GERMAN - MONGOLIAN INSTITUTE
FOR RESOURCES AND TECHNOLOGY

The present work was submitted to
the German-Mongolian Institute of Resources and Technology

Design and Development of Tariabot: A CNC-Based Farm Automation System

Bachelor's Thesis

By

KHUSELBAYAR Oyunbold

Study program: Mechatronic

Student ID: B2100317

1st Supervisor/Examiner: Odbileg Norovryenchin

2nd Supervisor/Examiner: Myagmarjav Bold

Ulaanbaatar/Nalaikh

2025

GERMAN - MONGOLIAN INSTITUTE FOR RESOURCES AND TECHNOLOGY

The present work was submitted to
the German-Mongolian Institute of Resources and Technology

Design and Development of Tariabot: A CNC-Based Farm Automation System

Bachelor's Thesis

By

KHUSELBAYAR Oyunbold

Study program: Odbileg Norovryenchin

Student ID: B2100317

1st Supervisor/Examiner: Odbileg Norovryenchin

2nd Supervisor/Examiner: Myagmarjav Bold

Ulaanbaatar/Nalaikh

2025

Statutory Declaration

Oyunbold, Khuselbayar	B2100317
Last Name, First Name	Student ID Number

I hereby affirm in lieu of an oath that I provided the submitted bachelor thesis

Design and Development of Tariabot: A CNC-Based Farm Automation System

I did not use any sources other than those stated. In case that the work is additionally submitted on a data medium, I declare that the written and the electronic form are completely identical. The work was not submitted in the same or similar form to any examination authority.

Ulaanbaatar/Nalaikh, 05/01/2025	
Place, Date	Signature

Abstract

This thesis presents the design, prototyping, and testing of Tariabot, a low-cost, semi-autonomous planting robot inspired by CNC principles. The objective was to create a scalable, indoor-optimized, and user-friendly farming system for apartment households and urban gardeners. Using open-source hardware and software tools, Tariabot automates essential agricultural tasks including seeding, watering, and environmental monitoring. The mechanical system is based on a Cartesian XYZ axis configuration, driven by NEMA 17 stepper motors and GT2 timing belts, while the Z-axis is operated using a rack-and-pinion mechanism. All core components, including structural parts and motor brackets, were 3D printed using PLA to reduce cost and fabrication complexity.

The system integrates an Arduino Uno and A4988 drivers to execute G-code-like commands received from a Node.js server. The web-based dashboard interface allows users to select planting positions visually, monitor temperature and humidity via DHT22 sensors, and control the watering system. Testing showed that Tariabot achieved ±5 mm axis repeatability. Watering output was consistent at approximately 350–400 mL per cycle. Finite element simulations validated the mechanical strength of key parts under applied loads, with safety factors above 70 for all components tested.

Tariabot fulfills its design goals with a budget-friendly approach and offers significant potential for smart farming applications in space-limited environments. The system demonstrates how mechatronic engineering, 3D printing, and web-based control can be combined into an accessible automation solution. Future developments may include camera integration, tool changers, full IoT support, and machine learning-based plant health analysis.

Acknowledgement

First, I would like to express my deepest gratitude to my primary supervisor, Professor Odbileg Norovryenchin, for his consistent guidance, encouragement, and valuable technical insights throughout the development of this thesis. His support and mentorship played a crucial role in shaping both the direction and quality of this project.

I also wish to sincerely thank my second supervisor, Mr. Myagmarjav Bold, for his thoughtful feedback, availability, and support throughout my research. His contribution helped me refine many aspects of my work, from system design.

Special thanks go to my friend Batmunkh, whose assistance with the 3D printing of mechanical components had a significant impact. His help greatly contributed to the successful hardware assembly and physical realization of Tariabot.

Lastly, I am deeply grateful to the German-Mongolian Institute for Resources and Technology (GMIT) for providing a learning environment rich in resources, knowledge, and motivation. The opportunity to carry out this thesis under such an institution has been a key milestone in my academic and professional development. To all those who have supported me on this journey, directly or indirectly, thank you.

Table of Contents

List of figures

List of table

1. Introduction

Over the past decade, achievements in robotics have significantly changed agriculture. Technologies such as GPS-guided equipment, automated irrigation systems, and AI-driven monitoring tools have increased efficiency and productivity in large-scale farming operations. However, these methods are only designed for large agricultural systems. Therefore, innovations such as household gardening are demanded for small-scale farming.

In recent years, Consumers have become more concerned about food security and accessibility because of pandemics and climate change. Recent events are the main reason for the growing demand for innovative and intelligent farming methods that allow individuals to grow food in limited spaces. Also, urbanization limits the arable land in cities, leading robotics to an effective and intelligent planting method on restricted land, including vertical farming, container farming, and a CNC-based farming system.

Due to food security and labor shortages, traditional farming methods are time-consuming, costly, and challenging. Hence, one solution for this is an automated farming system. So, the Global Farm Automation Market is expected to be worth around USD 29,416.8 million by 2033, up from USD 6,225.4 million in 2023, due to urbanization and advancement in robotics. (1)

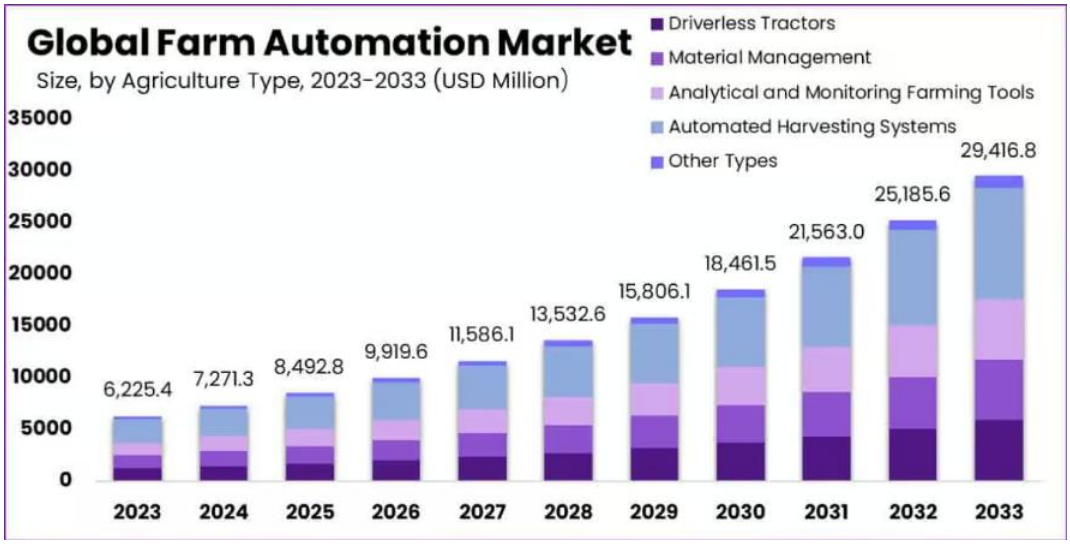


Figure 1: Global Farm Automation System market systems worth chart

Engineers have been developing various types of automation systems, including CNC-based systems for small-scale farming, to fulfill challenges in agriculture. This thesis presents the development and design of a CNC-based automation system for basic agricultural tasks such as watering plants, planting seeds, and monitoring temperature and humidity. This system, which we have named Tariabot, is a compact and user-friendly machine easily operated by individuals with minimal technical knowledge. As I wrote, large-scale farming is cost-effective, but planting individually offers significant advantages, including reduced labor costs, less waste, smaller operation space, and food stability. Therefore, some startup companies noticed this and developed systems such as Farmbot.

Farmbot is a CNC planting machine for the backyard with its own monitoring and handling system. While researching Farmbot, three significant disadvantages were recognized: the number of consumers is limited to a small number, consumers have to have basic knowledge of electronics, and it is prone to corrosion due to outdoor operation. (2) Therefore, I decided to develop and redesign it to suit apartment households and to make it non-corrosive, which is Tariabot. Although it may not be possible to fully solve large-scale food production issues, small-scale CNC-based systems can help supply households with fresh vegetables such as cucumbers, tomatoes, and peppers.

Furthermore, the potential of systems like Tariabot and Farmbot extends beyond household food production. These systems could evolve into fully autonomous smart gardens with the future integration of IoT technologies and data-driven growth optimization. They could also serve educational purposes, helping students and families learn about plant science, robotics, and sustainability, bridging the knowledge gap between technology and nature.

1.1 Background

Agriculture is one of the key factors in human civilization, dating back to around 11,000 years ago during the Neolithic Revolution. The earliest known example is the cultivation of rye at Tell Abu Hureyra around 13,000 years ago. (Encyclopædia Britannica, 2025; Wikipedia, 2025). (3) These agricultural achievements marked the starting point of the development of civilization, population growth, and food accessibility. During the Industrial Revolution, traditional tools were replaced with mechanized equipment, such as the mechanical reaper and the modern tractor. Also, the chemical industry introduced the Green Revolution and irrigation techniques, significantly increasing labor efficiency and large-scale food production. (4)

Recently, agriculture has taken a new step with automated farming. Technological innovations such as GPS-guided tractors, computerized irrigation systems, and AI-powered crop monitoring have upgraded cultivation's efficiency by reducing labor dependency and increasing yield precision. However, these systems are mainly focused on large-scale agricultural fields, and economically, the threshold investment is expensive, creating a gap between large-scale and household farmers. Hence, technologies like vertical farming, container farming, and CNC-based fully automated systems are being used as solutions to planting in urban environments and backyard farming.

FarmBot is a classic example of small-scale CNC-based automated farming. It fills the gap between large-scale and small-scale farming by automating planting and watering through computer monitoring. FarmBot is shipping worldwide and already has consumers.

1.2 Problem statement

Modern agriculture has grown significantly in large-scale farming. However, small-scale farming needs more improvement, especially in urban areas. Urbanization has been limiting arable land, and urban households want a green environment more and more nowadays. Systems like Farmbot, the only known CNC-based farming machine, have several problems, including their high cost, susceptibility to corrosion, Limited choice of operation area, and need for technical knowledge.

Furthermore, people are more concerned about food security and accessibility, so individuals and households are increasingly interested in growing food independently. However, few affordable and user-friendly solutions are on the market, especially for apartment households.

A solution to this gap is an accessible, scalable, intelligent farming system that supports household-level food production with minimal technical knowledge and space. This thesis proposes Tariabot, a redesigned CNC-based system that addresses these challenges and enables sustainable, low-maintenance, indoor urban farming.

