [ms] was chosen for overall system communication because it allows for periods of ample touch panel data collection and processing, and it supports system functionality with the timeliness of the IMU at an update cycle of 100 [Hz].

#### 5.2.4 User Interface Function

Since this device is required to be operated manually for setup and configuration, a serial user interface was chosen. A finite state machine for the user interface is shown in Figure 5.7. This interface walks the user through the platform setup process, including commutating the BLDC motors and zeroing the platforming. During the zeroing process, the ODrive system is commanded to position control mode of the BLDC motors, utilizing the on-board cascaded controller with controller gains found through previous experimental testing. The user zeroes the platform through the use of a bubble level puck that can be placed anywhere on the top plate of the platform. Using 'W', 'A', 'S', and 'D' serial inputs from the host PC changes the position set-point of the motors, tilting the platform in the direction commanded by the user. This process is designed to be used in tandem with a portable bubble level puck laying on the top plate, shown in Figure 5.8. Once the platform is considered level to the user, the user can disconnect from the serial interface and proceed to run their Simulink®

controller design, with the ODrive motor driver automatically switching to torque control upon receiving the command from the STM32 TTL serial communication.

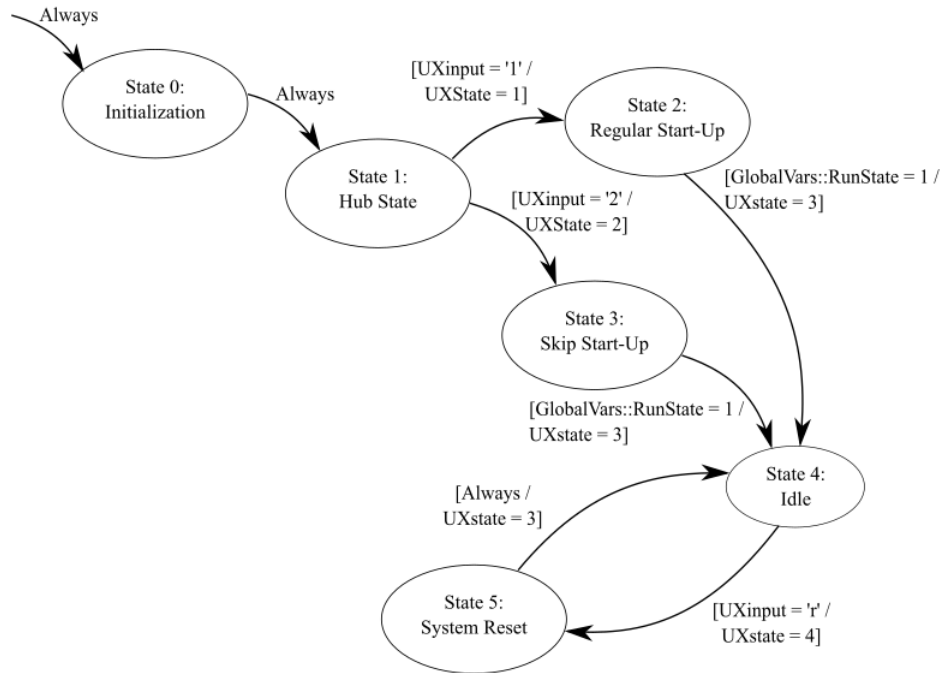


Figure 5.7: UI Finite State Machine.

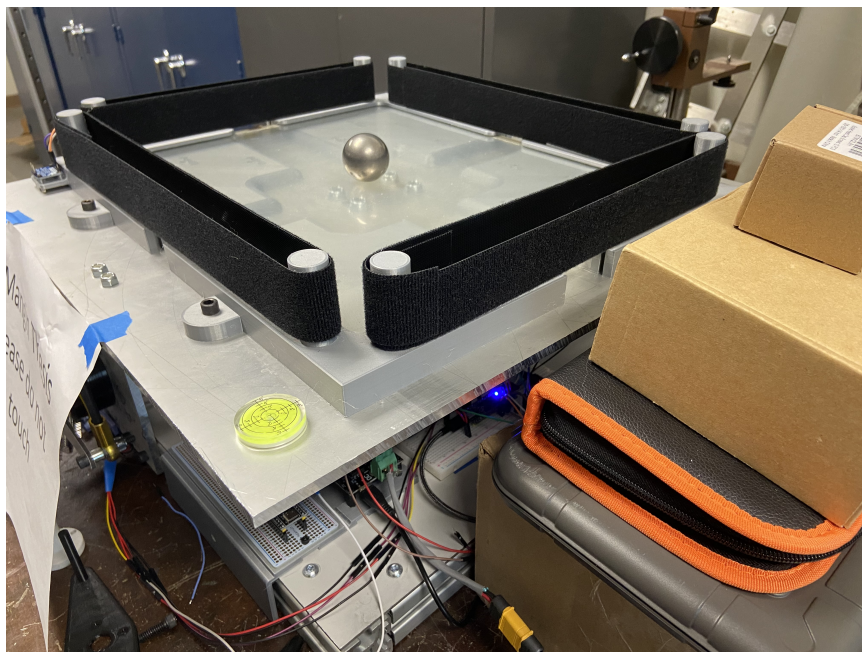


Figure 5.8: Platform Ball Balanced Using Bubble Level.

### 5.2.5 Data Collection

Data from the platform is stored through the use of the Simulink Desktop Real-Time™ model because the platform is tailored for use with the Simulink® environment. Typical “To Workspace” blocks are not supported in the real-time environment. Instead, each signal that the user would like to save must be logged.

## 5.3 ODrive Software

On top of the software required for the STM32, the ODrive Motor Driver also utilizes an STM32F405 MCU that can have important system parameters programmed through TTL or USB serial. Since this hardware package is from an open-source project, the code and ample documentation of how to use the code can be found on their website and GitHub pages. An important aspect of this being an open-source project is that new software revisions happen periodically throughout the year, which required updating the firmware on the ODrive through STM serial debuggers to ensure that the latest firmware version with all the newest features can be utilized. The active nature of this project required updating the firmware several times throughout the duration of the project to ensure functionality of key features utilized during the project, such as issuing commands to write to on-board ODrive parameters through TTL serial.

The ODrive also comes with a fully-featured on-board configuration management library that can be accessed through Python (using Anaconda for this project). This connectivity allows for changing parameters and even live plotting through the live-plotter library in Python. This accessibility was integral for device troubleshooting and integration. However, since the ODrive project has on-going development due to

its open source nature, much of the documentation on different parameters is incomplete. This necessitated analysis of the open-source firmware on GitHub for a better understanding of the system’s internal logic.

### 5.3.1 ODrive Firmware Configuration

When utilizing the ODrive motor driver system, the on-board STM32F4 MCU has its own unique parameters that must be tailored for full functionality with the ball-and-plate balancing platform. These include parameters such as end stop configuration, UART configuration, and many more. These parameter changes are handled through the UART chip using the TTL serial ASCII protocol. The motor commands are driven through the same TTL serial channel, commanding the ODrive motor on each axis to some value between  $-1.25 [N\text{-}m]$  to  $1.25 [N\text{-}m]$ . Due to using a high-resistance, low-current gimbal motor, this functionality had to be characterized to match the state of the hardware.

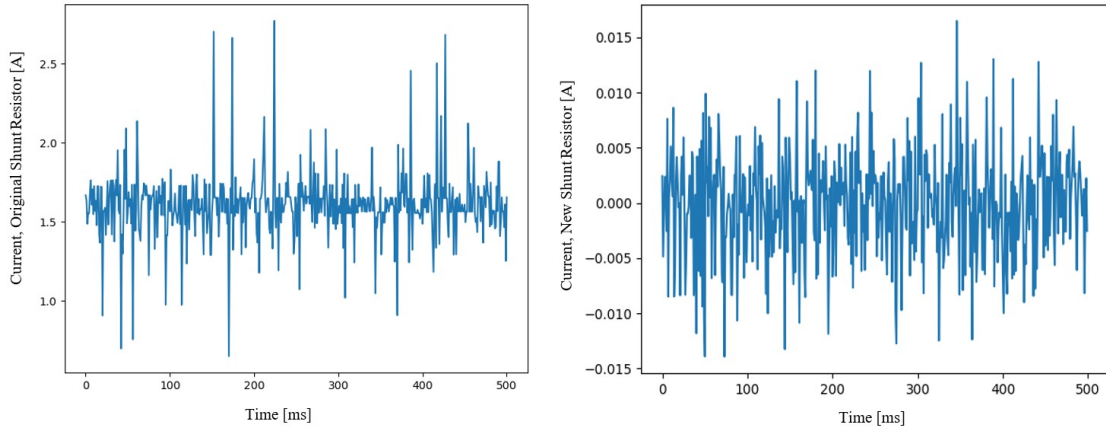
Since the ODrive motor driver is open-source, all source code for each release version can be reviewed to fully understand the software logic. This was especially vital for the integration of gimbal motors, as many system features are re-purposed for these types of motors due to their comparatively high phase-to-phase resistance and low current throughput than their design target. For instance, in initial testing with calibrating the motor through a commanded state on the ODrive, key motor parameters, such as phase resistance and inductance, were not calibrated. When analyzing the source code on the ODrive GitHub, it was found that, due to the previously mentioned chosen shunt resistors, accurate motor parameterization cannot be performed for gimbal motors. Instead, when the motor type is set to gimbal motors, the calibration sequence skips calibrating the motors. Many disparities between the software functionality of the gimbal motor type versus the high-current motor types



were explained through public forum posts and small sections of website documentation. Since the shunt resistor replacement was successful, many of these disparities found between the high current motor and gimbal motor mode no longer applied to the configuration of the ODrive.

### 5.3.2 ODrive Motor Control

The ODrive motor driver has several avenues for motor control. For high-current motors that are more in-line with the original design philosophy of the ODrive motor driver (motors that possess low-resistance with current requirements of 30-90 [A]), control via high-current mode is straightforward to implement. The use of gimbal type motors, however, poses a challenge. The ODrive possesses current shunt resistors that monitor each motor axis for the current applied to each motor; the shunt resistors provided with the ODrive have signal noise of  $\pm 1$  [A], which would be small for high currents. However, this amount of noise is not suitable for the current control of low-current motors. Therefore, the shunt resistors of the ODrive were replaced to allow for current control mode to be supported for the chosen BLDC motors. A comparison of the noise in the  $I_q$  measurement, or the measurement of current in-phase with the output of the BLDC motor stators, before and after replacing the shunt resistor can be seen in Figure 5.9. The left graph shows the noise in the current measurement for the original shunt resistance, and the right graph shows the noise in the current measurement for the replaced shunt resistors.



**Figure 5.9:  $I_q$  Current Measurement Noise Comparison.**

### 5.3.3 ODrive Motor Driver Firmware Development

Since the shunt resistors were changed to minimize the noise found on the current measurement, the resistances of the shunt resistor had to be updated in the ODrive firmware. This required changing the shunt resistance definition in the ODrive source code to the replaced shunt resistance value of  $0.020 \Omega$ . When attempting to set up the ODrive software toolchain to properly compile the firmware to flash it to the ODrive, the required dependency repositories, such as Tup, were abnormally difficult to install on a Windows OS. To fix this issue, configuring a laptop to dual-boot between the Ubuntu Linux OS and Windows OS was paramount. The Linux instruction for ODrive development were simple and straightforward, and compiling the ODrive firmware from the makefile was successful. An interesting aspect of working with the ODrive on Linux was also that the odrivetool, also available on Windows, had more features, signaling that the development environment targeted by the developers is through Linux.

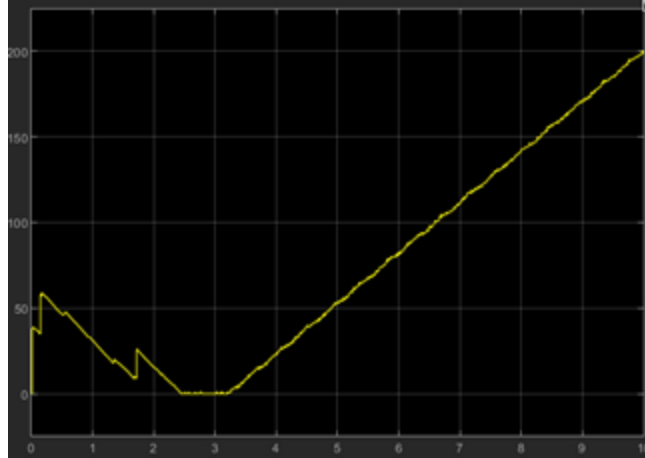
## 5.4 Host PC Firmware

Two options were considered for host PC firmware: a Python program, or Simulink Desktop Real-Time™. Simulink® was chosen as the preferred method because of its ubiquity in the development of simulated control systems in the Mechanical Engineering curriculum.

### 5.4.1 MATLAB/Simulink®

The goal of accommodating for a Simulink®-based control system is to allow for seamless translation between Simulink® models constructed with simulated plants to being able to use the same controller design during hardware testing. This goal was accomplished through the use of ASCII serial communication between the STM32 and MATLAB/Simulink® program.

To implement MATLAB/Simulink® functionality into the model, the use of the Simulink Desktop Real-Time™ solution was necessary. Without this integration, real-time synchronization between the model and the STM32 hardware would be highly inconsistent at best. This was apparent during preliminary testing; there was a linear relationship of missed CPU ticks over time, as shown Figure 5.10, where the y-axis shows the number of missed CPU ticks and the x-axis shows the duration of the test at ten seconds. Missed CPU ticks cause a desynchronization between the model clock and the clock of the host PC. This, along with inconsistent readings of the serial communication, underscored the necessity of using Simulink Real-Time™ functionality.

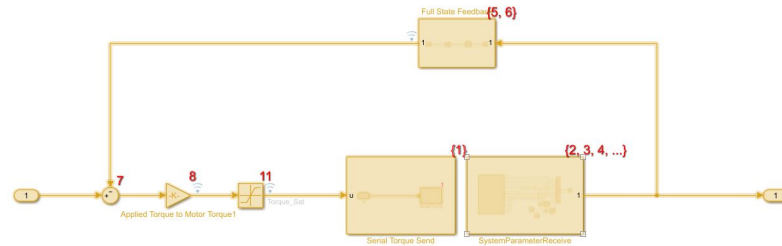


**Figure 5.10: Simulink<sup>®</sup> Scope Output of Missed CPU Ticks.**

When utilizing the Simulink Desktop Real-Time<sup>™</sup> system, several adjustments to the model were required. The use of the serial send/receive blocks was not applicable for a rapid-accelerated or Simulink Desktop Real-Time<sup>™</sup> systems. Instead, the use of Stream Input/Output is compliant with Simulink Desktop Real-Time<sup>™</sup> requirements. An additional modification that was necessary was in the process to store output data from the model. The use of the Data Analyzer was necessary for this project, as the “To Workspace” block is not supported when a Simulink<sup>™</sup> model runs in desktop real-time operations.

