Technical University of Applied Sciences Würzburg-Schweinfurt

Engineering Project SS – 23

Mechatronics IMC

Resistor Reel Cutter Machine

In cooperation with:

Center Robotics [CERI]

[Prof. Dr. Rainer Herrler and Mr. Fabian Dax]

Members	Matriculation Number
Sai Karthik Shankar	4019084
Sathvick Bindinganavale Srinath	4020025
Aniketh Padmakar	4019074
Samrat Shantesh Ukkali	4019124

Table of Contents

List of Figures

1. Introduction

Topic written by: Aniketh Padmakar

In modern electronics printed circuit board(PCB) assembly lines, pick and place machines play a crucial role in efficiently handling components from SMD reels and accurately placing them onto the PCB. However, this process becomes challenging for medium-scale production or when employing through-hole type(THT) components on the PCB. This limitation is particularly evident in low-cost board assemblies like LED drivers and transformerless power supplies[1]. In The Centre of Robotics campus of the Technical University of Applied Sciences Würzburg-Schweinfurt, Schweinfurt, one of the many labs organizes a line following bot development course for their Robotics students. These students need a set of resistors to build their bots. They needed a device to expedite the lab preparation process, hence we developed the Resistor reel-cutting device to address this requirement.

Our cutting machine is capable of handling various types of component reels, such as resistor reels, diode reels, and similar ones[1]. These component reels have the same taping specifications, formulated based on broad guidelines extracted from DIN-IEC-286-1 standards. By efficiently extracting a precise number of components from a reel and performing accurate cuts, our approach allows for the seamless division of an entire reel into the necessary smaller units. As a result, the component preparation phase of the assembly process is significantly streamlined, enabling smoother operations and enhanced efficiency.

2. Project Management

Topic written by: Aniketh Padmakar & Sathvick Bindinganavale Srinath

2.1. Team

Team Members	Roles
Sai Karthik Shankar	**Project Leader** Mechanical - Cutting Module & Housing
Sathvick Bindinganavale Srinath	**Software** - Arduino Code for Actuations Electrical - Wiring Plan
Aniketh Padmakar	**Minutes taker** and Admin-work Software - Display software and Display-Arduino code
Samrat Shantesh Ukkali	**Electrical Design** - PCB Design

After gaining a deeper understanding of the project's nature and requirements, we made a collective decision to divide the tasks among different engineering disciplines: Mechanical, Electrical, and Software. Each team member had the opportunity to choose the field they were most interested in and passionate about. The allocation of responsibilities is outlined in the table above.

To effectively manage and track the progress of the project, we adopted Notion, an all-in-one workspace that provides a comprehensive set of tools for organizing thoughts, creating plans, and streamlining projects[7]. By leveraging the features offered by Notion, we established a project tracking system specifically tailored to the Resistor Reel Cutter project.

Notion allowed us to centralize all project-related information, including documentation, task assignments, timelines, and milestones. It served as a collaborative platform where team members could collaborate, communicate, and update their progress in real-time.

We established a dedicated GitHub repository for streamlined collaboration and version control during the coding phase of the project. The repository served as a centralized hub where team members could share, review, and collaborate on the codebase.

To access the project's code, you can visit the GitHub repository at [2].

2.2. Work Packages Overview

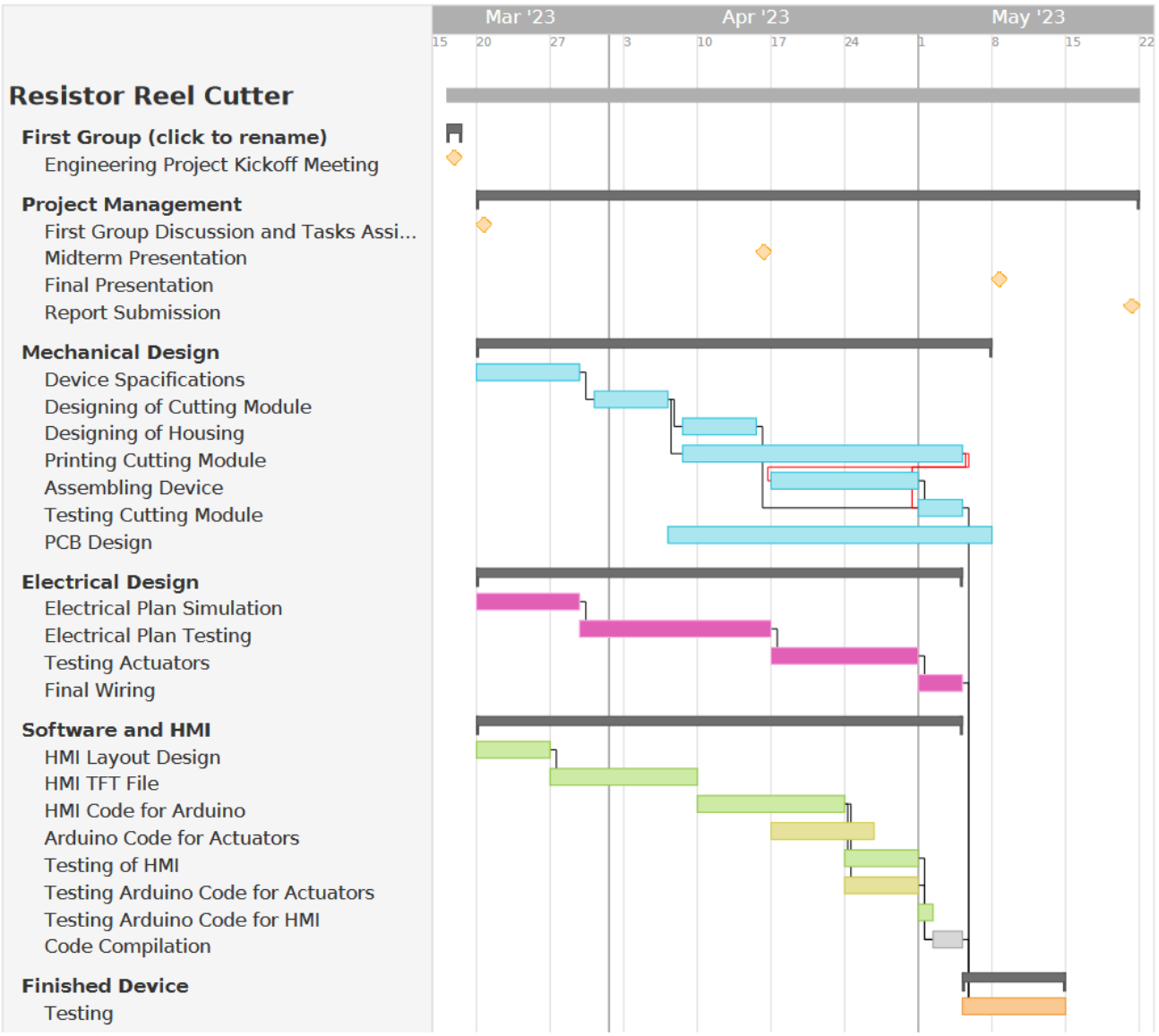


Figure 1: Gantt Chart

The provided diagram illustrates the distribution of tasks and their corresponding durations for each engineering aspect of the project. The table in section 2.1 presents a comprehensive overview of the specific aspects that were undertaken by different team members. It showcases how the responsibilities were divided among the team, highlighting their respective areas of expertise and contributions to the project.

3. Mechanical

Topic written by: Sai Karthik Shankar

Topic developed by: Sai Karthik Shankar

3.1. Project Specification

This section explains the derivation of the customer requirement importance rankings and the correlating engineering specifications for those requirements as established by discussions with the client. These requirements have been reevaluated during the design process, selection of the alpha design, and discussion of the alpha design with mentors.

Customer requirements and engineering specifications were initially developed and weighted for importance by the team. Adjustments and improvements to the importance and parameters of these requirements and specifications were made after multiple interviews with the client.

Scaled Rating	Importance
5	Critical
2-4	Moderate
1	Bonus Feature

Figure 2: Scaled Rating vs Importance

The importance of customer requirements is evaluated on a scale from 1 to 5. Table 2 below explains the rating correlation to importance level.

Two operating bands

as per the the client's request, this was given a rating of 5. The machine should be able to take in two separate reels of taped components, and be able to simultaneously process them into sets of the specified size. This increases the throughput of the device and reduces the time required to process the components.

Accessible HMI (Human machie interraction)

To interact with the device to either alter the settings, or provide the different input and calibration commands, it is crucial to have a simple and straightforward human machine interaction. requiring a secondary device(Laptop computers, mobile phones, etc…) to communicate with the device for regular operation would increase the complexity of the device and be cumbersome to use. Therefore this was assigned an importance rating of 4.

Safety

As the device is meant to cut taped components to process them into smaller sets, it is very likely that the device would contain high torque motors, rollers with a pinching hazard, and moving blades that produce a significant safety hazard during operation. Therefore having an enclosure/ safety measures to prevent such hazards are given an importance rating of 5.

Consumer Electronics Rating

The CE mark on a product indicates that the manufacturer or importer of that product affirms its compliance with the relevant EU legislation and the product may be sold anywhere in the European Economic Area (EEA). It is a criminal offence to affix a CE mark to a product that is not compliant or offer it for sale.

Due to restrictions in time and budget, also with the underlying criteria that this would be a one-off prototype and not a mass production product, a decision was taken not to focus on the CE conformity, but rather to be mindful of the design decisions to reduce the risk to the operator.

3.2. Initial Design Description

The design specification for the project called for a feeder and cutter module that emphasizes modularity. To meet this requirement, an initial brainstorming drawing was created, outlining the overall concept. The module was divided into three distinct functional parts, each serving a specific purpose.

The first part is the feeder guide, which serves as the entry point for the user to insert the reel of components. This guide ensures proper alignment and smooth feeding of the components into the module. It may include features such as alignment pins or channels to facilitate the correct positioning of the reel.

The second part is the active roller, which plays a crucial role in gripping and driving the reel of components. This roller is powered by a NEMA 17 stepper motor, which provides precise

control over the movement of the reel. By gripping the reel securely, the active roller ensures consistent feeding of the components during the cutting process.

The third and final part is responsible for cutting the reel into individual components. This part utilizes two servo motors to perform the cutting action. The precise control offered by the servo motors allows for accurate and clean cuts. The module may include a cutting mechanism such as a blade or a pair of shears to separate the components from the reel.

By splitting the feeder and cutter module into these three functional parts, modularity is achieved. Each part can be designed and assembled independently, making it easier to maintain, upgrade, or replace specific components if necessary. This modular approach also facilitates scalability, as multiple modules can be combined or rearranged.

3.3. Initial Prototype Description

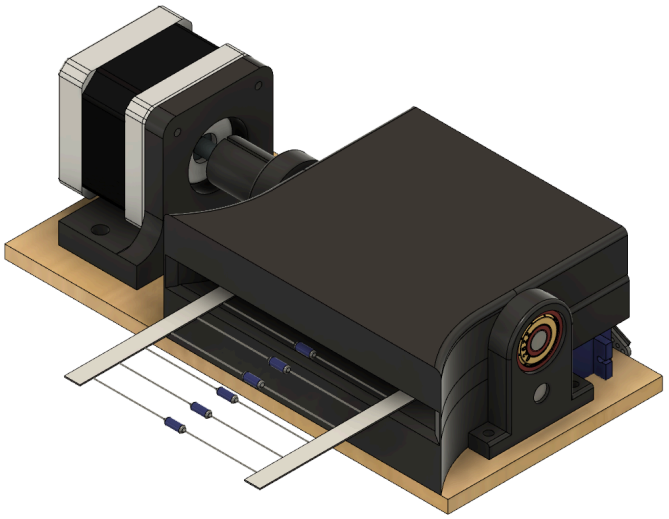


Figure 3: Feeder and cutter module Version 1

The above figure shows the complete feeder and cutter module. This sub section describes this version of the design in detail.

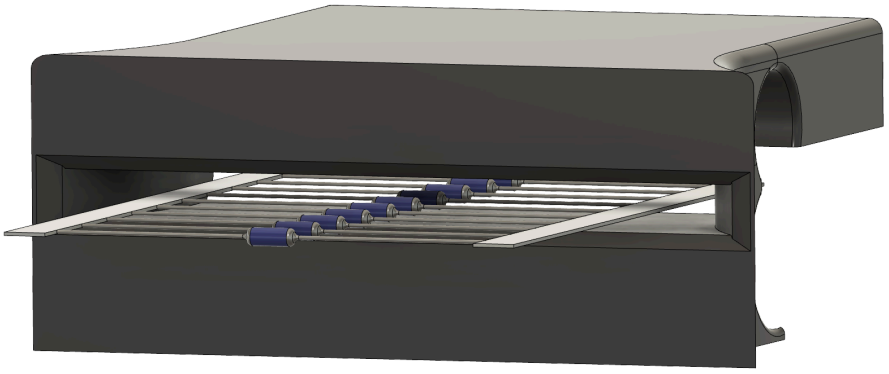


Figure 4: Feeder Guide Version 1.

The first part of it is the feeder guide. it was designed with the maximum width of such a reel taken into consideration. The feeder guide was also designed with a gradual curve in the opening to make it easier for the user to insert the reel into the module.

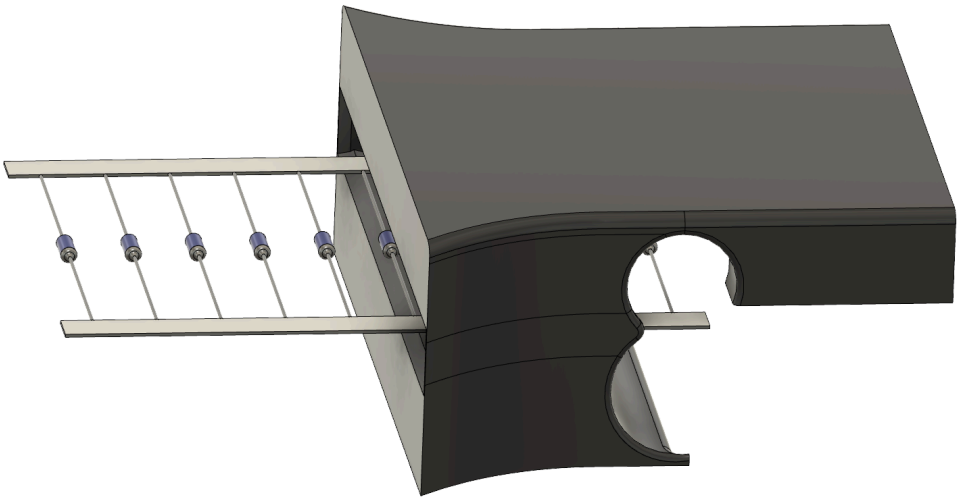


Figure 5: Feeder Guide with Sensor cover Version 1

The other side of the guide was also extended to provide a cover for the cutter module which encloses an Infrared reflective sensor to reduce the ingress of stray light, thereby reducing the noise in the readings and provide a more reliable reading.

In the design of the second part of the module, the active roller, a stepper motor was chosen as the driving mechanism for the rollers. The rollers themselves were 3D printed using TPU (Thermoplastic Polyurethane), a flexible and durable material. To ensure smooth

movement and provide traction to the reel, the rollers were mounted on a 6mm ground stainless steel rod.

To minimize frictional forces and prevent slippage, an additional rod with ball bearings was incorporated into the design. This arrangement allowed the reel to be securely gripped and driven towards the cutting module. The ball bearings provided smooth rotation and reduced the amount of friction between the roller and the reel, enabling efficient and reliable feeding of the components.

By using a stepper motor to drive the rollers, precise control over the movement of the reel is achieved. Stepper motors provide accurate positioning and can be easily controlled to rotate a specific number of steps, ensuring consistent feeding of the components during the cutting process.

The combination of the TPU rollers, the stainless steel rod, and the ball bearings creates an effective pinching mechanism that securely holds and drives the reel without introducing excessive friction. This design choice balances the need for traction with the requirement for smooth and reliable movement, ultimately enhancing the overall performance of the feeder and cutter module.

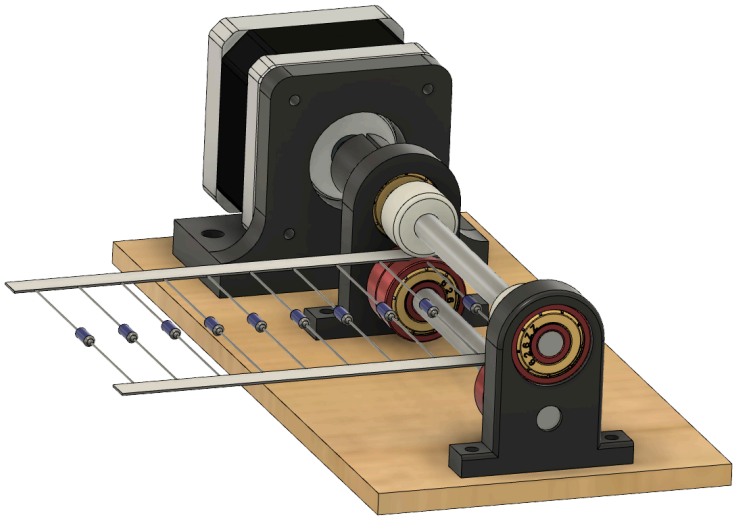


Figure 6: Active Roller Version 1.

The roller road was also supported on both the sides with ball bearings to provide a consistent pressure to both sides of the reel. Having inconsistent pressure on this would result in binding of the whole mechanism or bend the components in the reel.

The third part of the module is the cutter. This part is mainly comprised of two opposing SG90 9 gram servo motors with scalpel blades attached to it.