1.3 Objectives

The main objective of this thesis is to design and develop a CNC-based automated farming system, Tariabot, which is more suitable for small-scale farming in backyards and urban spaces. This system aims to provide an apartment household with green energy and vegetable necessities. The specific objectives of this research are:

- To design Tariabot as a scalable and flexible. Green energy is a classic interior design style, so it has to fit a suitable corner.
- To develop a user-friendly control system that enables easy operation without requiring technical knowledge.
- To find the cheapest way to assemble a CNC farming machine
- To find a way to automate basic agricultural tasks such as seeding, watering, and environmental monitoring.
- To design Tariabot as a suitable for green interior of the household

2. Literature review

2.1 Mechanical Design and Part System of Farmbot

FarmBot's mechanical design is inspired by CNC machine architecture, focusing on low-load precision movements for farming tasks. Its structure is divided into several main components: the XYZ movement system**,** the seeding mechanism, the watering system**,** and the supporting frame and drive components**.**

The XYZ axes are based on a Cartesian coordinate system, where each axis moves linearly along one dimension: the X-axis provides forward and backward motion, the Y-axis moves the gantry left and right, and the Z-axis allows vertical tool control. This configuration enables FarmBot to position its tools accurately anywhere within the growing area. Motion control is achieved using NEMA 17 stepper motors**,** GT2 timing belts**,** and pulleys for the X and Y axes, while the Z-axis often employs a lead screw for finer control (5).

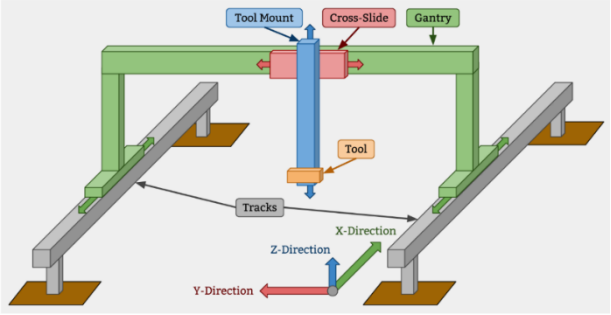


Figure 2: Farmbot movement system design

The seeding system in FarmBot uses a vacuum-based seed injector. A vacuum pump creates negative pressure through a small nozzle, enabling the tool head to pick up individual seeds. After the tool head moves to the specified planting location, the vacuum is turned off, releasing the seed into a small hole in the soil. The vacuum seed injector offers several advantages: it is simple, lightweight, and can be adapted to a wide range of seed sizes by changing the nozzle tip (6).

Figure 3: Watering tool of Farmbot

The watering system is designed to deliver precise amounts of water directly to each plant location, conserving water and improving plant health. FarmBot uses a watering nozzle tool connected to a pressurized water supply. When watering is required, the tool head moves to the plant's location, lowers the nozzle close to the soil, and activates a solenoid valve or pump to release a controlled quantity of water. This targeted delivery method minimizes water waste compared to traditional sprinkler systems and helps maintain consistent soil moisture levels critical for plant growth (6).

Figure 3: Watering tool of Farmbot

The combination of precise seeding and watering systems ensures that FarmBot can automate two of the most labor-intensive gardening tasks, planting and daily watering, with minimal human participation. Some FarmBot models also integrate automatic tool changers, allowing the robot to automatically switch between seeding, watering, and soil sensors without manual assistance. However, such features are more complex and costly (6).

The bot is equipped with other specialized tools to improve gardening automation. The soil sensor tool can measure soil electrical conductivity and moisture levels, providing critical data for watering decisions. A weeding tool**,** often designed as a small blade, disturbs and destroys unwanted plants by mechanically disrupting their root systems. Furthermore, some versions of FarmBot include a camera tool mounted on the gantry, enabling the system to perform plant recognition, monitor growth, and detect weeds. These additional tools expand FarmBot's functionality beyond basic planting and watering, moving toward a more autonomous and intelligent agricultural assistant (6).

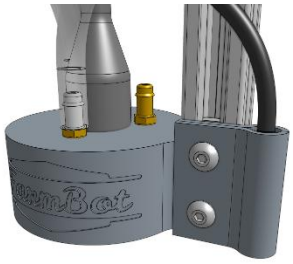


Figure 4: Observation camera of Farmbot

Figure 5: Rotary tool of Farmbot

Figure 6: Soil sensor of Farmbot

2.2 CNC-Based Automation Systems

CNC-based automated systems generally share a similar structural foundation, whether they are CNC milling machines, CNC laser engravers, or CNC plotters. However, their mechanical requirements and design differ significantly depending on their function. CNC milling machines, for example, typically require a rigid and heavy frame to resist cutting forces, vibrations, and tool deflections generated during material removal processes. This structural rigidity is critical to ensure dimensional accuracy and surface quality, especially when working with metals or harder materials. (7)

CNC plotters operate under minimal mechanical resistance, as they only involve the movement of a lightweight pen, marker, or tool head across a surface. The primary forces at play are due to inertia and gravity, which make the system less demanding in terms of structural strength. Therefore, CNC plotters can be constructed using lighter materials and simpler frame designs, often constructed with aluminum, 3D-printed brackets, or even wooden components in DIY builds. (8)

A key mechanical design difference lies in the Y-axis configuration. In milling machines, the Y-axis typically relies on a moving table, where the bed shifts under the cutting head. This setup, while effective for milling, is not ideal for lightweight robotic systems like Tariabot, especially due to its complexity and added mass. Instead, a CNC plotter-style design is more suitable, where the tool head moves in both X and Y directions, and the bed remains stationary. This minimizes moving mass, simplifies motor load distribution, and fits better with the compact, low-cost, and lightweight objectives of the Tariabot project.

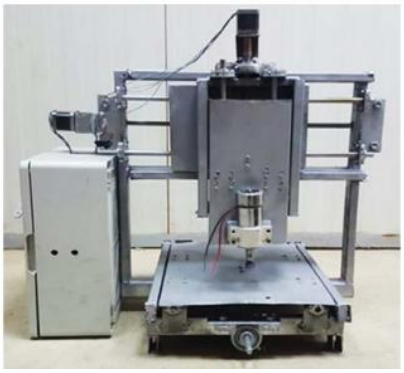


Figure 8: CNC milling machine

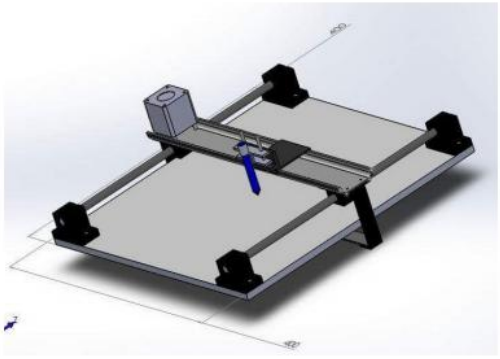


Figure 7: CNC plotter machine

2.3 Electrical Components

Motor Drivers

About similarities, the electrical design is similar in both cases. Using Arduino and the Arduino CNC shield is the most budget-friendly and best option. Many articles have used this setup to build CNC machines during my research. CNC milling and plotter machines commonly have three axes. These motors are controlled using stepper motor drivers such as A4988 and DRV8825, which have microstepping ability. The A4988 is much cheaper than the DRV8825, but the A4988 allows 1/16 micro-stepping, while the DRV8825 supports 1/32 micro-stepping, which is more precise. Depending on the task, each one is a good option. (9) (10)

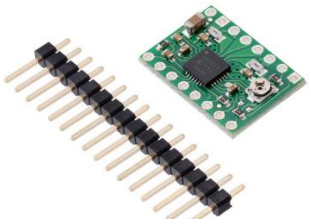


Figure 9: A4988 driver

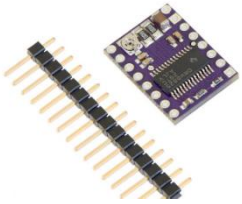


Figure 10: DRV8825 driver

Available motor options

Regarding the motor, NEMA 17 is commonly chosen by customers because it balances torque, size, and cost. The open-loop controller controls it, commanding the motor's position to move and hold at one of these steps without any feedback sensor. NEMA 17 stepper motors have 1.8° step angles and 200 steps per revolution. It can be increased up to 6400 microsteps. This motor produces 20 to 50 N · cm torque and operates at 12V to 24V DC. These properties make it more suitable for lightweight CNC machines like 3d printers and CNC plotter machines. The motors are usually connected through GT2 belts and pulleys or lead screws, depending on the design of the frame. Arduino-based controllers and GRBL firmware compatibility allow straightforward integration with low-cost electronics. (11)

Figure 11: NEMA 17 stepper motor

Microcontroller and CNC shield

Arduino UNO is the most suitable and budget-friendly microcontroller for a mini CNC machine project. It is based on the ATmega328P microcontroller, a low-power complementary Metal-oxide semiconductor with an 8-bit controller. Because of its simplicity and affordability, Arduino is one of the most widely used controllers in low-cost CNC machine projects. It offers 14 digital I/O pins, 6 analog inputs, and operates at 16MHz, making it suitable for interpreting G-code commands. USB connectivity simplifies uploading code and real-time control, and power consumption is comparatively low.

The Arduino CNC Shield is designed to be mounted on top of the Arduino Uno, allowing for minimal hardware design for small-scale CNC systems. It is intended to work with A4988 and DRV8825 drivers, providing dedicated slots for up to 4 axes of control (X, Y, Z, and A). It minimizes CNC Shield setup time and increases reliability in DIY CNC builds. The shield includes jumper pins for micro-stepping settings, limit switches, spindles, and laser control. Their system can be operated with standard G-code instructions and provides accurate three-axis movement, confirming the CNC shield's help as a reliable interface for educational and prototyping applications. (12)

Figure 13: Arduino UNO

Figure 12: Arduino CNC Shield

Power supply and sensors

A power supply is one of the key factors in operating an electrical system. The Arduino Uno can be powered through USB or a 9- 12V adapter, but controlling a stepper motor requires a higher current and voltage supply. NEMA 17 motors typically perform at 12V or 24V; higher voltage gives higher torque and faster response. (11) The CNC shield provides power for the stepper drivers from the external power input terminal, not Arduino (12). Electrical safety is paramount; setting the VREF (Voltage reference) in the driver is essential before connecting motors to the shield. The kind of driver you are using is necessary. For example for A4988 $current\ limit = VREF * 0.25$, while DRV8825 $current\ limit = VREF * 0.2$ by using potentiometer on the driver and multimeter, it is possible to configure. If you did not configure the VREF, the motor may be damaged.

