Department of Electronic & Telecommunication Engineering

University of Moratuwa

Sri Lanka

Design Document

Zero gravity lifting device

Group Members:

210005H	Abeyrathna S M S M B
210015M	Abeywardhane R N
210321X	Kumarasinghe R D
210687X	Weerasinghe C N

06/07/2024

Contents

1 General Overview and Project Background

1.1 Introduction

The Zero Gravity Lifting device represents a groundbreaking solution for efficient and safe material handling in industrial environments. This design document comprehensively outlines the development of this pioneering device, detailing the methodical planning, engineering factors considered, and creative approaches employed throughout the process. It encapsulates the combined work of our multi-disciplinary team, leveraging expertise spanning mechanical engineering, electronics, software development, and human-centered design principles. The design document delves into the core aspects that have shaped the Zero Gravity Lifting device. It covers the general concepts like market research, existing product analysis, design specifications and criteria. It then presents the conceptual designs evaluated for the mechanical structure, electronic control systems, user input mechanisms and sensing technologies. The rationale behind design choices, testing procedures, challenges faced and their solutions are thoroughly documented. Furthermore, it includes final drawings, schematics, renderings and details of the selected design approach. The document concludes with the prototype enclosure design details and acknowledgments. Additionally, it contains a daily log summarizing the key activities and progress made throughout the design and development phases.

1.2 The need of the system

The need for a Zero Gravity Lifting device arises from the significant demand within various industrial sectors for advanced lifting solutions capable of handling heavy, bulky, or irregularly shaped objects efficiently and safely. Traditional lifting equipment, such as cranes and hoists, often falls short in addressing the evolving challenges present in modern manufacturing environments, including limitations in maneuverability, accessibility, and flexibility. These constraints not only hinder operational efficiency but also pose potential safety risks, ergonomic hazards, and productivity bottlenecks.

To alleviate these issues, a Zero Gravity Lifting device offers a transformative solution tailored to the unique requirements of industrial settings. By leveraging innovative technology and design, this lifting device aims to revolutionize material handling processes by:

- Enhancing Efficiency and Safety: By providing precise control and lifting capabilities, the Zero Gravity Lifting device optimizes material handling operations, reducing the risk of workplace accidents and improving overall safety protocols.
- Mitigating Ergonomic Hazards: The device minimizes the physical strain on workers associated with manual lifting tasks, thereby reducing the incidence of musculoskeletal disorders and promoting better workplace ergonomics.
- Streamlining Operations: With its versatility, user-friendliness, and precise lifting capabilities, the device streamlines material handling processes within industrial environments, leading to enhanced productivity, workflow efficiency, and operational performance.

- Driving Innovation and Differentiation: The introduction of the Zero Gravity Lifting device represents a significant innovation in material handling technology, offering a differentiated solution that sets the company apart from traditional lifting equipment providers and positions it as a leader in the industry.
- Providing Tailored Solutions: The adaptability of the Zero Gravity Lifting device allows for customization to suit the unique challenges and requirements of diverse industrial applications, ensuring optimal performance and customer satisfaction.

In summary, the need for a Zero Gravity Lifting device stems from the imperative to address the shortcomings of existing lifting equipment and provide a transformative solution that enhances efficiency, safety, and productivity in industrial material handling processes.

1.3 Existing Product Analysis

We conducted a comprehensive analysis of existing zero-gravity lifting technologies and products available in the market. This analysis focused on industry-leading manufacturers and their cutting-edge solutions, including:

- Gorbel's Easy Arm Intelligent Lifting Arm: Incorporating G-Force lifting technology, this system offers flexible installation options, unmatched precision with speeds less than 1 fpm, force-sensing handle configurations, cable management systems, and a true zero-gravity float mode for precise load manipulation.
- Biotronik's Zero-Gravity Suspended Radiation Protection System: Designed to address the challenges faced by interventionists during fluoroscopic procedures, this system offers adaptable configurations, such as floor units, monorails, hinged swing arms, and combinations thereof, reducing cumulative radiation doses and orthopedic strain.
- INDEVA's Liftronic Series featuring advanced electronic control technology: Incorporating auto-weight sensing, auto-balancing capabilities, responsive electronic control, ergonomic design, remote troubleshooting through the App-Indeva, Industry 4.0 interconnectivity, efficient auto-diagnostic systems, and human extender technology for enhanced capabilities.

1.4 Stakeholder analysis

Understanding the interests and influence levels of these stakeholders is crucial for effective project management and successful implementation of zero gravity lifting technologies.

1.5 User analysis

The primary user group for the Zero Gravity Lifting device comprises industrial workers involved in material handling, assembly, and manufacturing operations across multiple sectors, including automotive, aerospace, heavy machinery, electronics, and healthcare. These users typically fall into the following roles:

- Assembly Line Workers: Responsible for assembling and handling various components, often involving lifting and positioning heavy or awkwardly shaped objects.
- Machine Operators: Tasked with loading and unloading materials from machines, as well as performing maintenance and adjustment tasks that require lifting and manipulation of components.
- Maintenance Technicians: Involved in the repair and servicing of industrial equipment, which frequently necessitates the lifting and positioning of machinery parts and tools.

- Material Handlers: Responsible for the movement and storage of goods, raw materials, and finished products within industrial facilities, requiring the lifting and transport of items of varying weights and sizes.

1.6 Desired Features and Requirements

Based on the identified pain points and user feedback, our research revealed several key features and requirements that industrial workers desire in a Zero Gravity Lifting solution:

- Ergonomic Design: The device should prioritize ergonomics, reducing the physical strain and risk of injury associated with manual lifting and handling tasks. Intuitive Operation: The user interface and control system should be intuitive and user-friendly, minimizing the need for extensive training and enabling seamless integration into existing workflows.
- Adaptability and Flexibility: The device should be capable of handling a wide range of materials with varying sizes, shapes, and weights, while also being able to operate in confined or cluttered workspaces.
- Precision and Control: Precise control over the lifting and positioning of materials is essential to ensure accuracy, prevent damage to components, and maintain a safe working environment.
- Durability and Robustness: Industrial environments can be harsh and demanding, necessitating a durable and robust design that can withstand the rigors of daily use and minimize maintenance requirements.
- Integration and Compatibility: The Zero Gravity Lifting device should seamlessly integrate with existing infrastructure, machinery, and processes within industrial facilities, minimizing disruptions and facilitating a smooth transition.
- Safety Features: Incorporating appropriate safety features, such as overload protection, emergency stops, and compliance with relevant industry standards and regulations, is crucial to ensuring the safe operation of the device and protecting workers.

1.7 Market Segment Analysis

The Zero Gravity Lifting device is designed to cater to a broad spectrum of industries that require efficient and safe material handling solutions. The primary market segments identified include:

- Automotive Industry: Assembly lines, part handling, and maintenance operations involving heavy components, engines, and body panels.
- Aerospace Industry: Manufacturing, assembly, and maintenance of aircraft components, including fuselages, wings, and intricate mechanical systems.
- Heavy Machinery: Material handling in the production, assembly, and maintenance of construction equipment, agricultural machinery, and industrial tools.
- Electronics Manufacturing: Assembly and handling of delicate electronic components, circuit boards, and finished products.
- Healthcare: Patient lifting and positioning, as well as handling and moving medical equipment in hospitals, clinics, and rehabilitation centers.
- Warehousing and Logistics: Lifting, moving, and storing goods, materials, and packages in distribution centers and storage facilities.
- Manufacturing (General): A wide range of manufacturing operations, including metal fabrication, plastics, and consumer goods production.

1.8 Draft of specifications

Scope

This specification outlines the design requirements for a Zero Gravity Lifting Device prototype, intended to aid in lifting and manipulating heavy objects within industrial settings. The device is being developed as part of an undergraduate project.

Applicable Documents

Relevant safety standards for machinery and material handling equipment, such as ISO 12100, EN 349, EN 614-1, shall be considered during the design process.

Requirements

- General:

  The device shall be capable of lifting and manipulating objects within a weight range of 5 kg to 25 kg. It should be suitable for operation in a controlled laboratory environment. Overall dimensions shall not exceed 1 m x 0.8 m x 2 m (L x W x H) when fully assembled.

- Performance:

  Lifting speed should be adjustable between 0.1 m/s and 0.3 m/s. Positioning accuracy should be ± 5 mm or better. The device shall provide a minimum lifting range of 1 m vertically. It should offer at least 2 degrees of freedom (vertical, horizontal, or rotational).

- Structural:

  The main frame shall be constructed using readily available materials, such as aluminum extrusions, plywood, or acrylic sheets. All load-bearing components must be designed with a suitable safety factor for the rated load.

- Mechanical:

  The lifting mechanism should employ a simple system, such as pulleys, cables, or linear actuators. The actuation system should consist of readily available electric motors or actuators. A basic load-balancing system or counterweight may be incorporated.

- Electrical and Control:

  The device shall operate on a standard 120/230V AC, 50/60Hz single-phase power supply available in the laboratory. The control system shall be based on a microcontroller or development board with appropriate sensor inputs and motor control outputs. The user interface should include simple controls, such as buttons, switches, or potentiometers, for basic operation. Feedback sensors may include rotary encoders or simple position sensors.

- Safety:

  The device must feature an emergency stop system or a means to immediately halt operation. Overload protection should prevent the device from lifting loads exceeding the rated capacity. Moving parts must be properly guarded or shielded to prevent potential hazards. Compliance with relevant safety standards for prototypes is essential.

- Documentation:

  A comprehensive project report covering the design, implementation, and testing of the device shall be provided. Basic technical documentation, including circuit diagrams must be available.

Testing and Evaluation

The device shall undergo basic functional testing and evaluation within the scope of the undergraduate project.

1.9 Standards Available

In the design and development of our Zero Gravity Lifting device, there are several international and industry standards to ensure safety, reliability, and compliance that we need to consider. The following standards are particularly relevant to our project:

- ISO 12100:2010 - Safety of Machinery: General principles for design — Risk assessment and risk reduction. This standard provides a framework for assessing and mitigating risks in machinery design, which is crucial for our lifting device.
- ISO 13849-1:2015 - Safety of Machinery: Safety-related parts of control systems — Part 1: General principles for design. This standard is essential for ensuring the safety and reliability of the control systems in our Zero Gravity Lifting device.
- EN 349:1993+A1:2008 - Safety of Machinery: Minimum gaps to avoid crushing of parts of the human body. This standard helps us design the device to prevent potential injuries to users or operators.
- EN 60204-1:2018 - Safety of Machinery: Electrical equipment of machines — Part 1: General requirements. This standard guides the design of the electrical systems in our lifting device, ensuring safety and reliability.
- ANSI/ASME B30.20-2021 - Below-the-Hook Lifting Devices. While our device is not strictly a below-the-hook system, this standard provides valuable insights into safe lifting practices.
- ISO 9927-1:2013 - Cranes: Inspections — Part 1: General. This standard informs our approach to designing the device for ease of inspection and maintenance.
- ISO 4301-1:2016 - Cranes and Lifting Appliances: Classification — Part 1: General. This standard helps us classify our lifting device appropriately and design it to meet the required specifications for its class.

Additionally, we are considering the following standards specific to medical and assistive devices:

- ISO 10535 - Hoists for the transfer of disabled persons – Requirements and test methods.
- ANSI/AAMI ES60601-1 - Medical electrical equipment - Part 1: General requirements for basic safety and essential performance.

2 Electronics

2.1 Electronic Sub-assemblies of the selected Conceptual design

2.1.1 Stepper Motor and the Motor Driver

The stepper motor and motor driver subsystem are critical components of the Zero Gravity Lifting device, responsible for controlling the movement and positioning of the lifting mechanism. The stepper motor provides precise and incremental rotation, enabling accurate adjustments in lifting height and direction. Paired with a suitable motor driver, this subsystem ensures smooth and controlled operation of the lifting device.

2.1.2 Load Cell and the Amplifier

The load cell and amplifier assembly form an essential part of the feedback system for the Zero Gravity Lifting device. The load cell measures the applied force or weight on the lifting platform, while the amplifier amplifies the output signal from the load cell for further processing. This feedback mechanism allows the device to adjust lifting parameters based on the detected load, ensuring safe and efficient material handling.

2.1.3 Feedback System

The feedback system integrates various sensors, including load cells, position encoders, and limit switches, to provide real-time feedback on the status and performance of the Zero Gravity Lifting device. By continuously monitoring key parameters such as load weight, position, and speed, the feedback system enables precise control and adjustment of lifting operations, enhancing safety and productivity.

2.1.4 Power Supply

The power supply unit is responsible for supplying the necessary electrical power to all electronic components of the Zero Gravity Lifting device. It converts the incoming AC voltage from the mains power source into the appropriate DC voltage levels required by the stepper motor, motor driver, load cell amplifier, and other subsystems. The power supply unit ensures reliable and stable operation of the device under varying load conditions and environmental factors.

2.2 Specifications of electronic sub assemblies

1. Stepper Motor
   (a) Model : NEMA 23HS8430D8L25-500
   (b) Diameter of rotating shaft : 8mm

(c) Length of rotating shaft : 25mm
(d) Current : 3A
(e) Step angle : 1.8°±5%
(f) Horizontal accuracy : 0.02mm Max
(g) Vertical accuracy : 01-0.3mm
(h) Ambient temperature: -20 ℃+50 ℃
(i) Insulation resistance : 500V DC 100M Min
(j) Insulation grade : 50Hz 1Minute 500V Min

2. Stepper motor driver
   (a) Model: TB6600
   (b) Compatible Motors: Nema 17/23/34 (42/57/86 Stepper Motor)
   (c) Control Signal: 3.3VDC–24VDC
   (d) Subdivision Accuracy: 1–32
   (e) Output Current: 4A
   (f) Voltage: 9 VDC–40 VDC
3. Load cell
   (a) Rated load: 20 Kg
   (b) Output: 2mv / v
   (c) Temperature zero drift: 0.1% F.S
   (d) Output sensitivity: ± 0.15mv / v
   (e) Temperature sensitivity: 0.05% F.S
   (f) Insulation resistance: 2000M
   (g) Excitation voltage: 5-10VDC
4. Load Cell Amplifier (HX7111)
   (a) Dual-channel 24 Bit Precision A/D
   (b) On-chip active low noise PGA with a selectable gain of 32, 64, and 128.
   (c) Operating Voltage (V): 5V max
   (d) Forward Current (mA): 20 ˜30
   (e) Operating Temperature(°C): -25°C to 85°C
5. Feedback system (AS5600 Magnetic Encoder)
   (a) Supply Voltage: 3-3.6 V and 4.5-5.5 V
   (b) Temperature Range (°C): -40 to +125
   (c) Output: Analog out / PWM / I^2C
   (d) Interface: I^2C
   (e) Resolution (bits): 12
6. Power Supply
   (a) Input Voltage: 230VAC
   (b) Output Voltage: 12VDC
   (c) Output Current: 5A (max)

2.3 Selection Criteria for used Parts

1. Stepper Motor
   (a) NEMA 23 stepper motor (23HS8430D8L25-2) is being used as it has 1.5Nm maximum torque which is sufficient for us to demonstrate the zero-gravity effect to a considerable amount of weight.
   (b) Additionally Gear system is used, therefore, effective maximum torque is higher than 1.5Nm.
   (c) Affordability is also taken into account in choosing this.
2. Motor Driver
   (a) Compatibility for the Nema 23 stepper motor
   (b) Maximum current of 4A which is below the maximum current drawn by the motor(3A)
   (c) Affordability is also considered in choosing this.
   (d) Effective Heat Management
3. Load cell and Amplifier.
   (a) Load cell does not measure the total weight directly. Only a component of the weight is measured and through calculation total weight can be derived. Therefore, 20Kg Load cell is more than sufficient.
   (b) HX7111 amplifier is used as it has a good accuracy and 12bit A-D converter. And resources and components necessary for the module are readily available.
4. Feedback System
   (a) AS5600 magnetic encoder is used for the Feedback system.
   (b) Ease of use
   (c) Affordability
   (d) Readily available
5. Buck Converter
   (a) TSOT26 chip is used as the buck converter as it's capable of providing 3A of maximum current with very low ripple.
6. Atmega328PU
   (a) The Atmega328PU is selected as the primary processor because of its dependable performance, cost-effectiveness, and the availability of readily accessible resources.

2.4 PCB Design

2.4.1 Introduction

The PCB design for the Zero Gravity Lifting Device is integral to its functionality, particularly in controlling the Nema23 stepper motor and ensuring precise movements essential for lifting and positioning. This section of the report focuses on detailing the design, development, and evaluation of the PCB.This section covers the schematic designs, layout designs, and final PCB design, along with the challenges encountered and the solutions developed throughout the PCB design process.

2.4.2 Design Specifications

1. Functional Requirements

   (a) **Decision Making**: Integrate the Atmega328P SMD microcontroller for decision-making processes within the control system of the Zero Gravity Lifting Device. This includes handling sensor inputs, calculating control algorithms, and executing commands for the stepper motor.

   (b) **Signal Amplifying**: Incorporate the LT110 Buck converter to regulate power supply voltages efficiently across the PCB. Ensure stable and reliable voltage levels suitable for the Atmega328P microcontroller, HX711 signal amplifier, and other associated components.

   (c) **Power Regulating**: Utilize the HX711 signal amplifier to enhance and amplify weak signals from sensors or other input devices. This component is crucial for accurate data acquisition and measurement, contributing to precise control and monitoring capabilities.

2. Technical Requirements

   (a) **Voltage and Current Rating**: Nominal Voltage for the components are 5V and the total current is less than 1A. Therefore we have used the TSOT26 buck converter which is capable of providing 5V at maximum current of 3A.

2.4.3 Schematics

Microprocessor Circuit

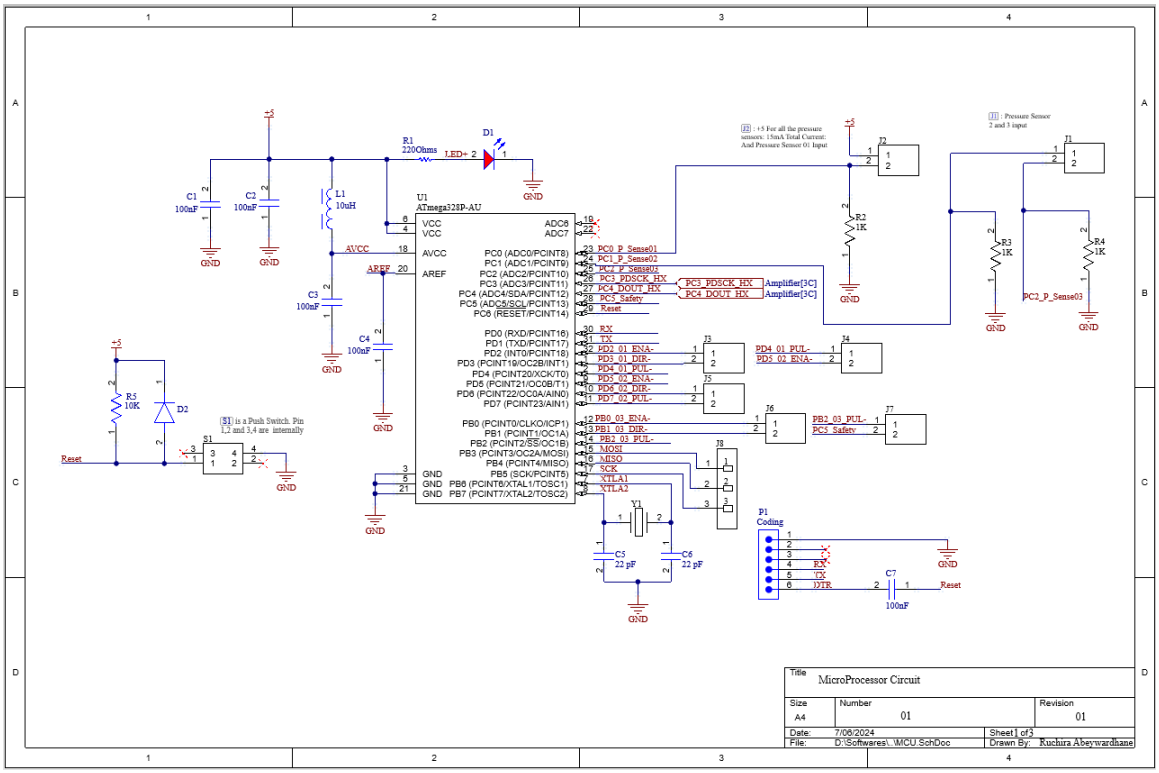


Figure 2.1: Microprocessor Circuit

Load Cell Amplifier Circuit

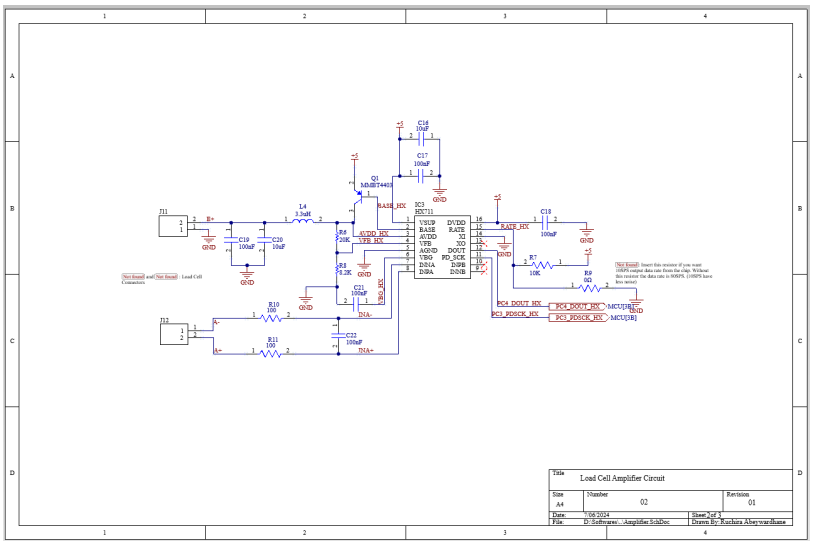


Figure 2.2: Load Cell Amplifier Circuit

Power Regulator Circuit

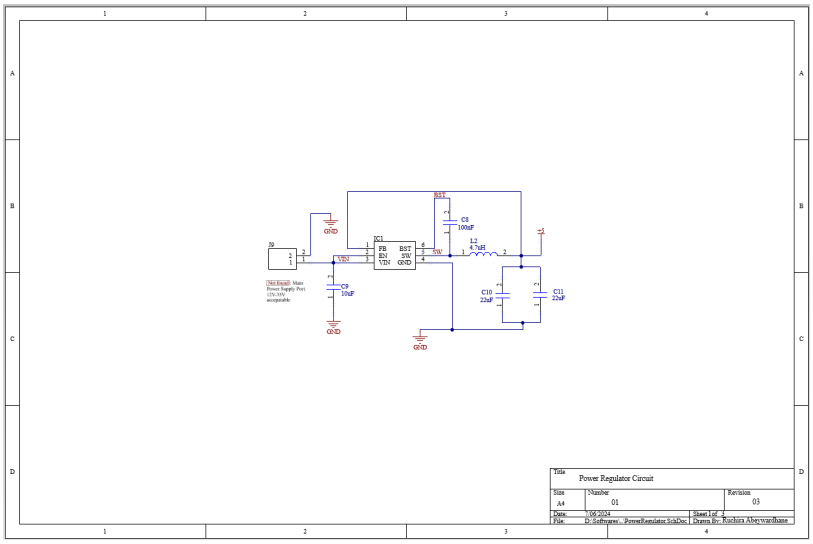


Figure 2.3: Power Regulator Circuit

2.4.4 PCB Layout

Component Placement

Components in the PCB for the Zero Gravity Lifting Device are grouped according to their subsystems. This approach optimizes signal integrity, reduces interference, and simplifies trace routing. It also enhances assembly and maintenance efficiency, aligning with effective electronic design practices.As a best practice, components within subsystems in the PCB for the Zero Gravity Lifting Device are strategically placed to minimize trace intersections and reduce the need for layer changes.