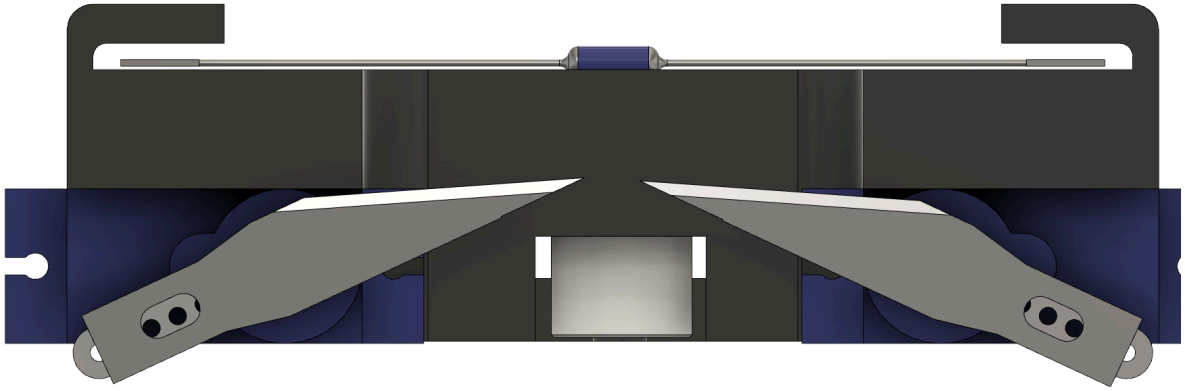


Figure 7: Cutting module Version 1.

Just before the tape reaches the blade in the cutting module, a lip is incorporated into the design. This lip serves multiple purposes. Firstly, it helps prevent bending of the reel during the cutting operation. By providing support to the tape, it ensures that the cut is consistent and repeatable, maintaining the desired accuracy in the process. Additionally, the lip helps to minimize deflection in the Y-axis caused by the blades.

To address the issue of components slipping in the X-axis during the cutting process, the servo motors are strategically mounted. The cutting action produced by the servo motors applies equal and opposing forces to both sides of the tape. This balanced force distribution helps to stabilize the tape, preventing it from shifting or sliding sideways. By maintaining proper alignment, the module ensures that the cutting action is precise and that the components are cleanly separated from the reel without any lateral displacement.

Overall, the incorporation of the lip and the balanced mounting of the servo motors contribute to the reliability and accuracy of the cutting module. These design features work together to ensure repeatable cuts, prevent bending or deflection, and minimize any unwanted movement of the components during the cutting process.

After prototyping the initial design, it became apparent that certain aspects of the design were effective, while others required improvement.

One issue that arose was the gradual curvature in the feeder, which did not adequately constrain the angular movement of the reel. This resulted in binding and interference with the rollers. To address this, a revised feeder design with better angular constraint could be

implemented. This would ensure smoother movement of the reel and prevent binding, enhancing the overall performance of the module.

Another area for improvement was the choice of TPU material for the rollers. It was found that the TPU material did not provide sufficient traction, which could lead to slippage and inconsistent feeding. Considering an alternative material with better grip properties could enhance the traction and gripping ability of the rollers.

Additionally, it was discovered that the SG90 servo motors lacked the required torque to effectively cut the reel. Upgrading to servo motors with higher torque ratings could address this limitation.

On a positive note, the use of opposing bearings to feed the reel was found to be effective. This arrangement helped to stabilize and guide the reel during the feeding process, minimizing any lateral movement or misalignment. Similarly, the choice of the NEMA 17 stepper motor provided precise movement and sufficient torque to feed the reel accurately. These successful aspects of the design could be retained in future iterations.

In terms of the cutting mechanism, the use of scalpel blades was found to yield clean cuts and offered easy replacement when dull. This feature simplifies maintenance and ensures consistent cutting performance. However, it is essential to consider safety measures and blade longevity, especially if the module is intended for prolonged or heavy-duty operation.

3.4. Final Prototype Description

After a thorough evaluation of the shortcomings in the previous design, a new and improved version of the module was developed. Several key modifications were implemented to address the identified issues and enhance the overall functionality.

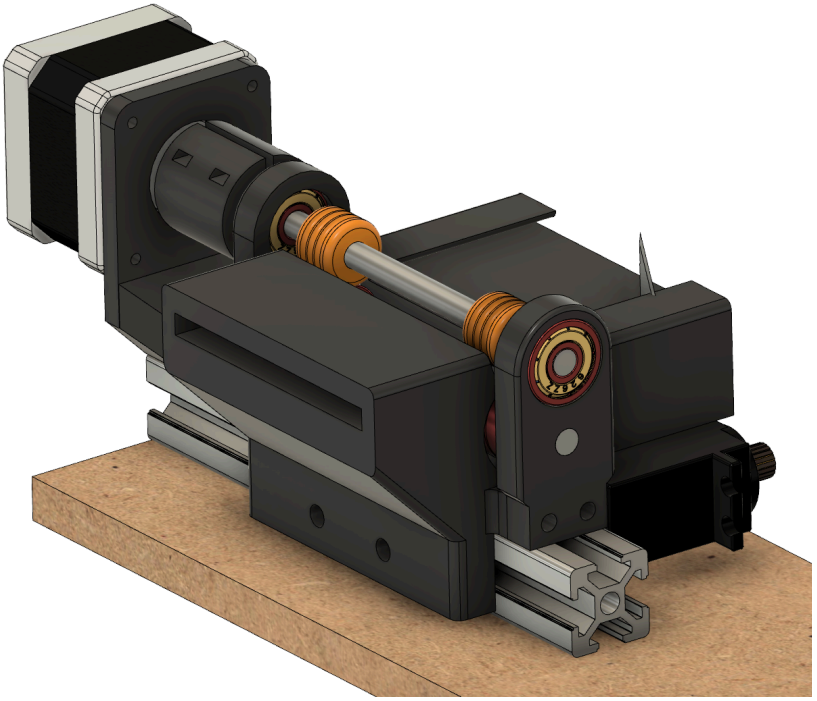


Figure 8: Feeder and Cutter Module Version 2

The feeder part underwent a significant redesign to eliminate the gradual curve that led to binding. By removing this feature, smoother movement of the reel was ensured, minimizing any interference or resistance. Additionally, the design now included specific mounts for the stepper motor, bearing assemblies, and a positive indexing feature. This indexing feature allowed for easy attachment of the module onto a 20x20mm Aluminium slot profile, making the entire feeder and cutter assembly fully modular. Each component could be replaced individually, enhancing flexibility and maintenance convenience.

Figure 9: Feeder Guide and Stepper mount Version 2

To overcome the problem of limited traction in the active roller part of the assembly, a cost-effective solution was implemented using rubber O-rings. These O-rings were carefully selected to provide increased traction. A nominal gap of 0.3mm was maintained between the O-rings and the opposing roller bearings, which was 0.2mm less than the nominal thickness of the reel. This deliberate gap allowed for compression of the rubber O-rings, ensuring improved grip on the reel. This design choice helped overcome any sliding resistance caused by the surface finish of the 3D-printed parts, resulting in more effective and reliable feeding of the components.

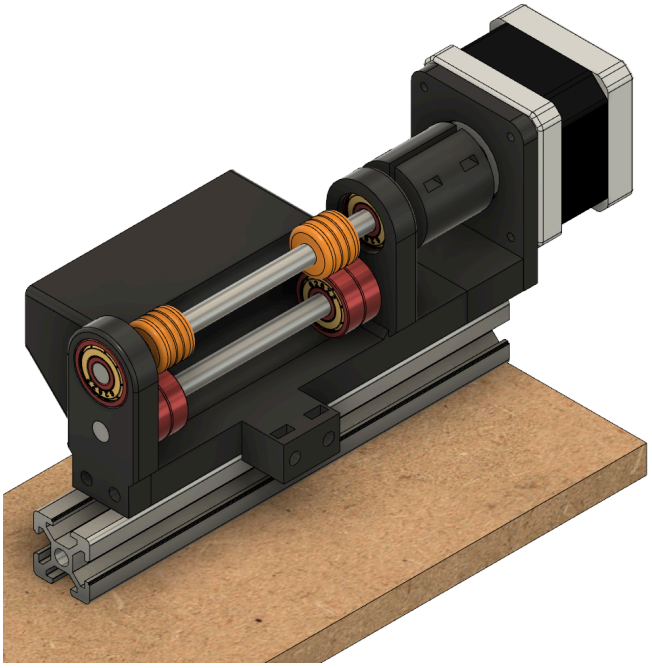


Figure 10: Active Roller Version 2

To address the issue of limited torque provided by the SG90 servo motor in the cutter section, a larger and more powerful servo motor, specifically the MG996R, was incorporated into the revised design. This servo motor offered a higher torque rating, enabling it to handle the cutting process more effectively. By upgrading to a servo motor with increased torque, the module could reliably and consistently cut through the reel without any performance limitations.

To ensure precision and minimize any play in the blades caused by the higher torque, a splined servo horn was introduced to connect the blade to the servo motor. This splined design provided a more secure and rigid connection, reducing any potential backlash or movement during the cutting action. Initially, a 3D-printed servo horn was attempted, but it was found that FDM 3D printers were unable to achieve the required level of precision for such fine features. As a result, an off-the-shelf injection-molded servo horn was modified to fit the servo motor and the blade securely.

In the process of revising the design, it was decided that incorporating a sensor to count the number of components passing through the module was not practical due to time and budget constraints. Therefore, the sensor mount was omitted from this iteration of the design. While a component-counting sensor could provide valuable data for quality control or process monitoring, the decision was made to prioritize other aspects of the module within the given limitations.

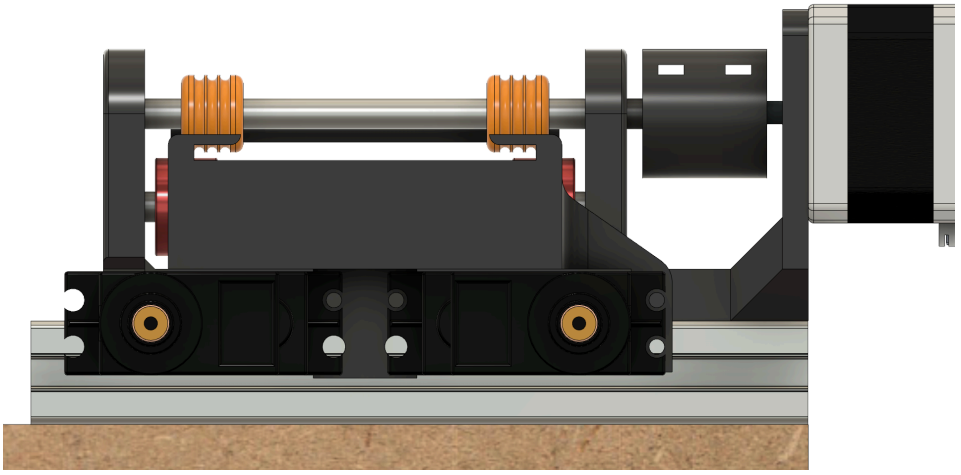


Figure 11: Cutting module Version 2

In summary, the new and improved version of the module addresses the shortcomings of the previous design. The feeder part was redesigned to eliminate binding, incorporating modular features and mounts for easy replacement and attachment. The use of rubber O-rings in the active roller part increased traction, ensuring reliable component feeding. Upgrading to the MG996R servo motor with higher torque resolved the cutting torque limitations, while a splined servo horn minimized blade play. Although the component-counting sensor was omitted due to time and budget constraints, the overall design improvements enhance the functionality and performance of the module.

3.5. Enclosure and HMI design

The device, which is used for cutting reels of components, poses several hazards to the user due to its blades and motors. Although the final device does not meet the European CE specification, we prioritize user safety. To address this concern and achieve an appealing design, we created an enclosure.

To provide a stable base for the enclosure design, we mounted the modules on an MDF plate using 20x20mm aluminum rails as guides. This setup ensured a solid foundation for the enclosure. The primary enclosure for the feeding and cutting modules was developed with a sleek and minimalist approach, incorporating a combination of 3D printed parts and laser-cut acrylic sheets.

One issue we encountered was the heat generated by prolonged use of the stepper motor, which could deform the PLA printed parts. To mitigate this, we positioned the stepper motors outside the printed parts to facilitate cooling through convection.

Figure 12: Acrylic Enclosure render with two modules

In order to prevent cutting hazards, we covered the output blades with acrylic sheets, leaving only a narrow 5mm slot for the components to pass through. This effectively prevents the insertion of fingers into the cutting area.

The top covering of the enclosure features a hole to accommodate the cable harness, which connects the various motors to the control and HMI unit to be mounted above. Originally, we intended to bend this feature from a 210x297mm sheet of 2mm thick acrylic. However, due to equipment limitations and the required large bend radius, we were unable to achieve successful bends. As a workaround, we 3D printed the part in separate pieces and used hot glue to join them together.

Figure 13: Acrylic Enclosure vs 3D printed enclosure

Additionally, we designed a control box to house the control electronics and the Human-Machine Interface (HMI) for operating the device. This control box is mounted on top of the existing enclosure and serves as a centralized location for the Emergency Stop switch and power cords. The control box is fully enclosed to minimize the risk of short-circuiting the electronics and ensures their safety.

Figure 14: Final Assembly with HMI and Electronics prototype vs Render

4. Electrical

4.1. Testing and Simulation

Topic written by : Sathvick Bindinganavale Srinath

Topic developed by : Sathvick Bindinganavale Srinath

To test our project initially, I utilized Tinkercad by Autodesk for simulation purposes. Tinkercad is a free-of-charge, online 3D modeling program that runs in a web browser. Within the software, we simulated the process of chopping the resistors by incorporating Arduino and some servo motors that were readily available in Tinkercad.

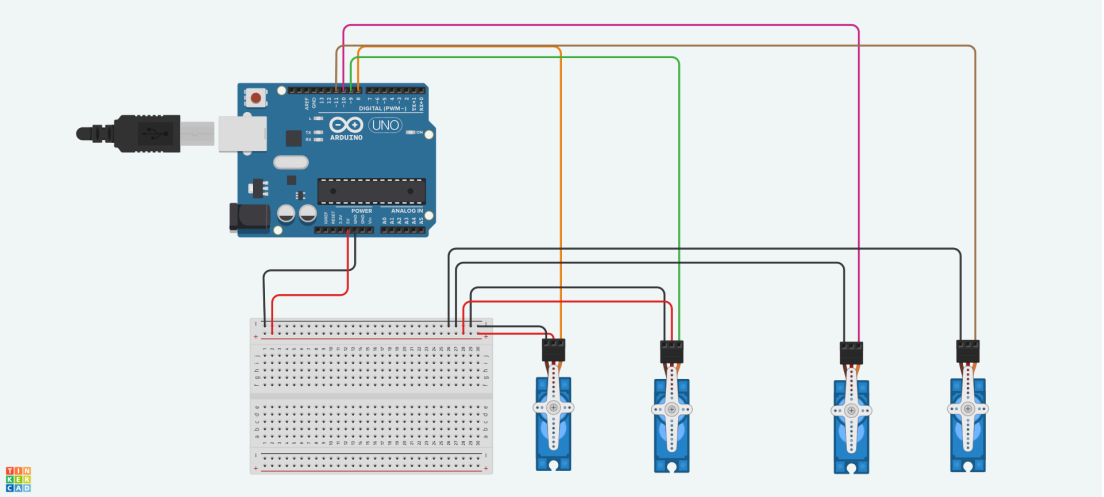


Figure 15 : Simulation of chopping using servo motors in Tinkercad [15]

Additionally, I also simulated the working of the stepper motor used on the machine using Tinkercad.

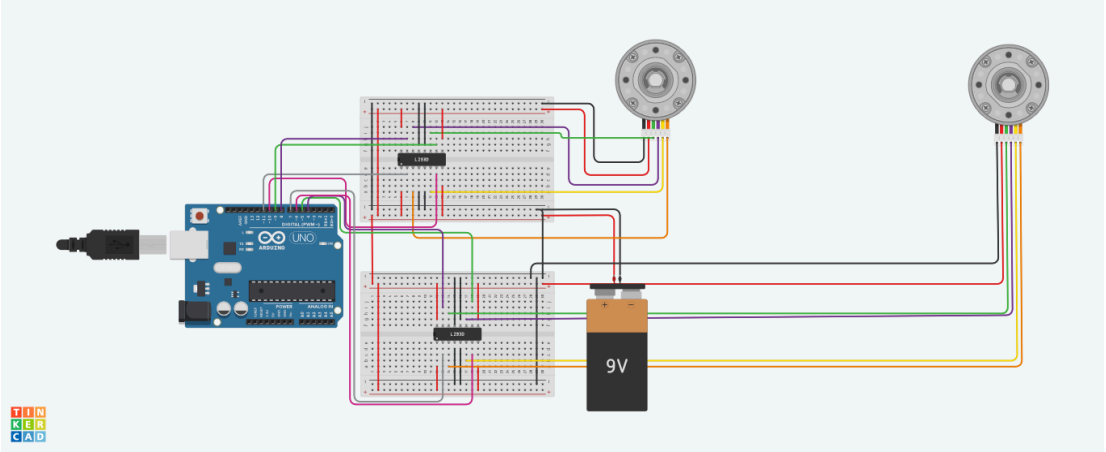


Figure 16 : Simulation of driving of stepper motors in Tinkercad [14]

One of the primary benefits of utilizing Tinkercad is the ability to program Arduino using an IDE within the software, allowing us to access all of the libraries and functionalities of an Arduino. However, the main limitation we encountered when using Tinkercad was the limited availability of electrical components for simulation purposes.

4.2. Wiring Plan

Topic written by : Sathvick Bindinganavale Srinath

Topic developed by : Sathvick Bindinganavale Srinath

In order to properly wire the components in your circuit, it is essential to have a wiring diagram. I utilized Fritzing [16], an open-source CAD software for designing electronics hardware, to create the wiring diagram for our project. One of the primary benefits of using Fritzing is the ability to create and customize electronic components according to our specific requirements. I was able to utilize components designed by other members of the Fritzing community to create our wiring plan. To hide the wiring in the machine, I used a heat sleeve to cover up the wires coming out of the components.

