

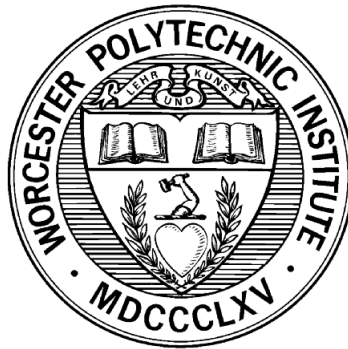
Automated Tool Prep/Crib

A Major Qualifying Project Report: Submitted to the faculty of the
WORCESTER POLYTECHNIC INSTITUTE In partial fulfillment of
the requirements for the Degree of Bachelor of Science

Wheeler, Ryan

Willcox, Eric

Zolotykh, Sergey



Advisors:

Prof. Craig Putnam, Major Advisor

Prof. Stephen S. Nestinger, Co-Advisor

Prof. Lifeng Lai, Major Advisor

Executive Summary

General Electric Aviation uses CNC machines for manufacturing machined parts. The process uses end mills inside of tool holders, called an assembly, that need to be replaced by hand after they are used for a milling process. To eliminate this manual task, GE wanted to use a robotic industrial arm they purchased from Applied Controls Technology (ACT) to try to start automating this task with four main goals. The robot needs to be able to receive used end mills from a worker, it needs to be able to replace the used end mill with a new one, and finally it needs to be able to properly tighten the collet nut.

A process was developed that will complete all the project goals and much of it was implemented, but not all. The processes planned and partially constructed throughout this project show potential for the final goals of the project that can be realized with some more work and refining.

A second goal of the project was to create a method that is capable of tracking tool holders within GE's facilities with the primary purpose of keeping their job histories. An RFID solution was designed and fully implemented using off the shelf consumer parts that met three design goals. The assemblies all need to have all jobs they work on logged with a timestamp, the current location needs to be tracked, and the total amount of time the tool holder has been used needs to be kept.

Testing shows that the project is completely feasible and could easily be implemented at GE's facility by just changing the RFID reader type to a more powerful industrial solution.

Contents

- 1. Introduction** **1**
 - 1.1. Project Description 3

- 2. Changing and Storage** **5**
 - 2.1. Current Changing Process 5
 - 2.2. Brainstorming 6
 - 2.2.1. Intake and Outtake 6
 - 2.2.2. Removing Sheaths 6
 - 2.2.3. Changing End Mills 7
 - 2.3. Flowcharts 8
 - 2.3.1. High Level 8
 - 2.3.2. Low Level 10
 - 2.4. Selected System Ideas 12
 - 2.4.1. Assembly Receiver/Dispatcher 12
 - 2.4.2. Vise 13
 - 2.4.2.1. Indexing Table 16
 - 2.4.2.2. Mounting 19
 - 2.4.2.3. Analysis 20
 - 2.4.3. Gripper 21
 - 2.4.3.1. Gripper Analysis 23

2.5. Models	27
2.5.1. Arm: MATLAB	27
2.5.2. Arm: SolidWorks	28
2.5.3. Work Cell Layout: SolidWorks	29
2.6. Implementation and Testing	30
2.6.1. Drawer	31
2.6.1.1. Performance	31
2.6.2. Vise	32
2.6.2.1. Performance	33
2.6.3. Gripper	35
2.6.3.1. Performance	36
2.6.4. Work Cell Layout	38
2.6.5. Programming and Putting it Together	38
2.7. Safety Considerations	40
2.8. Aesthetics	41
2.9. Social and Ethical Impact	42
3. Tool Tracking	43
3.1. Background	43
3.2. Preliminary Planning	44
3.3. Flowcharts	46
3.3.1. Changing Cell	46
3.3.2. Storage In and Out	47
3.3.3. CNC Carrousel	49
3.4. Reader Selection	51
3.5. Breadboard Circuit	52

3.6. Selecting a Communication Method	54
3.6.1. Overview	54
3.6.2. Wireless	54
3.6.3. Wired	55
3.7. Database Selection	56
3.8. Controller Selection	57
3.8.1. Arduino Uno and Ethernet Shield	57
3.8.2. Arduino Yun	58
3.9. Network Plan	59
3.10. Portable Hand Scanner	61
3.11. Implementation and Testing	63
3.11.1. Database	63
3.11.2. Web Interface	63
3.11.2.1. Basic Queries	64
3.11.2.2. Searching	65
3.11.2.3. Admin Station	69
3.11.3. Portable Scanner	69
3.11.3.1. Results	71
3.11.4. CNC Machine	71
3.11.4.1. Results	71
3.11.5. Changing and Storage	72
3.11.5.1. Results	72
4. Further Discussion and Future Work	73
4.1. Discussion	73
4.1.1. Repeatability Considerations	73
4.1.1.1. Tool Holder	73

4.1.1.2. Tool Tracking	74
4.1.2. Economic Considerations	74
4.1.2.1. Tool Changing	74
4.1.2.2. Tool Tracking	75
4.2. Future Work	76
4.2.1. Tool Changing	76
4.2.2. Tool Tracking	77
5. Conclusions	78
Appendices	80
A. FANUC DH Parameters	81
B. MATLAB Kinematics Code	83
B.1. translate.m	83
B.2. tlink.m	83
B.3. rotate.m	84
B.4. fulltrans.m	85
B.5. FANUC.m	85
C. New Yun Setup	87
D. Plates and Pointer	88
E. Reader Code	90
E.1. Arduino Side (C)	90
E.1.1. CNC Reader	90
E.1.2. Hand Scanner	95
E.1.3. All others	100

E.2. Linux Side (Python 2.7)	104
E.2.1. Admin Station	104
E.2.2. CNC Reader	109
E.2.3. Changing and Storage	113
E.2.4. Portable Reader	115
F. Web Code and Material	118
F.1. Home Page	118
F.2. Management Page	120
F.3. Adding a Tool Holder	121
F.3.1. User End	121
F.3.2. Back End	123
F.4. Editing Data	124
F.4.1. User End	124
F.4.2. Back End	127
F.5. Deleting Data	128
F.6. Searching	129
F.6.1. User End	129
F.6.2. Back End	132
F.7. Finding the History	136
F.8. Reading from Administration Station	137
F.8.1. User End	137
F.8.2. Back End	139
F.9. Text Files	140
F.9.1. Admin Station	140
F.9.2. Drawing Models	140
F.9.3. Machines and Other Locations	141

F.10. Calendar Script	141
F.11. CSS	147
F.11.1. Website	147
F.11.2. Calendar	148
F.12. Images	149

List of Figures

1.1. Tool holder exploded	1
1.2. CNC machine	2
1.3. Bladed Disk	2
1.4. Robotic arm with work cell	3
2.1. End mill with a sheath	7
2.2. Tool holder Channel	8
2.3. High level changing flowchart	9
2.4. Low level changing flowchart	11
2.5. Recieving/Dispatching Box	13
2.6. Locking Pins	14
2.7. Vise jaw internals	15
2.8. Jaws snapping into collet nut channels	15
2.9. Indexing table for spinning the assembly	16
2.10. Motor controller for indexing table	17
2.11. Relay schematic for controlling motor direction	18
2.12. H bridge for controlling indexer	18
2.13. Table for mounting vise	20
2.14. Deformation of jaw pins	21
2.15. Gripper	22

2.16. Simulation model	23
2.17. Gripper deformation	24
2.18. Gripper deformation	25
2.19. ABS gripper deformation	26
2.20. Gripper stress	27
2.21. MATLAB model	28
2.22. FANUC SolidWorks model	29
2.23. Planned work cell layout	30
2.24. Completed drawer assembly	31
2.25. Constructed vise	32
2.26. Constructed relays	33
2.27. Temporary fix for the vise jaws	34
2.28. Grippers made of PLA	35
2.29. Wooden and aluminum grippers	35
2.30. Severe deformation of the grippers	36
2.31. Actual cell layout	38
2.32. Teach pendant	39
2.33. Complete changing process	40
3.1. Preliminary RFID reader layout	45
3.2. Automated tool crib flowchart	47
3.3. Storage in flowchart	48
3.4. Storage out flowchart	49
3.5. CNC carousel flowchart	50
3.6. RFID reader and tag	52
3.7. Reader circuit design	53
3.8. Arduino Uno	58

3.9. Arduino Yun	59
3.10. Network plan	60
3.11. Portable RFID Scanner	62
3.12. Portable RFID scanner internals	62
3.13. Web interface	64
3.14. Searching for a tool holder	65
3.15. Date picker for filtering	66
3.16. Search results for jobs in a past machines	67
3.17. Job history of an individual tool holder	68
3.18. Web Admin page	69
3.19. Portable RFID scanner printed	70
3.20. Portable scanner output	71
D.1. Arm plates and Pointer	89
F.1. Website banner	149
F.2. Website icon	149

Acknowledgements

The team would like to thank the following individuals for their help with this MQP. Without their guidance, this project would not have been possible.

Pi Thanacha Choopojcharoen

Tracey Coetzee

David Ephraim

Joseph St. Germain

Lifeng Lai

Stephen Nestinger

Craig Putnam

Abstract

This MQP has two goals that aim to show the feasibility of automating CNC maintenance tasks for General Electric Aviation. The first goal was to design a system capable of changing used end mills within a tool holder used by CNC machines. Workers currently have to spend an entire day going around to every machine and changing out all of these used end mills with fresh ones for other jobs. A process using an industrial arm owned by GE was created capable of obtaining this goal with reasonable constraints.

The second goal was to create a method to automatically keep track of the location of all tool holders as well as keep a log of all jobs on which they have been used. GE currently does not keep track of their tool holders in any way which causes problems when a part becomes defective. Workers have to try to locate a single tool holder amongst the hundreds that they have at their facility which is a very difficult process.

The end results showed that both goals are possible and with more work could be implemented in a working production level system. The processes designed can be expanded for more functionality until GE either implements them or disbands the goal.

1. Introduction

The purpose of this Major Qualifying Project (MQP) was to develop a robotic process for General Electric Aviation (GE) to remove and replace end mills in industrial machine tool holders. The combination of a tool holder and all of its constituent parts (collet, collet nut, etc.) plus an end mill will be called an 'assembly'; an example can be seen in Figure 1.1.



Figure 1.1.: Exploded view of an assembly

GE uses these assemblies within their Computer Numerical Control (CNC) machines to create bladed disks (blisks) that can be seen in Figures 1.2 and 1.3 respectively.



Figure 1.2.: Example of a CNC machine

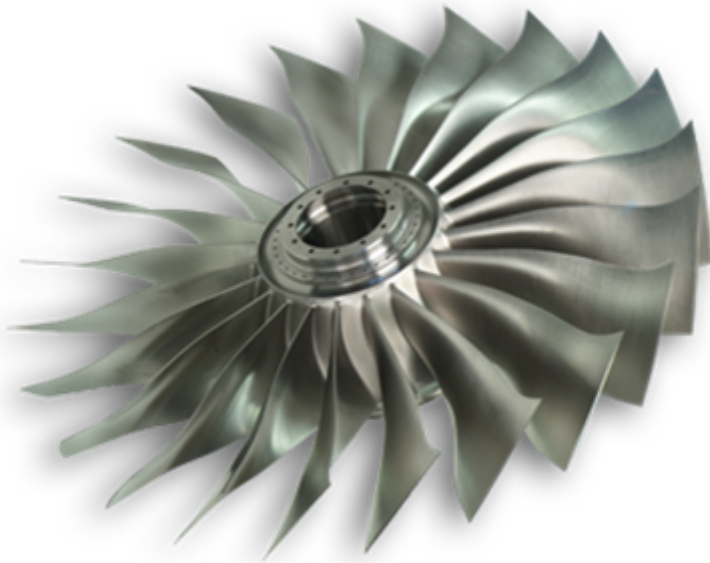
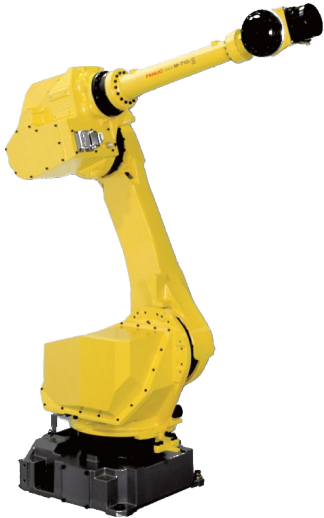


Figure 1.3.: An example of a blisk that can be made by a CNC process

GE's end vision is to have the robot work cell be able to receive a number of new end mills and used assemblies within a workers cart, and have the robot automatically replace the old end mills with new end mills just as a worker would.

GE has provided a robotic work cell for the MQP that contains a FANUC robotic arm, both of which can be seen in Figure 1.4. A budget of \$5,000 has also been generously provided for any additional tools and materials that are deemed necessary for meeting the end goals of the project.



(a) FANUC arm



(b) Outside of the work cell

Figure 1.4.: Robotic arm with work cell

1.1. Project Description

GE currently have a few processes that are all done by hand involving the assemblies. First, after an end mill has been used in a job it has to be replaced with a new one and the old one shipped off to be sharpened by an external vendor. Next, the new assembly is then either used immediately or stocked in a vending type machine that is able to store it until it is needed at a later time.

A problem with the tool holders is that they are currently not tracked in any way. If one should malfunction and it isn't caught immediately, it may be used in other jobs before the problem is detected in the first part. It's currently difficult to try to find all tool holders that were used on any given job making it hard to find where the broken

tool holder has gone to. GE would like to have a system created that is able to keep track of a tool holder's job history as well as the ability to find what end mill is currently in it and where it is located at any time.

The robot would ideally be able to verify the absolute length of the end mill within the tool holder using some sensor and communicate it with the CNC machines. The current process has the measurements being done within the CNC machine as is and there is not a lot of error handling on receiving wrong tools. The robot would then store all new assemblies in a storage system to be given out on an as-needed basis to workers, similar to the current vending machine process that is in use now. As tools enter and leave the storage area, they would be tracked for which jobs they are coming from and going to and the CNC machine would know if a worker installed an assembly into an incorrect carousel.

The problems were approached separately, and the tool changing and tool crib problems can be seen in Chapter 2 while tool tracking can be found in Chapter 3. Wrap-up comments as well as future recommendations are then given in Chapters 4 and 5.

2. Changing and Storage

To begin, each problem had to be broken down into smaller, manageable sizes so that it could be tackled. The first step was to come up with a process that the FANUC would be able to follow to complete each of the following four basic objectives:

1. Take an assembly in/out of the cell
2. Remove the end mill from the tool holder
3. Insert a new end to the proper length mill and tighten the tool holder
4. Move assembly to a storage area

2.1. Current Changing Process

To change a used end mill and tool holder combo (assembly), there are currently multiple steps that a worker needs to perform. The first step is to check the operational log on each machine to see which assemblies need to be removed from the system and, if any, take them out of the tool carousel. Once removed from the machine, the worker then has to bring them to a vise where they lock in the tool holder and, with a wrench, loosen its hold on the end mill and remove it from the collet that holds it. They then put the used end mill into a sheath to protect it and move it to an area to be sent off to be sharpened. Next, a new end mill is removed from its sheath and then placed into

the tool holder and is set with a special jig, and then the collet nut is tightened until it cannot physically turn any more. The tool holder then has to have its grip on the end mill loosened very slightly, called backing off, so the hold on the end mill is still firm but not so much as to cause damage. The new assembly can then be removed from the vise and stored in a holding area until a worker requests it from the tool crib.

2.2. Brainstorming

A lot of ideas were created to start trying to think about how to best approach each problem. A short general idea for each follow.

2.2.1. Intake and Outtake

Using one of the doors into the work cell that already exist, a mechanism can be devised to take advantage of the existing entryway into the cell. A drawer or a table can be created with holes in it to place assemblies along with new end mills still in their sheaths. A drawer method would prevent any workers from ever having any parts of their bodies within the cell as a hole could be put into the work cell that the drawer would slide in and out of. A table method however would involve a user opening one of the doors and reaching within the work cell, potentially putting them in danger if the robot were to move to their location. When a worker requests a new assembly, the drawer or table could just be loaded up by the robot with new ones from its storage and the worker can pull them out when ready.

2.2.2. Removing Sheaths

Before the end mills can be manipulated, the sheath they came in needs to be removed. A simple gripper that will hold one half of the sheath can be used to allow the robot to

lift off the other half, allowing for access to the end mill. The new end mill can then be replaced with the used one and repackaged in the reverse order and stored on the side to be resharpened. Figure 2.1 shows an end mill along with the two parts of the sheath it comes packaged in.



Figure 2.1.: End mill with a sheath

2.2.3. Changing End Mills

To change an end mill the assembly, namely the tool holder body, will need to be fixed in place. The vise that GE uses will either need to be utilized or something that performs the same function (with additional features) will need to be created. The tool holders have a vertical channel in them that the manual vise GE uses to hold them in place while a worker manipulates the tool holder as needed. These can also be used if a vise is created and can be used by the intake/outake system for alignment purposes to ensure a known position. This channel can be seen on a tool holder in Figure 2.2.

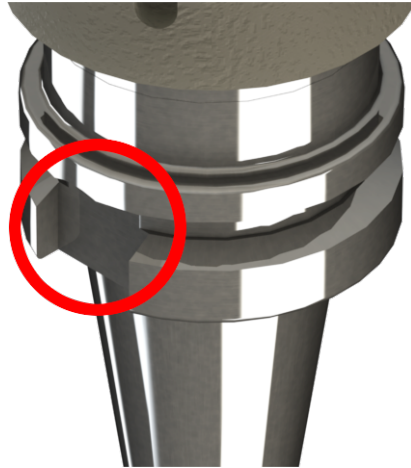


Figure 2.2.: Tool holder Channel

2.3. Flowcharts

Flowcharts detailing the overall system operation were then created using the brainstormed ideas to start figuring out how everything will interconnect.

2.3.1. High Level

Figure 2.3 shows the high level flowchart that breaks down the entire tool changing process.

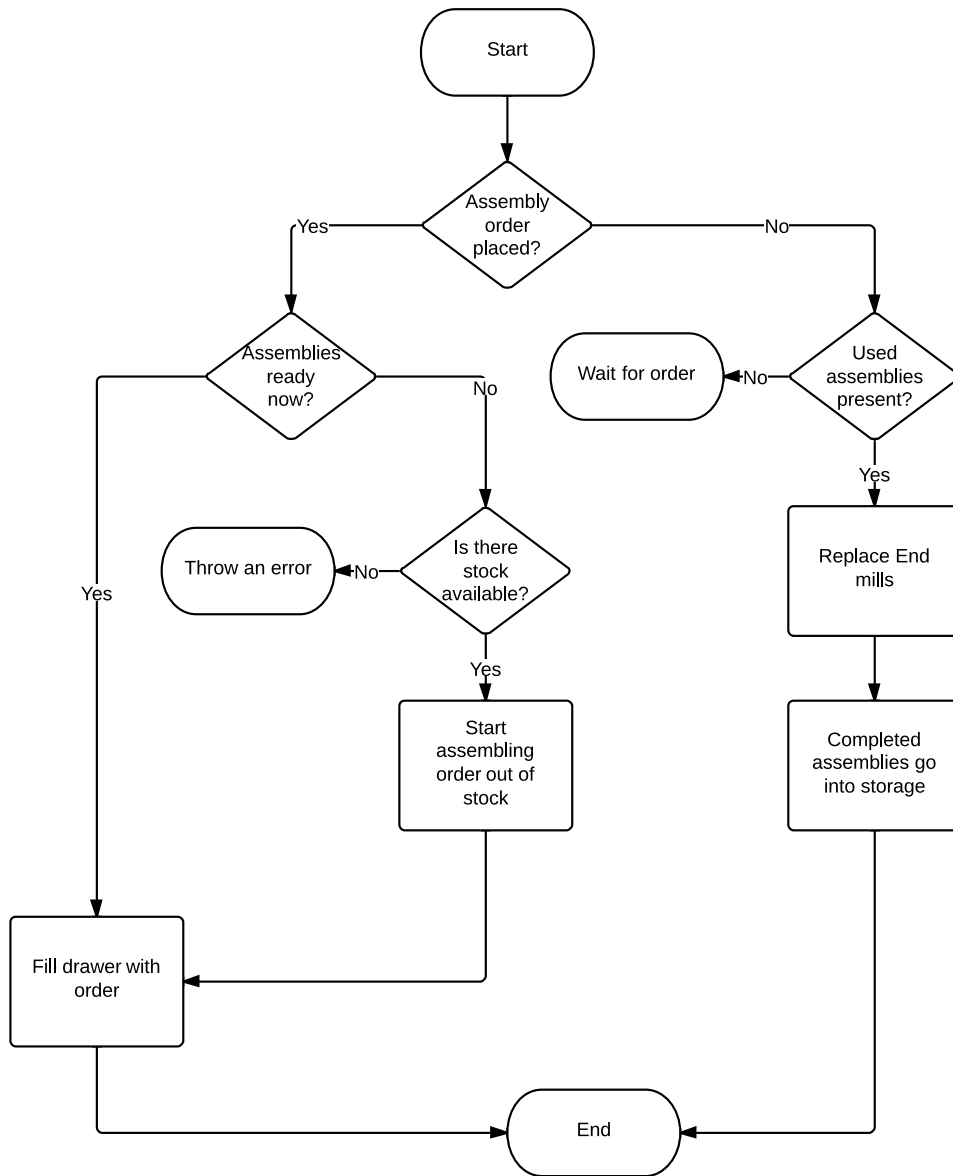


Figure 2.3.: Highest level flowchart for changing an assembly

The system will first check to see if any orders have been placed for assemblies. If order(s) have been placed the system then checks if it already has the assemblies in storage to fulfill the orders. If there are enough assemblies in storage, then the system

will populate the drawer with assemblies for the worker to pull out of the cell. If however there are not enough assemblies in storage, the system will check to see if it can create new ones with any existing stock or parts in the drawer. If it has no stock to go through (including the drawer) the system will alert the worker that their order cannot be fulfilled.

If no orders have been placed, the system will then check to see if there are any used assemblies in stock that need to have their end mills changed and will change any that exist. If there are no used assemblies in stock or any orders to fill, then the system will wait for an order to be placed or for new stock to be received.

2.3.2. Low Level

The flowchart for changing an end mill is shown in Figure 2.4.

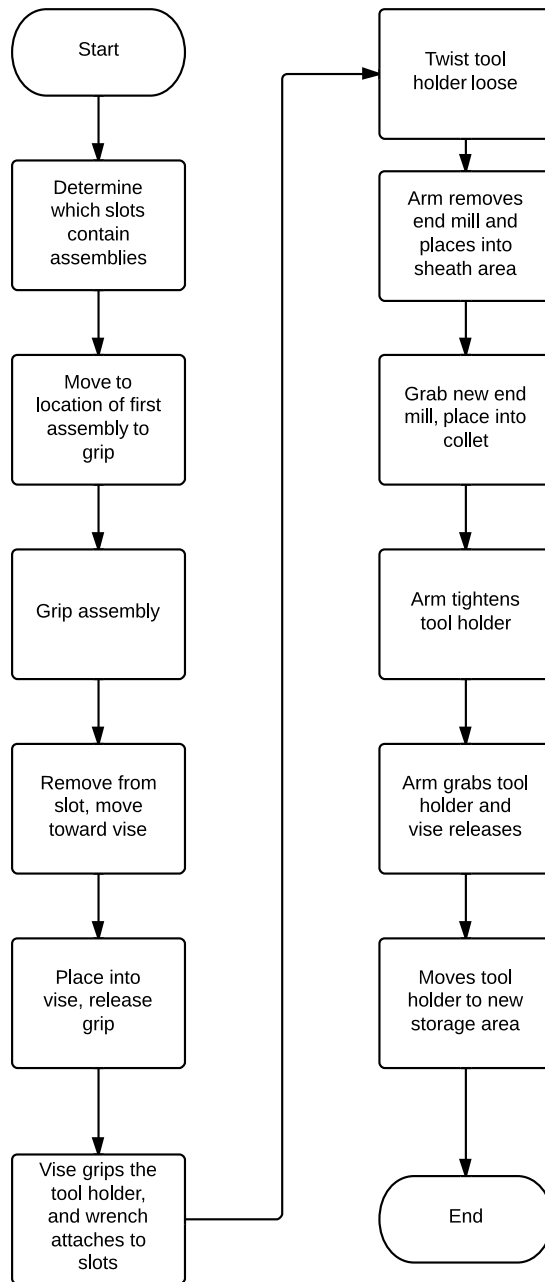


Figure 2.4.: Low level changing flowchart

The system will first sense the position of the assemblies by way of sensor or camera

I/O. The robot will then navigate the gripper to the tool holder that is most conveniently located and grip it. The robot will then lift the assembly out of its position and place it into the vice so it can be held while the end mill is removed.

The jaws will then close around the top half of the tool holder fitting into the slots that are used for securing or removing the end mill. The vice will then rotate, loosening the end mill from the tool holder. The robot will navigate the small jaws of the gripper around the end mill. The gripper jaws will close around the end mill. The robot will lift the end mill out of the tool holder and place it into the sheathing/desheathing area.

The robot will navigate the gripper around a new sheathed end mill and move it into a second gripper that will hold the bottom half of the sheath. The robot will lift the top half of the sheath off, revealing the end mill which it will place into the tool holder. The old end mill can then be placed into the sheath that had the new end mill and then repacked and placed to the side so that it can be resharpened later.

The vice will then rotate to tighten the tool holder and hold the end mill in place. The robot will move the completed assembly out of the vice and place it into a storage area where it can be retrieved later when an order is placed.

2.4. Selected System Ideas

Many designs were created to try to find viable solutions to solve the tool changing problem. The best ideas were selected and can be seen in the following subsections. The designs were selected by feasibility given the time constraints and the skill set of each team member as well as budget.

2.4.1. Assembly Receiver/Dispatcher

When a worker brings their used assemblies to the robot, they need a place to put assemblies that will not put workers in danger if the robot is in motion. A simple

drawer was designed that will be able to have the used assemblies along with new end mills within their sheaths placed into it and pushed into the cell on a rail system. Sensors can be placed within each slot to give the population of the box, or the Cognex camera can be utilized to see which positions are populated. To make sure that tool holders are always aligned a particular way so they can fit into the vise properly, raised tabs have been placed around the slots to make sure they are always oriented a certain way. A view of the drawer with a tool holder in it can be seen in Figure 2.5.

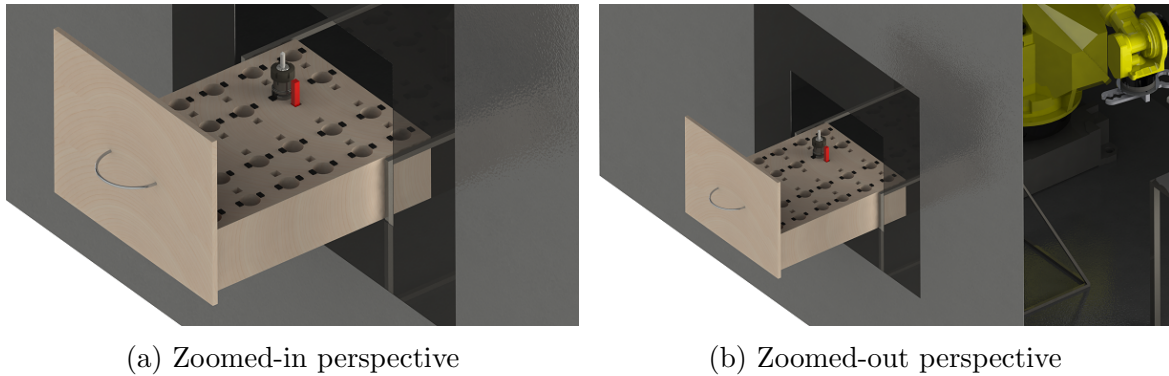


Figure 2.5.: Box to receive and return orders to workers

2.4.2. Vise

The vise is designed in two sections which each serve a specific purpose for end mill replacement.

The first part of the vise will be the inner part that spins the tool holder. It will rotate the tool holder until it is told to stop which, along with the upper part of the vise that doesn't rotate, will allow for the system to tighten or loosen an end mill from the grip of the tool holder. This can be seen in Figure 2.6 as the center of the two platters.

