IMPROVEMENTS IN FRICTION STIR WELDING CONTROL SYSTEM AND AN EXTREMUM CONTROLLER FOR TRACKING FRICTION STIR EXTRUSION PROCESS

By

Jay Thomas Reynolds

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Mechanical Engineering

December 16, 2017

Nashville, Tennessee

Approved:

Alvin M. Strauss, Ph.D.

George E. Cook, Ph.D.

ACKNOWLEGEMENTS

I want to thank Dr. Strauss and Dr. Cook for your guidance and support during this research and serving on my thesis committee. I also want to thank you both for giving me the opportunity to continue my education at Vanderbilt by making me a part of the VUWAL laboratory. As most research goes my work would not be possible without the foundation set up by the past graduate students, and I additionally want to thank my colleagues in the lab, Todd Evans, Brian Gibson, Kelsay Neely, Adam Jarrell, Jacob Matthews, and Devany Sweitzer, for your help and insights.

I would also like to thank the Vanderbilt staff and all my past professors. I want to thank the members of the ME office, Myrtle Daniels, Jean Miller, and Renee Tomlin for all your administrative help, and thanks to Linda Harris and Liz Leis for help formatting this thesis. Also, thank you Phil Davis, Bob Patchin, and John Fellestein for their machining expertise and troubleshooting on the lab's milling machine.

Lastly, I would like to thank Connie Thomas who not only cleaned the lab and building, but also on numerous late at nights let me inside when I was locked out and kept me supplied with peanut butter.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

NOMENCLATURE

Symbol	
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RPM	Rotations Per Minute
IPM	Inch Per Minute
FSW	Friction Stir Welding
FSE	Friction Stir Extrusion
GUI	Graphical User Interface
C#	Programing Language
ModBus	A Serial Communication Protocol
SLRT	Simulink Real-Time
NI	National Instruments
IO	Input Output
VUWAL	Vanderbilt University Welding and Automation Laboratory
FEA	Finite Element Analysis
Arduino	8 bit Microcontroller
SOIC	Small Outline Integrated Circuit
DAQ	Data Acquisition
PCB	Printed Circuit Board
KiCad	PCB design software
Eagle	PCB design software
PID	Proportional Integral and Derivative
PWM	Pulse Width Modulation

CHAPTER 1 - INTRODUCTION

Friction stir welding (FSW) is a maturing solid state joining process that has been adapted into numerous industries. The process does not involve melting of the base materials and avoids many of the traditional problems faced with fusion welding. In fact, in many cases the process refines the material and the welded joint can become stronger than the initial parent material. The key to the process is a rotating tool that is plunged into the pieces to be welded. The friction of the rotating tool generates heat and the materials becomes plasticized. The rotating tool also causes severe deformation of the plasticized materials, and this material is kept from exiting the weld area by a large amount of downward pressure supplied by the tool's shoulder. Then as the tool moves by the plasticized material reconsolidates welding the pieces together.

Force Measurement

The measurement of the process forces during FSW is important for many reasons. The simplest one would be avoiding damage to any equipment by overstressing them. Another is that the quality of the weld can be related to maintaining the process forces within certain bounds, and others have successfully used feedback control to maintain these measured forces [1] [2] [3]. Force measurement can also be used to detect the formation of flaws by examining the process forces both as they change in magnitude and in frequency [4]. Additional insight can be gained from instrumentation systems that allow for high frequency data acquisition. Balasubramanian gained better understanding on material flow and reconsolidation by examining the lateral and traverse forces as a function of the welding tool's angular position [5]. Force measurements along with other instrumentation are also key for validating the ongoing modeling effort to fully capture the FSW process.

Friction Stir Extrusion

Another capability of the FSW process is its ability to extrude material to fill voids. Evans has shown this process to be useful in creating novel ways to join dissimilar materials in a method called

Friction Stir Extrusion (FSE). This was done by creating features in the base material that once material is extruded into creates a mechanical interlock between the two materials. FSW tool is then used to extruded material to fill that feature. This was demonstrated first using a double-sided spot welding setup to create a rivet like structure for binding aluminum to steel [6], and this idea was progressed further to be combatable with linear welds by extruding aluminum into features like dove tails that were cut into the steel [7]. This process provides a way to join steel and aluminum in a mechanical interlock without investing in more expensive machines and tools necessary to weld steel.

Weld Tracking

Through the tool tracking is a process where the forces that the welding tool experiences are used as information to track a desirable weld feature. The idea was formulated after Fleming demonstrated the ability to estimate the amount of offset in a T joint using the measured signals from several welds after training a neural network [8]. He was then able to use this finding to design an extremum controller called Weavetrack to automatically track the weld in T joints [9]. The basic idea of Weavetack is to use a trapezoidal perturbation in the tools lateral position and then by monitoring the forces from the weld update the estimated center position of the T joint. This was possible as the T joint setup provided a local maximum in the axial force that the tool always tried to find, and this process can be seen in Figure 42. Fleming was also able to successfully apply the weave track method to track lap joints [10], and Gibson demonstrated the use of weave track to track applied blind sealant paths on lap joints for aerospace applications [11]. Tracking of these weld configurations is useful as the desired path is obscured by the top plate of the material, and methods of through the tool tracking provides the possibility to detect misalignment and perform path correction in real time.

CHAPTER 2 - VUWAL'S WELDING MACHINE

Introduction

In the past the code controlling the welding machine has all been done in C#. This language is great for making user interfaces, has large libraries supporting connections to peripherals, and has some multithreading capabilities built in for taking care of concurrent tasks. Students in the past have been able to develop some controls for the machine but most of what has been done are simple PID controllers. These can give satisfactory performance, however students faced a barrier in dealing with the C# to develop more complex controls. Not only did they have to be able to translate the controls they wanted into C# code, but they had to manage reading from sensors and sending motor commands. These all had their own respective delays to manage, so to simplify the control programming previous students only controlled one motor of the machine at a time to avoid making multithreaded programs for their control logic. To remedy this major modification to the welding systems software and hardware were changed. These changes also greatly lower the bar for more complex control work to be done in this lab in the future as well as improving the overall safety of the machine. The enhancement of the welding machines control platform and demonstration of its use is the focus of this thesis.

Welding Machine Overview

A WWII era Milwaukee milling machine serves as the base of the friction stir welder in the Vanderbilt Welding and Automation Laboratory (VUWAL). The milling machine has been extensively modified since its creation for FSW purposes which include the addition of controllers, motors and sensors to monitor and control the welding program. Most of these modifications have been covered in previously in past dissertations from the lab [2] [3], and most recently in master thesis including Paul Sinclair's and Brian Gibson's [12] [13]. Currently the changes that have been made to the welding system after it was last described by Gibson include the reintegration of the Kistler Rotating Cutting Force Dynamometer, the addition of magnetic angle sensors, a redundant system for emergency shut off, and the addition of a separate real-time computer for controlling welds and data acquisition. The software side

of the labs welding system was overhauled to provide better capabilities while improving user ability and safety. The actual logic for controlling a weld was given to a Simulink model running on a separate computer while the C# program primarily serves as a GUI and a conduit to route sensor data to and motor commands from the second computer. Simulink is a graphical programing language developed by Mathworks and is commonly used in teaching controls, including the courses at Vanderbilt. In Simulink systems are can be easily modeled with little user effort using its intuitive interface. The computer running the weld's Simulink model is running a Simulink Real-Time (SLRT) kernel which allows the code generated from the model to run with real time constraints. This allows data to be sampled, stored, processed, and new controls signals to be sent out in a uniform manner and at much higher speeds then what was possible before. An overview diagram of how information and commands are received and sent by the updated system can be seen in Figure 1. Detail on how these new systems were integrated can be seen in APPENDIX B, APPENDIX C, and APPENDIX D.

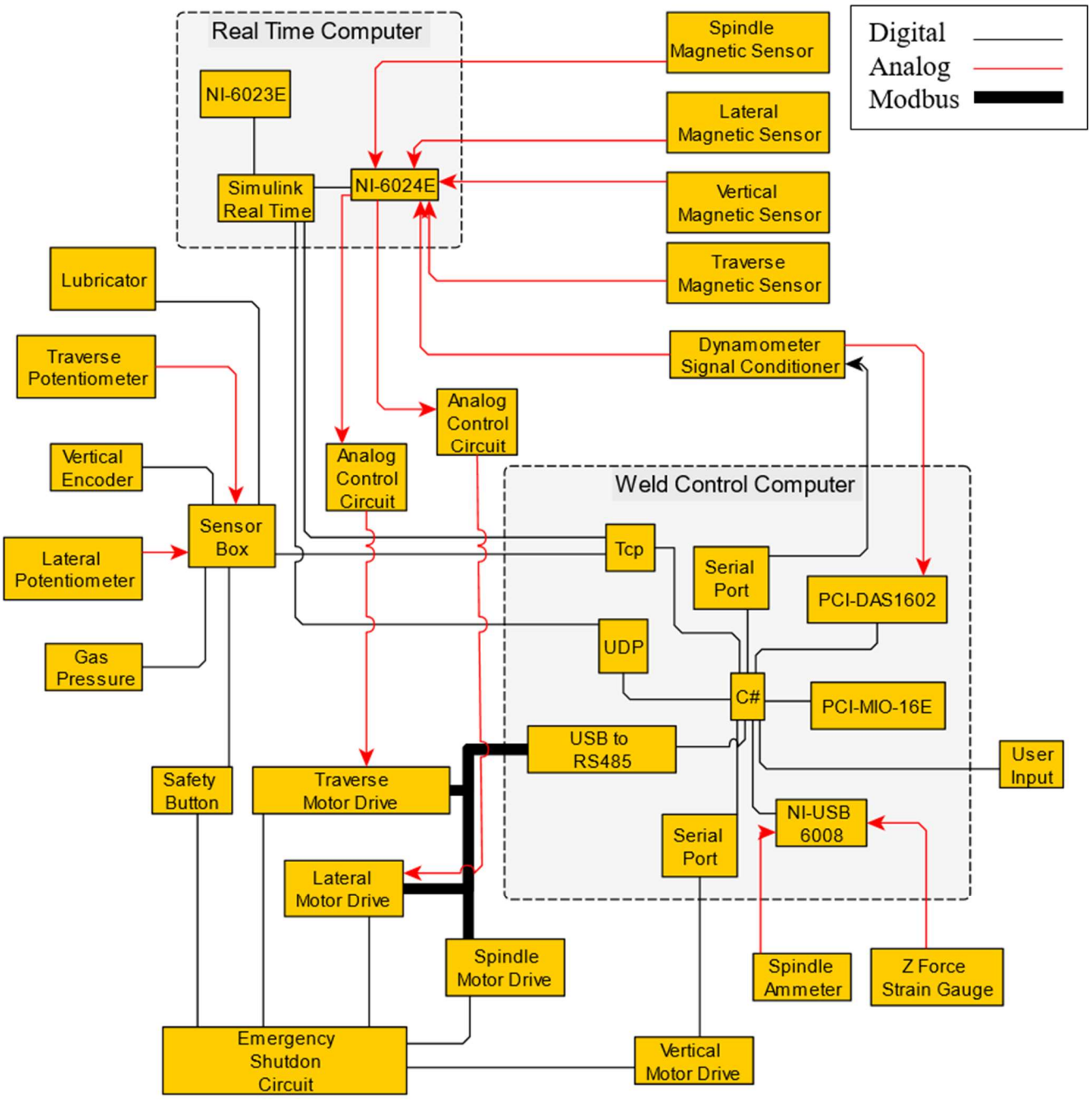


Figure 1: Block diagram showing how signals travel in the VUWAL's friction stir welding system

Weld Control computer

The Weld Control Computer is currently running windows 7 on an intel xenon quadcore that has hyperthreading. This is the computer that the user interacts with, and runs the C# program which is used

to interact seamlessly with the parts of the system connected to it. The program is multithreaded which allows different tasks to be run at different intervals or situations as needed. One of these threads is responsible with gathering sensor data, sending the data over UDP to the real-time computer, and then performing a safety check on the measured sensor data. Another thread handles the user interface. Other threads were created to handle communication to different shared resources using synchronization primitives. These threads include sending commands to the instance of Matlab running on the weld computer and over the shared bus that communicates with the lateral, traverse, and spindle motors. To help accomplish this two queues were created to store commands. These communication threads then simply wait for the queue to have something in it, and then they send out the commands in a timed manner to avoid overloading the shared communication bus. A simplified diagram of this can be seen in Figure 2 for dealing with motor commands. The same structure was made for routing commands to Matlab.

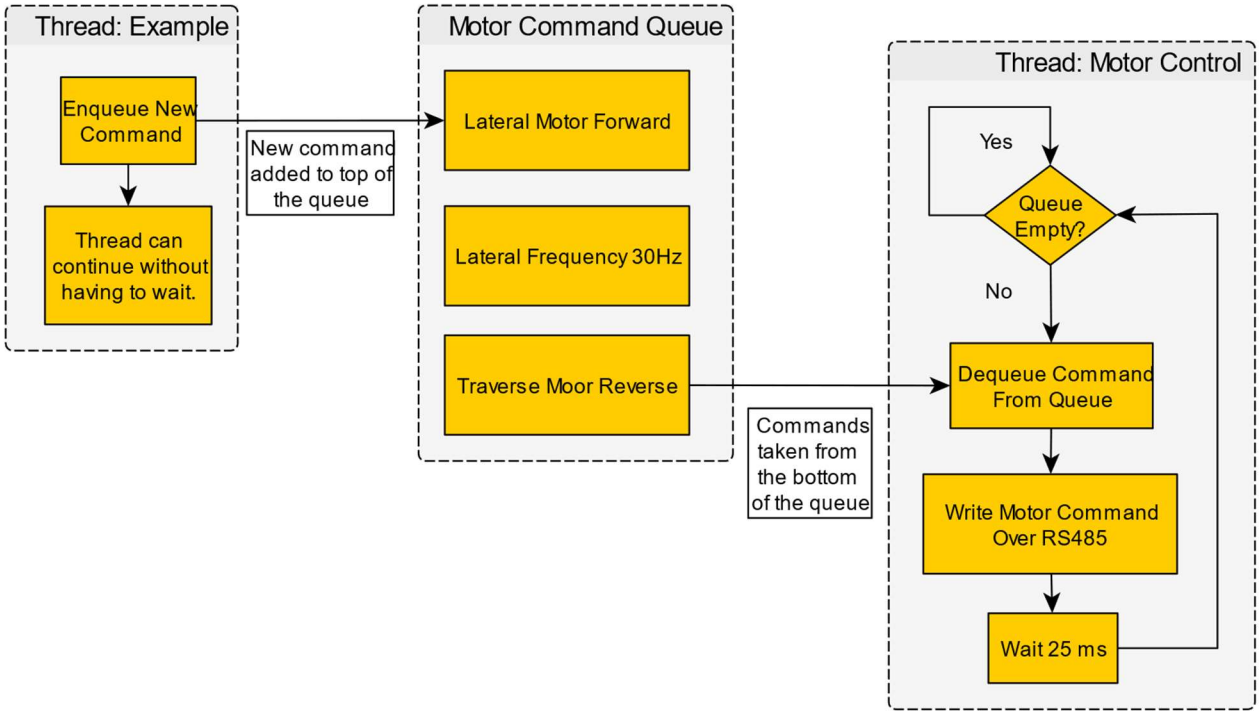


Figure 2: Diagram of how the command queue and motor control thread handle sending commands

This was needed as commands written to the shared bus too quickly would interfere with each other and cause messages to be lost. This system avoids this as commands are written out over the ModBus at the fastest rate possible without risking lost messages. In the actual code mutexes and semaphores were used to prevent deadlocks from occurring since multiple threads can access queue and only one thread at a time should be able to so. As a further precaution the final thread in the program is set up to be the only thread used to control the machine which is called the weld control thread. Its function changes based on the given commands, and this limits the system to only having one thread sending commands to the machine. This will avoid situations where two different threads compete for control which could lead to unstable and unpredictable behavior. This is also important for thread management so that if the user needs to abort a process the program knows what thread to kill which greatly improves the systems stability over time as in the past poor thread management caused system instability. The main GUI that the user interacts with can be seen in Figure 3 which shows the main window called the MainForm.

Figure 3: Mainform of the GUI created by the C# code running on the welding computer

The Mainform allows the user to control the welding system by connecting to various interfaces, manually control the motors, provide sensor information, and to present the user with a list of weld programs that have already been created. To perform a weld the user clicks on the word weld which can be seen on the upper tool strip of the MainForm. This brings down a dropdown menu that user can select the desired weld. This menu is auto populated from a list of text files that contain a list of welding parameters needed for that specific weld as well as the name of the corresponding Simulink file. This will allow researchers to design new weld controllers without ever having to code anything in C# or even close and recompile the C# program as changes to the weld parameter text files or the Simulink models can be modified and used independently.

Real-time computer

The “Real-Time Computer” was built to run the control logic during a weld and runs with an intel

Pentium G4600 and 8 gibibytes of RAM. This computer uses Matlab's Simulink Real-Time kernel which allows Simulink models to be compiled into code that is then run in real-time by this dedicated computer. A drawback of using this system that only a few IO boards are supported then what is available to C# running on a Windows system, and the cost of these boards can be expensive. The current IO board used in the computer is a used NI 6024E. This board can take up to 200 kS/s with 12 bit resolution on 16 analog inputs and two ±10 volt analog outputs. The computer also contains a NI 6023E DAQ card which can provide another 16 analog inputs for easy integration of new sensors in the future. As can be seen in Figure 1 the NI 6024E card reads in information from the dynamometer and magnetic angle sensors. The rest of the machines sensor data is feed to the real-time computer from the welding computer over UDP. This interaction between the two computers can be seen in Figure 4.

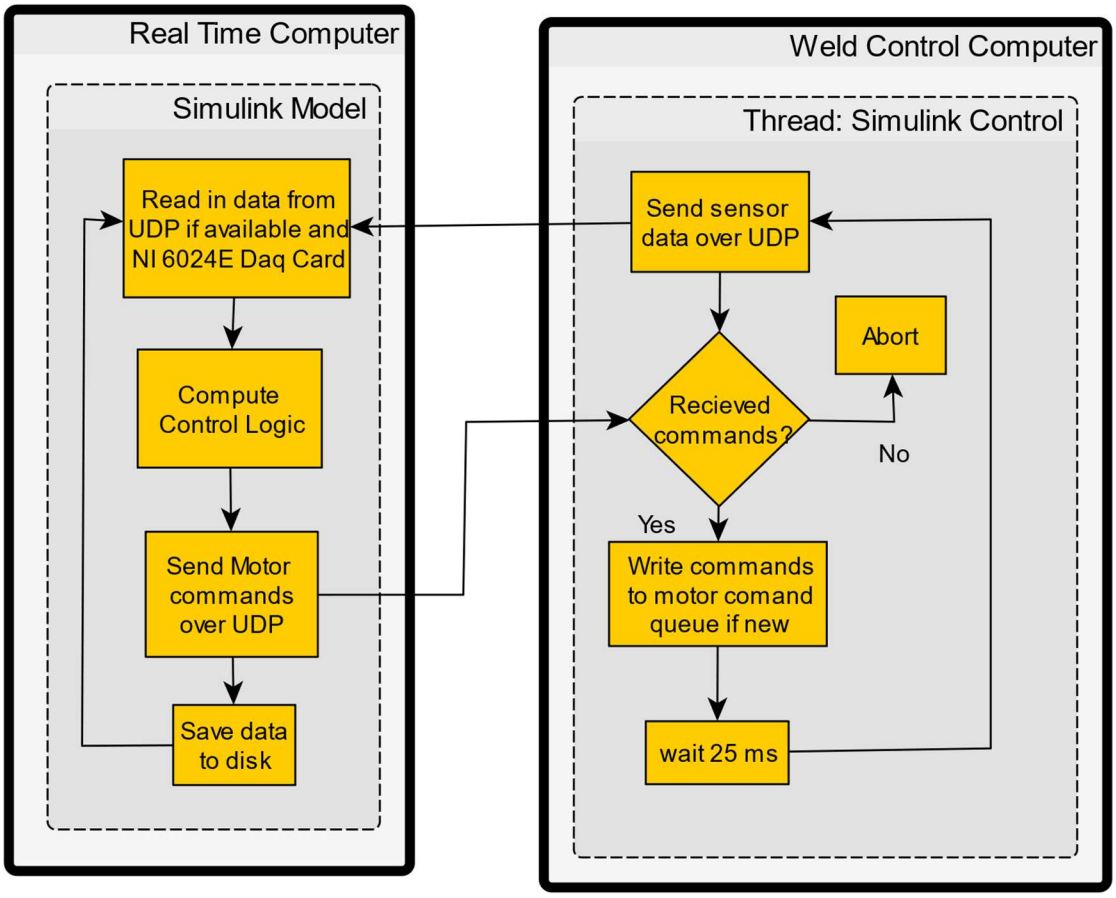


Figure 4: How the Weld control computer communicates with the Real-Time Computer during a weld.