Building the Tariabot monitoring system is an important part. To create a pleasant environment for the plant, it must at least accurately monitor soil moisture, temperature, and humidity. Functionality, budget-friendliness, and durability are considered during the selection process. No matter where the CNC machine operates, water may damage sensors, so waterproof or water-resistant sensors are recommended to ensure long-term reliability. Combined sensors that measure both temperature and humidity are efficient for small-scale farming. The DHT22 is one of those sensors. It is a widely used digital sensor with excellent accuracy, but it's not waterproof. When used in a moist environment, like outdoors or a plant house, DHT22 may be damaged in the long term. (13) Several waterproof sensor options exist, such as SHT20 or BME280 (with Protective casings). Their sensors offer better moisture resistance and are compatible with a microcontroller. (14)

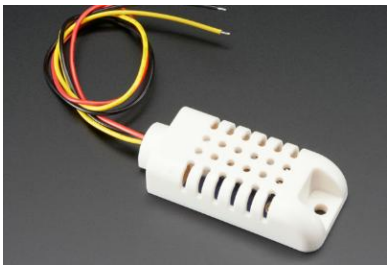


Figure 14: DHT22 temperature and moisture sensor

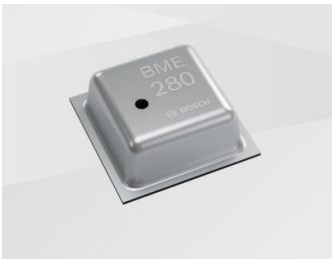


Figure 15:BME280 waterproof temperature and moisture sensor

Soil moisture is a key parameter for crop health. By measuring soil moisture, CNC machines can determine whether plants are thirsty. The SENO0193 soil moisture sensor is a great option. It works with Arduino and is dipped to the recommended depth into the soil. Avoiding direct electrical contact with the soil measures changes in soil dielectric properties. (15)

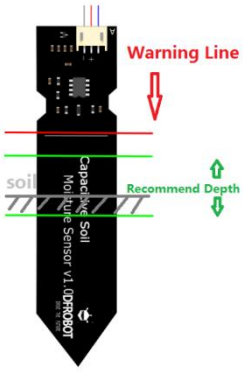


Figure 16: Soil moisture sensor

Available Pump options

A CNC-based farming system like Tariabot requires efficient water pumping for planting. For systems using a microcontroller like the Arduino Uno, voltage, current draw, and control interface are essential for pump selection. The amount of water a plant needs varies. On average, indoor plants require about 200-300 milliliters per plant, whereas outdoor vegetable plants may need up to 2 liters daily, especially during hot weather conditions. Integrating soil sensor data with real-time pump control ensures efficient water usage and reduces waste.

The RS-360S model is a mini submersible DC pump operating at 4V-12V. It is widely used in small-scale indoor plant-watering systems. These pumps offer a small size and submersibility and work with Arduino through transistors like TIP120 or relay modules. The flow rate ranges from 80 to 120 liters per hour. It is suitable for precise water injection for a single or a few plants. (16)

If consumers want to use it for bigger-scale farming, the JENENSERIES 12V DC pump is more suitable. These pumps provide higher flow rates, approximately 240 liters per hour, and increased pressure, making them ideal for systems requiring long tubing or multiple injectors. They are not submersible but can be safely installed to avoid water. Relay modules or MOSFETs are mainly used to control them. (17)

Figure 17: JENENSERIES 12V DC pump

Figure 18: RS-360S pump

2.4 Programming tools

G-code

One of the most significant parts of accomplishing Tariabot is programming. Coding precise software and programming logic helps to work with motors, sensors, and water pumps accurately. The most essential part of motion control is the G coding section, a standard programming language for CNC systems that defines movements like rapid positioning (G00) and controlled linear motion (G01). GBRL (G-Code Reference Block Library) works with G-code and converts it into stepper motor signals through drivers. Software tools such as Universal Gcode Sender (UGS)**,** BCNC**,** and LaserGRBL often upload G-code to Arduino. These tools help users to visualize movement and debug operations, allowing them to perform tasks like seed placement and watering.

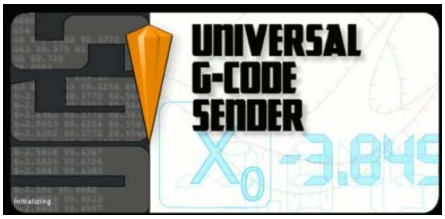


Figure 19: G-code sender

Figure 21: BCNC

Figure 20: LaserGBRL

Software and Communication Technologies

Programming is commonly done using the Arduino IDE, which is written in C/C++, for sensor control, pump activation, and general automation**.** Arduino simplifies coding and communication between electrical parts, making it a preferred option for DIY mini projects like Tariabot. In more advanced implementations, especially where wireless control and data saving are required, ESP32 microcontrollers and Python programming can be used. Python enables integration with IoT platforms and allows the collection and analysis of sensor data. (18)

To make Tariabot more interactive and user-friendly, a monitoring and control interface can be developed as a web-based dashboard. These interfaces allow users to:

- Start or stop operations remotely
- Monitor live sensor readings
- Set watering schedules
- View alerts (19)

On the other hand, a web dashboard can be built using Node.js, allowing Tariabot to send data to and receive commands from the cloud that interact with the Arduino or ESP32 via Wi-Fi or Bluetooth. (20)

Integrating a web-based interface into Tariabot is vital for designing a user-friendly system. Users can use a web-based interface to monitor sensor data, send commands, and manage gardening.

Node.js is an open-source JavaScript runtime environment that works on Windows, Linux, and macOS and executes JavaScript code outside a browser (21). Its non-blocking, event-driven architecture makes it highly efficient for developing fast, scalable network applications, particularly those that handle real-time sensor updates and user interactions (21) (22).

To accomplish my thesis project, Node.js is suitable for building a small server that manages real-time communication between the Tariabot and the user. Express.js is commonly combined with Node.js to simplify an HTTP server, while Socket.IO allows WebSocket-based real-time two-way communication, which is essential for monitoring sensor data. (23)

Figure 23: Node.js

Figure 22: Socket.IO

MySQL is one of the most widely used relational database management systems (RDBMS), known for its performance, reliability, and scalability. (24) It is well-suited for applications like Tariabot, where sensor data needs to be recorded.

Using MySQL, Tariabot can store key parameters such as soil moisture, temperature, and planting coordinates. MySQL also ensures data integrity, which is important for automated systems. Node.js acts as a middleware server that receives and updates from Tariabot, processes the data, and stores it in the MySQL database. Meanwhile, the web interface queries MySQL to retrieve live data, offering users control capabilities through a dashboard. (24) By combining Node.js and MySQL, Tariabot will become a user-friendly, scalable, and efficient platform for a small-scale automated farming system.

Figure 24: MySQL

3. Calculation

Motor Sizing Calculation

Assumptions:

- Gantry + Tool Mass: 2 kg
- Friction coefficient (LM8UU bearings): μ ≈ 0.01 [31]
- GT2 pulley radius:
    - Pitch diameter ≈ 12 mm → radius ≈ 6 mm = 0.006 m
- Safety factor: 2 (standard in engineering design)

1. Calculate Force Required to Move Load:

Normal force = Mass × Gravity

$$\mathbf{F_N = 2kg \times 9.\frac{81m}{s^2} = 19.62N}$$

Friction force = Friction coefficient × Normal force

$$\mathbf{F_{friction} = 0.01 \times 19.62N = 0.1962N}$$

2. Calculate Required Torque at Motor:

Torque = Force × Radius

$$\mathbf{T = 0.1962N \times 0.006m = 0.0011772\left(\frac{N}{m}\right) = 0.11772\left(\frac{N}{cm}\right)}$$

Apply Safety Factor 2:

$$\mathbf{T_{required} = 0.11772\left(\frac{N}{cm}\right) \times 2 = 0.23544\left(\frac{N}{cm}\right)}$$

Based on the mechanical load and friction assumptions, the calculated required torque for Tariabot is approximately 0.24 N·cm. The selected NEMA 17 stepper motor, providing around 40 N·cm holding torque, ensures highly reliable movement under load**.** (11)

Water Pump Sizing Calculation

The water pump selected is a 12V mini submersible pump (RS-360S), typically rated at 80–120 L/h**.** (16)

Assumptions:

- Each plant needs 300 mL (0.3 L) per watering.
- Assume 10 plants → total water needed per cycle:

$$\text{Water Needed} = 0.3\text{L} \times 10 = 3\text{L}$$

- Pump flow rate: 80 L/h (minimum rated value).

Convert flow rate: $80(\frac{\text{L}}{h}) = 1.33(\frac{\text{L}}{\text{min}})$

1. Calculate Time Needed to Water All Plants:

$$\text{Time} = \frac{3\text{L}}{1.33(\frac{\text{L}}{\text{min}})} = 2.26\text{minutes}$$

Thus, it takes about 2–3 minutes to water all plants fully.The water delivery requirement of 3 liters per cycle can be achieved within approximately 2.3 minutes using the selected RS-360S mini submersible pump, confirming its suitability for Tariabot's irrigation system.

4. Methodology and prototyping

4.1 Assembling hardware

Collecting mechanical and electrical parts

To Prototype Tariabot, the first task was to design the main frame and movement system. My main aim was to create a scalable, modular, and user-friendly CNC-based automation system for apartment and small-scale backyard farming. According to practical research, the structure of a CNC plotter machine was suitable for this purpose due to its simplicity, precision, and low structural load requirements.

During the research phase, an open-source 3D model of a CNC plotter machine was found, which closely matched the functional and mechanical needs of Tariabot, as shown in Figure 26. The model provided a complete design guide, including a model number of parts (bearings, screws, belts) and STL files for all custom 3D printed parts. This model significantly accelerated the initial prototyping stage, allowing focus on system adaptation rather than designing the frame from scratch.

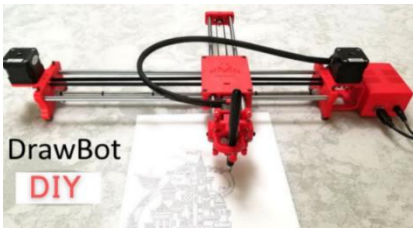


Figure 25: CNC Plotter [27]

The 3D printed parts sourced from the CNC plotter machine project and predesigned parts from Cults3d.com were selectively used to fit Tariabot's specific functions, such as movement along the XYZ axes and mounting gardening tools like watering nozzles. Each part was 3D printed using PLA filament with FDM (Fused Deposition Modeling) technology, providing a balance between strength, ease of manufacturing, and cost-effectiveness. Regarding the Z axis, I used a rack and pinion mechanism for a NEMA 17 motor (25)(Figure 27) and a GT2 timing belt-based movement system for the XY axis mechanism.