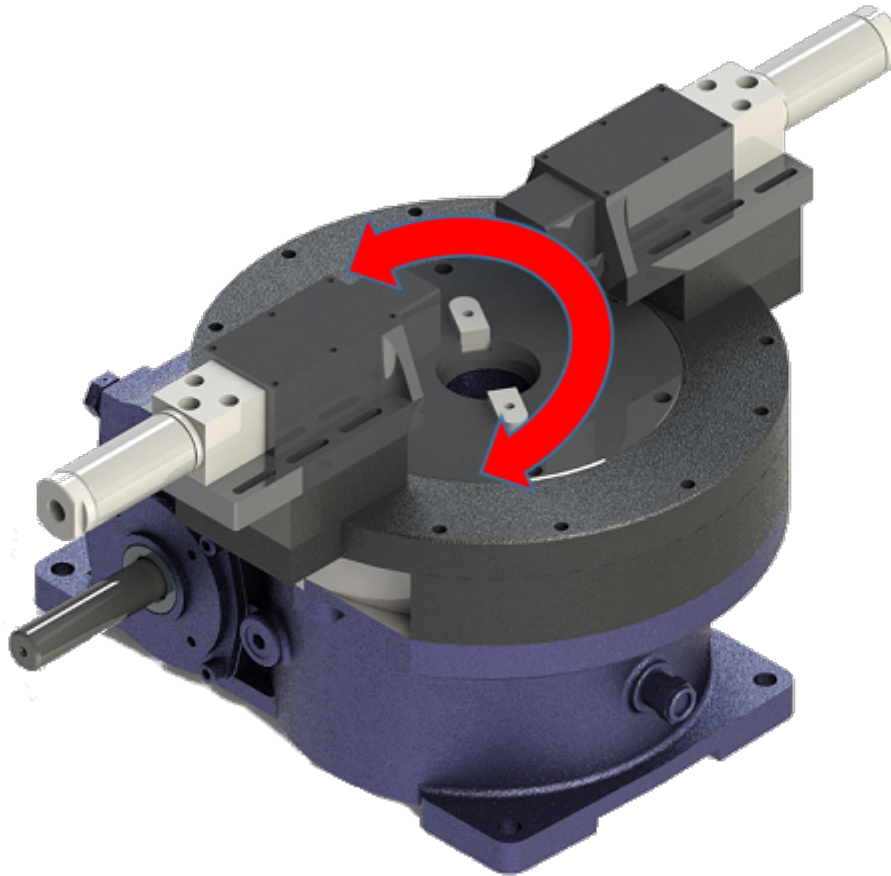


Figure 2.6.: Vise concept design

The next part of the vise will mimic the behavior of a wrench that grips the slots on the collet nut. Two jaws will be placed on either side of the tool holder and they will be controlled with a pneumatic piston. When the tool holder begins to spin, the pistons will extend and come in contact with the edges of the collet nut and ideally go into the slots immediately. More than likely it is going to miss the slots however and to address that the ends of the pistons are spring loaded to allow for them to have some play such that the springs will be compressed when not in a slot and extended when inside of a slot. Without the springs, the tool holder would likely not be able to be loosened or tightened and slots would have to be lined up for every tool holder which is not practical. An internal view of the jaws can be seen in Figure 2.7.

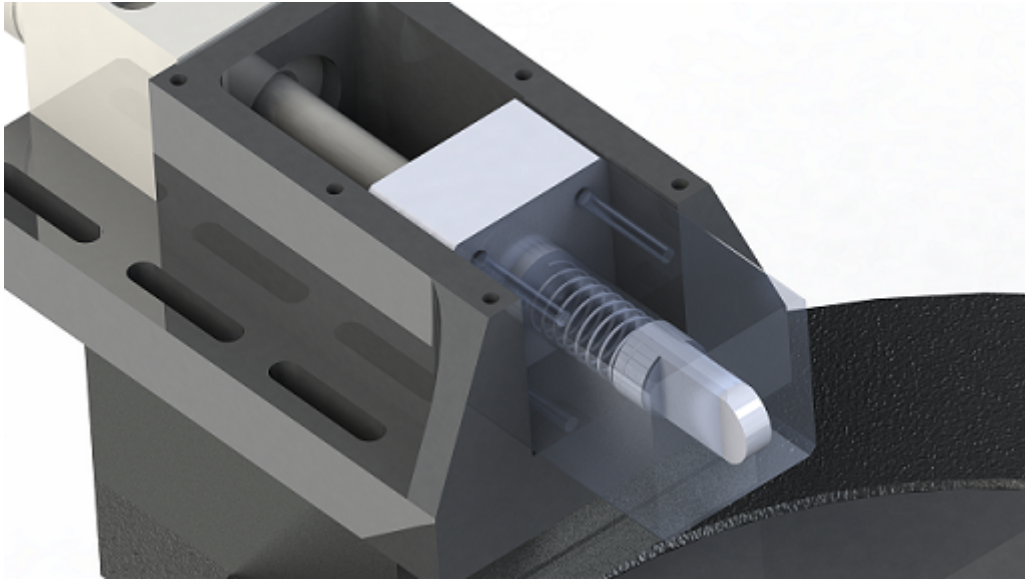


Figure 2.7.: Vise jaw internals

The jaws snapping into and gripping the collet nut of the tool holder can be seen in Figure 2.8.

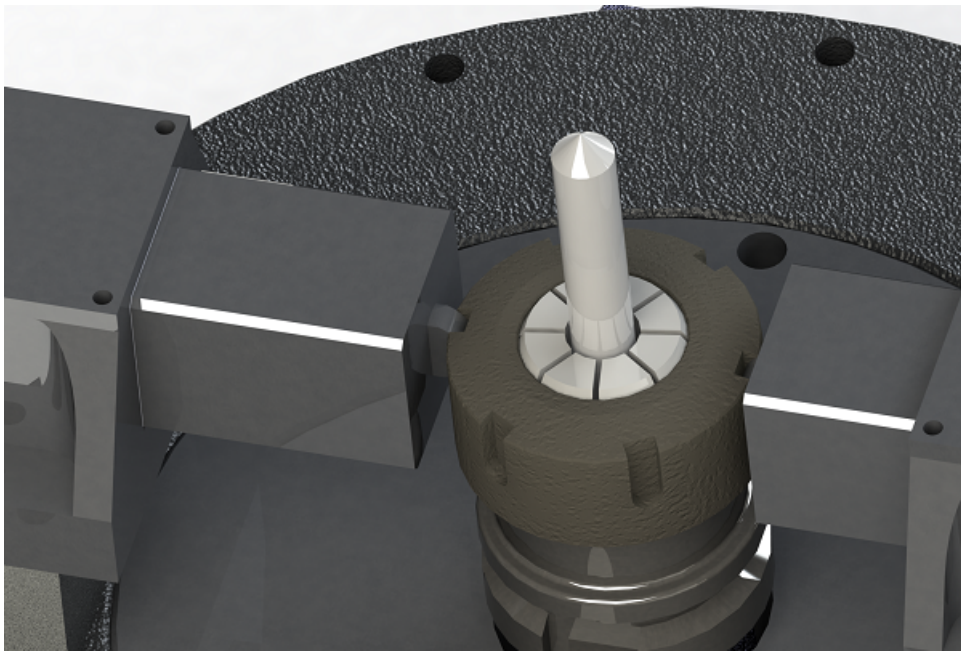


Figure 2.8.: Jaws snapping into collet nut channels

2.4.2.1. Indexing Table

The tool holders need to be rotated to enable the collet to be loosened or tightened whenever end mills are changed while in the vise. To do this, the rotating portion of the vise is connected to an indexing table that is able to spin in 45° increments in clockwise and counterclockwise directions. The indexing table can be seen below in Figure 2.9.

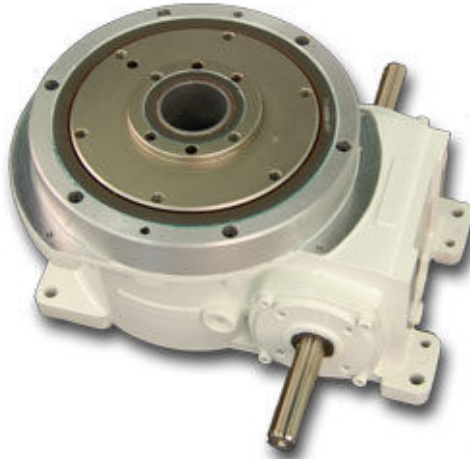


Figure 2.9.: Indexing table for spinning the assembly

To control the indexing table, a motor controller is used to drive as seen in Figure 2.10 it in conjunction with an H bridge design as seen in Figure 2.11.



Figure 2.10.: Motor controller for indexing table

This controller takes in an AC voltage from the grid and converts it into a usable 90V DC signal for use by the motor, however this particular model lacks a method to control the rotational direction of the motors. To address this problem another system needed to be added that will be able to take this 90V signal and apply it to the desired motor terminals allowing for it to be driven in either direction. The general system layout is in Figure 2.11 shown using a relay controlled by a microcontroller.

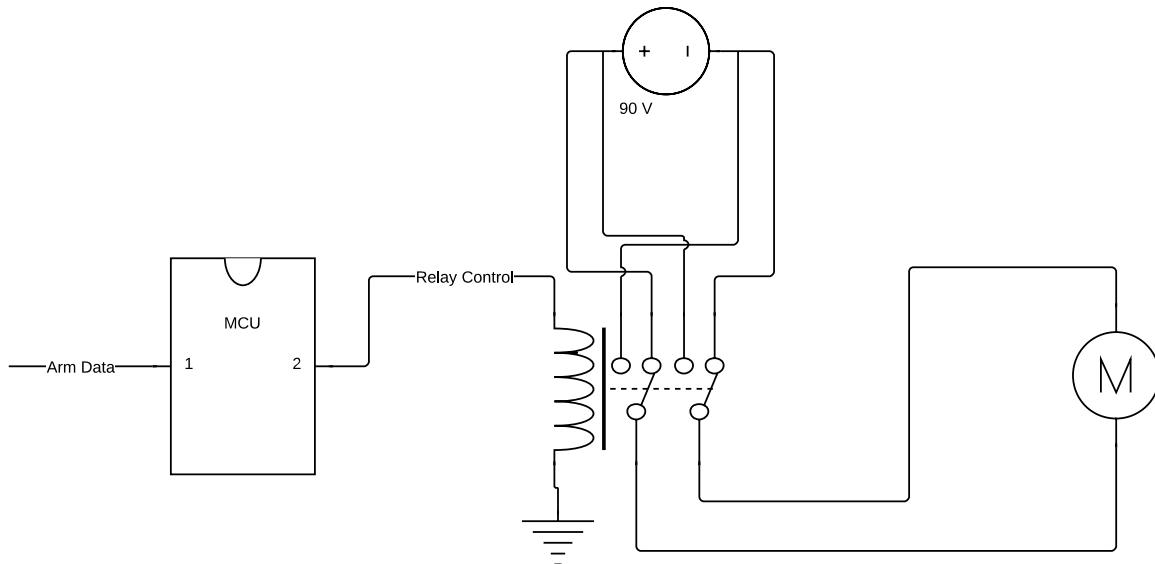


Figure 2.11.: Relay schematic for controlling motor direction

An attempt of creating this circuit was made by purchasing an H bridge that is a solid state version of a relay, utilizing MOSFETs instead of mechanical switches. The model purchased also had current sensing built in which will tell the behavior of the motors. When the motors are stalled the current is high meaning tool holder is tightened and when the current is low the collet nut is fully open. The H bridge was supposed to connect to the motor controller and with an input from a microcontroller will be able to switch the rotational movement at will as dictated by a current sensor included with the relay circuit. The unit purchased can be seen in Figure 2.12.

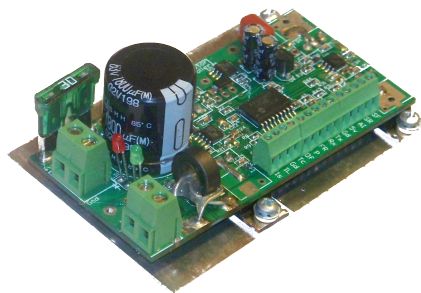


Figure 2.12.: H bridge for controlling indexer

This circuit however had a failure during initial testing when connected and destroyed

the microcontroller connected to the H bridge that it was controlling. This meant a new solution had to be found that would be able to control the direction of the indexing table as desired.

A second model of the motor controller exists that does support the reverse direction and is controlled with relays. Three relays are connected to control forward, reverse, and stop by switching a 24V signal that comes out of the arm controller. This eliminates the extra layer with the microcontroller and allows the arm to fully control the vise which is a better system, however this design lost the ability to sense current which was being used for when to stop. To address this, the vise will just be turned on for a set amount of time that was measured beforehand that tightens and loosens the tool holder.

2.4.2.2. Mounting

The entire vise needs to be in an area that the robot is able to reach so that assemblies can be placed into the vise. A simple table was constructed out of steel and the indexing table was mounted to it as shown in Figure 2.13.

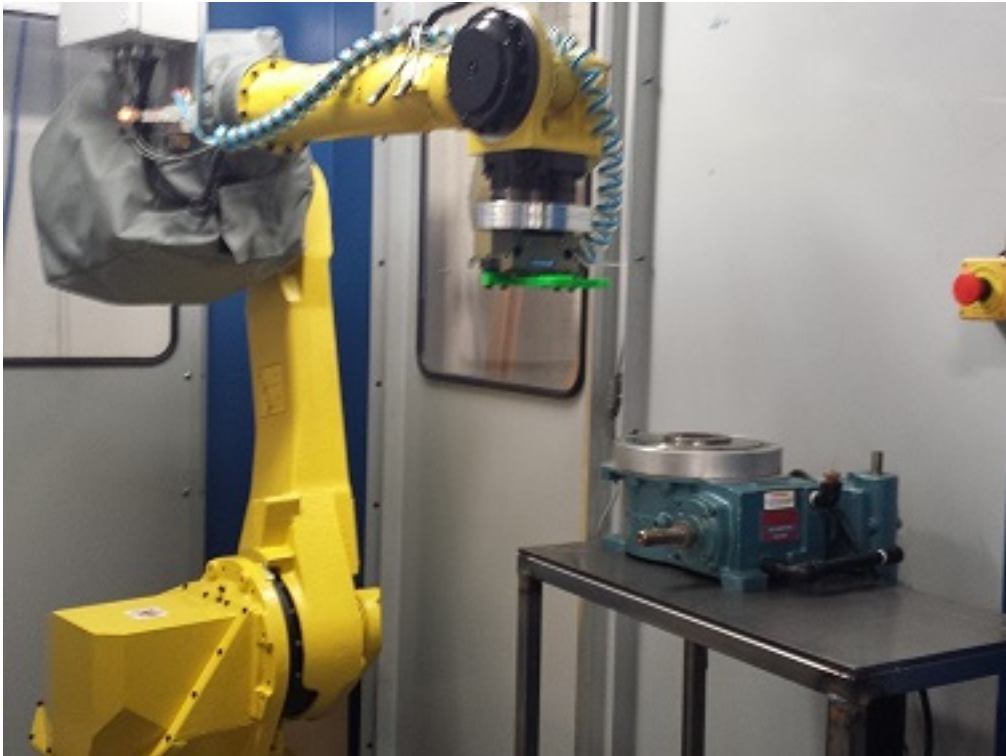


Figure 2.13.: Table for mounting vise

2.4.2.3. Analysis

A basic analysis was performed on the vise that looked at the pins in the jaws as that is the piece most likely to fail due to being so thin. An analysis similar to what was performed on the grippers was run in SolidWorks and the results can be seen in Figure 2.14.

Model name: jow-internal-v2
Study name: SimulationXpress Study
Plot type: Static displacement Displacement
Deformation scale: 2.08375

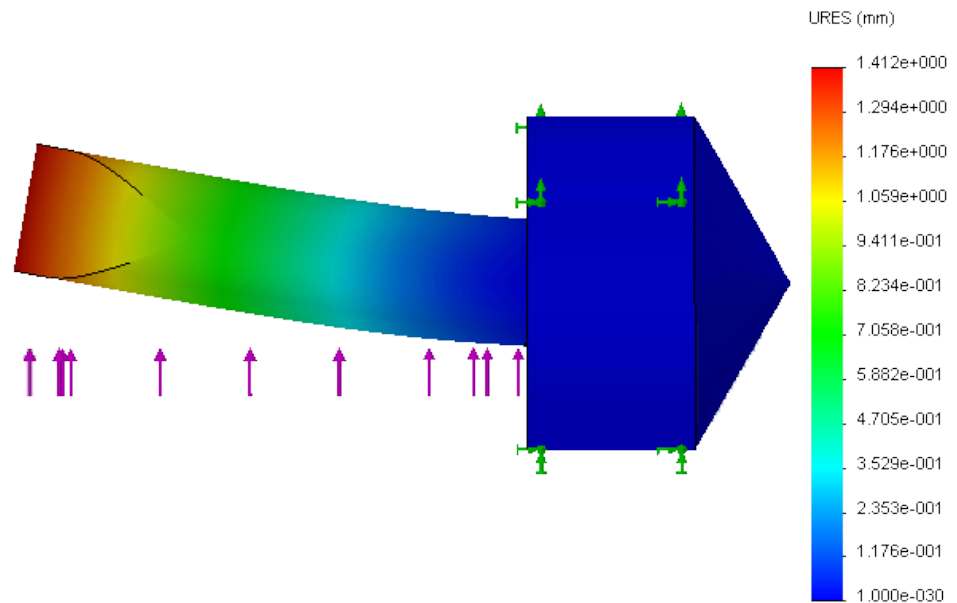


Figure 2.14.: Deformation of jaw pins

The bending of 1.4 mm was found at the tip which is not a large amount, but may be enough to begin wearing the jaw down over time and cause deformation. The pins will be bent once in both directions during the changing of every tool holder and this will eventually break the jaw pins causing them to need replacement.

To fix this in the future, the pins should be made out of a stronger metal that will not see this amount of deformation. As this is a contact point an even better reason to replace them in the future with a harder material is to stop chipping from happening when the jaws close.

2.4.3. Gripper

The gripper shown in Figure 2.15 will be 4 separate parts mounted onto a 2-jaw parallel actuator that allows for it to go over and around assemblies and end mills. As

the robot needs to manipulate sheaths, tool holders and end mills, the gripper needed to be designed to deal with all of them or be swapped out with another as needed. As there are only three parts of non-irregular shapes and sizes, it was better to keep one gripper and not overcomplicate anything. The left side of the gripper has a large circular hole that will go around the tool holder and be able to safely manipulate it as needed in and out of the vise. The right side has two holes, one that will be able to grab small circular end mills and slightly larger square sheaths. All holes will have a piece of soft rubber around the inside that will be able to deform when compressed so it can better grip the tool holder and conform to its shape and prevent it from slipping on the smooth metallic surfaces.

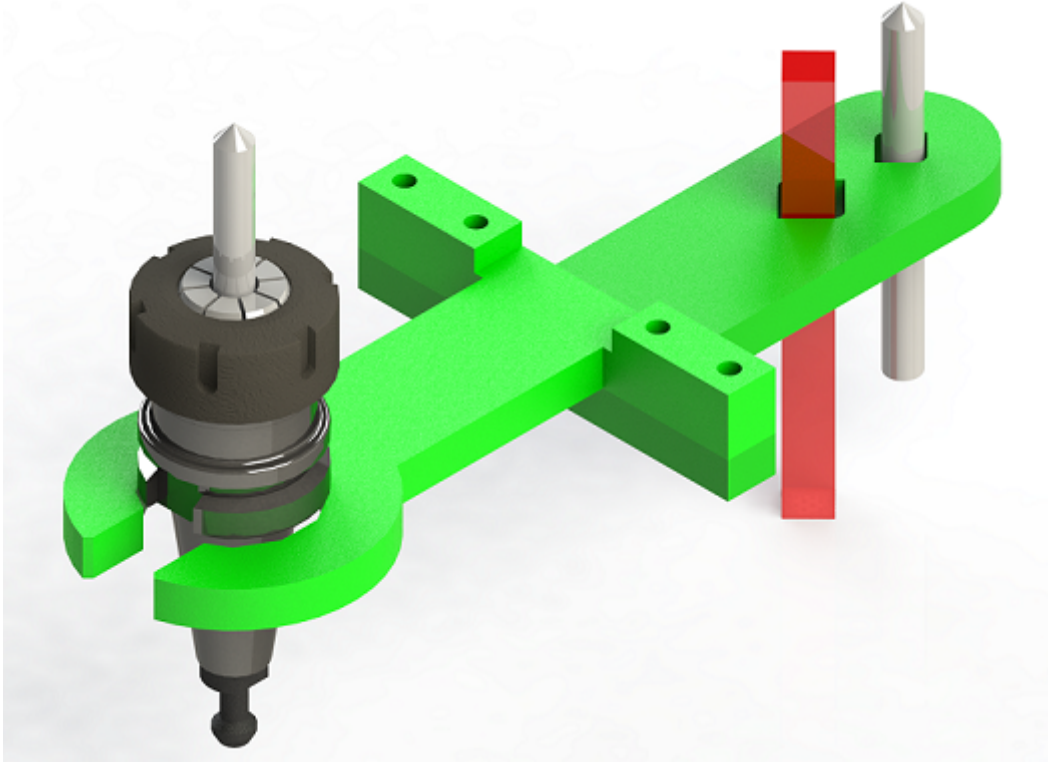


Figure 2.15.: Gripper

2.4.3.1. Gripper Analysis

To test out how well the design would work, a simulation was run in SolidWorks 2013 on a simplified gripper made out of aluminum 6061 as shown in Figure 2.16. Because only the tool holder has sufficient weight to cause any type of deformation, only the right half of the gripper was loaded with a force of 100 Newtons while the left half was left unloaded. This force is equivalent to about 22.4 pounds of force, much more than the assemblies weigh (about 10 pounds) which gives a safety factor of 2. This force is applied to the gray circle which represents an assembly where the force will be concentrated from the weight once grabbed.

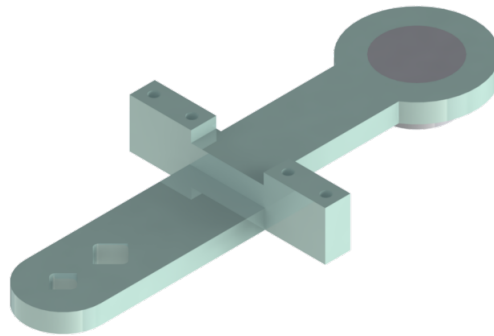


Figure 2.16.: Simplified gripper for simulation

The first analysis is the deformation in Figure 2.17. The scale is in millimeters and the color scale shows how much each part of the gripper deforms.

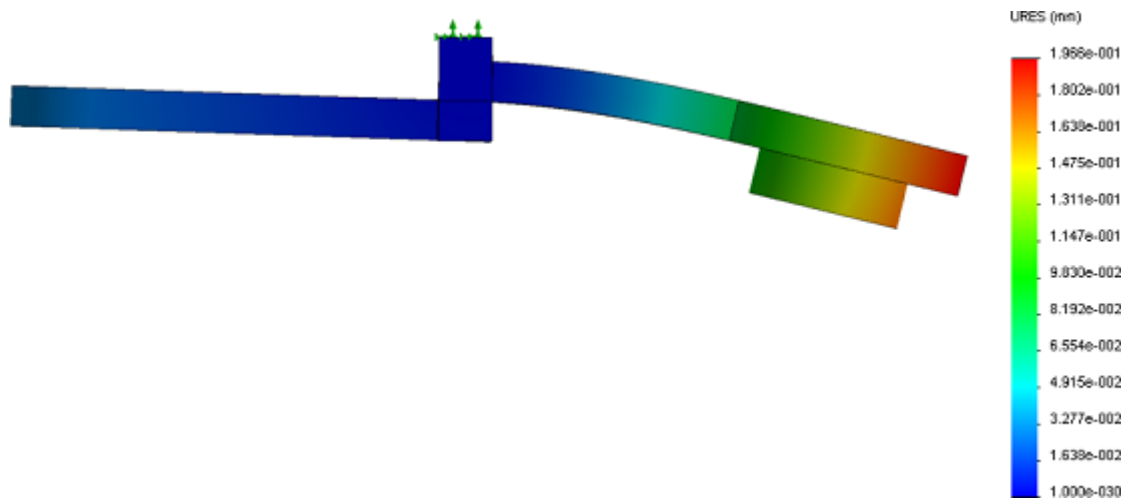


Figure 2.17.: Gripper deformation with 100N of force

Looking only at the image, it appears as if the gripper is bending by about 25 mm or more. This is because SolidWorks is scaling the deformation by about 145 times so that the effect is visible; looking at the scale to the right it shows that it is only at most 0.196 mm at the tip which is next to nothing.

If the bending of the gripper is treated as a triangle, the angle that the gripper is bending can be found which allows for the displacement of the end of the tool holder to be found. The angle was found to be 0.072° and the effect this has on the tool holder can be seen in Figure 2.18 using a tool holder with a length 305 mm where the black line is a tool holder deforming. An equation was also found to give the displacement for a tool holder of any length with a 100 N force which was found to be: $y = 793.650167x + 0$. Therefore, the effect of a 100 N force on the gripper is negligible.

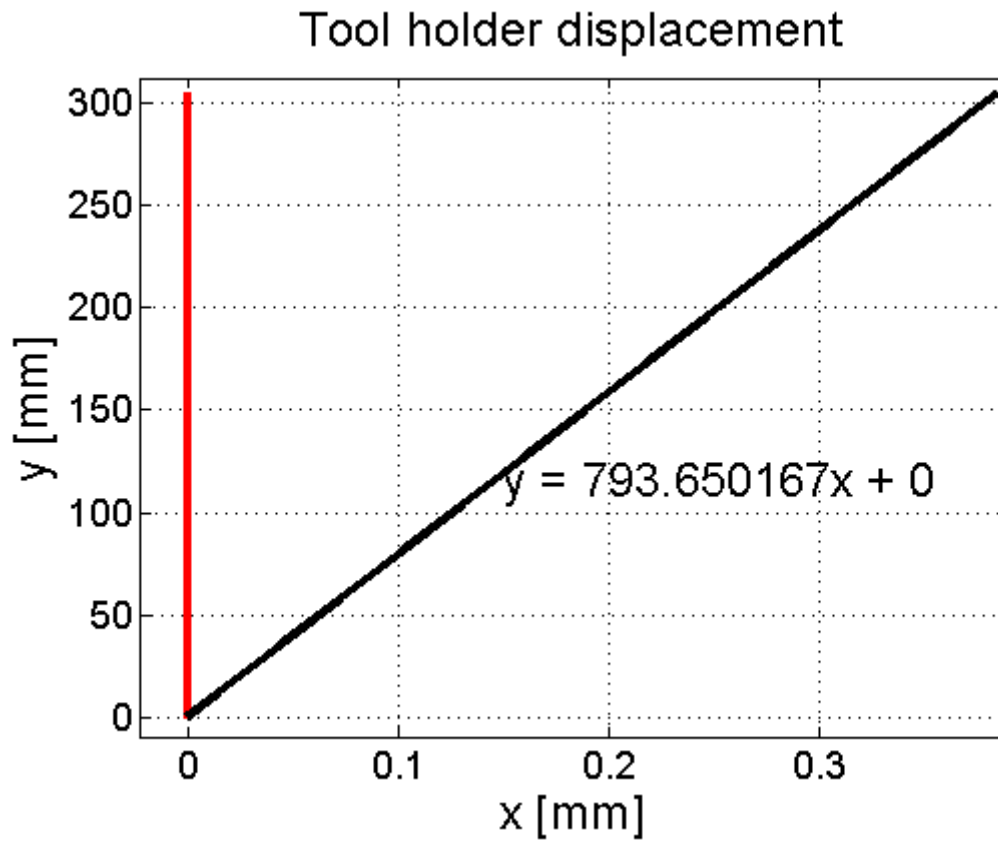


Figure 2.18.: Tool holder displacement with 100N of force

Initially, as a prototype to make sure that all measured dimensions that were believed to be needed were correct, a gripper was 3D printed out of polylactic acid (PLA). In SolidWorks, a simulation was run on a similar material (acrylonitrile butadiene styrene - ABS) which showed that the plastic gripper would not be able to be used with this force as shown in Figure 2.19.

