

University of Central Oklahoma

Masters Theses Dissertations and Theses

May 2023

## Design of Smart Tool Organizer

Mohmed Yaeesh Shaikh

University of Central Oklahoma

# **Design of Smart Tool Organizer**

A Thesis Presented

by

**MOHMED YAEESH SHAIKH**

Submitted to the Graduate School of the

University of Central Oklahoma in partial fulfillment of the

requirements for the degree of

**MASTER OF SCIENCE IN MECHANICAL ENGINEERING**

May 2023

**Mechanical Engineering**



# Design of Smart Tool Organizer

A Thesis Presented

by


MOHMED YAEESH SHAIKH

Approved as to style and content by:




---

Dr. Abdellah Ait Moussa, Chair



---

Dr. Nesreen Alsou, Member



---

Dr. Morshed Khandaker, Member

# Acknowledgements

I would like to appreciate the support I received from the UCO Engineering and Physics departments, my thesis committee members, my peers, friends and family throughout my journey of working on this thesis. It was a bumpy road with a lot of ups and downs but with all the support I received, I stayed persistent and was able to finish my prototype and report. I would like to thank Dr. Moussa for always being there when I needed him and for sharing his excellent knowledge and support through this process. I would also like to thank Dr. Khandaker and Dr. Alsbou for all their support, help and being on my thesis committee.

I always got help when I needed it from the department of Engineering and Physics at UCO. Dr. Moussa assisted me with many things throughout this process such as 3D printing, design suggestions and report writing & formatting suggestions. Dr. Khandaker and Dr. Alsbou also supported me with their assistance when needed. I would also like to thank my family for their support during the times of hardships and frustrations. I ran into many difficult situations during building and designing my prototype and had many difficult and frustrating moments but with the help and encouragement of my family, my supervisor Dr. Moussa, my thesis committee members Dr. Khandaker and Dr. Alsbou and the department of Engineering and Physics I was able to build a working prototype and test it.

# ABSTRACT

## Design of Smart Tool Organizer

MAY 2023

MOHMED YAEESH SHAIKH

B.E., UNIVERSITY OF CENTRAL OKLAHOMA

M.S.M.E., UNIVERSITY OF CENTRAL OKLAHOMA

Directed by: Dr. Abdellah Ait Moussa

The increasing demand for compact and energy efficient machines and mechanisms, led to the emergence of a series of scalable instruments and devices used for testing, storage, manufacturing, and prototyping. The emergence of these devices indeed provided the flexibility and low capital cost that were necessary for product development and personal use. Furthermore, the demand is expected to proliferate to all domestic and industrial sectors of the economy, which brings us to the subject of the current investigation.

The main objective of the proposed project is the design and prototyping of a compact electromechanical smart tool organizer that is capable of storing, tracking personal use and availability of machine shop tools in the college of Mathematics and Science at the University of Central Oklahoma.

The proposed design incorporates a micro-controlled electro-mechanical dispensing unit, an interactive digital interface with key activation and a database for data collection & tracking of

tools and personnel users. The dispensing unit consists of a CNC machine, linear actuator and a 3D printed mechanical clamp which enables the unit to efficiently hold and move various tools to the desired location. The skeleton of the CNC machine is assembled using five stainless steel v-slots operating on a belt and pinion system. Pinions are fitted to NEMA 17 stepper motors to achieve 2D motion by converting rotational motion to linear motion using a belt driven actuator. The assembly of the CNC machine utilizes various gantry plates to hold v-slots in position along with providing mounts for stepper motors and linear actuator. The linear actuator acts as a third axis which allows the motion of the dispensing unit to operate in three directions. This provides mobility to the machine to precisely take the mechanical clamp to a predetermined position within the frame of the dispensing unit.

The design of mechanical clamp includes assembly of the base of the clamp, rack and pinion system, two gripper arms and a servo motor. The pinion converts the rotational motion to linear motion of the racks enabling the grip to open and close as required. The final assembly of the mechanical clamp is mounted on the linear actuator using the 3D printed mount bracket on the base of the clamp. The electronics and controls of the smart tool organizer includes low and high voltage operating components such as Nema 17 stepper motors, MG996R servo, linear actuator, limit switches, buck converter, DS3231 RTC module, SD card module, L298N motor drive module, TB6600 motor drivers, a 7" touch screen LED display and an arduino mega. The firmware arduino IDE is used to program these electronic components and ASCII is used to program the Interactive GUI and HMI on the Nextion display which connect to arduino using serial port and synchronizes to achieve the goal of the smart tool organizer.

# TABLE OF CONTENTS

| Chapter   | Page number |
|---|-------------|
| 1. Motivation and Objective of Smart Tool Organizer .....       | 1           |
| 1. Introduction .....   | 1           |
| 1.1 Motivation.....   | 1           |
| 1.2 Objective of the Smart Tool Organizer .....                 | 2           |
| 2. Background of Smart Tool Organizer and related research..... | 4           |
| 2. Literature review .....                                      | 4           |
| 2.1 History of Automation and Robotics.....                     | 4           |
| 2.1.1 Organizer model for intelligent robotic systems.....      | 5           |
| 2.1.2 Organizer model operational procedures.....               | 8           |
| 2.2 Example of Previous work in Robotics.....                   | 9           |
| 2.2.1 The Robotic arm assembly.....                             | 9           |
| 2.2.2 The Kinematics of Robotic arm.....                        | 10          |
| 3. Mechanical Design and Fabrication.....                       | 15          |
| 3. Mechanical Design of the Smart Tool Organizer.....           | 15          |
| 3.1 Robotic Unit.....   | 16          |
| 3.1.1 The CNC machine .....                                     | 16          |
| 3.1.1.1 The Y-axis gantry plate assembly.....                   | 17          |
| 3.1.1.2 The X-axis gantry plate assembly .....                  | 18          |
| 3.1.1.3 The XY-axis V-slot assembly.....                        | 19          |
| 3.1.1.4 The CNC frame assembly .....                            | 21          |
| 3.1.1.5 The supporting frame assembly .....                     | 22          |
| 3.1.1.6 The Z-axis gantry plate assembly .....                  | 23          |
| 3.1.2 The electro-mechanical gripper.....                       | 24          |



|  |    |
|--|----|
| 3.2 Base of the Smart Tool Organizer.....                                    | 27 |
| 4. Electronics and Controls.....   | 31 |
| 4.1 High voltage circuit.....  | 31 |
| 4.1.1.1 Nema 17 Stepper Motors and TB6600 motor driver.....                  | 31 |
| 4.1.1.2 Programming of Nema 17 Stepper Motors and TB6600 motor driver.....   | 36 |
| 4.1.2.1 Linear actuator and L298N motor drive module.....                    | 39 |
| 4.1.2.2 Programming of the Linear actuator and L298N motor drive module..... | 42 |
| 4.1.3.1 Electro-Mechanical clamp and servo motor.....                        | 44 |
| 4.1.3.2 Programming of the Electro-Mechanical clamp and servo motor.....     | 46 |
| 4.1.4.1 Limit switches.....  | 47 |
| 4.1.4.2 Programming of the Limit switches.....                               | 48 |
| 4.2 Low voltage circuit.....   | 50 |
| 4.2.1.1 SD card module.....  | 50 |
| 4.2.1.2 Programming of the SD card module.....                               | 52 |
| 4.2.2.1 DS3231 RTC module.....   | 56 |
| 4.2.2.2 Programming of the DS3231 RTC module.....                            | 58 |
| 4.2.3.1 Arduino Mega.....  | 60 |
| 4.3 Wiring and connections.....  | 62 |
| 5. Interfacing and Firmware.....   | 63 |
| 5.1 User-Interface of the Smart Tool Organizer.....                          | 63 |
| 5.1.1 Nextion LED display.....   | 63 |
| 5.1.2 Nextion Editor and Arduino programming.....                            | 65 |
| 6. Results and Discussion.....   | 74 |
| 6.1 Performance and accuracy testing of the smart tool organizer.....        | 74 |
| 6.1.1 Stepper motor performance and accuracy testing.....                    | 74 |
| 6.1.2 Servo motor performance and accuracy testing.....                      | 82 |

|  |           |
|--|-----------|
| <b>6.2 Criteria for tool selection for the smart tool organizer.....</b> | <b>84</b> |
| <b>7. Conclusion and future works.....</b>                               | <b>88</b> |
| <b>References.....</b>   | <b>91</b> |
| <b>Appendix A - Smart Tool Organizer Code.....</b>                       | <b>95</b> |

# LIST OF FIGURES

| Figure   | Page number |
|--|-------------|
| 1. Intelligent robotic system hierarchical structure block diagram ..... | 7           |
| 2. Labeled diagram of robotic arm assembly.....                          | 10          |
| 3. Wheel assembly and drawing.....                                       | 16          |
| 4. Y axis gantry plate assembly.....                                     | 17          |
| 5. Stepper motor mount gantry plate model and drawing.....               | 18          |
| 6. V-slot timing belt and pulley assembly.....                           | 19          |
| 7. V-slot timing belt and pulley assembly model and drawing.....         | 20          |
| 8. X and Y V-slots assembly.....   | 21          |
| 9. Angled corner connector model and drawing.....                        | 22          |
| 10. Limit switch assembly.....   | 22          |
| 11. Electro-Mechanical clamp assembly.....                               | 24          |
| 12. Base of the clamp model and drawing.....                             | 25          |
| 13. Arm of the clamp model and drawing.....                              | 26          |
| 14. Linear actuator mounting bracket.....                                | 27          |
| 15. Base of the smart tool organizer assembly.....                       | 28          |
| 16. Supporting and mounting brackets for CNC machine.....                | 29          |
| 17. L bracket model and drawing.....                                     | 29          |
| 18. Push to Open dropbox assembly.....                                   | 30          |
| 19. Stepper motor winding diagram.....                                   | 32          |
| 20. Nema 17 stepper motor dimensional drawing.....                       | 33          |
| 21. Nema 17 Stepper motor torque curve.....                              | 34          |

|  |    |
|--|----|
| 22. Code defining direction and pulse for stepper motor..... | 36 |
| 23. Code setting stepper motor pins as output.....           | 37 |
| 24. Code setting stepper motor direction.....                | 37 |
| 25. Code for rotation of the stepper motor.....              | 38 |
| 26. Linear actuator dimensional drawings.....                | 39 |
| 27. L298N module pinouts.....                                | 41 |
| 28. Wiring diagram of linear actuator and L298N module.....  | 41 |
| 29. Code defining L298N pins.....                            | 43 |
| 30. Code setting and enabling L298N pins.....                | 43 |
| 31. Code for the linear motion of linear actuator.....       | 43 |
| 32. PWM signal representation of servo motor.....            | 45 |
| 33. Dimensional drawing of servo motor.....                  | 45 |
| 34. Code including servo library and defining variable.....  | 46 |
| 35. Code for the rotational motion of servo motor.....       | 47 |
| 36. Limit switch positioning on a V-slot.....                | 48 |
| 37. Limit switch conditions.....                             | 49 |
| 38. Code for bringing stepper back to home position.....     | 49 |
| 39. Logic level shifter.....                                 | 51 |
| 40. microSD card slot.....                                   | 51 |
| 41. SD card module pinouts.....                              | 52 |
| 42. Defining initializeSD function.....                      | 53 |
| 43. Defining createFile function.....                        | 53 |
| 44. Defining writeToFile function.....                       | 54 |
| 45. Defining closeFile function.....                         | 54 |
| 46. Defining openFile function.....                          | 55 |
| 47. Defining readLine function.....                          | 55 |

|  |    |
|--|----|
| 48. S3231 RTC chip.....  | 56 |
| 49. Graph of oscillator frequency's relation with temperature..... | 57 |
| 50. EEPROM chip and I2C address selection jumper.....              | 57 |
| 51. RTC module pinouts.....  | 58 |
| 52. Defining print2digits function.....                            | 59 |
| 53. Code for storing time and date to SD card.....                 | 59 |
| 54. Arduino mega pinouts and components.....                       | 61 |
| 55. Wiring diagram of the smart tool organizer's components.....   | 62 |
| 56. Nextion display components.....                                | 64 |
| 57. Selection of display type.....                                 | 66 |
| 58. Display orientation selection.....                             | 67 |
| 59. Main interface of the nextion editor.....                      | 67 |
| 60. Pages setup in nextion editor.....                             | 68 |
| 61. Home page of the nextion display.....                          | 69 |
| 62. Numeric keyboard for entering student ID.....                  | 69 |
| 63. ASCII coding for next button in nextion editor.....            | 70 |
| 64. Error page for invalid id number.....                          | 70 |
| 65. Tool selection status page.....                                | 71 |
| 66. Tool receiving selection page.....                             | 71 |
| 67. Tool returning selection page.....                             | 72 |
| 68. Final status page.....   | 73 |

# LIST OF TABLES

| <b>Table</b>   | <b>Page number</b> |
|--|--------------------|
| 1. Nema 17 stepper motor specifications .....                                  | 34                 |
| 2. TB6600 microstep and current settings.....                                  | 35                 |
| 3. Linear actuator specifications.....   | 40                 |
| 4. MG996R servo motor specifications.....                                      | 44                 |
| 5. Arduino mega specifications.....  | 62                 |
| 6. Nextion display general and electronics specifications.....                 | 65                 |
| 7. Nextion display memory specifications.....                                  | 65                 |
| 8. Accuracy test data for x-axis stepper motor.....                            | 75                 |
| 9. Accuracy test data for y-axis stepper motors.....                           | 76                 |
| 10. Test data for linear speed of the stepper motors with varying steps.....   | 77                 |
| 11. Test data for linear speed of the stepper motors with varying delays.....  | 79                 |
| 12. Test data for linear speed of the stepper motors with varying current..... | 81                 |
| 13. Test data for linear speed of the servo motors with varying voltage.....   | 83                 |
| 14. Test data for normal force applied to the tool by the gripper arms.....    | 85                 |
| 15. Test data for application of various tools.....                            | 86                 |
| 16. Requirements for the selection of tools.....                               | 87                 |

# LIST OF GRAPHS

| <b>Graphs</b>   | <b>Page number</b> |
|---|--------------------|
| 1. Linear speed of the stepper motors with varying steps.....   | 79                 |
| 2. Linear speed of the stepper motors with varying delay.....   | 80                 |
| 3. Linear speed of the stepper motors with varying current..... | 82                 |
| 4. Linear speed of the servo motors with varying voltage.....   | 84                 |

# List of Abbreviations

|              |                                    |
|--------------|------------------------------------|
| <b>AI</b>    | <b>Artificial Intelligence</b>     |
| <b>CNC</b>   | <b>Computer numerical control</b>  |
| <b>RPA</b>   | <b>Robotic process automation</b>  |
| <b>STP</b>   | <b>Straight through processing</b> |
| <b>IM</b>    | <b>Intelligent machines</b>        |
| <b>RA</b>    | <b>Robotic assemblies</b>          |
| <b>IRS</b>   | <b>Intelligent robotic systems</b> |
| <b>UIC</b>   | <b>User input commands</b>         |
| <b>CIC</b>   | <b>Compiled input command</b>      |
| <b>IREDD</b> | <b>Infrared emitting diode</b>     |
| <b>PSD</b>   | <b>Position sensitive detector</b> |
| <b>T</b>     | <b>Torque</b>                      |
| <b>F</b>     | <b>Force</b>                       |
| <b>L</b>     | <b>Length</b>                      |
| <b>A</b>     | <b>Area</b>                        |
| <b>W</b>     | <b>Weight</b>                      |
| <b>M</b>     | <b>Total degrees of freedom</b>    |
| <b>J</b>     | <b>Joints</b>                      |
| <b>PWM</b>   | <b>Pulse width modulation</b>      |
| <b>DIP</b>   | <b>Dual in-line package</b>        |
| <b>PCB</b>   | <b>Printed circuit board</b>       |
| <b>LDO</b>   | <b>Low dropout voltage</b>         |



|               |  |
|---------------|--|
| <b>SD</b>     | <b>Secure Digital</b>                                      |
| <b>MISO</b>   | <b>Master in Slave Out</b>                                 |
| <b>MOSI</b>   | <b>Master Out Slave In</b>                                 |
| <b>SCK</b>    | <b>Serial Clock</b>  |
| <b>CS</b>     | <b>Chip select</b>   |
| <b>SPI</b>    | <b>Serial peripheral interface</b>                         |
| <b>RTC</b>    | <b>Real time clock</b>                                     |
| <b>EEPROM</b> | <b>Electrically Erasable Programmable Read-only Memory</b> |
| <b>I2C</b>    | <b>Inter-integrated controller</b>                         |
| <b>TCXO</b>   | <b>Temperature compensated crystal oscillator</b>          |
| <b>SDA</b>    | <b>Serial Data</b>   |
| <b>SCL</b>    | <b>Serial clock line</b>                                   |
| <b>ADC</b>    | <b>Analog-to-digital converter</b>                         |
| <b>UART</b>   | <b>Universal asynchronous receiver-transmitter</b>         |
| <b>ICSP</b>   | <b>In-circuit serial programming</b>                       |
| <b>IOREF</b>  | <b>Input/output reference</b>                              |
| <b>AREF</b>   | <b>Analogue reference</b>                                  |
| <b>TXD</b>    | <b>Transmit Data</b>                                       |
| <b>RXD</b>    | <b>Received data</b>                                       |
| <b>HMI</b>    | <b>Human-Machine Interface</b>                             |
| <b>GUI</b>    | <b>Graphical user interface</b>                            |
| <b>GPIO</b>   | <b>General Purpose Input/Output</b>                        |
| <b>SRAM</b>   | <b>Static random-access memory</b>                         |

# CHAPTER 1

## Motivation and Objective of Smart Tool Organizer

### 1. Introduction

#### 1.1 Motivation

With advancements in technology, the fourth industrial revolution has led to significant automation of manual functions using robotics. The human-automation occurred in various dedicated domains starting in the 1990s when tasks that are time-sensitive and safety-critical were intensively researched and implemented [1]. This domain included automation of tasks such as flight monitoring [2], monitoring dynamic processes in factories and power plants and other professional settings [3]. Automation took a moderate turn to Embodied systems and AI in the 90s where systems interacted with the environment using sensors through which it perceived the world [4]. Human-robot interaction using embodied systems is some of the pioneering achievements in this domain leading to the rise of robotics. The use of automation technology was introduced to a wider domain in the last trend which was previously limited to professional use. The modern automation technologies don't require extensive technical training to operate which enables non-professional consumers to use automation and robotics. Robots as social companions, automated vehicles and various interactive systems emerged increasing the quality of life [5].

The utilization of robotics using electric and mechanical components has led to increase in automation and remarkably increased the output of industries as well as facilitating the betterment of

domestic life. Industries such as manufacturing, construction, information technology, space exploration, agriculture and automobiles are some of the major beneficiaries of automation. Automation assists with testing, prototyping, assembling and designing products with much more accuracy, ease and reduced cost. The advent of robotics has also had a major impact on domestic life making everyday tasks to be performed with ease in a more efficient manner. The current investigation motive is to take another step towards advancement of automation using robotics and prototyping a compact device easing some of the tedious and time-consuming tasks.

With the advancement of technology, there has been a surge in various types of machines and tools. The tools developed for various purposes come in different shapes and sizes for their specific applications. The storage and organization of the tools creates a challenge in a publicly accessible dynamic work environment where different groups or individuals work on their projects requiring a multitude of tools such as an engineering machine shop. Traditionally, hooks, compartments and drawers are used to separate and organize tools to make them accessible and easy to find when required. These methods can lead to missing and misplaced tools due to lack of tracking and management and can cause setbacks in timing of the project. The motivation for the smart tool organizer is to design a prototype of a compact and scalable device that tracks, organizes, manages and dispenses tools as per the requirement for the machine shop in UCO School of Engineering.

## **1.2 Objective of the Smart Tool Organizer**

The objective of the design and prototype of a smart tool organizer is to automate the tracking and managing of tools using numerous mechanical and electrical components in the machine shop. Accessing tools in a machine shop can sometimes be a laborious process due to the variety of tools and the unorganized nature of the machine shop. Misplacement of tools is a common occurrence. A

device which can manage, dispense and track tools would be ideal for a machine shop saving time and effort. The proposed prototype uses automation and robotics to achieve this outcome.

The principal objective of the proposed project is the design and prototyping of a compact electromechanical smart tool organizer that is capable of storing, tracking personal use and availability, dispensing and managing of machine shop tools in the school of engineering at the University of Central Oklahoma.

Following list states, the objective of finalized design of the prototype:

- 1) The smart tool organizer includes a mechanism for the automatic retrieval of operator requested tools
- 2) The smart tool organizer requests an operator ID for access and bookkeeping (including day, time)
- 3) The smart tool organizer includes a mechanism for appropriate tool return
- 4) The smart tool organizer is compact and operator friendly

# CHAPTER 2

## Background of Smart Tool Organizer and related research

### 2. Literature review

This chapter presents the prior research and literature review of the smart tool organizer. The review consists of detailed research about automation and its impact on everyday life, uses of robotics in machines and advantages of 3D motion control, linear actuators utilization and properties of other electronic components.

#### 2.1 History of Automation and Robotics

As discussed in chapter 1, automation and robotics has transformed the way interaction occurs between humans and machines. From the initial applications on time sensitive and safety critical settings to embodied systems in automobiles, planes etc. the advancements and utilizations of automation can be seen in every modern industry. The next major phase of automation came with the introduction of robotics where actuators and sensors work simultaneously with the help of a microcontroller or computer to perceive and work on its environment. This led to the introduction of automated devices to a wider population regardless of their technical proficiency or knowledge about the technology. Currently, many such smart devices are in use from smartphones and smart speakers to automatic transmission and vending machines.

### **2.1.1 Organizer model for intelligent robotic systems**

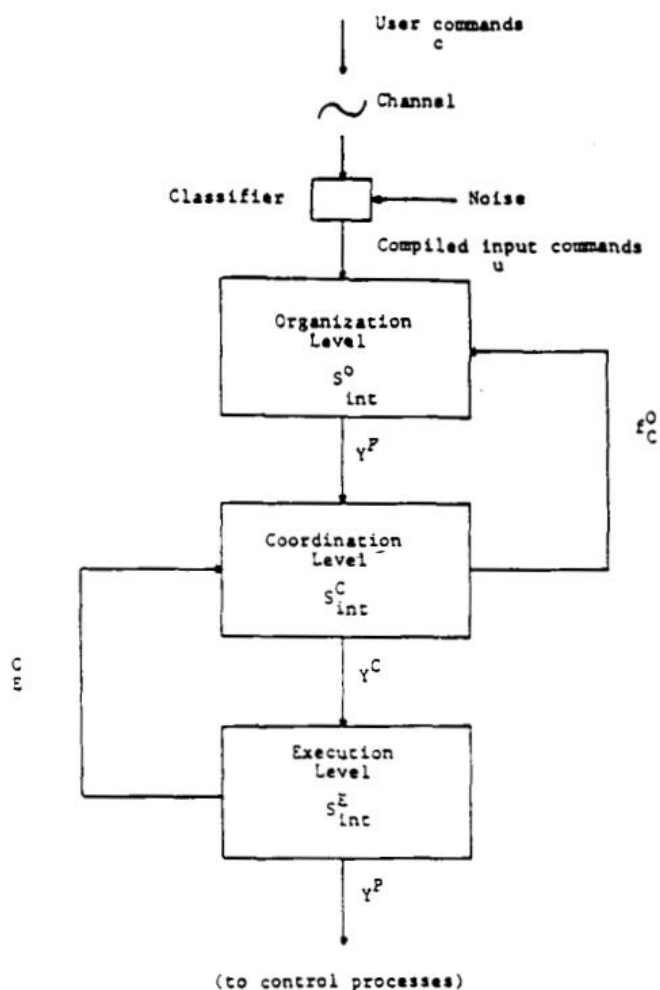
Automation and robotics continue to evolve and advance into the future with more functionalities and uses. With the developments in machine learning, data science and AI, new models of automations are developed such as robotic process automation (RPA) [8]. RPA allows tools to operate on other computers using their user interface. RPA executed scripts in the control dashboard by mapping the process in RPA tool language. This allows RPA to reduce the repetitive tasks for employees leading to reduced cost and return on investments (RoI). Commercial vendors such as AutomationEdge, Blue Prism and Softomotive offer RPA software which can quickly link applications together. RPA replaced many straight through processing (STP) software that were invented in the mid-nineties which allowed the information to electronically be transferred to required agents without human involvement [7]. The difference between RPA and STP is that the RPA uses an “outside-in” approach while STP uses an “inside-out” approach which allows RPA to remain unchanged [8]. It also allows RPA to keep the key content unchanged when the layout of the system changes. These applications and tools allowed for automation to make everyday tasks easier in information technologies.

Design and implementation of intelligent systems for the purpose of automation and robotics requires concepts involving mathematics and engineering [9]. Intelligent machines (IM's) consisting of robotic assemblies (RA's), artificial intelligence (AI's) and intelligent robotic systems (IRS's) design's major hurdle is the overall system coordination or simultaneous operation and not the operation of individual components [10]. To achieve the overall goal of an IM, it is imperative for all the components such as actuators, sensors and microcontroller to synthesize and follow a sequence of order. Programming and implementation of such systems is another major factor in its reliable operation. The RA and IRS designs have been studied considerably and numerous models of operation have been derived. One such model, hierarchically intelligent control, derived by Saridis

includes a tree-like structure of organization, coordination and execution of tasks by the IM's and IRS's with little interaction with humans [11]. This model follows the principle of decreasing precision with increasing intelligence derived mathematically. Another approach tackles the problems of intelligence control theory with nested hierarchical information structures derived by Meystel [12]. For AI modeling, planning systems, state representation and architecture of the system surround the central issue where logical models and symbolic models of distinct operations are used to achieve specific goals such as task planning and path planning. Here, task planning includes systems that don't include robotics and actuators while path planning and motion systems deal with robotics using mapping algorithms, geometry and metric fields [13].

In an effort to develop automated planning techniques for RA's powerful and efficient general procedures have been developed which is applicable to a wide variety of situations instead of specific instances [14]. In the development phase of an RA's automation, the effective planning of automation can be achieved using either real time planning or offline planning. In real time planning, the plan is executed and information is collected during the process to make a decision which is more efficient for a fuzzy or uncertain environment [15]. Offline planning is used when the system has learning capabilities to assist it to make a decision and execute a complex plan. For the robotic assemblies, AI planning is considered a dominating approach which uses operators, initial state and the goal of the system to generate a domain-independent engine [16]. Planning tasks for an RA includes a data structure that makes sure that the sequence of the task is followed. These data structures can be represented in various forms such as graphs, trees, triangle tables or a simple list of orders [17]. The IRS's task planning can be achieved using a hierarchical structure which consists of three levels such as organization which is the highest level where making decisions and planning takes place, coordination following the sequence of assigned tasks and finally the execution level [18]. This system allows improving the performance using self-learning and modifications of algorithms and

iterative processes to make a decision. Figure 1 below represents this hierarchical structure for an IRS.



**Figure 1-** Intelligent robotic system hierarchical structure block diagram [9]

The function of the organizer level is to interpret the user command received from the user interface and feedback received from coordination level and output a sequence of tasks in real-time to be executed. The output can include decision making, planning, learning feedback, reasoning and memory exchange for robotic assemblies and systems [11]. This level also processes huge user input or feedback data. The function of coordination level is to create an actual control action to be executed at the next level using coordinators which run a set of specified functions. This level transfers the information received from organization level to the execution level after converting it into executable steps [15]. The function of execution level is to execute the precise information or



commands received from coordinators. This level is usually made of actuators and hardware such as stepper motors.

For RA's and IRS's to work in a real world environment, they need to be able to interpret uncertainty or fuzziness if they run into imprecise descriptions of events or information. One such model of RA' and IRS's organization level was proposed by Valavanis and Stellakis where a unification and generalization of binary logic was researched to allow IM's to operate in fuzzy environments [9]. The model helps the organization level to pass down decisions to coordination level which then develop scenarios for execution. It also includes troubleshooting of IM's failures such as malfunction of a sensor, actuator or microcontroller. Another advantage of the model is that the system can operate on whether it's an idealized or fuzzy environment. A quick review of this model is discussed to better understand the proposed algorithms and operational procedures [9].

### **2.1.2 Organizer model operational procedures**

The initial information is provided to the organizer workspace which includes a set of definitions that establishes this workspace and a set of governing operational functions. The set of definitions includes the interpretation for fuzzy sets found in user input commands (UIC's) alongside certain or crisp information which is concerned with the operation of RA's or IRS's in an uncertain environment [9]. This interpretation pairs the fuzzy and crisp input in the form of  $(u_n, t_k)$  where  $n = 1, 2, 3, \dots, M$  and  $k = 0, 1, 2, \dots, \Psi$  where  $M$  and  $\Psi$  are finite number.  $n$  represents a number of commands and assigns a name to UIC and  $k$  represents a fuzzy or uncertain linguistic term such as "some", "more", "a lot", etc. Therefore, the set  $(u_n, t_k)$  is able to represent an imprecise UIC, for example a UIC "slice some" can be represented by  $(UIC)_2 = (u_2, t_k = \text{"some"})$ . The UIC depends on the specific application and the next term, for example overall  $(UIC)_2$  for a bread cutting intelligent robotic system will represent "slice some bread". The UIC is received by a user interface such as a

remote, an led screen or a smart speaker's microphone. Upon receiving this information, classification and conversion of fuzzy or crisp commands to coding language understandable by machine occurs at the organization level. The un in UIC is first converted to machine language since it carries critical information and following that  $t_k$  is classified [9]. This classification outputs a compiled input command (CIC) corresponding to a UIC. The CIC provides the RA's and IRS's with a well-defined environment to execute the UIC. It uses 5 functions in its set to achieve this task including compiled name, specific terms set, fuzzy set, 1-1 mapping of distribution functions and linguistic term. Using this model, a general organization model for intelligent robotic systems and robotic assemblies is achieved with sets of functions and algorithms for user input commands.