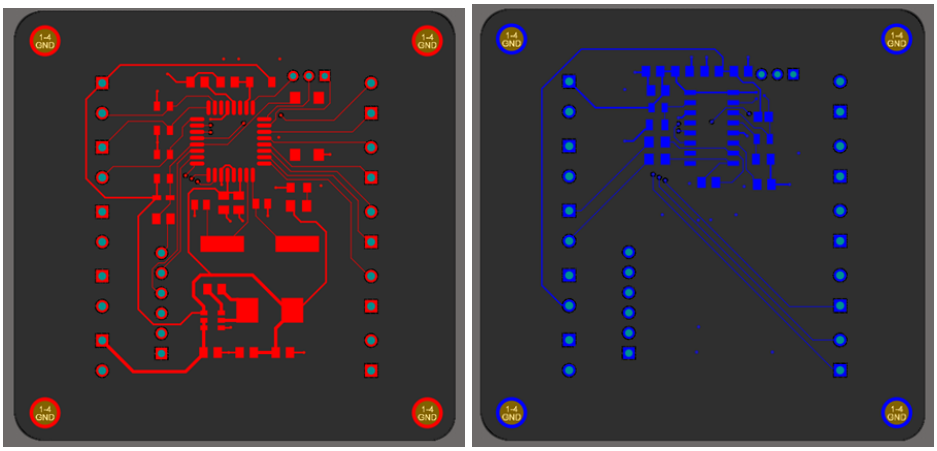


Figure 2.4: Routes in Top and Bottom Layers

Furthermore in order to keep the PCB design compact we have used SMD components therefore allowing us to place components on both sides of the board. Additionally, ICs are centrally located on both sides of the board to accommodate connections coming from multiple directions.

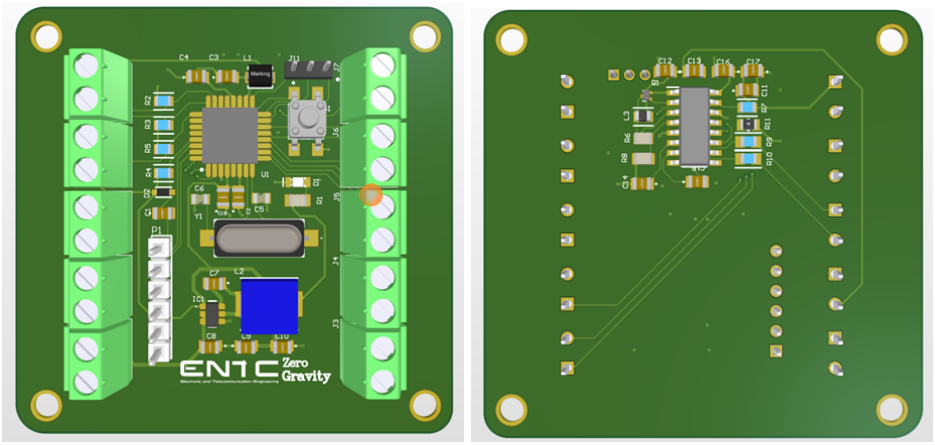


Figure 2.5: 3D View Top and Bottom

Trace Width

Trace widths are calculated according to the maximum currents they are going to accommodate. For the power lines they will be carrying less than 0.5A of current. Therefore trace width is chosen as 0.3mm. For signal lines they will be carrying less than 0.3A current therefore 0.15mm trace width is chosen.

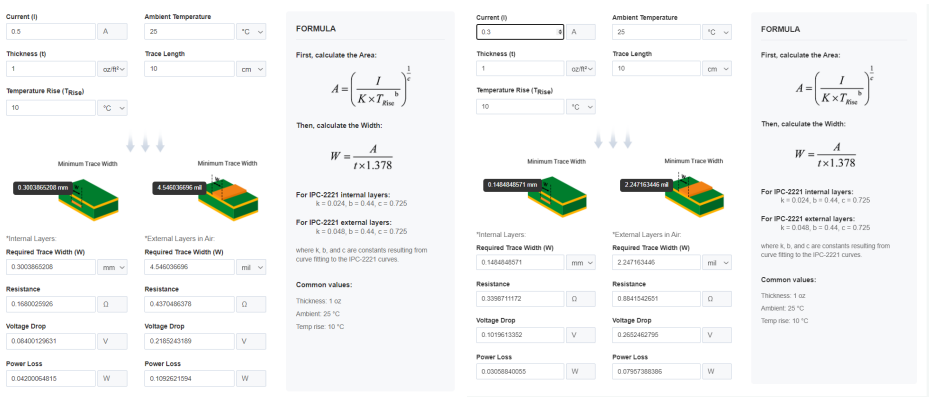


Figure 2.6: Trace Width Calculation

Layers

The PCB design utilizes a 4-layer configuration primarily to accommodate components on both sides of the board and to address specific considerations such as the placement of the HX711 signal amplifier on the bottom layer. This setup helps mitigate potential noise interference that could affect its performance. Two dedicated ground planes positioned in the middle of the PCB effectively shield against noise, ensuring signal integrity between layers and enhancing the overall reliability of the design. Routing is done only on outer layers (Top and bottom). No routings are done on inner layers(GND01, GND02).

#	Name	Material	Type	Weight	Thickness	Dk	Df
	Top Overlay		Overlay				
	Top Solder	Solder Resist	Solder Mask		0.01016mm	3.5	
1	Top Layer		Signal	1oz	0.03556mm		
	Dielectric 2	PP-006	Prepreg		0.07112mm	4.1	0.02
2	GND01	CF-004	Signal	1oz	0.035mm		
	Dielectric 1	FR-4	Dielectric		0.32004mm	4.8	
3	GND02	CF-004	Signal	1oz	0.035mm		
	Dielectric 3	PP-006	Prepreg		0.07112mm	4.1	0.02
4	Bottom Layer		Signal	1oz	0.03556mm		
	Bottom Solder	Solder Resist	Solder Mask		0.01016mm	3.5	
	Bottom Overlay		Overlay				

Figure 2.7: Layer Stack

2.4.5 Instructions passed to layout person

1. Be mindful when placing the amplifier circuit as the noise from the power circuit may affect its performance.
2. Keep a considerable amount of space between the buck converter IC and the atmega328P.
3. Keep analog traces as short as possible and away from digital signal traces as possible.
4. Keep terminal ports on the outer sides of the PCB.
5. Keep the design compact as possible.

2.4.6 Design Tools

For the PCB design of the Zero Gravity Lifting Device, Altium Designer serves as the primary software tool. It is utilized for creating schematic diagrams, laying out the PCB, and managing the design process efficiently. Additionally, SymCAS is employed to import component footprints, symbols, and for 3D modeling when necessary.

Figure 2.8: Layer Stack

2.4.7 Bill of Materials

RK73H2ATTD2200F		R1	RESC2013X60X30ML20T20	CMP-2001-02206-1	1
RK73H2ATTD1001D	1kΩ ±0.5% 0.25W Chip Resistor 0805 Thick Film AEC-Q200 Qualified	R2, R3, R4	FP-RK73H2A-IPC_C	CMP-04015-089471-1	3
RK73H2ATTED1002F	10kΩ ±1% 0.25W Chip Resistor 0805 Thick Film AEC-Q200 Qualified	R5, R7	FP-RK73H2A-IPC_C	CMP-04015-088604-1	2
RK73H2ATTD2002F		R6	RESC2013X60X30LL20T20	CMP-2001-02185-1	1
RK73H2ATTD8201F		R8	RESC2013X60X30NL20T20	CMP-2001-02517-1	1
RCC08050000Z0EA	RCC08050000Z0EA Res Thick Film 0805 0 Ohm Conformal Coated Pad SMD Automotive AEC-Q200 T/R	R9	FP-RCC0805-e3-MFG	CMP-00030-108211984-1	1
RK73H2ATTED1000F	100Ω ±1% 0.25W Chip Resistor 0805 Thick Film AEC-Q200 Qualified	R10, R11	FP-RK73H2A-IPC_A	CMP-04015-093159-1	2
TL3305BF160QG	Switch	S1	TL3305BF160QG	TL3305BF160QG	1
ATmega328P-AU	8-bit AVR Microcontroller, 32KB Flash, 1KB EEPROM, 2KB SRAM, 32-pin TQFP, Industrial Grade (-40°C to 85°C)	U1	32A_M	CMP-0095-00269-2	1
ABLS-16.000MHz-16-A-4-H-T	Crystal or Oscillator	Y1	ABLS	ABLS-16.000MHz-16-A-4-H-T	1

Figure 2.9: Bill of Materials Part 01

Comment	Description	Designator	Footprint	LibRef	Quantity
C0805Y104J3RACTU	Capacitor	C1, C2, C3, C4, C7, C8, C17, C18, C19, C21, C22	C0805	C0805Y104J3RACTU	11
C0805C220F1HACTU	C0805 22 pF X8R 30ppm/°C 1.00% 100 V	C5, C6	FP-C0805C-DN-MFG	CMP-03020-009085-1	2
C0805C106K4PAC7210	Capacitor	C9, C16, C20	C0805	C0805C106K4PAC7210	3
C0805C226K9PACTU	Capacitor	C10, C11	C0805	C0805C226K9PACTU	2
150080RS75000		D1	WL-SMCW_0805_150080xx75000	CMP-1426-00011-2	1
CDSU4148	Diode	D2	DIOM179X85N	CDSU4148	1
AP63205WU-7	Integrated Circuit	IC1	AP63205WU7	AP63205WU-7	1
HX711	Integrated Circuit	IC3	SOIC127P600X160-16N	HX711	1
1727010	Connector	J1, J2, J3, J4, J5, J6, J7, J9, J11, J12	1727010	1727010	10
TMM-103-03-T-S	CONN HEADER VERT 3POS 2MM	J8	FP-TMM-103-03-T-S-MFG	CMP-02766-001695-1	1
744764910	744764910 General Purpose Inductor, 10uH, 10%, 1 Element, SMD, 1210	L1	WE-GFH_3225	CMP-03913-61645378-1	1
SRP7028CC-4R7M	Inductor	L2	INDPM7366X300N	SRP7028CC-4R7M	1
MLZ2012A3R3WTD25	Shielded Multilayer Inductor 3.3uH ±20% 450mA 442mΩ AEC-Q200 SMD 0805	L4	FP-MLZ2012-0_85-IPC_C	CMP-08253-000097-1	1
TSW-106-07-L-S	0.025" SQ Post Header, Through-hole, Vertical, 55 to 125 degC, 2.54 mm Pitch, 6-Pin, Male, RoHS	P1	SMTC-TSW-106-07-X-S	CMP-1024-00386-1	1
MMBT4403M3T5G	PNP Switching Transistor, 3-Pin SOT-723, Pb-Free, Tape and Reel	Q1	ONSC-SOT-723-3-631AA-01_C_V	CMP-1048-00929-1	1

Figure 2.10: Bill of Materials Part 02

2.4.8 Testing

- Before the PCB implementation, the circuit was constructed on a breadboard and individually tested for functionality for separate subsystems.
- After individual functional testing, all components were interconnected and evaluated collectively to assess the overall performance of the product on the breadboard.
- After soldering the PCB, a continuity test is conducted to verify proper soldering and ensure electrical connections are intact across all components and traces.

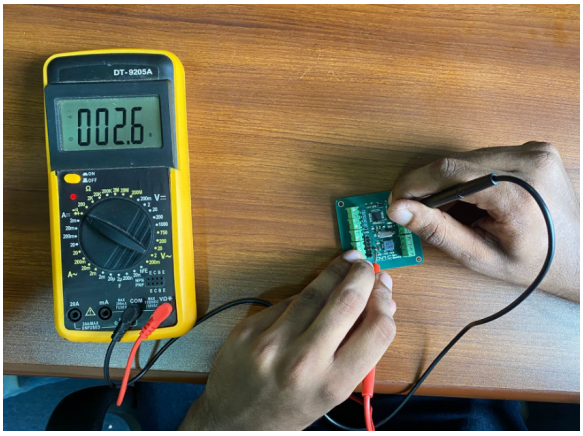


Figure 2.11: Continuty Testing

2.4.9 Specifications

- Dimensions - 54.104cm x 55.753cm
- Layers - 4
- Outer Copper Weight - 1oz
- Inner Copper weight - 0.5oz
- Material - FR4

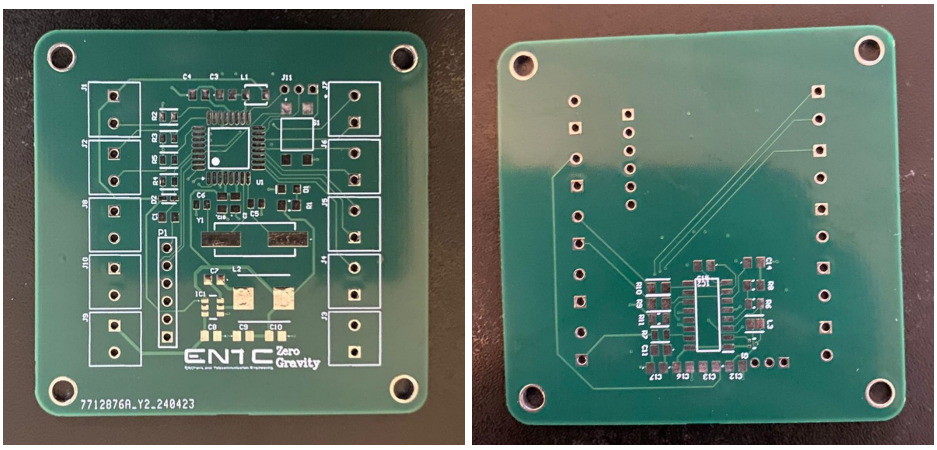


Figure 2.12: Bare PCB

2.4.10 Conclusion For the PCB Design Section

In summary, the PCB design for the Zero Gravity Lifting Device is thoughtfully structured to optimize functionality and reliability. Components are strategically placed within subsystems to minimize trace intersections, ensuring efficient signal routing. The use of a 4-layer configuration with dedicated ground planes effectively shields against noise interference, crucial for sensitive components like the HX711 signal amplifier. Post-soldering continuity testing further ensures the integrity of electrical connections, validating the strength of the PCB assembly. These design choices collectively support the device's essential operations with precision and efficiency.

3 Mechanical Design

3.1 Design of mechanical sub-assemblies

3.1.1 Pulling Device and Gear Box

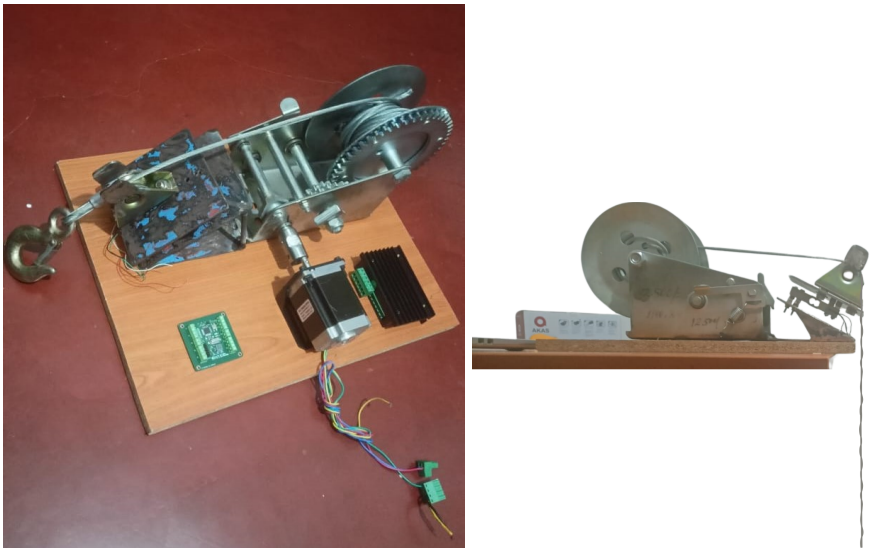


Figure 3.1: Pulling Device and Gear Box

We aimed to keep our mechanical design simple, focusing only on essential components due to the high costs associated with advanced functionalities, which are currently beyond our financial capabilities. Our primary requirement was for a pulling device equipped with a gear box. Gearbox was required to increase the torque provided from the NEMA23 stepper motor. We were able to source a ready mate winch which met for our specifications.

3.1.2 Load Cell Converter

To identify the user interactions with the lifted object (Moving upward or downward) it's required to calculate the tension of the cable. In order to calculate the tension of the cable directly from a Load cell we needed a S type Load cell. But S type Load cells were not available in Sri Lanka, and we were not in a position (financially) order it separately from overseas. Therefore, we implemented a mechanical structure that enabled us to determine the cable tension using a single-point beam load cell. After evaluating various conceptual designs, we opted for design 2.

(a) (b)

Figure 3.2: Load Cell Converter

3.2 Specifications of mechanical sub-assemblies

3.2.1 Pulling Device and GearBox

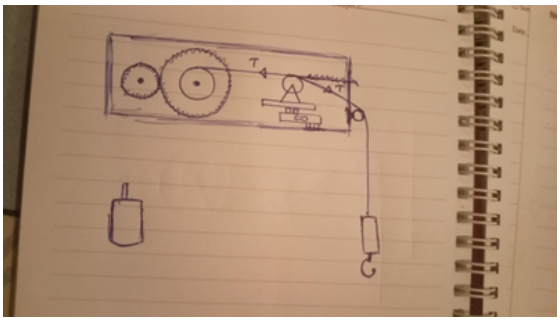
Primary Specifications is to increase the torque and therefore to lift at least 10Kg of weight consequently demonstrating a sufficient zero gravity effect on the managing the object.

Figure 3.3: Pulling Device and GearBox

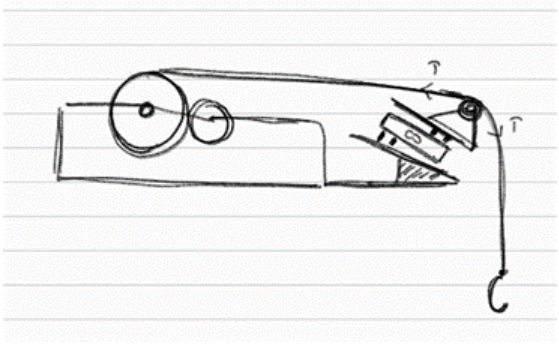
3.2.2 Load Cell Converter

Primary specification is to Direct a component of the Tension force in the cable to the single beam Load cell.

3.3 Rough Sketches



(a)



(b)

Figure 3.4: Rough Sketches 1

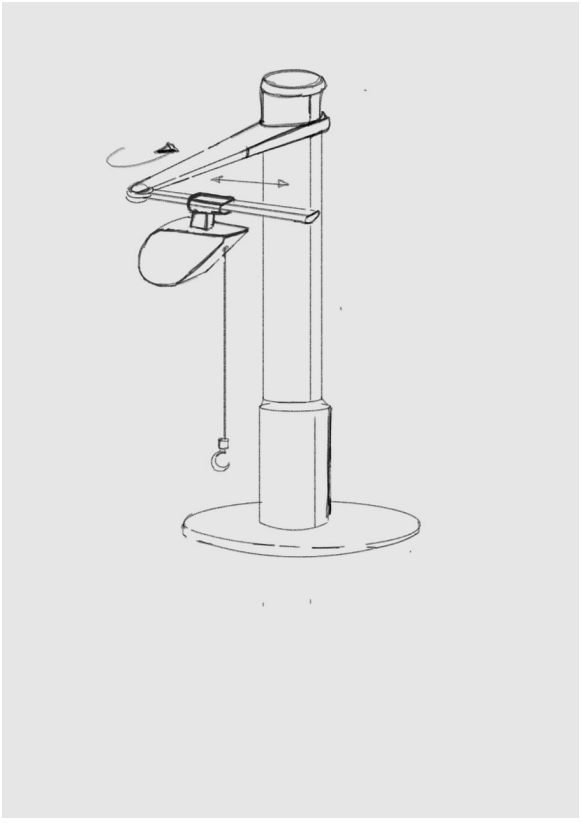


Figure 3.5: Rough Sketches 2

3.4 Selection Criteria for Various Parts

3.4.1 Pulling Device and GearBox

For the GearBox we needed at least a 1:4 ratio and we were able to find a 1:9 ratio gearbox which is much better. As for the winch we needed a device which is able to pull at least 10Kg of weight and we were able to find a Winch which can pull maximum of 1000Kg of weight.

3.4.2 Load Cell Converter

Material for the Load cell converter was selected considering it needs to withstand a considerable amount of force in the process. Therefore, iron plates were used to construct the mechanism.