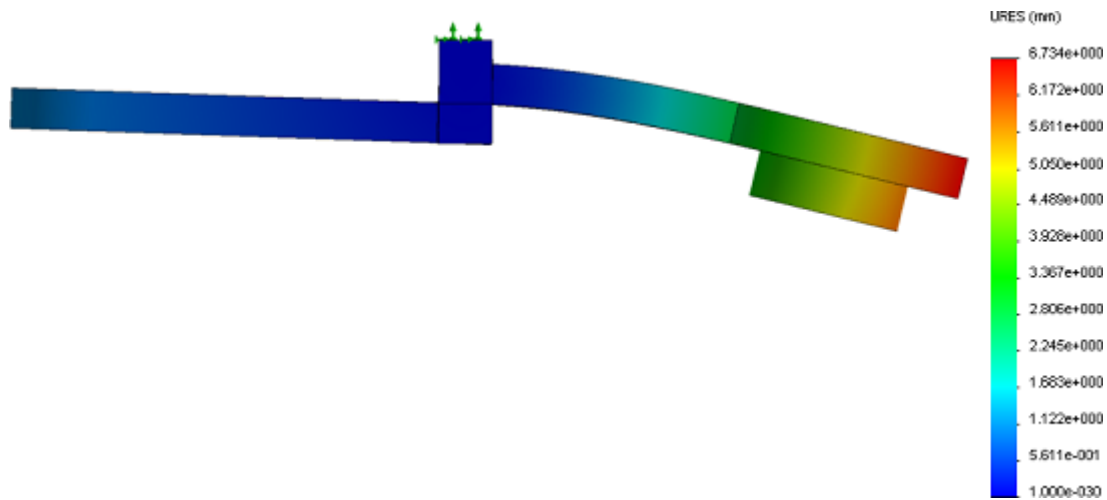


Figure 2.19.: Gripper deformation with 100N of force

This time there is 6.734 mm of deflection at the tip of the gripper. This would not be that great of a material to use for a gripper as it is just over 32 times worse than aluminum was. Equating the displacement of the actual tool holders end points was done once again and this time found to be almost 13 mm in the horizontal which is unacceptable. This would make it harder to place the assemblies into either the vise or storage position, and over time may even wear down as plastic does not have the durability of aluminum.

The analysis was done again with a force matching that of the actual tool holder and only a 5.5 mm displacement was found at the tip of the tool holder. This was an acceptable amount when tested and the PLA was used as it was already made and would work reliably throughout the project.

Next a stress analysis was done for aluminum in Figure 2.20.

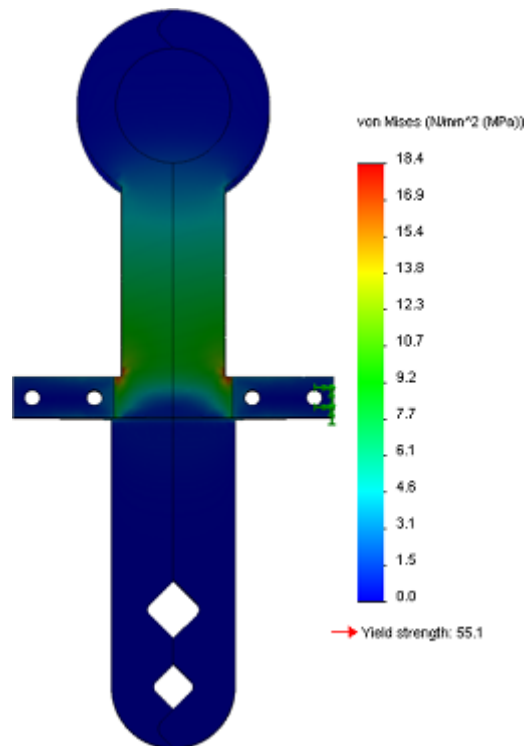


Figure 2.20.: Gripper deformation with 100N of force

Unsurprisingly, all of the stress is located back by the bolts, mainly at the inner 90 degree corner of the gripper. The stress exhibited is very low and is nothing that needs to be concerned about in this application.

2.5. Models

To help with the design, some models were created to represent the arm to make planning easier. They were used to make sure that the arm would be able to perform the actions desired and to help spot potential problems.

2.5.1. Arm: MATLAB

A MATLAB model was created that used forward kinematics to display the orientation of the arm with a given DH table. This model was not very useful in the end but creating

it did help understand the movements capable of the arm at the start of the project. An example output of the MATLAB script created can be seen in Figure 2.21 that shows the arm in some orientation and the MATLAB code can be found in the Appendices.

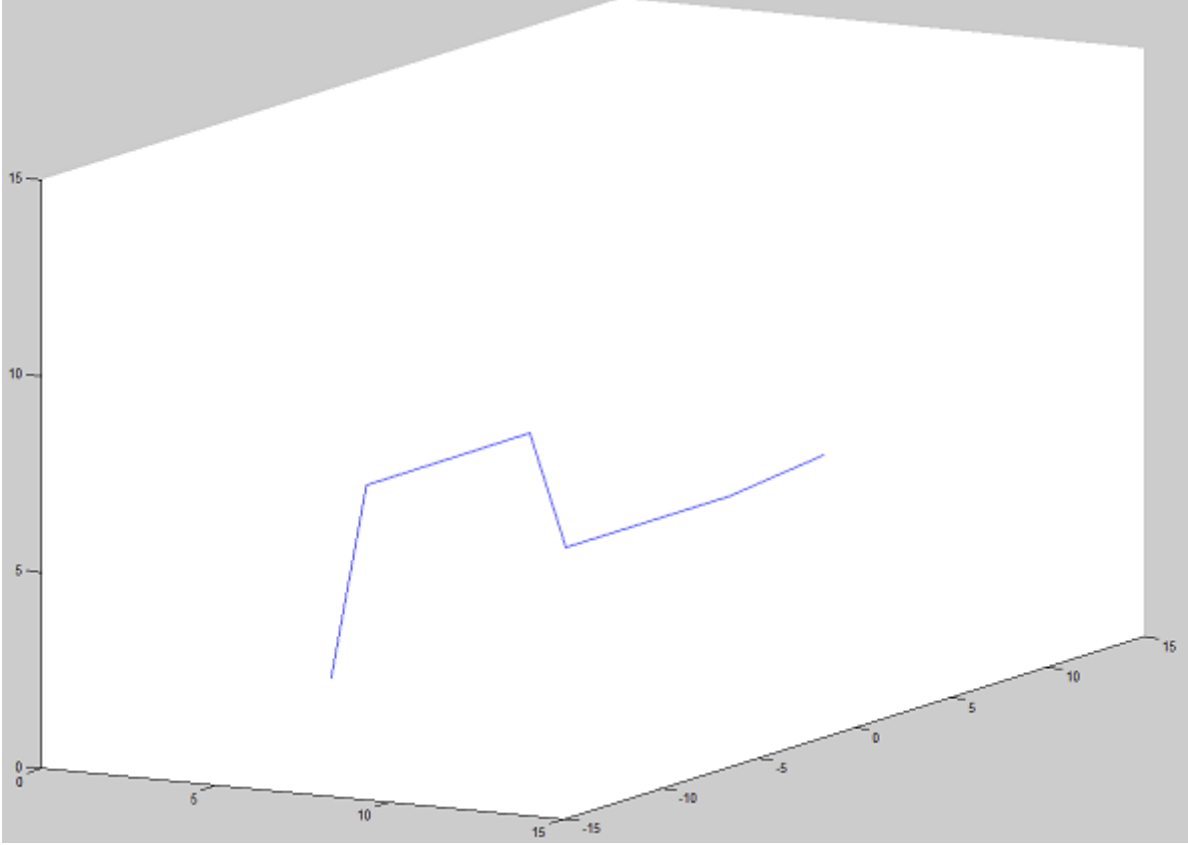


Figure 2.21.: FANUC MATLAB model

2.5.2. Arm: SolidWorks

A much more useful SolidWorks model of the FANUC arm was obtained from online.¹ This model allowed for a much better visualization of the arm and allowed for all other parts created in SolidWorks to be put in the same workspace as this arm. The modeled arm is shown in Figure 2.22

¹<https://grabcad.com/library/fanuc-m-710ic70>



Figure 2.22.: FANUC SolidWorks model

2.5.3. Work Cell Layout: SolidWorks

Using the SolidWorks model of the arm and all systems developed, a layout for the work cell was created as shown in Figure 2.23.

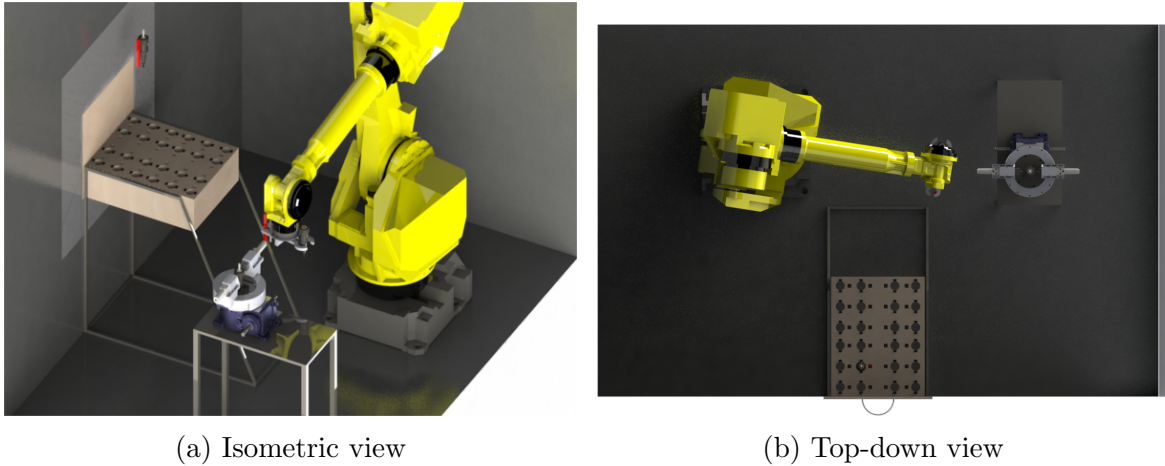


Figure 2.23.: Planned work cell layout

The drawer will go where an access door is already located on the work cell. This is an easy access point that is within the workspace of the robot so it can be easily reached by the arm. This is the only logical place to put the drawer at the moment as the work cell is not allowed to be significantly modified. The access door is able to easily be removed, and the existing safety features on the door can be re-used on the drawer as desired.

The vise will go directly in front of the arm and cables will run from the arm to the vise systems that allows for the controlling of the vise with pneumatics and relays. This uses up almost all of the available space in the work cell and if more is needed in the future, it would be required to either expand the work cell or purchase a new one.

2.6. Implementation and Testing

Each section was constructed one at a time to make sure there was no problems that would affect another system, calling for modifications that might be impossible after construction. Some of the original plans were deviated from during the construction process as needed when problems arose and were addressed as appropriate.

2.6.1. Drawer

The drawer was constructed out of wood and attached to the side of the cell where an access door used to be. The SolidWorks plan was deviated from due to time constraints and having some extra 80/20 on hand that was able to serve the same purpose as the support frame. This still allowed for workers to safely be able to remove and insert assemblies into the cell without being in danger of any moving parts. Figure 2.24 shows the drawer installed in the cell with a tool holder loaded into it.



Figure 2.24.: Completed drawer assembly

2.6.1.1. Performance

The drawer is able to function as desired and can safely insert and remove tool holders and end mills from the work cell. One thing that needs to be done is that the size of the drawer needs to be increased from the small test size that was created to make sure it would work. The drawer served its purpose for testing and therefore was not scaled

up as it was not a critical component that affected anything.

2.6.2. Vise

The vise was machined out of 6061 aluminum and was attached to the indexing table. The entirety of the vise was then attached to the previously constructed table as shown in Figure 2.25

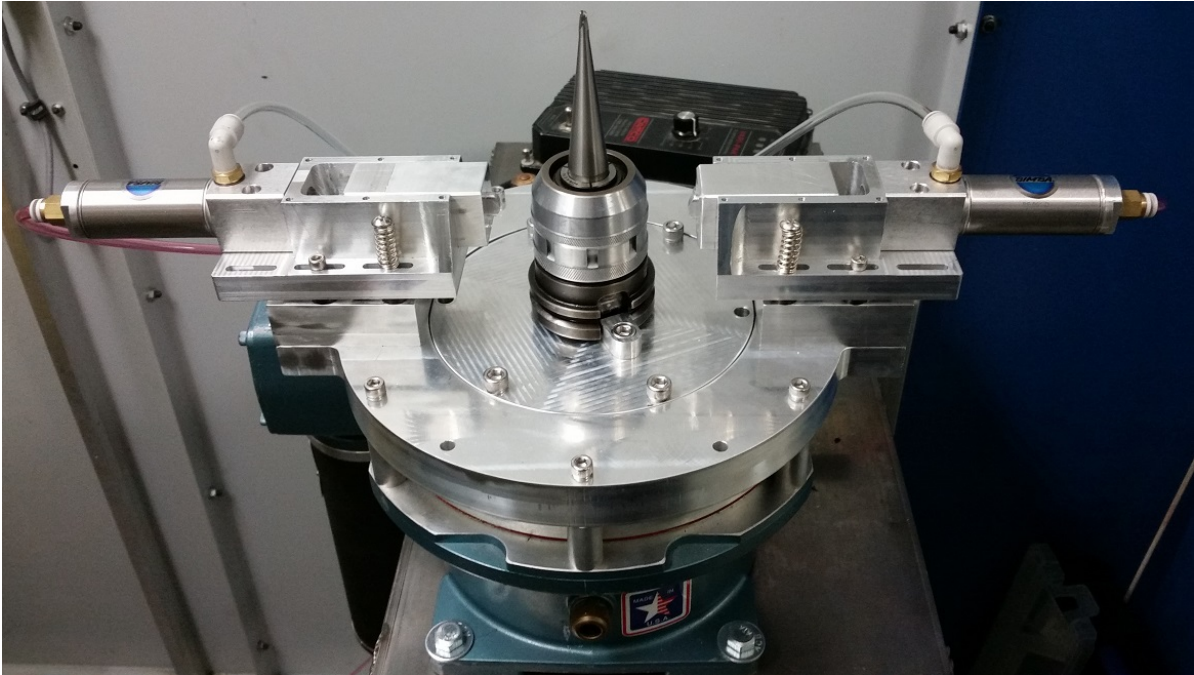
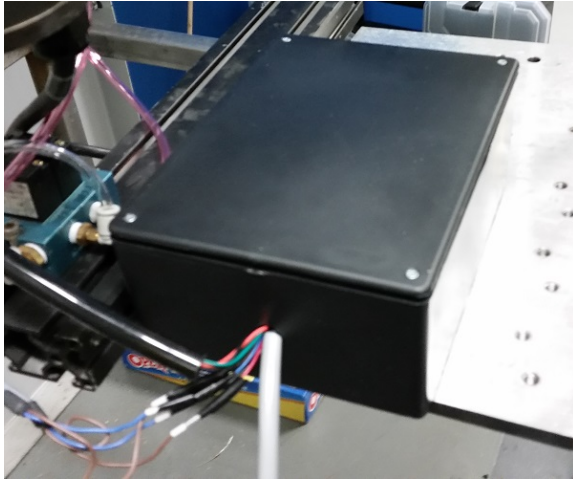
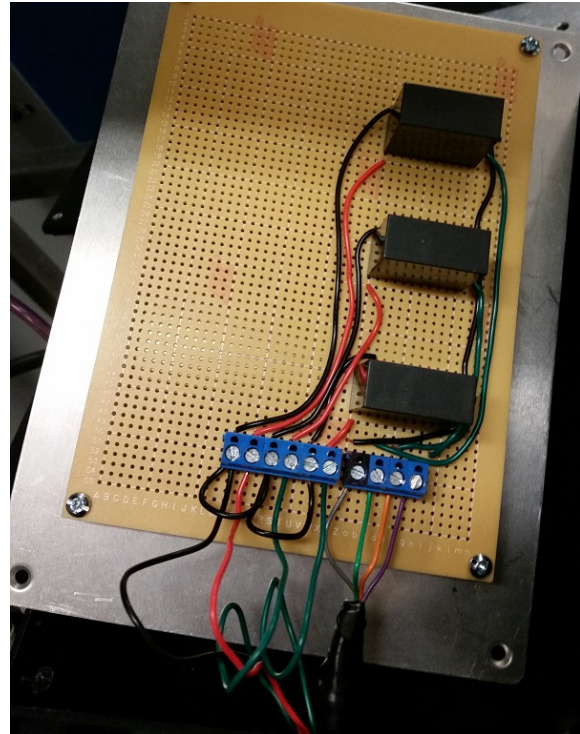


Figure 2.25.: Constructed vise

The relays that control the direction were put inside of a plastic project box as shown in Figure 2.26.



(a) Box that contains the relays



(b) Simple relay circuit for controlling the vise

Figure 2.26.: Constructed relays

2.6.2.1. Performance

The vise is able to perform the task of loosening and tightening the collet nut, however there are some issues with the overall design. When adjusted, the collet nut moves up and down approximately $\frac{1}{8}$ of an inch. This that was never taken into account during the planning phase, and means that there is bending in the jaws. An attempt was made to remedy this in the short term by placing rubber washers and springs underneath the jaws which allow it to move in the vertical as shown in Figure 2.27. This modification however allows the jaws to also move in the horizontal slightly which is undesirable.

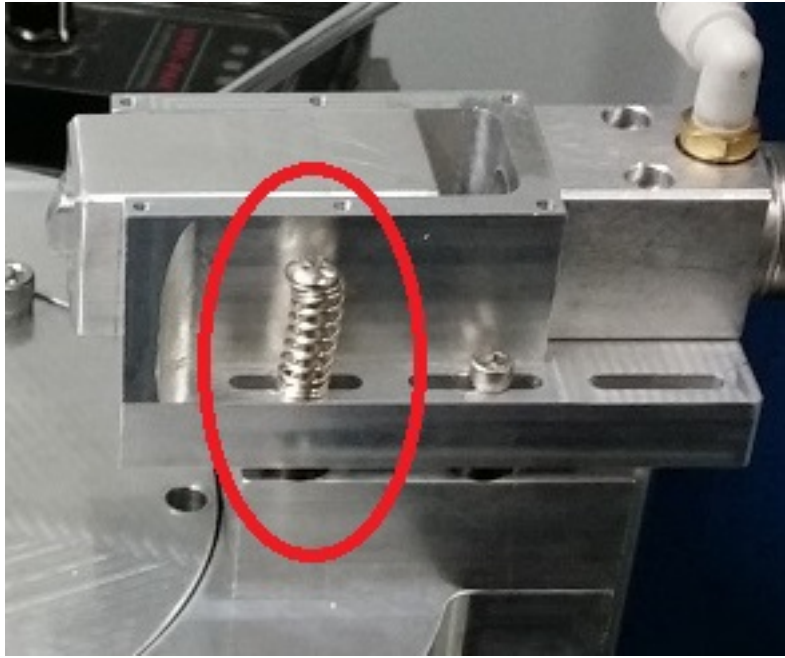


Figure 2.27.: Temporary fix for the vise jaws

Another problem is the dimensions of the tool holder were not properly taken and there is a small lip on the bottom that was not taken into account. Because of this the tool holder does not properly sit in the vise causing it to be able to wobble around when placed into the hole.

A possible solution to this is to make a way for the jaws of the vise to "float" allowing for them to freely rise and fall with the collet nut. This would be a more complicated solution however it is much more robust and allows for using different tool holders that each rise a different amount.

The last major problem with the vise is because there is no current sensing, the collet nuts were tightened by rotating for a fixed time. If the collet nut were to be over-tightened then it has the potential to bend the vise jaws, effectively destroying the vise. The only way to stop this from happening is the stop switch on the motor controller which immediately stops any rotation.

2.6.3. Gripper

The gripper was made out of PLA using a MakerBot 2 printer as shown in Figure 2.28.



Figure 2.28.: Grippers made of PLA

As already discussed in the analysis of the gripper, the PLA did not meet the initial hopes. Figure 2.29 shows the machined aluminum grippers for the tool holders that replaced the 3D printed grippers as well as wooden tooling grippers.

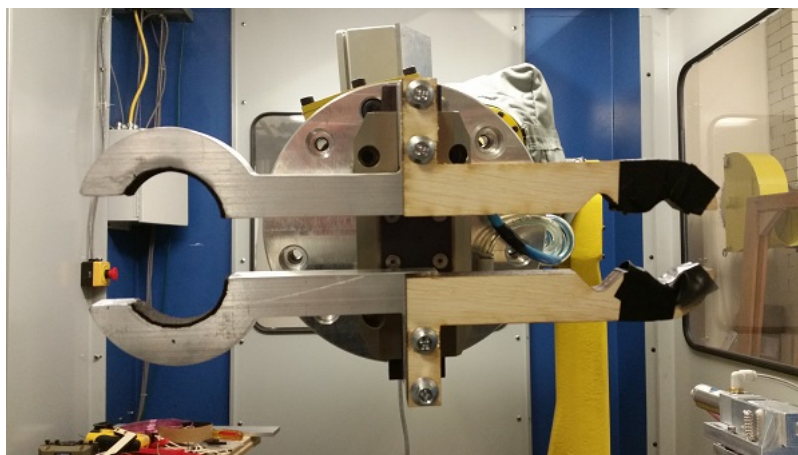


Figure 2.29.: Wooden and aluminum grippers

It was only possible to make the tool holder gripper out of aluminum due to problems with the CNC machines in WPI's Washburn Labs near the end of the project when the

grippers were being machined. The tool holder grippers were instead laser cut out of wood to eliminate the distortions in the 3D printed ones.

2.6.3.1. Performance

When the gripper was made out of PLA it was able to serve its purpose fine for a short while. However during the printing process the gripper deformed due to the limitations of the printer. Parts of the gripper became twisted which meant that objects became misaligned when held by the arm. Figure 2.30 shows the sheath and end mill grippers deformed from the print process.

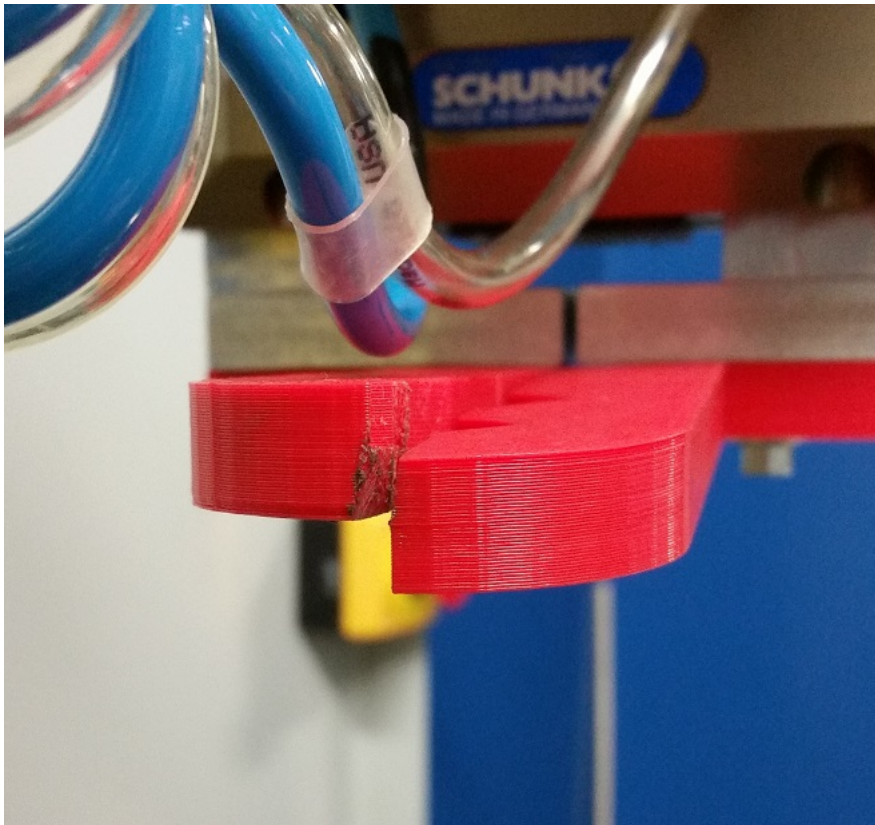


Figure 2.30.: Severe deformation of the grippers

Another problem not taken into account is when the gripper encountered any resistance or accidentally bumped into something, then the gripper would bend severely. Due

to the bending, the tool holder gripper was then made out of aluminum and the tooling grippers out of wood to make sure they weren't broken on accident during testing.

The aluminum tool holder gripper worked much better and did not have any deformations. It was able to successfully pick up tool holders and move them around as desired and putting extra force on them did not bend them. It was found that the tool holders were able to rotate so rubber was added to the gripper as was visible earlier in Figure 2.29.

The wooden grippers for the tooling did not work as well as the aluminum ones for the tool holders. Rubber was not able to be fixed to them as easily and they snapped when excessive force was accidentally applied to them. The laser cutter is not as accurate as the mills meaning that tolerances were poor and end mills were not gripped well. Electrical tape was wrapped around the gripper to try to fix with the holding issue as was seen in Figure 2.29 but it did not help much.

One other issue that was not considered that came up was that if there is higher than normal friction between the collet and end mill, the collet can be pulled out of the tool holder. This is a problem associated with the wooden grippers sagging slightly with weight at their ends and the deformation of the wrapped electrical tape which sometimes pulled the end mill out at an angle. This problem can be addressed in the future by making a simple mechanism that is able to stop the collet from rising as during testing the collet was easily pushed down and removed.

Another problem along the lines of the end mill becoming crooked is re-insertion. Whenever an end mill was picked up, it was often crooked and at a different angle than before, making inserting it into the collet impossible. This is something that may be able to be solved with the aluminum design, however it was not able to be tested due to all WPI's CNC machines being broken at the time.

2.6.4. Work Cell Layout

The final cell layout matches the initial plan and the implemented components can be seen in Figure 2.31.



Figure 2.31.: Actual cell layout

The arm was able to freely move between the two areas as desired with no issues whatsoever. The only change that would be made is that the table should be bolted to the floor as sometimes the table was moved accidentally which required some arm movement points to be re-created.

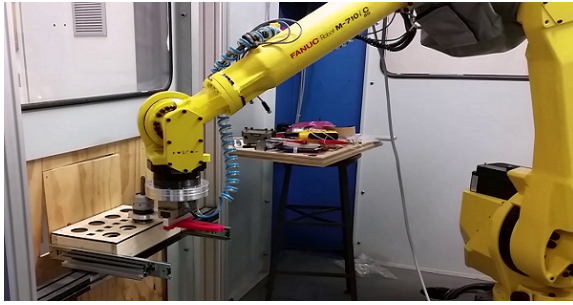
2.6.5. Programming and Putting it Together

The programming of the robot was relatively simple as multiple systems were never created which eliminated a lot of obstacles. Everything was programmed from the teach pendant shown in Figure 2.32 which allowed for moving the arm as desired and saving the motions as desired.

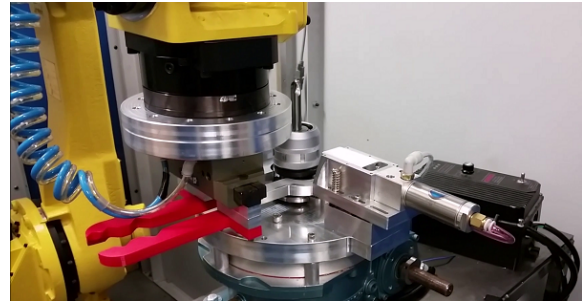


Figure 2.32.: Teach pendant

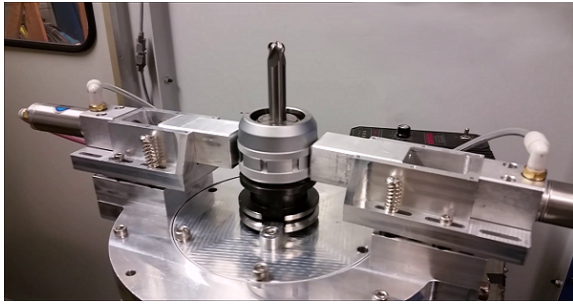
The functions of all sub-components were made into functions which were called when needed such as "open gripper" or "tighten collet" to make the code more readable and modular. Everything was then put together and a video of everything can be seen at <https://www.youtube.com/watch?v=74kvBIuBhkc&feature=youtu.be> and snapshots can be seen in Figure 2.33.