## **2.2 Example of Previous work in Robotics**

### **2.2.1 The Robotic arm assembly**

Using Mechatronics and the three branches: Mechanical design, electrical circuits-electronics and computer programming a robotic arm was developed by Tolis and Fragulis to automate the identification and sorting of different size objects. The light weight design of robotic arm uses infrared sensors and low cost actuators to reduce the cost of the design which was a major issue due to the complexity of the design and high cost of advanced actuators and sensors. The use of mechatronics assists in tackling problems related to power, compatibility, torque and kinetics for the complete design of the robotic arm. The robotics arm has five degrees of freedom and consists of five rotary joints along with a grip. The design of rotary joints include a rotation of the grip, a rotation of the wrist, a rotation of the elbow, a rotation of the shoulder and finally a base rotation providing the five degrees of freedom. The mechanical parts for the robotic arm are of AL5 type and were selected as per the requirement of the assembly using Lynxmotion. The assembly uses the infrared sensor which comprises IRED (infrared emitting diode), PSD (Position sensitive detector) and a circuit for processing signals made by Sharp [19]. The sensor Sharp 2Y0A21 F46 is a distance sensor and

correlates distance detected to voltage output. The position of the sensor is on the gripper which allows it to measure the distance of the objects. The Torque was calculated and used to select the servo motors by Hitec to control the rotation of the rotary arms. The operation of the robotic arm is controlled by a BotBoarduino microcontroller using a computer.

### 2.2.2 The Kinematics of Robotic arm

Forward and inverse kinematics were developed to study the operation of the robotic arm. The degree of freedom and torque were calculated to analyze the use of servo motors and determine the kinetics of the assembly. The weight of the object being lifted was used as the force ( $F$ ) in the vertical plane to calculate the torque ( $T$ ) along with the length of the arm from the pivot point ( $L$ ). The equation for force is given by: ( $F = g * m$ ) corresponding to the weight equation: ( $W = g * m$ ) of an object due to the gravitational force acting on the object. Here  $g$  represents the gravitational acceleration and  $m$  is the mass of the object. To calculate the required torque from the servos the torque equation: ( $T = F * L$ ) was used.

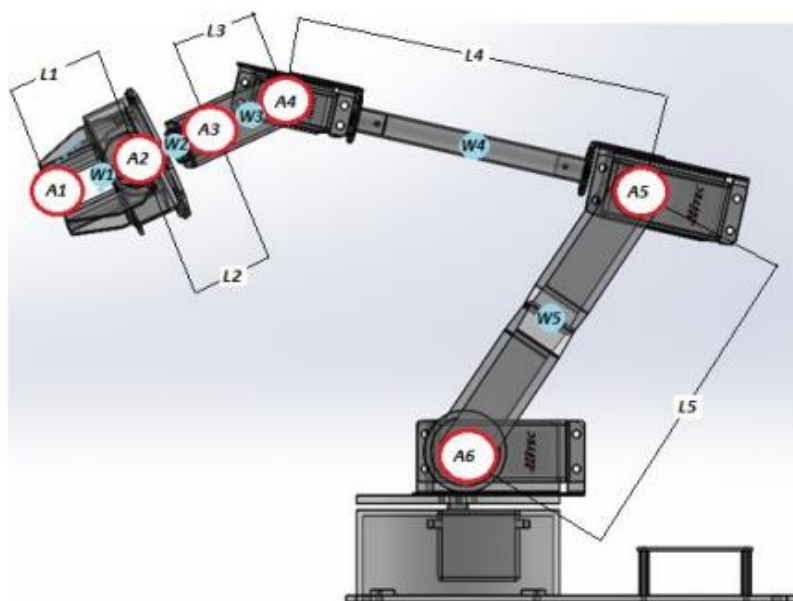


Figure 2- Labeled diagram of robotic arm assembly [19]

The figure 2 shows a labeled diagram of weights of sensor bracket, wrist bracket, grip, various servos and mechanical parts used in the assembly. It also shows the lengths of each rotary arm and the weight of the object held by the grip. Here, A1 is the weight held by the grip while A2, A3,.. A6 is the weight of the servo actuators used. The lengths L1 to L6 show the lengths of the arms while W1 to W5 shows the weight of the arms considering the centre of mass at the centre of the lengths. For this application, A2 for HS-422 servo is 45.5g, A3 for HS-225MG is 31g, A4 for HS-645MG with the arm ASB-24 is 55.2g + 7g = 62.2g, A5 for HS-755HB with the arm ASB-201 is 110g + 13g = 123g and A6 for HS-805BB along with the part ASB-204 = 197g + 18g=215g. The weight of the grip (W1) is 15.7g, weight of sensor bracket (W2) is 10g, weight of wrist bracket (W3) is 9g, combined weight of AT-04, ASB-06 and HUB-08 (W4) is 10g + 6g + 8g =24g and Combined weight of ASB-205 and ASB-203 (W5) is 16g + 15g = 31g. The lengths L1, L2, L3, L4 and L5 were measured and listed as 2.8 cm, 2.8cm, 2.85 cm, 18.73 cm and 14.6 cm respectively [19]. Now that all the parameters were measured, the calculation of torque was accomplished using the discussed equation. The T1 for HS-422 servo was calculated using the equation:

$$T_1 = A_1 * L_1 + W_1 * \frac{L_1}{2}$$

Plugging in the values mentioned above, for the T1 the torque was calculated to be 0.021 kg/cm. The torques for the servo HS-225MG T2 was calculated using equation:

$$T_2 = A_1 * (L_1 + L_2) + W_1 * \left(\frac{L_1 + L_2}{2}\right) + A_2 * L_2 + W_2 * \left(\frac{L_2}{2}\right)$$

The T2 resultant value is 0.207 kg/cm. The torque value for HS-645MG servo T3 was calculated using the equation:

$$T_3 = A_1 * (L_1 + L_2 + L_3) + W_1 * \left(\frac{L_1 + L_2 + L_3}{2}\right) + A_2 * (L_2 + L_3) + W_2 * \left(\frac{L_2 + L_3}{2}\right) + A_3 * (L_3) + W_3 * \left(\frac{L_3}{2}\right)$$

Using this equation, the value for T3 is calculated to be 0.511 kg/cm for the A3 servo motor. The torque for servo motor HS-755HB T4 was calculated using the equation:

$$T_4 = A_1 * (L_1 + L_2 + L_3 + L_4) + W_1 * \left(\frac{L_1 + L_2 + L_3 + L_4}{2}\right) + A_2 * (L_2 + L_3 + L_4) + W_2 * \left(\frac{L_2 + L_3 + L_4}{2}\right) + A_3 * (L_3 + L_4) + W_3 * \left(\frac{L_3 + L_4}{2}\right) + A_4 * (L_4 + L_5) + W_4 * \left(\frac{L_4 + L_5}{2}\right)$$

This equation gives a resultant value for T4 to be 5.122 kg/cm. The required torque value for the servo motor HS-805BB T5 was calculated using the equation:

$$T_5 = A_1 * (L_1 + L_2 + L_3 + L_4 + L_5) + W_1 * \left(\frac{L_1 + L_2 + L_3 + L_4 + L_5}{2}\right) + A_2 * (L_2 + L_3 + L_4 + L_5) + W_2 * \left(\frac{L_2 + L_3 + L_4 + L_5}{2}\right) + A_3 * (L_3 + L_4 + L_5) + W_3 * \left(\frac{L_3 + L_4 + L_5}{2}\right) + A_4 * (L_4 + L_5) + W_4 * \left(\frac{L_4 + L_5}{2}\right) + A_5 * (L_5) + W_5 * \left(\frac{L_5}{2}\right)$$

The torque value for servo motor A5 was calculated to be 12.25 kg/cm [19]. These calculated values were then compared to the nominal values of torque provided by the manufacturer. The nominal value for servo motor HS-422 is given to be 4.1 kg/cm, HS-225MG is 4.8 kg/cm, HS-665MG is 9.6 kg/cm, HS 755HB is 13.2 kg/cm and HS-805BB is 24.7 kg/cm. The torque values calculated above were using just the weights of the material used and servo motors without adding any load value.

Comparing these values with the nominal values shows that the robotic arm assembly should be able to operate without any load present. Next, an object of weight 100 g was added to the above equation to compare the torque values to the nominal values. The final calculated values for the torque using a load of 100 g were calculated to be 0.3 kg/cm for the HS-422 servo, 0.767 kg/cm for HS-225MG servo, 1.356 kg/cm for the HS-665MG servo, 7.84 kg/cm for the HS-755HB servo and 16.43 kg/cm for the HS-805BB servo. Comparing these values with the nominal values of the servo motors, it can be concluded that the robotic arm assembly will be able to withstand a load of 100 g. Next, the load values were increased to 300 g for the calculations and the calculated values for torque using the

given equation were 0.86 kg/cm for the HS-422 servo, 1.887 kg/cm for HS-225MG servo, 3.04 kg/cm for the HS-665MG servo, 13.27 kg/cm for the HS-755HB servo and 24.79 kg/cm for the HS-805BB servo. As these values reach the nominal values of the servo motors, it was concluded that the maximum load value for the robotic arm assembly is about 300 g.

The robotic arm assembly has five degrees of freedom. This can be determined using the five rotary actuators used in the assembly, not including the actuator responsible for opening and closing of the gripper. Mathematically, this was calculated using the Gruebler-Kutzbach equation:

$$M = (n - 1) * 3 - (J_1 - J_2) * 2$$

In this equation, n represents the links used in the assembly, joints with one degree of freedom are represented by J1 and joints with two degrees of freedom are represented by J2. M represents the total degrees of freedom of the system [19]. For this application, there are no rotary arms with two degrees of freedom, while there are five joints with one degree of freedom and from the figure it can be seen that there are a total of six links. Thus, plugging these values in the equations results in M or total degrees of freedom of the system to be five.

The power required for the operation of servos and the sensor is provided by two different sources. The servos operate on the power supplied by the adapter of the BotBoarduino which is connected to a power source. This adapter outputs a current of 2.25 A at 6 volts. The infrared sensor is supplied 0.5 A at 5 volts using a USB cable connected to the computer. This power source is regulated by the BotBoarduino at 1.5 A and 5 volts. The source of the adapter for the computer and BotBoarduino is supplied power using an outlet of 220 V and a frequency of 50 Hz. The computer's adapter receives a current at 6.5A at 18.5 volts [19]. It is imperative that all the components receive proper power for the robotic arm to be functional. The communication between the sensor and the actuators is carried out by the microcontroller which uses the output values of the sensor to control

and operate the actuators. The selection of the microcontroller BotBoarduino was carried out due to the fact that it can output two different current values and can match the needs of the servos and the infrared sensor. For this application, the servos are connected to the VS output of the microcontroller which acts as a 6V power source and the infrared sensor is connected to the VL output of the microcontroller which acts as a 5V source. The BotBoarduino microcontroller also has LD29150DT50R, an onboard regulator which can regulate a current up to 1.5 A. This current can be supplied using the VL input at 5V. For applications of devices that require more power input an adapter can be connected to the VS input like the one discussed earlier.

# CHAPTER 3

## Mechanical Design and Fabrication

### 3. Mechanical Design of the Smart Tool Organizer

With the intention to design a prototype that has the ability to organize, store and trade tools in an efficient manner, the design of smart tool organizer was developed and fabricated. The design development of the smart tool organizer was carried out to obtain an optimum performance and provide ease of use to the end user. The design incorporated a base which will be used to store and dispense the tools and a robotic unit which is capable of operating on three axes and has three degrees of freedom. As mentioned in chapter two, the degree of freedom can be calculated using the Gruebler-Kutzbach equation:

$$M = (n - 1) * 3 - (J_1 - J_2) * 2$$

In this equation, n represents the links used in the assembly, joints with one degree of freedom are represented by J1 and joints with two degrees of freedom are represented by J2. M represents the total degrees of freedom of the system [19]. In the Smart Tool Organizer, the links used for the assembly includes two links with stepper motors on the x and y axis, one link with the linear actuator on the z axis and one link with the electro-mechanical clamp attached to the linear actuator resulting in four total numbers of links in the assembly of the robotic unit. The joints used in the assembly offered one degree of freedom and three such joints were used enabling a motion in x-y-z direction. Using the equation mentioned above, the degree of freedom for the smart tool organizer is calculated where n is 4,  $J_1$  is 3 and  $J_2$  is 0 resulting in three degrees of freedom.

This chapter includes a description of the design components used in the assembly of the base and the robotic unit. It also includes the components used in the overall assembly and how each



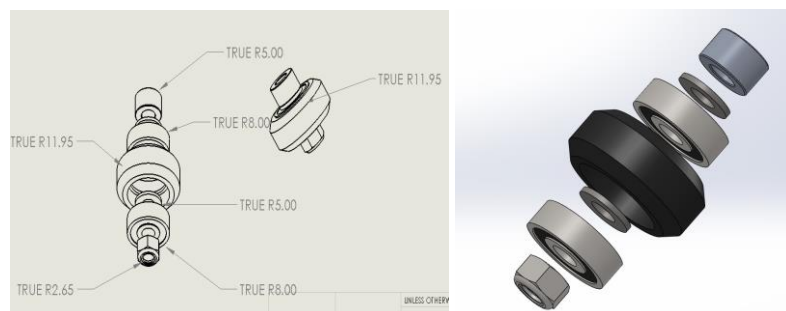
component plays an important role in realizing the idea of a compact, automated and efficient design of the smart tool organizer.

### 3.1 Robotic unit

#### 3.1.1 The CNC machine

The design of the smart tool organizer incorporates a CNC machine to assist it to move to various locations of tools. The CNC machine operates on stepper motors where rotational motion is converted to linear motion using a belt and pinion drive system. The assembly of the CNC machine includes 20 x 40 mm linear V-slots, acrylic gantry plates, solid V wheels, L-brackets, GT2-2M timing pulleys with 14 teeth, GT2 timing belt both 2 and 4 feet and two 20 x 20 V-slot linear rails for support. These parts were selected based on the research carried out for performance and the weight of a cnc machine to achieve a precise motion control and optimal performance of the smart tool organizer. This assembly enables the prototype to move in x-y direction linearly to the desired location to get access to the tools.

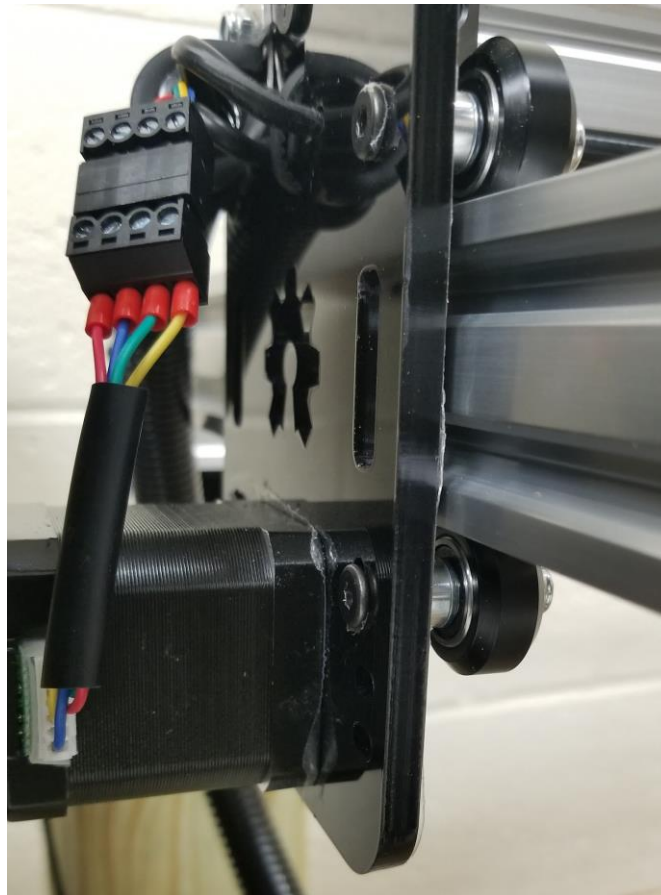
In the assembly of the CNC machine, twelve solid V wheels were used to achieve the linear motion. Sets of four wheels were attached to the acrylic gantry plate in a manner where they were allowed to roll in the slots of the V-slot linear rails. The wheels were assembled using a rubber wheel shell, two bearings to reduce friction, a hex nut to secure it and two precision shims for alignment.



**Figure 3-** Wheel assembly and drawing

### 3.1.1.1 The Y-axis gantry plate assembly

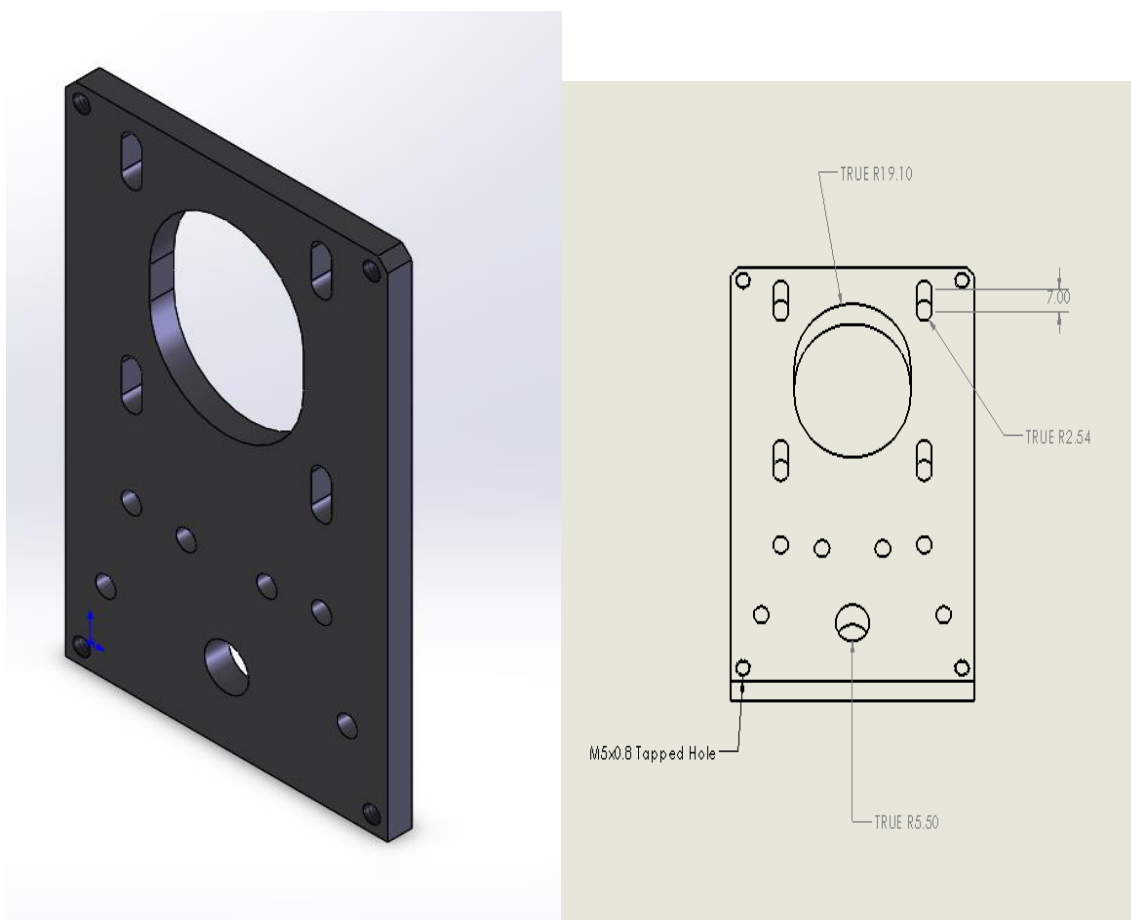
The y-axis acrylic gantry plate was assembled to achieve the motion in y direction over the v-slot using a Nema 17 stepper motor. This assembly consists of a Nema 17 motor to provide rotational motion, four solid V wheels, a 14 tooth timing pulley which was attached to the shaft of the Nema 17 stepper motor using a set screw, four M5 30 mm screws to attach the wheels to the acrylic gantry plate, 6 mm aluminium spacer to maintain clearance between the wheels and the acrylic gantry plate, four nylon hex nuts to secure the wheels in place along with precision shims for alignment and four M3 10 mm screws were used to secure the Nema 17 stepper motor to the acrylic gantry plate.



**Figure 4-** Y axis gantry plate assembly

This assembly would allow the timing belt to run through the 14 tooth timing pulley and achieve a linear motion in the y direction. A similar y-axis gantry plate was assembled for the opposite side using a different orientation for the stepper motor. This enabled the cnc machine to achieve precise motion in the y-axis using two stepper motors rotating in the opposite direction i.e.

One clockwise and another counter clockwise.

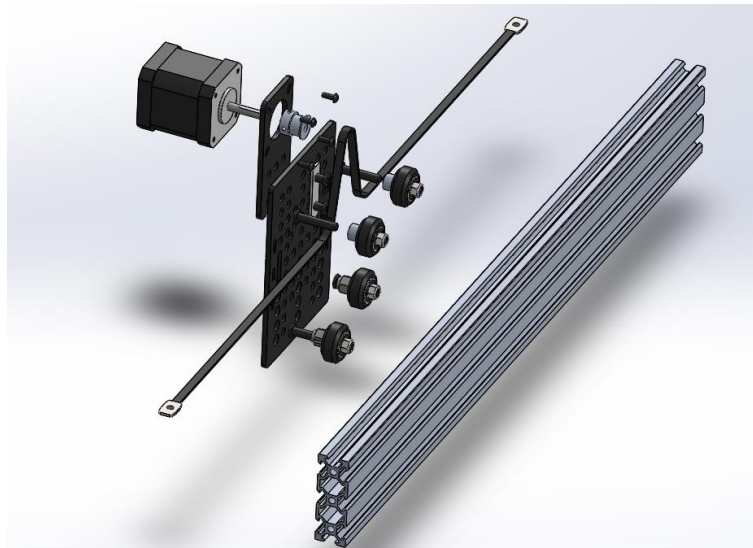


**Figure 5-** Stepper motor mount gantry plate model and drawing

### **3.1.1.2 The X-axis gantry plate assembly**

The motion in the x-axis was achieved by using a Nema 17 stepper motor which connects to a 14 tooth pulley and having a belt run through the pulley. For the assembly of the x-axis stepper motor on the 20 x 40 v-slot, two acrylic carriage plates were used with appropriate holes in position to insert screws for the wheels and the stepper motor. One carriage plate contains a clearance for the shaft of the stepper motor to make it accessible for the pulley and another carriage plate contains slotted and fixed holes for the assembly and adjustment of the wheels. The two plates are attached to hold the stepper motor and wheels between them in a sandwich configuration and use spacers to adjust the spacing between the plates. For the assembly of the wheels, four M5 40mm screws were used along with eight 6mm aluminium spacers, and for the assembly of the stepper motor onto the carriage plates

four M3 10 mm screws were used. To secure the assembly, nylon hex nut and precision shims were utilized. The holes on one carriage plate were aligned with the hole of the stepper motor in a way that the shaft of the stepper motor was accessible for the pulley through a shaft hole and M3 10 mm screws were inserted and secured to attach the stepper motor to the carriage plate. Four M5 4 mm screws were then inserted in the second carriage plate along with the 6mm spacers and precision shim. The wheels were added next on the M5 40 mm screw followed by the 6mm spacers and another set of precision shims. The plates were then attached using the nylon hex nuts to secure them in place. The 14 tooth timing pulley was attached to the shaft of the stepper motor using a set screw on the flat surface to secure it in place.

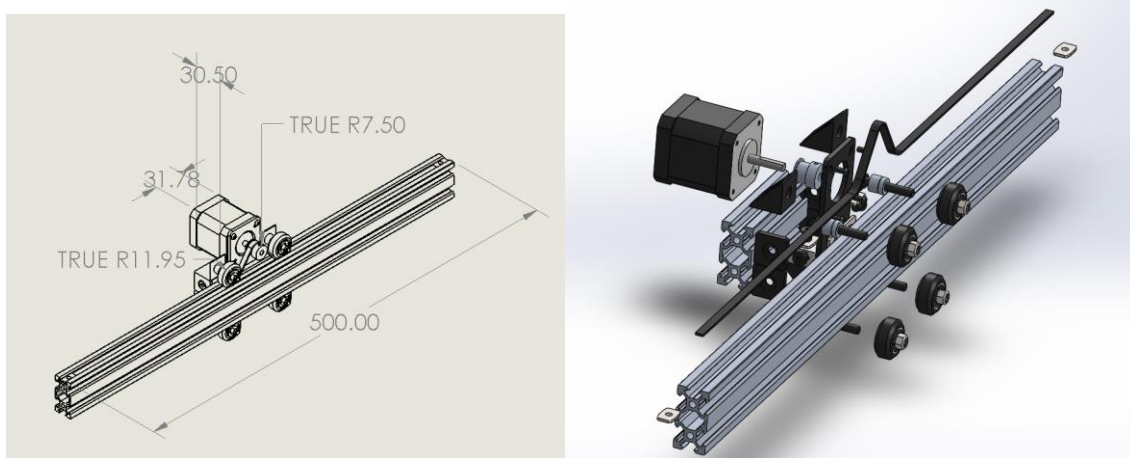


**Figure 6-** V-slot timing belt and pulley assembly

### **3.1.1.3 The XY-axis V-slot assembly**

The assembly of the x-carriage plates was then run across the 20 x 40 mm v-slot by adjusting the preload on the wheels in such a manner that the v-slot was held by two wheels on the bottom and two wheels on the top. The preload on the wheels was adjusted by using the slotted hole where the height of the bottom two wheels can be adjusted to achieve a smooth motion of the carriage plates on the v-slots. A GT-2 timing belt was inserted in the v-slot and attaching it on the 14 tooth pulley. Two

M5 tee nuts were slid in the v slot on each end to secure the belt in place. After adjusting the appropriate tension in the belt two M5 set screws were inserted and secured onto the tee nut. A similar process was applied for the assembly of two y axis v-slot with the y-axis gantry plate assembly. The preload on the wheels was adjusted to achieve a smooth motion on the v-slot followed by the insertion of the GT-2 timing belt in the v-slot. The belt was secure on the pulley and after adjusting appropriate tension on the belt a pair of set screws and M5 tee nuts were used on each side of each axis to secure the belt in place.



**Figure 7-** V-slot timing belt and pulley assembly model and drawing

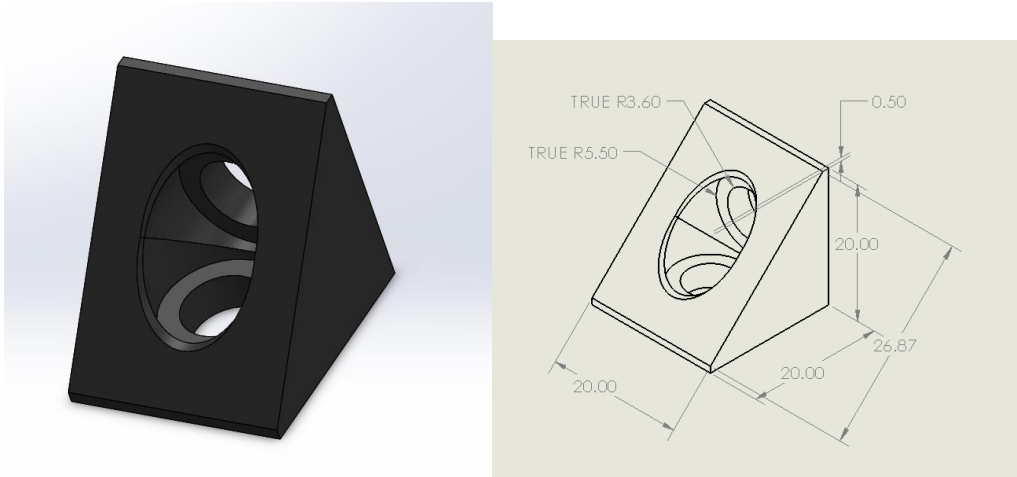
Following the assemblies of the x-axis and y-axis v-slots, acrylic end plates were attached to the y-axis assemblies to add support to the system. These end plates will be used to attach the 20 x 20 v-slots to the assembly to make a rigid assembly. The acrylic end plates include holes for attachment of y-axis 20 x 40 v-slots, two holes for attachment of a double tee nut which will slide in the support 20 x 20 v-slots and a hole at the bottom for the attachment of the L bracket which will be attached to the base assembly. The end plates were attached on each end of the 20 x 40 v-slots using self-tapping screws.



**Figure 8-** X and Y V-slots assembly

#### **3.1.1.4 The CNC frame assembly**

After successfully assembling the x and y axis assemblies, the CNC machine frame assembly was carried out. This assembly includes two y-axis assemblies and one x-axis assembly. The components needed for this assembly includes four M5 15 mm screws, four nylon hex nuts, four self tapping screws and four angled corner connectors to assemble the axes together. The four angled corner connectors were first attached to the ends of the x-axis on each side orthogonally. A tee nut was used along with an M5 10 mm screw for this attachment where the tee nut was slid into the v-slot and tightening the screw held it in place. The y-axis gantry plate consists of four holes for attachment to the x-axis assembly on the top of the plate. It also has slots for the extra GT2 timing belt to pass through the y-axis gantry plates to avoid any undesired spacing between the y-axis gantry plate and x-axis v-slots. The centre two holes were used to attach the end hole on the 20 x 40 v-slot of the x -axis using a self tapping screw. The holes of the sides were connected to the angled corner connector with M5 15 mm screws and secured by the nylon hex nuts of the other side of the y-axis gantry plate. The vertical alignment of the hole prevents any motion in y-direction and the horizontal alignment of the hole prevents any motion in x-direction during the operation of the CNC machine. This process was repeated for both sides of the x-axis assembly, securing the y-axis assemblies on each side of the x-axis v-slot.