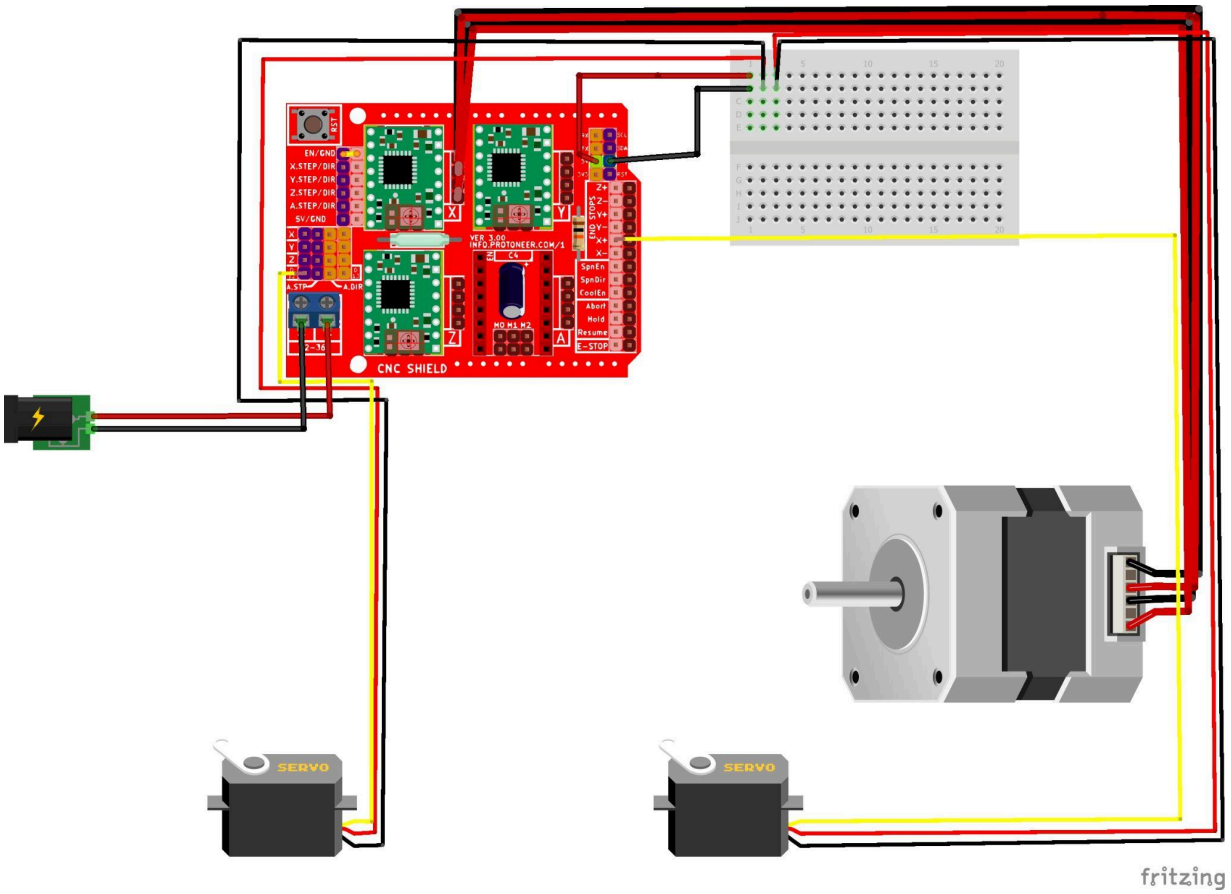


Figure 17 : Wiring plan in Fritzing

In the above figure, it has the wiring diagram only for 1 belt. The wiring is the same for belt 2.

In our project, to identify which belt is connected to which servo motors for the user to replace / repair a specific part, we have also color coded the wires of the servo motors to identify which color is connected to which pin on the CNC shield.

Color coding	Connection to CNC shield
Red	Limit Switch X+ / Digital pin 9
Purple	Limit Switch Y+ / Digital pin 10
Yellow	Limit Switch Z+ / Digital pin 11
Green	D12 / Digital / Digital pin 12

Figure 18 : Color coding of servo motors

4.3. Components Used

Topic written by : Sathvick Bindinganavale Srinath

Topic developed by : Sathvick Bindinganavale Srinath

In any technological machine, components are the building blocks that come together to create a functional and efficient whole. Components are the essential elements that allow devices to perform their intended tasks. Understanding the components used in a particular system is crucial for troubleshooting, maintenance and customization. In this section, we will explore the various components used in this project, their functions, and how they work together. We will dive into the intricate details of the components, their specifications, and the critical role they play in the overall performance of the system.

4.3.1. Seeeduino V4.3

Seeeduino V4.3 [9] is an arduino compatible board, which is based on ATmega328P MCU, Arduino UNO bootloader, and with an ATMega16U2 as a UART-to-USB converter. The board can be programmed via a micro USB cable. For our project, all our components are powered up using 5V supply voltage. We can use arduino IDE for programming the Seeeduino board also. We have to add a JSON file in the IDE for it to be compatible. In our project, we have powered up the arduino via the 5V pin. The main reason we chose an arduino UNO was that we can attach the CNC shield directly onto our board.

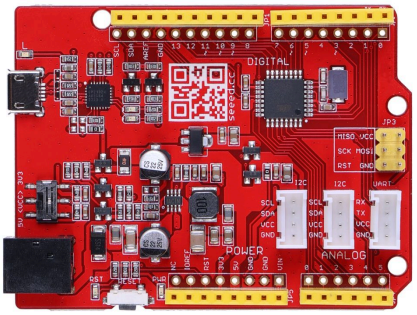


Figure 19 : Seeeduino V4.3 [8]

To provide 5V to almost all the components of the machine, we implemented a common 5V and a ground supply onto a perfboard, which powers almost all the components.

Features :

- ATMega328P microcontroller
- Arduino UNO bootloader

- 14 digital I/O pins
- 6 PWM channels
- 6 analog pins
- ISP header
- Micro USB programming and power supply
- 3.3V/5V system operating power switch
- Flash memory : 32 KB
- RAM : 2KB
- EEPROM : 1 KB
- Clock speed: 16 MHz

4.3.2. CNC Shield V3

CNC Shield V3 is an open source hardware used to control stepper motors. It allows the user to control 4 stepper motors simultaneously. The shield uses stepper motor drivers, like A4988 or DRV8835 to control the motors. Stepper motors are connected via 4 - pin connectors. The main reason for us to consider the CNC shield is to control the stepper motors easily with the stepper motor drivers and the libraries associated with it.

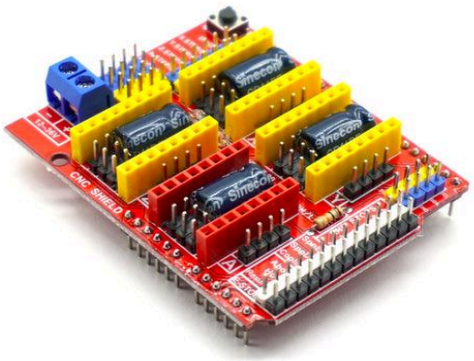


Figure 20 : CNC Shield V3 [10]

In our project, we have used a power supply providing a voltage of around 12V for the CNC shield. In the CNC shield, we have the same digital and analog pins present as we find on the arduino and have used limit switches X+, Y+, Z+ and D12 for connecting the servo motors.

Pin on CNC shield	Corresponding pin connected with arduino
X Step	Digital pin 2
X Direction	Digital pin 5
Y Step	Digital pin 3
Y Direction	Digital pin 6
Z Step	Digital pin 4
Z Direction	Digital pin 7
Enable	Digital pin 8
Limit switch X+ / X-	Digital pin 9
Limit switch Y+ / Y-	Digital pin 10
Limit switch Z+ / Z-	Digital pin 11
D12	Digital pin 12
D13	Digital pin 13
Abort	Analog pin A0
Hold	Analog pin A1
Resume	Analog pin A2
Cool En	Analog pin A3

Figure 21: Arduino CNC Shield pin connection

The A4988 stepper motor driver is capable of microstepping at 2X, 4X, 8X, or 16X the full step resolution, configured using pull-up jumpers located beneath each driver module on the CNC shield board(pins M0, M1 and M2 respectively). In our project, we have implemented microstepping at a resolution of 1/16 th of a step.

M0	M1	M2	Microstep Resolution
No	No	No	Full step

Yes	No	No	Half step
No	Yes	No	Quarter step
Yes	Yes	No	Eighth step
Yes	Yes	Yes	Sixteenth step

Figure 22 : Microstepping setup

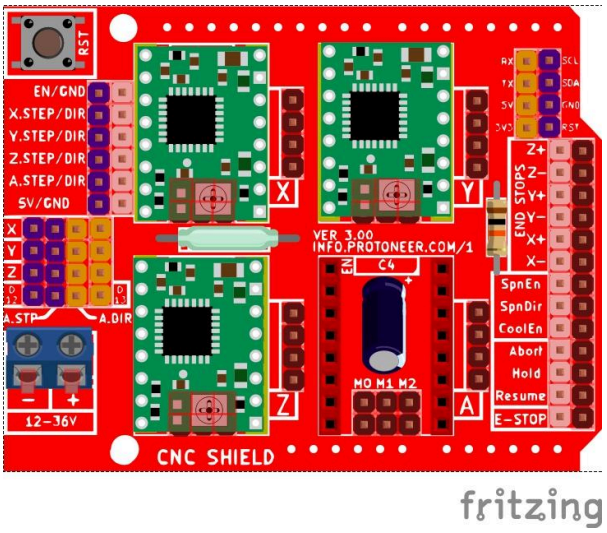


Figure 23: CNC shield with Stepper motor drivers attached in Fritzing

Features :

- 4 - Axis support (X, Y, , Z, A - can duplicate, X, Y, Z or do a full 4th axis with custom firmware)
- 2 end stops for each axis (6 in total - each axis pair shared by the same I/O pin)
- Spindle enable and direction connection
- Coolant enable connection

4.3.3. A4988 Stepper motor driver

The A4988 is a comprehensive microstepping motor driver that incorporates a built-in translator for simplified operation. It is specifically designed to drive bipolar stepper motors and supports various step modes, including full, half, quarter, eighth, and sixteenth steps. With an output drive capacity of up to 35V and a tolerance of 2A, the A4988 is capable of effectively powering stepper motors. Additionally, it features a fixed off-time

current regulator that can operate in either slow or mixed decay modes, enhancing its versatility in motor control applications..

Features :

- Mixed and slow current decay modes
- Crossover current protection
- 3.3V and 5V compatible logic supply
- Five selectable step modes : full, ½, ¼, ⅛, 1/16

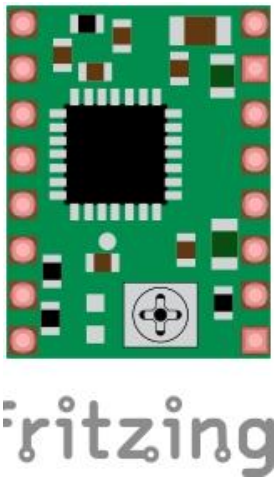


Figure 24: A4988 stepper motor driver in Fritzing

With this stepper driver, we can control our stepper motor using the

Step pin : Produces a short impulse which signals that the motor should do a step with predefined length.

Direction pin : Determines the direction of the movement. Example : `HIGH` for moving forward and `LOW` for moving backwards.

Enable pin : Switch the motor drive outputs on / off.

4.3.4. NEMA 17 Stepper motor

To facilitate the feeding of the resistor reel in our machine, we have integrated a NEMA 17 stepper motor, commonly found in many 3D printers for controlling axes. This motor is powered by a 12V voltage supply. While it can operate at lower voltages, it should be noted that the torque will be reduced accordingly. We opted for this stepper motor primarily due to its step angle of 1.8° and high torque characteristics, making it well-suited for effectively feeding the resistor reel.

Figure 25: NEMA 17 Stepper motor in Fritzing

Features :

- Rated voltage : 12 V
- Current : 1.2A at 4V
- Step angle : 1.8°
- Number of phases : 4
- Steps per revolution : 200

4.3.5. MG996R Servo motor

To achieve precise cutting of the resistor reel, we have employed servo motors equipped with a blade. Initially, we utilized an SG90 servo motor for this purpose. However, during the testing phase, we recognized the need for a servo motor with higher torque to effectively cut through the resistor reel tape. Consequently, we determined that the MG996R servo motor [11] was the optimal choice for our application, given its enhanced torque capabilities.

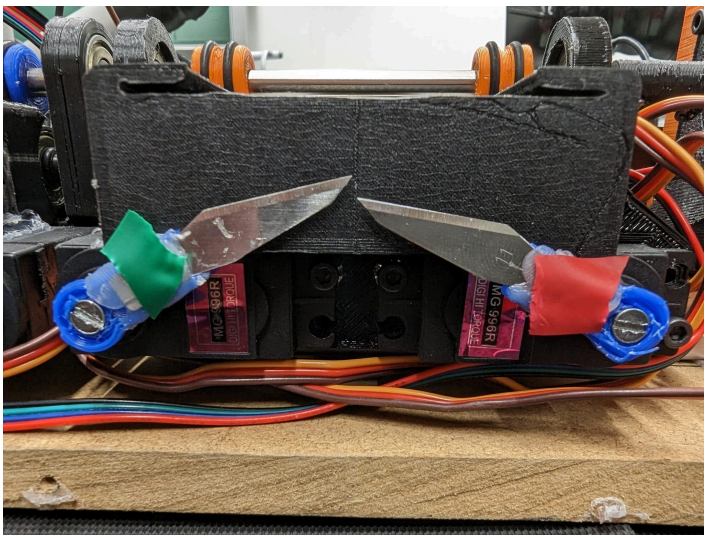


Figure 26: Blades attached to MG996R servo motor horns

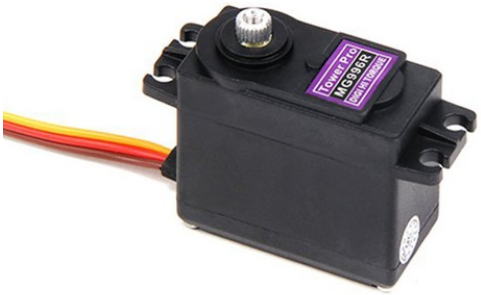


Figure 27: MG996R servo motor [12]

Features :

- Operating voltage : 4.8V - 6.6V
- Operating speed : 0.19 sec/60° (4.8V) ; 0.15 sec / 60° (6V)
- Rotates 180°

4.3.6. Buck convertor

Instead of powering the Arduino and the CNC shield separately, we opted for a single power source. However, a challenge arose due to the CNC shield requiring a 12V input voltage while the Arduino operates with around 5V. To resolve this issue, we incorporated a buck converter, also known as a step-down converter.

A buck converter is a DC-DC converter that reduces the voltage from the input to its output. In our setup, the input of the buck converter receives a voltage of approximately 12V from the power supply and supplies it to the CNC shield. The output of the buck converter provides around 5V, which is then distributed to all the other components in the system.

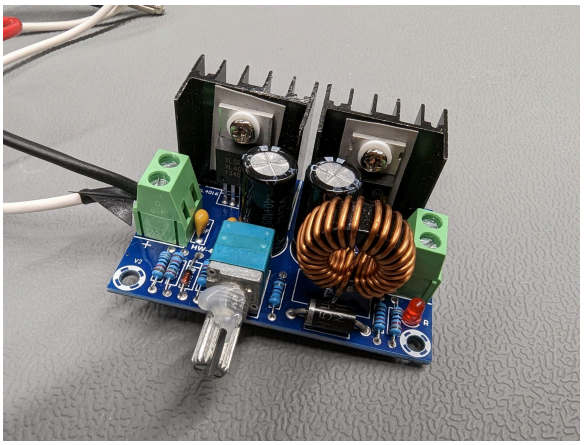


Figure 28: Buck Convertor

4.3.7. Emergency stop button

An emergency stop button, often referred to as an E-stop, is a safety device designed to quickly halt or interrupt the operation of a machine or process in emergency situations. It is typically a prominent, easily accessible button or switch that, when pressed, instantly stops the machinery, cutting off power and preventing any further movement or potential hazards. The purpose of an emergency stop button is to provide an immediate and reliable means of shutting down equipment in critical scenarios, such as when there is a risk of injury, damage, or an imminent danger. Its presence serves to enhance safety in various industrial, commercial, and even residential settings, ensuring that operators can swiftly respond to emergencies and mitigate potential risks.

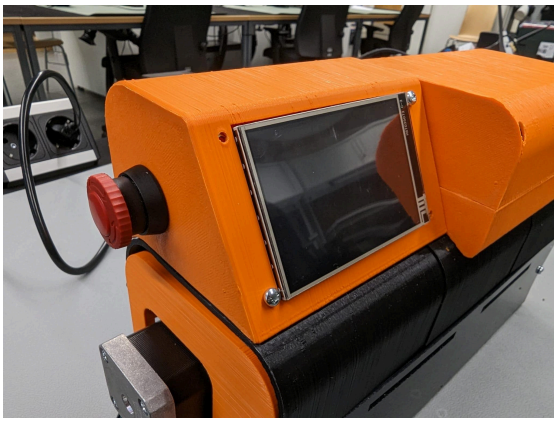


Figure 29: E- stop button installed on the device

In our project, we have connected the E-stop button to a normally closed switch positioned between the power source and the buck converter. When the button is pressed, the switch opens, interrupting the power supply to the device.

Figure 30: Emergency stop button

4.4. Electrical Design

Topic written by : Samrat Shantesh Ukkali

4.4.1. Introduction

This part of the report focuses on the development of the electrical design of the resistor reel cutting machine, including the use of the Wokwi online simulator to test the circuit design and the subsequent design of the Printed Circuit Board (PCB).

4.4.2. Wokwi Online Simulator

The Wokwi online simulator was a crucial tool in the design process of the PCB. It enabled the testing and evaluation of the circuit design before the design of the PCB. The simulator provided a virtual environment where the functionality of the servo motors, A4988 stepper drivers, stepper motors, Nextion display, emergency stop button and the functionality of the Arduino Uno was tested.