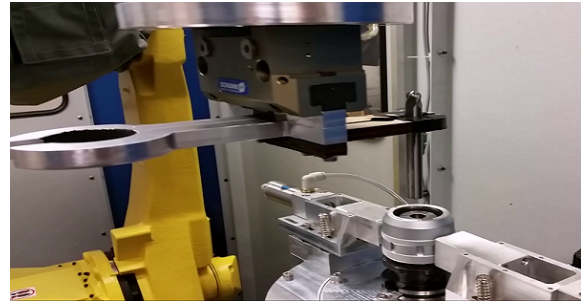
(a) Obtaining and returning the tool holder



(b) Placing and removing from the vise



(c) Tightening and loosening the collet nut



(d) Removing and replacing the end mill

Figure 2.33.: Complete changing process

2.7. Safety Considerations

The biggest concern with the changing process is a worker being able to get any part of their body within the work cell while it is in motion. This can happen in multiple spots, the most dangerous being the doors of the work cell as a person could fully enter the cell during operation.

When an employee wishes to enter the work cell they need to unlock one of the doors and leave their key on the outside, locked in. The doors are monitored by an interlock system that senses when a door is open and can stop the arm from making any movements or make it run at a reduced speed. If proper safety procedures are taken then there is no risk unless there is a malfunction with the equipment or someone removes another workers key and reactivates the arm.

The drawer currently has no safety systems implemented which can put workers at

risk. This created two hazards, the first is that if an employee pulled out the drawer they could stick in a body part and possibly be injured if the arm hits them. This could be addressed by tying the drawer into the interlock system so that if it is ever opened when the robot is operational, then it will react in the same way as a door being open.

The next problem with the drawer is that if the arm is holding something in the drawer there is nothing locking the drawer in place preventing a worker from attempting to pull it out of the cell. This can cause components to break which is an issue that will need to be fixed in the future with some kind of locking system to prevent anyone from accessing the drawer when the arm is using it.

Another large problem is that the vise system is not tied into any of the work cells safety systems. If a problem occurs while the vise is rotating such as over-torquing as discussed earlier, then the only way to stop the vise is to enter the work cell and flip the switch on the controller. Pressing the emergency stop buttons or breaking the fence on the work cell doors will currently not stop the vise from operating.

2.8. Aesthetics

The the vise and the tool holder gripper look aesthetically pleasing being made out of aluminum and look appropriate in an industrial setting. The drawer needs to be remade to look better which would involve making it out of metal instead of wood as well as doing a better mounting job. This also applies to the end mill and sheath gripper which was made out of wood instead of aluminum as it was supposed to, milling it out of aluminum will make it look more professional as it was meant to be.

2.9. Social and Ethical Impact

At the start of the project, GE explained that changing end mills is a job task that is rotated around within employees already within the company. This can eliminate a dull and slightly strenuous task from their employees workload and give them work that is better suited to their talents. With further development this can apply to the many other tool holder types GE has to eliminate those tasks as well and employees would no longer need to spend their days doing this job.

3. Tool Tracking

3.1. Background

For tracking assemblies, a solution was needed that could stand-up to the harsh environments of the CNC machines. Due to the environment, the only real solution was radio-frequency identification (RFID) where tags can be attached to the tool holders and scanned.

The other main solution considered briefly was an image recognition or QR-codes that would be scanned in by a camera. This however brought in multiple problems with it, the biggest being that oils and metal bits that can contaminate the image making it harder to read. Another problem is that there is no good place to put it on the tool holder that would be easy to read and all tool holders are off-the-shelf parts that would then need to go through a custom branding process to add an image. The final problem is that orientation and lighting can be tricky and make the image hard to read causing many false or failed readings.

Each tool holder will therefore have its own RFID tag which will stay with the tool holder throughout its lifetime at the company.

Due to budgetary concerns, GE indicated that they were okay with consumer grade electronics being used for RFID tracking as a proof of concept. They indicated that they would like to be able to track the following parameters:

- Location
 - Being changed
 - Storage
 - On the shop floor (in transit)
 - Carrousel, waiting to be used
 - Carrousel, ready to be removed
- Total time used
- Length of tool installed
- Job history

As the potential number of tool holders and the pertaining amount of information that is to be tracked is large, a database will be used to keep track of all data. GE has put no particular restrictions or preferences on what type of database should be used nor how to implement it.

3.2. Preliminary Planning

To track all of the positions, multiple RFID readers will be needed throughout the plant that will allow for as much as possible to be tracked without being redundant. Figure 3.1 shows how readers will be dispersed within the facility to be able to meet all the goals.

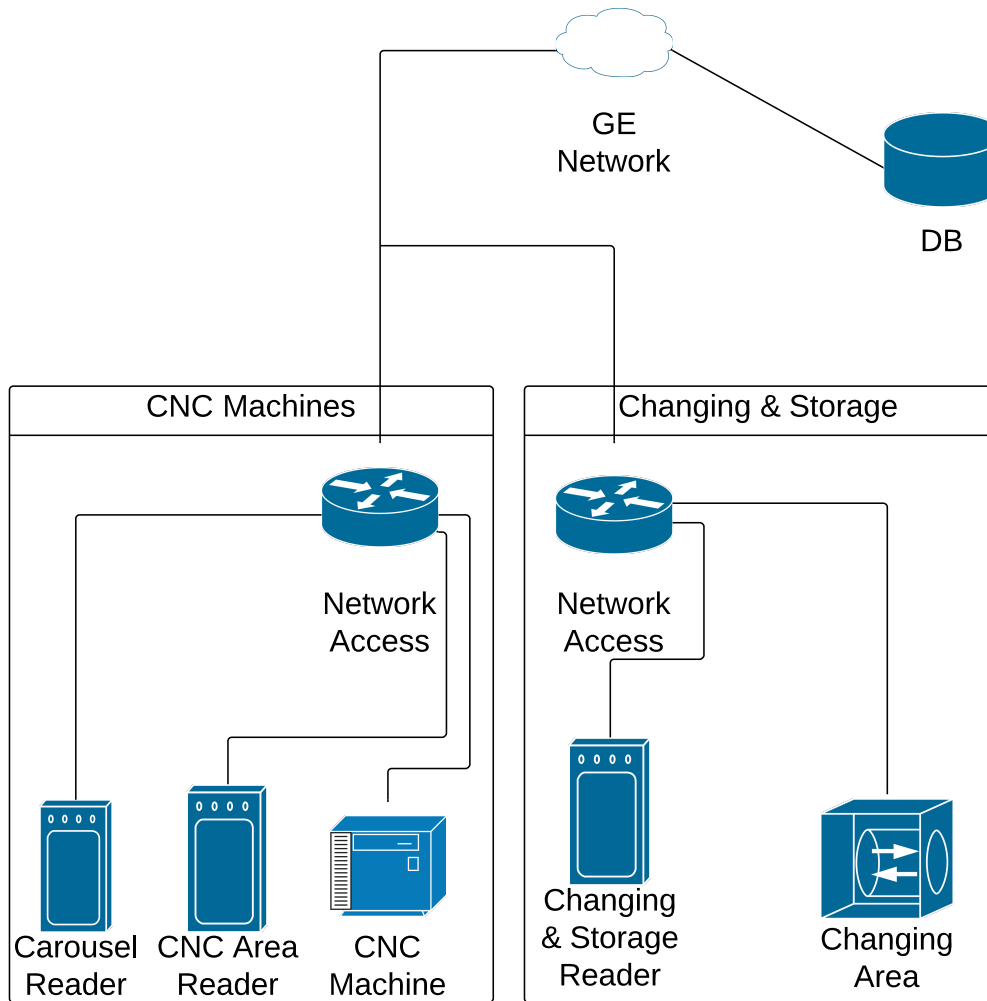


Figure 3.1.: Preliminary RFID reader layout

Each reader will be tied into the network by some communication method so that communications with the database can be done to retrieve and manipulate data as needed.

3.3. Flowcharts

To track a tool holder at each of the possible locations, different processes will be needed for entering and exiting each respective area. Flowcharts were created once all the areas were determined to help find potential problems and prepare for hardware purchases as well as coding later. One thing to note is that at every stage the tags are checked to see if they are in the database or not. This is a check to make sure workers have not put a tool holder anywhere in the line that it doesn't belong on accident.

3.3.1. Changing Cell

The flowchart for the process of taking a used assembly into the work cell and changing end mills can be seen in Figure 3.2.

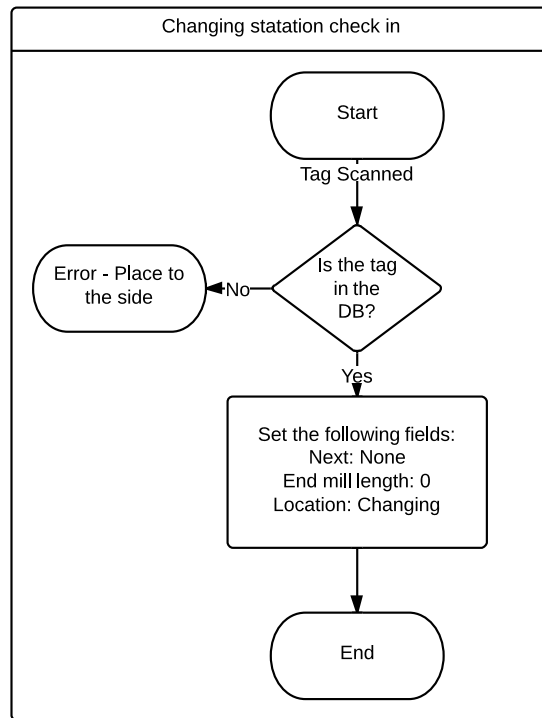


Figure 3.2.: Automated tool crib flowchart

When the robot removes a tool holder from the drawer it will pass it by an RFID scanner which will check it into the system and put the location in the database as "Changing end mill." However when the tool holder is scanned if that particular tool holder is not found in the database, the robot should place the tool holder to the side so that someone can add it in if desired when retrieved.

3.3.2. Storage In and Out

Figure 3.3 shows the flowchart for the process of taking a new assembly into the changing stations storage area.

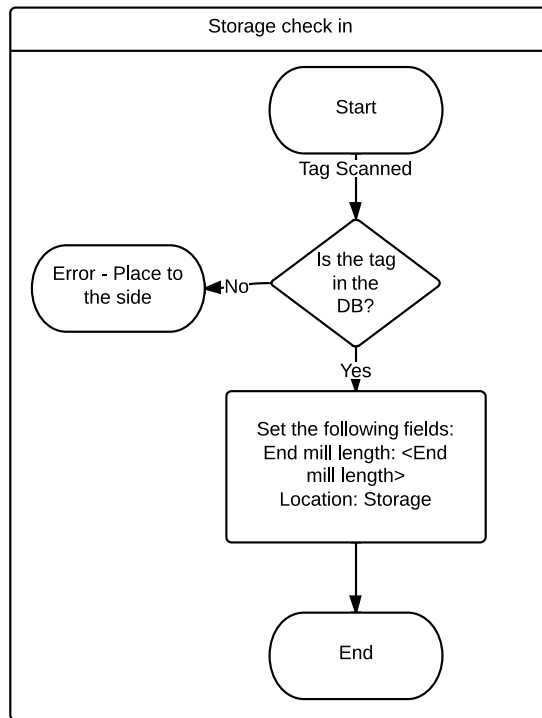


Figure 3.3.: Storage in flowchart

When an end mill is done being changed, the FANUC will bring it by an RFID scanner which will check to see if the tool holder exists in the database. If the tool holder is not in it, then the tool holder will be placed to the side as in the changing station entrance so it can be added if desired. If it was in the database then it will have its' current location updated as well as the length of the just installed end mill.

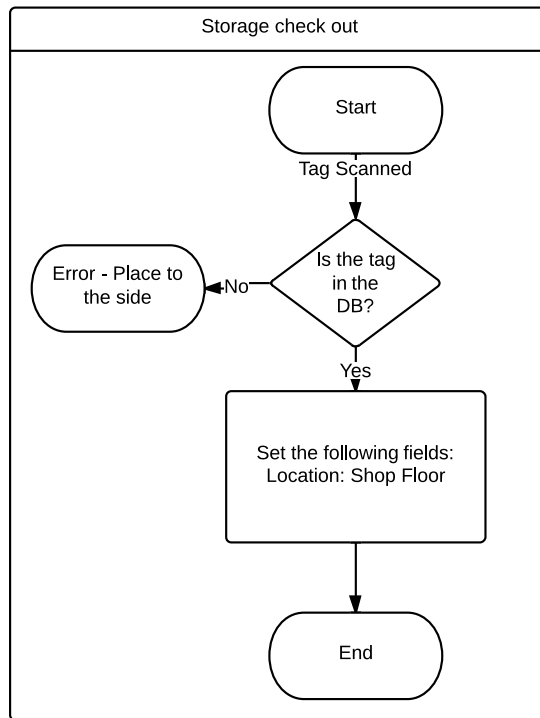


Figure 3.4.: Storage out flowchart

Figure 3.4 shows the flowchart for the process of taking a new assembly out of storage. When the arm scans a tool holder to check it out of the system, if it is not found in the database it will be set aside so it can be added later if desired and a different one can be grabbed instead. Otherwise it will just set the location database field to being on the shop floor.

3.3.3. CNC Carrousel

Figure 3.5 shows the flowchart for the process for taking a new assembly into a tool carrousel.

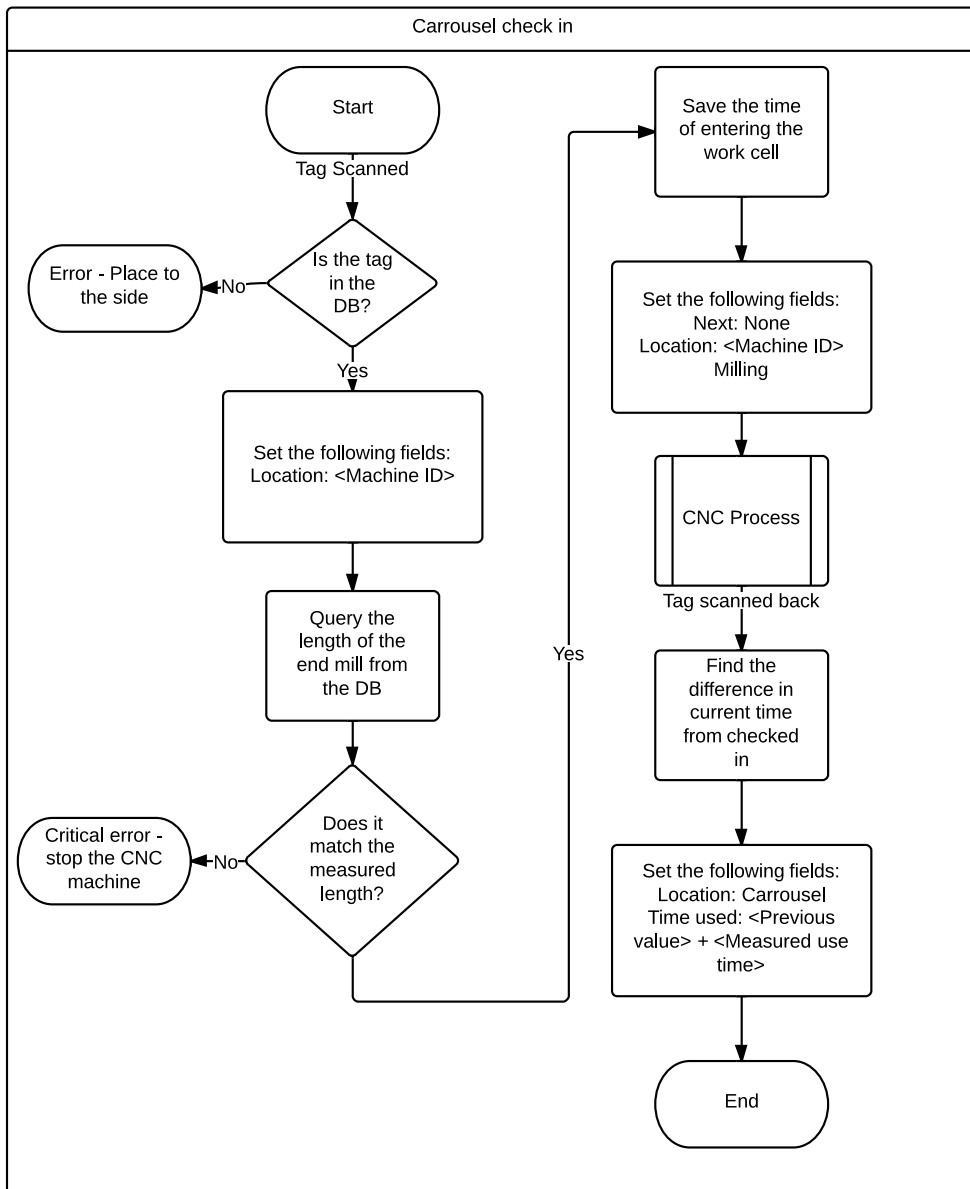


Figure 3.5.: CNC carrousel flowchart

When an employee puts the assembly into a CNC machine, it will be scanned and if it is not found in the database it will let the user know they have the wrong assembly in hand. Otherwise it will update the location field in the database to the current machine.

In the future, when the assembly goes into the CNC machine to be used its' end mill length will be queried from the database and checked to see if it matches what it thinks it has. If the length doesn't match, the CNC machine will report an error and need to have a worker come address it as the tool is not the correct length. However if the length is correct, the reader will save the time that the assembly is entering the machine and wait for it to come back out. When the assembly is scanned out of the machine, the difference in the time in and time out will be computed and it will have the time used field in the database updated.

3.4. Reader Selection

Multiple RFID readers will be needed for all the required location states and to make sure everything is automated so employees don't have to enter information unless absolutely necessary. To keep costs low, an off-the-shelf reader was selected from Sparkfun - the ID-12LA which was chosen due to previous experience with it and the ease of use experience it provided. It operates at 125 kHz has a reading range of approximately 4 inches and which is not a long range, but it will suffice for a proof-of-concept. This frequency would not likely be used by GE in a production environment, they would instead select 13.56 MHz as those tags are read/write and have more options for the physical packaging of the tag. An image of the selected reader can be seen in Figure 3.6 along with the tag style used.



Figure 3.6.: RFID reader and tag

3.5. Breadboard Circuit

Connecting the reader to a microprocessor is a very simple process; an Arduino Mega 2560 is used in Figure 3.7 for demonstration purposes.

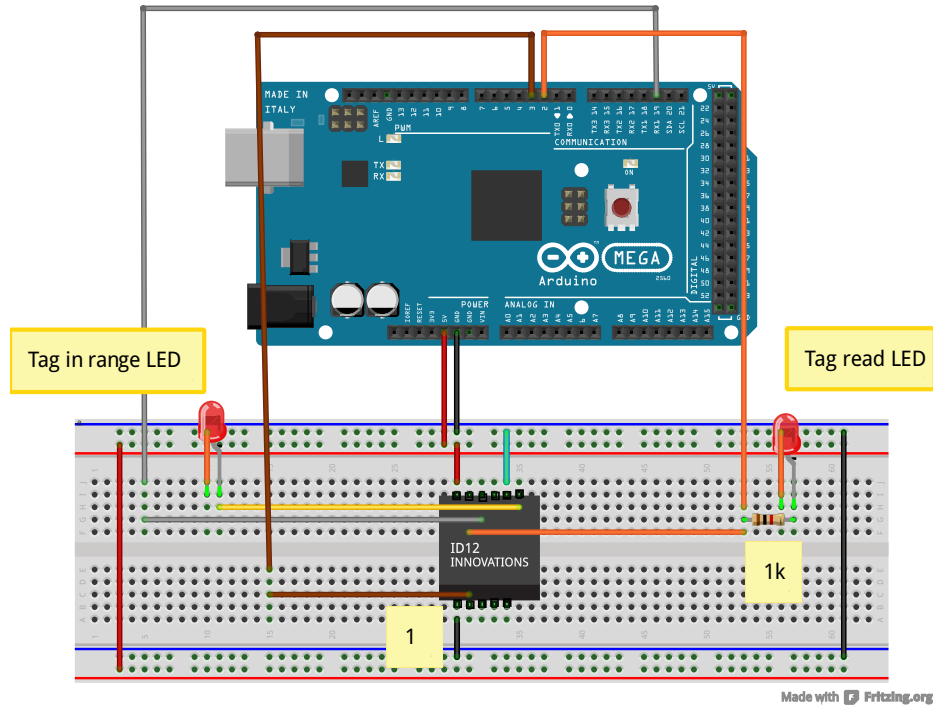


Figure 3.7.: Reader circuit design

There are 3 different data connections that have to go to the board: serial data, a tag in range line, and a tag read line; the only one that is really required for operation is the serial data line.

Reset is not supposed to be used according to the datasheet, however after some testing it seems there are no adverse affects of triggering a reset by pulling the pin temporarily to ground to make the reader restart. However if resetting capabilities are not needed, the reset line can just be tied to 5V as shown in Figure 3.7. This will be helpful when combined with the tag in range line; if there is a tag near the reader but no data was received then it can reset to try to re-read the tag. The last line is the data line which just sends serial data out to the microprocessor to be interpreted. The information stored on the tags being used for testing is 10 data bits followed by 2 checksum bits.

3.6. Selecting a Communication Method

3.6.1. Overview

For communicating from the RFID readers back to the database and vice versa, there are two possible methods: wired and wireless. For wired, the only real choice is Ethernet (IEEE 802.3) as the data has to possibly travel long distances from the two points - which 802.3 is meant to accomplish. Wireless has two different possible standards that are appropriate in this application, Wi-Fi (IEEE 802.11) and XBee (IEEE 802.15.4) that each have their own advantages and disadvantages. General Electric (GE) has expressed interest in using their Wi-Fi system once they revamp it in the coming year, however they are fine with using Wired as data drops are already installed at each CNC machine. They intend the system to grow up to a maximum of no more than 100 machines, each with potential to house 40 tool holders within at any given time and would like to be able to track 10,000 tool holders within the company.

3.6.2. Wireless

GE has already expressed interest in using their soon-to-exist infrastructure of 802.11, however looking at the number of machines involved this could potentially cause problems.

Each machine may have up to 2 readers within and assuming a worst case scenario of 2 wireless transmitters per machine, this can possibly cause issues with reliability. If every machine were reading in 4 tags per minute (10% of their capacity) that's 400 messages being sent every minute, or one every 150 ms on average. This essentially is still small, however paired with whatever else GE plans to put onto their network it can start to cause data to not reach it's destination which is not as easily detectable with wireless as there is no collision detection, only avoidance.

If an XBee based system were to be used, then the 900 MHz spectrum becomes an option which will not interfere with 802.11 and it is unlikely they have anything else on that spectrum (GE was unable to confirm at the time of writing). However, there is also another major problem, the environment cannot be tested beforehand so noise levels within the plant cannot be tested. The signal pathloss within the plant cannot be checked as the signal travels to the receiver so it is unknown if it is even an option, especially when they are revamping the system because the current one is unreliable.

A lot of these problems can be solved with software, and doing checks to make sure the database was appropriately modified (many of which will have to be done anyway), however, there is not much of a reason to be using wireless on a machine that will not be moving and is in a fixed position. It would be using a system that has numerous unknowns in the current situation and may not be feasible if implemented.

3.6.3. Wired

Wired would address all of the drawbacks with wireless, and allows for much easier expansion if ever desired as Ethernet can be easily segmented. As data ports are already available at every machine, the only real drawback would be if an inexpensive hub or switch is needed to support additional hosts.

Ethernet is designed to be able to span large distance (328 ft with Gigabit) and this will not be any concern with data drops right next to the reader. As data packets will all be small, and already calculated at 1 every 150 ms on average, it will not be a problem and create lots of collisions, and any that were to happen, data would just retransmit if using TCP/IP.

Choosing wired will be the safest choice for GE without conducting extensive testing on the performance of either wireless standard within their plant. There are just too many unknown variables that could cause wireless to be non-functional or suffer from

poor performance.

3.7. Database Selection

A database had to be selected that would be able to adapt to various different tool holders and likely other tools in the future. A list of requirements was made that needed to be met.

- Easily scalable
- Able to export to other formats
- Able to be accessed through the web by some plugin
- Easy to setup
- Ability to handle dynamic data

The next step was to look into the types of databases available. There were two different types of database solutions that were looked into: SQL and NoSQL. SQL is a relational database which stores data in with a key and value into tables that represent whatever is desired and everything in the database must follow this format. NoSQL allows for SQL like database queries to be performed if desired, but they are not true SQL and is not a relational structure. NoSQL was developed about 30 years after SQL so many advantages and disadvantages were already known and were addressed as each individual database manager programmer saw fit.

The solution chosen is called MongoDB which is a NoSQL database as it is meets every requirement and more. MongoDB is able to easily be interfaced with Python as well as PHP and performed well in initial tests making it a prime candidate.

3.8. Controller Selection

Going with the recommendation of using Ethernet, there are two main solutions that are feasible that will keep the cost low and be easy to integrate. Both solutions involve using an Arduino, however each solution has its own set of pros and cons. Other non-Arduino solutions are certainly possible such as a BeagleBone board, however also due to familiarity and the cost of an Arduino they were not selected. For a production level solution, a company such as Balluff would be used which specializes in this industry.

3.8.1. Arduino Uno and Ethernet Shield

Going for keeping costs low, an Arduino Uno with an Ethernet shield can be used to create a very basic interface to send data to and from the database. The combination kit sold by Cana Kit costs approximately \$50 on Amazon.com and would be able to send basic commands to the database to execute commands.

This combination however has many reports from users of not working well at times as a quick search on the web can yield. Some have reported that it is only an issue when trying to communicate with multiple end-points, while some have reported that the shield will not send data to even one destination and no answers were found to their problems. It is likely this would not affect the project judging by the seemingly small user group reporting failure and if it were encountered, could hopefully be solved quickly with troubleshooting. An Arduino Uno with Ethernet Shield can be seen in Figure 3.8.



Figure 3.8.: Arduino Uno

3.8.2. Arduino Yun

The Arduino Yun is a new product in the line and has brought some features that have not previously been seen in an Arduino.

The Yun has a second processor that has a Linux operating system running on it that would introduce a lot of possibilities to the RFID system. The most interesting of the possibilities is that the MongoDB has a Python library that is running on the server to manage the database. This means that some of the database work can be done right on the Arduino as well as the possibility to create more complicated programs to work directly with the RFID tags.

The Yun also has the added benefit of already having 802.11 and 802.3 built-in to it, along with USB host and an SD card slot for storage. With the SD card and Linux, each reader could have a webserver for management if so desired that would be very helpful

for managerial purposes, such as checking for potential problems.

All of the extra features that the Yun has makes it well worth the \$70 price tag as so much more can be done with it than with the Uno and shield, it was therefore selected for use in the project. The Yun can be seen in Figure 3.9.

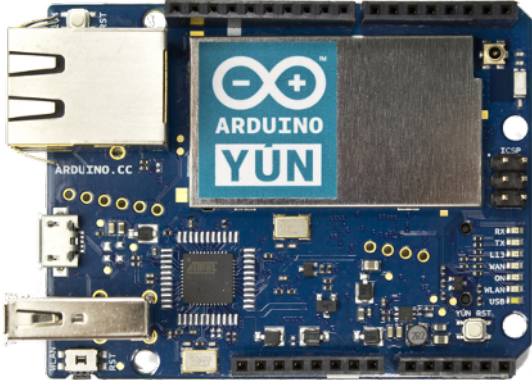


Figure 3.9.: Arduino Yun

3.9. Network Plan

To connect the Arduinos to the network, a network switch or hub is needed to let it connect with the machine it is paired with. Below in Figure 3.10 a plan for connecting the Yuns can be seen that segments each reader task into a different network branch.

