Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN2160 - Electronic Design Realization

3D Scanner - Design Document v3.0

Pathirana R.P.S. 210451U
Uduwaka S.S. 210663V

Contents

1. Introduction

The objective of our project is to develop a cost-effective and reliable 3D scanner that can scan an object, capture its dimensions, and generate a 3D map of the scanned object. Our 3D scanner comprises two main parts: a linear Z-axis motion slider and a turntable. During the scanning process, the object is placed on the turntable, which rotates to capture a full 360-degree scan. Once a complete rotation is achieved, the Z-axis slider, which holds the Time-of-Flight (ToF) sensor, moves upward incrementally, and the turntable begins another rotation. This process continues until the entire object has been scanned at various heights.

The ToF sensor mounted on the Z-axis slider measures the distance to the object's surface, and these distance values are transmitted to a computer in real time via serial communication. The collected data is stored in a .txt file, and upon completion of the scanning process, the data is used to generate a 3D point cloud map of the object.

This document provides a comprehensive overview of our design process, including the selection of components, the electronic design, the mechanical design, the software and coding aspects, and a log of our daily activities.

2. System Requirements

The system must have Python 3.x installed, along with the necessary Python packages. These packages include 'pyserial' for handling serial communication between the computer and the microcontroller, 'numpy' for numerical operations and data manipulation, 'matplotlib' for plotting and visualizing the 3D point cloud data, 'scipy' for signal processing and convolution operations, and 'mpl toolkits.mplot3d' for 3D plotting capabilities within 'Matplotlib'. To install these packages, users can use the following commands in their terminal or command prompt:

For installling those python packages user can use these cmd commands.

- pip install pyserial
- pip install numpy
- pip install matplotlib
- pip install scipy

A suitable Integrated Development Environment (IDE) or text editor is recommended for editing and running Python scripts, such as PyCharm, VS Code, or Jupyter Notebook. Additionally, necessary drivers should be installed for the USB-to-serial adapter used for communication with the microcontroller.

Users need to ensure the microcontroller is connected to the correct serial port (e.g., COM5 on Windows). The specific serial port may vary depending on the system and configuration. Moreover, users must have the necessary permissions to read from and write to files on their system, particularly the text file used for logging distance readings ('tof readings.txt').

By meeting these requirements, users will be able to operate the 3D scanner, capture real-time data, and generate accurate 3D point clouds for visualization and further analysis. Ensuring that all hardware components are correctly assembled and all necessary software packages are installed will facilitate smooth operation and optimal performance of the 3D scanning system.

3. Electronic Design

The electronic design of our 3D scanner involved careful selection of components and meticulous PCB design to ensure reliability, compactness, and efficient operation. This section details our process for selecting the appropriate components and the subsequent design and layout of the PCB.

3.1 Component Selection

3.1.1 NEMA 17 Stepper motor

NEMA 17 stepper motors are a crucial component in our 3D scanner, providing the necessary precision and torque for both the rotational movement of the turntable and the linear movement of the Z-axis. These motors were selected due to their high performance and compatibility with our 12V power supply, ensuring maximum efficiency and reliability.

Figure 1: NEMA 17 Stepper

Key Features:

- Shaft Size: 5mm
- Torque: 76 oz*in
- Step Angle: 1.8 degrees
- Peak Current: 1.68A per phase
- Mounting Holes: M3
- Wiring: 4 Wire Bi-Polar
- Operating Voltage: 12-24VDC

The use of NEMA 17 stepper motors in our 3D scanner project brings several significant advantages, making them an ideal choice for both the rotational movement of the turntable and the linear movement of the Z-axis slider. These motors provide a substantial torque of 76 oz*in, crucial for overcoming resistance and maintaining smooth, consistent movements during the scanning process. This high torque ensures that the motor can handle the load of the turntable and the Z-axis mechanism without compromising performance.

With a step angle of 1.8 degrees, NEMA 17 motors offer precise control over both rotational and linear positions. This precision is essential for 3D scanning applications, where accurate positioning directly impacts the quality and fidelity of the scanned data. These motors are designed to operate efficiently within a voltage range of 12-24VDC, making them perfectly suited for our system's 12V power supply. This compatibility ensures that the motors can function at their peak performance without requiring additional voltage regulation.

Featuring M3 mounting holes, NEMA 17 motors can be easily and securely attached to our mechanical components. Their 4-wire bi-polar configuration simplifies the wiring process and facilitates straightforward connections to the stepper motor drivers, streamlining the integration into our overall design. Known

for their durability, NEMA 17 motors offer consistent and dependable performance. Their robust construction reduces the likelihood of mechanical failures during operation, ensuring that our 3D scanner remains reliable and effective over prolonged use. In conclusion, the NEMA 17 stepper motors provide high torque, precision, power supply compatibility, ease of integration, and reliability. These advantages make them an excellent choice for our 3D scanner, contributing to the accurate and efficient movement required for successful 3D scanning operations.

3.1.2 Stepper Motor Driver - TB6600

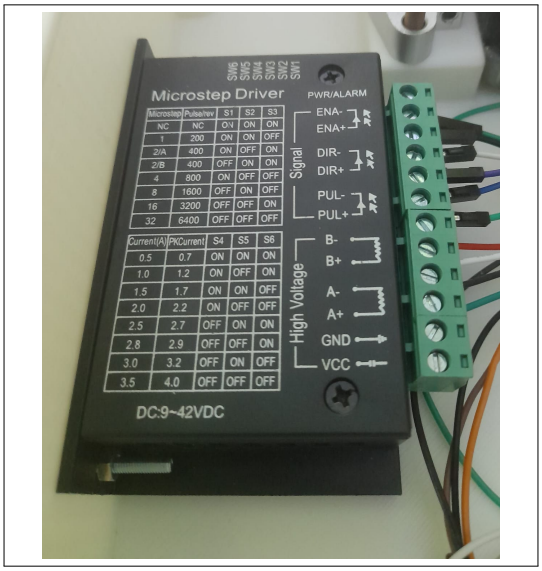


Figure 2: TB6600 Stepper Motor Driver

The TB6600 stepper motor driver is an essential component of our 3D scanner project, chosen for its robust performance and versatile capabilities. This driver supports a wide input voltage range of 9 42VDC, accommodating various power supply configurations. It can deliver a peak current of up to 4A, making it suitable for driving most stepper motors, including the NEMA 17 motors used in our design.

One of the standout features of the TB6600 driver is its support for speed and direction control, providing precise manipulation of motor movements. The driver allows for easy adjustment of micro-stepping and output current via 6 DIP switches, offering seven different micro-step settings (1, 2/A, 2/B, 4, 8, 16, 32) and eight current control options (0.5A, 1A, 1.5A, 2A, 2.5A, 2.8A, 3.0A, 3.5A). This flexibility enables fine-tuning of motor performance to match the specific requirements of our scanning application, enhancing accuracy and efficiency.

Additionally, the TB6600 incorporates high-speed optocoupler isolation for all signal terminals, significantly improving its resistance to high-frequency interference. This feature ensures stable and reliable operation even in environments with potential electrical noise, crucial for maintaining consistent performance during the scanning process.

Designed to drive two-phase and four-phase hybrid stepper motors, the TB6600 is compatible with a wide range of motor types, including 57 and 42-type stepper motors. Its professional-grade construction and advanced features make it a reliable choice for our 3D scanner, ensuring precise control and robust operation throughout the scanning process.

3.1.3 VL53L0X - ToF Sensor

The VL53L0X Time-of-Flight (ToF) sensor is chosen for our 3D scanner project due to its high accuracy and reliability in distance measurement using a narrow light beam. Unlike ultrasonic sensors that use sound

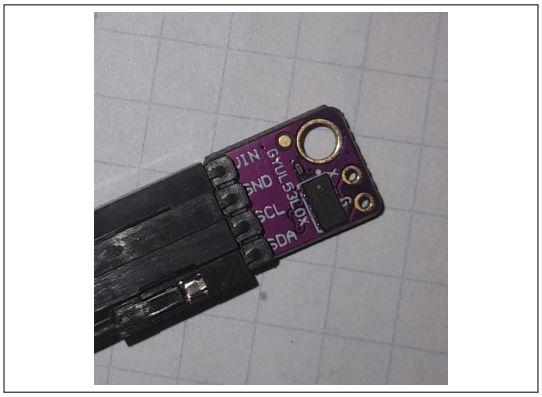


Figure 3: VL53L0X ToF Sensor

waves or IR sensors that measure reflected light intensity, the VL53L0X calculates the time it takes for a laser pulse to bounce off an object and return to the sensor. This method allows for precise distance measurements with minimal interference from ambient light or surface reflectivity variations.
Key technical specifications include:

- High Accuracy: Capable of measuring distances accurately within millimeters.
- Fast Measurement: Operates at speeds suitable for real-time applications.
- Narrow Field of View: Ensures precise measurement of the object directly in front of it without interference from nearby objects.
- No Linearity Issues: Unlike some IR sensors, it maintains consistent accuracy across different distances without ”double imaging” or other artifacts.

We use that ToF sensor because, its ability to provide reliable and precise distance measurements in real-time aligns perfectly with the requirements of your 3D scanner, ensuring high-quality 3D point cloud maps of scanned objects.

3.2 Component Selection for PCB

In designing our PCB, we carefully selected components to ensure optimal performance and compatibility. The key components used in our PCB design include the ATmega328P microcontroller and the CH340C USB-to-serial converter IC, along with supporting components such as oscillators and coupling capacitors.

We selected the ATmega328P microcontroller as the main controller unit for our 3D scanner. This microcontroller is well-suited for our application due to its high performance, low power consumption, and ease of programming. The ATmega328P operates at a frequency of 16MHz, which provides the necessary processing power to handle the real-time data acquisition and control tasks required by our system. Additionally, its wide range of available I/O pins allows for flexible interfacing with sensors, motors, and other peripherals.

For establishing serial communication via the USB com port, we chose the CH340C IC. This IC is specifically designed for USB-to-serial communication and includes an internal 12MHz oscillator, simplifying the design by eliminating the need for an external oscillator. The CH340C provides reliable and efficient data transfer between the microcontroller and the computer, ensuring smooth communication for real-time data logging and control.

To drive the ATmega328P microcontroller, we used a 16MHz oscillator. This frequency is ideal for achieving the required clock speed for our microcontroller, enabling it to perform the necessary computations and control tasks effectively. The choice of a 16MHz oscillator ensures that the microcontroller operates at its optimal performance level.

In selecting coupling capacitors, we considered the internal resistance of the oscillator and the electronic requirements of the overall circuit. Proper coupling capacitors are crucial for stabilizing the oscillator's frequency and ensuring reliable operation of the microcontroller. These capacitors help to filter out any noise and provide a stable voltage supply to the oscillator, which in turn ensures the accurate timing of the microcontroller.

Throughout the component selection process, we carefully considered the electronic requirements of our PCB design. This involved ensuring that all components were compatible in terms of voltage levels, current ratings, and signal integrity. We also took into account the power supply requirements and made sure that our power converter could adequately supply the necessary voltage and current to all components on the PCB.

By meticulously selecting the ATmega328P microcontroller, CH340C USB-to-serial converter, 16MHz oscillator, and appropriate coupling capacitors, we ensured that our PCB would meet the performance and reliability requirements of our 3D scanner project. Each component was chosen based on its specific characteristics and suitability for our application's needs, resulting in a robust and efficient design.

3.2.1 Bill of Materials

Comment	Description	Designator	Footprint	LibRef	Quantity
0.1uF	Multilayer Ceramic Capacitor 0.1uF 10V 10% SMD 0805 T/R	C1	FP-0805-L_2_0_0_1-W_1_25-IPC_C	CMP-14477-004636-2	1
10uF	Cap Ceramic 10uF 25V X7S 10% Pad SMD 0805 +125°C Automotive T/R	C2, C3	FP-CGA4J-IPC_C	CMP-08246-001526-1	2
100nF	Chip Capacitor, 100nF, +/-10%, 16V, -55 to 125 degC, 0402 (1005 Metric)	C4, C11-C13	CAPC1050X55X23ML05T13	CMP-2000-05439-1	4
10uF (Electrolytic)	Aluminum Electrolytic Capacitor, 10uF, +/-20%, 50V, -40 to 85 degC	C7, C8	CAPPR55-200-1100X500X1200	CMP-001-00025-7	2
Green LED	Chip LED, Green, 2.2V, -40 to 85 degC, 2-Pin SMD, Pb-Free, Tape and Reel	DS1, DS2	AVAG-HSMX-C170_V	CMP-2000-05062-1	2
ATmega328P	8-bit AVR Microcontroller, 32KB Flash, 1KB EEPROM, 2KB SRAM, 32-pin TQFP	IC1	32A_M	CMP-0095-00269-2	1
Header 4-Pin	Header, 4-Pin	J1, J2	HDR1X4	Header 4	2
Header 6-Pin	Header, 6-Pin	J3, J4	HDR1X6	Header 6	2
Terminal Block	PC Terminal Block, Pitch 6.35 mm, 1 x 2 Position, Height 21.5 mm	J5	PHOE-1714955_V	CMP-2000-05797-1	1
USB Connector	USB 2.0 Interface, Type A	J6	SAMTEC_USB-A-S-X-X-TH	USB-A-S-X-X-TH	1
10k Resistor	Chip Resistor, 10k Ohm, +/-1%, 125 mW, -55 to 155 degC, 0805 (2012 Metric)	R1	RESC2013X60X40LL15T20	CMP-2001-04472-1	1
1k Resistor	Chip Resistor, 1k Ohm, +/-1%, 125 mW, -55 to 155 degC, 0805 (2012 Metric)	R2, R3, R4, R5	RESC2013X60X35ML10T20	CMP-2100-03671-1	4
Switch	Push Button Switch	SW1	WS-TASV_L6W6H2.5_4304x30258x6	CMP-1455-00007-2	1
5V Regulator	Positive Voltage Regulator, 5V, 1A, 0 to 125 degC, 3-Pin TO-220, RoHS, Tube	T1	FAIR-TO-220-3	CMP-2000-04937-1	1
USB to Serial	USB to Serial Chip	U2	SOIC127P600X180-16N	CH340C	1
Crystal	Quartz Crystal	Y1	WE-XTAL_CFPX-180	CMP-03913-116876469-1	1

Figure 4: Bill of Materials

3.3 Schematic Design Overview

We have designed 3 main schematic designs for our PCB.

- Micro-controller
- Voltage Converter
- USB Serial Communication

3.3.1 Block Diagram

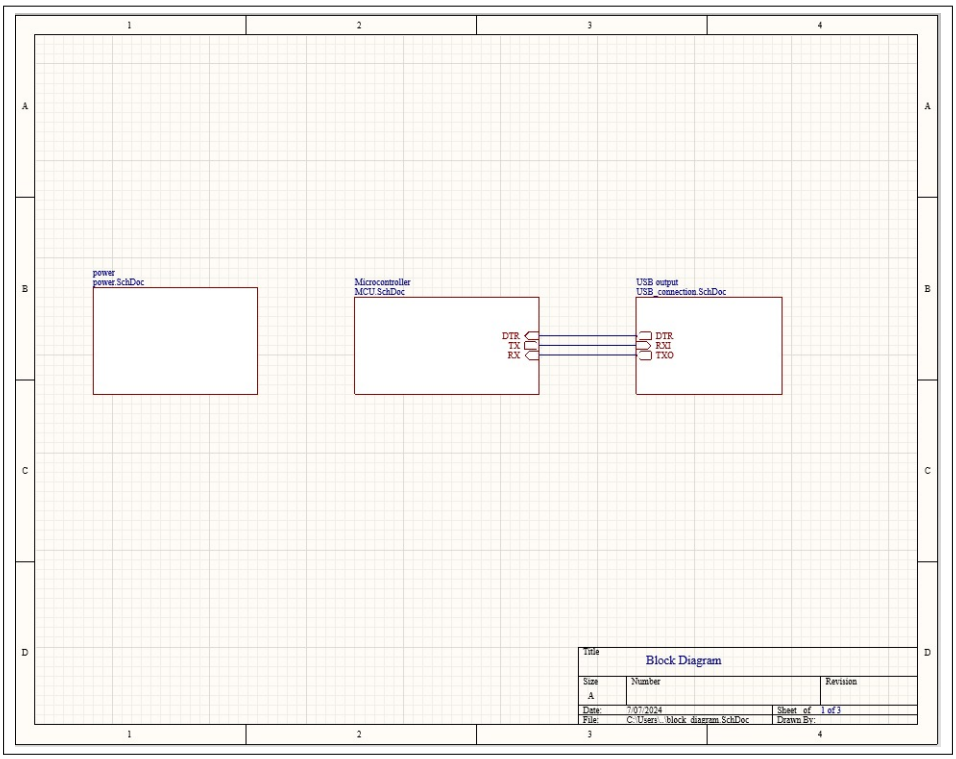


Figure 5: Functional Block Diagram

The voltage converter part is responsible for converting the input voltage to the 5V for different components on the board. It typically includes voltage regulator and capacitors to stabilize and filter the power supply. This section ensures that all parts of the PCB receive a stable and reliable power supply, preventing voltage fluctuations that could affect sensor accuracy or MCU performance.

The MCU part serves as the brain of the system, handling the control of the Z-axis motion slider, turntable, and the ToF sensor. It includes the Atmega328p microcontroller, 16MHz oscillator, and necessary passive components. The MCU is selected based on its processing power, available I/O pins, and compatibility with the ToF sensor and other peripherals. Proper layout and the inclusion of decoupling capacitors are crucial for reliable operation.

The USB serial communication part facilitates real-time data transfer between the PCB and the computer. This section includes a USB-to-serial converter IC CH340C, USB connectors, and necessary passive components. It ensures reliable and fast communication, allowing the real-time distance measurements from the ToF sensor to be transmitted to the computer for storage and processing. Proper ESD protection and signal integrity measures are implemented to maintain robust communication.

3.4 Microcontroller Unit Schematic design

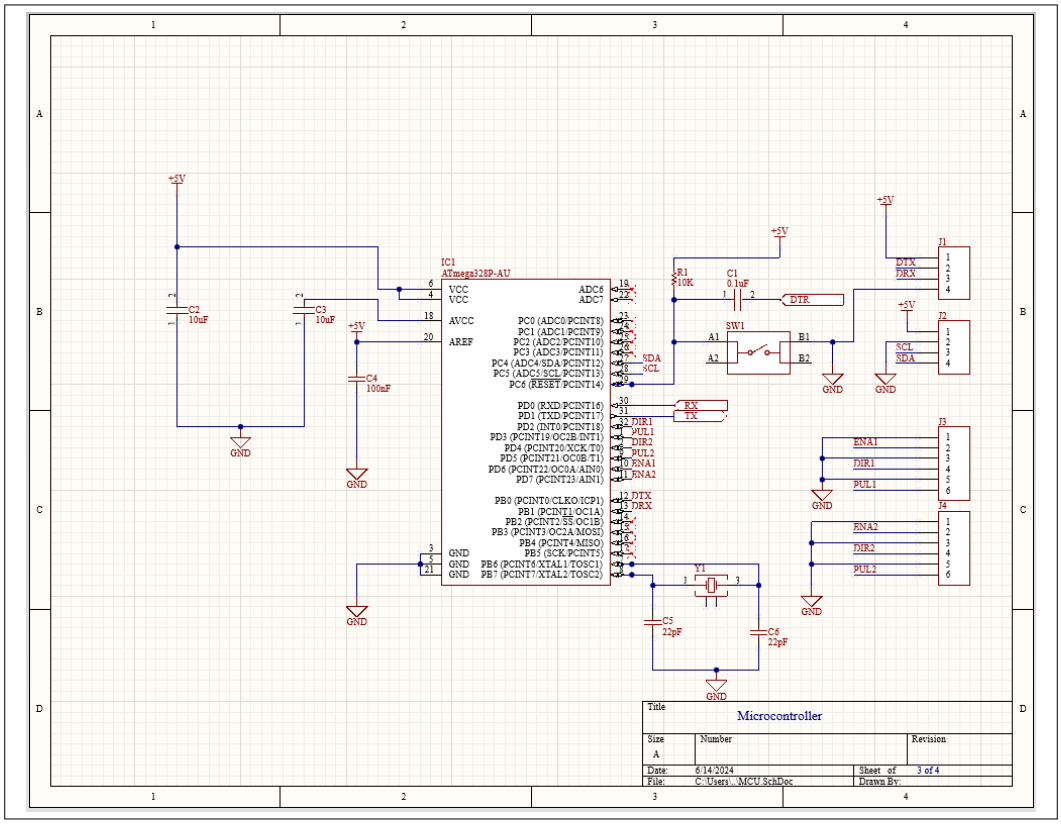


Figure 6: Micro Controller Unit

3.4.1 Microcontroller Selection:

For our 3D scanner project, we have selected the ATmega328P SMD microcontroller unit (MCU) due to its reliability, compact size, and capability to handle multiple inputs efficiently. The ATmega328P is a widely used microcontroller known for its robust performance and ease of programming.