3.5 PCB Dimensions

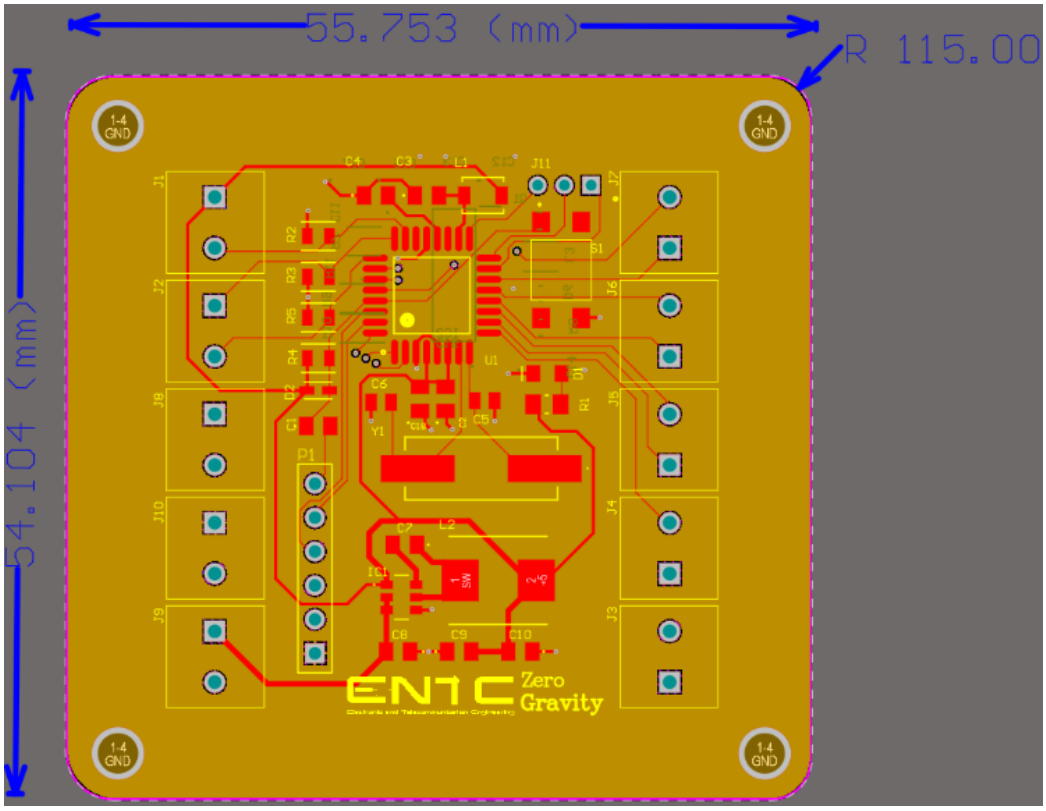


Figure 3.6: PCB dimensions

3.6 Solidworks Designs

3.6.1 Main Enclosure

Assembly

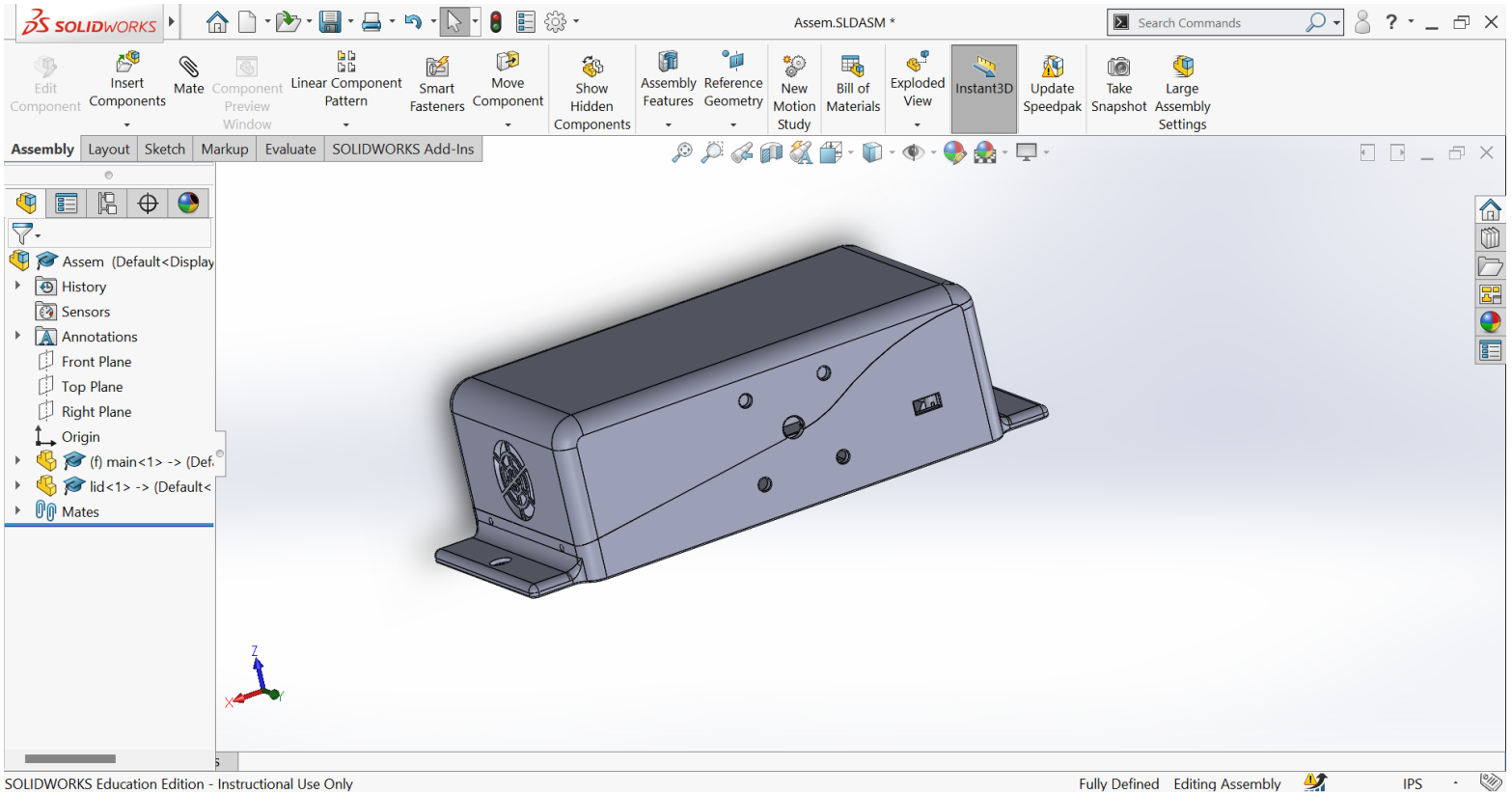


Figure 3.7: Main Enclosure: Assembly

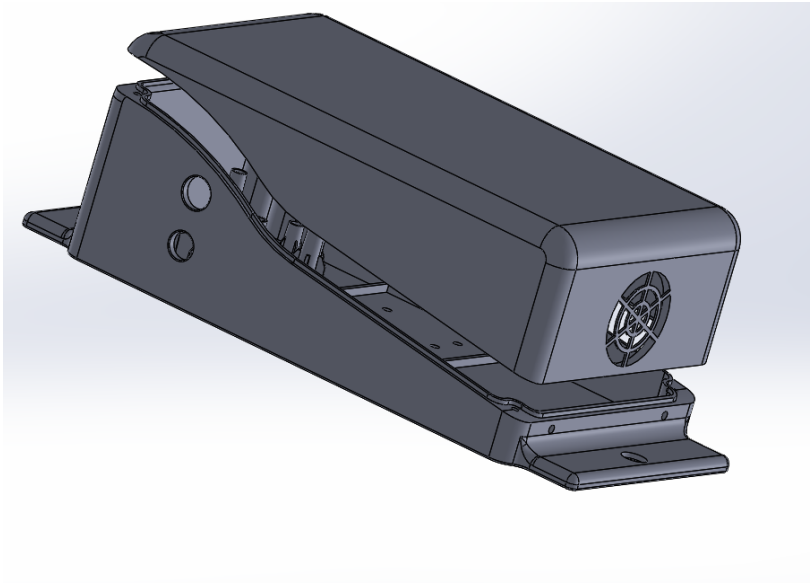


Figure 3.8: Main Enclosure: Exploded view 1

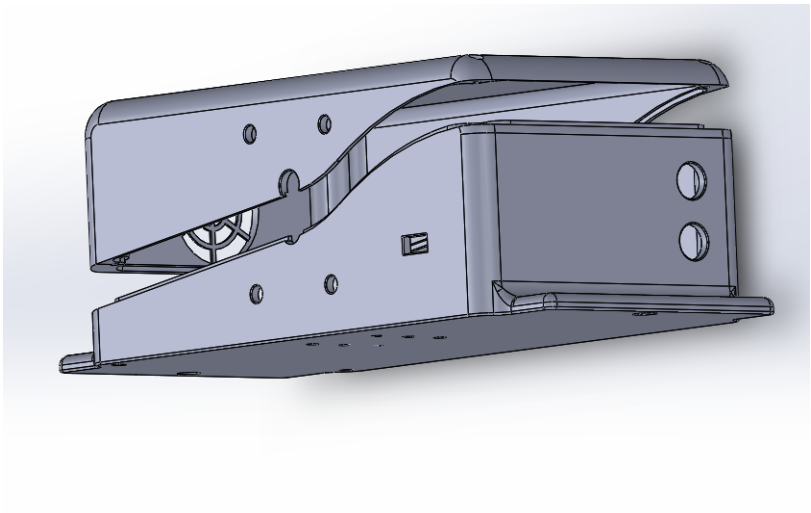


Figure 3.9: Main Enclosure: Exploded view 2

This is the main enclosure cover for the entire electronic assembly. Its design has been made to meet all industrial standards and be moldable. Its main purpose is user safety from any electrical leakage.