Another important aspect to consider about Simulink Desktop Real-Time<sup>™</sup> models is the execution order of the model. By default, the execution order of blocks within a Simulink<sup>®</sup> model can widely differ from the ideal execution order; however, the execution order can be influenced through several methods. The method that is implemented in this thesis was through the use of atomic subgroups. Using atomic subgroups, the execution of important blocks can be controlled. Using an information overlay, the execution order of Simulink<sup>®</sup> blocks can be seen on the top-right of each block shown in Figure 5.11, where all the relevant blocks to process the message received from the STM32 are grouped into one atomic subgroup to order the process-

ing of pertinent parameters for torque control before other blocks tailored for data collection.



**Figure 5.11: Execution Order of Simulink<sup>®</sup> Blocks.**

#### 5.4.2 Limitations of Simulink<sup>®</sup> Integration

While there are many benefits to utilizing Simulink<sup>®</sup> for testing controller designs, there are some limitations that must be taken into account. As covered in the previous section, many blocks are incompatible with use for Real-Time applications. The use of conventional blocks, like serial receive and send, are not Simulink Real-Time<sup>™</sup> kernel compatible. Therefore, future controllers must be tailored to work within the requirements of a Simulink Real-Time<sup>™</sup> environment. Since the testing controller utilized a simple linear regulator model for hardware testing, complex blocks were not included in the model. Instead, many parameters and necessary matrices were pre-calculated through system parameters found through the SolidWorks model. Instead of utilizing position-control, this platform utilizes direct torque control. This approach requires all aspects of the system to be much quicker than if commanding the system through a standard outer-loop position control algorithm; typically, torque control is handled in a faster, inner loop of a controller design to ensure system stability. However, since the amount of data transferred is limited, fast response speeds can be achieved to lower the effect that this has on the platform dynamics.

## Chapter 6

### HARDWARE TESTING

#### 6.1 Platform Requirements Overview

The hardware performance requirements for the ball-and-plate platform prototype are the following:

1. The complete ball-and-plate platform system shall be able to support an update frequency of 200 [Hz] at a minimum.
2. The ball-and-plate platform system shall produce the torque commanded by the Simulink<sup>®</sup> system to within 5%.

The following sections describe the results of the verification testing conducted to verify that the hardware performance requirements were satisfied.

#### 6.2 Motor Performance Testing

Motor performance testing consisted of testing the torque command step input response of the ODrive motor driver, determining the experimental torque constant of each BLDC motor, and characterizing the torque output of both motors over a range of values within the valid range of torque commands. All data described in this section can be found in Appendix C. All confidence intervals for the true mean are found using a 95% confidence and calculated using Minitab's One-Sample T function.

### 6.2.1 Torque Step Response

An important aspect of the ball-and-plate balancing prototype is to ensure the immediacy of the torque input response. To analyze this aspect of the platform, the  $I_q$  setpoint and  $I_q$  measurement were plotted using the built-in liveplotter tool from the odrivetool program for Python (an implementation of the Matplotlib Python library). Figure 6.1 shows the resulting torque step response when the motor is commanded from 0 to 1.2 [A], appearing to show a minimal time constant and minimal percent overshoot in response to the current setpoint command. However, this graph does not fully characterize the torque step response for important parameters such as the time constant, slew rate, and percent overshoot. Despite the lack of quantitative characterization, this result qualitatively shows that the ODrive is successful at handling torque inputs.

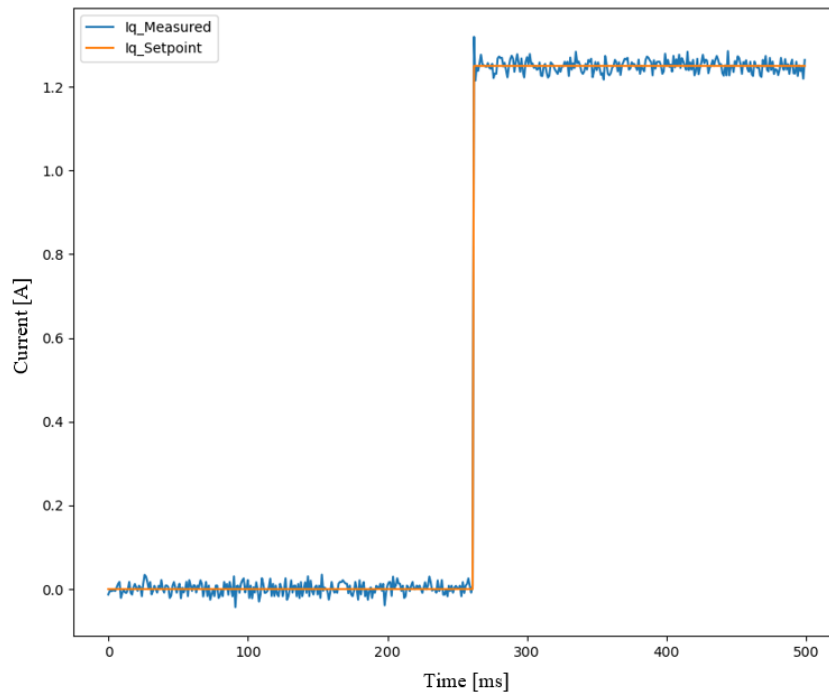
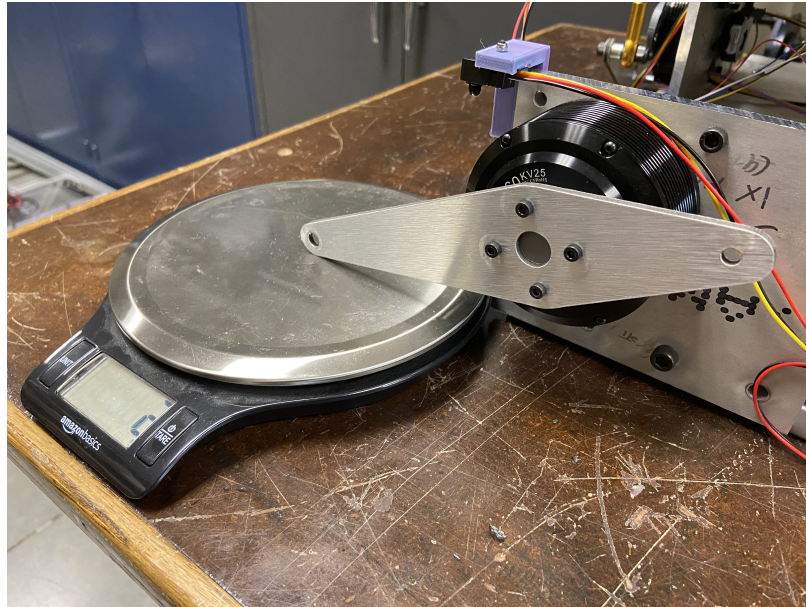


Figure 6.1: Step Response, Torque Command in Current Control Mode.

### 6.2.2 Torque Constant Testing

To verify that the platform will produce the commanded torque to within 5%, it was necessary to experimentally determine the precise torque constant of each motor axis. Several torque constants were tested for each motor axis, starting with the manufacturer's specification at  $0.4 [N\cdot m/A]$ . The experimental testing consisted of driving the platform motor arm into the center of a kitchen scale, converting the gram-force read by the scale into the torque output of the motor through the use of the horizontal distance between the point of contact and the center of the motor shaft, or the effective torque arm. The kitchen scale used is from the Amazon Basics line of products. The testing setup is shown in Figure 6.2.

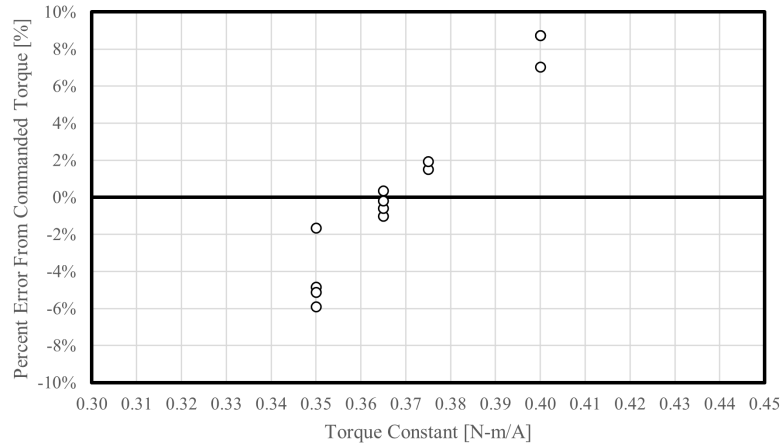


**Figure 6.2: Torque Test Experimental Setup.**

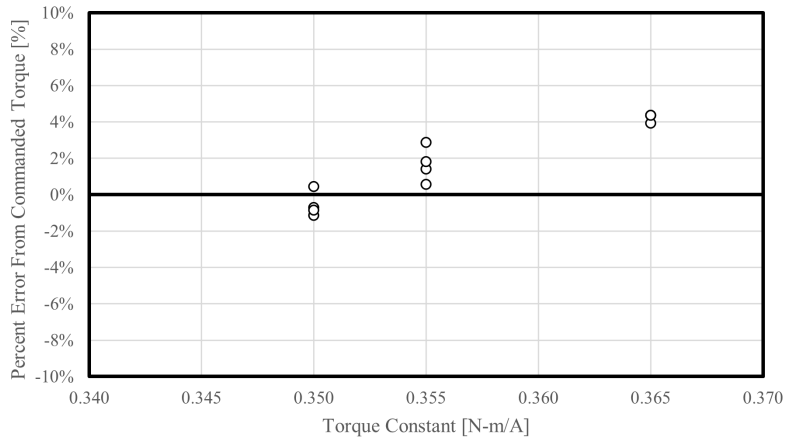
Figure 6.3 shows the results for motor axis 0 when tested for the experimental torque constant. Finding the experimental torque constant consisted of sending several torque commands of varying magnitude for each torque constant tested and plotting the percent error between the torque command and the experimental torque



output at each torque constant. As shown by the figure, the experimental torque constant for current control of the motor on axis 0 of the ODrive motor driver was found to be 0.365 [N-m/A]. Figure 6.3 shows the results for the motor on axis 1 using the same test procedure; the experimental torque constant for current control for the motor on axis 1 was found to be 0.35 [N-m/A].



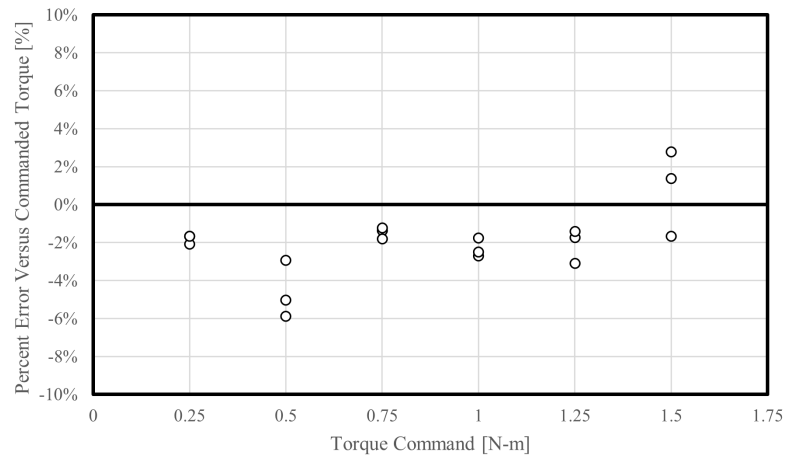
**Figure 6.3: Experimental Torque Constants for Axis 0 Motor.**



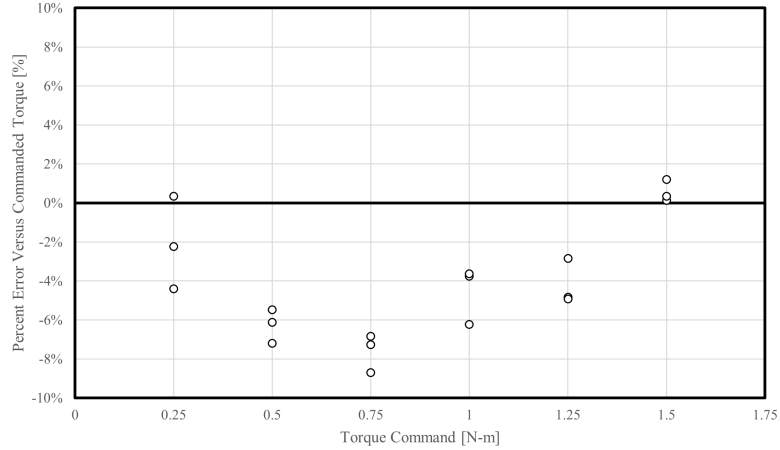
**Figure 6.4: Experimental Torque Constants for Axis 1 Motor.**

### 6.2.3 Motor Torque Testing - Gimbal Mode

Even though the ODrive shunt resistors were successfully replaced, the gimbal motor mode was also analyzed for torque output. The testing procedure for the gimbal mode torque testing utilized the same experimental setup as shown in Figure 6.2. Three experimental torque outputs were taken for each commanded torque, and the experimental torque output was compared to the commanded torque output for each data point. Figure 6.5 summarizes the test results for the axis 0 motor, and Figure 6.6 summarizes the test results for the axis 1 motor.



**Figure 6.5: Percent Error of Torque Output from Commanded Torque, Gimbal Mode, Axis 0.**



**Figure 6.6: Percent Error of Torque Output from Commanded Torque, Gimbal Mode, Axis 1.**

The importance of modifying the ODrive shunt resistors to utilize current control was found during gimbal mode torque testing, as at higher torque commands, the increase in current increased the temperature of the internal windings faster than expected. As such, the internal resistance within the motor increased and the magnetic field induced in the motor decreased, causing the  $I_q$  current output through the motor axis to rapidly decrease; this effect can be seen in the data output shown in Figure 6.7. This is because, since the gimbal motor control of the ODrive utilizes voltage as the parameter to control, using Ohm's law, if the resistance in the motor increases, the current supplied to the motor must decrease. The results of these tests show that, even within gimbal mode, the motors are able to produce the desired torque output for a wide range of torques provided that the motor windings do not heat up significantly.