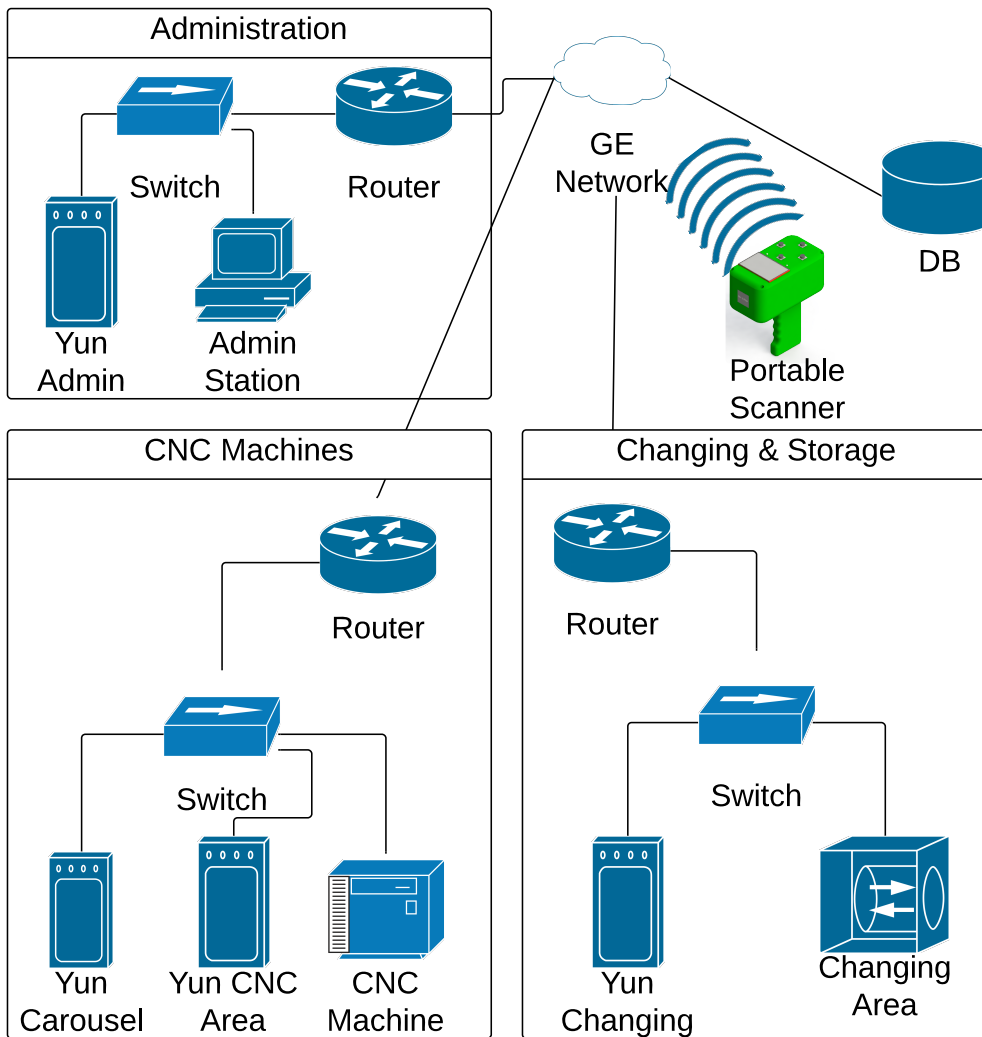


Figure 3.10.: Network plan

All of the network segments work in the following way. The clients are the end stations which all connect to a switch which allows multiple clients to connect to the network off of one data line. The routers are all optional, they can be used to segment networks for anything they want to contain within them, such as all (or some) of the CNC machines while still having access to the rest of the network. After the routers, everything will connect to the rest of GEs' network which will finally connect to the database where all

the tool holder data is contained.

The first segment in the upper left is the administration branch. This branch will be used when an employee wants to either look at the statistics of a tool holder in hand or they want to add a new one to the database.

The next branch can contain any number of CNC machines that will each have 2 Yuns for scanning first into the carrousel, and secondly into and out-of the machining area. This branch can have multiple machines and Yuns all on one router, but each group on their own switch.

The final branch is for the tool changing work cell. There will be one Yun which will work for both the changing and storage areas and will be able to track the multiple locations discussed earlier.

3.10. Portable Hand Scanner

Part way into the implementation phase, a requirement of a portable hand scanner was added in that should work over a wireless network so an employee can scan in data anywhere. The scanner will use the wireless connection to connect back to the database server to obtain all pertinent data about a tool holder. The design can be seen in Figure 3.11.

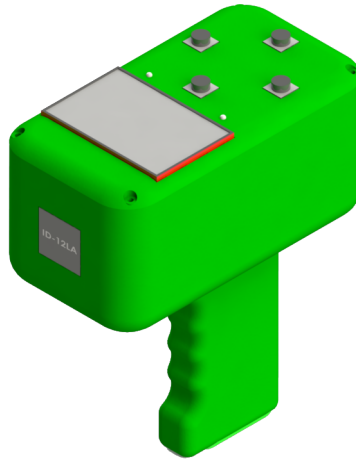


Figure 3.11.: Portable RFID Scanner

There is a large monochrome graphical display that will be able to show the information about the tool holder that was just scanned. Buttons and LEDs were included that are currently unfunctional and were included incase they were desired for anything in the future and a new part would not need to be created for something so simple.

The electronics will all be placed on the inside of the housing and a cutaway can be seen in Figure 3.12 which shows an Arduino Yun, an RFID reader, and a battery located in the handle.

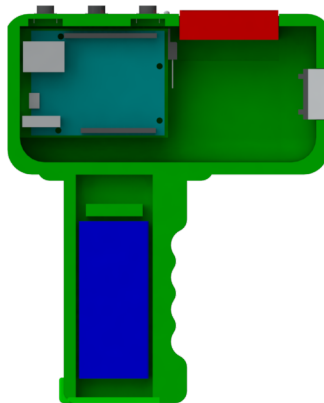


Figure 3.12.: Portable RFID scanner internals

This scanner can be printed out of PLA using a 3D printer in the same fashion that the tool changing grippers were. However none of the problems with the grippers will

be encountered as it only needs to hold electronics within it and is not subjected to any stresses.

3.11. Implementation and Testing

Everything was broken down and implemented incrementally starting first with the database and the website. From there each Arduino was tested one at a time starting with the simplest one of the administration station and then a portable hand scanner as these two only had to read data and not manipulate anything.

3.11.1. Database

MongoDB was a simple setup which just involved downloading the installer off the web from <https://www.mongodb.org/>. The 64-bit install was used which allows for virtually limitless space to be used for the database (compared to 2GB with the 32-bit version).

Test data was made using tag IDs that were purchased and test data was created to associate with it. Data was able to be successfully stored, manipulated, and retrieved by looking through the database with any search term.

3.11.2. Web Interface

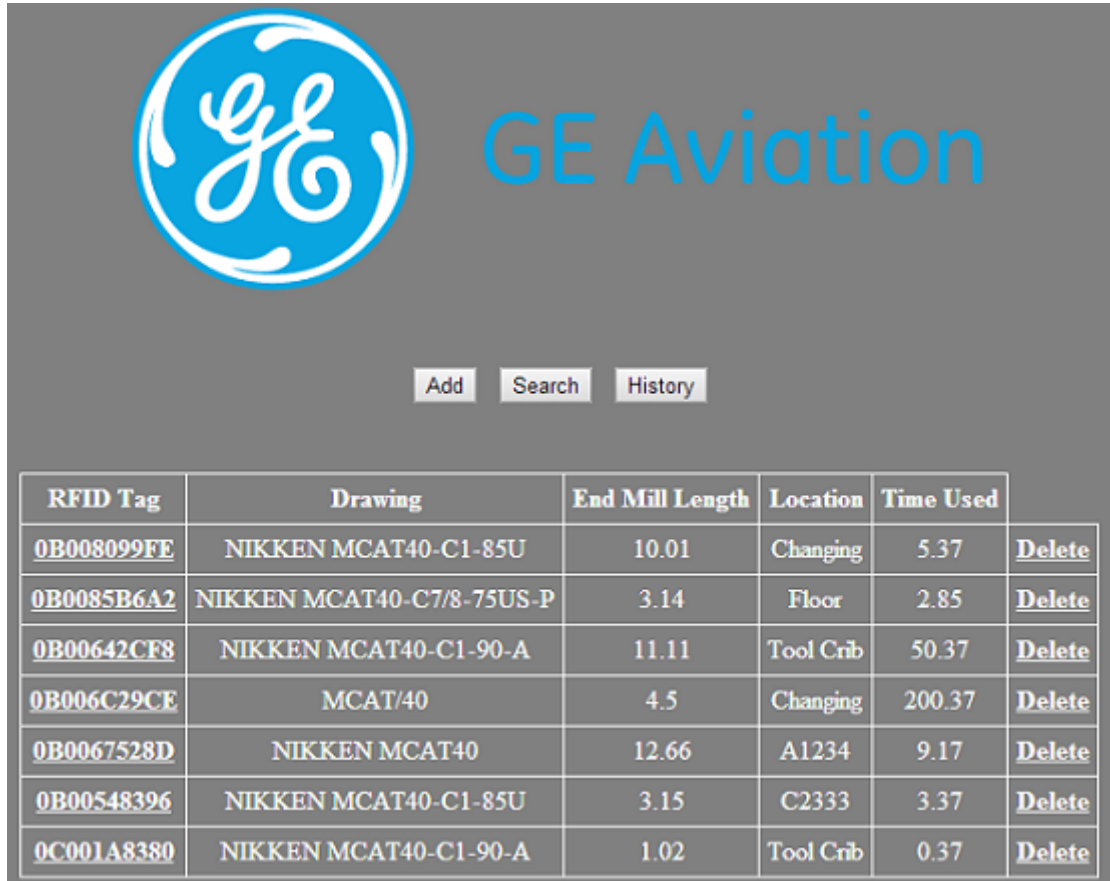
A simple website was developed to allow for GE to check everything that was outlined and was developed in HTML and PHP. Database queries are done by using a PHP plugin for MongoDB similar to PyMongo. Python was not used server side as PHP is more common place and is easier to setup.

To create a server testing environment, WampServer version 2.4 was used.¹

¹<http://www.wampserver.com/en/>

3.11.2.1. Basic Queries

The landing page of the website is shown in Figure 3.13 which pulls up everything in the database and shows it in a table.



The screenshot shows the GE Aviation web interface. At the top left is the GE logo, and to its right is the text "GE Aviation". Below the logo and text are three buttons: "Add", "Search", and "History". Below these buttons is a table with the following data:

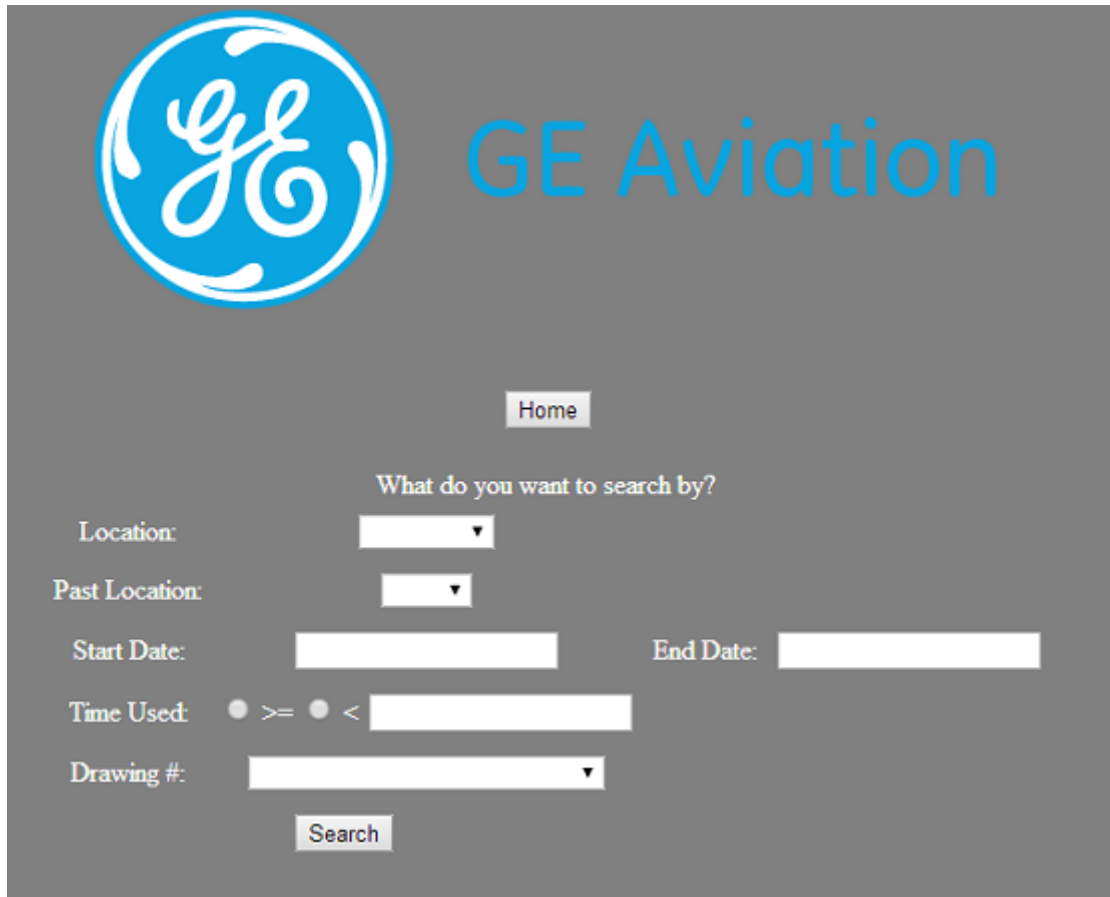
RFID Tag	Drawing	End Mill Length	Location	Time Used	
0B008099FE	NIKKEN MCAT40-C1-85U	10.01	Changing	5.37	Delete
0B0085B6A2	NIKKEN MCAT40-C7/8-75US-P	3.14	Floor	2.85	Delete
0B00642CF8	NIKKEN MCAT40-C1-90-A	11.11	Tool Crib	50.37	Delete
0B006C29CE	MCAT/40	4.5	Changing	200.37	Delete
0B0067528D	NIKKEN MCAT40	12.66	A1234	9.17	Delete
0B00548396	NIKKEN MCAT40-C1-85U	3.15	C2333	3.37	Delete
0C001A8380	NIKKEN MCAT40-C1-90-A	1.02	Tool Crib	0.37	Delete

Figure 3.13.: Web interface

All the RFID tags numbers are clickable which allows for editing the basic information: drawing number, length of currently installed end mills, the location, and the total time it has been used. The search button can be used to perform specific queries into the database and can search based off of any of the following parameters: location, past location, time used, and the drawing number. Clicking on the "Add" button allows an employee to create a new entry in the database with whatever tool holder they are looking to add.

3.11.2.2. Searching

Clicking on the search button brings up a page where a single set of parameters can be filled in as seen in Figure 3.14 for finding particular tool holders.



The screenshot shows the GE Aviation search interface. At the top left is the GE logo, and to its right is the text "GE Aviation". Below the logo is a "Home" button. The main heading is "What do you want to search by?". Below this are several search filters: "Location:" with a dropdown menu, "Past Location:" with a dropdown menu, "Start Date:" and "End Date:" with text input fields, "Time Used:" with two radio buttons labeled ">=" and "<" followed by a text input field, and "Drawing #:" with a dropdown menu. At the bottom is a "Search" button.

Figure 3.14.: Searching for a tool holder

Using any parameter except "past location" along with the date filters, will return results similar to the landing page. If the past location is used, the date filters can be used as well which allow for looking at jobs in a particular machine completed in a certain date range. To make date entering easier and ensure a user enters a date in a way that the server can interpret it, a date picking interface was created as shown in Figure 3.15.



Figure 3.15.: Date picker for filtering

The user can navigate months with the left and right arrows at the top or with the drop downs menus for month and year. The red date seen in the figure is just the current date for reference by the employee. Once a user does a search for all tool holders that have been through a machine, Figure 3.16 shows the structure of the results.



GE Aviation

Home

Machine: C

Tag	Time In	Time Out	End Mill Length
0B00642CF8	Fri Jan 31 08:02:30 2014	Wed Mar 19 19:55:50 2014	3.7
0B00642CF8	Mon Feb 24 12:35:00 2014	Mon Feb 24 12:46:39 2014	3.7

Figure 3.16.: Search results for jobs in a past machines

This is laid out as what tag was in the machine along with when it entered and exited the milling area and the end mill length used. Clicking on one of the links for a tag brings up the complete history for that tag along with the current information on it as seen in Figure 3.17. This page is also available when going to edit a tag; a button exists for easy access to the history.



GE Aviation

[Home](#)

[Edit](#)

Tag 0B00642CF8

Drawing NIKKEN MCAT40-C1-90-A

Used 50.37

Location Tool Crib

Start Date:

End Date:

[Filter](#)

Machine	Time In	Time Out	End Mill Length
C	Fri Jan 31 08:02:30 2014	Wed Mar 19 19:55:50 2014	3.7
C	Mon Feb 24 12:35:00 2014	Mon Feb 24 12:46:39 2014	3.7
D	Mon Feb 24 15:01:40 2014	Mon Mar 03 19:15:00 2014	2.2

Figure 3.17.: Job history of an individual tool holder

Similar to when searching, the history can be filtered by entering dates in the same manner then pressing the "Filter" button.

3.11.2.3. Admin Station

An Administration terminal was created that would allow for an employee to scan in any tag, and the information was then fed back to the webserver which would perform a search and display all appropriate data. This allows for an employee to edit the tool holder easily that is in their hand without having the problem of tracking down what they have by having to look up the tag manually. The webpage that shows what was scanned by this admin station is in Figure 3.18.

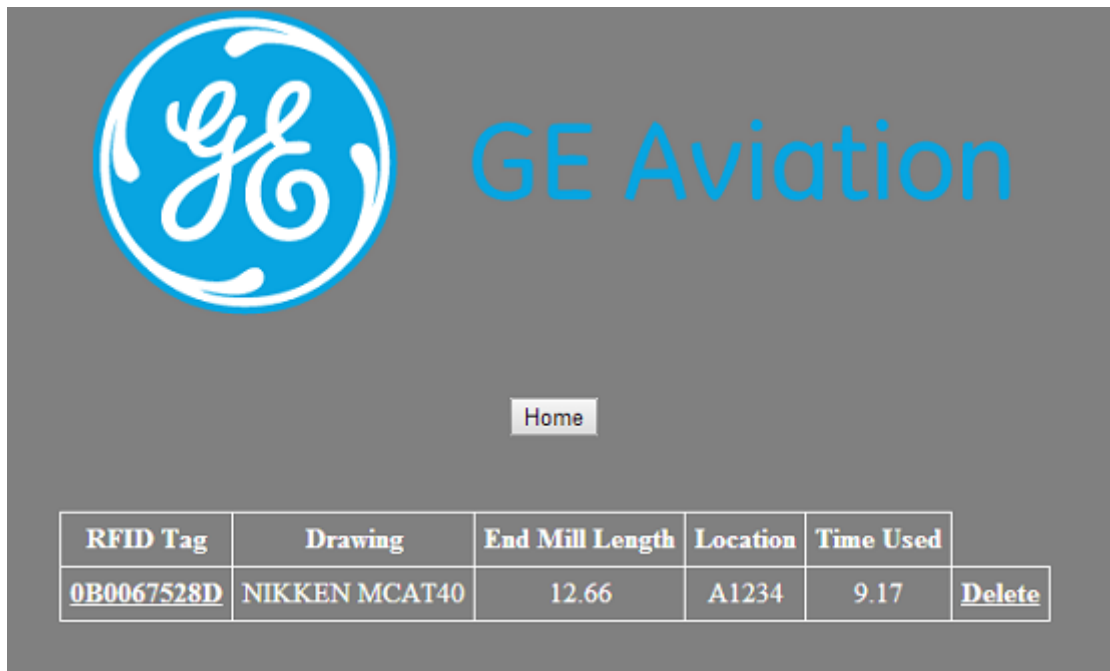


Figure 3.18.: Web Admin page

3.11.3. Portable Scanner

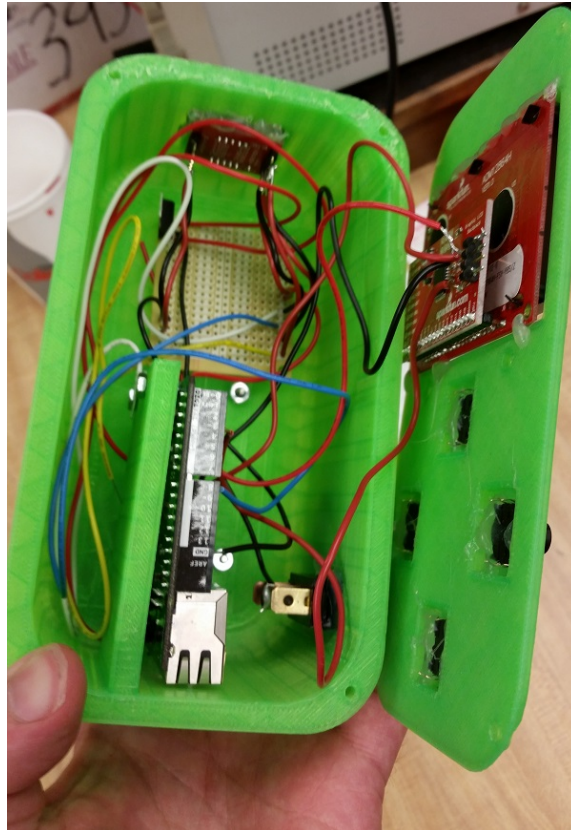
The scanner was printed on a 3D printer and the results are shown below in Figure 3.19.



(a) Isometric view



(b) Top view



(c) Interior

Figure 3.19.: Portable RFID scanner printed

There were some problems with the print warping in some places, however this was not an issue that affected anything as the only functions of the parts is to house electronics.

3.11.3.1. Results

Scanning in a tag produces the following screen output shown below in Figure 3.20.

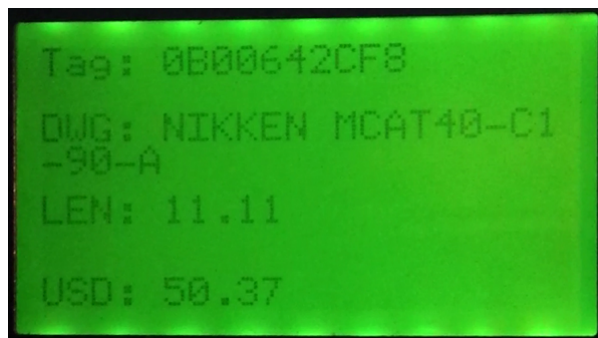


Figure 3.20.: Portable scanner output

As seen, all the data fits comfortably on the screen and is very easy to read. The wireless was reliable for retrieving from the database, and everything worked as desired.

3.11.4. CNC Machine

The CNC machine was the most complicated system and needed to be able to read from two different readers. The first reader checks the tool holder into the machine and the second one times how long it is in the machine for milling.

To accomplish this, all it required in addition to the previous code is a modification step in the code. When a tool holder enters the carousel, it gets its location updated, and when it completes a CNC job, it gets the total use time updated.

3.11.4.1. Results

The reader is able to modify the location of the tool holder as it enters and leaves each part of the CNC machine and identifies whether it is in milling or sitting in the carousel.

Checking the job history as a tool holder passes over the milling reader modifies the history as well as the use time used corresponding to how long the job was and reports the end mill length at the time. The functionality of the CNC readers were the most critical portion of the tool tracking objectives and the goals were all met with it.

3.11.5. Changing and Storage

The changing and storage reader was meant to be added into the tool changing part of the project, but because of the problems with that portion it was not fully implemented. The code is functional, however the robot was not programmed to automatically scan in tool holders as they are removed and replaced into the storage drawer. The new solution that would have to happen is that employees would have to bring the tool holder past a scanner themselves instead of the robot doing it for them. Also, because a storage system was never implemented the work cell location was changed from the separate changing and storage points to just "FANUC Changing/Storage" for database logging.

3.11.5.1. Results

The testing procedure was similar to the CNC machines and tags were just scanned in and checked to see if the current location was changed. Tool holders were able to be checked into and out of the cell as planned without any problems and worked as desired.

4. Further Discussion and Future Work

4.1. Discussion

4.1.1. Repeatability Considerations

4.1.1.1. Tool Holder

Much of the reliability problems were already discussed under each sub-systems performance section. The major points for each system are summarized below.

The largest problem for repeatability with the vise is the lack of current sensing and the issue of the jaws not adjusting to the height of the collet nut. The lack of current sensing allows for the potential to over-tighten the tool holders which doesn't allow the tool holders to be tightened properly all the time. Each tool holder may have a slightly different number of turns required for fully tightening and loosening that would need to be addressed in the future to reliably replace end mills.

The next problem is that installing an end mill at the right length was never addressed meaning that placing them into the collet at what worked once, may not work the next time. This can be fixed for this one tool holder by just installing a metal rod or an actuator that sticks out of the center of the vise and is inserted into the hole at the bottom of the tool holder.

The grippers biggest problem is the tooling one is made out of wood, not aluminum. This often caused the end mills to be picked up crooked which caused issues with re-

moving them from the collet as well as re-inserting the new ones.

There were no issues with the drawer that were found during testing. The only improvement that could be made is it could be expanded to the full design, however this would not have any foreseeable reliability affects and would only expand storage.

4.1.1.2. Tool Tracking

The tool tracking portion was perfectly repeatable during testing, however some modifications need to be made for a production level solution. The error checking outlined in the flowcharts needs to be implemented and likely more as the system is further tested.

Two issues may come up from moving from a clean laboratory environment to an industrial environment. The first is that the machinery that is located around the RFID readers may cause excessive interference that will affect the integrity of tag data being read and transmitted. The second issue is that in the lab, tags are always under ideal conditions which include orientation and distance which may not always be easy to ensure in a factory. Upgrading from the consumer grade product and ensuring all data lines are shielded should take care of both of these problems however.

4.1.2. Economic Considerations

4.1.2.1. Tool Changing

The tool changing portion of this project cost approximately \$1,500 considering the cost of everything purchased for testing and constructing. This price does not reflect that of the robot or any peripherals that were already owned by the university or GE, only that of additional materials purchased. The majority of this cost is from the aluminum stock that was purchased for creating the vise as well as the reversible motor driver and the H bridge.

Continuing the project in a linear fashion and adding support for one additional tool

holder per year, the project cost should be around the same to as the vise will remain the main component to be reconstructed. Another possible large cost in the future will be a change to the motor system, however it is not a critical part as it serves its purpose even though it is not an ideal solution.

There are a few major aspects to take into consideration for further development of the project that have the potential to cause price spikes, however they can be lessened if handled properly. As the vise is improved, the amount that it can be further expanded will saturate and another vise will need to be constructed to deal with different types of tool holders. This will require an expansion of the current one for more space. The robot has the workspace to have a bigger cell, and could even take advantage of things being placed on the walls or ceiling for more room. This will deal with any vises that could be constructed in the foreseeable future and should allow for many tool holders to be dealt with.

4.1.2.2. Tool Tracking

The cost of the tool tracking was almost entirely made up of Arduino Yun's and RFID readers. The total cost spent on the tool tracking was approximately \$700. The cost of a server running a database and a webserver was not included as a personal machine was used for testing which would normally need to be considered unless an existing server is extended with a virtual machine.

Going to a full-blown industrial solution would require much more money to be invested in the hardware at a minimum as the software was developed to be modular. However doing so would lose whatever features were already developed by a professional company so it would be best to switch both which could also affect server-side requirements as well with more powerful components.

4.2. Future Work

4.2.1. Tool Changing

There are several improvements that could be made to the tool changing process to further development. The first is to design the vise to work with a broader range of tool holders than just the one that it currently works with. Changes that would need to be made can't be determined until the corresponding tool holders are decided on, however they would likely revolve around the vise system. The vise would need a better way to be able to loosen and tighten the collet nut and a way to install end mills at different lengths. The final improvement to the vise is a way to reliably install end mills at a desired length. This could be done easily by installing an actuator that adjusts to the desired height of whatever end mill is being installed and would be a simple fix.

To deal with the collet nut, the most likely change would be making the height of the jaws adjustable so they can grab tool holders of different lengths. The locking mechanism would likely still be the same for similar tool holders and probably does not need adjusting.