**Figure 9-** Angled corner connector model and drawing

The limit switches were added on the left side of both the x and y axis using a M5 10 mm screw and a tee nut.



**Figure 10-** Limit switch assembly

### **3.1.1.5 The supporting frame assembly**

With the completion of assembling x and y axes together, the basic frame of the machine was accomplished with the exception of the supporting 20 x 20 v-slots which will provide support to the y-

axes v-slots. These supporting rails evenly distribute the compression stress due to the weight of the stepper motors, gantry plates assemblies and the v-slots. The end plates assembled earlier with double tee nuts were used to attach additional 20 x 20 v-slots to each side of the frame. The double tee nuts were slid into the supporting v-slots on parallel ends of the y-axis connecting the end plates and to the supporting v-slots parallel to x -axis. Similarly, the second supporting v-slot was attached to the frame of the CNC machine and was secure in place by using the double tee nuts. With the addition of the supporting v-slots the basic frame of the CNC machine was completed. The two stepper motors attached to the y-axis and one stepper motor attached to the x-axis allows the CNC machine to go to any desired position within the x-y plane in the constraints of the framework.

#### **3.1.1.6 The Z-axis gantry plate assembly**

The motion in the z-axis was achieved by using a linear actuator. A gantry plate was attached to the x-axis gantry plate assembly using four M5 15 mm screws and two double tee nuts. The four screws were inserted in the new gantry plate followed by a spacer and then through the slots in the existing gantry plate assembly. The new gantry plate for the attachment of the linear actuator was secure in place by attaching the M5 screws to the double tee nuts of the back of the gantry plate. The purpose of adding this new gantry plate to the existing assembly was to have more attachment options to the current CNC machine. The linear actuator was attached to the new gantry plate by using a 3D printed linear actuator holding bracket. This bracket was modeled using the dimensions of the linear actuator and the spacing of holes on the gantry plate to secure the linear actuator to the CNC assembly. After establishing the linear actuator to the CNC frame, the goal of achieving motion in all three directions i.e. x-y-z was accomplished.

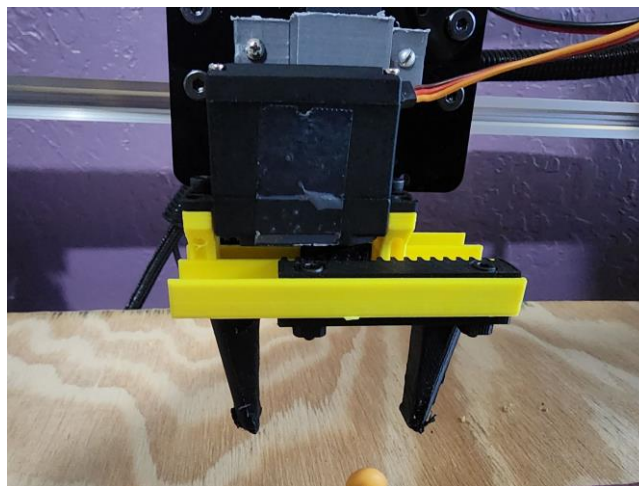
The attachment of the CNC machine to the base was accomplished by using the L-brackets which would attach to the end-plates and the base. The L-brackets contain holes on each face to



establish the connection between the CNC machine and the base. M5 15mm screws were used to secure the L-brackets to the end plates of the CNC machine and 1-1/4" wood screws were used to attach the L-brackets to the wooden base. To support the weight of the CNC machine, four 3D printed mounting brackets were installed in a way where they were attached to the wooden base and firmly held the CNC machine's supporting rails in the slots.

### 3.1.2 The electro-mechanical gripper

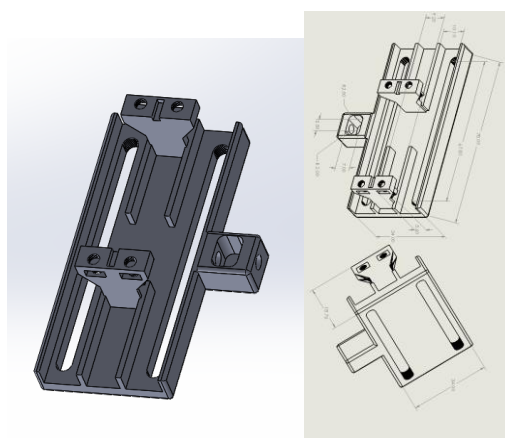
For the organization, dispensation and sorting of the tools, a 3D printed mechanical gripper or clamp was used. This gripper would allow the Smart Tool Organizer to grab and release the tools as required. The assembly of the clamp uses a rack and pinion system in order to achieve the motion of gripper arms to grab a hold of the tools with enough torque and release them. The rack and pinion system operates using a MG996R servo motor and converts the rotation motion of the servo shaft to linear motion of the gripper arms.



**Figure 11-** Electro-Mechanical clamp assembly

The base of the gripper for the robotic arm was designed with slots and holes in place to assemble the racks, pinion, MG996R servo motor, the gripper arm and the connecting linear actuator. The initial design of the base included a mounting slot for the linear actuator aligned the servo

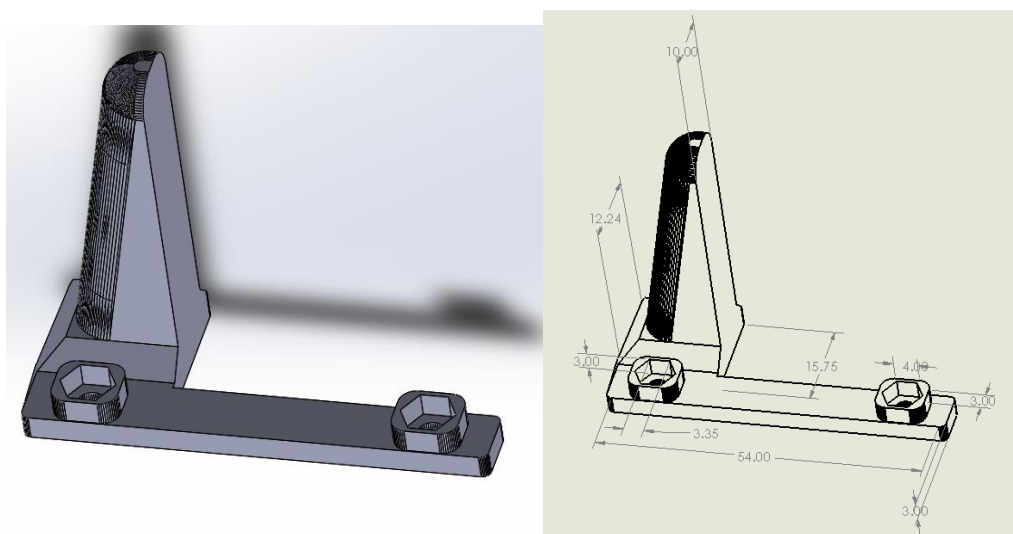
mounting holes but due to the constraints faced during the mounting process and non-alignment of the gripper arms with the desired orientation of the tools on the magnetic strip, modifications were made. The new design of the base of the gripper included the mounting slots for the linear actuator orthogonal to the position of the servo to align the arm of the gripper with the position of the tool on the magnetic strip on the base of the smart tool organizer. The slot of mounting the linear actuator was designed using the dimensions of the shaft of the linear actuator to accommodate a perfect fit of the assembly. The base of the gripper includes two slots for racks and a centre slot for positioning the pinion. The position of the pinion is aligned with the position of the servo motor. The pinion connected directly to the servo motor shaft gear increasing the radius of the servo motor shaft gear which in turn increased the torque provided by the servo motor for the motion of the gripper arms. The pinion is connected to the racks on each side using the linear gears on the rack to convert the rotational motion of the pinion into linear motion of the racks. The holes for mounting the servo are raised to accommodate the clearance between the mounting holes and the shaft of the servo motor. The servo motor was connected to the base of the gripper assembly using four M3 10 mm screws and M3 nuts were inserted in the slots designed below the hole to secure the servo motor on the base. These mounting holes are aligned in such a manner that the pinion fits in its slot after being connected to the servo motor and the gears engage with the rack slots on each side of the pinion.



**Figure 12-** Base of the clamp model and drawing

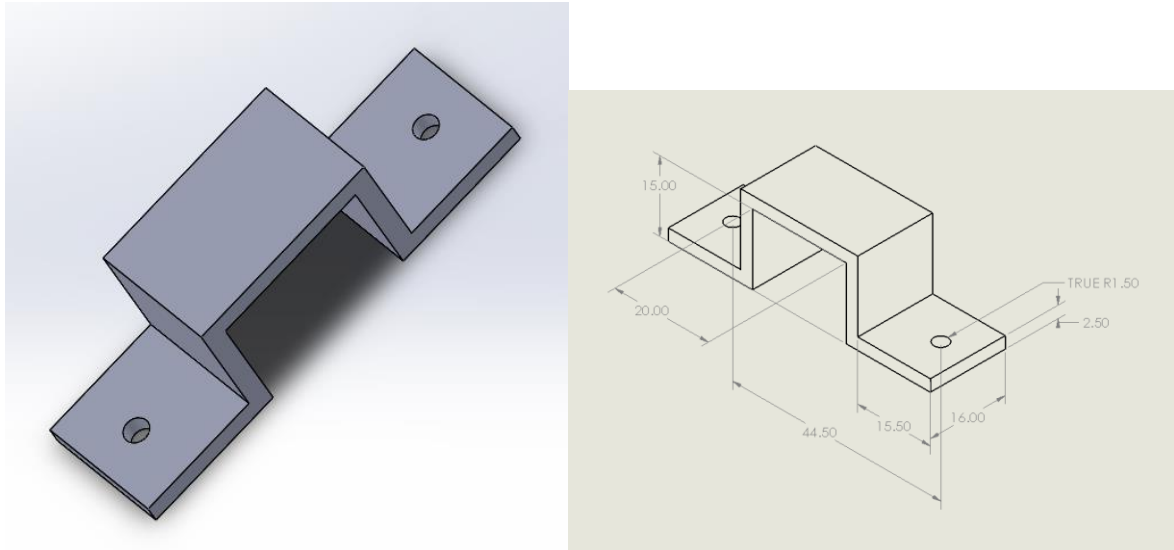
The gripper arms were connected to the rack on the opposite side of the base using four M3

15 mm screws and secured using four M3 nuts. The motion of the rack is translated to the gripper arms using the designed slots where the rack and gripper arms connect. A clockwise motion of the servo motor shaft results in clockwise motion of the pinion which moves the front rack and the gripper arm attached to the rack in left direction and the rack and gripper arm in the back towards the right direction translationally. This causes the gripper arms to close and hold the tools. A rotation of the servo motor in counter-clockwise direction produces the opposite motion in the gripper arm and allows the gripper to open the arms to release the tools.



**Figure 13-** Arm of the clamp model and drawing

The assembly of the gripper was connected to the linear actuator using the designed slot for the shaft of the linear actuator.



**Figure 14-** Linear actuator mounting bracket

The linear actuator was connected to a gantry plate using a 3D printed mounting bracket. This bracket holds the linear actuator in position on the x-axis gantry plate assembly using two 15 mm screws inserted in the holes designed for the screws on the mounting bracket and existing holes on the gantry plate. With this assembly in place, the assembly of the robotic unit of the smart tool organizer was accomplished.

### **3.2 Base of the Smart Tool Organizer**

The assembly of the CNC robotic unit is mounted on a base where the tools will be organized, dispensed and returned to. The base assembly is made of wood and includes styrofoam for holding the tools in place, mounting supports for the robotic unit and a push to open dropbox mechanism for access to tools.



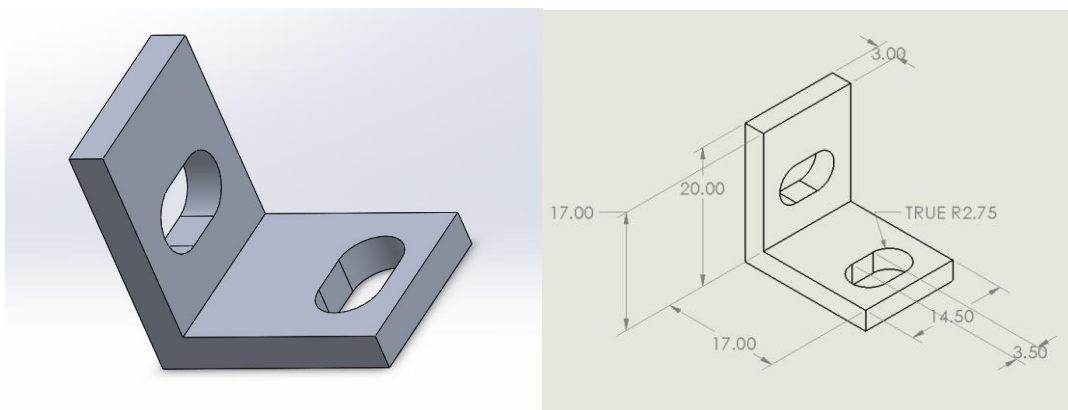
**Figure 15-** Base of the smart tool organizer assembly

For the assembly of the base, a half inch sheet of hardwood was used. This wood sheet was cut according to the dimension of the CNC machine base and a cut out for the dropbox was made. The clearance between the base and the CNC machine was measured and 3” of clearance for the linear actuator to extend fully and the robotic gripper arm to grab a hold of tools. The gantry plate provided a couple inches of adjustment for the linear actuator which allows the linear actuator to be lowered or raised accordingly. A 2 x 2” square pressure treated weather shield wood was used to design and make mounting supports for the robotic unit with the calculated clearance. Two styrofoam strips were used to hold the tools in position, one for the organization and storage of the tools within the smart tool organizer and another one for the dropbox for dispensation and retrieval of the tools. The position of these styrofoam strips was calculated using the working area of the robotic unit to ensure the accessibility of the tools to the gripper arm. These styrofoam strips were secured in place using four mounting tape. The four mounting supports for the robotic units were inserted in four corners of the base using 1-½ inch #10 wood screws after measuring the height of the supports to provide 3” clearance.



**Figure 16-** Supporting and mounting brackets for CNC machine

The CNC machine was mounted on the supporting brackets using L brackets which were attached to the supporting bracket using ½” wood screws and to the base of the CNC end plates using a M5 screw and nut.



**Figure 17-** L bracket model and drawing

A push to open mechanism was installed to make the dropbox accessible to the user. This mechanism was attached to the base using an assembly of four 12 x 2.5” wooden brackets which were attached to both faces of the push to open mechanical sliders using ½” wooden screws. Two

assemblies of mechanical slider were made and were attached to the base and the 12 x 4" dropbox designed with 1" wooden screws for accessibility. The base of the smart tool organizer was mounted on four 2 x 2 x 3" mounts to provide clearance for the dropbox and raise the smart tool organizer 3" above the table.



**Figure 18-** Push to Open dropbox assembly

# CHAPTER 4

## Electronics and Controls

### 4. Electronic components and software

The Smart Tool Organizer is designed to organize, store, track and trade tools automatically using an interface to communicate with the user. To achieve this outcome it utilizes various electronic components which enables the automation of the prototype. Along with the electronic components, automation also involves extensive programming of both the controller and the interface device to achieve successful outcomes. The design of the Smart Tool Organizer incorporates electronics such as Nema 17 stepper motors, MG996R servo, linear actuator, limit switches, buck converter, DS3231 RTC module, SD card module, L298N motor drive module, TB6600 motor drivers, a 7" touch screen LED display and an arduino mega. All these components work simultaneously to achieve a shared goal. In this chapter, the components used in Smart Tool Organizer and their controls are discussed.

#### 4.1 High voltage circuit

The High voltage circuit operates on 12 V and includes components such as stepper motors, TB6600 motor driver module, linear actuator, L298N motor drive module, MG996R servo motor and limit switches. These components and their programming are discussed in detail in this section.

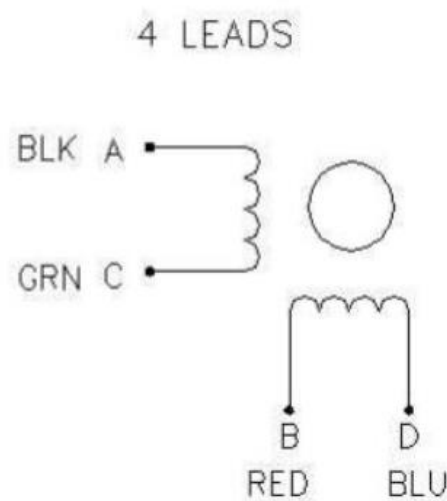
##### 4.1.1.1 Nema 17 Stepper Motors and TB6600 motor driver

The main component driving the CNC machine in the robotic unit of the smart tool organizer is Nema 17 stepper motors. These motors were installed on the gantry plates attached to the wheels and the rails of the CNC machine. Two stepper motors are used to drive the y-axis, and one to drive the x-axis. The Nema 17 stepper motor offers precise control over the positioning of the electro-



mechanical gripper which will be used to transfer tools to various positions. The shaft of the stepper motor is connected to a 14 tooth pulley which translates the rotational motion of the stepper motor to wheels attached to the rails using a connected 14 tooth timing belt. The assembly of the gantry plate moves translationally through the rotation of the pulley through the fixed belt as the rotation pushes the assembly forward or backward depending on the direction of rotation. This causes the four wheels to rotate over the slots in v-slot rails providing translation motion in one dimension. The y-axis stepper motors rotate at the same speed and in opposite direction to achieve linear motion in y direction and the x-axis stepper motor provides the linear motion in x direction.

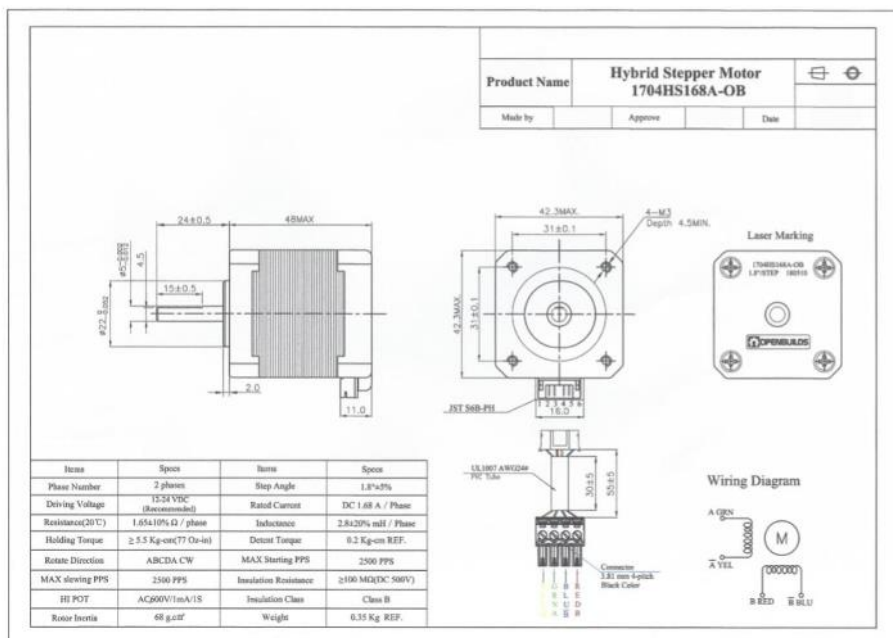
The Nema 17 stepper motor used in the CNC machine of the robotic unit is a unipolar, 4 phase stepper which uses a permanent magnet. This motor has two split windings and four wires attached to those windings. As shown in the figure 19 below, two wires are used in the first winding in which black and green wire acts as coil ends. Similarly, in the second winding, the red and blue wires act as the coil ends. During the wiring of the stepper motor, the ends of the coils are connected to the ground and powered alternatively by a motor driver. The order A+, B+, A-, B- is followed in the stator poles inside the motor arrangement.



**Figure 19-** Stepper motor winding diagram [20]

The motor faceplate is 1.7 x 1.7 inches and the diameter of the shaft of the motor is 5 mm.

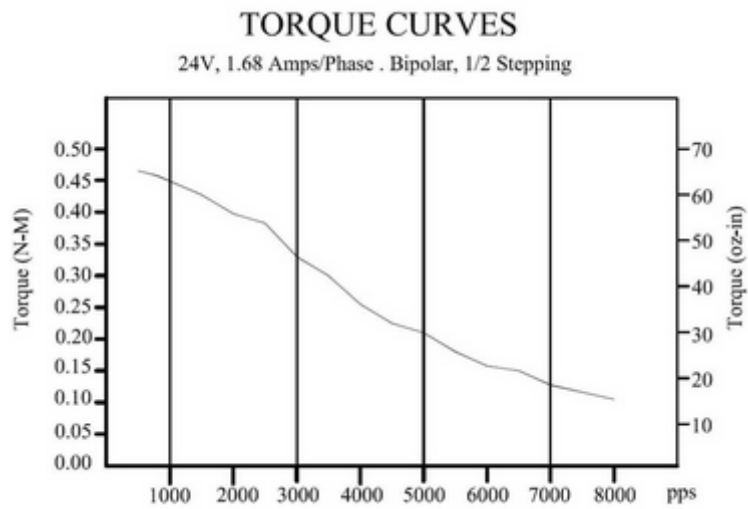
The dimensions of the Nema 17 stepper motor is shown in the figure 20 below.



**Figure 20-** Nema 17 stepper motor dimensional drawing [22]

It has a step angle of 1.8 degrees and to complete one full revolution, the motor takes 200 steps. The motor is operated on 12 V, 400 mA current with a peak current of 1.68A/phase[21] [22].

The three stepper motors are connected in parallel using a solder breadboard and supplied with the 12 volts using a 12 V power supply. The torque of the stepper motor is 76 oz\*in and it changes with the numbers of phases per second.



**Figure 21-** Nema 17 Stepper motor torque curve [22]

| Stepper Motor Specifications         |                                  |
|--------------------------------------|----------------------------------|
| <b>Mosaic Part No.:</b>              | STEPMOT-1                        |
| <b>Manufacturer Part No.:</b>        | 42BYG228                         |
| <b>Size:</b>                         | NEMA 17                          |
| <b>Drive system:</b>                 | Unipolar                         |
| <b>Step angle:</b>                   | 1.8° full step<br>0.9° half-step |
| <b>Phase/Windings:</b>               | 4/2                              |
| <b>Voltage &amp; Current:</b>        | 12V at 400 mA                    |
| <b>Resistance per Phase:</b>         | 30 ohms                          |
| <b>Inductance per Phase:</b>         | 23 mH                            |
| <b>Holding Torque:</b>               | 2000 g-cm                        |
| <b>Detent Torque:</b>                | 220 g-cm max                     |
| <b>Weight:</b>                       | 0.24 kg (0.5 lbs.)               |
| <b>Max continuous power:</b>         | 5 W                              |
| <b>Rotor Inertia:</b>                | 22 g-cm <sup>2</sup>             |
| <b>Bearings:</b>                     | Ball                             |
| <b>Leads:</b>                        | 18 in. 26 AWG UL 1007            |
| <b>Insulation resistance:</b>        | >100 MΩ at 500VDC                |
| <b>Dielectric strength:</b>          | 500V 50Hz/minute                 |
| <b>Mounting hole space diagonal:</b> | 1.73 in.                         |
| <b>Mounting screws:</b>              | 3 mm dia.<br>0.5 mm pitch        |
| <b>Shaft diameter:</b>               | 0.197 in. (5 mm)                 |
| <b>Motor footprint:</b>              | 1.7 in. × 1.7 in.                |
| <b>Motor height:</b>                 | 1.5 in.                          |
| <b>Ambient temperature:</b>          | -10°C to +55°C                   |

**Table 1-** Nema 17 stepper motor specifications [21]

The stepper motors in the smart tool organizer’s CNC machine are operated using a TB6600 motor driver. The TB6600 motor driver comes with overheating protection, under-voltage and over-current shutdown to protect and ensure the safety of the stepper motor operation. The TB6600 motor driver operating voltage is 9 - 42 V DC and 3.5 A per phase current with peak current being 4 A. the Tb6600 offers a wide range of microstep resolution including full microstep resolution, 1/2, 1/4, 1/8, 1/16 and 1/32 microstep resolution. This motor driver has a clock frequency of 200 kHz [23].

The TB6600 motor driver is connected to the stepper motor using the four wires from the stepper motor connected to A+, A-, B+ and B- pins on the TB6600. The VCC is connected to the power supply of 12 V and GND is connected to the ground of the power supply. The TB6600 module is connected to an arduino mega using DIR+, DIR-, Pul+ and Pul- pins to receive the signals from the arduino [23]. The DIR- and PUL- pins of the TB6600 module are connected to the ground and DIR+ and PUL+ is connected to digital pins of the arduino.

The microsteps and current supplied to the TB6600 module can be controlled using the DIP switches. The settings of microsteps and current using the DIP switch is printed on top of the TB6600 module. For the application of the stepper motor in the CNC machine, the microsteps are set to a full step meaning the stepper will make 200 steps per revolution and step size is 1.8. The current is set to 0.5 A with a peak current of 0.7 A according to the requirement of current from the stepper motor. The following **table** shows the adjustable microstep and current settings on a TB6600 motor driver.

| S1  | S2  | S3  | Microstep resolution | Pulse/rev | Current (A) | Peak current | S4  | S5  | S6  |
|-----|-----|-----|----------------------|-----------|-------------|--------------|-----|-----|-----|
| ON  | ON  | ON  | NC                   | NC        | 0.5         | 0.7          | ON  | ON  | ON  |
| ON  | ON  | OFF | Full step            | 200       | 1.0         | 1.2          | ON  | OFF | ON  |
| ON  | OFF | ON  | 1/2 step             | 400       | 1.5         | 1.7          | ON  | ON  | OFF |
| OFF | ON  | ON  | 1/2 step             | 400       | 2.0         | 2.2          | ON  | OFF | OFF |
| ON  | OFF | OFF | 1/4 step             | 800       | 2.5         | 2.7          | OFF | ON  | ON  |
| OFF | ON  | OFF | 1/8 step             | 1600      | 2.8         | 2.9          | OFF | OFF | ON  |
| OFF | OFF | ON  | 1/16 step            | 3200      | 3.0         | 3.2          | OFF | ON  | OFF |
| OFF | OFF | OFF | 1/32 step            | 6400      | 3.5         | 4.0          | OFF | OFF | OFF |

**Table 2-** TB6600 microstep and current settings [23]

#### 4.1.1.2 Programming of Nema 17 Stepper Motors and TB6600 motor driver

All three stepper motors in the CNC machine are connected to TB6600 motor drivers which receive signals from arduino mega for operational instructions. For the programming of the TB6600 motor driver, the arduino IDE was used. In the programming, first the signal pins on TB6600 module i.e. DIR+ and PUL+ were declared for all motors according to the connection of the digital pins on the arduino. To do this, #define function was used and pins for x and y-axis stepper motors were defined as follows:

```
#define dirPinA 4
#define stepPinAB 5
#define dirPinB 6
#define stepPinC 3
#define dirPinC 2
```

**Figure 22-** Code defining direction and pulse for stepper motor

Here, dirPinA refers to the direction pin of the y2 motor, dirPinB refers to the direction pin of the y1 motor and dirPinC refers to the direction pin of the x-axis motor. The direction pin sets the direction of revolution for the motors. According to the connection of the stepper motors to the TB6600 module, if the direction pin is set to high, it will result in a clockwise rotation and if the direction pin of the motor is set to low, it will result in counter-clockwise rotation. These pins are changed from low to high and vice versa depending on the direction of the x and y axis. The dirPinA is connected to digital pin 4 on the arduino, dirPinB is connected to the digital pin 6 on the arduino and the dirPinC is connected to the digital pin 2 of the arduino as shown in the code above. The stepPins are the PUL+ on the TB6600 motor driver and it represents the number of microsteps a stepper will rotate. Since the stepper motors on the y-axis are intended to move the same number of microsteps, the stepPins on the TB6600 motor driver for y1 and y2 were combined on the breadboard

and set to the same digital pin 5 on the arduino. The stepPinC is the microstep pin for the x-axis motor and is connected to digital pin 2 on the arduino mega.