A designed safety feature is that if the C# program on the welding computer detects that it has not received a command it will immediately abort the weld and shutdown the weld. Since the real-time computer runs at 5kHz the only reason a new command would not be received by the weld control computer is if an error had occurred somewhere in the system. After a weld is completed the Simulink model will signal the C# program that it is finished, and control of the motors will revert into manual control.

Example of Use

The creation of a new weld controller that works with this system involves the creation of two files. First a user needs to create a Simulink model for their weld controller and a corresponding text file. This text file is used by the C# code to add to the list of available welds and holds information used to

populate a new GUI for the weld. This window gives the user a place to input relevant parameters for the newly designed weld. The parameter values that the user inputs are then given over to an instance of Matlab running on the weld computer and then to the Simulink model on the real-time computer. An example text file for a simple linear weld opened in Excel can be seen in Figure 5.

	A	B	C	D	E	F
1	Simulink File					
2	LinearWeld.slx					
3						
4	Label Text	Matlab Parmater Name	Start Value	Minimum	Maximum	Number of Decimal Places
5	Plunge Depth(inch)	VerPlunge	0	0	0.3	5
6	Spindle cw RPM	SpiRPM	1000	-2000	2000	0
7	Traverse IPM	TraSpeed	0	0	10	1
8	Start Location	TraStart	9	0	30	3
9	End Location	TraEnd	9	0	30	3
10	Plunge Speed	PlungeSpeed	0.2	0	0.3	2
11	Tilt Angle	TiltAngle	0	-5	5	2

Figure 5: Text file to be used to generate GUI for a linear weld

This text file also sets maximal and minimal values for parameters. This helps avoid accidental input of parameters that could possibly damage the machine. Once the user selects a desired weld the C# program can parse in the data it needs from this text file. The result for how this information is transformed into an interface a user can interact with can be seen in the left side of Figure 6. In addition, the C# program will communicate with Matlab to open the matching Simulink model, compile it, and uploaded to the real-time computer. The Simulink model for a linear weld can be seen in the right side of Figure 6.

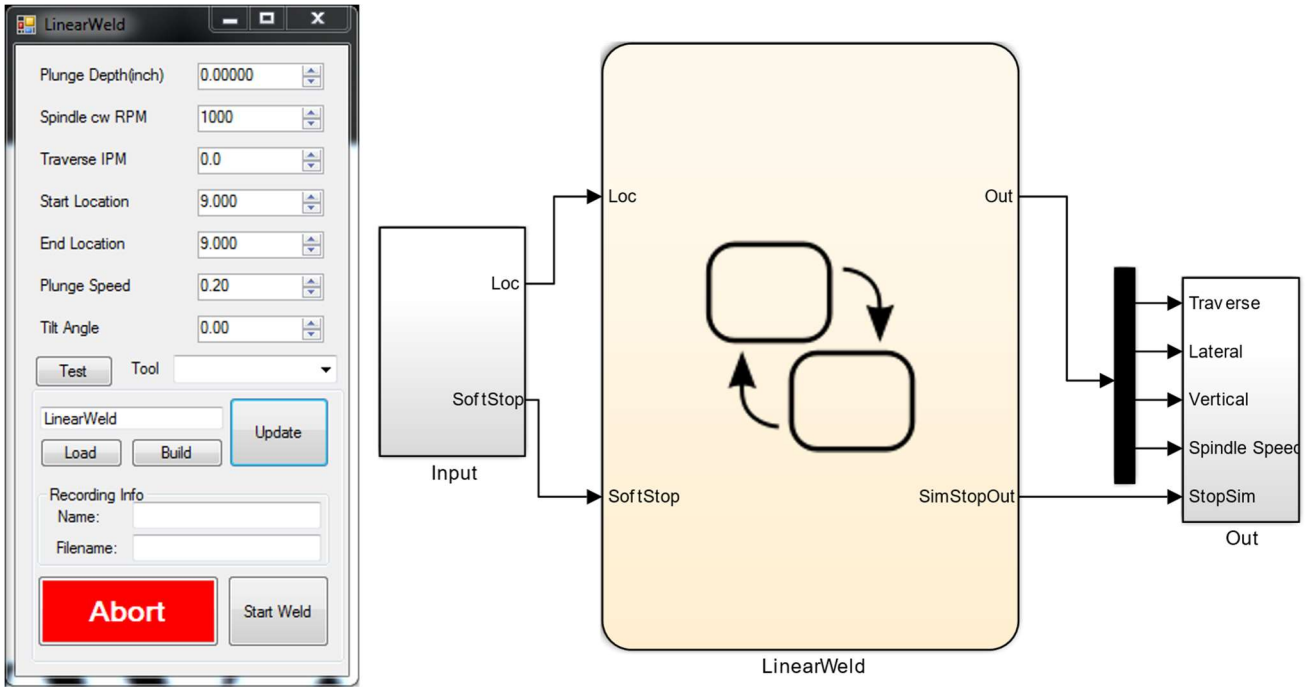


Figure 6: Form auto populated with desired inputs and Simulink model that uses those inputs for control

A simple linear weld is made by first plunging the tool to the desired depth with the desired rotation rate. The machine then welds at the desired traverse speed till the end location is reached. Once the desired welding parameters have been entered in the welding form, the start weld button is pressed to begin the weld. This will compile, upload and begin running the Simulink model to the real-time computer. In addition, a new thread in the C# code is created to route data to and from the real-time computer. This thread receives motor commands from the real-time computer using UDP and then sends the received commands to the vertical and spindle motors. The Simulink model for controlling this linear weld consists of two communication blocks that receive sensor data and send motor speed commands to the motors as well as performing a log of these signals. The middle block controls the logic behind controlling the weld and is a Stateflow block. Stateflow is Mathworks tool to develop and simulate logic in a state machine. The interior of this Stateflow block can be seen in Figure 7.

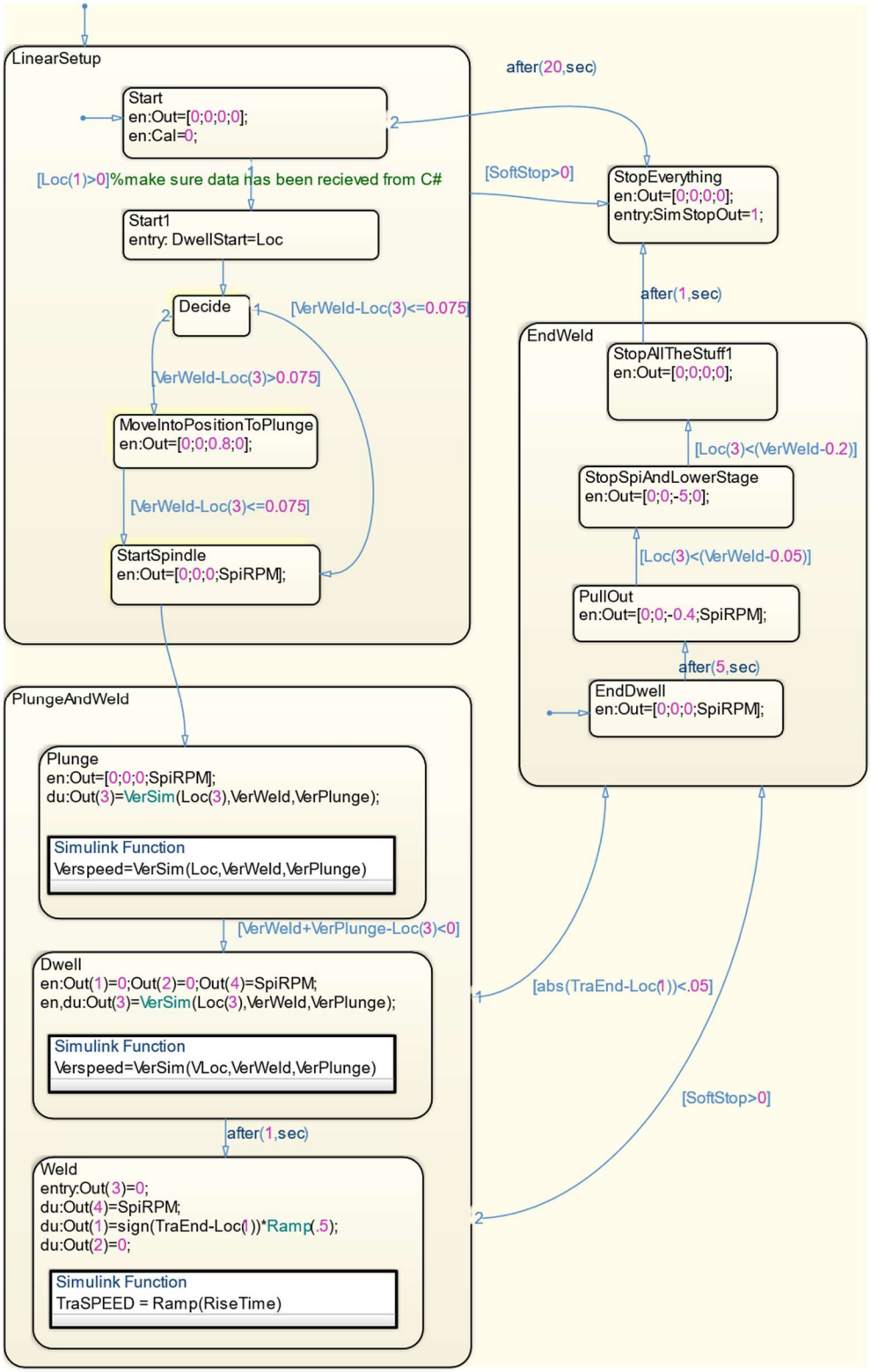


Figure 7: State machine logic to complete a simple linear weld.

This state machine guides the machine through the steps of the weld. These steps can be seen in the states shown in Figure 7, which include slowly plunging the tool into the material and dwelling for some time. Then traverse ramps up to the desired speed and once the end location is reached the tool dwells for a bit at the end of the weld. Finally, the tool is retracted from the welded sample. The state machine transitions from state to state once the logic condition governing the transition is met, and in addition allows Simulink functions to be used inside the states. The two Simulink functions used in this weld can be seen in Figure 8.

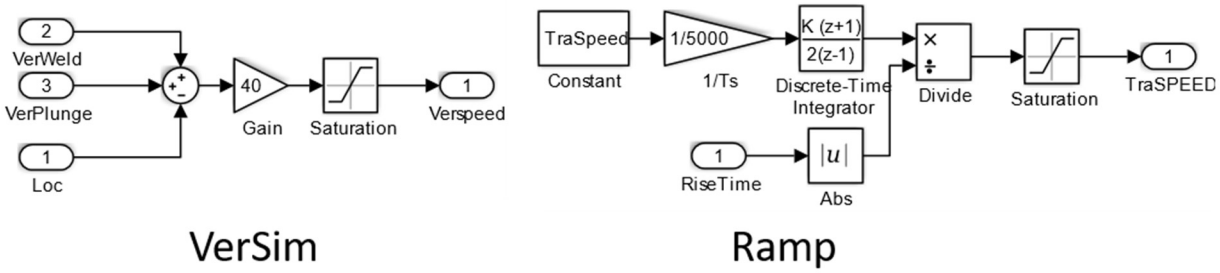


Figure 8: Simulink functions used in the linear weld state flow chart from Figure 7

The VerSim function is a proportional controller for achieving more accurate plunge depth, and the Ramp function is used to linearly increase the traverse speed to the desired speed. This is to avoid excessive force generation on the tool from sudden acceleration through the material. As can be seen later in Figure 32 a ramp time of 0.5 seconds appears to be too quick of an acceleration as a noticeable spike is seen in the X and Y forces of the weld. The soft stop transition seen in Figure 7 is a feature added to allow the user to immediately stop a weld and move to the retraction of the tool phase of the weld. This allows the user to stop an undesirable weld and safely retract the welding tool so it does not become stuck in the material which would otherwise happen during an emergency stop of the machine.

Demonstration of capabilities

This linear weld is a basic weld, however without too much effort it is not difficult to develop more complex welds using this system without a considerable time commitment. One such example was a

weld controller developed to follow an arbitrary path given the desired velocity and position of the tool at a given time. This was fed into a controller for the lateral and traverse motors given by equation

$$Speed_{Command} = Speed_{desired} + k * (Position_{desired} - Position_{measured}) \quad (1)$$

where the k is a gain multiplied by the error in the position to keep tool close to the desired position. Then for each time step of the controller a new desired speed and position was read into the controller from a predetermined path to be combined with new sensor information to update the commands given to the motors using equation (1). Then as numerous welds are made going through a collection of welding paths it is possible to write using friction stir welding in an automated fashion. The result of this process can be seen in Figure 9 and Figure 10.

Figure 9: Friction stir writing, VUWAL

Figure 10: Example of friction stir writing, vanity

Conclusion

By redesigning the software and hardware of the welding control system users now have a lower barrier to experiment with innovative ways to control and monitor the FSW process. The addition of a separate dedicated real-time computer allows for faster sampling rates, and the restructuring of the C# code provides a safe and cheap way to maintain compatibility with previous comment interfaces. This lower barrier for designing controllers was made possible due to the addition of a separate environment to create welding controllers in Simulink, and researchers can directly build off experiences they have from their control courses for FSW research.

CHAPTER 3 - SAFETY

Introduction

The FSW process is a relatively safe process without the use of extreme heat or use of toxic materials. However large forces and rotating machinery are present in the process which can pose a risk to the machine and its users if something were to go wrong. To help prevent this the welding machine has numerous sensors that can be used to automatically detect faults. Assuming the software or sensors are without error or infallible is dangerous as software can crash and parts of the system can break down. To help improve the safety of the machine improvements were made in the emergency stop button and the software fault detection.

Emergency Stop Button

Previously the signal from the weld machine's emergency stop button which can be seen in Figure 11 was fed into the sensor box. The sensor box then sent the status of the button over a TCP/IP connection to the welding computer which then would issue a stop command to all the motors if the button was pressed.

Figure 11: Emergency Stop Button

This was a safety hazard as if the weld computer, the sensor box, or the communication between them

was to experience an error while the motors were in motion the system would be unable to respond to the press of the emergency stop button. This would mean the operator of the machine would have to run and start flipping the main power switches to the motors to stop the machine before the machine damaged itself. The amount of time this could take could put both the welding machine in danger of damaging itself or the occupants of the lab. To fix this a more reliable way to react to emergencies was developed. This was done by designing a secondary system to shut down the machine's motor that is independent of the weld control computer or the sensor box. This system uses an Arduino Uno to monitor the signal from the emergency stop button and controls four different normally open relays connected to the different drive motors. A basic circuit diagram for this setup can be seen in Figure 12.

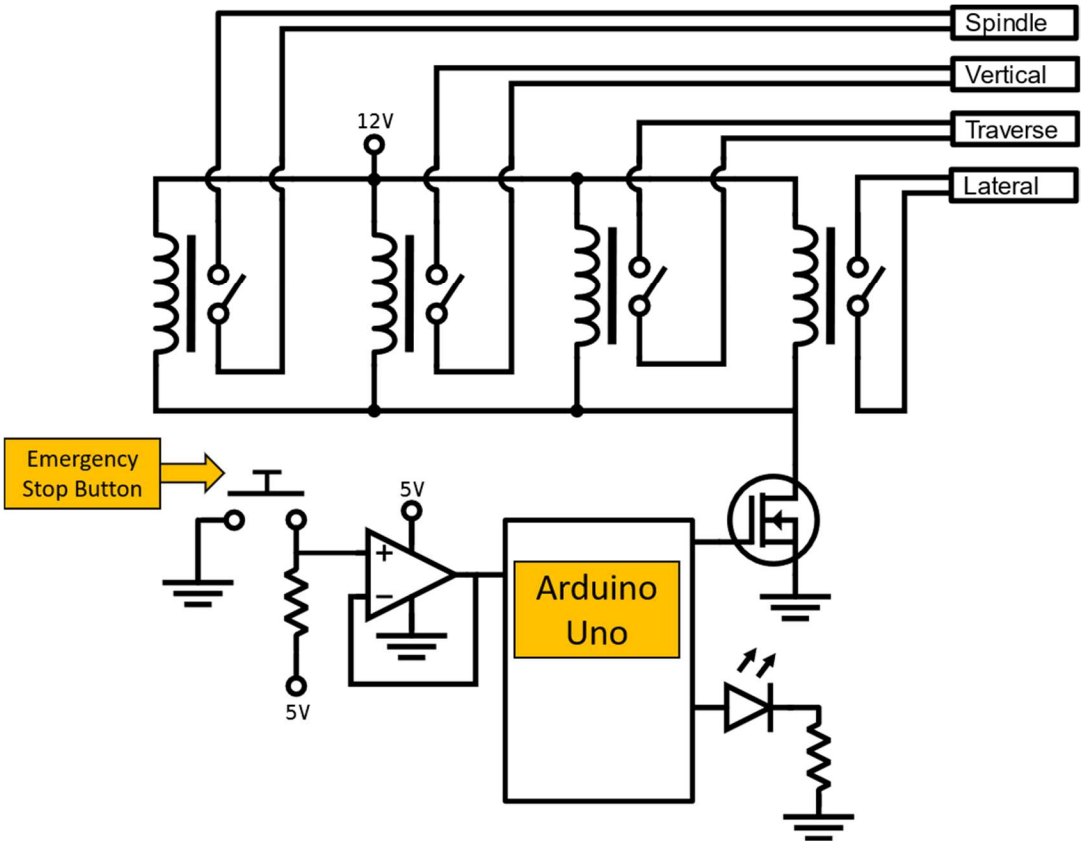


Figure 12: Circuit diagram of redundant safety circuit for emergency stops

The system works by interfacing with various digital inputs of the four motor drives of the welding machine, and these inputs are trigged when certain connections are either closed or open. The four relays

control those connections and the drives were programed and wired so that the motor drives would stop and become disabled if the relays are not energized. The vertical motor controller has a drive enable input which consists of the ENA(+) and ENA(-) ports. If these two ports are not connected the vertical motor will stop, turn off, and not accept any more commands. The spindle motor drive was reprogramed to be enabled if the COM and DIN3 ports were connected and to be disabled and stop the spindle if disconnected. The Lateral and Traverse motor drivers were both reprogramed to signal a fault if the ports DI5 and COM become disconnected. This will not only stop and disable the traverse and lateral motors when their relays are open but require the operator to clear the fault on the panel of the motor drive. The four relays are normally open so even power loss or a broken connection will disable the motors as an additional layer of protection. The program on the Arduino was kept as simple as possible and once the emergency button is pressed the Arduino needs to be reset to energize the relays and reengaged the motors. This redundant system to stop the welding process in case of an emergency should improve the safety of the lab and the welding machine. This circuit also easily allow the addition of limit switches to act as another fail safe to avoid the moving the mill table out its acceptable range of motion if desired. More information on how this system is integrated can be seen in APPENDIX C.

Safety Software

Improvements were also made on the software to detect faults. Previously an alarm feature was in place to monitor if the mill table moved beyond its safe limits and detect if the Z force passed a safety threshold. Once triggered the alarm function would attempt to stop all the motors and warn the user that something had gone wrong. The new alarm feature also checks for torque and XY force limits. Forces in the XY directions can be checked by simply taking the overall magnitude of the force which can be found in by

$$XY_{magnitude} = \sqrt{(X_{Tool})^2 + (Y_{Tool})^2} \tag{2}$$

If the limits are reached for any of these the machine will immediately attempt to stop all the motors. Additionally, if any control thread is running the alarm function will abort the thread and stop the creation

of any new control threads until the alarm is cleared. This monitoring function was also remade to have two different safety levels, where one is for when the machine is welding, and the other is for not welding. This allows the forces that will activate the alarm to be kept to a minimum when not welding. Hopefully this will help detect collisions of the tool and stop the machine before damage occurs when the machine is moving under non-welding conditions. These safety parameters can be changed if needed by clicking the word safety on the tool strip of the main welding form. This will open a new window which can be seen in Figure 13.