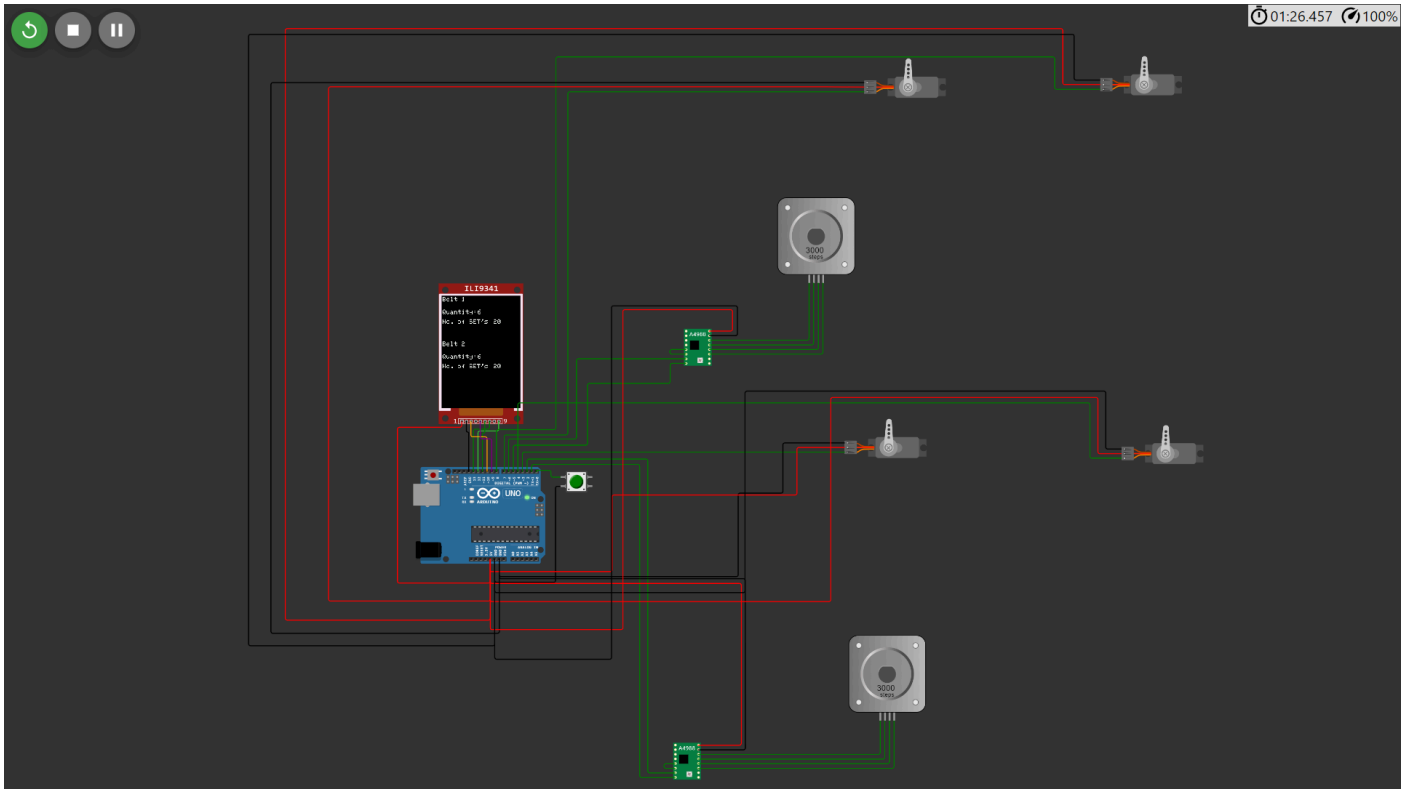


Figure 31: Wokwi Online Simulator

Link for Wokwi Online Simulation: (https://wokwi.com/projects/362191058017428481)

4.4.3. Schematic

The schematic was developed in KiCAD, an open-source software that facilitates the design of electronic circuits. The schematic represents the circuit design and serves as a guide for the placement of the components on the PCB.

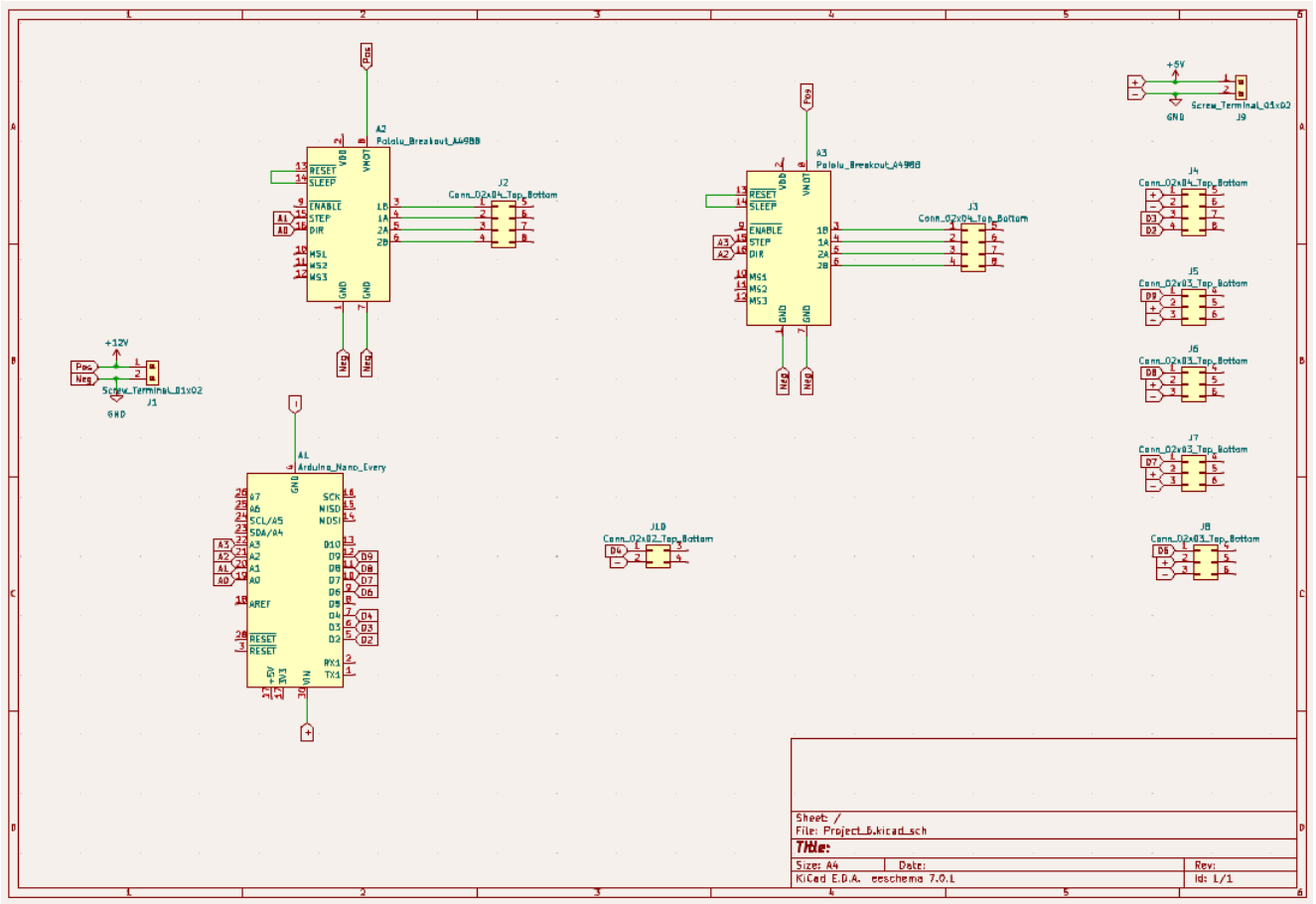


Figure 32: Schematic

4.4.4. PCB Design

The PCB design consists of three layers, namely the top copper layer, bottom copper layer, and silk screen layer. The top and bottom layers of the PCB serve as the mounting points for the components. The silkscreen layer, on the other hand, provides visual cues and indicators for the user (see Figure 33). KiCAD was used to design the PCB, which was straightforward and easy to use.

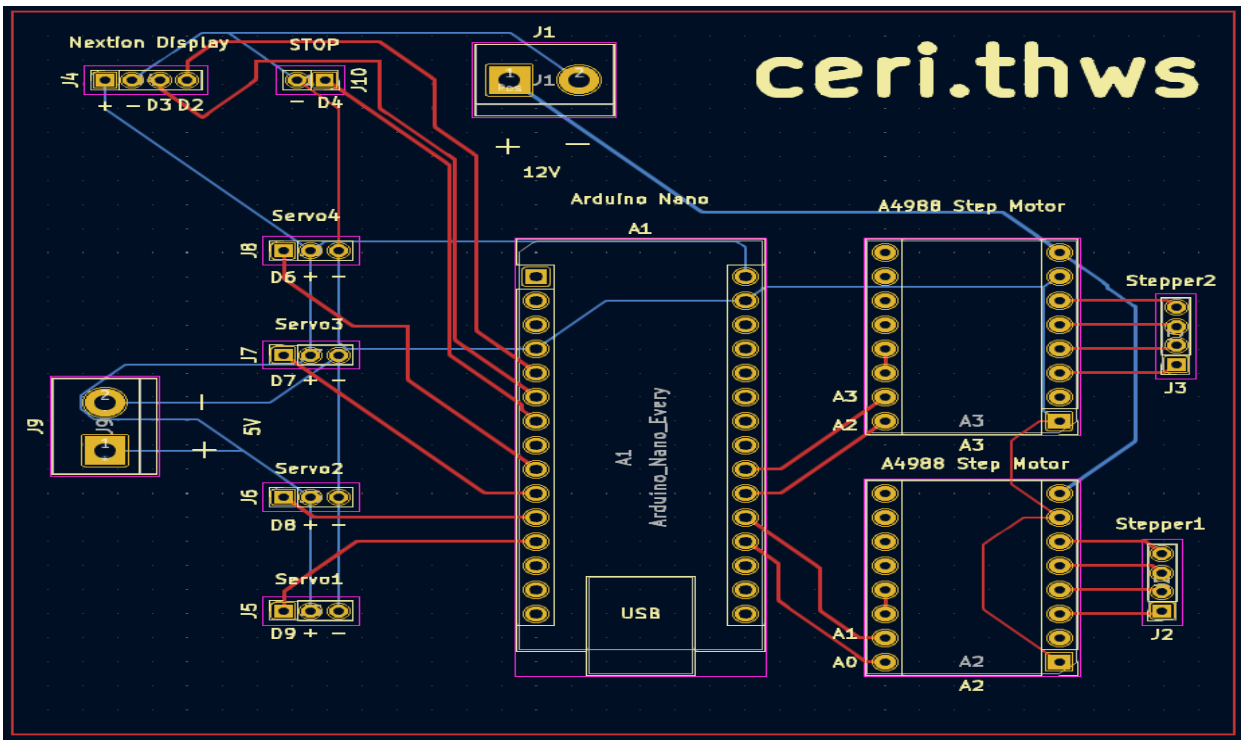


Figure 33: PCB Layers

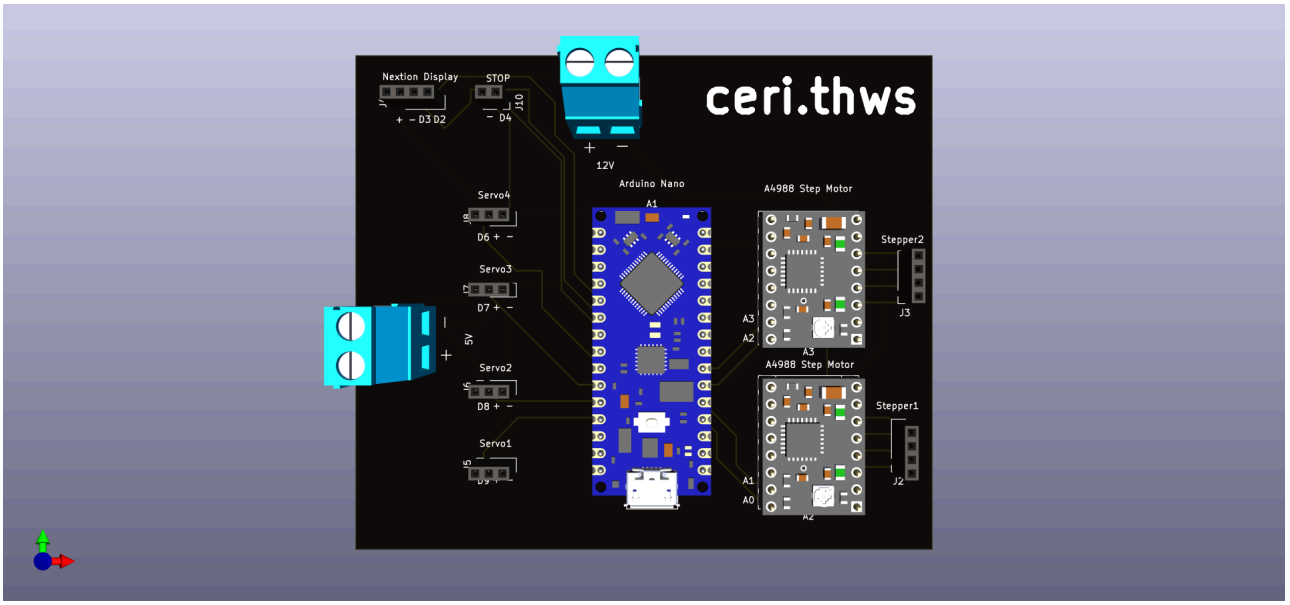


Figure 34: Printed Circuit Board (PCB)

4.4.5. Components driven by PCB

The PCB provides the functionality for several components, including (see Figure 34):

Servo motors: Used for the horizontal and vertical movement of the cutting blade. They are controlled by four digital pins (D6, D7, D8, and D9).

Stepper motors: Used for the rotation of the resistor reel. They are controlled by two A4988 Step Motor Drivers and two respective analog pins (A0 & A1) for Stepper1 and (A2 & A3) for Stepper2.

Nextion display: A user interface that displays relevant information about the machine, such as the current operation and the number of resistors cut. It is controlled by two digital pins (D2 & D3), providing feedback to the user.

Emergency stop button: A safety feature that stops the machine in case of an emergency. It is controlled by one digital pin D4.

Power Supply: The PCB is powered by two screw terminals, which are connected to a 5V and 12V power supply, respectively. The 5V power supply is given to Arduino Nano, servo motors, and Nextion display. Additionally, the 12V power supply is given to Stepper Motor Drivers. These power sources are critical to the functionality of the PCB and, therefore, the overall performance of the resistor reel-cutting machine.

4.4.6. Conclusion

In conclusion, the Wokwi online simulator played a crucial role in the design process of the PCB for the resistor reel-cutting machine. It provided a virtual environment for testing and evaluating the circuit design before the design of the PCB using KiCAD.

5. Software

5.1. Arduino code for actuation

Topic written by : Sathvick Bindinganavale Srinath

Topic developed by : Sathvick Bindinganavale Srinath

5.1.1. Introduction

This section contains the code for the actuation and feeding of the resistor reels. The code was written for the Arduino UNO board. The code is also available in the [2]Github repository. To simplify the process of calling the functions as needed, all the actuation code for the machine is organized into functions.

5.1.2. Setup and Installation of packages

The code for the Arduino UNO was written in Arduino IDE [13], which is necessary for verifying and uploading the code to the board. The board we used in the machine was Seeeduino V4.3, an Arduino compatible board. For the Seeeduino to work, we need to install third party drivers. Go to File -> Preferences, copy the .JSON link [18] and paste in the additional board managers URLs field.

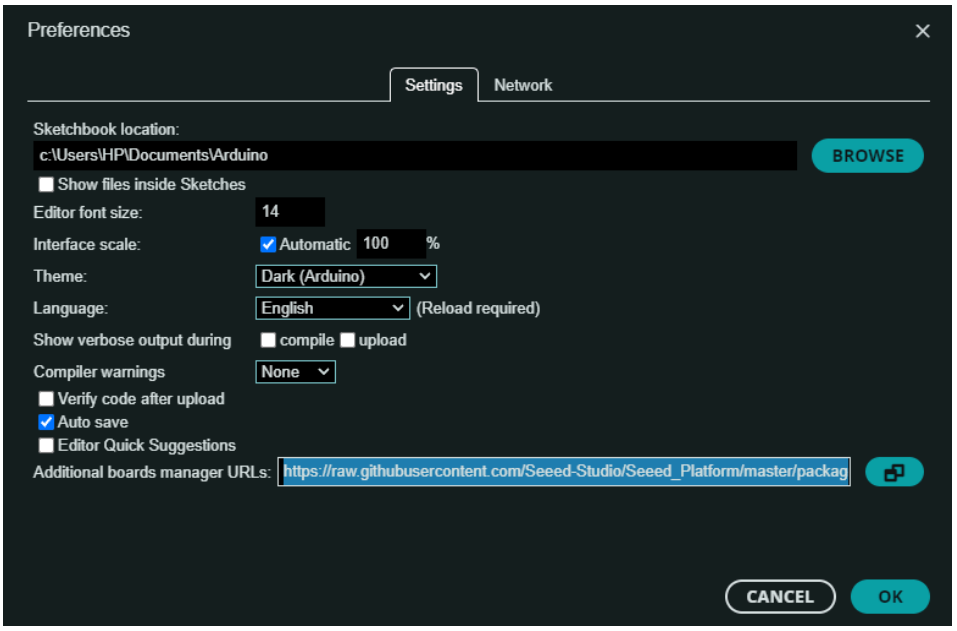


Figure 35: Preferences window in Arduino IDE

In the board manager, choose Seeeduino V4(Atmega328P) and select the appropriate COM port. For more information on adding boards in Arduino IDE, please see the references section [17].

5.1.3. Calibration of the resistor reel

During cutting of the resistor reel, accuracy is crucial in any device. For maintaining the accuracy in our device, initially we had installed a cny70 infrared sensor, which is usually seen on line follower robots. The IR sensor will detect the number of resistors and give feedback to the machine when to start chopping.

Figure 36: CNY70 IR sensor [19]

One limitation we encountered was related to the enclosure near the blades, where the sensor faced difficulty in identifying the resistors inside. This issue arose due to the inadequate distance between the design of the feeder module and the resistors, which made it challenging to distinguish the distance between the two accurately. To address the accuracy issue, we introduced a visual calibration feature that requires user input. The user is responsible for calibrating the resistor reel by ensuring that only one resistor is positioned outside the machine's housing. This calibration process helps establish a reference point for accurate cutting.

Additionally, we incorporated a resistor feeding functionality within the calibration feature. The user can utilize this feature to feed the resistor reel into the machine at different speeds. We have provided three speed options: slow, medium, and fast. The slow speed feeds in one resistor at a time, the medium speed feeds in approximately five resistors at a time, and the fast speed feeds in approximately ten resistors at a time. These speed options offer flexibility and convenience to the user based on their specific requirements. The code for the calibration process can be found in the corresponding section 10.1.1. of the document. Detailed information about the calibration procedure is provided in the dedicated section 6.3.3. as well.