Next, in the setup loop of the arduino IDE, these pins were set as outputs using the pinMode command. The dirPins and the stepPins will send the signal from the arduino to the TB6600 for the operating instructions of the stepper motors.

```
pinMode(stepPinAB, OUTPUT);  
pinMode(dirPinA, OUTPUT);  
pinMode(dirPinB, OUTPUT);  
pinMode(dirPinC, OUTPUT);  
pinMode(stepPinC, OUTPUT);
```

**Figure 23-** Code setting stepper motor pins as output

To achieve the rotation from the stepper motors, first the direction is set either clockwise or counterclockwise depending on the objective of the CNC machine and then a number of microsteps are defined according to the desired location of the x and y axis. To set the direction of the digitalWrite function was used to set dirPin to either high or low. The y axis motors are rotated in opposite directions to achieve linear motion in one direction as explained in chapter 3. The following figure 24 shows an example of setting direction for x and y axis stepper motors.

```
digitalWrite(dirPinA, HIGH);  
digitalWrite(dirPinB, LOW);  
digitalWrite(dirPinC, HIGH);
```

**Figure 24-** Code setting stepper motor direction

Once the directions for the stepper motors are set, the microstep pins are set to high to achieve the rotation of the stepper motors and then set to low to stop the stepper motor. Since the microstep is

set to a full step as explained earlier, one microstep will result in the motor turning 1.8 degrees and it will take 200 microsteps for the motor to complete a full revolution. To achieve the rotation of the stepper motor shaft multiple times, for function was used with an integer i, such that if i is less than the number specified the for function will repeat. In the for function, the stepPins of the arduino were set to high and after a certain delay were set to low to achieve continuous rotation of the motors. The following figure 25 illustrates an example of the rotation of the stepper motors.

```
for (int i = 0; i < 1500; i++) {  
    digitalWrite(stepPinAB, HIGH);  
    delayMicroseconds(1000);  
    digitalWrite(stepPinAB, LOW);  
    delayMicroseconds(1000);  
}  
for (int i = 0; i < 500; i++) {  
    digitalWrite(stepPinC, HIGH);  
    delayMicroseconds(1000);  
    digitalWrite(stepPinC, LOW);  
    delayMicroseconds(1000);  
}
```

**Figure 25-** Code for rotation of the stepper motor

This programming format was used to control the linear motion of the x and y axis with the goal of taking the electro-mechanical clamp to the desired position of different tools and dropbox. This enabled the CNC machine to retrieve and dispense various tools for various positions on the base of the smart tool organizer assembly.

#### 4.1.2.1 Linear actuator and L298N motor drive module

A linear actuator was used to achieve the motion in the z axis. This linear actuator was selected due to the size and the power requirements of the linear actuator. The length of the linear actuator motor bracket is 40.5 mm, the minimum installing length is 2.97" and maximum installing length is 3.77" [24]. The size of the linear actuator was appropriate for using it in the robotic unit. The linear actuator operates on a 12 V dc power supply which is provided using the power supply for the robotic unit by connecting the linear actuator in parallel with the stepper motors on the breadboard.

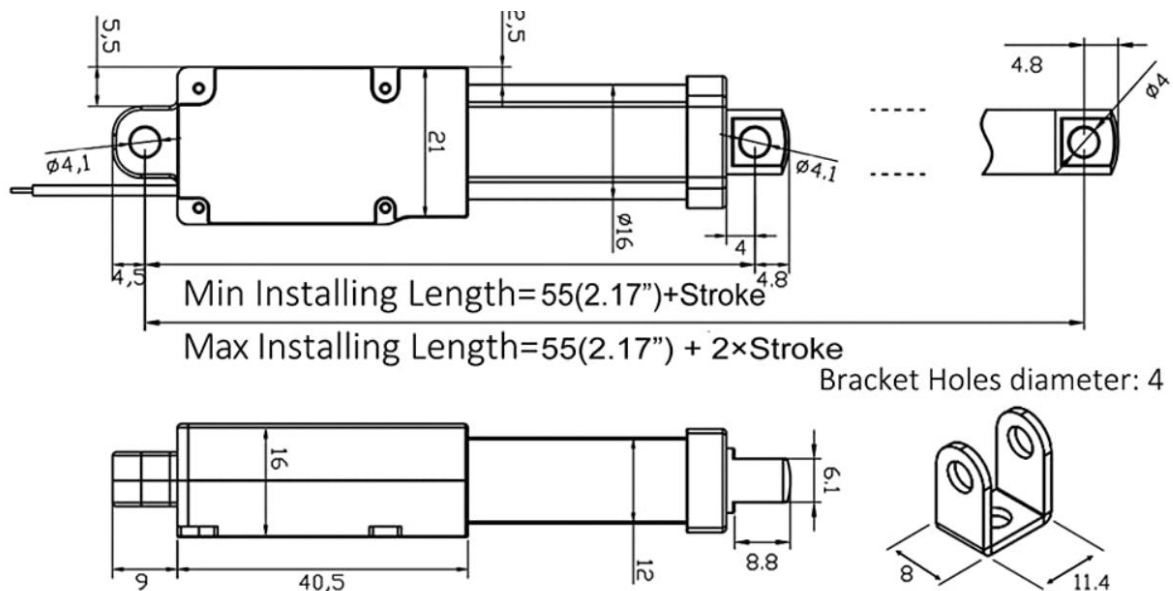


Figure 26- Linear actuator dimensional drawings [24]

The main components of the linear actuator are motor, limit switch, DC brushes, lead screw, gears and cylinder. The 12 V motor inside the linear actuator is responsible for the motion of the linear actuator. The limit switches in the linear actuator control the limit of the extension and retraction of the linear actuator. When the limit switches are triggered, it stops the motor avoiding any damage that could occur by over extending or retracting the actuator. The dc brushes conduct the current to the wires and the actuators. The lead screw provides linear motion by converting the rotational motion of the motor to linear motion and moves up and down depending on the polarity of



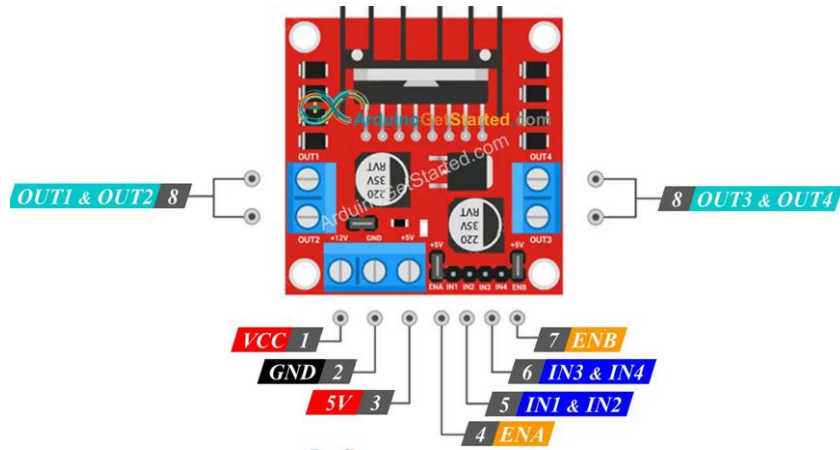
the motor. The lead screw is connected to the motor using gears which facilitates the motion of the actuator [26]. The cylinder contains the interconnected parts of the linear actuator and allows the shaft to move freely inside it.

|                      |                   |                         |   |
|----------------------|-------------------|-------------------------|---|
| <b>Material</b>      | Aluminum alloy    | <b>Duty cycle</b>       | 10%   |
| <b>Load capacity</b> | 4.5lbs / 20N      | <b>Waterproof</b>       | IP54  |
| <b>Speed</b>         | 50mm/s(1.97"/sec) | <b>Noise</b>            | < 50 dB   |
| <b>Input Voltage</b> | DC 12V            | <b>Package included</b> | 1 x Mini Linear Actuator Motor, 2 x Mounting Brackets |

**Table 3-** Linear actuator specifications [24]

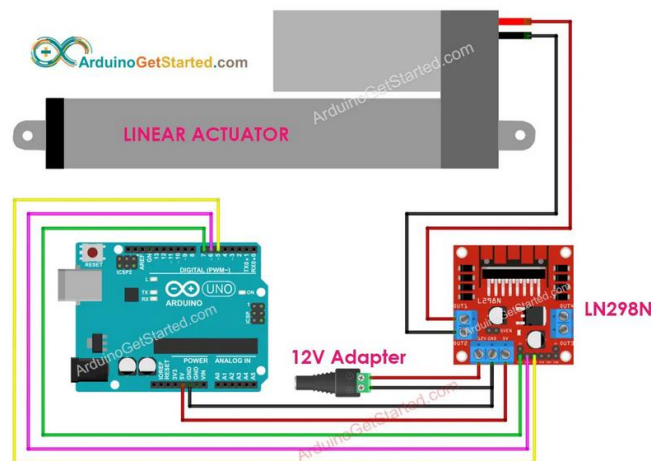
For the operation of the linear actuator, a 12 v power source is connected to the two wires from the linear actuator. If the 12 V positive side is connected to the positive wire and the negative side is connected to the ground, the linear actuator extends at 50mm/s. Similarly, if the linear actuator's negative wire is connected to the 12 V power source positive side and the positive wire of the linear actuator is connected to the ground, the linear actuator retracts at the same speed until it reaches the limit. If the power source is cut off during the extension of the retraction of the linear actuator, it holds the position of the shaft and stops the motion. The speed of the linear actuator can be controlled by decreasing the supplied voltage which will cause the linear actuator to move at slower speed. The speed of the linear actuator can also be controlled using a PWM such that if the duty cycle of the PWM is increased, it increases the speed of the linear actuator.

The linear actuator is controlled by an arduino mega which sends the linear actuator signal to extend or retract. But since the arduino cannot provide the linear actuator with 12 V power, an L298N motor drive module was used. The L298N motor drive module receives the signal from the arduino and amplifies it and it allows the arduino to change the polarity of the signal making the linear actuator extend or retract while keeping the same connection [25].



**Figure 27-** L298N module pinouts [25]

The L298N module is capable of controlling two linear actuators simultaneously since it has two channels. For this application, only channel 1 will be used. The Vcc pin is connected to a 12 V power source which supplies power to the linear actuator and 5 V pin is connected to a 5 V power source which supplies power to the L298 motor drive module. The 5 V power is received from the same power source by using a buck converter which will be discussed later in this chapter. The ground pin is connected to ground. The ENA pin is used to enable the L298N module and can also be used to control the PWM input and consequently the speed to the linear actuator. The IN1 and IN2 pins are connected to the arduino digital pins and OUT1 and OUT2 pins are connected to the wire of the linear actuator.



**Figure 28-** Wiring diagram of linear actuator and L298N module [25]

#### 4.1.2.2 Programming of the Linear actuator and L298N motor drive module

The linear actuator and L298N are programmed to extend and retract for location the electro-mechanical clamp in position where it has close proximity with tools and is able to grab or drop the tools.

The linear actuator extends when the CNC machine or x and y axis reaches its desired position or the position of the desired tool. This allows the clamp to grab the tool using its mechanical arms. Once the clamp holds the tool, the linear actuator retracts and it takes the tool to the required position using x and y axis stepper motors. This process repeats many times depending on the required output from the smart tool organizer.

For the programming of the L298N module, first the digital pins connected to the arduino were declared. Three pins on L298N are connected to the arduino i.e., ENA, IN1 and IN2. The ENA pin is connected to digital pin 7 on arduino mega, the IN1 pin is connected to digital pin 8 on arduino mega and IN2 pin is connected to digital pin 9 on arduino mega. These pins were declared using const int function.

```
const int ENA_PIN = 7;  
const int IN1_PIN = 8;  
const int IN2_PIN = 9;
```

**Figure 29-** Code defining L298N pins

After declaring the pins connections, the pins were set as outputs since the L298N motor drive module and linear actuator will receive signals from the arduino. To do this, pinMode function was used and the three pins ENA, IN1 and IN2 were set as output pins. The ENA pin was set to high

to enable the L298N module. This pin can also be used to limit the extension and retraction and the speed of the linear actuator but it was not necessary in this application.

```
pinMode(ENA_PIN, OUTPUT);  
pinMode(IN1_PIN, OUTPUT);  
pinMode(IN2_PIN, OUTPUT);  
  
digitalWrite(ENA_PIN, HIGH);
```

**Figure 30-** Code setting and enabling L298N pins

Next, to extend the linear actuator, the IN1 pin was set to high and IN2 pin was set to low using digitalWrite function. Similarly, to retract the linear actuator, the IN1 pin was set to low and the IN2 pin was set to high. By doing this, the L298N module reverses the polarity of the motor in the linear actuator causing it to rotate in opposite direction and the shaft of the linear actuator to retract. Some delays were added before and after the digitalWrite command to make sure the linear actuator extends and retracts fully.

```
delay(2000);  
digitalWrite(IN1_PIN, HIGH);  
digitalWrite(IN2_PIN, LOW);  
delay(2000);
```

**Figure 31-** Code for the linear motion of linear actuator

This programming format was used throughout the code to extend and retract the linear actuator as necessary.

### 4.1.3.1 Electro-Mechanical clamp and servo motor

The electro-mechanical clamp is the component which grabs and drops the tools as needed in the smart tool organizer. The clamp is attached to the shaft of the linear actuator as discussed in chapter 2. This clamp operates using a MG996R servo motor and converts the rotation of the servo motor into linear motion using the gear and pinion mechanism discussed in chapter 2. The clockwise rotation of the servo motor causes the mechanical arm of the clamp to close and the counterclockwise rotation of the clamp causes the mechanical arm to open.

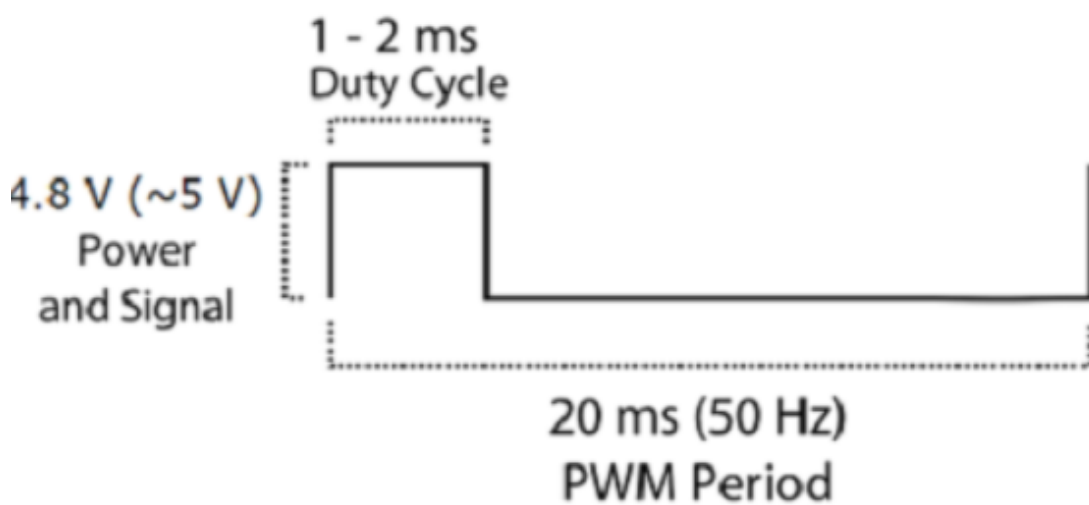
The MG996R operates using the duty cycle of the PWM signal received from the arduino mega. This motor has a stall torque of 9 kg/cm with maximum stall torque of 11 kg/cm and has a metal gear. The stall torque is more than enough for this application since the motor can hold the tool up to 11 kg at a distance of 1 cm. The motor has 3 connections i.e. Power, ground and signal. The brown wire of the servo motor is the ground connection, the red wire is power connection and the orange wire is the PWM signal connection which will be connected to the arduino. This motor operates on 5 V and 2.5 A current. The torque of the motor increases with increase in volts and has a rotation angle of 0 to 180 degrees. The circuit can be modified in order to achieve full 360 degrees of rotation. This motor typically operates at 0.17s/60 degree speed [27].

#### **MG996R Servo Motor Features**

- Operating Voltage is +5V typically
- Current: 2.5A (6V)
- Stall Torque: 9.4 kg/cm (at 4.8V)
- Maximum Stall Torque: 11 kg/cm (6V)
- Operating speed is 0.17 s/60°
- Gear Type: Metal
- Rotation : 0°-180°
- Weight of motor : 55gm
- Package includes gear horns and screws

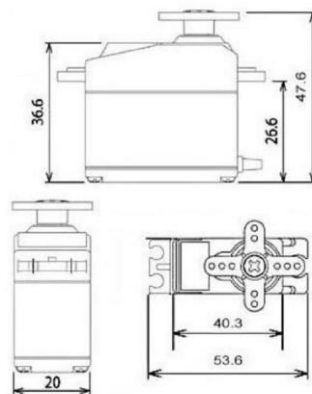
**Table 4-MG996R servo motor specifications [27]**

The PWM signal produced by the arduino has a frequency of 50 HZ or 20 ms. The degree of rotation of the motor can be controlled by using the on-time of the PWM signal which can vary from 1 to 2 ms. This means at 1 ms of on-time the motor will be at 0 degree and 2 ms of on-time the motor will be at 180 degree [27]. The interval being the 1 and 2 ms of on-time will result in the motor being at an intermediate degree such as at 1.5 ms the motor will be at 90 degrees.



**Figure 32-** PWM signal representation of servo motor [27]

The MR996R servo motor is 47.6 mm in overall length and has 20 mm thickness. The width of the screw mounts of the servo motor is 53.6 mm and the body of the servo motor is 40.3 mm wide [27].



**Figure 33-** Dimensional drawing of servo motor [27]

In the previous chapter, it was discussed how the servo motor is mounted using 4 screws onto the base of the mechanical clamp. The gear of the servo motor interacts with the linear gears on the rack using another gear which converts the rotational motion into linear motion.

#### 4.1.3.2 Programming of the Electro-Mechanical clamp and servo motor

The function of the electro-mechanical clamp and the servo motor is to grab and drop the tools as required. To do this the servo motor needs to rotate at various degrees and hold the tool in position till it is transported to the destination. The degree of rotation of the servo motor depends on the width of the tool it is grabbing which varies with different tools. Each tool was calibrated for different degrees of rotation of the servo motor to achieve this goal. The servo motor was powered using the same power source and was connected in parallel with the linear actuator. The signal for the servo motor was connected to arduino mega pin 10.

The programming of the servo motor was accomplished using the Servo.h in-built library in the arduino. First the library was included using the #include function and then an object named myservo was created using the Servo function. An integer was created next by the name of pos, short of position of the servo, and was set to 0 for the home position. This integer will be changed according to the required degrees of rotation from the servo.

```
#include <Servo.h>

Servo myservo;
int pos = 0;
```

**Figure 34-** Code including servo library and defining variable

The home position of the servo motor is set to 0 degrees. To enable the servo myservo.attach

function was used with the pin 10. This is an in-built function which needs to be called out before operating the servo motor. A for function was used to increase the degrees of rotation of the servo motor with the increment of one step. The function `myservo.write` was used to assign the value of `pos` to the servo which was used in the for loop. A delay of 15 ms was included for the servo to get in position. After the operation, the function `myservo.detach` is used to disable the servo motor. This process can be used to increase or decrease the degrees of rotation of the servo motor or in other words, open or close the mechanical clamp.

```
myservo.attach(10);
for (pos = 0; pos <= 120; pos += 1) {
  myservo.write(pos);
  delay(15);
}
myservo.detach();
```

**Figure 35-** Code for the rotational motion of servo motor

This format of programming was used throughout the code whenever the servo was needed to change the degree of rotation to grab or drop the tool and to bring the servo back to the home position.

#### **4.1.4.1 Limit switches**

Two limit switches were used on the CNC machine x and y axis to stop the rotational motion of the stepper motors when the gantry plates had reached the end of the axis. When the limit switch is triggered it disables the motors and stops the linear motion of the x and y gantry plate assemblies.

These limit switches were also used to specify a home position for the CNC machine where it would begin the operation and come back after the operation of the CNC machine.

The limit switches used in the assembly of the CNC machine operate on 12 V by using a PCB and have three wire connections. These wired connections include power +12 V wire, ground wire



and a signal wire. The power wire was connected to the 12 V power supply in parallel with the stepper motors, the ground wire was connected to the ground and the signal wire was connected to analog pins of arduino mega. The limit switch used in the assembly is normally open meaning when the switch is triggered by contact of the gantry plate it will close and send a signal to the arduino to stop the rotation of the stepper motors.



**Figure 36-Limit switch positioning on a V-slot**

#### **4.1.4.2 Programming of the Limit switches**

The limit switches were programmed to avoid any damage that would occur by collision of the x and y axis gantry plates and to specify a home position for the CNC machine. The signal pins from the limit switches were connected to arduino analog pins where x limit switch was connected to analog pin A1 on the arduino and y limit switch was connected to analog pin A0 of the arduino.

First the limit switches pins were declared using the #define function setting limitX to A1 and limitY to A0. For each action performed by the CNC machine an if loop was included before the action which made sure that the CNC machine was in home position. This was accomplished using conditions in the if loop which send signals to the arduino if the limit switches were triggered. This

made sure that the CNC machine only operated if the conditions were met or in other words the limit switches were triggered.

```
if ((digitalRead(limitY) == 0) && (digitalRead(limitX) == 0)) {
```

**Figure 37-** Limit switch conditions

After each operation of the CNC machine, the stepper motors were sent back to the home position. This was accomplished by first using an if condition meaning that if the limit switches were not triggered do the following. Then the direction for the home position was set using the digitalWrite function on dirPins. A do while loop was used next to bring the steppers back to the home position which told stepper to move toward the limit switches while the limit switches were not triggered. The homing of the stepper motor was carried out for both the x and y axis. A delay function was added for 2 sec after the homing to prepare the CNC machine to operate again.

```
if (!digitalRead(limitY) == 0) {  
    //direction back to HOME position  
    digitalWrite(dirPinA, LOW); //DOWN  
    digitalWrite(dirPinB, HIGH); //DOWN  
    //  
    do {  
        //moves Y-steppers till Y-limit is ON  
        digitalWrite(stepPinAB, HIGH);  
        delayMicroseconds(1000);  
        digitalWrite(stepPinAB, LOW);  
        delayMicroseconds(1000);  
    } while (!digitalRead(limitY) == 0);  
}  
if (!digitalRead(limitX) == 0) {  
    digitalWrite(dirPinC, LOW); //LEFT  
    do {  
        //moves X-stepper till X-limit is ON  
        digitalWrite(stepPinC, HIGH);  
        delayMicroseconds(1000);  
        digitalWrite(stepPinC, LOW);  
        delayMicroseconds(1000);  
    } while (!digitalRead(limitX) == 0);  
    delay(2000);  
}
```

**Figure 38-** Code for bringing stepper back to home position

This formatting of programming was used throughout the code whenever a tool retrieval or dissension request was placed.

## **4.2 Low voltage circuit**

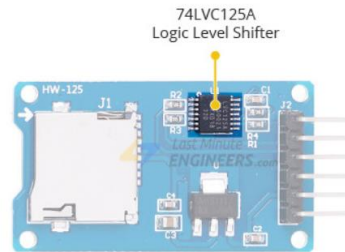
The low voltage circuit operates on 5 V and includes components such as SD card module, DS3231 RTC module, nextion display and arduino mega. The nextion display is discussed in detail in chapter 5 interface and firmware. The low voltage components were operated on the same 12 V power supply after reducing the voltage to 5 V using a buck converter. The buck converter comes with a potentiometer which can be turned to reduce the amount of outlet voltage. One side of the buck converter was connected to the 12 V power source and ground and the other side was connected to the opposite side of the breadboard which provided power to components requiring 5 V.

### **4.2.1.1 SD card module**

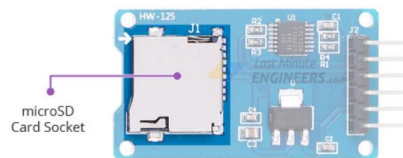
The SD card module is used to store data and entry logs of the user of the smart tool organizer. Anytime a user requests the dispensation or returns a tool the user's student ID number is stored along with the returned or dispensed tool number and date and time on a text file in the SD card. This data can be reviewed to determine which user received or returned which tool from the smart tool organizer with the exact date and time data. This data is very helpful to keep track of tools and manage and organize tools in the smart tool organizer.

The SD card module used for this application is Hiletgo SD card module with 6 pins SPI interface. This module has an on-board 3.3V LDO voltage regulator which converts the supplied 5V of dc power to 3.3 V. This voltage regulator is necessary because any voltage above 3.6 V may permanently damage the SD card. The SD card module also includes a 74LVC125A logic level shifter

chip which allows the communication between the SD card module and the microcontroller which is an arduino mega for this application. The SD card module includes a microSD memory card pocket where the microSD card up to 16 GB can be installed.



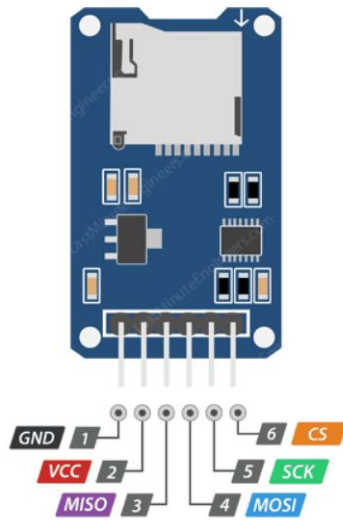
**Figure 39-** Logic level shifter [29]



**Figure 40-**microSD card slot [29]

The SD card module comes with six pins such as VCC, GND, MISO, MOSI, SCK and CS.

Here, the VCC is the power pin which is connected to the 5V power source, the GND pin is connected to the ground. The MISO (Master in Slave Out) is an SPI output for the SD card module, the MOSI (Master Out Slave In) is an SPI input for the SD card module, the SCK (Serial Clock) pin synchronizes the data transmission by using the clock pulse from the master or arduino mega, the CS pin regulates the SPI bus by selecting the control pin to select a slave [29].



**Figure 41-**SD card module pinouts [29]

The SD card module only accepts the microSD card formatted in FAT16 or FAT32 format. The microSD card used in the application was formatted in FAT32 format before inserting it in the SD card module. The VCC pin of the SD card module was connected to the breadboard 5 V power side and the GND pin was connected to the ground. The MOSI, MISO, SCK and CS pins were connected to the hardware SPI pins of the arduino mega to achieve the best performance while the transfer of the data. The hardware SPI pins of the arduino mega are pin 51 for MOSI pin, pin 50 for MISO pin, pin 52 for SCK pin and pin 53 for CS pin [29]. Following the connection of the SD card module to the arduino, the programming of the SD card module was accomplished which is discussed in the next section.

#### **4.2.1.2 Programming of the SD card module**

For the programming of the SD card module, the arduino in-built libraries SD.h and SPI.h were used. The #include function was used to include these libraries in the program. After including the libraries, the CS pin was declared using an int function to pin 53 of the arduino. The in-built file function was used to call out a file. For the operation of the SD card module, six functions were

defined such as initializeSD, createFile, writeToFile, closeFile, openFile and readLine. The initializeSD function set CS pin as an output and used SD.begin function to enable the SD card module.

```
void initializeSD()
{
  Serial.println("Initializing SD card...");
  pinMode(CS_PIN, OUTPUT);

  if (SD.begin())
  {
    Serial.println("SD card is ready to use.");
  } else
  {
    Serial.println("SD card initialization failed");
    return;
  }
}
```

**Figure 42-** Defining initializeSD function

The createFile function creates or opens a file in the SD card using the given name.

```
int createFile(char filename[])
{
  file = SD.open(filename, FILE_WRITE);

  if (file)
  {
    Serial.println("File created successfully.");
    return 1;
  } else
  {
    Serial.println("Error while creating file.");
    return 0;
  }
}
```

**Figure 43-** Defining createFile function

The writeToFile function enters the given data in the file created in the SD card using in-built file.println function.

```
int writeToFile(char text[])
{
  if (file)
  {
    file.println(text);
    Serial.println("Writing to file: ");
    Serial.println(text);
    return 1;
  } else
  {
    Serial.println("Couldn't write to file");
    return 0;
  }
}
```

**Figure 44-**Defining writeToFile function

The closeFile function uses file.close function to close the opened file.

```
void closeFile()
{
  if (file)
  {
    file.close();
    Serial.println("File closed");
  }
}
```

**Figure 45-** Defining closeFile function

The openFile function opens the file with a given name using SD.open in-built function.

```

int openFile(char filename[])
{
  file = SD.open(filename);
  if (file)
  {
    Serial.println("File opened with success!");
    return 1;
  } else
  {
    Serial.println("Error opening file...");
    return 0;
  }
}

```

**Figure 46-** Defining openFile function

The readLine function reads a given line in a file using file.read function and returns it as a string.

```

String readLine()
{
  String received = "";
  char ch;
  while (file.available())
  {
    ch = file.read();
    if (ch == '\n')
    {
      return String(received);
    }
    else
    {
      received += ch;
    }
  }
  return "";
}

```

**Figure 47-** Defining readLine function

These functions were used throughout the code to enter data in the SD card file whenever a



tool was dispensed or returned.

#### 4.2.2.1 DS3231 RTC module

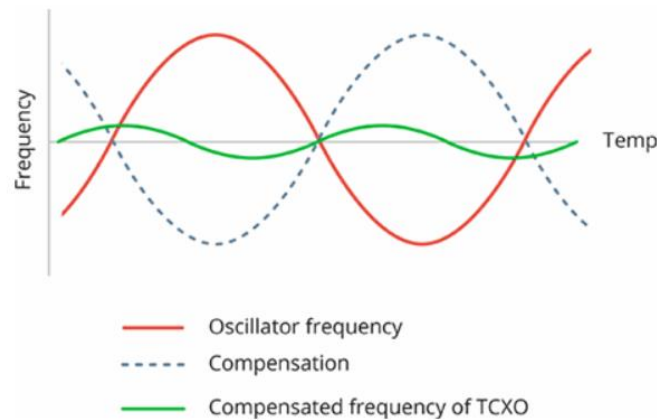
The DS3231 RTC module was used to keep track of date and time in the smart tool organizer. The date and time were stored whenever a tool request for dispensation or return was made to keep track of the tools. The time and date data was stored alongside the student ID number, the tool number and the status whether it is being dispensed or returned by the user.

The DS3231 RTC module used in this application includes a DS3231S RTC chip along with an AT24C32 EEPROM chip. The DS3231S chip keeps accurate timekeeping of years, months, days, hours, minutes and seconds. It also keeps tracks of leap years and can be used either in 12 hour or 24 hour format. The DS3231 chip includes INT/SQW pins which send square waves at 1, 4, 8 or 32 Hz and also provide interrupt signals. The DS3231 chip is also responsible for communication with the arduino using an I2C bus [31].