3.4.2 Pin Configuration and Connectivity

- GPIO Pins
  - ToF Sensor Connection: The ToF (Time-of-Flight) sensor is connected to the GPIO pins of the ATmega328P. These pins are used for both power and data communication, allowing the MCU to receive distance measurements from the sensor accurately.
  - Stepper Motors Connection: Two NEMA 17 stepper motors are connected to the GPIO pins via stepper motor drivers (such as the TB6600). The GPIO pins send pulse and direction signals to control the rotation and vertical movement of the stepper motors.
- Serial Communication
  - TX and RX Pins: The ATmega328P uses its TX (transmit) and RX (receive) pins for serial communication with a computer. This communication is facilitated through a USB port, which allows for data transmission and programming of the microcontroller.

- USB Connection: The USB connection not only powers the microcontroller but also serves as the interface for programming and data transfer. This setup ensures a seamless flow of scanned data from the microcontroller to the PC for further processing and visualization.

3.4.3 Programming and Operation

- Microcontroller Programming
  - The ATmega328P is programmed using the Arduino IDE, which provides a user-friendly environment for writing, compiling, and uploading code to the microcontroller via the USB connection.
  - The firmware developed for the ATmega328P handles the coordination between the ToF sensor and the stepper motors, ensuring accurate data collection and motor control.
- Data Handling
  - The microcontroller collects distance data from the ToF sensor as the object rotates and elevates. This data is transmitted to the computer in real-time through the serial communication interface.
  - On the computer, the received data is processed to generate a point cloud, representing the 3D scanned object.

3.4.4 Advantages of Using ATmega328P

- Reliability
  - The ATmega328P is known for its stable performance and reliability in various applications, making it ideal for our 3D scanner project.
- Compact Size
  - The SMD version of the ATmega328P is compact, allowing for a smaller PCB layout and overall system size.
- Efficient Input Handling
  - The microcontroller efficiently manages multiple inputs and outputs, ensuring smooth operation of the sensor and motors simultaneously.
- Wide Community Support
  - The ATmega328P benefits from extensive community support and resources, facilitating troubleshooting and development.

By utilizing the ATmega328P microcontroller, we ensure that our 3D scanner is capable of accurate, reliable, and efficient scanning, meeting the project's functional and technical requirements. This microcontroller forms the core of our system, enabling seamless integration and operation of all components involved in the scanning process.

3.5 Establishing USB Connection Between MCU and PC Schematic Design

To facilitate the USB connection between the ATmega328P microcontroller and the PC, we employ the CH340C IC. This section outlines the components and methods used to establish this connection, ensuring reliable data transfer and programming capabilities.

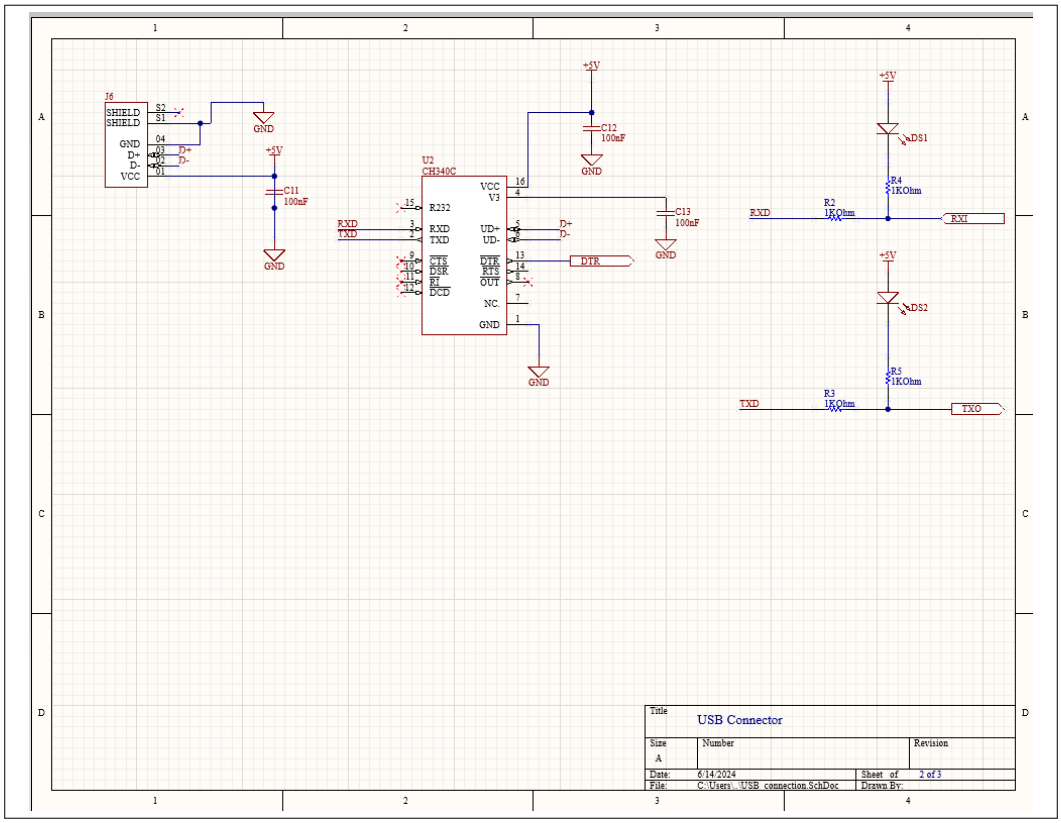


Figure 7: Micro Controller Unit

3.5.1 USB-to-Serial Conversion

- CH340C IC
  - The CH340C is a USB-to-serial converter IC that bridges the communication between the microcontroller's serial interface and the USB port on the PC.
  - Internal Oscillator: One of the key advantages of using the CH340C is its built-in internal oscillator, which eliminates the need for an external oscillator. This simplifies the circuit design and reduces component count.

3.5.2 Microcontroller Connections

- TX and RX Pins
  - The TX (transmit) and RX (receive) pins of the ATmega328P microcontroller are connected to the corresponding TXD and RXD pins of the CH340C IC. This setup allows serial data to be sent and received between the microcontroller and the PC.
  - Data Transmission: The TX pin of the microcontroller sends data to the RXD pin of the CH340C, while the RX pin of the microcontroller receives data from the TXD pin of the

CH340C. This bidirectional communication is essential for both data transfer and programming the microcontroller.

3.5.3 USB Connection

- USB Type A Female Header
  - A USB Type A female header is used on the PCB to facilitate the physical connection between the USB cable and the CH340C IC. This standard USB interface allows easy connectivity to the PC.
  - Power and Data Lines: The USB connection provides both power to the microcontroller and the data lines for serial communication. This dual functionality ensures that the system can be powered and communicate with the PC through a single USB cable. .

3.5.4 Circuit Design

- Simplified Design with CH340C
  - The use of the CH340C with its internal oscillator simplifies the PCB layout, as there is no need to accommodate an additional external crystal oscillator.
  - The CH340C IC integrates seamlessly with the ATmega328P microcontroller, providing a reliable and efficient USB-to-serial conversion.

3.5.5 Programming and Data Transfer

- Microcontroller Programming
  - The USB connection, established via the CH340C, allows for easy programming of the ATmega328P microcontroller. Using the Arduino IDE, code can be written, compiled, and uploaded directly to the microcontroller through this USB interface.
  - Real-Time Data Transfer: During operation, the CH340C facilitates real-time data transfer from the microcontroller to the PC. This enables the continuous transmission of scanned data, which is crucial for generating accurate 3D point clouds.

By incorporating the CH340C IC into our design, we ensure a robust and efficient USB connection between the ATmega328P microcontroller and the PC. This setup not only simplifies the circuit design but also enhances the reliability of data transmission and microcontroller programming. The use of a USB Type A female header provides a standard and user-friendly interface for connecting the 3D scanner to the PC.

3.6 Voltage Conversion Circuit Schematic Design

To ensure our 3D scanner operates efficiently and safely, we use a voltage conversion circuit to step down the input voltage from 12V to 5V. This section details the components and functionality of this circuit.

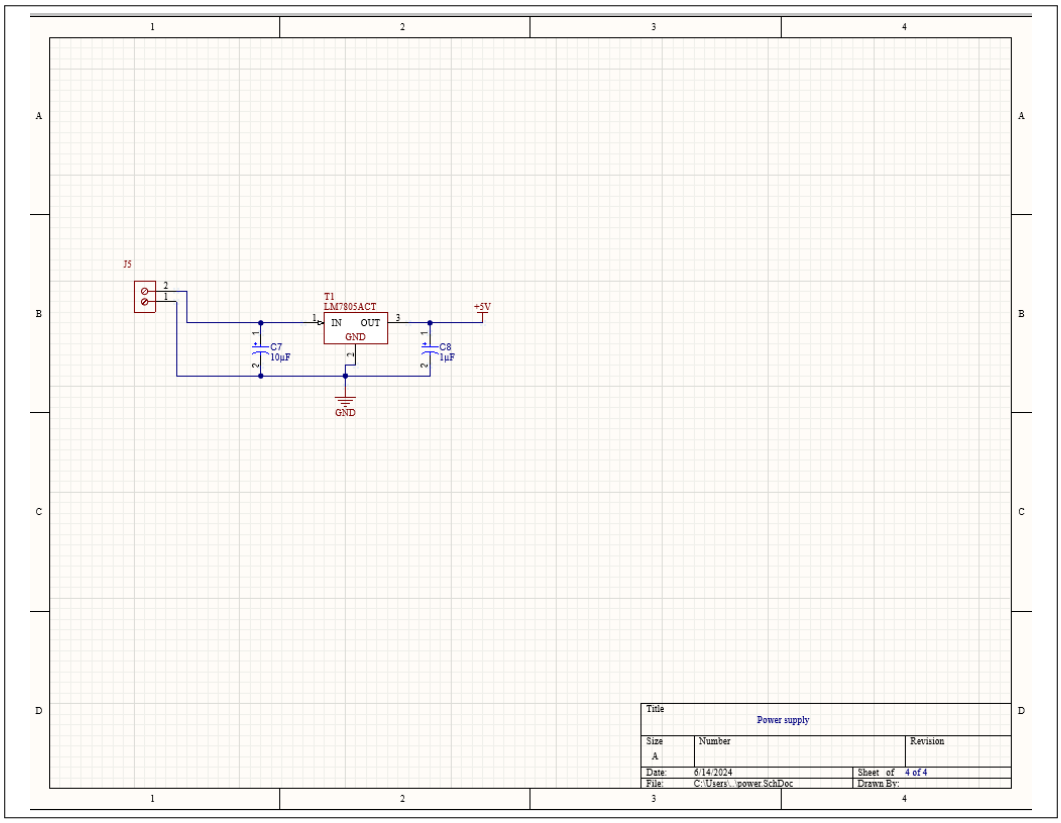


Figure 8: Voltage Conversion Circuit

3.6.1 Voltage Regulator

- LM7805 Voltage Regulator
  - The LM7805 is a linear voltage regulator that converts an input voltage of up to 35V to a stable 5V output.
  - Output Voltage: The LM7805 provides a regulated 5V DC output, which is essential for powering the ATmega328P microcontroller and other 5V components in our 3D scanner circuit.

3.6.2 Circuit Design

- Input Voltage (12V)
  - Our system is designed to be powered by a 12V DC input, which is common for many industrial and electronic applications.
  - Voltage Regulation: The 12V input is fed into the LM7805 voltage regulator, which then steps down the voltage to a stable 5V output.
- Component Connections
  - Input Capacitor: A capacitor is placed at the input of the LM7805 to filter out any high-frequency noise from the power supply, ensuring a smooth input voltage.

- Output Capacitor: Another capacitor is placed at the output of the LM7805 to stabilize the 5V output and filter out any voltage spikes, providing a clean and stable power supply to the microcontroller and other components.

To ensure our 3D scanner operates efficiently and safely, we utilize a voltage conversion circuit to step down the input voltage from 12V to 5V, employing the LM7805 voltage regulator. The LM7805 is highly beneficial due to its simplicity, requiring only two external capacitors for stable operation, making the circuit design straightforward and reliable. Additionally, it includes built-in thermal protection to prevent damage from overheating, thus ensuring the longevity of the device. The regulator also features short-circuit protection, safeguarding the circuit in case of faults or shorts in the connected components.

In our 3D scanner, the regulated 5V output from the LM7805 powers the ATmega328P microcontroller, the ToF sensor, and other 5V logic components, ensuring that all critical components receive a stable voltage supply for consistent and reliable operation. The voltage conversion circuit, including the LM7805 and the associated capacitors, is integrated into the PCB layout. This compact design minimizes space requirements and ensures efficient power distribution within the 3D scanner, enhancing overall system performance and reliability.

3.7 PCB Design

Our PCB for the 3D scanner machine is meticulously designed to integrate several critical functionalities, ensuring efficient operation and communication with the computer. The PCB is divided into three main sections: the power converter part, the MCU part, and the USB serial communication part.

3.7.1 PCB Layout

For routing the PCB design of our 3D scanner project, we adopt a cautious approach to ensure reliable current flow and minimize signal interference. Given that certain paths handle nearly 400mA of current, we opt for a trace width of 0.5mm in these critical areas, providing ample capacity to handle the current without overheating or voltage drops. This choice exceeds the typical doubling rule for trace widths, emphasizing an extra margin of safety. By carefully selecting trace widths based on current requirements and signal characteristics, we aim to optimize the performance and reliability of our PCB design for the 3D scanner.

