4/12/2022

EML4551-2

Team 501: Landing System for Uncertain Terrain

Saralyn R. Jenkins; Elzbieta Krekora; Andrew V. Sak; Julio A. Velasquez

FAMU-FSU College of Engineering 2525 Pottsdamer St. Tallahassee, FL. 32310

Abstract

Team 501 has designed a spacecraft landing system capable of landing on the hypothesized surfaces of the Psyche asteroid, an M-Type asteroid likely largely made of metal. Scientists believe that Psyche may be remnant core material from a planetesimal. Launching in 2022 and arriving in 2026, an orbiter will get a closer look at Psyche and gather information. Scientists are hopeful that this asteroid could provide useful information about the formation of the solar system and the cores of rocky planets like Earth. Potential motivation from the findings could result in future mission teams proposing to land a spacecraft on Psyche. Currently, the surface of the asteroid is unknown but thought to be uneven, with a mix of rock and metal, unlike other solar system bodies visited before. The proposed landing attachment will be able to land a spacecraft on the asteroid's different hypothesized surfaces. The selected design uses three legs to support the spacecraft. Each leg features a shock absorber to take the impact force of landing. The shock absorber is a piston-like assembly with an aluminum honeycomb-filled cylinder that deforms on impact. To stabilize the spacecraft, each leg adjusts its height independently for a leveled position. At the bottom of the leg is a pin screen foot, modeled after the popular children's toy. The pin screen toy is known for being able to form to the shape of any three-dimensional relief it is placed on. Displayed on the pin screen is the relief's shape with pins. Consisting of closely packed metal pins that slide back and forth in slots, the pin screen feet form to the uneven terrain of Psyche. The pins prevent the feet from slipping on the terrain as well. Team 501's design achieves the goal to create a landing system that can successfully land a spacecraft on Psyche's range of hypothesized surfaces.

Keywords: Psyche, spacecraft, lander, damper, pinscreen

Disclaimer

Acknowledgement

Our team would like to thank our sponsor, Dr. Cassie Bowman at Arizona State University for providing us with this opportunity to develop and strengthen our engineering abilities, as well as all the skills necessary to fulfil a design project. Thank you for all the resources and support you've offered throughout the project.

To our academic advisor, Dr. Camilo Ordóñez, thank you for volunteering your time and effort for our project and extending your knowledge of mechatronics to our team. We appreciate the support you've given us helping to bring our project to fruition.

Thank you, Bobby Jenkins, and Philip Cooksey with Kelly Sheet Metal Inc. for providing materials to make our model. Thank you for assembling the base of our model for us, as well as sharing your knowledge of machining with our team.

Thank you Dr. McConomy for conducting the Senior Design capstone course and dedicating your time towards preparing our team for our respective careers. Thank you for all the knowledge, materials, support, and words of wisdom you've passed on to us.

Table of Contents

List of Tables

List of Figures

Notation

A17	Steering Column Angle
A27	Pan Angle
A40	Back Angle
A42	Hip Angle
AAA	American Automobile Association
AARP	American Association of Retired Persons
AHP	Accelerator Heel Point
ANOVA	Analysis of Variance
AOTA	American Occupational Therapy Association
ASA	American Society on Aging
BA	Back Angle
BOF	Ball of Foot
BOFRP	Ball of Foot Reference Point
CAD	Computer Aided Design
CDC	Centers for Disease Control and Prevention
CU-ICAR	Clemson University - International Center for Automotive Research
DDI	Driver Death per Involvement Ratio
DIT	Driver Involvement per Vehicle Mile Traveled

	Difference between the calculated and measured
Difference	BOFRP to H-point
DRR	Death Rate Ratio
DRS	Driving Rehabilitation Specialist
EMM	Estimated Marginal Means
FARS	Fatality Analysis Reporting System
FMVSS	Federal Motor Vehicle Safety Standard
GES	General Estimates System
GHS	Greenville Health System
H13	Steering Wheel Thigh Clearance
H17	Wheel Center to Heel Pont
H30	H-point to accelerator heel point
HPD	H-point Design Tool
HPM	H-point Machine
HPM-II	H-point Machine II
HT	H-point Travel
HX	H-point to Accelerator Heel Point
HZ	H-point to Accelerator Heel Point
IIHS	Insurance Institute for Highway Safety
L6	BFRP to Steering Wheel Center

Chapter One: EML 4551C

1.1 Project Scope

1.1.1 Project Description

The NASA Psyche Mission is set to orbit the asteroid 16 Psyche in 2026. After studying the asteroid many scientists and engineers might be interested in proposing a mission to land on it at a future date. The objective of this project is to design a landing system capable of safely landing on the assumed range of hypothesized surfaces and terrains of 16 Psyche.

1.1.2 Key Goals

The goal of this project is to create a landing system that will provide safe landing on 16 Psyche for a future spacecraft. To provide a safe landing, the landing system must be able to account for various types of terrain such as high relief terrain, terrain with rocky debris, and metallic terrain. The landing system must also be able to prevent the spacecraft from tipping and flipping. If the spacecraft tips or flips, then the spacecraft could be potentially damaged and rendered useless. The landing system must be able to absorb the impact upon landing the spacecraft to prevent any component failure within the spacecraft.

1.1.3 Market

This project will primarily appeal to NASA for any future landing spacecrafts that are scheduled to land in places where there are no guaranteed flat surfaces. This project will have a secondary appeal to private space companies and the robotics industry. The design could be used by other private or foreign space agencies for landing missions where the landing terrain is not optimal. It may appeal to the robotics industry because a large issue is being able to make robots that can overcome dynamic terrains.

1.1.4 Assumptions

Throughout the project there are assumptions that are made. We assume that the landing system will be operated in minimal gravity and that it will be able to operate in space temperatures and conditions. We assume that the spacecraft will be able to perform a soft landing on uncertain hypothesized terrains that may include mostly flat metallic surface, flat metallic with metal and/or rocky debris, rough/high relief metallic and/or rocky terrain, and high relief metallic crater walls. We assume that the landing system designed will be able to be attached to a future spacecraft without issue. The power to operate the system is assumed to be supplied from the spacecraft. This system is assumed to be controlled autonomously without manual maneuvering of the system.

1.1.5 Stakeholders

Those who are stakeholders in this project include both administrations and people alike. Arizona State University (ASU) is a stakeholder through the sponsorship of the project. The National Aeronautics and Space Administration (NASA) is a stakeholder since they are partnered with ASU and are making the mission to 16 Psyche possible. Individuals who are also stakeholders include Dr. Cassie Bowman, co-investigator on the Psyche Asteroid Mission with ASU, as she provides us information related to our project. Dr. Shayne McConomy is a stakeholder as he has invested time into setting up the sponsorship; expending time/effort to assist us as students with our project to make sure that we as a team, have the skills necessary for a successful engineering future. As our advisor, Dr. Camilo Ordonez is investing time and effort into meeting with our team, providing guidance and advice. Stakeholders are also those interested in scientific discovery, as well as taxpayers, who collectively fund NASA's budget.

1.2 Customer Needs

To determine the specific needs of this project, questions were directed towards Dr. Cassie Bowman. The question that best represented the essence of this project was "What are the possible landing sites at Psyche?". The sponsor stated that the main problem is the lack of accurate information on the topography of Psyche, this impacted our scope. Further questions focused on controls needed and spacecraft specifications, as these are important design parameters for the project. The questions made to the sponsor can be found in Table A, along with the interpreted needs based on the responses. This customer response data was obtained through video calls and over Slack, the preferred method of communication established by the sponsor. The interpreted needs are based on the underlying issue inherent in the customer statements. Several responses, or customer statements, resulted in the same interpreted need.

The responses to the questions asked impacted the scope of our project. Most of the answers gathered were leaning towards the ability to handle the different terrains. The second point of concern we received from the customer was regarding the payload of the spacecraft. The customer argued that the system must be able to land the spacecraft while carrying a rover as a payload. We organized our interpreted needs in 3 groups, Control Needs, Landing Conditions Needs, and Payload Needs. We decided to prioritize the Landing Conditions Needs since a greater portion of the customer complaints revolve around that.

Table 1.

Questions to Customer, Customer Statement, and Interpreted Needs

Question Asked	**Customer Statement**	**Interpreted Need**
What is the possible size/weight of the spacecraft the landing gear will support?	"Look at previous missions to small planets for reference sizes. Look at other landers and the rovers they carried, but we don't want to send something big and expensive."	The landing system supports the weight/size of the spacecraft based off previous missions.
Does the spacecraft have storage underneath?	"Yes, look at the rover previously made by a FAMU-FSU Team for a reference size."	The system can support the CHONKE Rover without damaging it.
What is the estimated impact velocity of the spacecraft?	"It will be similar to that of previous space missions to land on small planets."	The device can withstand or dissipate the potential energy from the fall and impact velocity.
What are the possible landing sites at Psyche?	"Let everyone know that the lander will be able to handle the hypothesized terrains. Better knowledge of where to land will	The system is capable of successfully landing on the hypothesized terrains of Psyche ie.

	come after completion of the upcoming orbiter mission. From the orbiter we can determine where the best place is to land and set the lander to go there.”	rocky, mostly metal, ect.
Is the team responsible for the control of the impact velocity of the spacecraft?	“Assume the lander has been brought to a reasonable impact velocity by other equipment. This impact velocity would be based off previous space missions and what you conclude.”	Ability to withstand impact and land from assumed impact velocity.
Is the spacecraft returning to Earth? If so, is the team responsible for the landing system for the return?	“No, assume the spacecraft is staying on Psyche.”	The system does not have to be reusable.
Does the landing system require any remote controls for manual maneuvering?	“The system needs to be autonomous. Psyche is too far to pilot any spacecraft.”	The system is autonomous.

After discussing needs with our sponsor, necessary data for our project was collected that will be reflected in our design concepts. The system created will need to be robust enough to withstand a given range of landing speeds onto an unknown terrain. It will also need to house and protect the CHONKE rover as a payload, which we do not have exact dimensions of and will estimate values forThe controls for the system will be autonomous and the landing system will be left on the asteroid after completing its mission.

1.3 Functional Decomposition

Our functional decomposition shows all the necessary functions for our landing system to successfully work on Psyche. Information gathered from our customer needs influenced the functions in our functional decomposition. The largest piece of information gathered was that there are many unknowns regarding Psyche. The unknowns are regarding the type of surface terrain on Psyche and the size of the payload that will be carried. We also were informed that the approach from space to surface should be modeled as well as the landing system.

Due to the unknown nature of the terrain and the need to simulate the approach a major function of Controls is needed. The Controls function will control the speed upon landing and control the landing system to adjust to the uneven terrain. For our landing system to know when to deploy a major function of Sensors is needed. There will be both internal and external sensors to measure values about the lander and values about the environment. As the lander secures position on the surface a major function of Support was created. The support function includes the landing system being able to stabilize on the surface as well as protecting the payload and the other components on the lander.

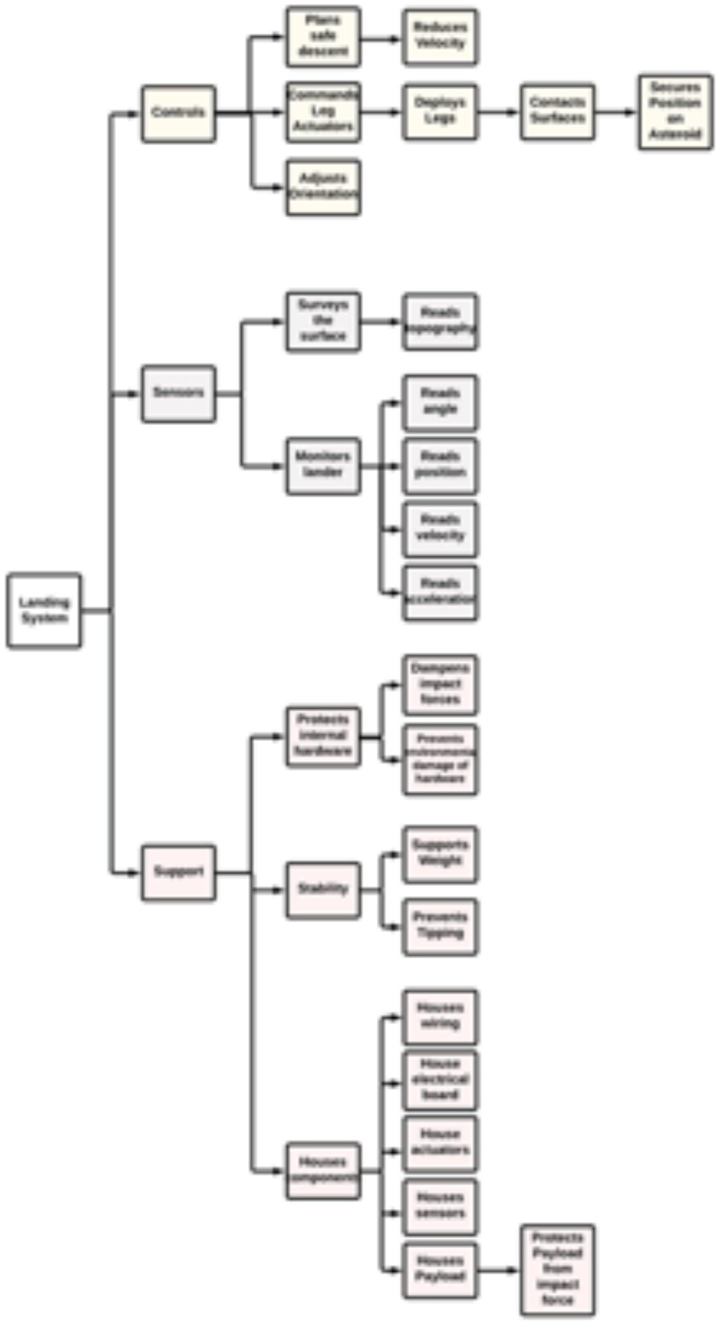


Figure 1. Functional Decomposition Flow Chart

This functional decomposition (Figure 1) shows the required components of the landing system that our team is designing to land on the hypothesized terrain of Psyche. The overall system is broken down into major and minor components. The major components consist of *Support, Sensors,* and *Controls.* These major components are broken into smaller components that all connect to produce what will be the landing system.

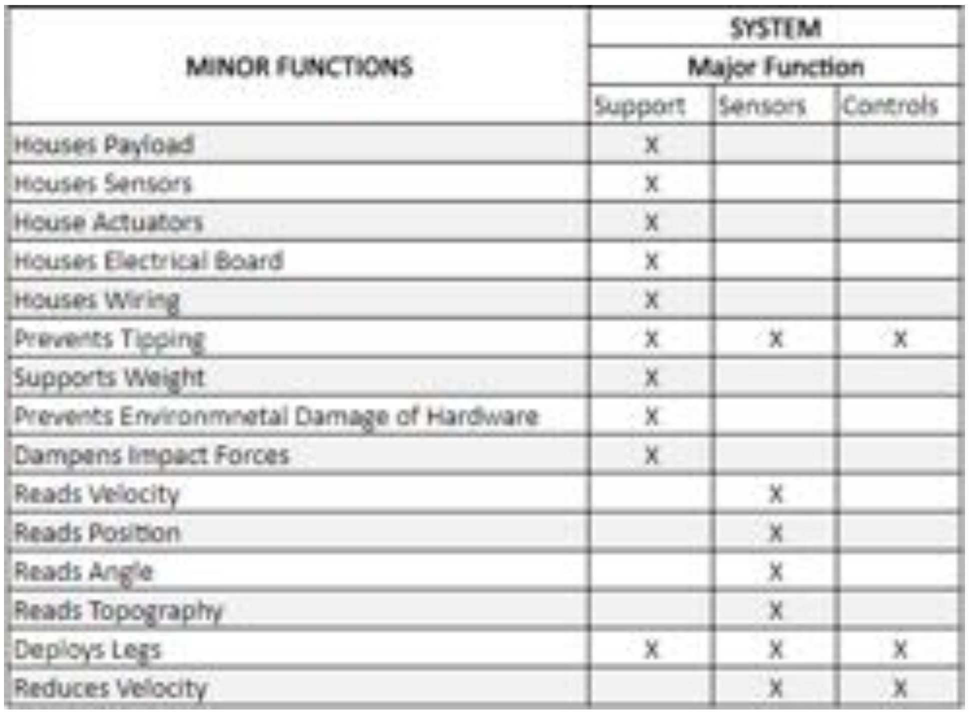
The *Support* component represents the physical responsibility of the system to stabilize, organize and protect what is being landed on Psyche. With this being said, the minor components of this include *Houses Components, Stability,* and *Protects Internal Hardware. Housing Components* main function is to store the necessary operating components of the system; therefore, this is broken into sub-systems that include *Houses Payload, Houses Sensors, House Actuators, House Electrical Board,* and *Houses Wiring*. *Stability* focuses on keeping the system in the desired position so that the other components remain unharmed. This is broken into the subsystems of *Prevents Tipping* and *Supports Weight*, narrowing down what the support system needs to accomplish to be stable. *Protects Internal Components* consists of the subsystems *Prevents Environmental Damage of Hardware* and *Dampens Impact Forces.* These subsystems focus on the damage control of essential pieces of the landing system.

The *Sensors* component represents the responsibility of the system to adjust to the surroundings based on what it encounters. The minor components of this include *Monitors Lander* and *Surveys the Surface*. *Monitors Lander* include the subcomponents *Reads Angle, Reads Position, Reads Velocity,* and *Reads Acceleration.* These represent how the lander will be monitored and what will need to be sensed to adjust the physical landing system to ensure stability. *Surveys the Surface* is broken into the subcomponent *Reads Topography*. This falls into what was stated previously, the surface and its topography need to be sensed to modify aspects of the physical landing system.

The *Controls* component represents the responsibility of the system to send signals and commands to various components, adjusting the way they operate. The minor components include *Adjust Orientation, Commands Leg Actuators,* and *Plans Safe Descent. Adjust Orientation* is meant to physically modify the position of the landing system to land in a stable manner. *Commands Leg Actuators* are broken into the subcomponents *Deploys Legs, Contacts Surfaces,* and *Secures Position on Asteroid.* This represents the process in which the legs of the lander are commanded to make contact on Psyche with the goal of achieving a stable landing. *Plans Safe Decent* is broken into the subcomponent *Reduces Velocity*. This is meant to slow the landing system to a safe velocity while sensing position and other factors.

Table 2.

Functional Decomposition Matrix



MINOR FUNCTIONS	SYSTEM		
	Major Function		
	Support	Sensors	Controls
Houses Payload	X		
Houses Sensors	X		
House Actuators	X		
Houses Electrical Board	X		
Houses Wiring	X		
Prevents Tipping	X	X	X
Supports Weight	X		
Prevents Environmnetal Damage of Hardware	X		
Dampens Impact Forces	X		
Reads Velocity		X	
Reads Position		X	
Reads Angle		X	
Reads Topography		X	
Deploys Legs	X	X	X
Reduces Velocity		X	X

1.3.1 Smart Integration

Secure position (controls) → prevent tipping/support weights (stability in support)

Monitors lander (sensors) → plans safe descent/reduces velocity (controls)

Protects payload from impact forces (houses components) → secures position on asteroid (controls)

On our functional decomposition chart, as shown in Table B, the major functions of *Sensors* and *Controls* are connected. They are connected through the minor function of *Securing its Position* on the asteroid which lies under the major function of *Controls,* and this is connected to the minor functions of *Preventing Tipping* and supporting weight of the lander which lie under the major function of *Support*. To be secure on the uneven surface of the asteroid, the landing system needs to be able to prevent the entire payload from tipping during the process of securing itself. It needs to be able to support the weight of the lander once the landing system is stabilized and is secure on the surface.

The major functions of *Sensors* and *Controls* are further related. They are connected through the minor function of *Monitoring the Lander* and its sub functions which lie in the major function of *Sensors*. They are connected to the minor functions of *Plans Safe Descent* and its sub functions. The control system onboard will plan a safe descent which involves reducing the velocity and adjusting the orientation of the lander. To be able to control those variables the lander needs to be able to sense its own position, velocity, and angle relative to the surface on Psyche.

The major functions of *Controls* and *Support* are also connected. They are connected through the minor functions of *Protecting the Payload from Impact Forces* and *Dampens Impact Forces* which are located under the major function of *Controls.* In addition to this, they're connected through the minor function of *Securing the Position on the Asteroid* which is located under the major function of *Support*. While the landing system is securing its position on the surface of Psyche it is important that the process does not damage the payload and all the components onboard. To successfully protect the payload, the impact forces will need to be dampened.

1.3.2 Action and Outcomes

When the lander is released from a spacecraft around Psyche, it will need to use its sensors to monitor the lander's position relative to the surface on Psyche. As the lander approaches the surface the control system on the lander will plan a safe descent which involves being able to reduce velocity to a safe impact velocity. Once the internal sensor senses the lander is at a safe velocity the controls system will begin to command the leg actuators to modify their position to secure the landing on Psyche. While the lander is securing its position, the landing system will need to be able to support the weight of an attached payload and will need to protect this payload from being damaged.

1.4 Target Summary

Houses Payload

To ensure that the mission is successful the system must be capable of successfully landing while carrying the payload. As specified by the sponsor, the payload our system will be carrying is the CHONKE rover built by a previous class. The system must be able to house 0.65 x 0.48 x 0.3 m as payload. This is an estimate for the actual dimensions of CHONKE. To test this function, a box with the dimensions of the rover will be used as a dummy. Our prototype must be able to perform a landing while carrying.

Houses Sensors

The landing system will need to carry several sensors on its mission, requiring those for acceleration, velocity, surrounding topography and position relative to it, angular orientation, and total contact between Psyche and appropriate landing surfaces. Sensors for acceleration, velocity, and angular position are relatively small, ranging around 0.254-12.7 mm in length and volume of about 1.29-25.81 mm^2. Technology used to detect objects and measure distance are larger than accelerometers, using about 0.929-2.787 m^2 of space. Force sensors will confirm contact with the asteroid, having a volume around 5.81-19.35 mm^2. Adequate spacing will be needed to cushion any impact forces that may affect any instruments.

Houses Actuators

The landing system will require motors to fulfill its function and react as necessary. Motors on this device will be expected to take up to 0.0186 m^2.

Houses Electrical Board

The electrical board that the landing system will need to supply electricity and such to the components will be 37 g in weight, and 101.52 x 53.3 mm. The landing system will need to be able to accommodate the size of this electrical board.

Houses Wiring

The wiring transfers electricity to different components in this project and needs to be able to fit inside the landing system. The wires used will have a diameter of about 0.05 mm each.

Prevents Tipping

The system must be able to prevent the lander from tipping. The landing system must be able to correct a 20-degree tipping angle of the lander in order to prevent tipping. This will be measured by subjecting the system to an angle of 20 degrees. To determine if the target is meet, the system must not tip over after being subjected to this inclination.

Support Weight

The system must be able to support the payload and other instruments that might be on board the lander. Based on research done by the team and advice from the sponsor, the team has set a target weight of 21.6 Newtons. In order to test the landing system, different weights will be placed on top of the physical model to see what the landing system can support.

Prevents Environmental Damage of Hardware

To prevent hardware damage and keep the lander operational, the system must be able to keep regolith and any order external debris from entering the hardware housing. The hardware needs to function while the lander is descending and while the lander is on the surface. This means that no regolith (space dust) penetrates through the housing and is on any internal components. Typical size of this dust is 0.1 mm. To test this, particles the size of regolith can be poured over the housing and then be checked inside if any dust gets through.

Dampens Impact Force

The system must be able to dissipate or dampen the impact force from the landing. The team has set a target of 6 m/s as the maximum impact velocity the system must be able to withstand. Using the kinetic energy formula, the energy that the system must be able to dissipate can be calculated. The system must be able to dampen and/or dissipate 2700 Joules. This is assuming a maximum weight of 150 kilograms. To test this, we will subject the system to a drop test that would replicate the conditions it will experience at the asteroid. Using the potential energy equation, we can determine what altitude will produce an impact of the same force. Therefore, the system will be dropped from an altitude of 1.84 meters.

Reads Acceleration

The acceleration of the lander is an important property to know because impact force is directly related to the acceleration of the lander. The acceleration of the lander relative to Psyche will be measured with a sensor. The target of the sensor is to be able to sense the acceleration of the lander at a resolution of 0.1 m/s^2 and the error must not exceed 10%. To test the sensor, we can attach it to objects and do drop tests. The value on the sensor will be compared to the gravity of Earth.

Reads Velocity

The velocity of the lander will be derived from the position or acceleration sensor. This means that the computation of the velocity of the lander needs to be able to be computed at a resolution of 0.1 m/s and be within 10% of the real value. To test this, the computation software can be given acceleration values or position values for a time interval and the velocity will be calculated from those values.

Reads Position

As the lander approaches the surface of Psyche, the distance from the lander to the surface needs to be sensed during the descent. The target of the sensor is to work at a resolution of 0.1 m and the error must not exceed 10%. To test the position sensor, the sensor will be pointed at objects at a known distance and the value of the sensor will be compared to the known value. Position will be determined from the surrounding topography.

Reads Angle

The angle of the lander with respect to the surface of Psyche is important to make sure that nothing besides the landing system encounters the surface. A sensor is needed on the lander to monitor angle relative to the surface. The target of the sensor is to work at a resolution of 0.1 degrees and the sensor must not have an error of more than 10%. To test the sensor, the sensor can be attached to an object that is normal to the surface and able to be titled. The value of the sensor will be compared to the real value of the tilt given to the object.

Reads Topography

A scanner will be used to survey the terrain surrounding the lander and determine the distance from the system to surrounding objects in meters. The system must be able to deliver topography information to the legs to adjust the angle accordingly. The sensors must be accurate enough to measure elevations on the surface with an error of 0.2 meters. To test this, the sensors will be placed in a test rig where the elevations of the surface are known. The sensors must deliver the elevation values within 0.2 meters of the known parameters.

Adjusts Orientation

The landing system needs to be able to adapt to different terrains that may not be flat. There will be actuators that can adjust the angle of the lander relative to the surface of Psyche. The target of the actuator will be to make the angle of tilt of the lander base less than 20 degrees. The landing system will be put on different surface slopes and tested to see whether the landing system adjusts to be under 20 degrees of tilt. A set position will be selected before landing begins and information from position and angular orientation will be used to calculate what adjustments are reasonably needed for our lander to be successful.

Secures Position on Asteroid

The position of the lander must not change after a set amount of time after touchdown. The team set a target of 10 seconds after landing to give time for the legs to adjust the angle of the lander. After 10 seconds have passed, the lander must not have any change in position. To test this target, a physical, full-scale prototype will be dropped from a set distance onto an uneven terrain. After the set time has passed the prototype must not change position to be deemed successful. Secured position will be measured with force sensors on the system expected to come in contact with the asteroid. The actual value will then be compared against the expected value.

Reduces Velocity

As the lander approaches the surface it will need to be able to reduce its velocity or the landing system will not be able to dissipate the impact energy. There will be a control system on board that is able to reduce the velocity to less than 3 m/s before touchdown. The control system for reducing velocity will be simulated through software.

Support Mass of the Lander

Not discussed/shown in our functions from our functional decomposition is the target mass of the lander that the landing system will need to support. This will be estimated at 150 kg. In testing the mass will be simulated by a box.

Impact Velocity

Not discussed/shown in our functions from our functional decomposition is the impact velocity with which the landing system will need to withstand. Previously mentioned is the target of reducing the velocity to 3 m/s before touching the surface. However, the landing system will ideally withstand an impact velocity of 6 m/s in the case that the methods to reduce impact velocity fails.

The targets and metrics were based on previous landers on small planetary bodies and on Mars missions. The missions to the small planetary bodies have similar conditions and constraints to our mission while the Mars missions have more modern equipment to account for the different variables needed to consider. From this list of targets and metrics the critical functions were dampens impact forces, prevents tipping, reads topography, and secures position on asteroid. These were chosen as our critical targets and metrics because the goal of the project is to create a landing system for a lander that will contact hypothesized terrains on Psyche. For the lander to get on the surface of Psyche safely, the landing system should be able to dissipate any impact energy to prevent it from damaging the rest of the lander and the instruments. In order to further reduce risk to the lander, the landing system should prevent the entire lander from tipping over and damaging the instruments onboard. The Psyche surface is currently unknown which makes it important that the landing system can land on any of the hypothesized terrains such as high relief, rocky, or metallic. Finally, the last critical target and metrics ensures that the lander is stable on Psyche and not moving around or achieving escape velocity. These are the four critical things the landing system must be able to do to potentially succeed in the project.

1.5 Concept Generation

Concepts were generated for our project and eight concepts were sketched/modeled. A variety of methods were used to generate ideas. The first method used was brainstorming. All four members of the team added any concepts that came to mind to the concept list. This provided a large variety of concepts that team members were further able to bounce ideas off to create a few more. The next method used was biomimicry. Biomimicry was used to look at plants and animals that can absorb energy and adjust its level. Finally, crap shoot and forced analogy were the final methods used. The crap shoot method had each member list six items for each category of people involved, common activities, and potential resources for the involved project. Then three numbers were chosen, and a concept had to be generated from the items selected from each category. This allowed more creative concepts to be easily produced. Forced Analogy is done in a similar manner where the team members first generated a random list of things. Concepts were then generated using attributes of the objects on the list. The result is a list of one hundred concepts in Appendix E. From this list, ideas were combined and altered to produce the following eight concepts below.

Medium Fidelity Concepts

Concept 1

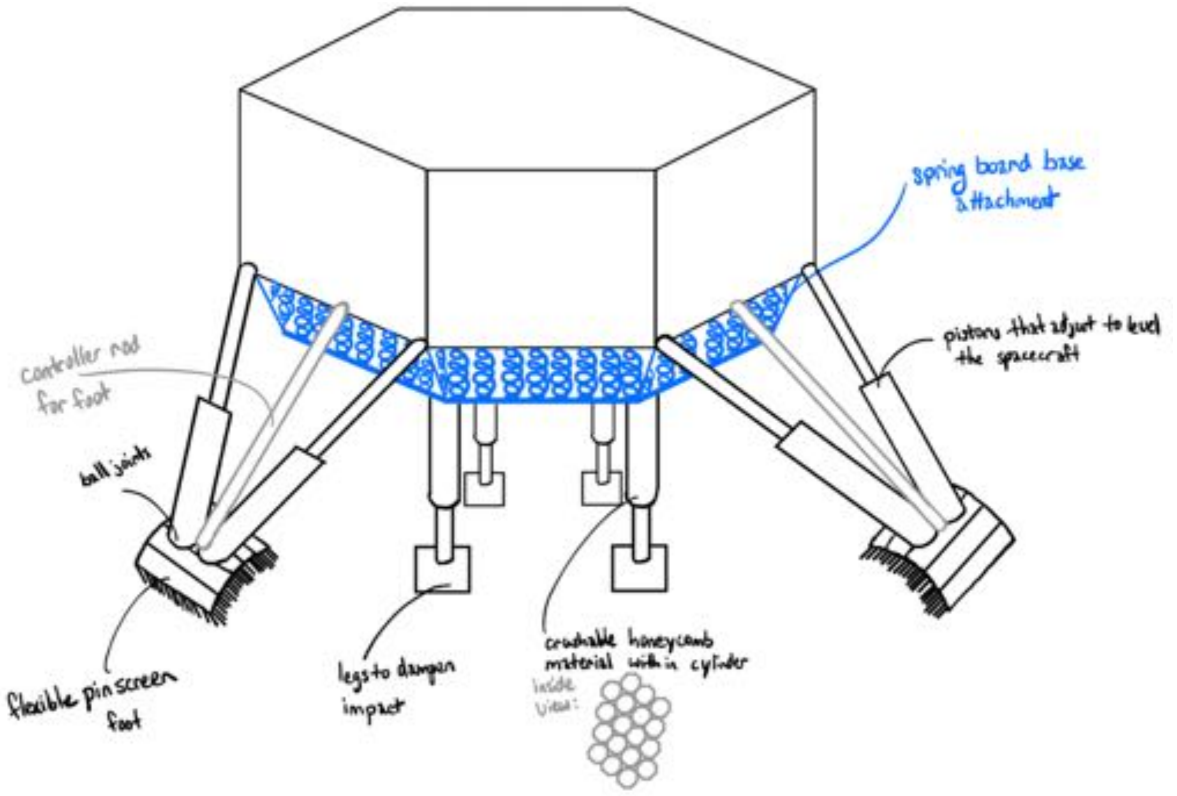

spring board base attachment
controller rod for foot
ball joints
pistons that adjust to level the spacecraft
flexible pin screen foot
legs to dampen impact
crushable honeycomb material within cylinder
inside view:

Figure 2. Concept 1

Figure 2 shows the first medium fidelity concept. This design consists of a base modeled as a hexagon featuring a springboard base attachment that is connected to four legs. Both the four legs and the springboard are meant to absorb the initial impact energy, acting as a damper for the system. The legs would consist of a honeycomb structured material located inside of cylinder and as the system lands the leg would be pushed into the honeycomb cylinder, crushing the honeycomb structure inside. These legs would touch the surface first. After the landing system is slowed down, three additional legs would deploy for stability. These legs would be attached to the base of the spacecraft in a triangle with pistons on the two connecting arms. The pistons would be able to maneuver the position angle of the spacecraft. Feet are attached to the legs via ball joints. The feet at the end of the stability legs are modeled after a pin screen. As the feet contact an uneven surface, the pins would conform to that terrain. The pins would be pushed up and into a small gap in the foot where a backing of metal would stop them, locking it in place. The feet are also flexible and would slightly wrap if the system were to land on more rocky terrain where gripping was needed.

Concept 2

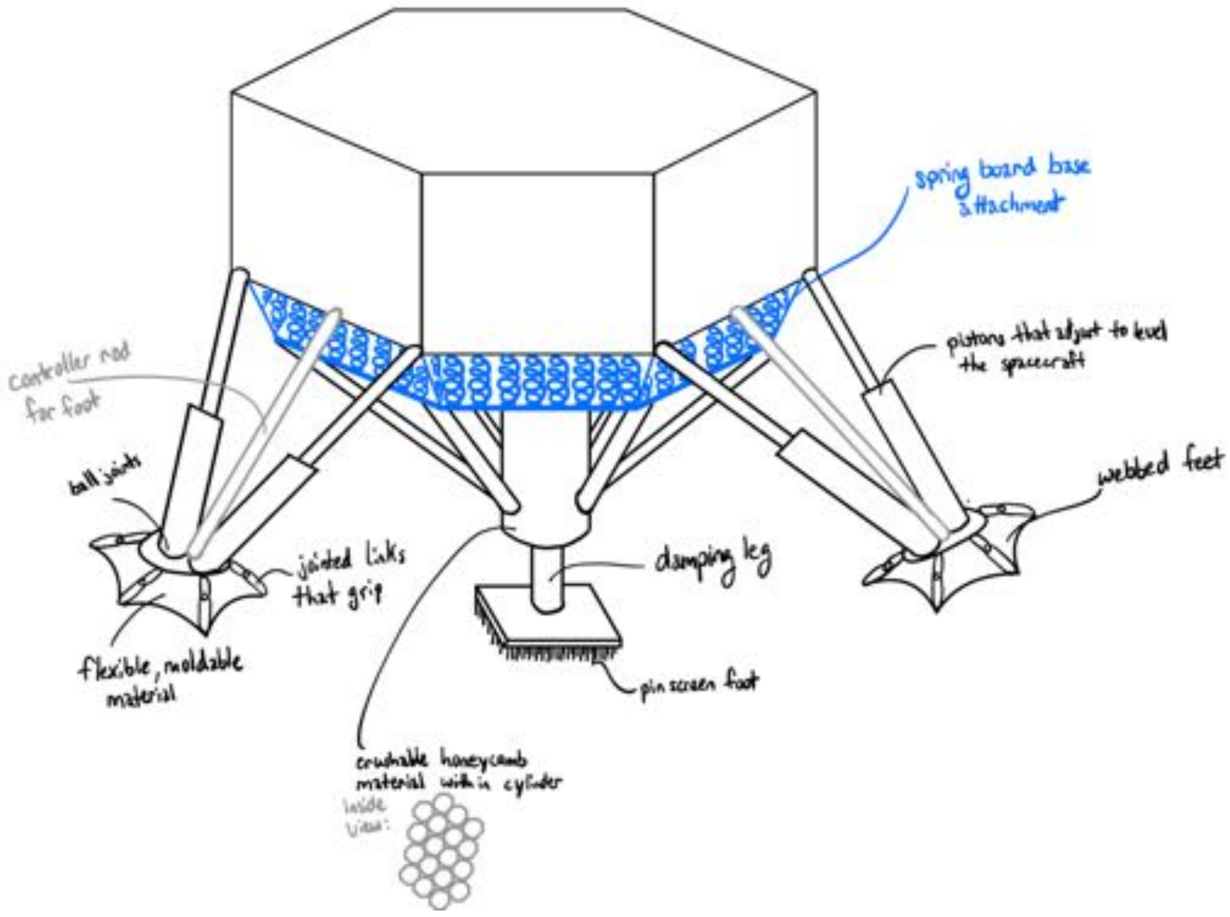

spring board base attachment
controller rod for foot
pistons that adjust to level the spacecraft
ball joints
webbed feet
jointed links that grip
damping leg
flexible, moldable material
pin screen foot
crushable honeycomb material within cylinder
inside view:

Figure 3. Concept 2

Figure 3 shows the second medium fidelity concept. This design consists of a base modeled as a hexagon featuring a springboard base attachment that is connected to one large leg support. This large leg would be connected to each outer corner of the springboard attachment via metal rods to distribute the energy. Both the large leg and the springboard are meant to absorb the initial impact energy, acting as a damper for the system. The large leg would consist of a honeycomb structured material located inside of the cylinder and as the system lands the leg would be pushed into the honeycomb cylinder, crushing the honeycomb structure inside.

Attached to this leg is a pin screen foot. As the foot contacts an uneven surface, the pins would conform to that terrain. The pins would be pushed up and into a small gap in the foot where a backing of metal would stop them, locking it in place. This leg would touch the surface first. After the landing system is slowed down, three additional legs would deploy for stability. These legs would be attached to the base of the spacecraft in a triangle with pistons on the two connecting arms. The pistons would be able to maneuver the position angle of the spacecraft. Feet are attached to the legs via ball joints. The feet at the end of the stability legs are modeled as webbed feet. The webbed portion would be made of a moldable medal and there would be jointed "fingers." These fingers would be able to grip onto the surface while the webbing would assist in molding to any uneven terrain that may be encountered. The motion of these fingers would be controlled via wires leading up to the control panel through the control rod.

Concept 3

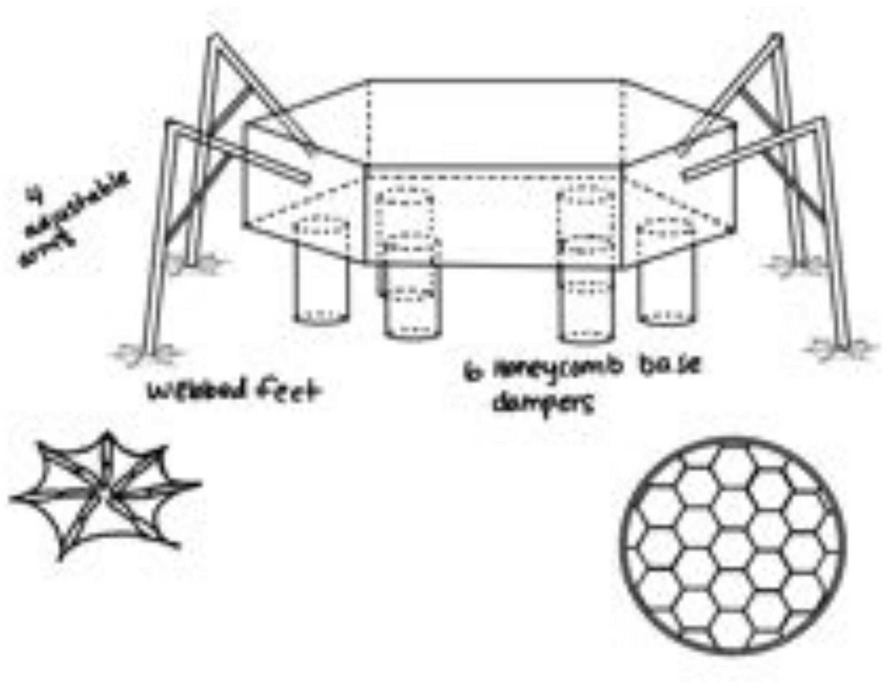


Figure 4. Concept 3

The third medium fidelity concept is shown in Figure 4. The design uses 6 honeycomb legs advantageously positioned for the most efficient impact force distribution. These base legs consist of a honeycomb structure that will deform on impact, damping the applied load from landing. On the sides of the hexagonal base are the positioning legs. They are attached to the side of the base and provide stability to the craft. They move at 2 joints on the legs, where the leg and base meet and at the elbow. At the end of the legs are the webbed feet attached with ball joints, which are made of flexible material and 'fingers', as described earlier. They are able to grip on to the asteroid surface, conforming to the uneven edges of the ground.