Bottom part

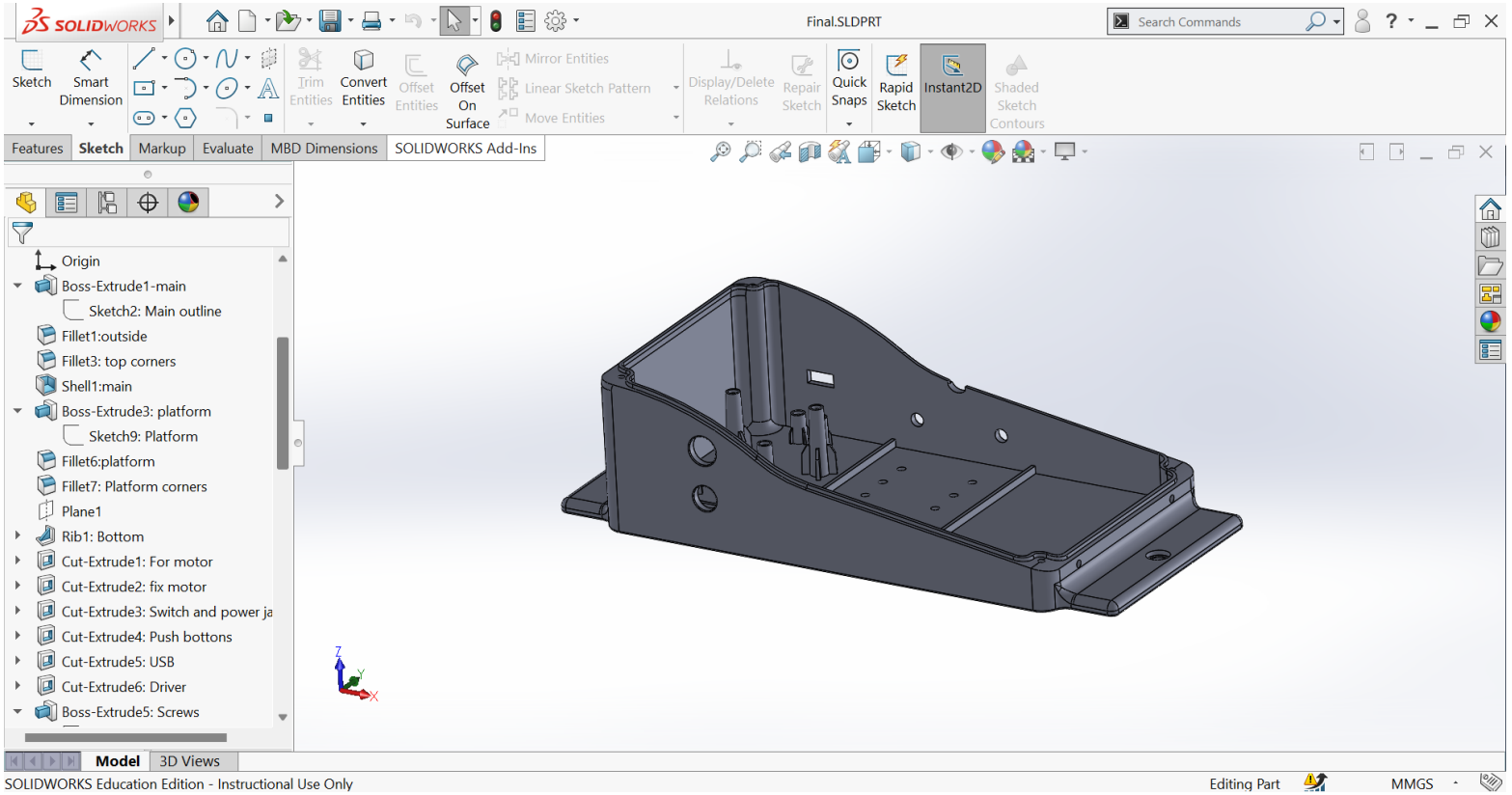


Figure 3.10: Main enclosure: Bottom part

Bottom part: Mold

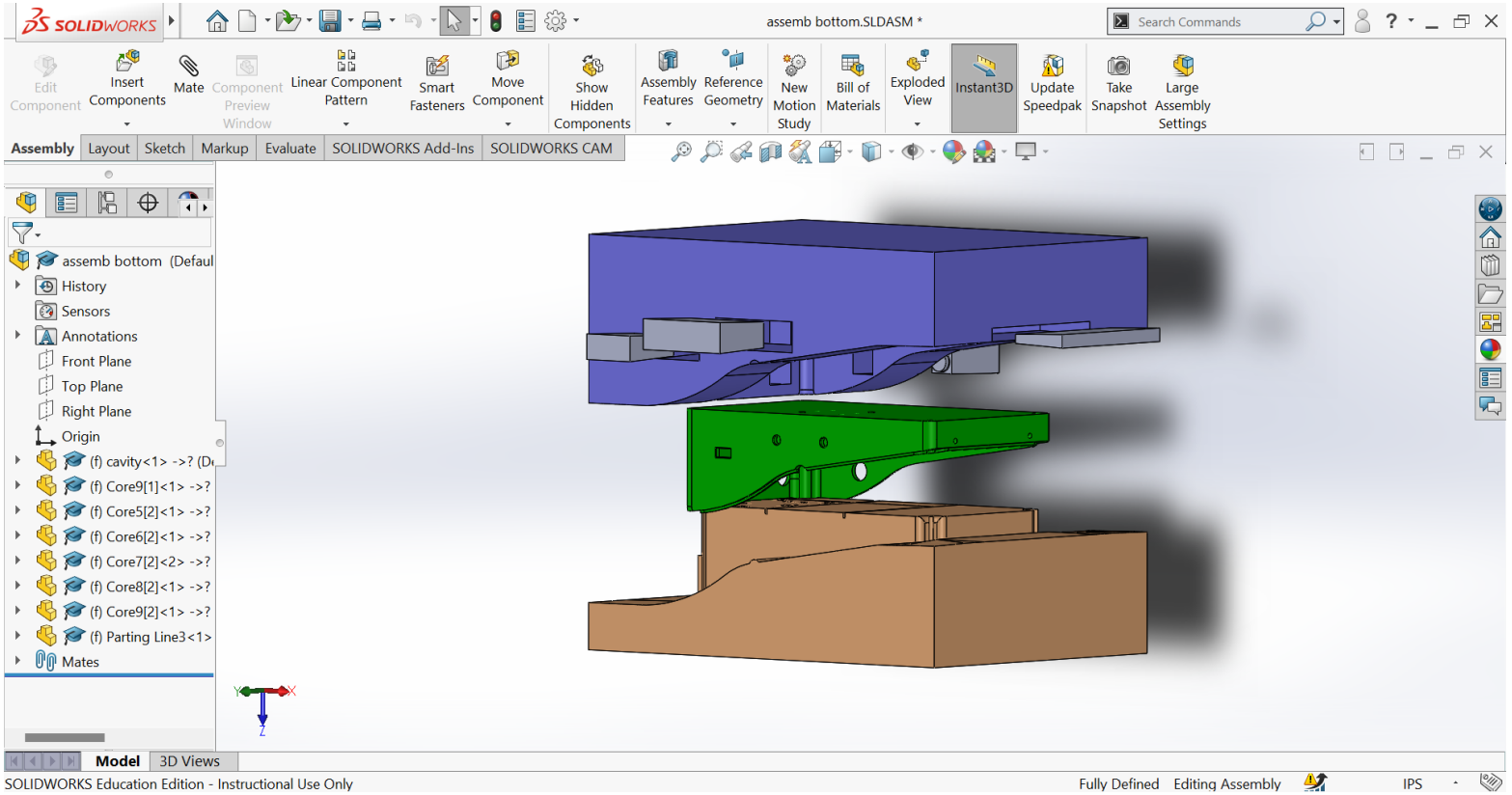


Figure 3.11: Main enclosure: Bottom part-Mold: Exploded view 1

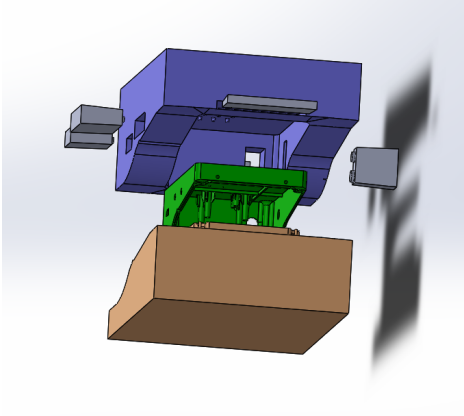


Figure 3.12: Main enclosure: Bottom part-Mold: Exploded view 2

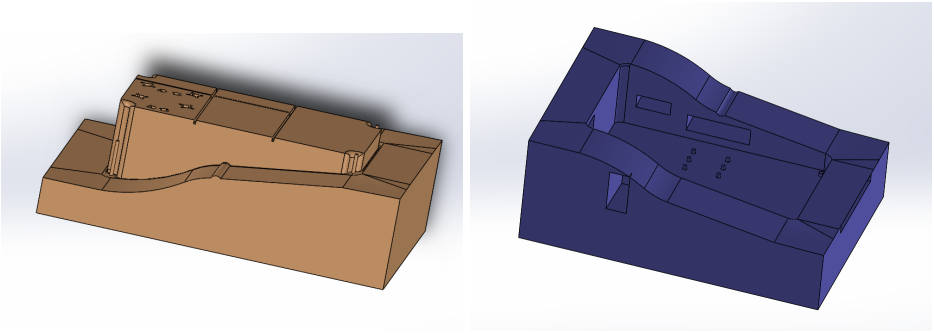


Figure 3.13: Core and Cavity

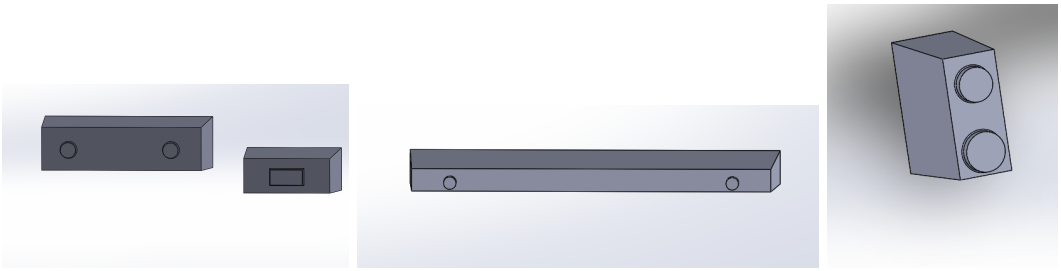


Figure 3.14: Core 1,2,3,4

Top part

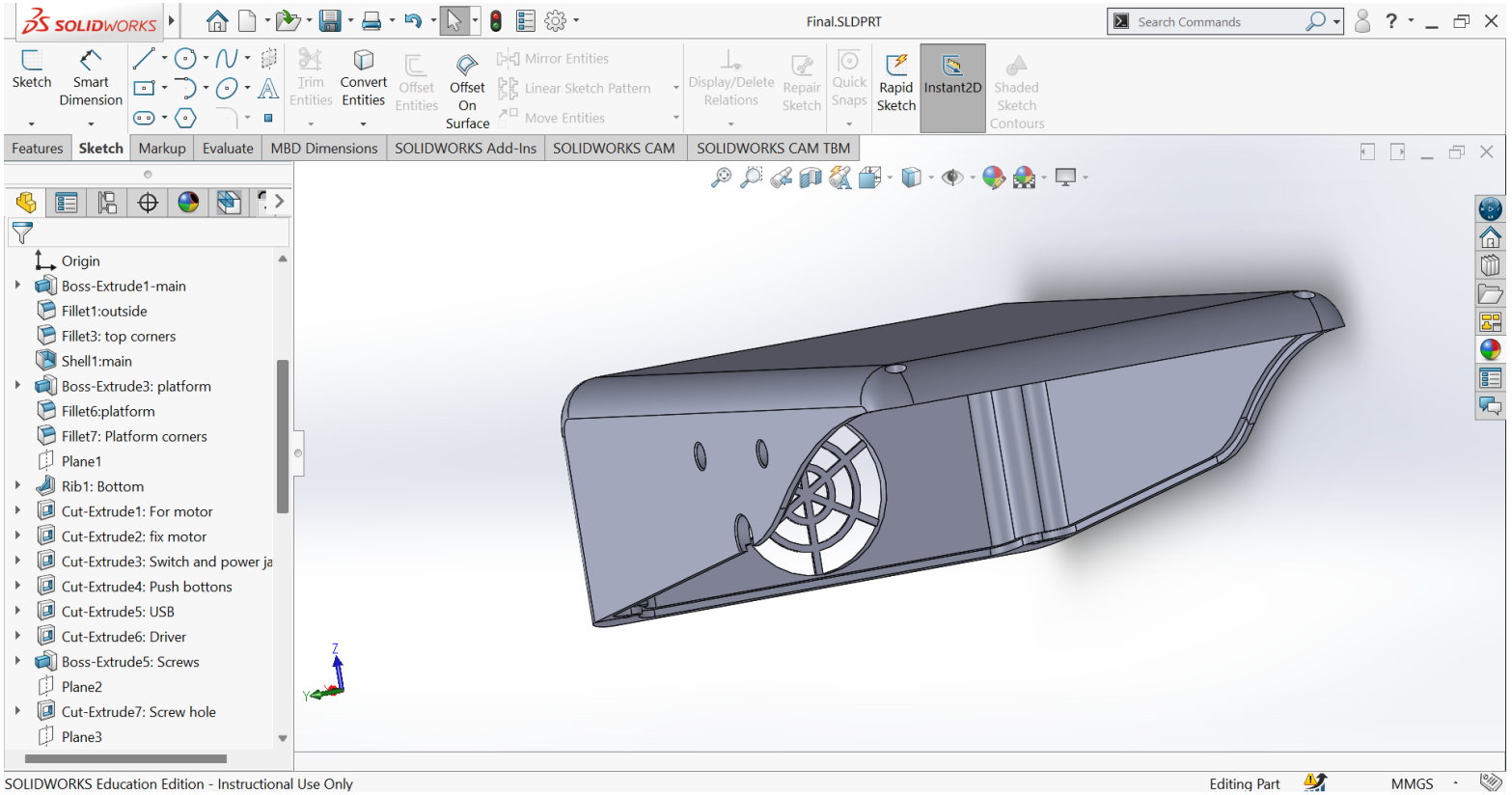


Figure 3.15: Enter Caption

Top part: Mold

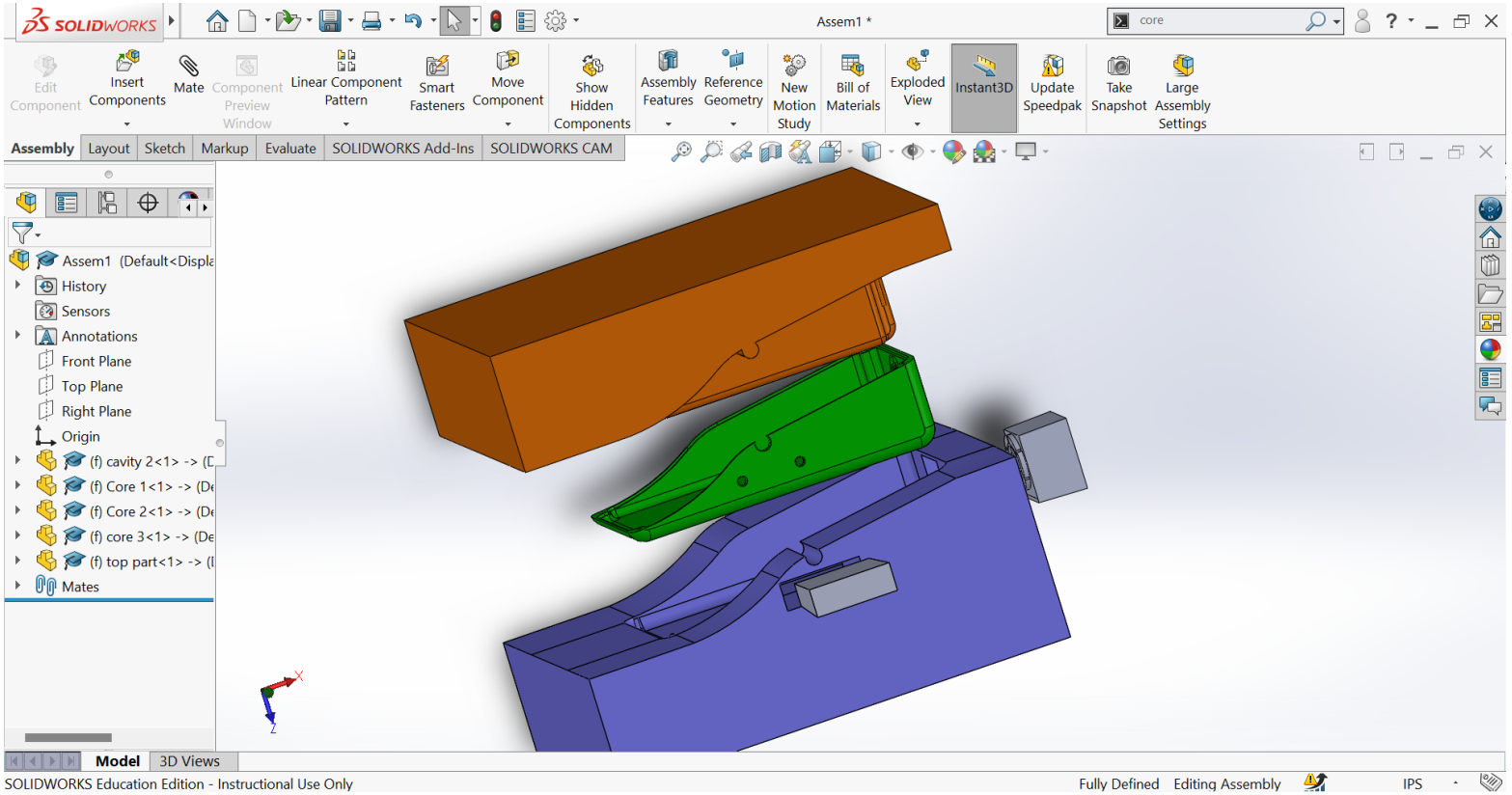


Figure 3.16: Main enclosure: Top part-Mold: Exploded view 1

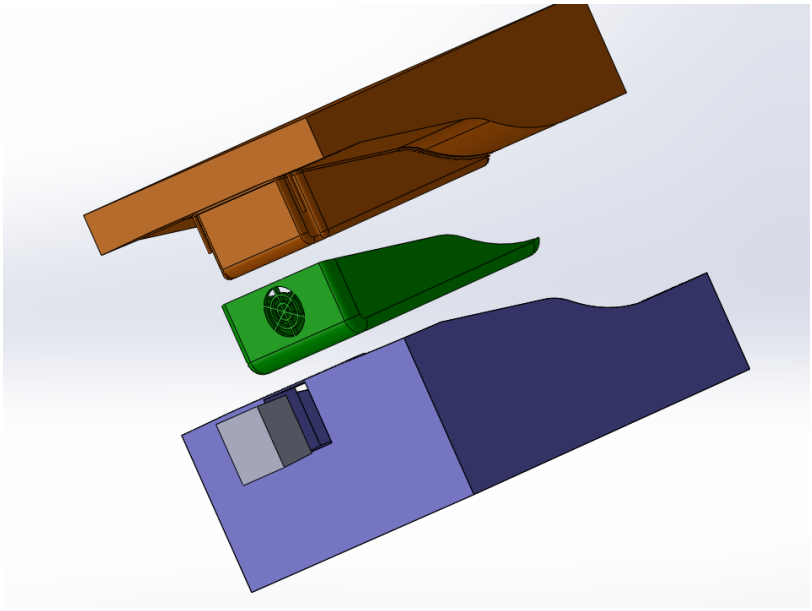


Figure 3.17: Main enclosure: Top part-Mold: Exploded view 2

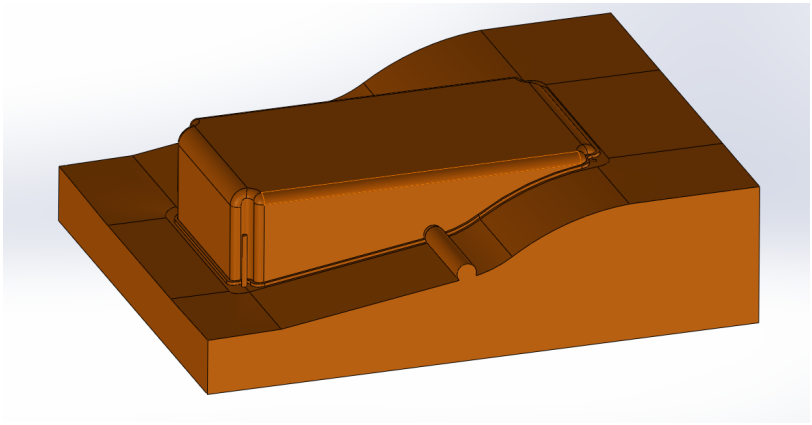


Figure 3.18: Core

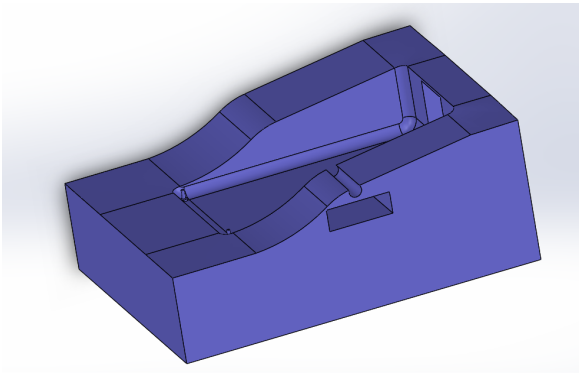


Figure 3.19: Cavity

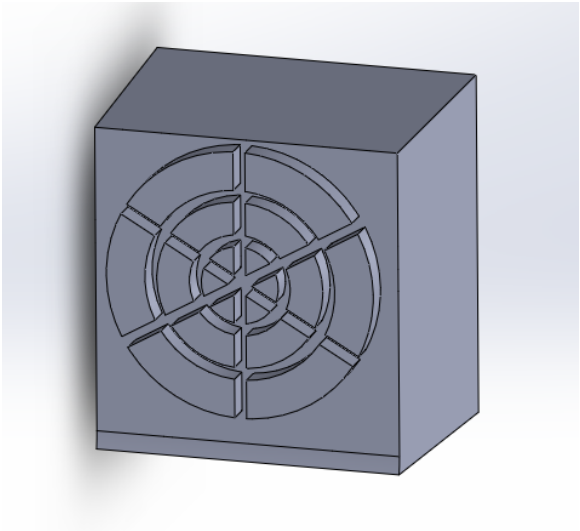


Figure 3.20: Core 1

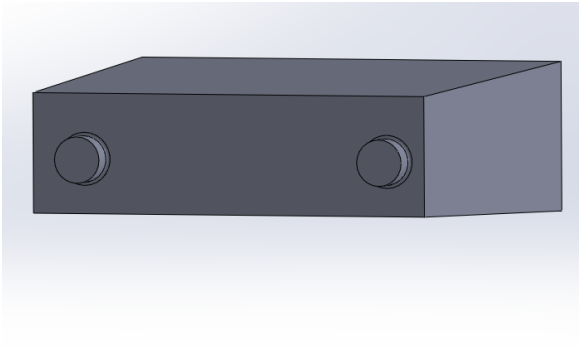


Figure 3.21: Core 2

3.6.2 Feedback Enclosure

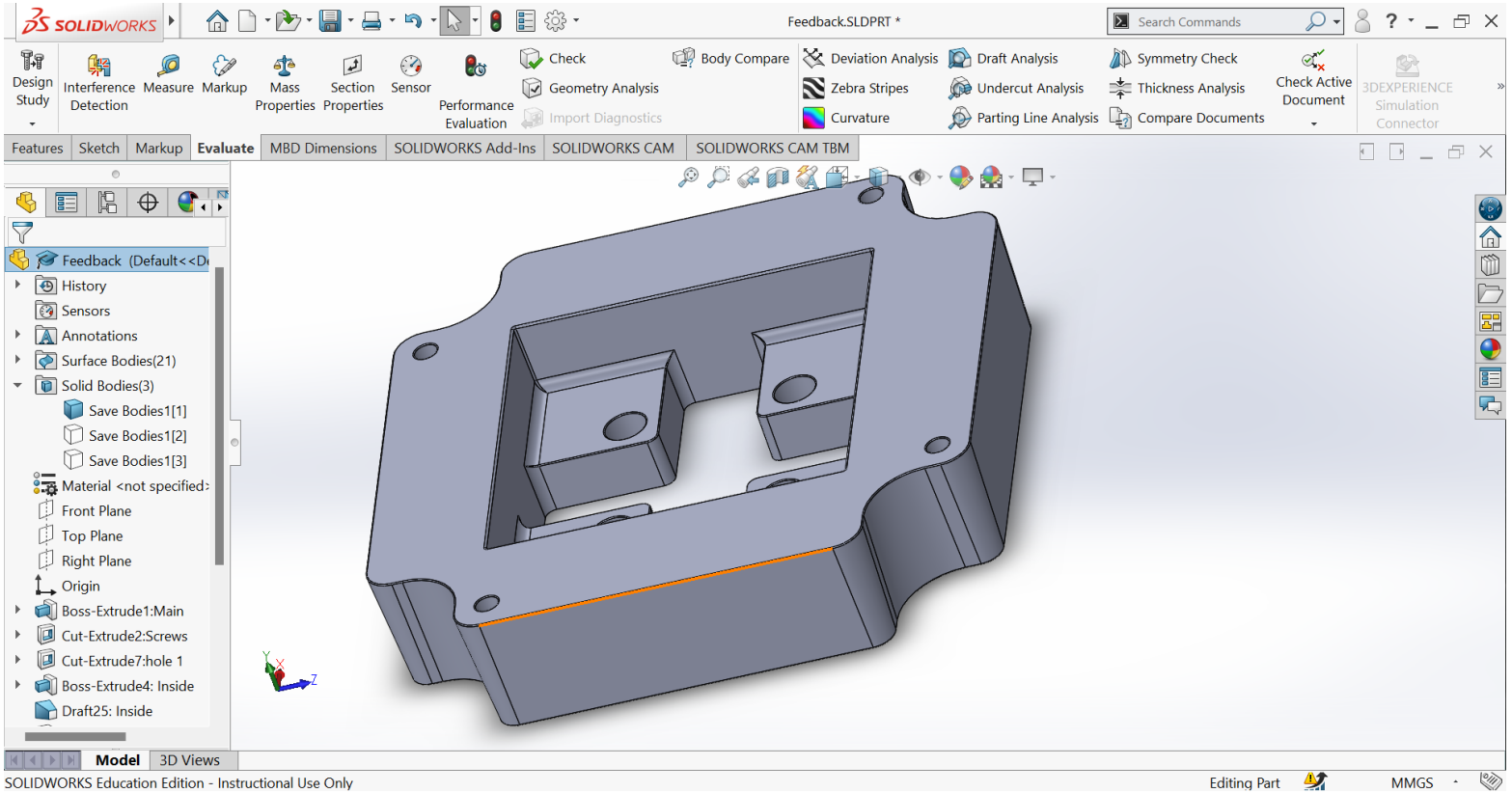


Figure 3.22: Feedback enclosure

Feedback enclosure: Mold

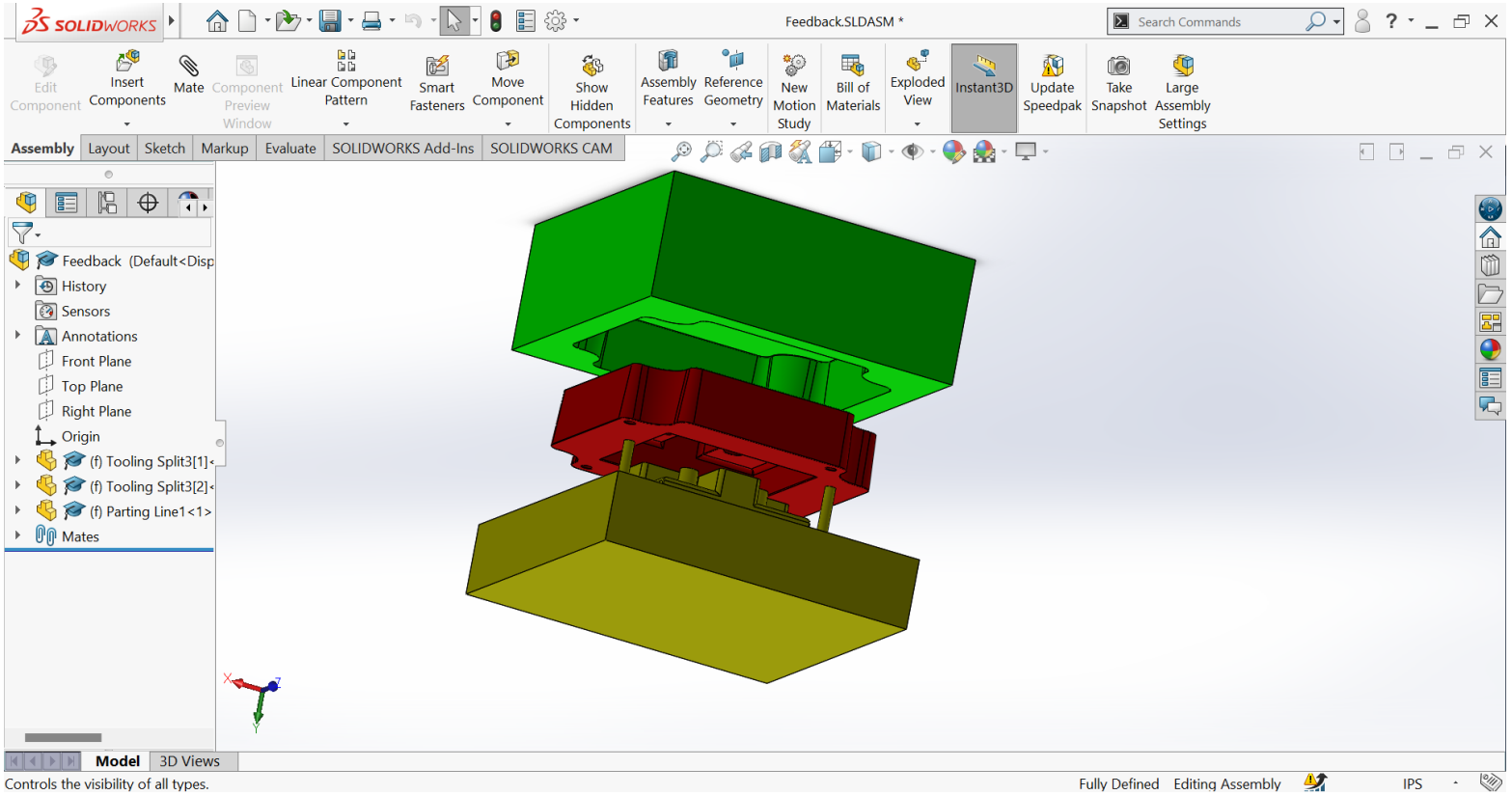


Figure 3.23: Feedback enclosure: Mold: Exploded View 1

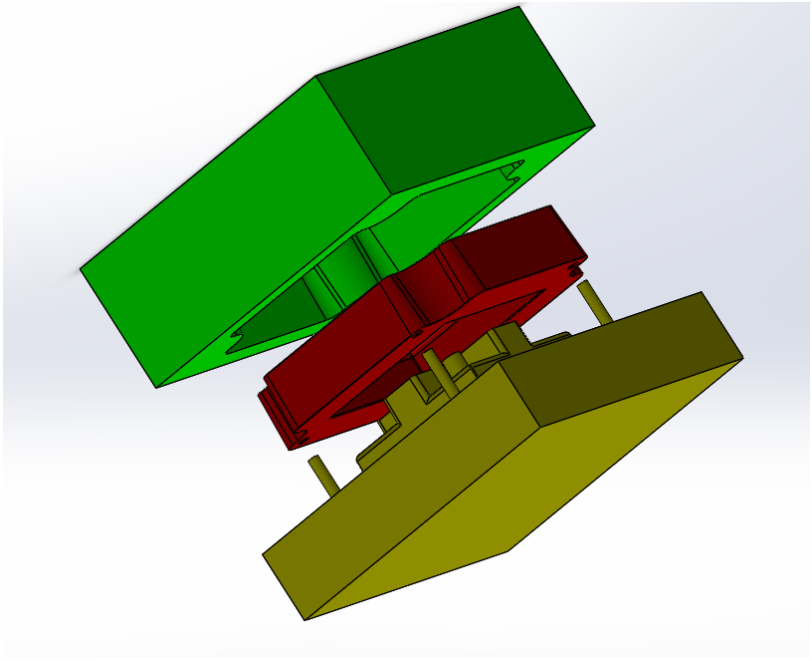


Figure 3.24: Feedback enclosure: Mold: Exploded View 2

This is the enclosure of a feedback system, mainly covering the magnetic encoder specifically designed to fit with the stepper motor. Its main purpose is to improve structural integrity and user safety.