Another vise improvement is with the motor system. The indexing in 45° increments only slows down the whole process and with a regular motor, could be done faster. With a current sensor added in as originally planned, the motor could begin to slow down and eventually stop once it is tightened completely and this would work better than running the motor for a set length of time.

The drawer and the grippers may not need to be adjusted depending on if the bottom part of the tool holder stays the same or not. If it is of the same length and diameter then neither would need adjusting as they will still function in the same way.

The final and most important work that needs to be done is all of the safety considerations discussed in Section 2.7.

4.2.2. Tool Tracking

The only real future work with regards to the tool tracking is to try to test out an industrial solution and compare the results. The consumer electronics were able to perform perfectly fine but the only real question is with regards to how it would deal with an electrically noisy factory environment. If the consumer grade electronics were deemed to be acceptable by GE for their needs, then the biggest improvement that can be recommended is to get a reader with a better antenna. The four inch range of the current reader was the only real draw back that was found during testing and other reader support external antennas that have much better ranges that could work better.

Once a final reader style is selected then a tag style can be as well. The tags used in this project were merely for testing but would not work in an industrial environment as they would not be able to be affixed to the tool holder in any way. There are multiple styles available that were not discussed due to the final reader not being known however many options do exist that are made specifically for this purpose.

One other component that could be improved is the portable hand scanner. A better housing solution should be developed that is not warped like the 3D printed parts were, and a keypad could very easily be installed into it to allow for more functionality. A full QWERTY keypad could be added that would allow for manipulating data on-the-spot instead of having to go back to a terminal to make changes to a tool holder.

5. Conclusions

This project aimed to improve the CNC work General Electric Aviation does on a daily basis. The first part of the project developed and constructed a method of replacing end mills that were used during a milling job. Multiple components were created capable of completing the whole process from the intake, to the changing, and to the returning a new assembly for a new milling job.

To take assemblies in and out of the work cell, a drawer was created for workers to use. The drawer worked as expected however it still needs safety systems implemented to keep workers safe from the robot. The robot moves assemblies between the drawer and a vise that is capable of spinning the collet nut that holds the end mills. The vise worked but had many reliability and safety concerns that will need to be addressed before it works as desired, however it is a good start. Finally, the grippers attached to the robot that move the tool holders around worked exactly as desired but the ones for the tooling still needs to be made out of aluminum and tested. The tool changing process still needs a lot of work but it has certainly made progress this project iteration.

The next goal of the project created a way to make tracking CNC tool holders possible. An RFID solution was designed and implemented that reads tags attached to each tool holder and a database contains pertinent information corresponding to each tag. Readers were designed to be spread across GE's facility so that everything can be tracked autonomously and everything worked as expected. The largest desired capability was to track the job history of each tool holder which was met by having at least one RFID

reader in the entrance of the milling area of each CNC machine.

This MQP was able to meet some of the initial goals, but not all. The largest result from the project shows that the goals GE would like to accomplish are feasible with more development time. With further development GE should be able to incorporate the project at their facility which will make their operations run smoother and be more profitable at the end of the day.

Appendices

A. FANUC DH Parameters

The DenavitHartenberg parameters were found and can be seen below in the table while the constant values are shown in the dimensions on the following page. This table was used to construct the MATLAB model and the code is in the following appendix which takes in this table to construct the arm at a given orientation.

Link	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	$\pi/2$
2	θ_2	0	a_2	0
3	θ_3	0	0	$\pi/2$
4	θ_4	d_4	0	$-\pi/2$
5	θ_5	0	0	$\pi/2$
6	θ_6	d_6	0	0

B. MATLAB Kinematics Code

The MATLAB code for the forward kinematics of the arm takes in a DH table as shown in the FANUC.m section to construct the arm. This produces the graphical model as was shown in Figure 2.21.

B.1. translate.m

```
1 function [ T ] = translate( steps , axis )
2 %TRANSLATE Generate a transformation matrix along an axis a given amount
3
4 %create a 4x4 I matrix
5 R = eye(4);
6
7 %puts the step amount in the either the x, y or z position
8 if axis == 'x'
9     R(1,4) = steps;
10 elseif axis == 'y'
11     R(2,4) = steps;
12 elseif axis == 'z'
13     R(3,4) = steps;
14 end
15 T = R;
16
17 end
```

matlab/fanuc/translate.m

B.2. tlink.m

```

1 function [ T ] = tlink( theta , d , a , alpha )
2 %TLINK Create a transformation matrix for the links
3
4 %get the transformation matrix
5 T = rotate(theta , 'z') * translate(d, 'z') * translate(a, 'x') * ...
6     rotate(alpha , 'x');
7
8 end

```

matlab/fanuc/tlink.m

B.3. rotate.m

```

1 function [ Rot ] = rotate( r_deg , axis )
2 %ROTATE Generate a rotation matrix about an axis by given amount in
3     degrees
4
5 %covert degrees to radians
6 r_rad = r_deg * (pi/180);
7
8 %create a 4x4 I matrix
9 R = eye(4);
10
11 %add in the rotations to the appropriate spots for the desired axis
12 if axis == 'x'
13     R(2,2) = cos(r_rad);
14     R(2,3) = -sin(r_rad);
15     R(3,2) = sin(r_rad);
16     R(3,3) = cos(r_rad);
17 elseif axis == 'y'
18     R(1,1) = cos(r_rad);
19     R(1,3) = sin(r_rad);
20     R(3,1) = -sin(r_rad);
21     R(3,3) = cos(r_rad);
22 elseif axis == 'z'
23     R(1,1) = cos(r_rad);
24     R(1,2) = -sin(r_rad);
25     R(2,1) = sin(r_rad);
26     R(2,2) = cos(r_rad);
27
28 end
29 Rot = R;
30
31 end

```

matlab/fanuc/rotate.m

B.4. fulltrans.m

```
1 function [ T ] = fulltrans( dhP )
2 %FULLTRANS Returns the position and orientation of the end effector and
3 %plots
4
5 T = zeros(4,4,6);
6 Temp = eye(4);
7
8 for i = 1:6
9     Temp = Temp*tlink(dhP(i,1), dhP(i,2), dhP(i,3), dhP(i,4));
10    T(:,:,i) = Temp;
11 end
12
13 for i = 1:6
14     Temp2 = T(:,:,i);
15     x1 = Temp2(1, 4);
16     y1 = Temp2(2, 4);
17     z1 = Temp2(3, 4);
18     if(i == 1)
19         x2 = 0;
20         y2 = 0;
21         z2 = 0;
22     else
23         Temp2 = T(:,:,i-1);
24         x2 = Temp2(1, 4);
25         y2 = Temp2(2, 4);
26         z2 = Temp2(3, 4);
27     end
28     hold on;
29     plot3([x1 x2], [y1 y2], [z1 z2]);
30 end
31 end
```

matlab/fanuc/fulltrans.m

B.5. FANUC.m

```
1 clear all; clc; close all;
2 %theta3 = 45+theta2;
3
4 dh = [ 0 5 1 -90;...
5       -20 0 5 180;...
```

```
      -90 0 3 90;...  
7      0 -5 0 -90;...  
      5 0 0 90;...  
9      0 -3 0 180];  
11 T = fulltrans(dh);  
    axis([-15 15 -15 15 0 15]);  
13 axis equal;
```

matlab/fanuc/FANUC.m

C. New Yun Setup

The following commands need to be run on an Arduino Yun to set them up for the first time. They install all necessary packages needed to control any of the readers and all are necessary except for the nano installation command.

```
opkg update
```

```
opkg install distribute
```

```
opkg install python-openssl
```

```
opkg install nano (optional editor instead of vim)
```

```
easy_install pip
```

- Sometimes required (was once during all setups):

```
pip install setuptools --no-use-wheel --upgrade
```

```
pip install pymongo
```

```
pip install ntplib
```


D. Plates and Pointer

Before anything can be attached to the FANUC, attachment plates had to be designed and machined. One plate would be for attaching to the robot, with a secondary one being used for mounting the desired end of arm tooling where each end effector would have a separate plate. To help create custom coordinate frames for movement while programming the robot, a rudimentary pointer was also designed that would just serve as a visual guide to help with that process. Once designed, steel and aluminum blanks were purchased and brought to Washburn Shops on campus to mill into the final product, seen in Figure D.1.

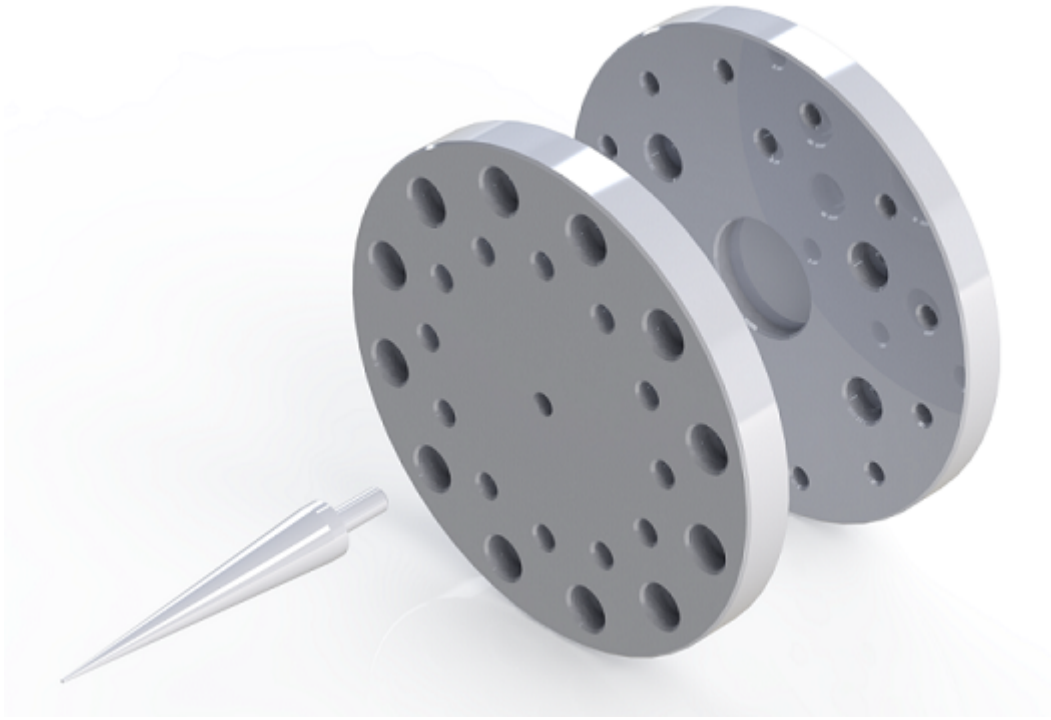


Figure D.1.: Plate for mounting to the arm, along with a plate for mounting end of arm tooling and a pointer

E. Reader Code

Following in the code for each of the readers, Arduino and Linux side.

E.1. Arduino Side (C)

The Arduino code is the same for all readers except for the hand scanner and CNC machine.

E.1.1. CNC Reader

```
1 //Admin Station
2 #include <SoftwareSerial.h>
3 #include <Bridge.h>
4
5
6
7 /*
8  * Pins - milling area
9  * RX = 8
10 * TX = 9
11 * Reset = 7
12 */
13 #define rxMill 8
14 #define txMill 9
15 #define resetPMill 7
16
17 /*
18 * Pins - carousel
19 * RX = 4
20 * TX = 3
21 * Reset = 2
22 */
```

```

23 #define rxCar 4
24 #define txCar 3
25 #define resetPCar 2
26
27 /*
28 * Start bit has a value of           0x2
29 * Newline bit has a value of         0xD
30 * Carriage return bit has a value of 0xA
31 * Stop bit has a value of           0x3
32 */
33 #define bBit 0x2
34 #define nLine 0xD
35 #define cReturn 0xA
36 #define sBit 0x3
37
38 /*
39 * Simple IDs for what reader was read
40 */
41 #define MILL_ID 0
42 #define CAR_ID 1
43
44 // The Yun needs to use software serial
45 SoftwareSerial RFID_MILL(rxMill, txMill);
46 SoftwareSerial RFID_CAR(rxCar, txCar);
47
48
49 /*
50 * The tag will give 12 ASCII chars
51 * The first 10 are the ID values
52 * The last 2 are the checksum
53 */
54 String rTagMill = "";
55 String rTagCar = "";
56
57
58
59 ////////////////////////////////////////////////////
60 /*
61 * Reads in data from the RFID reader, recieves 12 data bits
62 * and stores it into a global string called rTag
63 */
64 void readTag(){
65 // Checks to see if there is anything in the serial buffer
66 while(RFID_MILL.available()){
67 // Read in the available byte
68 char rChar = RFID_MILL.read();
69 // Only store it if it's data
70 if(rChar != bBit && rChar != nLine && rChar != cReturn && rChar !=
71 sBit)
72 rTagMill.concat(rChar);
73 // Need a short delay to let the next serial byte shift into the reg

```

```

73     delay(10);
74 }
75 while(RFID_CAR.available()){
76     // Read in the available byte
77     char rChar = RFID_CAR.read();
78     // Only store it if it's data
79     if(rChar != bBit && rChar != nLine && rChar != cReturn && rChar != sBit)
80         rTagCar.concat(rChar);
81     // Need a short delay to let the next serial byte shift into the reg
82     delay(10);
83 }
84 }
85
86
87
88 /*
89 * Prints out the tag and checksum through serial
90 */
91 void printTag(){
92     // If a tag was read
93     if(rTagMill != ""){
94         Serial.print("Mill Tag:\t");
95         for(char i = 0; i<10; i++){
96             Serial.print(rTagMill[i]);
97         }
98         Serial.print("\tChecksum:\t");
99         for(char i = 10; i<12; i++){
100             Serial.print(rTagMill[i]);
101         }
102         Serial.println();
103     }
104     if(rTagCar != ""){
105         Serial.print("Car Tag:\t");
106         for(char i = 0; i<10; i++){
107             Serial.print(rTagCar[i]);
108         }
109         Serial.print("\tChecksum:\t");
110         for(char i = 10; i<12; i++){
111             Serial.print(rTagCar[i]);
112         }
113         Serial.println();
114     }
115 }
116
117
118
119 /*
120 * Checks the integrity of the data
121 */
122 boolean chksum(String tagCheck){
123     // Convert the read ASCII data into hex nibbles
124     char hData[12];

```

```

125 for(char i = 0; i < 12; i++){
126     if(tagCheck[i] < 58) hData[i] = tagCheck[i] - 48;
127     else hData[i] = tagCheck[i] - 55;
128 }
129
130 // Combine every-other nibble into a byte meaning
131 // [0 1] [2 3] [4 5] [7 8] [9 10] [11 12]
132 // Byte #:
133 //   1     2     3     4     5     6
byte bytes[6];
135 char index = 0;
136 for(char i = 0; i < 12; i += 2){
137     char hByte = (hData[i] << 4) & 0xF0;
138     char lByte = hData[i+1] & 0x0F;
139     bytes[index] = (hByte | lByte) & 0xFF;
140     index++;
141 }
142
143 // Compares the computed checksum against the recieved
144 // Done by XORing all the data bits and seeing if it matches
145 // the checksum
byte xChecksum = 0;
147 for(char i = 0; i < 5; i++){
148     xChecksum ^= bytes[i];
149 }
150
151 // If it matches, return true
152 if(xChecksum == bytes[5]){
153     return true;
154 }
155
156 // Otherwise, return false
157 return false;
158 }
159
160
161
162 /*
163 * Sends data out the serial port to the computer
164 */
165 void sendTag(String tagCheck){
166     // Takes only the identifying portion of the tag (no checksum)
167     String dataOut = "";
168     for(char i = 0; i < 10; i++){
169         dataOut.concat(tagCheck[i]);
170     }
171     // Append the eol character
172     dataOut.concat('\n');
173     Serial.print(dataOut);
174 }
175

```

```

177
179 /*
181 * Trips the reset to try to re-read the tag before
182 * it's out of range
183 */
184 void reRead(int readerID){
185     Serial.println("oops");
186     // Toggle reset
187     if(readerID == MILL_ID){
188         digitalWrite(resetPMill, LOW);
189         delay(1);
190         digitalWrite(resetPMill, HIGH);
191     }
192     else if(readerID == CAR_ID){
193         digitalWrite(resetPCar, LOW);
194         delay(1);
195         digitalWrite(resetPCar, HIGH);
196     }
197 }
198
199 /*
200 * Posts the tag to the bridge.
201 */
202 void postTag(String tagCheck, int readerID){
203     if(readerID == MILL_ID){
204         // Takes only the identifying portion of the tag (no checksum)
205         String dataOut = "";
206         for(char i = 0; i < 10; i++){
207             dataOut.concat(tagCheck[i]);
208         }
209         Bridge.put("milltag", dataOut);
210     }
211     else if(readerID == CAR_ID){
212         // Takes only the identifying portion of the tag (no checksum)
213         String dataOut = "";
214         for(char i = 0; i < 10; i++){
215             dataOut.concat(tagCheck[i]);
216         }
217         Bridge.put("cartag", dataOut);
218     }
219 }
220
221
222 ///////////////////////////////////////////////////
223 void setup(){
224     // Open serial communications and wait for port to open:
225     Serial.begin(57600);
226     while (!Serial); // wait for serial port to connect.
227     RFID_MILL.begin(9600);

```

```

229 RFID_CAR.begin(9600);
    Bridge.begin();
231 }

233
void loop() {
235     // Read in the tag
    readTag();
237     if(rTagMill != "") {
        if(chksum(rTagMill)) {
239             sendTag(rTagMill);
            postTag(rTagMill, MILL_ID);
241         }
        // If we read a tag but got bad data, reset the reader and get a new
        // reading
243         else reRead(MILL_ID);
    }
245     if(rTagCar != "") {
        if(chksum(rTagCar)) {
247             sendTag(rTagCar);
            postTag(rTagCar, CAR_ID);
249         }
        // If we read a tag but got bad data, reset the reader and get a new
        // reading
251         else reRead(CAR_ID);
    }
253
255     // Clear the tag
    rTagMill = "";
257     rTagCar = "";
}

```

code/cnc_yun3/cnc_yun3.ino

E.1.2. Hand Scanner

```

//Admin Station
2 #include <SoftwareSerial.h>
#include <Bridge.h>
4 #include <SerialGraphicLCD.h> //(2,3) (rx[nc],tx)

6 #define bitCheck(var, pos) (!((var) & (1 << (pos))))

8 /*
* Pins
10 * RX = 8
* TX = 9

```



```

12 *   Reset = 3
13 */
14 #define rx 8
15 #define tx 9
16
17
18 /*
19 *   Start bit has a value of           0x2
20 *   Newline bit has a value of         0xD
21 *   Carriage return bit has a value of 0xA
22 *   Stop bit has a value of           0x3
23 */
24 #define bBit 0x2
25 #define nLine 0xD
26 #define cReturn 0xA
27 #define sBit 0x3
28
29 // Serial pixel control
30 #define maxX 127
31 #define maxY 63
32
33 byte center(char xy, int len){
34     if(xy == 'x')
35         return ( ((maxX/2) - (len * 3)) );
36     else
37         return ( (maxY/2) - 4 );
38 }
39
40 // Instantiate the LCD
41 LCD LCD;
42
43 // The RFID reader needs soft serial
44 SoftwareSerial RFID(rx, tx);
45
46
47
48
49
50 /*
51 *   The tag will give 12 ASCII chars
52 *   The first 10 are the ID values
53 *   The last 2 are the checksum
54 */
55 String rTag = "";
56
57
58 ///////////////////////////////////////////////////
59 /*
60 *   Reads in data from the RFID reader , recieves 12 data bits
61 *   and stores it into a global string called rTag
62 */

```

```

64 void readTag(){
    // Checks to see if there is anything in the serial buffer
66 while(RFID.available()){
    // Read in the available byte
68 char rChar = RFID.read();
    // Only store it if it's data
70 if(rChar != bBit && rChar != nLine && rChar != cReturn && rChar !=
    sBit) rTag.concat(rChar);
    // Need a short delay to let the next serial byte shift into the reg
72 delay(10);
    }
74 }

76

78 /*
    * Print tag data to the screen.
80 * TAG #
    * DWG #
82 * End Mill Length
    * Time Used
84 */
void printScreen(){
86     static int len = 21;
    char screen[len];
88     LCD.clearScreen();

90     Bridge.get("tagData", screen, len);
    //tag
92     LCD.setY(0);
    LCD.setX(0);
94     LCD.printStr("Tag: ");
    delay(25);
96     LCD.printStr(screen);
    delay(25);

98     //dwg
100    Bridge.get("dwgData", screen, len);
    LCD.setY(20-4);
102    LCD.setX(0);
    LCD.printStr("DWG: ");
104    delay(25);
    LCD.printStr(screen);
106    delay(25);

108    //length
    Bridge.get("lenData", screen, len);
110    LCD.setY(40-4);
    LCD.setX(0);
112    LCD.printStr("LEN: ");
    delay(25);
114    LCD.printStr(screen);

```

```

116     delay(25);
118     //time used
118     Bridge.get("usedData", screen, len);
120     LCD.setY(56);
120     LCD.setX(0);
120     LCD.printStr("USD: ");
122     delay(25);
122     LCD.printStr(screen);
124     delay(25);

126     //clear the ready flag
126     Bridge.put("ready", "N");
128 }

130

132 /*
132 * Checks the integrity of the data
134 */
134 boolean chksum(){
136     // If a tag was read
136     if(rTag != ""){
138         // Convert the read ASCII data into hex nibbles
138         char hData[12];
140         for(char i = 0; i < 12; i++){
140             if(rTag[i] < 58) hData[i] = rTag[i] - 48;
142             else hData[i] = rTag[i] - 55;
144         }

144         // Combine every-other nibble into a byte meaning
146         // [0 1] [2 3] [4 5] [7 8] [9 10] [11 12]
146         // Byte #:
148         //   1     2     3     4     5     6
148         byte bytes[6];
150         char index = 0;
150         for(char i = 0; i < 12; i += 2){
152             char hByte = (hData[i] << 4) & 0xF0;
152             char lByte = hData[i+1] & 0x0F;
154             bytes[index] = (hByte | lByte) & 0xFF;
154             index++;
156         }

158         // Compares the computed checksum against the recieved
158         // Done by XORing all the data bits and seeing if it matches
160         // the checksum
160         byte xChksum = 0;
162         for(char i = 0; i < 5; i++){
162             xChksum ^= bytes[i];
164         }

166         // If it matches, return true

```

```

168     if(xChksum == bytes[5]){
170         return true;
172     }
174     // Otherwise, return false
176     else{
178         return false;
180     }
182 }
184 // We didn't do anything, return false
186 return false;
188 }
190
192 /*
194 * Posts the tag to the bridge.
196 */
198 void postTag(){
200     if(rTag != ""){
202         // Takes only the identifying portion of the tag (no checksum)
204         String dataOut = "";
206         for(char i = 0; i < 10; i++){
208             dataOut.concat(rTag[i]);
210         }
212         Bridge.put("tag", dataOut);
214     }
216 }
218
220 ////////////////////////////////////////////////////////////////////
222 void setup(){
224     // Open serial communications and wait for port to open:
226     Serial.begin(57600);
228     while (!Serial); // wait for serial port to connect.
230     Bridge.begin();
232     delay(1000); // wait for the splash screen
234     LCD.setBaud(9600);
236     LCD.setHome(); // zero the cursor
238     LCD.clearScreen(); // clear screen
240     LCD.printStr("Ready!");
242     delay(1000);
244     LCD.setHome();
246     LCD.clearScreen();
248     RFID.begin(9600);
250 }
252
254 void loop() {
256     char readyFlag[2];

```

```

220 Bridge.get("ready", readyFlag, 2);
    if(readyFlag[0] == 'Y') printScreen();

222 // Read in the tag
    readTag();
224 // Verify the data integrity -
    // if it's good, send the tag ID to serial & bridge
226 if(chksum()){
    //sendTag();
228     postTag();
    }
230 // Clear the tag
    rTag = "";
232 }

```

code/portable_yun4/portable_yun4.ino

E.1.3. All others

```

//Admin Station
2 #include <SoftwareSerial.h>
#include <Bridge.h>
4
6
8 /*
 * Pins
 * RX = 8
10 * TX = 9
 * Reset = 3
12 */
#define rx 8
14 #define tx 9
#define resetP 3
16
18
20 /*
 * Start bit has a value of           0x2
 * Newline bit has a value of         0xD
22 * Carriage return bit has a value of 0xA
 * Stop bit has a value of            0x3
24 */
#define bBit 0x2
26 #define nLine 0xD
#define cReturn 0xA
28 #define sBit 0x3

```

```

30 // The Yun needs to use software serial
32 SoftwareSerial RFID(rx, tx);
34
36 /*
37 * The tag will give 12 ASCII chars
38 * The first 10 are the ID values
39 * The last 2 are the checksum
40 */
41 String rTag = "";
42
43
44 ////////////////////////////////////////////////////
45 /*
46 * Reads in data from the RFID reader, recieves 12 data bits
47 * and stores it into a global string called rTag
48 */
49 void readTag(){
50 // Checks to see if there is anything in the serial buffer
51 while(RFID.available()){
52 // Read in the available byte
53 char rChar = RFID.read();
54 // Only store it if it's data
55 if(rChar != bBit && rChar != nLine && rChar != cReturn && rChar !=
56 sBit) rTag.concat(rChar);
57 // Need a short delay to let the next serial byte shift into the reg
58 delay(10);
59 }
60 }
61
62
63
64 /*
65 * Prints out the tag and checksum through serial
66 */
67 void printTag(){
68 // If a tag was read
69 if(rTag != ""){
70 Serial.print("Tag:\t");
71 for(char i = 0; i<10; i++){
72 Serial.print(rTag[i]);
73 }
74 Serial.print("\tChecksum:\t");
75 for(char i = 10; i<12; i++){
76 Serial.print(rTag[i]);
77 }
78 Serial.println();
79 }
80 }

```

```

82
84 /*
85 * Checks the integrity of the data
86 */
87 boolean chksum(){
88     // If a tag was read
89     if(rTag != ""){
90         // Convert the read ASCII data into hex nibbles
91         char hData[12];
92         for(char i = 0; i < 12; i++){
93             if(rTag[i] < 58) hData[i] = rTag[i] - 48;
94             else hData[i] = rTag[i] - 55;
95         }
96
97         // Combine every-other nibble into a byte meaning
98         // [0 1] [2 3] [4 5] [7 8] [9 10] [11 12]
99         // Byte #:
100        //   1     2     3     4     5     6
101        byte bytes[6];
102        char index = 0;
103        for(char i = 0; i < 12; i += 2){
104            char hByte = (hData[i] << 4) & 0xF0;
105            char lByte = hData[i+1] & 0x0F;
106            bytes[index] = (hByte | lByte) & 0xFF;
107            index++;
108        }
109
110        // Compares the computed checksum against the recieved
111        // Done by XORing all the data bits and seeing if it matches
112        // the checksum
113        byte xChecksum = 0;
114        for(char i = 0; i < 5; i++){
115            xChecksum ^= bytes[i];
116        }
117
118        // If it matches, return true
119        if(xChecksum == bytes[5]){
120            return true;
121        }
122
123        // Otherwise, return false
124        else{
125            return false;
126        }
127    }
128
129    // We didn't do anything, return false
130    return false;
131 }
132

```

```

134
136 /*
136 * Sends data out the serial port to the computer
136 */
138 void sendTag(){
138     if(rTag != ""){
140         // Takes only the identifying portion of the tag (no checksum)
140         String dataOut = "";
142         for(char i = 0; i < 10; i++){
142             dataOut.concat(rTag[i]);
144         }
144         // Append the eol character
146         dataOut.concat('\n');
146         Serial.print(dataOut);
148     }
148 }
150
152
154 /*
154 * Trips the reset to try to re-read the tag before
154 * it's out of range
156 */
156 void reRead(){
158     Serial.println("oops");
158     // Toggle reset
160     digitalWrite(resetP, LOW);
160     delay(1);
162     digitalWrite(resetP, HIGH);
162 }
164
166
168 /*
168 * Posts the tag to the bridge.
168 */
170 void postTag(){
170     if(rTag != ""){
172         // Takes only the identifying portion of the tag (no checksum)
172         String dataOut = "";
174         for(char i = 0; i < 10; i++){
174             dataOut.concat(rTag[i]);
176         }
176         Bridge.put("tag", dataOut);
178     }
178 }
180
182
184 //////////////////////////////////////
184 void setup(){