Concept 4

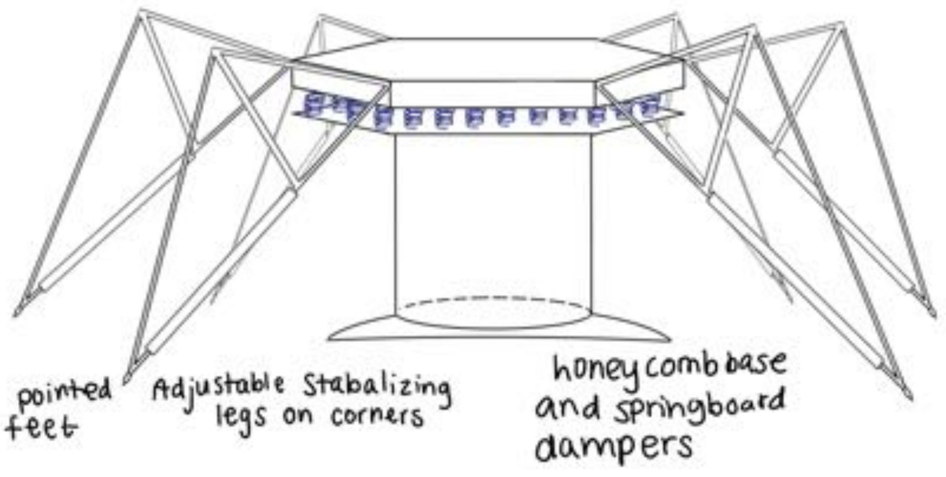

pointed feet
Adjustable Stabalizing legs on corners
honeycomb base and springboard dampers

Figure 5. Concept 4

The fourth medium fidelity design shown in Figure 5 uses a springboard base for extra damping. The baseboard will provide extra support to the base leg. The design uses a large honeycomb leg in the middle of the base that will absorb most of the impact energy, transferring any excess upwards to the springboard base that will take in the rest of it. Surrounding the perimeter of the hexagonal base are legs that can rotate at the joint between the leg and the base and have pistons that can extend and reach crevices on the uneven surface for the pointed feet to push against, providing stability for the craft.

Concept 5

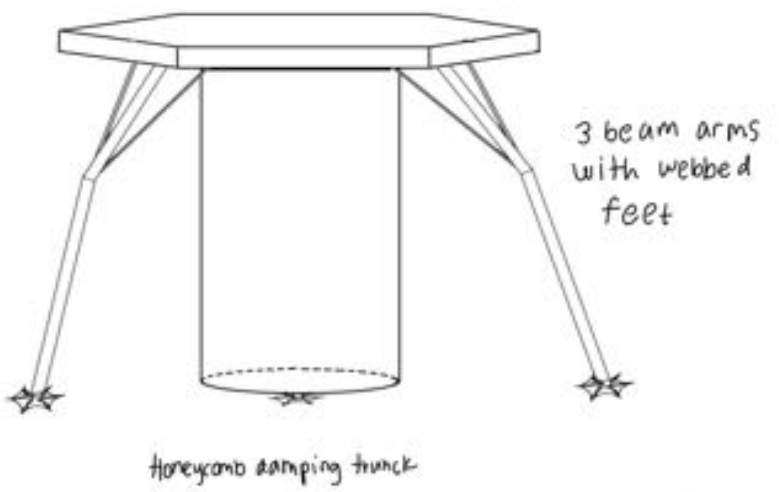


Figure 6. Concept 5

The final medium fidelity concept shown in Figure 6 uses a large base to bear the impact force from landing. Inside is a honeycomb structure that will crush upward and dissipate the force from landing. The bottom of the trunk is made of deformable material that will conform to the shape of the surface and provide extra grip for the craft. Surrounding the base leg are 3 stabilized legs made of three branches. The larger middle branch is made of the same honeycomb structure that is in the base leg. The ends of the legs are connected to the same webbed feet as earlier with a ball joint, providing a large range of rotation for the feet to find the proper orientation to grip on to the surface to.

High Fidelity Concepts

For the high-fidelity concepts below, the feet are all designed to be a flexible pin screen shown in Figure 7. The feet feature pins that would conform to the terrain as it contacts the uneven surface. The pins would be pushed up and into a small gap in the foot where a backing of metal would stop them, locking it in place. This design also includes a flexible "spine" of hinged, overlapping plates to allow for the foot to be able to grip onto rocky, uneven terrain.

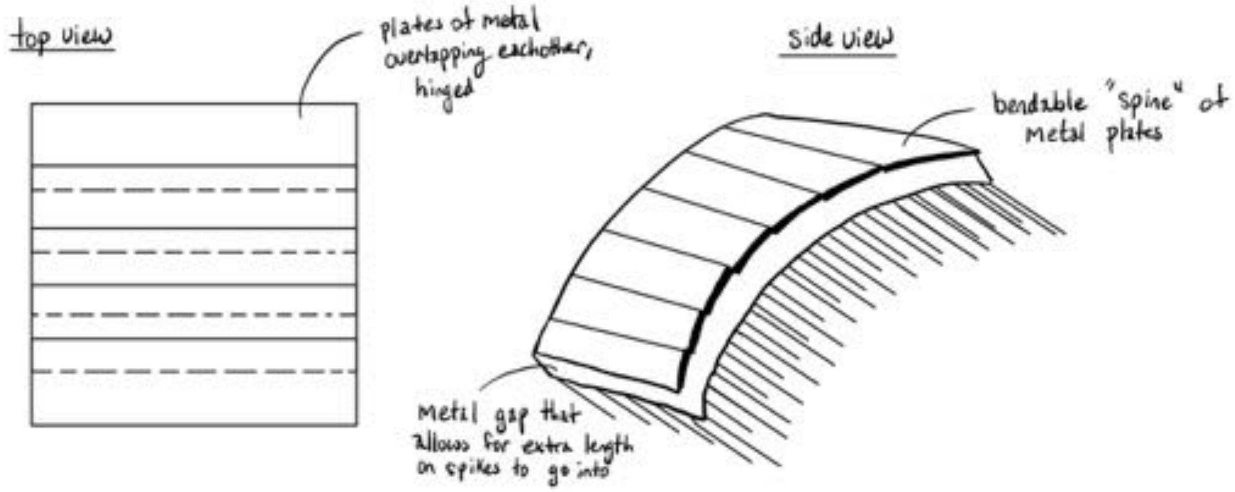

top view
plates of metal overlapping eachother, hinged
side view
bendable "spine" of metal plates
metal gap that allows for extra length on spikes to go into

Figure 7. Pin Screen Foot Design for Concepts 6-8

Concept 6

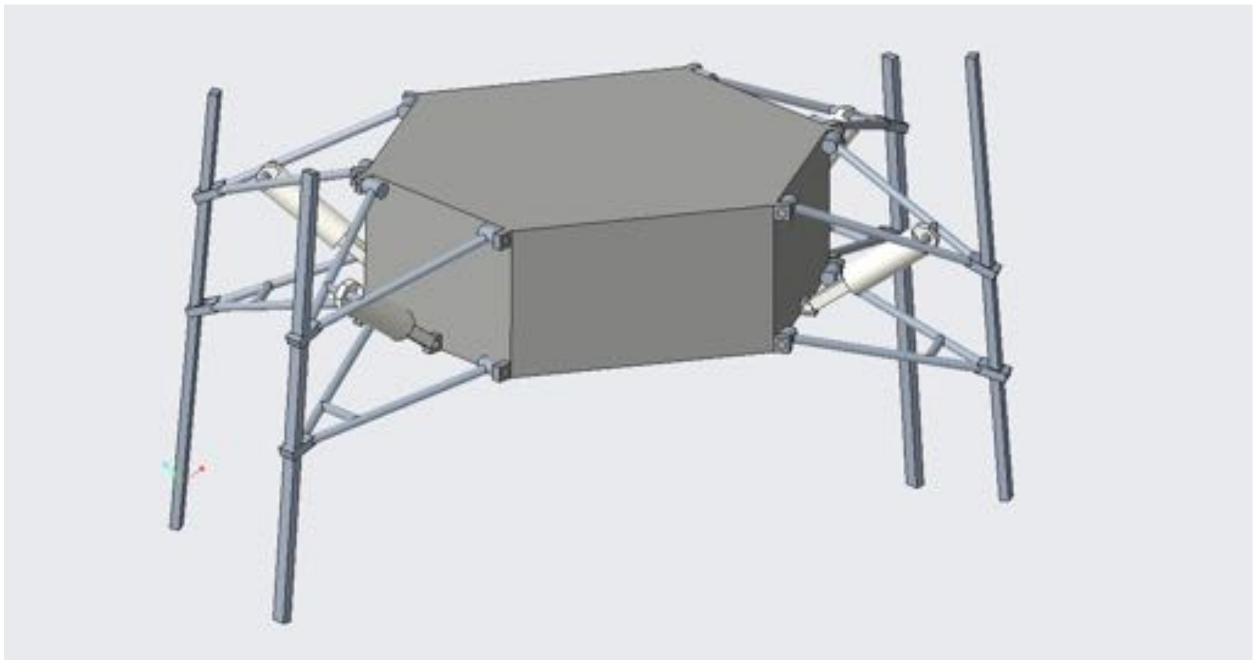


Figure 8. Concept 6

Figure 8 shows the first high-fidelity concept featuring a hexagon shaped spacecraft with four double A-arm suspension legs. These legs have a cylinder device in between the two A-arms that are meant to absorb the impact energy, acting as a damper for the system. This device would consist of a honeycomb structured material located inside of cylinder and as the system lands the rod would be pushed into the honeycomb cylinder, crushing the honeycomb structure inside. Connected to both ends of the two A-arms is an adjustable rod that would adjust its position to level the spacecraft to its desired angular position. At the end of these sliding adjustable rods would be pin screen feet shown in Figure 7 above. These would be attached via ball joints.

Concept 7

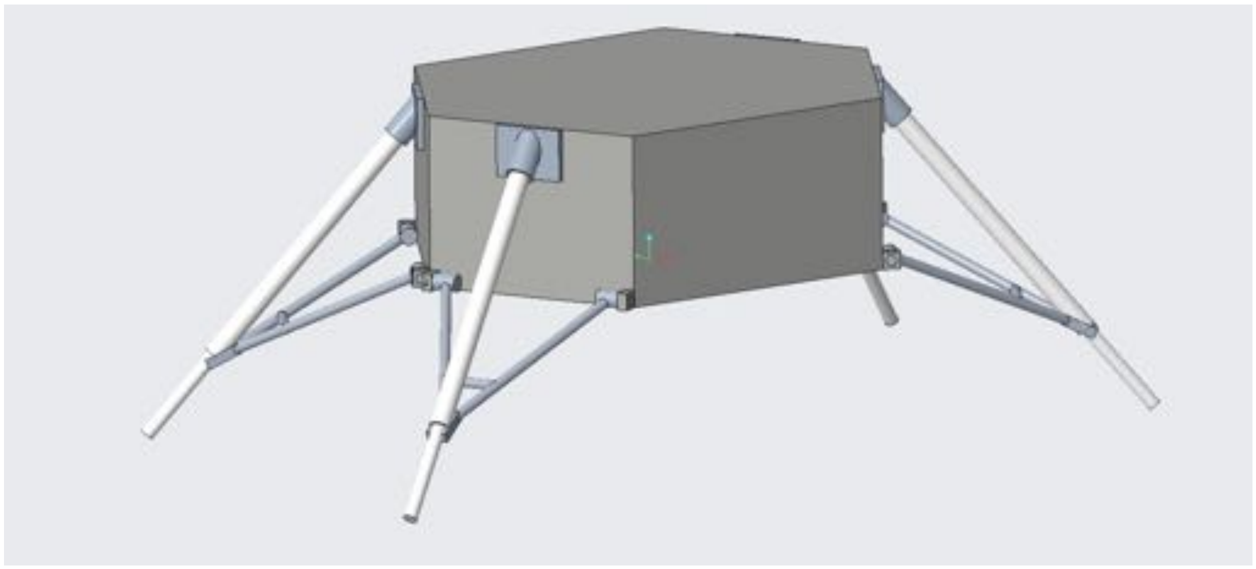


Figure 9. Concept 7

Figure 9 shows the second high fidelity concept featuring a hexagon shaped spacecraft with four adjusting legs attached. These legs feature a piston assembly rigidly attached to the spacecraft that allows for the rod inside the piston cylinder to slide in and out, adjusting the angular position of the spacecraft to make sure it is level. This piston assembly is support by an A-arm at the bottom of the base. At the end of these adjustable legs would be the pin screen foot shown in Figure 7. These feet would be attached via ball joints.

Concept 8

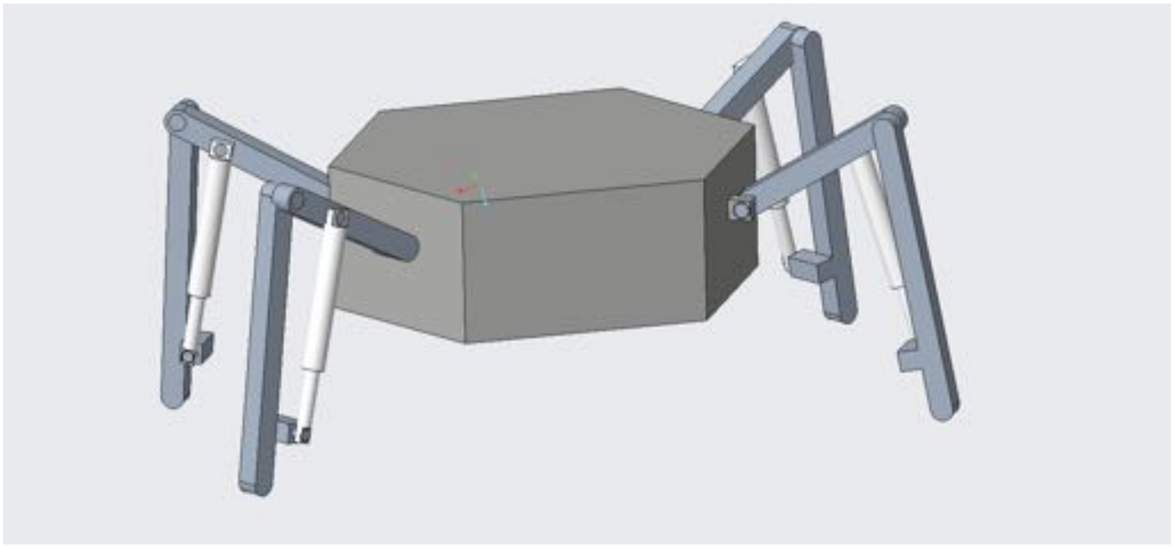


Figure 10. Concept 8

Figure 10 shows the third high fidelity concept. It consists of a hexagonal base with four legs attached to it. The legs mimic grasshopper legs with a piston in each leg for dampening and adjustment. The height of the piston can be adjusted to change the overall height of each leg, allowing for independent height adjustment of each leg. Upon landing the legs will be fully extended to dampen the impact from touchdown, after touchdown the legs will adjust their height to level the spacecraft. Attached to the end of the legs would be the pin screen foot shown in Figure 7. These feet will be attached with ball joints for better adjustment.

1.6 Concept Selection

Binary Pairwise

Binary Pairwise comparison sequentially examines two customer needs at a time to order the needs in terms of most important to least important to the purpose of the project. The customer needs are written out twice, along the horizontal and vertical axes. Then, starting with the first column, the customer needs lined up with that cell are compared, that row customer need is compared against the column customer need, resulting in either a '1' or a '0'. A '1' is assigned when the row need is more valuable than the column need. Our binary pairwise comparison found the customer need of withstanding the potential energy from the fall to be most important.

House of Quality

The house of quality shown in Table 3 ultimately gives us a ranking of the engineering characteristics governing our project from most important (1) to least important (8).

Table 3.

House of Quality

		Engineering Characteristics							
Improvement Direction		↓	↑	↑	↑	↑	↑	↓	↓
Units		m^2	kg	m; m/s; m/s^2;deg	deg to tip	N	m	deg	m
Customer Requirements	Importance Weight Factor	Houses Components/Hardware	Supports Weight	Reads Lander Data	Prevents Tipping	Dampens Impact Energy	Senses Sourrounding Topography	Adjusts Orientation	Secures Position on Asteroid
The system is autonomous	1	1		9	9		9	9	9
Supports the spacecraft and associated components	2	3	9		3	3		3	3
Withstands or dissipates the potential energy from the fall and impact velocity	3		9	3		9			1
Adjusts to the hypothesized terrains of Psyche	1			9	9	1	9	9	9
The system does not have to be reusable	0	1				3			3
Raw Score	206	7	45	27	24	34	18	24	27
Relative Weight %		3.40	21.84	13.11	11.65	16.50	8.74	11.65	13.11
Rank Order		8	1	3	5	2	7	5	3

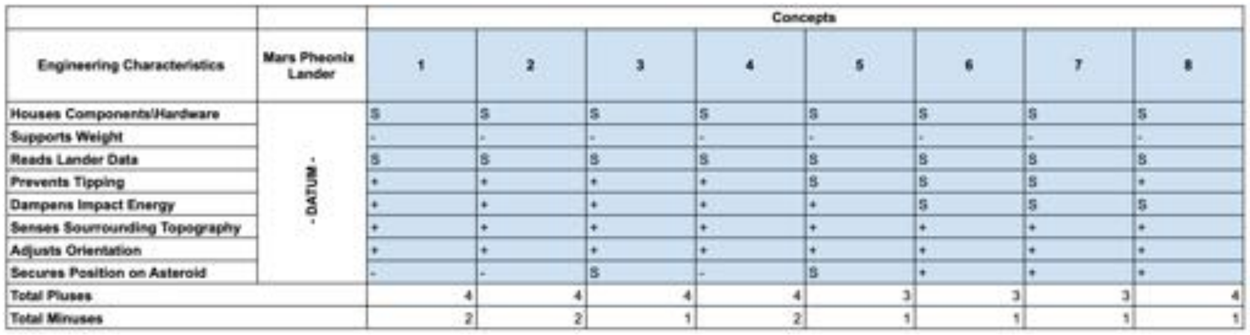
This table was constructed by putting in our customer requirements in the far-left column. An importance weight factor was assigned from the results of the binary pairwise table done previously (Appendix F). The columns in blue are the engineering characteristics from our functional decomposition that were thought to be most impactful to our project. Each of these were assigned units. To get the ranking, we went row by row asking whether the engineering criteria will contribute to fulfilling the customer requirement. Values of 0,1,3, or 9 were assigned, with 0 being no contribution and 9 being the highest level of contribution. The numerical values in the rows were multiplied by their importance weight factor and the values were then added by column. The relative weight is the percentage of each characteristic in relation to the total raw score. Based on these weights the ranking of engineering characteristics was achieved. Supports weight came out to be our most important engineering characteristic in satisfying our customer requirements, followed by dampens impact energy, reads lander data, and secures position on asteroid.

Pugh Charts

The first Pugh chart is shown in Table 4 which shows how each concept is ranked against a datum concept. A "+" is given if the concept is better than the datum, "-" if the concept is worse than the datum, and "S" if the concept is similar to the datum. Once a Pugh chart is filled in then the pluses and minuses in a column were summed. The worst concept is removed, the middle ground concept is made the datum, and the Pugh chart is restarted.

Table 4.

First Pugh Chart



		Concepts							
Engineering Characteristics	Mars Pheonix Lander	1	2	3	4	5	6	7	8
Houses Components/Hardware	- DATUM -	S	S	S	S	S	S	S	S
Supports Weight		-	-	-	-	-	-	-	-
Reads Lander Data		S	S	S	S	S	S	S	S
Prevents Tipping		+	+	+	+	S	S	S	+
Dampens Impact Energy		+	+	+	+	+	S	S	S
Senses Sourrounding Topography		+	+	+	+	+	+	+	+
Adjusts Orientation		+	+	+	+	+	+	+	+
Secures Position on Asteroid		-	-	S	-	S	+	+	+
Total Pluses		4	4	4	4	3	3	3	4
Total Minuses		2	2	1	2	1	1	1	1

In the first Pugh chart, the Mars Phoenix Lander was chosen to be the first datum concept. The Mars Phoenix Lander had three legs and each leg had two components for stability and one spring component for damping. The chosen landing location for the Mars Phoenix Lander had flat surface topography.

The two engineering characteristics that were the same for all the concepts as the datum was houses components/hardware and reads lander data. This is because all of our concepts have made space for more instrumentation to be added on top and all of our concepts will use sensors on the landing system. All of our concepts got a minus compared to the datum because the Mars Phoenix Lander supported 550 kg while we are aiming for 100 kg. All of our concepts have pluses in the senses surrounding topography and adjusts orientation engineering characteristics categories because all of our designs are specifically created to account for different possible topographies. Concepts 1, 2, and 4 have minuses in the secures position on asteroid category because this engineering characteristic was defined as the distance moved after touchdown. These three concepts have springboards which the team assumes will cause a higher probability to move around if there are oscillations in the springboard. All concepts besides 6, 7, and 8 were seen to be able to absorb more impact energy than the datum because crushable structure and springs were viewed as able to absorb more energy than just piston which were used in 6, 7, and 8. From this chart, there was not a clear worst concept nor a middle of the road concept.

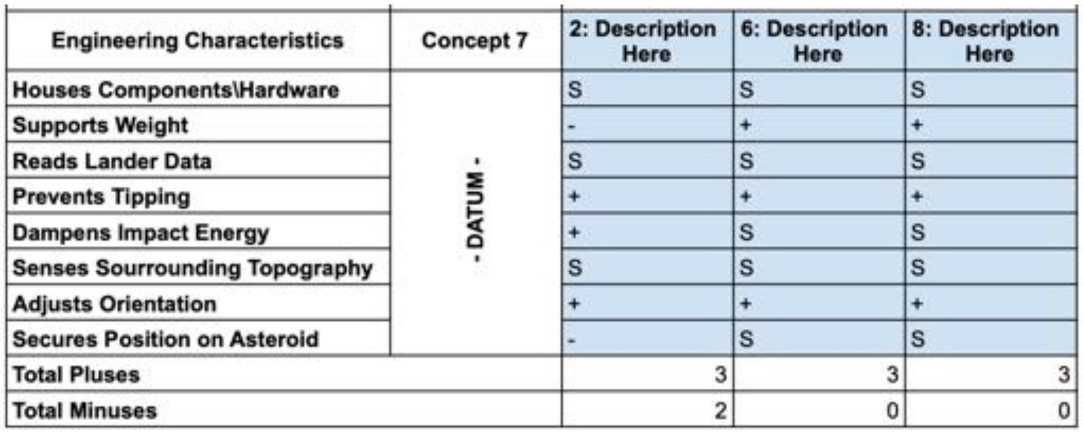
The second chart is in Appendix G Table G1 and the datum concept is concept 3. In the first Pugh chart all concepts were seen to be similarly better than the datum. As a team we decided that concept 3 was the concept that was most similar to existing spacecrafts while also fulfilling our customer needs. This allows for our innovative ideas to be compared against a concept that is similar to an already tested and tried design. The worst concept was concept 5 with zero pluses and two minuses. Concept 5 was seen as worse in the prevents tipping category due to there being little ability to adjust the legs and was worse in the supports weight category. The new datum was chosen to be concept 1 with one plus and one minus. Concepts 6 and 8 both are the leadings concepts in this chart with three pluses and no minuses each.

The third Pugh chart is in Appendix G Table G2 with concept 1 being the datum. Concept 4 was the worst concept with two minuses and zero pluses. Concept 7 became the datum with one plus and two minuses. Concept 6 and 8 are both leading concepts again with four pluses and one minus each.

The final Pugh chart is seen in Table 5 with the datum being concept 7. Concept 7 had rigidly attached pistons with extendable legs to control the angle of the lander. Concept 6 and 8 were able to support more weight due to the geometry of the design being better for more load. All the concepts were able to prevent tipping better than concept 7 due to their leg geometries being better shaped for stability. All concepts were also able to adjust the orientation better than concept 7. This is because concept 7 can only adjust orientating by extending or retracting a rod while the other designs are able to adjust in more stable methods. This final Pugh chart shows that concepts 6 and 8 are the strongest concepts when compared to our other concepts and an already existing spacecraft lander.

Table 5.

Last Pugh Chart



Engineering Characteristics	Concept 7	2: Description Here	6: Description Here	8: Description Here
Houses Components\Hardware	- DATUM -	S	S	S
Supports Weight		-	+	+
Reads Lander Data		S	S	S
Prevents Tipping		+	+	+
Dampens Impact Energy		+	S	S
Senses Sourrounding Topography		S	S	S
Adjusts Orientation		+	+	+
Secures Position on Asteroid		-	S	S
Total Pluses		3	3	3
Total Minuses		2	0	0

AHP

The Analytical Hierarchy Process (AHP) breaks down complex decisions into a series of one-on-one comparisons. In the first stage, shown in Table 6, the engineering qualities are ranked against each other with 1 denoting equal weight and 9 denoting a strong preference to one over the other. This first iteration arrives at a weight factor for each engineering characteristic. These weight factors are then used to compare the engineering characteristics of the three high fidelity concepts and the result yields the best concept for the design.

Table 6.

AHP

	Supports Weight	Dampens Impact Energy	Prevents Tipping	Secures Position on Asteroid	Reads Lander Data
Supports Weight	**1.00**	3.00	1.00	1.00	0.33
Dampens Impact Energy	0.33	**1.00**	1.00	0.33	0.11
Prevents Tipping	1.00	1.00	**1.00**	1.00	3.00
Secures Position on Asteroid	1.00	3.00	1.00	**1.00**	9.00
Reads Lander Data	3.00	9.00	0.33	0.11	**1.00**
Sum	6.33	17.00	4.33	3.44	13.44

The AHP was then repeated to compare the engineering characteristics against the three different concepts selected from the Pugh Charts. A total of five different AHP charts were done, and they can be found on Appendix H. The results from these were put together in a single table, as shown in Table 7.

Table 7.

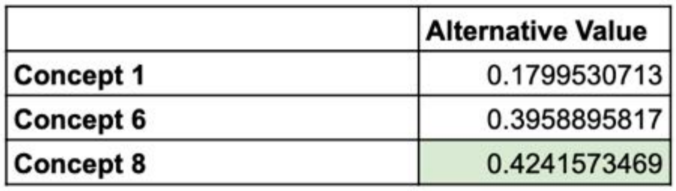
Final Rating Matrix

Final Rating Matrix			
	Concept 1	**Concept 6**	**Concept 8**
Supports Weight	0.14	0.43	0.43
Dampens Impact Energy	0.05	0.32	0.63
Prevents Tipping	0.14	0.43	0.43
Secures Position on Asteroid	0.14	0.43	0.43
Read Lander Data	0.33	0.33	0.33

The values from Table 7 were then multiplied by the engineering characteristics weight factors obtained in Table 6. The results are shown in Table 8 below. This table shows the design concept that best meets the customer requirements.

Table 8.

Alternative Value Matrix



	Alternative Value
Concept 1	0.1799530713
Concept 6	0.3958895817
Concept 8	0.4241573469

Final Selection

Based on the analysis on the five medium fidelity concepts and the 3 high fidelity concepts, the design picked by the team is the Double A-arm Rack and Pinion System (Concept 6) shown in Figure 8 above. This design consistently ranked high in our charts and meets all the customer requirements. Although this is not the concept that the charts ranked highest, it was a close second. This design is also more feasible and is thought to have more dampening potential than concept 8. It is a design that has been proven effective across different applications.

Chapter Two: EML 4552C

2.1 Restated Project Definition and Scope

The objective of this project is to design a landing system capable of safely landing on the assumed range of hypothesized surfaces and terrains of 16 Psyche. The NASA Psyche Mission is set to orbit the asteroid 16 Psyche in 2026. After studying the asteroid many scientists and engineers might be interested in proposing a mission to land on it at a future date.

The goal of this project is to create a landing system that will provide safe landing on 16 Psyche for a future spacecraft. To provide a safe landing, the landing system must be able to account for various types of terrain such as high relief terrain, terrain with rocky debris, and metallic terrain. The landing system must also be able to prevent the spacecraft from tipping and flipping. If the spacecraft tips or flips, then the spacecraft could be potentially damaged and rendered useless. The landing system must be able to absorb the impact upon landing the spacecraft to prevent any component failure within the spacecraft.

This project will primarily appeal to NASA for any future landing spacecrafts that are scheduled to land in places where there are no guaranteed flat surfaces. This project will have a secondary appeal to private space companies and the robotics industry. The design could be used by other private or foreign space agencies for landing missions where the landing terrain is not optimal. It may appeal to the robotics industry because a large issue is being able to make robots that can overcome dynamic terrains.

Throughout the project there are assumptions that are made. We assume that the landing system will be operated in minimal gravity and that it will be able to operate in space temperatures and conditions. We assume that the spacecraft will be able to perform a soft landing on uncertain hypothesized terrains that may include mostly flat metallic surface, flat metallic with metal and/or rocky debris, rough/high relief metallic and/or rocky terrain, and high relief metallic crater walls. We assume that the landing system designed will be able to be attached to a future spacecraft without issue. The power to operate the system is assumed to be supplied from the spacecraft. This system is assumed to be controlled autonomously without manual maneuvering of the system.

Those who are stakeholders in this project include both administrations and people alike. Arizona State University (ASU) is a stakeholder through the sponsorship of the project. The National Aeronautics and Space Administration (NASA) is a stakeholder since they are partnered with ASU and are making the mission to 16 Psyche possible. Individuals who are also stakeholders include Dr. Cassie Bowman, co-investigator on the Psyche Asteroid Mission with ASU, as she provides us information related to our project. Dr. Shayne McConomy is a stakeholder as he has invested time into setting up the sponsorship; expending time/effort to assist us as students with our project to make sure that we as a team, have the skills necessary for a successful engineering future. As our advisor, Dr. Camillo Ordonez is investing time and effort into meeting with our team, providing guidance and advice. Stakeholders are also those interested in scientific discovery, as well as taxpayers, who collectively fund NASA's budget.

2.2 Results

Tab*le 9. Validation Legs Extended*

Validate the Legs Extended 20.32 cm			
Test Run #	Leg 1 Length (cm)	Leg 2 Length (cm)	Leg 3 Length (cm)
1	19.65	21.55	21.40
2	19.21	20.86	21.51
3	19.45	21.16	21.87
4	19.10	21.18	21.77
5	19.33	21.40	21.99
Desired Length	20.32	20.32	20.32
% Error	4.8%	4.5%	6.8%

Table 10. Validate Tipping Prevented

Validate Tipping Prevented		
Test Run #	Pitch (deg)	Roll (deg)
1	-4	-5
2	-3	-5
3	-6	-4
4	-5	-3
5	-4	-5
Maximum Angle (deg)	-5	-5
% Failure	20%	0%

Table 11. Condition of System

Condition of Damping System and Components			
Test Run #	Wiring Inside	Dampers	Legs
1	2	2	2
2	2	2	2

3	2	2	2
4	2	2	2
5	2	2	2
Ranking: 2 = Good 1 = Acceptable 0 = Unacceptable			

Table 12. Validation of Impact Velocity

Validate Impact Velocity			
Test Run #	Distance (m)	Time (s)	Velocity (m/s)
1	0.0701	0.067	1.05
2	0.0809	0.067	1.21
3	0.0775	0.067	1.16
4	0.0627	0.067	0.935
5	0.0732	0.067	1.09
Desired Velocity (m/s)		0.92	
% Error		18%	

Table 13. Validation of Surface Accommodation

Validate the Legs Accommodate Surface			
Test Run #	Leg 1 Length (cm)	Leg 2 Length (cm)	Leg 3 Length (cm)
1	7.60	16.51	33.10
2	6.35	15.26	33.08
3	5.06	13.97	34.22
4	5.12	16.55	35.48
5	6.42	16.48	35.66
Desired Length	5.08	15.24	35.56
% Error	20%	3.4%	3.5%

2.3 Discussion

From these tests, the following parameters were collected; legs extended 20.32 cm, prevented tipping, undamaged hardware, impact velocity, and legs accommodated surface. The first step for the mechatronics is to extend the linear actuators halfway out, 20.32 cm. The linear actuators do not have encoders, motion in the linear actuators is dictated by the amount of time power is supplied to them in a certain direction, distance travelled in given by stroke rate, so errors are expected. The highest rate of error was seen on leg 3, at 6.8%. This is shown in Table 9. The different errors are attributed to the varying stroke rate on each leg from wear and the power supplied to them. After landing on the uneven testbed, the pitch and roll were collected to check the intensity of tipping, this is shown in table 3. A successful landing was characterized by orientation angles of less than 5°. Out of 5 tests, only one test resulted in one angular offset greater than 5°. After testing, the damage to the lander was assessed. The impact test has the risk of causing electronics to come out of place and deforming the dampers and legs. These components were visually inspected to ensure all they were in proper condition. Test scripts were also run to check the mechatronic connections. After each test, the prototype was in satisfactory condition to run the test again. The results of the condition after each test are shown in Table 11. The prototype was created to withstand a 0.92 m/s impact velocity for the earth prototype. Table 12 shows the impact velocities the prototype approached the surface with for each run. Every impact velocity tested was above the desired velocity 0.92 m/s. Our model successfully landed and withstood the impact energy at the goal impact velocity. From our testing bed, the length of the legs is known for a levelled orientation. After landing and running checks on the leg lengths the leg lengths were measured. The measurements are shown in table 13. There was a 20% error on leg 1 which could be explained by an inconsistent stroke rate from the amount of current supplied under the load of carrying the prototype's weight.

2.4 Conclusions

The uneven, unknown surface of Psyche presents a design challenge to create a landing system based on what is currently known about the asteroid. From our testing and results, a successful design was created to land on an asteroid with uneven terrain over a hypothesized range of surfaces. Our feet were able to adapt to the rocky terrain of the test bed and provide ample grip to secure the position of the lander. The legs were able to achieve a nearly levelled orientation without any feedback control, as well as successfully transfer impact energy to the suspension without breaking. The damping systems on board were able to support the base without causing any damage to the system. The model created poses solutions to securely landing on the range of hypothesized surfaces on asteroid 16 Psyche.

2.5 Future Work

Moving forward, the model can be tested on a wider variety of test beds. The testing was done on one test surface with only 2 of the provided materials, lava rocks and black washed gravel. Given more time, different rocks and metal materials can be added to the testing surface.

Some of the components have Psyche counter parts, so further testing is needed for the honeycomb dampers. The honeycomb testing is destructive and requires a lot of material to determine how much damping is required.

Appendices

Appendix A: Code of Conduct

MISSION STATEMENT

To work collaboratively as part of the Psyche mission team to create an original, innovative design capable of successfully landing on a range of hypothesized surfaces.

DRESS CODE

All team members must wear business formal attire for presentations. During team meetings no specific attire is required. For sponsor meetings and faculty advisor meetings business casual attire is required. During class time casual attire is accepted. For any other events, the team will decide on casual or business casual attire depending on the event.

ETHICS

All team members will abide by the mission statement.

All team members will communicate their ideas respectfully.

All team members will deliver their tasks in a timely manner.

All team members will review deliverables before they are submitted.

All team members will respect the decisions and commitments made as a team.

All team members will attend all regularly scheduled team meetings.

OUTSIDE OBLIGATIONS

All team members are expected to include their regularly scheduled outside obligations in the team calendar. If any other obligation is to unexpectedly occur, it is expected that the team member will notify the other members accordingly.

TEAM ROLES

Point of Contact - Saralyn Jenkins

This person is responsible for sending emails to the sponsor and advisor. They will notify the team via the primary mode of communication when the emails have been sent.

Mechanical Systems Engineer - Saralyn Jenkins

The Mechanical Systems Engineer is responsible for the design and testing of the moving parts and their tolerances in the design. The Mechanical Systems Engineer must ensure there are no interferences between moving parts.

Materials Engineer - Elzbieta Krekora

This Materials Engineer is responsible for the research, and material selection of the design. The Materials Engineer must verify that the material selected can withstand the loads applied to the system.

Controls Engineer - Andrew Sak

The Controls Engineer is responsible for researching the hardware needed to accomplish the motion of the design. They are also responsible for the wiring and programming of hardware.

The Controls Engineer must ensure that the hardware is able to complete the motion within the time required.

Manufacturing Engineer - Julio Velasquez

The Manufacturing Engineer is responsible for the fabrication of the prototype. They are also responsible for the purchase or the required material and hardware. The Manufacturing Engineer must ensure the prototype is delivered by the required due date and that it meets the team's standards.

Other duties will be assigned according to skillsets of the individual team members and who is thought to be the best fit for a task based on majority team opnion.

COMMUNICATION STANDARDS

The primary mode of communication will be the group text chat to set up team meetings and address anything pertaining to the project. The second mode of communication will be email for non-time-sensitive matters and file transferring. Team members are expected to check the team group text chat and school email regularly, responding within 24 hours of original message being sent. A OneDrive folder will be used to post all finalized versions of documents. A Google Drive folder will be used for all documents that are being worked on. Communication with team sponsors and faculty advisors will be done via email or video calls unless otherwise specified by the sponsor or faculty advisor.

ATTENDANCE POLICY

Team meetings outside of class time will be held every other week at an agreed location and time. Team members will be notified of time and location of meetings 24 hours in advance. Team members are required to attend every regularly scheduled team meeting unless unforeseen circumstances arrive. Team members are required to notify the rest of the team as soon as an unforeseen circumstance presents itself. Team attendance will be kept via Microsoft Word document, documented members who did not attend.

Repeated absences will result in a team intervention to address the situation. If repeated absences continue after the team intervention, the team will contact Dr. McConomy and request a meeting with the team member to discuss the team's concerns and possible grade repercussions for the individual.

Use of a vacation day for team assignments must be agreed by all team members. The decision to use the vacation day must be made at least 48 hours prior to the due date.

AMENDING THE CODE OF CONDUCT

All team members agree that the code of conduct may be amended at any point during the project. In order to amend the document the majority of members must be in agreement of the change.

STATEMENT OF UNDERSTANDING

By signing below as a team member of Team 501: Landing System for Uncertain Terrain, I agree to follow this code of conduct during the period I am part of the team.