5.1.4. Counting the number of resistors

In our system, the user provides input through the Nextion display indicating the desired number of resistors. This count value is then passed as input to the function responsible for counting up the resistors. Counting the resistors accurately posed a challenge as we did not employ any specific sensor for detection. However, we utilized the DIN-IEC-286-1 standard [20], which ensures a consistent distance of 10 mm between the resistors and other components on the taped reel, with a tolerance of ±0.5 mm. By

employing trial and error, we determined the appropriate speed for moving one resistor at a time. The movement of the stepper motors for the calibration was incorporated using the library Stepper, a library provided by Arduino. To calculate the total time required for the desired number of resistors, we multiplied the count value provided by the user with the speed required for moving one resistor. This approach allowed us to estimate the time and accurately count the specified number of resistors. The code for counting the number of resistors is given in the section 10.1.3.

5.1.5. Chopping of resistors

Once the calibration and resistor counting processes are complete, the next step is to cut the resistor reel. To achieve this, we programmed the MG996R servo motors, with their motor horns connected to scalpel blades, to serve as the chopping mechanism. The process of chopping the resistor reel is simple and straightforward. Firstly, we installed the servo motors and calibrated their starting and end points. We ensured that the motors would move approximately 90° towards the resistor reel, allowing them to cut through it effectively. To control the servo motors, we utilized the Servo library, a library provided by Arduino. Once the chopping is performed, the motor will return to its original position, ready for the next operation. The code for the chopping of resistors is explained in more detail in the section 10.1.4.

Figure 37: Blades attached to the motor for chopping

5.2. Nextion Arduino Code

Developed by: Aniketh Padmakar
Topic written by: Aniketh Padmakar

5.2.1. Nextion Package for Arduino

First the Nextion Package has to be installed with all the libraries in Arduino. The library file can be installed from the Nextion website at [4] Github repo. After downloading this the zip file has to be extracted where all the other libraries are and then has to be renamed to “Nextion”. A few lines of the code in Nextion have to be changed and commented out in the NexConfig file. The instructions on how to install and make the changes can be found in [6] Youtube video.

5.2.2. Arduino Code for Nextion

Once the packages have been successfully imported, as demonstrated in section 10.1.2.1, the initialization process for all page elements that possess touch-based interactions on the screen commences. Each element across different pages necessitates individual initialization, taking into account the specific type of Nextion element it represents. During initialization, it is crucial to have knowledge of the page number, object ID, and object number associated with each element, as these parameters are essential for the initialization procedure.

The Home Page, being the initial page, features two elements that require tracking of their touch actions: the Belt1 and Belt2 on and off buttons. As these buttons are Dual state button elements and located on the first page, their initialization process is outlined in section 10.1.2.2. The integer variables "bt1s" and "bt2s" are employed to store the current On/Off states of belts 1 and 2, respectively. They are initially set to "1," indicating the On state. The functions depicted in section 10.1.2.6 are triggered when the Dual State Buttons are touched on the display, resulting in the modification of the tracking variables "bt1s" and "bt2s" to reflect the corresponding changes. These alterations can also be monitored through the Serial Monitor on Arduino by utilizing the print statements included within these functions.

The Number Pad page does not entail any elements that necessitate communication with the Arduino, as its primary function is to serve as an input method for the user. In contrast, the Calibration page contains the highest number of elements that require tracking when touched on the display. The page includes several key elements for belt selection and calibration speed adjustment. The primary element is the Dual state button, which enables users to choose the desired belt for calibration. The current state of this button is stored in the integer variable "which_belt," allowing the system to track the user's selection accurately.

Additionally, there are three buttons that must be initialized to enable users to select the calibration speed: fast, medium, and slow. The corresponding integer variable "speed" keeps track of the chosen speed, with values ranging from 675 to 6400. These values represent the specific number of steps (micro-stepping) that the stepper motor will move during the calibration process.

To facilitate speed selection, touch-bound functions associated with setting the speeds can be found in section 10.1.2.12. These functions ensure that users can easily adjust the speed based on their requirements, enhancing the overall calibration experience.

Once the user has selected the belt and speed for calibration, they can utilize the arrow buttons displayed on the interface to initiate calibration in the desired direction and at the chosen speed. The touch-action-controlled function, as shown in section 10.1.2.8, triggers the corresponding actuation function based on the user's selected parameters. This

flexibility allows users to freely modify these parameters as needed, ensuring a seamless and customizable calibration experience.

To ensure proper initialization and functionality, all the aforementioned page elements and variables have been correctly initialized, as demonstrated in section 10.1.2.3. Alongside the previously mentioned elements, there are a few additional crucial components that have been initialized on the same page. The most significant among them is the start button, responsible for initiating the cutting process. Furthermore, there are four variable elements that store the values inputted by the user on the Home page.

To capture and store these values, four integer variables have been initialized in Arduino: "b1_sets," "b1_comps," "b2_sets," and "b2_comps." These variables correspond to the number of sets and components for each belt. The values from the Nextion display are transferred to these variables using the 32-bit unsigned integer "number" variable, as demonstrated in section 10.1.2.7. This data transfer is facilitated by utilizing the functions available in the imported Nextion library. Once the user selects the start button, the transfer process is initiated to obtain the inputted values. Subsequently, the resistor-cutting functions are invoked based on these values. The progress of the cutting process for each belt is tracked by two integer variables, "b1_done" and "b2_done," which are initialized in section 10.1.2.4. These variables effectively monitor and update the status of the cutting progress for their respective belts.

Once the user selects the start button on the calibration page, the display transitions to the Cutting Progress page. While this page doesn't require communication with the Arduino, it does feature a Stop button that redirects the display to the Stop confirmation page. On the Stop confirmation page, there is a crucial button, the Yes button, which communicates with the Arduino. This button is initialized in section 10.1.2.4 and is tracked by the integer variable "stop". When the Yes button is touched, the touch-bound function is invoked, resulting in a change of the tracking variable's value to 1. This action interrupts the cutting process and resets all the tracking variables, as demonstrated in section 10.1.2.9.

The Nextion elements that have been initialized and require tracking for touch actions need to be entered into an array as pointers, ending with "NULL" as depicted in 10.1.2.5. This array serves the purpose of indicating the elements that need to be monitored for touch events. The array is then passed to a Nextion function, informing it about the specific elements that should be tracked.

Now, these touch-sensitive elements need to be associated with their respective touch-bound functions, as discussed earlier. This binding process is carried out within the setup function of Arduino, as demonstrated in 10.1.2.10. It is crucial to determine whether the elements send their component IDs when touched or when released, as this distinction affects the function calls and the association between the element and the function.

In the loop function of Arduino, we pass the array we created containing pointers to the touch-sensitive elements to the function responsible for tracking them, as shown in 10.1.2.11. This ensures that the elements are continuously monitored for any touch actions. Additionally, within this loop function, there is an "if" statement that checks if the cutting process has been completed. If so, it transitions to the Cutting Done page, resets all the tracking values, and returns the setup to its initial state, ready for the next cutting cycle.

6. GUI Software

Developed by: Aniketh Padmakar

Topic written by: Aniketh Padmakar

6.1. Nextion HMI GUI Touch Display

Nextion is a China-based company renowned for delivering cutting-edge solutions in Human Machine Interface (HMI) which are Arduino, Raspberry PI, and systems with ESP8266 microcontroller compatible. Their offerings comprise an onboard processor, memory touch display, and Nextion Editor software, specifically designed for the development of HMI graphical user interfaces (GUIs). With meticulous consideration of our project's timeframe and requirements, we opted for a Nextion display. Given the project's eight-week duration, we aimed to secure a touch display that could be seamlessly programmed using an integrated development environment (IDE), making Nextion the ideal choice. Furthermore, their website offers a hassle-free, complimentary installation of the IDE.

Nextion Intelligent NX4827P043-011R is the display we have used for the Resistor Reel Cutter Machine, which has the following features:

- Resolution: 480 x 272
- Diagonal: 4.3 inches
- Touch Panel: Resistor
- Interface: UART

The display also comes with an adapter for USB power supply and connection cables which are finished with plugs.

The subsequent sections provide a comprehensive explanation of the .tft code, which was meticulously crafted utilizing the Nextion IDE's invaluable assistance. The Nextion IDE HMI file and the TFT file to be uploaded onto the Nextion can be found in [2] GitHub repo.

6.2. Using the Nextion IDE

The Nextion IDE stands out for its user-friendly and intuitive nature. It can be downloaded at [5] for free. When starting a new project, the initial step involves selecting the specific Nextion model being used. Following this, the display direction, determining the orientation, is chosen. This sets the foundation for a blank file, consisting of a single empty page.

The initial task involves selecting an appealing background for the display. This entails downloading the desired image and resizing it to match the display size, which can be conveniently achieved using [3] for free. Additionally, it is recommended to create various fonts of different sizes to cater to different text fields on the display.

With the groundwork set, the user is ready to incorporate various types of elements into their project. An essential consideration is to determine whether an element should trigger an action in the Arduino. In such cases, it is crucial to select the option to send the component ID, either upon touch or upon release. Clarifying the specific timing is essential, as it must be specified when the component sends its ID to the Arduino. During the initialization process in Arduino, it is imperative to be aware of the page number, component ID, and component name associated with each element. If some value is to be extracted from the Nextion by Arduino that code has to be written in the Nextion as well.

By following these aforementioned steps and adhering to the provided tips, users can effectively leverage the Nextion IDE without encountering any difficulties. Once the HMI layout and code are ready the user has to download the .tft file and upload it onto the Nextion display via an SD card. The youtube video [6] is also helpful for this.

6.3. Pages Layout

Figure X illustrates the Pages within the HMI display, while the arrows indicate the transitions between them. Each page's specific functionality is outlined within the accompanying boxes, and their detailed explanations will be presented in the subsequent sections. When designing the pages, careful consideration was given to ensuring that there were no dead-end pages, where the user would become trapped or unable to navigate further.

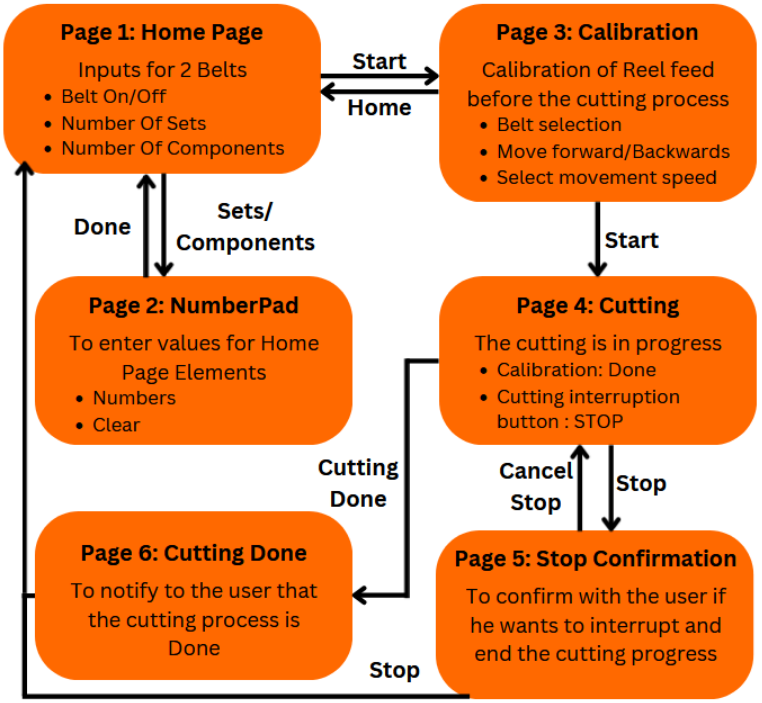


Figure 38: Pages Flowchart

6.3.1. Page 1: Home Page

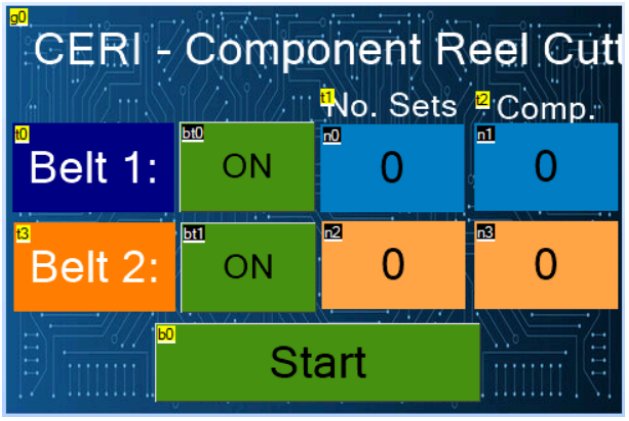


Figure 39 : Home Page on Nextion
Page ID: 0/ Home
Page Elements: 13

As indicated in Figure X, the home page serves as a platform for user input regarding the activation status of Belt 1 and/or Belt 2. Subsequently, depending on the utilization of the respective belts, the user is able to enter the desired quantities of sets and components for each belt through the number pad page, visually presented and explained in section 6.3.2.

The components featured on this page are elucidated through the assistance of the table provided below. In case there is accompanying code for a particular page component, the reference to that code can be found in the appendix. The scope of each component is explicitly stated, but only if it pertains to a global scope.

Sl.No.	Component Type (variable scope)	Component ID	Object Name	Component Function	Element code: Appendix Reference
1	Picture	1	p0	Page Background	N/A
2	Scrolling Text	2	g0	Device Title: “CERI - Component Reel Cutter”	N/A
3	Button	3	b0	Start button that transitions to Calibration page	10.2.1.1.
4	Number (Global)	4	n0	1. Touch to enter the number of sets for belt 1. 2. Transitions to numPad page while changing text element t12 in numpad page to "B1:Sets". 3. Save Belt 1 and Belt 2 on/off status into two variables on the numPad	10.2.1.2.

				page.	
5	Number (Global)	9	n1	1. Touch to enter the number of components for belt 1. 2. Transitions to numPad page while changing text element t12 in numpad page to "B1:Comp". 3. Save Belt 1 and Belt 2 on/off status into two variables on the numPad page.	10.2.1.3.
6	Number (Global)	12	n2	1. Touch to enter the number of sets for belt 2. 2. Transitions to numPad page while changing text element t12 in numpad page to "B2:Sets". 3. Save Belt 1 and Belt 2 on/off status into two variables on the numPad page.	10.2.1.4.
7	Number (Global)	13	n3	1. Touch to enter the number of components for belt 2. 2. Transitions to numPad page while changing text element t12 in numpad page to "B2:Comp". 3. Save Belt 1 and Belt 2 on/off status into two variables on the numPad page.	10.2.1.5.
8	Text	5	t0	Text box: “Belt 1:”	N/A
9	Text	10	t3	Text box: “Belt 2:”	N/A
10	Dual State Button (Global)	6	bt0	1. To turn Belt 1 On/Off. 2. Text displayed on the button “ON”/”OFF” dependent on the button state.	10.2.1.6.
11	Dual State Button	11	bt1	1. To turn Belt 2 On/Off. 2. Text displayed on the	10.2.1.7.

	(Global)			button “ON”/”OFF” dependent on the button state.	
12	Text	7	t1	Text box: “No. Sets”	N/A
13	Text	8	t2	Text box: “Comp.”	N/A

The Component IDs of the following elements are sent to Arduino:

- Dual State Button bt0: On Touch Release
- Dual State Button bt1: On Touch Release

The Arduino can now determine what state the user wants the respective belts as every time the element is touched the ID is sent. Variables in Arduino keep track of this change.

When the user interacts with the Start button “b0” on the calibration page, the values of the numbers n0, n1, n2, and n3 are transmitted to the Arduino. This occurs because the calibration page marks the point at which the user finalizes the values for the number of sets and components.

6.3.2. Page 2: Number Pad

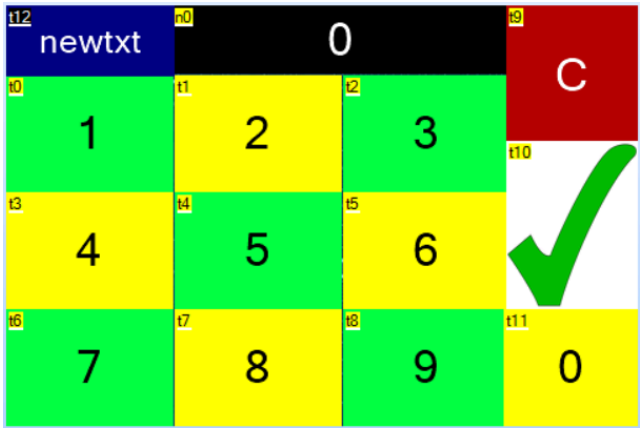


Figure 40 : Number Pad Page on Nextion
Page ID: 1/numPad
Page Elements: 17

The Number Pad page, depicted in the aforementioned figure, facilitates the input of the number of sets and components for each belt. After the user has entered the desired values in the selected field, they are directed back to the Home Page. Additionally, a clear button is available to rectify any incorrect numbers that may have been entered.

The components featured on this page are elucidated through the assistance of the table provided below. In case there is accompanying code for a particular page component, the reference to that code can be found in the appendix. The scope of each component is explicitly stated, but only if it pertains to a global scope.

Sl.No.	Component Type (variable scope)	Component ID	Object Name	Component Function	Element code: Appendix Reference
1.	Picture	1	p0	Page Background	N/A
2.	Number	2	n0	Stores the value entered by the user Data Type: Integer	N/A
3.	Variable (Global)	3	va0	To store Belt 1 On/Off state	N/A
4.	Variable (Global)	17	va1	To store Belt 2 On/Off state	N/A
5.	Text	4	t0	1. Text Box: “1” 2. When touched by the user multiply the value of number n0 by 10 and adds 1	10.2.2.1.
6.	Text	5	t1	1. Text Box: “2” 2. When touched by the user multiply the value of number n0 by 10 and adds 2	10.2.2.2.
7.	Text	6	t2	1. Text Box: “3” 2. When touched by the user multiply the value of number n0 by 10 and adds 3	10.2.2.3.
8.	Text	7	t3	1. Text Box: “4” 2. When touched by the user multiply the value of number n0 by 10 and adds 4	10.2.2.4.
9.	Text	8	t4	1. Text Box: “5” 2. When touched by the user multiply the value of number n0 by 10 and adds 5	10.2.2.5.
10.	Text	9	t5	1. Text Box: “6” 2. When touched by the	10.2.2.6.

				user multiply the value of number n0 by 10 and adds 6	
11.	Text	10	t6	1. Text Box: “7” 2. When touched by the user multiply the value of number n0 by 10 and adds 7	10.2.2.7.
12.	Text	11	t7	1. Text Box: “8” 2. When touched by the user multiply the value of number n0 by 10 and adds 8	10.2.2.8.
13.	Text	12	t8	1. Text Box: “9” 2. When touched by the user multiply the value of number n0 by 10 and adds 9	10.2.2.9.
14.	Text	15	t11	1. Text Box: “0” 2. When touched by the user multiply the value of number n0 by 10	10.2.2.10.
15.	Text	13	t9	1. Text Box: “C” 2. Divides number n0 by 10	10.2.2.11.
16.	Text	14	t10	1. No Text, Background set to a green tick mark. 2. Depending on which field was chosen to be inputted on the home page and sends the number n0 value to that respective field 3. Resets number n0 to 0 4. Sets the Belt states saved in variables va0 and va1 5. Goes back to Home page	10.2.2.12.
17.	Text (Global)	15	t12	Text Box: Depends on what field was chosen on the Home page	N/A

None of the elements on this page need to communicate with Arduino, hence none of them send their component IDs when touched.