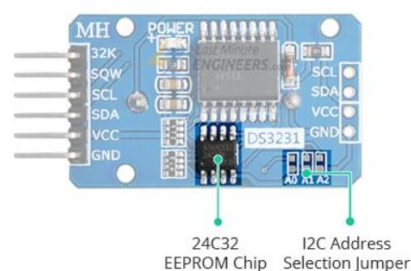
**Figure 48-** DS3231 RTC chip [31]

The DS3231 keeps track of time using the oscillation frequency and includes a 32 kHz TCXO (Temperature compensated crystal oscillator) which can operate at wide external temperature differentials[31]. To compensate for the frequency changes due to external temperature, the TCXO includes an internal temperature sensor and control logic which adjust the time by adding or removing seconds to keep an accurate track of time.



**Figure 49-** Graph of oscillator frequency’s relation with temperature [31]

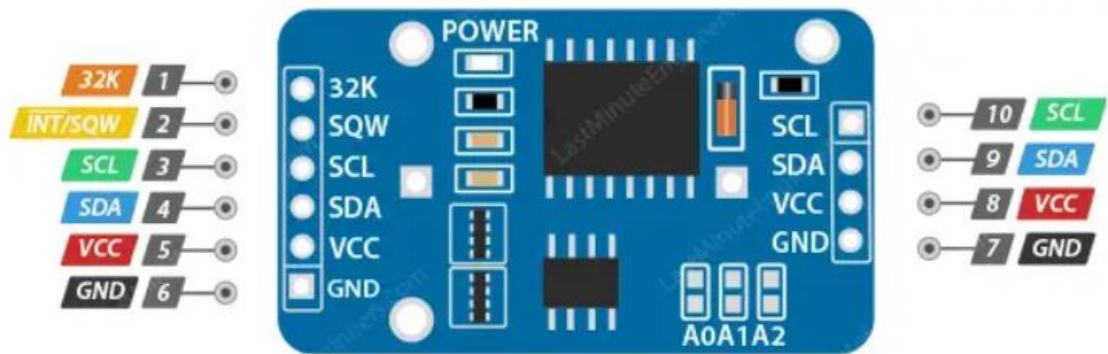
The DS3231 RTC module comes with a backup battery of 3V which is inserted on the back of the chip. The module uses this battery power when the power is cut off on the VCC pin of the module to keep an accurate timekeeping. The 32-byte AT24C32 EEPROM chip included on the DS3231 RTC module is used for data logging and storing[31]. It uses the same I2C bus to communicate with arduino as the DS3231S chip.



**Figure 50-** EEPROM chip and I2C address selection jumper [31]

The DS3231 RTC module has 6 pins such as 32K, INT/SQW, SCL, SDA, VCC and GND.

The 32K pin outputs a reference clock which is compensated for temperature and is accurate. The Int/SQW pin sends square waves at 1, 4, 8 or 32 Hz and also provides interrupt signals. For the I2C interface, the SCL pin is used as a serial clock pin and the SDA pin is used as a serial data pin. These pins are connected to the I2C pins on the arduino mega i.e. SCL pin is connected to pin 21 and SDA pin is connected to pin 20 of the arduino mega. The Vcc pin is used to supply 5V of power and the GND pin is connected to the ground.



**Figure 51-** RTC module pinouts [31]

#### **4.2.2.2 Programming of the DS3231 RTC module**

The DS3231 RTC module was programmed using the in-built libraries Wire.h, TimeLib.h and DS1307RTC.h. First these libraries were included using #include function. The in-built function tmElements\_t was used to declare a variable tm which would receive time and date from the RTC module. A function print2digits was used to print only 2 digits of each hour, minute and second. This was needed to ensure only two digits were stored excluding any extra zero before or after the time.

```

void print2digits(int number) {
    if (number >= 0 && number < 10) {
        Serial.write('0');
    }
    Serial.print(number);
}

```

**Figure 52-** Defining print2digits function

The time and date were read using the inbuilt functions of tm.Hour, tm.Minute, tm.Second, tm.Day, tm.Month and tm.Year. This data was displayed on the serial monitor using Serial.print command and was stored in various strings for storing it on the SD card. All the acquired data was combined in a string now and converted to character using a buffer and toCharArray function. This was necessary because to be able to store data on an SD card, it has to be in character format and not string format. This data was written to the file in the SD card. The following figure 53 displays how the time and date data was manipulated for each iteration and stored on the SD card.

```

if (RTC.read(tm)) {
    Serial.print("Ok, Time = ");
    print2digits(tm.Hour);
    String h = String (tm.Hour);
    Serial.write(':');
    print2digits(tm.Minute);
    String m = String (tm.Minute);
    Serial.write(':');
    print2digits(tm.Second);
    String s = String (tm.Second);
    Serial.print(", Date (D/M/Y) = ");
    Serial.print(tm.Day);
    String d = String (tm.Day);
    Serial.write('/');
    Serial.print(tm.Month);
    String mn = String (tm.Month);
    Serial.write('/');
    Serial.print(tmYearToCalendar(tm.Year));
    String y = String (tmYearToCalendar(tm.Year));
    Serial.println();
    now = ("Time = " + h + ":" + m + ":" + s + ",Date(D/M/Y)= " + d + "/" + mn + "/" + y );
    Serial.println("hey" + now);
    char buf[200];
    now.toCharArray(buf,now.length()+1);
    Serial.println(buf);
    writeFile(buf);
    closeFile();
}

```

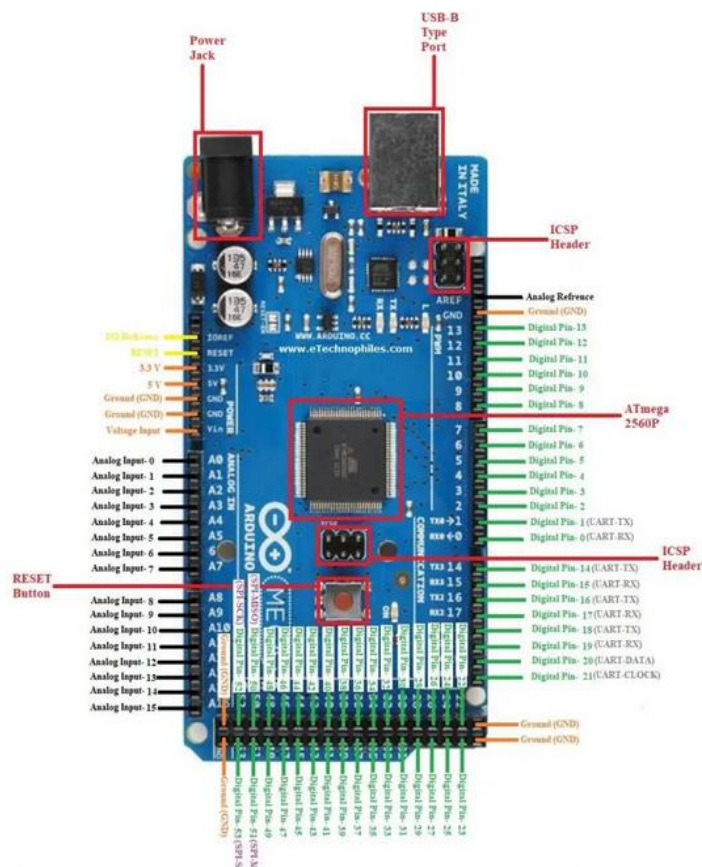
**Figure 53-** Code for storing time and date to SD card

### 4.2.3.1 Arduino Mega

The microcontroller arduino mega was used to control all the previously mentioned components making it the brain of the smart tool organizer. All components including stepper motor, TB6600 motor driver, servo, linear actuator, L298N module, limit switches, RTC module, SD card module and nextion display were programmed using the arduino mega and were connected to arduino in some way. Arduino mega synchronizes all the components and makes them work together to achieve the goal of the prototype. It facilitates the organization, tracking, dispensation, retrieval along with interfacing for the smart tool organizer.

The arduino mega uses an 8-bit microcontroller chip Atmega2560P which in a single clock cycle allows the arduino to execute instructions and provide excellent processing speed with low power consumption. The microcontroller includes 54 digital input/output digital pins which includes 14 PWM pins, 16 analog pins and 6 hardware serial ports. The arduino digital pins can be in either state 1 or 0 which is either high or low depending if there is voltage in those pins. All pins from 0 to 53 can be used as a digital pin. The arduino analog pins use ADC or analog to digital converter and they can be used either as an analog pin or a digital pin. The analog pins can return a value ranging from 0 to 1023. Since the arduino operates on 5 V, each value of the analog pin represents 4.88 mV of power meaning if the value is 4.88 mV, it will return 1 as analog value and increase to 1023 in the increments of 4.88 mV. The UART pins or hardware serial ports allow the arduino to communicate with other devices using the serial port. The UART TXD pins on arduino are pins 1, 18, 16 and 14 and UART RXD pins on arduino are pins 0, 19, 17 and 15. The SPI pins on arduino are 50, 51, 52 and 53 which allows the arduino to communicate with various peripheral devices. The three pins SCK, MISO and MOSI for peripheral devices were discussed earlier in the SD card module section which uses these pins to transfer data from arduino to the SD card. The SS or CS pin used for peripheral devices is used for master slave communication which is dependent on whether the CS pin is high or

low to allow communication between multiple devices. Arduino mega also includes an in-built LED which is connected to pin 13 and turns on or off depending on the state of that pin. The on-board 6 ICSP (In-Circuit Serial Programming) pins on the arduino mega can be used to program the arduino[33].



**Figure 54-**Arduino mega pinouts and components [32]

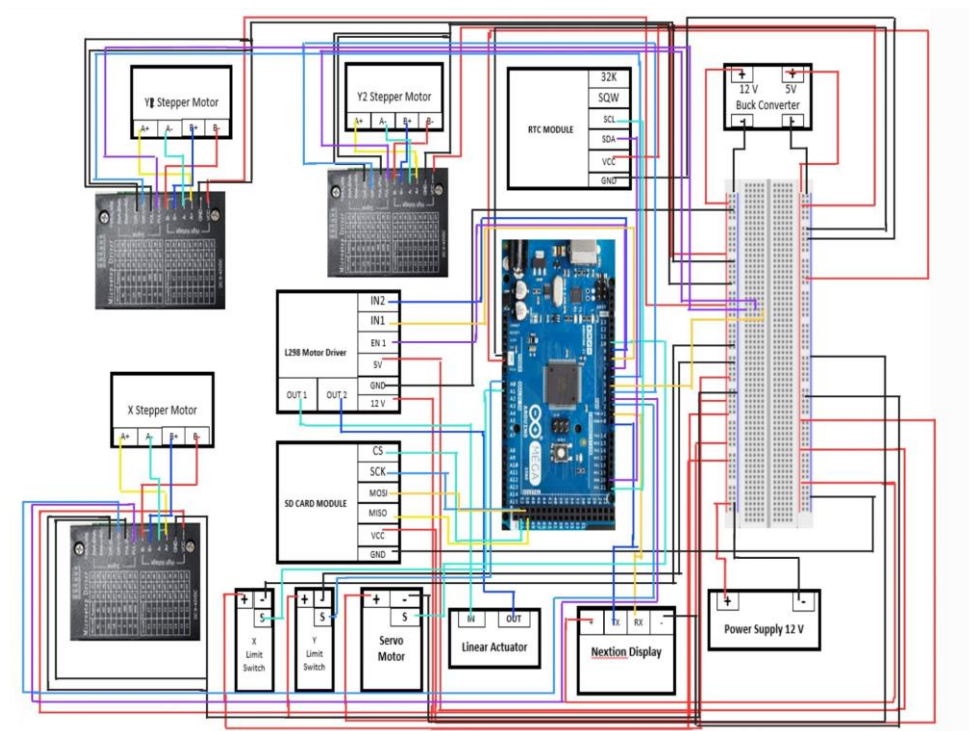
Arduino mega includes a crystal oscillator which provides basic timing and clock signal to the board. The reset button on arduino can be used to reset the board by pressing and holding the button till on-board power led flashes. The I/O reference voltage pin or IOREF pin is used as a reference at what voltage the microcontroller is currently operating and the AREF or analogue reference pin is used as a reference pin to compare the analog voltages[33]. For this application, the arduino mega was powered using the Vin pin which was connected to a 5 V power source and the ground pin was connected to the ground of the power source. The Following table includes the specifications of the arduino mega.

|                                |                |
|--------------------------------|----------------|
| Microcontroller                | ATmega2560     |
| Operating Voltage              | 5 V            |
| Power supply                   | 7 V – 12 V     |
| Current consumption            | 50 mA – 200 mA |
| Current consumption Deep Sleep | 500 $\mu$ A    |
| Digital I/O Pins               | 54             |
| Digital I/O Pins with PWM      | 15             |
| Analog Input Pins              | 16             |
| DC Current per I/O Pin         | 40 mA          |
| DC Current for 3.3V Pin        | 50 mA          |
| Flash Memory                   | 256 KB         |
| SRAM                           | 8 KB           |
| EEPROM                         | 4096 bytes     |
| Clock Speed                    | 16 MHz         |
| Length                         | 102 mm         |
| Width                          | 53 mm          |
| Power jack                     | yes            |
| USB connection                 | yes            |

**Table 5-** Arduino mega specifications [32]

### 4.3 Wiring and connections

The smart tool organizer uses 17 components and synchronizes these components to work together and achieve the common goal of organizing, dispensing, retrieving, tracking and keeping record of data. The following wiring diagram shows how all these components are connected and wired together.



**Figure 55-**Wiring diagram of the smart tool organizer's components

# CHAPTER 5

## Interfacing and Firmware

### 5.1 User-Interface of the Smart Tool Organizer

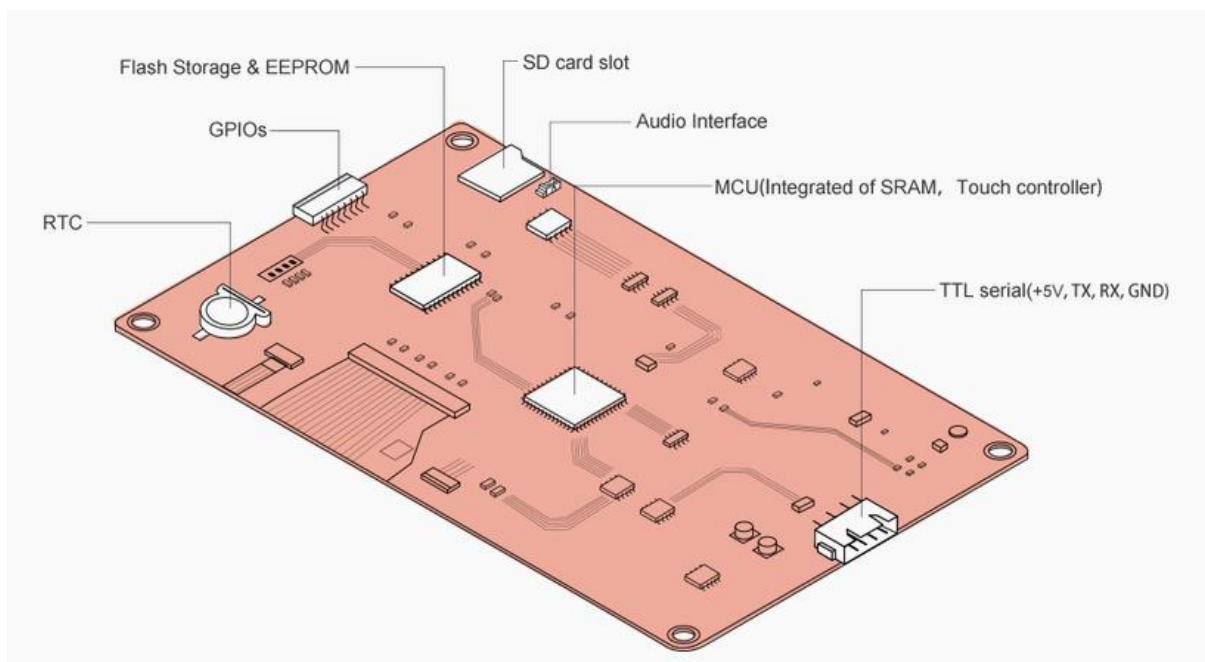
The Smart Tool Organizer is a compact device which allows the user to choose the desired tool and receive or return it using a dropbox. It also keeps track of user identity by requesting for student ID number and storing it alongside the tools number, tool status whether it was received or returned and accurate date and time. To operate the smart tool organizer, the user needs to interact with the machine and provide information. The interaction of the user with the smart tool organizer is designed to be on a 7" touch screen LED display which will take the user through the step of interaction to accomplish the goal of providing the user with requested tools or returning them. The interface designed for the end user is very easy to use, allowing anyone with the right credentials to operate the smart tool organizer. In this chapter, the interface design and user interaction with the smart tool organizer is discussed.

#### 5.1.1 Nextion LED display

The nextion display is an interactive user interface or HMI which can be connected to an arduino through the serial port and send and receive signal. The connection of arduino mega and the nextion display is established using a peripheral TTL serial connection. The nextion display comes with 4 connections: 5V power connection, a ground connection, TX connection and RX connection. The serial connection interacts with the TX and RX pins on the arduino and sends and receives signals upon touch event. The touch event can be set to when the display is touched or released. The nextion



display is used as a GUI (Graphical User Interface) and HMI (Human Machine Interface) for the interaction with the smart tool organizer. The nextion display operates on 5V and 430mA current. It typically uses 9600 bps baud rate through the serial port which can be changed to a minimum of 2400 bps and maximum of 921600 bps if required. The nextion display comes with an SD card slot which is used to upload files to the display which contains instructions about the GUI interface. The nextion display also includes flash storage to store data and files on-board, GPIOs, RTC to keep track of time, an audio interface, a touch controller, MCU and an SRAM [33].



**Figure 56-** Nextion display components [34]

The display sizes of the nextion can vary from 4.3” up to 10.1” and has multiple product families such as basic, enhanced and intelligent. For the application of the smart tool organizer a 7” intelligent series nextion display was selected. The touch screen is a 65K RGB resistive and capacitive display and the on-board MCU is 200MHz and contains flash memory of 128 MB. The SRAM is 512 KB and has 1024 Byte of EEPROM. The following table includes specifications of the nextion display[34].

### Specifications

|                              | Data                       | Description                                  |
|------------------------------|----------------------------|--|
| Color                        | 65K 65536 colors           | 16 bit 565, 5R-6G-5B                         |
| Layout size                  | 181mm(L)×108mm(W)×9.3mm(H) | NX8048P070-011R                              |
| Active Area (A.A.)           | 164.90mm(L)×100.00mm(W)    |  |
| Visual Area (V.A.)           | 154.08mm(L)×85.92mm(W)     |  |
| Resolution                   | 800×480 pixel              | Also can be set as 480×800                   |
| Touch type                   | Resistive                  |  |
| Touches                      | > 1 million                |  |
| Backlight                    | LED                        |  |
| Backlight lifetime (Average) | >30,000 Hours              |  |
| Brightness                   | 300nit                     | 0% to 100%, the interval of adjustment is 1% |
| Weight                       | 265g                       |  |

### Electronic Characteristics

|                   | Test Conditions             | Min  | Typical | Max | Unit |
|-------------------|-----------------------------|------|---------|-----|------|
| Operating Voltage |                             | 4.75 | 5       | 6.5 | V    |
| Operating Current | VCC=+5V, Brightness is 100% | -    | 430     | -   | mA   |
|                   | SLEEP Mode                  | -    | 170     | -   | mA   |

Power supply recommend : 5V, 1.0A, DC

### Working Environment & Reliability Parameter

|                     | Test Conditions  | Min | Typical | Max | Unit |
|---------------------|------------------|-----|---------|-----|------|
| Working Temperature | 5V, Humidity 60% | -20 | 25      | 70  | °C   |
| Storage Temperature |                  | -30 | 25      | 85  | °C   |
| Working Humidity    | 25°C             | 10% | 60%     | 90% | RH   |

### Interfaces Performance

|                           | Test Conditions | Min  | Typical | Max    | Unit |
|---------------------------|-----------------|------|---------|--------|------|
| Serial Port Baudrate      | Standard        | 2400 | 9600    | 921600 | bps  |
| Output High Voltage (TXD) | IOH=1mA         | 3.0  | 5.0     | Vin    | V    |
| Output Low Voltage (TXD)  | IOL=-1mA        |      | 0.1     | 0.2    | V    |

**Table 6-** Nextion display general and electronics specifications [33]

### Memory Features

| Memory Type        | Test Conditions        | Min | Typical | Max  | Unit |
|--------------------|------------------------|-----|---------|------|------|
| FLASH Memory       | Store fonts and images |     |         | 120  | MB   |
| User Storage       | EEPROM                 |     |         | 1024 | BYTE |
| RAM Memory         | Store variables        |     |         | 512  | KB   |
| Instruction Buffer | Instruction Buffer     |     |         | 4096 | BYTE |

### Audio Features

| Speaker | Parameter | Min | Typical | Max | Unit |
|---------|-----------|-----|---------|-----|------|
| Power   | -         | 0.5 | -       | 1.5 | W    |

Audio Connector Type: 1.25T-2-2A (1.25mm pitch 2-pin housing)

**\* NX8048P070-011R do not have speaker in the package.**

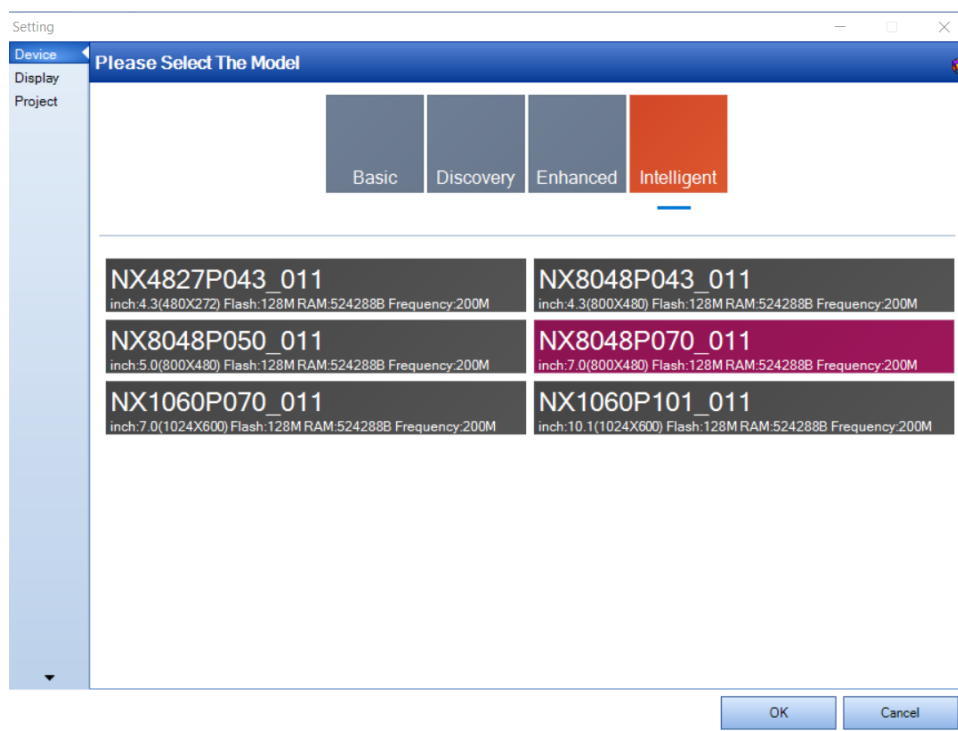
**Table 7-** Nextion display memory specifications [33]

## 5.1.2 Nextion Editor and Arduino programming

The nextion display uses nextion editor software to design the HMI software and GUI layout. It uses ASCII for programming the display functions such as touch event or touch release event and allows to assign attributes to value components at runtime. It also includes a built-in debugger which

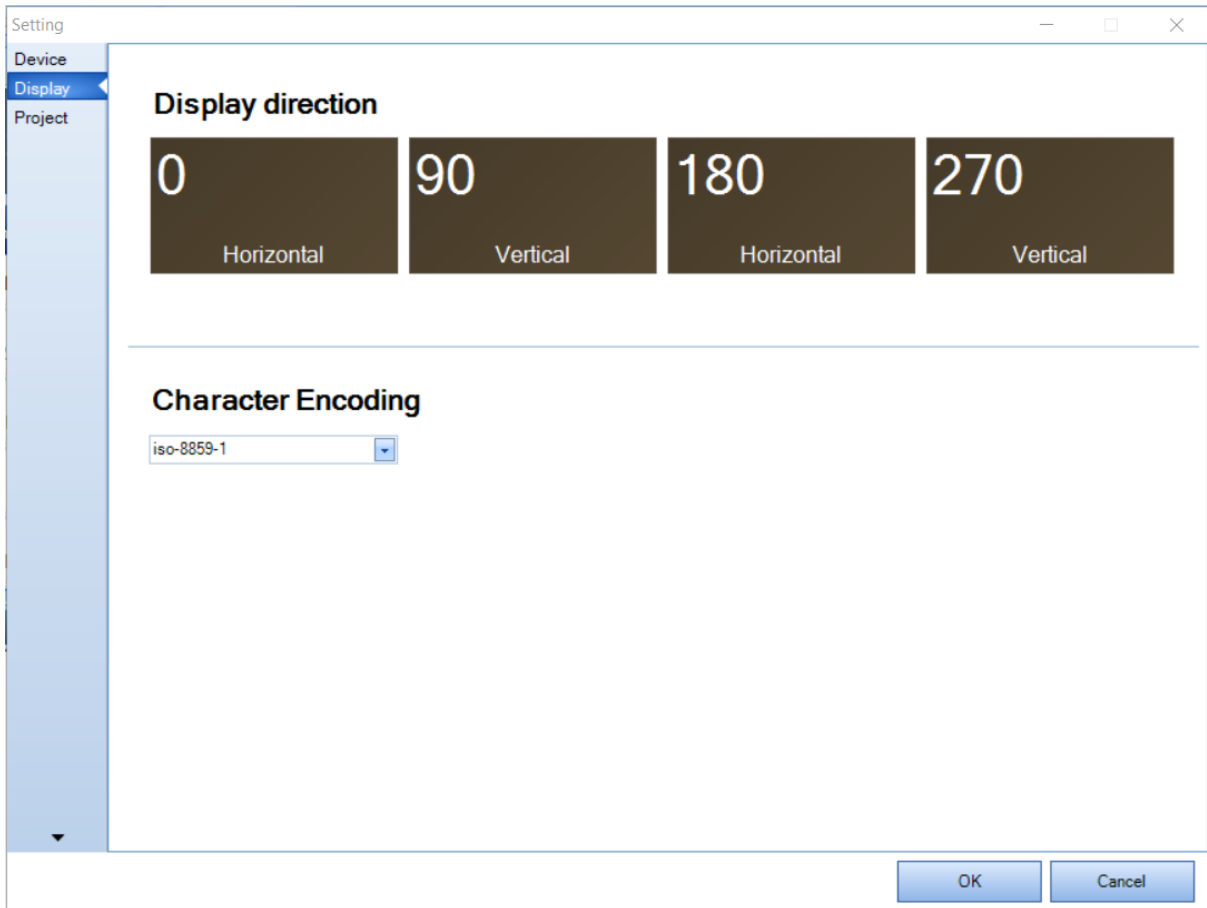
allows the user to simulate the GUI interface before uploading it to the nextion display.

During the process of making a new project, the device needs to be selected to configure the nextion editor according to the device. In the device selection, the series of devices and the model number is required. The device was selected by entering the series as intelligent and the model number NX8048P070\_11 to match the model being used in the smart tool organizer.



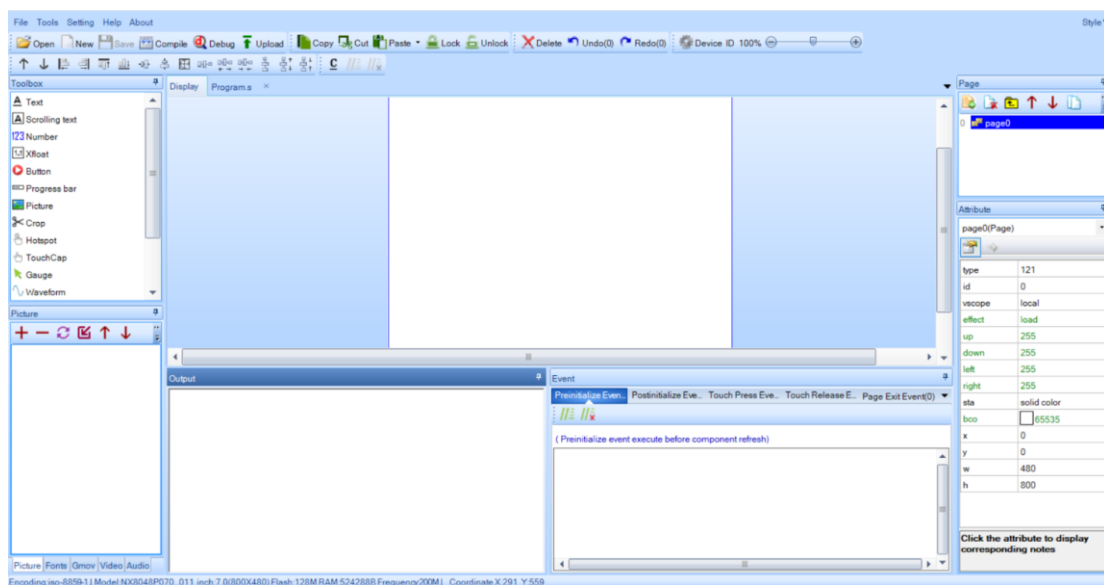
**Figure 57-** Selection of display type

In the next screen the viewing angle and the orientation of the screen was selected to be vertical at 270. This selection affects how the screen will be orientated. The vertical orientation was selected for designing the user interface of the smart tool organizer.