Appendix B: Functional Decomposition

Figure 1: Functional Decomposition Flow Chart

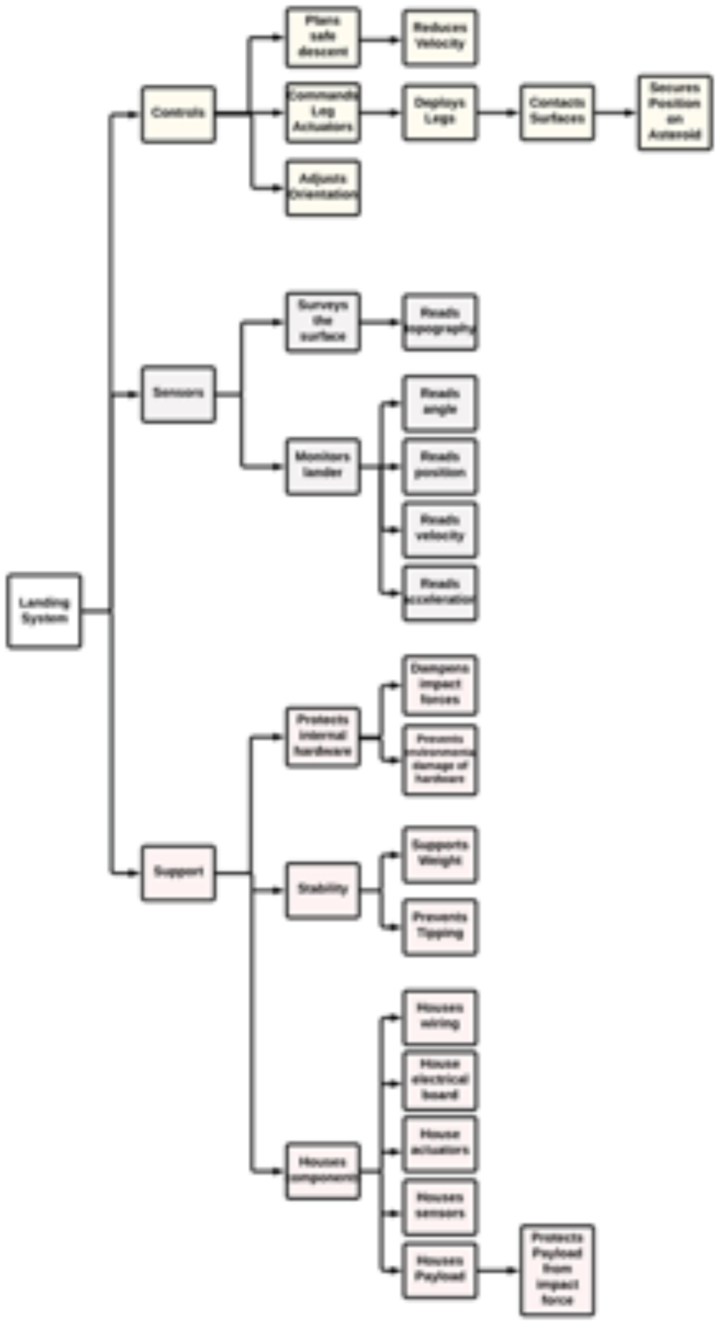
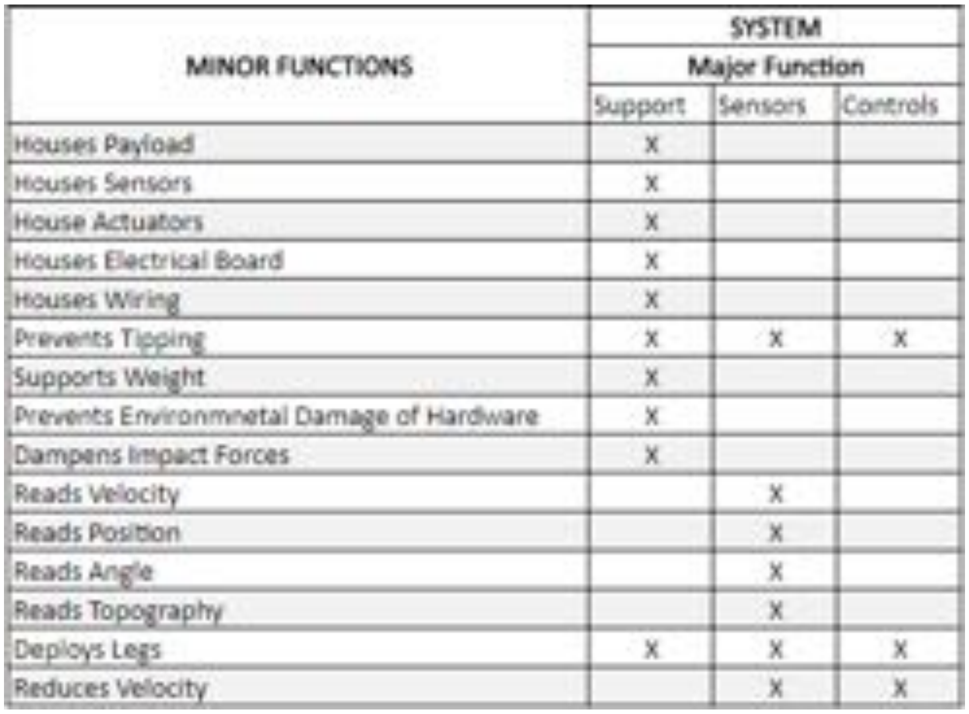


Figure 2: Functional Decomposition Matrix

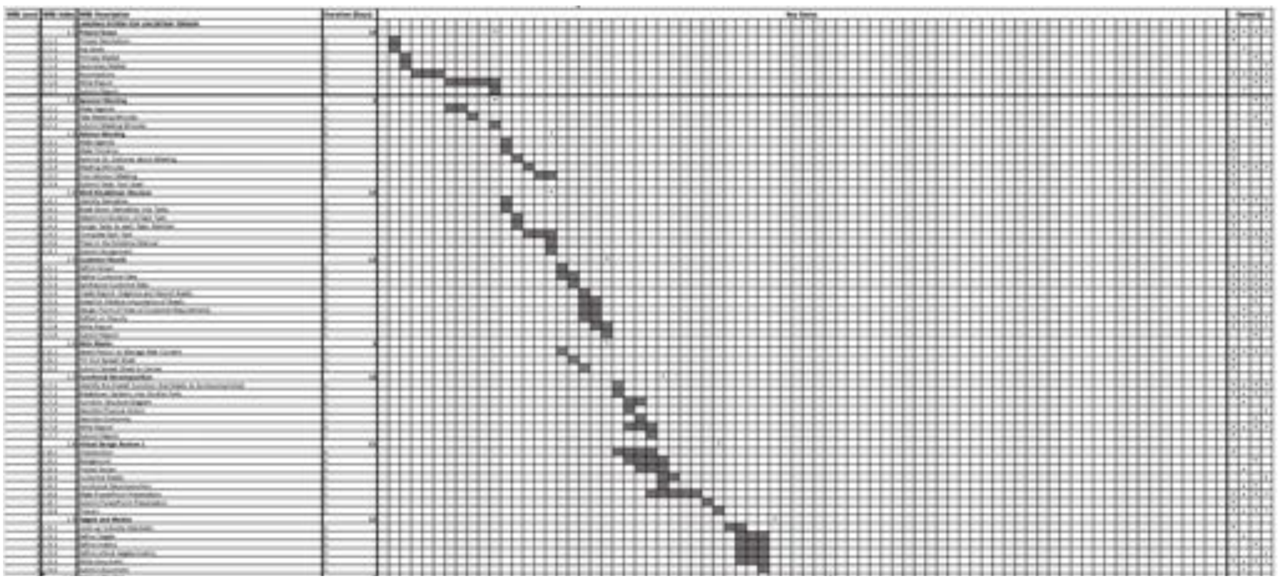


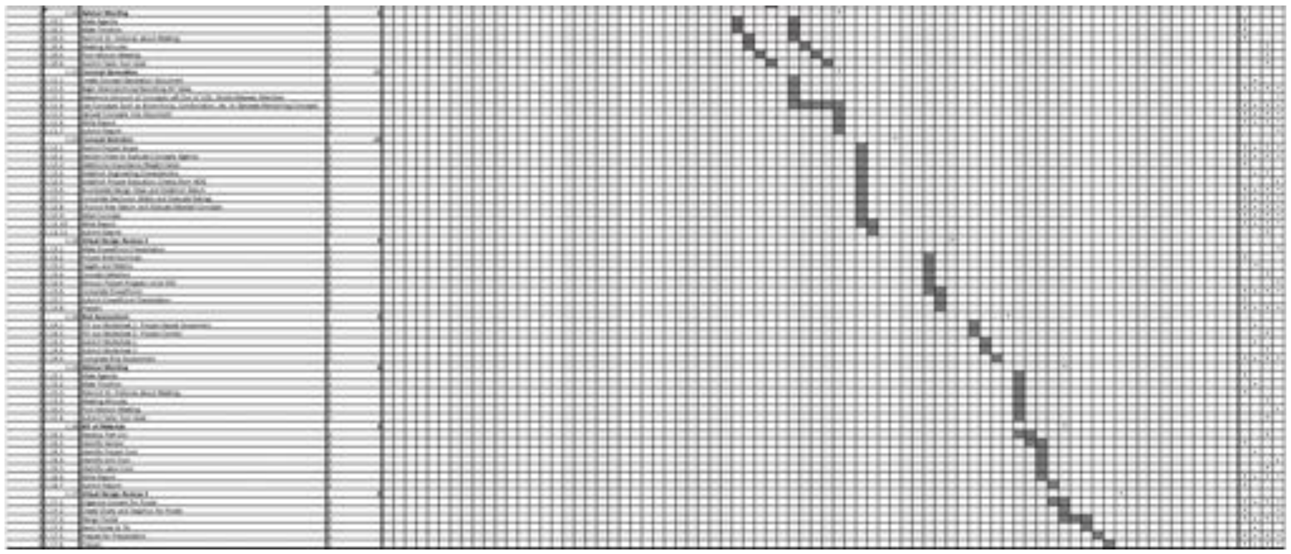
MINOR FUNCTIONS	SYSTEM		
	Major Function		
	Support	Sensors	Controls
Houses Payload	X		
Houses Sensors	X		
House Actuators	X		
Houses Electrical Board	X		
Houses Wiring	X		
Prevents Tipping	X	X	X
Supports Weight	X		
Prevents Environmnetal Damage of Hardware	X		
Dampens Impact Forces	X		
Reads Velocity		X	
Reads Position		X	
Reads Angle		X	
Reads Topography		X	
Deploys Legs	X	X	X
Reduces Velocity		X	X

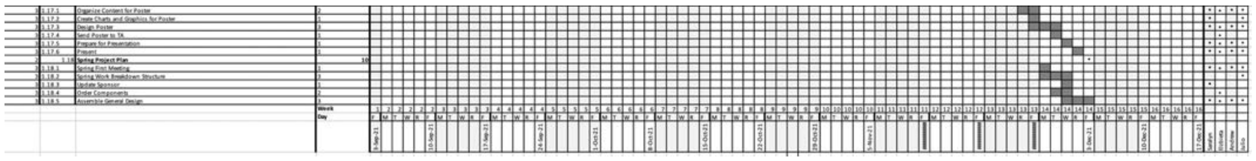
Appendix C: Target Catalog

Operation	What's Being Measured/Quantified	Target
Houses Payload	Dimensions of CHONKE (estimated)	0.65 m x 0.48 m x 0.3 m
Houses Sensors	Acceleration, velocity, angular position sensors length and volume	0.254-12.7 mm; 1.29-025.81 mm^2
	Technology used to detect objects and measure distance (volume)	0.929-2.787 m^2
	Force sensors (volume)	5.81-19.35 mm^2
Houses Electrical Board	Dimensions	101.52 x 53.3 mm
Houses Actuators	Volume of actuators	0.019 m^2
Houses Wiring	Diameter of each wire	1.29 mm
Prevents Tipping	Tipping angle to correct	10 degrees
Support Weight	Weight of spacecraft, payload, and other instruments	21.6 N
Prevents Enviromental Damage of Hardware	Cosmic dust and other particles size	0.1 mm
Dampens Impact Energy	Amount it needs to dissipate/dampen	2700 J
Impact Velocity	Maximum impact velocity	6 m/s
Reads Acceleration	Acceleration resolution to read	0.1 m/s^2 and within 10% of real value
Mass of Lander	Mass	150 kg
Reads Velocity	Velocity resolution to read	0.1 m/s and within 10% of real value
Reads Position	Position resolution to read	0.1 m and within 10% of real value
Reads Angle	Angle resolution to read	0.1 degrees and within 10% of final value
Reads Topography	Measure elevations within range	~0.2 m
Adjusts Orientation	Angle of lander relative to surface (in any direction)	20 degrees
Secures Position on Astroid	Change in position on all three axis	0 degrees
Reduces Velocity	Velocity of the lander as it touches down on surface	<3 m/s

Appendix D: Work Breakdown Structure







3	1.17.1	Organize Content for Poster	2
3	1.17.2	Create Charts and Graphics for Poster	1
3	1.17.3	Design Poster	3
3	1.17.4	Send Poster to TA	1
3	1.17.5	Prepare for Presentation	1
3	1.17.6	Present	1
2	1.18	**Spring Project Plan**	10
3	1.18.1	Spring First Meeting	1
3	1.18.2	Spring Work Breakdown Structure	3
3	1.18.3	Update Sponsor	1
3	1.18.4	Order Components	2
3	1.18.5	Assemble General Design	3
			Week
			Day

Appendix E: List of 100 Concepts

1. Airbag system
2. Legs with probes
3. Parachute
4. Legs that break fall and snap
5. Thrusters to levitate
6. Legs with terrain sensors
7. Legs controlled by gears and varied gear ratio
8. Collapsible legs
9. Balancing feet
10. Sensors on feet
11. Feet that conform to rocky terrain
12. Bouncing legs/feet
13. Deployable landing surface that feet will stick to
14. Legs that absorb shock and bounce (kangaroo)
15. Deployable wings to slow the lander
16. Gliding wings to smooth landing
17. Feet with pronged flexible “toes” or “fingers” to grab onto surface
18. Sticky feet
19. Long arms/legs extended outside of spacecraft at a bend
20. 8 legs with joints in the middle
21. Aluminum legs
22. Legs that absorb the shock from the feet and adjust accordingly
23. One leg lands before the other
24. Flexible feet moving independently from legs
25. Short legs, low to the ground
26. Webbed feet for gliding and grabbing
27. Different sets of feet that absorb shock/slow down system at a different rate
28. Magnetic legs
29. Large tread and area that conforms to terrain upon landing
30. Retrorockets
31. Single leg/foot Osairus Rex inspired
32. Pushrod System
33. Curved legs
34. Legs with protruding feet to get underneath the surface as it lands
35. Landing system with goat like hooves to be able to stand on thinner flat sections
36. Claw like feet to grab the rock as it lands
37. Use a large amount of small flexible tubing as landing system
38. Hover system with attached sensor to move accordingly
39. Pull Rod System
40. Crushable System
41. Magnetic Hook

42. Harpoon Hook
43. Bendable Legs
44. Cushioned Legs
45. Macpherson System
46. Air Suspension System
47. Crane like legs
48. Kangaroo like legs
49. Spider web feet
50. Hawk claws
51. Spider like legs
52. Grasshopper extending and collapsing landing legs
53. Aviation absorber system
54. Shock absorbing liquid cushion
55. Alligator/crocodile jaw
56. Gecko feet
57. Protracting claws
58. Flexible fibers system
59. Lowering winch device
60. Hydraulic legs
61. Hind legs
62. Solar powered reverse thrusters
63. Solar sail landing system
64. Solar powered drill to attach it to the asteroid
65. Solar powered gun to shoot the Asteroid
66. 3D printed legs
67. Retractable legs with grabbing feet
68. Lever that hits and extends out stabilizing landing legs
69. Non-Newtonian fluid cushion
70. Side extended landers; launch out sideways
71. Cushion to soften fall and crushable legs to take the rest of the shock
72. Round feet with bouncy material
73. Self-leveling extendable cylinders
74. Metal conformable teeth feet that lock in place
75. Screw on legs that drills into terrain
76. Dampened Base
77. One leg purely for energy absorption
78. Multiple legs for energy absorption
79. Rack and pinion-controlled legs
80. Ball jointed feet
81. One leg lands before the other + McPherson structure
82. Legs with several joints
83. Force toggling locking system
84. Footpad with fingers that are adjust to the topography

85. Double A arm suspension
86. Semi crushable link within leg
87. Triple support leg with center crushable shock absorber
88. Springboard base attachment
89. Multiple crushable legs
90. Arm with three branches and center branch as shock absorber
91. Feet landing with center palm and deployable ‘fingers’
92. Moving and locking chains for foot pads
93. Hexagon base with 3 stability legs and 3 damping legs
94. Y-shaped legs
95. Mechanically jointed legs with adjustable leg angles
96. Stability legs that touch down fist with center spring energy absorber
97. Triangle shaped legs with adjusting pistons
98. Bent legs that extend outside of the base
99. Large dampening leg with pin screen foot
100. Flexible pin screen feet with the ability to conform to the surface

Appendix F: Binary Pairwise Comparison

Binary Pairwise Matrix						
	The system is autonomous	Supports the spacecraft and associated components	Withstands or dissipates the potential energy from the fall and impact velocity	Adjusts to the hypothesized terrains of Psyche	The system does not have to be reusable	Total
The system is autonomous	-	0	0	0	1	1
Supports the spacecraft and associated components	1	-	0	1	1	2
Withstands or dissipates the potential energy from the fall and impact velocity	1	1	-	1	1	3
Adjusts to the hypothesized terrains of Psyche	1	0	0	-	1	1
The system does not have to be reusable	0	0	0	0	-	0

Appendix G: Pugh Charts

Table G1. Pugh Chart 2

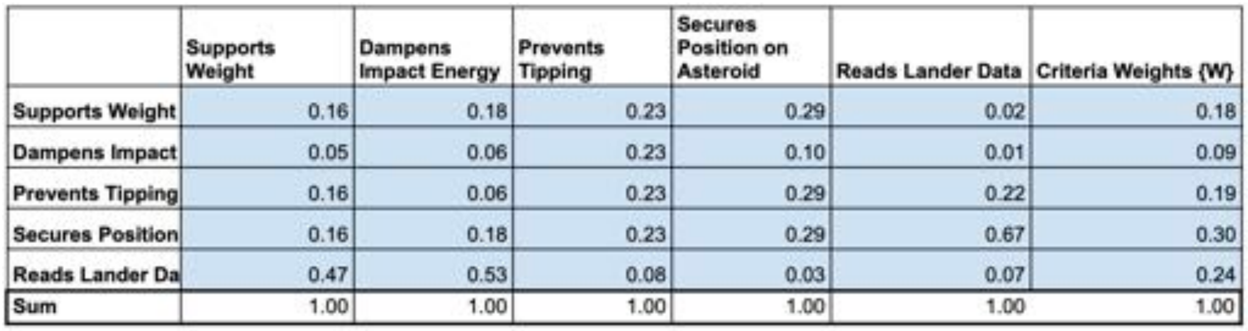
		Concepts						
Engineering Characteristics	Concept 3	1	2	4	5	6	7	8
Houses Components\Hardware	- DATUM -	S	S	S	S	S	S	S
Supports Weight		S	-	-	-	+	S	+
Reads Lander Data		S	S	S	S	S	S	S
Prevents Tipping		S	-	-	-	+	S	+
Dampens Impact Energy		+	+	+	S	S	-	S
Senses Sourrounding Topography		S	S	S	S	S	S	S
Adjusts Orientation		S	S	S	S	S	-	S
Secures Position on Asteroid		-	-	-	S	+	+	+
Total Pluses		1	1	1	0	3	1	3
Total Minuses		1	3	3	2	0	2	0

Table G2. Pugh Chart 3

Engineering Characteristics	Concept 1	2	4	6	7	8
Houses Components\Hardware	- DATUM -	S	S	S	S	S
Supports Weight		S	S	+	-	+
Reads Lander Data		S	S	S	S	S
Prevents Tipping		S	-	+	S	+
Dampens Impact Energy		S	S	-	-	-
Senses Sourrounding Topography		S	S	S	S	S
Adjusts Orientation		S	-	+	S	+
Secures Position on Asteroid		S	S	+	+	+
Total Pluses		0	0	4	1	4
Total Minuses		0	2	1	2	1

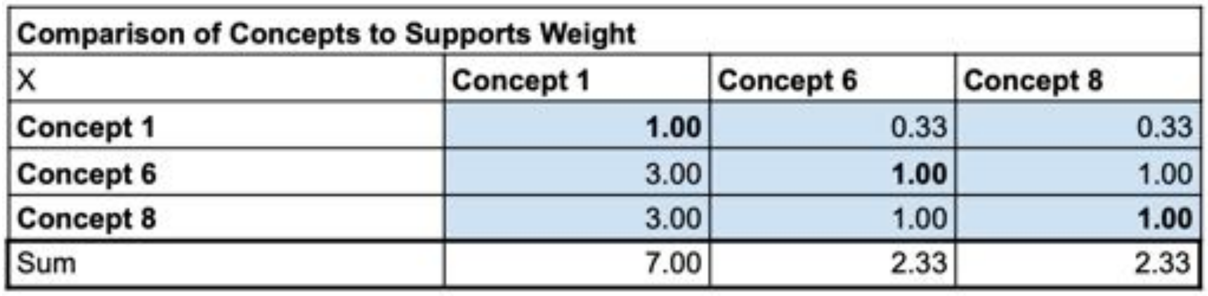
Appendix H: AHP Charts

Table H1. Normalized AHP

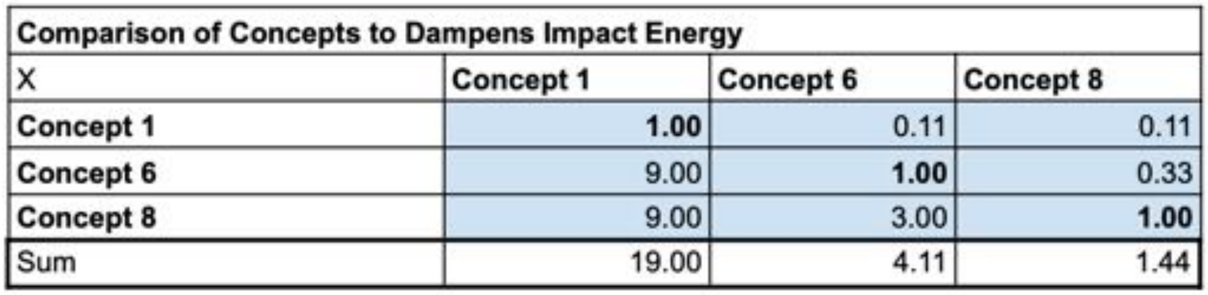


	Supports Weight	Dampens Impact Energy	Prevents Tipping	Secures Position on Asteroid	Reads Lander Data	Criteria Weights {W}
Supports Weight	0.16	0.18	0.23	0.29	0.02	0.18
Dampens Impact	0.05	0.06	0.23	0.10	0.01	0.09
Prevents Tipping	0.16	0.06	0.23	0.29	0.22	0.19
Secures Position	0.16	0.18	0.23	0.29	0.67	0.30
Reads Lander Da	0.47	0.53	0.08	0.03	0.07	0.24
Sum	1.00	1.00	1.00	1.00	1.00	1.00

Comparison Charts



Comparison of Concepts to Supports Weight			
X	Concept 1	Concept 6	Concept 8
Concept 1	**1.00**	0.33	0.33
Concept 6	3.00	**1.00**	1.00
Concept 8	3.00	1.00	**1.00**
Sum	7.00	2.33	2.33



Comparison of Concepts to Dampens Impact Energy			
X	Concept 1	Concept 6	Concept 8
Concept 1	**1.00**	0.11	0.11
Concept 6	9.00	**1.00**	0.33
Concept 8	9.00	3.00	**1.00**
Sum	19.00	4.11	1.44

Comparison of Concepts to Prevents Tipping			
X	**Concept 1**	**Concept 6**	**Concept 8**
Concept 1	**1.00**	0.33	0.33
Concept 6	3.00	**1.00**	1.00
Concept 8	3.00	1.00	**1.00**
Sum	7.00	2.33	2.33

Comparison of Concepts to Secures Position on Asteroid			
X	**Concept 1**	**Concept 6**	**Concept 8**
Concept 1	**1.00**	0.33	0.33
Concept 6	3.00	**1.00**	1.00
Concept 8	3.00	1.00	**1.00**
Sum	7.00	2.33	2.33

Comparison of Concepts to Reads Lander Data			
X	**Concept 1**	**Concept 6**	**Concept 8**
Concept 1	**1.00**	1.00	1.00
Concept 6	1.00	**1.00**	1.00
Concept 8	1.00	1.00	**1.00**
Sum	3.00	3.00	3.00

Normalized Comparison Charts

Normalized Comparison of Concepts to Supports Weight				
X	Concept 1	Concept 6	Concept 8	Design Alternative Priorities
Concept 1	**0.14**	0.14	0.14	0.14
Concept 6	0.43	**0.43**	0.43	0.43
Concept 8	0.43	0.43	**0.43**	0.43
Sum	1.00	1.00	1.00	1.00

Normalized Comparison of Concepts to Dampens Impact Energy				
X	Concept 1	Concept 6	Concept 8	Design Alternative Priorities
Concept 1	**0.05**	0.03	0.08	0.05
Concept 6	0.47	**0.24**	0.23	0.32
Concept 8	0.47	0.73	**0.69**	0.63
Sum	1.00	1.00	1.00	1.00

Normalized Comparison of Concepts to Prevents Tipping				
X	Concept 1	Concept 6	Concept 8	Design Alternative Priorities
Concept 1	**0.14**	0.14	0.14	0.14
Concept 6	0.43	**0.43**	0.43	0.43
Concept 8	0.43	0.43	**0.43**	0.43
Sum	1.00	1.00	1.00	1.00

Normalized Comparison of Concepts to Secures Position on Asteroid				
X	Concept 1	Concept 6	Concept 8	Design Alternative Priorities
Concept 1	**0.14**	0.14	0.14	0.14
Concept 6	0.43	**0.43**	0.43	0.43
Concept 8	0.43	0.43	**0.43**	0.43
Sum	1.00	1.00	1.00	1.00

Normalized Comparison of Concepts to Reads Lander Data				
X	Concept 1	Concept 6	Concept 8	Design Alternative Priorities
Concept 1	**0.33**	0.33	0.33	0.33
Concept 6	0.33	**0.33**	0.33	0.33
Concept 8	0.33	0.33	**0.33**	0.33
Sum	1.00	1.00	1.00	1.00

Appendix I: Operation Manual

TEAM 501: LANDING SYSTEM FOR UNCERTAIN TERRAIN

Syllabus

Senior Design Team 501:

Saralyn Jenkins | Mechanical Systems Engineer

Elzbieta Krekora | Materials Engineer

Andrew Sak | Controls Engineer

FAMU-FSU College of Engineering | 2525 Pottsdamer St. Tallahassee, FL. 32310

List of Figures

I. Overview

This device is a possible solution to landing on asteroid 16 Psyche with uncertain terrain, hypothesized to range from rocky to metallic, if a future landing mission were to be proposed. Our design addresses concerns of damping impact force and adapting to an uneven surface. The lander has two main areas, mechanical design and mechatronic components, the mechanical design breaks down further into three subsystems- the suspension, legs, and feet. The lander is tested using a structure to support the landing system and base attachment over a terrain comparable to Psyche's. The structure aids in lifting and controlling the lander's descent using a pulley and counterweight.

II. Components

The landing system designed for this project is attached to 3 walls of a base modeled by a hexagonal box. The model shown is intended for earth testing and does not include the components required for a space mission but are replaced by comparable parts. The fully assembled CREO model is shown below in figure 1. All individual parts of the landing system prototype are in Appendix A.

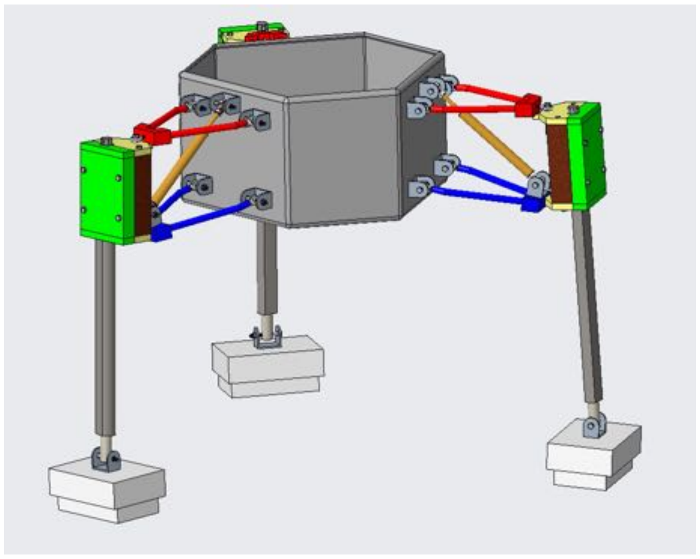


Figure 1 : Fully Assembled Landing System Prototype

2.1 Suspension

The suspension used in this design is modeled after a suspension commonly used in cars, double A-arms. The A-Arms are made from steel rods. Both the upper A-arm and the lower A-arm are shown in Figure 2.

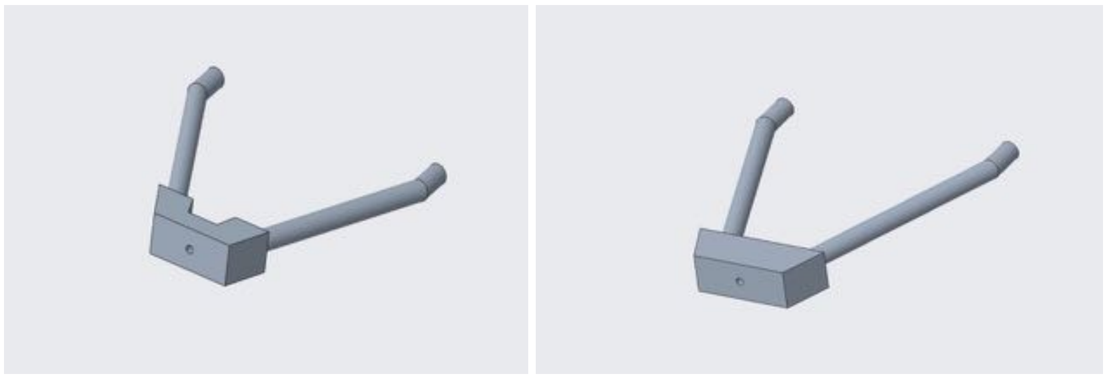


Figure 2 : (A) Upper A-Arm (B) Lower A-Arm

A gas spring is attached from the base to the bottom A-arm to aid in damping the impact force and will be changed to a honeycomb piston like assembly for space use. The gas spring damper used in this prototype is a 30-pound damper that is 12 inches long. The damper is secured to the bottom A-arm and base by U-joints. The gas spring damper is shown in figure 3 and the U-joint is shown in figure 4.

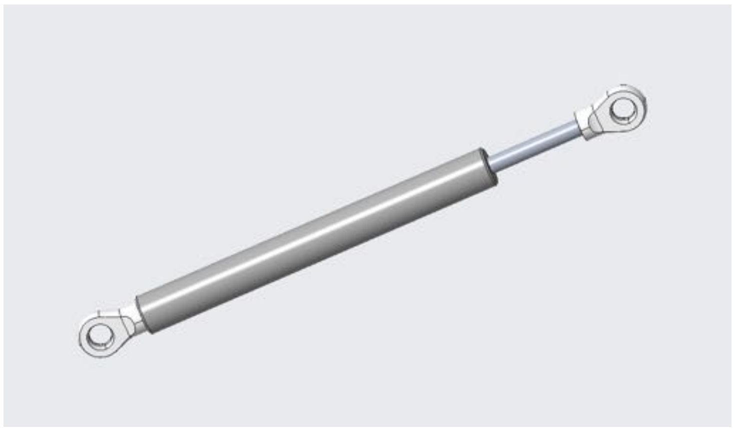


Figure 3: Gas Spring Damper

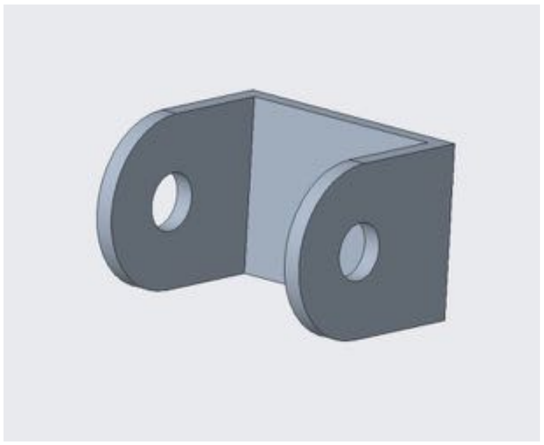


Figure 4: U-Bracket

The A-arms are secured on to the base and attached to the knuckle using Heim joints. The Heim joints are constructed by using M6x1 spherical rod end ball joints on each of the connecting pieces. The Heim joints are connected to the base using the same U-bracket shown in figure 4 above. The spherical rod end ball joint used is shown below in figure 5.

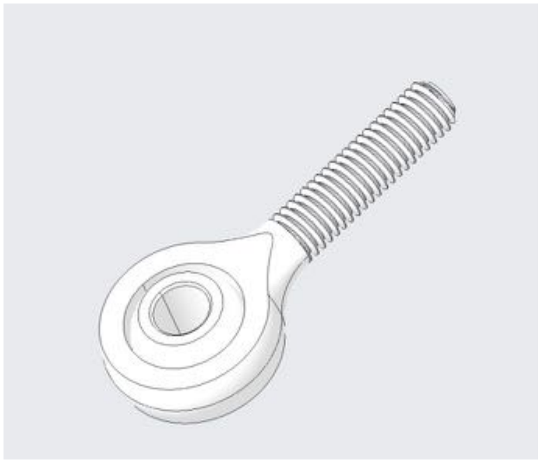


Figure 5: Spherical Rod End Ball Joint

2.2 Leg and Knuckle

Linear actuators are used as adjustable legs on this project. The prototype uses three linear actuators with 40.64 cm of adjustable length. It is controlled by 12V of power. It extends or retracts when voltage is supplied at a rate of 10 mm/s, and it locks in place when power is shut off. This allows the motor to be able to be shut off once the prototype has landed, hence the motor does not have to be running the whole time. The linear actuator used is shown below in figure 6.

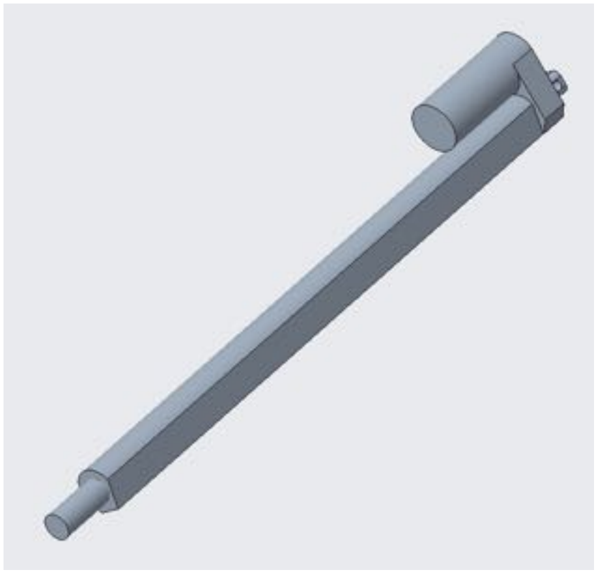


Figure 6: Linear Actuator

The knuckle used is split into four parts and secured around the linear actuator. The inside of the knuckle is a negative mold of the top of the linear actuator. There are two main clamp pieces that grip most of the linear actuator: the knuckle mid-plate (figure 7) and the knuckle cap (figure 8).

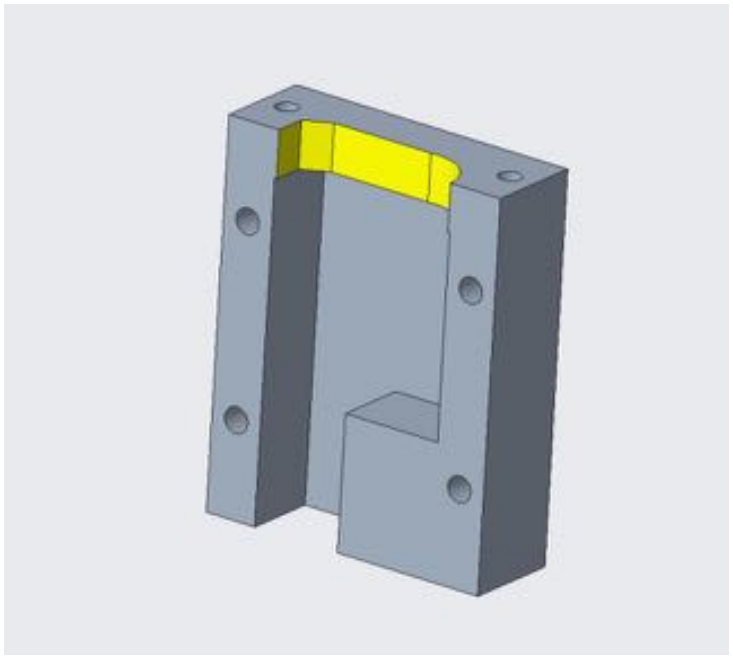


Figure 7: Knuckle Mid-Plate

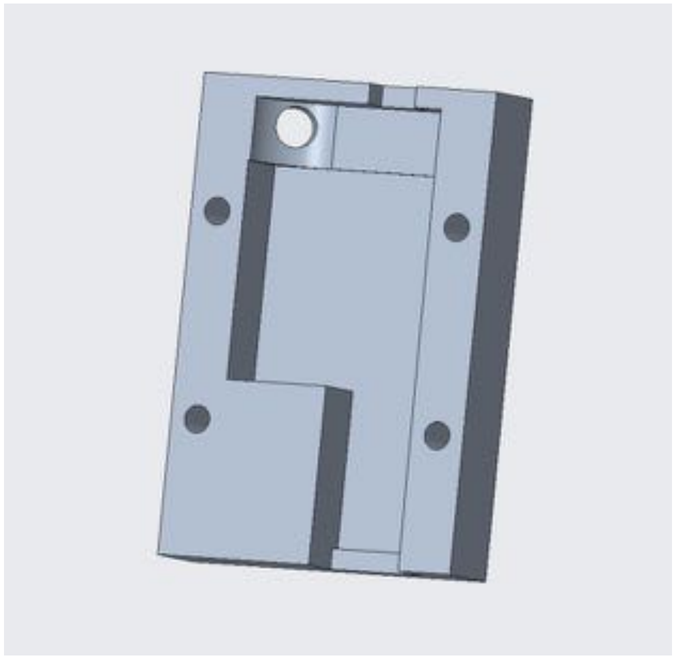


Figure 8: Knuckle Cap

There are two plates on the top and bottom of the knuckle to connect the two larger knuckle pieces shown above. All four pieces of the knuckle are composed of 6061 aluminum. The top plate is shown in figure 9 and the bottom plate is shown in figure 10.

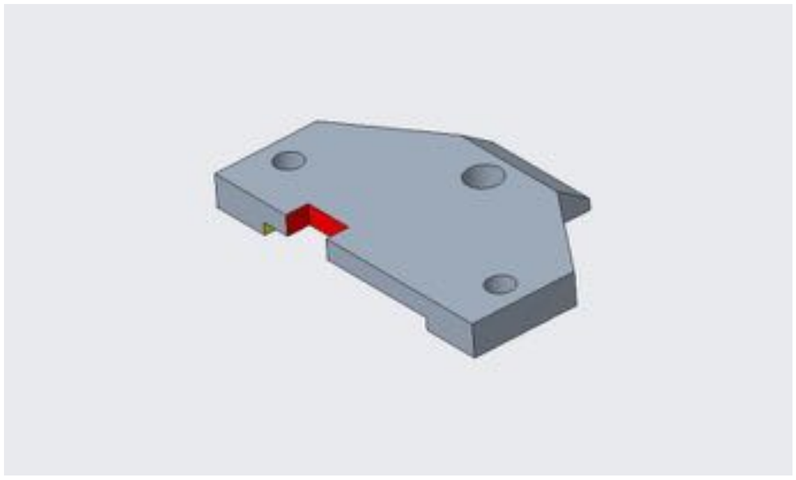


Figure 9: Top Plate

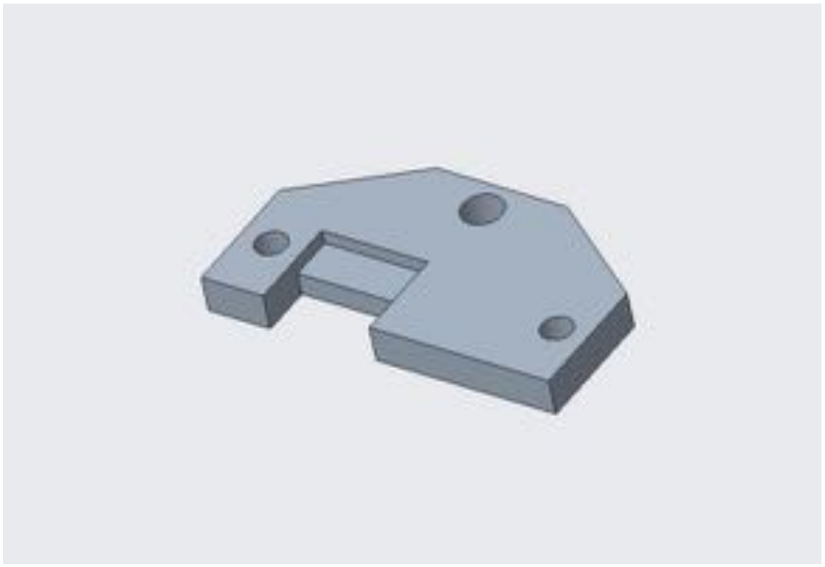


Figure 10: Bottom Plate

2.3 Feet

The feet used in this design are repurposed pin screens, commonly used as toys, chosen for their ability to for to the shape of object placed against it, allowing the lander to 'grip' on to Psyche's surface. The plastic parts of the pin screen are replaced with diamond plate steel to reinforce the component. A U-bracket is welded on to the foot and connected to another U-bracket to allow for rotation in two directions to adjust to any incline on the terrain. These two U-brackets make a U-joint. The rotation is limited to the point that the foot will not fold underneath itself. The pin screen foot and the U-joint are shown in figure 11 and figure 12.