6.3.3. Page 3: Calibration

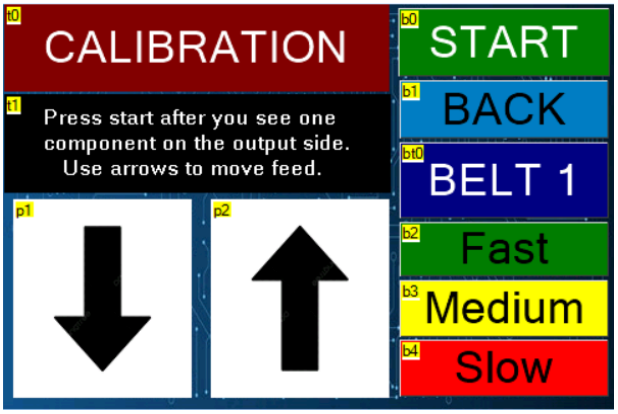


Figure 41: Calibration Page on Nextion
Page ID: 2/Setup
Page Elements: 15

After the user has entered the status of the belts and the desired cutting requirements, the display seamlessly transitions to the Calibration page. Since the Resistor reel-cutting device does not employ a sensor, it becomes crucial to calibrate the device to an established zero point. In this case, the calibration involves ensuring that only one component from the reel is visible on the output side. To accomplish this, the user is required to first load the reel and then exert control over the reel's movement on the belt using the speed and direction controllers, as depicted in the figure above. It's important to note that the calibration process is performed on one belt at a time, and the user has the flexibility to revisit and modify the input values for the cutting process if needed. Once the user has finished calibrating he can start the cutting process.

The components featured on this page are elucidated through the assistance of the table provided below. In case there is accompanying code for a particular page component, the reference to that code can be found in the appendix. The scope of each component is explicitly stated, but only if it pertains to a global scope.

Sl.No.	Component Type (variable scope)	Component ID	Object Name	Component Function	Element code: Appendix Reference
1	Picture	1	p0	Page Background	N/A
2	Picture	2	p1	Down facing arrow which moves the chosen belt forward at the speed chosen by the user for calibration	N/A
3	Picture	6	p2	Up facing arrow which moves the	N/A

				chosen belt backward at the speed chosen by the user for calibration	
4	Text	3	t0	• Text Box: “CALIBRATION” • Page Title	N/A
5	Text	4	t1	• Text Box: “Press start after you see one component on the output side. Use arrows to move feed.” • Calibration Instructions	N/A
6	Button	5	b0	• Start button to start cutting process • Obtains the values inputted by the user for the number of sets and components from the home page and stores it in variables • Sends the values stored in these variables to Arduino • Transition to Cutting progress page	10.2.3.1.
7	Variable (Global)	7	va0	Variable to store the number of sets for Belt 1	N/A
8	Variable (Global)	8	va1	Variable to store the number of components for Belt 1	N/A
9	Variable (Global)	9	va2	Variable to store the number of sets for Belt 2	N/A
10	Variable (Global)	10	va3	Variable to store the number of components for Belt 2	N/A
11	Button	11	b1	Button to go back to the Home page	10.2.3.2.
12	Button	12	b2	Button to set calibration speed to Fast	N/A
13	Button	13	b3	Button to set calibration speed to Medium	N/A
14	Button	14	b4	Button to set calibration speed to Slow	N/A

15	Dual State Button	15	bt0	Button for user to switch between Belt1 and Belt2 during calibration	10.2.3.3.

The Component IDs of the following elements are sent to Arduino:

- Picture p1: On Touch Release
- Picture p2: On Touch Release
- Button b2: On Touch Release
- Button b3: On Touch Release
- Button b4: On Touch Release
- Dual State Button bt0: On Touch Release

The component IDs of Pictures p1 (Downward arrow) and p2 (Upward arrow) establish communication with the Arduino by transmitting their respective component IDs. This enables the Arduino to detect touch inputs and initiate the corresponding action on the selected belt. Similarly, buttons b2, b3, and b4 facilitate communication to inform the Arduino about the desired actuation speed. Additionally, the Dual state button establishes communication to indicate the belt that the user intends to calibrate.

The variables va0, va1, va2, and va3 serve as temporary storage on this page, retaining the number of sets and components for each belt. This allows for the subsequent transmission of these values to the Arduino, which triggers the cutting actuation process. The values are sent to the Arduino when the user interacts with the Start button b0.

6.3.4. Page 4: Cutting Progress Page

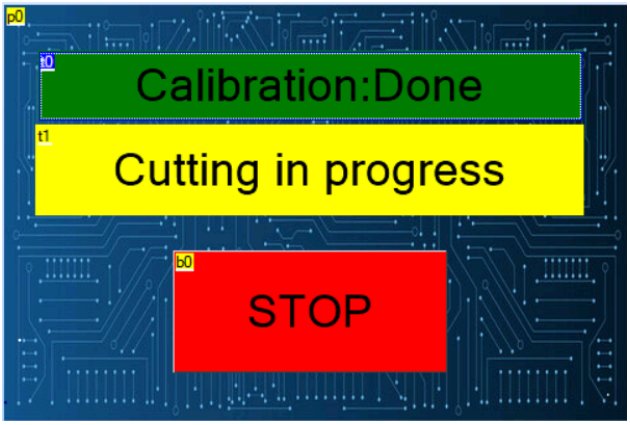


Figure 42: Cutting Progress Page on Nextion
Page ID: 3/final
Page Elements: 4

Upon completion of the input and calibration phases, the user proceeds to initiate the cutting progress, which then directs them to the Cutting Progress page, as depicted in the above figure. On this page, the user patiently awaits the completion of the cutting process.

Moreover, they have the option to interrupt the cutting process prematurely in the event of any issues or concerns.

The components featured on this page are elucidated through the assistance of the table provided below. In case there is accompanying code for a particular page component, the reference to that code can be found in the appendix. The scope of each component is explicitly stated, but only if it pertains to a global scope.

Sl.No.	Component Type (variable scope)	Component ID	Object Name	Component Function	Element code: Appendix Reference
1	Picture	1	p0	Page Background	N/A
2	Text	2	t0	Text Box: “Calibration:Done”	N/A
3	Text	4	t1	Text Box: “Cutting in progress”	N/A
4	Button	3	b0	A Stop button that redirects the user to the Stop Page	10.2.4.1.

The components present on this page do not require communication with the Arduino.

6.3.5. Page 5: Cutting Done Page

Figure 43: Cutting Done Page on Nextion
Page ID: 4/Done
Page Elements: 3

Upon the successful completion of the cutting process, the Nextion display seamlessly transitions to the Cutting Done page, as depicted above. From this page, the user can easily navigate back to the home page, bringing the display back to its initial state, ready for the next cutting cycle. Out of all the transitions observed so far, the movement from the Cutting Progress page back to the Cutting Done page is the only one governed by the Arduino. In contrast, previous transitions occurred solely when the user interacted with buttons on the display.

The components featured on this page are elucidated through the assistance of the table provided below. In case there is accompanying code for a particular page component, the reference to that code can be found in the appendix. The scope of each component is explicitly stated, but only if it pertains to a global scope.

Sl.No.	Component Type (variable scope)	Component ID	Object Name	Component Function	Element code: Appendix Reference
1	Picture	1	p0	Page Background	N/A
2	Text	2	t0	Text Box: “JOB COMPLETE!”	N/A
3	Button	3	b0	• Done button • Reset values on Home page: 1. Belt States to ON 2. Sets and Components values to 0 • Transition to Home Page	10.2.5.1.

The components featured on this page operate independently and do not necessitate communication with the Arduino. The page transition occurs irrespective of these components, allowing for a seamless transition.

6.3.6. Page 6: Stop Confirmation Page

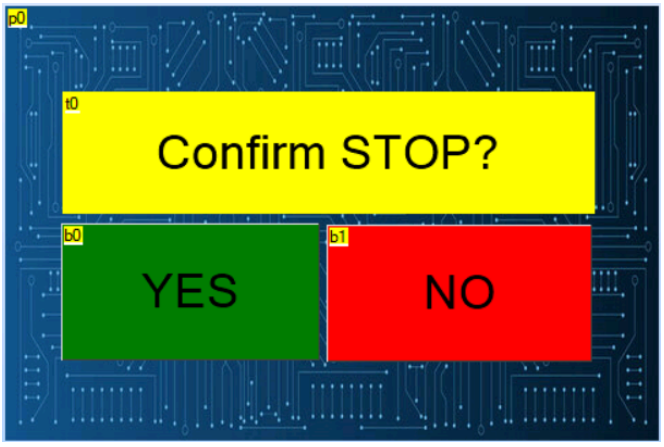


Figure 44 : Stop Confirmation Page on Nextion
Page ID: 5/stop
Page Elements: 4

In the event of any issues encountered during the cutting process, the user can utilize the stop button located on the Cutting Progress page. Upon pressing this button, the display transitions to the Stop Confirmation page, as depicted above. On this page, the user is prompted to confirm whether they genuinely wish to halt the process or if it was pressed

unintentionally. If it was pressed by mistake, they can choose not to confirm the stop and will be redirected back to the Cutting Progress page. However, if the user decides to confirm the stop, the cutting operation is interrupted, and the display promptly returns to the Home page, restoring the display to its initial state and preparing it for the subsequent cutting cycle.

The components featured on this page are elucidated through the assistance of the table provided below. In case there is accompanying code for a particular page component, the reference to that code can be found in the appendix. The scope of each component is explicitly stated, but only if it pertains to a global scope.

Sl.No.	Component Type (variable scope)	Component ID	Object Name	Component Function	Element code: Appendix Reference
1	Picture	1	p0	Page Background	N/A
2	Text	2	t0	Text Box: “Confirm STOP?”	N/A
3	Button	3	b0	• Yes Button to confirm stop • Reset Values in Home Page: The Sets and components values • Transition to Home Page	10.2.6.1.
4	Button	4	b1	• No button • Go back to Cutting Progress Page	10.2.6.2.

The Component IDs of the following elements are sent to Arduino:

- Button b0: On Touch Release

When the user confirms the stop of the cutting process, it is crucial to inform the Arduino. This communication is accomplished by the Yes button b0 on the Stop Confirmation page, which transmits its component ID to the Arduino. Upon the user's touch of this button, the Arduino promptly responds by halting the cutting process as intended.

7. Scope for Improvement

Sensor:

In order to address the occasional slipping of resistor reels and ensure consistent feeding, it would be beneficial to incorporate a sensor that can accurately count the number of resistors being fed into the machine before they are cut. This addition not only provides valuable feedback from the machine but also helps to maintain smooth and reliable operation. One suitable option for this purpose is the MOC7811 optocoupler sensor. By integrating this sensor into the system, it becomes possible to monitor the resistor feeding process and make necessary adjustments as needed.

Display:

Considering the time constraints of the project, we made the decision to utilize the Nextion display, despite it being a more expensive option compared to other Arduino-compatible LCD displays available. However, as an improvement option, it could be worth considering replacing the Nextion display with a more cost-effective LCD display. This alternative display could be programmed directly through Arduino C++ code using open-source UTFT libraries, potentially reducing the overall cost of the device. This improvement can be explored if there is sufficient time available for further enhancements.

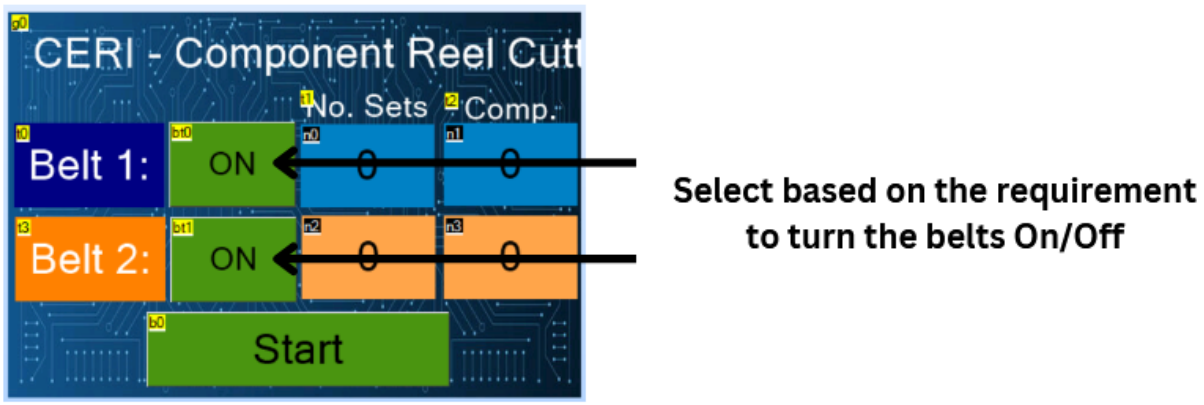
8. Conclusion and Learnings

Working collaboratively in a simulated corporate environment, our team successfully completed the Resistor Reel Cutter project while gaining valuable insights into professional dynamics. Each member's expertise in mechanical, electrical, and software engineering fostered a collaborative atmosphere, allowing us to learn from one another and enhance our skills across domains. Despite time constraints, we made informed decisions, such as selecting the Nextion display over cost-effective alternatives, while also considering potential improvements. Throughout the project, we prioritized client requirements, including the incorporation of dual operating bands for increased throughput and an accessible Human Machine Interaction interface. Safety measures received utmost importance, and although CE conformity was not fully addressed due to project constraints, we focused on reducing risks for the operator. Overall, this project provided us with invaluable experience in project management, collaboration, and professional presentation, strengthening our engineering capabilities for future endeavors.

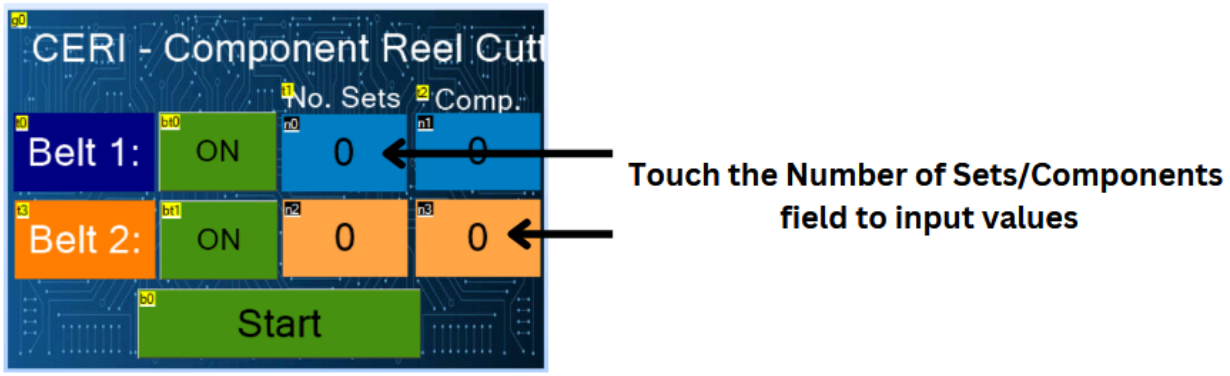
9. User Manual

Display User Manual

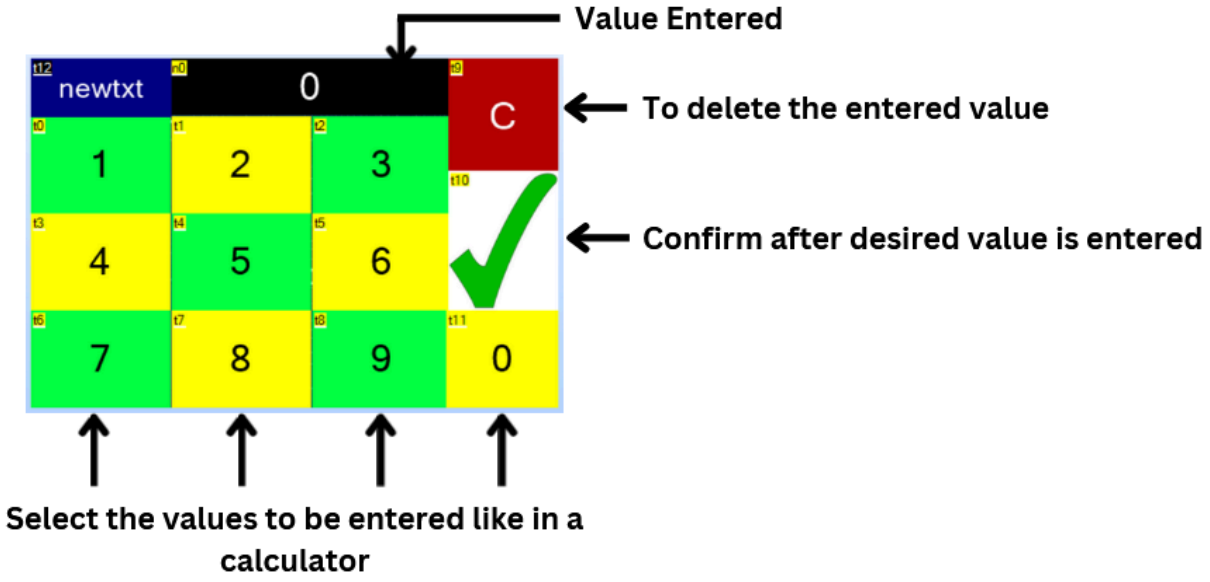
Step 1:



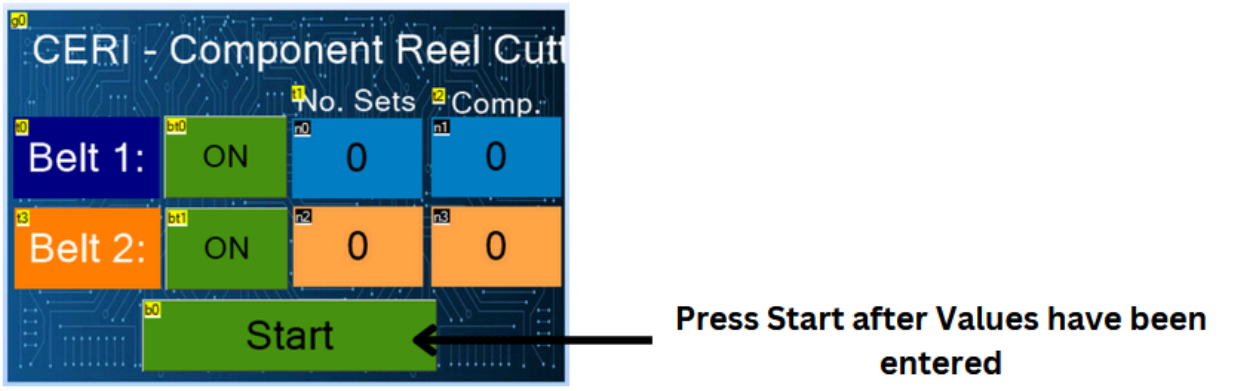
Step 2:



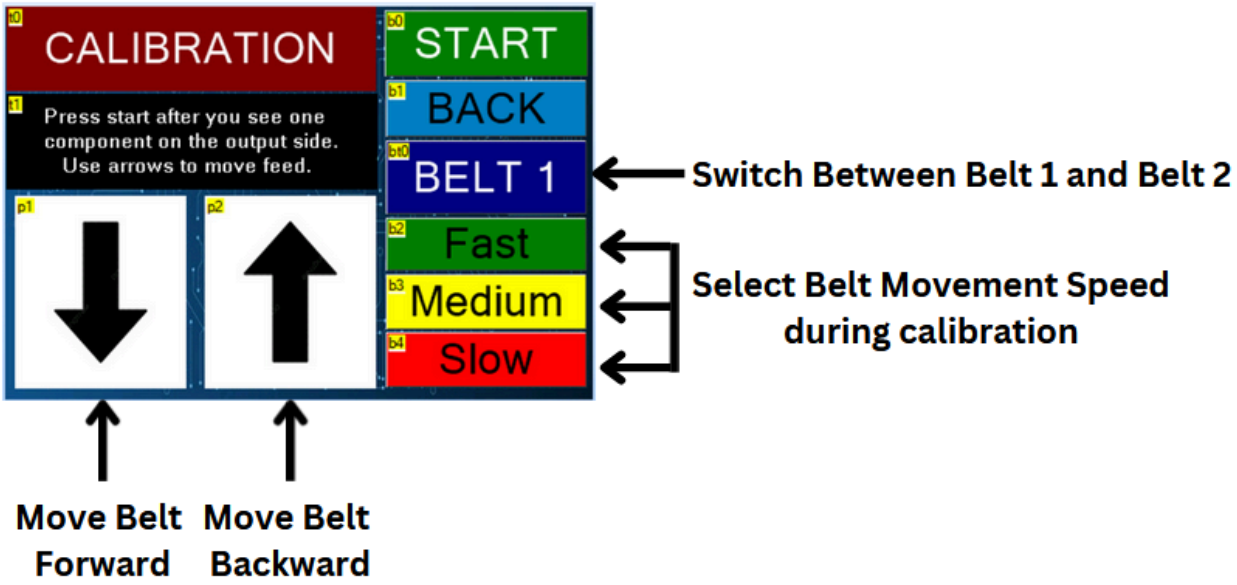
Step 3: Entering the value for the selected field



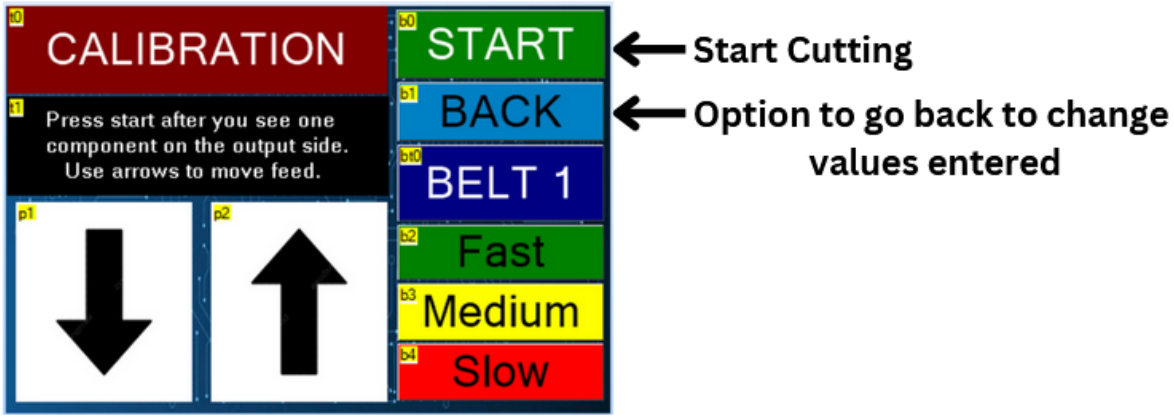
Step 4:



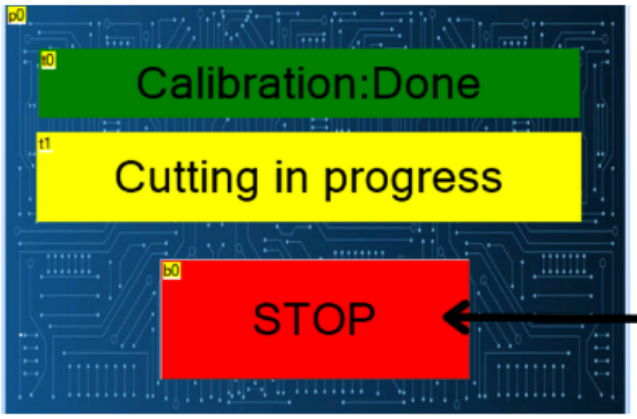
Step 5: Calibration of the Belts



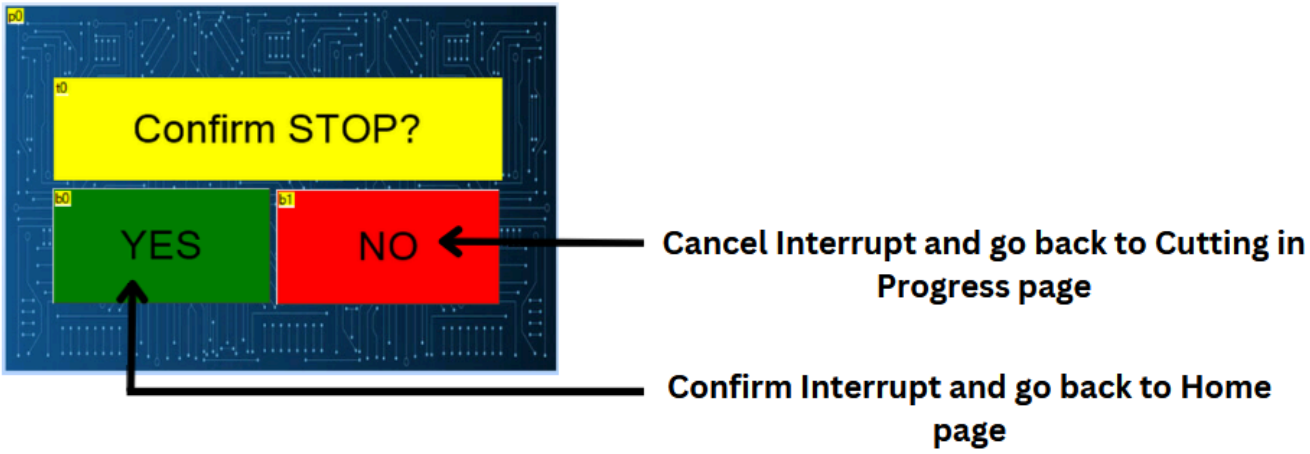
Step 6: Start Cutting



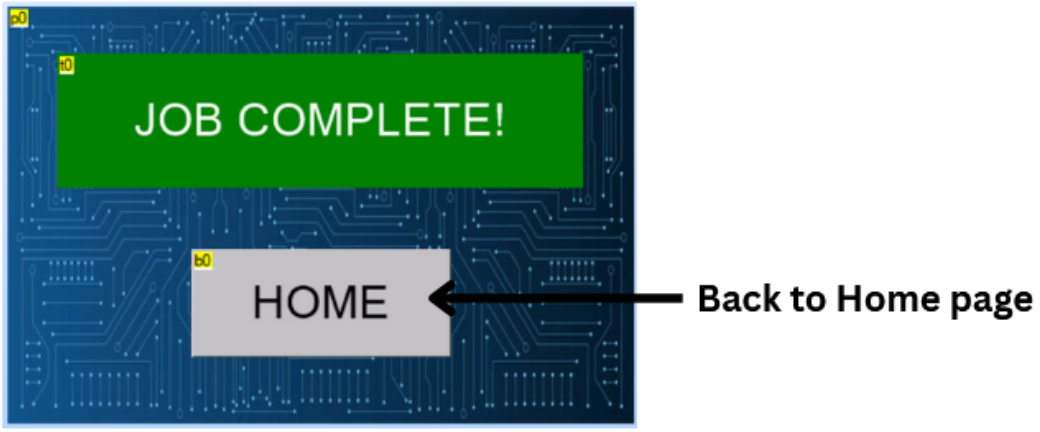
Step 7: Wait for the Cutting Process to complete/ Interrupt the Cutting Process



Step 8: Confirm Stop



Step 9: Back to Home after Cutting Process



10. Appendix

10.1. Code for Section 5: Software

10.1.1. Arduino Code for Calibration

Developed by: Sathvick Bindinganavale Srinath

Topic written by: Sathvick Bindinganavale Srinath

```
#include <Stepper.h>

const int enable_pin = 8;
const int dir_pin = 5;
const int step_pin = 2;
const int dir_pin_2 = 6;
const int step_pin_2 = 3;

// Calibration for belt 1 moving forward
void calibration_forward_1(int speed)
{
  int Step_count  = 0;
  for(Step_count; Step_count< speed; Step_count++)
      // 1350 - one resistor rotation
      // 3200 - half rotation
      //6400 - full rotation
      {
      digitalWrite(enable_pin, HIGH); // enables the stepper motor
      digitalWrite(dir_pin, HIGH);
      digitalWrite(step_pin, LOW);
      delayMicroseconds(500);
      digitalWrite(step_pin, HIGH);
      delayMicroseconds(500);
      }
}

// Calibration for belt 2 moving backwards
void calibration_back_2(int speed)
{
  int Step_count  = 0;
  for(Step_count; Step_count< speed; Step_count++)
      // 1350 - one resistor rotation
```

```
      // 3200 - half rotation
      //6400 - full rotation
      {
      digitalWrite(enable_pin, HIGH); // enables the stepper motor
      digitalWrite(dir_pin_2, HIGH);
      digitalWrite(step_pin_2, LOW);
      delayMicroseconds(500);
      digitalWrite(step_pin_2, HIGH);
      delayMicroseconds(500);
      }
}

// Calibration for belt 1 moving backwards
void calibration_back_1(int speed)
{
  int Step_count  = speed;
  for(Step_count; Step_count> 0; Step_count--)
      // 1350 - one resistor rotation
      // 3200 - half rotation
      //6400 - full rotation
      {
      digitalWrite(enable_pin, HIGH); // enables the stepper motor
      digitalWrite(dir_pin, LOW);
      digitalWrite(step_pin, LOW);
      delayMicroseconds(500);
      digitalWrite(step_pin, HIGH);
      delayMicroseconds(500);
      }
}

// Calibration for belt 2 moving forward
void calibration_forward_2(int speed)
{
  int Step_count  = speed;
  for(Step_count; Step_count> 0; Step_count--)
      // 1350 - one resistor rotation
      // 3200 - half rotation
      //6400 - full rotation
      {
      digitalWrite(enable_pin, HIGH); // enables the stepper motor
      digitalWrite(dir_pin_2, LOW);
```

```
      digitalWrite(step_pin_2, LOW);
      delayMicroseconds(500);
      digitalWrite(step_pin_2, HIGH);
      delayMicroseconds(500);
      }
}
```

10.1.2. Arduino Code for Nextion

Developed by: Aniketh Padmakar

Topic written by: Aniketh Padmakar

1.

```
include <Nextion.h>
```

2.

```
//Page 0 : Home
NexDSButton bt1 = NexDSButton(0,6,"bt0");//Belt 1 ON/OFF
NexDSButton bt2 = NexDSButton(0,11,"bt0");//Belt 2 ON/OFF
int bt1s = 1;//Belt 1 On
int bt2s = 1;//Belt 2 On
```

3.

```
//Page 1 : Numberpad no data transfer from this page to arduino
//Page 2 : Setup/Calibration
NexDSButton blt = NexDSButton(2,15,"bt0");//belt 1/belt 2 for
calibration
int which_belt = 1;//2 for belt 2
NexPicture forward = NexPicture(2,2,"p1");// Forward arrow (DOWN)
NexPicture back = NexPicture(2,6,"p2");// Forward arrow (DOWN)

NexButton start = NexButton(2,5,"b0");//Start cutting process
NexVariable sets1 = NexVariable(2,7,"va0");
NexVariable comps1 = NexVariable(2,9,"va2");
NexVariable sets2 = NexVariable(2,8,"va1");
NexVariable comps2 = NexVariable(2,10,"va3");

long b1_sets = 0;
long b1_comps = 0;
```

```
long b2_sets = 0;
long b2_comps = 0;
uint32_t number = 0;//number pointer to get values from nextion

NexButton fast = NexButton(2,12,"b2");//Belt config speed: Fast
NexButton medium = NexButton(2,13,"b3");//Belt config speed: Fast
NexButton slow = NexButton(2,14,"b4");//Belt config speed: Fast
int speed = 2550;//2550-slow 3200-medium 6400-fast
```

4.

```
//page 3: Final, after process has been started
//page 4: After cutting done
//page 5: Stop page, to confirm stop
NexButton stop_yes = NexButton(5,3,"b0");//Stop has been
confirmed
int stop = 0;
int b1_done = 0;
int b2_done = 0;
```

5.

```
//Register button objects to the touch event list
NexTouch *nex_listen_list[] = {
  &bt1,
  &bt2,
  &forward,
  &back,
  &start,
  &fast,
  &medium,
  &slow,
  &stop_yes,
  &blt,
  NULL
};
```

6.

```
//belt 1 ON/OFF
void bt1PopCallBack(void *ptr){
  if(bt1s == 1){
      bt1s = 0;
```

```
      Serial.println("Belt 1:OFF");
  }
  else{
      bt1s = 1;
      Serial.println("Belt 1:ON");
  }
}

//belt 1 ON/OFF
void bt2PopCallBack(void *ptr){
  if(bt2s == 1){
      bt2s = 0;
      Serial.println("Belt 2:OFF");
  }
  else{
      bt2s = 1;
      Serial.println("Belt 2:ON");
  }
}
```

```
//Belt selection for Calibration
void bltPopCallBack(void *ptr){
  if(which_belt == 1){
  which_belt=2;
  Serial.println("Belt 2");}
  else{
      which_belt=1;
      Serial.println("Belt 1");
  }
}
```

7.

```
//Start button on calibration page
void startPopCallBack(void *ptr){
  stop = 0;
  Serial.println("Cutting Start");

  sets1.getValue(&number);
  Serial.println();
  b1_sets = number;
```

```
  Serial.print("Sets1:");
  Serial.println(b1_sets);
  Serial.println();

  comps1.getValue(&number);
  Serial.println();
  b2_sets = number;
  Serial.print("Sets2:");
  Serial.println(b2_sets);
  Serial.println();

  sets2.getValue(&number);
  Serial.println();
  b1_comps = number;
  Serial.print("Comps1:");
  Serial.println(b1_comps);
  Serial.println();

  comps2.getValue(&number);
  Serial.println();
  b2_comps = number;
  Serial.print("Comps2:");
  Serial.println(b2_comps);

  if(stop == 0){
      if(bt1s == 1){
  cut_resistors_1_1(b1_sets,b1_comps);}
  b1_done = 1;
  delay(200);
  if(bt2s == 1){
  cut_resistors_2_1(b2_sets,b2_comps);}
  b2_done = 1;
  }
}
```

8.

```
void forwardPopCallBack(void *ptr){
  Serial.println("Forward");
  if(which_belt == 1){
  calibration_forward_1(speed);
  }
```

```
  else{
  calibration_forward_2(speed);
  }
}

void backPopCallBack(void *ptr){
  Serial.println("Back");
  if(which_belt == 1){
  calibration_back_1(speed);
  }
  else{
      calibration_back_2(speed);
  }
}
```

9.

```
void stop_yesPopCallBack(void *ptr){
  stop = 1;
  Serial.println("STOP!!!!");
  which_belt = 1;
  b1_sets = 0;
  b1_comps = 0;
  b2_sets = 0;
  b2_comps = 0;
  b1_done = 0;
  b2_done = 0;
}
```

10.

```
  Serial.begin(9600);
  nexInit();
  bt1.attachPop(bt1PopCallBack,&bt1);
  bt2.attachPop(bt2PopCallBack,&bt2);
  slow.attachPop(slowPopCallBack,&slow);
  medium.attachPop(mediumPopCallBack,&medium);
  fast.attachPop(fastPopCallBack,&fast);
  start.attachPop(startPopCallBack,&start);
  stop_yes.attachPop(stop_yesPopCallBack,&stop_yes);
  blt.attachPop(bltPopCallBack,&blt);
  forward.attachPop(forwardPopCallBack,&forward);
  back.attachPop(backPopCallBack,&back);
```