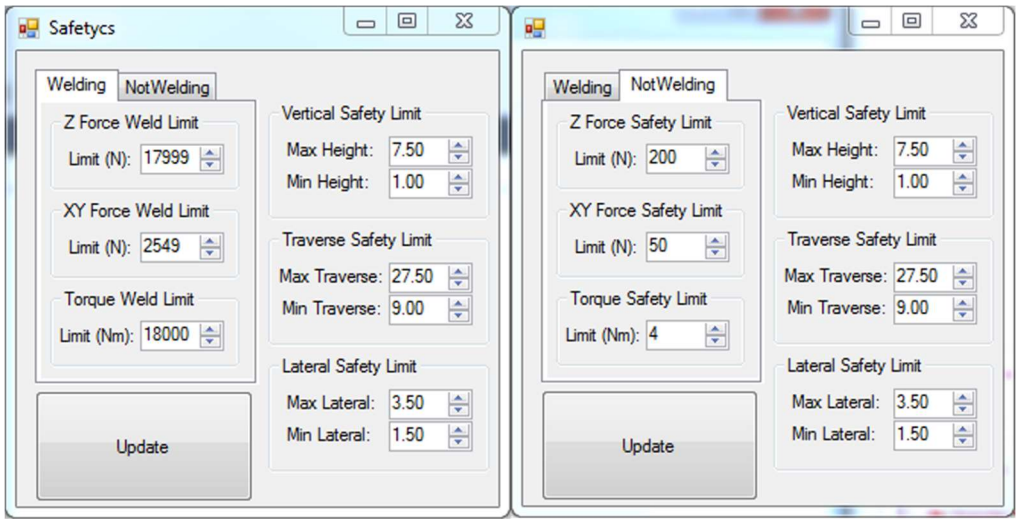


Figure 13: Safety form showing settings for both welding and not welding

Safety limits for forces in the XY direction and the torque are important to ensure that the Kistler dynamometer is not used outside of it's safe operating range. The specifications of the dynamometer appear like it would be able to handle higher XY forces, however the main limitations arise from the moment these forces cause the dyno to experience when they are applied to the tip of the welding tool. The maximal bending stress that the dynamometer is only rated for is 400Nm so if the distance from the end of the dyno to the tool tip was 0.125m the calculation for the maximal XY forces would be

$$\frac{400Nm}{0.125m + 0.035m} = 2{,}500N \tag{3}$$

The extra 0.035m is needed for the maximal loading calculation given in the dynamometers user manual. This force value is half the amount that the dyno is rated to handle in the XY direction so in the future if

higher XY forces are needed the tool and or tool holder will need to be shortened to shorten the moment arm. This however could lead to trouble thermally as then there would be less material between the heat generation in the weld and the dyno since the dyno is only rated to operate safely up to 60°C. Future work to help prevent damage to the dynamometer should include a system to monitor its temperature. If temperature issues are found during welding fins could be added to either the tool holder or the tool to be used injunction with the vortex air cooler.

CHAPTER 4 - LEVEL TABLE

Introduction

The friction stir welding process depends on maintaining a stable plunge depth into the material. If the tool is plunged too deeply excessive flash can occur causing loss of material in the weld thus decreasing its overall strength [4]. On the other hand, if the tool is not plunged far enough into the material the weld could have insufficient forging pressure and cause voids to form [14]. Given the high forces involved deflection can occur. Depending on the deflection severity this can cause sensor readings of the tool depth to be inaccurate. This can be overcome by using a controller that tries to maintain a known axial force, or having an accurate model of how the welding system would respond to the forces and account for the estimated deflection.

Measurement and Correction Setup

Since the welding setup at VUWAL is a modified milling machine deflection of the tool in the z direction is minimal than what would be experienced in a multijointed serial robotic arm. Welding takes place on a large piece of steel called the welding table that attaches to the mill table. The welding table has numerous threaded holes that allow for various clamps to be used to hold down the workpieces, one which can be seen in Figure 14.

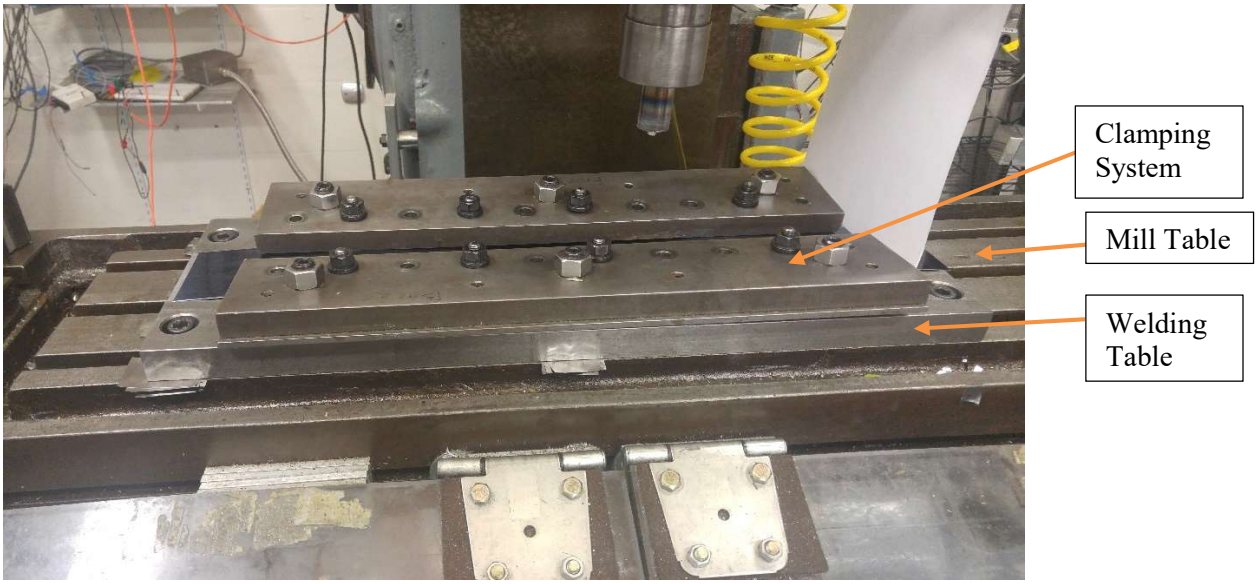


Figure 14: Location of welding table

Plunge depth for most of the lab's welding processes assumed that the welding table is level and that as the table is moved in the XY plane the welding tool's plunge depth would remain constant. To verify this a program was created to measure how level the welding table was. This program works by having the machine automatically move to a location then preform a measurement of the table height. This measurement process is detailed in Gibson's Thesis [12], and has only been slightly modified to fit into the restructuring of how threads are managed by the weld program and the inclusion of using the dynamometer for more accurate force readings. In summary, the process used for determining the table height works by slowly raising the weld table and using the force sensors to detect when the tool just begins to touch the weld table. The machine then stops the table and then records the position when that happened. The process is then repeated over many points across the weld table. The recorded points can then be analyzed and plotted to visualize the slope of the table. To correct for any deviation shims were then added between the milling table and the welding table to improve the trueness of the welding table. This process was done recursively until a desired level of improvement was met.

Results

The results for the welding table were plotted and then linear interpolation was used to generate

the height for the rest of the table. This can be seen in Figure 15 where the recorded points are the colored circles on the image and the rest of the color gradient shows the pricewise linear interpolation

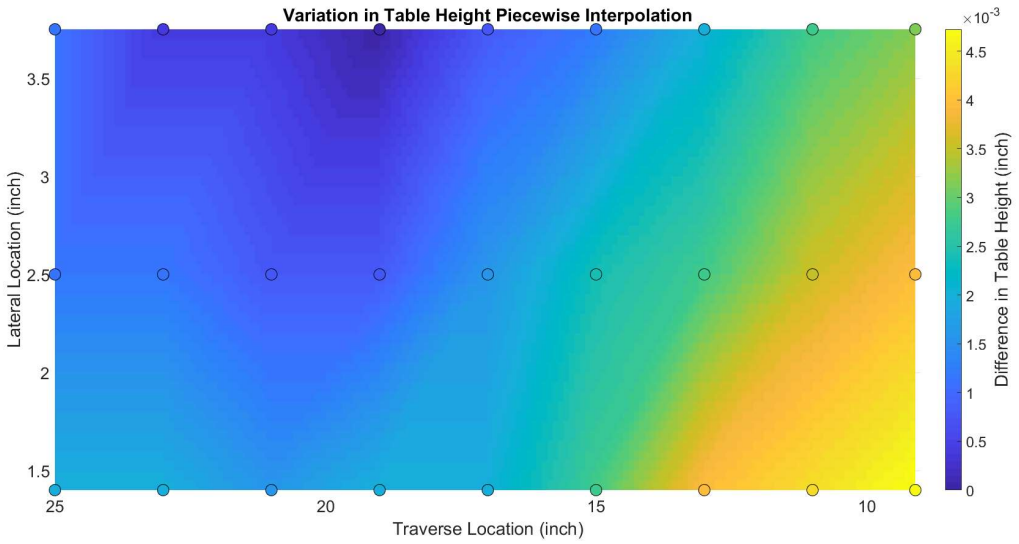


Figure 15: Table height, no shims, piecewise interpolation

Assuming the table is flat, a plane of best fit can be generated to give an estimate of the tables tilt which can be seen in Figure 16.

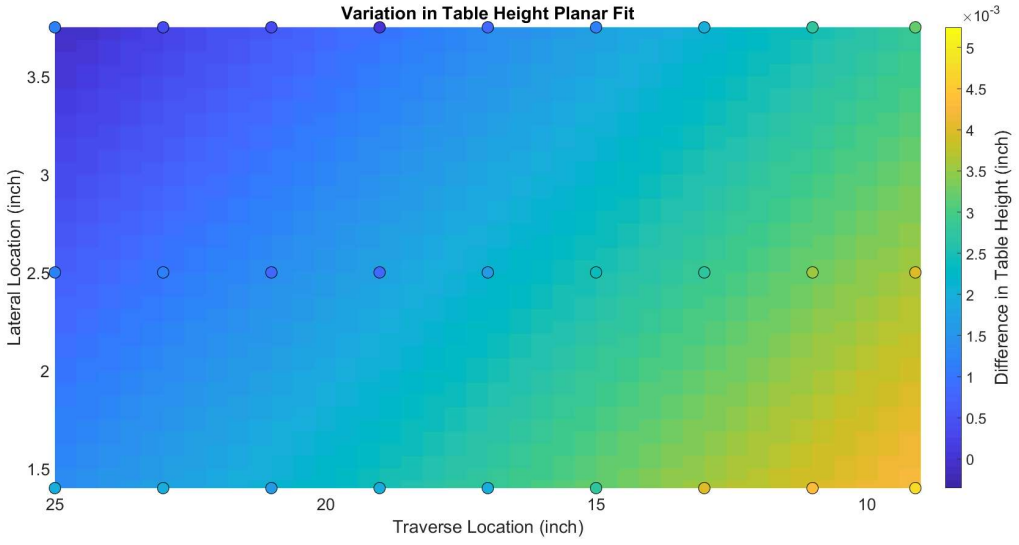


Figure 16: Table height, no shims, planar fit

The welding table was then shimmed with pieces of 0.001 inch shim stock to correct for the tilt of the table and then remeasured. This process was repeated until the planer fit became close to level. The raw

data along with the Matlab code used to do these calculations and generate these figures can be found in APPENDIX G. The result of the table's leveling process can be seen in Figure 17 for the planer fit which show a significant improvement from the state of the welding table before shims that was seen in Figure 16.

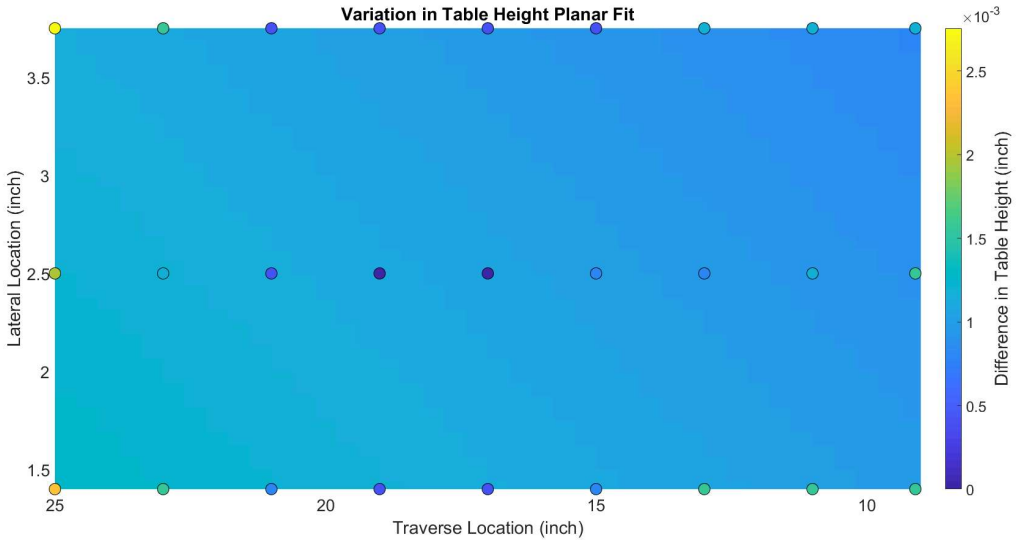


Figure 17: Height map of fitted plane to table height final test

However, examining the actual data points there is still quite some variation in the table height which can be seen more clearly in the linear piecewise interpolation of the data which can be seen in Figure 18.

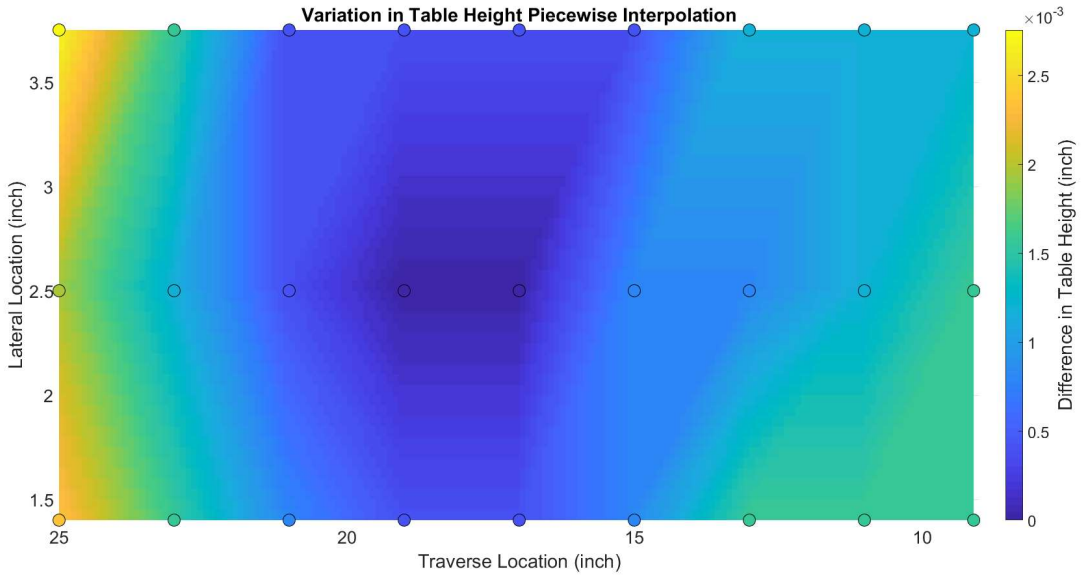


Figure 18: Shimmed table height map with linear piece wise interpolation

The variation in height was still reduced by a factor of two by just comparing the raw data points, and while the table is more level than without shims this data suggests that the welding table itself may not be flat and that the surface might need to be refinished as there appears to be a valley like feature around the 18-inch mark in the traverse direction. A table showing the differences in the measured slopes can be seen in Table 1 where the coefficients for the equation of a plane are given by

$$z = a\,x + b\,y + c \tag{4}$$

Where z is the table height, a is the slope of the plane in the traverse direction x, and b is the slope of the plane in the lateral direction y. C is the offset of the plane and is not important for determining the slope of a plane.

Test	Coefficient a 10^{-3}	Coefficient b 10^{-3}
No Shims	-0.185	-0.683
Shim Test_1	-0.165	-0.424
Shim Test_2	-0.167	-0.145
Shim Test_3	-0.131	-0.127
Final Test	0.020	-0.069

Table 1: Coefficients for plane of best fit of measured table height

The difference of the slope between the welding table without shims and the shimmed table show a great

reduction in the overall slope of the table by almost a factor of 10. The raw data for these tests can be found in APPENDIX G.

Conclusion

In summary, the surface of the welding table was measured by collecting the height of the table of multiple points. A plane of best fit was then generated that best correlated to those points, and then shims were added to the welding table to correct for any measured tilt of the welding table. This will allow for less variation in the height of the workpiece during welds which should translate into more accurate plunge depth of the tool during welding. However, the welding table appears to no longer to be a flat surface as a valley like feature is clearly seen in Figure 18. This could be due to the welding table not being flat or possibly the milling table not moving in a uniform manner with some tilting occurring as more of the table overhangs the base of the machine. To fix any height deviation a milling tool could be fitted to the machine and the welding table could be milled flat, and since the milling machine has already been fitted with strain gauges [15] the deflection that takes place could in theory be corrected using that data and the results of the FEA simulations.

CHAPTER 5 - RESOLVING XY FORCES

Introduction

Measurement of the forces perpendicular to the welding tool can be a valuable tool for analyzing a weld. To acquire these forces welds can be done on an instrumented welding table, or sensors can be placed to measure forces on the tool. Hattingh [16] placed strain gages on the tool and by plotting vs the tool angle was to generate what he called a force footprint seen in Figure 19.

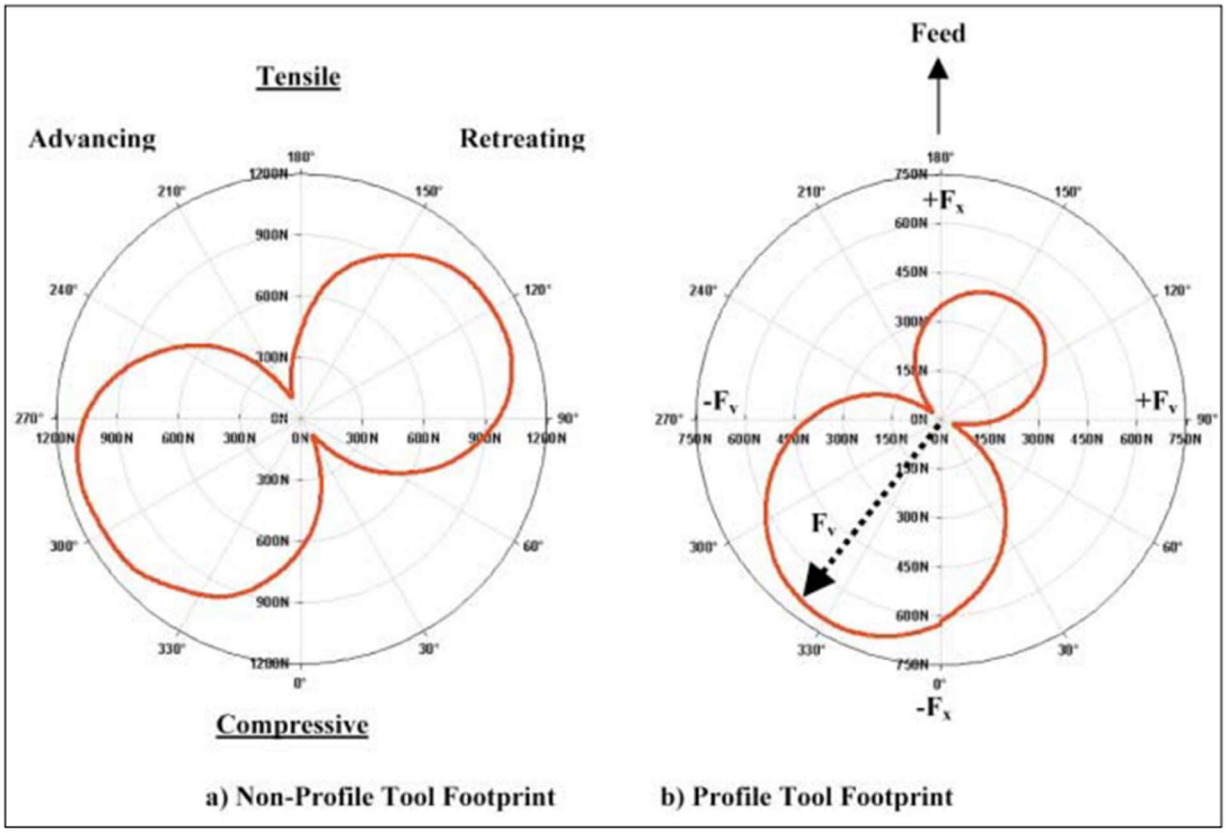


Figure 19: Typical 2D force footprint, 300 RPM and traverse rate of 120 mm/min [16]

These two plots reveal significant differences in the forces when comparing two different tools, and it can be seen the tool with a non-symmetrical pin profile gave a non-symmetric force foot print. While Gibbson [12] developed a custom low-cost alternative to the Kistler rotating cutting force dynamometer to measure welding forces, his solution lacked the ability to measure forces in the X and Y direction. The Kistler dynamometer has this capability, however the forces it measures are in relation to the tool not the welding

table. To resolve the forces back into the weld's reference frame a way to measure the rotation of the welding tool is needed. Previously an optical encoder with fins spaced every 36 degrees was set to trigger a force reading on the PCI-DAS1602/16 DAQ card 10 times per revolution This required the DAQ card to be put into a mode where it waited to be triggered and then recorded data. To facilitate this a second computer was used to record data from the dyno as this setting on the DAQ card would cause a control program to be unusable as too much time would be spent waiting for these trigger signals. An example of the results achievable by the older system can be seen in Figure 20 which is from David Lammlein's Ph.D. dissertation [17].