The movement along each axis moves through a closed-loop timing belt connected to stepper motors and idler bearings. As shown in Figure 28, when the motors rotate, the belt tension drives the gantry structure forward, backward, left, or right, depending on the motor direction. This configuration allows for fast and precise bidirectional control. The GT2 belts have a 2 mm tooth pitch, providing good positional resolution when paired with 1.8° stepper motors and microstepping drivers. (26)

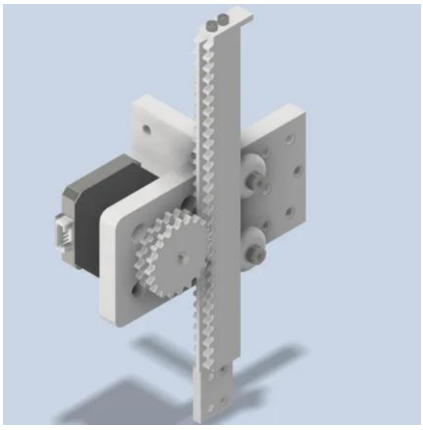


Figure 27: Z axis mechanism rack and pinion [28]

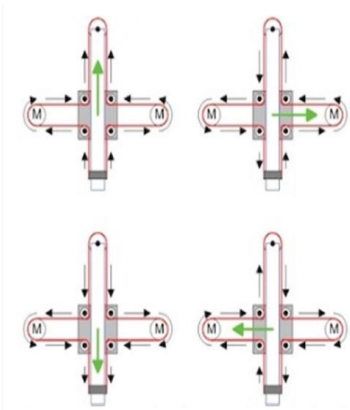
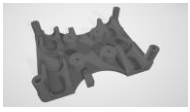
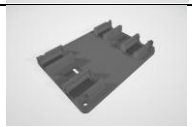
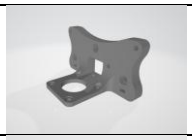
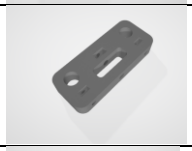
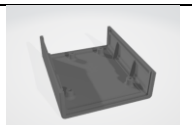
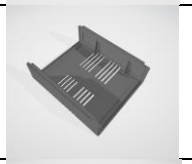
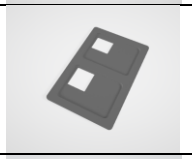
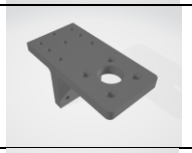
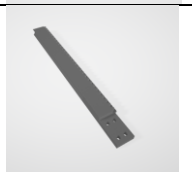
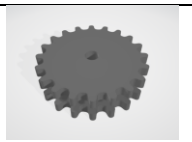
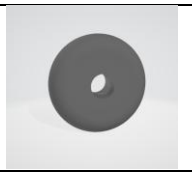


Figure 26: GT2 timing belt-based movement system [27]

To assemble the prototype of Tariabot, several custom 3D-printed components were printed based on the CNC plotter model and the actuator design. These parts were important for constructing the movement system, mounting components, and supporting gardening tools. Table 1 lists the main 3D printed parts used in the Tariabot system, their functions, materials, and important manufacturing notes.

No	Image	Part name	Quantity	Material	Note
1.		Bottom of the XY clamshell	1	PLA	Pulley, linear bearing, and linear rod are located on it and are designed for the movement of the X axis.
2.		Pulley for GT2 belt	4	PLA	Helps to change the direction of the GT2 belt.
3.		Top of the XY clamshell	1	PLA	The linear bearing and rod are located on it and are designed for the Y-axis's movement.

4.		X-axis support	2	PLA	NEMA 17 and linear are mounted on it.
5.		Back of the Y-axis support	1	PLA	Helps to support the Y-axis.
6.		Front of the Y-axis support	1	PLA	It helps support the Y-axis and Z-axis cover, which is mounted on it.
7.		Bottom of Arduino and CNC shield protective shield	1	PLA	Arduino is mounted on it.
8.		Top of Arduino and CNC shield protective shield	1	PLA	It covers the electrical parts, like Arduino and CNC shields.
9.		Back of Arduino and CNC shield protective shield	1	PLA	Electrical wires go through the hole in it.
10.		The front of the Arduino and CNC shield protective shield	1	PLA	Power supply connects through the hole.
11.		Z-axis support	1	PLA	The Z-axis motor is mounted on it.
12.		Rack of the Z-axis	1	PLA	Responsible for the movement of the Z-axis, and the nozzle is connected to the bottom of it.
13.		Pinion of the Z-axis	1	PLA	Transmits the rotation of the motor to the rack. Contributes to converting rotation into linear motion.
14.		Support of Rack	2	PLA	Prevent the rack from falling.

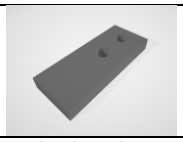
15.		Limiting part of the Rack	1	PLA	Limits Rack.

Table 1: 3d printed parts

Several important electronic and mechanical parts were purchased to complete the Tariabot prototype. Almost all components were sourced through the Taobao online marketplace to maintain cost-effectiveness and ensure part compatibility. The purchased parts include stepper motors, drivers, control boards, timing belts, and bearings. Table 2 summarizes the key purchased components, their specifications, quantities, sources, and intended functions within the Tariabot system.

No	Image	Part name	Quantity	Note	Cost
1.		GT2 belt's Pulley	2	Connects the rotation of motor to the belt	0.66$
2.		GT2 belt 6mm width, 2000mm long	1	Transmits Motion	3.41$
3.		M8 nuts	4	Fix the Threaded rod into the X-axis support	1.35$
4.		624ZZ bearing	5	Helps to move the pulley more smoothly	2.75$
5.		LM8UU linear bearing	8	Helps to move the linear rod more smoothly	7.42$
6.		12V 2A Power supply	1	Supply power to the CNC shield	1.65$
7.		M8 Linear rod 500mm	4	Helps to slide the clamshell freely	6.05$

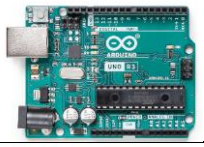
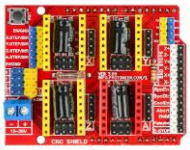
8.		M8 threaded rod 500mm	1	Fix the size of the X-axis	2$
9.		Arduino Uno	1	Reads G-code and sensor data, and sends commands to motors.	27.6 $
10.		CNC shield	1	Connects Arduino to stepper drivers and motors easily	9.99$
11.		A4988 Stepper motor driver	4	Controls the NEMA 17 motors by converting Arduino signals into motor steps	22.77$
12.		Nema 17 Stepper motor	3	Moves the CNC machine along X, Y, and Z axes	38.97$
				Total cost	124.62$

Table 2: mechanical and electrical parts

Testing and Validation of Electronics Components

Before assembling the Tariabot, the electronic components should be individually tested to ensure their correct operation. This testing phase focused on checking the Arduino Uno, CNC Shield, A4988 stepper motor drivers, and NEMA 17 stepper motors. The Arduino Uno has to be programmed with Arduino Uno C++ code. Connection to the CNC Shield was checked, ensuring all step and direction signal pins were correctly connected. Each A4988 driver was installed on the CNC Shield, and the VREF voltage (current limit) was adjusted using a multimeter to protect the stepper motors from overcurrent damage. For A4988, $current\ limit = VREF * 2.5$ is used to calculate VREF. According to the datasheet of the

Nema 17 stepper motor, the Current limit is around 1.5A, so VREF is around 0.6V. To measure VREF using the multimeter, I checked the voltage between GND and calibrating screw. If the motor driver is not calibrated, the current overload may damage the motor.

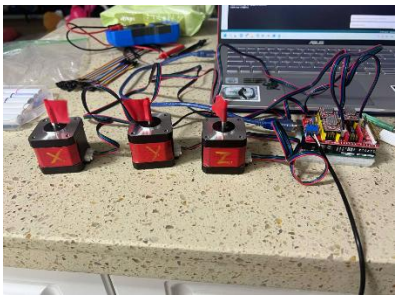


Figure 28: Calibration process of motor

Figure 29: Calibrated A4988 driver

The NEMA 17 stepper motors were wired correctly to the CNC Shield motor outputs. Initial movement commands were sent through serial commands to the Arduino. The motors successfully rotated, confirming that signal transmission, driver function, and motor coil wiring were correctly set. This testing confirmed that all core electronic components functioned properly, ensuring a reliable foundation for building the complete Tariabot system.

Assembling electrical and mechanical part

First, the linear and threaded rods were cut to the required lengths. 2 were cut 450mm long for the x axis, and the rest were cut at least 360mm long for the y axis movement. An extra 10 mm is needed to tighten the belt. The threaded rod was cut 470mm long at least, because to fix the support of the X-axis, it has to be tightened with the bolt. (Figure 31)

Figure 30: Cutted linear rods

The bearings and pulleys were assembled onto the clamshell structure in the next step. The bottom part of the clamshell was printed by fixing the bearings and positioning the pulley. Additionally, four linear bearings were installed on the top part of the clamshell to guide the Y-axis motion. The pulley was put between the bearings to ensure smooth belt operation. All pulleys were fixed in place using M3 screws in the desired hole in the middle of the clamshell.

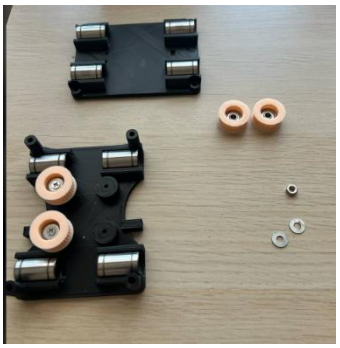


Figure 31: Fixed bearings and pulleys

The cut linear rods were slightly inserted through the linear bearings into the base structure. The top and bottom parts of the clamshell were then assembled and securely fastened together using M4 screws at the four corners of the clamshell. The two X-axis supports were connected to the bottom clamshell through the linear rods and tightened using the threaded rod and fixed with screws at the bottom of each support, as illustrated in Figure 33.

Figure 32: X-axis support's connection

Following this, the back and front Y-axis supports were mounted onto the top side of the clamshell using linear rods, and tightened with screws at the bottom of each support to ensure a stable Y-axis frame structure. Back of the Y-axis should deliver rotation of the belt, so fifth bearing is located inside of it and tightened with M4 screws. (Figure 34)

Figure 33: Back of the Y-axis

After assembling the supports, the pulleys for the X and Y axis motors were installed and fixed. The stepper motor for the X-axis movement was mounted securely onto the X-axis support part, completing the motor installation for the movement system. (Figure 35)

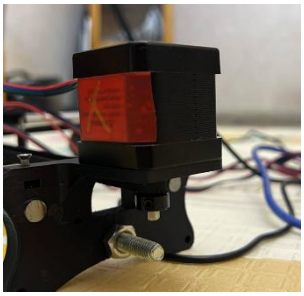


Figure 34: Fixed X-axis motor

In the next stage, the GT2 timing belt was connected to the system as illustrated in Figure 28. The belt was routed around all pulleys, including motor and idler pulleys, forming a closed-loop path for the movement system. The belt was then tensioned and glued to the front section of the Y-axis structure to secure it and prevent slippage during operation. (Figure 36)

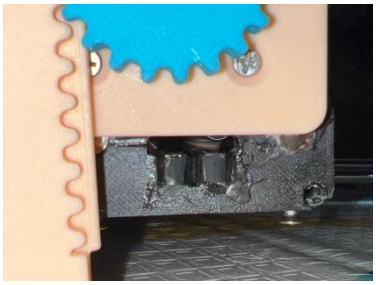


Figure 35: Front of the Y axes support

The next task was the assembly of the Z-axis system. M3 screws were used to mount the NEMA 17 stepper motor, the rack supports, and the limiting part onto the Z-axis support structure. As shown in Figure 37, the back side of the Z-axis support was manually cut to fit the assembly requirements. (Figure 37)

The pinion gear was inserted onto the motor shaft, and the rack was positioned and secured by the rack supports and the pinion gear. This configuration allows the rotational motion of the motor to be converted into linear vertical movement.After completing the assembly of all Z-axis components, the entire Z-axis support structure was carefully glued to the front section of the Y-axis support, forming an integrated frame for vertical movement.