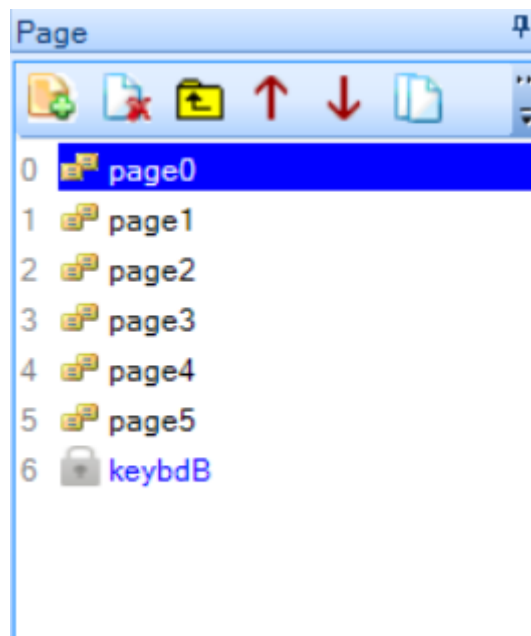
**Figure 58-** Display orientation selection

After selecting the orientation, the nextion editor displays the main user interface where the project will be edited and designed.



**Figure 59-** Main interface of the nextion editor

The page panel on the right side of the screen helps manage view delete and modify the pages in the editor. It includes functions such as add, delete, copy, insert, move up, move down, delete all, etc. The pages will be displayed one after the another on the nextion display as the user interacts with the smart tool organizer. The page can be named with up to 14 characters and the names are case sensitive. For smart tool organizer, six pages were used including a page for numeric keyboard for entering student ID. The six pages includes a home page, a tool status page which asks the user if it is a receive tool request or return tool request, two tools selection pages, a final status page and an error page.



**Figure 60-** Pages setup in nextion editor

On the home page, the user will be asked to enter their student id number and the numeric keyboard page will be activated when the user touches the number field. The student id will then be verified against the list of allowed student ids and the user will be taken to the tool selection status page if the student id is valid or shown the error page if the student id is not valid. This action will occur when the user touches and releases the next button on the home page. The next button is also

programmed to save the student id and send it to the arduino via serial port. The program for the next button is written as ASCII and it first converts the text field to a number field using the covx command and then prints it on the serial port a string value to send it to the arduino. The tsw command prevents users from accidentally touching the button twice by disabling the button for a second after it is pressed. The arduino will convert the student id number received in string format to char format and save it on the sd card.



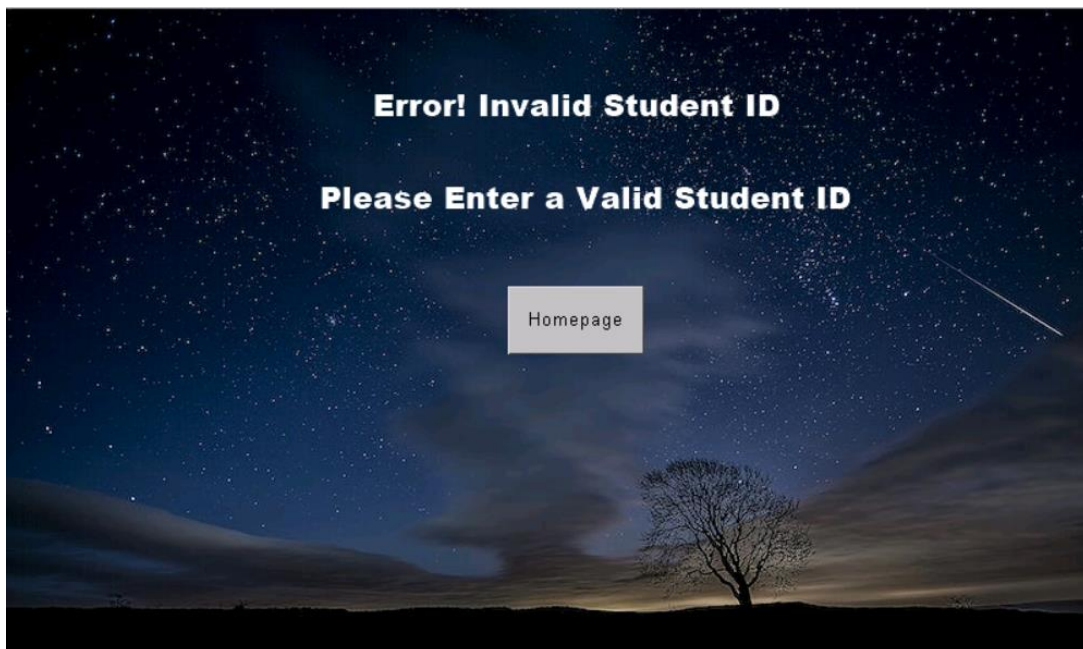
Figure 61- Home page of the nextion display



Figure 62- Numeric keyboard for entering student ID

```
Event
Touch Press Event(0) Touch Release Event(14)
Send Component ID
tsw b1,0
va0.txt="txt1"
covx n0.val,va1.txt,8,0
va0.txt+=va1.txt
prints va0.txt,0
delay=1000
tsw b1,1
if(va0.txt=="txt120411111"||va0.txt=="txt120411112"||va0.txt=="txt120411113"||va0.txt=="txt
{
page 1
}
else
{
page 5
}
```

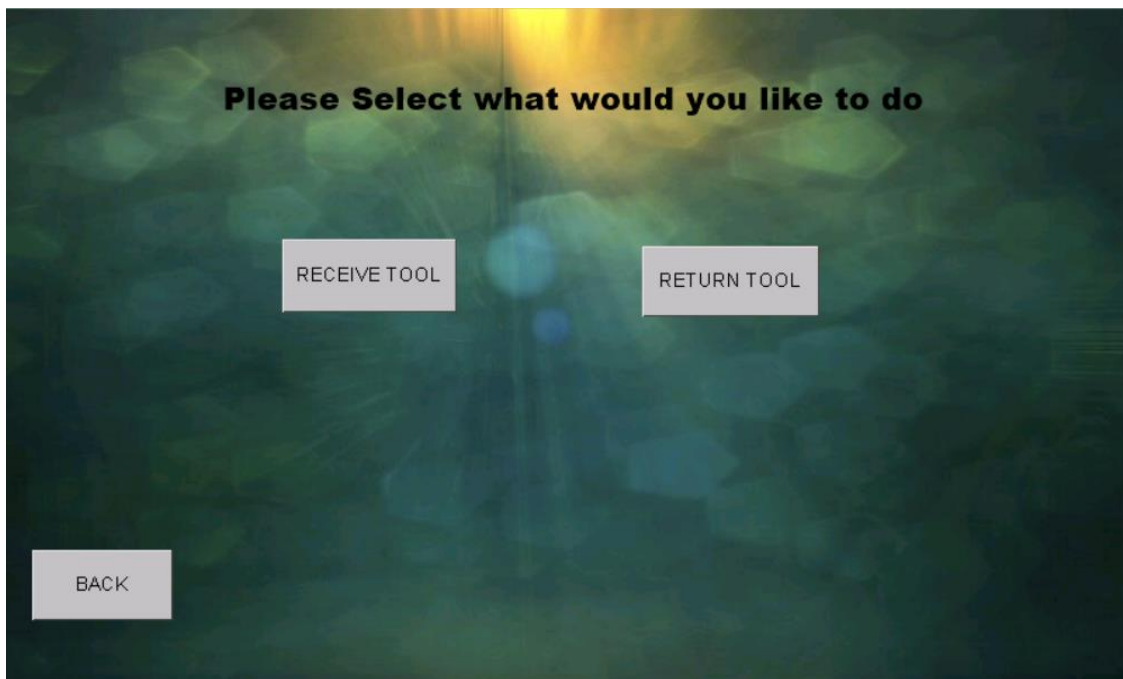
**Figure 63-** ASCII coding for next button in nextion editor



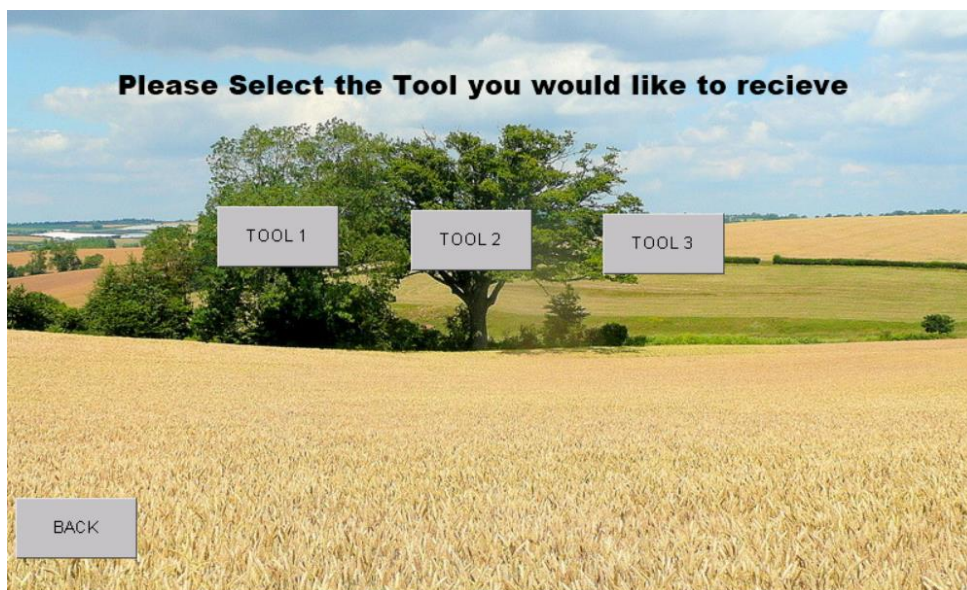
**Figure 64-** Error page for invalid id number

Upon entering a valid student id number the user is taken to the next page where the request

type will be prompted asking the user if the tool is to be received or returned. If the user selects the receive tool option, page 2 will be displayed and if the user selects the return tool option, page 3 will be displayed. These pages have the same buttons but the programming of the buttons on these pages are different.

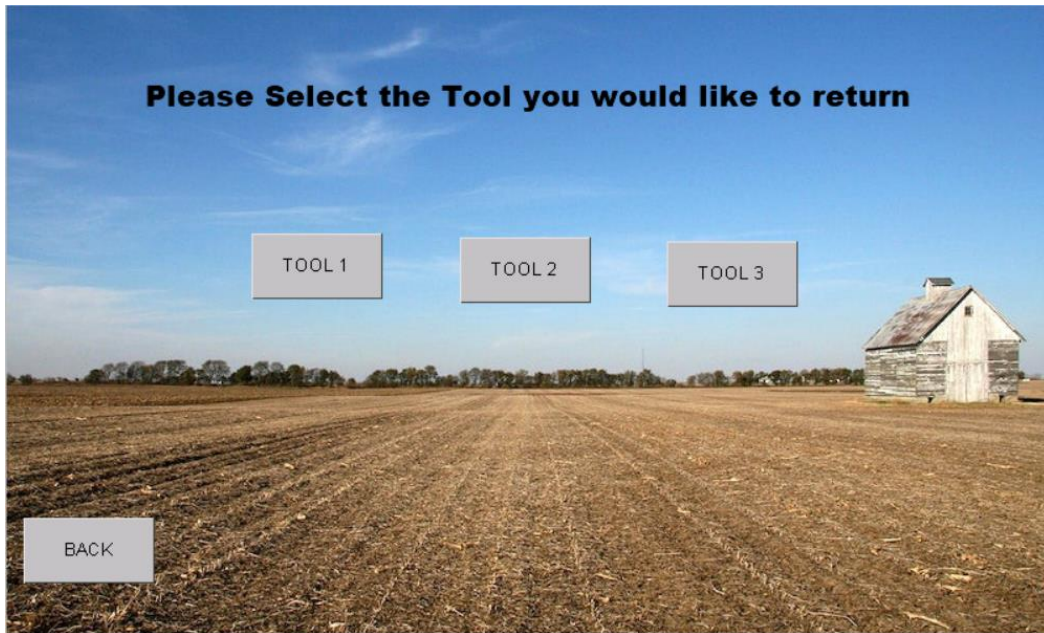


**Figure 65-** Tool selection status page



**Figure 66-** Tool receiving selection page





**Figure 67-** Tool returning selection page

The tool options will be displayed on these pages for receiving and returning. Each button is programmed to send a unique string to the arduino. When the button is touched and released arduino receives the unique string signal for the particular button. The arduino is programmed to save the date and time using the RTC module on the SD card along with the unique tool identity. When it receives that unique signal, the arduino synchronizes all the previously stated electronics components in chapter 4 such as CNC machine, linear actuator, the servo motor, TB6600 module and L298N module accordingly to either dispense or retrieve the tool selected by the user. The code arduino uses to achieve this goal is stated in appendix A. After the user received or returned the tool successfully, the final status page is displayed which thanks the user for using the smart tool organizer.



**Figure 68-** Final status page

# CHAPTER 6

## Results and Discussion

### 6.1 Performance and accuracy testing of the smart tool organizer

The smart tool organizer was tested for accuracy and performance and the test data was acquired. The data acquired for the performance and accuracy testing was analyzed and the optimum operating parameters were selected to achieve the best and most accurate performance of the smart tool organizer.

#### 6.1.1 Stepper motor performance and accuracy testing

The smart tool organizer is designed to carry various tools to different locations on the base to organize, dispense and retrieve tools as per the request of the user. To achieve this task, linear motion in x, y and z direction is necessary. This motion is achieved by using a CNC machine which operates on NEMA 17 stepper motors and a linear actuator. Three stepper motors are utilized to achieve linear motion in x and y direction such that one motor controls the motion in x direction and two motors controls the motion in y direction. Due to the rectangular frame of the CNC machine made of stainless-steel V-slots, the x axis is twice the length of the y axis and therefore, to achieve precise motion on two y axes, which are spread over a larger distance, two stepper motors are used. The stepper motors on y axes operate with synchronization of steps and pulses per second in opposite directions to achieve linear motion in one direction. This was accomplished by combining the pulse input for both motors enabling them to receive identical inputs for the number of pulses per second.

The stepper motors operate on two inputs i.e, pulse and direction. The direction input instructs the stepper motors to rotate either clockwise or counter-clockwise and the pulse input energizes the winding or the phases in the motor. This either magnetizes or demagnetizes the coils and enables the magnetic rotor to spin accordingly. The step size of the stepper motors controls the

angle of rotation of the rotor, which is 1.8 degrees for a full step. The speed of the stepper motor can be controlled by changing the steps sizes. In a full step mode, the motor takes 200 steps to complete one rotation.

For the accuracy testing of the stepper motors, the stepper motors were operated in both axes at different pulse/sec rates and data regarding the distance traveled was acquired. For each number of steps set, three trails were performed and mean of the three trails were calculated. The standard deviation of the three trails represents the accuracy of the stepper motors using a full step at the operating current of 0.5 A at 12 volts. The following table shows the data acquired for accuracy testing of the stepper motor used for linear motion in the x axis.

| <b>X FULL STEP</b> |                        |                      |               |                        |                      |               |                        |                      |
|--------------------|------------------------|----------------------|---------------|------------------------|----------------------|---------------|------------------------|----------------------|
| <b>Trails</b>      | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> | <b>Trails</b> | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> | <b>Trails</b> | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> |
| Trail 1            | 100                    | 1.42                 | Trail 1       | 200                    | 2.8                  | Trail 1       | 300                    | 4.18                 |
| Trail 2            | 100                    | 1.4                  | Trail 2       | 200                    | 2.78                 | Trail 2       | 300                    | 4.2                  |
| Trail 3            | 100                    | 1.41                 | Trail 3       | 200                    | 2.8                  | Trail 3       | 300                    | 4.19                 |
|                    | Mean                   | 1.41666667           |               | Mean                   | 2.79333333           |               | Mean                   | 4.19                 |
|                    | STD.                   | 0.008164965809       |               | STD.                   | 0.009428090416       |               | STD.                   | 0.008164965809       |
|                    |                        |                      |               |                        |                      |               |                        |                      |
| <b>Trails</b>      | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> | <b>Trails</b> | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> | <b>Trails</b> | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> |
| Trail 1            | 400                    | 5.6                  | Trail 1       | 600                    | 8.4                  | Trail 1       | 800                    | 11.19                |
| Trail 2            | 400                    | 5.59                 | Trail 2       | 600                    | 8.38                 | Trail 2       | 800                    | 11.18                |
| Trail 3            | 400                    | 5.58                 | Trail 3       | 600                    | 8.4                  | Trail 3       | 800                    | 11.2                 |
|                    | Mean                   | 5.58333333           |               | Mean                   | 8.39333333           |               | Mean                   | 11.19                |
|                    | STD.                   | 0.008164965809       |               | STD.                   | 0.009428090416       |               | STD.                   | 0.008164965809       |
|                    |                        |                      |               |                        |                      |               |                        |                      |
| <b>Trails</b>      | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> | <b>Trails</b> | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> | <b>Trails</b> | <b>Steps (pul/sec)</b> | <b>Distance (cm)</b> |
| Trail 1            | 1000                   | 13.89                | Trail 1       | 1200                   | 16.78                | Trail 1       | 1400                   | 19.58                |
| Trail 2            | 1000                   | 13.99                | Trail 2       | 1200                   | 16.79                | Trail 2       | 1400                   | 19.6                 |

|               |                            |                          |               |                            |                          |               |                            |                          |
|---------------|----------------------------|--------------------------|---------------|----------------------------|--------------------------|---------------|----------------------------|--------------------------|
| Trail 3       | 1000                       | 13.9                     | Trail 3       | 1200                       | 16.8                     | Trail 3       | 1400                       | 19.58                    |
|               | Mean                       | 13.92666<br>667          |               | Mean                       | 16.79                    |               | Mean                       | 19.58666<br>667          |
|               | STD.                       | 0.044969<br>12521        |               | STD.                       | 0.008164<br>965809       |               | STD.                       | 0.009428<br>090416       |
|               |                            |                          |               |                            |                          |               |                            |                          |
| <b>Trails</b> | <b>Steps<br/>(pul/sec)</b> | <b>Distance<br/>(cm)</b> | <b>Trails</b> | <b>Steps<br/>(pul/sec)</b> | <b>Distance<br/>(cm)</b> | <b>Trails</b> | <b>Steps<br/>(pul/sec)</b> | <b>Distance<br/>(cm)</b> |
| Trail 1       | 1600                       | 22.39                    | Trail 1       | 1800                       | 25.2                     | Trail 1       | 2000                       | 27.99                    |
| Trail 2       | 1600                       | 22.37                    | Trail 2       | 1800                       | 25.19                    | Trail 2       | 2000                       | 27.97                    |
| Trail 3       | 1600                       | 22.39                    | Trail 3       | 1800                       | 25.19                    | Trail 3       | 2000                       | 27.98                    |
|               | Mean                       | 22.38333<br>333          |               | Mean                       | 25.19333<br>333          |               | Mean                       | 27.98                    |
|               | STD.                       | 0.009428<br>090416       |               | STD.                       | 0.004714<br>045208       |               | STD.                       | 0.008164<br>965809       |

**Table 8-** Accuracy test data for x-axis stepper motor

The data shows the standard deviation of the Nema 17 stepper motor in x axis for different number of steps is normally less than a 100th of a centimeter with the exception of 1000 steps where the standard deviation is maximum at 0.044969 cm. For the purpose of the Smart Tool Organizer, the accuracy of the stepper motors at a full step mode is practical and viable for the performance of the CNC machine.

The following table shows the data acquired of the y axis which operates using two stepper motors at a distance of 61 cm. The mean values and the standard deviation of the three trials are calculated to measure the accuracy of the CNC machine in y direction.

| <b>Y FULL STEP</b> |                            |                          |               |                            |                          |               |                            |                          |
|--------------------|----------------------------|--------------------------|---------------|----------------------------|--------------------------|---------------|----------------------------|--------------------------|
| <b>Trails</b>      | <b>Steps<br/>(pul/sec)</b> | <b>Distance<br/>(cm)</b> | <b>Trails</b> | <b>Steps<br/>(pul/sec)</b> | <b>Distance<br/>(cm)</b> | <b>Trails</b> | <b>Steps<br/>(pul/sec)</b> | <b>Distance<br/>(cm)</b> |
| Trail 1            | 100                        | 1.38                     | Trail 1       | 200                        | 2.75                     | Trail 1       | 300                        | 4.1                      |
| Trail 2            | 100                        | 1.4                      | Trail 2       | 200                        | 2.78                     | Trail 2       | 300                        | 4.08                     |
| Trail 3            | 100                        | 1.4                      | Trail 3       | 200                        | 2.78                     | Trail 3       | 300                        | 4.09                     |
|                    | Mean                       | 1.393333<br>333          |               | Mean                       | 2.77                     |               | Mean                       | 4.09                     |
|                    | STD.                       | 0.009428<br>090416       |               | STD.                       | 0.014142<br>13562        |               | STD.                       | 0.008164<br>965809       |
|                    |                            |                          |               |                            |                          |               |                            |                          |

| Trails  | Steps (pul/sec) | Distance (cm)     | Trails  | Steps (pul/sec) | Distance (cm)      | Trails  | Steps (pul/sec) | Distance (cm)     |
|---------|-----------------|-------------------|---------|-----------------|--------------------|---------|-----------------|-------------------|
| Trail 1 | 400             | 5.47              | Trail 1 | 600             | 8.31               | Trail 1 | 800             | 11.11             |
| Trail 2 | 400             | 5.5               | Trail 2 | 600             | 8.29               | Trail 2 | 800             | 11.01             |
| Trail 3 | 400             | 5.49              | Trail 3 | 600             | 8.32               | Trail 3 | 800             | 11.02             |
|         | Mean            | 5.486666<br>667   |         | Mean            | 8.306666<br>667    |         | Mean            | 11.04666<br>667   |
|         | STD.            | 0.012472<br>19129 |         | STD.            | 0.012472<br>19129  |         | STD.            | 0.044969<br>12521 |
|         |                 |                   |         |                 |                    |         |                 |                   |
| Trails  | Steps (pul/sec) | Distance (cm)     | Trails  | Steps (pul/sec) | Distance (cm)      | Trails  | Steps (pul/sec) | Distance (cm)     |
| Trail 1 | 1000            | 13.78             | Trail 1 | 1200            | 16.75              | Trail 1 | 1400            | 19.3              |
| Trail 2 | 1000            | 13.85             | Trail 2 | 1200            | 16.73              | Trail 2 | 1400            | 19.5              |
| Trail 3 | 1000            | 13.87             | Trail 3 | 1200            | 16.75              | Trail 3 | 1400            | 19.49             |
|         | Mean            | 13.83333<br>333   |         | Mean            | 16.74333<br>333    |         | Mean            | 19.43             |
|         | STD.            | 0.038586<br>12301 |         | STD.            | 0.009428<br>090416 |         | STD.            | 0.092014<br>49161 |

**Table 9-** Accuracy test data for y-axes stepper motors

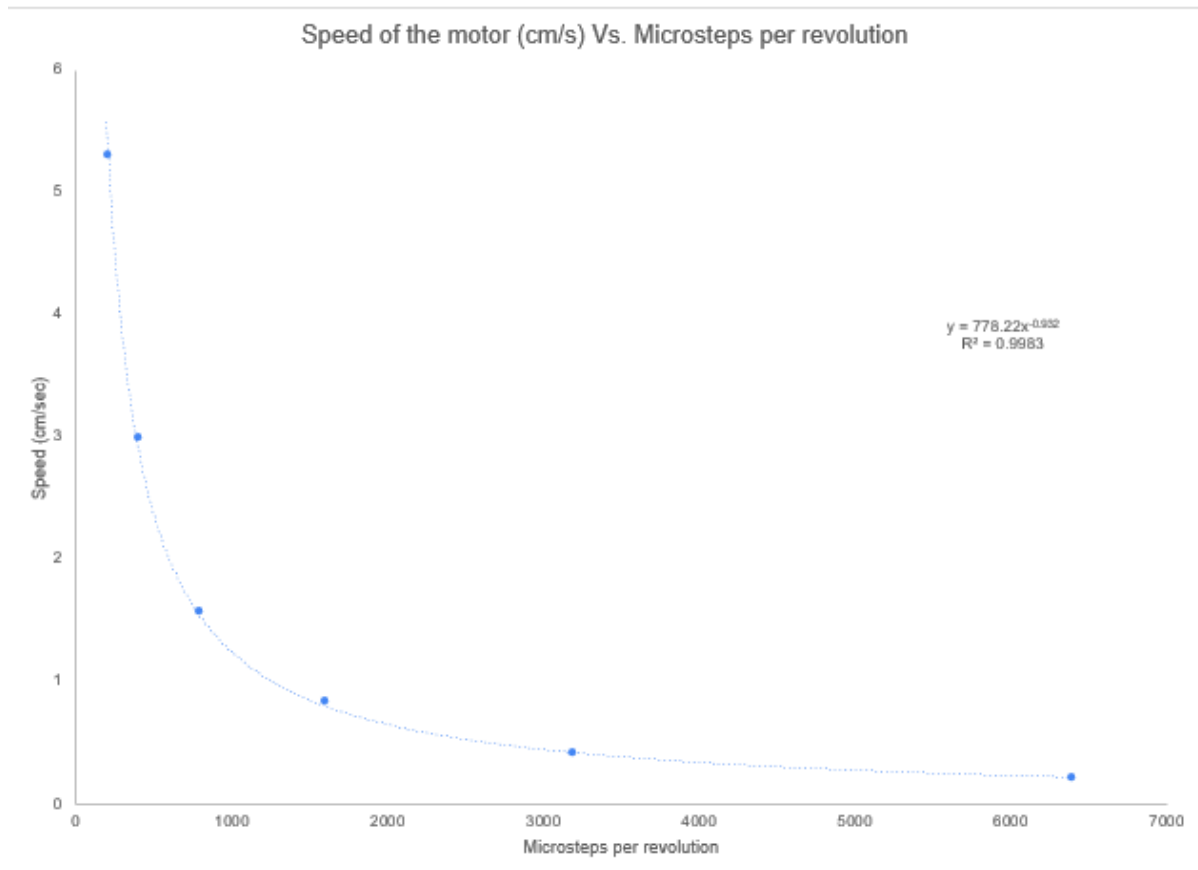
The above data shows the accuracy of two stepper motors operating together to achieve the linear motion in y direction. The maximum standard deviation from the data above is less than a mm at 0.920144 mm. For the practical use of the smart tool organizer this standard deviation is viable. The error in the accuracy of the stepper motor could result from friction between the wheels and the v-slot. This error is not compounded because the CNC machine goes back to the home position after every iteration.

Next, to maximize the performance and rapid positioning of the smart tool organizer, the linear speed of the stepper motors using a belt and pulley were measured at various pulse/sec, microseconds delays and current supplied to the stepper motors. The following table shows the changes in the speed of the stepper motors with changing number of steps.

| <b>Microsteps per revolution</b> | <b>Delay time between steps</b> | <b>distance (cm)</b> | <b>time (sec)</b> | <b>Speed of the motor (cm/s)</b> |
|----------------------------------|---------------------------------|----------------------|-------------------|----------------------------------|
| 200                              | 1000                            | 14                   | 2.64              | 5.303030303                      |
| 400                              | 1000                            | 14                   | 4.69              | 2.985074627                      |
| 800                              | 1000                            | 14                   | 8.85              | 1.581920904                      |
| 1600                             | 1000                            | 14                   | 16.91             | 0.8279124778                     |
| 3200                             | 1000                            | 14                   | 33.21             | 0.4215597712                     |
| 6400                             | 1000                            | 14                   | 66                | 0.2121212121                     |

**Table 10-** Test data for linear speed of the stepper motors with varying steps

The data above shows a trend of linear speed of the stepper motors decreasing by increasing the microsteps per revolution. The accuracy and the precision increases with increasing the microsteps per revolution but since the full steps accuracy data in Table 10 proved the accuracy of the stepper motor to be viable and practical in the operation of the smart tool organizer, maximizing the speed at full step is opted for to increase the performance of the smart tool organizer. The following graph shows the trend of the linear speed of the stepper motors vs. microsteps per revolution.