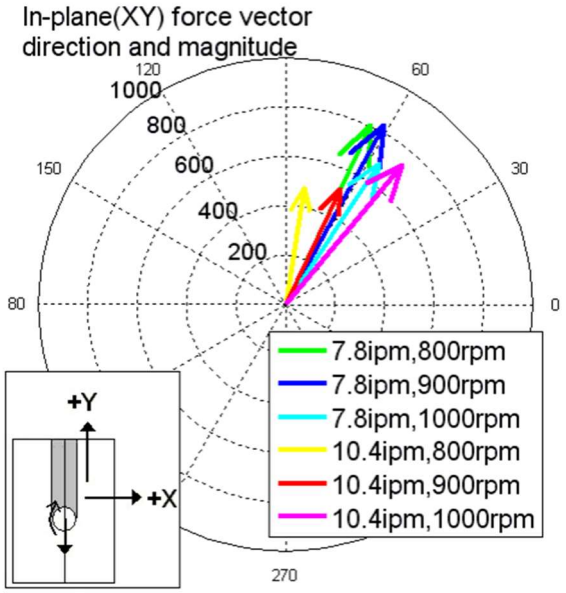


Figure 115: (Unsupported, conical tool) The tool is deflected towards the advancing, trailing quadrant.

Figure 20: XY force vector for welds made with a unsupported conical tool [17]

Dedicating an entire computer for this purpose is unnecessary and would add unnecessary complexity to the control system. To remedy this a different method for measuring the tool angle was developed.

Implementation

An analog sensor was used to measure the angle of the tool. This was accomplished using a magnetic rotary position sensor and a magnet attached to the top of the spindle of the machine. The

rotary sensor used was Austria Mikro Systeme's (AMS) AS5600 chip, which is a programmable 12 bit sensor in a SOIC-8 package that has either PWM or analog voltage output. The sensor was selected because of its sampling rate, and the ease of which it could be incorporated into the overall welding system. The sensor's sampling rate is important for resolving the XY forces as a sampling rate that was too slow would cause an undesirable amount of lag between the spindles actual position and the value reported by the sensor. The sensor has a sampling rate of 150 µs, and most of the welds done by the lab tend to be at or below 1500 RPM. The spindle rotating at that speed would only introduce a 1.35 degree lag in the measurements of the tools angular position. To utilize the sensor a PCB breakout board was designed in Eagle and then sent to be manufactured. The components were then soldered together, and a low-cost rotary sensor was created. A schematic of the PCB design can be seen in APPENDIX F. Four of these sensors were created for every motor of the system, and the total cost of the sensor is outlined in Table 2.

Item	Description	Qty	Price	Amount
PCB	OSH Park, PCB Breakout	6	1.63	9.80
Resistor 4.7 kΩ	DigiKey, 4.7k Ohm ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	10	0.015	0.15
Capacitor 1 uF	DigiKey, 1µF ±10% 50V Ceramic Capacitor X5R 0603 (1608 Metric)	4	0.18	0.72
Capacitor 0.1 uF	DigiKey, 0.1µF ±5% 16V Ceramic Capacitor X7R 0603 (1608 Metric)	4	0.12	0.48
Magnetic Rotary Sensor	DigiKey, AS5600, IC SENSOR MAG ROTARY 12BIT 8SOIC	4	3.88	15.52
Header	Digikey, 12 Positions Header, Breakaway Connector 0.100" (2.54mm) Through Hole Tin	2	0.90	1.80
Magnets	Digikey, Magnet Neodymium Iron Boron (NdFeB) N35 0.236" Dia x 0.098" H (6.00mm x 2.50mm)	4	0.32	1.28
			Total:	29.75

Table 2: Cost breakdown of magnetic rotary sensors

The sensor for the tool was placed above the spindle of the milling machine and a diametrically magnetized disk magnet was attached to the top of the spindle. These were then brought into close

alignment using visual inspection and this can be seen in Figure 21.

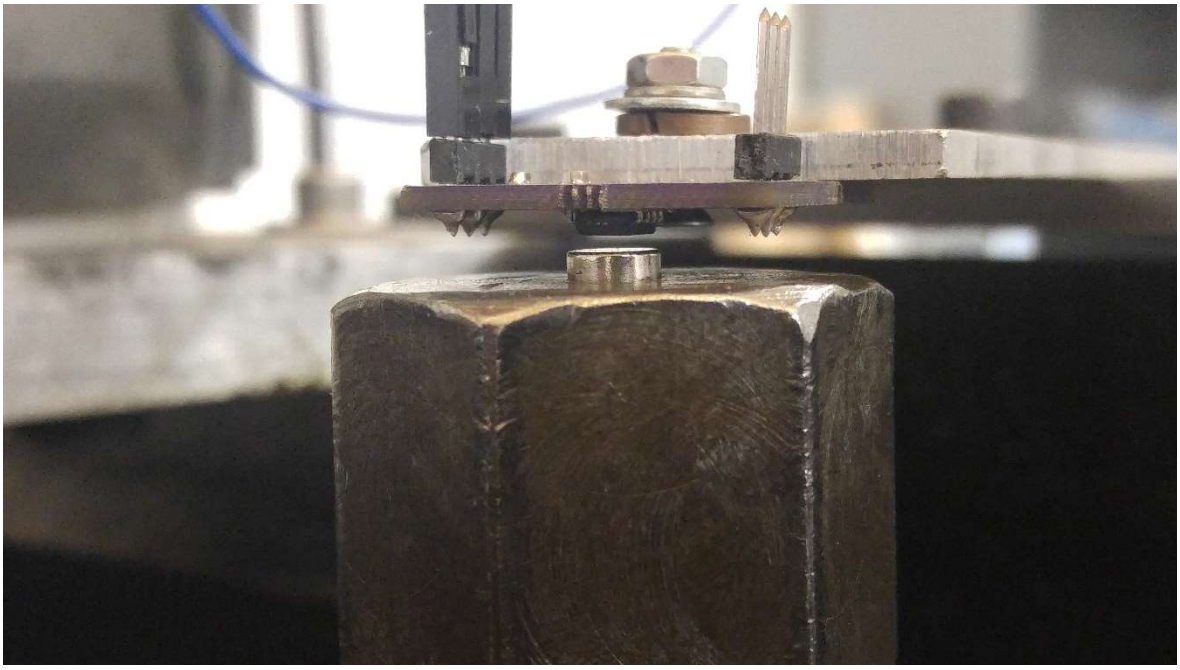


Figure 21: Picture of magnet and sensor for determining the spindles rotation

However negative aspects of this sensor location is that as the machine turns there is some wobble in the top of the spindle and the sensor is only properly aligned for welds performed with zero tilt angle. A different method to attach the sensor would need to be designed for tilted welds. The sensor gives a voltage from 0 to 5 volts corresponding to 0 to 360 degrees which is sampled by the NI 6024E Daq card on the real-time computer at 5kHz. Once this angle is measured a rotation matrix is used to resolve the X and Y the forces in the frame of the weld instead of the frame of the tool using equation (5). The coordinate frame of the weld has the X axis along the traverse of the weld and the positive Z axis is aligned with the axial direction which can be seen in Figure 22.

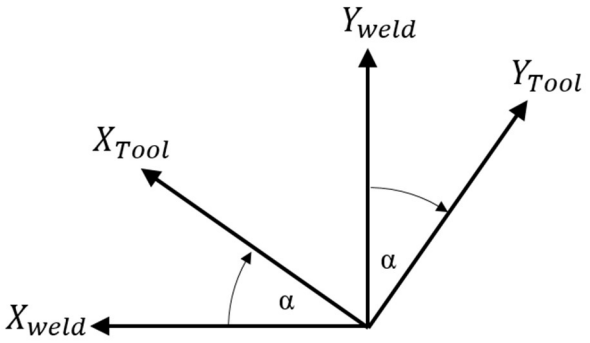


Figure 22: Diagram of coordinate frame rotation

$$\begin{bmatrix} X_{Weld} \\ Y_{Weld} \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} X_{Tool} \\ Y_{Tool} \end{bmatrix} \quad (5)$$

XY Force Test

To test this setups ability to reconcile the forces from the dyno's frame to the weld's frame a force was applied to the tool only in the negative x direction. This was done using a 0.5 inch bearing around the tool which was then attached to a table using a spring seen in Figure 23. This allows the force to be controlled by moving the table in the traverse direction which stretched the spring increasing the force applied by a proportional constant.

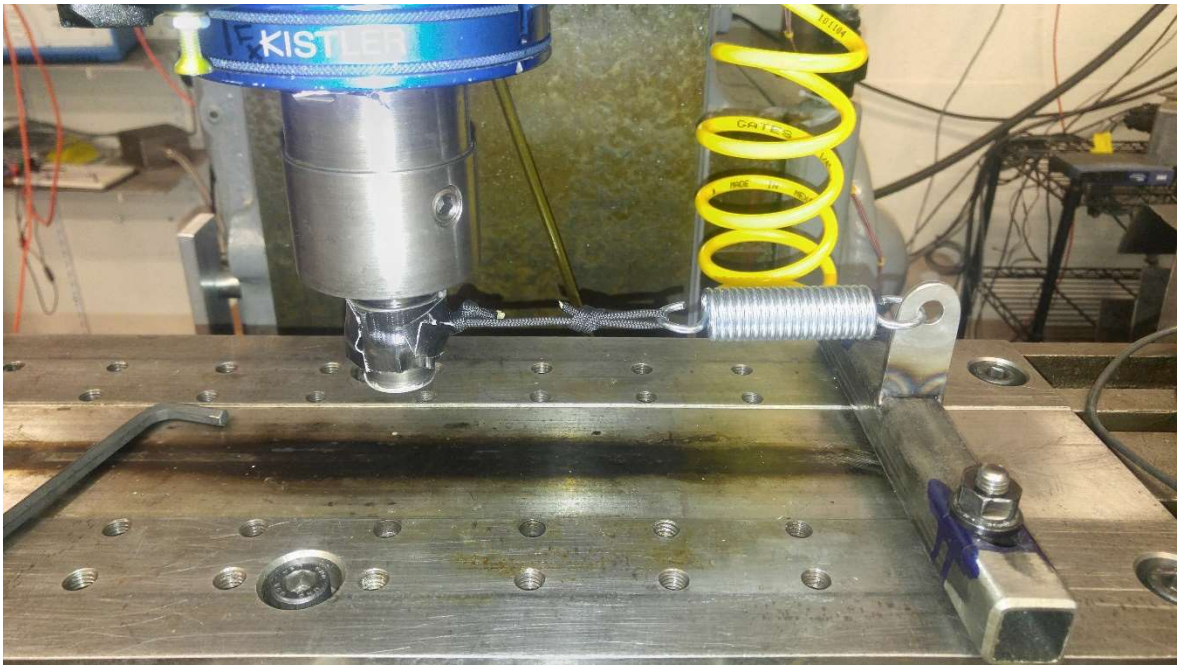


Figure 23: Test setup for resolving XY forces

For the first test, the table was moved to a fixed position and then the machine was run through a variety of spindle speeds which can be seen in Figure 24.

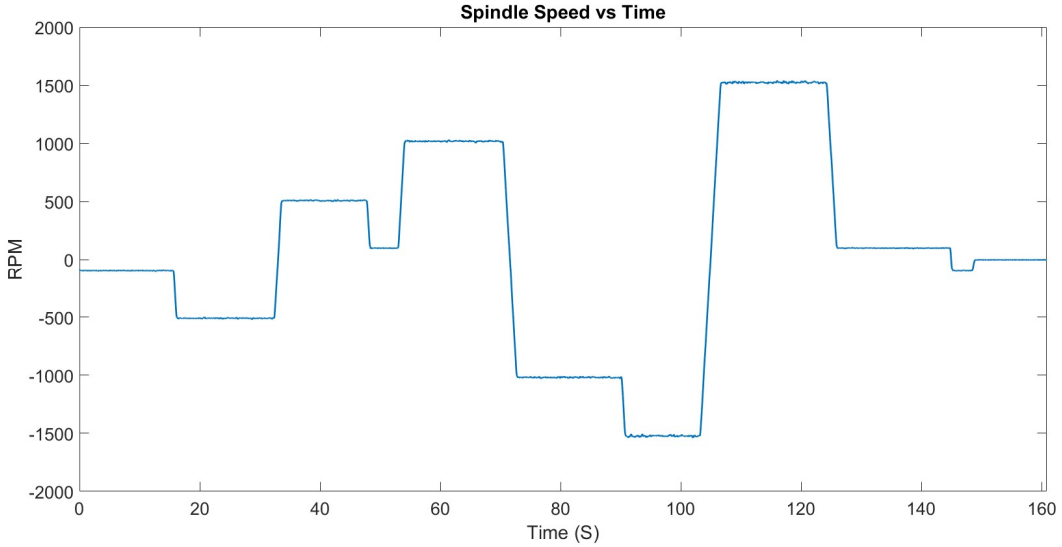


Figure 24: Spindle speed vs time for XY force test

The speeds ranged from -1500 to 1500 RPM, and additional benefit of measuring the angle of the tool is that it allows for measurement of the rotational speed by taking the angular positions derivative. The X

and Y forces that correspond to this test can be seen in Figure 25

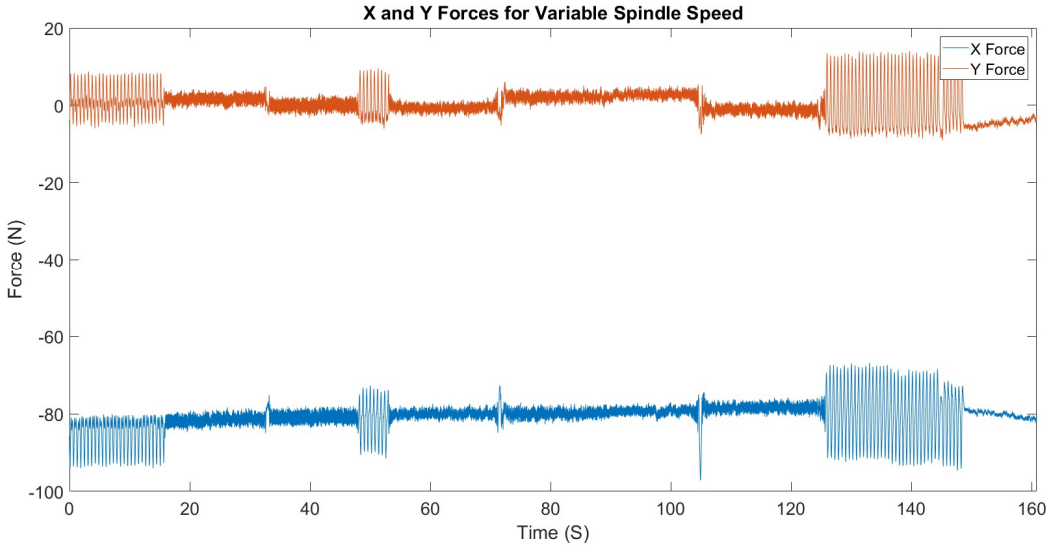


Figure 25: X and Y Force on tool as spindle speed varied

As can be seen in Figure 25 the forces in the X and Y direction remained consistent as the rotational speed was varied. The data was sent through a first order low pass filter with a time constant of 0.1 which explains why the areas with higher rotation rates seem to have less oscillation. For the second test, a constant spindle speed of 1500 RPM was maintained while the table was moved to vary the force applied to the tool. The position of the table can be seen in Figure 26.

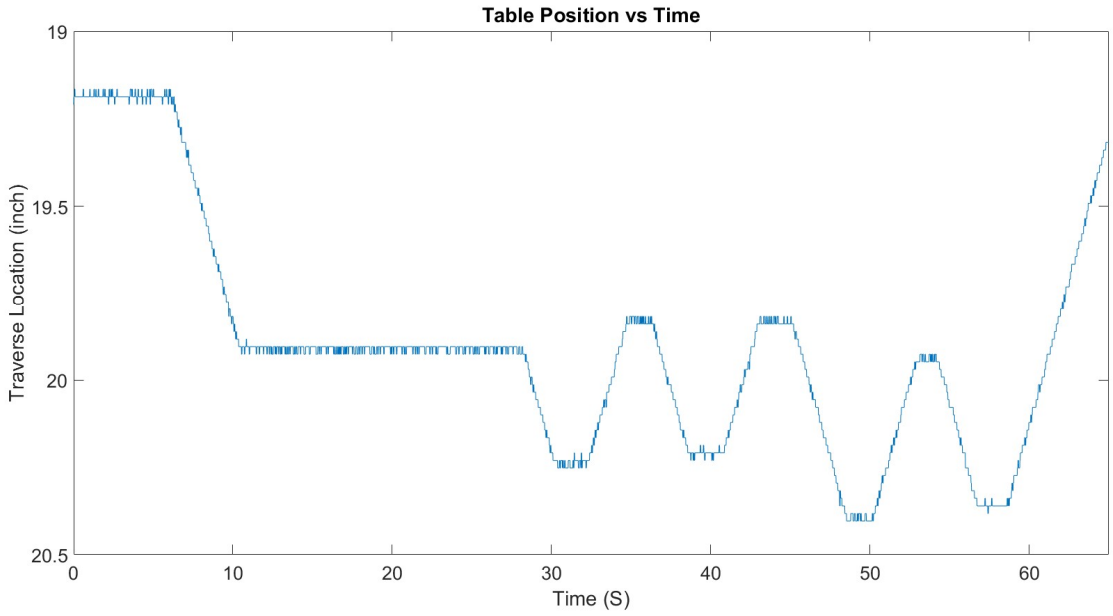


Figure 26: Table position vs time for varying force applied on tool

Do to the spring, the changes in the position of the table should be proportional to the change of the force on the tool and this can be seen in Figure 27 where the force applied in the X direction corresponds to the position of the table with almost no effect on the force read in the Y direction.

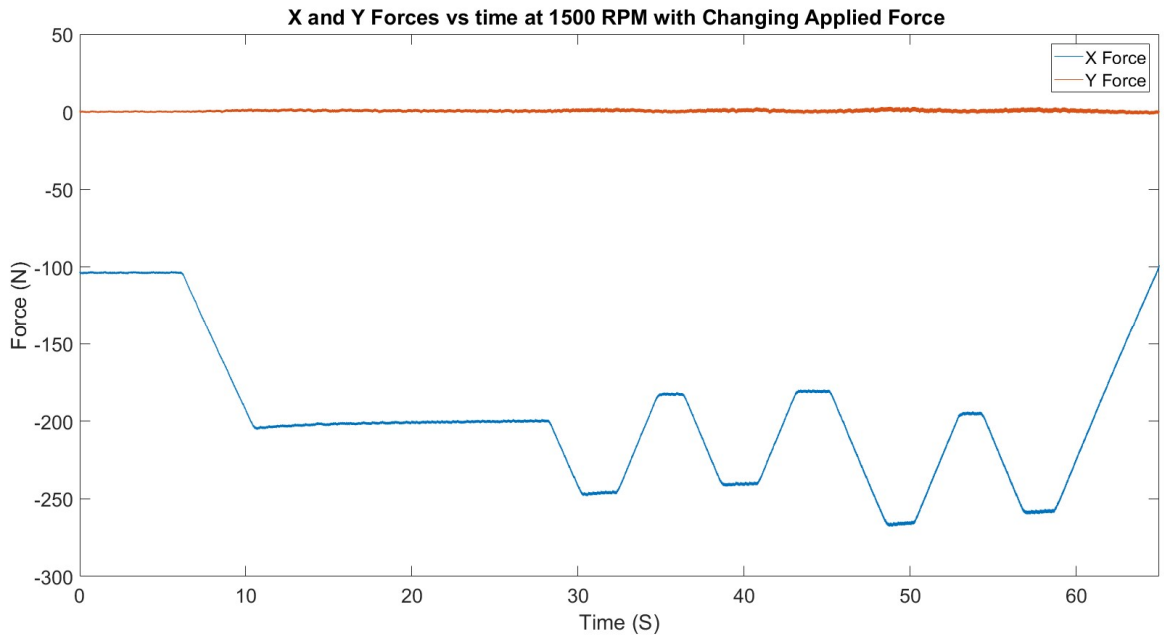


Figure 27: XY forces for variable traverse force at 1500RPM

Again, the forces were passed through a first order low pass filter with a time constant of 0.1 seconds to eliminate most of the effects due to noise, spindle wobble, and tool runout for the creation of Figure 27. Another important use of the ability to sense the X and Y forces on the weld tool is to look at the forces the tool experiences during the rotation of the tool. However, when trying to look at the forces during a rotation the limited sampling frequency of 5kHz does not allow much room to filter out the noise from the motors, as at 1500RPM you are only getting about 200 samples per rotation and any filtering would have major effect on the phase of the forces. Since filtering would add an undesirable amount of phase lag for looking at the forces during a rotation of the tool a better understanding of the level of electrical noise in the system is needed.

One way to look at the effects of electrical noise on the system is to activate various motors and look at how that effects various sensor readings. This was done by examining the force readings from the dynamometer with no load and then periodically activating the vertical motor, traverse, and spindle motor. The full results of these test can be seen in APPENDIX H, however for the X Y force tests only the spindle motor and traverse motor were in use and their effect on the force measurements can be seen

in Figure 28 and Figure 29.