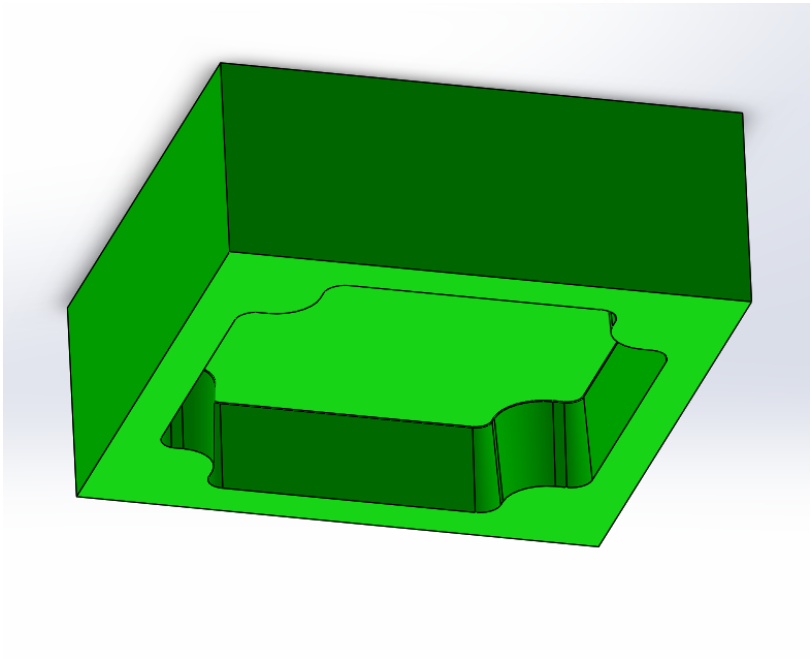


Figure 3.25: Cavity

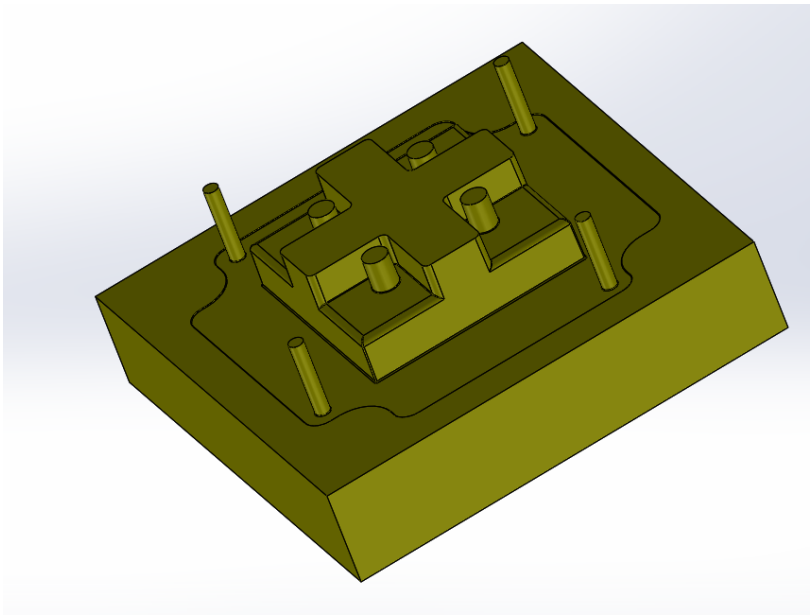


Figure 3.26: Core

3.6.3 Material and Mass Properties

Mass Properties of The Bottom Part Using ABS Material

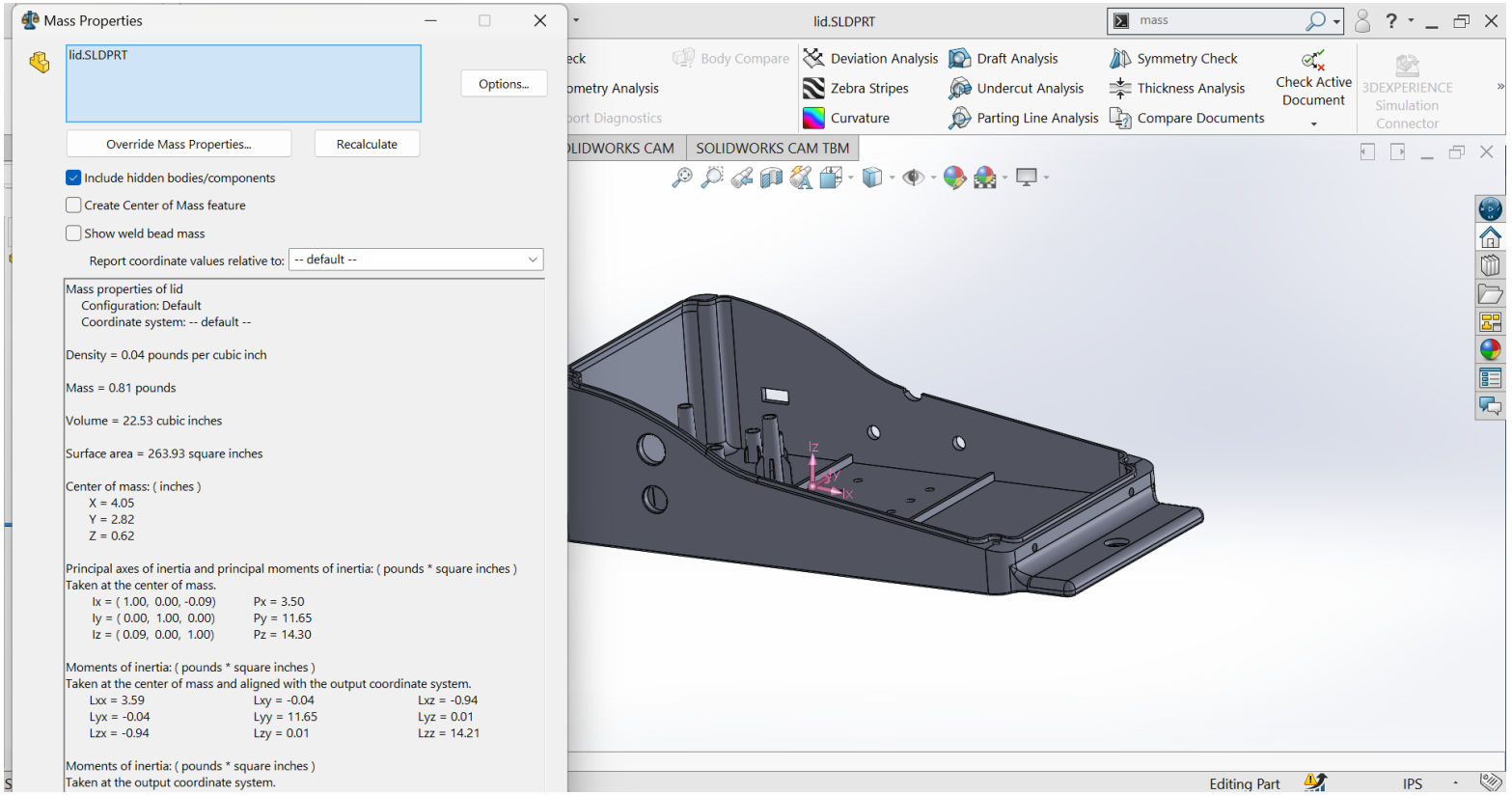


Figure 3.27: Mass Properties of The Bottom Part Using ABS Material

Mass Properties of The Top part Using ABS Material

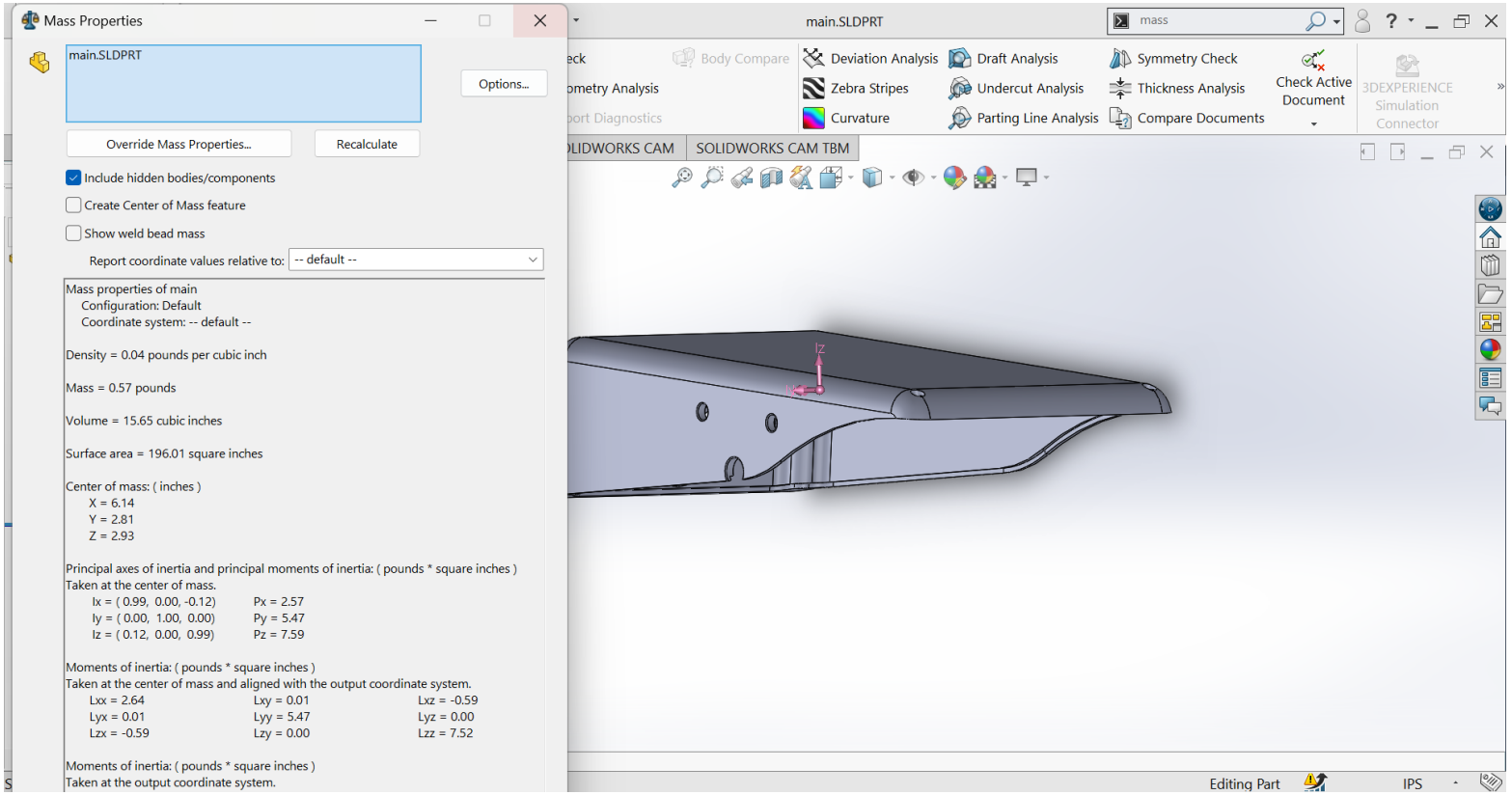


Figure 3.28: Mass Properties of The Top part Using ABS Material

Mass Properties of The Feedback Using ABS Material

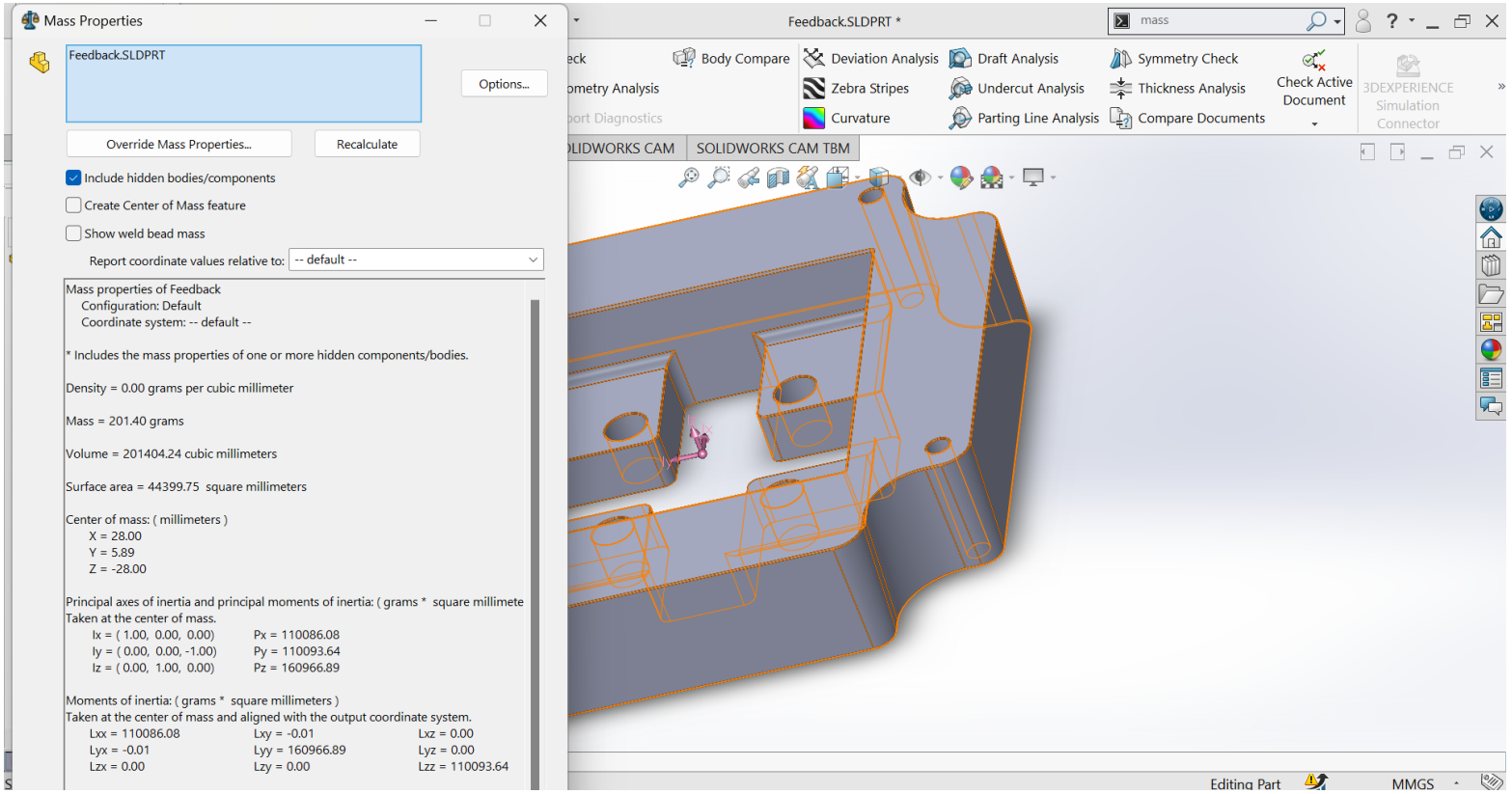
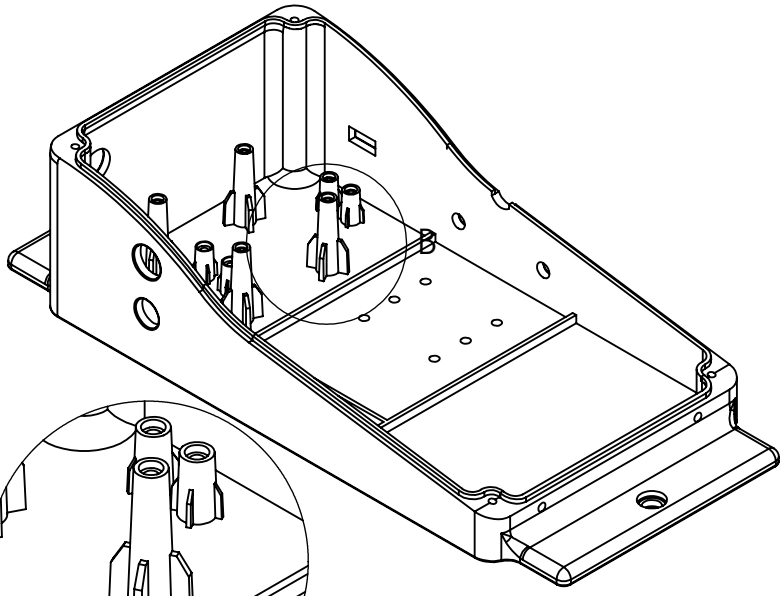
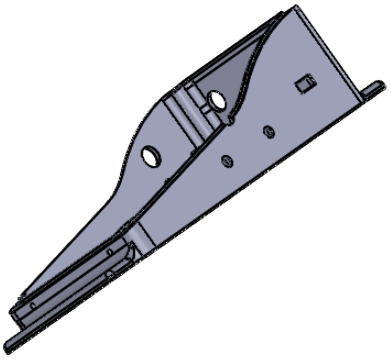
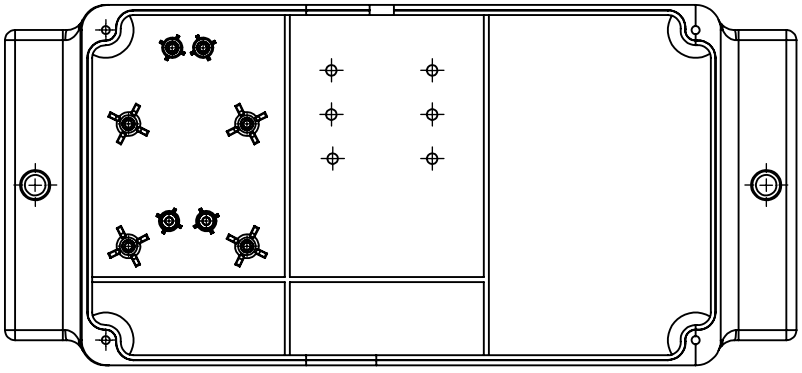


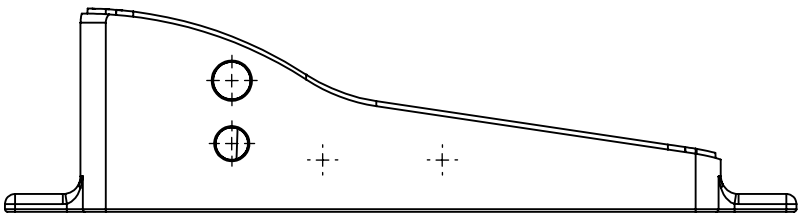
Figure 3.29: Mass Properties of The Feedback Using ABS Material





DETAIL B
SCALE 2 : 3





UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING / REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

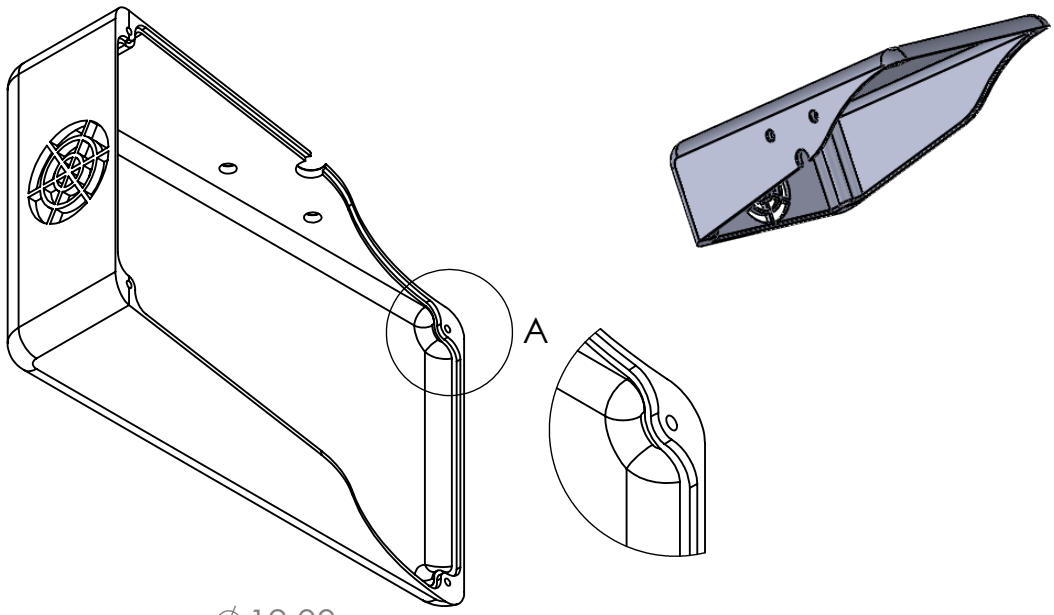
DWG NO. lid

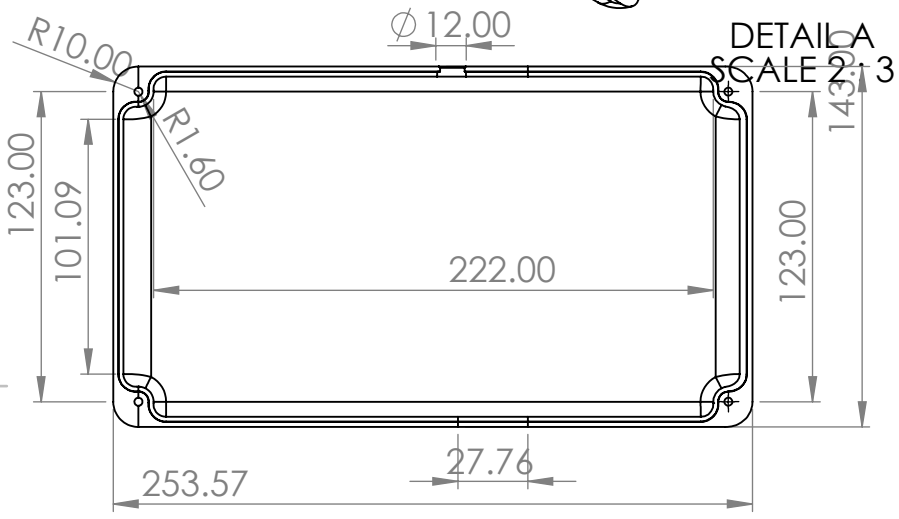
A4

WEIGHT:

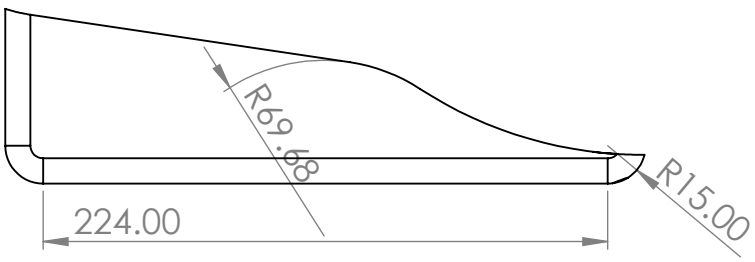
SCALE:1:5

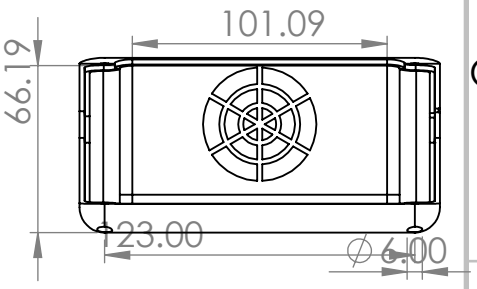
SHEET 1 OF 1



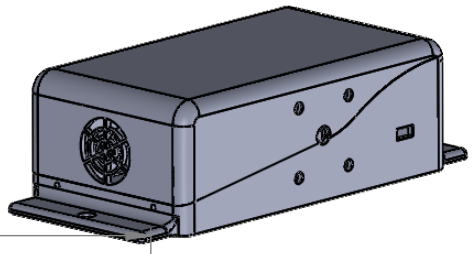


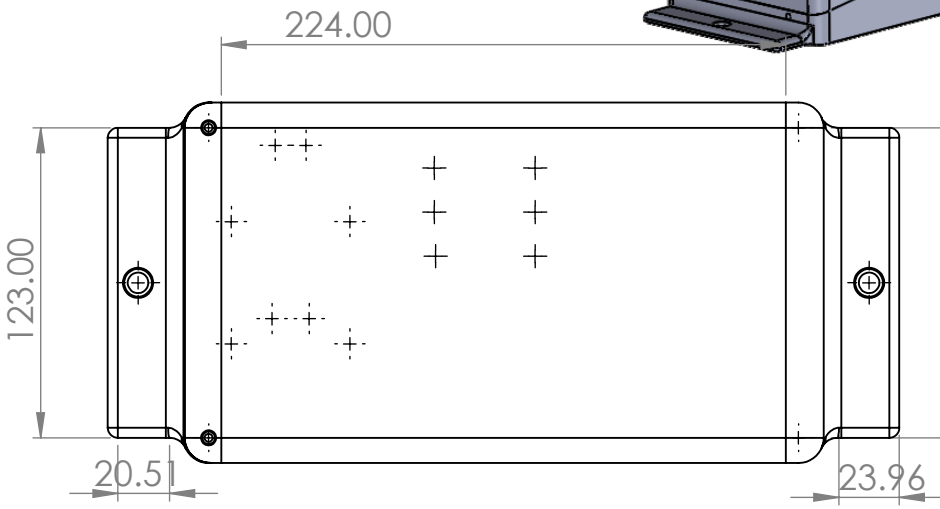
DETAIL A
SCALE 2 : 3

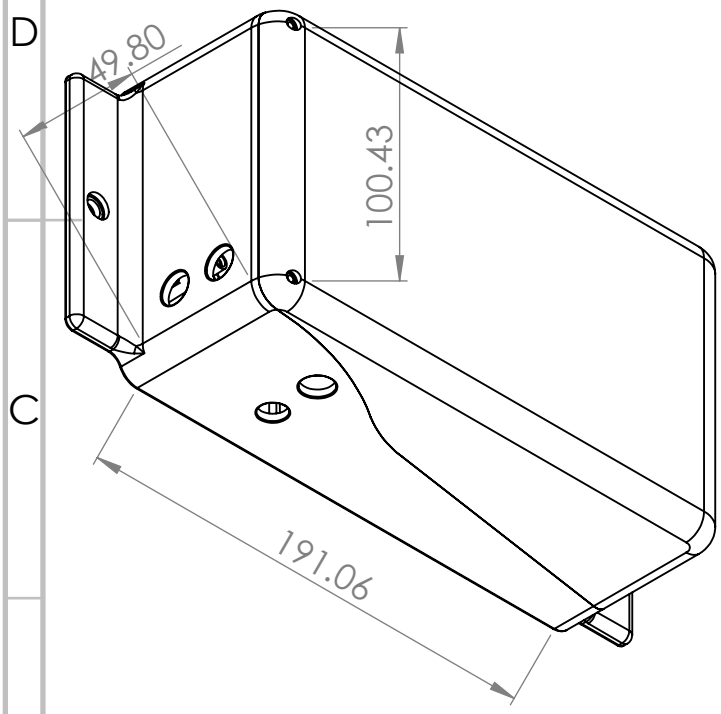


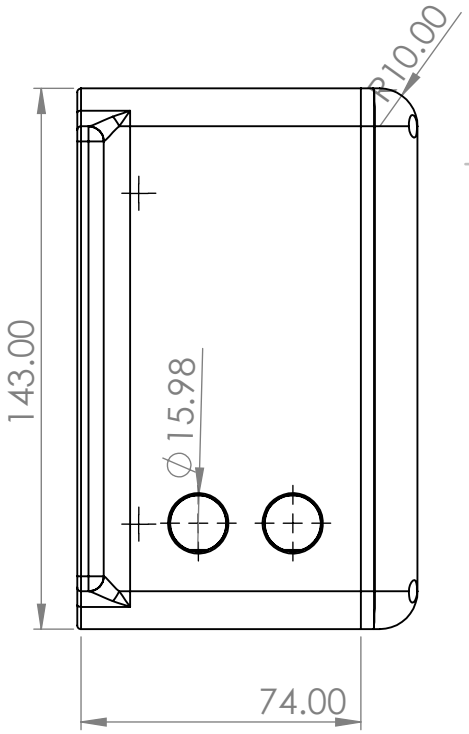


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE				TITLE:
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO.	A4
						Final	
				WEIGHT:		SCALE:1:5	SHEET 1 OF 1









UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING / REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

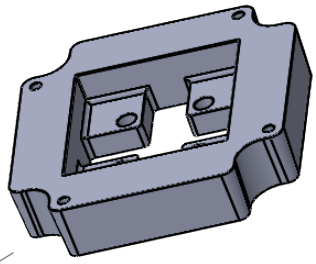
DWG NO. Final

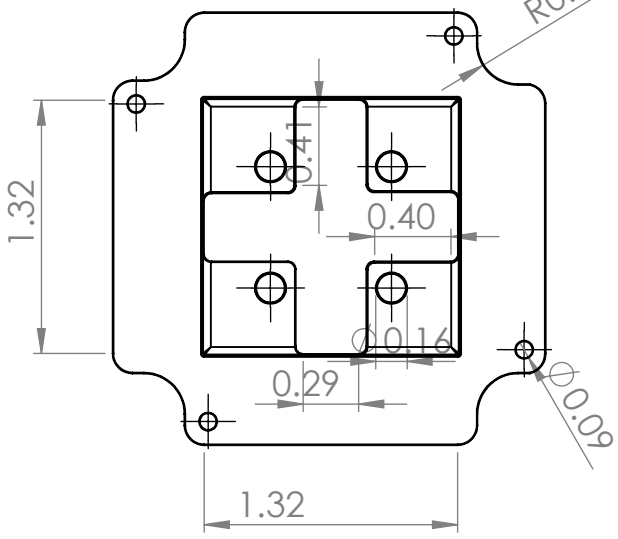
A4

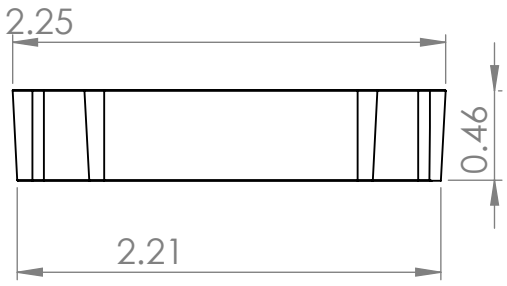
WEIGHT:

SCALE:1:5

SHEET 1 OF 1







UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:		DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION		
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. Feedback	A4
				WEIGHT:		SCALE:1:2	SHEET 1 OF 1

4 Software

The software development for this project primarily utilized the ATmega328P microcontroller, programmed using the Win AVR. The ATmega328P is a widely used microcontroller in embedded systems due to its versatility and ease of use.