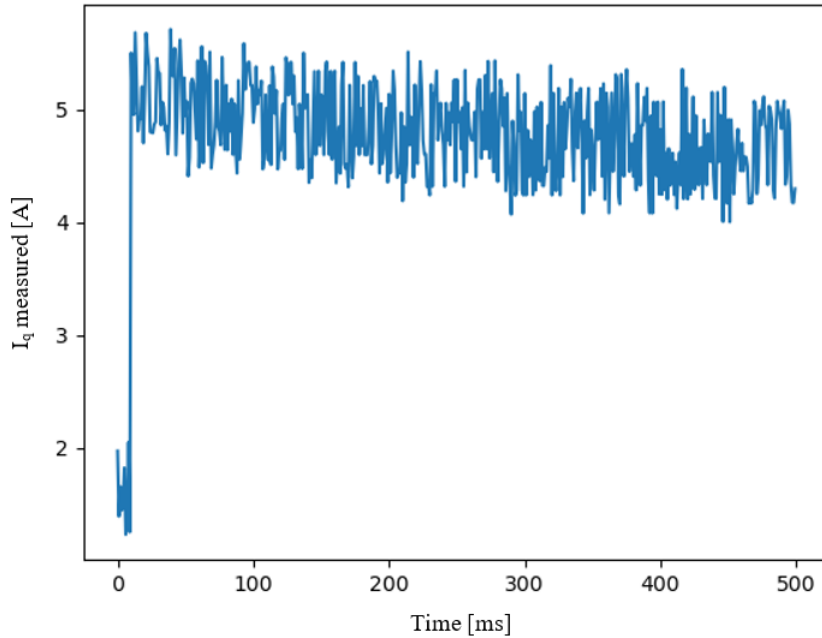
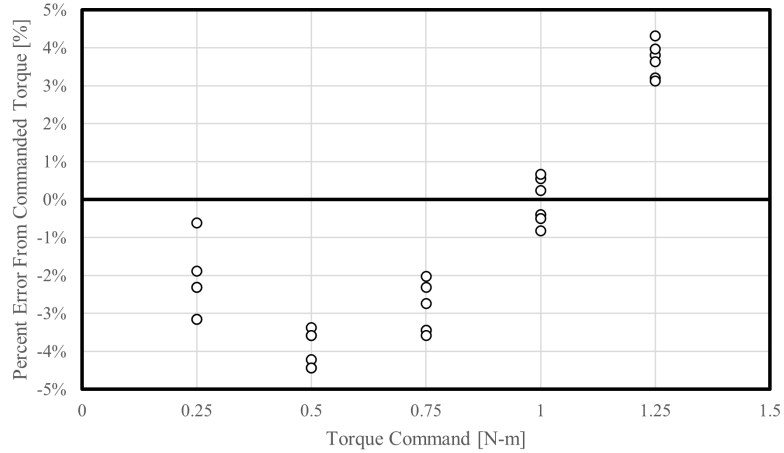


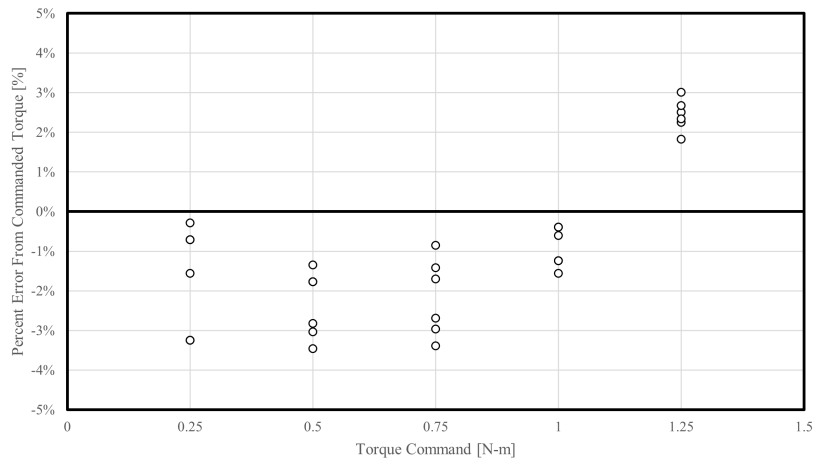
Figure 6.7: Stall  $I_q$  Output in Gimbal Mode, ODrive.

#### 6.2.4 Motor Torque Testing - Current Control Mode

Similar to the testing procedure done in the gimbal mode, the torque control when operating in current mode was also tested. For this testing procedure, six data points were taken for each torque command tested, with three data points being taken in ascending order and three data points taken in descending order. This was to ensure that the internal temperature of the motor was accounted for in the torque results. Figure 6.8 shows the torque testing results for the motor on axis 0, and Figure 6.9 shows the torque testing results for the motor on axis 1. Unlike the gimbal mode torque results, the current control mode was able to stay within the desired 5% error range from the commanded torque value.



**Figure 6.8: Percent Error of Torque Output from Commanded Torque, Current Control, Axis 0.**



**Figure 6.9: Percent Error of Torque Output from Commanded Torque, Current Control, Axis 1.**

### 6.3 Software Timing Characterization

Another vital aspect of ensuring that this project was a success was in verification testing of the timing of each subsystem. Typical position control outer loops can take place over a slower period of time when compensated for with a faster inner motor control loop. However, since this project is focusing on developing a system

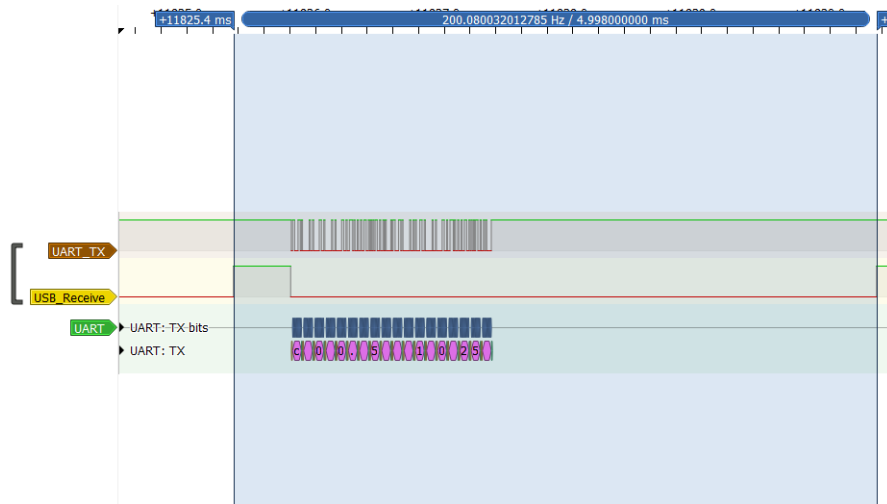
to facilitate torque control, rapid communication between all system components is necessary to minimize system instability. To analyze system timing, a logic analyzer and dedicated GPIO pin were utilized to characterize the timing for different tasks; by setting the output of the dedicated GPIO pin to HIGH (+5V) or LOW (GND), timing of tasks can be analyzed through the use of data collection software. The chosen logic analyzer was the KeeYees 24 MHz USB Logic Analyzer. The data collection software chosen is a common open-source software known as PulseView, developed as a part of the sigrok project. This software is useful for a variety of reasons, but one of the most useful aspects of this software is its built-in communication protocol decoder, whereby messages can be decoded from the TTL serial or I<sup>2</sup>C signal lines.

### 6.3.1 Communication Timing

There are several communication channels utilized for this project: serial over USB between the host PC and STM32 MCU, I<sup>2</sup>C between the IMU and STM32 MCU, and TTL serial between the STM32 MCU and the ODrive motor driver system. An important detail between all of these communication channels is that, no matter how fast or slow a channel is communicating at, since all communication channels have been off-loaded from the CPU to two DMA controllers, the communication is not impacting the performance of the CPU in calculation-based tasks, such as in the filtering of the touch panel position and velocity data. Therefore, when demonstrating the timing of UART workflows, note that the selected UART baudrate, 115200, is slower than the finalized UART baudrate, 410800, because of the limited capture rate of the chosen logic analyzer when communication is occurring at a high baudrate.

First, the overall system timing was analyzed for timeliness based on its compatibility with handling periodic communication from the Simulink<sup>®</sup> model at 200 [Hz]. Figure 6.10 shows the general timing of messages being received from the host PC, where

the captured signal on the bottom utilizes the dedicated GPIO pin to capture the timing of system components. For this test, the pin output is set to HIGH when the message is received in the USB buffer, and the pin output is set to LOW after the DMA is set to transfer the motor torque command over TTL serial to the ODrive motor driver. In this captured snippet, the time difference between receiving the two USB messages was 4.998 [ms]. When a test period of five seconds was analyzed for variations in this timing difference, the timing between received messages was found to be:  $4.9995 \pm 0.0054$  [ms]. This result shows that the STM32 is able to successfully handle the desired periodic communication timing of 5 [ms].



**Figure 6.10: Time Duration Between Host PC USB Messages.**

Another important aspect of the communication system is the period of time between when the STM32 receives a USB message in buffer, and when it is able to process this message, send a message back to the Simulink<sup>®</sup> system, and finally send the necessary torque command over TTL serial. Figure 6.11 shows the discretization of the response to a serial message from the host PC, with Table 6.1 showing an average and standard deviation for the amount of time taken up by each section utilizing several data points from a five second live test.



Figure 6.11: Sections of Response Upon Receiving USB Message.

Table 6.1: USB Response Overview and Timings.

Section	Description	Average Time [ $\mu s$ ]	Standard Deviation [ $\mu s$ ]
1	sprintf	421.33	18.65
2	USB CDC Response	33.90	0.16
3	Torque Command Process	160.57	0.39
-	Overall Period Timing	615.80	18.63

Section one highlights the time required for the sprintf function to compile all relevant system parameters into a string to be sent back to the Simulink<sup>®</sup> model. The time that the sprintf function requires to accomplish this task can be represented as:  $421.33 \pm 13.35$  [ $\mu s$ ]. Since the system parameters are represented as floats and can vary in magnitude, this section creates a message of variable length. This variance is the reason for small timing differences in the response message to the host PC and the torque command for the ODrive.

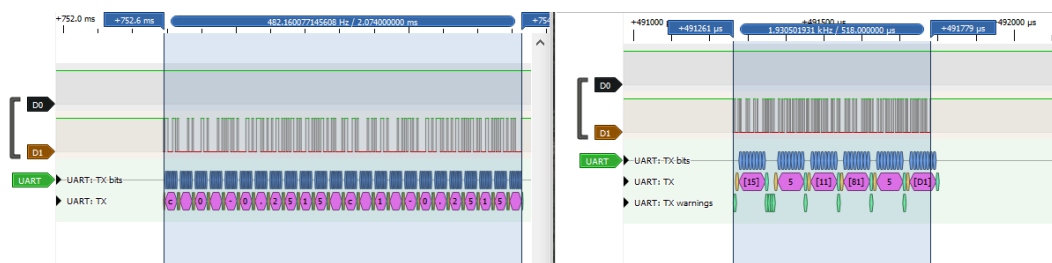
Section two highlights the time required to send the completed string from the previous section back to the Simulink<sup>®</sup> model. The timing for this section of the function was found to be:  $33.901 \pm 0.114$  [ $\mu s$ ].



Section three highlights the time required to process the incoming torque command string into two torque commands that are then sent over TTL serial to the ODrive. The timing for this section of the function was found to be:  $160.57 \pm 0.23 [\mu s]$ .

Overall, the timing for the communication response function is:  $615.80 \pm 13.33 [\mu s]$ . As mentioned previously, the biggest contribution to the variance of the time required to fully process the torque command and USB response is due to the variance of the float parameters being processed by the sprintf function.

Lastly, the baudrate of the serial connection between the ODrive and STM32 has a significant impact on the timeliness of the communication system. Figure 6.12 shows the difference in timing between a TTL serial signal comprised of 115200 baudrate (left side) and 460800 (right side). As the comparison test utilized the longest possible message, comprised of a constant negative torque value with four decimal places, there is no variation in message duration. The message utilizing the 115200 baudrate takes a time duration 2074  $[\mu s]$ , and the message utilizing the 460800 baudrate takes a time duration of 518  $[\mu s]$ . This result is expected but important to verify, as 460800 baudrate is four times higher than 115200 baudrate (with some microsecond variance).



**Figure 6.12: Comparison Between Baudrates.**

### 6.3.2 IMU Timing

The IMU system utilizes one of the STM32 timer channels to regulate the IMU polling to 10 [ms]. This was verified through the use of the logic analyzer, an example of the situation shown in Figure 6.13. Analyzing ten samples in a regular test duration, the time duration between sampling the IMU is:  $10.011 \pm 0.018$  [ms].

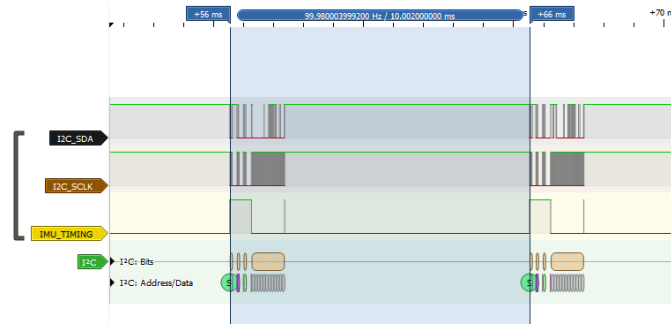


Figure 6.13: Cycle Time Between I<sup>2</sup>C Messages.

When looking at the timing of the IMU communication, several aspects are apparent. One aspect that is interesting to note is how the DMA aspect of the I<sup>2</sup>C STM32 code works. I<sup>2</sup>C protocol requires several read/write procedures to set up to read desired registers from the IMU device, with the STM32 utilizing blocking-mode I<sup>2</sup>C for set up. Despite this quirk of using DMA to handle I<sup>2</sup>C communications, the overall CPU performance hit is minimal due to this process only occurring once every 10 [ms]. Figure 6.14 shows an example of the setup period of reading the desired IMU registers. Over a sample period of five seconds, the period that facilitated setting up the DMA for I<sup>2</sup>C communication occurred for:  $680.40 \pm 23.80$  [ $\mu$ s]. The final time period shown in Figure 6.14 of the IMU\_Timing signal line shows the time duration when converting the received buffer values for the top plate orientation into angle and angular velocities. This process takes on average  $3.500 \pm 0.377$  [ $\mu$ s].

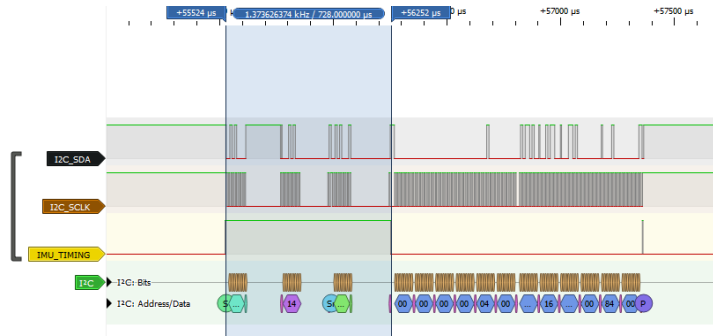


Figure 6.14: Example of I<sup>2</sup>C Setup Time Duration.

### 6.3.3 Touch Panel Timing

Akin to the IMU, the touch panel utilizes a timer channel and DMA to collect and process the raw voltage data from the ADC into the ball's approximate position and velocity. Following similar data collection procedures to previous sections, the touch panel's overall timeliness and CPU overhead were characterized. Over a sample period of five seconds, the time required to fully collect the voltage readout for the x and y axis of the touch panel was:  $371.60 \pm 0.91 [\mu s]$ . However, this time also includes the time required for the touch panel to come to steady state upon switching the sensing axis. In actuality, the amount of computing time to set up the DMA for this data collection occurs for:  $9.6 \pm 0.6 [\mu s]$ .

The touch panel class takes up a large portion of processing time when it comes to processing the ADC data into the ball position and velocity. The time required for this process was found to be:  $1.96 \pm 0.23 [ms]$ . The majority of this time is spent on the merge sort algorithm to allow for the median filtering of the ADC values. This value is still within acceptable system timeliness, as several position data measurements can be processed within the 5 [ms] communication round-trip time.

## Chapter 7

### CONCLUSIONS AND FUTURE WORK

This platform is the first iteration of the overhaul for the existing ball-and-plate balancing system designed for the ME 305 and ME 405 courses at California Polytechnic University, San Luis Obispo, and many lessons were learned in its creation.