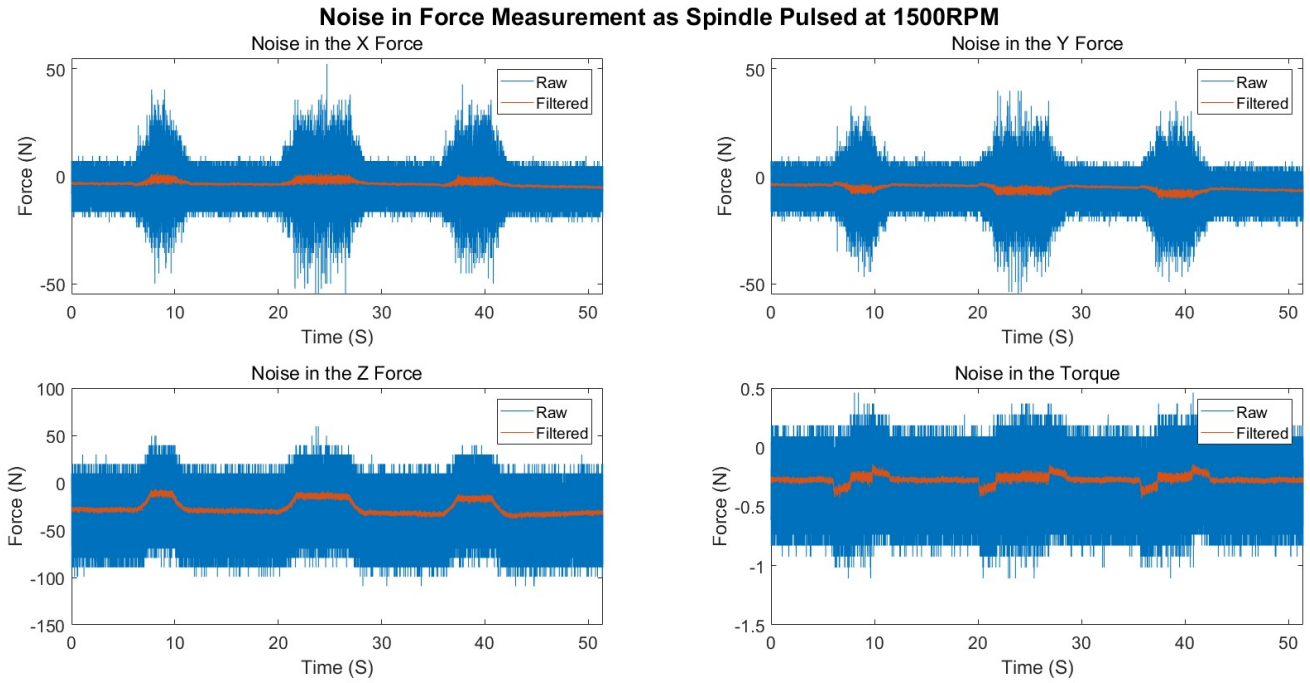


Figure 28: Noise in force measurements from running the spindle at 1500RPM

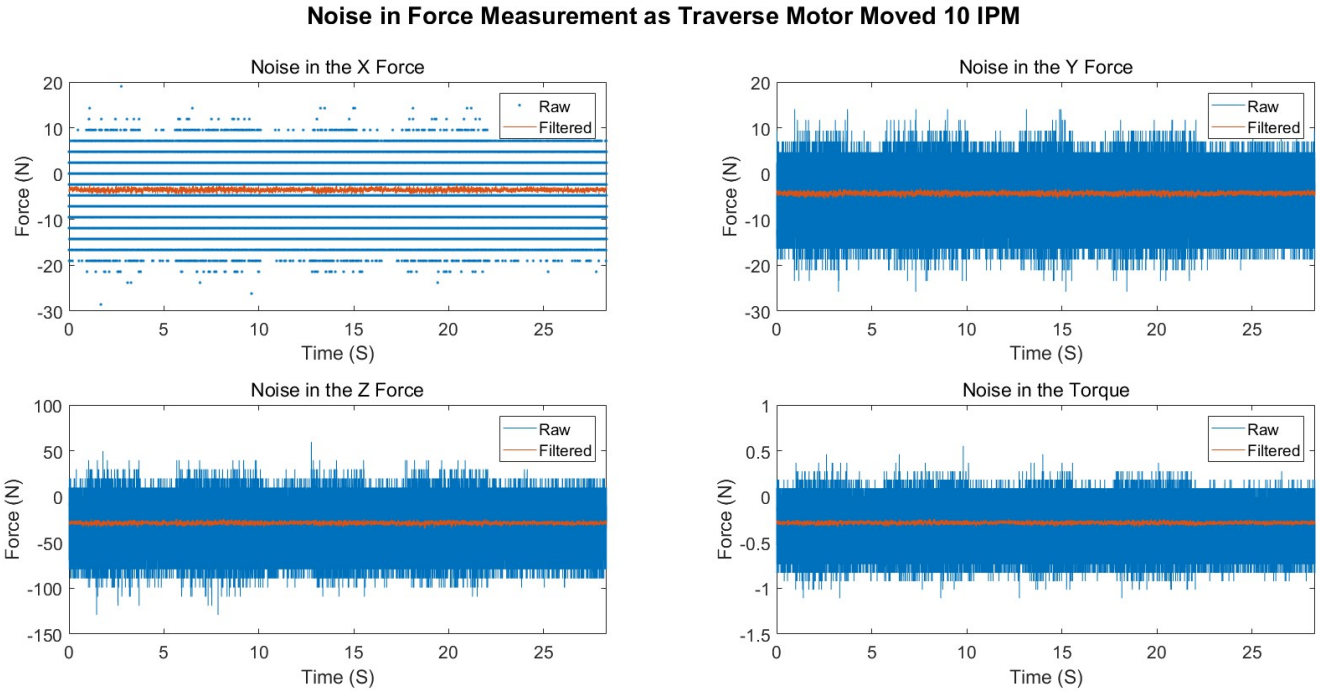


Figure 29: Noise in force measurements from traverse motor

Looking at the plots only the activation of the spindle motor had much effect on the noise on the system

increasing it from around a magnitude of 20N to that of about 50N. However, some of the noise seen in the XY forces might be due to tool runout, either way the noise is low enough not affect the test data terribly for being raw unfiltered data. This can be seen in Figure 30, and the data for this was taken from the same test shown in Figure 27.

Figure 30: Unfiltered X and Y forces for one rotation at 42.5478s

Some lower frequency variation in the forces can be seen and these can possibly be attributed again to tool runout and/or the wobble in the top of the spindle causing slight misalignment of the angle sensor. Evidence for this is given from examining multiple rotations that the variations are periodic and appear to be a function of the tool angle. The issue of noise from the electrical system of the machine will be even less of a factor during welds as the forces are much higher than what was done for this test, and will increase the signal to noise ratio. It can also be useful to look at the resultant force vector experienced by the tool where the magnitude of the force was already calculated using equation (2). The angle of this magnitude can be calculated form this equation.

$$\theta = tan^{-1}\frac{Y_{Weld}}{X_{Weld}} \tag{6}$$

Then the resultant force vector can be seen in the Figure 31 which shows the corresponding force on the tool plotted in polar coordinates.

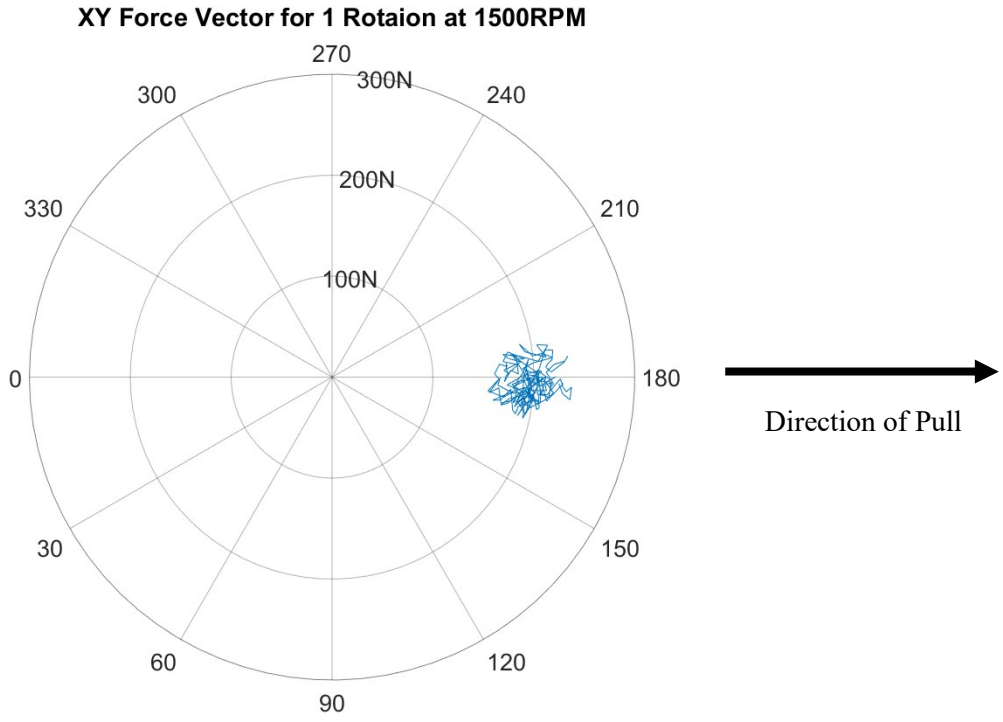


Figure 31: Polar plot force vector for 1 rotation

XY Forces during a weld

Now that this system of resolving X and Y forces from the dynamometer works, the forces of welds can be examined confidently knowing that the system is returning the correct information. An example for the XY forces experienced by the tool during a weld run at 1400 RPM and 6 IPM can be seen in Figure 32.

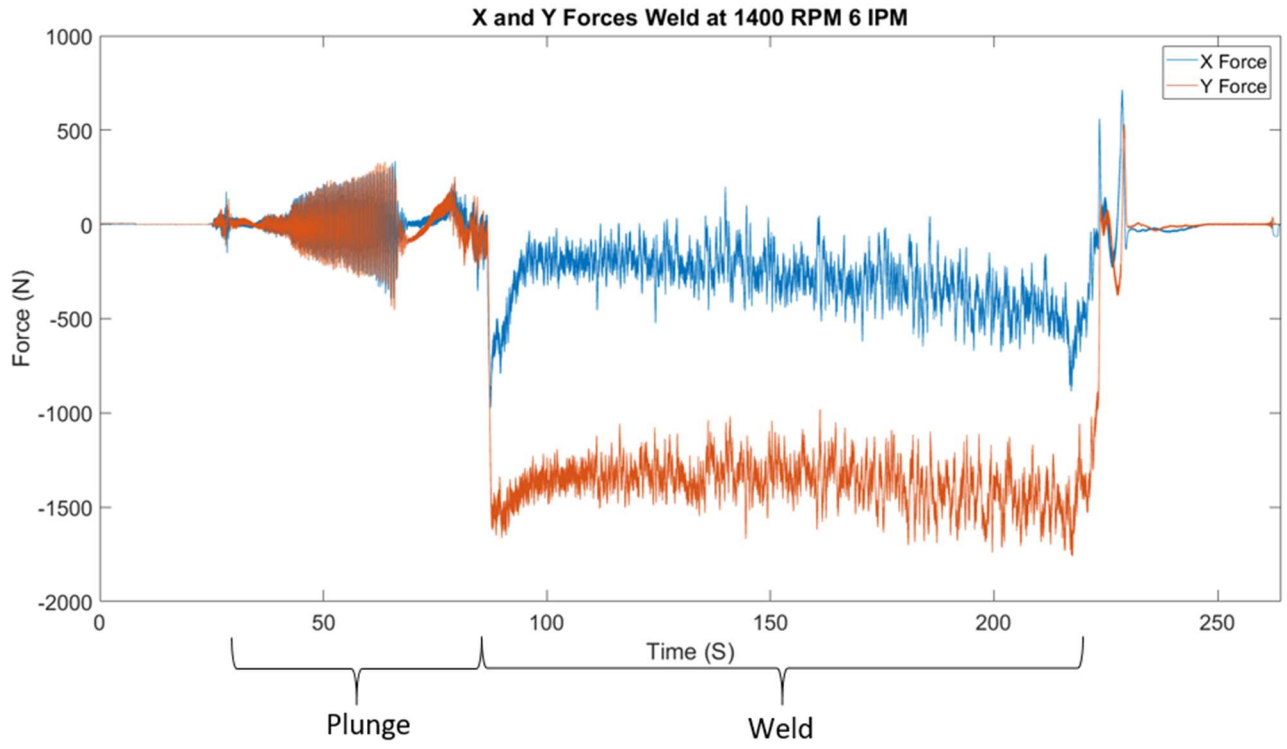


Figure 32: Filtered X and Y forces vs time during a weld

These forces were passed through a first order low pas filter before being plotted with a time constant of 0.1 seconds. The plot shows that most of the forces the tool experiences occur once the tool starts to move through the material. A spike in the forces can also be seen once the weld starts to move in the traverse direction, and is more pronounced in the X direction. This force is directly opposing the tool moving through the material, and as was stated earlier to avoid such spikes in the forces a longer ramp up to traverse speeds will need to be used to reduce this spike. The forces experienced by the tool in one rotation at time 102.7758 can be seen in Figure 33 and a plot of the resultant force vector for that rotation can be seen in Figure 34.

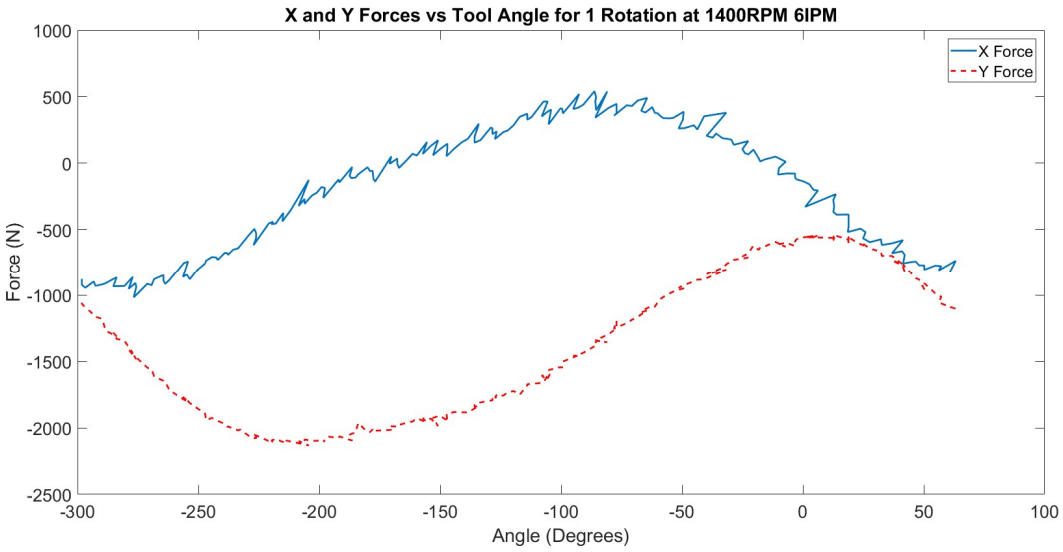


Figure 33: X and Y forces acting on the tool vs time for one rotation at 102.7758 s

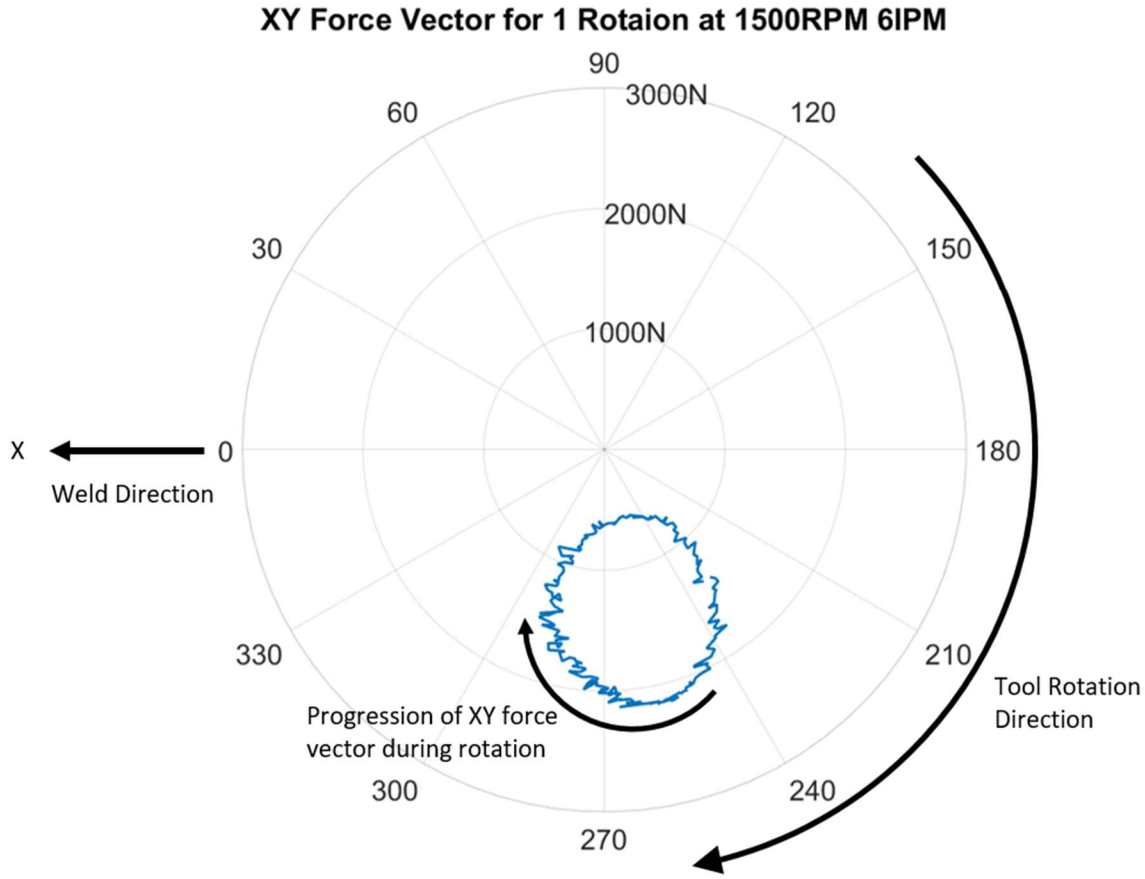


Figure 34: Polar Plot of forces acting on the tool for one rotation at time 102.7758

Most of the force occurs in the Y direction and matches what would be expected as the rotating tool encounters new material based on its direction of rotation. Also, oscillations can be seen in the forces which could arise from the runout of the tool and of the oscillatory nature of the welding process which can be seen in the common formation of the traditional onion ring pattern.

Conclusion

By using this analog magnetic rotary sensor, the XY forces from the rotating dynamometer can be successfully resolved from the frame of the tool to that of the weld. These forces are useful for gaining a better understanding of what is happening during the friction stir welding process. This ability could serve the lab to detect defect formations online and be a new set of data for validating modeling efforts. In addition, the angle sensor of the tool provides a simple method to apply feedback control on the tool's rotation rate, since previously the lab mostly relied on open loop control to set the speed of the spindle motor based on past calibrations.

CHAPTER 6 - BACKLASH AND IMPROVED POSITION MEASUREMENT

Introduction

VUWAL's friction stir welding machine being a modified milling machine from WW2 has undoubtedly seen a lot of use during its long service as both a milling machine and a research tool. This use has led to quite a bit of wear in the internal mechanisms that translate the machine's rotational inputs into the linear movements of the welding table. This wear most prominently presents itself as backlash in the system which adds a nonlinear term that needs to be dealt with when translating the angular displacement of the driving shaft to the linear motion of the stage. However, before that can be done the backlash of the system needs to be measured and quantified. This was accomplished by adding magnetic rotary position sensors to the welding table's input shafts, and then this additional information can be used with the old sensors to calculate the backlash in the system.

Implementation

The magnetic sensors used were the same ones used to measure the rotation angle of the spindle and the senor data was also fed into the same NI PCI-6024E DAQ card on the real-time computer. This allows the data to be read in directly using the Simulink portion of the welding code so new data is read in and processed at 5 kHz. The sensors however only output a voltage corresponding to a rotated angle so a shaft rotating constantly will generate a sawtooth waveform as can be seen in Figure 35.