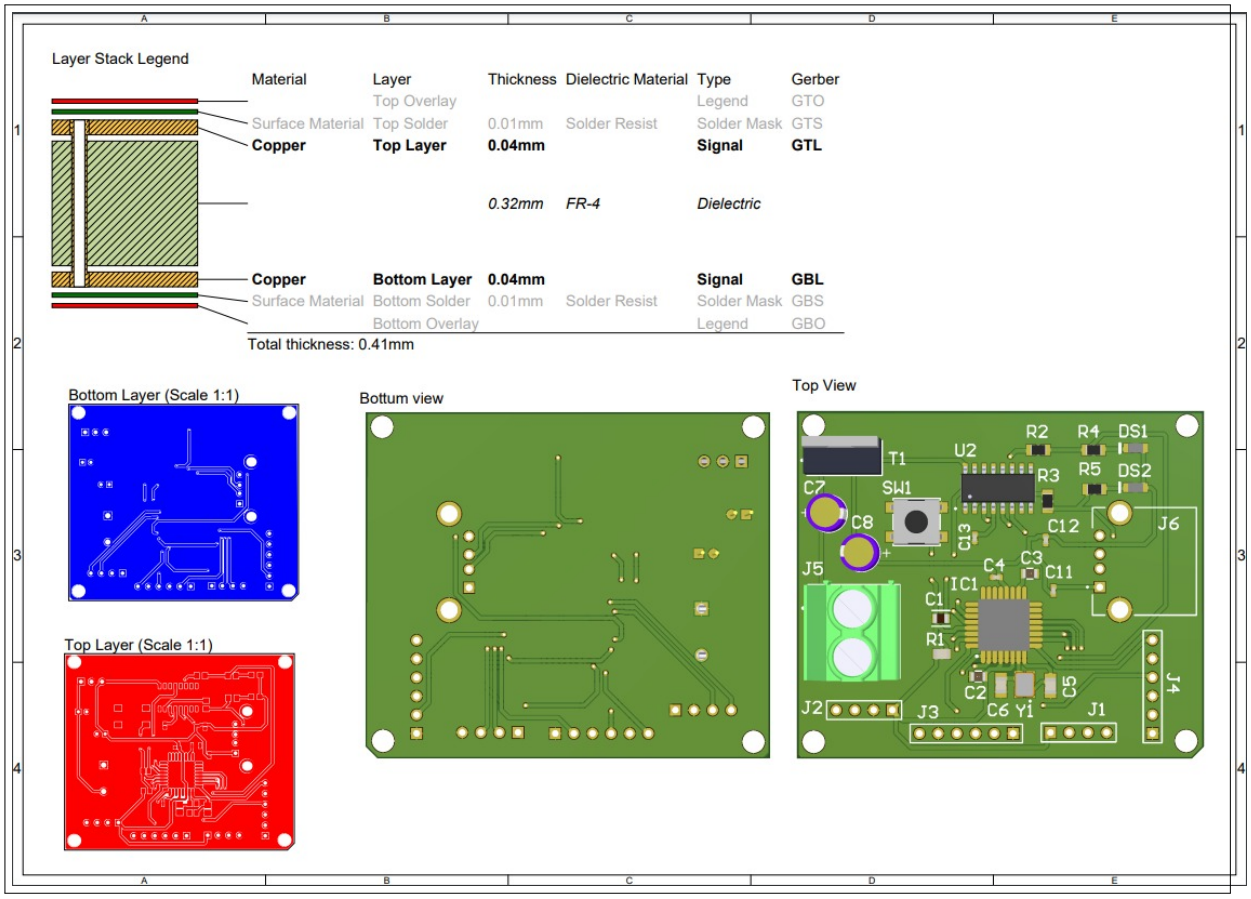


Figure 9: PCB Layer Stack

3.7.2 Top Layer

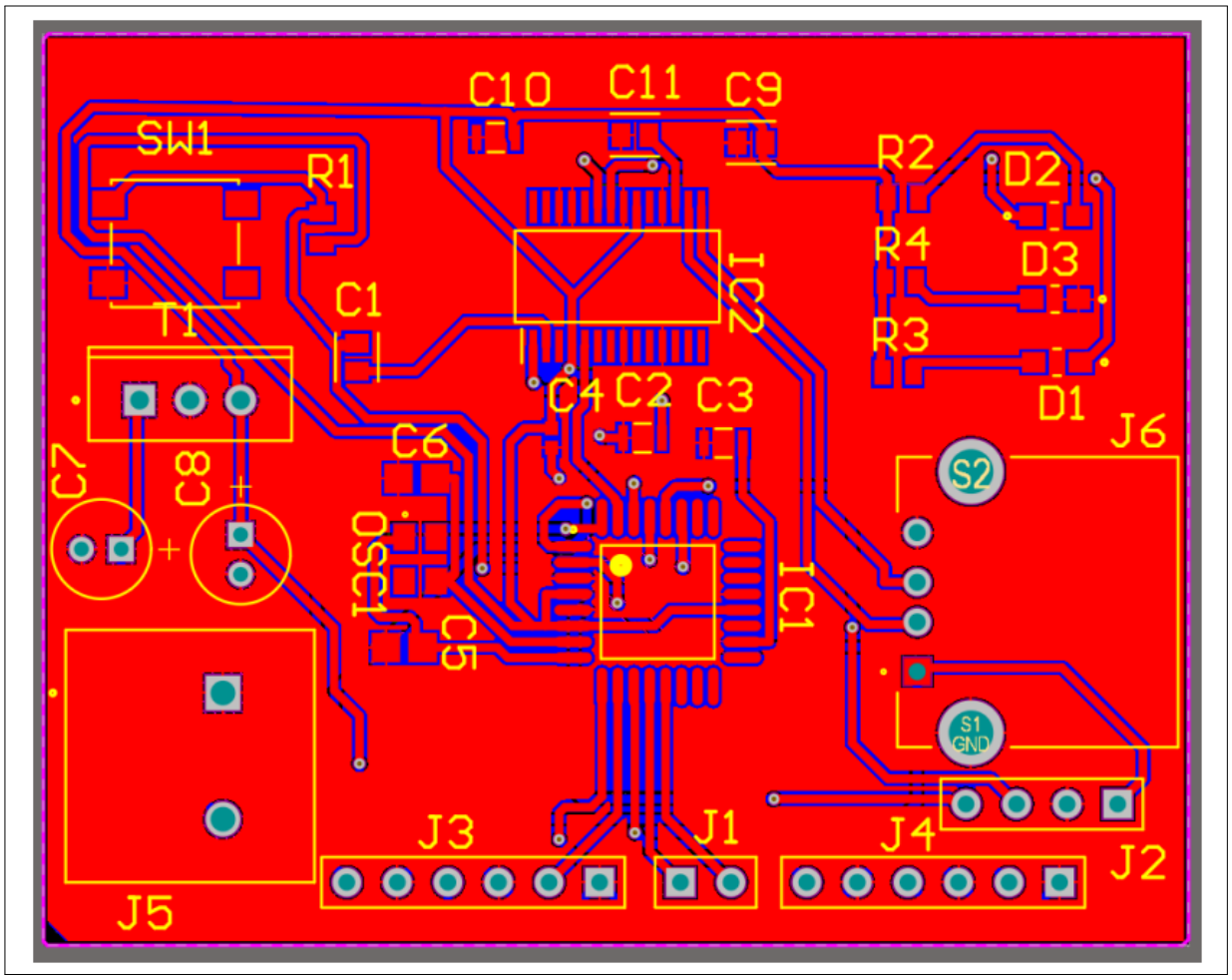


Figure 10: PCB top layer

3.7.3 Bottom Layer

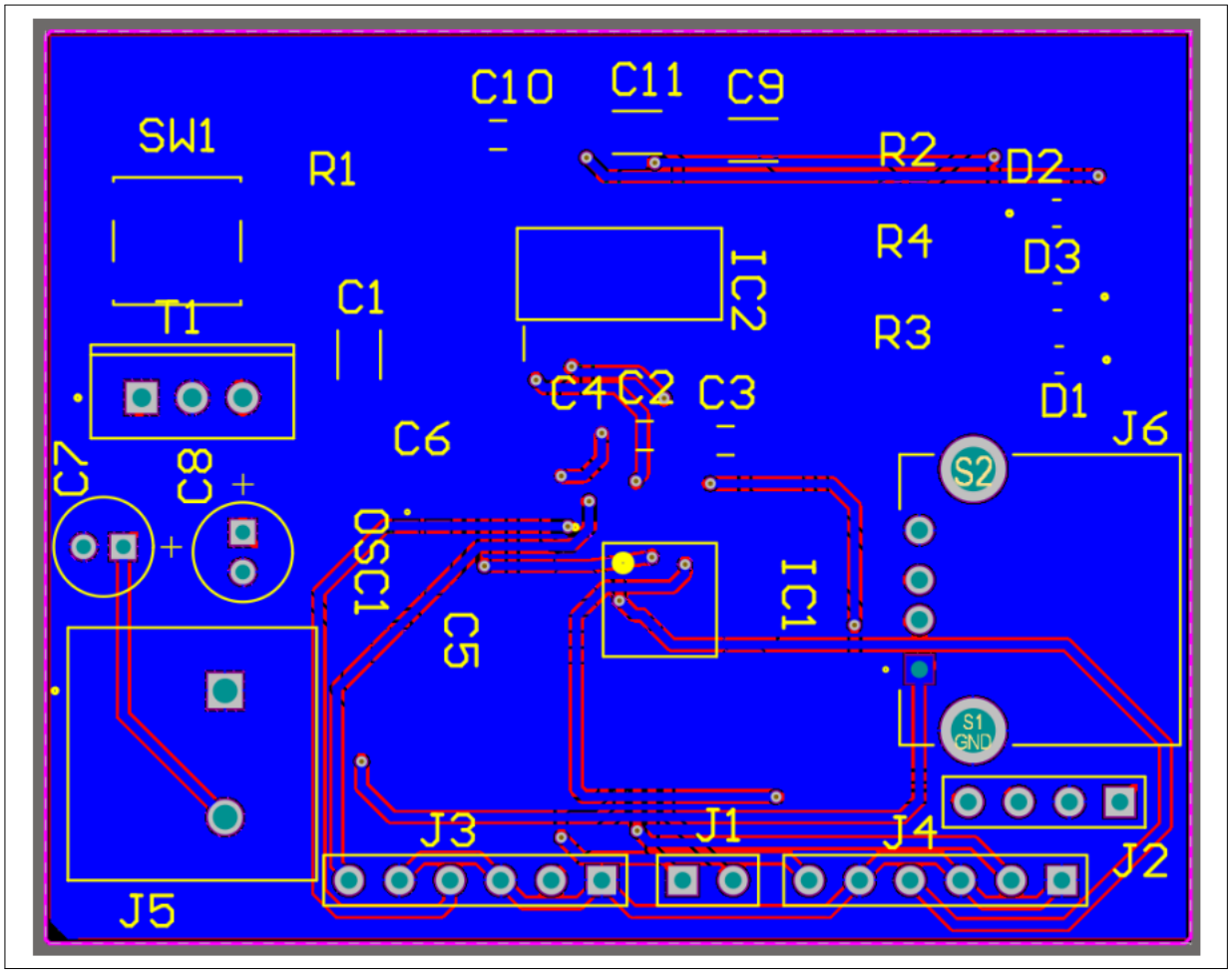


Figure 11: PCB bottom layer

3.7.4 3D View

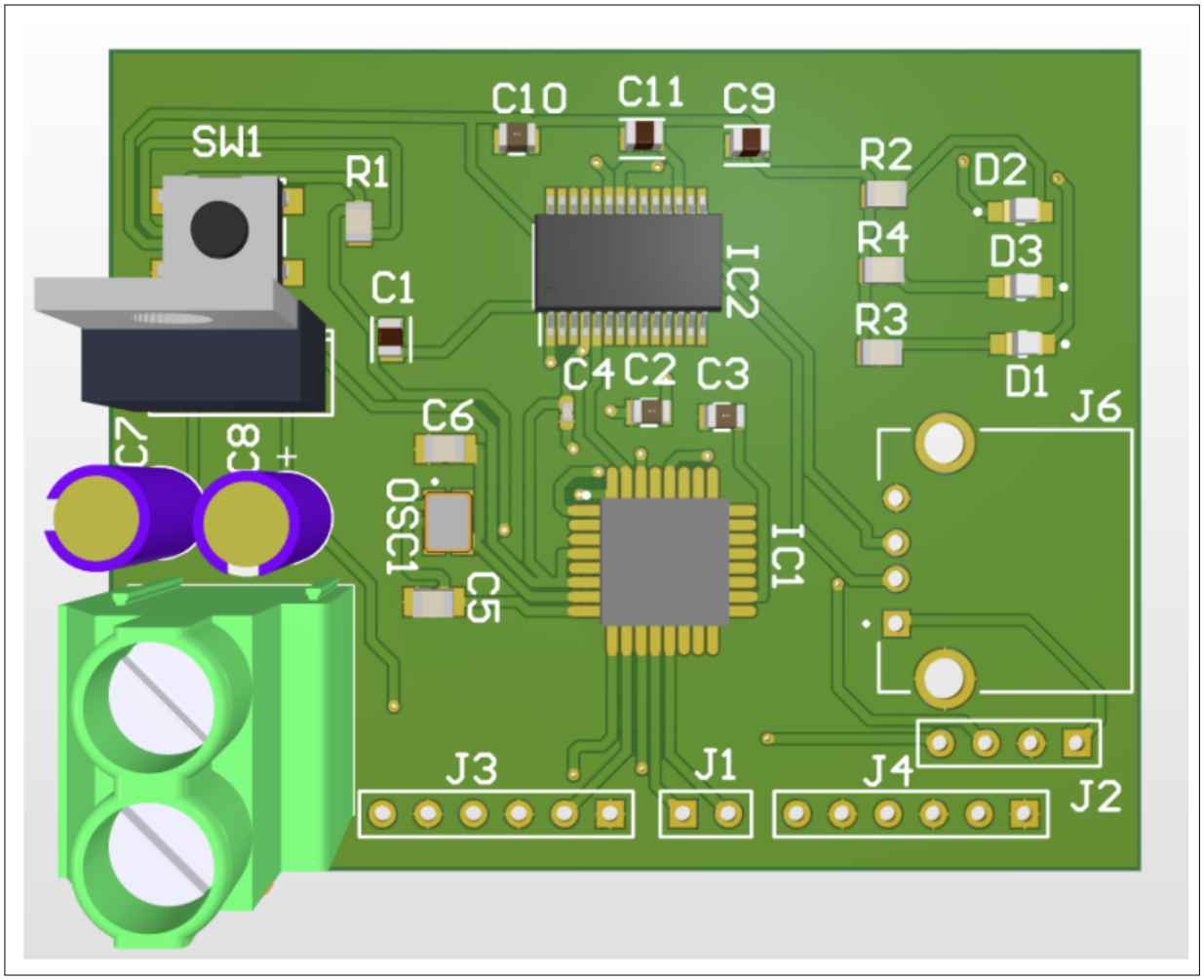


Figure 12: PCB with components

3.7.5 Drilling Details

Drilling specifications ensure precise hole placement for mounting components and connectors. Hole dimensions are defined according to component specifications and PCB manufacturing requirements.

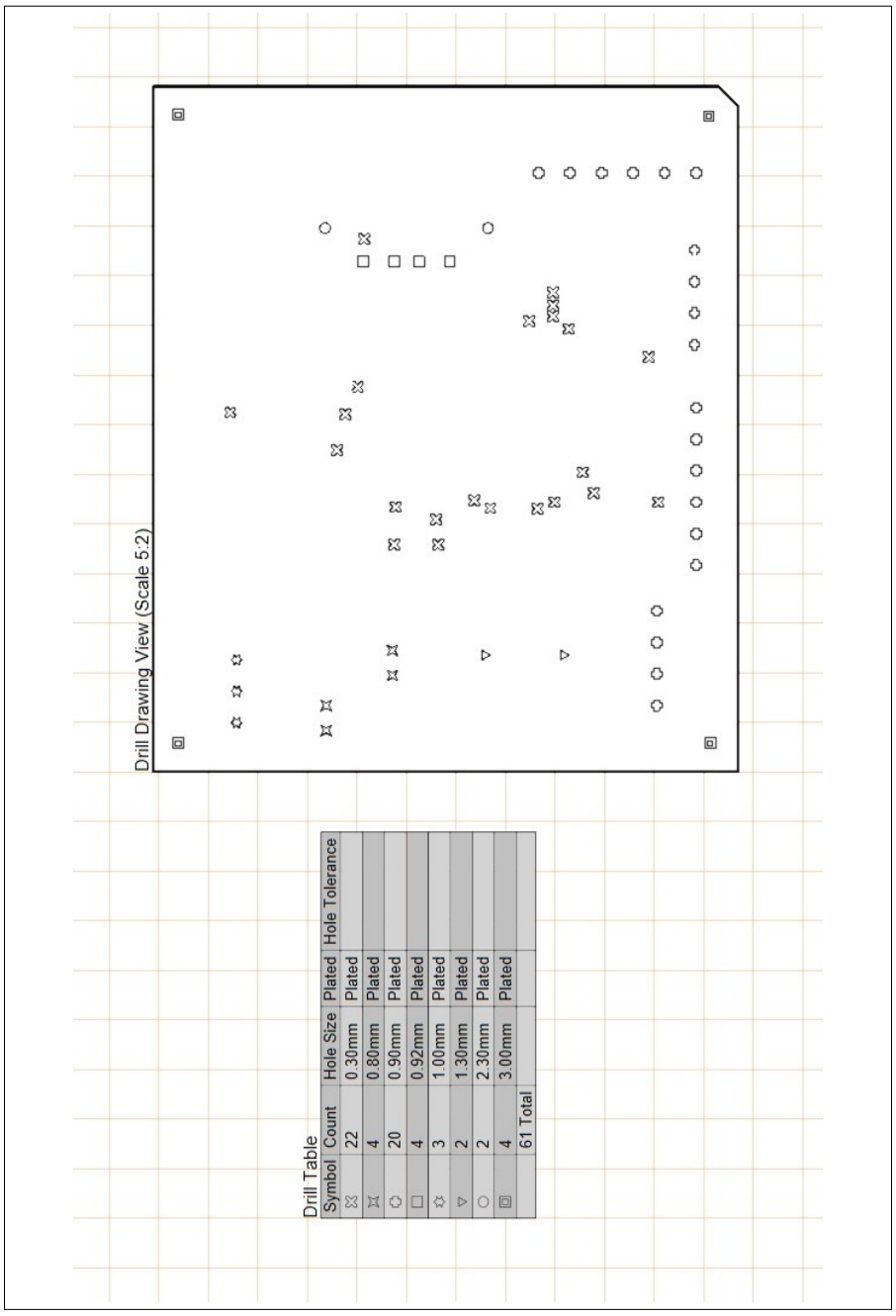


Figure 13: Drilling Details

3.7.6 Printed PCB

After finalizing the routing and drilling details, the design files for the 3D scanner PCB are generated and sent for manufacturing. The PCB is sent to JLCPCB and the manufacturer uses these files to produce the printed PCB according to the specified design parameters.

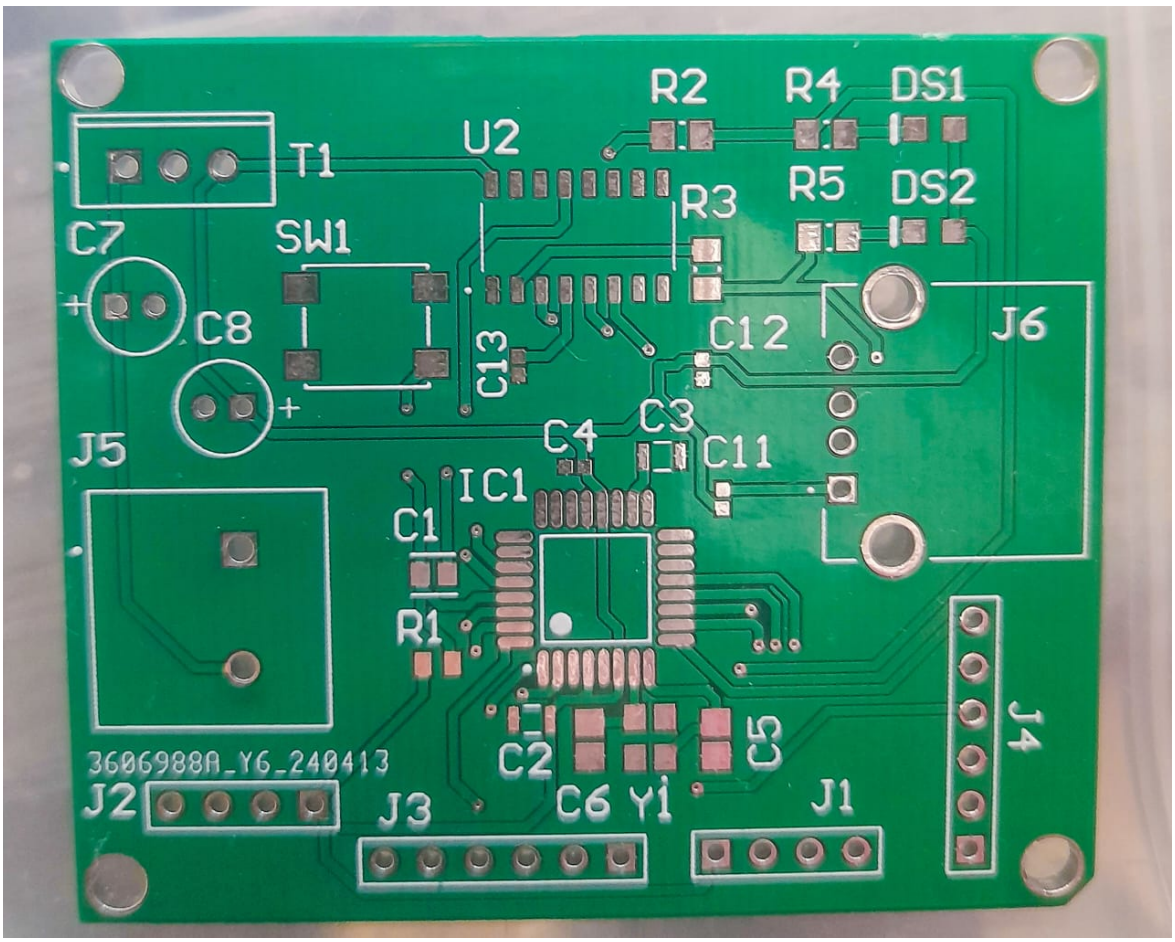


Figure 14: Printed PCB

3.7.7 Soldered PCB

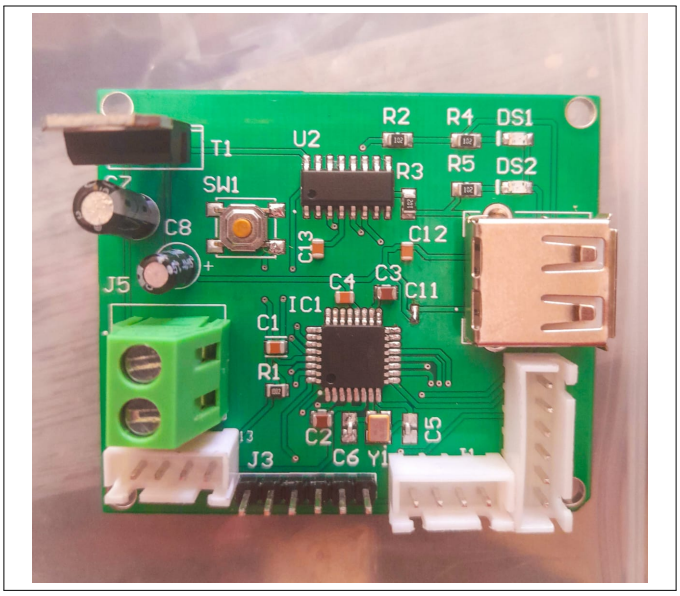


Figure 15: Soldered PCB

Figure 16: Soldered PCB

3.7.8 PCB Testing

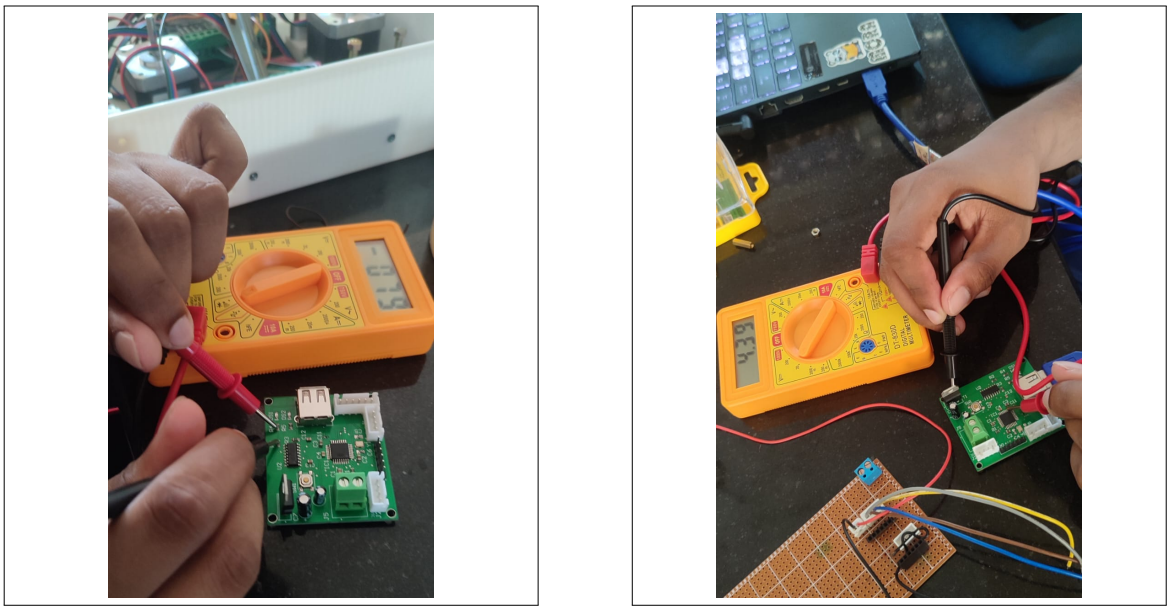


Figure 17: PCB Testing Images

4. Mechanical Design

4.1 Mechanical components

The mechanical design of our 3D scanner machine focuses on ensuring precise and smooth vertical movement of the ToF sensor, which is crucial for accurate distance measurements. The design utilizes several key components:

- Smooth Shafts: Two smooth shafts provide stability and guide the vertical movement of the ToF sensor. These shafts ensure that the movement is straight and free of wobble, which is essential for accurate scanning.