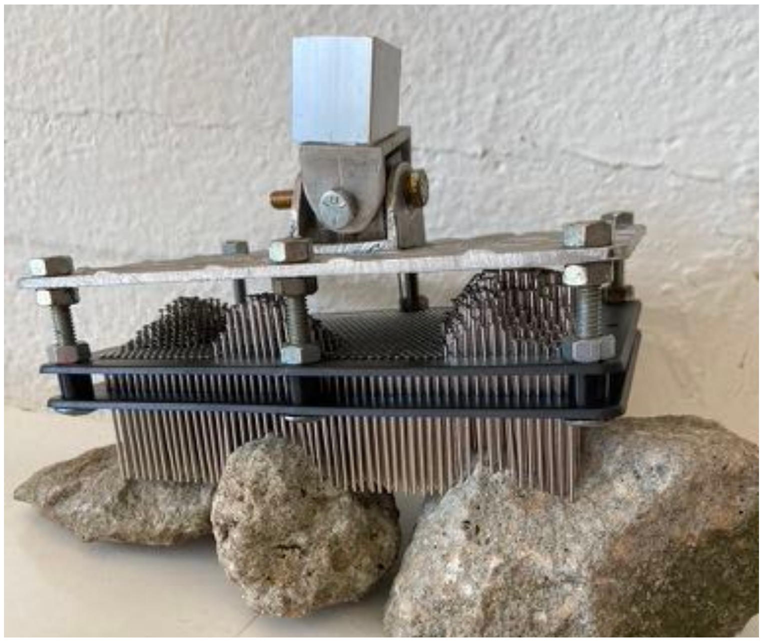


Figure 11: Reinforced Pin Screen Foot

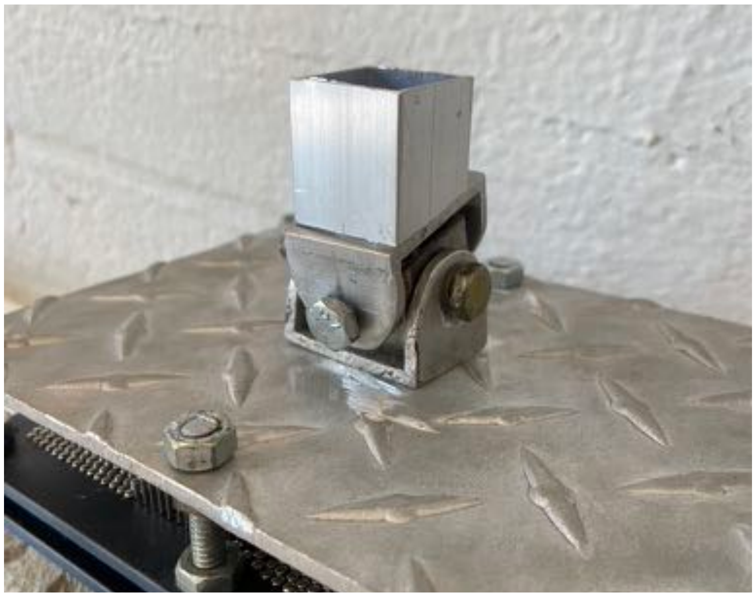


Figure 12: U-Joint on Foot

2.4Mechatronics

All mechatronic components communicate using an Arduino Mega 2560 and are powered by a 12 V power supply. These components are needed to read the distance between each leg on the lander and the surface directly below it, adjust the linear actuators, and read the orientation offset from gravity.

To do the necessary things listed above the following components were used: Arduino Mega 2560, TOF Sense Laser Range Sensor, BNO055 Inertial Measurement Unit, Tower Pro SG92R Micro servo, and a Windynation 12V Linear Actuator.

Laser Sensors

The laser sensors used on this project were TOF Sense Laser Range 5M which can read distances spanning from 1cm to 5m. Three sensors are used in cascading form to communicate with each other and reduce the number of COM ports used on the Arduino. To read in the data from the sensors, a USB to TLL device was used to match the rate of information going into rate of information collected by the computer.

Setting Up Laser Sensors

To setup the hardware of the TOF Laser Range Sensor the TOFAssisstant software is needed to be downloaded at https://www.waveshare.com/wiki/TOF_Laser_Range_Sensor. Click the “Resources” tab and download the software on a Windows PC.

To communicate between the sensor hardware and the TOFAssistant software a USB to TTL (transistor-transistor logic) device is needed to connect the computer to the sensor. Figure 13 shows what each wire corresponds to. Receive and transmit on the sensor means that sensor receives information from the blue wire and transmits information to the green wire. When connecting to the USB to TTL device, the receive wire from the sensors connects to the transmit pin of the device. This is because the device will transmit through that pin and the sensor receiving wire will take in the information. The connection from the laser sensor to the USB to TTL device is seen in figure 14.

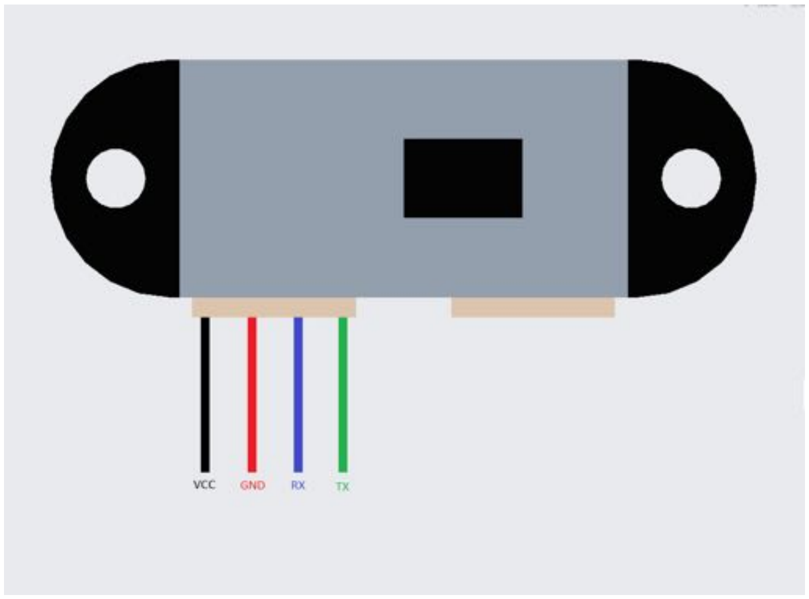


Figure 13: Wire Set-Up for the TOF Laser Range Sensor- VCC is Power, GND is Ground, RX is Serial Receive, and TX is Serial Transmit

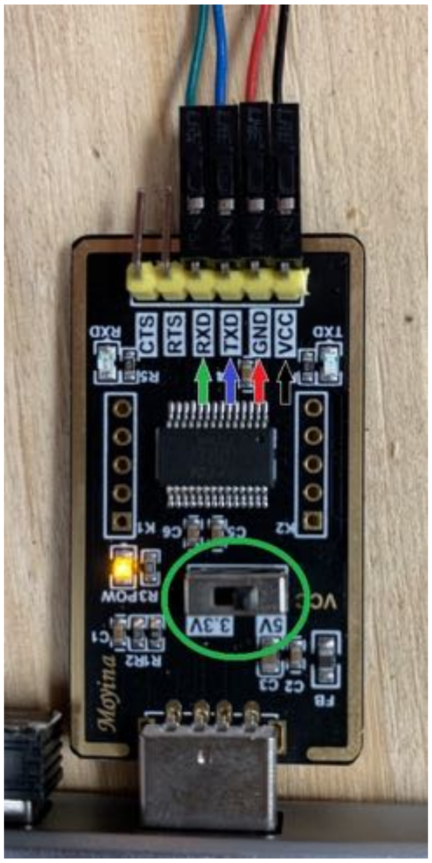
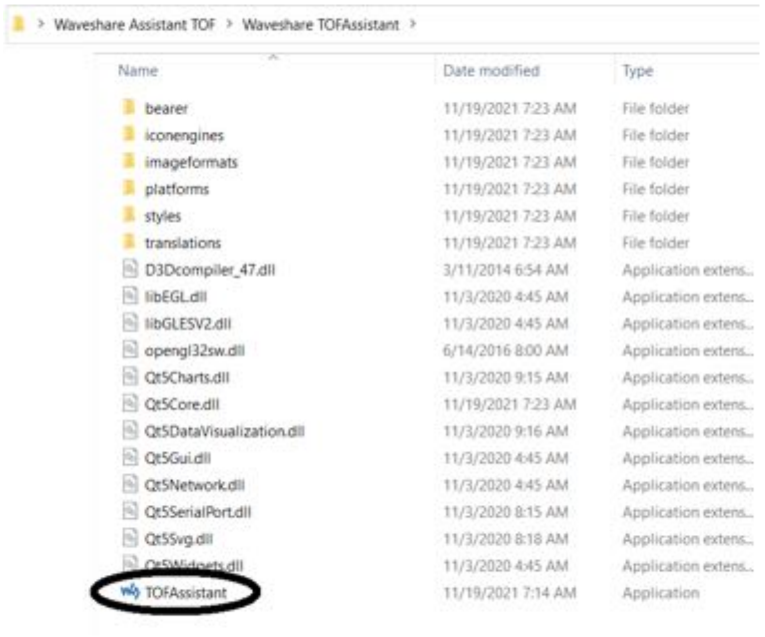


Figure 14: USB to TTL Connection - Green Circle Indicates Power Switch to 5V. Green, Blue, Red, and Black Arrow Show Where to Connect the Transmit, Receive, Ground, and Power from the Sensor to the Device

After downloading the TOFAssistant files and connecting the laser sensor to the USB to TTL device, access the "Waveshare TOFAssistant" folder at the location of download. Click on the "TOFAssistant" application as seen in figure 15. An application will launch as seen in figure 16.



> Waveshare Assistant TOF > Waveshare TOFAssistant >

Name	Date modified	Type
bearer	11/19/2021 7:23 AM	File folder
iconengines	11/19/2021 7:23 AM	File folder
imageformats	11/19/2021 7:23 AM	File folder
platforms	11/19/2021 7:23 AM	File folder
styles	11/19/2021 7:23 AM	File folder
translations	11/19/2021 7:23 AM	File folder
D3Dcompiler_47.dll	3/11/2014 6:54 AM	Application extens...
libEGL.dll	11/3/2020 4:45 AM	Application extens...
libGLESV2.dll	11/3/2020 4:45 AM	Application extens...
opengl32sw.dll	6/14/2016 8:00 AM	Application extens...
Qt5Charts.dll	11/3/2020 9:15 AM	Application extens...
Qt5Core.dll	11/19/2021 7:23 AM	Application extens...
Qt5DataVisualization.dll	11/3/2020 9:16 AM	Application extens...
Qt5Gui.dll	11/3/2020 4:45 AM	Application extens...
Qt5Network.dll	11/3/2020 4:45 AM	Application extens...
Qt5SerialPort.dll	11/3/2020 8:15 AM	Application extens...
Qt5Svg.dll	11/3/2020 8:18 AM	Application extens...
Qt5Widgets.dll	11/3/2020 4:45 AM	Application extens...
TOFAssistant	11/19/2021 7:14 AM	Application

Figure 15: TOFAssistant Application Software File

Figure 16: TOFAssistant Software Launching

Through this software we can change parameters on the sensor hardware and how the microcontroller will communicate with the hardware. The first thing we change is the baud rate. The max baud rate the Arduino mega 2560 microcontroller can handle is 115200 and the lowest output the laser sensor can do is 115200. In the top left of the software there is a drop-down menu to change the baud rate as seen in figure 17. Next, the correct computer communication port needs to be chosen in the TOFAssistant software in the drop-down menu as seen in figure 18. The communication port used can be viewed in the "device manager" application on most windows machines. Once the correct communication port is selected, communication is now ready to occur between the software and the hardware. Click on the icon with two disconnected ports to start communication as seen in figure 19. If the hardware is brand new, the software will start to output data as seen in figure 20. The sensors settings need to be changed for the purpose of this project and the settings menu can be reached by clicking the three circles and gear icon as seen in figure 21. Once the settings menu is open, put in the desired sensor id number, a baud rate of 115200, UART mode, INQUIRE mode, LONG range, and any desired FOV as seen in figure 22. If the sensor ID is set to be 0, then using the eight-byte hex code from figure 23, click on the serial port icon next to the COM drop down menu to open the serial port assistant as seen in figure 24. The eight-byte code can be placed in the black box in figure 24 and a transmitted 16-byte code will appear in the red box in figure 24. If a 16-byte code appears, then the sensor is working correctly. Now this process needs to be done to all three sensors and then they will be ready to wire to the microcontroller. The wiring for the sensors to the microcontroller is seen in figure 25.

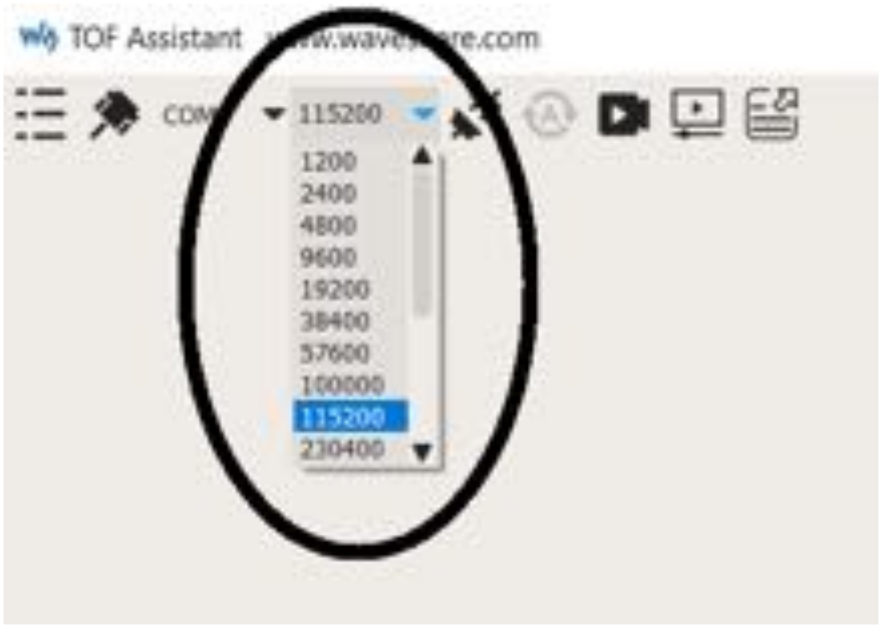


Figure 17: Changing the Baud Rate on the TOFAssistant Software

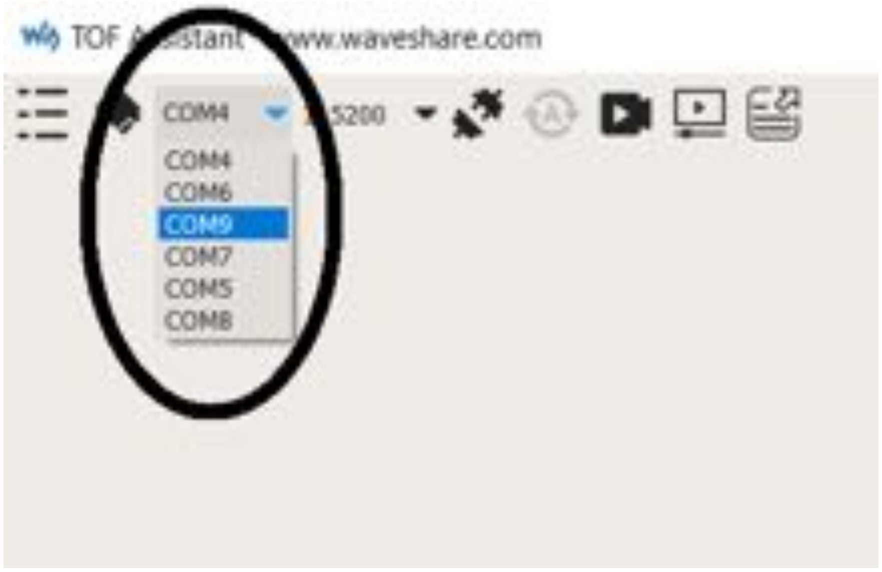
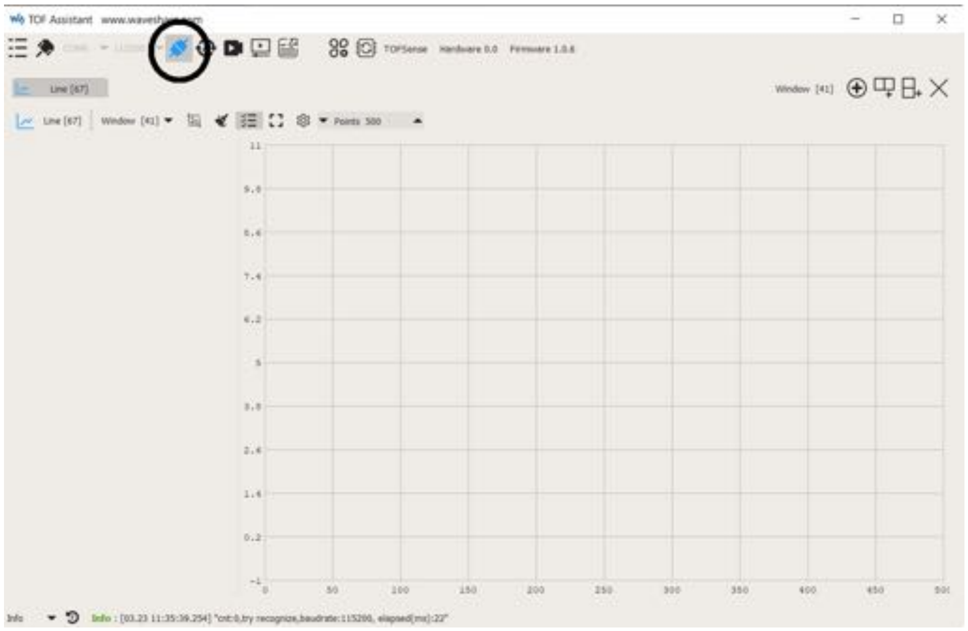
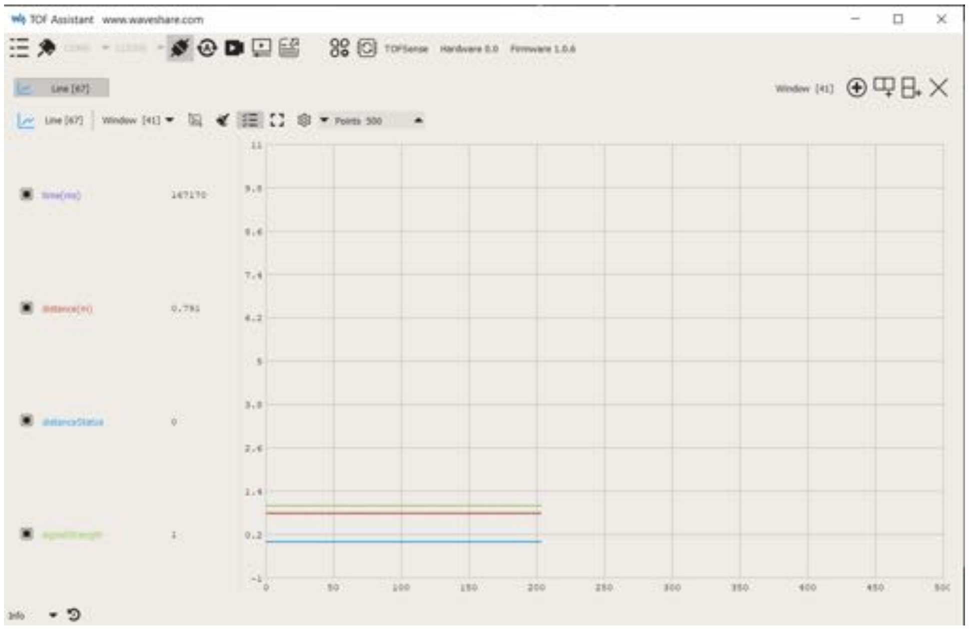


Figure 18: Change the COM Port on the TOFAssistant Software



```
Info : [03.23 11:35:39.254] "cnt:0,try recognize,baudrate:115200, elapsed(ms):22"
```

Figure 19: Connecting Communication Between the Software and the Laser Hardware



```
time(ms)          147170
distance(m)       0.791
distanceStatus    0
signalStrength    1
```

Figure 20: Data Being Output from the Sensors

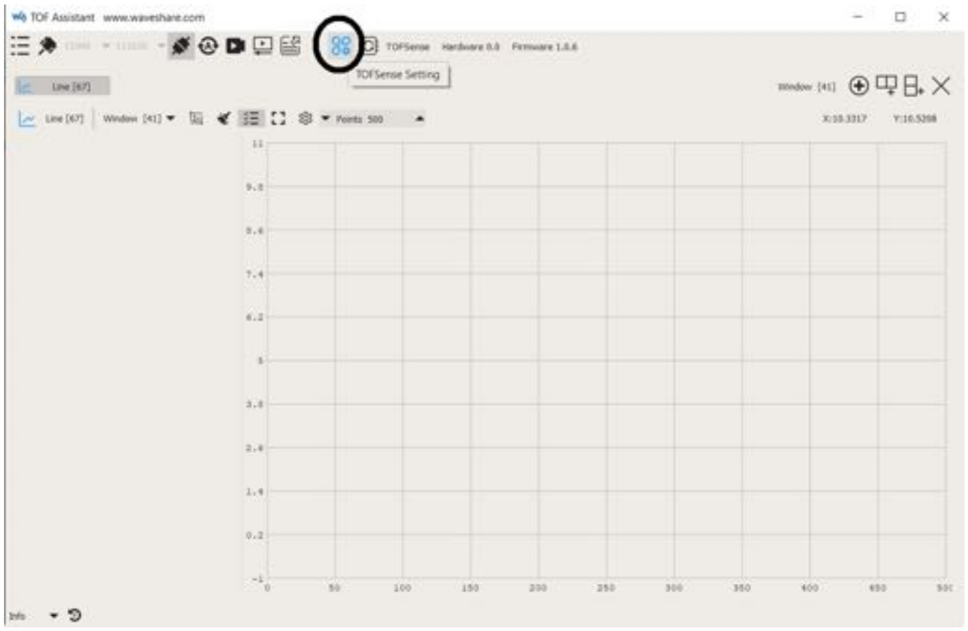
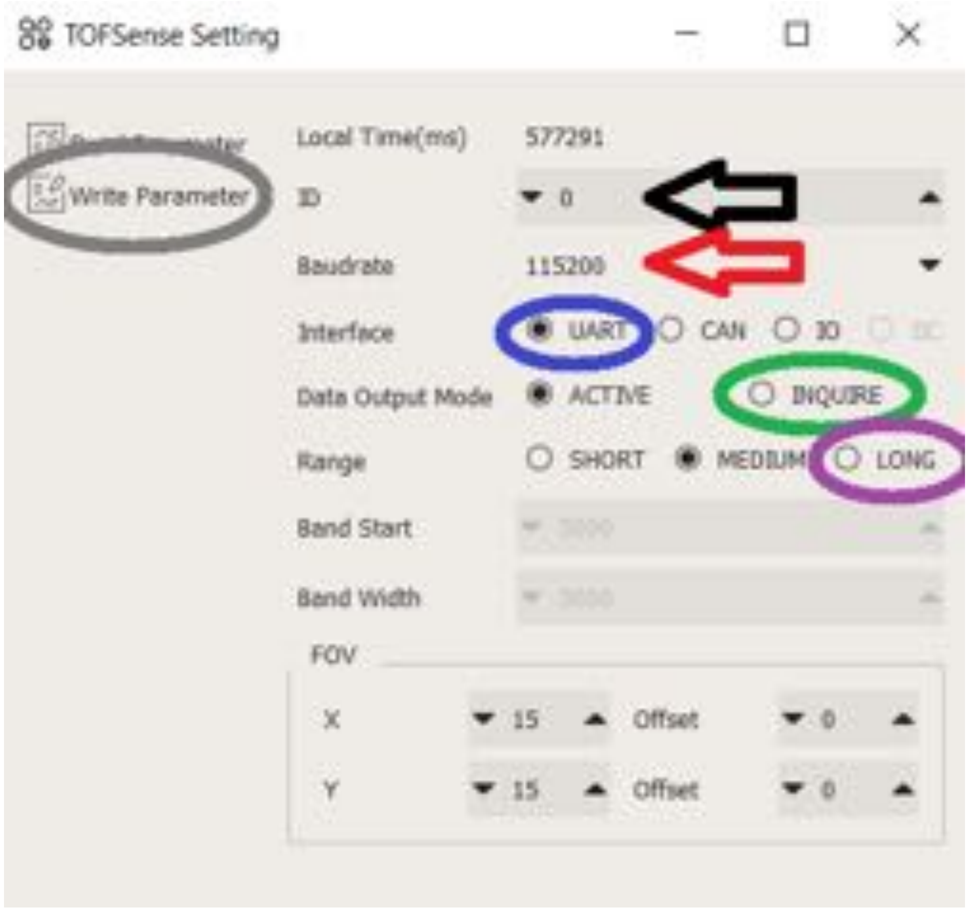

TOF Assistant www.waveshare.com
TOFSense Hardware 0.0 Firmware 1.0.6
TOFSense Setting
Line [67]
Window [41]
Points 500

Figure 21: Accessing the Settings Menu on the TOFAssistant Software



Local Time(ms) 577291

ID 0

Baudrate 115200

Interface [✓ UART] CAN IO IIC

Data Output Mode [✓ ACTIVE] INQUIRE

Range SHORT [✓ MEDIUM] LONG

Band Start 3000

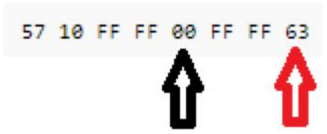
Band Width 3000

FOV

X 15 Offset 0

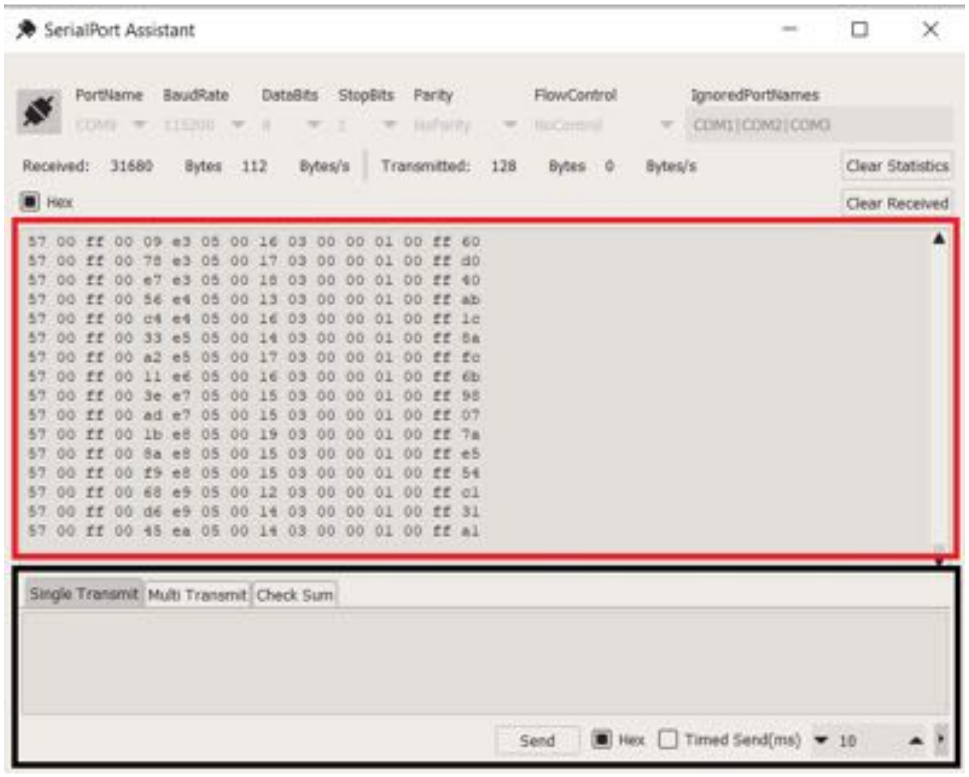
Y 15 Offset 0

Figure 22: The Settings Circled are the Ones Needed- After All the Settings Are Put in, the Write Parameter Button is Clicked



```
57 10 FF FF 00 FF FF 63
```

Figure 23: Eight-Byte Hex Code to Query Information - Black Arrow Indicates the ID - Red Arrow Indicates the Sum of 0x57, 0x10, and the ID



```
COM1|COM2|COM3
Received:  31680  Bytes  112  Bytes/s | Transmitted:  128  Bytes  0  Bytes/s
Hex
57 00 ff 00 09 e3 05 00 16 03 00 00 01 00 ff 60
57 00 ff 00 78 e3 05 00 17 03 00 00 01 00 ff d0
57 00 ff 00 e7 e3 05 00 18 03 00 00 01 00 ff 40
57 00 ff 00 56 e4 05 00 13 03 00 00 01 00 ff ab
57 00 ff 00 c4 e4 05 00 16 03 00 00 01 00 ff 1c
57 00 ff 00 33 e5 05 00 14 03 00 00 01 00 ff 8a
57 00 ff 00 a2 e5 05 00 17 03 00 00 01 00 ff fc
57 00 ff 00 11 e6 05 00 16 03 00 00 01 00 ff 6b
57 00 ff 00 3e e7 05 00 15 03 00 00 01 00 ff 98
57 00 ff 00 ad e7 05 00 15 03 00 00 01 00 ff 07
57 00 ff 00 1b e8 05 00 19 03 00 00 01 00 ff 7a
57 00 ff 00 8a e8 05 00 15 03 00 00 01 00 ff e5
57 00 ff 00 f9 e8 05 00 15 03 00 00 01 00 ff 54
57 00 ff 00 68 e9 05 00 12 03 00 00 01 00 ff c1
57 00 ff 00 d6 e9 05 00 14 03 00 00 01 00 ff 31
57 00 ff 00 45 ea 05 00 14 03 00 00 01 00 ff a1
```

Figure 24: Serial Port Assistant to Check Sensor Query

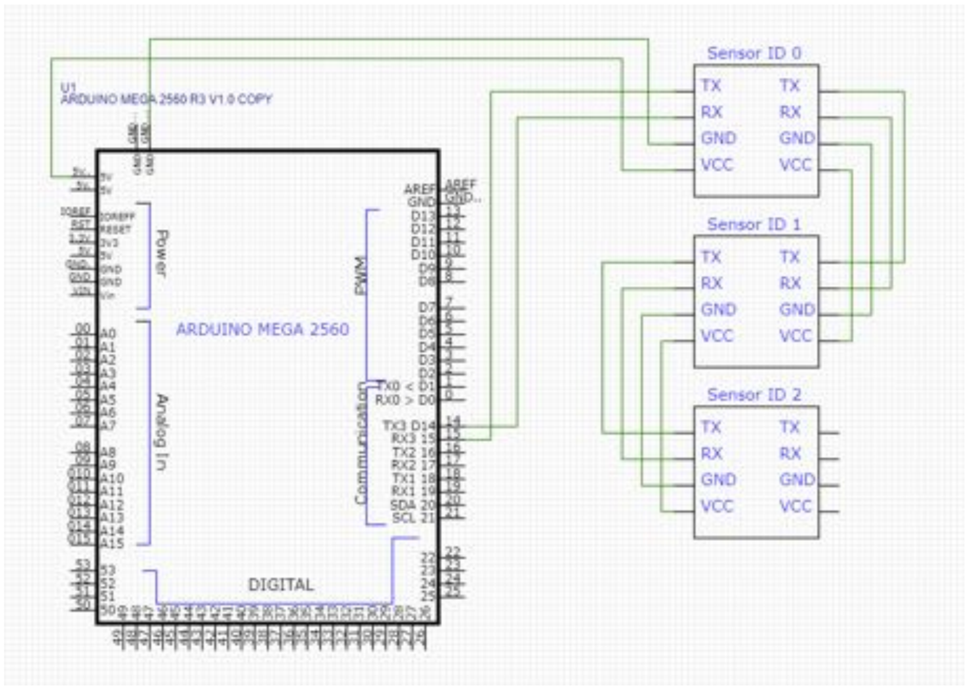


Figure 25: Connection Diagram for the Laser Sensors to the Microcontroller

Internal Measurement Unit

The Internal Measurement Unit (IMU) used is the Adafruit 9-axis accelerometer. The device is used to determine the Euler angles of orientation based on the force of gravity. Our project is concerned with rotation about the X and Y axis. Figure 26 shows the connection diagram for the IMU to the Arduino microcontroller.

For the BNO055 IMU to work in the Arduino program, the Adafruit_Sensor library and Adafruit_BNO055 library need to be downloaded onto the computer doing the coding. The library can be found through the Arduino programing by going to “Sketch > Include Library > Manage Libraries …" seen at the top of the Arduino program.

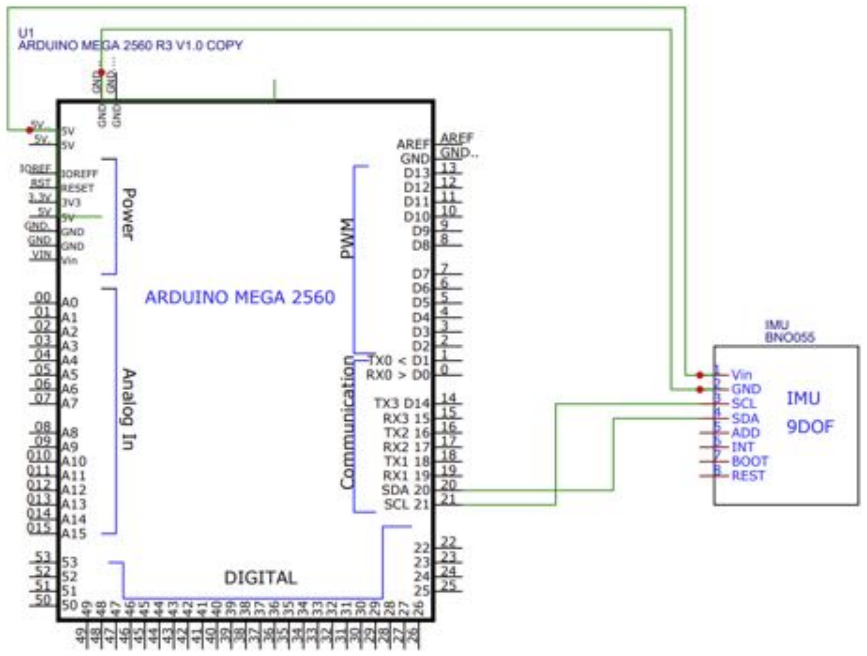


Figure 26: Connection Diagram for the IMU

Linear Actuators

12 Volt 40.64 cm linear actuators are used in this assembly, with stroke rate of 10 mm/sec. Linear encoders are not used in this project and are replaced by measuring the distance the linear actuators moved with laser distance sensors. A servo motor that is attached to the moving part of the actuator is activated to extend a flap that will obstruct the laser distance sensor vision to the terrain below and now the distance sensor will read how far the flap has moved to know how far the linear actuator has been extended. The connection diagram for the linear actuators to the motor drivers and microcontroller is shown in figure 27.

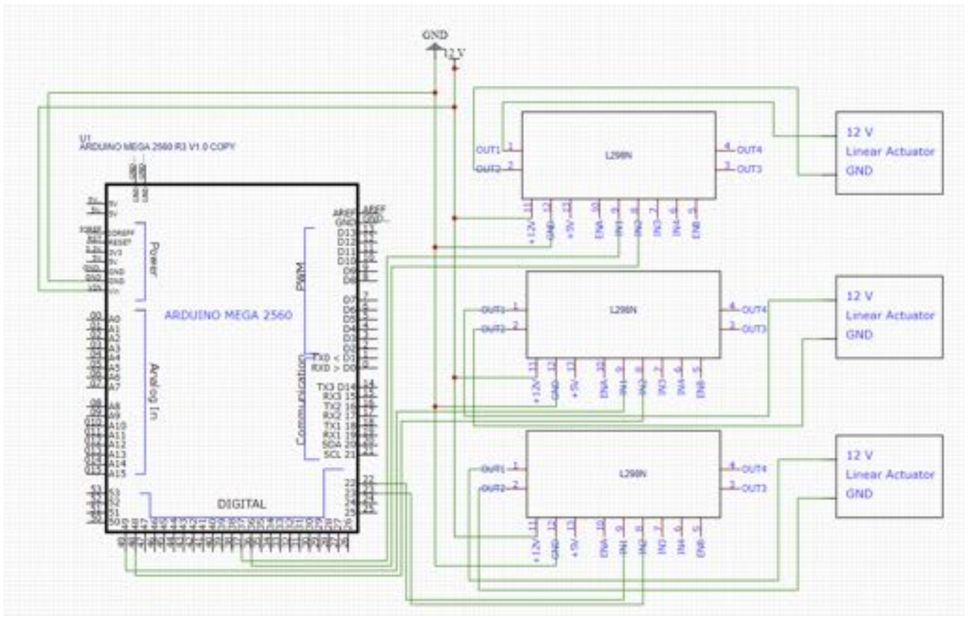


Figure 27: Connection Diagram for the Linear Actuators to Motor Drivers to Microcontroller

Servo Motors

Servo motors are used to extend a flap that will intersect the laser's view to change the reading from the ground to the fixed position of the 3D printed flap. The servo motor is connected to the end of the moving part on the linear actuator and will rotate the flap 90° to obstruct the sensors view of the terrain to be used as an encoder for the linear actuator.

Controls

The Arduino control program for the entire lander can be downloaded from team 501's senior design website. The program will go through the following stages. First, a reset button will be clicked to retract the legs and then extend them back out to 8 inches. During this stage, the first LED light will blink and a LCD screen will display "Reset in progress…". After, there is a

10 second delay to allow the users to ensure the lander is in the correct place in the test rig. During this stage, all LED lights will blink and a LCD screen will display “Next Stage in: ” while counting down the time.

Next, the lander takes 10 seconds to read the terrain below and make the adjustment length calculations. During this stage, the second LED light will blink and a LCD screen will display “Acquiring Data…”. After the distance data is acquired, the servo motors will activate and extend a flat piece that will be used for linear actuator encoding with the laser distance sensors. The linear actuators will now start moving according to the calculations done. During this stage, the third LED light will blink and a LCD screen will display “Adjusting Legs…”.

Once the legs are done adjusting, a delay of 20 second starts and the lander is ready to be dropped. During this stage, all LED lights will blink and a LCD screen will display “Drop Ready”. The users will now have to let the lander drop in the test rig. After 20 seconds, the lander will start to sense the orientation through the IMU. During this stage, the fourth LED light will blink and a LCD screen will display “Sensing Orientation…”. After the program is done sensing the orientation, the legs will start to adjust to the calculated lengths. During this stage, all LED lights will blink and a LCD screen will display “Adjusting Legs”. Once done with adjusting legs, all LED lights will stay on and the LCD screen will display the final orientation as well as the touchdown velocity.

III. Integration

The hexagonal base holds the electronic components and has wires running through the A-Arms to connect the power and software to the linear actuators and laser sensors. The feet have two U brackets that attach to the legs with a nut and bolt through a hole at the bottom of the linear actuator. The knuckle has four pieces and is form fitted to the top of the linear actuator and clamps around it to hold the leg and foot system. The A-arms arm hollow tubes have one spherical joint at the end of the top and bottom A-arm that attaches to the knuckle, and they are secured with a nut and bolt on the plates. The other ends of the A-arm also have spherical joints used as base attachments. The base has U brackets welded on to the side that secure the spherical joints of the A-arms with a nut and bolt. The gas spring has eyelet end fittings that attach to the U brackets at the top of the base sidewall and the bottom A-arm secured with a screw and bolt. The laser sensors are attached at the bottom of the knuckle with the servo motor and 3D printed flap attached at the bottom of the leg. The distance between the flap and laser sensor with the linear actuators halfway extended is measured and put into the code to calculate how much the legs moved. The model has three walls with a landing system attached, each landing system wall being separated by an empty wall that attaches to the testing rig.

IV. Test Rig

To test our prototype a testing set up was planned and built. The test rig is meant to simulate the prototype landing onto the surface of Psyche at an assumed impact velocity of 0.92 m/s on

Earth. This velocity is simulated by letting our prototype down with a rope and a counterweight attached to the other end. The mass of the counterweight used is 20.77 kg. The counterweight mass needed was calculated by using dynamic relations between pulley systems. The mass of our lander plays a part in the calculations. The counterweight chosen was based off a 23-kilogram landing spacecraft prototype with an impact velocity of 0.92 m/s. Our chosen impact velocity is related to the prototype mass as well.

The bottom surface of our test rig is modeled with materials set by the Arizona State University for what is assumed to be materials that replicate Psyche's assumed surface. The materials used for the terrain are as follows: black washed gravel, lava rocks, and basalt gravel. These rocky materials are set in place on a canvas drop cloth with polyurethane glue over sandbags for different variations in height. The sandbags are adjusted for multiple tests with the difference between the highest sandbag and the lowest point on the terrain (the test rig base) being no more than 40.64 centimeters. The full test rig set up is shown below in figure 28.