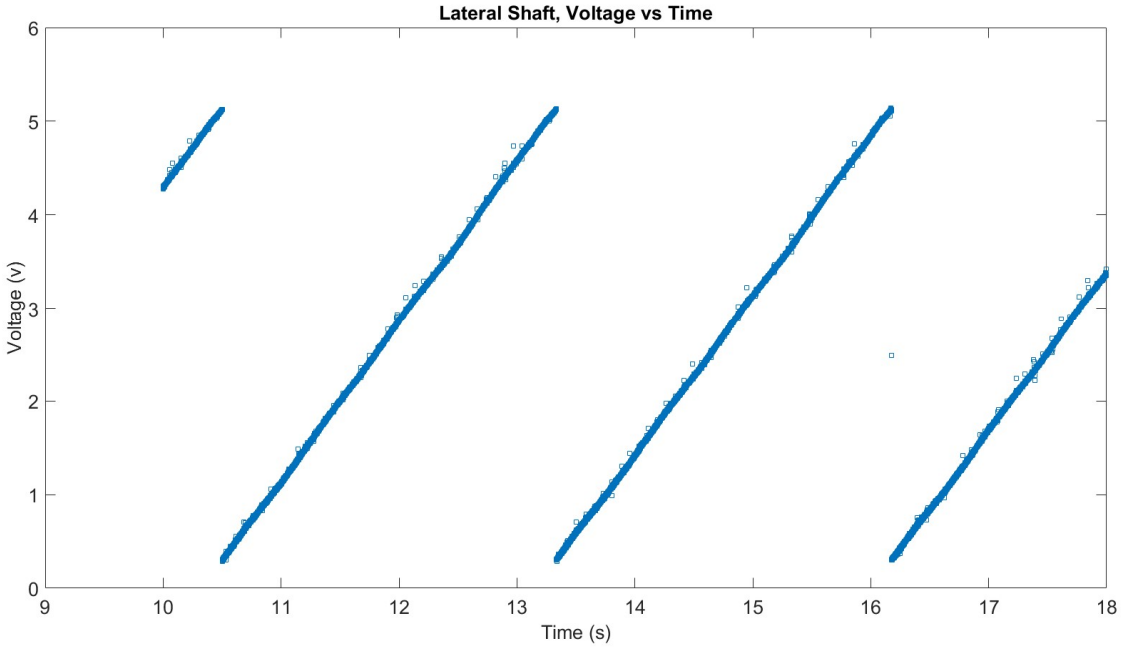


Figure 35: Raw Voltage Measurement from Lateral Shaft Angle Sensor

After examining the data, the transitions were clean with only an occasional point reported between jumps which can be seen a little bit past the 16 second point in Figure 35. Since at most there is only one-point in-between transitions only a three-point moving window is used to analyze the data looking for these transitions, and when detected increasing or decreasing the number of rotations. The processed signal is then just calculated as the current measured voltage plus five times the current number of rotations which generates a smooth signal proportional to the total angular displacement of the shaft which can be seen in Figure 36.

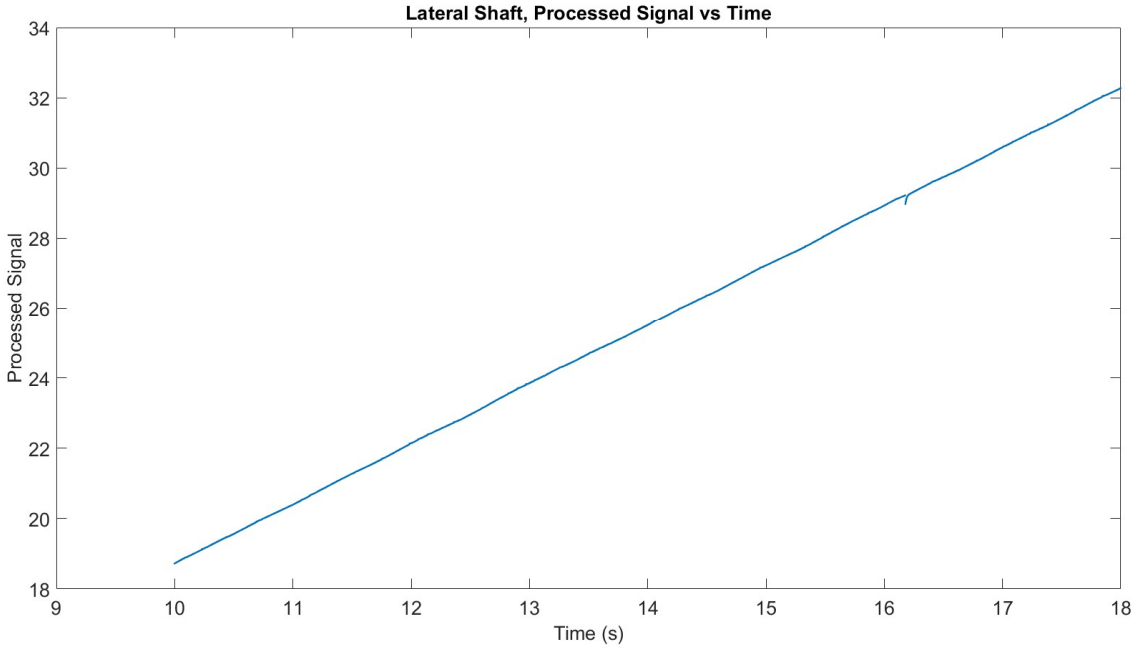


Figure 36: Processed Lateral Shaft Signal

The machine was then made to follow a square wave like pattern along each axis to gather data for calculating the backlash and gear ratio of the system. For a perfect gear system relating the rotation of the shaft to the linear location of the table (x) would just be the rotation angle (α) multiplied by a gear ratio term (R) plus an offset term (C).

$$x = R\alpha + C \tag{7}$$

So, using linear regression the optimal gear ratio and offset value could be found by comparing the processed data from the rotary sensors to the location data provided by the string potentiometers. Backlash however introduces a non-linearity in the system and is harder to calculate. To find the backlash in each drive the gathered angular displacement data was passed through a model of backlash using many different values. Then to account for the differences in delays between the signals the derivative of the angular data was used to calculate the portions of the data where the table was stationary. This difference in delay comes from the fact that the data from the magnetic sensor is read directly by the real-time computer, while the data of the string potentiometers is read by the sensor box passed over TCP

to the C# program which then finally passes the data to the real-time computer over UDP. If all the data was used there would be a slight offset between the two data sets as the machine moved, which would then become unwanted error when doing the regression. These portions of data where the table was not moving were then averaged and using least squares the data was used to fit the model in equation (7). Then the resulting estimate of the table position was subtracted from the measured and then the average of the squared error was found for multiple backlash values for lateral, traverse, and vertical directions. The result of this is graphed in Figure 37.

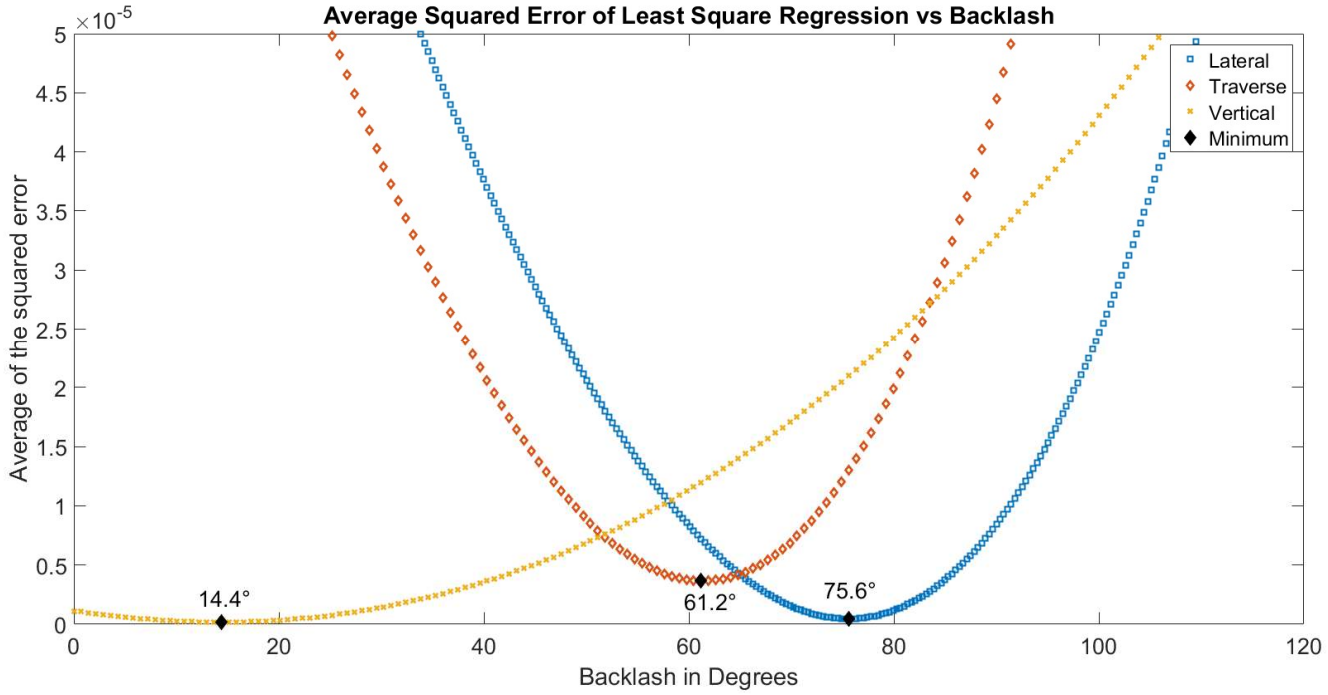


Figure 37: Plot of Average Squared Error vs Backlash

The backlash of the system was then found by picking the backlashed value that minimized the square error which can also be seen in Figure 37. A plot showing the large disparity that occurs in not accounting for the backlash can be seen in Figure 38..

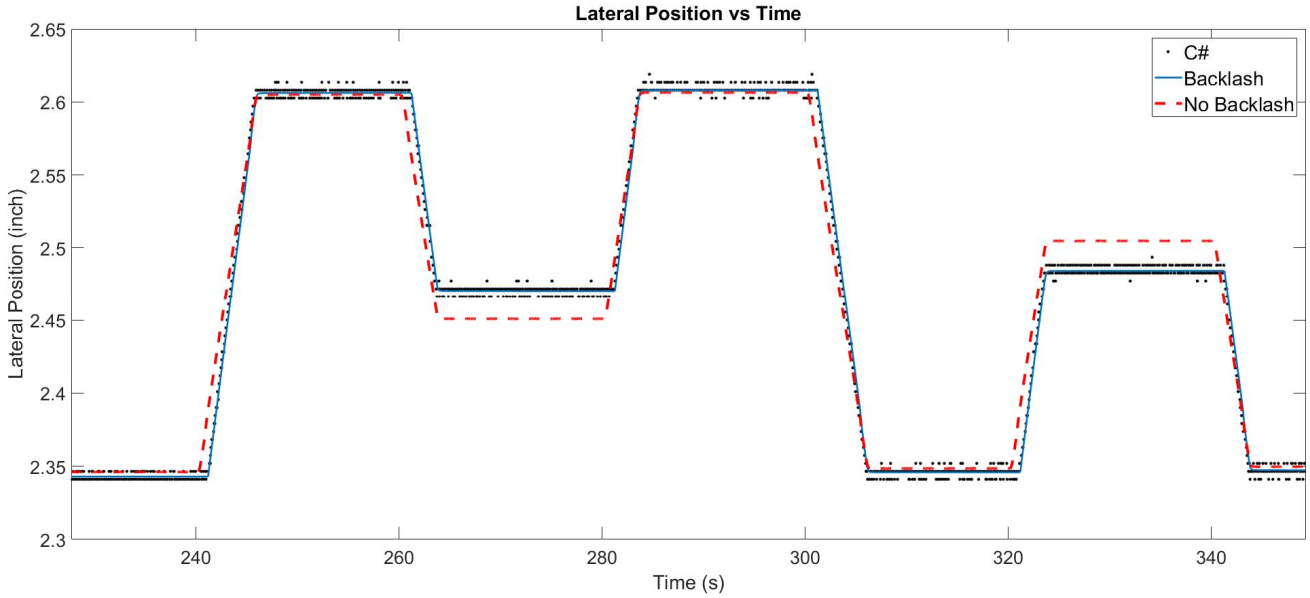


Figure 38: Plot of lateral position vs time

For Figure 38 the potentiometer data is in black labeled C#, and the data from the magnetic sensors is labeled 'Backlash' and 'No Backlash' An additional benefit of these rotational magnetic sensors is that they can offer a much more accurate reading of tables position when compared to the string potentiometers which can be seen in Figure 39.

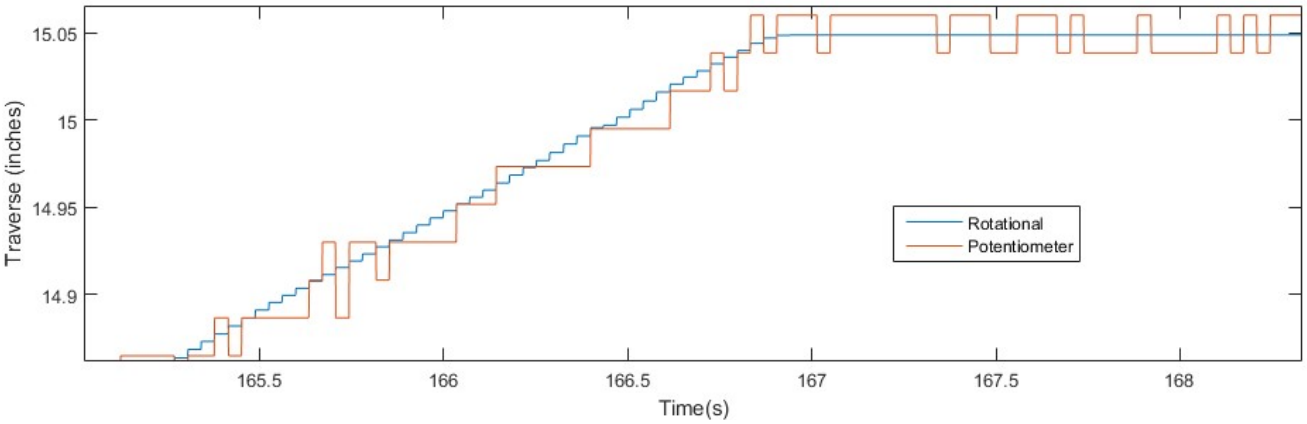


Figure 39: Example of increased positional accuracy offered by new sensors

Conclusion

These angular sensors provide a way to measure the tables backlash which will allow that nonlinearity to be more successfully controlled in the future. These sensors also provide greater resolution to measure the tool's position with greater accuracy. The signal generated by these sensors is also much

cleaner than old potentiometers and is easier to differentiate to find the tables velocity. This measured velocity could then be used in future controllers to actively set the desired speed of the machine instead of relying on open loop control set with previous calibration.

CHAPTER 7 - ANALOG MOTOR CONTROL

Introduction

An important part of implementing a control system is to try to minimize the delay between sampling data and issuing commands to the system. The ability of the angular measurements of the input shafts to be accurately resolved into the position of the table gave the real-time computer the position of the tool at 5kHz. However, commands to the motors had to be sent back to the weld computer and then directed to the correct motor drive which adds some delay to the system. This problem was made worse by the fact that the traverse, lateral, and spindle motor drives are all connected over a shared ModBus connection. This means messages to different motors cannot be sent at the same time as otherwise they would interfere with each other. The speed of which commands could be sent down this channel to an individual motor was found to be around 25 ms before commands would start to be not successfully received by the motor drives. Combine this with the fact that two separate commands are needed for motor speed and direction means that at least 150 ms is needed to ensure there is always enough time to send commands to all 3 motors. To help fix this large potential delay in sending motor commands a new way to control the lateral and traverse motors directly from the two analog outputs of the NI-6024E DAQ card was developed.

Implementation

This was done by having the C# code program the lateral and traverse drives before control of the machine was given to the real-time controller. The drives were programed to accept an analog input from 0-10V to control the speed of the motor and the direction of the motor was controlled by whether two terminals of the motor were connected. The DAQ card can output a voltage from -10-10V so a circuit was designed in KiCad to take the absolute value of the input voltage and activate a relay based on the sign of the input voltage. The circuit schematic, PCB design, component list, and price breakdown can be seen in APPENDIX D The part of the circuit responsible for taking the absolute value of the voltage was taken from The Art of Electronics [18], and pictures of an assembled circuit can be seen in Figure 40.

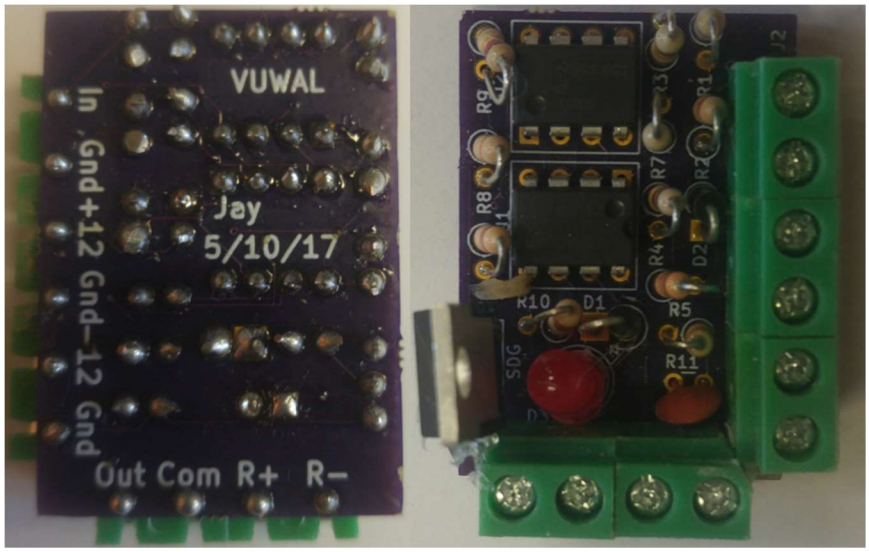


Figure 40: Assembled PCB for analog control of lateral and traverse motors

As can be seen Figure 40 this circuit is simply added into the system using the connected screw terminals with reference information printed on the bottom of the PCB to avoid confusion. It would have been easier to program the motors to run directly from a 0-10 volt signal where 0 volts would correspond to the maximum speed in one direction and 10 volts to the maximal speed in the other direction. The system programed this way would not need this separate circuit, however this method has some major downsides. If the welding computer were to crash or fail in that setup the analog outputs of the DAQ card would immediately fall to zero which in turn would send the machine out of control at maximum speed. This circuit remedy's that possibly catastrophic failure as a system crash or power loss will result in the stopped motors with the added benefit of increasing the resolution since the full range of the output is used to set the speed of the motors.

Conclusion

With this circuit in place the lateral and traverse drives receive commands directly from the real-time computer while the vertical and spindle have their commands routed to the weld computer when the machine is performing a weld. This reduces the amount of time commands are sent to the lateral and traverse motors while increasing the rate at which commands can be sent to the spindle motor. This means

new commands can be sent every 25 ms if needed as the spindle will never have to change direction during in a weld. In addition, this system improves the safety of the machine as the analog control will automatically shut off in the case of a power loss or if the software were to crash.

CHAPTER 8 - GROOVE TRACKING

Introduction

Previously all the through the tool tracking done in this lab has been done using the Weavetrack method. This is an extremum controller that seeks a local maximum or minimum in the welding process forces and torque. This force signature is unique to the welding setup used. For the case of tracking a T joint the tracked force is a local maximum due to a specialized clamping system as can be seen in Figure 41.

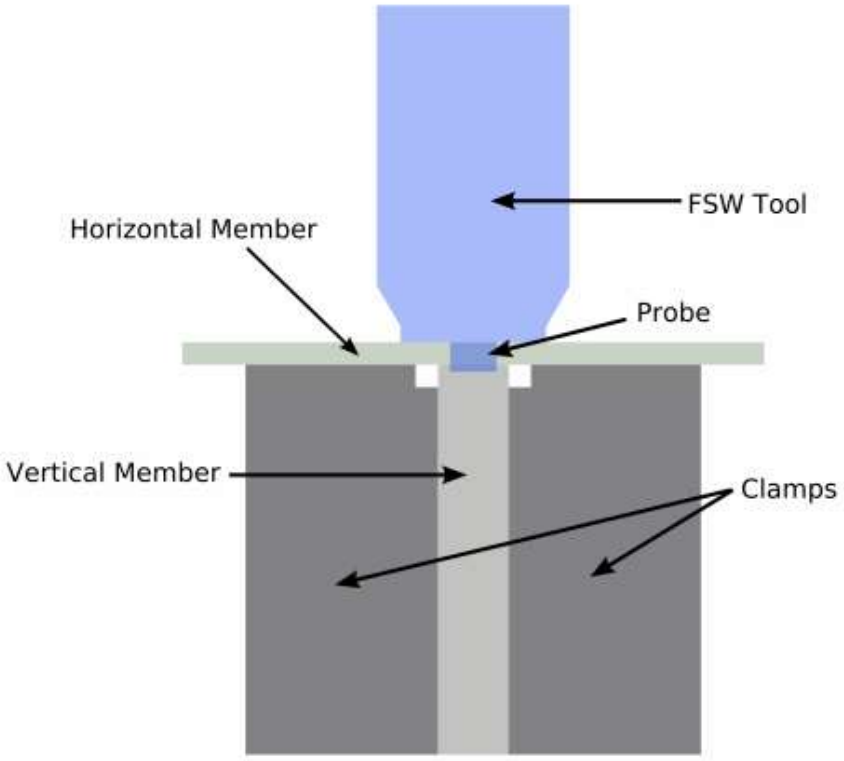


Figure 41: "Open air clamps" used for blind T-joint tracking [8]

This is the case because as the tool deviates from the weld center material the gaps left in the fixturing provide a place for material to be extruded. This causes the forces to diminish and provides a local maximum to be tracked along the joint. The reverse is true for tracking a groove. As the tool deviates from the location of the groove less material is extruded, and the weld forces increase making a local minimum to track. To find this local minimum or maximum the tools location is varied and by examining

the changes in the forces in relation to the change of position the extrema can be found. This type of tracking is useful for these weld configurations as the feature being tracked is completely obscured by the top material and optical methods would not work, and this method would automatically adapt to fix misalignments.