```



```

186 // Open serial communications and wait for port to open:
Serial.begin(57600);
188 while (!Serial); // wait for serial port to connect.
RFID.begin(9600);
Bridge.begin();
190 }
192
194 void loop() {
// Read in the tag
readTag();
196 // Verify the data integrity -
// if it's good, send the tag ID to serial & bridge
198 if(chksum()){
sendTag();
200 postTag();
}
202 // If we read a tag but got bad data, reset the reader and get a new
reading
else if (rTag != "") reRead();
204 // Clear the tag
rTag = "";
206 }

```

code/admin_yun1/admin_yun1.ino

E.2. Linux Side (Python 2.7)

E.2.1. Admin Station

```

"""
2 This script takes in data from the bridge (Arduino and Linino link) for
the
last tag scanned on the RFID reader. It then does a PHP post to the
4 webserver that will show the corresponding information about the tool
holder. If the tag does not exist in the database already you will have
the
6 option to add it.
8 Also supports some basic DB functionality from wihtin the script, however
this is not a big focus of this script.
10 """
12 # Imports
import urllib
14 import urllib2

```

```

16 from threading import Timer
17 from pymongo import MongoClient
18 import sys
19 sys.path.insert(0, '/usr/lib/python2.7/bridge/')
20 from bridgeclient import BridgeClient as bridgeclient
21 bc = bridgeclient()
22
23 """
24 Timer – for running tasks every x seconds
25 """
26 class RepeatedTimer(object):
27     def __init__(self, interval, function, *args, **kwargs):
28         self._timer = None
29         self.interval = interval
30         self.function = function
31         self.args = args
32         self.kwargs = kwargs
33         self.is_running = False
34         self.start()
35
36     def _run(self):
37         self.is_running = False
38         self.start()
39         self.function(*self.args, **self.kwargs)
40
41     def start(self):
42         if not self.is_running:
43             self._timer = Timer(self.interval, self._run)
44             self._timer.start()
45             self.is_running = True
46
47     def stop(self):
48         self._timer.cancel()
49         self.is_running = False
50
51 """
52 *****
53 ***** MongoDB *****
54 *****
55 """
56 #Open a connection to the databse
57 c = MongoClient('130.215.241.97')
58 #Connect to the tool holders DB
59 db = c.ToolHolders
60 #Connect to the ChuckHolders collection
61 col = db.ChuckHolders
62
63 #The tag wasn't found in the DB
64 noDBtag = None
65 #The last tag scanned in – used to check if we scanned in anything new

```

```

lastTag = None
68 locs = [ "Changing", "Tool Crib", "Floor", "Carousel" ]

70 ""
72 ***** Disp Data *****
74 def pTLabels():
    labels = "{0:^15}{1:^15}{2:^15}{3:^15}{4:^15}".format("DWG", "Length",
    "Location", "Used", "Next")
76     print labels

78 def pTable(TH):
    # Data
80     dwg = TH["dwg"]
    leng = TH["len"]
82     loc = TH["loc"]
    used = TH["used"]
84     nxt = TH["next"]
    print "{0:^15}{1:^15}{2:^15}{3:^15}{4:^15}".format(dwg, leng, loc, used,
    nxt)
86

88 def findTag(tag):
    global noDBtag
90     # If the tags in the database, find it
    dTag = col.find_one({"_id": tag})
92     if dTag is None:
        print
94         print "The tag is not in the system, you should add it first..."
        print "Tag ID is: ", tag
96         noDBtag = tag.strip()
    else:
98         print
        # Cats
100         pTLabels()
        pTable(dTag)
102         print

104 def chLoc(location):
    print
106     pTLabels()
    for CH in col.find({"loc": location}):
108         pTable(CH)

110 def searchDB():
    print
112     print "1 to search by last tag\n2 to search by entering a tag\n3 to
    search by location"
    srTrm = raw_input()
114     if srTrm == '1':
        findTag(lastTag)

```

```

116 elif srTrm == '2':
    sTag = raw_input("Enter your tag: ")
118     findTag(sTag)
    elif srTrm == '3':
120         print
        for i in range(len(locs)):
122             print i, " for ", locs[i]
        sLoc = raw_input("Enter the location to search: ")
124         searchIn = locs[int(sLoc)]
        chLoc(searchIn)
126
128 def printCH():
    # Cats
130     print
    pTLabels()
132     for CH in col.find():
        # Data
134         pTable(CH)
136     """
    ***** Change Data *****
138     """
140 def addCH():
    global noDBtag
142     print "Please enter..."
    if noDBtag is not None:
144         print "Do you want to use the tag from memory? ", noDBtag
        mem = raw_input("(y/n): ")
146         if mem == 'y':
            id = noDBtag
148             noDBtag = None
        else:
150             noDBtag = None
            id = raw_input("Tag ID: ")
152     len = float(raw_input("Tool length: "))
    loc = "Tool Crib"
154     usd = float(raw_input("Time used: "))
    nxt = None
156     pst = None
    dwg = raw_input("Dwg: ")
158
    newCH = { "_id": id,
160              "len": len,
              "loc": loc,
162              "used": usd,
              "next": nxt,
164              "past": pst,
              "dwg": dwg
166              }

```

```

168 col.insert(newCH)
170 def updateCH():
171     print
172     print "Do you want to update the last scanned tag? ", lastTag
173     useScanned = raw_input("(y/n): ")
174     if useScanned == 'y':
175         uTag = lastTag
176     else:
177         uTag = raw_input("Enter the tag: ")
178
179     updt = raw_input("Enter the new location: ")
180     newLoc = {"loc": updt}
181     col.update({"_id": uTag}, {"$set": newLoc}, upsert = False, multi =
182         False)
183
184 def delCH():
185     print
186     if lastTag is None:
187         uTag = raw_input("Enter the tag you wish to delete: ")
188     else:
189         print "Do you want to delete the last scanned tag? ", lastTag
190         useScanned = raw_input("(y/n): ")
191         if useScanned == 'y':
192             uTag = lastTag
193         else:
194             uTag = raw_input("Enter the tag: ")
195     print repr(uTag)
196     col.remove({"_id": uTag})
197
198 """
199
200 *****
201 ***** Requests *****
202 *****
203 """
204
205 def reqTag():
206     global lastTag, bc
207     tag = bc.get('tag')
208     if(lastTag == tag):
209         pass
210     else:
211         lastTag = tag
212         url = 'http://130.215.241.97/admin_write.php?tag=' + tag
213         req = urllib2.Request(url)
214         response = urllib2.urlopen(req)
215         the_page = response.read()
216         findTag(tag)
217
218

```

```

220 rt = RepeatedTimer(2, reqTag) # it auto-starts, no need of rt.start()
222 """
224 *****
224 ***** Main *****
224 *****
224 """
226 def main():
228     while 1:
228         print
228         print "1 to view the data base"
230         print "2 to add a tool holder"
230         print "3 to delete an entry"
232         print "4 to edit an entry"
232         print "6 to search the DB"
234         inData = raw_input("$")

236         if inData == '1':
236             print "View"
238             printCH()
238         elif inData == '2':
240             print "Add"
240             addCH()
242         elif inData == '3':
242             print "Delete"
244             delCH()
244         elif inData == '4':
246             print "Edit"
246             updateCH()
248         elif inData == '6':
248             print "Search"
250             searchDB()

252 if __name__ == "__main__":
254     try:
254         main()
254     finally:
256         # Stop the timer
256         rt.stop()
258         print "Exiting safely..."

```

code/admin_yun1/admin_ctrl.py

E.2.2. CNC Reader

```

# Imports
2 import ntplib
from threading import Timer

```

```

4 from pymongo import MongoClient
import sys
6 sys.path.insert(0, '/usr/lib/python2.7/bridge/')
from bridgeclient import BridgeClient as bridgeclient
8 bc = bridgeclient()

10 machine = "A"

12 """
14 Timer
15 """
16 class RepeatedTimer(object):
17     def __init__(self, interval, function, *args, **kwargs):
18         self._timer = None
19         self.interval = interval
20         self.function = function
21         self.args = args
22         self.kwargs = kwargs
23         self.is_running = False
24         self.start()

26     def _run(self):
27         self.is_running = False
28         self.start()
29         self.function(*self.args, **self.kwargs)

30     def start(self):
31         if not self.is_running:
32             self._timer = Timer(self.interval, self._run)
33             self._timer.start()
34             self.is_running = True

36     def stop(self):
37         self._timer.cancel()
38         self.is_running = False

40

42 """
43 *****
44 ***** MongoDB *****
45 *****
46 """
47 # Connect to the mongo
48 c = MongoClient('130.215.241.97')
49 # Connect to the db in use for tool holders
50 db = c.ToolHolders
51 # Connect to the collection for chuck style tool holders
52 col = db.ChuckHolders

54 noDBtag = None
lastTag = None

```

```

56
58 """
***** Change Data *****
60 """
time = ntplib.NTPClient()
62
def getTime():
64     tStamp = time.request('us.pool.ntp.org')
    return tStamp.tx_time
66
cHolder = [0.0, 0.0, 0.0]
68
#need to see if the machine is in the list, if not add it and put in a
list
70 def inOut(tag, jobStart, toolLen):
    global cHolder
72     holder = col.find_one({"_id": tag})

    if jobStart:
74         time = getTime()
76         cHolder[0] = time
        cHolder[2] = holder["len"]
78     else:
        time = getTime()
80         cHolder[1] = time
        past = holder
82         col.update({"_id": holder["_id"]}, {"$push": {"past": "past."+machine: cHolder
        }}, upsert = False, multi = False)

84         used = holder["used"]
        jobTime = cHolder[1] - cHolder[0]
86         jobTime = (jobTime / 60) / 60 #sec in an hour
        used = round(used + jobTime, 3) #new total time
88         col.update({"_id": holder["_id"]}, {"$set": {"used": used}}, upsert =
        False, multi = False)

90         cHolder = [0.0, 0.0, 0.0]

92
94 def updateLoc(tag, loc):
    col.update({"_id": holder["_id"]}, {"$set": {"loc": loc}})
96
"""
98 *****
***** Requests *****
100 *****
"""
102
def reqTag():
104     global lastTag, bc

```



```

106 millTag = str(bc.get('milltag'))
    carTag = str(bc.get('cartag'))

108 #milling
    if(millTag != "x"):
110         if(lastTag == millTag):
            inOut(millTag, False, 0)
112             lastTag = "" #clear the last tag, we're done with it
            bc.put("milltag", "x")
114             updateLoc(millTag, machine+" (milling)")
        else:
116             inOut(millTag, True, lenInstalled)
            lastTag = millTag
118             bc.put("milltag", "x")
            updateLoc(millTag, machine+" (carousel)")
120
122 #carousel
    if(cartag != "x"):
        holder = col.find_one({"_id": carTag})
124         location = holder["loc"]

126         if location == machine+" (carousel)":
            updateLoc(millTag, "Floor")
128         else:
            updateLoc(millTag, machine+" (carousel)")
130
132         bc.put("cartag", "x")

134 rt = RepeatedTimer(2, reqTag) # it auto-starts, no need of rt.start()
136 """
138 *****
140 ***** Main *****
142 *****
144 """

146 def main():
    while 1:
148         continue

150 if __name__ == "__main__":
    try:
152         print "Begin!"
        main()
    finally:
        # Stop the timer
        rt.stop()
        print "Exiting safely..."

```

code/cnc_yun3/cnc_ctrl.py

E.2.3. Changing and Storage

```
1 # Imports
2 from threading import Timer
3 from pymongo import MongoClient
4 import sys
5 sys.path.insert(0, '/usr/lib/python2.7/bridge/')
6 from bridgeclient import BridgeClient as bridgeclient
7 bc = bridgeclient()
8
9
10 """
11 Timer
12 """
13 class RepeatedTimer(object):
14     def __init__(self, interval, function, *args, **kwargs):
15         self._timer = None
16         self.interval = interval
17         self.function = function
18         self.args = args
19         self.kwargs = kwargs
20         self.is_running = False
21         self.start()
22
23     def _run(self):
24         self.is_running = False
25         self.start()
26         self.function(*self.args, **self.kwargs)
27
28     def start(self):
29         if not self.is_running:
30             self._timer = Timer(self.interval, self._run)
31             self._timer.start()
32             self.is_running = True
33
34     def stop(self):
35         self._timer.cancel()
36         self.is_running = False
37
38 """
39
40 *****
41 ***** MongoDB *****
42 *****
43 """
44 # Connect to the mongo
45 c = MongoClient('130.215.241.97')
46 # Connect to the db in use for tool holders
47 db = c.ToolHolders
48 # Connect to the collection for chuck style tool holders
```

```

col = db.ChuckHolders
50
noDBtag = None
52 lastTag = None

54 """
56 ***** Change Data *****
58 """

60 def inOut(tag):
    holder = col.find_one({"_id": tag})
62     location = holder["loc"]

64     if location == "FANUC Changing/Storage":
        col.update({"_id": holder["_id"]}, {"$set": {"loc": "Floor"}}) #check
        out
66     else:
        col.update({"_id": holder["_id"]}, {"$set": {"loc": "FANUC Changing/
        Storage"}}) #check in

68
70 """
72 ***** Requests *****
74 """

76 def reqTag():
    global lastTag, bc
78     tag = str(bc.get('tag'))
    if(tag != "x"):
80         if(lastTag == tag):
            inOut(tag, False, 0)
82             lastTag = "" #clear the last tag, we're done with it
            bc.put("tag", "x")
84         else:
            inOut(tag, True, 4.23)
86             lastTag = tag
            bc.put("tag", "x")

88
rt = RepeatedTimer(2, reqTag) # it auto-starts, no need of rt.start()
90
92 *****
94 ***** Main *****
96
98 def main():
    while 1:

```

```

    continue
100
if __name__ == "__main__":
102     try:
        print "Begin!"
104         main()
    finally:
106         # Stop the timer
        rt.stop()
108         print "Exiting safely..."

```

code/fanuc_yun2/fanuc_ctrl.py

E.2.4. Portable Reader

```

# Imports
2 from threading import Timer
  from pymongo import MongoClient
4 import sys
  sys.path.insert(0, '/usr/lib/python2.7/bridge/')
6 from bridgeclient import BridgeClient as bridgeclient
  bc = bridgeclient()
8
10 """
  Timer
12 """
  class RepeatedTimer(object):
14     def __init__(self, interval, function, *args, **kwargs):
        self._timer = None
16         self.interval = interval
        self.function = function
18         self.args = args
        self.kwargs = kwargs
20         self.is_running = False
        self.start()
22
        def _run(self):
24             self.is_running = False
            self.start()
            self.function(*self.args, **self.kwargs)
26
        def start(self):
28             if not self.is_running:
                self._timer = Timer(self.interval, self._run)
30                 self._timer.start()
                self.is_running = True
32

```

```

34     def stop(self):
35         self._timer.cancel()
36         self.is_running = False
37
38     """
39
40     *****
41     ***** MongoDB *****
42     *****
43     """
44     # Connect to the mongo
45     c = MongoClient('130.215.241.97')
46     # Connect to the db in use for tool holders
47     db = c.ToolHolders
48     # Connect to the collection for chuck style tool holders
49     col = db.ChuckHolders
50
51     noDBtag = None
52     lastTag = None
53
54     """
55     *****
56     ***** Bridge Put *****
57     *****
58     """
59
60     """
61     For now, I am sending:
62     !DATA READY FLAG!
63
64     Tag (I KNOW THIS IS REDUNDANT)
65     DWG
66     Length
67     Used
68     """
69     def sendTag(tag):
70         dbInfo = col.find_one( {'_id': str(tag)} )
71
72         #tag
73         bc.put("tagData", str(dbInfo['_id']))
74         #dwg
75         bc.put("dwgData", str(dbInfo['dwg']))
76         #len
77         bc.put("lenData", str(dbInfo['len']))
78         #used
79         bc.put("usedData", str(dbInfo['used']))
80         #ready
81         bc.put("ready", "Y")
82
83     """
84     *****

```

```

86 ***** Requests *****
87 *****
88 """
90 def reqTag():
91     global lastTag, bc
92     tag = str(bc.get('tag'))
93     if(lastTag == tag):
94         pass
95     else:
96         sendTag(tag)
97         lastTag = tag
98
100 rt = RepeatedTimer(2, reqTag) # it auto-starts, no need of rt.start()
101
102 """
103 *****
104 ***** Main *****
105 *****
106 """
107
108 def main():
109     while 1:
110         time.sleep(1)
111
112 if __name__ == "__main__":
113     try:
114         main()
115     finally:
116         # Stop the timer
117         rt.stop()
118         print "Exiting safely..."

```

code/portable_yun4/portable_ctrl.py

F. Web Code and Material

The following is all of the PHP and HTML code needed to run the webserver along with the supporting images used. The webserver was WAMP 2.4 which uses Apache 2.4.4 and PHP 5.4.12. The only additional requirement is the MongoDB PHP extension.

F.1. Home Page

```
1 <html>
  <head>
3 <title>GE – View Chuck Holders</title >
  <link rel="stylesheet" type="text/css" href="/styles/base.css">
5 <link rel="shortcut icon" href="/favicon.ico" />
  </head>
7 <body>
9 <div align="center">
  <object type="image/svg+xml" data="/images/logo_vec.svg"></object>
11 </div>
  <br /><br />
13
15 <div align="center">
  <table class="tnob">
  <tr>
17
  <td class="tdnob">
19 <form action="/add.php">
    <input type="submit" value="Add">
21 </form>
  </td><td class="tdnob">
23 <form action="/search.php">
    <input type="submit" value="Search">
```

```

25 </form>
26 </td>
27 </td><td class="tdnob">
28 <form action="/management.php">
29     <input type="submit" value="Management">
30 </form>
31 </td>
32 </td>
33 </tr>
34 </table>
35 </div>
36
37 <br />
38
39 <div align="center">
40 <table>
41
42 <tr>
43     <th>RFID Tag</th>
44     <th>Drawing</th>
45     <th>End Mill Length</th>
46     <th>Location</th>
47     <th>Time Used</th>
48 </tr>
49
50 <?php
51 // Config
52 $dbhost = 'localhost';
53 $dbname = 'ToolHolders';
54
55 // Connect to test database
56 $m = new Mongo("mongodb://$dbhost");
57 $db = $m->$dbname;
58
59 // select the collection
60 $collection = $db->ChuckHolders;
61
62 // pull a cursor query
63 $cursor = $collection->find();
64
65 foreach($cursor as $document) {
66     echo '<tr>';
67     echo '<td><a href="/edit.php?_id=' . $document['_id'] . ">' .
68     $document['_id'] . '</a></td>';
69     echo '<td>' . $document['dwg'] . '</td>';
70     echo '<td>' . $document['len'] . '</td>';
71     echo '<td>' . $document['loc'] . '</td>';
72     echo '<td>' . round($document['used'], 3) . '</td>';
73     echo '<td><a href="/delete.php?_id=' . $document['_id'] . ">' .
74     'Delete' . '</a></td>';
75     echo '</tr>';
76 }

```



```
75 ?>
77
79 </table>
81 </div>
</body>
</html>
```

www/view.php

F.2. Management Page

This file allows for editing of the dropdown boxes that allow selection of location and drawing model. The only thing missing is just adding code to allow editing the second file which is as easy as making a second page and changing the file which is edited.

```
<html>
2 <head>
  <title>GE – Management</title>
4 <link rel="stylesheet" type="text/css" href="/styles/base.css">
  <link rel="shortcut icon" href="/favicon.ico" />
6 </head>
  <body>
8
  <div align="center">
10 <object type="image/svg+xml" data="/images/logo_vec.svg"></object>
  </div>
12 <br /><br />
14 <div align="center">
  <table class="tnob">
16 <tr>
  <td class="tdnob">
18 <form action="/view.php">
    <input type="submit" value="Home">
20 </form>
  </td>
22 </tr>
  </table>
24 </div>
26
  <div align="center">
```

```

28 <?php
30 $fn = "txts/locs.txt";
32 if (isset($_POST['content'])){
34     $content = stripslashes($_POST['content']);
36     $fp = fopen($fn,"w") or die ("Error opening file in write mode!");
38     fputs($fp,$content);
40     fclose($fp) or die ("Error closing file!");
42 }
44 ?>
46 <table>
48 <tr>
50 <td>
52 <form action="<?php echo $_SERVER["PHP_SELF"] ?>" method="post">
54     <textarea rows="25" cols="40" name="content"><?php
56     if(file_exists($fn)) {
58         readfile($fn);
60     }
62     else echo "Error - Please contact an admin!"; #the text file doesn't
64     exist
66     ?></textarea>
    </td></tr>
    <tr><td>
        <input type="submit" value="Save">
    </td></tr>
</form>
</td></tr>
</table>
</div>
</body>
</html>

```

www/management.php

F.3. Adding a Tool Holder

F.3.1. User End

```

1 <html>
2 <head>
3 <title>GE – Add a Chuck Holder</title>
4 <link rel="stylesheet" type="text/css" href="/styles/base.css">
5 <link rel="shortcut icon" href="/favicon.ico" />
6 </head>
7
8 <body>
9
10 <div align="center">
11 <object type="image/svg+xml" data="/images/logo_vec.svg"></object>
13 </div>
14 <br /><br />
15
16 <div align="center">
17 <table class="tnob">
18 <tr>
19 <td class="tdnob">
20 <form action="/view.php">
21 <input type="submit" value="Home">
22 </form>
23 </td>
24 </tr>
25 </table>
26 </div>
27
28 <div align="center">
29 <form action="/insert.php" method="post">
30 <table class="tnob">
31 <tr><td class="tdnob">RFID Tag: </td><td class="tdnob"><input type="text"
32 name="_id" <?php if(isset($_GET['_id'])) echo 'value="' . $_GET['_id']
33 . '"'; ?></td></tr>
34 <tr><td class="tdnob">Drawing #: </td><td class="tdnob">
35
36 <select name="dwg">
37 <?php
38 $path = "./txts/dwgs.txt";
39 $file = fopen($path, 'r');
40 $data = fread($file, filesize($path));
41 fclose($file);
42
43 $lines = explode(PHP_EOL, $data);
44 foreach($lines as $line) {
45     if($line) {
46         echo '<option value="' . urlencode($line) . '">' . $line . '</
47 option>';
48     }
49     else {
50         continue;
51     }
52 }
53 </select>
54 </div>

```

```

48     }
49     }
50 ?>
51 </select >
52
53 </td></tr>
54 <tr><td colspan="2" class="tdnob"><input type="submit" value="Submit
    Change"></td></tr>
55 </form>
56 </table>
57 </div>
58
59 </body>
60 </html>

```

www/add.php

F.3.2. Back End

```

1 <html>
2 <head>
3 <title>GE - Added!</title >
4 <link rel="stylesheet" type="text/css" href="/styles/base.css">
5 <link rel="shortcut icon" href="/favicon.ico" />
6 </head>
7
8 <body>
9
10 <div align="center">
11 <object type="image/svg+xml" data="/images/logo_vec.svg"></object >
13 </div>
14 <br /><br />
15
16 <div align="center">
17 Successfully Added!
18
19 <table>
20
21 <tr>
22 <th>RFID Tag</th>
23 <th>End Mill Length</th>
24 <th>Location</th>
25 <th>Time Used</th>
26 </tr>
27 <?php
28 $connection = new MongoClient();
29 $collection = $connection->ToolHolders->ChuckHolders;

```

```

30 $doc = array(
    "_id" => $_POST["_id"],
32    "len" => 0,
    "loc" => "Changing",
34    "used" => 0,
    "next" => "None",
36    "past" => array(),
    "dwg" => urldecode($_POST["dwg"]),
38 );

40 $collection->insert( $doc );

42

44 echo '<tr>';
echo '<td>' . $doc['_id'] . '</td>';
46 echo '<td>' . $doc['len'] . '</td>';
echo '<td>' . $doc['loc'] . '</td>';
48 echo '<td>' . $doc['used'] . '</td>';
echo '</tr>';
50 ?>
</table>
52 </div>

54 <div align="center">
<form action="/view.php">
56     <input type="submit" value="Continue">
</form>
58 </div>

60 </body>
</html>

```

www/insert.php

F.4. Editing Data

F.4.1. User End

```

1 <html>
<head>
3 <title>GE - Edit an Entry</title>
<link rel="stylesheet" type="text/css" href="/styles/base.css">
5 <link rel="shortcut icon" href="/favicon.ico" />
</head>

```

```

7 <body>
9 <div align="center">
11 <object type="image/svg+xml" data="/images/logo_vec.svg"></object>
</div>
13 <br /><br />

15 <div align="center">
<table class="tnob">
17 <tr>
<td class="tdnob">
19 <form action="/view.php">
    <input type="submit" value="Home">
21 </form>
</td>
23 </tr>
<tr>
25 <td class="tdnob">
<?php
27 echo '<a href="/th_hist.php?start=&end=&.id=';
echo $_GET["_id"]';
29 echo '>'>';
echo "<button type='button'> History </button>";
31 echo "</a>";
?>
33 </td>
</tr>
35 </table>
</div>

37 <div align="center">
39 Editing Tag: <?php echo $_GET['_id']; ?>
</div>
41 <br />

43 <?php
// Config
45 $dbhost = 'localhost';
$dbname = 'ToolHolders';
47
// Connect to test database
49 $m = new Mongo("mongodb://$dbhost");
$db = $m->$dbname;
51
// select the collection
53 $collection = $db->ChuckHolders;

55 $tag = array( '_id' => $_GET['_id'] );
57 // pull a cursor query

```

```

$cursor = $collection->find($tag);
59
foreach($cursor as $document) {
61     $id = $document['_id'];
        $len = $document['len'];
63     $loc = $document['loc'];
        $used = $document['used'];
65     $dwg = $document['dwg'];
    }
67 ?>

69 <div align="center">
    <form action="/update.php" method="post">
71 <input type="hidden" name="_id" value="<?php echo $id; ?>">
    <table class="tnob">
73 <tr><td class="tdnob">Drawing #:</td><td class="tdnob">

75 <select name="dwg">
    <?php
77     //Pull all the possible drawing models from a text file
        $path =    "./txts/dwgs.txt";
79     $file =    fopen($path, 'r');
        $data =    fread($file, filesize($path));
81     fclose($file);

83     $lines = explode(PHP_EOL, $data);
        foreach($lines as $line) {
85         if($line) {
            if($line == $dwg) {
87                 echo '<option selected="selected" value="' . urlencode($line) .
                "">' . $line . '</option>';
            }
89             else {
                echo '<option value="' . urlencode($line) . "">' . $line . '</
option>';
91             }
            }
93         else {
            continue;
95         }
        }
97 ?>
</select>
99
</td></tr><tr><td class="tdnob">Length:</td><td class="tdnob"> <input type
="text" name="len" value="<?php echo $len; ?>"></tr></td>
101 <tr><td class="tdnob">Location:</td><td class="tdnob">

103 <select name="loc">
    <?php
105     //Pull all the possible locations from a text file
        $path =    "./txts/locs.txt";

```

```

107     $file      =    fopen($path, 'r');
108     $data     =    fread($file, filesize($path));
109     fclose($file);

111     $lines    =    explode(PHP_EOL, $data);
112     foreach($lines as $line) {
113         if($line) {
114             if($line == $loc) {
115                 echo '<option selected="selected" value="' . urlencode($line) .
116                 '>' . $line . '</option>';
117             }
118             else {
119                 echo '<option value="' . urlencode($line) . '>' . $line . '</
120                 option>';
121             }
122             else {
123                 continue;
124             }
125         }
126     }
127 </select>
128 </tr></td>
129 <tr><td class="tdnob">Time Used:</td><td class="tdnob"> <input type="text"
130     name="used" value="<?php echo $used; ?>"></tr></td>
131 <tr><td class="tdnob" colspan="2"><input type="submit" value="Change Tag"
132     ></td></tr>
133 </table>
134 </form>
135 </div>
136 </body>
137 </html>

```

www/edit.php

F.4.2. Back End

```

1 <?php
2 $id = array("_id" => $_POST["_id"]);
3 $dwg = array("dwg" => urldecode($_POST["dwg"]));
4 $len = array("len" => (float) $_POST["len"]);
5 $loc = array("loc" => urldecode($_POST["loc"]));
6 $used = array("used" => (float) $_POST["used"]);
7
8