The O-Drive motor driver was chosen both for its affordability and its access to many features not found on competing BLDC-compatible motor drivers. However, due to its open-source nature, much of the documentation for system functionality was lacking in some important details, and troubleshooting required engagement with the company's forum posts. While a document explaining the hurdles that had to be overcome was created as a reference for future project work with this platform design, other commercial motor drivers also should be considered if simplicity of operation is desired over cost-effectiveness and a plethora of features. Another issue occurred when building and flashing custom firmware to the ODrive using the Windows OS, as many of the dependencies have no easily-available instructions for proper installation of each dependency. Installing Ubuntu to dual-boot between Linux and Windows fixed this issue, as the Linux instructions are straightforward due to the availability of all listed repositories.

The communication standard used for the communications between the host PC, STM32, and the O-Drive was primarily driven through the TTL serial ASCII and USB CDC interfaces. While using the ASCII format is simpler implement, and the speed of the ASCII interface was adequate for the purposes of this project, a binary-

based serial implementation would be vastly more efficient for both communication-line speeds and message decomposition.

This platform iteration did not incorporate a PCB design for component integration, as many of the components utilized on the platform were break-out board devices (STM32 MCU, IMU, O-Drive). This design decision led to requiring careful wire management to minimize any effects from potential electromagnetic interference. However, a PCB tailored to the ball-and-plate platform would allow for reduced noise in the TTL serial connection between the STM32 and the ODrive.

## BIBLIOGRAPHY

- [1] ASIAA. “Instrumentation Overview.” the Yuan Tseh Lee Array (Formerly AMiBA), 20 Oct. 2017, <http://ytla.asiaa.sinica.edu.tw/instru.php>.
- [2] Awtar, Shorya, et al. “Mechatronic Design of a Ball-on-Plate Balancing System.” *Mechatronics*, vol. 12, no. 2, 2002, pp. 217–228., [https://doi.org/10.1016/s0957-4158\(01\)00062-9](https://doi.org/10.1016/s0957-4158(01)00062-9).
- [3] Bang, Heeseung and Lee, Young Sam. “Implementation of a Ball and Plate Control System Using Sliding Mode Control.” *IEEE Access*, vol. 6, 21 May 2018, pp. 32401–32408., <https://doi.org/10.1109/access.2018.2838544>.
- [4] Mathema, Chitiz. “What’s the Difference Between Resistive and Capacitive Touchscreens?” *Electronic Design*, 17 Apr. 2015, <https://www.electronicdesign.com/technologies/displays/article/21800710/whats-the-difference-between-resistive-and-capacitive-touchscreens>.
- [5] ODrive Robotics. “Control Structure and Tuning.” ODrive Documentation, 2021, <https://docs.odriverobotics.com/v/latest/control.html>.
- [6] Richter, Zachary. “Nonlinear Model Development and Validation for Ball and Plate Control System.” *California Polytechnic University*, Digital Commons, 2021. <https://digitalcommons.calpoly.edu/theses/2383/>.
- [7] Zeeshan, A., et al. “Design, Control and Implementation of a Ball on Plate Balancing System.” *Proceedings of 2012 9th International Bhurban Conference on Applied Sciences & Technology (IBCAST)*, Jan. 2012, <https://doi.org/10.1109/ibcast.2012.6177520>.

## APPENDICES

### Appendix A

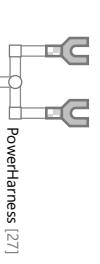
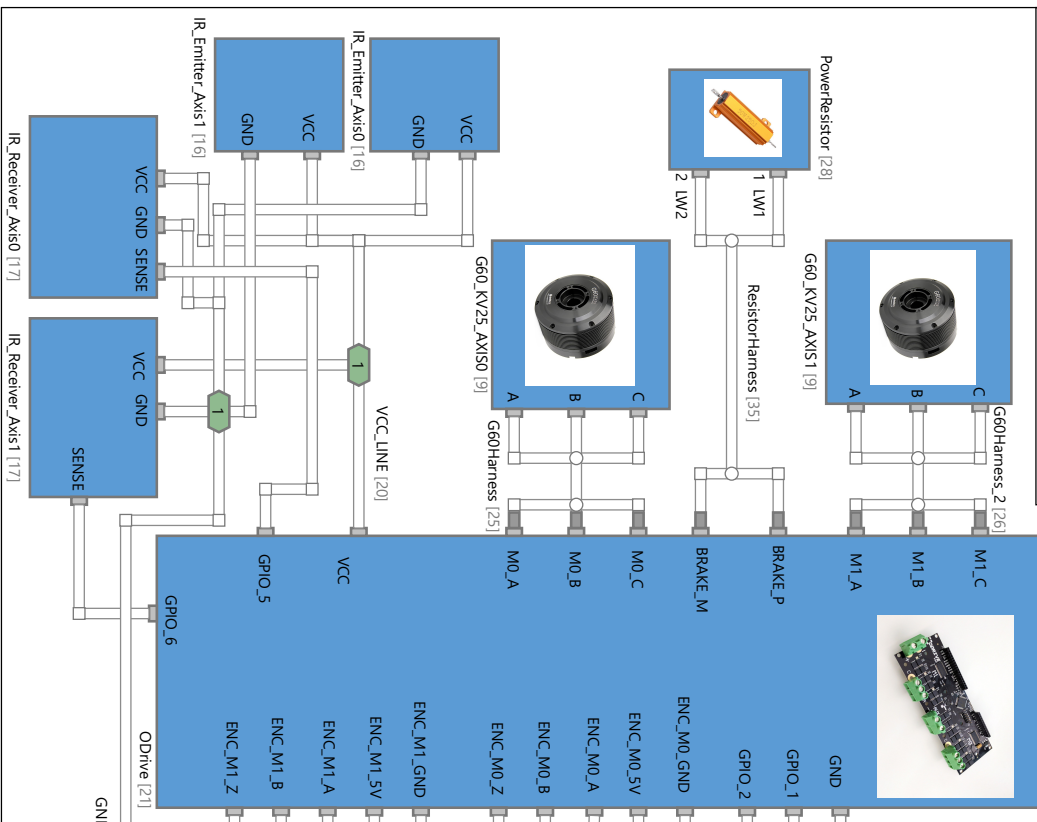
#### WIRING DIAGRAM

Revisions

Rev	Date	Author	Description
A	3/4/2022	Scott Mangin	

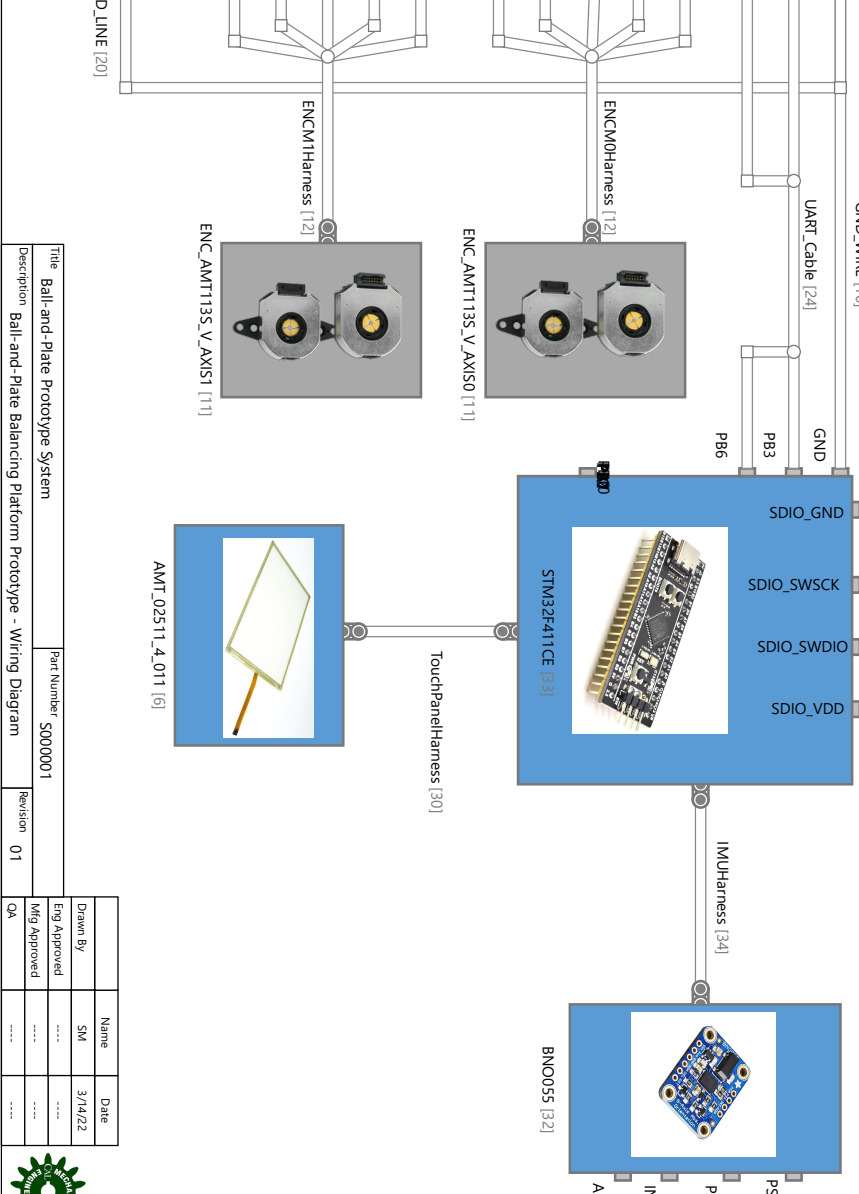
Notes

Notes
-------



**Bill of Materials**

Id	Type	Manufacturer	Part Number	Quantity
6	Device	AM Touch	02511-4-011	1
9	Device	T Motor	G60	2
10	Harness	N/A	28 Gauge Wire	5
11	Device	Anaheim Automation	ENC-AMT1135-V	2
12	Harness	N/A	Encoder Connector	2
16	Device	Adafruit	BreakBeamSensor_Emitter	2
17	Device	Adafruit	IR_Signal_Bundle	2
20	Harness	N/A	IR_Signal_Bundle	2
21	Device	Robotics	ODrive	1
24	Harness	General Cable	CE20035G-100	1
25	Harness	N/A	Motor/Harness	1
26	Harness	N/A	Motor/Harness	1
27	Harness	N/A	PowerHarness	1
28	Device	N/A	50W/RSI	1
30	Harness	N/A	TouchPanelHarness	1
32	Device	Adafruit	BNO055	1
33	Device	W&A	STM32F41TCE	1
34	Harness	N/A	IMUHarness	1
35	Harness	N/A	ResistorHarness	1



Title	Ball-and-Plate Prototype System	Part Number	S000001
Description	Ball-and-Plate Balancing Platform Prototype - Wiring Diagram	Revision	01

Drawn By	Name	Date
SM	SM	3/14/22






Revisions

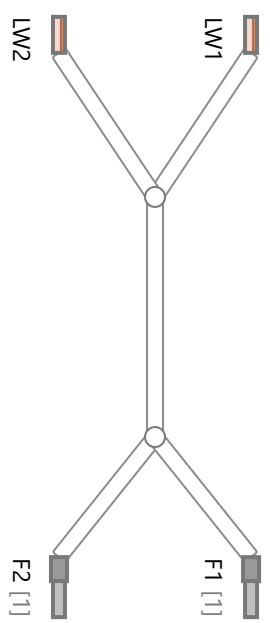
Rev.	Date	Author	Description
A	3/16/2022	Scott Mangin	


Bill of Materials

Id	Type	Manufacturer	Part Number	Quantity
1	Ferrule	N/A	14-12 AWG Gray	2
2	Wire	N/A	UL1015 12 AWG Violet	2

From	To	Conductor	Color	Gauge
LW1	F1	W1.Violet		12 AWG

From	To	Conductor	Color	Gauge
LW2	F2	W2.Violet		12 AWG



From	To	Conductor	Color	Gauge
F1	LW1	W1.Violet		12 AWG

From	To	Conductor	Color	Gauge
F2	LW2	W2.Violet		12 AWG

Title	Power Resistor Harness	Part Number	H001
Description	N/A	Revision	01

	Name	Date
Drawn By	SM	3/16/22
Eng Approved	.....	.....
Mfg Approved	.....	.....
QA	.....	.....



Revisions

Rev.	Date	Author	Description
A	3/16/2022	Scott Mangin	

Bill of Materials

Id	Type	Manufacturer	Part Number	Quantity
1	Connector	N/A	PEC05SFBN	1
2	Connector	N/A	PPTC051LFBN-RC	1
3	Wire	N/A	UL1007 28 AWG Yellow	1
4	Wire	N/A	UL1007 28 AWG Green	1
5	Wire	N/A	UL1007 28 AWG Blue	1
6	Wire	N/A	UL1007 28 AWG Red	1
7	Wire	N/A	UL1007 28 AWG Orange	1

From	To	Conductor	Color	Gauge	Notes
STM32.1	TouchPanel.3	W1.Yellow		28 AWG	TS_SENSE (PA7)
STM32.2	TouchPanel.5	W2.Green		28 AWG	TS_LL (PB0)
STM32.3	TouchPanel.4	W3.Blue		28 AWG	TS_UL (PB1)
STM32.4	TouchPanel.2	W4.Red		28 AWG	TS_LR (PB2)
STM32.5	TouchPanel.1	W5.Orange		28 AWG	TS_UR (PB10)

From	To	Conductor	Color	Gauge
TouchPanel.1	STM32.5	W5.Orange		28 AWG
TouchPanel.2	STM32.4	W4.Red		28 AWG
TouchPanel.3	STM32.1	W1.Yellow		28 AWG
TouchPanel.4	STM32.3	W3.Blue		28 AWG
TouchPanel.5	STM32.2	W2.Green		28 AWG

STM32 [2]

TouchPanel [1]

Title	Touch Panel Harness	Part Number	H002
Description	N/A	Revision	01

	Name	Date
Drawn By	SM	3/16/22
Eng Approved	.....	.....
Mfg Approved	.....	.....
QA	.....	.....