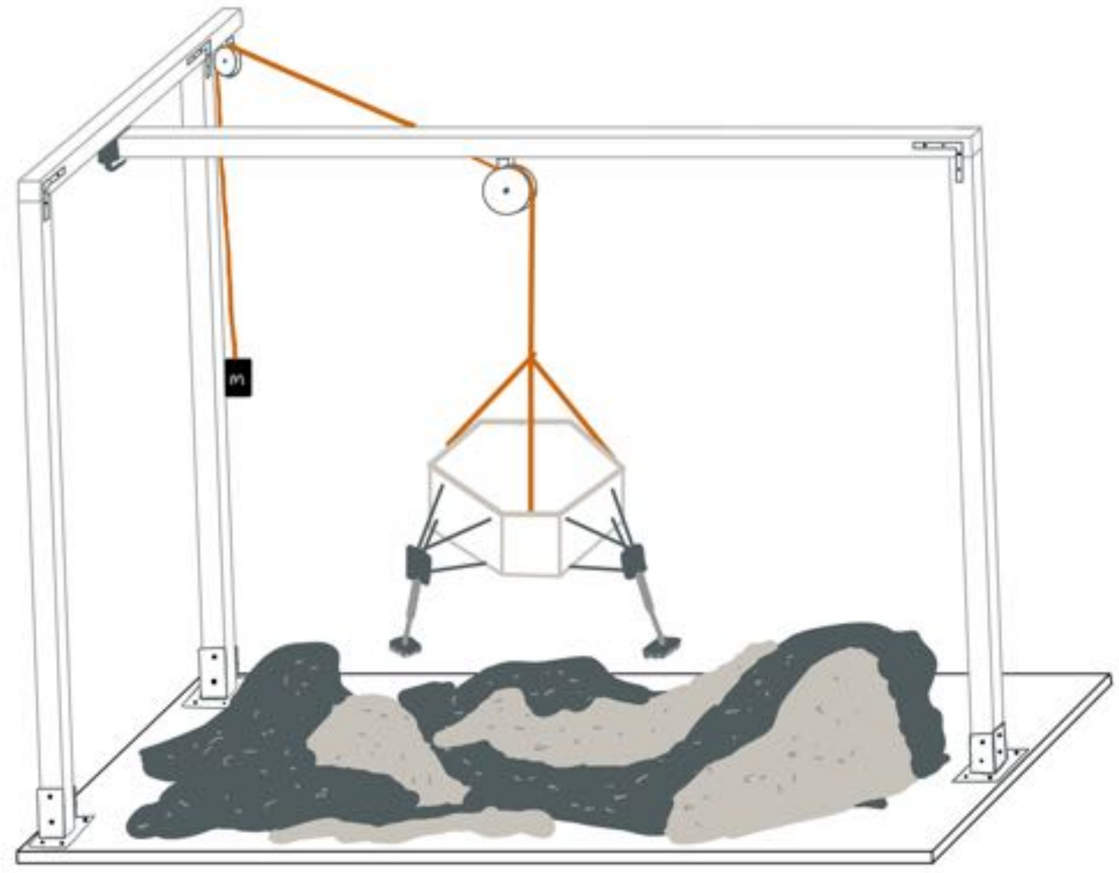


Figure 28: Fully Assembled Test Rig with Terrain and Prototype

The test rig consists of a plywood base to which each of the three main support beams are mounted to using floor mounts and the appropriate screws to the respective mounts. Two of the main vertical support beams are the same height, joined together at their tops with horizontal support beam. The horizontal support beam is attached to the vertical support beam with a 90-degree angle bracket on two sides for each attachment point. One end of the beam extends past the side of the vertical support beam by 30.48 centimeters. This is done so that a pulley is attached to the underneath side of the horizontal extension. This is where the counter mass is suspended. The third main vertical support beam is attached on the opposite side of the square plywood base. It is 5.08 centimeters shorter than the other two vertical support beams. It is placed directly in between the other two vertical supports, just across from them. A separate support beam is mounted on top of the third vertical support, extending horizontally from the third vertical support, intersecting perpendicular to the beam connecting the first two vertical supports. The separate support beam is mounted to the third vertical support by two 90-degree brackets on opposite sides. The perpendicular beam falls right underneath the beam connecting the first two main vertical supports for an adjustable swivel mount to connect the two perpendicular pieces. The adjustable swivel mount is mounted onto the top support beam and the connecting piece extended from the mount are connected to the lower beam. This allows for the two top perpendicular beams to be secured together. In the middle of the perpendicular beam is a pulley. Rope is connected to the lander on the three other sides of the hexagon base that does not have the double A-arm suspension assembly attached. The rope from the three sides is joined in the middle and a single rope suspends it from the middle pulley. The pulley suspends the prototype directly over the modeled Psyche terrain with rope. The rope is extended to the other pulley and connected to the counterweight.

V. Operation

For the operation of the lander and the purposes of testing the prototype, the test rig set up as described is used. First, the prototype is suspended 1 meter off the bottom base midplane. The legs are already in their starting position with the linear actuator halfway extended. This allows for 20.32 centimeters of movement up or down, for a total of 40.64 centimeters. While still suspended in the air, the system will be told to begin the control algorithm as if it were approaching Psyche from space. Since we are not working with as much distance between the lander and the ground, the lander begins the control algorithm. The lander includes buttons and an LCD screen to aid in operation. The sensors will then read the distance between each sensor and the terrain directly below it. The smallest and largest distances of the three distances collected are averaged to identify a midplane and the distance between the midplane and closest and furthest sensors is the corresponds to the amount of time each legs moves, either extending or retracting. Once the legs readjust their position, the sensors will read the distance between each sensor and a flap on a servo motor to check the distance the legs actually move, making necessary adjustments. The lander then hits the terrain and experiences the impact force. Once it lands, the IMU will measure the Euler angles of the lander to check the lander is upright and will make the final leg length adjustments to become perpendicular to the force of gravity. Impact velocity and Euler angles are displayed on the LCD screen.

VI. Troubleshooting

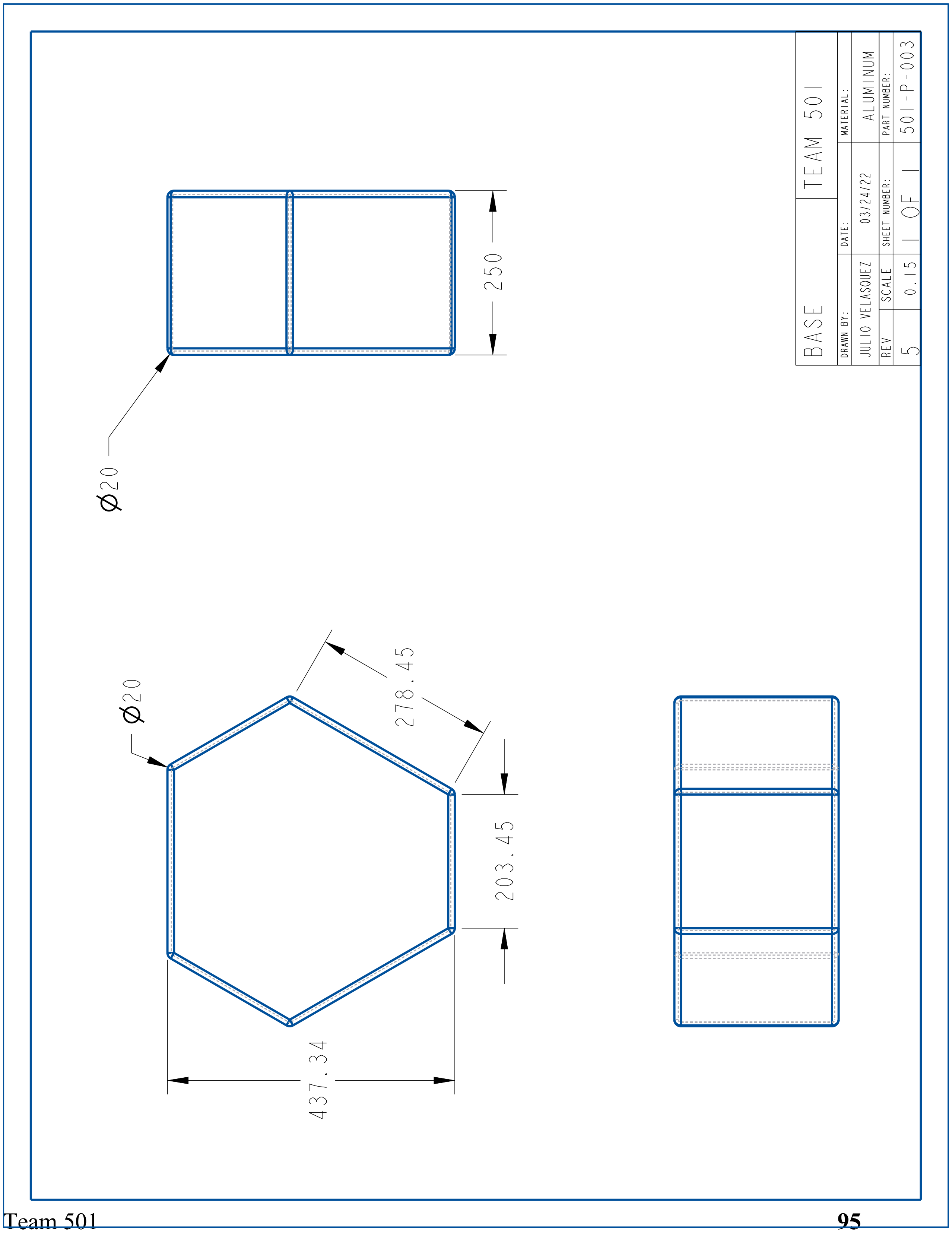
Information from laser sensors may not always be collected by the computer, a USB to TLL device needs to be connected to adjust the baud rate from those that are standard for Arduino to the baud rate of the sensors which is 115200.

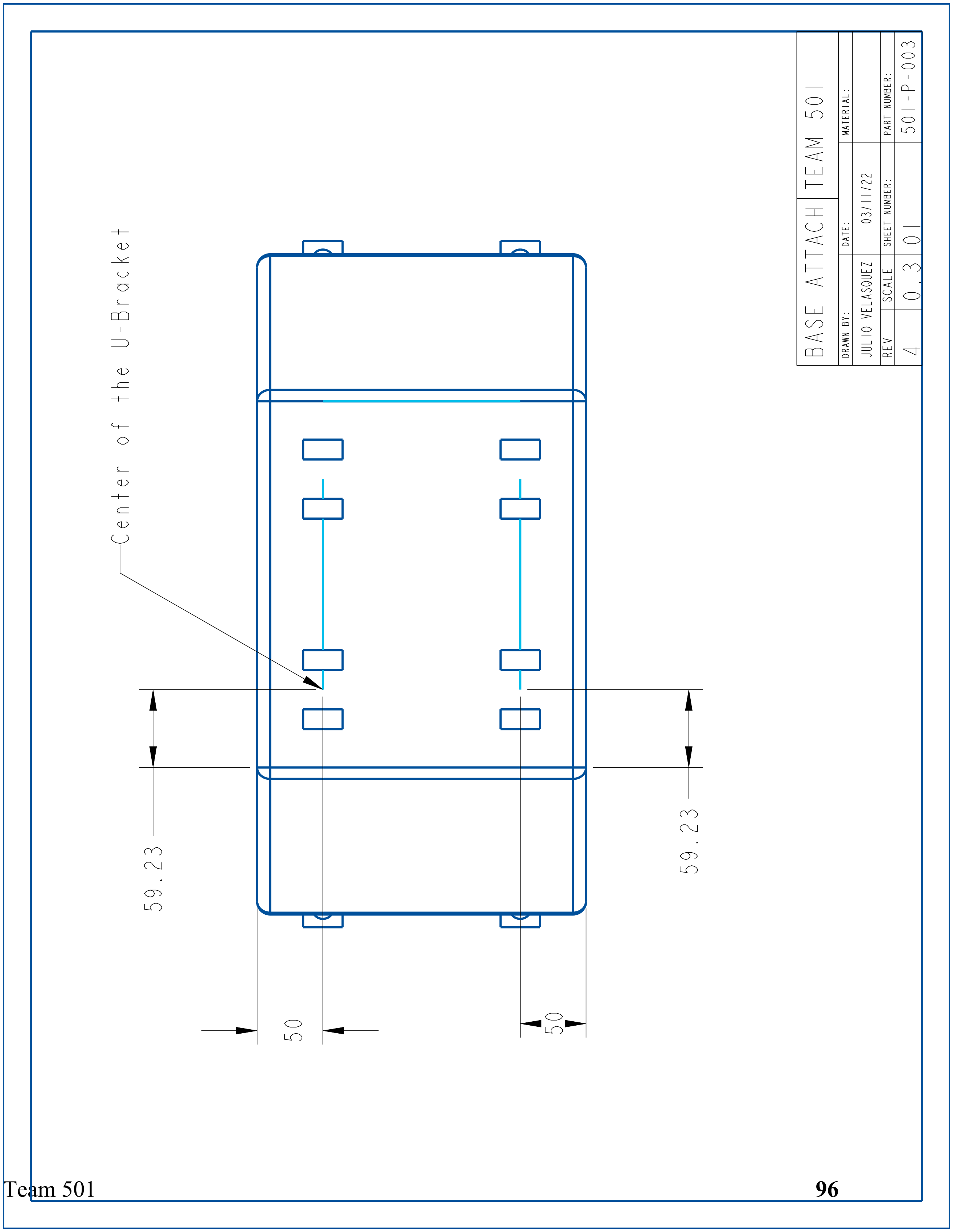
The wires on the motor drivers can sometimes loosen. It is important to immediately check the motor drivers and linear actuator connections since running electricity through them can burn out the motor. Make sure the motor drivers are plugged in, tighten screws.

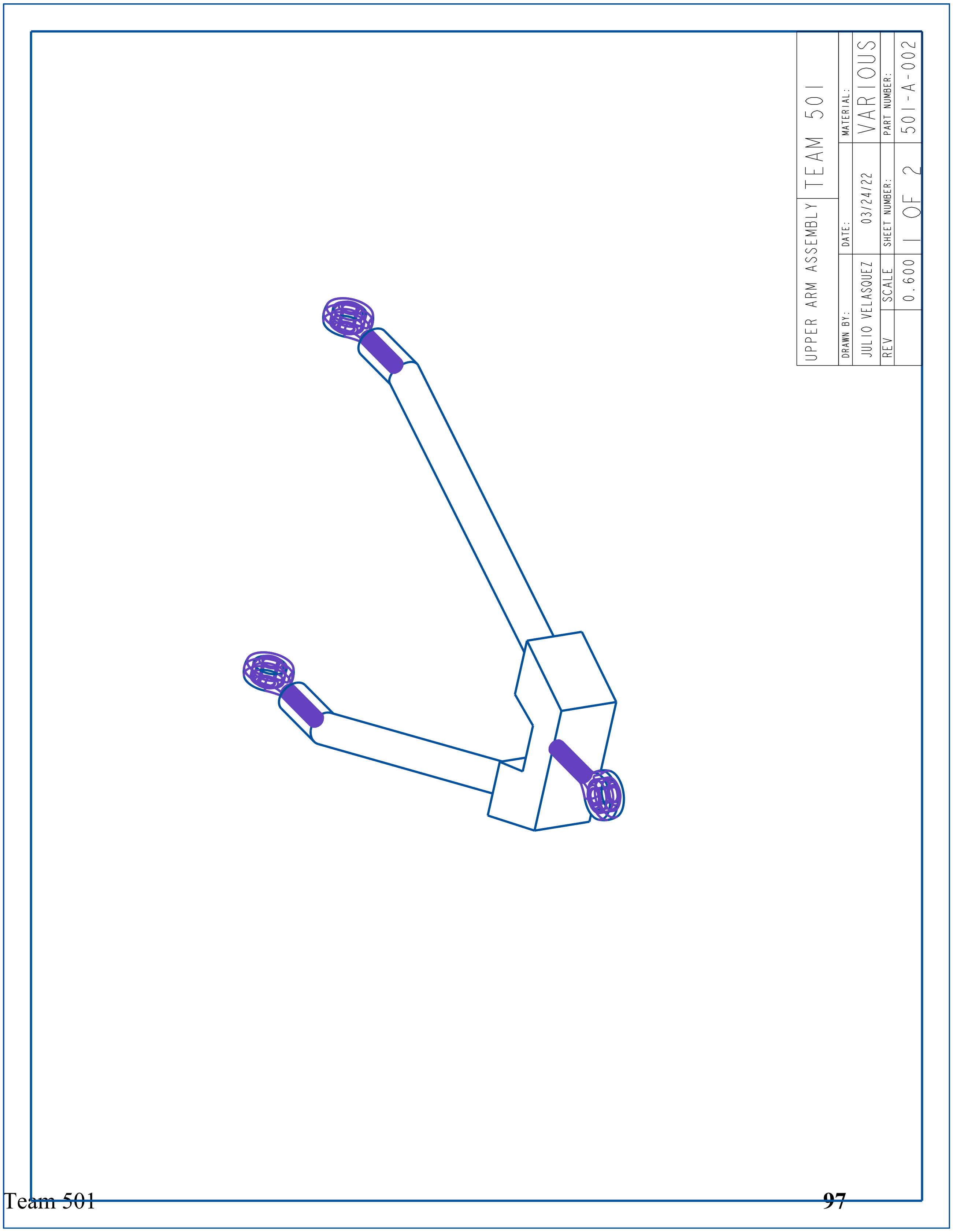
If the legs are not moving at the same rate, restart the experiment. A reset button is on the breadboard and pressing it will cause the legs to fully retract and start the algorithm all over again.

Appendix A: Landing System CREO Drawings

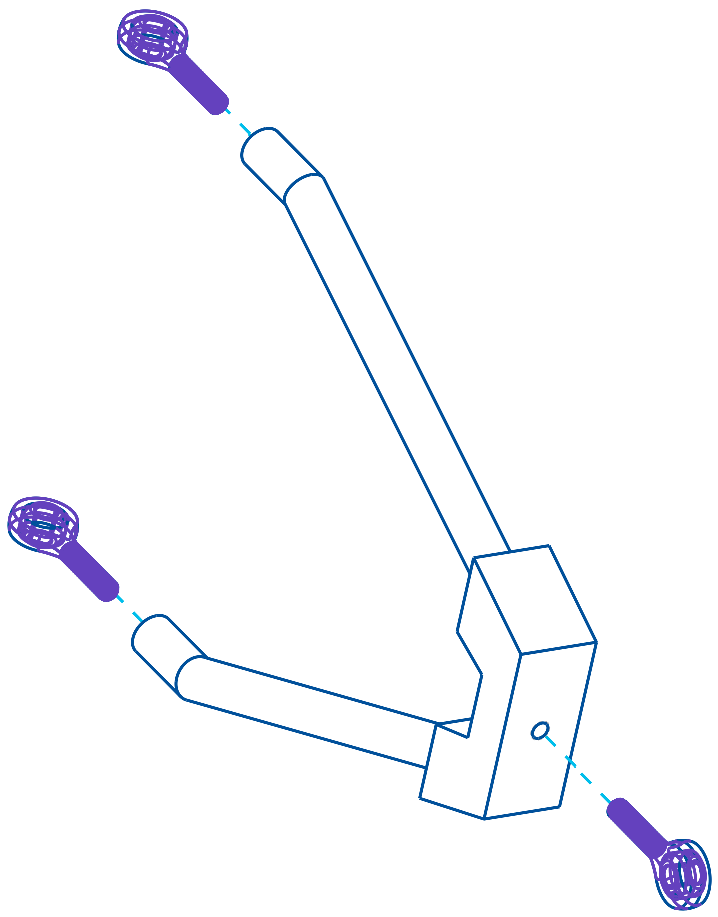
(<u>Note</u>: all dimensions on the CREO drawings in Appendix A are in millimeters)


BASE
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
ALUMINUM
REV
5
SCALE
0.15
SHEET NUMBER:
1 OF 1
PART NUMBER:
501-P-003
250
⌀20
⌀20
278.45
203.45
437.34

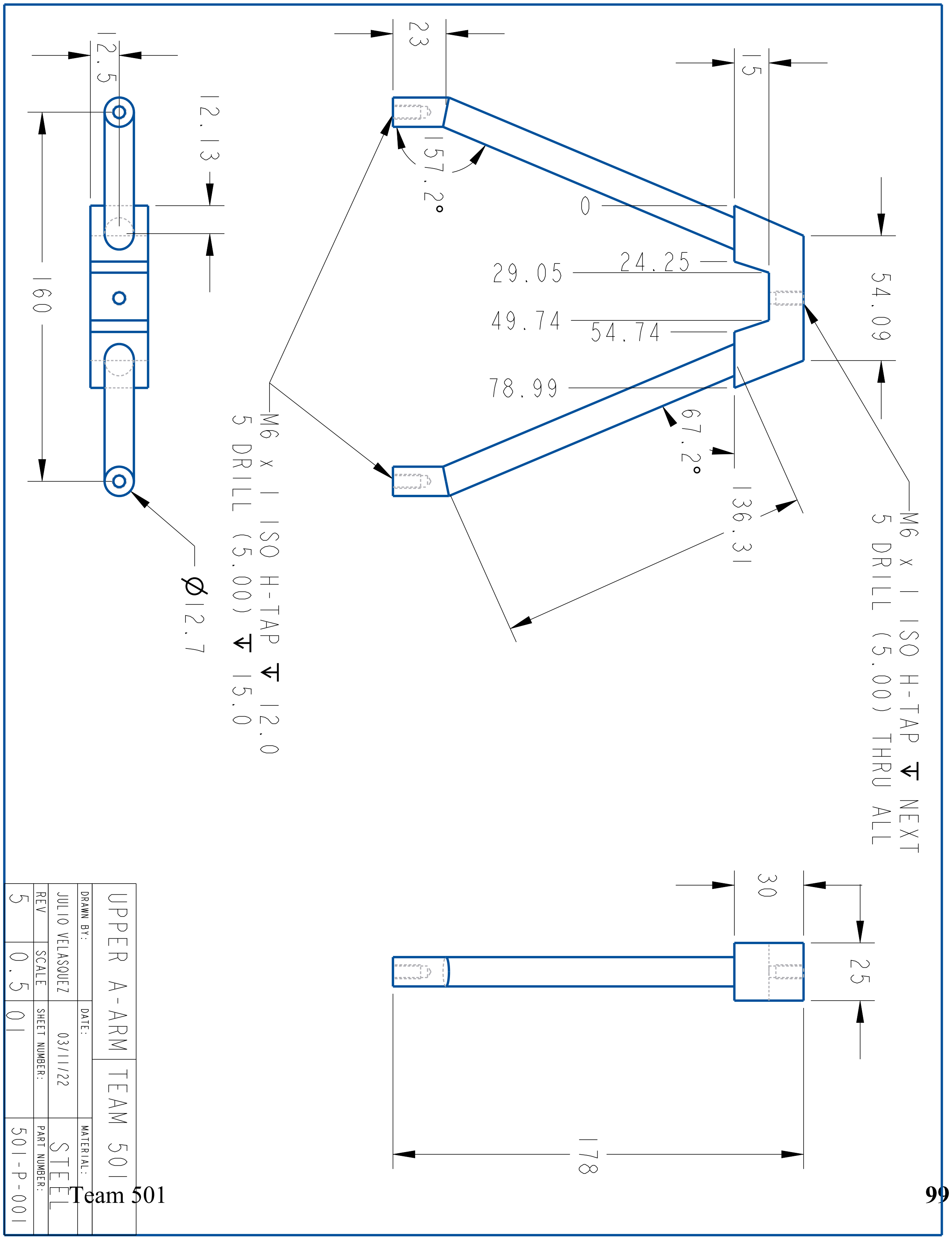

Center of the U-Bracket
59.23
59.23
50
50
BASE ATTACH TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/11/22
MATERIAL:
REV
4
SCALE
0.3
SHEET NUMBER:
01
PART NUMBER:
501-P-003

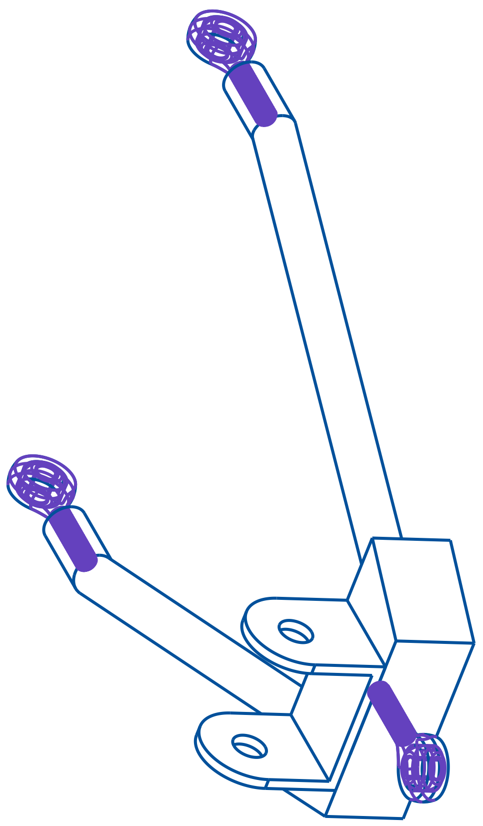


UPPER ARM ASSEMBLY		TEAM 501	
DRAWN BY: JULIO VELASQUEZ		DATE: 03/24/22	MATERIAL: VARIOUS
REV	SCALE 0.600	SHEET NUMBER: 1 OF 2	PART NUMBER: 501-A-002

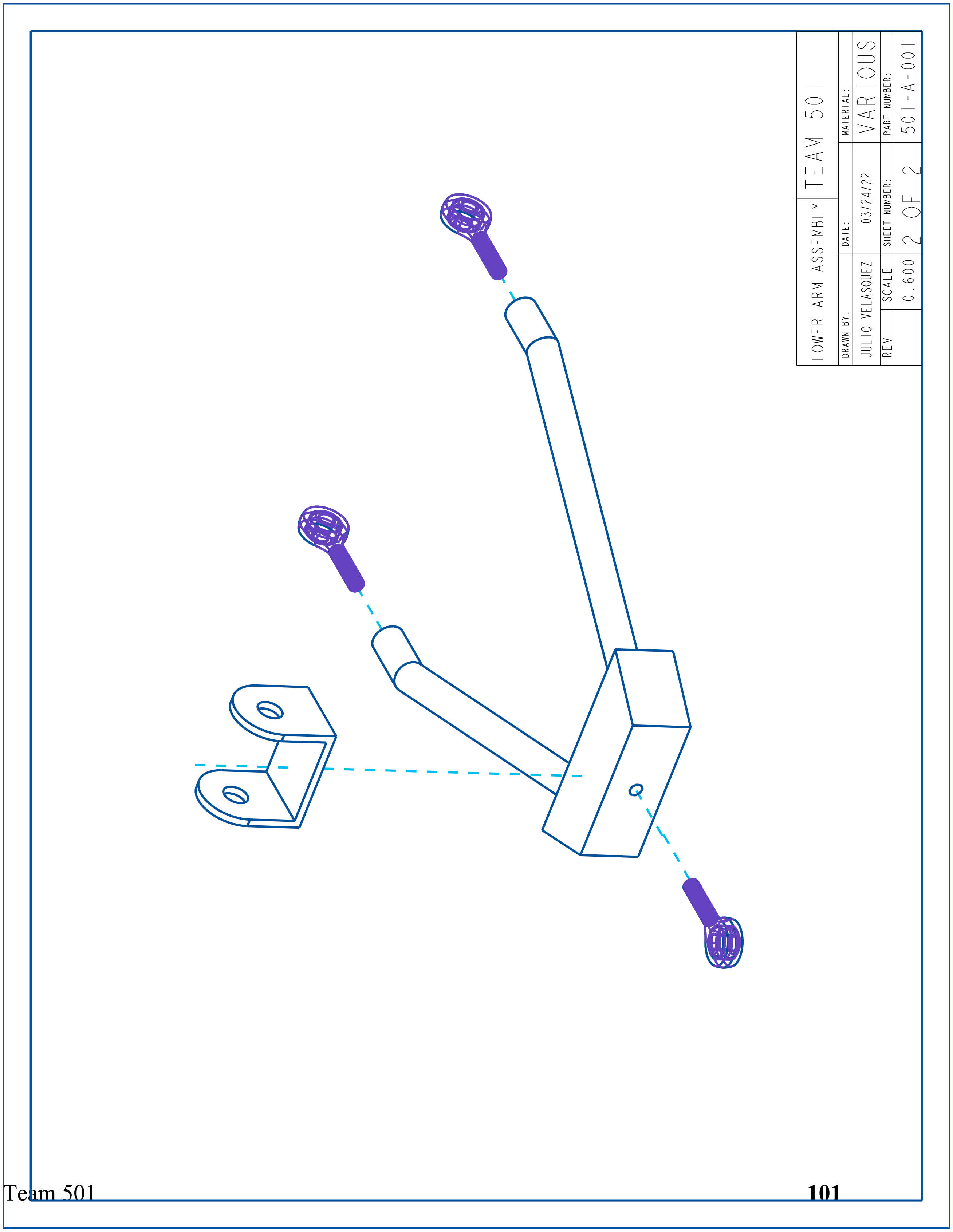


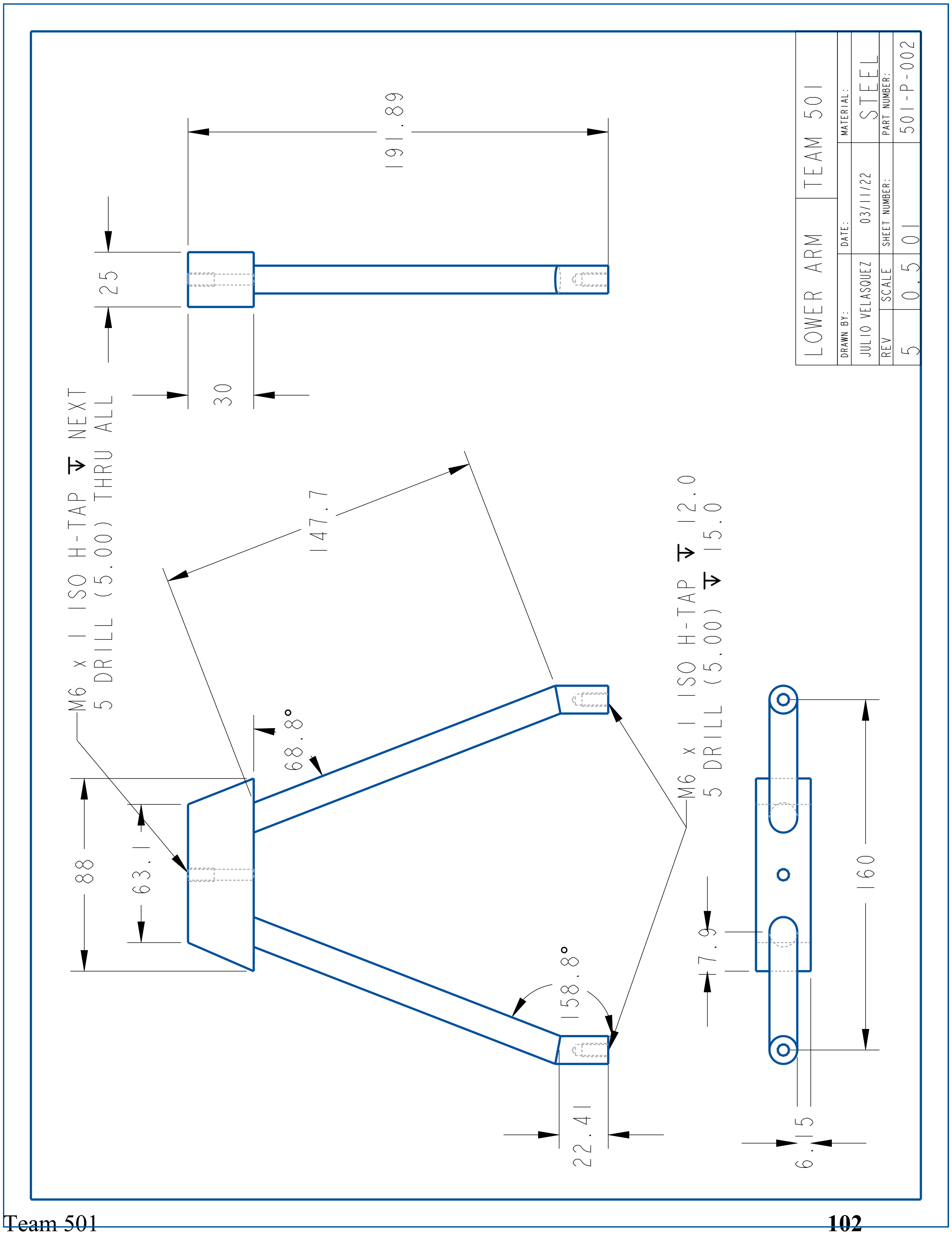
UPPER ARM ASSEMBLY		TEAM 501	
DRAWN BY: JULIO VELASQUEZ	DATE: 03/24/22		MATERIAL: VARIOUS
REV	SCALE 0.600	SHEET NUMBER: 2 OF 2	PART NUMBER: 501-A-002


12.5
12.13
160
Ø12.7
23
157.2°
M6 x 1 ISO H-TAP ⊤ 12.0
5 DRILL (5.00) ⊤ 15.0
15
0
24.25
29.05
49.74
54.74
78.99
67.2°
136.31
54.09
M6 x 1 ISO H-TAP ⊤ NEXT
5 DRILL (5.00) THRU ALL
30
25
178
UPPER A-ARM
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/11/22
MATERIAL:
STEEL
REV
5
SCALE
0.5
SHEET NUMBER:
01
PART NUMBER:
501-P-001

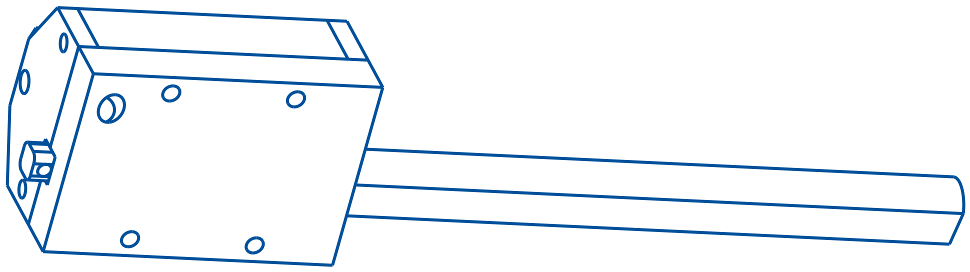


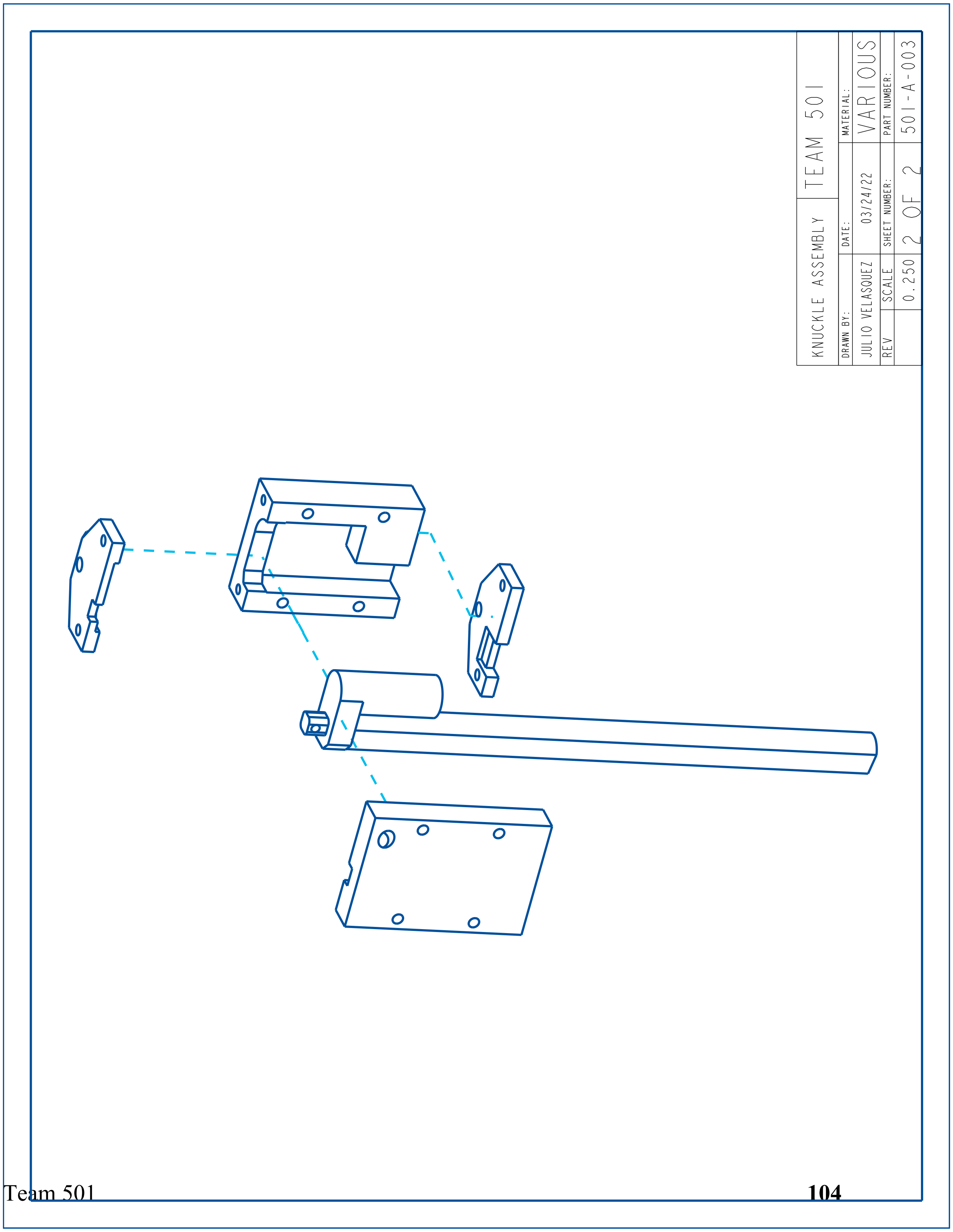
LOWER ARM ASSEMBLY		TEAM 501	
DRAWN BY:		DATE:	MATERIAL:
JULIO VELASQUEZ		03/24/22	VARIOUS
REV	SCALE	SHEET NUMBER:	PART NUMBER:
	0.600	1 OF 2	501-A-001

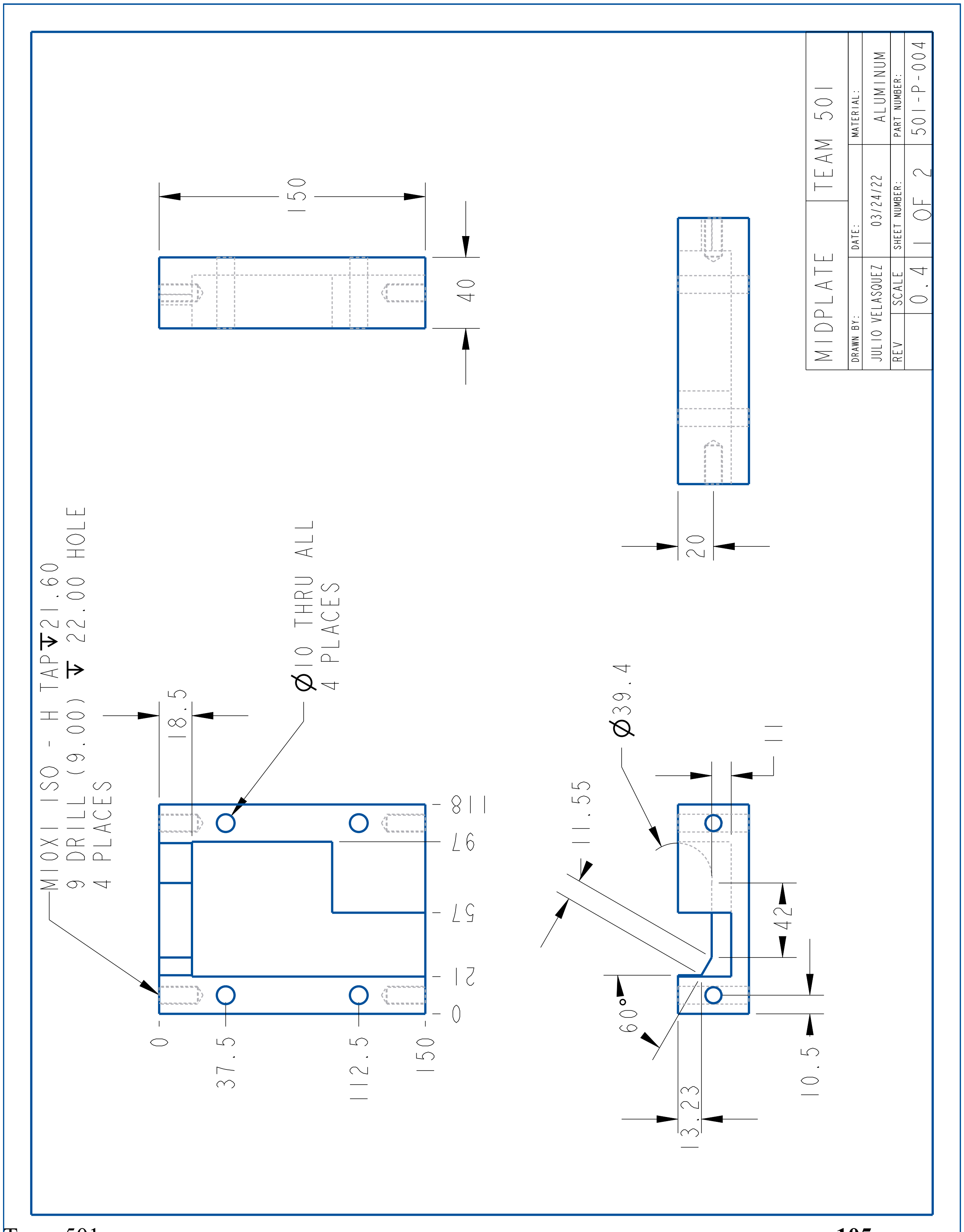

LOWER ARM ASSEMBLY
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
VARIOUS
REV
SCALE
0.600
SHEET NUMBER:
2 OF 2
PART NUMBER:
501-A-001