```



```

10 // Config
   $dbhost = 'localhost';
12 $dbname = 'ToolHolders';

14 // Connect to test database
   $m = new Mongo("mongodb://$dbhost");
16 $db = $m->$dbname;

18 // select the collection
   $collection = $db->ChuckHolders;

20
   $collection -> update($id, array('$set' => $dwg));
22 $collection -> update($id, array('$set' => $len));
   $collection -> update($id, array('$set' => $loc));
24 $collection -> update($id, array('$set' => $used));

26 header('Location: /view.php');

28 ?>

```

www/update.php

F.5. Deleting Data

```

<?php
2 // Config
   $dbhost = 'localhost';
4 $dbname = 'ToolHolders';

6 // Connect to test database
   $m = new Mongo("mongodb://$dbhost");
8 $db = $m->$dbname;

10 // select the collection
   $collection = $db->ChuckHolders;

12
   $id = array("_id" => $_GET["_id"]);
14
   $collection -> remove($id, array('justOne' => true));
16
   header('Location: /view.php');

18
   ?>

```

www/delete.php

F.6. Searching

F.6.1. User End

```
1 <html>
2 <head>
3 <title>GE – View Chuck Holders History</title>
4 <link rel="stylesheet" type="text/css" href="/styles/base.css">
5 <link rel="shortcut icon" href="/favicon.ico" />
6 <link rel="stylesheet" type="text/css" href="/styles/calendar.css">
7 <script type="text/javascript" src="/scripts/calendar.js"> </script>
8 <script type="text/javascript">
9     function init () {
10         calendar.set("start");
11         calendar.set("end");
12     }
13 </script>
14
15 </head>
16 <body onload="init()">
17
18
19 <div align="center">
20 <object type="image/svg+xml" data="/images/logo_vec.svg"></object>
22 </div>
23 <br /><br />
24
25 <div align="center">
26 <table class="tnob">
27 <tr>
28 <td class="tdnob">
29 <form action="/view.php">
30     <input type="submit" value="Home">
31 </form>
32 </td>
33 </tr>
34 </table>
35 </div>
36
37 <div align="center">
38 What do you want to search by?
39 <br />
40 <table class="tnob">
41 <tr>
42 <form action="/sresults.php" method="get">
43 <td class="tdnob">Location: </td><td class="tdnob">
```

```

45 <select name="loc">
    <?php
47     $path  =  "./txts/locs.txt";
        $file  =  fopen($path, 'r');
49     $data  =  fread($file, filesize($path));
        fclose($file);

51     $lines = explode(PHP_EOL, $data);

53     echo '<option value=""> </option>';
55     foreach($lines as $line) {
        if($line) {
57         echo '<option value="' . urlencode($line) . '">' . $line . '</
option>';
        }
59         else {
            continue;
61         }
        }
63 ?>
</select>

65 </td>
67 <td class="tdnob"> </td>
</tr>
69 <tr>
<form action="/sresults.php" method="get">
71 <td class="tdnob">Past Location: </td><td class="tdnob">

73 <select name="past">
    <?php
75     $path  =  "./txts/locs.txt";
        $file  =  fopen($path, 'r');
77     $data  =  fread($file, filesize($path));
        fclose($file);

79     $lines = explode(PHP_EOL, $data);

81     echo '<option value=""> </option>';
83     foreach($lines as $line) {
        if($line == 'Tool Crib' OR $line == 'Changing' OR $line == 'Floor'
or !$line) {
85         continue;
        }
87         else {
            echo '<option value="' . urlencode($line) . '">' . $line . '</
option>';
89         }
        }
91 ?>
</select>
93

```

```

95     </td>
96     <tr>
97     <td class="tdnob">Start Date: </td> <td class="tdnob"> <input id="start"
        name="start" type="text"> </td>
98     <td class="tdnob">End Date: </td> <td class="tdnob"> <input id="end"
        name="end" type="text"> </td>
99     </tr>
100 </tr>
101 <tr>
102 <form action="/sresults.php" method="get">
103 <td class="tdnob">Time Used: </td><td class="tdnob">
        <input type="radio" name="comp" value="gt"> >=
105     <input type="radio" name="comp" value="lt"> <
        <input type="text" name="used"> </td>
107 </tr>
108 <tr><td class="tdnob">Drawing #:</td><td class="tdnob">
109 <select name="dwg">
110 <?php
111     $path    =    "./txts/dwgs.txt";
112     $file    =    fopen($path, 'r');
113     $data    =    fread($file, filesize($path));
114     fclose($file);
115
116     $lines   =    explode(PHP_EOL, $data);
117
118     echo '<option value=""> </option>';
119     foreach($lines as $line) {
120         if($line) {
121             echo '<option value="' . urlencode($line) . '">' . $line . '</
option>';
122         }
123         else {
124             continue;
125         }
126     }
127 >?>
128 </select>
129 </td></tr>
130 <tr>
131 <td colspan="2" class="tdnob">
        <input type="submit" value="Search"> </td>
132 </td>
133 </tr>
134 </form>
135 </table>
136 </div>
137 </body>
138 </html>

```

F.6.2. Back End

```
1 <html>
2 <head>
3   <meta charset="UTF-8" />
4   <meta name="google" content="notranslate">
5   <meta http-equiv="Content-Language" content="en" />
6   <title>GE - View Chuck Holders</title>
7   <link rel="stylesheet" type="text/css" href="/styles/base.css">
8   <link rel="shortcut icon" href="/favicon.ico" />
9 </head>
10 <body>
11
12 <div align="center">
13   <object type="image/svg+xml" data="/images/logo_vec.svg"></object>
15 </div>
16 <br /><br />
17
18 <div align="center">
19   <table class="tnob">
20     <tr>
21       <td class="tdnob">
22         <form action="/view.php">
23           <input type="submit" value="Home">
24         </form>
25       </td>
26     </tr>
27   </table>
28 </div>
29
30 <br />
31
32 <div align="center">
33   <?php
34   // Config
35   $dbhost = 'localhost';
36   $dbname = 'ToolHolders';
37
38   // Connect to test database
39   $m = new Mongo("mongodb://$dbhost");
40   $db = $m->$dbname;
```

```

42 // select the collection
   $collection = $db->ChuckHolders;
44
46 // pull a cursor query
   $cursor = $collection->find();
48 date_default_timezone_set('America/New_York');
50 function tableData() {
   echo '<table>';
52   echo '<tr>';
   echo '<th>RFID Tag</th>';
54   echo '<th>Drawing</th>';
   echo '<th>End Mill Length</th>';
56   echo '<th>Location</th>';
   echo '<th>Time Used</th>';
58   echo '</tr>';
   }
60
62 function tablePast() {
   echo '<table>';
   echo '<tr>';
64   echo '<th>Tag</th>';
   echo '<th>Time In</th>';
66   echo '<th>Time Out</th>';
   echo '<th>End Mill Length</th>';
68   echo '</tr>';
   }
70
72 if (!empty($_GET["loc"])){
   $sterm = "loc";
74   tableData();
   }
76 elseif (!empty($_GET["past"])){
   $sterm = "past";
78   echo "<h1>Machine: " . $_GET["past"] . "</h1><br />";
   tablePast();
80 }
82 elseif (!empty($_GET["used"])){
   $sterm = "used";
   tableData();
84 }
86 elseif (!empty($_GET["dwg"])){
   $sterm = "dwg";
   tableData();
88 }
90 else{
   $sterm = "";
   tableData();
92 }

```

```

94 function dataRow($document){
    echo '<tr>';
96     echo '<td><a href="/edit.php?_id=' . $document['_id'] . ">' . $document
        ['_id'] . '</a></td>';
    echo '<td>' . $document['dwg'] . '</td>';
98     echo '<td>' . $document['len'] . '</td>';
    echo '<td>' . $document['loc'] . '</td>';
100    echo '<td>' . $document['used'] . '</td>';
    echo '<td><a href="/delete.php?_id=' . $document['_id'] . ">' . 'Delete
        ' . '</a></td>';
102    echo '</tr>';
    }
104
    $lastTag = "";
106
    function pastRow($tag, $job){
108         global $lastTag;
        $id = $tag;
110         #if($tag == $lastTag) $id = "";
        #else $lastTag = $tag;
112         echo '<tr>';
        echo '<td><a href="/th_hist.php?start=&end=&_id=' . $id . ">' . $id
            . '</a></td>';
114         echo '<td>' . date('D M d H:i:s Y', $job[0]) . '</td>';
        echo '<td>' . date('D M d H:i:s Y', $job[1]) . '</td>';
116         echo '<td>' . $job[2] . '</td>';
        echo '</tr>';
118    }

120 function filterDate($tag, $job){
    if(!empty($_GET["start"]) and empty($_GET["end"])){ //only start has a
        date
122         $start = strtotime( (string) $_GET["start"] );
        if($job[0] >= $start) pastRow($tag, $job);
124    }
    elseif(empty($_GET["start"]) and !empty($_GET["end"])){ //only end has a
        date
126         $end = strtotime( (string) $_GET["end"] );
        $end = $end + ((60 * 60 * 24) - 1); // make the cut-off time 12:59:59
128         if($job[0] <= $end) pastRow($tag, $job);
    }
130    else {
        $start = strtotime( (string) $_GET["start"] );
132         $end = strtotime( (string) $_GET["end"] );
        $end = $end + ((60 * 60 * 24) - 1); // make the cut-off time 12:59:59
134
        if($job[0] >= $start and $job[0] <= $end) pastRow($tag, $job);
136    }
    }
138
140 if($sterm == "loc" or $sterm == "dwg") {

```

```

142     foreach($cursor as $document) {
143         if(urldecode($_GET["loc"]) == $document['loc'] or urldecode($_GET["
144             dwg"]) == $document['dwg']) {
145             dataRow($document);
146         }
147     }
148
149     elseif($sterm == "past") {
150         foreach($cursor as $document) {
151             $past = $document['past'];
152             if(array_key_exists((string)$_GET["past"], $past)) {
153                 $arrKeys = array_keys($past);
154                 foreach($arrKeys as $key) {
155                     if($key == $_GET["past"]){
156                         foreach($past[$key] as $job){
157                             $tag = $document["_id"];
158                             if(empty($_GET["start"]) and empty($_GET["end"])) pastRow(
159                                 $tag, $job);
160                             else filterDate($tag, $job);
161                         }
162                     }
163                 }
164             }
165         }
166
167     elseif($sterm == "used") {
168         foreach($cursor as $document) {
169             if($_GET["comp"] == "lt") {
170                 if($document["used"] < (float) $_GET["used"]) {
171                     dataRow($document);
172                 }
173             }
174
175             if($_GET["comp"] == "gt") {
176                 if($document["used"] >= (float) $_GET["used"]) {
177                     dataRow($document);
178                 }
179             }
180         }
181     }
182
183     ?>
184 </table>
185 </div>
186
187 </body>
188 </html>

```

www/sresults.php

F.7. Finding the History

```
1 <html>
2 <head>
3 <title>GE - Added!</title>
4 <link rel="stylesheet" type="text/css" href="/styles/base.css">
5 <link rel="shortcut icon" href="/favicon.ico" />
6 </head>
7
8 <body>
9
10 <div align="center">
11 <object type="image/svg+xml" data="/images/logo_vec.svg"></object>
13 </div>
14 <br /><br />
15 <div align="center">
16 Successfully Added!
17
18 <table>
19
20 <tr>
21 <th>RFID Tag</th>
22 <th>End Mill Length</th>
23 <th>Location</th>
24 <th>Time Used</th>
25 </tr>
26 <?php
27 $connection = new MongoClient();
28 $collection = $connection->ToolHolders->ChuckHolders;
29
30 $doc = array(
31 " _id" => $_POST[" _id" ],
32 " len" => 0,
33 " loc" => "Changing" ,
34 " used" => 0,
35 " next" => "None" ,
36 " past" => array() ,
37 " dwg" => urldecode($_POST[" dwg" ] ) ,
38 );
39
40 $collection->insert( $doc );
41
42
43
44 echo '<tr>';
45 echo '<td>' . $doc[' _id ' ] . '</td>';
46 echo '<td>' . $doc[' len ' ] . '</td>';
47 echo '<td>' . $doc[' loc ' ] . '</td>';
```

```

49     echo '<td>' . $doc[ 'used ' ] . '</td>';
    echo '</tr>';
?>
51 </table>
</div>
53
<div align="center">
55 <form action="/view.php">
    <input type="submit" value="Continue">
57 </form>
</div>
59
</body>
61 </html>

```

www/insert.php

F.8. Reading from Administration Station

F.8.1. User End

```

1 <html>
  <head>
3 <title>GE – Admin Station</title>
  <link rel="stylesheet" type="text/css" href="/styles/base.css">
5 <link rel="shortcut icon" href="/favicon.ico" />
7
  </head>
9 <body>
11 <div align="center">
  <object type="image/svg+xml" data="/images/logo_vec.svg"></object>
13 </div>
  <br /><br />
15
  <div align="center">
17 <table class="tnob">
  <tr>
19 <td class="tdnob">
  <form action="/view.php">
21   <input type="submit" value="Home">
  </form>
23 <form action="/admin.php">
  <input type="submit" value="Refresh">

```

```

25 </form>
26 </td>
27 </tr>
28 </table>
29 </div>

31 <br />

33 <div align="center">

35 <?php
36 $path =    ".\/txts\/admin.txt";
37 $file =    fopen($path, 'r');
38 $data =    fread($file, filesize($path));
39 fclose($file);

41 // Config
42 $dbhost = 'localhost';
43 $dbname = 'ToolHolders';

45 // Connect to test database
46 $m = new Mongo("mongodb://$dbhost");
47 $db = $m->$dbname;

49 // select the collection
50 $collection = $db->ChuckHolders;

51 //id we are looking for
52 $id = array('_id' => $data);

55 // pull a cursor query, limit to 1 result for now
56 $cursor = $collection->find($id)->limit(1);

57

59 function foundTag($document){
60
61     echo '<table>';
62     echo '<tr>';
63     echo '<th>RFID Tag</th>';
64     echo '<th>Drawing</th>';
65     echo '<th>End Mill Length</th>';
66     echo '<th>Location</th>';
67     echo '<th>Time Used</th>';
68     echo '</tr>';

69

70     echo '<tr>';
71     echo '<td><a href="/edit.php?_id=' . $document['_id'] . '>' . $document
72         ['_id'] . '</a></td>';
73     echo '<td>' . $document['dwg'] . '</td>';
74     echo '<td>' . $document['len'] . '</td>';
75     echo '<td>' . $document['loc'] . '</td>';
76     echo '<td>' . $document['used'] . '</td>';

```

```

77     echo '<td><a href="/delete.php?_id=' . $document['_id'] . "'>' . 'Delete
78     ' . '</a></td>';
79 }
80
81 function makeNew(){
82     global $data;
83     echo "Tag " . $data . " does not exist! <br />";
84     echo "Would you like to ";
85     echo "<a href='/add.php?_id=" . $data . "'>";
86     echo "add it now?";
87     echo "</a>";
88 }
89
90 $holder = $cursor->getNext();
91
92 if(!empty($holder)) {
93 //This doesn't need to be a loop obviously as we're only using 1 tag
94 //I made it as a loop for easier expansion in the future however as only
95 //a few changes are needed to print multiple things (namely foundTag())
96     foreach($cursor as $document) {
97
98
99         if($data == $document['_id']) {
100             foundTag($document);
101         }
102     }
103 }
104 else{
105     makeNew();
106 }
107
108 ?>
109
110
111 </div>
112
113 </body>
</html>

```

www/admin.php

F.8.2. Back End

```

<!-- This file is where the admin_ctrl.py file goes to write a text file
with

```

```

2 the tag that was just scanned in. The webpage can then read from this
   file and
   display the appropriate information in admin.php.
4 —>
6 <?php
   $dwgs = './txts/admin.txt';
8 $handle = fopen($dwgs, 'w') or die('Cannot open file: '.$dwgs);
   $data = $_GET["tag"];
10 fwrite($handle, $data);
   fclose($handle);
12 ?>

```

www/admin_write.php

F.9. Text Files

The following files are read from for the corresponding section headings. For the dropdown menus, each line break represents a new selectable option.

F.9.1. Admin Station

As per the code, this just has the last tag that was read in by the reader at the Administration Station.

```
0B008099FE
```

www/txts/admin.txt

F.9.2. Drawing Models

```

1 NIKKEN MCAT40-C1-85U
  NIKKEN MCAT40-C7/8-75US-P
3 NIKKEN MCAT40-C1-90-A
  NIKKEN MCAT40

```

www/txts/dwgs.txt

F.9.3. Machines and Other Locations

```
1 Tool Crib
2 Changing
  Floor
4 A
  B
6 C
  D
8 E
  F
10 G
12 5917
13 5919
14 5920
  5925
  6210
```

www/txts/locs.txt

F.10. Calendar Script

```
1 /**
2  * Calendar Script
3  * Creates a calendar widget which can be used to select the date more
  * easily than using just a text box
4  * http://www.openjs.com/scripts/ui/calendar/
5  *
6  * Example:
7  * <input type="text" name="date" id="date" />
8  * <script type="text/javascript">
9  *   calendar.set("date");
10 * </script>
11 */
12 calendar = {
13   month_names: ["January", "February", "March", "April", "May", "June", "July", "
    August", "September", "October", "November", "December"],
14   weekdays: ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"],
15   month_days: [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31],
16   //Get today's date - year, month, day and date
17   today : new Date(),
18   opt : {},
19   data: [],
20
21   //Functions
```

```

23  /// Used to create HTML in a optimized way.
    wrt: function(txt) {
25      this.data.push(txt);
    },

27  /* Inspired by http://www.quirksmode.org/dom/getstyles.html */
    getStyle: function(ele, property){
29      if (ele.currentStyle) {
          var alt_property_name = property.replace(/\-(\w)/g, function(m,c){
31          return c.toUpperCase();}); //background-color becomes backgroundColor
          var value = ele.currentStyle[property] || ele.currentStyle[
            alt_property_name];

33      } else if (window.getComputedStyle) {
          property = property.replace(/([A-Z])/g, "-$1").toLowerCase(); //
35          backgroundColor becomes background-color

          var value = document.defaultView.getComputedStyle(ele, null).
37          getPropertyValue(property);
        }

39      //Some properties are special cases
        if(property == "opacity" && ele.filter) value = (parseFloat( ele.
41        filter.match(/opacity\=([^\s]*)/)[1] ) / 100);
        else if(property == "width" && isNaN(value)) value = ele.clientWidth
43        || ele.offsetWidth;
        else if(property == "height" && isNaN(value)) value = ele.clientHeight
45        || ele.offsetHeight;
        return value;
    },
47  getPosition: function(ele) {
        var x = 0;
        var y = 0;
        while (ele) {
49          x += ele.offsetLeft;
          y += ele.offsetTop;
51          ele = ele.offsetParent;
        }
53        if (navigator.userAgent.indexOf("Mac") != -1 && typeof document.body.
            leftMargin != "undefined") {
55          x += document.body.leftMargin;
          scrollTop += document.body.scrollTop;
        }

57        var xy = new Array(x,y);
59        return xy;
    },
61  /// Called when the user clicks on a date in the calendar.
    selectDate: function(year, month, day) {
63        var ths = _calendar_active.instance;
        if(ths.opt['onDateSelect']) ths.opt['onDateSelect'].apply(ths, [year,
            month, day]); // Custom handler if the user wants it that way.

```

```

65     else {
        document.getElementById(this.opt["input"]).value = year + "-" + month
        + "-" + day; // Date format is :HARDCODE:
67     this.hideCalendar();
    }
69 },
    /// Creates a calendar with the date given in the argument as the
    selected date.
71 makeCalendar:function(year, month, day) {
    year = parseInt(year);
73     month= parseInt(month);
    day = parseInt(day);
75
    //Display the table
77     var next_month = month+1;
    var next_month_year = year;
79     if(next_month>=12) {
        next_month = 0;
81     next_month_year++;
    }
83
    var previous_month = month-1;
85     var previous_month_year = year;
    if(previous_month< 0) {
87     previous_month = 11;
        previous_month_year--;
89     }

91     this.wrt("<table>");
    this.wrt("<tr><th><a href='javascript:calendar.makeCalendar(\"+(
    previous_month_year)+\", \"+(previous_month)+\"');' title='\"+this.
    month_names[previous_month]+\" \"+(previous_month_year)+\"'>&lt;</a></th>\"
    );
93     this.wrt("<th colspan='5' class='calendar-title'><select name='
    calendar-month' class='calendar-month' onChange='calendar.makeCalendar(
    "+year+\",this.value);'>");
    for(var i in this.month_names) {
95     this.wrt("<option value='"+i+"'");
        if(i == month) this.wrt(" selected='selected'");
97     this.wrt(">"+this.month_names[i]+"</option>");
    }
99     this.wrt("</select>");
    this.wrt("<select name='calendar-year' class='calendar-year' onChange
    ='calendar.makeCalendar(this.value, "+month+\"');'>");
101     var current_year = this.today.getYear();
    if(current_year < 1900) current_year += 1900;
103
    for(var i=current_year-70; i<current_year+10; i++) {
105     this.wrt("<option value='"+i+"'");
        if(i == year) this.wrt(" selected='selected'");
107     this.wrt(">"+i+"</option>");
    }
}

```



```

109     this.wrt("</select></th>");
    this.wrt("<th><a href='javascript:calendar.makeCalendar("+
next_month_year)+"','"+(next_month)+"');' title='"+this.month_names[
next_month]+" "+(next_month_year)+"'>&gt;</a></th></tr>");
111     this.wrt("<tr class='header'>");
    for(var weekday=0; weekday<7; weekday++) this.wrt("<td>"+this.weekdays
[weekday]+"</td>");
113     this.wrt("</tr>");

115     //Get the first day of this month
    var first_day = new Date(year,month,1);
117     var start_day = first_day.getDay();

119     var d = 1;
    var flag = 0;

121

123     //Leap year support
    if(year % 4 == 0) this.month_days[1] = 29;
    else this.month_days[1] = 28;

125

127     var days_in_this_month = this.month_days[month];

129     //Create the calender
    for(var i=0;i<=5;i++) {
        if(w >= days_in_this_month) break;
131         this.wrt("<tr>");
        for(var j=0;j<7;j++) {
133             if(d > days_in_this_month) flag=0; //If the days has overshooted
the number of days in this month, stop writing
                else if(j >= start_day && !flag) flag=1;//If the first day of this
month has come, start the date writing

135
                if(flag) {
137                     var w = d, mon = month+1;
                    if(w < 10) w = "0" + w;
139                     if(mon < 10)mon = "0" + mon;

141                     //Is it today?
                    var class_name = '';
143                     var yea = this.today.getYear();
                    if(yea < 1900) yea += 1900;

145
                    if(yea == year && this.today.getMonth() == month && this.today.
getDate() == d) class_name = " today";
                    if(day == d) class_name += " selected";

147
                    class_name += " " + this.weekdays[j].toLowerCase();

149
                    this.wrt("<td class='days"+class_name+" '><a href='javascript:
calendar.selectDate(\""+year+"\",\""+mon+"\",\""+tw+"\") '>"+tw+"</a></td>
");
151                     d++;

```

```

153     } else {
154         this.wrt("<td class='days'>&nbsp;&nbsp;&nbsp;</td>");
155     }
156 }
157 this.wrt("</tr>");
158 }
159 this.wrt("</table>");
160 this.wrt("<input type='button' value='Cancel' class='calendar-cancel'
161 onclick='calendar.hideCalendar();' />");
162
163 document.getElementById(this.opt['calendar']).innerHTML = this.data.
164 join("");
165 this.data = [];
166 },
167
168 // Display the calendar - if a date exists in the input box, that will
169 // be selected in the calendar.
170 showCalendar: function() {
171     var input = document.getElementById(this.opt['input']);
172
173     //Position the div in the correct location...
174     var div = document.getElementById(this.opt['calendar']);
175
176     if(this.opt['display_element']) var display_element = document.
177     getElementById(this.opt['display_element']);
178     else var display_element = document.getElementById(this.opt['input']);
179
180     var xy = this.getPosition(display_element);
181     var width = parseInt(this.getStyle(display_element, 'width'));
182     div.style.left=(xy[0]+width+10)+"px";
183     div.style.top=xy[1]+"px";
184
185     // Show the calendar with the date in the input as the selected date
186     var existing_date = new Date();
187     var date_in_input = input.value;
188     if(date_in_input) {
189         var selected_date = false;
190         var date_parts = date_in_input.split("-");
191         if(date_parts.length == 3) {
192             date_parts[1]--; //Month starts with 0
193             selected_date = new Date(date_parts[0], date_parts[1], date_parts
194 [2]);
195         }
196         if(selected_date && !isNaN(selected_date.getYear())) { //Valid date.
197             existing_date = selected_date;
198         }
199     }
200
201     var the_year = existing_date.getYear();
202     if(the_year < 1900) the_year += 1900;
203     this.makeCalendar(the_year, existing_date.getMonth(), existing_date.
204 getDate());

```

```

199     document.getElementById(this.opt['calendar']).style.display = "block";
200     _calendar_active_instance = this;
201 },
202
203 /// Hides the currently show calendar.
hideCalendar: function(instance) {
204     var active_calendar_id = "";
205     if(instance) active_calendar_id = instance.opt['calendar'];
206     else active_calendar_id = _calendar_active_instance.opt['calendar'];
207
208     if(active_calendar_id) document.getElementById(active_calendar_id).
style.display = "none";
209     _calendar_active_instance = {};
210 },
211
212 /// Setup a text input box to be a calendar box.
set: function(input_id, opt) {
213     var input = document.getElementById(input_id);
214     if(!input) return; //If the input field is not there, exit.
215
216     if(opt) this.opt = opt;
217
218     if(!this.opt['calendar']) this.init();
219
220     var ths = this;
221     if(this.opt['onclick']) input.onclick=this.opt['onclick'];
222     else {
223         input.onclick=function(){
224             ths.opt['input'] = this.id;
225             ths.showCalendar();
226         };
227     }
228 },
229
230 /// Will be called once when the first input is set.
init: function() {
231     if(!this.opt['calendar'] || !document.getElementById(this.opt['
calendar'])) {
232         var div = document.createElement('div');
233         if(!this.opt['calendar']) this.opt['calendar'] = 'calender_div_'+
Math.round(Math.random() * 100);
234
235         div.setAttribute('id',this.opt['calendar']);
236         div.className="calendar-box";
237
238         document.getElementsByTagName("body")[0].insertBefore(div,document.
getElementsByTagName("body")[0].firstChild);
239     }
240 }
241 }
242 }
243 }

```

www/scripts/calendar.js

F.11. CSS

F.11.1. Website

```
body{
2 color: white;
  background-color: gray;
4 }

6 table
  {
8 border-collapse: collapse;
  }

10 table,th, td
12 {
14 border: 1px solid white;
  text-align: center;
  padding: 5px;
16 }

18 .tnob
  {
20 border-collapse: collapse;
  border: 0px hidden;
  padding: 0px;
22 }

24 .tdnob
26 {
  border: 0px hidden;
28 }

30 a:link {
  color: white;
32 font-weight: bold;
  }

34 a:visited {
36 color: white;
  }

38 a:hover {
40 color: navy;
  }

42 a:active {
44 color: white;
  }
```

F.11.2. Calendar

```
1 .calendar-box {
2     display:none;
3     background-color:gray;
4     border:1px solid #444;
5     position:absolute;
6     width:250px;
7     padding: 0 5px;
8 }
9 .calendar-box select.calendar-month {
10     width:90px;
11 }
12 .calendar-box select.calendar-year {
13     width:70px;
14 }
15 .calendar-box .calendar-cancel {
16     width:100%;
17 }
18 .calendar-box table td {
19     width:14%;
20 }
21 .calendar-box .calendar-title {
22     text-align:center;
23 }
24 .calendar-box a {
25     text-decoration:none;
26 }
27 .calendar-box .today a {
28     padding:0 5px;
29     margin-left:-5px;
30     background-color:gray;
31     color:red;
32 }
33 .calendar-box .selected a {
34     padding:0 5px;
35     margin-left:-5px;
36     background-color:gray;
37 }
```

F.12. Images



Figure F.1.: Website banner



Figure F.2.: Website icon