Figure 18: Smooth Shafts

- Screw Shaft: The screw shaft is responsible for the vertical displacement of the ToF sensor. It is driven by a stepper motor, which allows for precise control of the movement. The screw shaft ensures that the vertical position of the ToF sensor can be finely adjusted.

Figure 19: Screw Shaft

- Screw Shaft Coupler: A screw shaft coupler is used to connect the stepper motor to the screw shaft. This component ensures that the rotational movement of the stepper motor is accurately transferred to the screw shaft, facilitating precise vertical movement.

Figure 20: Screw Shaft Coupler

- Linear Bearings: Linear bearings are used to allow the connector part with the ToF sensor to slide smoothly along the smooth shafts. These bearings minimize friction and ensure smooth, consistent motion, which is critical for maintaining the accuracy of the scanning process.

Figure 21: Linear Bearings

- Shaft Holders: Shaft holders are used to securely hold the smooth shafts in place. They provide the necessary support to keep the shafts aligned and stable, ensuring the overall stability of the vertical motion system.

Figure 22: Shaft Holder

- Switches and Push Buttons: Various switches and push buttons are incorporated into the design to control the functionalities of the 3D scanner. These controls provide a user-friendly interface for operating the machine, allowing for actions such as starting and stopping the scan, adjusting the position of the ToF sensor, and other operational commands.

Figure 23: Switch and Push Button

The mechanical components of our 3D scanner are carefully chosen to provide stability, precision, and ease of control. The smooth shafts and linear bearings ensure smooth vertical movement, the screw shaft and coupler provide precise control, and the switches and push buttons offer a convenient interface for operating the scanner.

4.2 Solidwork Design Details

4.2.1 Turnable Table

- Diameter and Design
  - The turnable table has a diameter of 14 cm, making it suitable for scanning a variety of objects.
  - Attachment to Stepper Motor: The table is precisely mounted on a NEMA 17 stepper motor, ensuring accurate rotational movement. The stepper motor is controlled by the microcontroller to achieve the necessary 360-degree scans.

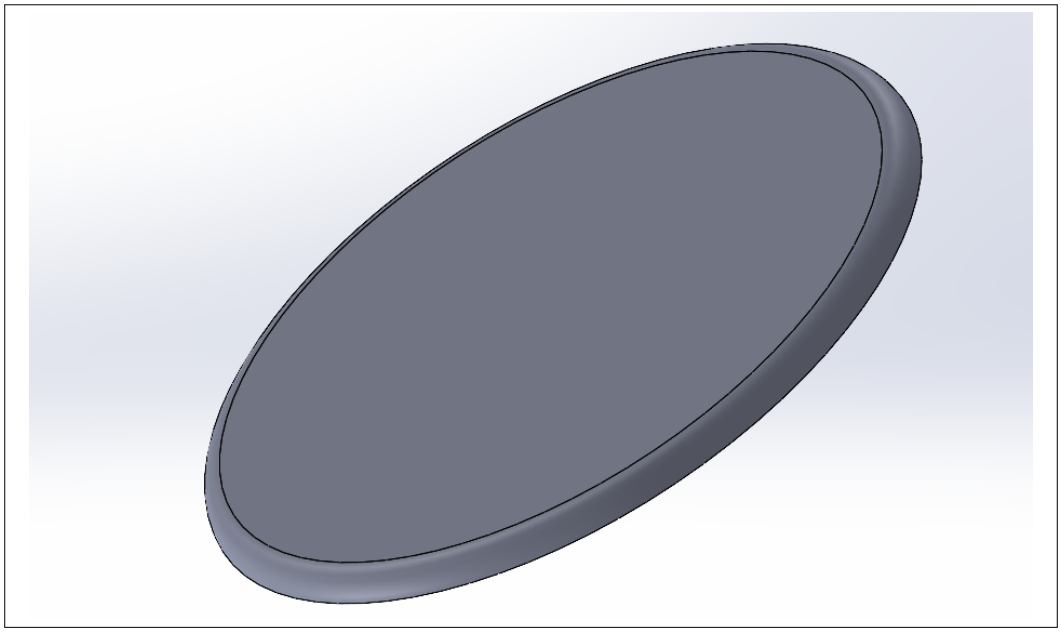


Figure 24: 3D Scanner Turnable Table top

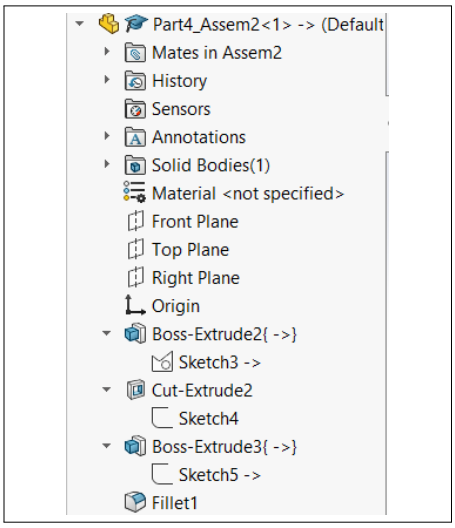


Figure 25: Design Tree

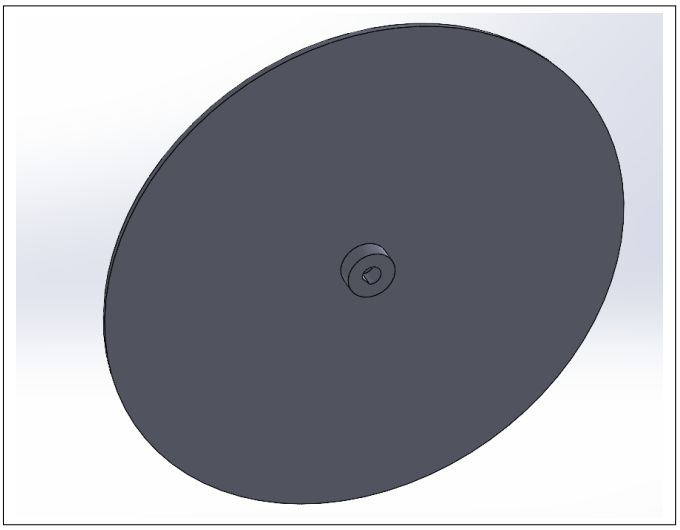


Figure 26: Tray Bottom

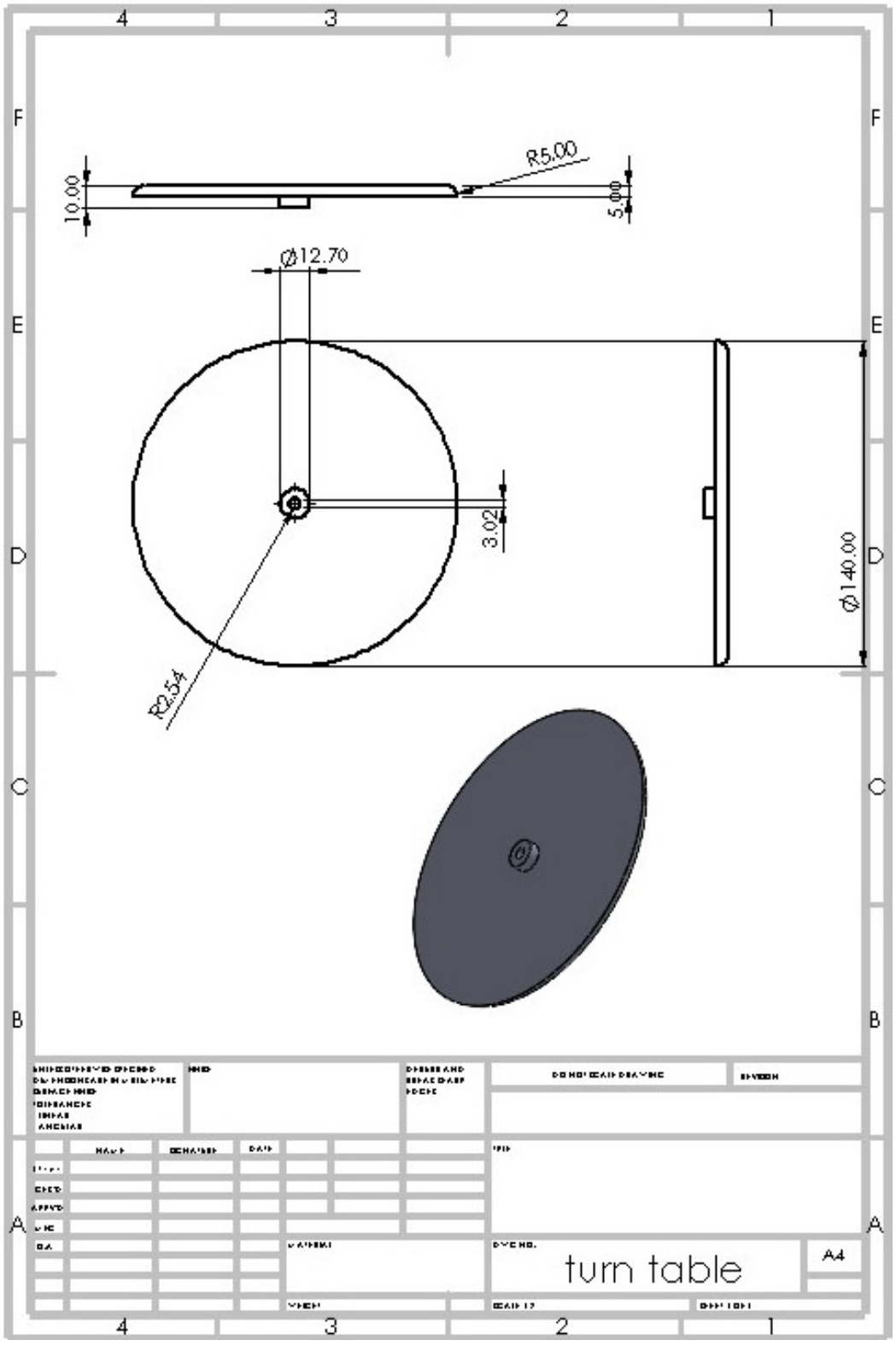


Figure 27: Turnable Table

Figure 28: 3D Scanner Tray

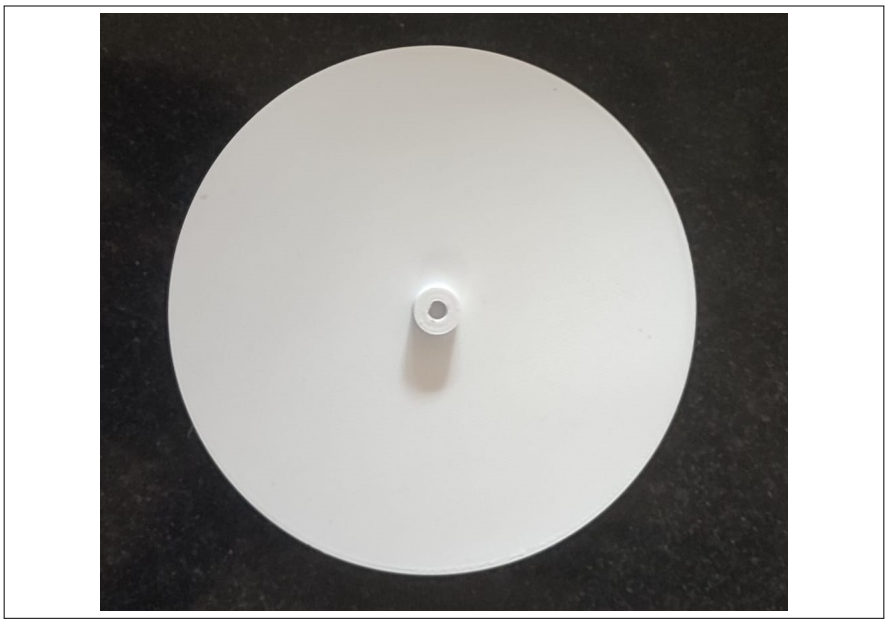


Figure 29: Tray bottom

4.2.2 Linear Slider Connnector Part

The linear slider part of our 3D scanner machine is designed to hold the ToF sensor and facilitate its smooth and precise vertical movement along the Z-axis. This component is integral to ensuring accurate distance measurements during the scanning process.

The linear slider, measuring 10 cm in width and 6 cm in height, is engineered to slide through two smooth shafts, which provide stability and guide the movement. These shafts ensure that the slider moves in a straight line without any wobble, maintaining the accuracy of the ToF sensor's readings.

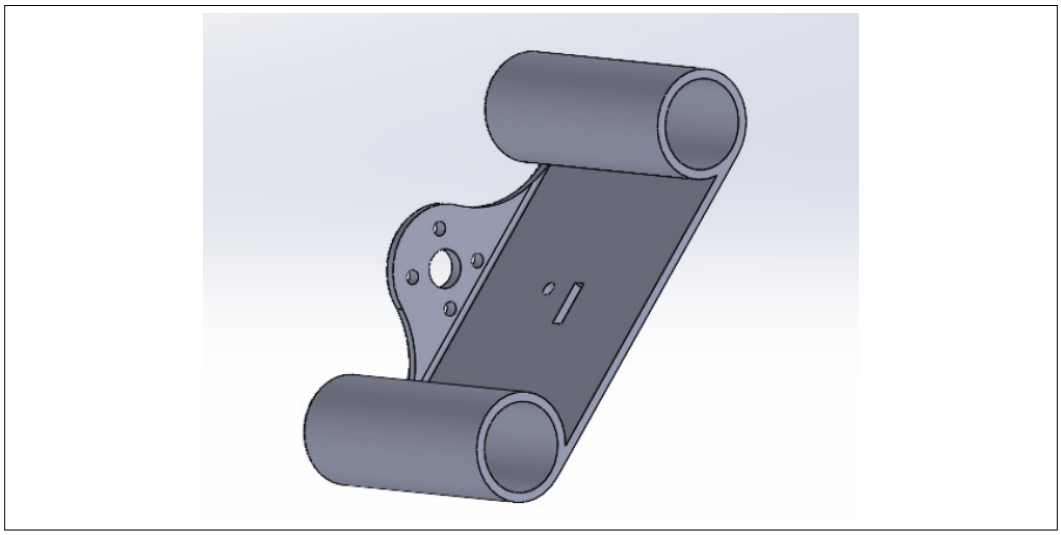


Figure 30: 3D Scanner Moving Holder

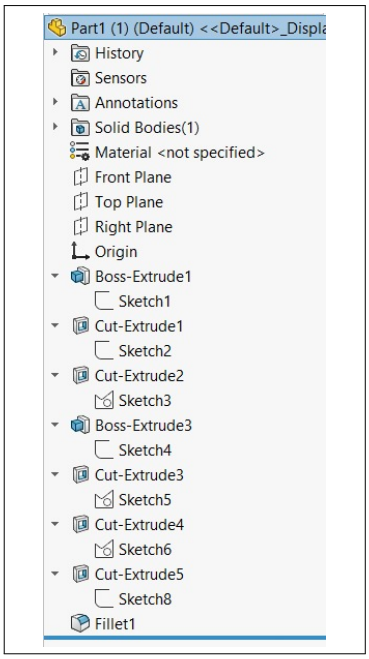


Figure 31: Design Tree

Figure 32: Connector part

To achieve the vertical displacement of the ToF sensor, a screw shaft is utilized. The screw shaft, driven by a stepper motor, precisely controls the vertical position of the slider. The screw shaft ensures that the

movement is finely adjustable, allowing the ToF sensor to move in small, controlled increments along the Z-axis.

Linear bearings are incorporated into the design of the slider to facilitate smooth sliding along the smooth shafts. These bearings minimize friction and ensure consistent, fluid motion, which is essential for maintaining the precision of the scanning process.