LOWER ARM
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/11/22
MATERIAL:
STEEL
REV
5
SCALE
0.5
SHEET NUMBER:
01
PART NUMBER:
501-P-002
191.89
25
30
M6 x 1 ISO H-TAP ↧ NEXT
5 DRILL (5.00) THRU ALL
147.7
68.8°
88
63.1
M6 x 1 ISO H-TAP ↧ 12.0
5 DRILL (5.00) ↧ 15.0
158.8°
22.41
17.9
160
6.15

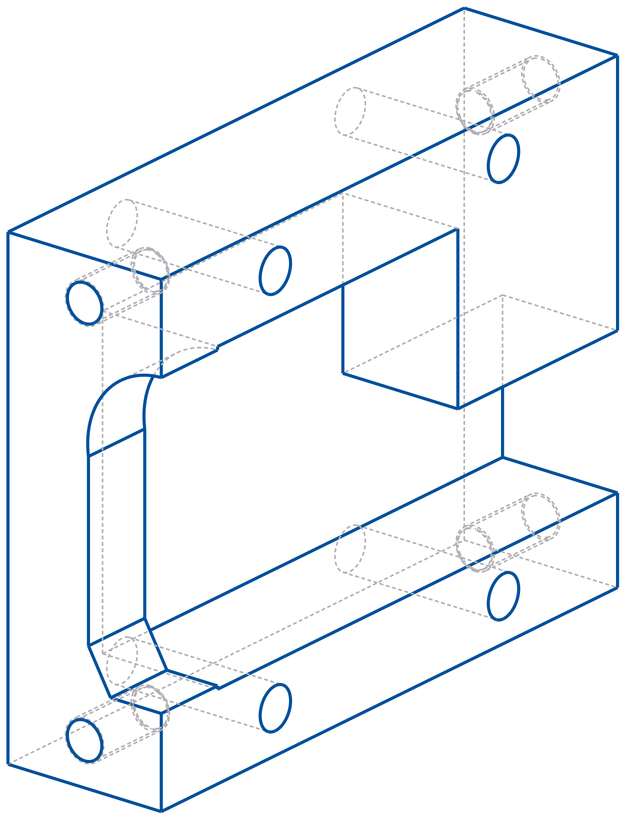
KNUCKLE ASSEMBLY		TEAM 501
DRAWN BY: JULIO VELASQUEZ	DATE: 03/24/22	MATERIAL: VARIUOS
REV	SCALE 0.300	SHEET NUMBER: 2 OF 2
		PART NUMBER: 501-A-003

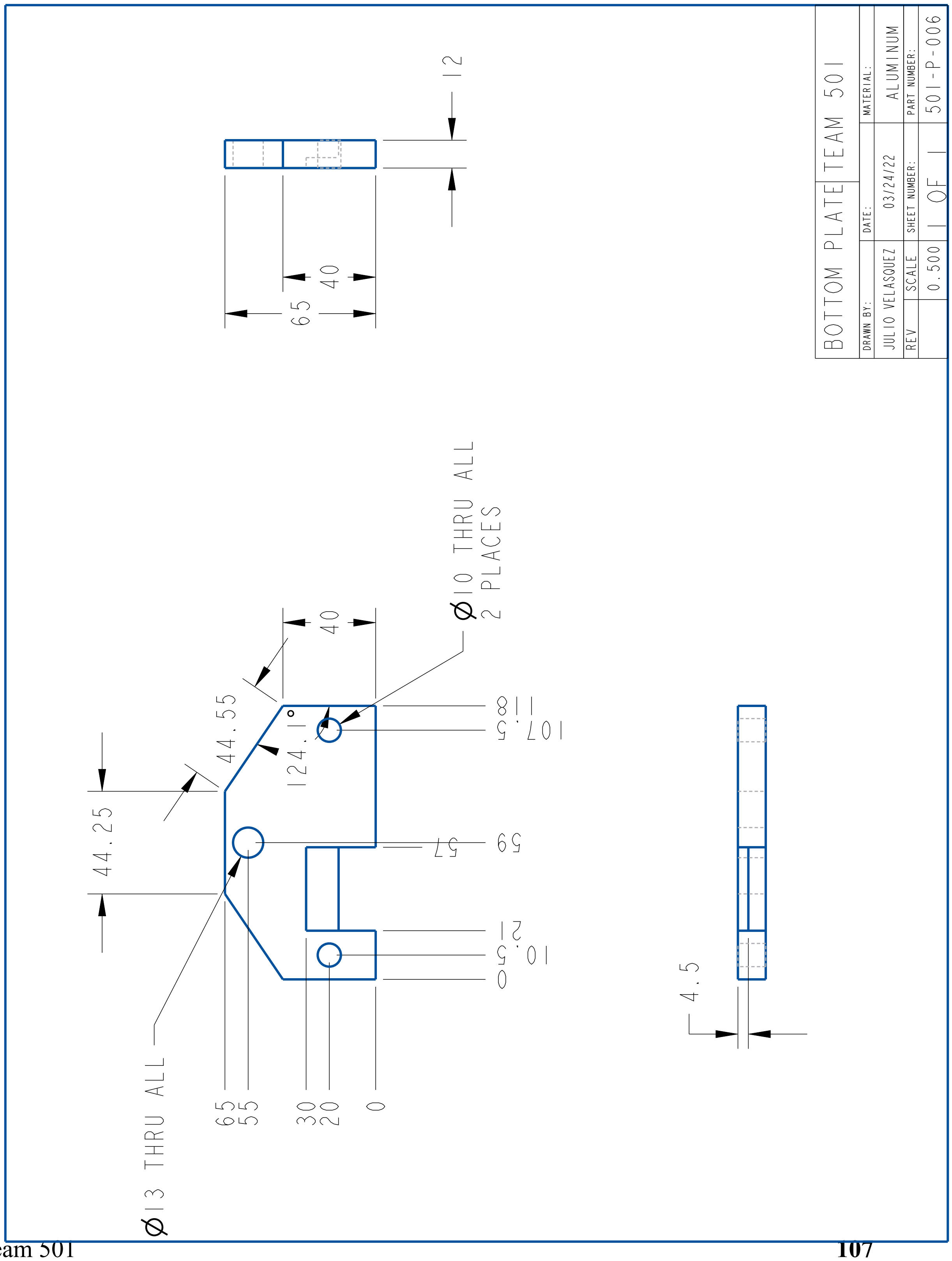


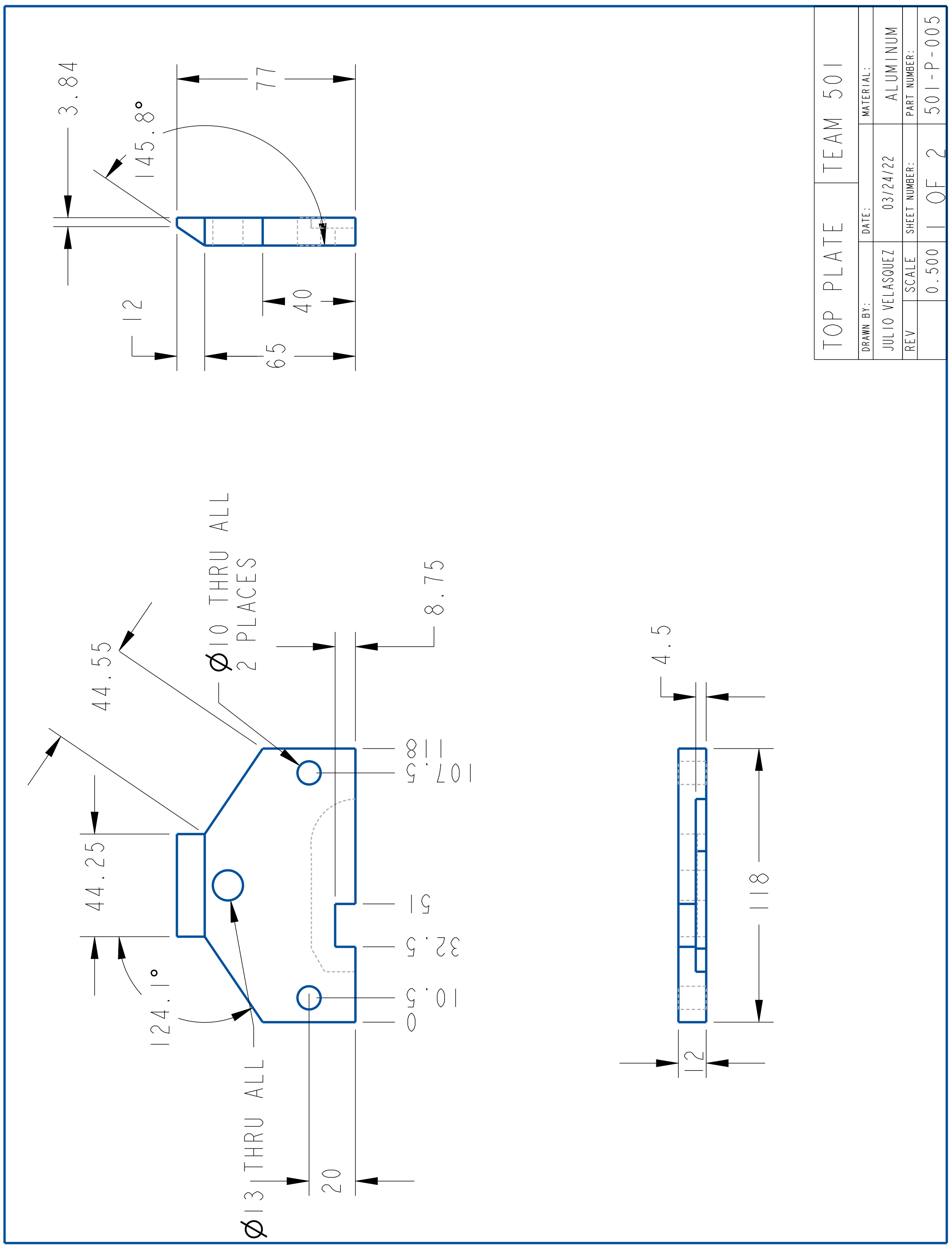

KNUCKLE ASSEMBLY
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
VARIOUS
REV
SCALE
0.250
SHEET NUMBER:
2 OF 2
PART NUMBER:
501-A-003

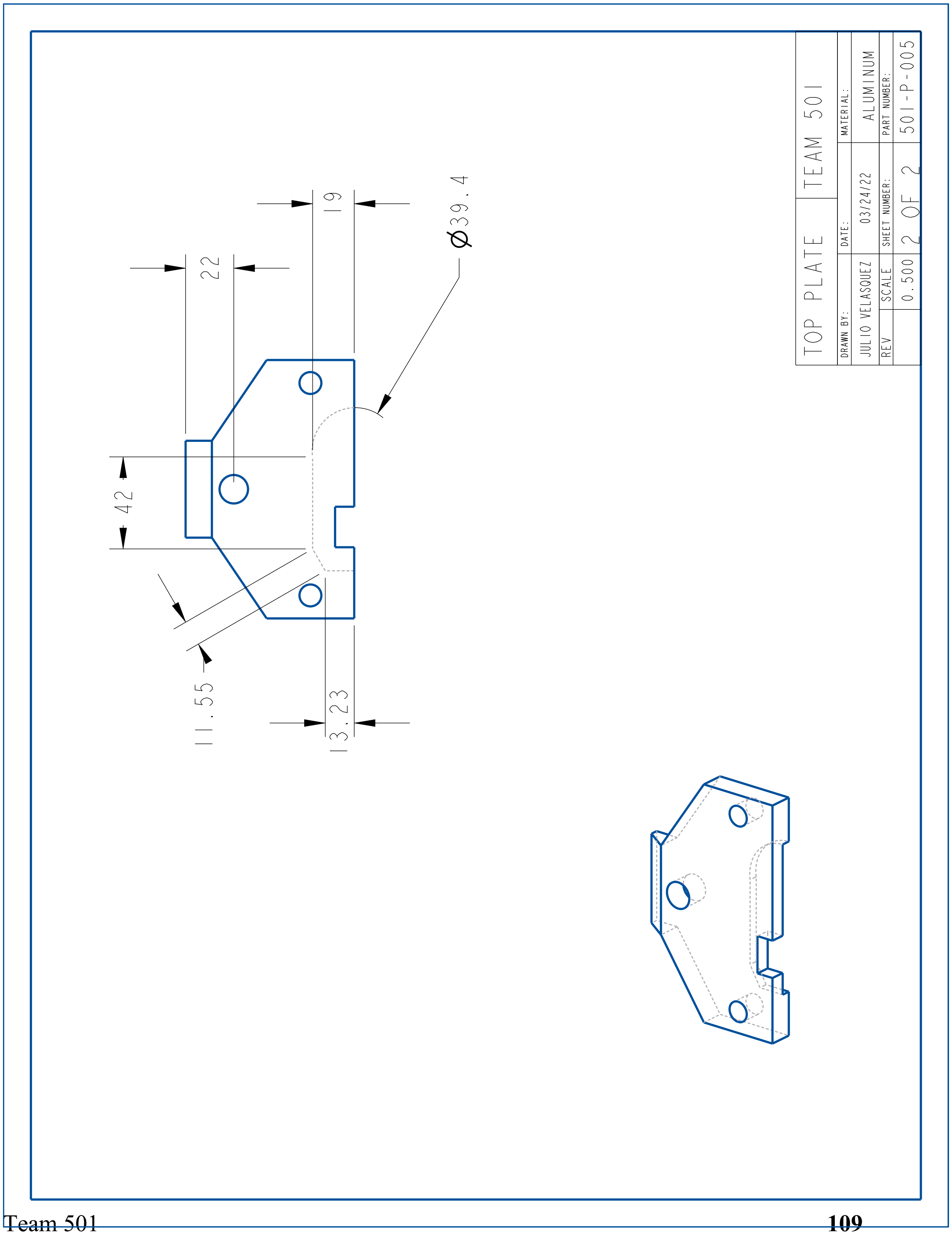

MIDPLATE
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
ALUMINUM
REV
SCALE
0.4
SHEET NUMBER:
1 OF 2
PART NUMBER:
501-P-004
M10X1 ISO - H TAP ⊽ 21.60
9 DRILL (9.00) ⊽ 22.00 HOLE
4 PLACES
Ø10 THRU ALL
4 PLACES
150
40
20
18.5
118
97
57
21
0
0
37.5
112.5
150
11.55
Ø39.4
11
42
60°
10.5
13.23

MIDPLATE		TEAM 501	
DRAWN BY: JULIO VELASQUEZ		DATE: 03/24/22	MATERIAL: ALUMINUM
REV	SCALE 0.800	SHEET NUMBER: 2 OF 2	PART NUMBER: 501-P-004




BOTTOM PLATE TEAM 501
DRAWN BY: JULIO VELASQUEZ
DATE: 03/24/22
MATERIAL: ALUMINUM
REV
SCALE 0.500
SHEET NUMBER: 1 OF 1
PART NUMBER: 501-P-006
12
65
40
Ø10 THRU ALL
2 PLACES
Ø13 THRU ALL
44.55
44.25
124.1°
118
107.5
59
57
12
10.5
0
65
55
30
20
0
4.5


TOP PLATE
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
ALUMINUM
REV
SCALE
0.500
SHEET NUMBER:
1 OF 2
PART NUMBER:
501-P-005
3.84
145.8°
77
12
40
65
Ø10 THRU ALL
2 PLACES
8.75
44.55
44.25
124.1°
Ø13 THRU ALL
20
118
107.5
51
32.5
10.5
0
4.5
12



TOP PLATE		TEAM 501	
DRAWN BY:	JULIO VELASQUEZ	DATE: 03/24/22	MATERIAL: ALUMINUM
REV	SCALE 0.500	SHEET NUMBER: 2 OF 2	PART NUMBER: 501-P-005

Appendix B: Test Rig Assembly Dimensions and Set Up

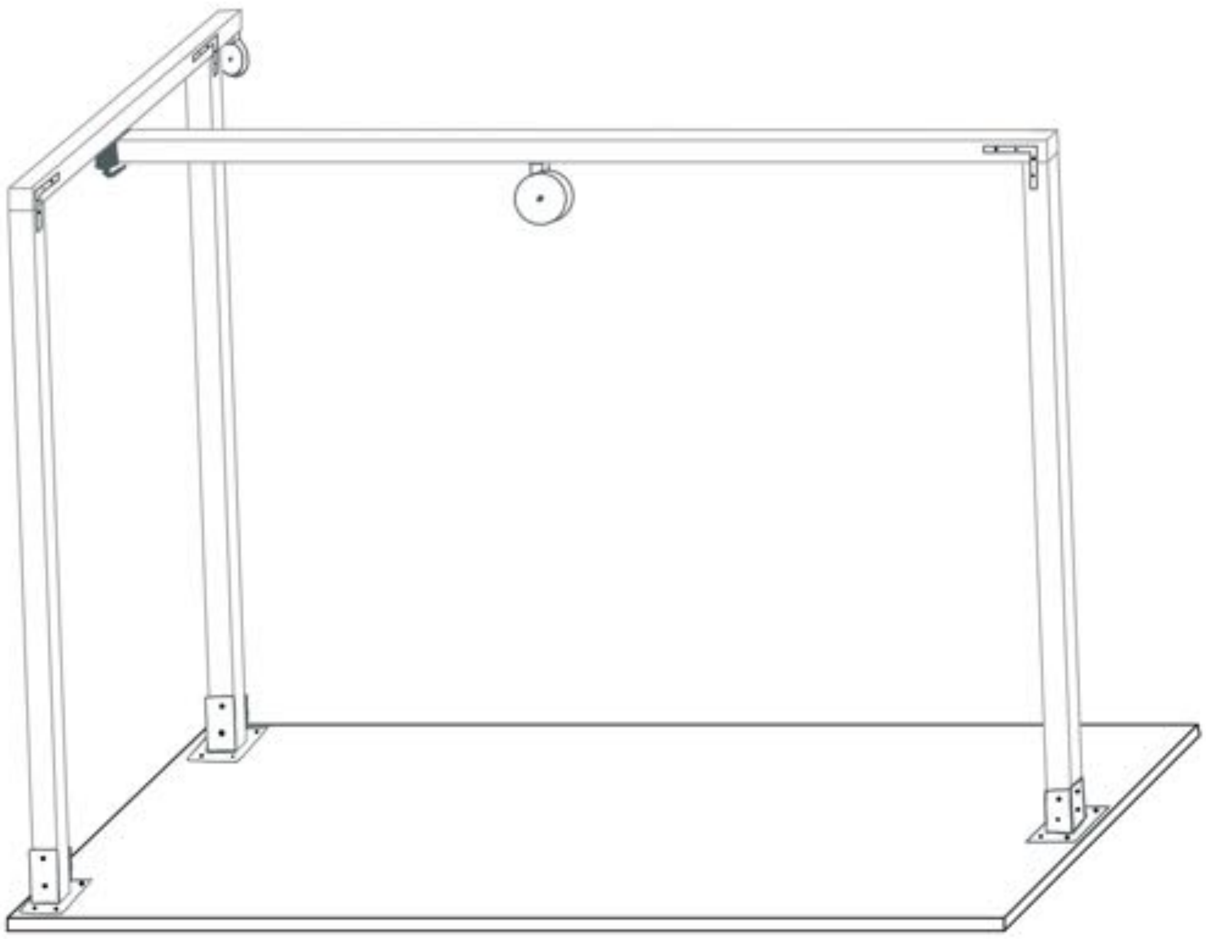


Figure B.1: Test Rig Fully Assembled Without Terrain or Prototype Components

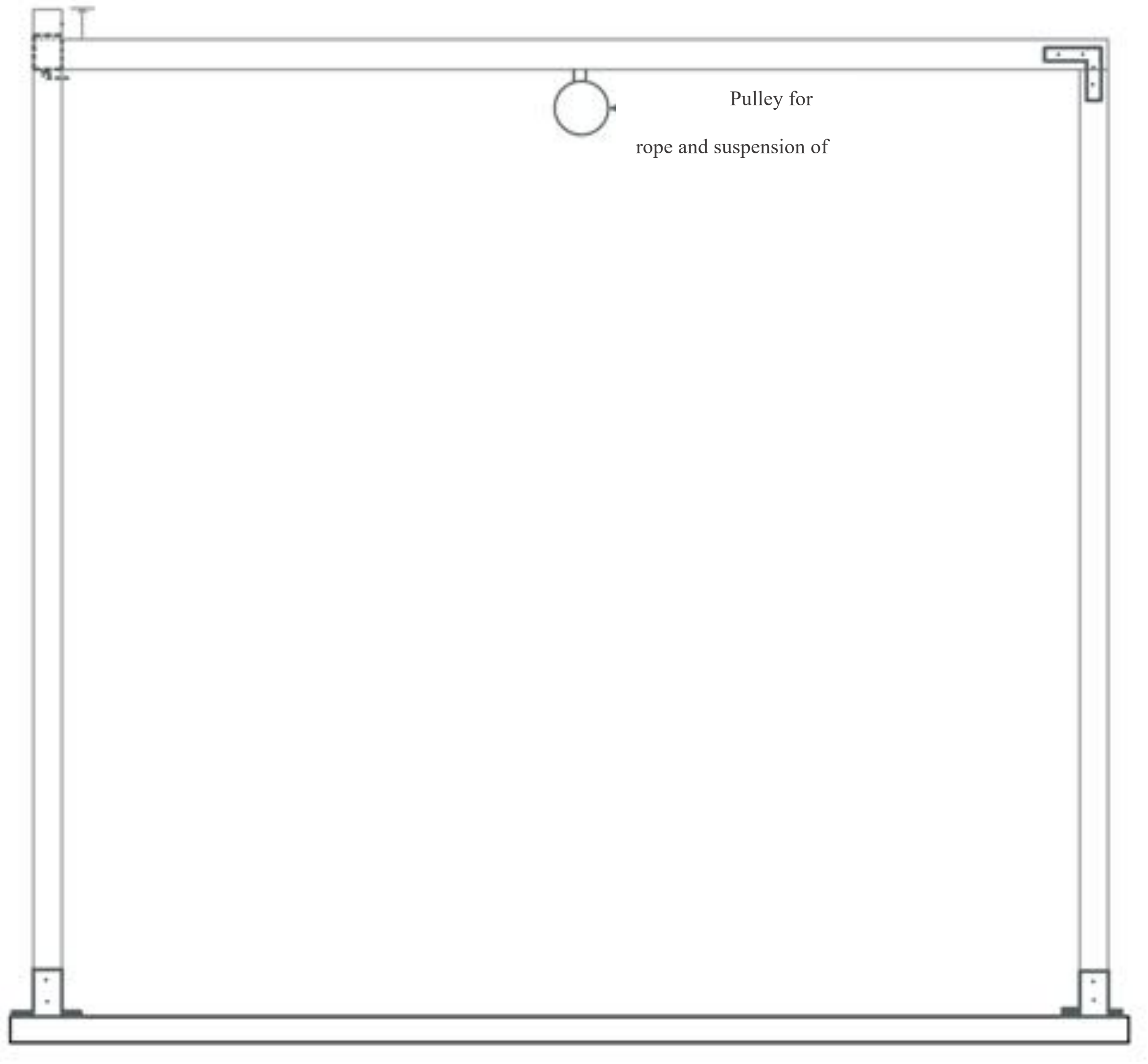


Figure B.2: Test Rig Side View (No Terrain)

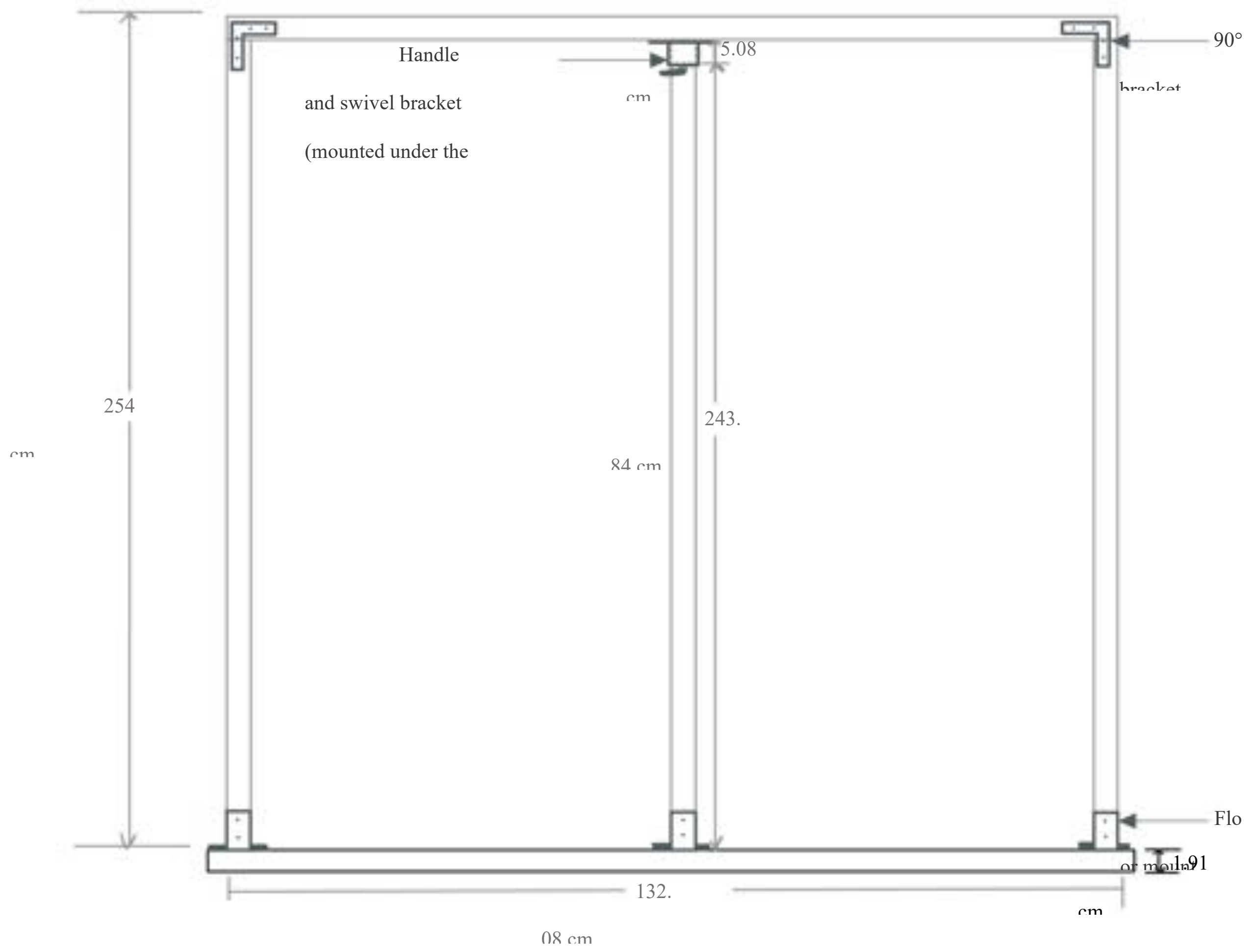


Figure B.3: Test Rig Front View Without Terrain

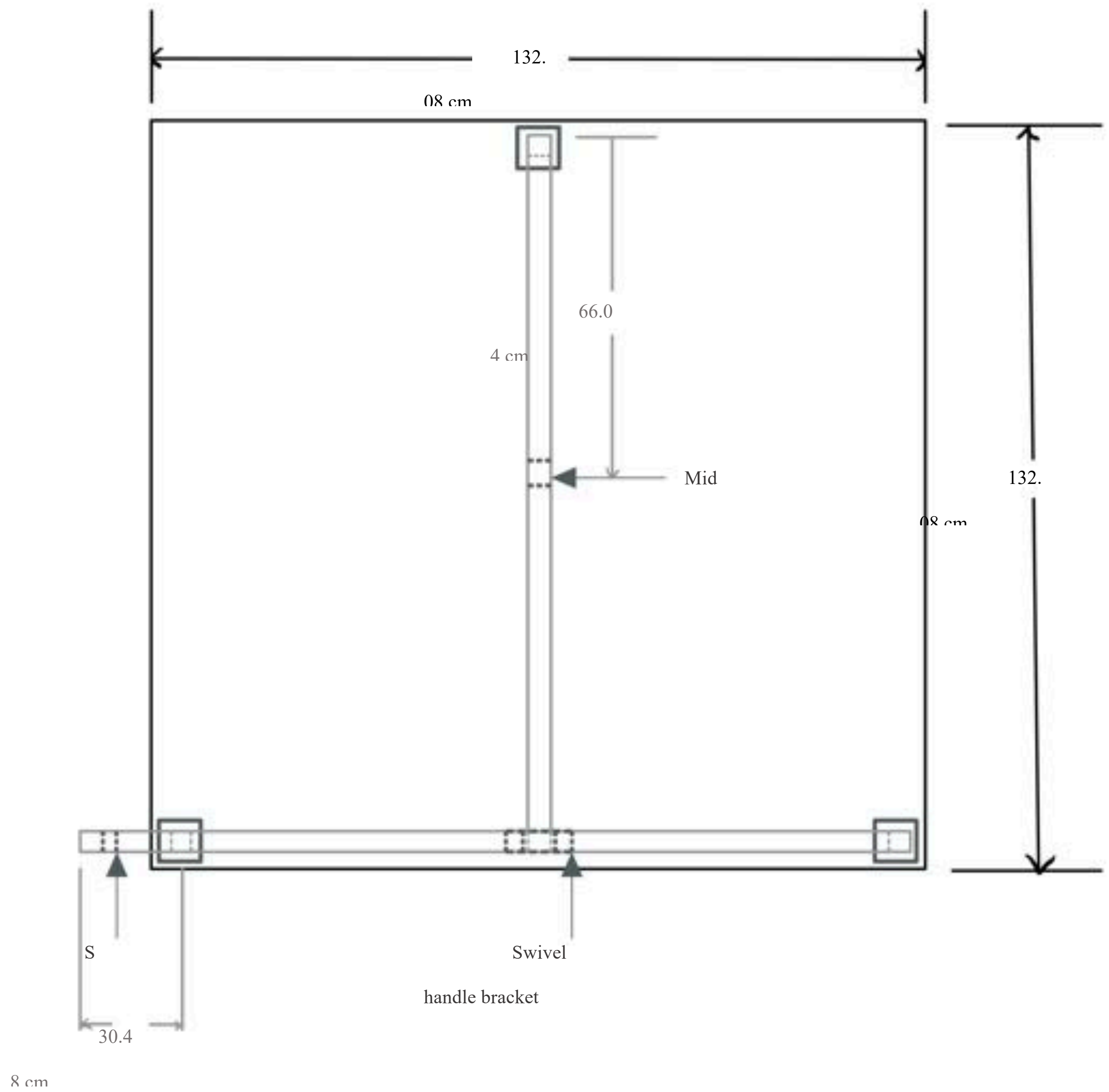


Figure B.4: Test Rig Top View Without Terrain

Appendix C: Sensor and Linear Actuator Code

```
// Uses 12V and 0.5 - 2.5 A

//  https://www.windynation.com/cm/Actuator%20Manual_R3.pdf

// or google "LIN-ACT1-16 documentation" and its by

windynation

// in2 = Pin 36 (PC1), in1 = Pin 37 (PC0) -> linear

actuator 1

// in1 = Pin 49(PL0),in2 = Pin 48 (PL1) -> linear actuator 2

// in1 = Pin 22(PA0),in2 = Pin 23 (PA1) -> linear actuator 3





// Arduino Microcontroller Documentation

// website for all documentation: https://docs.arduino.

cc/hardware/mega-2560

// Pinout:

https://docs.arduino.cc/static/edb006e0a180e80b9ea9cf8b4859d

cd4/A000067-full-pinout.pdf

// or on website

// IMPORTANT NOTES:

// For external interrupts only pin 2,3,18,19,20,21 can be

used
```

```
// External interrupts are currently used for ultrasonic
sensors(3)
// and for microswitches(3) in order to reset legs
//
// Active PORTS: A (motor), B (trigger - sensors), C
(motor), D (echo - sensors), L (motor), E (microswitch)



//   Distance Sensors
//
//   SENSOR 1
//   Using pin21 (PD0) for ECHO with an external interrupt
//   Employing pin 53 (PB0) for the trigger
double distance; //distance reading from sensor
byte lastEchoState;
int echoWidth;
unsigned long timer_1;
int j = 0; // counter variable for distance sensor
// SENSOR 2
// Using pin20 (PD1) for ECHO with an external interrupt
// Employing pin 52 (PB1) for the trigger
```

```
double distance_2; //distance reading from sensor

byte lastEchoState_2;

int echoWidth_2;

unsigned long timer_2;

// SENSOR 3

// Using pin19 (PD2) for ECHO with an external interrupt

// Employing pin 51 (PB2) for the trigger

double distance_3; //distance reading from sensor

byte lastEchoState_3;

int echoWidth_3;

unsigned long timer_3;


// Microswitches

// Pin 18(PD3), 2(PE4), 3(PE5)

int switch1 = 18;

int switch2 = 2;

int switch3 = 3;


// TIMER 1 Interrupt Variables

const uint16_t t1_load = 0; // resets timer to 0

const uint16_t t1_comp = 6250; // need 6250 ticks for 0.1

seconds at prescalar of 256
```

```
double time_cur; //s, current time


// Distance variables inside of TIMER ISR Function

int k = 0; // counter variable

double dist1_hold; //cm, holds current distance reading

for leg 1

double dist1_avg = 0; //cm, holds average distance rading

for leg 1

double dist2_hold;

double dist2_avg = 0;

double dist3_hold;

double dist3_avg = 0;

double dist_avg = 0; // holds the average value for all

three distance sensors

                    // IDEALLY ON FLAT SURFACE: the avg is

equal to 0 IF all placed at same height

double dist1_sum = 0; // holds sum for all distance

measurements

double dist2_sum = 0; // initializes as 0, as to not mess

up the math

double dist3_sum = 0;
```

```
// Linear Actuator Variables

// Place red in out2 and black in out1

int fwd = 2; //linear fwd -> red in out2 , in2 = bit1

int rev = 1; // linear rev -> black in out1, in1 = bit0

double speed_actuator = 10; // mm/s

double speed_act = 1; // cm/s

double time_leg1; //time needed to move leg1

double time_leg2; //time needed to move leg1

double time_leg3; //time needed to move leg1

double time_start; // time legs started to move

bool leg1_reset = false; // if reset is DONE then TRUE, if

reset NOT DONE then FALSE

bool leg2_reset = false;

bool leg3_reset = false;

// Global Variables

int state = 0; // holds state for switch, default is always

at 0

int reset_on = 0; // button connected to Pin41(PG0) and it

will read that bit value until pressed
```

```
                    // when pressed it will read 0
double halfway_extend = 20.32; //cm, this is 8 inches
converted to cm
double time_halfway_extend = 20.32; //sec, actuator moves
at 1 cm/s


boolean leg1_adjust_done = false; // Checks if leg1 has
been done adjusting
boolean leg2_adjust_done = false;
boolean leg3_adjust_done = false;
boolean leg_adjust_ready = false; // Checks if operator
says its okay for legs to adjust


double move_leg1_avg = 0; // Average of leg moving distances
double move_leg2_avg = 0;
double move_leg3_avg = 0;


double move_leg1_sum = 0; // Sum of all move legs
double move_leg2_sum = 0;
double move_leg3_sum = 0;


// Debouncing buttons
```

```
int debounce_time_delay = 1; // seconds, the amount of time
to read the button input
double debounce_time_start = 0; // seconds, holds the
current time to know when to start debounce
int reset_button_last = 1; // reset button is bit 0, so
initialize as it has not been pressed
int adjust_leg_button_last = 2; // adjust leg button is bit
1, so initialized at it has not been pressed
void setup() {
  // put your setup code here, to run once:

  // START OF TIMER STUFF
  // Resets Timer1 Control Register to default
  TCCR1A = 0;

  // Sets to CTC mode (after it compares   it falls back to 0)
  // For CTC mode on timer 1 need to set   WGM12 to "1"
  TCCR1B &= ~(1 << WGM13); // sets WGM13   to 0
  TCCR1B |= (1 << WGM12); // sets WGM12    to 1

  // Setting prescalar of 256 ( 1 0 0 for CS12 CS11 CS10
bits)
```

```
  TCCR1B |= (1 << CS12); //sets to 1
  TCCR1B &= ~(1 << CS11); // sets to 0
  TCCR1B &= ~(1 << CS10); // sets to 0

  // Reset Timer1 value and sets compare value
  TCNT1 = t1_load; // sets timer to 0
  OCR1A = t1_comp; // compare value of 6250 for 0.1 second
intervals

  // Enable Timer1 compare interrupt
  TIMSK1 = (1 << OCIE1A); // enables compare match A output
for timer 1

  // Enable global interrupts
  sei();
  // END OF TIMER STUFF

  // Microswitches:
   attachInterrupt(digitalPinToInterrupt(18), ISR_leg1_off,
RISING);
   attachInterrupt(digitalPinToInterrupt(2), ISR_leg2_off,
RISING);
```

```
  attachInterrupt(digitalPinToInterrupt(3), ISR_leg3_off,
RISING);

  // For distance sensor:
  // The interrupt will be to read the the ultrasonic
measurement
  attachInterrupt(digitalPinToInterrupt(21), my_ISR,
CHANGE);
  attachInterrupt(digitalPinToInterrupt(20), my_ISR_2,
CHANGE);
  attachInterrupt(digitalPinToInterrupt(19), my_ISR_3,
CHANGE);

  // PORT REGISTERS:

  // Port B holds Trigger for Distance Sensors and Trigger
is an OUTPUT we send
  DDRB = 0xFF; // the TRIGGER will be an output because we
send it out
  // Port D reads the ECHO measurement as INPUT back from
ultrasonic sensor
  DDRD = 0x00; // set ECHO as input to read
  PORTD = 0xFF; // pull up resistors


  // Button Register

  DDRG = 0b00000000; // reset button connected to Pin 41

bit 0 -> INPUT

                    // adjust leg button connected to Pin

40 bit 1 -> INPUT

  PORTG = 0b00000011; // pull up resistor for bit 0, bit 1


  // Port C, L, A control direction for linear actuators,

set those bits as OUTPUT since we control

  DDRC = 0b00000011; // Sets up both as ouput for linear 1

  DDRL = 0b00000011; // output for linear 2

  DDRA = 0b00000011; // output for linear 3


  // Start Ports for actuators to   be off

  PORTC = 0b00000000; // Both are   off at start linear 1

  PORTL = 0b00000000; // linear 2   off at start

  PORTA = 0b00000000; // linear 3   off at start -> bit7 in

input of reset button


  // Initialization of Variables
```

```
 Serial.begin(9600);


   // TESTING PRINTS

 //

 // This is for testing the buttons and the debounce

 /*

  Serial.print("time");

  Serial.print("\t");

  Serial.print("state");

  Serial.print("\t");

  Serial.print("rst");

  Serial.print("\t");

  Serial.print("adj");

  Serial.print("\t");

  Serial.print("dbnce");

  Serial.print("\t");

  Serial.println("minus");

 */

 //

 // This is for testing the reset state and the
microswitches
```

```
 /*
  Serial.print("time");
  Serial.print("\t");
  Serial.print("state");
  Serial.print("\t");
Serial.print("mode");
Serial.print("\t");
Serial.print("leg1");
Serial.print("\t");
Serial.print("leg2");
Serial.print("\t");
Serial.println("leg3");
*/
// This is for testing the distance sensors
/*
Serial.print("time");
Serial.print("\t");
Serial.print("state");
Serial.print("\t");
Serial.print("dis1");
Serial.print("\t");
Serial.print("dis2");
Serial.print("\t");

Serial.println("dis3");

*/

// This is for testing the average of distances

/*

Serial.print("time");

Serial.print("\t");

Serial.print("state");

Serial.print("\t");

Serial.print("d1");

Serial.print("\t");

Serial.print("d1sum");

Serial.print("\t");

Serial.print("d1avg");

Serial.print("\t");

Serial.print("d2");

Serial.print("\t");

Serial.print("d2avg");

Serial.print("\t");

  Serial.print("d2avg");

  Serial.print("\t");

  Serial.print("d3");
  Serial.print("\t");
  Serial.print("d3avg");
  Serial.print("\t");
  Serial.print("d3avg");
  Serial.print("\t");
  Serial.println("k");
 */
 // This is for testing the farthest dist, closest dist,
z-midplane, z-ref for legs, and movement
 /*
  Serial.print("time");
  Serial.print("\t");
  Serial.print("state");
  Serial.print("\t");
  Serial.print("d1");
  Serial.print("\t");
  Serial.print("d2");
  Serial.print("\t");
  Serial.print("d3");
  Serial.print("\t");
  Serial.print("d_cl");
  Serial.print("\t");
  Serial.print("d_far");

  Serial.print("\t");

  Serial.print("z_ref1");

  Serial.print("\t");

  Serial.print("z_ref2");

  Serial.print("\t");

  Serial.print("z_ref3");

  Serial.print("\t");

  Serial.print("z_mid");

  Serial.print("\t");

  Serial.print("m1");

  Serial.print("\t");

  Serial.print("m1avg");

  Serial.print("\t");

  Serial.print("m2");

  Serial.print("\t");

  Serial.print("m2avg");

  Serial.print("\t");

  Serial.print("m3");

  Serial.print("\t");

  Serial.print("m3avg");

  Serial.print("\t");
```

```
  Serial.println("k");
 */
 // This is for testing the farthest dist, closest dist,
z-midplane, z-ref for legs, and movement EAISER TO SEE

 Serial.print("time");
 Serial.print("\t");
 Serial.print("state");
 Serial.print("\t");
 Serial.print("d1");
 Serial.print("\t");
 Serial.print("d2");
 Serial.print("\t");
 Serial.print("d3");
 Serial.print("\t");
 Serial.print("d_cl");
 Serial.print("\t");
 Serial.print("d_far");
 Serial.print("\t");
 Serial.print("z_ref1");
 Serial.print("\t");
 Serial.print("z_ref2");
```

```
  Serial.print("\t");

  Serial.print("z_ref3");

  Serial.print("\t");

    Serial.print("z_mid");

    Serial.print("\t");

    Serial.print("m1avg");

    Serial.print("\t");

    Serial.print("m2avg");

    Serial.print("\t");

    Serial.print("m3avg");

    Serial.print("\t");

    Serial.println("k");




}




void loop() {

  // put your main code here, to run repeatedly:
    // PORT C motor1, PORT L motor2, PORT A motor3


  int mode; // mode for linear actuators

  int reset_button = PING & 0b00000001; // PING &

0b00000001 is reading the bit button connected (PG0) Pin 41

                                          // Will read 1 if

not pushed and read 0 if pushed

  int adjust_leg_button = PING & 0b00000010; // This read

bit connect to adjust leg button (PG1) Pin 40

                                            // Will read 2

if not pushed and read 0 if pushed





  // Closest and farthest point to lander (refrence to

sensors on lander)

  double closest_dist; // holds value to closest distance

measurement

  double farthest_dist;
  // Midplane reference variables
  // This has the closest point as reference z0 = 0
  double z_ref_leg1;
  double z_ref_leg2;
  double z_ref_leg3;
  double z_ref_midplane; // midplane
  double move_leg1; // leg 1 distance from midplane needed
to move
  double move_leg2;
  double move_leg3;




  // ** START OF LOOP LOGIC *
  //
  //
  // This statement is always checked.
  // So if at any point during the process the reset button
is hit, all the legs will immediately begin retracting.
  // This is done for safety.
  //
  //
  // READING THE DIFFERENT BUTTONS
  //
  // ** CAUTION: CANNOT READ BOTH BUTTONS AT SAME TIME **
  //
  // ** REPEAT:    CANNOT READ BOTH BUTTONS AT SAME TIME **
  //
  // Buttons work by PRESSING and HOLDING for over the
"debounce_delay" vairable.
  // Currently: HOLD and PRESS for 1 SECOND
  //
  // READING THE RESET BUTTON
  // The "reset_button" variable reads the CURRENT BUTTON
READING (PORT G, bit 0)
  // "reset_button_last" is initialized as 1, if buttons
arent pressed then it reads a 1.
  // This variable will be set to the button reading at
the end of the loop.
  // Its initialized as 1 so both variables previously
mentioned are equal to each other.
  //
  // The first if-block says if the current button reading
```

```
DOES NOT equal the previous reading
  // then set the "debounce_time_start" to the current
time.
  // Since the "reset_button_last" is initialized as "1",
they will not equal each other when the
  // reset button is pressed an "reset_button" is equal to
"0". At this moment the start of the debounce timer
  // will each to the current time. At the end of the loop
the "reset_button_last" variable will be set to
  // current reading of the button

if ( reset_button != reset_button_last ){
   debounce_time_start = time_cur;
}

  // This if-block says if the DIFFERENCE between the
current time and the start time of the debounce is
  // LARGER than the debounce delay the RESET BUTTON
READING will be evaluated
  // If the button is STILL BEING HELD DOWN then the state
will be 0 and the RESET PROCESS START
  // If the button is NO LONGER HELD DOWN SINCE LAST
```

```
PRESSED then the RESET PROCESS IS IGNORED
  // THE DEFAULT: RESET PROCESS IS IGNORED
  if ( (time_cur - debounce_time_start) >
debounce_time_delay ){
    // If within the delay, the reset button is STILL
PRESSED then the STATE SET TO 1,
    // and the legs will START RESET PROCESS
    // "leg_reset" is set to false, this variable controls
the microswitches.
    // When they are false that means the microswitches are
not pressed and the legs need to retract.
    if ( reset_button == 0 ){
      state = 1;
      leg1_reset = false;
      leg2_reset = false;
      leg3_reset = false;
    }
  }

  // READING THE ADJUST LEG BUTTON
  // Same exact process as the reset button
  // If want to know more, look at the description at the
```