Controller Design

Weave track used a trapezoidal movement to vary the lateral position of the tool which can be seen in Figure 42.

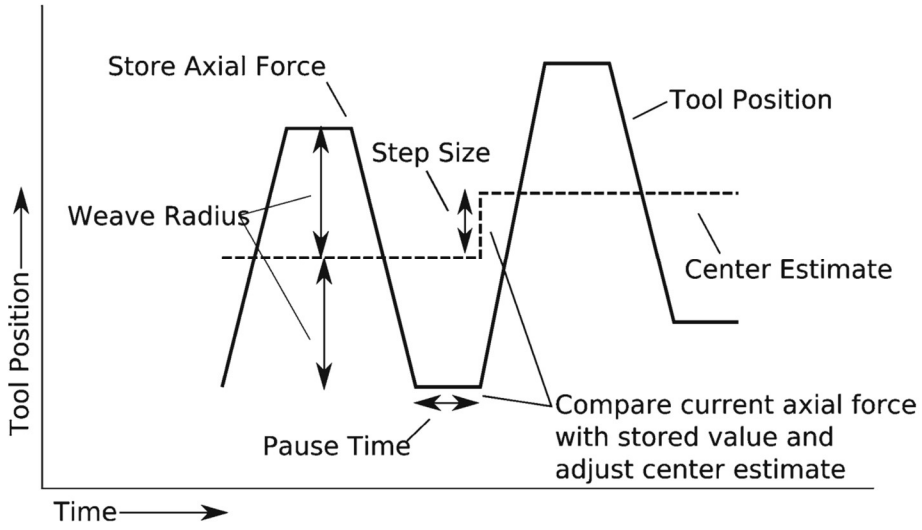


Figure 42: Illustration of two WeaveTrack cycles [9]

This movement profile has sharp movement transitions, and it was shown in Figure 32 and reported by others that this causes disturbances to occur in the measured forces [8]. An attempt to remedy this was to vary the tool position using a sinusoidal perturbation path in the lateral direction instead of a trapezoidal one with the hope that a smooth motion profile would minimize this type of disturbance. This level of control was now possible after the numerous improvements to the machines control system. After every period of the perturbation signal the forces would be examined depending if they were increasing or decreasing the center position was either advanced or retreated by one half the amplitude of the perturbation signal. The primary parameters for this controller are the transverse speed, the amplitude of the perturbation signal, and frequency of the perturbation signal. Varying these parameters affected the

controller's ability to track the groove as well as the overall quality of the weld. The complete controller can be seen in Figure 43.

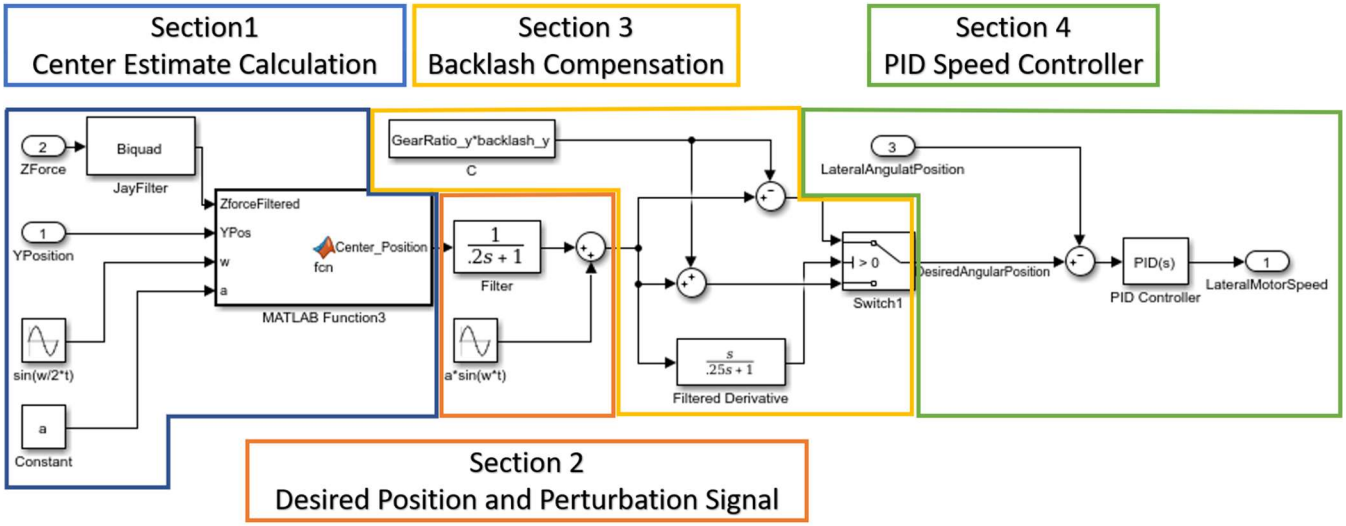


Figure 43: Block diagram of extremum controller

Section 1 of the controller takes in axial force and the lateral position of the tool then calculates the current center estimate every period of the perturbation signal. The center estimate is then passed through a filter and combined with the perturbation signal to find the desired position of the tool in section 2 of the controller. The filter is important so that the tool desired position is always continuous and never causes a sudden jump in the desired position. Section 3 of the controller is an attempt to compensate for the backlash in the system. The logic for this is that is it assumed that when the desired velocity is positive or negative the machine will need to move in that same direction to follow the desired position. Then depending on the sign of the desired speed the backlash times the gear ratio is added or subtracted from the desired position. This will provide a discontinuity in the signal to be tracked however in section 4 the angular position times the gear ratio is subtracted to provide the error signal for the PID controller. The jump in the tracking signal when changing direction cause the machine to quickly run through the slack caused by backlash and then the PID stops the tool from over shooting and changing direction too quickly.

Results

The ability of this controller to track a groove for extrusion was tested by letting the controller correct itself from being initially offset from the groove location. After numerous tests varying the parameters a working a set was found that could track the groove and still give decent weld quality. The traverse speed was set at 2 IPM and the perturbation signal was set with an amplitude of 0.05 inches with a frequency of 1 rad/s. Plots for three welds can be seen in Figure 44.

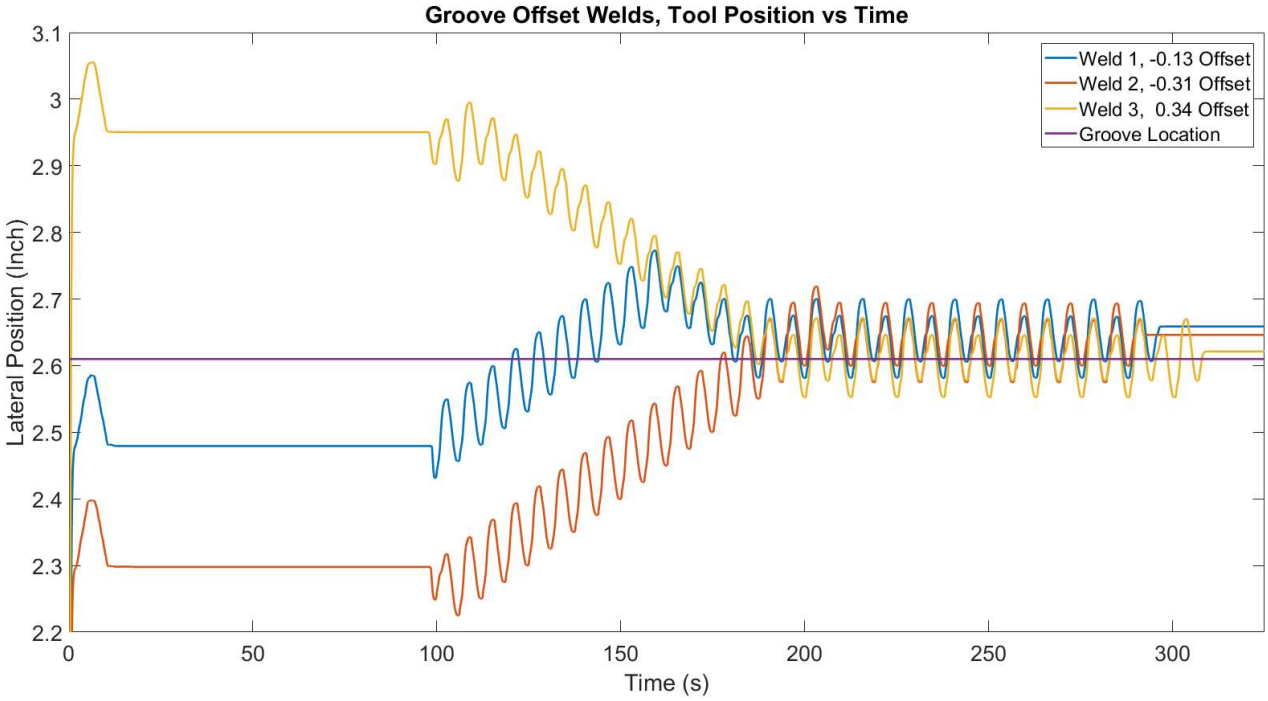


Figure 44: Tool lateral position vs time during extrusion tracking

These three welds show how all the welds converted to the same location slightly offset from the center location of the groove. This corresponds to some preliminary findings in this lab where maximum extrusion for this type of tool occurred with the pin slightly of center of the groove's center location. The controller had some overshoot which is quite prominent in weld 1, but further experimentation with different parameters might yield better performance. Figures for the two welds with the greatest offsets, 2 and 3, are present to give a visual representation of how this automatic realignment to the grooves location greatly improved the amount of extrusion. A top, bottom and side view of weld three can be seen

in Figure 45 and the same for weld two is given in Figure 46.

Figure 45: Top, bottom, and side view of weld 3

Figure 46: Top Bottom and Side view of Weld 2

These pictures show how at the start of the weld almost no material is extruded into the groove and how the groove becomes entirely filled as the controller converges to follow the local minimum of the axial force.

CHAPTER 9 - CONCLUSION

The redesign of the welding control systems software and hardware improves the and expands the capability of the welding machine. Having a separate real-time computer controlling the weld allows for faster sampling rates of 5kHz and combined with the new ability to measure the angle of the tool allow the process forces to be monitored at much greater detail. This will give better insight for tool design, identifying weld defects, and provide new avenues for control research. Moving controller design away from the C# program to Simulink allows for faster development of complex controllers and less training is required to develop new controls. The developed magnetic angle sensors also allowed the backlash of the machine to be quantified and provide a higher resolution signal to monitor the welding table's position. In addition, various safety improvements made throughout the system which reduces the risk of damage to the machine and injury to operators. Finally, a control scheme was developed to automatically track the friction stir extrusion process and tested with success using the improved control system for the friction stir welding machine.

CHAPTER 10 - FUTURE WORK

Future work includes developing new complex controls using this updated system with the ability to quickly incorporate new sensors to the system as currently the NI-6023E and PCI-MIO-16E DAQ cards are unused and can provide an additional 32 analog channels to sample from. The new ability to measure forces as the tool rotates could provide valuable data in the terms of tool design, flaw detection, and the overall understanding and modeling of the friction stir welding process. While many improvements were made to the control system undoubtedly there are still flaws to be worked out in the system, and eventual less reliance on the C# program during the weld will be ideal. With the reintegration of the Kistler dynamometer and the magnetic angle sensors there are many of the same measurements that can be made with different sensors. This would allow the possibility of sensor fusion to improve the accuracy of measurements a provide a way to automatically detect failures of sensors which would provide greater safety. Many improvements could still be made to the extrusion tracking process improving its speed and robustness. The backlash companion used in this controller was crude and further refinement would benefit the overall control of the position of the tool as backlash was found in all the axis of the machine. Additionally, the use of pattern recognition to locate the grooves location would be an interesting research avenue.

BIBLIOGRAPHY

[1] W. R. Longhurst, C. D. Cox, B. T. Gibson, G. E. Cook, A. M. Strauss and D. R. DeLapp, "Applied torque control of friction stir welding using motor current as feedback," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* vol. 228, no. 8, pp. 947-958, 2014.

[2] P. A. Fleming, *MONITORING AND CONTROL IN FRICTION STIR WELDING,* Vanderbilt University: Ph.D. dissertation, 2009.

[3] R. W. Longhurst, *FORCE CONTROL OF FRICTION STIR WELDING,* Vanderbilt University,: Ph.D. dissertation, 2009.

[4] B. T. Gibson, D. H. Lammlein, T. J. Prater, R. R. Longhurst, C. D. Cox, M. C. Ballun, K. J. Dharmaraj, G. E. Cook and A. M. Strauss, "Friction stir welding: Process, automation, and control," *Journal of Manufacturing Processes,* vol. 16, no. 1, pp. 56-73, 2014.

[5] N. Balasubramanian, B. Gattu and R. S. Mishra, "Process forces during friction stir welding of aluminum alloys," *Science and Technology of Welding and Joining,* vol. 14, no. 2, pp. 141-145, 2013.

[6] W. T. Evans, C. Cox, B. T. Gibson, A. M. Strauss and G. E. Cook, "Two-sided friction stir riveting by extrusion: A process for joining," *Journal of Manufacturing Processes,* vol. 23, pp. 115-121, 2016.

[7] W. T. Evans, B. T. Gibson, J. T. Reynolds, A. M. Strauss and G. E. Cook, "Friction Stir Extrusion: A new process for joining dissimilar materials," *Manufacturing Letters,* vol. 5, pp. 25-28, 2015.

[8] P. A. Fleming, D. H. Lammlein, D. M. Wilkes, G. E. Cook, A. M. Strauss, D. R. DeLapp and D. A. Hartman, "Misalignment detection and enabling of seam tracking for friction stir welding," *Science and Technology of Welding and Joining,* vol. 14, no. 1, pp. 93-96, 2009.

[9] P. A. Fleming, C. E. Hendricks, D. M. Wilkes, G. E. Cook and A. M. Strauss, "Automatic seam-tracking of friction stir welded T-joints," *Int J Adv Manuf Technol,* vol. 45, pp. 490-495, 2009.

[10] P. A. Fleming, C. E. Hendricks, G. E. Cook, D. M. Wilkes, A. M. Strauss and D. H. Lammlein, "Seam-Tracking for Friction Stir Welded Lap Joints," *ASM International,* vol. 19, no. 8, pp. 1128-1132, 2009.

[11] B. Gibson, C. Cox, M. Ballun, G. Cook and A. Strauss, "Automatic Tracking of Blind Sealant Paths in Friction Stir Lap Joining," *AIAA Journal of Aircraf,* vol. 51, no. 3, pp. 824-832, 2014.

[12] B. T. Gibson, *CUSTOM LOW-COST FORCE MEASUREMENT METHODS IN FRICTION STIR WELDING,* Vanderbilt University: M.S. Thesis, 2011.

[13] P. Sinclair, *HEATED FRICTION STIR WELDING: AN INVESTIGATION INTO HOW PREHEATING ALUMINUM 6061 AFFECTS PROCESS FORCES,* Vanderbilt University: M.S. Thesis, 2009.

[14] P. Kah, R. Rajan, J. Martikainen and S. Raimo, "Investigation of weld defects in friction-stir welding and fusion welding of aluminium alloys," *International Journal of Mechanical and Materials Engineering,* vol. 10, no. 26, 2015.

[15] B. T. Gibson, C. D. Cox, W. R. Longhurst, A. M. Strauss and G. E. Cook, "Exploiting robotic link deflection for low-cost force measurement in manufacturing," *Measurement,* vol. 45, no. 1, pp. 140-143, 2011.

[16] D. G. Hattingh, T. v. Niekerk, C. Blignault, G. Kruger and M. N. James, "Analysis of the FSW Force Footprint and its Relationship with Process Parameters to Optimise Weld Performance and

Tool Design," *Welding in the World,* vol. 48, no. 1-2, pp. 50-58, 2004.

[17] D. H. Lammlein, *FRICTION STIR WELDING OF SPHERES, CYLINDERS, AND T-JOINTS: DESIGN, EXPERIMENT, MODELLING, AND ANALYSIS,* Vanderbilt University: Ph.D. dissertation, 2010.

[18] P. Horowitz and W. HIll, The Art of Electronics: Second Edition, Cambridge: Cambridge University Press, 1989.

[19] J. H. Yan, M. A. Sutton and A. P. Reynolds, "Processing and banding in AA2524 and AA2024 friction stir welding," *Science and Technology of Welding and Joining,* vol. 12, no. 5, pp. 390-401, 2007.

[20] P. A. Fleming, C. E. Hendricks, D. M. Wilkes, G. E. Cook and A. M. Strauss, "Automatic seam-tracking of friction stir welded T-joints," *Int J Adv Manuf Technol,* vol. 45, no. 5-6, pp. 490-495, 2009.

APPENDIX

APPENDIX A: Backlash and gear ratio

Matlab code that was used to calculate backlash and fit lateral angular data to the data from the potentiometer

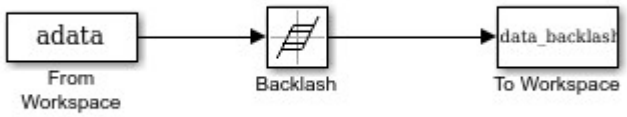
```
clc
clear blarray sqearray
fontsize=22;
%time from slrt has a few messed up values
%this sould correct for those time values that
%are equal to -1
Ts=0.0002;%sample time
t=NIDaq.data(:,13);%get time
n=1:length(t);
[k f]=find((t<0).*1);%find times that have messed up
%fix times with the correct ones
for i=1:length(k)
    t(k(i))=t(k(i)+1)-.0002;
end
%
plot(k,f,'o',n,t)

traf=TRA';
latf=LAT';
verf=VER';

VerLoc=CSharp.data(:,2);
LatLoc=CSharp.data(:,3);
TraLoc=CSharp.data(:,4);
T=CSharp.data(:,14);

open_system('simbacklash');

xa=latf;
%linear interpolate Csharp data to be the same size as NIDaq
x=interp1(T,LatLoc,t);

%remove NAN elements from the interoperation process and resize the data
% to match
row= find(isnan(x));
x=x(1:row(1)-1);
xa=xa(1:row(1)-1);
t=t(1:row(1)-1);

%get data ready to be used in backlash simulink model
adata=[t xa];
endtime=t(end);
%filter data
sys = tf([1],[0.1 1]);
sysd = c2d(sys,Ts);
xa=filter(sysd.Numerator{1},sysd.Denominator{1},xa);
```

```
%get dirivative of angle data
xap = gradient(xa,0.0002);
sys = tf([1],[0.1 1]);
sysd = c2d(sys,Ts);
xapf=filter(sysd.Numerator{1},sysd.Denominator{1},xap);
%find places where the data is not changing
[zzz kkk]=find((abs(xapf)<.03));

%find segments of data that are not changing and add the index of when
% these segments start and end to an array
j=1;
clear index
index(1,1)=zzz(1);
for i=2:length(zzz)
    if(zzz(i)-zzz(i-1)>10000)
        index(j,2)=zzz(i-1);
        j=j+1;
        index(j,1)=zzz(i);
    end
end
index(end,2)=zzz(end);

qqq=index(:,1);
qq=index(:,2);
%find the means of the data for the stationary portions and replace
%values with said mean
xw=x;
zzz=[];
%remove first index values
index(1,:)=[];
clear xpoints
for i=1:length(index)
    xw(index(i,1):index(i,2))=mean(xw(index(i,1):index(i,2)));
    xpoints(i)=mean(x(index(i,1):index(i,2)));
    zzz=[zzz index(i,1):index(i,2)];
end

plot(t,x,t,xw,t(qqq),x(qqq),'o',t(qq),x(qq),'*')
plot(t,x,t(zzz),xw(zzz),'.')

%loop to find optimal backlash that matches the system
backlash_x=0;
sqe1=1;
sqe2=2;
% rr=[1 -.1 .01 -.001];
rr=[1 -.1 .01];
backlash_x=backlash_x-rr(1);
Ts=0.0002;
i=1;
xraw=x;
xmean=xw;
%get data ready to be used in backlash simulink model
adata=[t xa];
endtime=t(end);
```