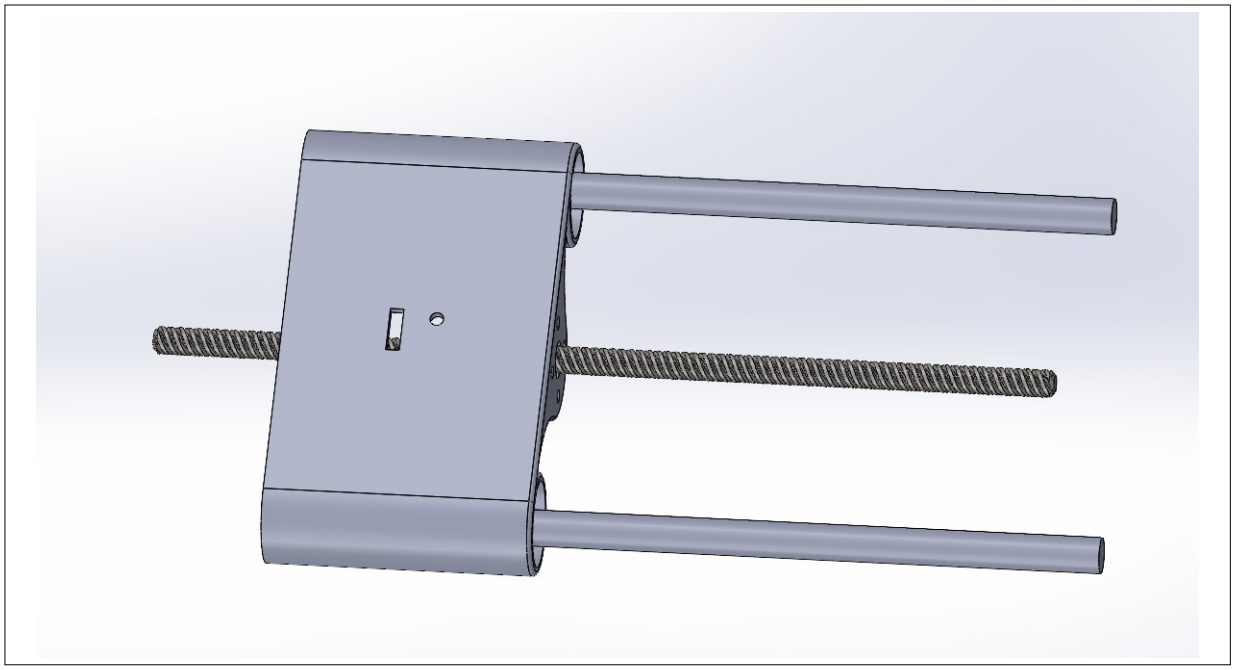


Figure 33: Slider Connection

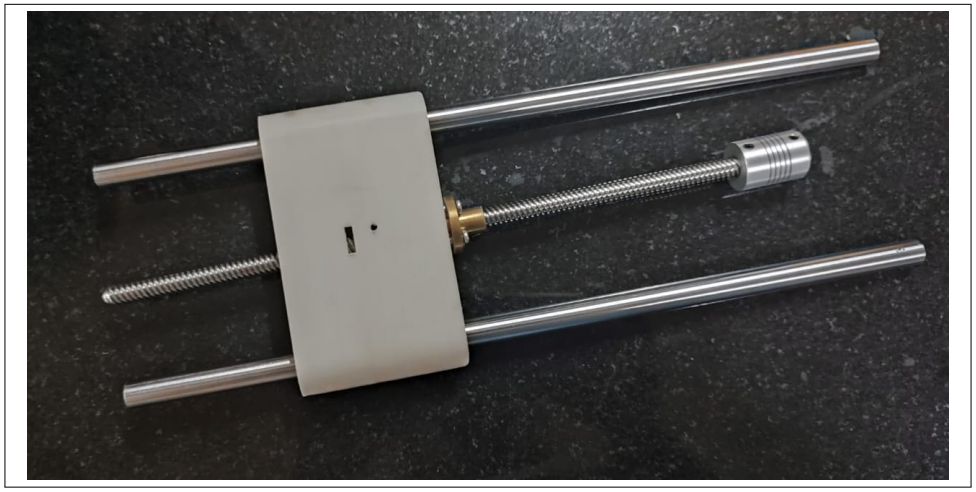


Figure 34: Sub system Integration

Overall, the linear slider part is a critical component of your 3D scanner, designed to hold the ToF sensor securely while enabling precise and stable vertical movement. The combination of smooth shafts, screw shaft, and linear bearings ensures high accuracy and reliability in the scanning process.

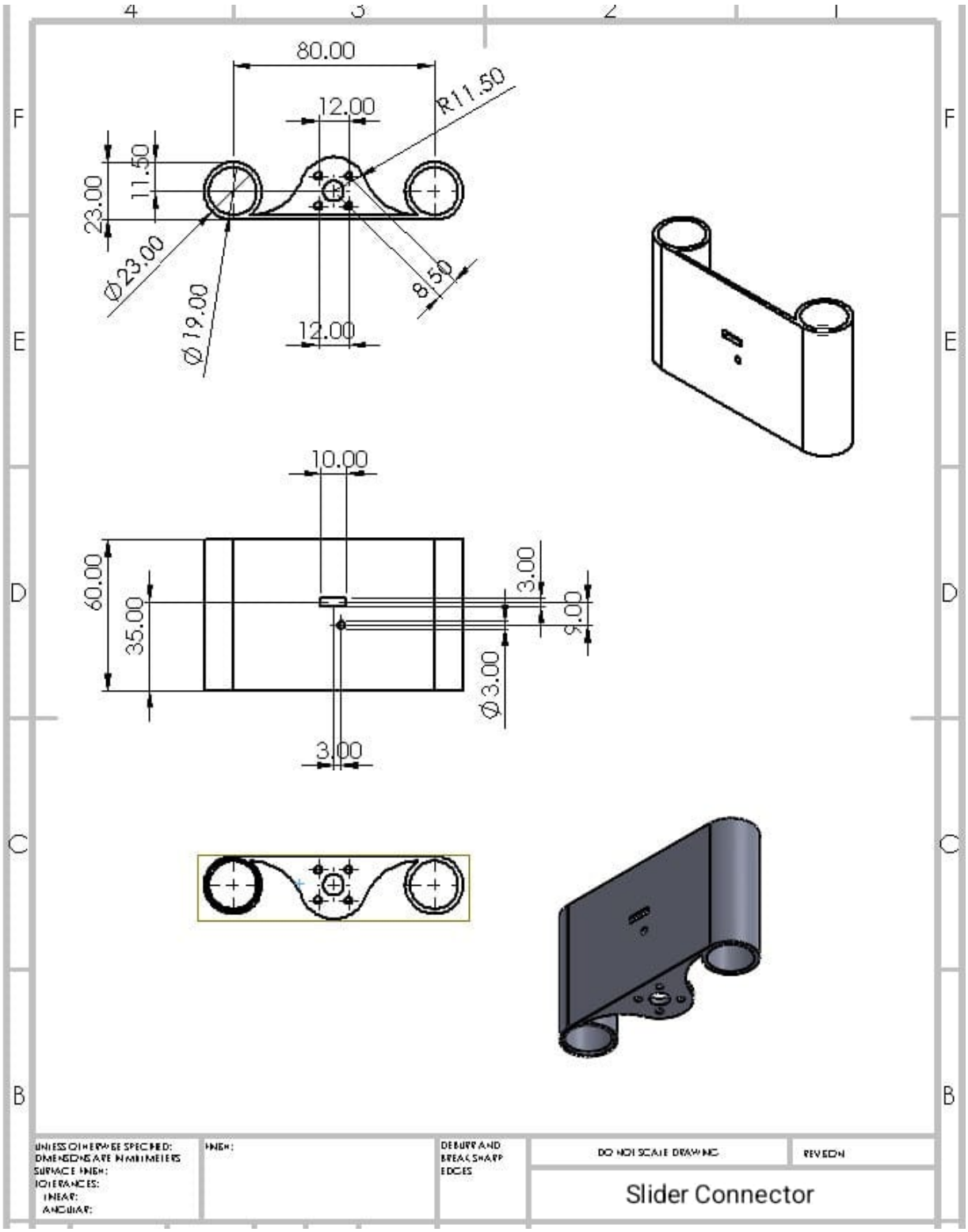
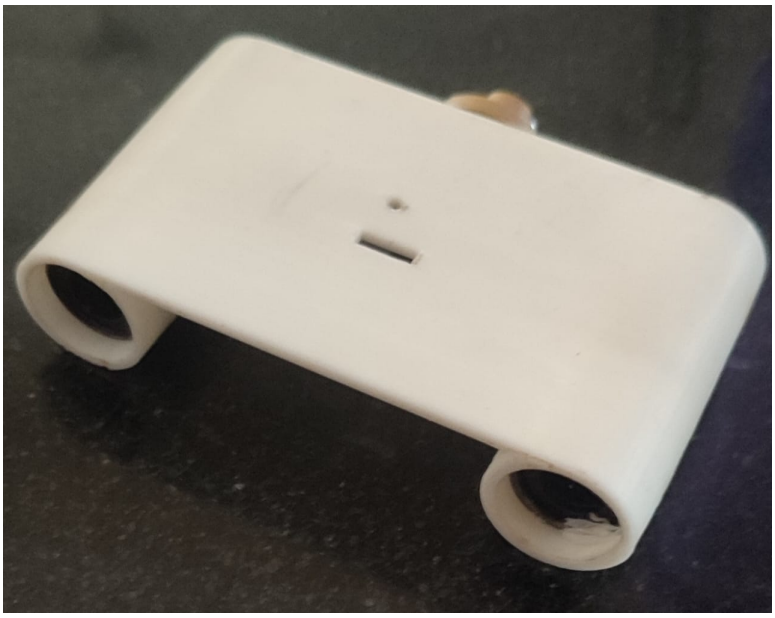


Figure 35: Slider Connection



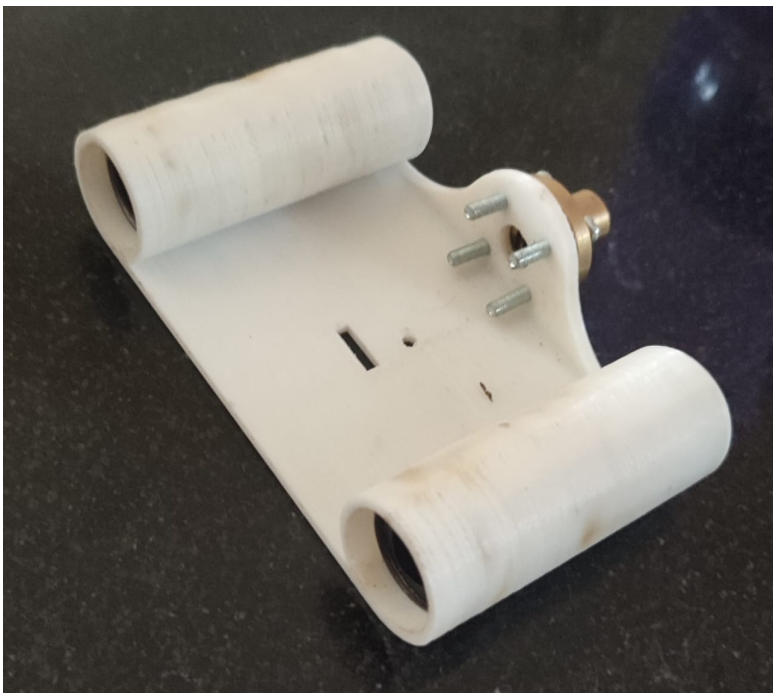


Figure 36: Shafts Connector

4.3 Enclosure Design

4.3.1 Main Base

The main base of our 3D scanner machine is a crucial component that houses and supports various essential parts, ensuring the overall stability and functionality of the system. This base is designed to hold the PCB, stepper motors, shaft holders, and stepper motor drivers securely within its structure.

The base is meticulously crafted to include specific features for mounting and organizing these components. Holes are strategically placed to hold the PCB in a vertical position, which optimizes space and allows for easy access to connections and adjustments. This vertical orientation of the PCB also facilitates better airflow and cooling.

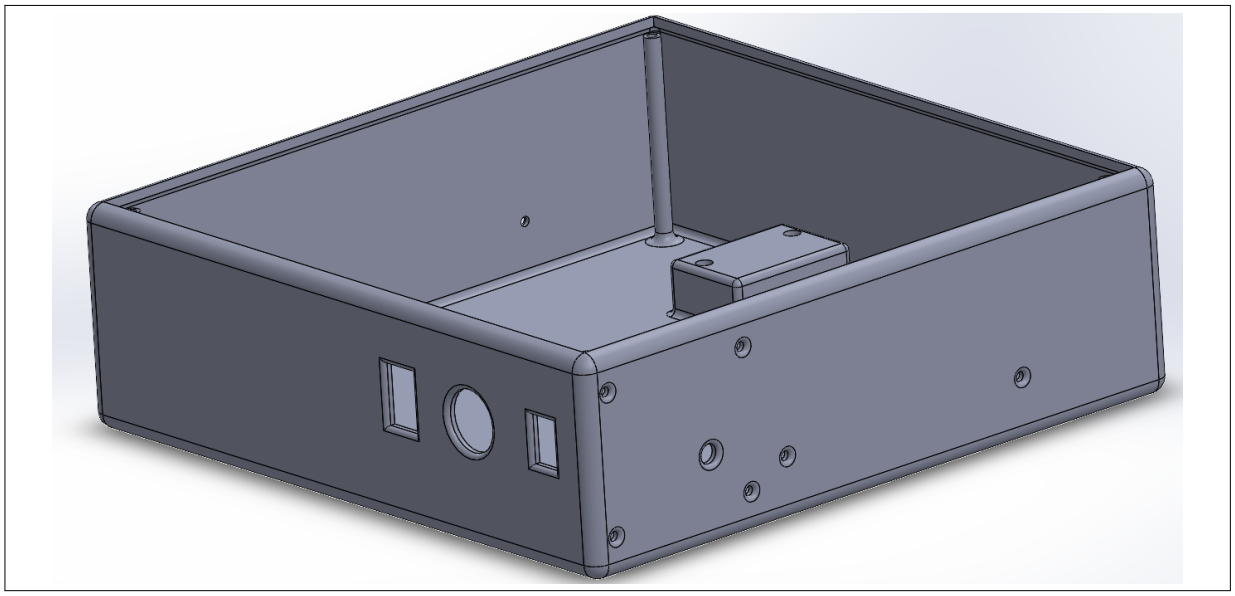


Figure 37: 3D Scanner Main base

Figure 38: Main base Design Tree

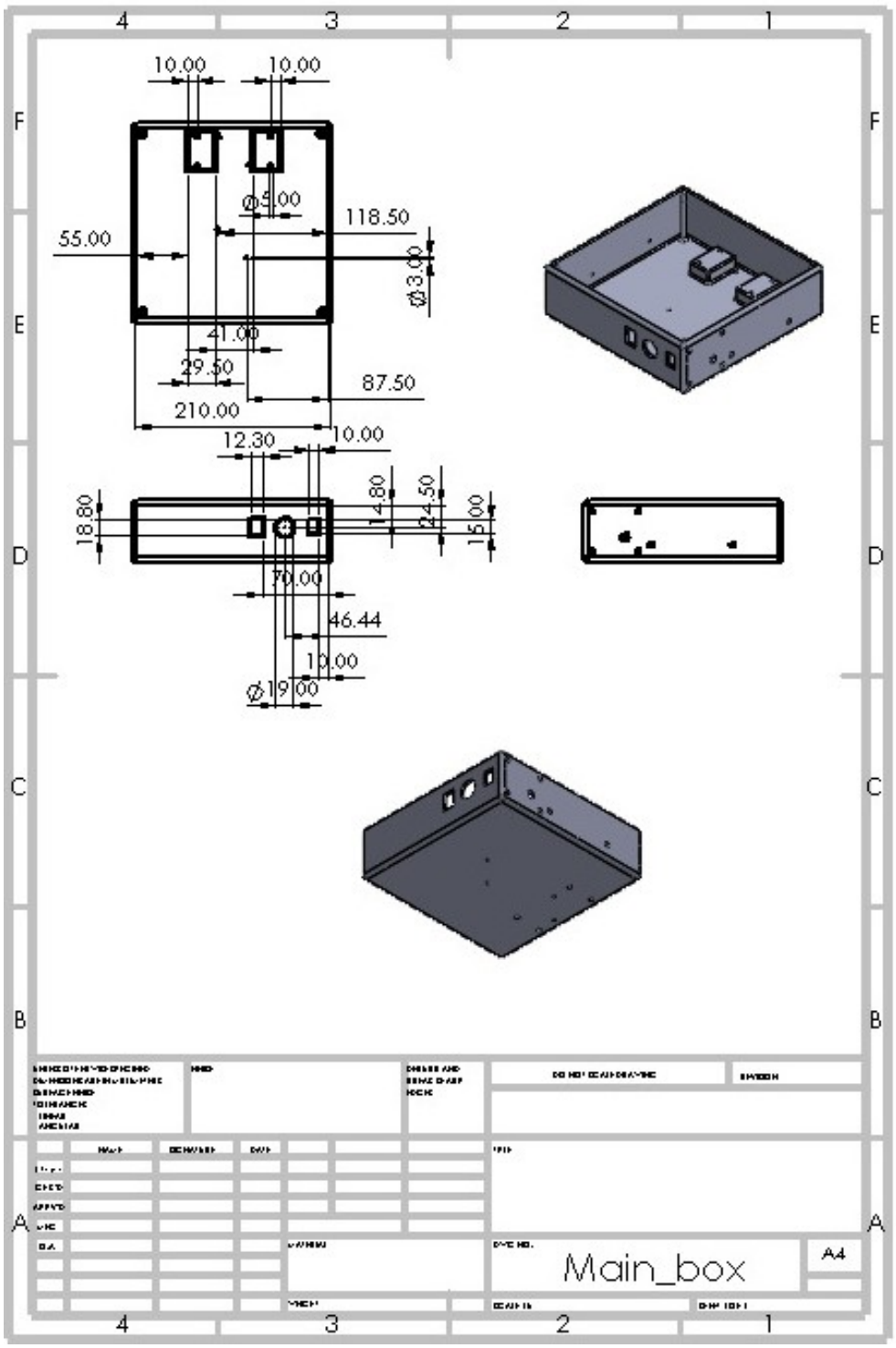


Figure 39: Main Base - Enclosure

Within the main base, stepper motors are securely mounted to drive the screw shaft and turntable mechanisms. Shaft holders are also fixed within the base to ensure the smooth shafts are held firmly in place, providing stability for the linear slider's movement.

Stepper motor drivers are installed inside the main base to control the motors with precision. These drivers are positioned to maintain efficient wiring and signal integrity, minimizing interference and ensuring reliable operation.