"READING THE RESET BUTTON" section

```
  if ( adjust_leg_button != adjust_leg_button_last ){
    debounce_time_start = time_cur;
  }

  if ( (time_cur - debounce_time_start) >
debounce_time_delay ){
    // If within the delay, the adjust leg button is STILL
PRESSED then the STATE SET TO 4,
    // and the legs will START ADJUST based on the READINGS
FROM STATE 3
    if ( adjust_leg_button == 0 ){
      state = 4;
       leg_adjust_ready == true;
      }
  }

  // Sets the last reset reading variable to the current
reading of the buttons
  reset_button_last = reset_button;
  adjust_leg_button_last = adjust_leg_button;
  // This switch statement will follow the order of the
block diagram. In google docs -> Coding Documentation
  // The state will mostly be increasing sequentially to be
able to track how far in the process the robot is.
  switch (state){
    case 0:


        break;


      case 1:


    // This will control the legs to reverse all the way in
(extension = 0 inch)
    // The first if-else block checks all the
microswitches. If any of them have been
    // pressed then the "legX_reset" variable is equal to
true and that motor will shut off.
    // The mode variable describes which combination of
```

```
legs are still retracting.
    //
    // After the if-else block is checked, there is another
switch statement that is controlled by the "mode".
    // The "mode" variable is changed based on which motors
are on.
    // The switch statement checks what "mode" it is in,
and then will enable and disable the appropriate motors.

        if (leg1_reset == false && leg2_reset == false &&
leg3_reset == false){
          mode = 1;

        }else if (leg1_reset == true && leg2_reset ==false &&
leg3_reset == false){
          mode = 2;

        }else if (leg1_reset == true && leg2_reset == true &&
leg3_reset == false){
          mode = 3;

        }else if (leg1_reset == true && leg2_reset == true &&
leg3_reset == true){

         mode = 4;


      }else if (leg1_reset == false && leg2_reset == true

&& leg3_reset == false){

        mode = 5;


      }else if (leg1_reset == false && leg2_reset == true

&& leg3_reset == true){

        mode = 6;


      }else if (leg1_reset == false && leg2_reset == false

&& leg3_reset == true){

        mode = 7;


      }else if (leg1_reset == true && leg2_reset == false

&& leg3_reset == true){

        mode = 8;

      }

      // The switch block that will control which motors

are reversing and which motors are off

      switch (mode){
```

```
    case 1:
      PORTC = rev;
      PORTL = rev;
      PORTA = rev;
      break;
    case 2:
      PORTC = 0b00000000;
      PORTL = rev;
      PORTA = rev;
      break;
    case 3:
      PORTC = 0b00000000;
      PORTL = 0b00000000;
      PORTA = rev;
      break;
    case 4:
      PORTC = 0b00000000;
      PORTL = 0b00000000;
      PORTA = 0b00000000;
       state = 2; // all microswitches have been hit and
can move onto next state
       time_start = time_cur; // time_start keeps track
```

```
at what time we move onto next state.

                                        // this is done to try to

extend the actuators for the appropriate amount of time.

            break;

         case 5:

            PORTC = rev;

            PORTL = 0b00000000;

            PORTA = rev;

            break;

         case 6:

            PORTC = rev;

            PORTL =   0b00000000;

            PORTA =   0b00000000;

            break;

         case 7:

            PORTC =   rev;

            PORTL =   rev;

            PORTA =   0b00000000;

            break;

         case 8:

            PORTC =   0b00000000;

            PORTL =   rev;
```

```
      PORTA =   0b00000000;
      break;
    }
    break;

  case 2:
  // Extends the legs out to 8 inches
  // "time_start" variable captures the time at which the
state is changed to 2.
  // Then we want the motors to extend to 8 inches
(~20cm).
  // The motors extend at 0.39 inch/s or 1 cm/s.
  // time_halfway_extend is equal to ~20 secs
  //
  // The if-else-if block checks if the time that has
passed since we arrived in this state is less than
  // the amount of time needed to extend. If the time
passed is less than the time needed, the legs will continue
to extend.
  // If the time passed is over the time needed to
extend, then the motors shut off.
```

```
        if ( (time_cur - time_start) < time_halfway_extend
){
          PORTC = fwd;
           PORTL = fwd;
           PORTA = fwd;
          } else if ( (time_cur - time_start) >
time_halfway_extend ){
           PORTC = 0b00000000;
           PORTL = 0b00000000;
           PORTA = 0b00000000;
           state = 3;
           time_start = time_cur;
        }
      break;

      case 3:
      // Send out the signals to read the distances, find
the max and min distance from sensors,
      // find the midplane between those two points (where
the closest point is reference z = 0),
      // calculate distance to move the legs, move the legs
for the correct time
```

```
         // This block: Sends out function to read the
distance from ultrasonic sensor
        sendTrigger();
         sendTrigger_2();
         sendTrigger_3();

           // This block: Calculates the average of all
readings
        // Used for testing on a flat surface, 0 if sensors
are level
         dist_avg = (distance + distance_2 + distance_3) /
3; // holds average of all readings

         // This block: Variable to hold the distance read
from each sensor
        dist1_hold = distance;
         dist2_hold = distance_2;
         dist3_hold = distance_3;

        // This block: Variable     to hold the TOTAL SUM of
ALL distance sensor readings
```

dist1_sum = dist1_sum +   dist1_hold;

dist2_sum = dist2_sum +   dist3_hold;

dist3_sum = dist3_sum +   dist3_hold;

// This block: Calculates the average reading for

each sensor from ALL READINGS TAKEN

dist1_avg = (dist1_sum + dist1_hold) / (k+1);

dist2_avg = (dist2_sum + dist2_hold) / (k+1);

dist3_avg = (dist3_sum + dist3_hold) / (k+1);

// METHOD 1 FOR KEEPING ORIENTATION : SUPER SIMPLE

METHOD

//

// This block: This will calculate the distance to

how far they are relative to each other.

//   Will sort out order from farthest to

closest (1 being farthest, 3 being closest).

//   A refrence distance will be made

that is at a mid point between highest and lowest point.

```
        //              A distance will be calculated from
this reference plane for all 3 points.
        //              The distance calculated will be the
amount the legs need to move to keep the plane flat
(assuming the legs
        //              (are currently positioned so they
all equally stick out 8 inches.
        // This block: To find the smallest sensor distance
        // Compares each average sensor reading to the
other two
        // If the reading is smaller than both the other
readings then it will
        // be labeled as the closest point to the lander
and be stored
        // ** DISTANCE is in REFERENCE to the SENSOR
onboard lander **
        if ( dist1_hold <= dist2_hold && dist1_hold <=
dist3_hold ){
          closest_dist = dist1_hold;
        } else if ( dist2_hold < dist1_hold && dist2_hold <
dist3_hold ){
          closest_dist = dist2_hold;
        } else if ( dist3_avg < dist1_avg && dist3_avg <
dist2_avg){
          closest_dist = dist3_hold;
        }


        // This block: To find the farthest sensor distance
        // Compares each acerage sensor reading to the
other two
        // If the reading is farther than both the other
readings then it will
        // be labeled as the farthest point to the lander
        // ** DISTANCE is in REFERENCE to the SENSOR
onboard lander **
        if ( dist1_hold >= dist2_hold && dist1_hold >=
dist3_hold ){
          farthest_dist = dist1_hold;
        } else if ( dist2_hold > dist1_hold && dist2_hold >
dist3_hold ){
          farthest_dist = dist2_hold;
        } else if ( dist3_hold > dist1_hold && dist3_hold >
dist2_hold ){
          farthest_dist = dist3_hold;
        }


        // This block: Sets the new reference point ( z_ref
= 0 ) to be the closest point
        // to the lander
        // ** DISTANCE is in REFERENCE to the CLOSEST POINT
to lander **
        z_ref_leg1 = dist1_hold - closest_dist;
        z_ref_leg2 = dist2_hold - closest_dist;
        z_ref_leg3 = dist3_hold - closest_dist;


        // This block: Find the midplane between the
highest and lowest point
        // ** DISTANCE is in REFERENCE to the CLOSEST POINT
to lander **
        z_ref_midplane = (farthest_dist - closest_dist) /
2; // Reference (z=0) is closest dist, this is a POSITIVE
DISTANCE
                                        //
that is BELOW REFERENCE
                                        //
Hence, POSITIVE Z AXIS POINTS DOWN FROM CLOESEST POINT
```

```
        // This block: Find the distance from the midplane
to each point
        // If the DISTANCE is POSITIVE then the linear
actuators EXTEND ( MOVE FWD )
        // If the DISTANCE is NEGATIVE then the lienar
actuators RETRACT ( MOVE REV )
        // ** DISTANCE is in REFERENCE to the MIDPLANE
found from REFERENCE TO CLOSEST POINT to lander **
        move_leg1 = z_ref_leg1 - z_ref_midplane;
        move_leg2 = z_ref_leg2 - z_ref_midplane;
         move_leg3 = z_ref_leg3 - z_ref_midplane;

         move_leg1_sum = move_leg1_sum + move_leg1;
         move_leg2_sum = move_leg2_sum + move_leg2;
         move_leg3_sum = move_leg3_sum + move_leg3;

         move_leg1_avg = (move_leg1_avg + move_leg1_sum) /
(k+1);
         move_leg2_avg = (move_leg2_avg + move_leg2_sum) /
(k+1);
         move_leg3_avg = (move_leg3_avg + move_leg3_sum) /
```

```
(k+1);


          // This block: Finds the time needed to move each
leg
         // The linear actuators extend at 1 cm/s
("speed_act" variable)
         // time_leg = move_leg/speed_act = cm/(cm/s) =
cm*(s/cm) = s
         time_leg1 = abs( move_leg1 / speed_act );
         time_leg2 = abs( move_leg2 / speed_act );
         time_leg3 = abs( move_leg3 / speed_act );


         // This block: Will check if over 100 distances
have been averaged (reading "k" variable)
        //              and then will wait for the button to
be pushed to adjust legs and keep reading
        //              distances while waiting for the
button push
        if ( k>100 ){
          state = 4;
          time_start = time_cur;
        }
```

```
    break;


    case 4:
      // Leg1 PORTC
      // Leg2 PORTL
      // Leg3 PORTA
      // Leg1 Block Movement
       // This block: Will compare the time start from
case 3 to the current time and find difference,
       // This difference is then compared with how long
it should take to get the leg to correct poistion
       // If the difference in time is less than the time
it should take then the leg will move, if it is over
      // the leg will stop
       if ( (time_cur - time_start) < time_leg1 ){
         if ( move_leg1 > 0 ){
           PORTC = fwd;
         } else if ( move_leg1 < 0 ){
           PORTC = rev;
         } else if ( move_leg1 == 0){
           PORTC = 0b00000000;
         }
```

```
 } else if ( (time_cur - time_start) > time_leg1 ){
   PORTC = 0b00000000;
   leg1_adjust_done = true;
}

// Leg2 Block Movement
if ( (time_cur - time_start) < time_leg2 ){
  if ( move_leg2 > 0 ){
    PORTL = fwd;
  } else if ( move_leg2 < 0 ){
    PORTL = rev;
  } else if ( move_leg2 == 0){
    PORTL = 0b00000000;
  }
} else if ( (time_cur - time_start) > time_leg2 ){
  PORTL = 0b00000000;
    leg2_adjust_done = true;
}

//Leg3 Block Movement
if ( (time_cur - time_start) < time_leg3 ){
  if ( move_leg3 > 0 ){
          PORTA = fwd;
        } else if ( move_leg3 < 0 ){
          PORTA = rev;
        } else if ( move_leg3 == 0){
          PORTA = 0b00000000;
        }
      } else if ( (time_cur - time_start) > time_leg3 ){
        PORTA = 0b00000000;
        leg3_adjust_done = true;
      }


      if ( leg1_adjust_done == true && leg2_adjust_done
== true && leg3_adjust_done == true ){
        state = 5;
      }
    break;
  case 5:


    break;
```

```
}




int poop1 = digitalRead(18);

int poop2 = digitalRead(2);

int poop3 = digitalRead(3);


// TESTING PRINTS

//

// This is for testing the buttons and the debounce

/*

 Serial.print(time_cur);

 Serial.print("\t");

 Serial.print(state);

 Serial.print("\t");

 Serial.print(reset_button);

 Serial.print("\t");

 Serial.print(adjust_leg_button);

 Serial.print("\t");
```

```
   Serial.print(debounce_time_start);
   Serial.print("\t");
   Serial.println((time_cur - debounce_time_start));
  */
  //
  // This is for testing the reset state and the
microswitches
  /*
   Serial.print(time_cur);
   Serial.print("\t");
   Serial.print(state);
   Serial.print("\t");
   Serial.print(reset_button);
   Serial.print("\t");
   Serial.print(leg_adjust_button);
   Serial.print("\t");
   Serial.print(debounce_time_start);
   Serial.print("\t");
   Serial.println(time_cur - debounce_time_start);
  */
  // This is for testing the distance sensors
/*
```

```
Serial.print(time_cur);

Serial.print("\t");

Serial.print(state);

Serial.print("\t");

Serial.print(distance);

Serial.print("\t");

Serial.print(distance_2);

Serial.print("\t");

Serial.print(distance_3);

Serial.print("\t");

Serial.println(dist_avg);

*/

// This is for testing the average of distances

/*

Serial.print(time_cur);

Serial.print("\t");

Serial.print(state);

Serial.print("\t");

Serial.print(dist1_hold);

Serial.print("\t");

Serial.print(dist1_sum);

Serial.print("\t");
Serial.print(dist1_avg);
Serial.print("\t");
Serial.print(dist2_hold);
Serial.print("\t");
Serial.print(dist2_sum);
Serial.print("\t");
Serial.print(dist2_avg);
Serial.print("\t");
Serial.print(dist3_hold);
Serial.print("\t");
Serial.print(dist3_sum);
Serial.print("\t");
Serial.print(dist3_avg);
  Serial.print("\t");
  Serial.println(k);
 */
 // This is for testing the farthest dist, closest dist,
z-midplane, z-ref for legs, and movement
 /*
  Serial.print(time_cur);
  Serial.print("\t");
  Serial.print(state);
  Serial.print("\t");

  Serial.print(dist1_avg);

  Serial.print("\t");

  Serial.print(dist2_avg);

  Serial.print("\t");

  Serial.print(dist3_avg);

  Serial.print("\t");

  Serial.print(closest_dist);

  Serial.print("\t");

  Serial.print(farthest_dist);

  Serial.print("\t");

  Serial.print(z_ref_leg1);

  Serial.print("\t");

  Serial.print(z_ref_leg2);

  Serial.print("\t");

  Serial.print(z_ref_leg3);

  Serial.print("\t");

  Serial.print(z_ref_midplane);

  Serial.print("\t");

  Serial.print(move_leg1);

  Serial.print("\t");

  Serial.print(move_leg1_avg);
```

```
  Serial.print("\t");

  Serial.print(move_leg2);

  Serial.print("\t");

  Serial.print(move_leg2_avg);

  Serial.print("\t");

  Serial.print(move_leg3);

  Serial.print("\t");

  Serial.println(move_leg3_avg);

 */

 // This is for seeing the farthest dist, closest dist,

z-midplane, z-ref for legs, and movement EASIER TO SEE


 Serial.print(time_cur);

 Serial.print("\t");

 Serial.print(state);

 Serial.print("\t");

 Serial.print(dist1_avg);

 Serial.print("\t");

 Serial.print(dist2_avg);

 Serial.print("\t");

 Serial.print(dist3_avg);

 Serial.print("\t");
```

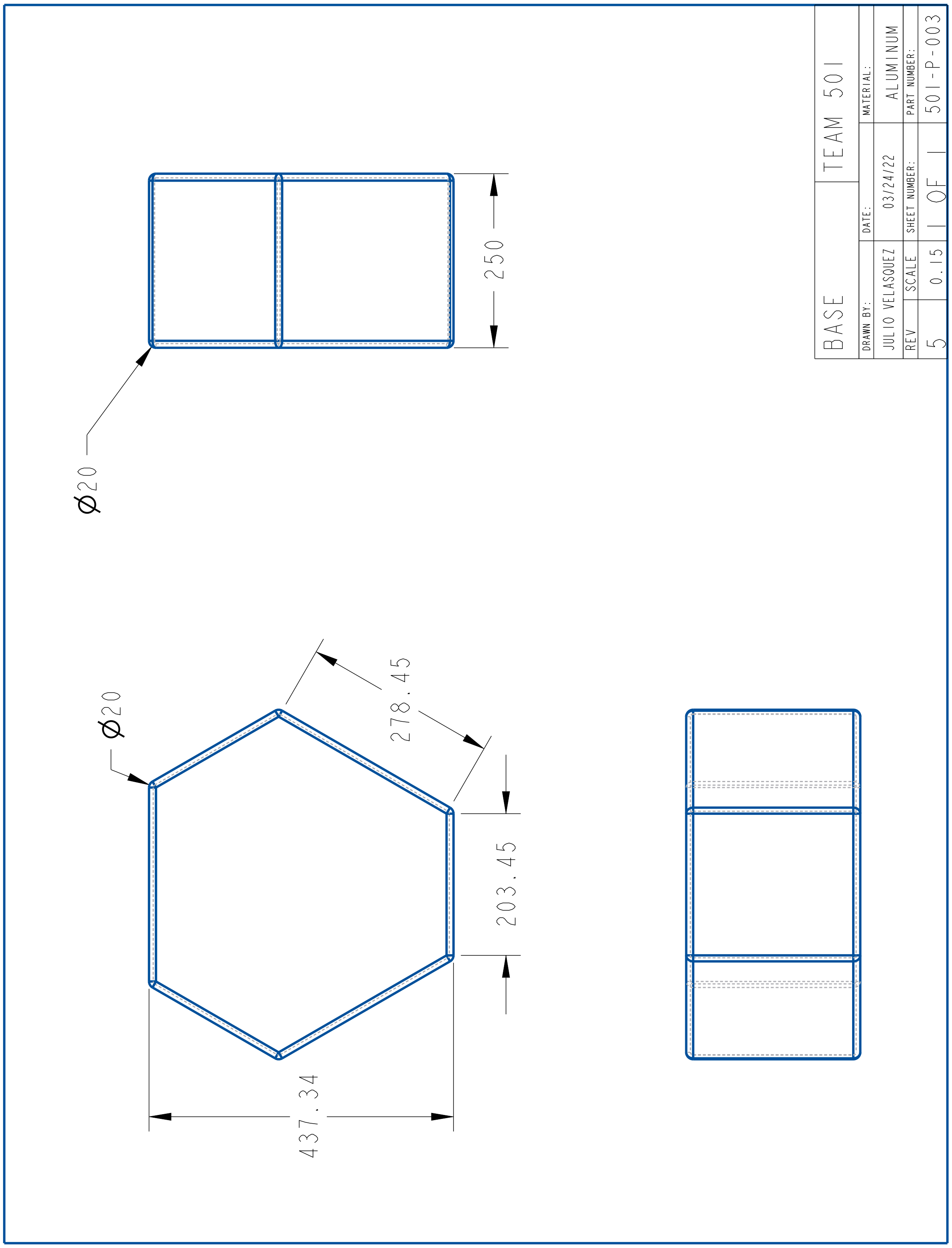
```
  Serial.print(closest_dist);
  Serial.print("\t");
  Serial.print(farthest_dist);
  Serial.print("\t");
  Serial.print(z_ref_leg1);
  Serial.print("\t");
  Serial.print(z_ref_leg2);
  Serial.print("\t");
  Serial.print(z_ref_leg3);
  Serial.print("\t");
  Serial.print(z_ref_midplane);
  Serial.print("\t");
  Serial.print(move_leg1_avg);;
  Serial.print("\t");
  Serial.print(move_leg2_avg);
  Serial.print("\t");
  Serial.println(move_leg3_avg);
}


// Timer1 ISR Function
// Counts every 0.1 seconds
ISR(TIMER1_COMPA_vect){
    time_cur = time_cur + 0.1;


}


// Calculates the distance and prevents bounce

void my_ISR(){

  if( (lastEchoState == 0) && (PIND & 0b00000001) ){

//change from zero to one

    lastEchoState = 1;

    timer_1 = micros();

  }else if ( (lastEchoState == 1) && !(PIND & 0b00000001)

){ //change from one to zero

    lastEchoState = 0;

    echoWidth = micros() - timer_1; //pulse width in

microseconds

  }


    distance = (echoWidth/2.0)*0.0343; // distance in cm

}


void my_ISR_2(){
  if( (lastEchoState_2 == 0) && (PIND & 0b00000010) ){
//change from zero to one
    lastEchoState_2 = 2;
    timer_2 = micros();
  }else if ( (lastEchoState_2 == 2) && !(PIND & 0b00000010)
){ //change from one to zero
    lastEchoState_2 = 0;
    echoWidth_2 = micros() - timer_2; //pulse width in
microseconds
  }

    distance_2 = (echoWidth_2/2.0)*0.0343; // distance in cm
}

void my_ISR_3(){
  if( (lastEchoState_3 == 0) && (PIND & 0b00000100) ){
//change from zero to one
    lastEchoState_3 = 4;
    timer_3 = micros();
  }else if ( (lastEchoState_3 == 4) && !(PIND & 0b00000100)
){ //change from one to zero
    lastEchoState_3 = 0;
    echoWidth_3 = micros() - timer_3; //pulse width in
microseconds
  }

    distance_3 = (echoWidth_3/2.0)*0.0343; // distance in cm
}

// function to senda 10 us trigger pulse for distance
sensors
void sendTrigger(){
  PORTB = 0b00000000; // start low
   delayMicroseconds(2);
  PORTB = 0b00000001; // high for 10 usec
   delayMicroseconds(10);
  PORTB = 0b00000000; // start low back to low
}

void sendTrigger_2(){
  PORTB = 0b00000000; // start low
   delayMicroseconds(2);
  PORTB = 0b00000010; // high for 10 usec
   delayMicroseconds(10);
```

```
    PORTB = 0b00000000; // start low back to low
}


void sendTrigger_3(){
  PORTB = 0b00000000; // start low
   delayMicroseconds(2);
  PORTB = 0b00000100; // high for 10 usec
   delayMicroseconds(10);
  PORTB = 0b00000000; // start low back to low
}




// When right microswitch is hit it will change DC motor
direction to left
void ISR_leg1_off(){
  PORTC = 0b00000000; //turns leg 1 off
  leg1_reset = true;
}


// When left microswitch is hit it will change DC motor
direction to right
```

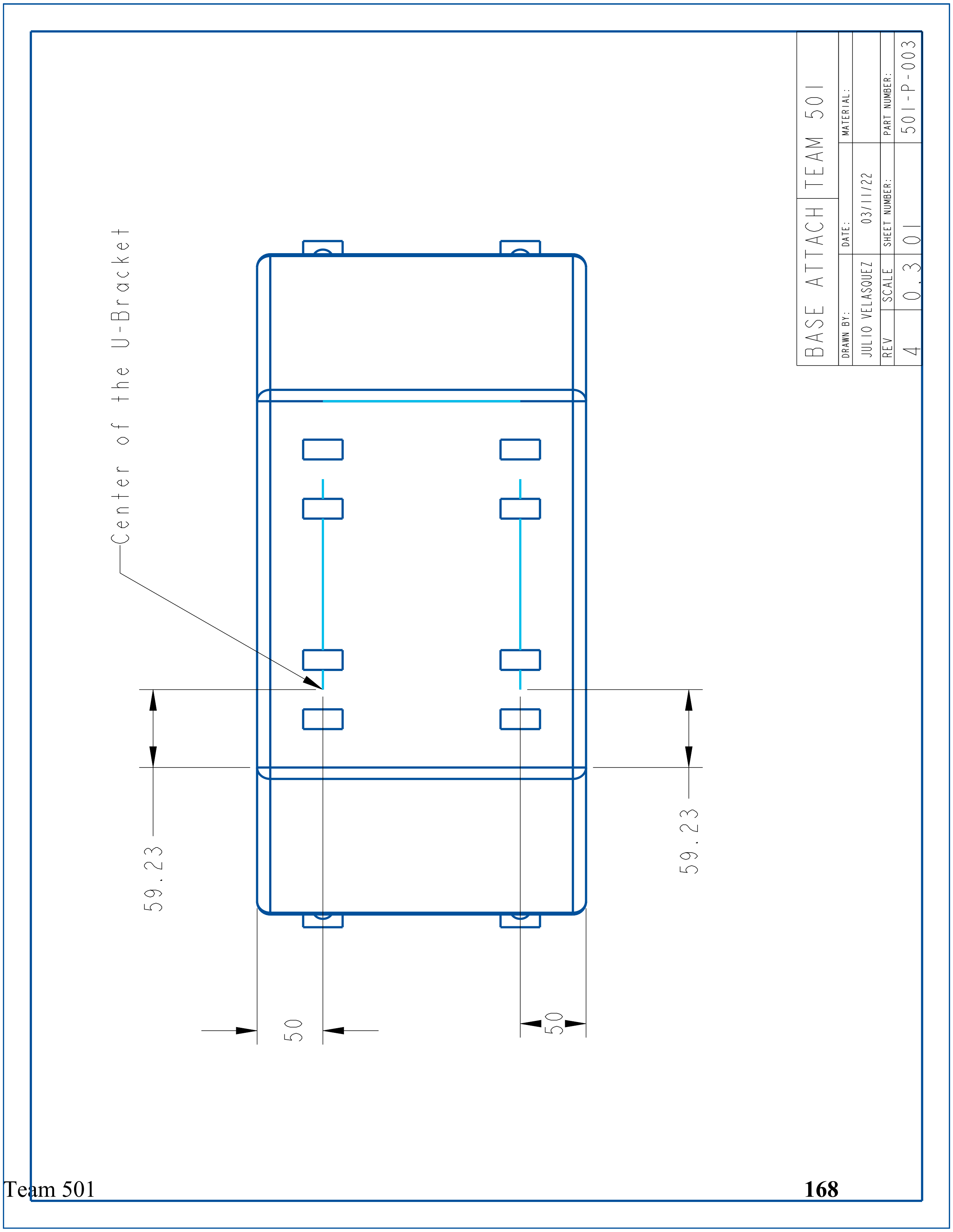
```
void ISR_leg2_off(){
  PORTL = 0b00000000; //turns leg 2 off
  leg2_reset = 1;
}
// When right microswitch is hit it will change DC motor
direction to left
void ISR_leg3_off(){
  PORTA = 0b00000000; //turns leg 3 off
  leg3_reset = true;
}
```

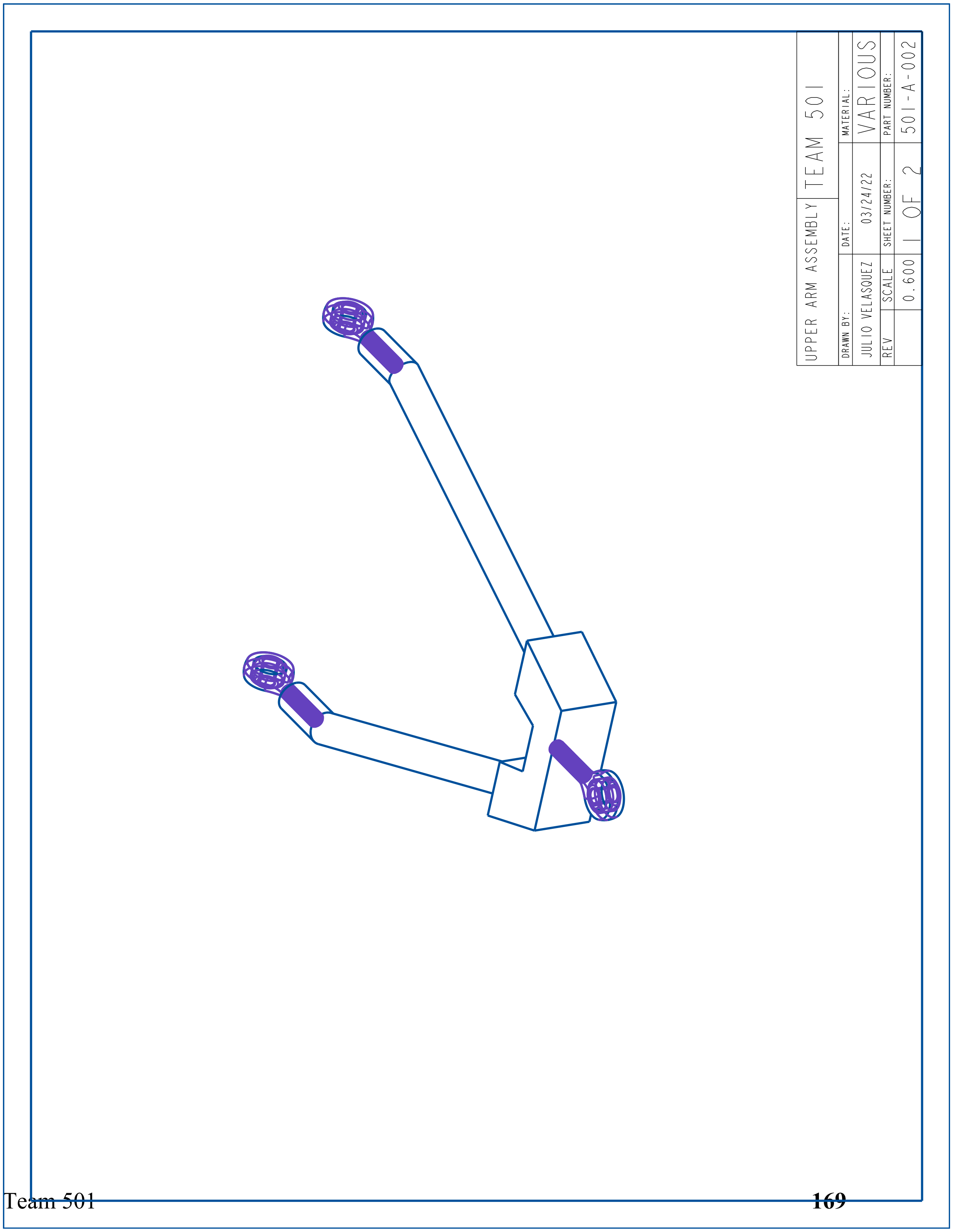
Appendix J: Engineering Drawings

(Note: All dimensions on CREO drawings are in millimeters)

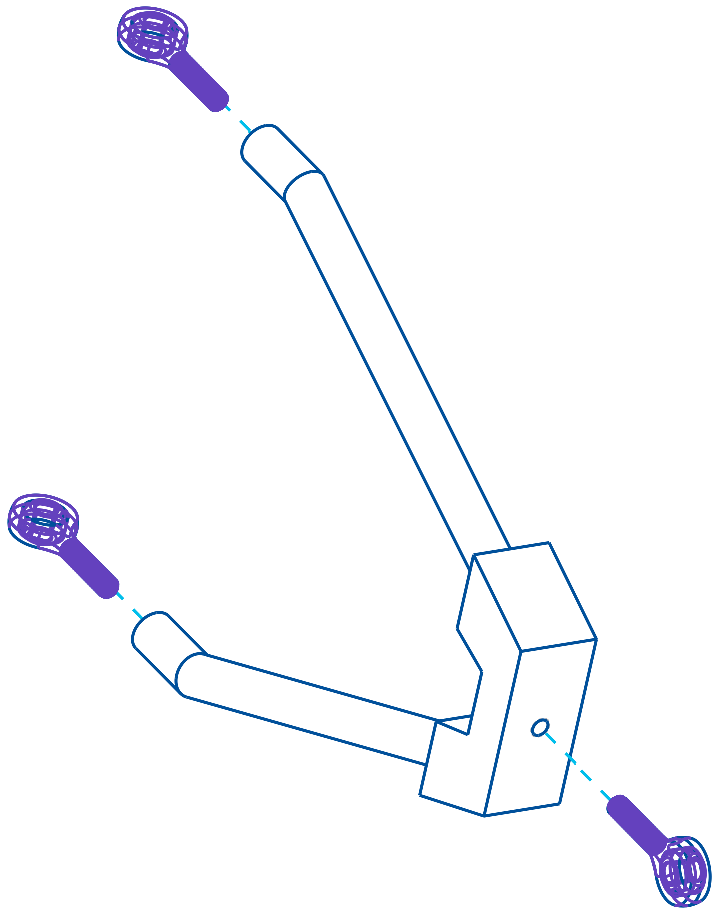


BASE
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
ALUMINUM
REV
5
SCALE
0.15
SHEET NUMBER:
1 OF 1
PART NUMBER:
501-P-003
250
Ø20
Ø20
278.45
203.45
437.34


Center of the U-Bracket
59.23
59.23
50
50
BASE ATTACH TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/11/22
MATERIAL:
REV
4
SCALE
0.3
SHEET NUMBER:
01
PART NUMBER:
501-P-003



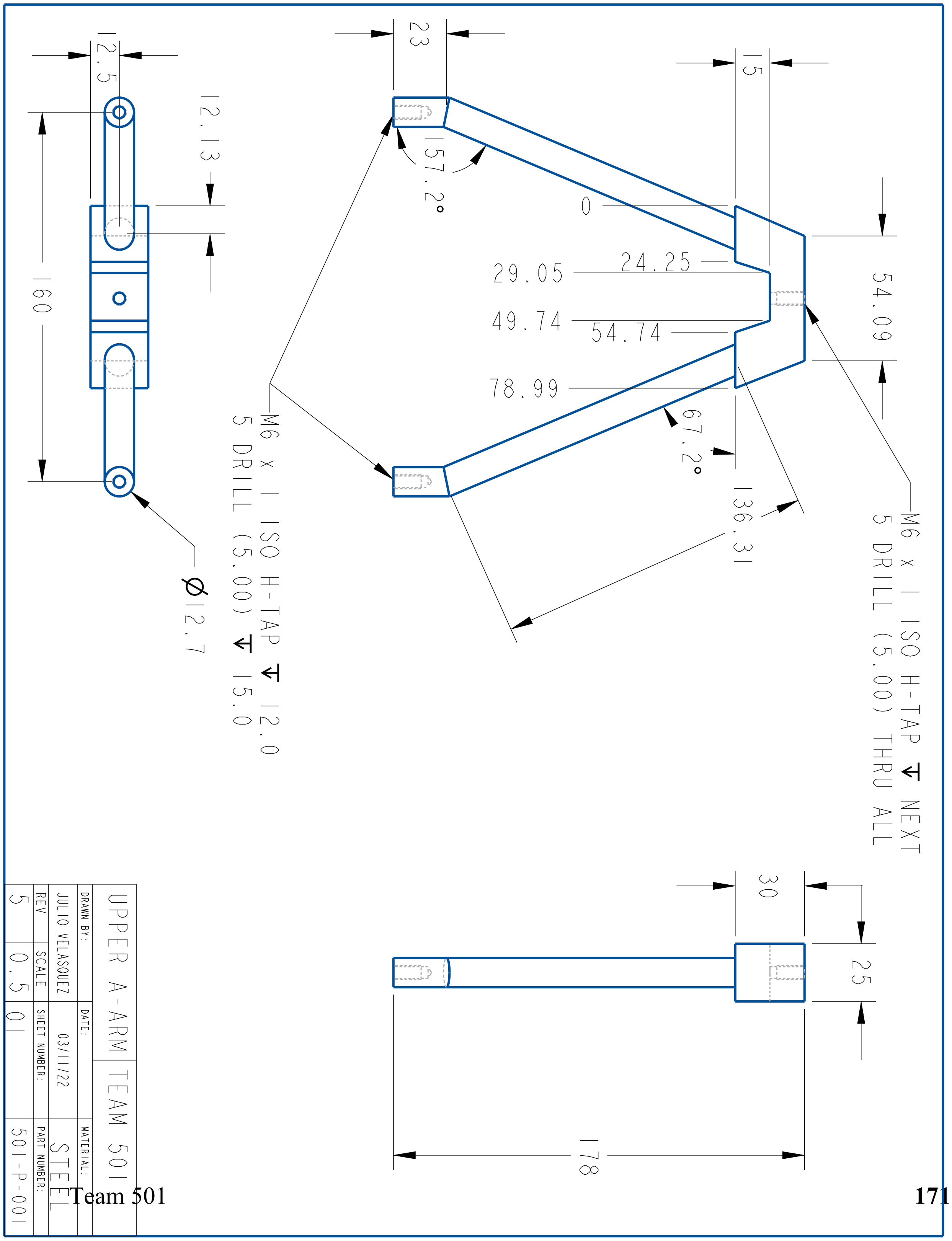
UPPER ARM ASSEMBLY		TEAM 501	
DRAWN BY: JULIO VELASQUEZ	DATE: 03/24/22		MATERIAL: VARIOUS
REV	SCALE 0.600	SHEET NUMBER: 1 OF 2	PART NUMBER: 501-A-002