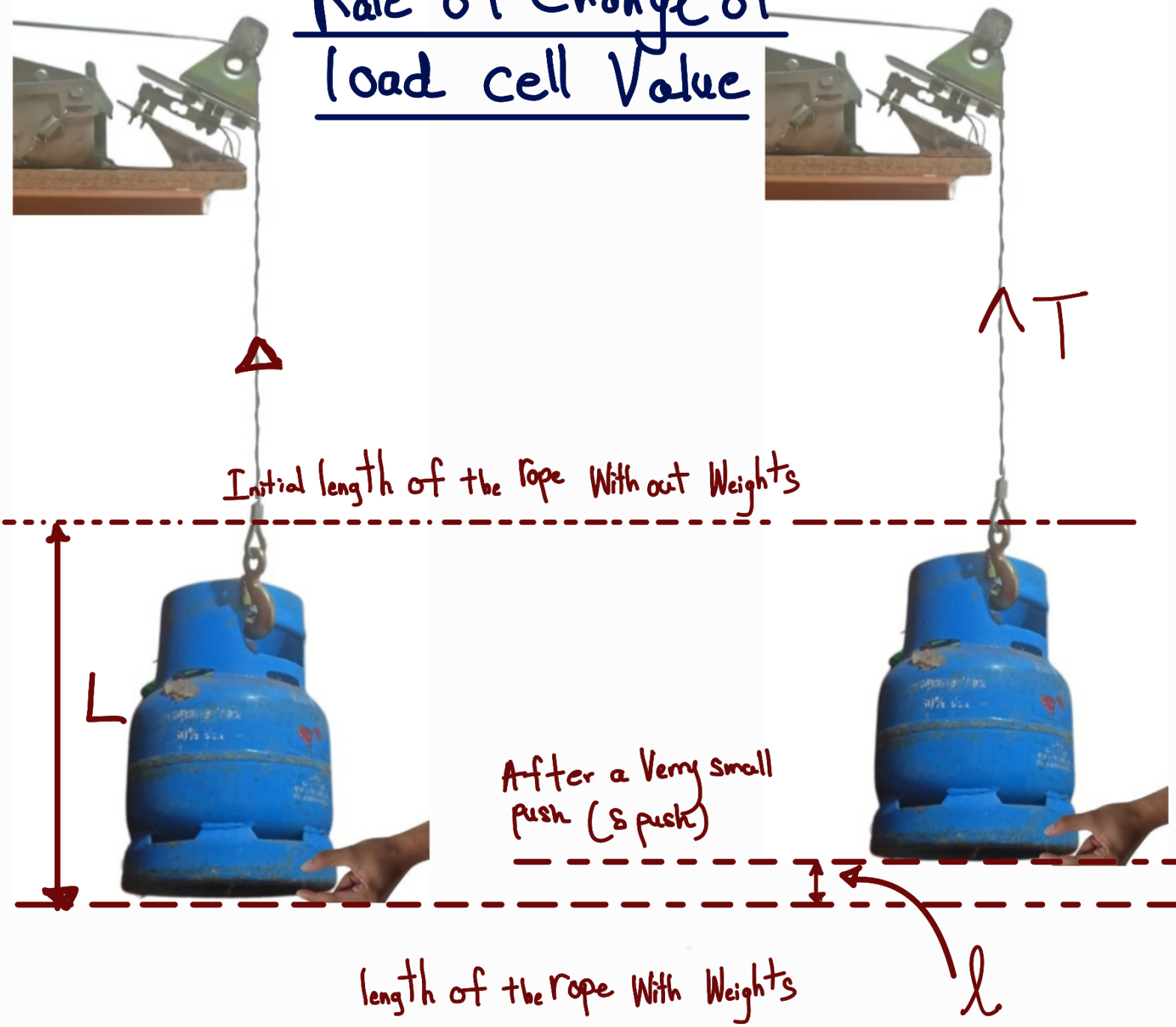
4.1 Microcontroller Details

The ATmega328P microcontroller features:

- 8-bit AVR architecture
- Clock speed up to 20 MHz
- 32KB Flash memory for program storage
- 2KB SRAM and 1KB EEPROM for data storage
- Integrated peripherals including timers, UART, SPI, and I2C interfaces

4.2 Mathematical Proof and Simulation of the Algorithm

Rate of change of load cell Value



$T = K(L - l)$ (K, Spring constant)

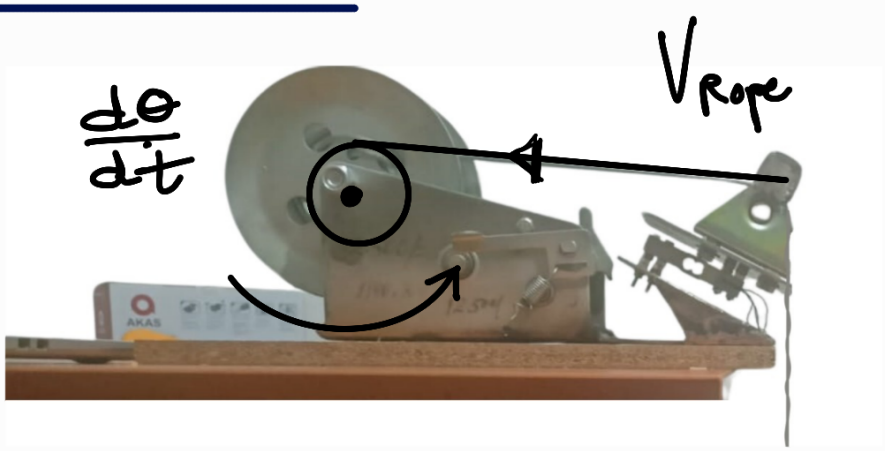
$\dot{T} = -K\dot{l}$

$$\frac{dT}{dt} = K\frac{dl}{dt} \quad \text{— (1)}$$

$\therefore$ $\frac{dl}{dt}$ = Velocity of the string − Velocity of the hand

$$\frac{dT}{dt} = K\left(V_{Rope} - V_{hand}\right)$$

Velocity of the Rope



$$V_{rope} = \frac{d\theta}{dt} \cdot r = 2\pi \frac{dN}{dt} \cdot r$$

$$= 2\pi n r \quad \text{——} (2)$$

$$n = rps = pps \times \frac{360^\circ}{\text{Angle per pulse}}$$

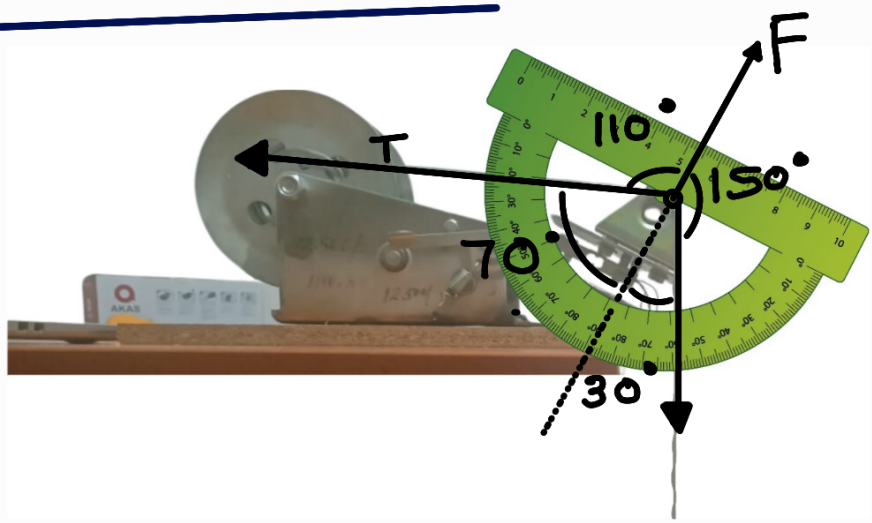
$$= pps \times \frac{360^\circ}{1.8^\circ}$$

$$n = rps = 200pps \quad \text{——} (4)$$

$$V_{rope} = 2\pi r \times 200\,[pps]$$

$$= 400\pi r(pps)$$

$\frac{df}{dt}$ as a function of pps



$$F = T(\cos 30^\circ + \cos 70^\circ)$$

$$F = (\cos 70^\circ + \cos 30^\circ)T = \alpha T$$

$$\frac{dT}{dt} = \frac{dF}{dt} \times 1/\alpha \quad \text{——} (5)$$

(5) = (5)

$$1/\alpha \frac{df}{dt} = k(400\pi r\{pps\} - V_{hand})$$

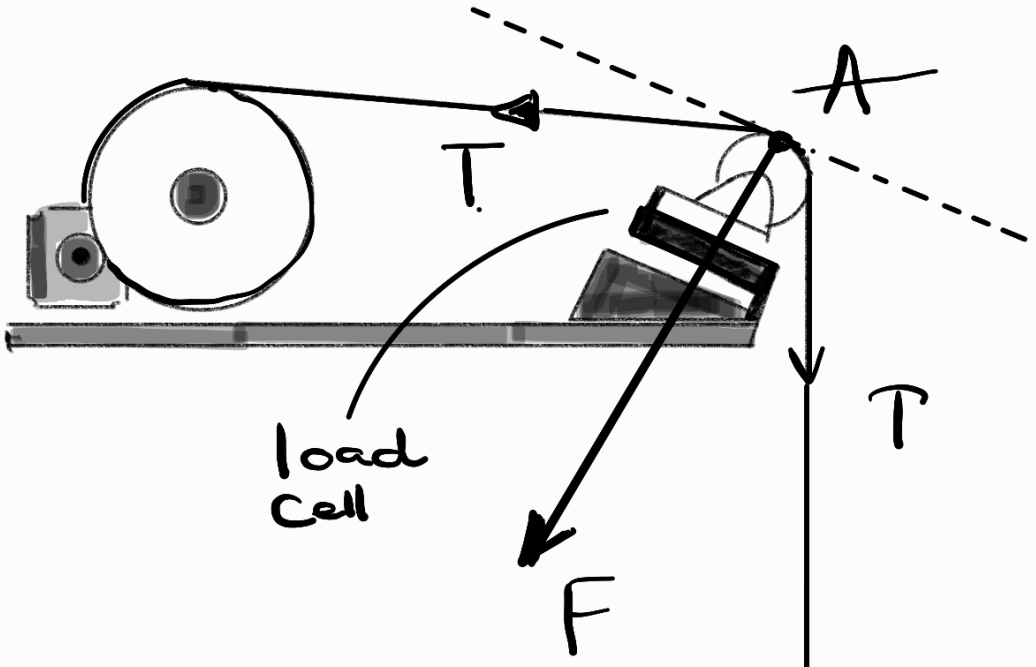
$$\frac{df}{dt} = 400\pi r k\alpha\{pps\} - k\alpha V_{hand}$$

$\frac{df}{dt}$ = Rate of change of load cell value

r - Radius

$\alpha = (\cos 30^\circ + \cos 70^\circ)$

k = spring constant of the Rope



$F \propto T$

$$\frac{dF}{dt} \propto \frac{dT}{dt}$$

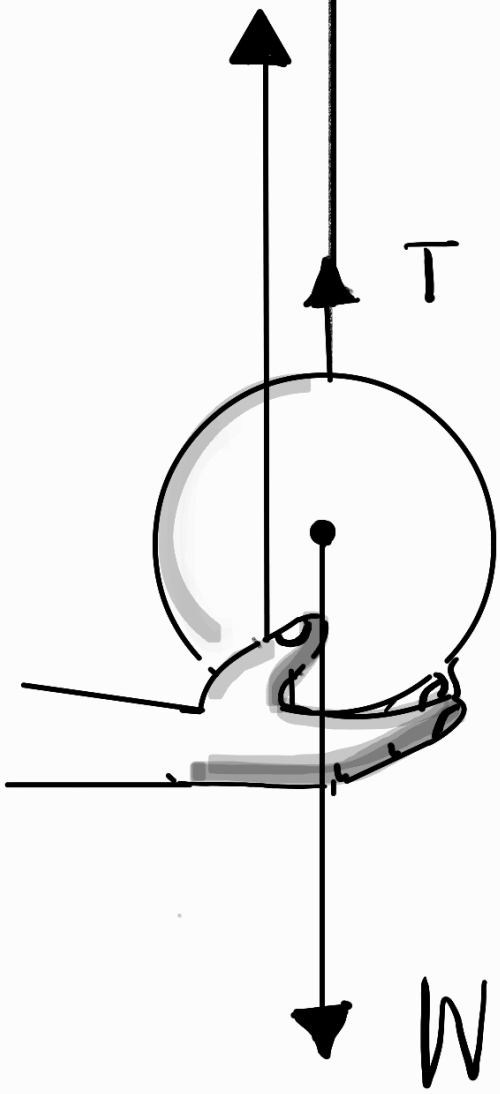
for zero gravity
Effect

$F_{hand} \simeq 0$

$T = W$

$$\frac{dT}{dt} = 0$$

$$\frac{df}{dt} = 0$$



We need to maintain this $\frac{df}{dt} = 0$

so We thought to use a PID Controler

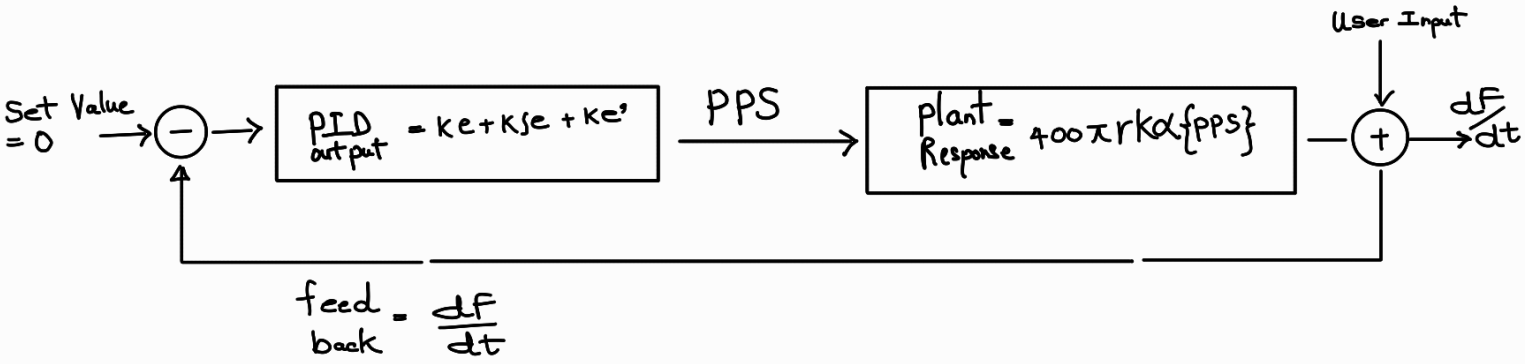
$\therefore$ the set Value of the PID Controller Should be 0

$$error = 0 - \frac{df}{dt}$$

$$e = -\frac{df}{dt}$$

$$\text{PID}_{\text{output}} = Ke + K\int e + Ke'$$

$$\text{plant Response} = 400\pi rk\alpha \{PPS\}$$



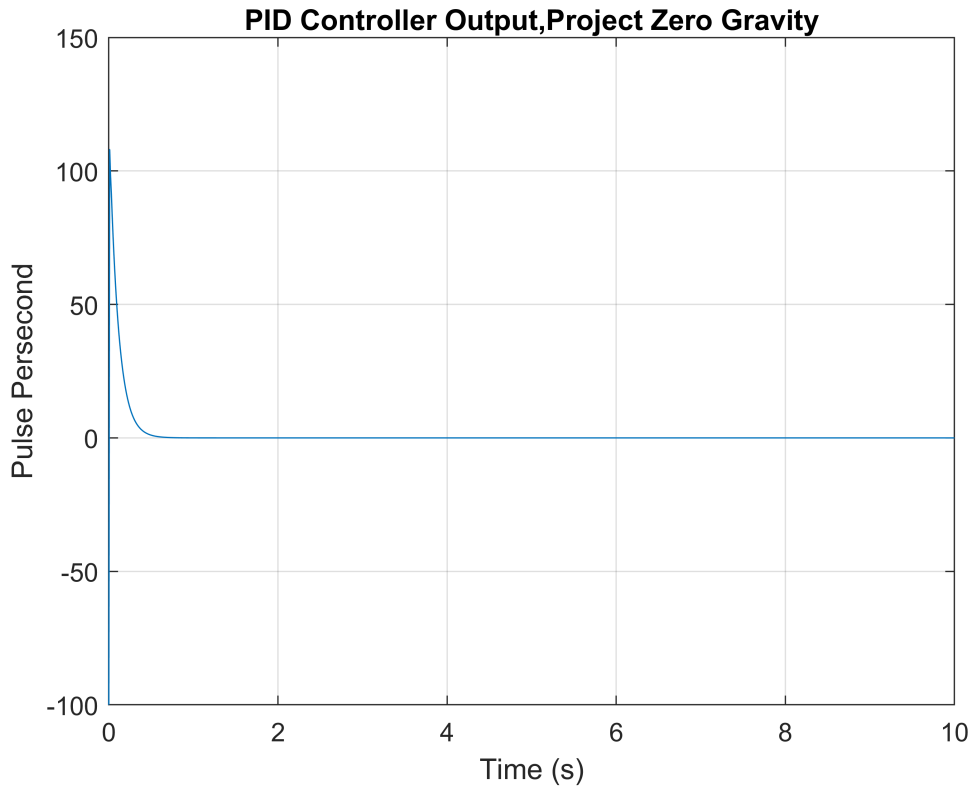
the discreate Implementation of PID

F'_{n+1} = new error f'_n = previous error $f_n = -\frac{df}{dt_n}$

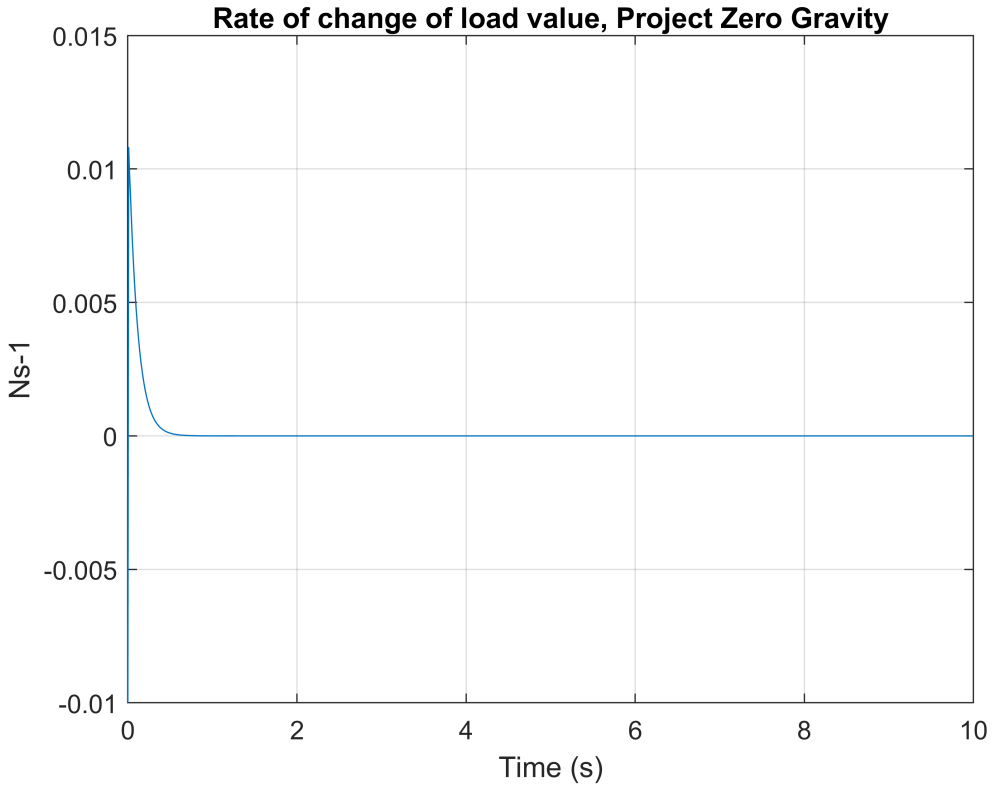
$$F'_{n+1} = 400\pi rk\alpha \left\{ K_p F'_n + K_I \sum_1^n F'_n + K_D (F'_n - F'_{n-1}) + U \right\}$$

Simulation of the Algorithm to get the Zero Gravity effect

```
% PID controller gains
Kp = -0.01; % Proportional gain "Need to be practicaly calculated"
Ki = 9; % Integral gain  "Need to be practicaly calculated"
Kd = 0.0001; % Derivative gain  "Need to be practicaly calculated"

% Simulation parameters
dt = 0.01; % Time step
t_end = 10; % End time
time = 0:dt:t_end; % Time vector

% Initial conditions
set_value = 0; % Desired rate of change of lord value is 0 to get Zero Gravity effect
external_impact_by_user = -10;% external impact on rate of change of lord value
previous_error = 0; % Previous error
Plant_gain = 0.0001; %"Need to be practicaly calculated"
integral = 0; % Integral term
derivative = 0; % Derivative term
output = zeros(size(time)); % Output array
output(1) = external_impact;

% PID controller loop
for i = 1:length(time)-1
    % Compute error
    error = set_value - output(i);

    % Compute integral term
    integral = integral + error * dt;

    % Compute derivative term
    if i > 1
        derivative = (error - previous_error) / dt;
    end

    % Compute PID output
    pid_output = Kp * error + Ki * integral + Kd * derivative;

    % Update the system output (this example assumes a simple system model)
   output(i+1) = pid_output - external_impact; % You can replace this with your system's respon
    % Store current error for next iteration
    previous_error = error;
end

% Plot
plot(time, output);
xlabel('Time (s)');
ylabel('Pulse Persecond');
title('PID Controller Output,Project Zero Gravity');
grid on;
```




```
% Plot
plot(time, output*Plant_gain);
xlabel('Time (s)');
ylabel('Ns-1');
title('Rate of change of load value, Project Zero Gravity');
grid on;
```