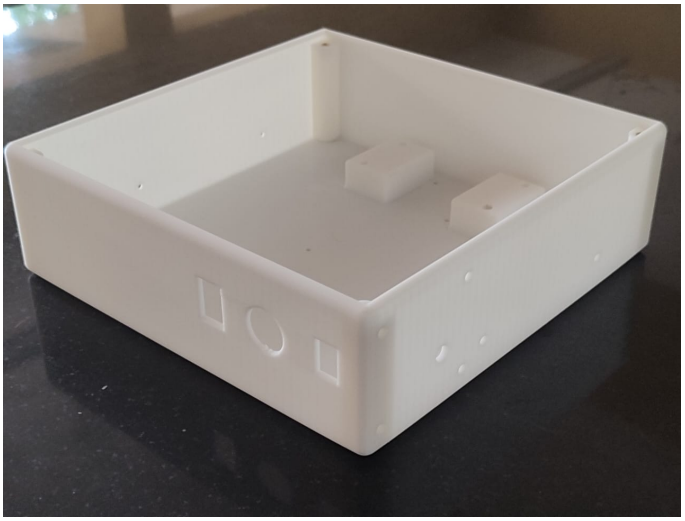
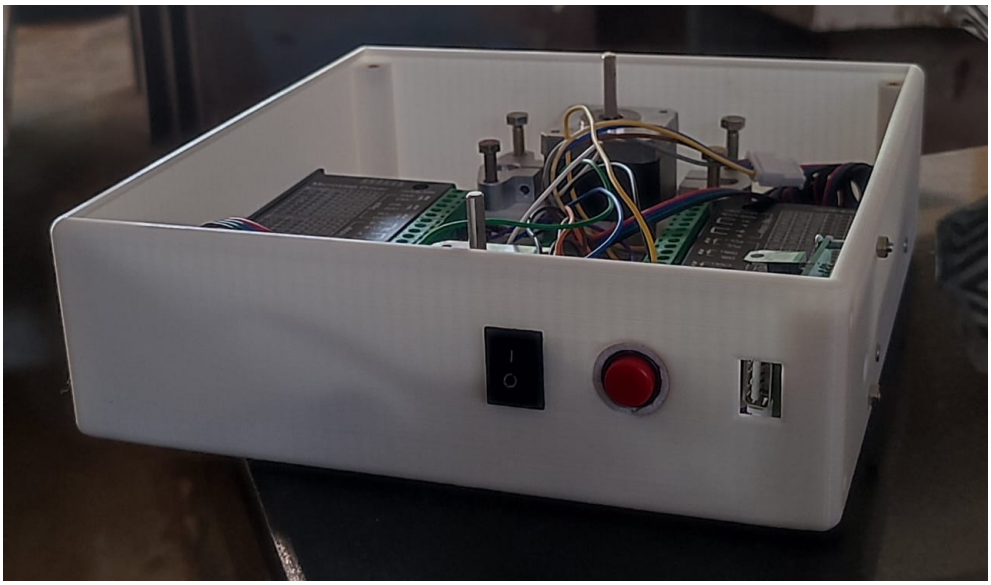


Figure 40: 3D Scanner Main base

Figure 41: Main base top

Additionally, the base is designed with user interaction in mind. It includes cutouts and placements for a USB socket, an on/off switch, and a push button. The USB socket allows for easy connection to the computer for real-time data transfer, while the on/off switch and push button provide convenient controls for powering the system and initiating or stopping the scanning process.

The main base of our 3D scanner is a well-designed structure that organizes and supports the PCB, stepper motors, shaft holders, and stepper motor drivers. It ensures stability, efficient wiring, and user-friendly access to controls, contributing to the overall functionality and reliability of the scanning system.



4.3.2 Tray Lid Part

The lid for the main base part of our 3D scanner machine is designed to provide both protection and functionality, while allowing necessary components to interface seamlessly with the rest of the system. This lid plays a vital role in ensuring the smooth operation of the scanner and maintaining the alignment and stability of critical parts.

The lid is precisely cut to include holes for the motor shafts and smooth shafts. These holes are strategically positioned to allow the shafts to pass through the lid without obstruction, ensuring the continued smooth and accurate movement of the linear slider and other mechanical components. The alignment of these holes is crucial for maintaining the stability and precision of the Z-axis motion.

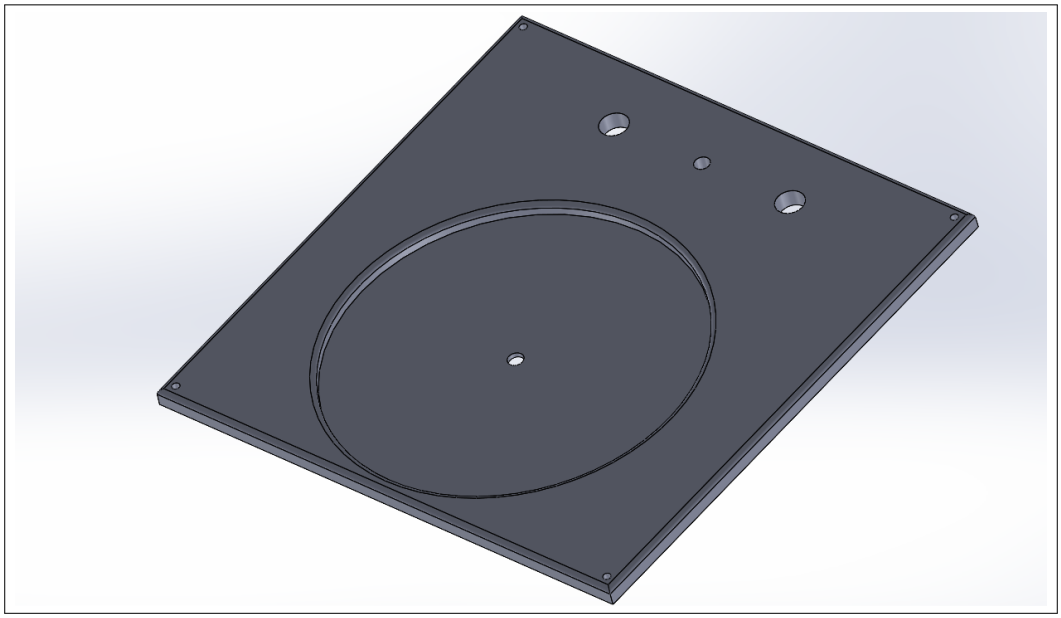


Figure 42: 3D Scanner Top Lid

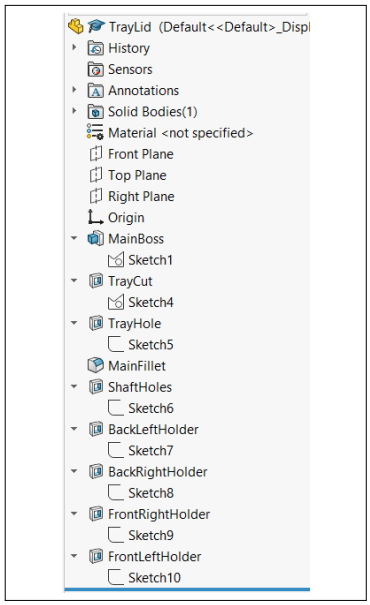


Figure 43: Design Tree

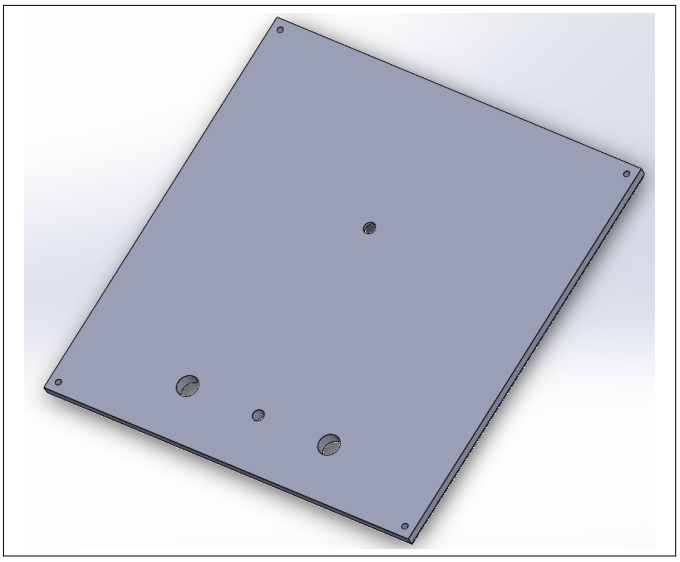


Figure 44: Lid Bottom

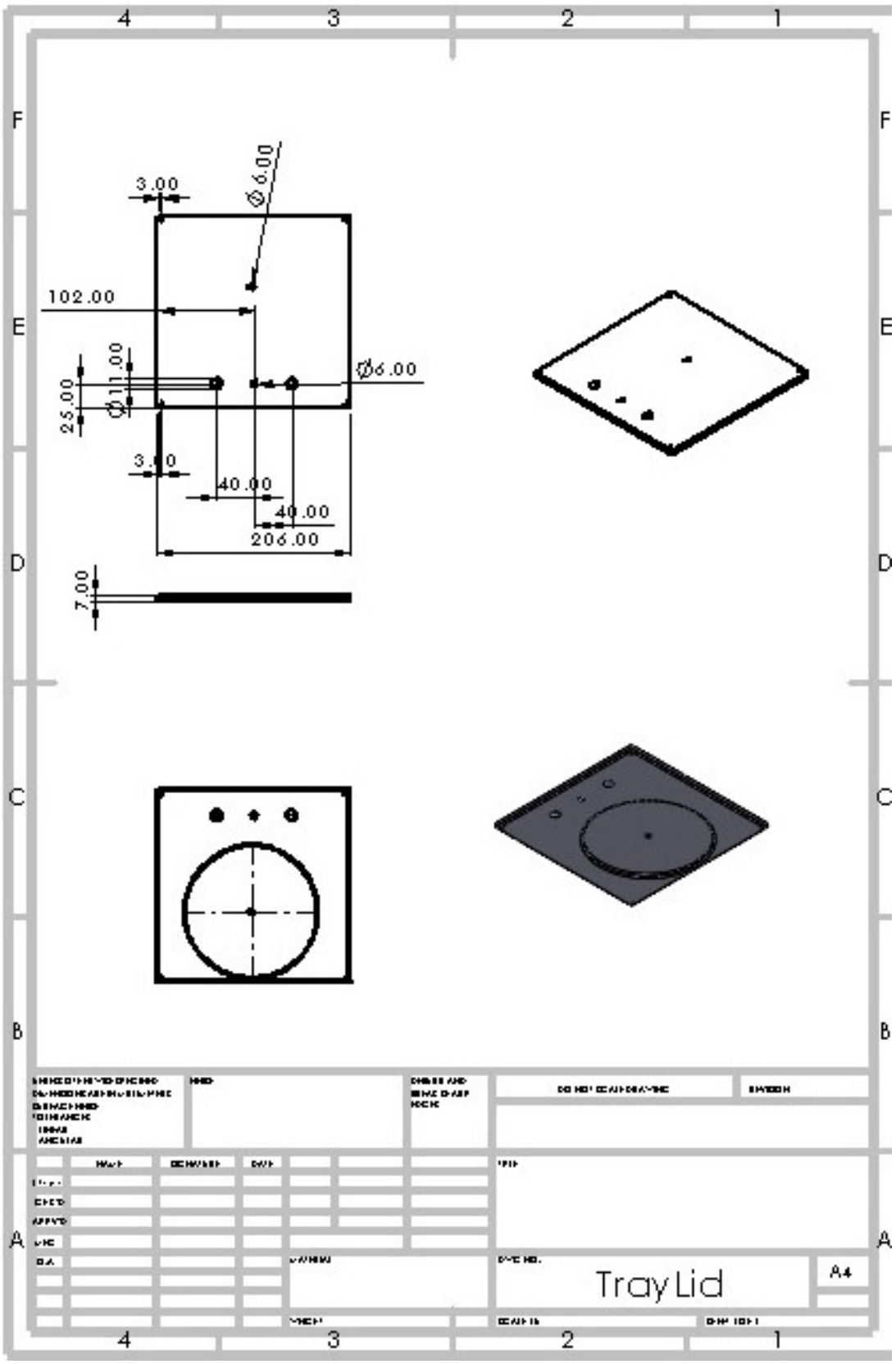


Figure 45: Tray Lid - Enclosure

Additionally, a section of the lid is cut out to accommodate the turnable table. This cutout is carefully sized to securely hold the turnable table in place while allowing it to rotate freely during the scanning process. The integration of the turnable table into the lid ensures that the scanned object is positioned correctly and remains stable throughout the scan.

The design of the lid also takes into consideration the need for easy access and maintenance. The cutouts and holes are designed to facilitate quick assembly and disassembly, making it convenient to access internal components such as the stepper motors, shafts, and electronic parts for adjustments or repairs.

The lid for the main base part of our 3D scanner is a well-engineered component that enhances the functionality and reliability of the machine. It includes precise holes for motor shafts and smooth shafts, as well as a cutout for the turnable table, ensuring smooth operation and easy access to critical components.

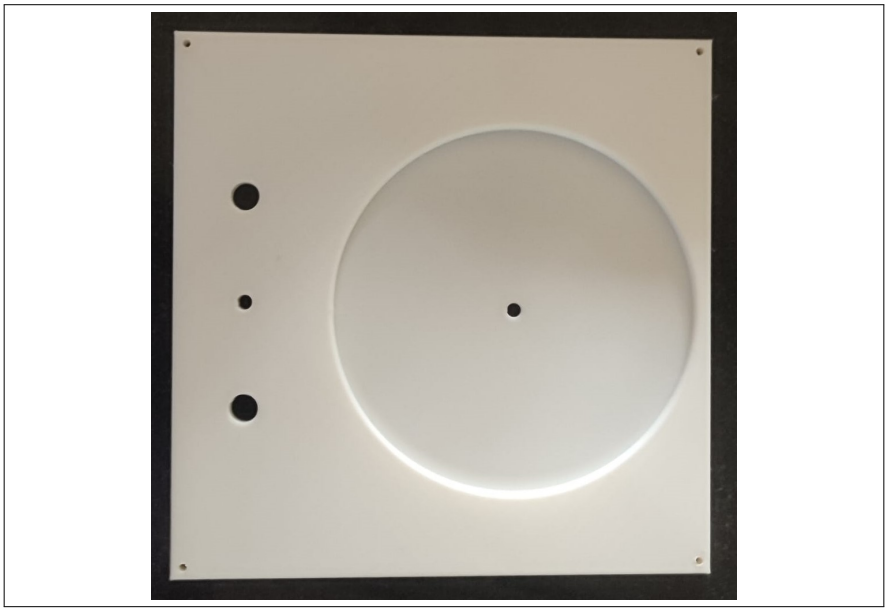


Figure 46: 3D Scanner Top lid

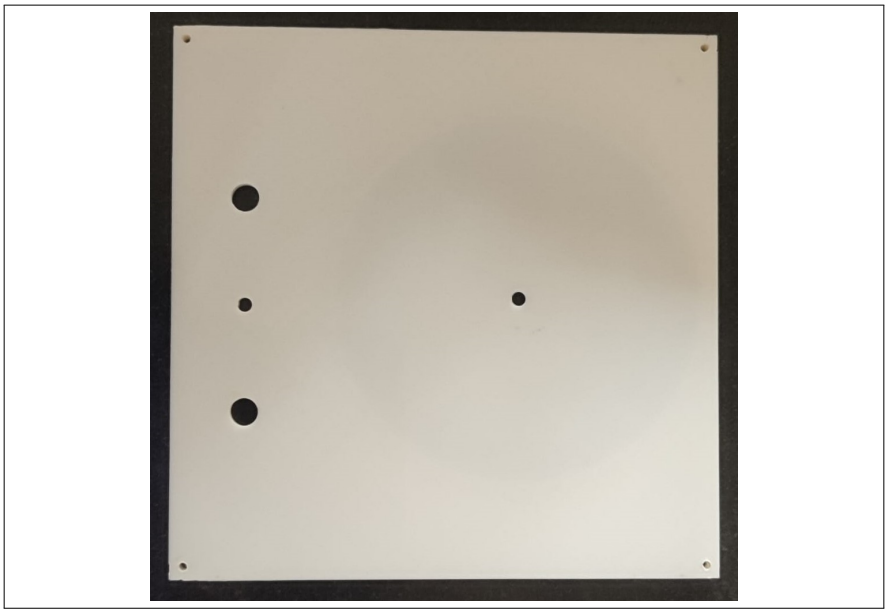


Figure 47: Top lid bottom

4.4 Final assembly - Enclosure

4.4.1 Solidwork Asseembly

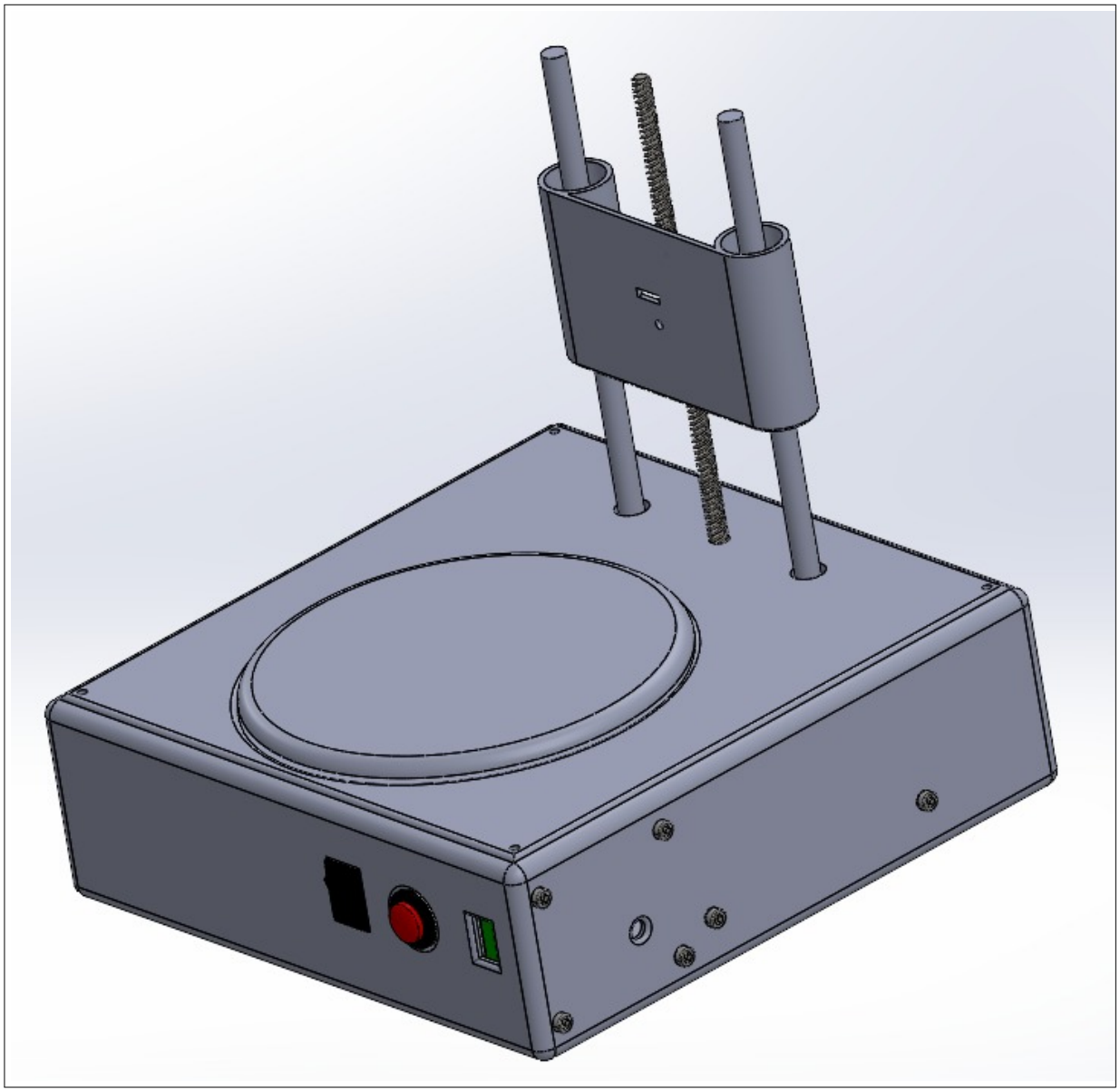


Figure 48: 3D Scanner - Final Assembly

4.4.2 Final Product

Figure 49: Final 3D Scanner Assemble

5. Software Details

5.1 Scanner Code

In the development of our 3D scanner machine, we opted for register-level programming and port manipulation instead of relying on standard Arduino libraries. This approach allowed us to optimize the code for speed and efficiency, which is critical for the real-time requirements of our scanning system. By directly accessing the hardware registers and manipulating ports, we achieved faster execution times and greater control over the hardware components.