Revisions			
Rev.	Date	Author	Description
A	3/16/2022	Scott Mangin	

Bill of Materials				
Id	Type	Manufacturer	Part Number	Quantity
1	Wire	N/A	UL1015 12 AWG Red	1
2	Wire	N/A	UL1015 12 AWG Green	1
3	Wire	N/A	UL1015 12 AWG White	1
4	Ferrule	N/A	14-12 AWG Gray	3

From	To	Conductor	Color	Gauge
A	A_ODrive	W1.Red	<span style="color:red">■</span>	12 AWG



From	To	Conductor	Color	Gauge
A_ODrive	A	W1.Red	<span style="color:red">■</span>	12 AWG

A\_ODrive [4]

From	To	Conductor	Color	Gauge
B	B_ODrive	W2.Green	<span style="color:green">■</span>	12 AWG



From	To	Conductor	Color	Gauge
B_ODrive	B	W2.Green	<span style="color:green">■</span>	12 AWG

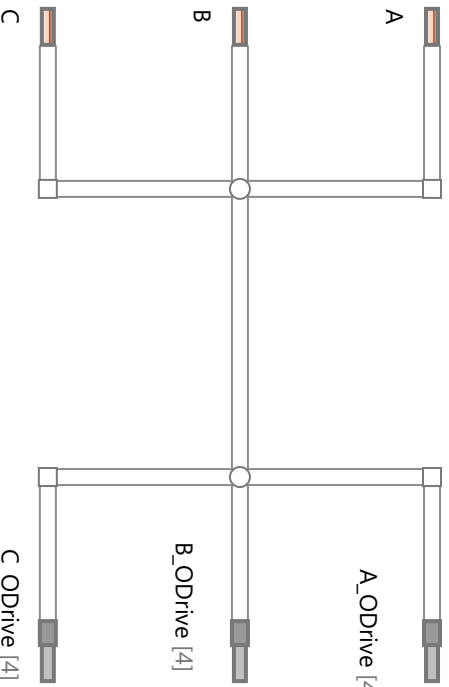
B\_ODrive [4]

From	To	Conductor	Color	Gauge
C	C_ODrive	W3.White	<span style="color:white">■</span>	12 AWG



From	To	Conductor	Color	Gauge
C_ODrive	C	W3.White	<span style="color:white">■</span>	12 AWG

C\_ODrive [4]



Title	Motor Harness - Axis 0	Part Number	H003
Description	N/A		
Revision	01		

Name	Date
Drawn By SM	3/16/22
Eng Approved Mfg Approved	
QA	



Revisions

Rev.	Date	Author	Description
A	3/16/2022	Scott Mangin	

Bill of Materials

Id	Type	Manufacturer	Part Number	Quantity
1	Ferrule	N/A	14-12 AWG Gray	3
2	Wire	N/A	UL1015 12 AWG Violet	
3	Wire	N/A	UL1015 12 AWG Yellow	
4	Wire	N/A	UL1015 12 AWG Black	

From	To	Conductor	Color	Gauge
C	C_ODrive	W3.Black		12 AWG



From	To	Conductor	Color	Gauge
C	C	W3.Black		12 AWG

From	To	Conductor	Color	Gauge
B	B_ODrive	W2.Yellow		12 AWG



From	To	Conductor	Color	Gauge
B	B	W2.Yellow		12 AWG

From	To	Conductor	Color	Gauge
A	A_ODrive	W1.Violet		12 AWG



From	To	Conductor	Color	Gauge
A	A	W1.Violet		12 AWG

Title	Motor Harness - Axis 1	Part Number	H004
Description	N/A	Revision	01

	Name	Date
Drawn By	SM	3/16/22
Eng Approved	.....	.....
Mfg Approved	.....	.....
QA	.....	.....

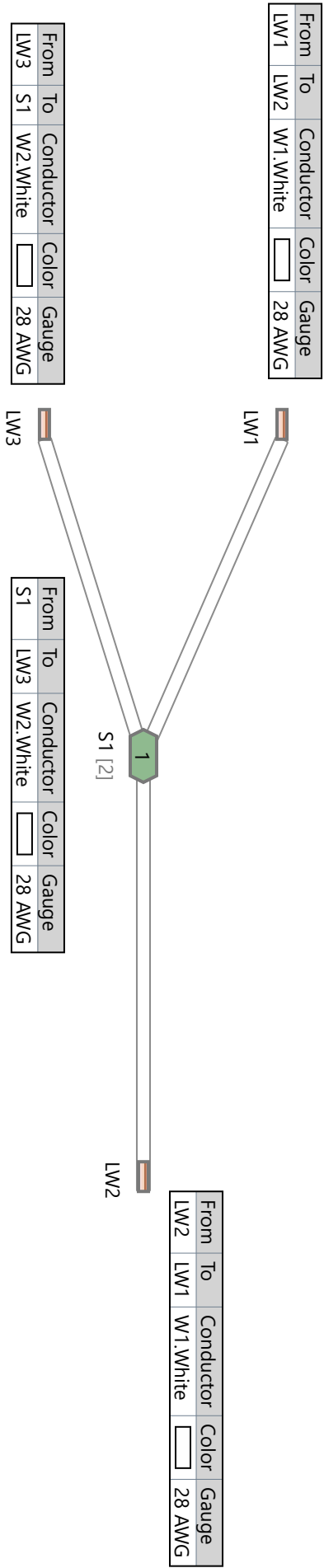


Revisions

Rev.	Date	Author	Description
A	3/14/2022	Scott Mangin	

Bill of Materials

Id	Type	Manufacturer	Part Number	Quantity
1	Wire	N/A	UL1007 28 AWG White	5
2	Splice	N/A	Generic Splice	1



Title		Part Number	
IR Signal Harness		H005	
Description		Revision	
N/A		01	

Drawn By	Name	Date
Eng Approved	SM	3/16/22
Mfg Approved	QA	



Appendix B

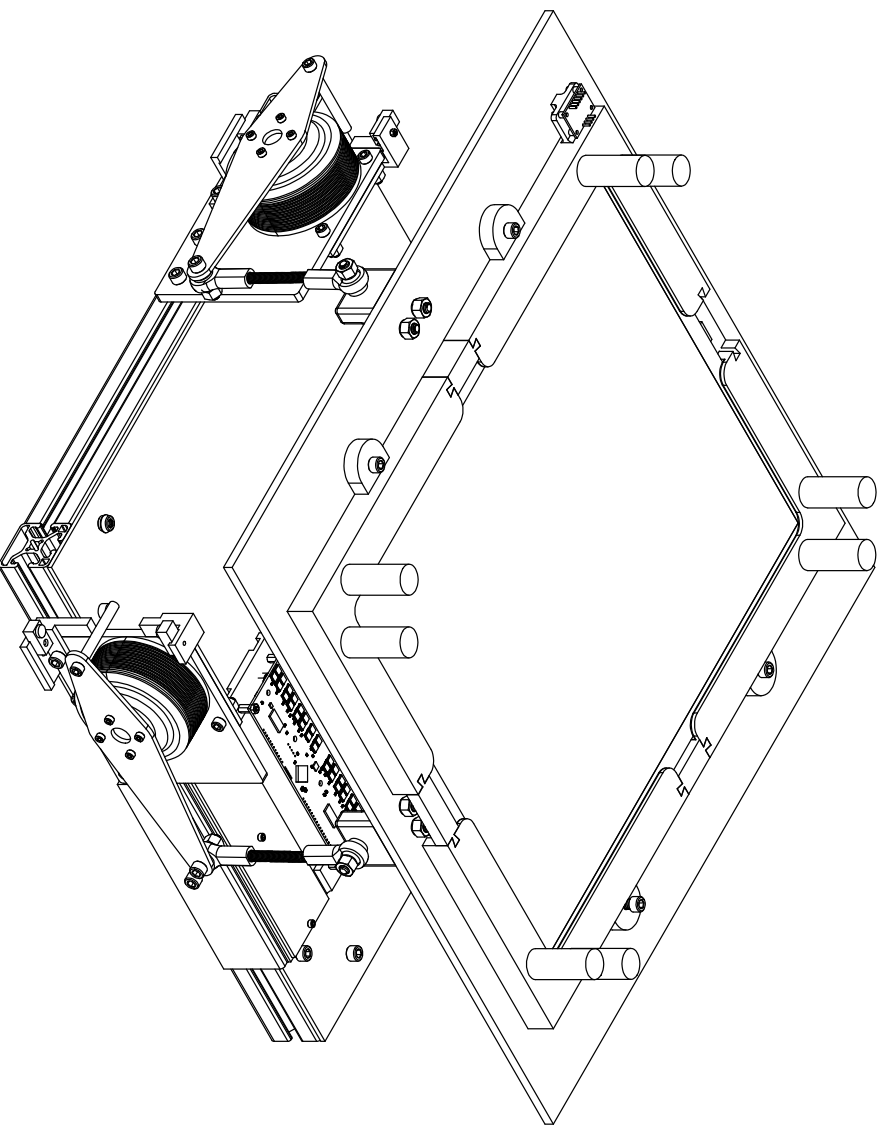
PLATFORM PART DRAWINGS

4

3

2

1



B

A

B

A

**PROPRIETARY AND CONFIDENTIAL**  
 THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF PERFORMANT PARTS, INC. AND IS NOT TO BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, WITHOUT THE WRITTEN PERMISSION OF PERFORMANT PARTS, INC.  
 \*INSERT COMPANY NAME HERE\* IS PROHIBITED.

UNLESS OTHERWISE SPECIFIED:		DRAWN	NAME	DATE
DIMENSIONS ARE IN MM		CHECKED	SM	3/14/22
TOLERANCES:		ENG APPR.		
FRACTIONAL ±		MTG APPR.		
ANGULAR: MACH ± BEND ±		COMMENTS:		
TWO PLACE DECIMAL ±		Q.A.		
THREE PLACE DECIMAL ±				
INTERPRET GEOMETRIC TOLERANCING PER:				
MATERIAL				
FINISH				
N/A				
DO NOT SCALE DRAWING				
N/A				
NEW ASSY	USED ON			
APPLICATION				

TITLE:  
**BALL AND PLATE  
 PLATFORM**

SIZE: DWG. NO. REV  
**B 001 01**

SCALE: 1:5 WEIGHT: SHEET 1 OF 10

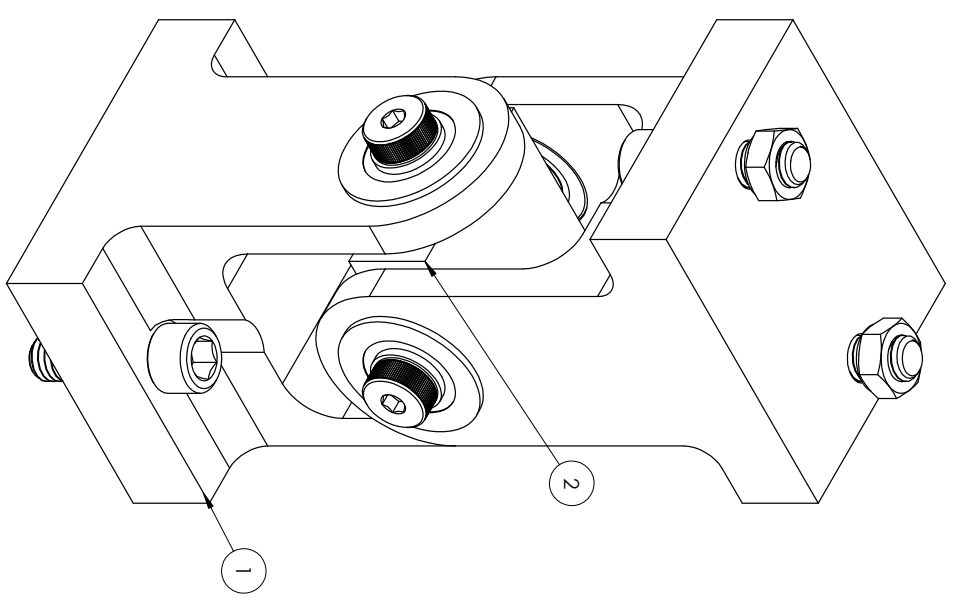
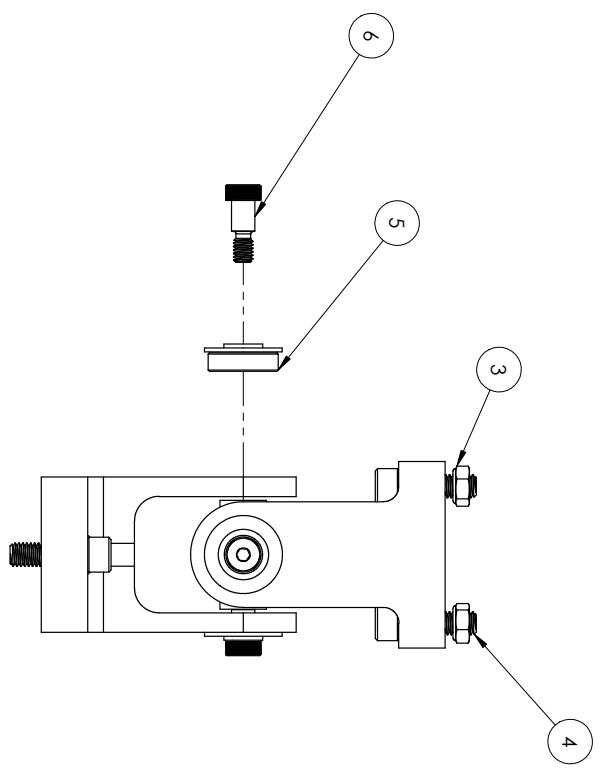
4

3

2

1

4 3 2 1



6	91259A619	SHOULDER SCREW, 3/8" DIAMETER, 5/16"-18 THREAD	4	ALLOY STEEL
5	6384K348	SEALED BALL BEARING, 3/8" SHAFT, 1-1/8" ID	4	STEEL
4	91251A626	SOCKET HEAD SCREW, 3/8"-1 6 1-1/4" LONG	4	ALLOY STEEL
3	90566A031	LOW PROFILE LOCK NUT, 3/8"-16	2	ALLOY STEEL
2	000002	U-JOINT CROSS BAR	1	DELRIN
1	000001	U-JOINT PILLOW BLOCK	2	PLA
ITEM NO.	PART NUMBER	DESCRIPTION	QTY.	MATERIAL

**PROPRIETARY AND CONFIDENTIAL**  
 THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF FORTY TWO INNOVATIONS, INC. AND IS TO BE USED ONLY FOR THE PARTS AND ASSEMBLIES IDENTIFIED HEREIN. ANY REPRODUCTION OR TRANSMISSION OF THIS INFORMATION WITHOUT THE WRITTEN PERMISSION OF FORTY TWO INNOVATIONS, INC. IS PROHIBITED.