Rate of change of load value, Project Zero Gravity
0.015
0.01
0.005
0
-0.005
-0.01
Ns-1
0
2
4
6
8
10
Time (s)

4.3 Implementation of the Algorithm in a WinAVR code

4.3.1 millis.h

```
/*
 * Project: Lightweight millisecond tracking library
 * Author: Zak Kemble, contact@zakkemble.co.uk
 * Copyright: (C) 2013 by Zak Kemble
 * License: GNU GPL v3 (see License.txt)
 * Web: http://blog.zakkemble.co.uk/millisecond-tracking-library-for-avr/
 */

#ifndef MILLIS_H_
#define MILLIS_H_

/**
* Milliseconds data type \n
* Data type       - Max time span     - Memory used \n
* unsigned char     - 255 milliseconds    - 1 byte \n
* unsigned int      - 65.54 seconds     - 2 bytes \n
* unsigned long     - 49.71 days      - 4 bytes \n
* unsigned long long  - 584.9 million years - 8 bytes
*/
typedef unsigned long millis_t;

#define MILLIS_TIMER0 0 /**< Use timer0. */
#define MILLIS_TIMER1 1 /**< Use timer1. */
#define MILLIS_TIMER2 2 /**< Use timer2. */

#define MILLIS_TIMER MILLIS_TIMER2 /**< Which timer to use. */

/**
* Alias of millis_get().
*
* @note Not availble for  since millis() is already used.
*/
#define millis() millis_get()
#endif
#ifdef __cplusplus
extern "C" {
#endif

/**
* Initialise, must be called before anything else!
*
* @return (none)
*/
void millis_init(void);

/**
* Get milliseconds.
*
* @return Milliseconds.
*/
millis_t millis_get(void);

/**
* Turn on timer and resume time keeping.
*
* @return (none)
*/
void millis_resume(void);

/**
* Pause time keeping and turn off timer to save power.
*
* @return (none)
```

```
*/
void millis_pause(void);

/**
* Reset milliseconds count to 0.
*
* @return (none)
*/
void millis_reset(void);

/**
* Add time.
*
* @param [ms] Milliseconds to add.
* @return (none)
*/
void millis_add(millis_t ms);

/**
* Subtract time.
*
* @param [ms] Milliseconds to subtract.
* @return (none)
*/
void millis_subtract(millis_t ms);

#ifdef __cplusplus
}
#endif

#endif /* MILLIS_H_ */
}
```

Listing 4.1: Lightweight millisecond tracking library

4.3.2 HX711 . h

```
#ifndef HX711_h
#define HX711_h

    #define PD_SCK_PORT         PORTD                           // Power Down and
    Serial Clock Input Port
    #define PD_SCK_DDR          DDRD                              // Power Down and
    Serial Clock DDR
    #define PD_SCK_PIN          PD5                               // Power Down and
    Serial Clock Pin

    #define PD_SCK_SET_OUTPUT   PD_SCK_DDR |= (1<<PD_SCK_PIN)

    #define PD_SCK_SET_HIGH     PD_SCK_PORT |= (1<<PD_SCK_PIN)
    #define PD_SCK_SET_LOW      PD_SCK_PORT &= ~(1<<PD_SCK_PIN)

    #define DOUT_PORT           PORTD                             // Serial Data Output
     Port
    #define DOUT_DDR            DDRD                              // Serial Data Output
     DDR
    #define DOUT_INPUT          PIND                              // Serial Data Output
     Input
    #define DOUT_PIN            PD6                             // Serial Data Output Pin
    #define DOUT_READ           (DOUT_INPUT & (1<<DOUT_PIN))    // Serial Data Output
     Read Pin

    #define DOUT_SET_HIGH       DOUT_PORT |= (1<<DOUT_PIN)
    #define DOUT_SET_LOW        DOUT_PORT &= ~(1<<DOUT_PIN)
    #define DOUT_SET_INPUT      DOUT_DDR &= ~(1<<DOUT_PIN); DOUT_SET_HIGH
    #define DOUT_SET_OUTPUT     DOUT_DDR |= (1<<DOUT_PIN); DOUT_SET_LOW
```

```
23
24      uint8_t GAIN;                   // amplification factor
25      double OFFSET;                  // used for tare weight
26      float SCALE;                      // used to return weight in grams, kg, ounces,
        whatever
27
28    // define clock and data pin, channel, and gain factor
29    // channel selection is made by passing the appropriate gain: 128 or 64 for channel
        A, 32 for channel B
30    // gain: 128 or 64 for channel A; channel B works with 32 gain factor only
31    void HX711_init(uint8_t gain);
32
33    // check if HX711 is ready
34    // from the datasheet: When output data is not ready for retrieval, digital output
        pin DOUT is high. Serial clock
35    // input PD_SCK should be low. When DOUT goes to low, it indicates data is ready
        for retrieval.
36    int HX711_is_ready(void);
37
38    // set the gain factor; takes effect only after a call to read()
39    // channel A can be set for a 128 or 64 gain; channel B has a fixed 32 gain
40    // depending on the parameter, the channel is also set to either A or B
41    void HX711_set_gain(uint8_t gain);
42
43    // waits for the chip to be ready and returns a reading
44    uint32_t HX711_read(void);
45
46    // returns an average reading; times = how many times to read
47    uint32_t HX711_read_average(uint8_t times);
48
49    // returns (read_average() - OFFSET), that is the current value without the tare
        weight; times = how many readings to do
50    double HX711_get_value(uint8_t times);
51
52    // returns get_value() divided by SCALE, that is the raw value divided by a value
        obtained via calibration
53    // times = how many readings to do
54    float HX711_get_units(uint8_t times);
55
56    // set the OFFSET value for tare weight; times = how many times to read the tare
        value
57    void HX711_tare(uint8_t times);
58
59    // set the SCALE value; this value is used to convert the raw data to "human
        readable" data (measure units)
60    void HX711_set_scale(float scale);
61
62    // get the current SCALE
63    float HX711_get_scale(void);
64
65    // set OFFSET, the value that's subtracted from the actual reading (tare weight)
66    void HX711_set_offset(double offset);
67
68    // get the current OFFSET
69    double HX711_get_offset(void);
70
71    // puts the chip into power down mode
72    void HX711_power_down(void);
73
74    // wakes up the chip after power down mode
75    void HX711_power_up(void);
76
77      // Sends/receives data. Modified from  source
78    uint8_t shiftIn(void);
79
80    unsigned long HX711_Read(void);
81
82  #endif /* HX711_h */
```

```
}
```

Listing 4.2: This is an advanced library which has been tested with 20kg load cell and can measure 1 gram minimum if proper change rate setting is applied.

4.3.3 Main.c

```
#include <stdint.h>
#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>
#include "HX711.h"
#include "millis.h"

// Define pins for HX711 load cell amplifier
#define HX711_DT_PIN  PC4  // Data pin
#define HX711_SCK_PIN PC3  // Clock pin

// Define pins for stepper motor control
#define STEP_PIN PD6
#define DIR_PIN PD7

// Stepper motor configuration
#define STEPS_PER_REVOLUTION 200
#define DELAY_MICROSECONDS 1000 // Adjust this value to control the motor speed

// PID controller constants
#define P_FACTOR 1
#define I_FACTOR 0
#define D_FACTOR 1
#define SCALING_FACTOR 128
#define MAX_INT 32767
#define MAX_I_TERM (MAX_INT / 2)

// Struct to hold PID data
struct PID_DATA {
    int16_t sumError;
    int16_t lastProcessValue;
    int16_t P_Factor;
    int16_t I_Factor;
    int16_t D_Factor;
    int16_t maxError;
    int16_t maxSumError;
};

// Initialize HX711 module
void HX711_init() {
    // Set SCK pin as output
    DDRD |= (1 << HX711_SCK_PIN);
    // Set DT pin as input
    DDRD &= ~(1 << HX711_DT_PIN);
}

// Read data from HX711 module
uint32_t HX711_read() {
    uint32_t count = 0;
    while (PIND & (1 << HX711_DT_PIN)); // Wait until DT goes low
    for (uint8_t i = 0; i < 24; i++) {
        PORTD |= (1 << HX711_SCK_PIN); // Set SCK high
        count = count << 1;
        PORTD &= ~(1 << HX711_SCK_PIN); // Set SCK low
        if (PIND & (1 << HX711_DT_PIN)) {
            count++;
        }
    }
    PORTD |= (1 << HX711_SCK_PIN); // Set SCK high for the 25th pulse
```

```
    count = count ^ 0x800000; // Make the count signed
    PORTD &= ~(1 << HX711_SCK_PIN); // Set SCK low
    return count;
}

// Initialize PID controller
void pid_Init(int16_t p_factor, int16_t i_factor, int16_t d_factor, struct PID_DATA *pid) {
    // Set initial values for PID controller
    pid->sumError = 0;
    pid->lastProcessValue = 0;
    pid->P_Factor = p_factor;
    pid->I_Factor = i_factor;
    pid->D_Factor = d_factor;
    pid->maxError = MAX_INT / (pid->P_Factor + 1);
    pid->maxSumError = MAX_I_TERM / (pid->I_Factor + 1);
}

// PID controller function
int16_t pid_Controller(int16_t setPoint, int16_t processValue, struct PID_DATA *pid_st) {
    // Calculate error
    int16_t errors = setPoint - processValue;

    // Calculate P term
    int16_t p_term;
    if (errors > pid_st->maxError) {
        p_term = MAX_INT;
    } else if (errors < -pid_st->maxError) {
        p_term = -MAX_INT;
    } else {
        p_term = pid_st->P_Factor * errors;
    }

    // Calculate I term
    int32_t i_term;
    int32_t temp = pid_st->sumError + errors;
    if (temp > pid_st->maxSumError) {
        i_term = MAX_I_TERM;
        pid_st->sumError = pid_st->maxSumError;
    } else if (temp < -pid_st->maxSumError) {
        i_term = -MAX_I_TERM;
        pid_st->sumError = -pid_st->maxSumError;
    } else {
        pid_st->sumError = temp;
        i_term = pid_st->I_Factor * pid_st->sumError;
    }

    // Calculate D term
    int16_t d_term = pid_st->D_Factor * (pid_st->lastProcessValue - processValue);

    pid_st->lastProcessValue = processValue;

    // Calculate final output
    int32_t ret = (p_term + i_term + d_term) / SCALING_FACTOR;
    if (ret > MAX_INT) {
        ret = MAX_INT;
    } else if (ret < -MAX_INT) {
        ret = -MAX_INT;
    }

    return ((int16_t)ret);
}

// Reset PID integrator
void pid_Reset_Integrator(struct PID_DATA *pid_st) {
    pid_st->sumError = 0;
}
```

```
126
127 // Setup stepper motor pins
128 void setup_stepper() {
129     DDRB |= (1 << STEP_PIN) | (1 << DIR_PIN);
130 }
131
132 // Control stepper motor based on velocity
133 void loop_stepper(double velocity) {
134     // Convert velocity to steps per second
135     uint16_t steps_per_second = (uint16_t)(velocity * STEPS_PER_REVOLUTION);
136
137     // Set direction based on velocity sign
138     if (velocity >= 0) {
139         PORTB &= ~(1 << DIR_PIN);
140     } else {
141         PORTB |= (1 << DIR_PIN);
142     }
143
144     // Generate step pulses
145     for (uint16_t i = 0; i < steps_per_second; i++) {
146         PORTB |= (1 << STEP_PIN);
147         _delay_us(DELAY_MICROSECONDS / 2);
148         PORTB &= ~(1 << STEP_PIN);
149         _delay_us(DELAY_MICROSECONDS / 2);
150     }
151 }
152
153 int main(void) {
154     // Initialize USART, millis, and enable interrupts
155     USART_Init(MYUBRR);
156     millis_init();
157     sei();
158
159     // Initialize and calibrate HX711
160     HX711_init();
161     HX711_set_scale(1.f);
162     HX711_set_gain(128);
163     HX711_tare(10);
164     double tare_point_128 = HX711_get_offset();
165     double calibration_128 = 222138.f;
166     _delay_ms(500);
167
168     // Initialize PID controller
169     struct PID_DATA pid;
170     pid_Init(P_FACTOR, I_FACTOR, D_FACTOR, &pid);
171
172     // Variables for weight derivative calculation
173     double derivative = 0.0, previous_weight_128 = 0.0;
174     millis_t time_mid, time_start = millis();
175
176     // Setup stepper motor
177     setup_stepper();
178
179     while(1) {
180         // Read current weight from HX711
181         double current_weight_128 = HX711_read() - tare_point_128;
182         current_weight_128 = current_weight_128 / calibration_128;
183
184         // Calculate time elapsed and weight derivative
185         time_mid = millis();
186         derivative = (current_weight_128 - previous_weight_128) / (time_mid -
    time_start);
187
188         time_start = time_mid;
189         previous_weight_128 = current_weight_128;
190
191         // Calculate PID output using weight derivative as error
192         int16_t pid_output = pid_Controller(0, (int16_t)(derivative * SCALING_FACTOR)
```

```
    , &pid);

        // Convert PID output to velocity
        double velocity = (double)pid_output / SCALING_FACTOR;

        // Control stepper motor based on calculated velocity
        loop_stepper(velocity);

        // Print debug information
        printf("D = %.5f, PID Output = %d\n", derivative, pid_output);
    }

    return 0;
}
```

Listing 4.3: This code implements a control system that uses a load cell (via HX711 module) to measure rate of change of tention, processes this information through a PID controller, and uses the output to control a stepper motor's velocity

Appendices

A Project Daily Progress

Week 1 (January 29 - February 4, 2024):

- **February 1**: Initiated the Zero Gravity Lifting project which was proposed by Prof. Jayasinghe with an official proposal. The team discussed project objectives, potential impacts, and set initial goals.

Week 2 (February 5 - February 11, 2024):

- **February 3**: Met on zoom with Professor Jayasinghe to discuss industrial applications and servo motors. His insights were invaluable in shaping our project direction.
- **February 7**: Conducted comprehensive research on zero gravity technology, gathering information from academic papers and industry reports.
- **February 9**: Continued research, deepening our understanding of the complexities involved in zero gravity systems.

Week 3 (February 12 - February 18, 2024):

- **February 12**: Expanded our team to include four members, selecting individuals with complementary skills to tackle various project challenges.
- **February 14**: Held a team meeting to assign roles and responsibilities, ensuring each member had clear tasks aligned with their expertise.
- **February 16**: Engaged in a brainstorming session to sketch initial mechanical structure ideas and potential component lists.

Week 4 (February 19 - February 25, 2024):

- **February 19**: Refined mechanical structure concepts through a collaborative team session, integrating different ideas and approaches.
- **February 21**: Compiled a list of potential components, researching materials and electronics to enhance the device's performance.
- **February 23**: Met with Professor Jayasinghe again, receiving guidance on integrating multiple engineering principles into our design.

Week 5 (February 26 - March 3, 2024):

- **February 27**: Studied various engineering principles, focusing on their application to our project, including control systems and biomechanics.
- **February 29**: Held a team meeting to incorporate new knowledge into our design concepts, evaluating different principles' impact on device performance.

- **March 1**: Dedicated the day to intensive research on existing zero gravity devices, analyzing their strengths and weaknesses.

Week 6 (March 4 - March 10, 2024):

- **March 4**:Performed a stakeholder analysis, considering how our device could best serve the industry .Compiled feedback and requirements from potential stakeholders, gaining insights into real-world needs. Planned the next steps to be done assigning tasks among team members.

- **March 6**: Reviewed key research papers and datasheets, extracting insights on current technologies and methodologies in zero gravity lifting devices.

- **March 7**: Synthesized research findings into a comprehensive document to guide our future design decisions.

- **March 8**: Developed three distinct solutions based on different engineering principles, each offering a unique approach to achieving zero gravity.

Week 7 (March 11 - March 17, 2024):

- **March 11**: Sketched three mechanical structures, each representing a different concept for our lifting device, showcasing team creativity.

- **March 13**: Conducted a thorough review of our sketches, critically evaluating each design and refining our ideas further.

- **March 15**: Analyzed specific research papers to extract key insights relevant to our project.

Week 8 (March 18 - March 24, 2024):

- **March 18**: Discussed paper analysis findings as a team, identifying applicable concepts for our design.

- **March 20**: Began drafting initial design specifications, translating research and discussions into concrete parameters.

- **March 23**: Presented our conceptual designs to Prof. Jayasinghe. His insights and ideas were valuable in selecting final design.

Week 9 (March 25 - March 31, 2024):

- **March 25**: Finalized our device's schematic, bringing our concept closer to reality. Started work on the PCB layout, carefully considering component placement and connections. Started focusing on implementation of the selected mechanical design

- **March 28**:Adjusted design specifications based on stakeholder analysis, ensuring our device meets market demands. Conducted a team review of the schematic and initial PCB layout, ensuring alignment with our design goals.

- **March 29**: Further refined the PCB. Began designing the device's enclosure, focusing on functionality and manufacturability.

Week 10 (April 1 - April 7, 2024):

- **April 1**: Created 3D models of our enclosure design, visualizing our concepts in a virtual space.
- **April 2**: Held a team review session to refine the enclosure design, balancing aesthetics with practicality.
- **April 3**: Presented our PCB layout to Professor Jayasinghe for expert feedback.

Week 11 (April 8 - April 14, 2024):

- **April 8**: Implemented the professor's suggestions, fine-tuning our PCB design for optimal performance.
- **April 9**: Made final adjustments to the PCB design, optimizing trace widths and copper usage. Completed the PCB layout, a significant milestone in our electronic design phase.

Week 12 (April 15 - April 21, 2024):

- **April 15**: Conducted a final team review of the PCB design, ensuring it met all requirements and standards.
- **April 17**: Prepared a detailed bill of materials, listing all components needed for our prototype.
- **April 19**: Sent our finalized PCB layout to JLCPCB for manufacturing, marking a major step towards realizing our design.

Week 13 (April 22 - April 28, 2024):

- **April 22**:Placed orders for all necessary components from Mouser.com, selecting parts for quality and compatibility. Started documentation.
- **April 24**: Began breadboard implementation of critical circuits, testing and refining designs.
- **April 26**: Started assembling the mechanical structure using steel components, bringing our physical design to life.
- **April 28**:Continued breadboard testing, making adjustments based on performance.

Week 14 (April 29 - May 5, 2024):

- **April 30**: Received manufactured PCBs and ordered components. Began soldering and initial assembly.
- **May 1**: Completed PCB soldering and conducted initial electrical tests to ensure circuit functionality.
- **May 4**: Integrated the PCB with the mechanical structure, seeing our device take its final form.

Week 15 (May 6 - May 12, 2024):

- **May 6**: Performed first full system test, observing the device in action.
- **May 8**: Completed documentation. Refined the device's software, optimizing performance and user interface.

June 1 - June 7, 2024:

- Micro-controller Programming using WinAVR

B Project Photographs

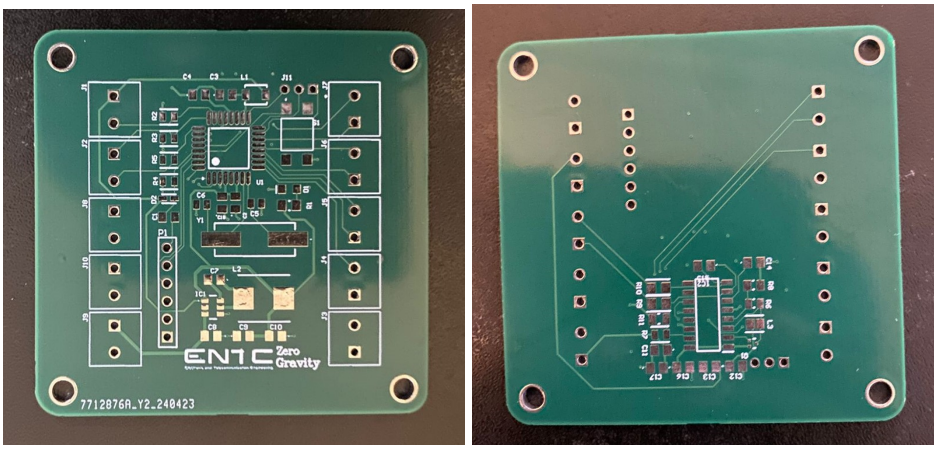


Figure B.1: Bare PCB- Front and Back

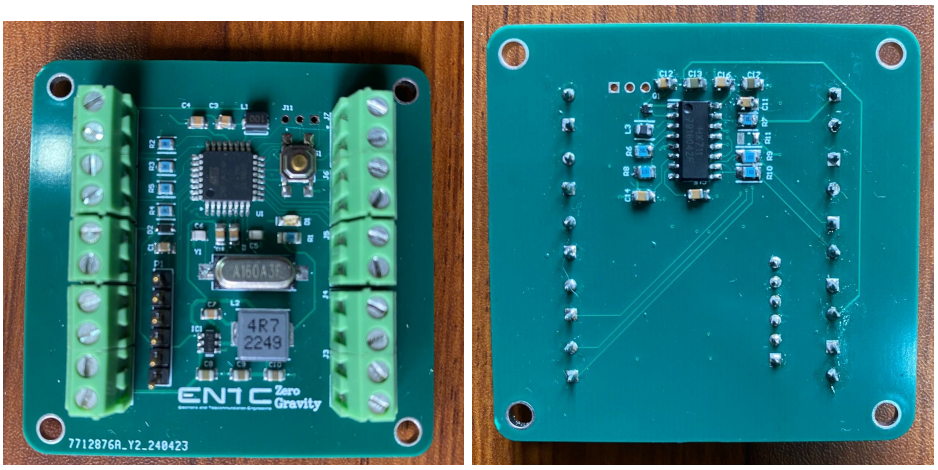


Figure B.2: Soldered PCB- Front and Back

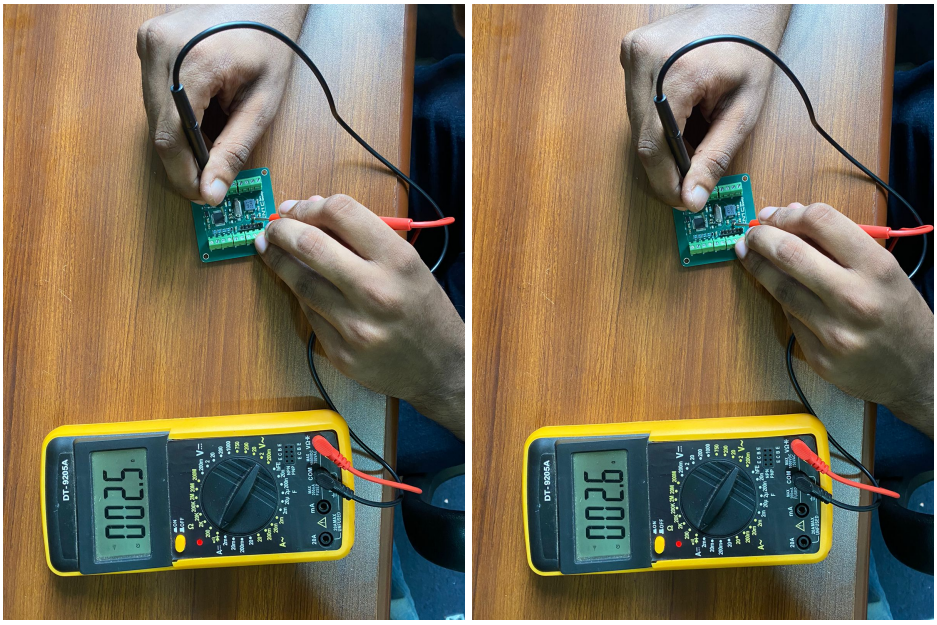


Figure B.3: PCB Continuity testing

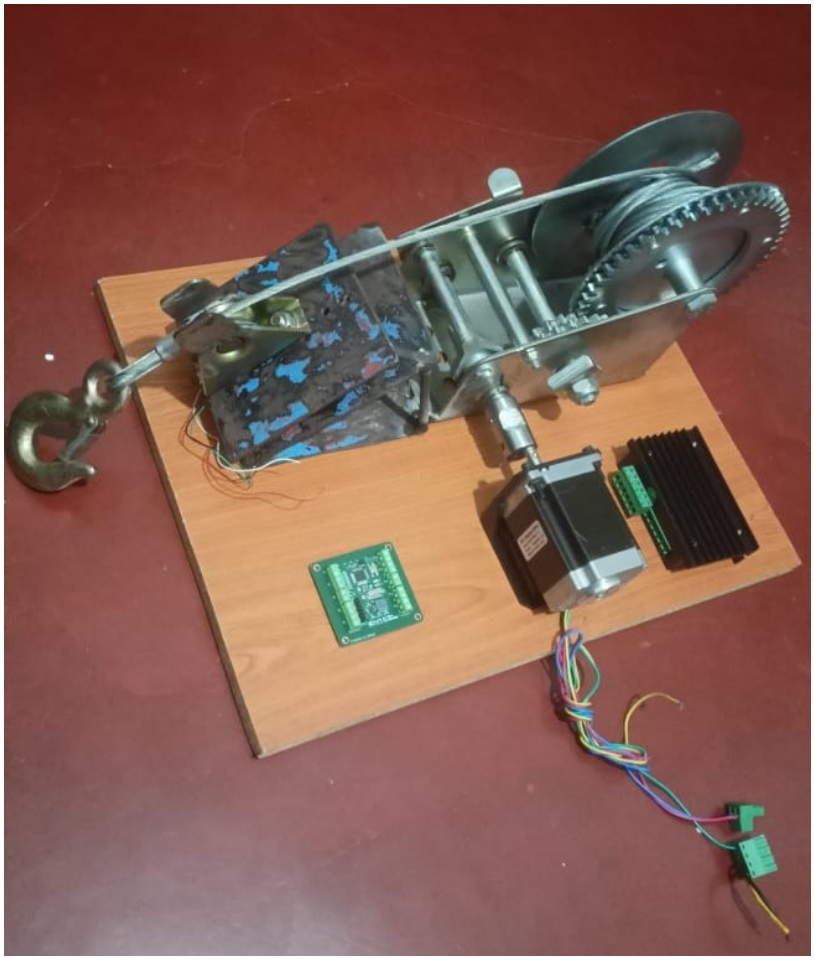
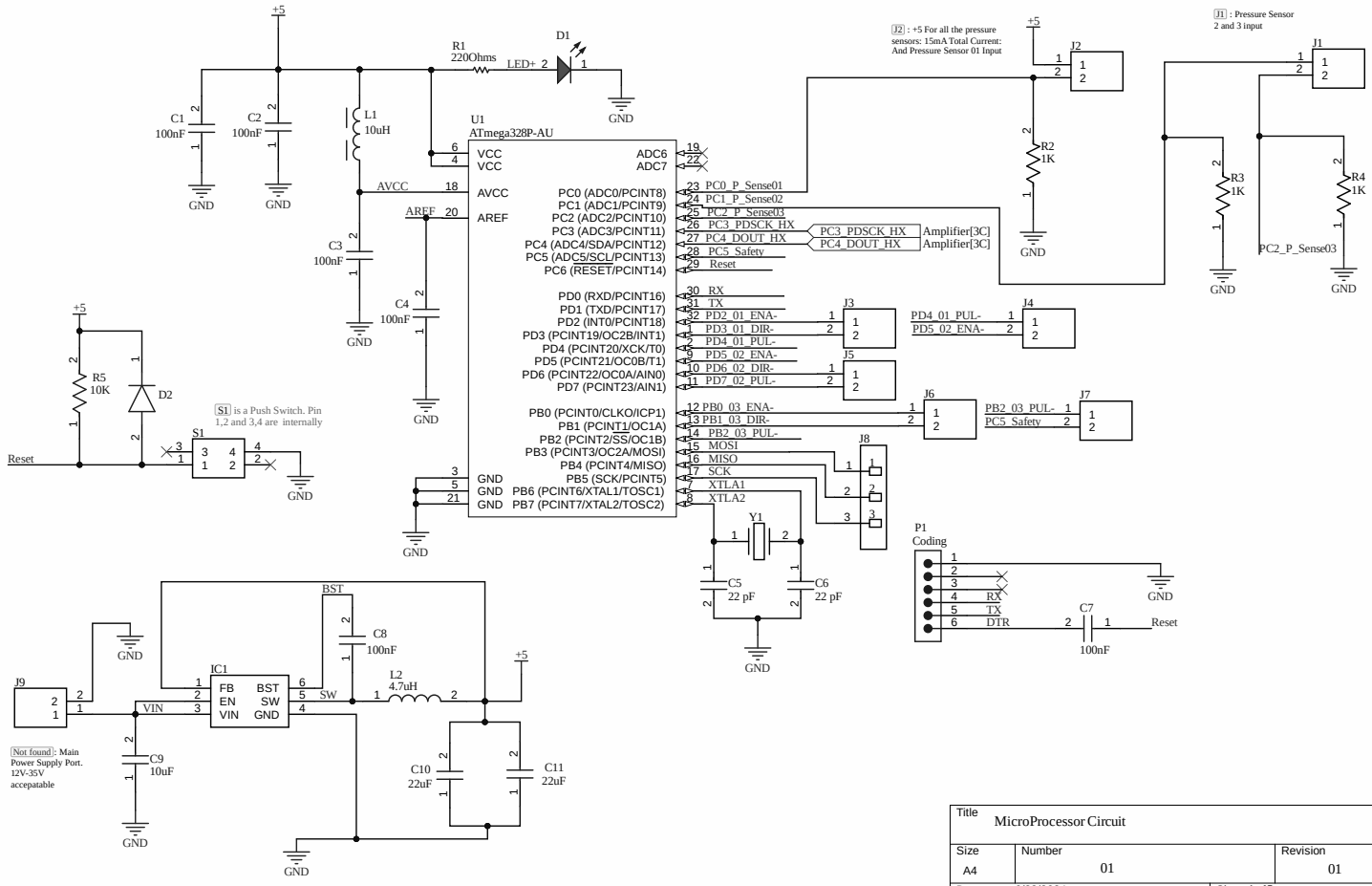
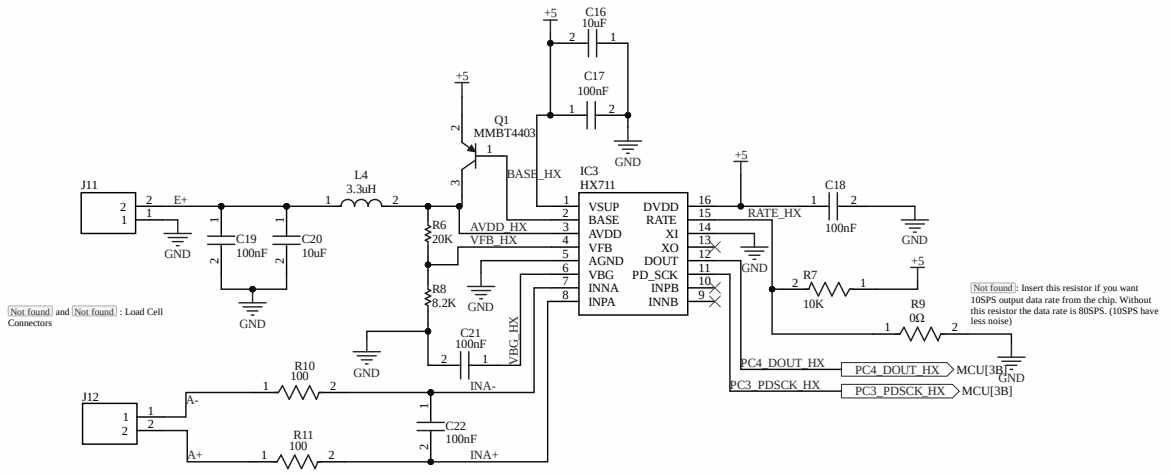


Figure B.4: Mechanical design

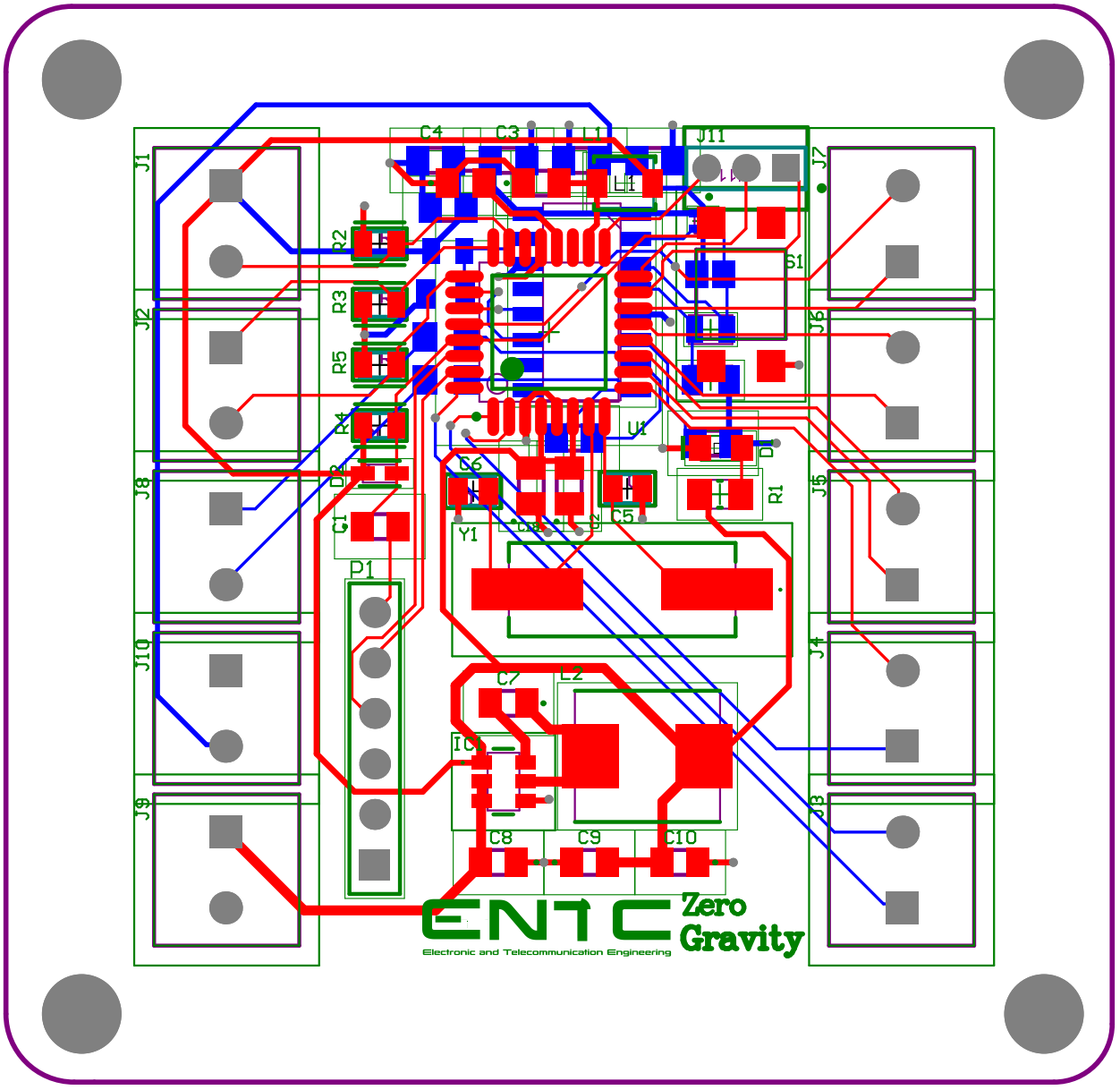
Figure B.5: Mechanical design

C Schematic


+5
R1
220Ohms
LED+
D1
GND
C1
100nF
C2
100nF
L1
10uH
U1
ATmega328P-AU
VCC
VCC
AVCC
AREF
GND
C3
100nF
C4
100nF
ADC6
ADC7
PC0 (ADC0/PCINT8)
PC1 (ADC1/PCINT9)
PC2 (ADC2/PCINT10)
PC3 (ADC3/PCINT11)
PC4 (ADC4/SDA/PCINT12)
PC5 (ADC5/SCL/PCINT13)
PC6 (RESET/PCINT14)
PC0_P_Sense01
PC1_P_Sense02
PC2_P_Sense03
PC3_PDSCK_HX
PC4_DOUT_HX
PC5_Safety
Reset
PC3_PDSCK_HX
PC4_DOUT_HX
Amplifier[3C]
Amplifier[3C]
J2 : +5 For all the pressure sensors: 15mA Total Current: And Pressure Sensor 01 Input
R2
1K
J1 : Pressure Sensor 2 and 3 input
J1
R3
1K
R4
1K
PC2_P_Sense03
PD0 (RXD/PCINT16)
PD1 (TXD/PCINT17)
PD2 (INT0/PCINT18)
PD3 (PCINT19/OC2B/INT1)
PD4 (PCINT20/XCK/T0)
PD5 (PCINT21/OC0B/T1)
PD6 (PCINT22/OC0A/AIN0)
PD7 (PCINT23/AIN1)
RX
TX
PD2_01_ENA-
PD3_01_DIR-
PD4_01_PUL-
PD5_02_ENA-
PD6_02_DIR-
PD7_02_PUL-
J3
J4
J5
PD4_01_PUL-
PD5_02_ENA-
PB0 (PCINT0/CLKO/ICP1)
PB1 (PCINT1/OC1A)
PB2 (PCINT2/SS/OC1B)
PB3 (PCINT3/OC2A/MOSI)
PB4 (PCINT4/MISO)
PB5 (SCK/PCINT5)
PB6 (PCINT6/XTAL1/TOSC1)
PB7 (PCINT7/XTAL2/TOSC2)
PB0_03_ENA-
PB1_03_DIR-
PB2_03_PUL-
MOSI
MISO
SCK
XTLA1
XTLA2
J6
J7
J8
PB2_03_PUL-
PC5_Safety
Y1
C5
22 pF
C6
22 pF
P1
Coding
RX
TX
DTR
C7
100nF
Reset
R5
10K
D2
S1 is a Push Switch. Pin 1,2 and 3,4 are internally
S1
Reset
BST
C8
100nF
IC1
FB
EN
VIN
BST
SW
GND
SW
L2
4.7uH
J9
VIN
C9
10uF
Not found : Main Power Supply Port. 12V-35V accepatable
C10
22uF
C11
22uF
Title
MicroProcessor Circuit
Size
A4
Number
01
Revision
01
Date:
6/23/2024
Sheet 1 of 2
File:
C:\Users\..\MCU.SchDoc
Drawn By:
Ruchira Abeywardhane



Title	Load Cell Amplifier Circuit		
Size	Number		Revision
A4	02		01
Date:	6/23/2024	Sheet 2 of 2	
File:	C:\Users\..\Amplifier.SchDoc	Drawn By: Ruchira Abeywardhane	


C4
C3
L1
J11
J1
J7
R2
S1
J2
J6
R3
R5
R4
U1
D2
D1
J8
J5
C6
R1
C1
C5
Y1
P1
J10
J4
C7
L2
IC1
J9
J3
C8
C9
C10
ENTC
Zero
Gravity
Electronic and Telecommunication Engineering

D Code for micro controller in previous report

```
#include <AccelStepper.h>

// Define the motor interface type
#define motorInterfaceType 1

// Pins for the stepper motor driver
#define stepPin 2
#define dirPin 3



// Create an instance of the stepper motor class
AccelStepper stepper(motorInterfaceType, stepPin, dirPin);

void setup() {
  // Initialize serial communication for debugging
  Serial.begin(9600);

  // Set the maximum speed and acceleration
  stepper.setMaxSpeed(1000);  // Maximum speed in steps per second
  stepper.setAcceleration(500);  // Acceleration in steps per second^2
}

void loop() {
  // Read the analog values from the pressure sensors
  int sensorValue1 = analogRead(pressureSensor1Pin);
  int sensorValue2 = analogRead(pressureSensor2Pin);

  // Calculate the average force value from both sensors
  int averageForce = (sensorValue1 + sensorValue2) / 2;

  // Output the average force value to the serial monitor for debugging
  Serial.print("Sensor 1: ");
  Serial.print(sensorValue1);
  Serial.print("  Sensor 2: ");
  Serial.print(sensorValue2);
  Serial.print("  Average Force: ");
  Serial.println(averageForce);

  // Map the average force value to speed and angle
  float speed = map(averageForce, 0, 1023, 0, 1000);  // Speed from 0 to 1000 steps/
    second
  int angle = map(averageForce, 0, 1023, 0, 200);  // Angle from 0 to 200 degrees

  // Convert angle to steps (assuming 200 steps per revolution)
  long steps = (angle / 360.0) * 200;

  // Set the speed of the stepper motor
  stepper.setSpeed(speed);

```

```
  // Move the stepper motor to the desired position
  stepper.moveTo(steps);

  // Run the motor
  stepper.runSpeedToPosition();

  // Delay for a short period to avoid reading the sensors too quickly
  delay(100);
}
```

Listing D.1: Code for Stepper Motor Control with Load Cells

```
#include <AccelStepper.h>

// Define the motor interface type
#define motorInterfaceType 1

// Pins for the stepper motor driver
#define stepPin 2
#define dirPin 3

// Define analog input pins for the pressure sensors
#define pressureSensor1Pin A0
#define pressureSensor2Pin A1

// Create an instance of the stepper motor class
AccelStepper stepper(motorInterfaceType, stepPin, dirPin);

void setup() {
  // Initialize serial communication for debugging
  Serial.begin(9600);

  // Set the maximum speed and acceleration
  stepper.setMaxSpeed(1000);  // Maximum speed in steps per second
  stepper.setAcceleration(500);  // Acceleration in steps per second^2
}

void loop() {
  // Read the analog values from the pressure sensors
  int sensorValue1 = analogRead(pressureSensor1Pin);
  int sensorValue2 = analogRead(pressureSensor2Pin);

  // Calculate the average force value from both sensors
  int averageForce = (sensorValue1 + sensorValue2) / 2;

  // Output the average force value to the serial monitor for debugging
  Serial.print("Sensor 1: ");
  Serial.print(sensorValue1);
  Serial.print("  Sensor 2: ");
  Serial.print(sensorValue2);
  Serial.print("  Average Force: ");
  Serial.println(averageForce);

  // Map the average force value to speed and angle
  float speed = map(averageForce, 0, 1023, 0, 1000);  // Speed from 0 to 1000 steps/
    second
  int angle = map(averageForce, 0, 1023, 0, 200);  // Angle from 0 to 200 degrees

  // Convert angle to steps (assuming 200 steps per revolution)
  long steps = (angle / 360.0) * 200;

  // Set the speed of the stepper motor
  stepper.setSpeed(speed);

  // Move the stepper motor to the desired position
  stepper.moveTo(steps);

  // Run the motor
```

```
  stepper.runSpeedToPosition();

  // Delay for a short period to avoid reading the sensors too quickly
  delay(100);
}
```

Listing D.2: Arduino Code for Stepper Motor Control with Pressure Sensors

```
#include <AccelStepper.h>
#include <Encoder.h>

// Define the motor interface type
#define motorInterfaceType 1

// Pins for the stepper motor driver
#define stepPin 2
#define dirPin 3

// Define encoder pins
#define encoderPinA 4
#define encoderPinB 5

// Create an instance of the stepper motor class
AccelStepper stepper(motorInterfaceType, stepPin, dirPin);

// Create an instance of the encoder class
Encoder encoder(encoderPinA, encoderPinB);

// Define target position and speed
long targetPosition = 0;
float targetSpeed = 500;  // Initial target speed

void setup() {
  // Initialize serial communication for debugging
  Serial.begin(9600);

  // Set the maximum speed and acceleration for the stepper
  stepper.setMaxSpeed(1000);  // Maximum speed in steps per second
  stepper.setAcceleration(500);  // Acceleration in steps per second^2

  // Move the stepper motor to the home position
  stepper.moveTo(0);
  stepper.runToPosition();
}

void loop() {
  // Read the encoder value
  long encoderValue = encoder.read();

  // Calculate the current speed based on encoder value
  static long lastEncoderValue = 0;
  static unsigned long lastTime = 0;
  unsigned long currentTime = millis();
  long encoderDelta = encoderValue - lastEncoderValue;
  unsigned long timeDelta = currentTime - lastTime;

  float currentSpeed = (encoderDelta / (float)timeDelta) * 1000;  // Steps per second

  // Output the current speed and position to the serial monitor
  Serial.print("Encoder Value: ");
  Serial.print(encoderValue);
  Serial.print("  Current Speed: ");
  Serial.println(currentSpeed);

  // Adjust the stepper speed if necessary
  if (abs(currentSpeed - targetSpeed) > 10) {  // Allowable error margin
    stepper.setSpeed(targetSpeed);
  }
```

```
61
62    // Move the stepper motor to the target position
63    stepper.moveTo(targetPosition);
64    stepper.runSpeedToPosition();
65
66    // Update the last encoder value and time
67    lastEncoderValue = encoderValue;
68    lastTime = currentTime;
69
70    // Delay for a short period to avoid overwhelming the serial output
71    delay(100);
72  }
```

Listing D.3: Code for Stepper Motor Control with Encoder Feedback

E Datasheets

1. Atmega328P Microproceesor
   https://www.mouser.com/datasheet/2/268/Atmel_7810_Automotive_Microcontrollers_ATmega328P_-3445629.pdf
2. Switching Voltage Regulators
   https://www.mouser.com/datasheet/2/115/DIOD_S_A0007089856_1-2542907.pdf
3. HX711 24-Bit Analog-to-Digital Converter
   https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf
4. Crystal Oscillator 16.000 MHz 18 pF
   https://www.mouser.com/datasheet/2/122/csm_7x-1841291.pdf
5. PNP Signal Transistor
   https://www.mouser.com/datasheet/2/115/DIOD_S_A0013022777_1-2513168.pdf
6. Small Signal Switching Diodes
   https://www.mouser.com/datasheet/2/80/CDSU4148_HF_RevC216291-2505199.pdf
7. Stepper Motor Driver
   https://bulkman3d.com/wp-content/uploads/2019/06/TB6600-Stepper-Motor-Driver-BM3D-v1.1.pdf
8. Stepper Motor
   https://uamper.com/products/datasheet/23HS8430.pdf