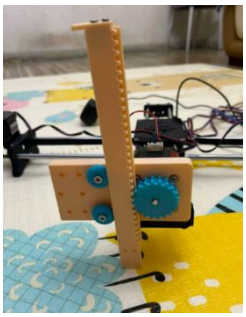


Figure 36: Z-axis system

4.2 Assembling electrical part

The final assembly step involved installing the protective shield for the electrical system. In this stage, the Arduino Uno was mounted onto the bottom section of the shield. For safety, non-conductive washers or screws were used to isolate the Arduino to prevent any short circuits electrically.

Once the Arduino was fixed, the CNC Shield was mounted directly on top of it, and the A4988 stepper motor drivers, pump, and DHT22 sensor were installed into the sockets and pins on the CNC Shield. (Figure 39) After completing the stacking, the front panel of the protective shield was inserted and attached through the threaded inserts on the bottom base. (Figure 39)

=The stepper motor wires were connected to the terminals on the CNC Shield, and the Power supply should connect to the terminal on the CNC Shield, double-checking the terminal sign. Finally, the protective shield's back cover and top cover were placed, completing the electrical system enclosure and ensuring both functionality and protection for the control electronics. (Figure 38)

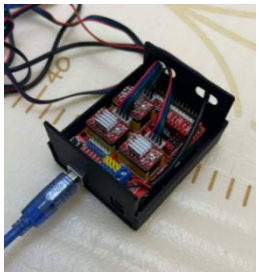


Figure 38: Protective shield of the electrical system

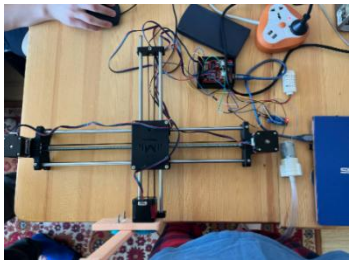


Figure 37: Fully assembled Tariabot

4.3 Software and Programming

The software architecture of Tariabot was divided into two main components: the microcontroller firmware (Arduino) and the server-side application (Node.js with a web interface).

Arduino Firmware

The Arduino Uno was programmed using C++ through the Arduino IDE. The firmware controls the movement of the X, Y, and Z axes based on received commands and manages basic operations such as watering. The Arduino gets G-code-like coordinate commands through serial communication from the server and converts them into motor actions.The firmware responsibilities include:

- Receiving X and Y target coordinates via the serial port.
- Moving the stepper motors in the appropriate directions by activating the CNC Shield drivers.
- Managing the watering operation when a "plant" command is received.
- Return real-time sensor data, such as temperature and humidity, to the server for monitoring.

This communication ensures that Tariabot's movement and planting operations are precisely controlled.

Node.js Server and Web Interface

The Tariabot control system features a small server built using Node.js and Express.js frameworks. The server communicates with the Arduino through a serial port, processes movement and planting commands, and hosts a web dashboard for user interaction. The server's responsibilities include:

- Establishing a serial connection with the Arduino via COM20 at 9600 baud rate.
- Listening for and illustrating sensor data sent from the Arduino.
- Serving a web page developed using EJS templating to users.
- Handling incoming HTTP POST requests for user commands
- Forwarding commands to the Arduino over serial communication.

The live environmental data, such as temperature and humidity are visualized on the web dashboard using a real-time updating line chart created with Chart.js.

Web Dashboard

The front-end interface, built using EJS and HTML, allows users to:

- Monitor live temperature and humidity data updates.
- Visualize historical trends using a live-updating graph.
- Click on a graphical field area to specify planting coordinates.
- Send movement, planting, or homing commands interactively.

Users interact with a drawn 2D grid representing the planting bed. Clicking on this interface translates the click position into X and Y coordinates, which are sent to the Arduino to move and perform planting tasks.

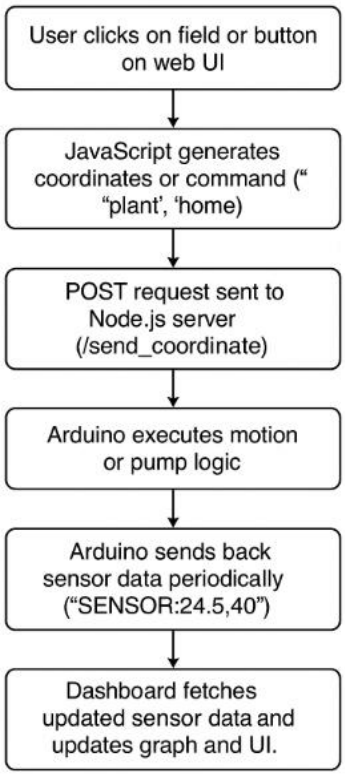


Figure 39: sequence of motion coding process

5. Stress Analysis/ simulation

5.1 Force calculation

Bottom Clamshell

Tension from the timing belt pulling on the pulley creates horizontal force, and due to the total weight of mounted components like the motor and the gantry, it places a vertical load on the clamshell.

Assumptions:

- Pulley tension: 4 N, typical GT2 belt under moderate tension. Based on the
- Total vertical weight:
    - Z axis's NEMA 17 motor is 400 g in weight
    - The tool head, support, and gantry have a total weight of 1.5 kg

$$\text{Total vertical force} = 1.9\text{kg} \times 9.81(\frac{\text{m}}{\text{s}^2}) \approx 18.6\ \text{N}$$

X-Axis Support

Assumptions:

The motor is mounted on a plate that weights 400 g. The rods and other parts weight pulling down through the hole, which weight around 2 kg.

$$\text{Motor weight} = 0.4\text{kg} \times 9.81\left(\frac{\text{m}}{\text{s}^2}\right) = 3.92\ \text{N}$$

$$\text{Rods and other parts load } = 2\text{kg} \times 9.81\left(\frac{\text{m}}{\text{s}^2}\right) = 19.6\ \text{N}$$

$$\text{Total force } = \ 3.92\ \text{N} \ + \ 19.6\ \text{N} = 23.52\ \text{N}$$

5.2 Stress Analysis of 3D Printed Components

Bottom clamshell

To check the mechanical reliability of key 3D printed parts in the Tariabot system by Fusion360, the bottom clamshell and X-axis motor support are tested using estimated loads.

The bottom clamshell supports both the pulley tension force from the GT2 belt and the vertical weight of the gantry and attached components. In simulation, a combined load of 20 N was applied primarily to the pulley mounting zones, with fixed constraints at the bearing mounting holes.

As shown in Figure 41, the maximum stress was found to be 0.548 MPa, concentrated near the inner support ribs. The PLA material with a typical yield strength of 60 MPa has a safety factor of over 100, confirming safe operation under belt tension and component load.

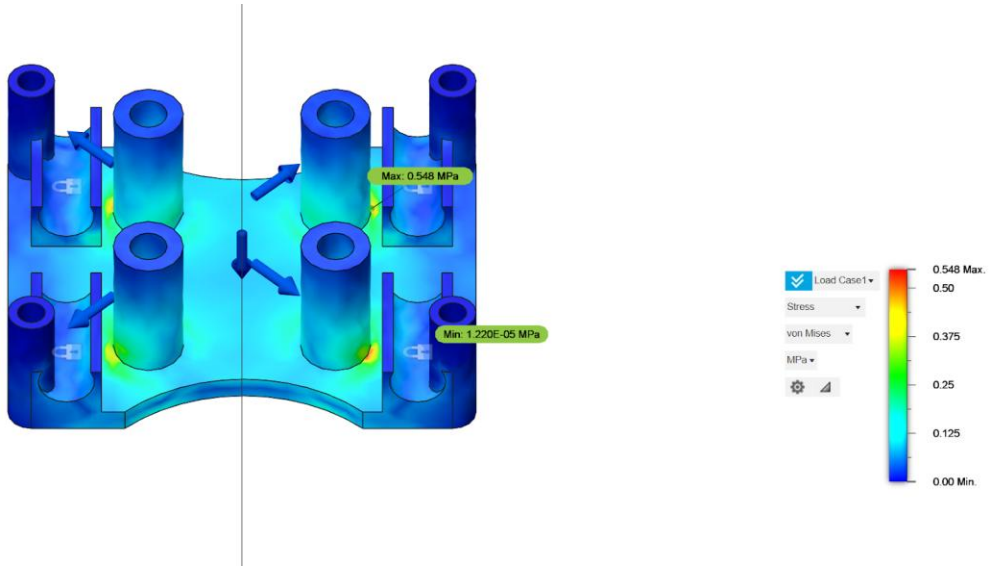


Figure 40: Stress analysis of Bottom clamshell

X-axis support

The X-axis support bracket bears the weight of both the mounted NEMA 17 stepper motor, with 400 g, the long structural rods, with 250g, and other components are weighing 600g. A total vertical load of 23.52 N was applied to the motor mount plate and the lower slot area where the rod passes through. Mounting holes were used as fixed constraints in the simulation.

As shown in Figure 42, the maximum stress was approximately 0.751 MPa**,** located at the interface between the motor plate and structural curve. The result remains well below PLA's yield strength, ensuring structural integrity with a safety factor greater than 70**.**

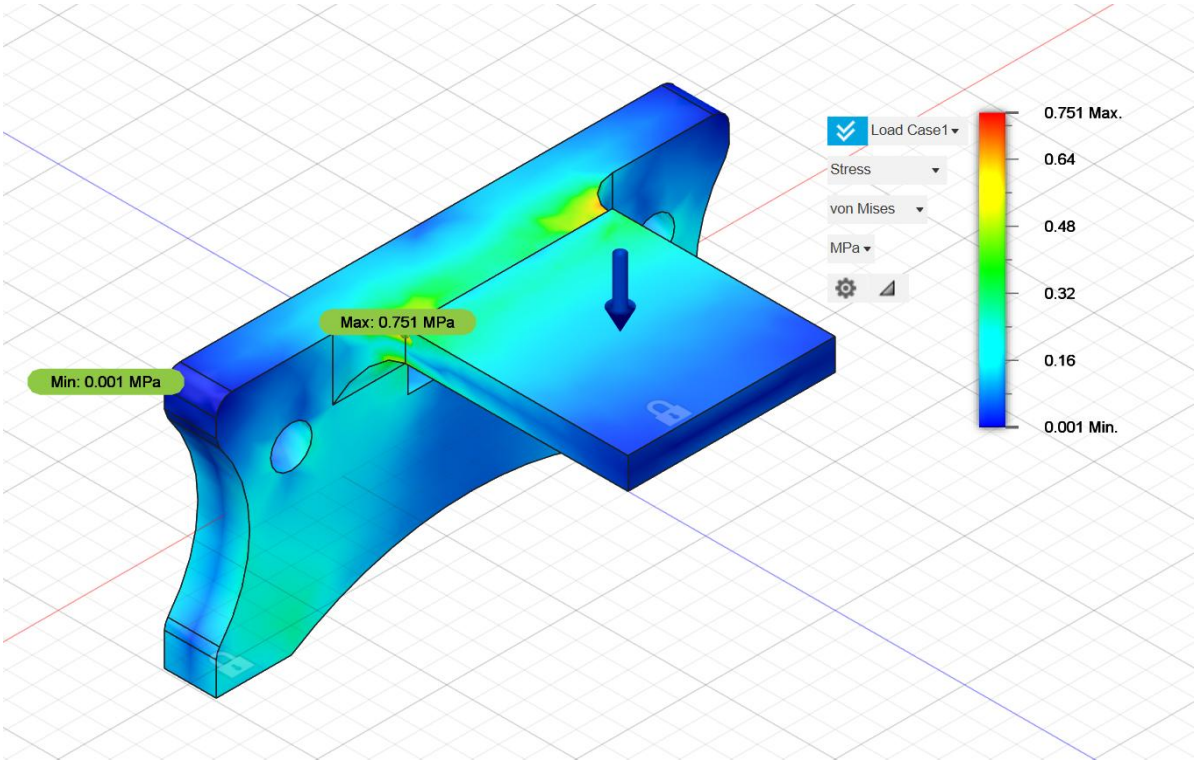


Figure 41: Stress analysis of X axis support

The updated stress analysis confirms that both the bottom clamshell and X-axis motor support are structurally safe under actual load conditions, with max stresses far below material failure limits. The design is suitable for lightweight robotic applications and allows reliable operation of the Tariabot system.