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ± INTERPRET GEOMETRIC TOLERANCING PER: MATERIAL FINISH DO NOT SCALE DRAWING	DRAWN S.M.	DATE 3/14/22	TITLE: <b>U-JOINT ASSEMBLY</b>	SIZE: <b>B</b>	DWG. NO.: <b>002</b>	REV: <b>01</b>
NEW ASSY	USED ON	APPLICATION	SCALE: 1:2	WEIGHT:	SHEET 2 OF 10	

4 3 2 1

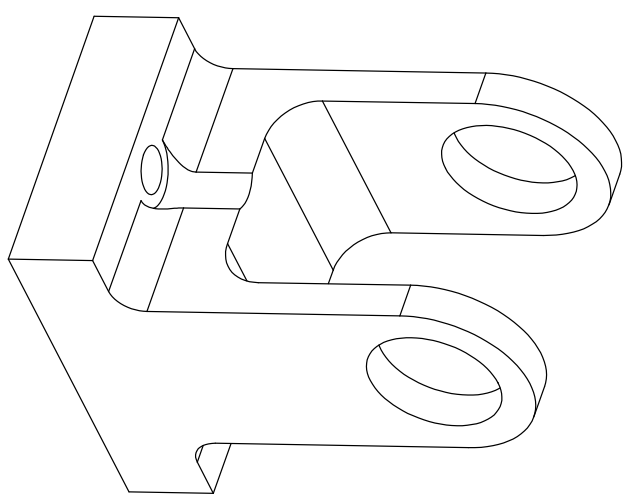
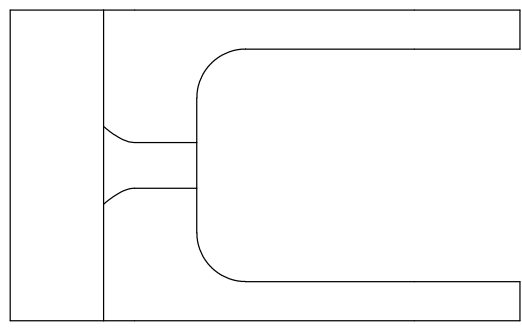
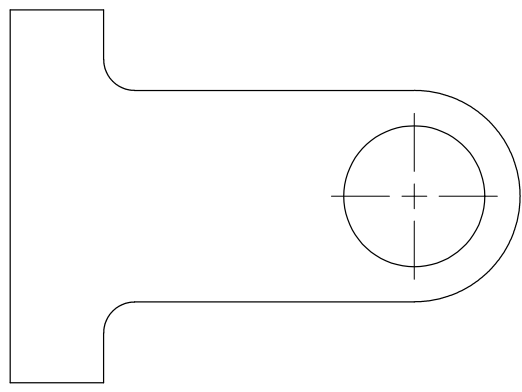
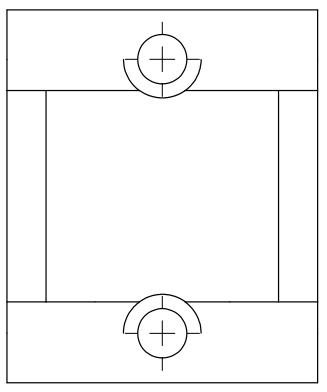


4

3

2

1



**U-JOINT PILLOW BLOCK**  
SCALE 1:1

MATERIAL: PLA

**NOTES:**

1. INTENDED FOR 3D PRINTING; NO DIMENSIONING REQUIRED.

**PROPRIETARY AND CONFIDENTIAL**  
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF FORTY TWO INNOVATIONS, INC. ANY REPRODUCTION OR TRANSMISSION OF THIS DRAWING WITHOUT THE WRITTEN PERMISSION OF FORTY TWO INNOVATIONS, INC. IS PROHIBITED.  
<INSERT COMPANY NAME HERE> IS

UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES		SM	3/14/22
TOLERANCES:			
FRACTIONAL: ±		CHECKED	
ANGULAR: MATCH ±	BEND ±	ENG APPR.	
TWO PLACE DECIMAL ±	THREE PLACE DECIMAL ±	MFG APPR.	
INTERPRET GEOMETRIC TOLERANCING PER:		Q.A.	
MATERIAL: PLA		COMMENTS:	
FINISH:			
NEW ASSY	USED ON		
APPLICATION	DO NOT SCALE DRAWING		

**U-JOINT PILLOW BLOCK**

SIZE: DWG. NO. **B 003** REV **01**  
SCALE: 1:2 WEIGHT: SHEET 3 OF 10

4

3

2

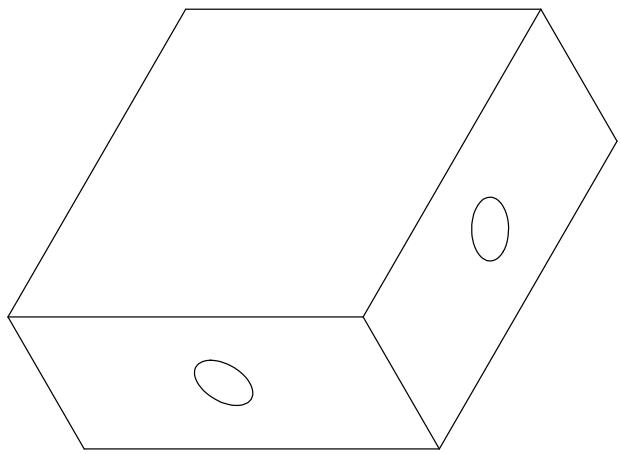
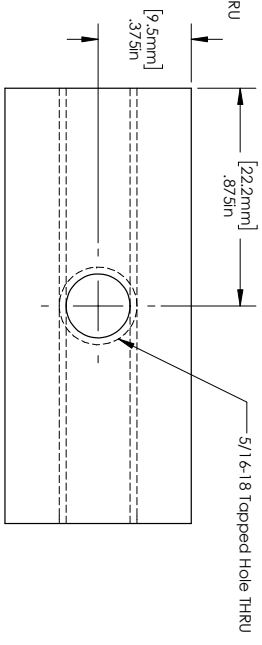
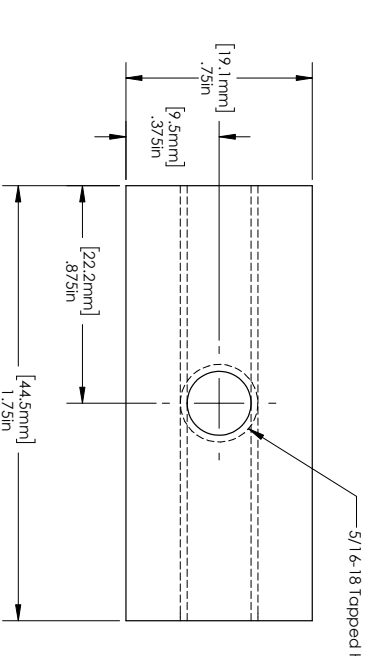
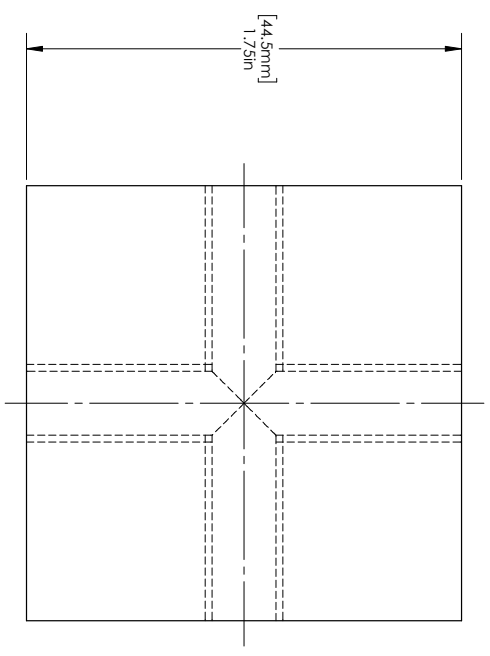
1

4

3

2

1



**U-JOINT CROSS BAR**  
 SCALE 2:1  
 MATERIAL: DELRIN

**PROPRIETARY AND CONFIDENTIAL**  
 THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF DELRIN. IT IS TO BE USED FOR THE PART IDENTIFIED HEREIN AND IS NOT TO BE REPRODUCED, COPIED, OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, WITHOUT THE WRITTEN PERMISSION OF DELRIN. COMPANY NAME HERE IS PROHIBITED.

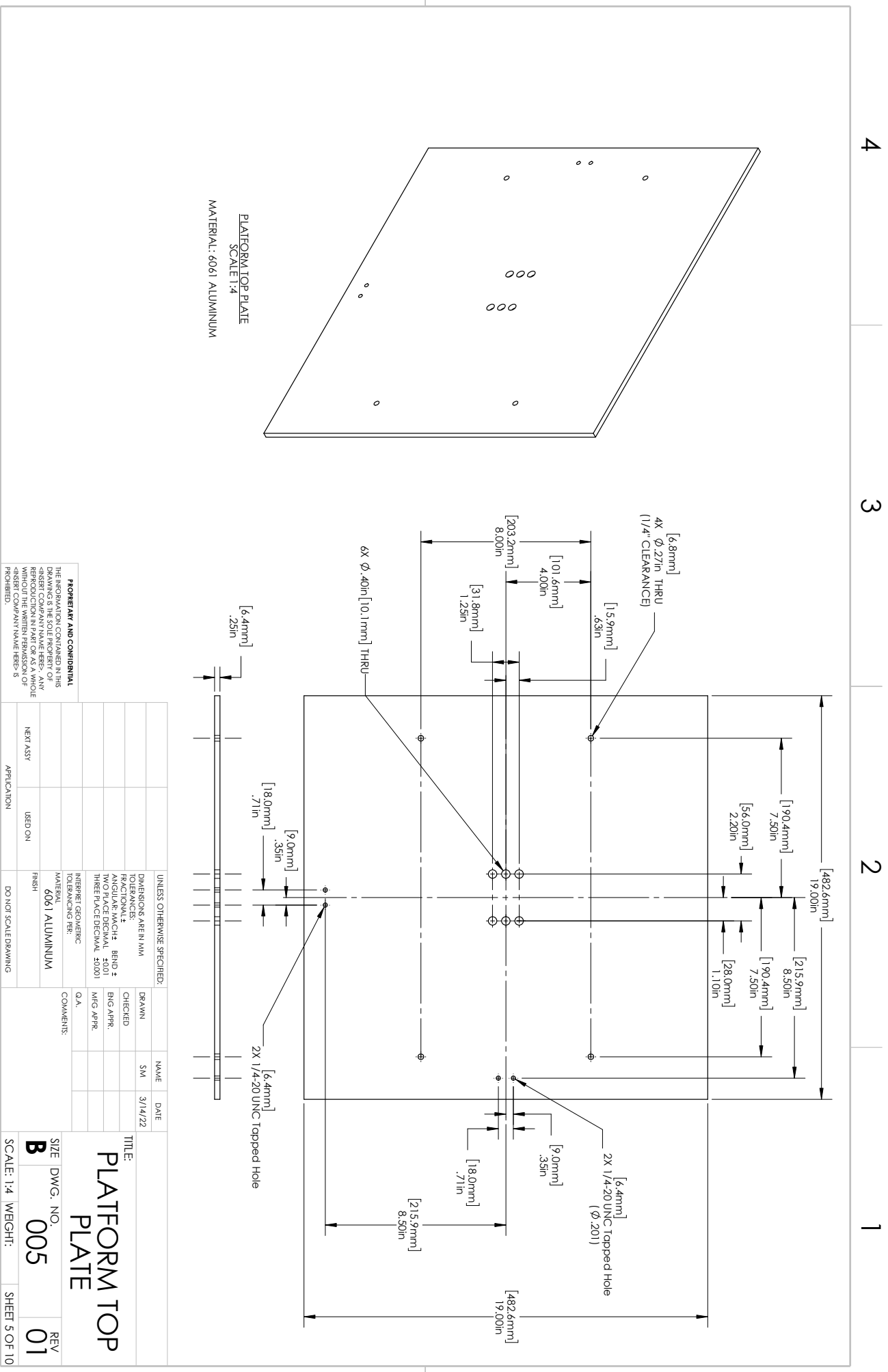
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: TWO PLACE DECIMAL ± .01 THREE PLACE DECIMAL ± .005	NAME SM	DATE 3/14/22
INTERPRET/GEOMETRIC TOLERANCING PER: MATERIAL DELRIN	CHECKED ENG APPR.	COMMENTS Q.A.
NEW ASSY	USED ON	FINISH N/A
APPLICATION	DO NOT SCALE DRAWING	
TITLE: <b>U-JOINT CROSS BAR</b>		SIZE DWG. NO. <b>B 004</b>
SCALE: 2:1		WEIGHT: SHEET 4 OF 10
REV <b>01</b>		

4

3

2

1



UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MM		SM	3/14/22
TOLERANCES:		CHECKED	
FRACTIONAL ±		ENG APPR.	
ANGULAR: MACH ±	BEND ±	MFG APPR.	
TWO PLACE DECIMAL ±	5.001		
THREE PLACE DECIMAL ±	50.001		
INTERPRET GEOMETRIC TOLERANCING PER:		COMMENTS:	
MATERIAL:	6061 ALUMINUM		
FINISH:			
NEW ASSY	USED ON		
APPLICATION	DO NOT SCALE DRAWING		

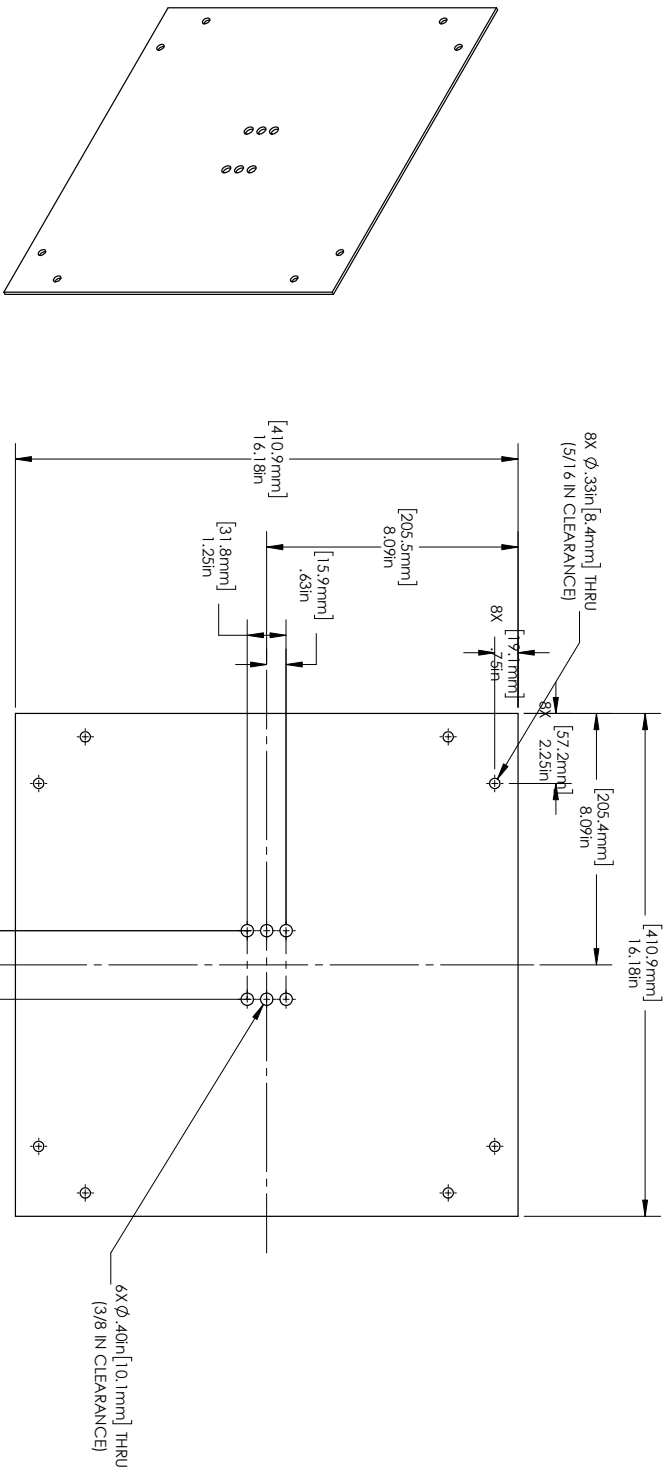
TITLE:	<b>PLATFORM TOP PLATE</b>	
SIZE:	DWG. NO.	REV
<b>B</b>	<b>005</b>	<b>01</b>
SCALE: 1:4	WEIGHT:	SHEET 5 OF 10

4

3

2

1



PLATFORM BOTTOM PLATE  
SCALE: 1:4  
MATERIAL: 6061 ALUMINIUM

**PROPRIETARY AND CONFIDENTIAL**  
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF PERIODIC COMPANY AND IS TO BE USED ONLY FOR THE PART AND ASSEMBLY IDENTIFIED HEREIN. ANY REPRODUCTION OR TRANSMISSION OF THIS DRAWING WITHOUT THE WRITTEN PERMISSION OF PERIODIC COMPANY IS PROHIBITED.

UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MM		SM	3/14/22
TOLERANCES:			
FRACTIONAL ±			
ANGULAR: MACH ± BEND ±			
TWO PLACE DECIMAL ±.001			
THREE PLACE DECIMAL ±.0001			
INTERPRET GEOMETRIC TOLERANCING PER: Q.A.			
MATERIAL: 6061 ALUMINIUM	COMMENTS:		
FINISH			
NEW ASSY	USED ON		
APPLICATION	DO NOT SCALE DRAWING		

TITLE:  
**PLATFORM  
BOTTOM PLATE**

SIZE: DWG. NO. **B 006** REV **01**

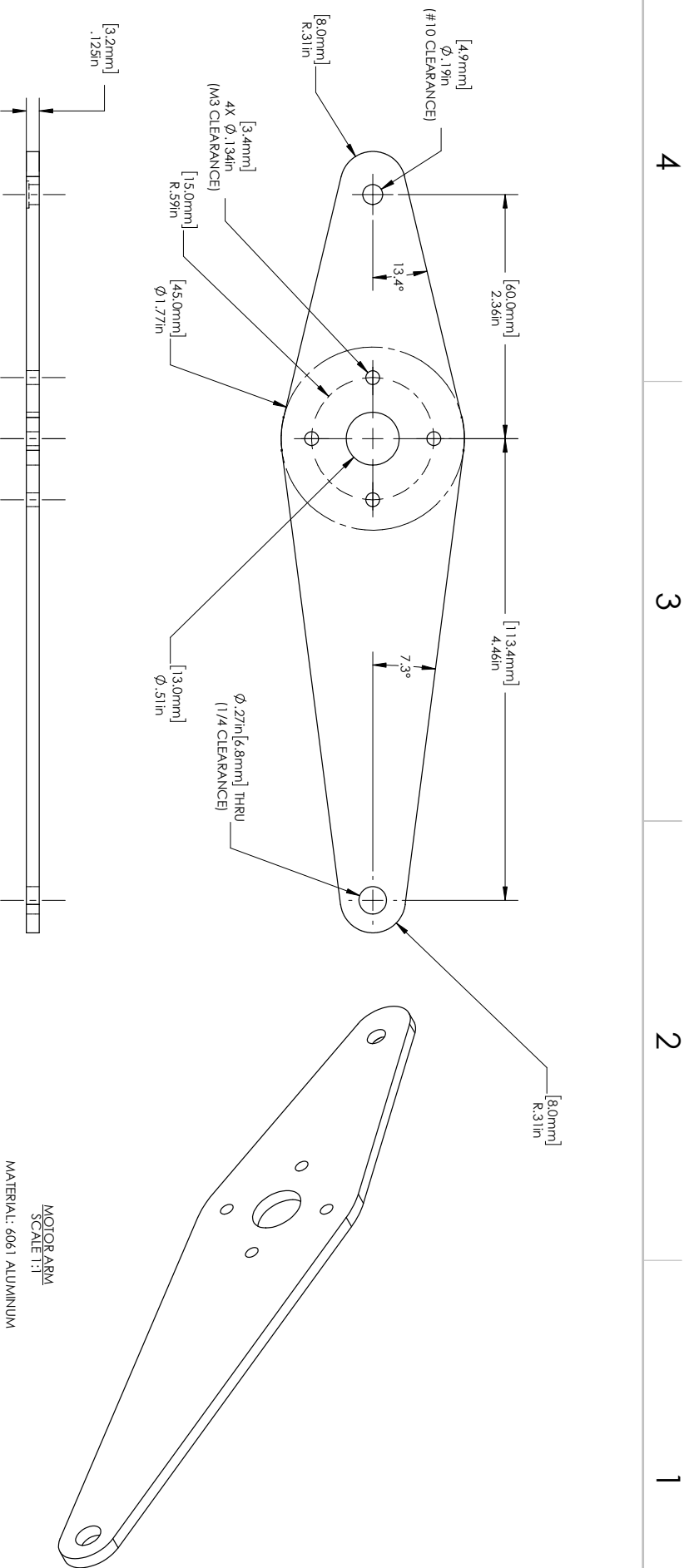
SCALE: 1:4 WEIGHT: SHEET 6 OF 10

4

3

2

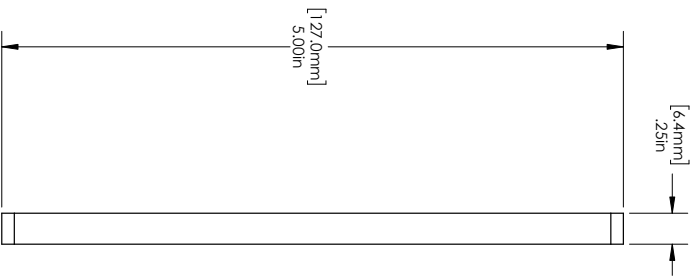
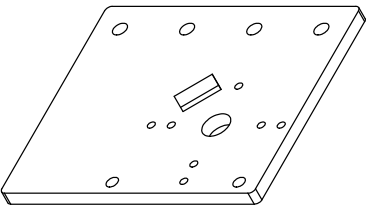
1



**PROPRIETARY AND CONFIDENTIAL**  
 THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF PERFORMANT PARTS, INC. AND IS UNCLASSIFIED. ANY REUSE OR REPRODUCTION OF THIS DRAWING WITHOUT THE WRITTEN PERMISSION OF PERFORMANT PARTS, INC. IS PROHIBITED.

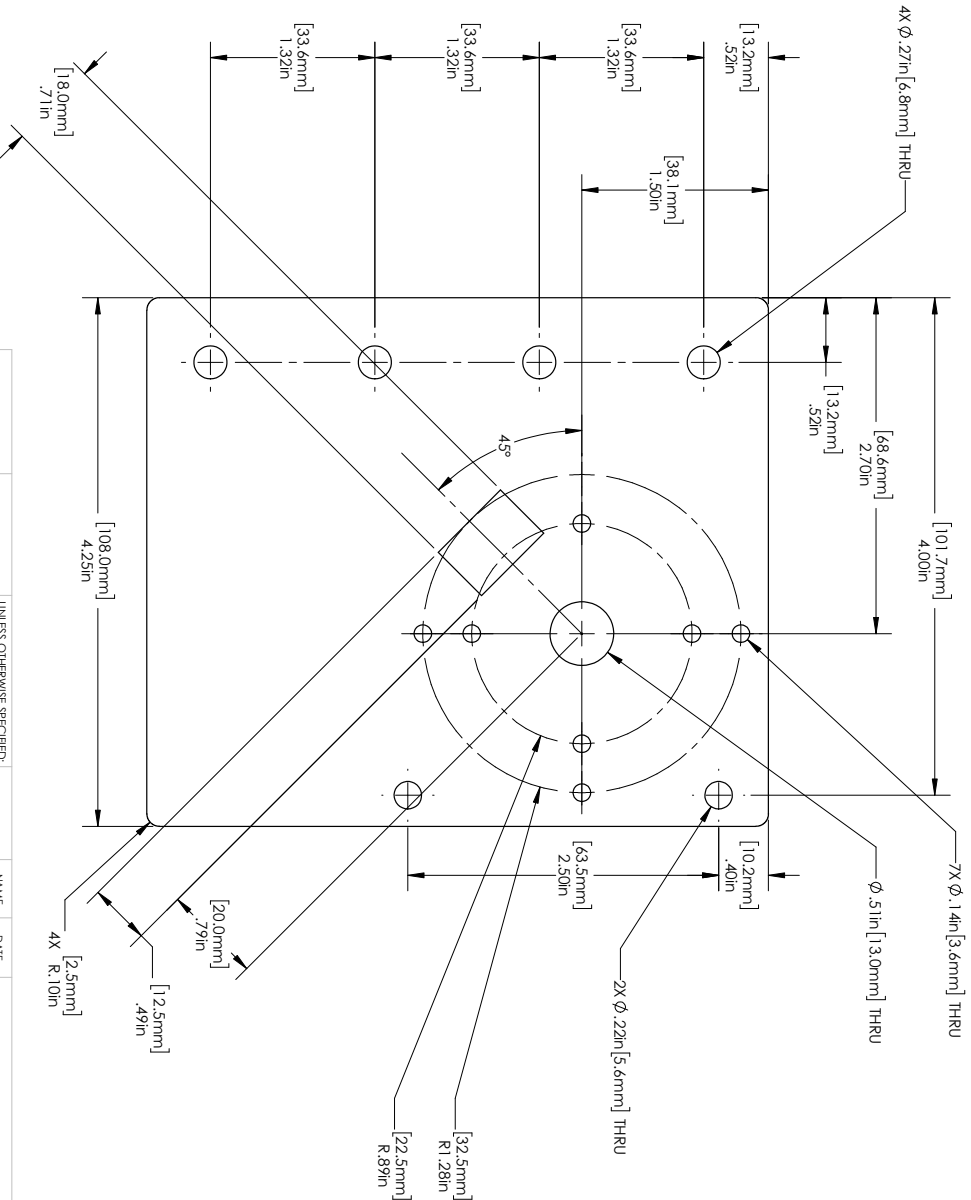
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MM		SM	3/14/22
TOLERANCES:			
FRACTIONAL ±			
ANGULAR: MACH ± BEND ±			
TWO PLACE DECIMAL ±0.01			
THREE PLACE DECIMAL ±0.001			
INTERPRET GEOMETRIC TOLERANCING PER: Q.A.			
MATERIAL: 6061 ALUMINUM	COMMENTS:		
FINISH:			
NEW ASSY	USED ON		
APPLICATION			

SIZE: <b>B</b>	DWG. NO: <b>007</b>	REV: <b>01</b>
SCALE: 1:1	WEIGHT:	SHEET 7 OF 10



MOTOR MOUNT PLATE  
SCALE 1:1

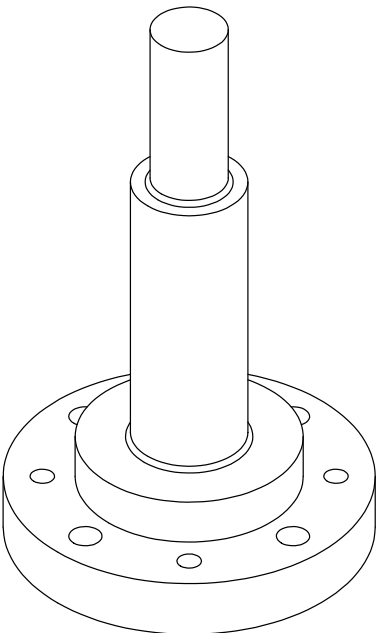
MATERIAL: 6061 ALUMINUM



**PROPRIETARY AND CONFIDENTIAL**  
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF PERFORMA COMPANY AND IS TO BE KEPT CONFIDENTIAL AND NOT TO BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, WITHOUT THE WRITTEN PERMISSION OF PERFORMA COMPANY. NAME HERE IS PROHIBITED.

UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN MM		SIM	3/14/22
TOLERANCES:		CHECKED	
FRACTIONAL ±		ENG APPR.	
ANGULAR: MACH ±	BEND ±	MFG APPR.	
TWO PLACE DECIMAL ±	THREE PLACE DECIMAL ±	Q.A.	
THREE PLACE DECIMAL ±	5.0001	COMMENTS:	
INTERPRET/GEOMETRIC TOLERANCING PER:			
MATERIAL:			
6061 ALUMINUM			
FINISH:			
DO NOT SCALE DRAWING			
NEW ASSY		USED ON	
APPLICATION			
SIZE	DWG. NO.	REV	
<b>B</b>	<b>008</b>	<b>01</b>	
SCALE: 1:1	WEIGHT:	SHEET 8 OF 10	

# MOTOR PLATE

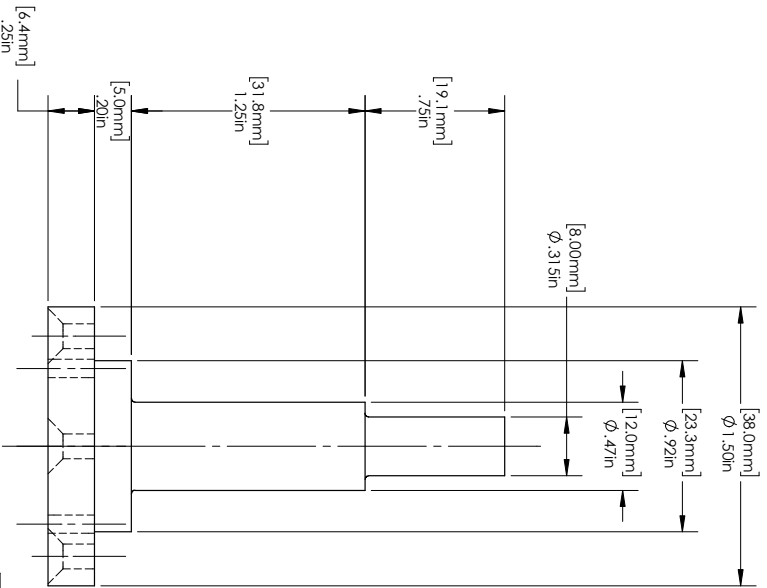


**MOTOR-ENCODER CONNECTOR**  
SCALE 3:2

MATERIAL: NYLON

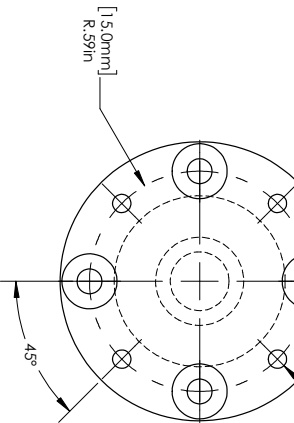
**NOTES:**

1. ALL EDGES BROKEN



4X  $\phi$ .131in [3.4mm] THRU  
 $\phi$ .30in [7.5mm] X 90°  
 EVENLY SPACED ON  $\phi$ 30mm BOLT CIRCLE

4X M3 THRU  
 TAP DRILL DIA.  $\phi$ .10in [2.5mm]  
 EVENLY SPACED ON  $\phi$ 30mm BOLT CIRCLE



**PROPRIETARY AND CONFIDENTIAL**  
 THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF PERIODIC COMPANY AND IS TO BE USED ONLY FOR THE PART AND QUANTITY SPECIFIED HEREIN. NO REPRODUCTION OR TRANSMISSION OF THIS DRAWING IS PERMITTED WITHOUT THE WRITTEN PERMISSION OF PERIODIC COMPANY. NAME HERE IS PROHIBITED.