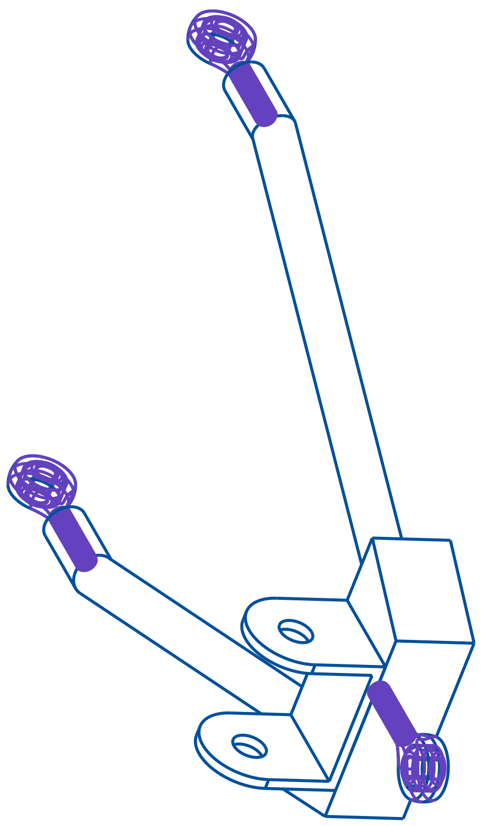
UPPER ARM ASSEMBLY

TEAM 501

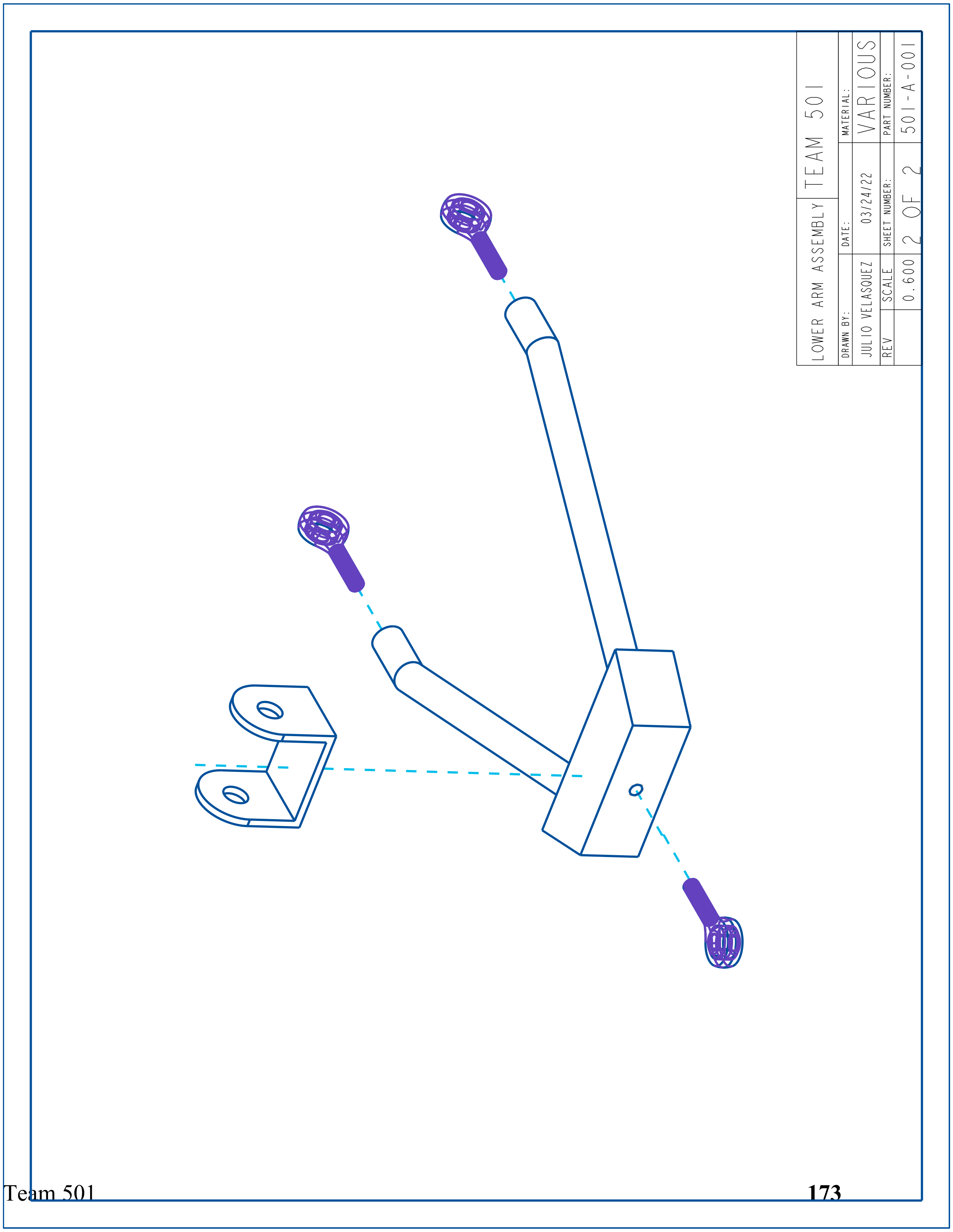
DRAWN BY:	JULIO VELASQUEZ
DATE:	03/24/22
MATERIAL:	VARIOUS
REV	
SCALE	0.600
SHEET NUMBER:	2 OF 2
PART NUMBER:	501-A-002

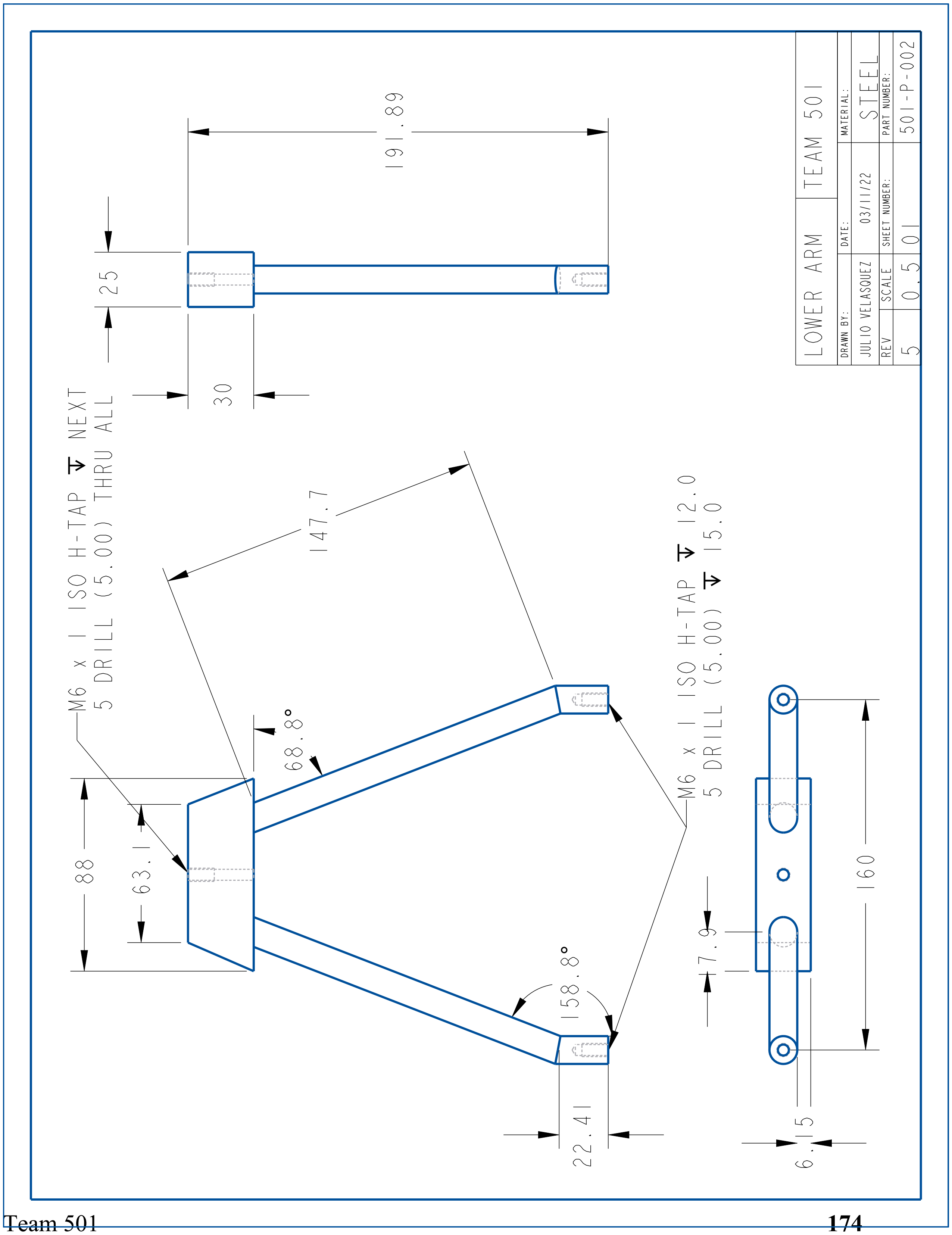


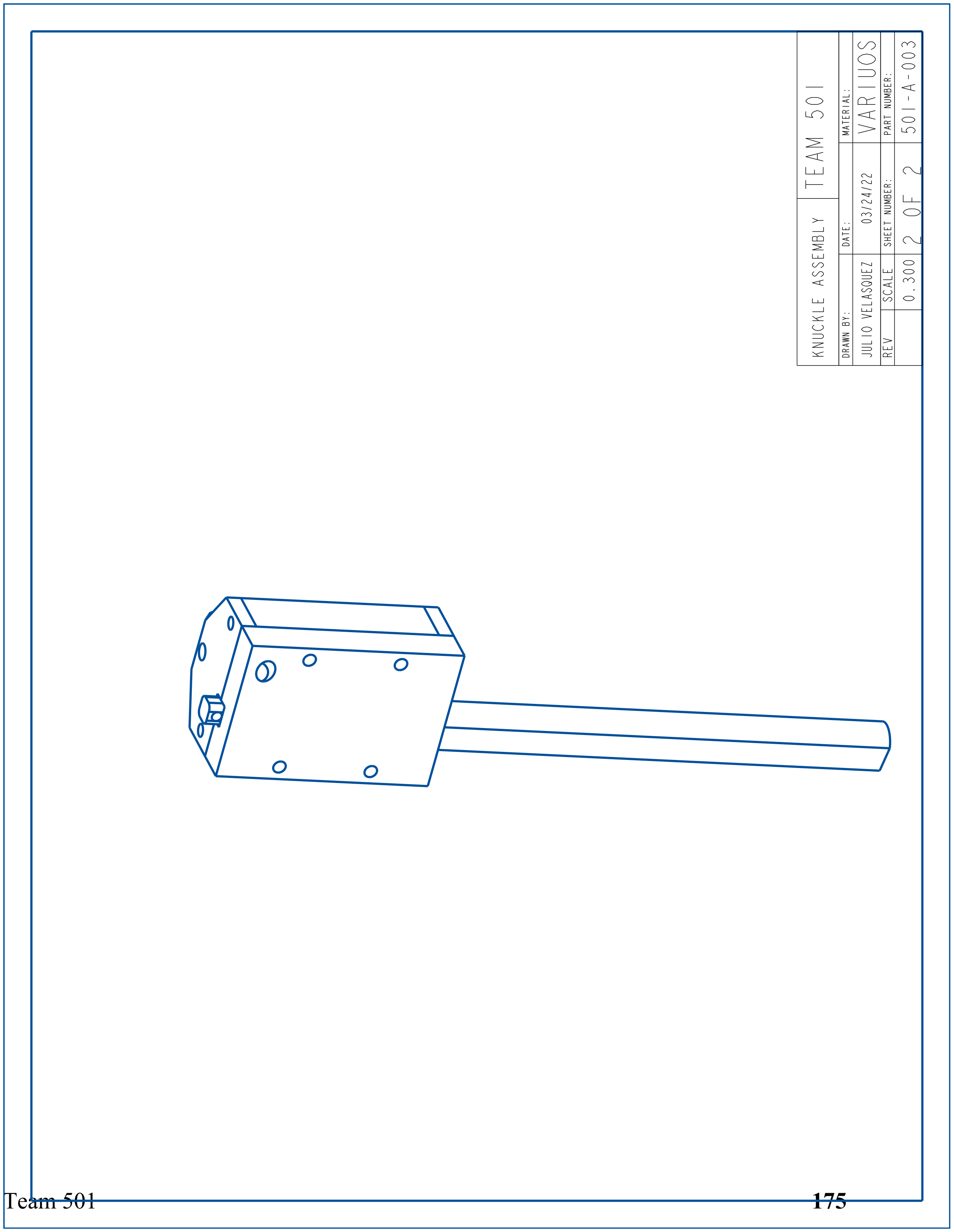
23
15
12.5
12.13
157.2°
0
24.25
29.05
49.74
54.74
78.99
54.09
67.2°
136.31
160
Ø12.7
M6 x 1 ISO H-TAP ⊤ 12.0
5 DRILL (5.00) ⊤ 15.0
M6 x 1 ISO H-TAP ⊤ NEXT
5 DRILL (5.00) THRU ALL
30
25
178
UPPER A-ARM
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/11/22
MATERIAL:
STEEL
REV
5
SCALE
0.5
SHEET NUMBER:
01
PART NUMBER:
501-P-001

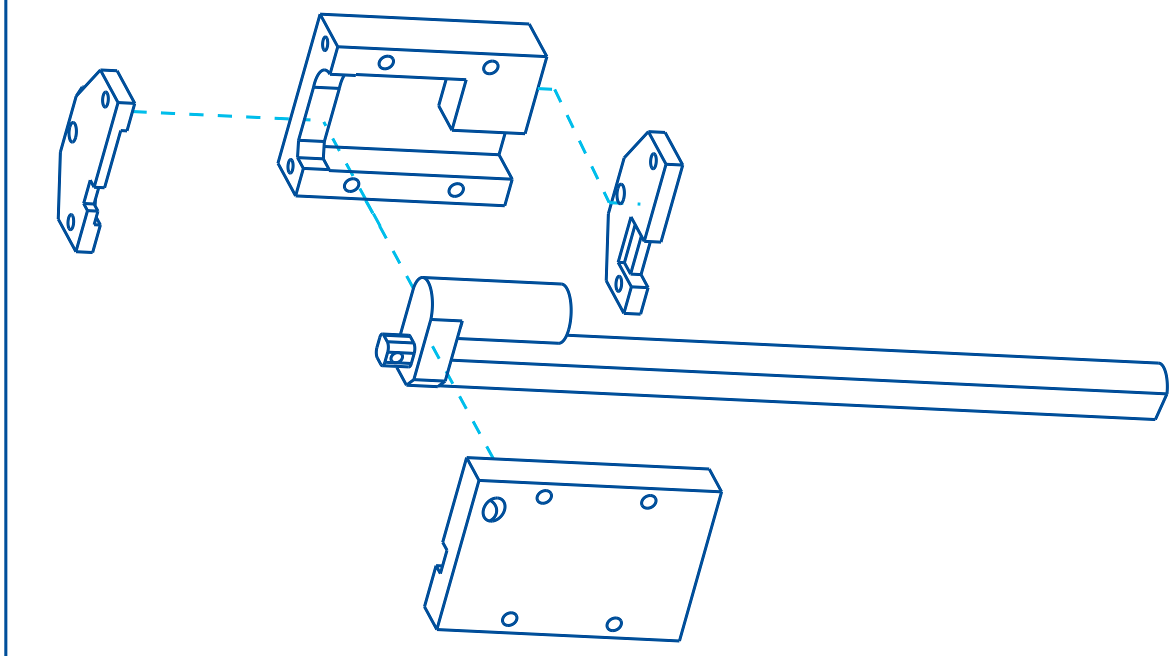


LOWER ARM ASSEMBLY		TEAM 501
DRAWN BY: JULIO VELASQUEZ	DATE: 03/24/22	MATERIAL: VARIOUS
REV	SCALE 0.600	SHEET NUMBER: 1 OF 2	PART NUMBER: 501-A-001

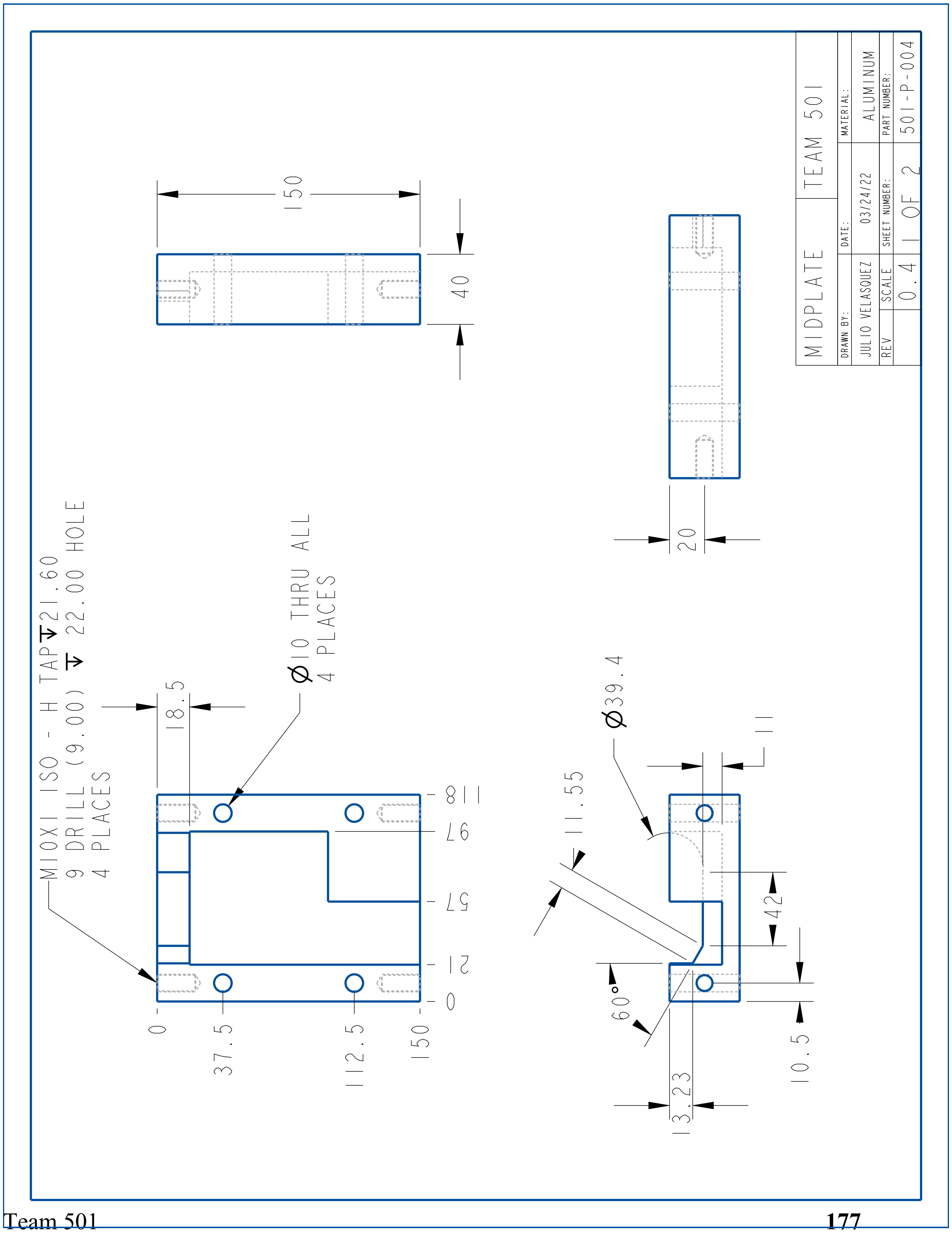

LOWER ARM ASSEMBLY
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
VARIOUS
REV
SCALE
0.600
SHEET NUMBER:
2 OF 2
PART NUMBER:
501-A-001


191.89
25
30
M6 x 1 ISO H-TAP ⊽ NEXT
5 DRILL (5.00) THRU ALL
147.7
68.8°
88
63.1
M6 x 1 ISO H-TAP ⊽ 12.0
5 DRILL (5.00) ⊽ 15.0
158.8°
22.41
17.9
160
6.15
LOWER ARM
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/11/22
MATERIAL:
STEEL
REV
5
SCALE
0.5
SHEET NUMBER:
01
PART NUMBER:
501-P-002

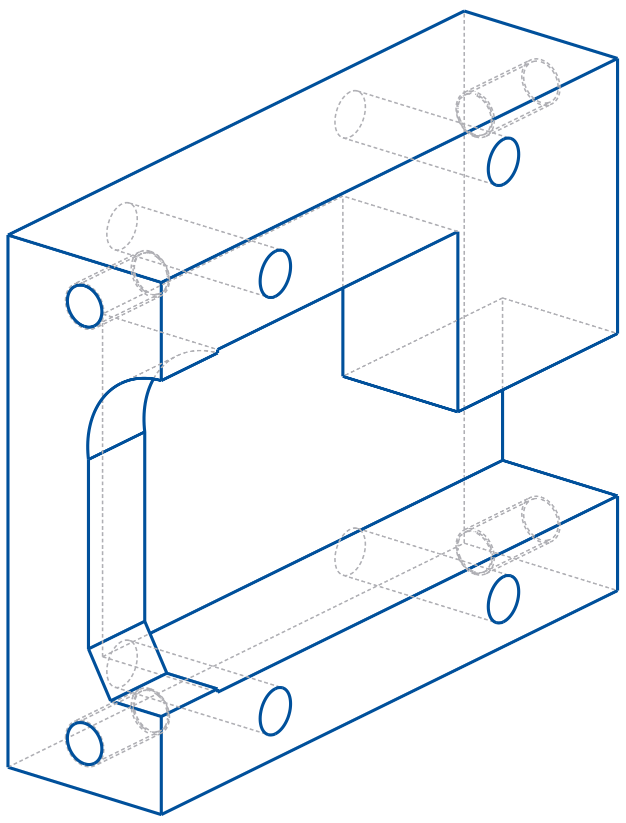

KNUCKLE ASSEMBLY
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
VARIUOS
REV
SCALE
0.300
SHEET NUMBER:
2 OF 2
PART NUMBER:
501-A-003

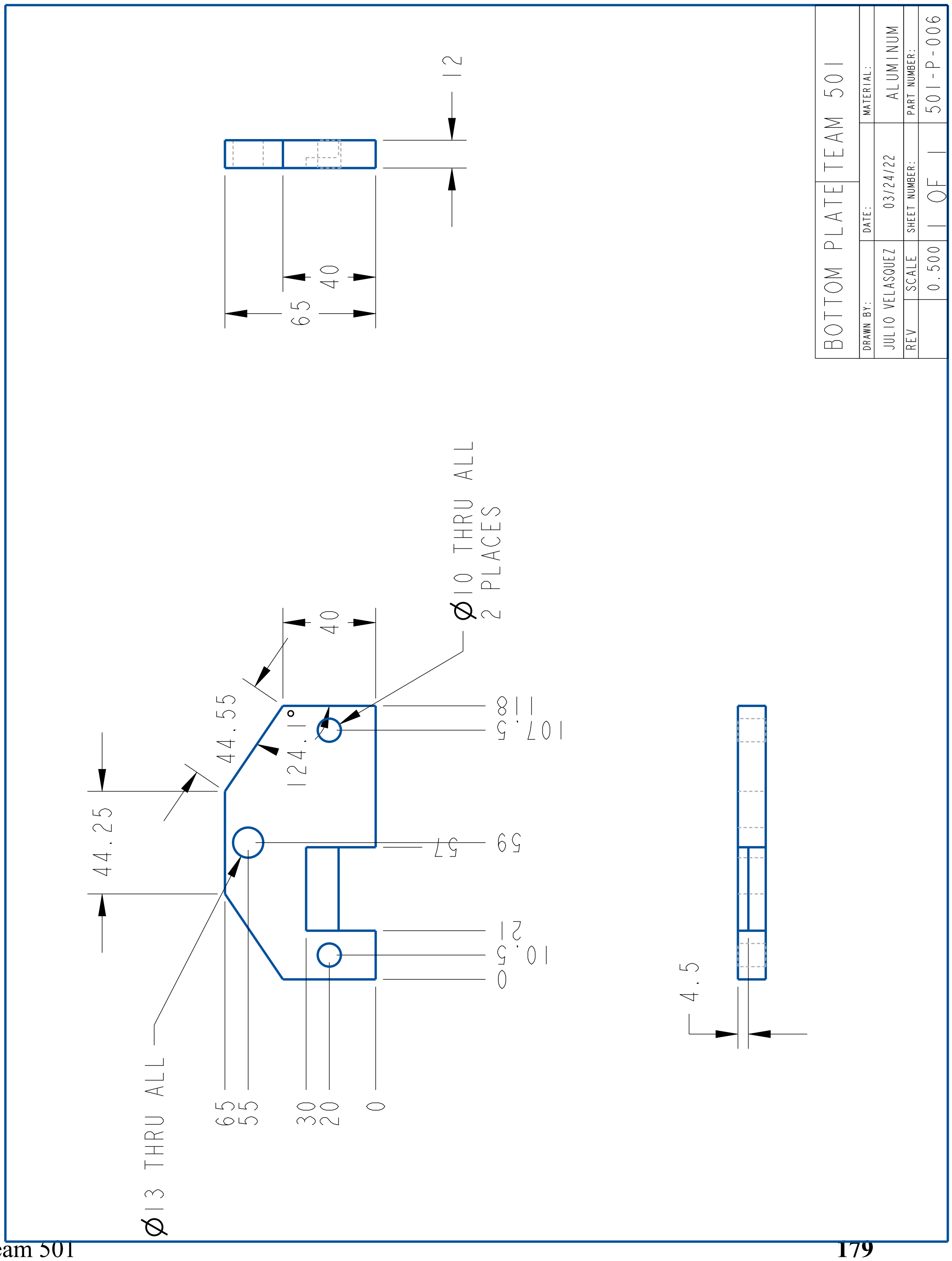


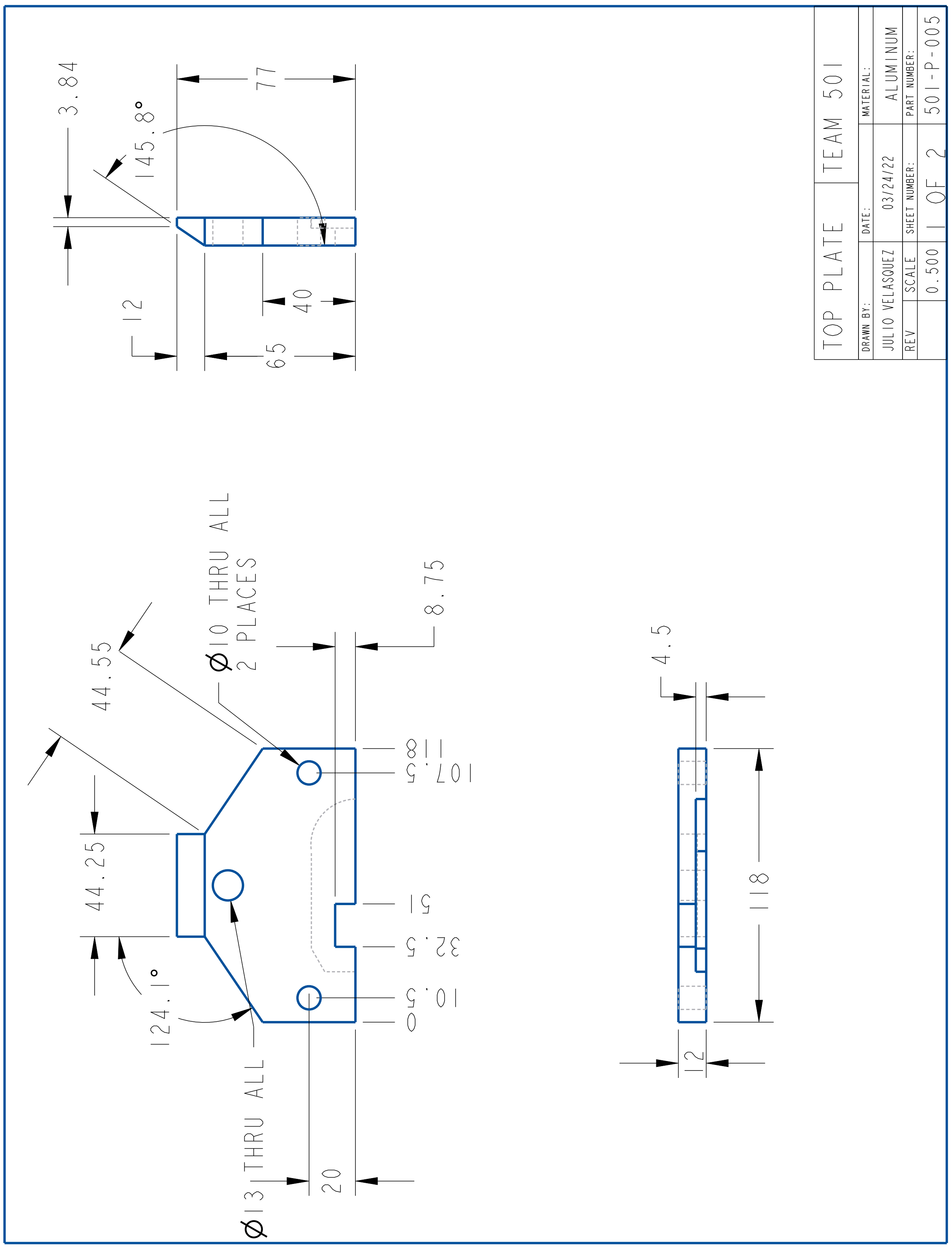
KNUCKLE ASSEMBLY		TEAM 501	
DRAWN BY: JULIO VELASQUEZ	DATE: 03/24/22	MATERIAL: VARIOUS	
REV	SCALE 0.250	SHEET NUMBER: 2 OF 2	PART NUMBER: 501-A-003

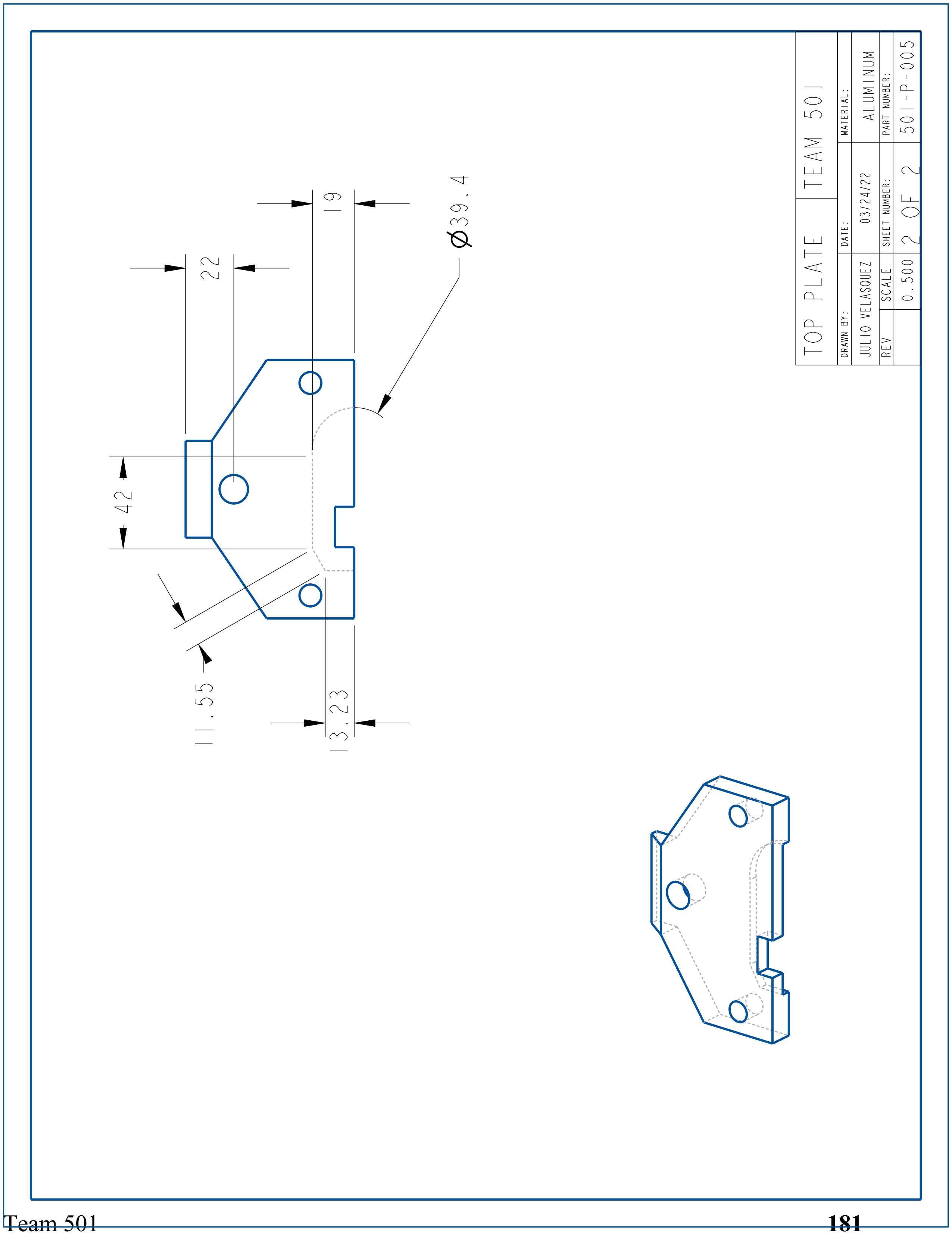

MIDPLATE
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
ALUMINUM
REV
SCALE
0.4
SHEET NUMBER:
1 OF 2
PART NUMBER:
501-P-004
M10X1 ISO - H TAP ↧ 21.60
9 DRILL (9.00) ↧ 22.00 HOLE
4 PLACES
Ø10 THRU ALL
4 PLACES
Ø39.4
150
40
20
18.5
11.55
11
42
60°
13.23
10.5
0
37.5
112.5
150
0
21
57
97
118

MIDPLATE		TEAM 501
DRAWN BY: JULIO VELASQUEZ	DATE: 03/24/22	MATERIAL: ALUMINUM
REV	SCALE 0.800	SHEET NUMBER: 2 OF 2	PART NUMBER: 501-P-004




BOTTOM PLATE TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
ALUMINUM
REV
SCALE
0.500
SHEET NUMBER:
1 OF 1
PART NUMBER:
501-P-006
12
40
65
Ø10 THRU ALL
2 PLACES
Ø13 THRU ALL
44.55
44.25
124.1°
118
107.5
59
57
12
10.5
0
65
55
30
20
0
4.5


TOP PLATE
TEAM 501
DRAWN BY:
JULIO VELASQUEZ
DATE:
03/24/22
MATERIAL:
ALUMINUM
REV
SCALE
0.500
SHEET NUMBER:
1 OF 2
PART NUMBER:
501-P-005
3.84
145.8°
77
12
40
65
Ø10 THRU ALL
2 PLACES
44.55
8.75
44.25
124.1°
Ø13 THRU ALL
20
118
107.5
51
32.5
10.5
0
4.5
12
118



TOP PLATE		TEAM 501
DRAWN BY: JULIO VELASQUEZ	DATE: 03/24/22	MATERIAL: ALUMINUM
REV	SCALE 0.500	SHEET NUMBER: 2 OF 2	PART NUMBER: 501-P-005

Appendix K: Calculations

The weight which the lander needs to support is calculated by:

$$Weight = mass * gravity = 23kg * 9.81\frac{m}{s} = 225.63N$$

The impact energy that the lander needs to be able to handle and dissipate:

$$Impact\ Energy = \frac{1}{2} * mass * velocity^2 = \frac{1}{2}(23kg)\left(0.92\frac{m}{s^2}\right)^2 = 9.74\,J$$

The counterweight mass needed (m_1) was calculated using pulley equations:

$$m_1 = \frac{m_2(1 - {}^{a}/_{g})}{(1 + {}^{a}/_{g})}$$

Where the gravity on the specified planet is given as g = 9.81 m/s^2 for Earth, the mass of the lander (m_2) is measured, and the acceleration (a) is calculated by:

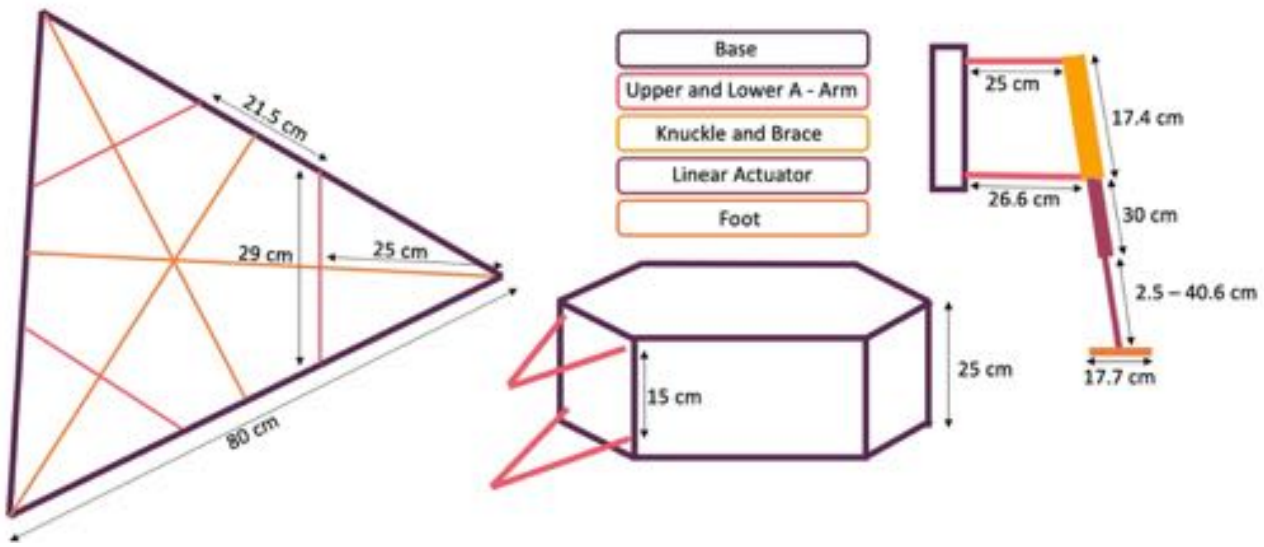
$$a = \frac{1}{2h}$$

Where h is the height from which it is dropped. The calculation for the counterweight needed is shown below:

$$a = \frac{1}{2h} = \frac{1}{2(1m)} = \frac{0.5m}{s^2}$$

$$m_1 = \frac{m_2(1 - {}^{a}/_{g})}{(1 + {}^{a}/_{g})} = \frac{23kg(1 - {}^{0.5\frac{m}{s^2}}/_{9.81\frac{m}{s^2}})}{(1 + {}^{0.5\frac{m}{s^2}}/_{9.81\frac{m}{s^2}})} = 20.77kg$$

The size of the lander was calculated geometrically by determining the center of gravity and the instant center:



The length of the bottom A-arm shown in the figure above was calculated by trigonometry with the 5 degree offset between the leg and the base side.

References

There are no sources in the current document.