6. Testing phase

6.1 Mechanical Testing

Each axis was tested individually using manual serial commands and web inputs. The stepper motors (NEMA 17) driven by A4988 drivers moved the X, Y, and Z axes smoothly. GT2 belts were adjusted for proper tension to eliminate backlash. The Z-axis, redesigned for rigidity, showed stable vertical motion during repeated watering actions.

Pulley alignment**,** motor bracket rigidity, and bearing friction were all manually inspected. Belt tensioning was improved by extending the Y-axis linear rods by 20 mm, providing sufficient space for adjustment.

6.2 Electrical and Sensor Testing

All electronic components were tested individually before system integration. The Arduino Uno with the CNC Shield and three A4988 drivers responded correctly to serial commands. VREF tuning was performed to ensure correct current output to the motors, avoiding overheating. The DHT22 temperature-humidity sensor was connected and monitored via the Arduino serial monitor. Both provided consistent readings under changing conditions, for example, blowing air and a humid environment.

6.3 Water Pump Test

The RS-360S 12V submersible pump was tested by activating it through a digital pin from the Arduino. The pump successfully delivered ~300–400 mL of water, which was deemed sufficient for a small planting task. It was powered through a 12V supply and triggered via relay control.

6.4 Web Interface Testing

The Node.js server, serial interface, and web interface were tested across multiple sessions. Movement commands such as home, move, and plant were sent via the interface and confirmed by Arduino response and observed motion. Sensor values were visualized using Chart.js, updating every 5 seconds. The graph plotted real-time temperature and humidity trends. Communication stability was evaluated under real conditions.

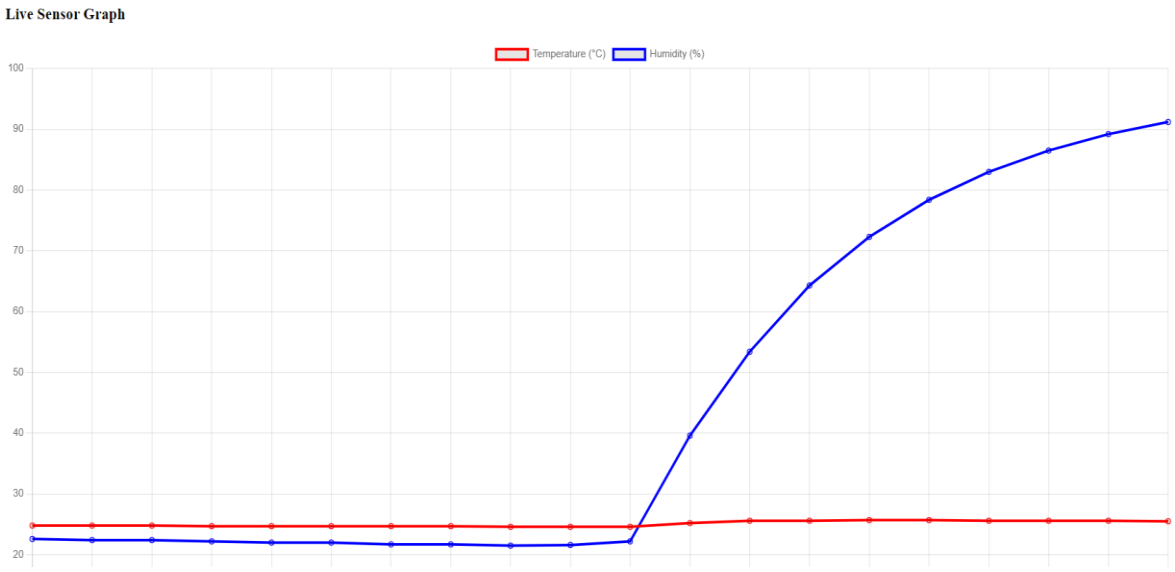


Figure 42: Real-time data graph

6.5 Axis Accuracy and Movement Testing

To evaluate the precision of Tariabot's motion system, a test was conducted using a flat surface marked with a grid pattern. The robot's tool head was commanded to move to predefined coordinate points such as (0, 50 mm)**,** (50, 50 mm)**,** and (100, 50 mm) using both manual serial commands and web-based inputs. Each coordinate was tested ten times, and the actual landing point of the tool head was visually inspected and measured. The average positional deviation was found to be ±5 mm**,** with the largest errors occurring during long diagonal moves. These deviations are acceptable for small-scale planting systems where seed spacing tolerances are relatively relaxed. The system's accuracy was enhanced by enabling 1/16 microstepping on the A4988 stepper drivers, providing finer resolution without sacrificing torque. After proper belt tensioning and motor calibration, the motion became smoother and more reliable. These results confirmed the effectiveness of the Cartesian belt-driven motion system for indoor planting automation.

An integrated planting test was conducted by:

1. Clicking a target position on the grid interface
2. Observing X/Y motion
3. Triggering the Z-axis movement cycle
4. Logging returned position and timing

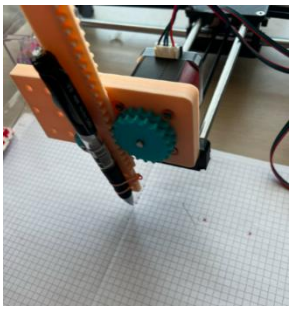


Figure 43: Result of accuracy test

7. Result and discussion

Overview of Testing Objectives

The testing phase was conducted to evaluate the functional performance of the Tariabot prototype across mechanical, electrical, and software components. After full assembly and firmware deployment, individual subsystems were tested, followed by integrated system tests. The primary goals were to:

- Assess the accuracy and stability of XYZ axis movements.
- Validate the planting mechanism for repeatability and reliability.
- Confirm the effectiveness of the watering system.
- Evaluate sensor response for real-time monitoring.
- Assess usability and responsiveness of the web-based user interface.
- Identify and overcome design flaws or limitations discovered during real-world operation.

Each of these goals was translated into real-world experiments conducted in a controlled indoor environment.

Axis Movement and Positional Accuracy

The Tariabot's motion is based on a Cartesian XYZ axis configuration controlled by NEMA 17 stepper motors. Each axis was tested using a series of manual commands and web-based coordinate inputs. The microstepping (1/16) enabled fine motion control with belt-driven movement on the X and Y axes and a rack-and-pinion mechanism on the Z-axis. To measure movement accuracy, a test grid was created on a flat surface. The tool head was commanded to move to specific coordinates (e.g., 0×50 mm, 50×50 mm, 100×50 mm) repeatedly across ten trials. The average deviation from the expected landing point was ±5 mm**,** which is acceptable for small-scale planting tasks. Movement was smooth after proper belt tensioning and motor calibration.

During early tests, belt slippage was observed in the Y-axis. To resolve this, the linear rods were extended by 20 mm, allowing additional space to pull and tension the GT2 timing belt properly. This improved both motion stability and repeatability. Another issue emerged with the Z-axis support alignment. The original 3D-printed bracket has additional unnecessary surface. This was resolved by manually redesigning the Z-axis mount and securely gluing it to the front of the Y-axis vertical support (Figure 45).

Figure 44: Back of the Z axis support

Watering System Testing

The RS-360S 12V submersible pump was tested by activating it through an Arduino-controlled digital pin. The pump was mounted in a small water reservoir and connected via flexible tubing to the tool head. Each watering event delivered approximately 350–400 mL of water, measured using a graduated cylinder. The flow was sufficient for small trays or planters. To automate irrigation, a DHT22 sensor was used as a trigger: if the humidity value dropped below a defined threshold, the pump activated until the moisture rose again. The system operated in an open-loop for the initial tests, with a delay-based timer, and later switched to moisture feedback logic. This addition allowed semi-autonomous watering, a key feature for real-world use.

Sensor Feedback and Monitoring

Environmental monitoring was implemented using a DHT22 sensor for temperature and humidity. The sensors were connected to the Arduino and transmitted data through the serial port to the Node.js server. Data was visualized using Chart.js**,** with line graphs plotted in real time on the web dashboard. Values were updated every 5 seconds, showing live trends. During testing, temperature values varied from 24°C to 27°C, and relative humidity ranged between 34% and 43%, matching the nearby room temperature and humidity level.

Web Dashboard Functionality

The Tariabot's web interface, developed using EJS templates and served via Express.js, was successfully deployed and tested. Key features include:

- A visual grid representing the planting area
- Live sensor data graphs
- Buttons for sending "home," "plant," and "move" commands
- Real-time feedback of action status

Users could click on the grid to send X/Y coordinates to the Arduino. The Node.js backend handled this via serial communication, converting user inputs into simplified G-code commands. The Arduino received and interpreted the commands correctly, executing movement and tool control. However, it handles data correctly, but the illustrated delay was higher than expected. The system remained stable during 1-hour test sessions.

Design Comparison and Functional Reflection

Compared to more complex systems like FarmBot**,** Tariabot provides a minimalist, lower-cost solution. A feature-by-feature comparison is shown below:

Feature	**FarmBot**	**Tariabot**
Cost	> $3,000	<$200
Axis system	Outdoor aluminum with waterproof motors	3D-printed PLA with indoor use only
Seeding	Vacuum tool changer	Fixed tool with stepper lift
Web UI	Full-featured, simulation	Simplified EJS dashboard
Sensor suite	Camera, pH, water flow, and moisture	Temp, humidity, moisture
Tool changing	Yes	No
Assembly complexity	Moderate to High	Low to Medium
Programming	Advanced	Beginner to Intermediate

Table 3: Comparison between Farmbot and Tariabot

While FarmBot supports modular tool changes and sophisticated analytics, Tariabot focuses on essential automation for students, hobbyists, or resource-limited growers. Its simplified hardware and software stack make it accessible and easy to adapt.

Challenges and Lessons Learned

During prototyping, several challenges occurred:

- Belt slippage: Fixed by increasing Y-axis rod length by 20 Mm.
- Excessive design Z-mount: Solved through manual redesign and gluing.
- Node.js serial instability: Resolved by adjusting the USB cable and error handling.