**Graph 1-** Linear speed of the stepper motors with varying steps

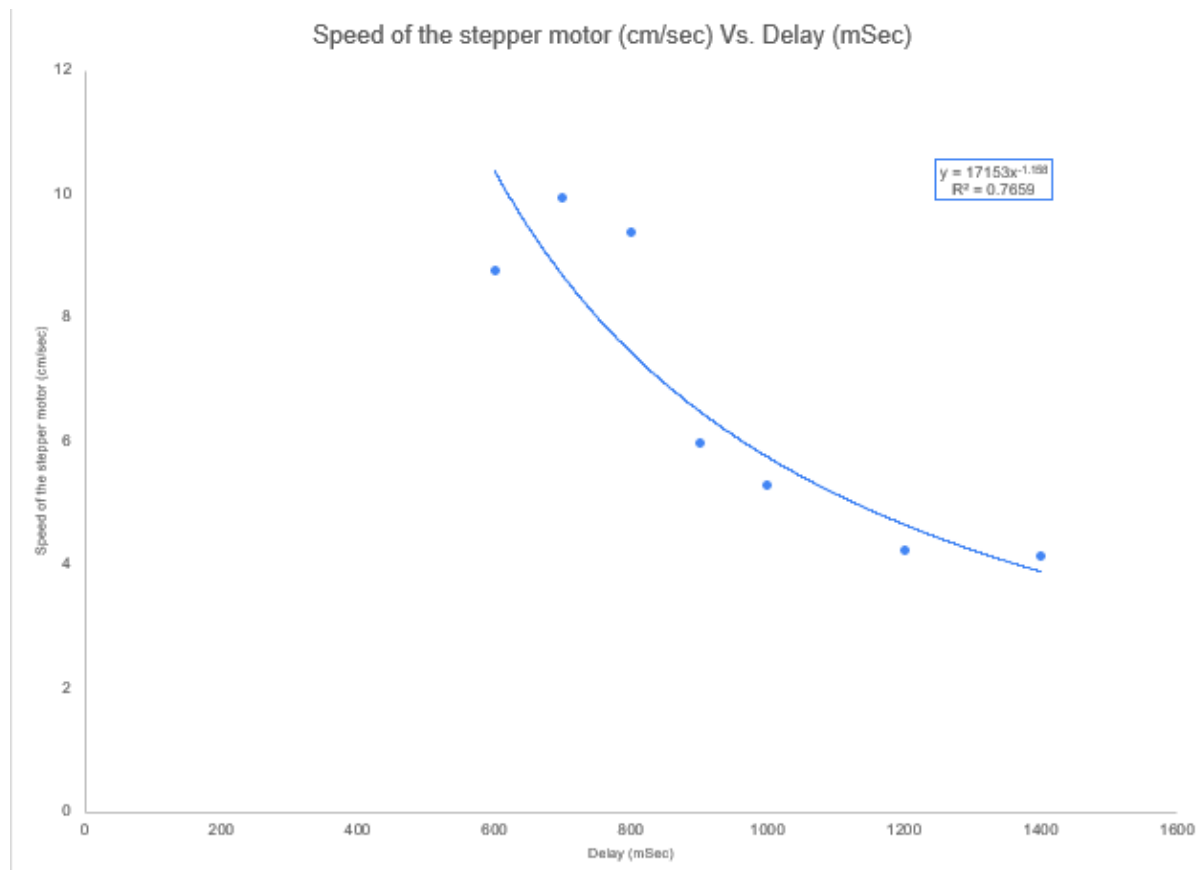
The microsecond delay between the steps was next compared to the linear speed of the stepper motor and the trend was plotted. The following table represents the change in the speed of the stepper motor with change in the microsecond delay in the code.

| <b>Microsteps per revolution</b> | <b>Delay time between steps</b> | <b>distance (cm)</b> | <b>time (sec)</b> | <b>Speed of the motor (cm/s)</b> |
|----------------------------------|---------------------------------|----------------------|-------------------|----------------------------------|
| 200                              | 600                             | 14                   | 1.6               | 8.75                             |
| 200                              | 700                             | 14                   | 1.41              | 9.929078014                      |
| 200                              | 800                             | 14                   | 1.49              | 9.395973154                      |
| 200                              | 900                             | 14                   | 2.34              | 5.982905983                      |
| 200                              | 1000                            | 14                   | 2.64              | 5.303030303                      |
| 200                              | 1200                            | 14                   | 3.31              | 4.229607251                      |
| 200                              | 1400                            | 14                   | 3.38              | 4.142011834                      |

**Table 11-** Test data for linear speed of the stepper motors with varying delays



Based on the data acquired for microseconds delay, the fastest linear speed achievable by stepper motors is approximately 9.93 cm/s at a delay of 700 microseconds. Further attempts to reduce the microsecond delay below 600 resulted in failure for operation of the stepper motors. This sets the limit of minimum delay between each step to be 600 microseconds. The following graph shows the trend of linear speed of the stepper motor with change in delay.



**Graph 2-** Linear speed of the stepper motors with varying delay

The Graph 2 shows that the linear speed of the stepper motor peaks at 700 microseconds of delay. The delay was thus chosen to be 700 microseconds for the optimal performance of the stepper motors.

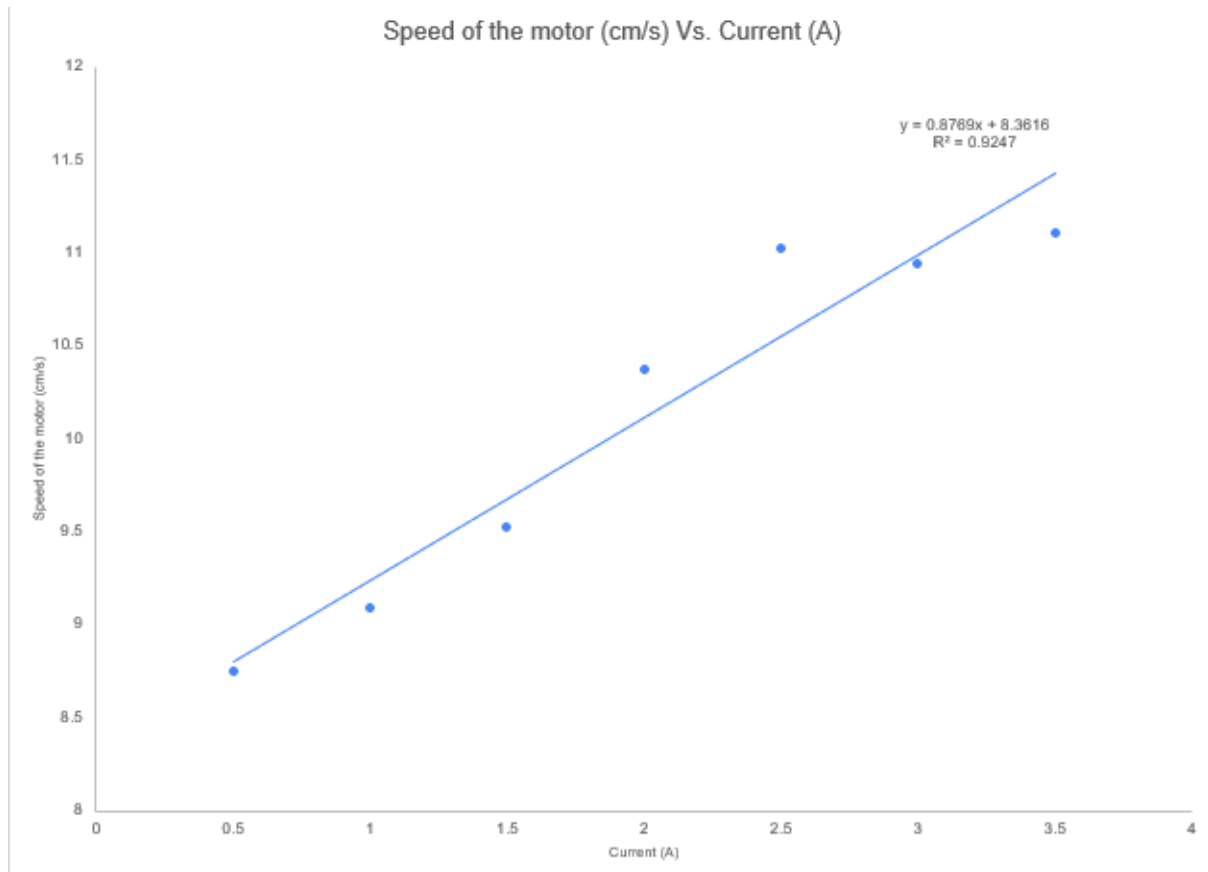
Next, the current supplied to the stepper motor was optimized for maximum performance of the smart tool organizer. The current was changed for each data point and the speed of the stepper

motors were recorded. The current ranges from 0.5A to a maximum of 3.5 A. The following table shows the effects of changing the current supplied to the stepper motors on the linear speed of the stepper motors on V-slots.

| <b>Current</b> | <b>Microsteps per revolution</b> | <b>Delay time between steps</b> | <b>distance (cm)</b> | <b>time (sec)</b> | <b>Speed of the motor (cm/s)</b> |
|----------------|----------------------------------|---------------------------------|----------------------|-------------------|----------------------------------|
| 0.5            | 200                              | 700                             | 14                   | 1.6               | 8.75                             |
| 1              | 200                              | 700                             | 14                   | 1.54              | 9.090909091                      |
| 1.5            | 200                              | 700                             | 14                   | 1.47              | 9.523809524                      |
| 2              | 200                              | 700                             | 14                   | 1.35              | 10.37037037                      |
| 2.5            | 200                              | 700                             | 14                   | 1.27              | 11.02362205                      |
| 3              | 200                              | 700                             | 14                   | 1.28              | 10.9375                          |
| 3.5            | 200                              | 700                             | 14                   | 1.26              | 11.11111111                      |

**Table 12-** Test data for linear speed of the stepper motors with varying current

According to the data gathered by changing the input current to the stepper motor, a trend of increasing linear speed of the stepper motor is observed when the current supplied is increased. The maximum speed of the stepper motor is observed at an input current of 3.5A increasing the speed of the motor to approximately 11.11 cm/s. With increasing current, a higher tendency of the stepper motors to heat was observed and the noise produced by the motors was increased. The NEMA 17 stepper motors used in the CNC machine are rated at 0.45 A and to achieve the repeatability, consistency and prevent overheating of the stepper motors used in the CNC machine the current of 0.5A was selected to be the optimum current. The following graph shows the trend of changing current Vs. the speed of the stepper motor.



**Graph 3-** Linear speed of the stepper motors with varying current

A linear trend is observed in the graph above representing the linear speed of the stepper motor in cm/s with change in current with the increments of half an Amp.

Using the data collected for the linear speed of the stepper motor, the parameters for the optimum performance of the stepper motors were determined. The number of steps per revolution, supplied current and the microseconds delay between each step affects the speed and performance of the stepper motor. Based on the data provided in tables above and the trends observed from the graphs above, 200 steps/revolution, 0.5A input current and 700 microseconds of delay between each step was selected to be the most optimum conditions to achieve highest performance of the smart tool organizer.

### **6.1.2 Servo motor performance and accuracy testing**

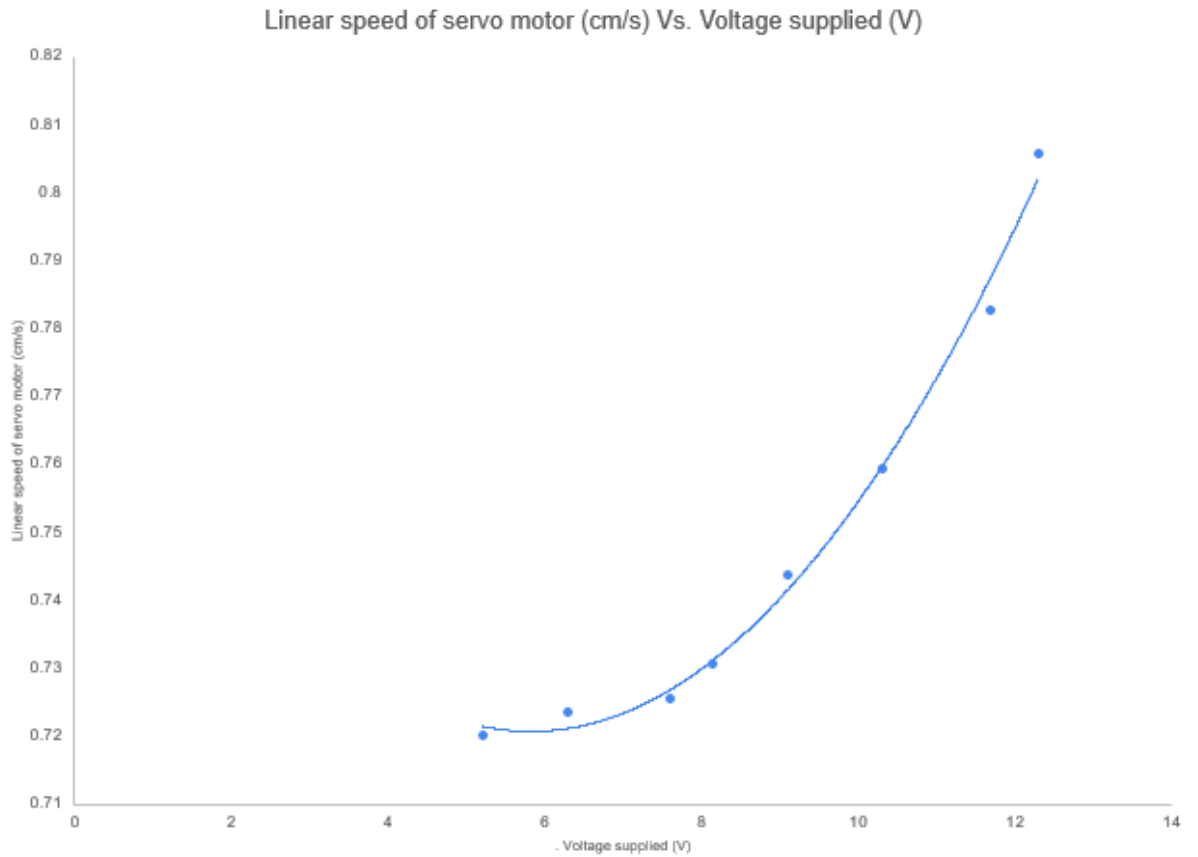
The servo motor is used to operate the mechanical 3D printed gripper. The servo motor shaft

is connected to a geared pinion which translates the rotational motion of the servo motor to linear motion of the geared rack. Two racks are connected tangent to the pinion and parallel to each other. The rotation of the pinion connected to the shaft causes these racks to move linearly in opposite directions. The racks are connected to the gripper arm on the bottom of the base which causes this gripper arm to come closer closing the gripper and move away opening the gripper. The voltage supplied to the servo motor was compared to the speed of the servo motor. The linear distance traveled by the gripper arm was measured along with the time it took to cover the distance to calculate the speed of the servo motor. The MG996R servo motor can rotate 0-180° and it requires 160° to close the gripper arms. The following table shows the data acquired for linear speed and voltage of the servo motor.

| <b>MG996R Servo</b> |                      |                   |                     |
|---------------------|----------------------|-------------------|---------------------|
| <b>Voltage</b>      | <b>distance (mm)</b> | <b>time (sec)</b> | <b>speed (cm/s)</b> |
| 5.2                 | 13.4                 | 1.861             | 0.7200429876        |
| 6.3                 | 13.4                 | 1.852             | 0.7235421166        |
| 7.6                 | 13.4                 | 1.847             | 0.7255008121        |
| 8.15                | 13.4                 | 1.834             | 0.7306434024        |
| 9.1                 | 13.4                 | 1.802             | 0.743618202         |
| 10.3                | 13.4                 | 1.765             | 0.7592067989        |
| 11.7                | 13.4                 | 1.712             | 0.7827102804        |
| 12.3                | 13.4                 | 1.663             | 0.8057726999        |

**Table 13-** Test data for linear speed of the servo motors with varying voltage

From the data acquired for the servo motor, a trend of increasing velocity with increase in voltage is observed. The voltage supplied to the servo motor was incremented by approximately 1 V using a potentiometer and the time for the linear geared rack to move a constant distance was measured. Using this data, the speed of the gripper arms was calculated. The trend observed from the data in the table was plotted and is shown below.



**Graph 4-** Linear speed of the servo motors with varying voltage

## 6.2 Criteria for tool selection for the smart tool organizer

The gripper arm is in contact with the tool when the smart tool organizer is relocating the tool, either from the original position to the dropbox or from the dropbox to its original position. The tool will be held by two gripper arms from both sides exerting normal force onto the tools. The coefficient of friction between the gripper arm connected to the servo motor using a gear and rack system and the surface of the tool in contact with the gripper arm can be calculated using the normal force exerted by the gripper arms on the tool and the weight of the tool. The normal force applied to the surface of the tool stays constant and the weight of the tool will change depending on the tool. This changes the friction coefficient for different tools. Since the linear actuator can lift the tool up

to 4.5 lbs, the limiting factor for the weight of the tool is the friction coefficient between the tool and the gripper arms. To determine if a tool is compatible and operable in the smart tool organizer, the friction coefficient of the tool, the surface area in contact with the gripper arms and the width of the tool are used. For a tool to be operable in the smart tool organizer, all these requirements must be met. The normal force acting on a tool by the gripper arm was measured using a weighing scale between the arms of the gripper. The average of 15 trials is used as the normal force for further calculations. The following table shows the data acquired for the normal force with a rotation angle of 160°.

| <b>Trail</b>       | <b>Normal Force (N)</b> |
|--------------------|-------------------------|
| 1                  | 2.062338495             |
| 2                  | 2.14373369              |
| 3                  | 1.88679946              |
| 4                  | 2.14569502              |
| 5                  | 2.01428591              |
| 6                  | 1.934852045             |
| 7                  | 1.685763135             |
| 8                  | 1.72400907              |
| 9                  | 1.697531115             |
| 10                 | 1.748525695             |
| 11                 | 1.92602606              |
| 12                 | 1.770100325             |
| 13                 | 1.68282114              |
| 14                 | 1.711260425             |
| 15                 | 1.991730615             |
| Mean               | 1.87503148              |
| Standard deviation | 0.1691972967            |

**Table 14-** Test data for normal force applied to the tool by the gripper arms

The friction coefficient for various tools were calculated using the normal force and the weight of the tool. The friction coefficient calculated is used to determine if the tool can be operated

in the smart tool organizer. The limiting factor for a tool to be used in the smart tool organizer includes weight, width and surface area. A variety of tools with different shapes, sizes and forms were tested with the smart tool organizer to establish a tool selection criterion. The following table shows the test results of the smart tool organizer operating with various tools.

| Tool                          | Normal force (N) | Weight(N)  | Friction Coefficient | Surface area (mm) | width(m m) | Operable | Limitation           |
|-------------------------------|------------------|------------|----------------------|-------------------|------------|----------|----------------------|
| Screwdriver 1                 | 1.87503148       | 0.10591182 | 17.7037037           | 5.24              | 10         | yes      | None                 |
| Screwdriver 2                 | 1.87503148       | 0.11179581 | 16.77192982          | 5.24              | 10         | yes      | None                 |
| Screwdriver 3                 | 1.87503148       | 0.12258312 | 15.296               | 5.24              | 10         | yes      | None                 |
| Wrench 1                      | 1.87503148       | 0.94536106 | 1.98340249           | 6.2               | 12.4       | no       | Friction Coefficient |
| Wrench 2                      | 1.87503148       | 0.50210048 | 3.734375             | 6.2               | 10.7       | yes      | None                 |
| Allen key 1                   | 1.87503148       | 0.22849494 | 8.206008584          | 7.6               | 5.7        | no       | Min Width            |
| Allen key 2                   | 1.87503148       | 0.13337044 | 14.05882353          | 5.2               | 4.8        | no       | Min Width            |
| Metallic screwdriver 1        | 1.87503148       | 0.29812216 | 6.289473684          | 3.72              | 7.1        | yes      | None                 |
| Metallic screw driver 2       | 1.87503148       | 0.09022118 | 20.7826087           | 2.25              | 4.3        | yes      | None                 |
| handle of rubber screw driver | 1.87503148       | 0.20397832 | 9.192307692          | 5.92              | 11.3       | yes      | None                 |
| Ball driver                   | 1.87503148       | 0.23830159 | 7.868312757          | 10.3              | 21.2       | no       | Max Width            |

**Table 15-** Test data for application of various tools

The data acquired through testing the smart tool organizer helps establish the criteria for tools that can be operated in the smart tool organizer. The criteria include minimum and maximum width of the tool, the weight of the tool and the friction coefficient of the tool. The first three screwdrivers with friction coefficient of 17.7, 16.8 and 15.2 and with a width of 10 mm were operable by the smart tool organizer. The wrench 1 was not operable because the friction coefficient is 1.98. Therefore, the criteria for the friction coefficient of the tool is set to a minimum of 2.5 with a factor of safety of 1.26. The allen keys were not operable because the minimum width parameter was less than the space between the gripper arm when it is in fully closed position. The minimum space

between the gripper arms was measured and the minimum width of the tool requirement was set to 6.5 mm with a factor of safety of 1.14. The limiting factor for maximum tool width is set by the distance between the lowest point of the gripper arms and the position of the tool on the base of the smart tool organizer. The space between the arms of the gripper is 28.4 mm and the space between the lowest point of the gripper arms and the position of the tool on the base of the smart tool organizer is 20.3 mm. Therefore, the maximum height of the tool requirement is set to 18 mm and the maximum width of the tool requirement is set to 25 mm. These requirements are set with a factor of safety of 1.128 and 1.136 respectively. The following table shows the minimum and maximum requirements for the tool used in the smart tool organizer.

| <b>Limiting factors</b>                        | <b>Minimum</b> | <b>Maximum</b> |
|--|----------------|----------------|
| <b>Width of the tool (mm)</b>                  | 6.5            | 25             |
| <b>Height of the tool (mm)</b>                 | None           | 18             |
| <b>Length of the tool (mm)</b>                 | None           | 157.6          |
| <b>Friction of Coefficient</b>                 | 2.5            | None           |
| <b>Weight of the tool (g)</b>                  | None           | 76.48          |
| <b>Surface are in contact (mm<sup>2</sup>)</b> | 3.5            | None           |

**Table 16-** Requirements for the selection of tools

The current design of the smart tool organizer requires the tool to meet all the requirements mentioned in the table above. All the requirements include a factor of safety and can be used for selection of tools intended to be used in the smart tool organizer. For future work, the design of the gripper can be modified to extend the criteria for the tools and include more tools.



# CHAPTER 7

## Conclusion and future works

The goal of the prototype of the smart tool organizer is to have a compact electromechanical smart tool organizer that is capable of storing, tracking personal use and availability of machine shop tools in the college of Mathematics and Science at the University of Central Oklahoma. The following are the deliverable and the design requirements for the design of the smart tool organizer.

Design requirements:

- 1) The device must include a mechanism for the automatic retrieval of operator requested tools
- 2) The device must request an operator ID for access and bookkeeping (including day, time)
- 3) The device must include a mechanism for appropriate tool return
- 4) The device must be compact and operator friendly

The prototype of the smart tool organizer met all the requirements mentioned above using electronics, mechanics, controls, firmware and interfacing.

The machine shop in the UCO engineering department has many unorganized tools and as a result, often the tools are misplaced and missing. The smart tool organizer is a prototype designed to tackle this problem by pristinely organizing, storing, tracking, dispensing and retrieving the tools to and from the authorized user. The smart tool organizer is a compact design which operates on a 12 V power source which can be plugged into a 120 V 60 Hz wall outlet. The smart tool organizer has a 7” LED touch screen display for user interface which is used to interact with the prototype. The smart tool organizer has an automatic tool dispensation and retrieval mechanism which keeps track of the users, tools, date and time for bookkeeping.

The design of the smart tool organizer includes a microcontroller, electro-mechanical clamping device, a CNC machine, an interactive digital interface and a data-base for collecting & tracking the tools. The CNC machine uses a belt and pinion system to convert the rotation motion of the stepper motors into linear motion and operates on a skeleton made of 5 stainless steel v-slots. The acrylic gantry plates are used to assemble the stepper motors, belt and pinion system, wheels and V-slots together. The CNC machine only operates in the x and y axis, so to obtain the motion in the z axis a linear actuator was incorporated which gave the robotic unit three degrees of freedom. The mechanical clamp operating using a servo motor was used as a gripper to grab and drop the tools as required. Limit switches were used to safely operate the robotic unit avoiding any damage that could occur from collisions of gantry plates and x and y axis. Motor drivers such as TB6600 and L298N were used to operate the stepper motors and linear actuator. The DS3231 RTC module was used to keep track of time and date and an SD card module was used to store the data. The user interface display was used to collect information from the user such as student ids and tool numbers and stored accordingly in the sd card. The smart tool organizer uses ten electronics components and various mechanical components to accomplish the goal of pristinely organizing, storing, tracking, dispensing and retrieving the tools.

During the process of designing the smart tool organizer, many mechanical, electronics and controls problem were encountered such as not being able to synchronize the components to work together, problematic communication between the nextion display and arduino, inoperable components, problems with the dropbox mechanism, lack of friction between the rails and the wheels of the gantry plates, various mechanical constraints, etc. These problems were resolved using ingenious ideas and troubleshooting techniques.

Future work on the prototype of smart tool organizer can include face recognition of the user

to identify the user without the need for student id number, tool recognition for automatic dispensation and retrieval of tools, expanding the capabilities and tools selection to include more bigger and heavier tools in the smart tool organizer by increasing the torque of the gripper, uploading the acquired data on a cloud for remote data access and the use of lead screw mechanism to convert the rotational motion of the stepper motors into linear motion to operate the smart tool organizer in vertical orientation such as mounting on a wall.

## References

- [1] Janssen, Christian P., et al. "History and Future of Human-Automation Interaction." *International Journal of Human-Computer Studies*, vol. 131, 2019, pp. 99–107, <https://doi.org/10.1016/j.ijhcs.2019.05.006>.
- [2] Singh, Indramani L., et al. "Automation-Induced Monitoring Inefficiency: Role of Display Location." *International Journal of Human-Computer Studies*, vol. 46, no. 1, 1997, pp. 17–30, <https://doi.org/10.1006/ijhc.1996.0081>.
- [3] Lee, John D., and Neville Moray. "Trust, Self-Confidence, and Operators' Adaptation to Automation." *International Journal of Human-Computer Studies*, vol. 40, no. 1, 1994, pp. 153–84, <https://doi.org/10.1006/ijhc.1994.1007>.
- [4] Pfeifer, Rolf, and Christian Scheier. *Understanding intelligence*. MIT press, 2001.  
[Understanding Intelligence - Rolf Pfeifer, Christian Scheier - Google Books](#)
- [5] Leite, Iolanda, et al. "The Influence of Empathy in Human–robot Relations." *International Journal of Human-Computer Studies*, vol. 71, no. 3, 2013, pp. 250–60, <https://doi.org/10.1016/j.ijhcs.2012.09.005>.
- [6] Hollnagel, Erik, and Andreas Bye. "Principles for Modelling Function Allocation." *International Journal of Human-Computer Studies*, vol. 52, no. 2, 2000, pp. 253–65, <https://doi.org/10.1006/ijhc.1999.0288>.
- [7] Van Hee, Kees. *Workflow Management*. 1st ed., vol. 1, The MIT Press, 2002, <https://doi.org/10.7551/mitpress/7301.001.0001>.
- [8] Van der Aalst, Wil M. P., et al. "Robotic Process Automation." *Business & Information Systems Engineering*, vol. 60, no. 4, 2018, pp. 269–72, <https://doi.org/10.1007/s12599-018-0542-4>.
- [9] Valavanis, K. P., and K. M. Stellakis. "A General Organizer Model for Robotic Assemblies and Intelligent Robotic Systems." *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 2, 1991, pp. 302–17, <https://doi.org/10.1109/21.87079>.

- [10] A. C. Sanderson, “Applications of neural networks in robotics and automation for manufacturing,” CIRSSE Document no. 30, Rensselaer Polytechnic Institute, Troy, NY, 1988.
- [11] Saridis, G. “Intelligent Robotic Control.” *IEEE Transactions on Automatic Control*, vol. 28, no. 5, 1983, pp. 547–57, <https://doi.org/10.1109/TAC.1983.1103278>.
- [12] A. Meystel, “Intelligent control in robotics,” *J. Robotic Syst.*, vol. 5, no. 4, Aug. 1988
- [13] Charniak, Eugene., and McDermott, Drew V. *Introduction to Artificial Intelligence*. Addison-Wesley, 1985.
- [14] Homem de Mello, L. S., and A. C. Sanderson. “AND/OR Graph Representation of Assembly Plans.” *IEEE Transactions on Robotics and Automation*, vol. 6, no. 2, 1990, pp. 188–99, <https://doi.org/10.1109/70.54734>.
- [15] K. P. Valavanis, “A mathematical formulation for the analytical design of intelligent machines,” Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, NY, 1986
- [16] Homem de Mello, L. S., and A. C. Sanderson. “A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences.” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, 1991, pp. 228–40, <https://doi.org/10.1109/70.75905>.
- [17] Homem de Mello, L. S., and A. C. Sanderson. “Planning Repair Sequences Using the AND/OR Graph Representation of Assembly Plans.” *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, IEEE Comput. Soc. Press, 1988, pp. 1861–1862 vol.3, <https://doi.org/10.1109/ROBOT.1988.12341>.
- [18] Saridis, G. N. “Toward the Realization of Intelligent Controls.” *Proceedings of the IEEE*, vol. 67, no. 8, 1979, pp. 1115–33, <https://doi.org/10.1109/PROC.1979.11407>.
- [19] Tolis, Christos, and George F. Fragulis. *An Experimental Mechatronic Design and Control of a 5 DOF Robotic Arm for Identification and Sorting of Different Sized Objects*. 2017, <https://doi.org/10.48550/arxiv.1711.03808>.
- [20] Utmel. “Nema17 Stepper Motor: Datasheet PDF, 1.5 A 1.8° Stepper Motor and Dimensions.”

*Utmel*, Utmel Electronics, 29 Nov. 2021,

<https://www.utmel.com/components/nema17-stepper-motor-datasheet-pdf-1-5-a-1-8%C2%B-stepper-motor-and-dimensions?id=914>.

[21] Clifford, Paul. “Stepper Motor Specifications, NEMA 17 1.8 Degree 200 Steps-per-Revolution Four-Phase Unipolar Permanent-Magnet Stepper-Motor.” *Find Controllers for Instrumentation and Automation at the Mosaic Industries Site*, Mosaic Industries, Inc.,

<http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/stepper-motors/specifications>.

[22] Styger, Gary, et al. “Nema 17 Stepper Motor.” *OpenBuilds Part Store*,

<https://openbuildspartstore.com/nema-17-stepper-motor/>.