For the stepper motor control, we developed a custom library to precisely control the stepper motor that drives the screw shaft for the Z-axis movement. This involved configuring the timer registers to create accurate timing intervals for the steps and using port manipulation to toggle the control pins of the stepper motor driver. This direct control allowed us to generate the precise step pulses required for smooth and accurate movement of the linear slider.

In terms of sensor reading, we directly accessed the I2C registers to communicate with the VL53L0X ToF sensor, bypassing the standard Wire library. This required setting up the I2C communication by configuring the relevant control and status registers, sending commands to initiate distance measurements, and reading the results directly from the sensor's registers. This low-level control over the I2C communication process resulted in faster data acquisition and reduced overhead, which is crucial for real-time distance measurements.

We also implemented a custom two-wire interface (TWI) library to facilitate communication between the microcontroller and I2C devices, such as the ToF sensor. By directly configuring and controlling the I2C hardware of the microcontroller, we set the correct baud rate for I2C communication, handled start and stop conditions, and managed data transfer with precise control over ACK/NACK signals. This provided us with reduced latency and improved reliability in data communication.

For creating precise timing delays required for various operations, we developed custom delay functions using timer registers. By configuring the timer to generate interrupts at specific intervals, we created accurate microsecond and millisecond delays. This method ensured the precision of the delays, which is essential for coordinating the movements of the stepper motor and the timing of sensor readings.

In summary, our use of register-level programming and port manipulation allowed us to achieve a high level of performance and precision in our 3D scanner machine. This approach provided us with the necessary control to optimize the system for real-time operation, ensuring accurate and efficient scanning. By bypassing standard libraries and directly interacting with hardware registers, we minimized latency, increased execution speed, and maintained precise control over all hardware components.

5.1.1 Main Code

```
//main function

#include <avr/io.h>
#include "delay.h"
#include "uart.h"
#include "TWI.h"
#include "ToF.h"
#include "stepper.h"

int main(){
  init_uart();
  i2c_init();
  vl53l1x_init();
```

```
  init_stepper();
  DDRD |= (1<<DDD4);  //pin PD5 output - z_dir

  while(1){

    //trans_string("Distance: ");
    //delay_ms(100);


    uint8_t vert_distnce = 10;   /*max Z axis distance*/
    uint8_t z_steps = 200;  // z steps per rotation
    uint32_t z_counts = (200/1*vert_distnce)/z_steps;  //total z_counts
        ntil return home
    uint8_t theta_counts = 200;

    PORTD &= (~(1<<PORTD4)); // z_dir LOW
    for(uint32_t j=0;j<z_counts;j++)
    {
      for(uint8_t i=0; i<theta_counts;i++)
      {
        rotate_tmotor(1);   //start rotate turn table
        //double sense_distance = 0; //reset distance measurement
        vl53l1x_measure_distance();
        uint16_t distance = vl53l1x_read_distance();
        trans_num(distance);
        trans_string("\n");

      }
      rotate_zmotor(z_steps); //start rotating z axis motor
      delay_ms(1000);
      //serial print(9999);
      trans_num(9999);
      trans_string("\n");
    }
    PORTD |= (1<<PORTD4); // move z to home
    delay_ms(10); //delay 10ms
    for(uint32_t k=0;k<z_counts;k++){
      rotate_zmotor(z_steps);
      delay_ms(10);
    }

  }
  return 0;
}
```

Listing 1: Main.cpp

5.1.2 Stepper Code

```
//stepper library
void init_stepper(){
  //setup stepper motor pins
```

```
  //setup turn table motor
        DDRD |= (1<<DDD6);  //pin PD6 output - t_enable
        DDRD |= (1<<DDD3);  //Pin PD3 Output - t_pulse
        DDRD |= (1<<DDD2);  //pin PD2 output - t_dir

  //set up z axis motor
        DDRD |= (1<<DDD7);  //pin PD7 output - z_enable
        DDRD |= (1<<DDD5);  //pin PD5 output - z_pulse
        DDRD |= (1<<DDD4);  //pin PD5 output - z_dir

        PORTD &= (~(1<<PORTD6));  //t_enable low
        PORTD &= (~(1<<PORTD7));  //z_enable low

}

//rotate turntable motor
void rotate_tmotor(uint8_t steps){
        for(uint8_t t=0;t<steps;t++){
                PORTD &= (~(1<<PORTD3)); //low t_pulse
                delay_ms(1);
                PORTD |= (1<<PORTD3); //high t_pulse
                delay_ms(1);

        }

}


//rotate z axis motor
void rotate_zmotor(uint8_t steps){
        for(uint8_t i=0;i<steps;i++)
        {
                PORTD &= (~(1<<PORTD5)); //low t_pulse
                delay_ms(1);
                PORTD|= (1<<PORTD5); //high t_pulse
                delay_ms(1);
        }
}
```

Listing 2: stepper.cpp

```
#ifndef STEPPER
#define STEPPER

void init_stepper();
void rotate_tmotor(uint8_t steps);
void rotate_zmotor(uint8_t steps);

#endif
```

Listing 3: stepper.h

5.1.3 Two Wire Interface (I2C) Implementation code

```
//two wire interface

void i2c_init() {
    // Set the clock speed
    TWSR = 0x00;
    TWBR = ((F_CPU/400000)-16)/(2*pow(4,(TWSR&((1<<TWPS0)|(1<<TWPS1))))
       );
}

void i2c_start() {
    // Send start condition
    TWCR = (1 << TWSTA) | (1 << TWEN) | (1 << TWINT);
    while (!(TWCR & (1 << TWINT)));
}

void i2c_stop() {
    // Send stop condition
    TWCR = (1 << TWSTO) | (1 << TWEN) | (1 << TWINT);
}

void i2c_write(uint8_t data) {
    // Write data to the I2C bus
    TWDR = data;
    TWCR = (1 << TWEN) | (1 << TWINT);
    while (!(TWCR & (1 << TWINT)));
}

uint8_t i2c_read_ack() {
    // Read data from the I2C bus with acknowledgment
    TWCR = (1 << TWEN) | (1 << TWINT) | (1 << TWEA);
    while (!(TWCR & (1 << TWINT)));
    return TWDR;
}

uint8_t i2c_read_nack() {
    // Read data from the I2C bus without acknowledgment
    TWCR = (1 << TWEN) | (1 << TWINT);
    while (!(TWCR & (1 << TWINT)));
    return TWDR;
}
```

Listing 4: TWI.cpp

```
#ifndef TWI
#define TWI

void i2c_init();
void i2c_start();
void i2c_stop();
void i2c_write(uint8_t data);
uint8_t i2c_read_ack();
```

```
uint8_t i2c_read_nack();

#endif
```

Listing 5: TWI.h

5.1.4 ToF Sensor Reading code

```
//ToF sensor library



#define VL53L1X_ADDR 0x52

void vl53l1x_init() {
    // Example initialization sequence
    i2c_start();
    i2c_write(VL53L1X_ADDR); // Device address
    i2c_write(0x00); // Register address
    i2c_write(0x01); // Data to write
    i2c_stop();

}

void vl53l1x_measure_distance(){
  i2c_start();
  i2c_write(VL53L1X_ADDR); // Device address
  i2c_write(0x00);
  i2c_write(0x02);
  i2c_stop();
}

uint16_t vl53l1x_read_distance() {
    uint16_t distance;

    // Read distance high byte
    i2c_start();
    i2c_write(VL53L1X_ADDR); // Device address
    i2c_write(0x14 + 10); // Register address for distance high byte
    i2c_start(); // Repeated start
    i2c_write(VL53L1X_ADDR | 0x01); // Read mode
    uint8_t high_byte = i2c_read_ack();

    // Read distance low byte
    uint8_t low_byte = i2c_read_nack();
    i2c_stop();

    distance = (high_byte << 8) | low_byte;
    // distance = low_byte;

    return distance;
}
```

Listing 6: TOF.cpp

```
#ifndef ToF
#define ToF

void vl53l1x_init();
void vl53l1x_measure_distance();
uint16_t vl53l1x_read_distance();

#endif
```

Listing 7: TOF.h

5.1.5 UART Communication Implementation (Serial communication)

```
//UART Transmission


//receive one byte data
unsigned char USART_Receive()
{
  while(!(UCSR0A & (1<<RXC0)));  //receive buffer empty
  return UDR0; //receive reg
}


//transmit one byte
void UART_Txchar(char ch)
{
  while(!(UCSR0A & (1<<UDRE0))); //WAIT UNTIL EMPTY TRANSMIT BUFFER
  UDR0 = ch; // after transmit buffer empty load data to the uart data
     register

}


//transmit string
void trans_string(char str[]){
  int lenght = strlen(str);
  char trans;
  for(int i=0;i<lenght;i++){
    trans = str[i];
    UART_Txchar(trans);
  }
}

//transmit number
void trans_num(uint16_t num){
  String number = String(num);
  int lenght = number.length();
  char trans;
```

```
  for(int i=0;i<lenght;i++){
    trans = number[i];
    UART_Txchar(trans);
  }
  // UART_Txchar('\n');
}

void init_uart(){
  //Enable transmitter and recirever bits - Use USART0

  UCSR0B |= (1<<RXEN0) | (1<<TXEN0); // ENABLE RX AND RX

  //SET THE DATA SIZE FOR COMMUNIATION
  UCSR0C &= (~(1<<UMSEL00)) & (~(1<<UMSEL01));   //ENABLE ASYNCHRONOUS USART COMMUNICATION
  UCSR0C &= (~(1<<UPM00)) & (~(1<<UPM01));   // DISABLE PARITY BIT
  UCSR0C &= (~(1<<USBS0)); //CHOOSE ONE STOP BIT

  //SET DATA LENGHT TO BE 8 BITS
  UCSR0B &= (~(1<<UCSZ02));
  UCSR0C |= (1<<UCSZ00) | (1<<UCSZ01);  //SET 8BITS 011 BITS IN UCSZ02, UCSZ00,UCSZ01

  //SET THE SPEED FOR TRANSMISSION

  UCSR0A |= (1<<U2X0); //SELSCT HIGH SPEED MODE
  // UCSR0A &= ~(1<<U2X0); // low speed

  //BAUDRATE
  UBRR0 = 207; //9600 BAUDRATE FROM DATASHEET DIRECT TABLE FOR U2X0=1 if u2x=0 , ubrr0 = 103
}
```

Listing 8: uart.cpp

```
#ifndef UART
#define UART

unsigned char USART_Receive();
void UART_Txchar(char ch);
void trans_string(char str[]);
void trans_num(uint16_t num);
void init_uart();

#endif
```

Listing 9: uart.h

5.1.6 Delay Function - Time Delay

```
// delay function

```

```
3
4  void delay_ms(unsigned int ms) {
5    for (unsigned int i = 0; i < ms; i++) {
6        // Configure Timer1
7        TCCR1A = 0;  // Clear Timer1 Control Register A
8        TCCR1B = 0;  // Clear Timer1 Control Register B
9        TCNT1 = 0;   // Clear Timer1 Counter
10
11       // Set Timer1 to CTC mode and set prescaler to 64
12       TCCR1B |= (1 << WGM12) | (1 << CS11) | (1 << CS10);
13
14       // Set compare match register for 1ms delay
15       OCR1A = 249;
16
17       // Wait for compare match flag to be set
18       while (!(TIFR1 & (1 << OCF1A)));
19
20       // Clear compare match flag
21       TIFR1 |= (1 << OCF1A);
22   }
23 }
```

Listing 10: delay.cpp

```
1 #ifndef DELAY
2 #define DELAY
3
4 void delay_ms(unsigned int ms);
5
6 #endif
```

Listing 11: delay.cpp

5.2 Getting Serial Data to the PC

We implement a Python script to facilitate real-time data acquisition from a device connected via serial communication (specifically on port 'COM5' at a baud rate of 9600). It employs the serial module to establish and manage the serial connection, while a custom 'ReadLine' class efficiently reads incoming data in line-by-line fashion. Upon initialization, the script opens a file named 'tofreadings.txt' in write mode to store received data. Inside a continuous loop, it reads lines of text from the serial connection using the 'readline' method of the 'ReadLine' class instance. Each line is printed to the console and simultaneously written to the text file. The file.flush() command ensures that data is promptly saved to disk after each write operation. This approach ensures that all data transmitted over the serial connection is captured and logged persistently, making it suitable for applications such as logging sensor readings or monitoring device output in real-time.

```
1 import serial
2
3 class ReadLine:
4     def __init__(self, s):
5         self.buf = bytearray()
6         self.s = s
7
```

```
    def readline(self):
        i = self.buf.find(b"\n")
        if i >= 0:
            r = self.buf[:i+1]
            self.buf = self.buf[i+1:]
            return r.decode("utf-8")
        while True:
            i = max(1, min(2048, self.s.in_waiting))
            data = self.s.read(i)
            i = data.find(b"\n")
            if i >= 0:
                r = self.buf + data[:i+1]
                self.buf[0:] = data[i+1:]
                return r.decode("utf-8")
            else:
                self.buf.extend(data)

ser = serial.Serial('COM5', 9600)
rl = ReadLine(ser)
printed = False

with open("tof_readings.txt", "w") as file:
    printed = False
    while not printed:
        line = rl.readline()
        print("Received:", line)

        # Write the received line to the text file
        file.write(line + "\n")
        file.flush()
```

Listing 12: readings.py

5.3 3D Mapping and Point Cloud Generation

The software component of our 3D scanner project involves the processing and visualization of point cloud data obtained from the ToF sensor. We used numpy, matplotlib,scipy and mpl toolkt library for mapping.The process begins with loading the raw distance measurements from a text filet using numpy.loadtxt(). This file contains the distance readings obtained from the ToF sensor during the scanning process. Erroneous scan values, such as negative distances, are identified and set to zero to prevent inaccuracies in the 3D model. The delimiter 9999 is used in the text file to indicate the end of each z-height scan. These indices are located and stored to help segment the data into distinct z-layers. The raw data is then reshaped into a matrix where each row corresponds to one z-height, allowing for structured processing of the distance measurements.

Next, we perform distance calibration by offsetting the distances by the known distance from the scanner to the center of the turntable. This calibration ensures that the measurements are referenced from the turntable's center of rotation. Distances greater than a specified maximum and less than a minimum are set to NaN to exclude outliers. Values within a small threshold around zero are also set to NaN to remove noise close to the scanning origin. A theta matrix is created with angles corresponding to each column in the distance matrix, and a z-height array is generated with each row representing a specific height based on the known vertical increment. These matrices are used to convert polar coordinates (distance and angle) to Cartesian coordinates (x, y).

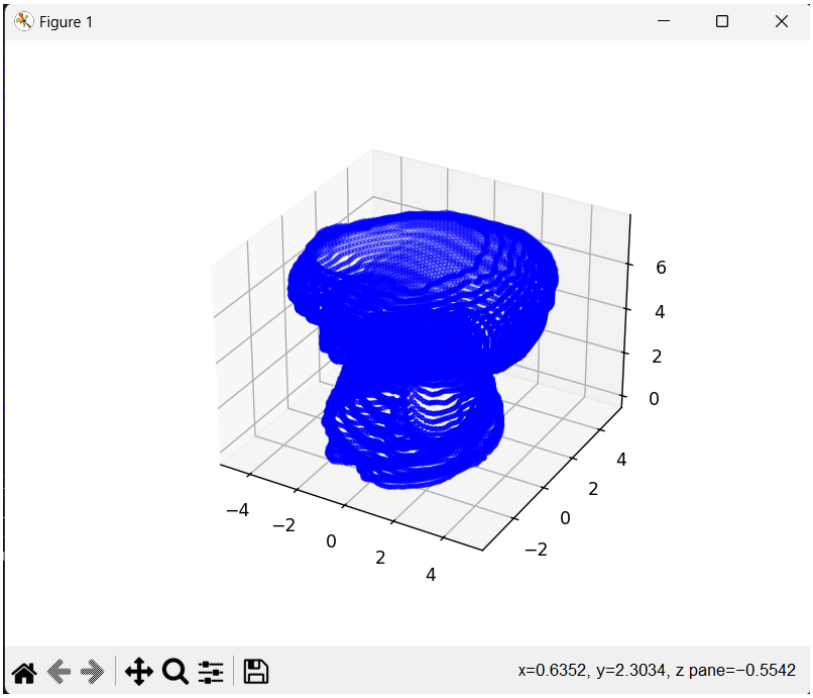


Figure 50: Example of 3D view of Scanned point cloud of a small statue

To handle missing data, NaN values in the x and y coordinates are replaced with the nearest valid neighbor value within the same height layer. This step ensures continuity in the data and prevents gaps in the 3D model. The data is then resampled to match the desired mesh resolution , reducing the data size and focusing on significant points. An average filter is applied to the x and y coordinates to smooth out noise. Convolution is used with a defined window size to average the values, resulting in a cleaner dataset.

The processed data is visualized using a 3D scatter plot with matplotlib's Axes3D, rendering the point cloud and allowing for verification of the data processing steps. The code includes a placeholder for exporting the processed point cloud to an STL file format, which can be implemented using external libraries like numpy-stl or meshio. Exporting to STL will enable the creation of a 3D model suitable for further analysis or 3D printing.

This detailed software workflow transforms the raw ToF sensor data into a structured and visualized 3D point cloud, providing a foundation for creating accurate 3D models of scanned objects. The combination of data preprocessing, calibration, conversion, and visualization ensures the reliability and accuracy of the final 3D representation.

```
1
2  import numpy as np
3  from scipy.signal import convolve2d
4  from mpl_toolkits.mplot3d import Axes3D
5  import matplotlib.pyplot as plt
6
7  # Processing variables
8  max_distance = 20  # Upper limit -- raw scan value only scanning "air"
9  min_distance = 0   # Lower limit -- raw scan value error: reporting
       negative reading
10 mid_thresh_upper = 0.5  # Offset radius threshold around 0
11 mid_thresh_lower = -mid_thresh_upper  # Offset radius threshold around
       0
12 window_size = 3  # Window size for average filter to clean up mesh
13 interp_res = 1  # Interpolation resolution, i.e. keep every interRes-th
       row
14 center_distance = 10.3  # [cm] - Distance from scanner to center of
       turntable
15 z_delta = 0.1
16 raw_data = np.loadtxt('farmer.txt')  # Load text file from SD Card
17 raw_data[raw_data < 0] = 0  # Remove erroneous scans from raw data
18 indices = np.where(raw_data == 9999)[0]  # Find indices of '9999'
       delimiter in text file, indicating end of z-height scan
19
20 # Arrange into matrix, where each row corresponds to one z-height
21 r = np.array([raw_data[indices[i - 1] + 1:indices[i]] for i in range(1,
       len(indices))])
22 r = center_distance - r  # Offset scan so that distance is with respect
       to turntable center of rotation
23 r[np.isnan(r)] = 0  # Replace NaN values with 0
24
25 # Remove scan values greater than maxDistance and less than minDistance
26 r[r > max_distance] = np.nan
27 r[r < min_distance] = np.nan
28
29 # Remove scan values around 0
30 mid_thresh_idx = np.logical_and(r > mid_thresh_lower, r <
       mid_thresh_upper)
31 r[mid_thresh_idx] = np.nan
32
33 # Create theta matrix with the same size as r -- each column in r
       corresponds to specific orientation
34 theta = np.linspace(0, 2 * np.pi, r.shape[1])
35 theta = np.tile(theta, (r.shape[0], 1))
36
37 # Create z-height array where each row corresponds to one z-height
38 z = np.arange(0, r.shape[0] * z_delta, z_delta).reshape(-1, 1)
39 z = np.tile(z, (1, r.shape[1]))
40
41 # Convert polar coordinates to Cartesian coordinates
42 x, y = np.cos(theta) * r, np.sin(theta) * r
```

```
43
44 # Replace NaN values in x, y with nearest neighbor at the same height
45 for i in range(1, x.shape[0]):
46     if np.all(np.isnan(x[i])):
47         x = x[:i]
48         y = y[:i]
49         z = z[:i]
50         break
51
52     latest_value_idx = np.where(~np.isnan(x[i]))[0][0]
53     latest_x, latest_y = x[i, latest_value_idx], y[i, latest_value_idx]
54     for j in range(x.shape[1]):
55         if not np.isnan(x[i, j]):
56             latest_x, latest_y = x[i, j], y[i, j]
57         else:
58             x[i, j], y[i, j] = latest_x, latest_y
59
60 # Resample array based on desired mesh resolution
61 interp_idx = np.arange(0, x.shape[0], interp_res)
62 x_interp, y_interp, z_interp = x[interp_idx, :], y[interp_idx, :], z[
    interp_idx, :]
63
64 # Smooth data to eliminate more noise
65 h = np.ones((1, window_size)) / window_size  # Define average filter
66 x_interp = np.pad(x_interp, ((0, 0), (0, window_size - 1)), mode='
    symmetric')  # Add symmetric duplicate padding along rows
67 y_interp = np.pad(y_interp, ((0, 0), (0, window_size - 1)), mode='
    symmetric')  # Add symmetric duplicate padding along rows
68 x_interp = convolve2d(x_interp, h, mode='valid')  # Filter x
69 y_interp = convolve2d(y_interp, h, mode='valid')  # Filter y
70
71 # Plot point cloud as a mesh to verify that processing is correct
72 fig = plt.figure()
73 ax = fig.add_subplot(111, projection='3d')
74 ax.scatter(x_interp, y_interp, z_interp, color='b', marker='.')
75 plt.show()
```

Listing 13: readings.py

6. APPENDIX 1: Daily Log Entries

6.1 Prepare Project Proposal and Background Research (February 1, 2024)

- Prepared a comprehensive project proposal outlining the goals, objectives, and scope of our 3D scanner project. The proposal included a detailed description of the project's significance and potential applications in various industries.