These issues led to iterative design improvements that strengthened the final system and highlighted the importance of mechanical alignment, electrical safety, and software fail-safes.

Summary of Results

Test Area	**Result**
Axis repeatability	±1.3 mm
Watering output	350–400 mL per activation
Temperature accuracy	±1°C
Humidity accuracy	±1% RH
UI command response	<1 second delay
Sensor data response	< 5-second delay

Table 4: Testing result

8. Conclusion

Future Work

While Tariabot fulfills its core objectives, several opportunities exist for improvement and expansion:

1. Full Automation of Seeding and Watering

Currently, watering is triggered via user input. Future versions can implement fully autonomous cycles based on plant schedules, soil conditions, and environmental data.

2. Tool Changing System

Inspired by FarmBot's multi-tool head concept, a magnetic or mechanical tool changer could allow switching between seeders, waterers, and sensors without manual intervention.

3. Camera and AI Integration

Adding a camera module and integrating computer vision would allow plant growth monitoring, weed detection, and more precise seeding.

4. Wireless Communication

Switching from USB serial to Wi-Fi or Bluetooth using ESP32 would remove the tethered connection, enabling wider deployment flexibility and mobile access.

5. Weather-Proofing and Outdoor Scales

Future versions may use corrosion-resistant materials like aluminum and ABS and waterproof enclosures to operate in balcony or outdoor environments. Motor housings and cabling would need to be reinforced.

6. Real-Time Data Logging and Analytics

Sensor values could be logged in a database and visualized through dashboards, allowing users to track environmental trends and optimize planting decisions.

Final Reflection

The Tariabot project reflects how engineering principles, open-source technologies, and iterative problem-solving can be applied to real-world agricultural automation. Although modest in scale, the system serves as a proof-of-concept for how smart farming technologies can be simplified, made more affordable, and accessible to students, researchers, and urban gardeners. The experience gained through this project, including mechanical design, sensor integration, software communication, and system testing, has further developed the author's ability to carry out engineering design. Future development of Tariabot will continue to push toward higher autonomy, intelligence, and usability in smart agricultural systems.

Summary

This thesis presented the design, development, and testing of Tariabot, a low-cost, small-scale, semi-autonomous planting robot. Inspired by open-source systems like FarmBot, Tariabot was designed to be simpler, more accessible, and optimized for indoor or compact space gardening. The system integrates mechanical motion, watering, and environmental sensing, all controllable through a web interface. The project successfully combined CNC motion principles with sensor-based automation using open-source components: NEMA 17 stepper motors, A4988 drivers, Arduino Uno, soil and climate sensors, and 3D printed parts. The Node.js-powered server enabled real-time user control and feedback via a web dashboard, while Chart.js provided live sensor visualization.

Through testing and simulation, Tariabot was shown to:

- Accurately execute motion and planting sequences
- Deliver water properly
- Display real-time sensor data for temperature and humidity
- Operate stably with a simplified G-code-like communication system

Mechanical issues like belt tensioning and Z-axis alignment were identified early and resolved through design iteration. The final system met its functional goals with minimal cost, proving that essential agricultural automation can be achieved using accessible tools and knowledge. This project demonstrates how mechatronics engineering, 3D printing, and software integration can be combined into practical automation systems that serve educational, research, or even commercial purposes in controlled environments.

9. References

1. Market.us. Farm Automation Market Size, Trends and Forecast to 2033. 2024.
2. Inc. F. Frequently Asked Questions – FarmBot Genesis V1.5. [Online].; 2024. Available from: https://genesis.farm.bot/v1.5/FarmBot-Genesis-V1.5/intro/faq.html.
3. Britannica E. Agricultural technology. [Online].; 2025. Available from: https://www.britannica.com/technology/agricultural-technology.
4. Britannica E. History of agriculture. [Online].; 2025. Available from: https://www.britannica.com/topic/agriculture.
5. Rexroth B. Linear Motion Technology Handbook 2021: Bosch Rexroth; 2021.
6. Inc. F. FarmBot Genesis Documentation: Motors and Movement Systems. [Online].; 2024. Available from: https://genesis.farm.bot/.
7. Mohsin SMAaH. Design and Fabrication of 3-Axis Mini CNC Milling Machine. IOP Conference Series: Materials Science and Engineering. 2021; 1094(1).
8. Hasan H. Implementation and Manufacturing of a 3-Axis Plotter Machine by Arduino and CNC Shield. In Al-Qadisiyah University; 2018.
9. Electronics PRa. A4988 Stepper Motor Driver Carrier. [Online].; 2025. Available from: https://www.pololu.com/product/1182.
10. Electronics PRa. Pololu. [Online].; 2025. Available from: DRV8825 Stepper Motor Driver Carrier, High Current.
11. M. Y. Javed STHRMASKAOBNAAaKS. Low Cost Computer Numeric Controller Using Open Source Software and Hardware. Science International (Lahore). 2015; 27(5).
12. Elangovan RRaS. Design and Development of a Low-Cost Multipurpose Arduino CNC Plotter for Industrial Applications. In Proceedings of the 2024 International Conference on Advanced Mechatronics, Design, and Manufacturing Systems (ICAMDMS); 2024; Coimbatore, India.
13. Industries A. DHT22 Temperature-Humidity Sensor Datasheet. [Online].; 2020 [cited 2025 4 30. Available from: https://www.adafruit.com/product/393.
14. Sensortec B. BME280 Combined Humidity and Pressure Sensor. [Online].; 2017 [cited 2025 4 30. Available from: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/.
15. DFRobot. Gravity: Analog Capacitive Soil Moisture Sensor - Corrosion Resistant. [Online].; 2021 [cited 2025 4 30. Available from: https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193.

16. Electronics R. Water Pumping Motor RS-360 Mini DC 4–12V. [Online]. [cited 2025 4 30. Available from: https://rcl.lt/en/products/svs01-water-pumping-motor-rs-360-mini-dc-4-12v.

17. JENENSERIES. 12V Solar Water Pump, 2.5-4.5L/min Submersible Micro Pump. [Online]. [cited 2025 4 30. Available from: https://www.amazon.co.uk/JENENSERIES-12V-Solar-Flow%EF%BC%8C2-submersible/dp/B091B9PNQ3.

18. Vukovic L. ESP32 and Python for Internet of Things Applications. [Online].; 2019. Available from: https://zerynth.com/blog/esp32-and-python-for-internet-of-things-applications/.

19. Toxigon. ESP32 MicroPython: Connecting IoT Sensors for Smart Projects. [Online].; 2025.

20. Factory A. How to use Python Programming Language for IoT (Internet of Things). [Online]. Available from: https://www.articlesfactory.com/articles/programming/how-to-use-python-programming-language-for-iot-internet-of-things.html.

21. S. Tilkov SV. Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing. 2010; 14(6): 80.

22. M. Cantelon MHTHNR. Node.js in Action. 2nd ed. Shelter Island, NY: Manning Publications; 2013.

23. Shafranovich Y. Best Practices for Real-Time Web Applications with Node.js and WebSocket. Communications of the ACM. 2019; 62(2): 32.

24. DuBois P. MySQL. 5th ed.: Addison-Wesley Professional; 2008.

25. Test3DPrints. Homework Writing Machine – Arduino CNC Plotter Project. [Online].; 2024. Available from: https://test3dprints.com/arduino/homework-writing-machine/.

26. Inc. F. FarmBot Whitepaper. [Online].; 2024. Available from: https://farm.bot/pages/whitepaper.

27. Market.us. Farm Automation Market Size, Trends and Forecast to 2033. [Online].; 2024. Available from: https://market.us/report/farm-automation-market/.

10. Appendices

Figure 45: Web Interface design

Motorwithoutweb.ino

```
#include "DHT.h"

// CNC Shield Motor Pins
const int stepA = 2;
const int dirA  = 5;
const int stepB = 3;
const int dirB  = 6;
const int stepZ = 4;
const int dirZ  = 7;
const int enPin = 8;
const int pumpPin = A1;  // Relay control

// DHT22 Sensor
#define DHTPIN A0
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// Position Tracking
const float stepsPerMM_XY = 7.0;
const float stepsPerMM_Z = 7.0;
long currentX_steps = 0;
long currentY_steps = 0;
long currentZ_steps = 0;
float offsetX_mm = 0.0;
float offsetY_mm = 0.0;

// Sensor reading timing
unsigned long lastSensorTime = 0;
const unsigned long sensorInterval = 5000; // 5 seconds

void setup() {···
}

void loop() {···
}

void handlePlantingCommand(String command) {···
}

void moveToHomeCenter() {···
}

void setSoftwareOriginHere() {···
}

void moveToTargetMM(float targetX_mm, float targetY_mm, float targetZ_mm) {···
}

void moveZSafeHeight() {···
}

void moveZWorkingHeight(float workZ_mm) {···
}

void moveZto(float targetZ_mm) {···
}

void moveXY(float targetX_mm, float targetY_mm) {···
}
```