[23] “TB6600 Stepper Motor Driver with Arduino.” *MYTECTUTOR*,

<https://mytectutor.com/tb6600-stepper-motor-driver-with-arduino/>.

[24] *Mini Electric Linear Actuator Stroke 0.4"–Force 4.5 Lbs–12v | High ...*

<https://www.amazon.com/Actuator-Force-High-Speed-sec-Weight-Intelligent-Automation/dp/B0B4R8X95Q>.

[25] “Arduino - Actuator: Arduino Tutorial.” *Arduino Getting Started*,

<https://arduinogetstarted.com/tutorials/arduino-actuator>.

[26] Olson, Doug. “Guide to Working of Electric Linear Actuators.” *Venture Mfg. Co.*, 12 June 2019,

<https://www.venturemfgco.com/blog/guide-working-electric-linear-actuators/>.

[27] “MG996R Servo Motor.” *Components101*,

<https://components101.com/motors/mg996r-servo-motor-datasheet>.

[28] Dale, et al. “Xtension Limit Switch Kit.” *OpenBuilds Part Store*,

<https://openbuildspartstore.com/xtension-limit-switch-kit/>.

[29] Last Minute Engineers. “In-Depth Tutorial to Interface Micro SD Card Module with Arduino.”

*Last Minute Engineers*, Last Minute Engineers, 10 Oct. 2022,

<https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/>.

[30] “5pcs Micro SD Carte TF Adater Reader Module.” *5pcs Micro SD Carte TF Adater Reader Module*, Shenzhen HiLetgo Technology Co., Ltd,

<http://www.hiletgo.com/ProductDetail/2158021.html>.

[31] Last Minute Engineers. “In-Depth: Interface DS3231 Precision RTC Module with Arduino.” *Last Minute Engineers*, Last Minute Engineers, 10 Oct. 2022,

<https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/>.

[32] Howell, Bernarr. “Ultimate Guide to Arduino Mega 2560 Pinout, Specs & Schematic.” *ETechnophiles*, 3 Apr. 2023,

<https://www.etechnophiles.com/arduino-mega-pinout-pin-diagram-schematic-and-specifications-in-detail/#arduino-mega-pinoutdetailed-board-layout>.

[33] “NX8048P070-011R.” *Nextion*, 8 Nov. 2022,

<https://nextion.tech/datasheets/NX8048P070-011R/>.

[34] “Intelligent Series Introduction.” *Nextion*, 1 Dec. 2022,

<https://nextion.tech/intelligent-series-introduction/>.

[35] “The Nextion Editor Guide.” *Nextion*, 23 Dec. 2022, [https://nextion.tech/editor\\_guide/](https://nextion.tech/editor_guide/).

# Appendix A

## Smart Tool Organizer Code

```
#include <Wire.h>
#include <TimeLib.h>
#include <DS1307RTC.h>
#include <SD.h>
#include <SPI.h>
#include <Servo.h>           //servo

Servo myservo;
int pos = 0;

#define dirPinA 4           //stepper motors
#define stepPinAB 5
#define dirPinB 6
#define stepPinC 3
#define dirPinC 2

#define limitX A1          //limit switches
#define limitY A0

const int ENA_PIN = 7;
const int IN1_PIN = 8;
const int IN2_PIN = 9;

int dly=500;

int CS_PIN = 53;

File file;
String dfd = "";

union {
  char charByte[4];
  long valLong;
} value;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(500);
  pinMode(ENA_PIN, OUTPUT);
```



```

pinMode(IN1_PIN, OUTPUT);
pinMode(IN2_PIN, OUTPUT);

digitalWrite(ENA_PIN, HIGH);

pinMode(stepPinAB, OUTPUT);    //stepper motor pins
pinMode(dirPinA, OUTPUT);
pinMode(dirPinB, OUTPUT);
pinMode(dirPinC, OUTPUT);
pinMode(stepPinC, OUTPUT);

pinMode(limitY, INPUT);        //limit switches
pinMode(limitX, INPUT);

}

void loop() {

if(Serial.available()) {
dfd += char(Serial.read());
}

if ((dfd.substring(0,3)=="txt") & (dfd.length()>=12)){
Serial.println(dfd);
String test = dfd.substring(4);
Serial.println(test);
initializeSD();
createFile("test.txt");
char buf1[200];
test.toCharArray(buf1,test.length()+1);
Serial.println(buf1);
writeToFile(buf1);
closeFile();
dfd = "";
}

if (dfd == "ToolA"){
Serial.println("Tool 1");
while (!Serial) ;
delay(200);
Serial.println("-----");

initializeSD();
createFile("test.txt");

```

```

String now = "";

tmElements_t tm;

if (RTC.read(tm)) {
  Serial.print("Ok, Time = ");
  print2digits(tm.Hour);
  String h = String (tm.Hour);
  Serial.write(':');
  print2digits(tm.Minute);
  String m = String (tm.Minute);
  Serial.write(':');
  print2digits(tm.Second);
  String s = String (tm.Second);
  Serial.print(" , Date (D/M/Y) = ");
  Serial.print(tm.Day);
  String d = String (tm.Day);
  Serial.write('/');
  Serial.print(tm.Month);
  String mn = String (tm.Month);
  Serial.write('/');
  Serial.print(tmYearToCalendar(tm.Year));
  String y = String (tmYearToCalendar(tm.Year));
  Serial.println();
  now = ("Tool 1 Dispensed Time = " + h+ ":" +m+ ":" +s+ ",Date(D/M/Y)= "+ d + "/" + mn + "/" + y );
  Serial.println("hey" + now);
  char buf[200];
  now.toCharArray(buf,now.length()+1);
  Serial.println(buf);
  writeToFile(buf);
  closeFile();

} else {
  if (RTC.chipPresent()) {
    Serial.println("working");
  } else {
    Serial.println("not working");
  }
  delay(9000);
}
////////////////////////////////////Tool 1

if ((digitalRead(limitY) == 0) && (digitalRead(limitX) == 0)) {

          //direction from HOME position
  digitalWrite(dirPinA, HIGH);      //UP

```

```

digitalWrite(dirPinB, LOW);    //UP
digitalWrite(dirPinC, HIGH);   //Right

                                //steppers move to grab tool
for (int i = 0; i < 1525; i++) {
  digitalWrite(stepPinAB, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinAB, LOW);
  delayMicroseconds(1000);
}
for (int i = 0; i < 500; i++) {
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinC, LOW);
  delayMicroseconds(1000);
}
delay(2000);
digitalWrite(IN1_PIN, HIGH);    //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);            //CLOSE SERVO
for (pos = 0; pos <= 160; pos += 1) {
  myservo.write(pos);
  delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);     //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

                                //direction from Tool 1 position
digitalWrite(dirPinA, LOW);    //down
digitalWrite(dirPinB, HIGH);   //down
digitalWrite(dirPinC, HIGH);   //RIGHT

                                //steppers move to drop tool
for (int i = 0; i < 2450; i++) {
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
}

```

```

digitalWrite(stepPinC, LOW);
delayMicroseconds(1000);

}

for (int i = 0; i < 1245; i++) {
  digitalWrite(stepPinAB, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinAB, LOW);
  delayMicroseconds(1000);
}

delay(2000);
digitalWrite(IN1_PIN, HIGH);    //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(580);
digitalWrite(ENA_PIN, LOW);

delay(2000);
myservo.attach(10);            //OPEN SERVO
for (pos = 160; pos >= 0; pos -= 1) {
  myservo.write(pos);
  delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(ENA_PIN, HIGH);

digitalWrite(IN1_PIN, LOW);    //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

delay(2000);
}
if (!digitalRead(limitY) == 0) {

    //direction back to HOME position
    digitalWrite(dirPinA, LOW);    //DOWN
    digitalWrite(dirPinB, HIGH);    //DOWN
    //

do {

```

```

                //moves Y-steppers till Y-limit is ON
        digitalWrite(stepPinAB, HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepPinAB, LOW);
        delayMicroseconds(1000);
    } while (!digitalRead(limitY) == 0);
}
if (!digitalRead(limitX) == 0) {
    digitalWrite(dirPinC, LOW);    //LEFT
    do {
                //moves X-stepper till X-limit is ON
        digitalWrite(stepPinC, HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepPinC, LOW);
        delayMicroseconds(1000);
    } while (!digitalRead(limitX) == 0);

    delay(2000);

}

delay(1000);
dfd = "";
}

if (dfd == "ToolB"){
    Serial.println("Tool 2");
    while (!Serial) ;
    delay(200);
    Serial.println("-----");

    initializeSD();
    createFile("test.txt");

    String now = "";

    tmElements_t tm;

    if (RTC.read(tm)) {
        Serial.print("Ok, Time = ");
        print2digits(tm.Hour);
        String h = String (tm.Hour);
        Serial.write(':');
        print2digits(tm.Minute);
        String m = String (tm.Minute);
        Serial.write(':');
        print2digits(tm.Second);
    }
}

```

```

String s = String (tm.Second);
Serial.print(" , Date (D/M/Y) = ");
Serial.print(tm.Day);
String d = String (tm.Day);
Serial.write('/');
Serial.print(tm.Month);
String mn = String (tm.Month);
Serial.write('/');
Serial.print(tm.YearToCalendar(tm.Year));
String y = String (tm.YearToCalendar(tm.Year));
Serial.println();
now = ("Tool 2 Dispensed Time =" + h+ ":" +m+ ":" +s+ ",Date(D/M/Y)= "+ d + "/" + mn + "/" + y );
Serial.println("hey" + now);
char buf[200];
now.toCharArray(buf,now.length()+1);
Serial.println(buf);
writeToFile(buf);
closeFile();

} else {
if (RTC.chipPresent()) {
Serial.println("working");
} else {
Serial.println("not working");
}
delay(9000);
}

////////////////////////////////////Tool 2

if ((digitalRead(limitY) == 0) && (digitalRead(limitX) == 0)) {

//direction from HOME position
digitalWrite(dirPinA, HIGH); //UP
digitalWrite(dirPinB, LOW); //UP
digitalWrite(dirPinC, HIGH); //Right

//steppers move to grab tool
for (int i = 0; i < 1535; i++) {
digitalWrite(stepPinAB, HIGH);
delayMicroseconds(1000);
digitalWrite(stepPinAB, LOW);
delayMicroseconds(1000);
}
}

```

```

}
for (int i = 0; i < 1000; i++) {
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinC, LOW);
  delayMicroseconds(1000);

}
delay(2000);
digitalWrite(IN1_PIN, HIGH);      //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);              //CLOSE SERVO
for (pos = 0; pos <= 160; pos += 1) {
  myservo.write(pos);
  delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);      //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

                                //direction from Tool 3 position
digitalWrite(dirPinA, LOW);      //down
digitalWrite(dirPinB, HIGH);     //down
digitalWrite(dirPinC, HIGH);     //RIGHT

                                //steppers move to drop tool
for (int i = 0; i < 1970; i++) {
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinC, LOW);
  delayMicroseconds(1000);

}

for (int i = 0; i < 1235; i++) {
  digitalWrite(stepPinAB, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinAB, LOW);
  delayMicroseconds(1000);

}

```

```

delay(2000);
digitalWrite(IN1_PIN, HIGH);    //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);           //OPEN SERVO
for (pos = 160; pos >= 0; pos -= 1) {
  myservo.write(pos);
  delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);    //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

delay(2000);
}
if (!digitalRead(limitY) == 0) {

    //direction back to HOME position
    digitalWrite(dirPinA, LOW);    //DOWN
    digitalWrite(dirPinB, HIGH);    //DOWN
    //

do {

    //moves Y-steppers till Y-limit is ON
    digitalWrite(stepPinAB, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinAB, LOW);
    delayMicroseconds(1000);
} while (!digitalRead(limitY) == 0);
}
if (!digitalRead(limitX) == 0) {
    digitalWrite(dirPinC, LOW);    //LEFT
do {

    //moves X-stepper till X-limit is ON
    digitalWrite(stepPinC, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinC, LOW);
    delayMicroseconds(1000);
} while (!digitalRead(limitX) == 0);

delay(2000);

```



```

}

dfd = "";
}

if (dfd == "ToolC"){
    Serial.println("Tool 3");
    while (!Serial) ;
    delay(200);
    Serial.println("-----");

    initializeSD();
    createFile("test.txt");

    String now = "";

    tmElements_t tm;

    if (RTC.read(tm)) {
        Serial.print("Ok, Time = ");
        print2digits(tm.Hour);
        String h = String (tm.Hour);
        Serial.write(':');
        print2digits(tm.Minute);
        String m = String (tm.Minute);
        Serial.write(':');
        print2digits(tm.Second);
        String s = String (tm.Second);
        Serial.print(", Date (D/M/Y) = ");
        Serial.print(tm.Day);
        String d = String (tm.Day);
        Serial.write('/');
        Serial.print(tm.Month);
        String mn = String (tm.Month);
        Serial.write('/');
        Serial.print(tm.YearToCalendar(tm.Year));
        String y = String (tm.YearToCalendar(tm.Year));
        Serial.println();
        now = ("Tool 3 Dispensed Time = " + h + ":" + m + ":" + s + ",Date(D/M/Y)= " + d + "/" + mn + "/" + y );
        Serial.println("hey" + now);
        char buf[200];
        now.toCharArray(buf,now.length()+1);
        Serial.println(buf);
        writeToFile(buf);
        closeFile();

    } else {

```

```

if (RTC.chipPresent()) {
  Serial.println("working");
} else {
  Serial.println("not working");
}
delay(9000);
}

```

```

////////////////////////////////////Tool 3

```

```

if ((digitalRead(limitY) == 0) && (digitalRead(limitX) == 0)) {

```

```

    //direction from HOME position

```

```

    digitalWrite(dirPinA, HIGH);    //UP
    digitalWrite(dirPinB, LOW);     //UP
    digitalWrite(dirPinC, HIGH);    //Right

```

```

    //steppers move to grab tool

```

```

    for (int i = 0; i < 1515; i++) {
      digitalWrite(stepPinAB, HIGH);
      digitalWrite(stepPinC, HIGH);
      delayMicroseconds(1000);
      digitalWrite(stepPinAB, LOW);
      digitalWrite(stepPinC, LOW);
      delayMicroseconds(1000);
    }

```

```

    for (int i = 0; i < 60; i++) {
      digitalWrite(stepPinAB, HIGH);
      delayMicroseconds(1000);
      digitalWrite(stepPinAB, LOW);
      delayMicroseconds(1000);

```

```

    }
    delay(2000);
    digitalWrite(IN1_PIN, HIGH);    //OPEN ACTUATOR
    digitalWrite(IN2_PIN, LOW);
    delay(2000);
    myservo.attach(10);            //CLOSE SERVO
    for (pos = 0; pos <= 160; pos += 1) {
      myservo.write(pos);
      delay(15);
    }
    myservo.detach();

```

```

delay(2000);
digitalWrite(IN1_PIN, LOW);    //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);
                                //direction from Tool 3 position
digitalWrite(dirPinA, LOW);    //down
digitalWrite(dirPinB, HIGH);   //down
digitalWrite(dirPinC, HIGH);   //RIGHT

                                //steppers move to drop tool
for (int i = 0; i < 1460; i++) {
digitalWrite(stepPinC, HIGH);
delayMicroseconds(1000);
digitalWrite(stepPinC, LOW);
delayMicroseconds(1000);

}

for (int i = 0; i < 1285; i++) {
digitalWrite(stepPinAB, HIGH);
delayMicroseconds(1000);
digitalWrite(stepPinAB, LOW);
delayMicroseconds(1000);

}

delay(2000);
digitalWrite(IN1_PIN, HIGH);   //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);           //OPEN SERVO
for (pos = 160; pos >= 0; pos -= 1) {
myservo.write(pos);
delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);    //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

```

```

    delay(2000);
}
if (!digitalRead(limitY) == 0) {

    //direction back to HOME position
    digitalWrite(dirPinA, LOW);    //DOWN
    digitalWrite(dirPinB, HIGH);   //DOWN
    //

    do {

        //moves Y-steppers till Y-limit is ON
        digitalWrite(stepPinAB, HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepPinAB, LOW);
        delayMicroseconds(1000);
    } while (!digitalRead(limitY) == 0);
}
if (!digitalRead(limitX) == 0) {
    digitalWrite(dirPinC, LOW);    //LEFT
    do {

        //moves X-stepper till X-limit is ON
        digitalWrite(stepPinC, HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepPinC, LOW);
        delayMicroseconds(1000);
    } while (!digitalRead(limitX) == 0);

    delay(2000);

}

dfd = "";
}
if (dfd == "ToolD"){
    Serial.println("Return Tool 1");
    while (!Serial) ;
    delay(200);
    Serial.println("-----");

    initializeSD();
    createFile("test.txt");

    String now = "";

    tmElements_t tm;

    if (RTC.read(tm)) {

```

```

Serial.print("Ok, Time = ");
print2digits(tm.Hour);
String h = String (tm.Hour);
Serial.write(':');
print2digits(tm.Minute);
String m = String (tm.Minute);
Serial.write(':');
print2digits(tm.Second);
String s = String (tm.Second);
Serial.print(" , Date (D/M/Y) = ");
Serial.print(tm.Day);
String d = String (tm.Day);
Serial.write('/');
Serial.print(tm.Month);
String mn = String (tm.Month);
Serial.write('/');
Serial.print(tmYearToCalendar(tm.Year));
String y = String (tmYearToCalendar(tm.Year));
Serial.println();
now = ("Tool 1 Return Time = " + h + ":" + m + ":" + s + ",Date(D/M/Y)= " + d + "/" + mn + "/" + y );
Serial.println("hey" + now);
char buf[200];
now.toCharArray(buf,now.length()+1);
Serial.println(buf);
writeToFile(buf);
closeFile();

} else {
  if (RTC.chipPresent()) {
    Serial.println("working");
  } else {
    Serial.println("not working");
  }
  delay(9000);
}

```

```

//////////////////////////////////////Return Tool 1

```

```

if ((digitalRead(limitY) == 0) && (digitalRead(limitX) == 0)) {

    //direction from HOME position
    digitalWrite(dirPinA, HIGH);    //UP
    digitalWrite(dirPinB, LOW);    //UP
    digitalWrite(dirPinC, HIGH);    //Right

```

```

                //steppers move to grab tool
for (int i = 0; i < 290; i++) {
  digitalWrite(stepPinAB, HIGH);
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinAB, LOW);
  digitalWrite(stepPinC, LOW);
  delayMicroseconds(1000);
}
for (int i = 0; i < 2660; i++) {
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinC, LOW);
  delayMicroseconds(1000);
}
delay(2000);
digitalWrite(IN1_PIN, HIGH);      //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);              //CLOSE SERVO
for (pos = 0; pos <= 160; pos += 1) {
  myservo.write(pos);
  delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);      //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);
                //direction from dropbox Tool 1 position
digitalWrite(dirPinA, HIGH);     //up
digitalWrite(dirPinB, LOW);      //up
digitalWrite(dirPinC, LOW);      //LEFT

                //steppers move to drop tool
for (int i = 0; i < 1235; i++) {
  digitalWrite(stepPinAB, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinAB, LOW);
  delayMicroseconds(1000);
}

```

```

}
  for (int i = 0; i < 2475; i++) {
    digitalWrite(stepPinC, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinC, LOW);
    delayMicroseconds(1000);
  }

delay(2000);
digitalWrite(IN1_PIN, HIGH);    //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);            //OPEN SERVO
for (pos = 160; pos >= 0; pos -= 1) {
  myservo.write(pos);
  delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);    //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

delay(2000);
}
if (!digitalRead(limitY) == 0) {

    //direction back to HOME position
    digitalWrite(dirPinA, LOW);    //DOWN
    digitalWrite(dirPinB, HIGH);    //DOWN
    //

do {

    //moves Y-steppers till Y-limit is ON
    digitalWrite(stepPinAB, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinAB, LOW);
    delayMicroseconds(1000);
  } while (!digitalRead(limitY) == 0);
}
if (!digitalRead(limitX) == 0) {
  digitalWrite(dirPinC, LOW);    //LEFT
do {

```

```

                //moves X-stepper till X-limit is ON
        digitalWrite(stepPinC, HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepPinC, LOW);
        delayMicroseconds(1000);
    } while (!digitalRead(limitX) == 0);

    delay(2000);

}

dfd = "";
}

if (dfd == "ToolE"){
    Serial.println("Return Tool 2");
    while (!Serial) ;
    delay(200);
    Serial.println("-----");

    initializeSD();
    createFile("test.txt");

    String now = "";

    tmElements_t tm;

    if (RTC.read(tm)) {
        Serial.print("Ok, Time = ");
        print2digits(tm.Hour);
        String h = String (tm.Hour);
        Serial.write(':');
        print2digits(tm.Minute);
        String m = String (tm.Minute);
        Serial.write(':');
        print2digits(tm.Second);
        String s = String (tm.Second);
        Serial.print(", Date (D/M/Y) = ");
        Serial.print(tm.Day);
        String d = String (tm.Day);
        Serial.write('/');
        Serial.print(tm.Month);
        String mn = String (tm.Month);
        Serial.write('/');
        Serial.print(tmYearToCalendar(tm.Year));
        String y = String (tmYearToCalendar(tm.Year));
        Serial.println();
    }
}

```



```

now = ("Tool 2 Return Time =" + h+ ":" +m+ ":" +s+ ",Date(D/M/Y)=" + d + "/" + mn + "/" + y );
Serial.println("hey" + now);
char buf[200];
now.toCharArray(buf,now.length()+1);
Serial.println(buf);
writeToFile(buf);
closeFile();

} else {
if (RTC.chipPresent()) {
  Serial.println("working");

} else {
  Serial.println("not working");
}
delay(9000);
}

////////////////////////////////////Return Tool 2

if ((digitalRead(limitY) == 0) && (digitalRead(limitX) == 0)) {

          //direction from HOME position
digitalWrite(dirPinA, HIGH);      //UP
digitalWrite(dirPinB, LOW);       //UP
digitalWrite(dirPinC, HIGH);      //Right

          //steppers move to grab tool
for (int i = 0; i < 290; i++) {
  digitalWrite(stepPinAB, HIGH);
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinAB, LOW);
  digitalWrite(stepPinC, LOW);
  delayMicroseconds(1000);

}
for (int i = 0; i < 2660; i++) {
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinC, LOW);
  delayMicroseconds(1000);

}
}

```

```

delay(2000);
digitalWrite(IN1_PIN, HIGH);      //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);              //CLOSE SERVO
for (pos = 0; pos <= 160; pos += 1) {
  myservo.write(pos);
  delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);      //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

                                //direction from dropbox Tool 1 position
digitalWrite(dirPinA, HIGH);     //up
digitalWrite(dirPinB, LOW);      //up
digitalWrite(dirPinC, LOW);      //LEFT

                                //steppers move to drop tool
for (int i = 0; i < 1240; i++) {
  digitalWrite(stepPinAB, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinAB, LOW);
  delayMicroseconds(1000);
}
for (int i = 0; i < 1965; i++) {
  digitalWrite(stepPinC, HIGH);
  delayMicroseconds(1000);
  digitalWrite(stepPinC, LOW);
  delayMicroseconds(1000);
}

delay(2000);
digitalWrite(IN1_PIN, HIGH);     //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);              //OPEN SERVO
for (pos = 160; pos >= 0; pos -= 1) {
  myservo.write(pos);
  delay(15);
}

```

```

}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);    //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

delay(2000);
}
if (!digitalRead(limitY) == 0) {

    //direction back to HOME position
    digitalWrite(dirPinA, LOW);    //DOWN
    digitalWrite(dirPinB, HIGH);    //DOWN
    //

do {

    //moves Y-steppers till Y-limit is ON
    digitalWrite(stepPinAB, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinAB, LOW);
    delayMicroseconds(1000);
} while (!digitalRead(limitY) == 0);
}
if (!digitalRead(limitX) == 0) {
    digitalWrite(dirPinC, LOW);    //LEFT
do {

    //moves X-stepper till X-limit is ON
    digitalWrite(stepPinC, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinC, LOW);
    delayMicroseconds(1000);
} while (!digitalRead(limitX) == 0);

delay(2000);

}

dfd = "";
}

if (dfd == "ToolF"){
    Serial.println("Return Tool 3");
    while (!Serial) ;
delay(200);
Serial.println("-----");
}

```

```

initializeSD();
createFile("test.txt");

String now = "";

tmElements_t tm;

if (RTC.read(tm)) {
  Serial.print("Ok, Time = ");
  print2digits(tm.Hour);
  String h = String (tm.Hour);
  Serial.write(':');
  print2digits(tm.Minute);
  String m = String (tm.Minute);
  Serial.write(':');
  print2digits(tm.Second);
  String s = String (tm.Second);
  Serial.print(", Date (D/M/Y) = ");
  Serial.print(tm.Day);
  String d = String (tm.Day);
  Serial.write('/');
  Serial.print(tm.Month);
  String mn = String (tm.Month);
  Serial.write('/');
  Serial.print(tm.YearToCalendar(tm.Year));
  String y = String (tm.YearToCalendar(tm.Year));
  Serial.println();
  now = ("Tool 3 Return Time = " + h + ":" + m + ":" + s + ",Date(D/M/Y)= " + d + "/" + mn + "/" + y );
  Serial.println("hey" + now);
  char buf[200];
  now.toCharArray(buf,now.length()+1);
  Serial.println(buf);
  writeToFile(buf);
  closeFile();

} else {
  if (RTC.chipPresent()) {
    Serial.println("working");
  } else {
    Serial.println("not working");
  }
  delay(9000);
}

```

```

////////////////////////////////////Return Tool 3

```

```

if ((digitalRead(limitY) == 0) && (digitalRead(limitX) == 0)) {

    //direction from HOME position
    digitalWrite(dirPinA, HIGH);    //UP
    digitalWrite(dirPinB, LOW);    //UP
    digitalWrite(dirPinC, HIGH);    //Right

    //steppers move to grab tool
    for (int i = 0; i < 290; i++) {
        digitalWrite(stepPinAB, HIGH);
        digitalWrite(stepPinC, HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepPinAB, LOW);
        digitalWrite(stepPinC, LOW);
        delayMicroseconds(1000);
    }
    for (int i = 0; i < 2660; i++) {
        digitalWrite(stepPinC, HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepPinC, LOW);
        delayMicroseconds(1000);
    }
    delay(2000);
    digitalWrite(IN1_PIN, HIGH);    //OPEN ACTUATOR
    digitalWrite(IN2_PIN, LOW);
    delay(2000);
    myservo.attach(10);            //CLOSE SERVO
    for (pos = 0; pos <= 160; pos += 1) {
        myservo.write(pos);
        delay(15);
    }
    myservo.detach();
    delay(2000);
    digitalWrite(IN1_PIN, LOW);    //CLOSE ACTUATOR
    digitalWrite(IN2_PIN, HIGH);
    delay(2000);

    //direction from dropbox Tool 1 position
    digitalWrite(dirPinA, HIGH);    //up
    digitalWrite(dirPinB, LOW);    //up
    digitalWrite(dirPinC, LOW);    //LEFT

```

```

                //steppers move to drop tool
for (int i = 0; i < 1275; i++) {
    digitalWrite(stepPinAB, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinAB, LOW);
    delayMicroseconds(1000);

}
for (int i = 0; i < 1465; i++) {
    digitalWrite(stepPinC, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinC, LOW);
    delayMicroseconds(1000);

}

delay(2000);
digitalWrite(IN1_PIN, HIGH);    //OPEN ACTUATOR
digitalWrite(IN2_PIN, LOW);
delay(2000);
myservo.attach(10);            //OPEN SERVO
for (pos = 160; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(15);
}
myservo.detach();
delay(2000);
digitalWrite(IN1_PIN, LOW);    //CLOSE ACTUATOR
digitalWrite(IN2_PIN, HIGH);
delay(2000);

delay(2000);
}
if (!digitalRead(limitY) == 0) {

                //direction back to HOME position
digitalWrite(dirPinA, LOW);    //DOWN
digitalWrite(dirPinB, HIGH);   //DOWN
//

do {

                //moves Y-steppers till Y-limit is ON

```

```

    digitalWrite(stepPinAB, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinAB, LOW);
    delayMicroseconds(1000);
  } while (!digitalRead(limitY) == 0);
}
if (!digitalRead(limitX) == 0) {
  digitalWrite(dirPinC, LOW);    //LEFT
  do {
    //moves X-stepper till X-limit is ON
    digitalWrite(stepPinC, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPinC, LOW);
    delayMicroseconds(1000);
  } while (!digitalRead(limitX) == 0);

  delay(2000);

}
dfd = "";
}

}
void print2digits(int number) {
  if (number >= 0 && number < 10) {
    Serial.println("working");

    Serial.write('0');
  }
  Serial.println("working");
  Serial.print(number);
}
void initializeSD()
{
  pinMode(CS_PIN, OUTPUT);

  if (SD.begin())
  {
  } else
  {
    return;
  }
}

int createFile(char filename[])
{
  file = SD.open(filename, FILE_WRITE);

```

```

if (file)
{
    return 1;
} else
{
    return 0;
}
}

int writeToFile(char text[])
{
    if (file)
    {
        file.println(text);
        Serial.println(text);
        return 1;
    } else
    {
        return 0;
    }
}

void closeFile()
{
    if (file)
    {
        file.close();
    }
}

int openFile(char filename[])
{
    file = SD.open(filename);
    if (file)
    {
        return 1;
    } else
    {
        return 0;
    }
}

String readLine()
{
    String received = "";
    char ch;

```



```
while (file.available())
{Serial.println("working");
  ch = file.read();
  if (ch == '\n')
  {Serial.println("working");
    return String(received);
  }
  else
  {Serial.println("working");
    received += ch;
  }
}
return "";
}
```