UNLESS OTHERWISE SPECIFIED:	NAME	DATE
PRIMARY DIMENSIONS ARE IN MM	DRAWN	3/14/22
TOLERANCES: ONE DECIMAL, +.1	CHECKED	
TWO PLACE DECIMAL, ±.01	ENG APPR.	
	MFG APPR.	
	Q.A.	
INTERPRET/GEOMETRIC TOLERANCING PER:	COMMENTS:	
MATERIAL: NYLON		
FINISH:		
NEW ASSY	USED ON	
APPLICATION	DO NOT SCALE DRAWING	

TITLE:  
**MOTOR-ENCODER CONNECTOR**

SIZE: **B**  
 DWG. NO.: **009**

SCALE: 3:2 WEIGHT: SHEET 9 OF 10

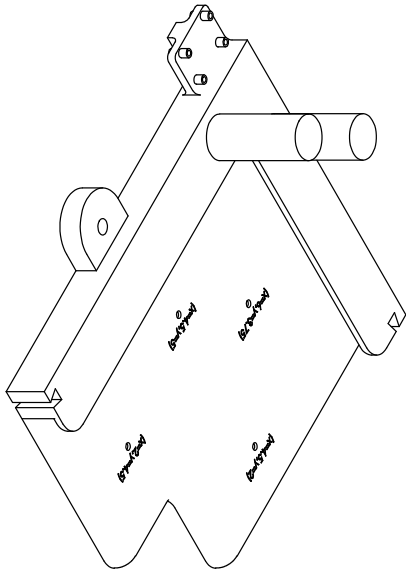
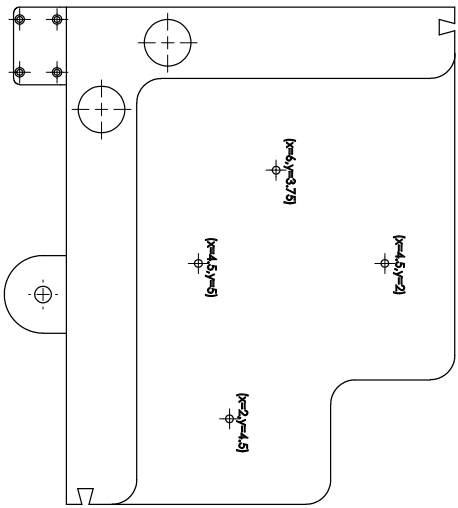
REV **1**

4

3

2

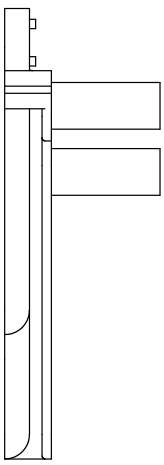
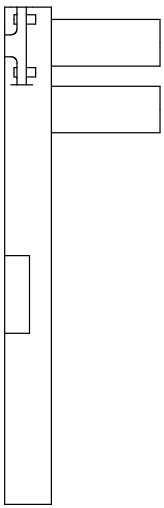
1



TOUCH PANEL SPACER  
 SCALE 3:2  
 MATERIAL: PLA

**NOTES:**

- 1. INTENDED FOR 3D PRINTING. NO DIMENSIONING REQUIRED.



A

A

B

B

**PROPRIETARY AND CONFIDENTIAL**  
 THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF INTERPRET GEOMETRIC TECHNOLOGIES, INC. ("INTERPRET GEOMETRIC") AND IS LOANED TO YOU BY INTERPRET GEOMETRIC. IT IS TO BE USED ONLY FOR THE PROJECT AND WITHOUT THE WRITTEN PERMISSION OF INTERPRET GEOMETRIC. ANY REPRODUCTION OR DISTRIBUTION OF THIS DRAWING WITHOUT THE WRITTEN PERMISSION OF INTERPRET GEOMETRIC IS PROHIBITED.

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES		NAME	DATE
DRAWN		SM	3/14/22
CHECKED			
ENG APPR.			
MFG APPR.			
Q.A.			
COMMENTS:			
MATERIAL: PLA			
FINISH:			
APPLICATION: NEW ASSY			
USED ON:			
DO NOT SCALE DRAWING			
TITLE: <b>TOUCH PANEL SPACER</b>		SIZE DWG. NO.	REV
SCALE: 1:2		<b>B</b> 010	01
WEIGHT:		SHEET 10 OF 10	

4

3

2

1



Appendix C

TORQUE TESTING DATA

Motor Axis 0 - Torque Constant Testing

Effective Arm Length:		0.10822		m	
Axis 0					
Torque Constant [N-m/A]	Commanded Torque [N-m]	Average I <sub>q</sub> [A]	Scale Measurement [g-f]	Experimental Torque [N-m]	Percent Error from Commanded
0.4	0.25	0.6	215	0.228	8.7%
	0.5	1.25	438	0.465	7.0%
	0.75	1.87	645	0.685	8.7%
	1	2.5	846	0.898	10.2%
0.35	0.25	0.71	247	0.262	-4.9%
	0.5	1.43	499	0.530	-5.9%
	0.75	2.14	743	0.789	-5.1%
	1	2.855	958	1.017	-1.7%
0.375	0.25	0.67	232	0.246	1.5%
	0.5	1.33	462	0.490	1.9%
	0.75	0.685	238	0.253	-1.0%
0.365	0.5	1.37	474	0.503	-0.6%
	0.75	2.055	708	0.751	-0.2%
	1	2.74	939	0.997	0.3%

Motor Axis 0 - Current Control Mode Ascending

Effective Arm Length:		0.10822		m		
Torque Constant:		0.365		N-m/A		
Current Limit		5.9		A		
Phase-to-Phase Resistance		6		Ohm		
Phase-to-Phase Inductance		2.72		mH		
Ambient Room Temperature		71		F		
Torque Command [N-m]	Average I <sub>q</sub> [A]	Scale Measurement [g-f]	Converted Force [N]	Experimental Torque [N-m]	Percent Error from Command	Motor Case Maximum Temperature [F]
0.25	0.685	243	2.383	0.258	-3.2%	75.6
	0.685	243	2.383	0.258	-3.2%	75.8
	0.685	241	2.363	0.256	-2.3%	75.6
0.5	1.37	488	4.786	0.518	-3.6%	75.8
	1.37	492	4.825	0.522	-4.4%	76
	1.37	487	4.776	0.517	-3.4%	76.2
0.75	2.055	723	7.090	0.767	-2.3%	76.4
	2.055	726	7.120	0.771	-2.7%	76.8
	2.055	721	7.071	0.765	-2.0%	77.4
1	2.74	937	9.189	0.994	0.6%	78.2
	2.74	940	9.219	0.998	0.2%	79
	2.74	936	9.179	0.993	0.7%	80.2
1.25	3.4	1127	11.052	1.196	4.3%	80.6
	3.3	1140	11.180	1.210	3.2%	81
	3.4	1141	11.190	1.211	3.1%	81.2

Motor Axis 0 - Current Control Mode Descending

Effective Arm Length:		0.10822		m		
Torque Constant:		0.365		N-m/A		
Current Limit		5.9		A		
Phase-to-Phase Resistance		6		Ohm		
Phase-to-Phase Inductance		2.72		mH		
Ambient Room Temperature		71		F		
Torque Command [N-m]	Average I <sub>q</sub> [A]	Scale Measurement [g-f]	Converted Force [N]	Experimental Torque [N-m]	Percent Error from Command	Motor Case Maximum Temperature [F]
1.25	3.4	1133	11.111	1.202	3.8%	78.4
	3.4	1131	11.092	1.200	4.0%	78.6
	3.4	1135	11.131	1.205	3.6%	78.6
1	2.74	946	9.277	1.004	-0.4%	78.8
	2.74	947	9.287	1.005	-0.5%	79.4
	2.74	950	9.317	1.008	-0.8%	79.8
0.75	2.055	731	7.169	0.776	-3.4%	80.6
	2.055	731	7.169	0.776	-3.4%	81
	2.055	732	7.179	0.777	-3.6%	82
0.5	1.37	491	4.815	0.521	-4.2%	82.6
	1.37	492	4.825	0.522	-4.4%	82.6
	1.37	488	4.786	0.518	-3.6%	83.2
0.25	0.685	240	2.354	0.255	-1.9%	82.2
	0.685	241	2.363	0.256	-2.3%	81.8
	0.685	237	2.324	0.252	-0.6%	81.6

Motor Axis 1 - Torque Constant Testing

Effective Arm Length:	0.10787	m	Axis 1			
Torque Constant [N-m/A]	0.35	N-m/A	Scale Measurement [g-f]	Experimental Torque [N-m]	Percent Error from Commanded	
0.365	0.25	0.685	226	0.239	4.4%	
	0.5	1.37	454	0.480	3.9%	
	0.75	2.055	678	0.717	4.4%	
0.355	0.25	0.705	235	0.249	0.6%	
	0.5	1.41	466	0.493	1.4%	
	0.75	2.115	696	0.736	1.8%	
0.35	1	2.815	918	0.971	2.9%	
	0.25	0.715	238	0.252	-0.7%	
	0.5	1.425	478	0.506	-1.1%	
	0.75	2.145	715	0.756	-0.9%	
	1	2.855	941	0.995	0.5%	

Motor Axis 1 - Current Control Mode Ascending

Effective Arm Length:	0.10787	m					
Torque Constant:	0.35	N-m/A					
Current Limit	5.9	A					
Phase-to-Phase Resistance	6	Ohm					
Phase-to-Phase Inductance	2.72	mH					
Ambient Room Temperature	71	F					
Torque Command [N-m]	Average I <sub>q</sub> [A]	Scale Measurement [g-f]	Converted Force [N]	Experimental Torque [N-m]	Percent Error from Command	Motor Case Maximum Temperature [F]	
0.25	0.715	244	2.393	0.258	-3.2%	82.4	
	0.715	238	2.334	0.252	-0.7%	81.6	
	0.715	244	2.393	0.258	-3.2%	81.8	
0.5	1.43	486	4.766	0.514	-2.8%	81.2	
	1.43	489	4.796	0.517	-3.5%	81.2	
	1.43	487	4.776	0.515	-3.0%	81.2	
0.75	2.145	728	7.139	0.770	-2.7%	81.2	
	2.145	733	7.189	0.775	-3.4%	81.4	
	2.145	730	7.159	0.772	-3.0%	82	
1	2.855	957	9.385	1.012	-1.2%	82.2	
	2.855	960	9.415	1.016	-1.6%	83.4	
	2.855	957	9.385	1.012	-1.2%	84.4	
1.25	3.5	1146	11.239	1.212	3.0%	85	
	3.5	1152	11.298	1.219	2.5%	86.2	
	3.5	1160	11.376	1.227	1.8%	86.4	

Motor Axis 1 - Current Control Mode Descending

Effective Arm Length:	0.10787	m					
Torque Constant:	0.35	N-m/A					
Current Limit	5.9	A					
Phase-to-Phase Resistance	6	Ohm					
Phase-to-Phase Inductance	2.72	mH					
Ambient Room Temperature	71	F					
Torque Command [N-m]	Average I <sub>q</sub> [A]	Scale Measurement [g-f]	Converted Force [N]	Experimental Torque [N-m]	Percent Error from Command	Motor Case Maximum Temperature [F]	
1.25	3.5	1155	11.327	1.222	2.3%	86.4	
	3.5	1154	11.317	1.221	2.3%	86.4	
	3.5	1150	11.278	1.217	2.7%	86.6	
1	2.855	951	9.326	1.006	-0.6%	87.2	
	2.855	949	9.307	1.004	-0.4%	87.4	
	2.855	949	9.307	1.004	-0.4%	88.8	
0.75	2.145	715	7.012	0.756	-0.9%	89.4	
	2.145	719	7.051	0.761	-1.4%	89.8	
	2.145	721	7.071	0.763	-1.7%	87	
0.5	1.43	481	4.717	0.509	-1.8%	86.8	
	1.43	479	4.698	0.507	-1.3%	86.8	
	1.43	481	4.717	0.509	-1.8%	86.6	
0.25	0.715	238	2.334	0.252	-0.7%	86.6	
	0.715	237	2.324	0.251	-0.3%	86.8	
	0.715	240	2.354	0.254	-1.6%	86.4	

Motor Axis 0 - Gimbal Mode

Effective Arm Length:	0.10753 m					
Torque Constant:	0.11 N-m/V					
Current Limit	24 V					
Phase-to-Phase Resistance	6 Ohm					
Phase-to-Phase Inductance	2.72 mH					
Ambient Room Temperature	72 F					
Torque Command [N-m]	Average I <sub>q</sub> [A]	Scale Measurement [g-f]	Converted Force [N]	Experimental Torque [N-m]	Percent Error from Command	Motor Case Maximum Temperature [F]
0.25	0.7	241	2.36	0.2541	-1.7%	71.8
	0.7	242	2.37	0.2552	-2.1%	71.8
	0.7	244	2.39	0.2573	-1.7%	72
0.5	1.4	502	4.92	0.5294	-5.9%	72.4
	1.4	498	4.88	0.5252	-5.0%	72.8
	1.4	488	4.79	0.5146	-2.9%	73.2
0.75	2.1	721	7.07	0.7603	-1.4%	73.8
	2.05	724	7.10	0.7635	-1.8%	74.8
	2.1	720	7.06	0.7593	-1.2%	75.4
1	2.8	965	9.46	1.0176	-1.8%	77.2
	2.8	974	9.55	1.0271	-2.7%	77.8
	2.8	972	9.53	1.0250	-2.5%	78
1.25	3.6	1222	11.98	1.2887	-3.1%	78.8
	3.6	1206	11.83	1.2718	-1.7%	79
	3.6	1202	11.79	1.2676	-1.4%	79.6
1.5	4.9	1403	13.76	1.4795	1.4%	80.4
	4.9	1446	14.18	1.5249	-1.7%	78.8
	5	1383	13.56	1.4584	2.8%	79.4

Motor Axis 1 - Gimbal Mode

Effective Arm Length:	0.10997 m					
Torque Constant:	0.115 N-m/V					
Current Limit	24 V					
Phase-to-Phase Resistance	6 Ohm					
Phase-to-Phase Inductance	2.72 mH					
Ambient Room Temperature	72 F					
Torque Command [N-m]	Average I <sub>q</sub> [A]	Scale Measurement [g-f]	Converted Force [N]	Experimental Torque [N-m]	Percent Error from Command	Motor Case Maximum Temperature [F]
0.25	0.75	231	2.27	0.2491	0.3%	77.5
	0.75	237	2.32	0.2556	-2.2%	83.8
	0.75	242	2.37	0.2610	-4.4%	79.8
0.5	1.5	497	4.87	0.5360	-7.2%	80
	1.5	492	4.83	0.5306	-6.1%	79.6
	1.5	489	4.80	0.5274	-5.5%	79.6
0.75	2.15	743	7.29	0.8013	-6.8%	80.2
	2.15	756	7.41	0.8153	-8.7%	76.6
	2.15	746	7.32	0.8045	-7.3%	77.4
1	2.8	985	9.66	1.0623	-6.2%	77.8
	2.8	962	9.43	1.0375	-3.7%	78.2
	2.8	961	9.42	1.0364	-3.6%	79
1.25	3.6	1215	11.92	1.3103	-4.8%	78.6
	3.7	1216	11.93	1.3114	-4.9%	78.8
	3.6	1192	11.69	1.2855	-2.8%	79.2
1.5	10	1374	13.47	1.4818	1.2%	80.2
	11	1389	13.62	1.4980	0.1%	81
	11	1386	13.5925	1.4948	0.3%	81.4