Figure 46: Motor Control code

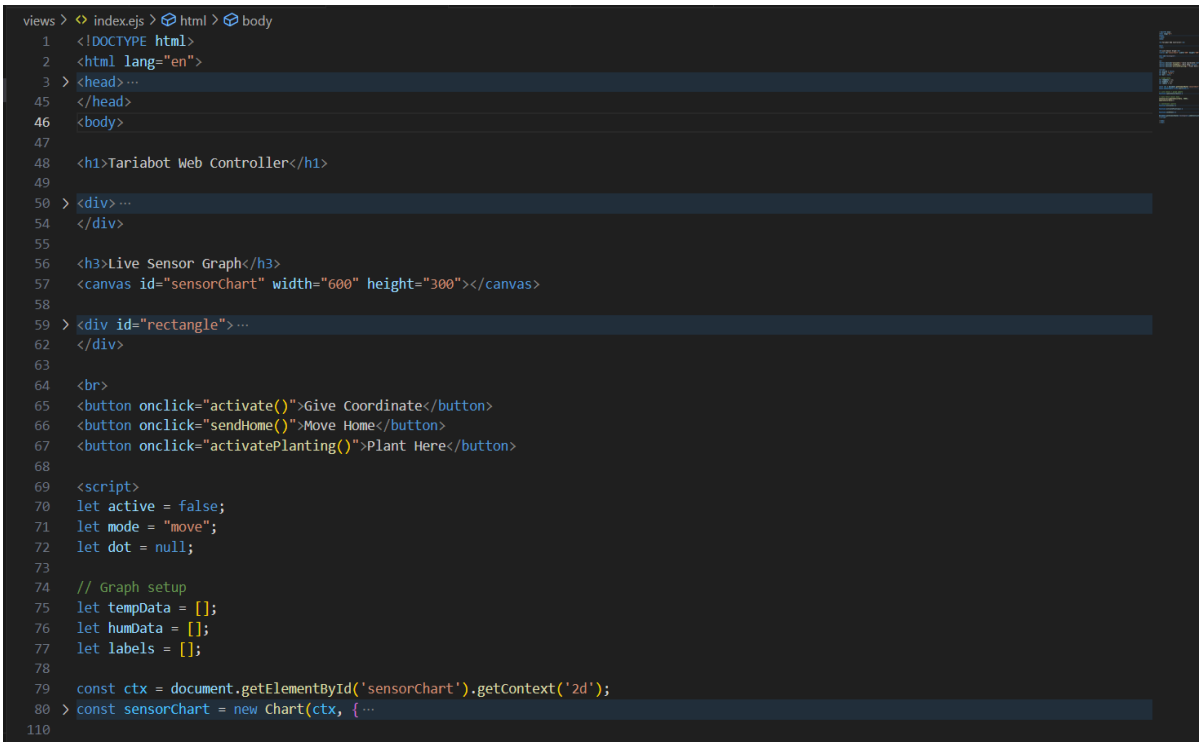
sketch_may1a.ino

```
// defines pins numbers
const int stepX = 2;
const int dirX  = 5;
const int stepY = 3;
const int dirY  = 6;
const int stepZ = 4;
const int dirZ  = 7;
const int enPin = 8;
void setup() {
  // Sets the two pins as Outputs
  pinMode(stepX,OUTPUT);
  pinMode(dirX,OUTPUT);
  pinMode(stepY,OUTPUT);
  pinMode(dirY,OUTPUT);
   pinMode(stepZ,OUTPUT);
  pinMode(dirZ,OUTPUT);
  pinMode(enPin,OUTPUT);
  digitalWrite(enPin,LOW);
  digitalWrite(dirX,HIGH);
  digitalWrite(dirY,LOW);
  digitalWrite(dirZ,HIGH);
}
void loop() {
  // Enables the motor to move in a particular direction
  // Makes 200 pulses for making one full cycle rotation
  for(int x = 0; x < 800; x++) {
    digitalWrite(stepX,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepX,LOW);
    delayMicroseconds(1000);
  }
delay(1000); // One second delay
 for(int x = 0; x < 800; x++) {
  digitalWrite(stepY,HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepY,LOW);
  delayMicroseconds(1000);
 }
 delay(1000); // One second delay
for(int x = 0; x < 800; x++) {
  digitalWrite(stepZ,HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepZ,LOW);
  delayMicroseconds(1000);
}
 delay(1000); // One second delay
}
```

Figure 47: Motor testing code

```
const express = require('express');
const path = require('path');
const connection = require('./db');
const bodyParser = require('body-parser');
const { SerialPort } = require('serialport');
const { ReadlineParser } = require('@serialport/parser-readline');

const app = express();
const PORT = 3000;

// Connect to Arduino
const port = new SerialPort({ path: 'COM20', baudRate: 9600 });
const parser = port.pipe(new ReadlineParser({ delimiter: '\n' }));
port.on('open', () => {···
});
let latestSensorData = { temperature: null, humidity: null };
parser.on('data', data => {···
});
// Middlewares
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
app.use(express.static(path.join(__dirname, 'public')));
app.use(bodyParser.json());
// Home page
app.get('/', (req, res) => {···
});
// Get latest sensor data
app.get('/get_sensor', (req, res) => {···
});
// Handle sending coordinates
app.post('/send_coordinate', (req, res) => {···
});
// Start server
app.listen(PORT, () => {···
});

```

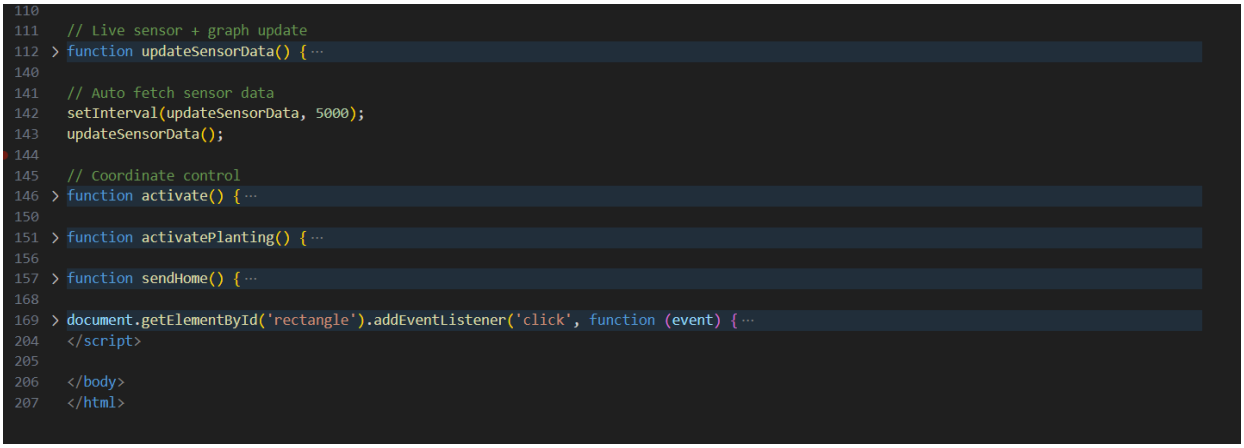
Figure 48: Backend code Server.js



views > index.ejs > html > body

```
<!DOCTYPE html>
<html lang="en">
<head>···
</head>
<body>

<h1>Tariabot Web Controller</h1>

<div>···
</div>

<h3>Live Sensor Graph</h3>
<canvas id="sensorChart" width="600" height="300"></canvas>

<div id="rectangle">···
</div>

<br>
<button onclick="activate()">Give Coordinate</button>
<button onclick="sendHome()">Move Home</button>
<button onclick="activatePlanting()">Plant Here</button>

<script>
let active = false;
let mode = "move";
let dot = null;

// Graph setup
let tempData = [];
let humData = [];
let labels = [];

const ctx = document.getElementById('sensorChart').getContext('2d');
const sensorChart = new Chart(ctx, {···
```

Figure 49: Frontend code Index.ejs



```
// Live sensor + graph update
function updateSensorData() {···

// Auto fetch sensor data
setInterval(updateSensorData, 5000);
updateSensorData();

// Coordinate control
function activate() {···

function activatePlanting() {···

function sendHome() {···

document.getElementById('rectangle').addEventListener('click', function (event) {···
</script>

</body>
</html>
```

Figure 50: Frontend code Index.ejs

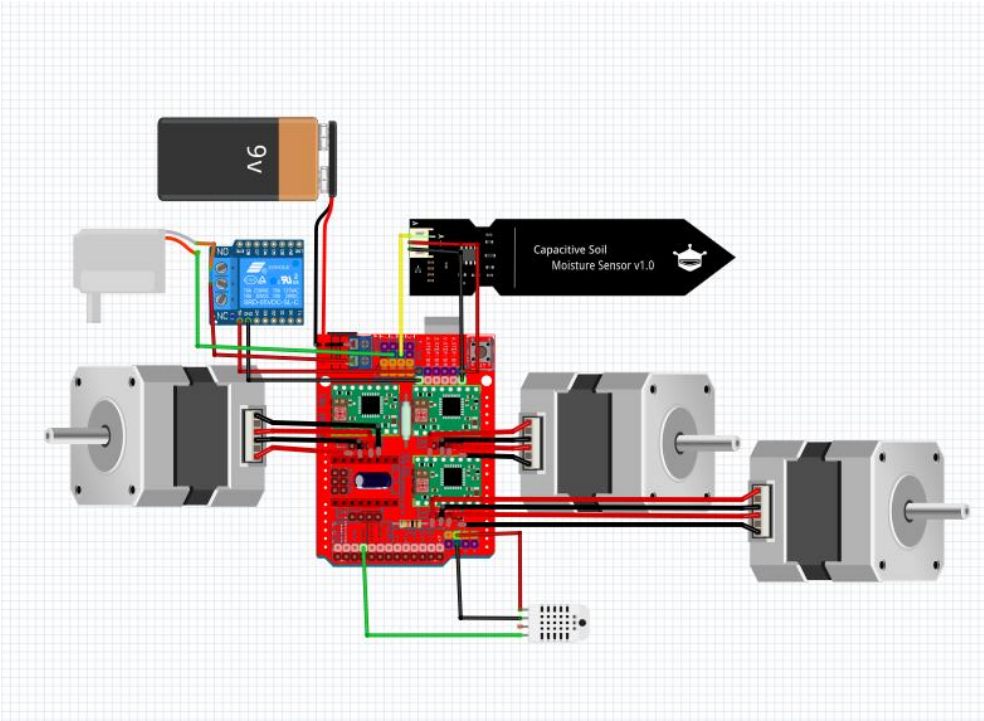


Figure 53: Electrical scheme

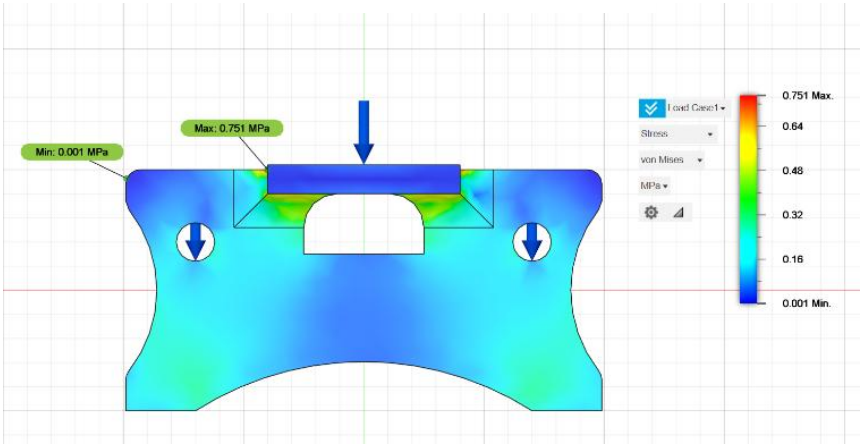


Figure 51: X axis support Force direction

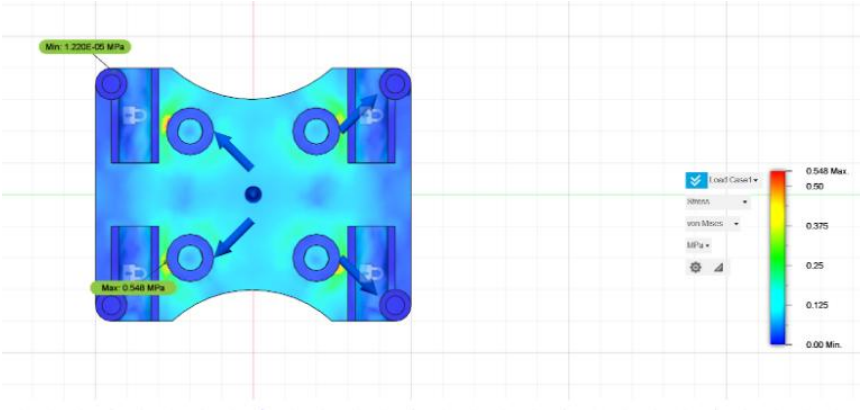


Figure 52: Bottom clamshell Force direction