11.

```
nexLoop(nex_listen_list);
  if(b1_done == 1 && b2_done == 1){
  Serial.print("page 4");
  Serial.write(0xff);
  Serial.write(0xff);
  Serial.write(0xff);
  delay(200);
  which_belt = 1;
  b1_sets = 0;
  b1_comps = 0;
  b2_sets = 0;
  b2_comps = 0;
  b1_done = 0;
  b2_done = 0;
  pb.setValue(0);}
```

12.

```
//slow speed
void slowPopCallBack(void *ptr){
  speed = 675;
  Serial.println("Speed:Slow");
  //done_page_trans();
}

//medium speed
void mediumPopCallBack(void *ptr){
  speed = 3200;
  Serial.println("Speed:Medium");
}

//fast speed
void fastPopCallBack(void *ptr){
  speed = 6400;
  Serial.println("Speed:Fast");
}
```

10.1.3. Arduino Code for Counting number of resistors

Topic written by : Sathvick Bindinganavale Srinath

Topic developed by : Sathvick Bindinganavale Srinath

```
const int x = 200;
int rotations_per_resistor = 2550;

Stepper myStepper(x,2,5);
Stepper myStepper1(x,3,6);
void cut_resistors_1_1(long number_of_sets, long
number_of_resistors){
  myStepper.setSpeed(1000);
  myStepper.step(-rotations_per_resistor*2);
  delay(200);
  for(int i = 0; i<number_of_sets; i++)
  {
      if(stop == 0){
      int Step_count = 0;
      for(Step_count; Step_count< number_of_resistors*675
;Step_count++)
      {
      digitalWrite(enable_pin, HIGH); // enables the stepper motor
      digitalWrite(dir_pin, HIGH);
      digitalWrite(step_pin, LOW);
      delayMicroseconds(500);
      digitalWrite(step_pin, HIGH);
      delayMicroseconds(500);
      }
      delay(200);
      //Turn on both the servo motors for cutting.
      chop_belt1();
      nexLoop(nex_listen_list);
      }
  }

  Serial.println("Cutting 1 done");
}

void cut_resistors_2_1(long number_of_sets, long
number_of_resistors){
  myStepper1.setSpeed(1000);
  myStepper1.step(rotations_per_resistor*2);
```

```
  delay(200);
  for(int i = 0; i<number_of_sets; i++)
  {
      if(stop == 0){
      long Step_count = number_of_resistors*675;
      for(Step_count; Step_count> 0 ; Step_count--)
      {
      digitalWrite(enable_pin, HIGH); // enables the stepper motor
      digitalWrite(dir_pin_2, LOW);
      digitalWrite(step_pin_2, LOW);
      delayMicroseconds(500);
      digitalWrite(step_pin_2, HIGH);
      delayMicroseconds(500);
      }
      delay(200);
      //Turn on both the servo motors for cutting.
      chop_belt2();
      nexLoop(nex_listen_list);
      }
  }

  Serial.println("Cutting 1 done");
}
```

10.1.4. Arduino Code for Chopping of Resistors

Topic written by : Sathvick Bindinganavale Srinath

Topic developed by : Sathvick Bindinganavale Srinath

```
#include <Servo.h>

Servo motor1;
Servo motor2;
Servo motor3;
Servo motor4;

void chop_belt1()
{
      motor1.write(120); //initial
      motor2.write(0);
```

```
      delay(500);

      motor1.write(35); //initial
      motor2.write(90);

      delay(500);

}

void chop_belt2()
{
      motor3.write(80); //initial
      motor4.write(90);

      delay(500);

      motor3.write(150); //initial
      motor4.write(0);

      delay(500);

}
```

10.2. Code for Section 6: GUI Software

Developed by: Aniketh Padmakar

Topic written by: Aniketh Padmakar

10.2.1. Home Page Components Code:

1.

```
page Setup
```

2.

```
numPad.t12.txt="B1:Sets"
numPad.va0.val=bt0.val
numPad.va1.val=bt1.val
page numPad
```

3.

```
numPad.t12.txt="B1:Comp"
numPad.va0.val=bt0.val
numPad.va1.val=bt1.val
page numPad
```

4.

```
numPad.t12.txt="B2:Sets"
numPad.va0.val=bt0.val
numPad.va1.val=bt1.val
page numPad
```

5.

```
numPad.t12.txt="B2:Comp"
numPad.va0.val=bt0.val
numPad.va1.val=bt1.val
page numPad
```

6.

```
 if(bt0.val==0)
{
 bt0.txt="ON"
}
if(bt0.val==1)
{
bt0.txt="OFF"
}
```

7.

```
  if(bt1.val==0)
 {
  bt1.txt="ON"
 }
 if(bt1.val==1)
 {
 bt1.txt="OFF"
 }
```

10.2.2. Number Pad Page Components Code:

1.
```
n0.val=n0.val*10+1
```
2.
```
n0.val=n0.val*10+2
```
3.
```
n0.val=n0.val*10+3
```
4.
```
n0.val=n0.val*10+4
```
5.
```
n0.val=n0.val*10+5
```
6.
```
n0.val=n0.val*10+6
```
7.
```
n0.val=n0.val*10+7
```
8.
```
n0.val=n0.val*10+8
```
9.
```
n0.val=n0.val*10+9
```
10.
```
n0.val=n0.val*10
```
11.
```
n0.val=n0.val/10
```
12.
```
if(t12.txt=="B1:Sets")
{
 Home.n0.val=n0.val
```

```
}
if(t12.txt=="B1:Comp")
{
 Home.n1.val=n0.val
}
if(t12.txt=="B2:Sets")
{
 Home.n2.val=n0.val
}
if(t12.txt=="B2:Comp")
{
Home.n3.val=n0.val
}
n0.val=0
Home.bt0.val=va0.val
Home.bt1.val=va1.val
page Home
```

10.2.3. Calibration Page Components Code:

1.

```
va0.val=Home.n0.val
va1.val=Home.n1.val
va2.val=Home.n2.val
va3.val=Home.n3.val
get va0.val
get va2.val
get va1.val
get va3.val
page final
```

2.

```
page Home
```

3.

```
if(bt0.val==0)
{
 bt0.txt="BELT 1"
}
if(bt0.val==1)
```

```
{
 bt0.txt="BELT 2"
}
```

10.2.4. Cutting Progress Page Components Code:

1.

```
page stop
```

10.2.5. Cutting Done Page Components Code:

1.

```
Home.n0.val=0
Home.n1.val=0
Home.n2.val=0
Home.n3.val=0
Home.bt0.val=0
Home.bt1.val=0
page Home
```

10.2.6. Stop Confirmation Page Components Code:

1.

```
Home.n0.val=0
Home.n1.val=0
Home.n2.val=0
Home.n3.val=0
page Home
```

2.

```
page final
```

10.3. Engineering Project Master Documents

Industrieprojekt-Masterplan/Engineering Project Master Plan:

FHWS Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt

Sommer Semester 2023 Option SSa: 15.03. - 08.05.

Fakultät Maschinenbau/Faculty of mechanical engineering

Industrieprojekt (BM, Modul-Nr. 29, 8 ECTS, 4 SWS)

Studiengang Mechatronik/Mechatronics

Projektarbeit (BMC, Modul-Nr. 29, 7 ECTS, 4 SWS / Engineering Project IMC, Module Nr. 29, 7 ECTS, 4 credit points)

Thema/Topic : Resistor Reel Cutter

Industriepartner/Industry Partner:

Betreuer FHWS/Supervisor FHWS: Prof. Rainer Herrler

Betreuer Industrie/Industry Supervisor:

Projekt-Team/Project Team: 7 (Nr.)

Shankar, Sai Karthik, IMC- Mechatronics, 8, 4019084

Name/Last Name, Vorname/First Name, Studiengang/Degree Programme, Semester, Matrikel-Nr./Matriculation number

Bindinganavale Srinath, Sathvick, IMC-Mechatronics, 6, 4020025

Name/Last Name, Vorname/First Name, Studiengang/Degree Programme, Semester, Matrikel-Nr./Matriculation number

Shantesh Ukkali, Samrat, IMC-Mechatronics, 8, 4019124

Name/Last Name, Vorname/First Name, Studiengang/Degree Programme, Semester, Matrikel-Nr./Matriculation number

Padmakar, Aniketh, IMC-Mechatronics, 8, 4019074

Name/Last Name, Vorname/First Name, Studiengang/Degree Programme, Semester, Matrikel-Nr./Matriculation number

Name/Last Name, Vorname/First Name, Studiengang/Degree Programme, Semester, Matrikel-Nr./Matriculation number

Name/Last Name, Vorname/First Name, Studiengang/Degree Programme, Semester, Matrikel-Nr./Matriculation number

Terminplan/Schedule:

Auftaktveranstaltung FHWS intern/Launch event FHWS	16th March, 2023
Auftaktveranstaltung mit Firma/Launch event industry partner	
1. Testat/Test	22nd March, 2023
2. Testat/Test	31st March, 2023
3. Testat/Test	6th April, 2023
Vortrag/Presentation	14th April, 2023
Benotete Meetings in English/graded meetings in english	14th April, 2023
4. Testat/Test	3rd May, 2023
5. Testat/Test - Endpräsentation/Final presentation	8th May, 2023
6. Testat/Test - Projektdokumentation/project documentation	19th May, 2023
Abgabe Bericht in Englisch über eigene Tätigkeit im Projekt (nur für BM) - only applicable for mechanical engineering programme	

Industrieprojekt-Masterplan/Engineering Project Master Plan:

FH·W-S
Hochschule
für angewandte Wissenschafter
Würzburg-Schweinfurt

Sommer Semester 2023 Option SSa: 15.03. - 08.05.

Fakultät Maschinenbau/Faculty of mechanical engineering

Industrieprojekt (BM, Modul-Nr. 29, 8 ECTS, 4 SWS)

Studiengang Mechatronik/Mechatronics

Projektarbeit (BMC, Modul-Nr. 29, 7 ECTS, 4 SWS / Engineering Project IMC, Module Nr. 29, 7 ECTS, 4 credit points)

Thema/Topic : Resistor Reel Cutter

Industriepartner/Industry Partner:
Betreuer FHWS/Supervisor FHWS: Prof. Rainer Herrler
Betreuer Industrie/Industry Supervisor:

Aufgabenstellung/Task:

The objective is to construct a cost-effective Resistor Reel Cutting machine for the Soldering lab at the Centre of Robotics in THWS. This device is specifically designed to handle and cut various taped component reels, including resistors, diodes, and more, with the aim of expediting the lab preparation process.

10.4. Meeting Minutes Summary

Date: 22nd March 2023	**Kickoff Meeting**	**Duration:** 13:35 to 14:00
Attendees: Mr. Fabian Dax, Aniketh Padmakar, Samrat Shantesh Ukkali, Sathvick Bindinganavale Srinath, Sai Karthik Shankar **Note Taker: Aniketh**		
Agenda Items: ● To understand the project task ● To prioratize the device requirements **Action Items:** ● To decide on a team management method ● To come up with an initial plan and design for the device		

Date: 31st March 2023	**Weekly Meeting**	**Duration:** 12:15 to 12:45
Attendees: Mr. Fabian Dax, Aniketh Padmakar, Samrat Shantesh Ukkali, Sathvick Bindinganavale Srinath **Note Taker: Aniketh**		
Agenda Items: ● Project Progress: Our Idea for the Project and the Progress so far ● Team Structure ● Paperwork related to the project **Action Items:** ● To change the input number pad: To make the device look more aesthetically pleasing ● To do all the Project paperwork digitally ● Send Bill of materials for products which have to be ordered through AZ-Delivery to Mr.Fabian Dax ● To add a physical stop button		

Date: 6th April, 2023	**Weekly Meeting**	**Duration:** 12:30 - 14:30
Attendees: Mr.Fabian Dax, Sai Karthik Shankar, Aniketh Padmakar, Samrat Shantesh Ukkali, Sathvick Bindinganavale Srinath **Note Taker: Sathvick**		
Agenda Items: • Project Progress: Mechanical Design of the Cutting Module • Integrated Numberpad into Display • Device Software Features • Electrical Components and PCB **Action Items:** • Add a Calibration Feature • To reconsider the Display if the budget is exceeding		

Date: 14th April 2023	**Midterm Presentation**	**Duration:** 13:35 to 14:00
Attendees: Prof. Rainer Herrler, Mr.Fabian Dax, Sai Karthik Shankar, Aniketh Padmakar, Samrat Shantesh Ukkali, Sathvick Bindinganavale Srinath **Note Taker: Aniketh**		
Agenda Items: • To demonstrate the ready cutting module • To demonstrate the HMI GUI **Action Items:** • To check the power source based on the current required for full device		

Date: 3rd May 2023	**Weekly Meeting**	**Duration:** 13:35 to 14:00
Attendees: Prof. Rainer Herrler, Sai Karthik Shankar, Aniketh Padmakar, Samrat Shantesh Ukkali, Sathvick Bindinganavale Srinath **Note Taker: Aniketh**		
Agenda Items: • To demonstrate the ready cutting modules with the device housing • To demonstrate the HMI froma user perspective • To show the change in the top enclosure of the device		

- To demonstrate a test cutting cycle

Action Items:

- To finalize the wiring
- Finish the electronics, HMI, and stop button housing
- To test the completed Device

Date: 8th May 2023	**Final Presentation**	**Duration:** 13:15 to 14:15

Attendees: Prof. Rainer Herrler, Mr. Fabian Dax, Sai Karthik Shankar, Aniketh Padmakar, Samrat Shantesh Ukkali, Sathvick Bindinganavale Srinath

Note Taker: Aniketh

Agenda Items:

- To demonstrate the fully assembled device with two modules, all electronics housed, a stop button and the HMI display
- To present our work and explain the components and the working of the device

Action Items:

- To fix the small bugs and to finish the project

10.5. Bill of Materials

No.	Item Name	Vendor	Quantity	Total Cost
1	NEMA 17 Stepper Motors	Amazon	2	18.99 €
2	SG90 Servo Motors	Amazon	6	13.59 €
3	Acrylic Sheets	Amazon	2	23.98 €
4	Scalpel Blade	Amazon	1	5.99 €
5	626ZZ Ball Bearing	Amazon	20	12.49 €
6	3.5" NEXTION TFT Display	Amazon	1	59.90 €
7	MG996R Servo Motors	Amazon	5	30.99 €
8	Emergency Stop Switch	Amazon	1	13.99 €
9	O ring kit	Amazon	1	5.99 €
10	10mm MDF plate	Bauhaus	1	6.99 €
11	Arduino UNO	AZ Delivery	1	9.99 €
12	CNC Shield	AZ Delivery	1	8.79 €
13	A4988 Stepper Driver	AZ Delivery	2	10.58 €
			Grand Total	**222.26 €**

11. References

[1]"**Arduino Based Resistor Reel Cutting Machine**." https://circuitdigest.com/microcontroller-projects/arduino-resistor-reel-cutting-machine (accessed 10 May 2023).

[2]"**Resistor reel cutter machine**". https://github.com/Sathvick11/Resistor-reel-cutter-machine.

[3]"**PicResize**" .https://picresize.com/. (accessed 14 May 2023).

[4]"**ITEADLIB_Arduino_Nextion** ".https://github.com/itead/ITEADLIB_Arduino_Nextion. (accessed 14 May 2023).

[5]"**NEXTION EDITOR**". https://nextion.tech/nextion-editor/#_section1. (accessed 14 May 2023).

[6]"**NEXTION HMI DISPLAY WITH ARDUINO -Getting Started with LED ON/OFF".** https://www.youtube.com/watch?v=RbPfo8wW74I. (accessed 14 May 2023).

[7]"**Using Notion for Project Management: Pros, Cons, & More**". https://www.unleash.so/a/blog/using-notion-for-project-management-pros-cons-and-more. (accesed 18 May 2023)

[8] **Seeeduino V4.3, digital image,** https://media-cdn.seeedstudio.com/media/catalog/product/cache/bb49d3ec4ee05b6f018e93f896b8a25d/2/-/2-102010026-seeeduino-font.jpg(accessed 11 May 2023)

[9] **Seeeduino V4.0** https://wiki.seeedstudio.com/Seeeduino_v4.0/ (Accessed 11 May 2023)

[10] **CNC Shield V3, digital image,** https://cdn.shopify.com/s/files/1/0014/4313/5560/products/DSC02398_535x.jpg?v=1619889785 (Accessed 11 May 2023)

[11] **MG996R servo motor**, https://www.towerpro.com.tw/product/mg996r/ (Accessed 15 May 2023).

[12] **MG996R servo motor, digital image,** https://img.gkbcdn.com/p/2015-08-10/towerpro-mg996r-digital-high-torque-servo-metal-gear-compatible-with-jr-futaba-1571995609590._w500_.jpg (Accessed 15 May 2023).

[13] **Arduino IDE**, https://www.arduino.cc/en/software

[14] **Tinkercad,** Feeding the resistor reel, https://www.tinkercad.com/things/jxtvMKAL95w, (Accessed 16 May 2023).

[15] **Tinkercad,** Mechanism for chopping the reel, https://www.tinkercad.com/things/jpApdXrInj1, (Accessed 16 May 2023).

[16] **Fritzing,** https://fritzing.org(Accessed 16 May 2023).

[17] **Adding Seeeduino board into Arduino IDE,**
https://wiki.seeedstudio.com/Seeeduino_v4.0/#driver-installation(Accessed 16 May 2023).

[18] **.JSON file link for the IDE**,
https://raw.githubusercontent.com/Seeed-Studio/Seeed_Platform/master/package_legacy_seeeduino_boards_index.json(Accessed 16 May 2023).

[19] **CNY70 IR sensor, digital image,**
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.vishay.com%2Fdocs%2F83751%2Fcny70.pdf&psig=AOvVaw0f8adVXmNo8bxkjiuot81R&ust=1684340968817000&source=images&cd=vfe&ved=0CBEQjRxqFwoTCOCeleCn-v4CFQAAAAAdAAAAABAE (Accessed 16 May 2023).

[20] **Taped Components Specifications,**

https://htr-india.com/faqs1/tape-ammo-pack-specifications-hia-hta-vhia-frs-rl-series-resistors/ (Accessed 17 May 2023).

[21] **Servo library,** https://www.arduino.cc/reference/en/libraries/servo/

(Accessed 17 May 2023).

[22] **Stepper library,** https://www.arduino.cc/reference/en/libraries/stepper/ (Accessed 17 May 2023).