- Conducted extensive background research to understand the current state of 3D scanning technology. This included reading several relevant research papers to gain insights into existing methods, technologies, and challenges associated with 3D scanning.
- Watched numerous YouTube videos to visualize practical implementations and gather ideas for our design. These videos provided practical examples of 3D scanner builds and usage, helping us to identify key components and design considerations.

6.2 Stakeholder Mapping, Market Research, and Planning (February 8, 2024 - February 14, 2024)

- Stakeholder Map Creation - Developed a comprehensive stakeholder map to identify and understand the key stakeholders for our 3D scanner project. This included potential users, investors, suppliers, and industry experts. By mapping out the stakeholders, we aimed to ensure that our project would meet the diverse needs and expectations of all involved parties.
- User Observation - Conducted observations of potential users to gather insights into their needs, preferences, and pain points related to 3D scanning. This user-centric approach helped us to tailor our design to better suit the end-users.
- Market Research - Researched existing 3D scanner machines to understand the current market landscape. This included analyzing the features, capabilities, and pricing of various models. We identified cost-effective components and materials that would allow us to design a product with comparable or superior features at a lower cost.
- Cost Analysis - Compiled data on the costs and prices of existing 3D scanners. This analysis provided a benchmark for our product development, enabling us to strategically plan to make our product more affordable without compromising on quality.

6.3 Conceptual Design Comparison and Evaluation (February 20, 2024 - March 5, 2024)

- Conceptual Design Comparison - Evaluated multiple conceptual designs for our 3D scanner. Each design was assessed based on various criteria including durability, safety, ease of use, and manufacturing feasibility. This comparative analysis helped us identify the strengths and weaknesses of each design.
- Durability Considerations - Focused on the materials and structural integrity of each design to ensure long-term durability. This involved selecting materials that could withstand repeated use and environmental factors such as temperature and humidity.
- Safety Aspects - Considered the safety features of each design to ensure user safety during operation. This included assessing potential hazards and incorporating safety mechanisms to prevent accidents and injuries.
- Evaluated additional factors such as ease of assembly, maintenance requirements, functionality, simplicity and heat dissipation. These considerations were crucial for ensuring that the final design would be practical and user-friendly.

6.4 Component Selection for mechanical part (March 6, 2024 - March 15, 2024)

- Enclosure Arrangement - Designed the enclosure to house all essential components of the 3D scanner. The enclosure was planned to securely hold the stepper motors, motor drivers, microcontroller, and other electronics while providing easy access for maintenance and adjustments.
- Component Selection
  - Shafts and Couplers - Selected appropriate shafts and couplers to ensure smooth and reliable mechanical motion. These components were chosen based on their compatibility with the stepper motors and the mechanical design requirements.
  - Motors and Motor Drivers - Chose NEMA 17 stepper motors for their reliability and precision. Corresponding motor drivers were selected to efficiently control the motors and provide the necessary torque for the scanning process. So, we chose TB6600 stepper motor drivers to control the sepper motors in our product.
  - Distance Sensor - Decided to use the VL53L0X ToF sensor due to its high accuracy and precision. The VL53L0X was preferred over IR sensors as it simplifies distance measurement without needing complex trigonometric calculations.
- Design for Turntable and Connecting Parts - Created detailed designs for the turntable and other mechanical parts that connect the various components. The turntable design ensured stable rotation for accurate 3D scanning, while the connecting parts were engineered to maintain alignment and smooth linear motion along the z-axis.

6.5 Code Implementation (March 17, 2024 - March 25, 2024)

- Stepper Motor Control - Implemented initial code for controlling the stepper motors using the AccelStepper library. Despite several attempts, the library did not function as expected for our specific requirements.
- Direct Control via PWM and Step Counting - Developed and tested code to control the stepper motors using PWM signals and manual step counting. This approach provided greater control and customization for our specific scanning needs
- ToF Sensor Distance Readings - Implemented code to obtain distance measurements from the VL53L0X ToF sensor. Conducted research on potential sources of error in the readings and explored methods to minimize these errors.
- Sensor Calibration - Performed calibration of the ToF sensor to enhance accuracy. This involved adjusting the sensor's settings and validating the readings against known distances.
- Final Code Implementation - Integrated the final code for both distance measurement and stepper motor control. The code was refined to ensure synchronized operation of the motors and accurate distance readings during the scanning process.

6.6 Implemented data logging and visualization code (March 26, 2024 - March 31, 2024)

- Developed and tested a Python script to read distance measurements from the 3D scanner and write the values to a text file. This script ensures that all scanned data is accurately logged for further processing.

- Implemented another Python script to visualize a 3D graph of the scanned object. This visualization code processes the stored distance values, converting them into a 3D point cloud to create a visual representation of the scanned object. This step is crucial for verifying the accuracy and completeness of the scanning process and for subsequent analysis or refinement of the scanning technique.

6.7 Component selection for PCB and start to make schematic design (4th April, 2024 - 10th April, 2024)

The components were selected based on their functionality, compatibility, market availability, feasibility, robustness, ease of soldering, and cost effectiveness. Alternative options were considered to ensure optimal performance and cost effectiveness.

- Selected the ATmega328P microcontroller as the central control unit, chosen for its robust performance and versatility in handling project-specific tasks.
- Implemented the CH340 IC for serial UART communication, facilitating seamless data exchange between our scanner machine and PC.
- Integrated two 6-pin connectors to interface with stepper drivers, ensuring precise control and movement in the system.
- Utilized a 4-pin connector to connect the Time-of-Flight (ToF) sensor, enabling accurate distance measurement and object detection capabilities.
- Components were selected based on stringent criteria including functionality, compatibility, and practical considerations such as ease of integration and cost-efficiency.

6.8 Make the PCB Layout (14th April, 2024 - 18th April, 2024)

After schematic validation and verification, we proceeded to place components on the PCB. Our primary objective was to achieve a compact design footprint while ensuring optimal functionality and manufacturability.

- Key efforts were focused on
  - Strategically placing components to maximize use of available space.
  - Adhering strictly to design constraints and specifications throughout the placement process.
  - Implementing initial routing strategies aimed at minimizing signal interference and establishing efficient signal paths.
- Applied JL PCB rules meticulously to safeguard signal integrity and reduce potential noise.
- Continued prioritizing compactness while maintaining clear segregation between high-speed and low-speed signal traces.

6.9 Completion of PCB Design Validation and send for Manufacturing (19th April, 2024

Today marks the culmination of our PCB design journey with the finalization and thorough validation of our design. Following meticulous routing adhering to JLC PCB guidelines, we have successfully generated essential Gerber and NC drill files crucial for manufacturing.

6.10 SolidWorks Design Process for Product Components (20th April, 2024 - 26th April, 2024)

- Turntable Design - Utilizing SolidWorks, we meticulously designed a turntable mechanism tailored to our product's specifications. This included detailed modeling of the rotating platform, ensuring smooth operation and precise alignment with functional requirements.
- Shaft Connecting Part - We employed SolidWorks to create a robust shaft connecting component. This involved precise modeling and simulation to ensure compatibility and durability, crucial for maintaining operational integrity under varying conditions.
- Enclosure Design: The enclosure design phase in SolidWorks focused on creating a protective housing that encapsulates and secures internal components. Emphasis was placed on optimizing space utilization, ergonomic considerations, and aesthetic appeal while ensuring ease of assembly and serviceability.

6.11 Arrived the pcb and start to print enclosure parts (1st May, 2024 - 5th May, 2024)

Arrived the manufactured PCB and we print some parts for testing. Also we solder our PCB.

6.12 Documentation and Final Preparations (17th May, 2024 - 20th June, 2024)

Completed the documentation process including the final report for the 3D Scanner and also the Final Design Details Document with all the details we gathered and filtered for the documentation part through the entire process of developing the concept, design and other functionalities of the product '3D Scanner'

7. APPENDIX 2: Document Reviews

Cross check by the other team - Depth Camera

We have thoroughly reviewed the design documentation for the 3D Scanner Machine. The document includes a hierarchical schematic and detailed SolidWorks designs. It is comprehensive,covering all necessary aspects with detailed codes, graphs, pictures, and appendices. I confirm that these comments reflect my unbiased critique of the provided documentation.

we hereby declare that we have reviewed and approved the report of the project 3D Scanner done by Pathirana R.P.S. and Uduwaka S.S.

Index	Name	Signature
210071E	BOTHEJU W.G.W.	
210022G	ADIKARI A.A.S.D.	

8. APPENDIX 3: Previously used Arduino code

```
#include <Adafruit_VL53L0X.h>

int tenable = 6;
int tpulse = 3;
int tdir = 2;

int zenable = 7;
int zpulse = 5;
int zdir = 4;

#define pb_start 8


Adafruit_VL53L0X lox = Adafruit_VL53L0X();

void setup() {
  // put your setup code here, to run once:
  pinMode(tenable,OUTPUT);
  pinMode(tpulse,OUTPUT);
  pinMode(tdir,OUTPUT);

  pinMode(zenable,OUTPUT);
  pinMode(zpulse,OUTPUT);
  pinMode(zdir,OUTPUT);

  pinMode(pb_start,INPUT);

  Serial.begin(9600);

  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while (1);
  }

  digitalWrite(tenable,LOW);
  digitalWrite(zenable,LOW);
}

void loop() {
  // put your main code here, to run repeatedly:

  int vertDistance=10; //Total desired z-axis travel
  int noZSteps=200; //No of z-steps per rotation.  Distance = noSteps
     *0.05mm/step  20 default
  int zCounts=(200/1*vertDistance)/noZSteps; //Total number of zCounts
     until z-axis returns home
  //int thetaCounts=400;
  int thetaCounts=200;

```

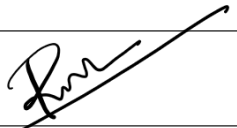
```
48    // Scan object
49    digitalWrite(zdir,LOW);
50    for (int j=0; j<zCounts; j++) //Rotate z axis loop
51    {
52      for (int i=0; i<thetaCounts; i++)   //Rotate theta motor for one
          revolution, read sensor and store
53      {
54        rotateMotor(tpulse, 1); //Rotate theta motor one step
55        //delay(200);
56        double senseDistance=0; //Reset senseDistanceVariable;
57        senseDistance=readingDistance(); //Read IR sensor, calculate
            distance
58        // writeToSD(senseDistance); //Write distance to SD
59        Serial.println(senseDistance);
60      }
61
62      rotateMotor(zpulse, noZSteps); //Move z carriage up one step
63      delay(1000);
64      Serial.println(9999); //Write dummy value to SD for later parsing
65    }
66
67    // Scan complete.  Rotate z-axis back to home and pause.
68    digitalWrite(zdir,HIGH);
69    delay(10);
70    for (int j=0; j<zCounts; j++)
71    {
72      rotateMotor(zpulse, noZSteps);
73      delay(10);
74    }
75
76    for (int k=0; k<3600; k++) //Pause for one hour (3600 seconds), i.e.
        freeze until power off because scan is complete.
77    {
78      delay(100);
79    }
80
81
82  }
83
84
85  // get readings from ToF
86
87  double readingDistance(){
88    VL53L0X_RangingMeasurementData_t measure;
89
90    lox.rangingTest(&measure, false);
91
92    if (measure.RangeStatus != 4) {  // phase failures have incorrect
        data
93      // Serial.print(F("Distance (mm): "));
94      double x =(measure.RangeMilliMeter);
95      return x;
```

```
  }
  else {
    return 28;
  }

  //delay(100);
}

void rotateMotor(int pinNo, int steps)
{

  for (int i=0; i<steps; i++)
  {
    digitalWrite(pinNo, LOW); //LOW to HIGH changes creates the
    delayMicroseconds(1000);
    digitalWrite(pinNo, HIGH); //"Rising Edge" so that the EasyDriver
       knows when to step.
    delayMicroseconds(1000);
    //delayMicroseconds(500); // Delay so that motor has time to move
    //delay(100); // Delay so that motor has time to move
  }
}
```

Listing 14: readings.py

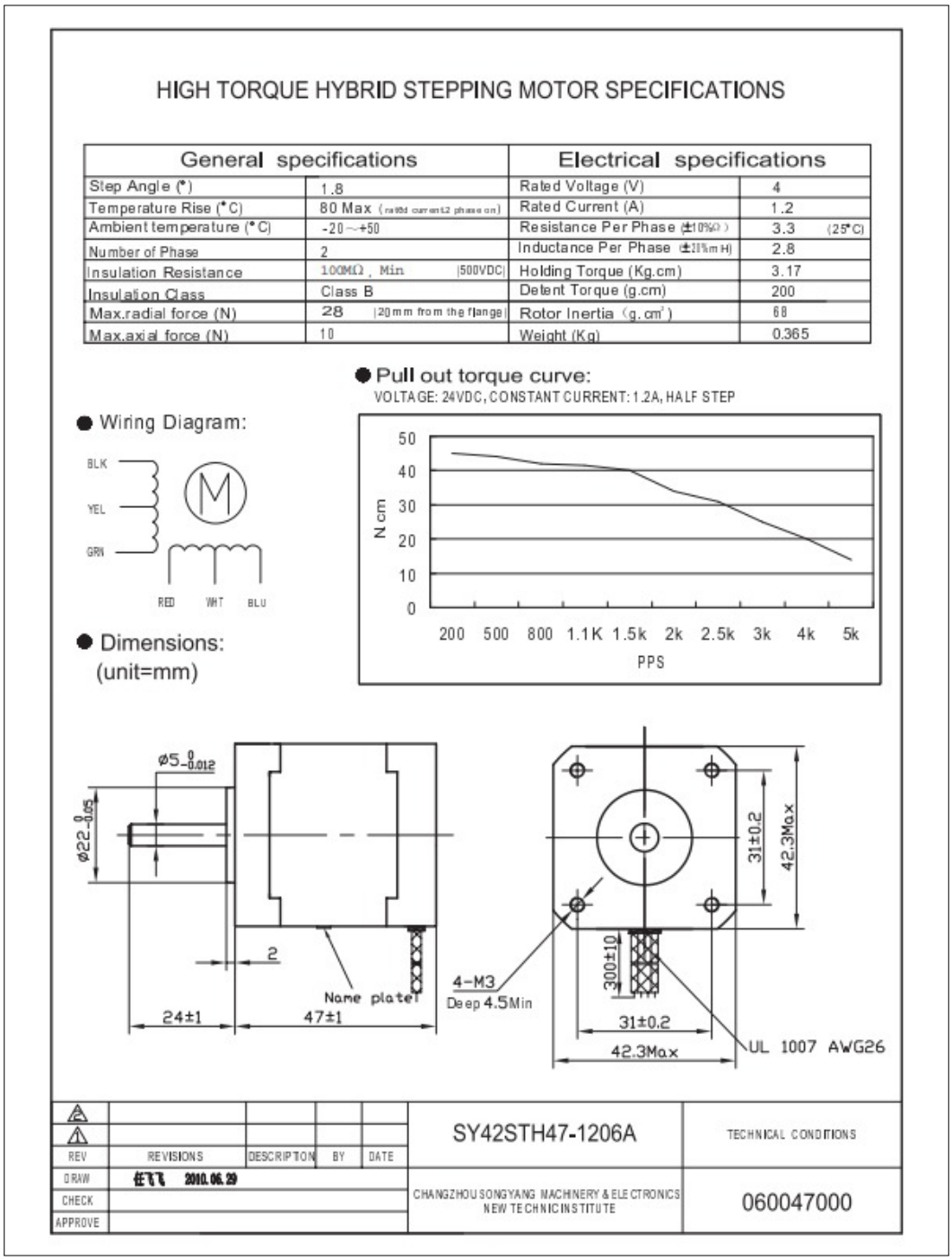
9. APPENDIX 4: Declaration

We declare that the original code did not include any Arduino-type coding or utilize Arduino libraries.

Index	Name	Signature
210451U	Pathirana R. P. S.	
210663V	Uduwaka S. S.	

10. APPENDIX 5: Useful Datasheets

10.0.1 Nemma 17 Stepper Motor



HIGH TORQUE HYBRID STEPPING MOTOR SPECIFICATIONS

General specifications		Electrical specifications	
Step Angle (°)	1.8	Rated Voltage (V)	4
Temperature Rise (°C)	80 Max (rated current,2 phase on)	Rated Current (A)	1.2
Ambient temperature (°C)	-20~+50	Resistance Per Phase (±10%Ω)	3.3 (25°C)
Number of Phase	2	Inductance Per Phase (±20%mH)	2.8
Insulation Resistance	100MΩ, Min (500VDC)	Holding Torque (Kg.cm)	3.17
Insulation Class	Class B	Detent Torque (g.cm)	200
Max.radial force (N)	28 (20mm from the flange)	Rotor Inertia (g.cm²)	68
Max.axial force (N)	10	Weight (Kg)	0.365

● Pull out torque curve:

VOLTAGE: 24VDC, CONSTANT CURRENT: 1.2A, HALF STEP

● Wiring Diagram:

● Dimensions:
(unit=mm)

REV	REVISIONS	DESCRIPTION	BY	DATE
DRAW	任飞飞 2010.06.29			
CHECK				
APPROVE				

SY42STH47-1206A

TECHNICAL CONDITIONS

CHANGZHOU SONGYANG MACHINERY & ELECTRONICS NEW TECHNIC INSTITUTE

060047000