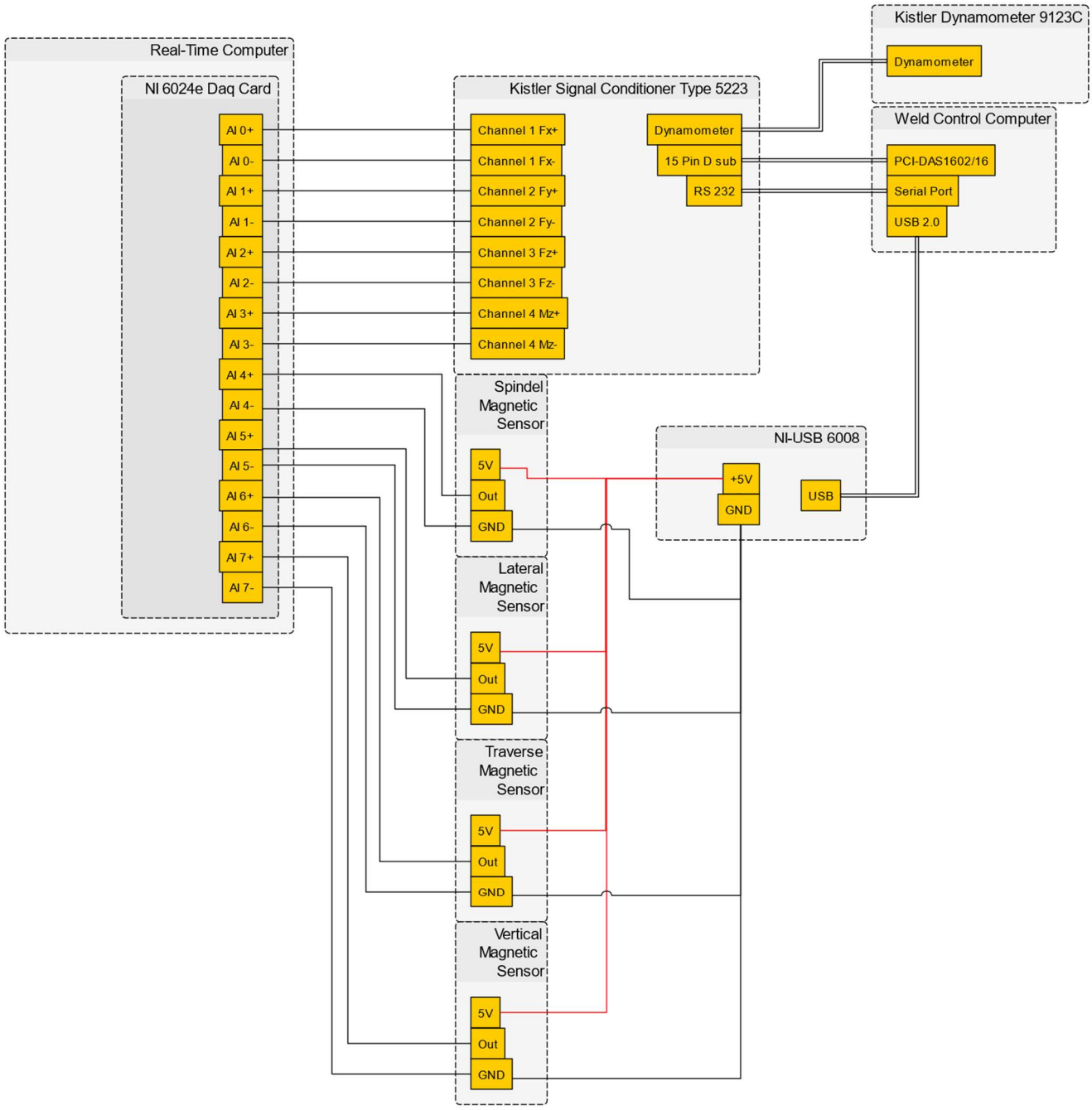
```
backlash=0:.005:2;
set_param('simbacklash', 'StopTime', num2str(endtime))
sim('simbacklash')
xa=adata_backlash;
s=size(adata_backlash);
for i=1:s(2)
    clear xapoints
    xa=adata_backlash(:,i);
    for j=1:length(index)
        xapoints(j)=mean(xa(index(j,1):index(j,2)));
    end
    xa=xapoints';
    x=xpoints';

         % minimize l-2 norm
        psi=[xa ones(length(xa),1)];
        theta=inv(psi'*psi)*psi'*x;
        thetaArray(i,:)=theta;

        xa=theta(1)*xa+theta(2);
        sqe=xa-x;
        sqe1=sum(sqe.*sqe)/length(sqe);%average of the squared error
        sqe2=max(sqe);
        sqearray(i,1)=sqe1;
        sqearray(i,2)=sqe2;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1)
plot(backlash*360/5,sqearray(:,1),'.')
title({'Lateral';'Average Squared Error of Least Square Regression vs
Backlash'},'FontSize',fontsize)
xlabel('Backlash in Degrees','FontSize',fontsize) % x-axis label
ylabel('Average of the squared error','FontSize',fontsize) % y-axis label
%legend('C#','Backlash','No Backlash')
xt = get(gca, 'XTick');
set(gca, 'FontSize', 20)
xlim([0 110])
x=xraw;
[n,k]=min(sqearray(:,1));
xab=adata_backlash(:,k);
theta=thetaArray(k,:)
xa=theta(1)*xab+theta(2);
%
xab=adata_backlash(:,1);
theta=thetaArray(1,:)
xa2=theta(1)*xab+theta(2);
figure(2)

clc
LATbacklash=backlash;
LATsqearray=sqearray;
plot(T,LatLoc,'.',t,xa,t,xa2)
title('Table Height vs Time','FontSize',fontsize)
xlabel('Time (s)','FontSize',fontsize) % x-axis label
ylabel('Height (inch)','FontSize',fontsize) % y-axis label
```

```
legend('C#','Backlash','No Backlash')
xt = get(gca, 'XTick');
set(gca, 'FontSize', 20)
ylim([2.3 2.65])
thetaArray(1,:)
thetaArray(k,:)
backlash(k)
```

Simulink model used to model backlash

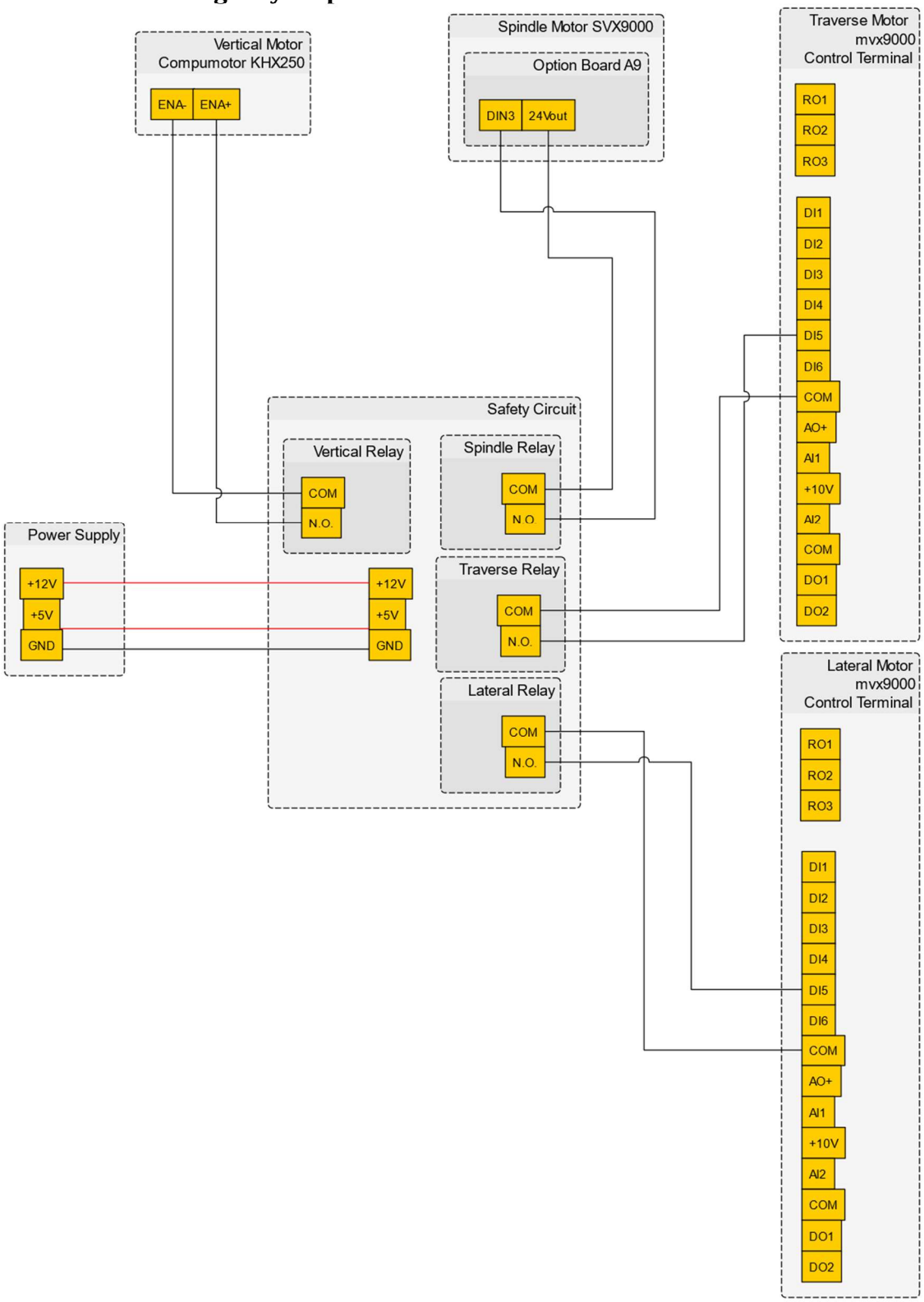


APPENDIX B: Sensor Integration Schematic

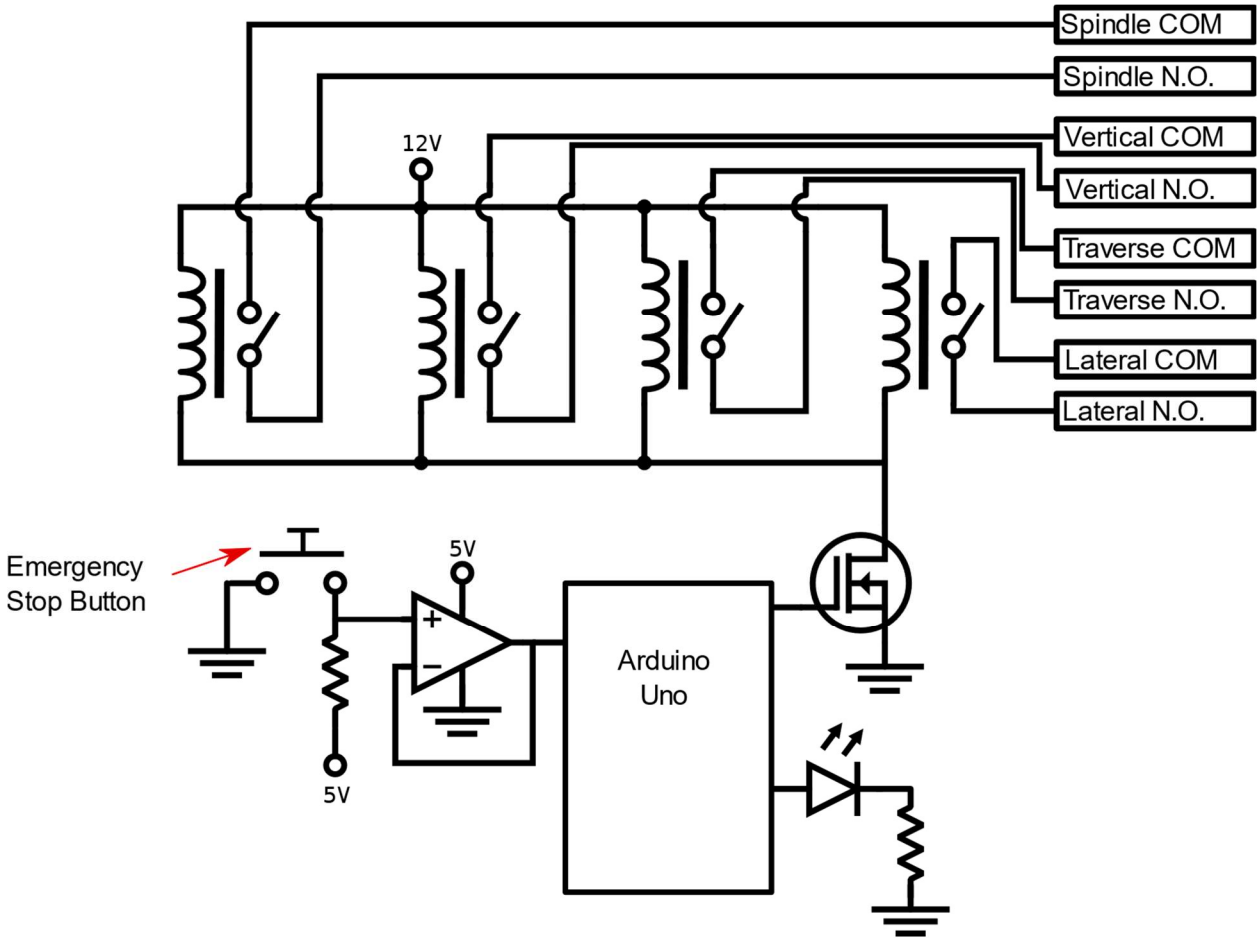


Appendix Figure 1: Schematic of Dynamometer and Angle sensor connections

APPENDIX C: Emergency stop

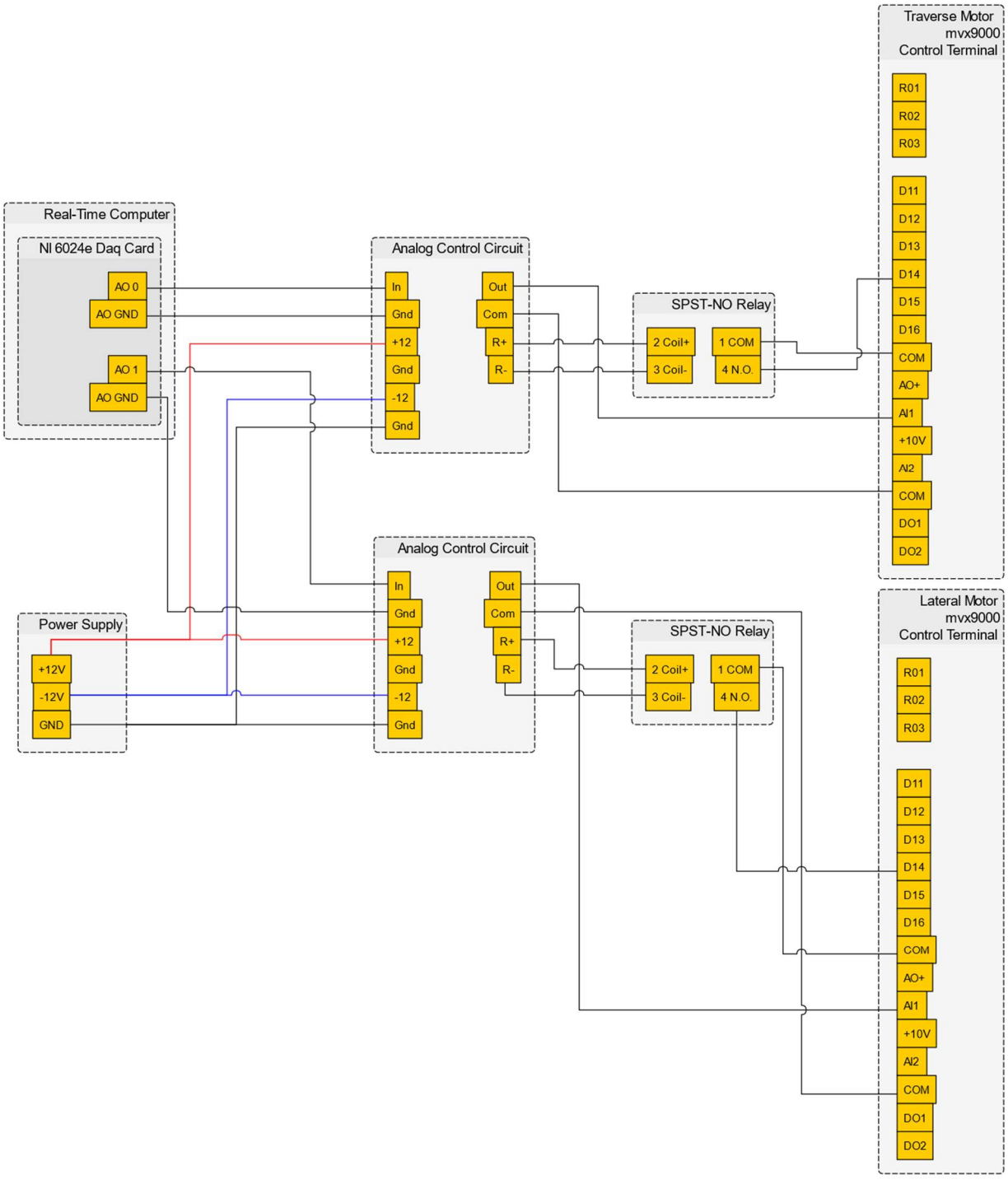


Appendix Figure 2: Schematic Showing how the emergency stop safety circuit is incorporated



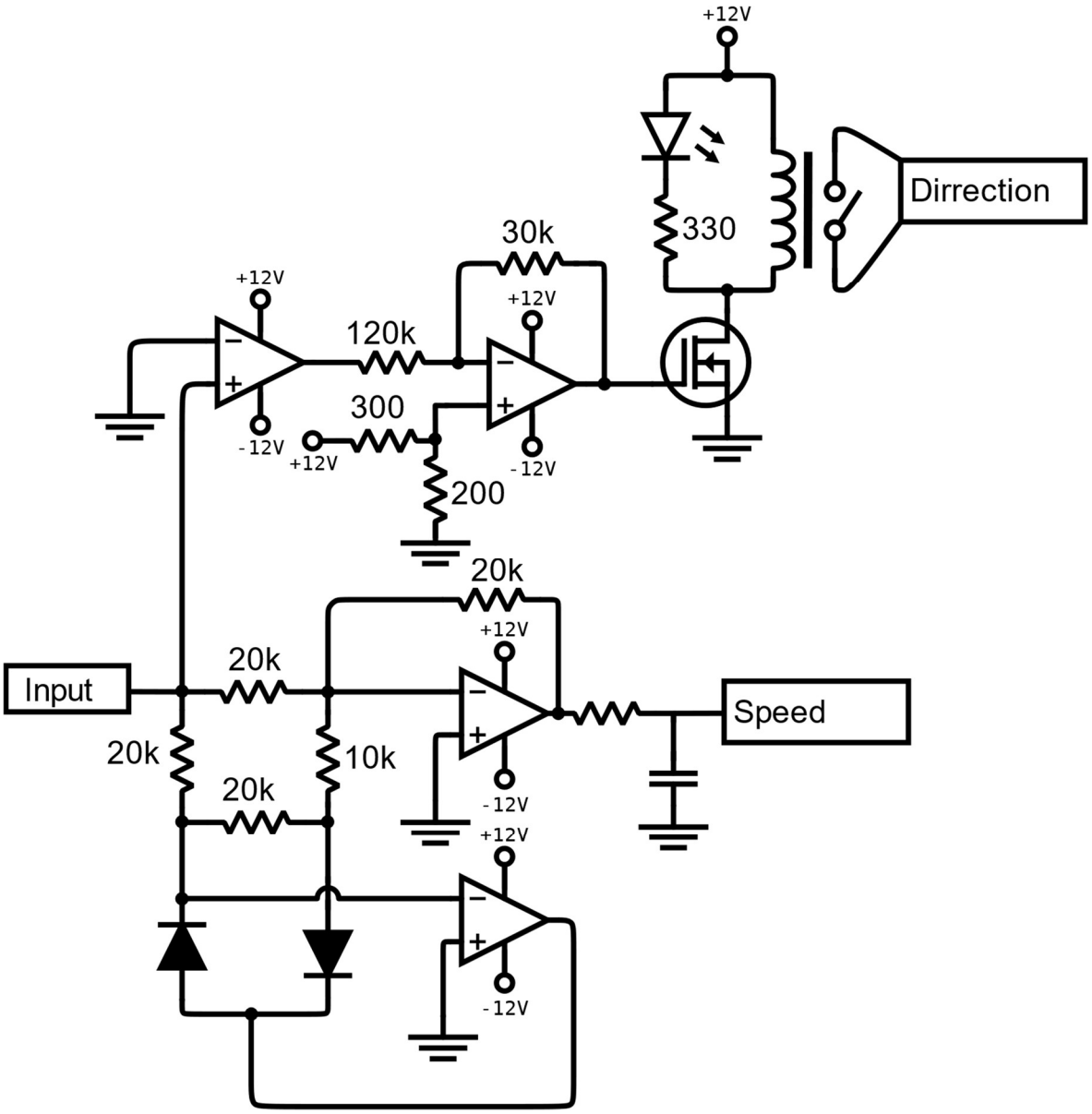
Appendix Figure 3:Circuit Diagram of the Emergency stop safety circuit

APPENDIX D: Analog Circuit design for Lateral and Traverse Motors

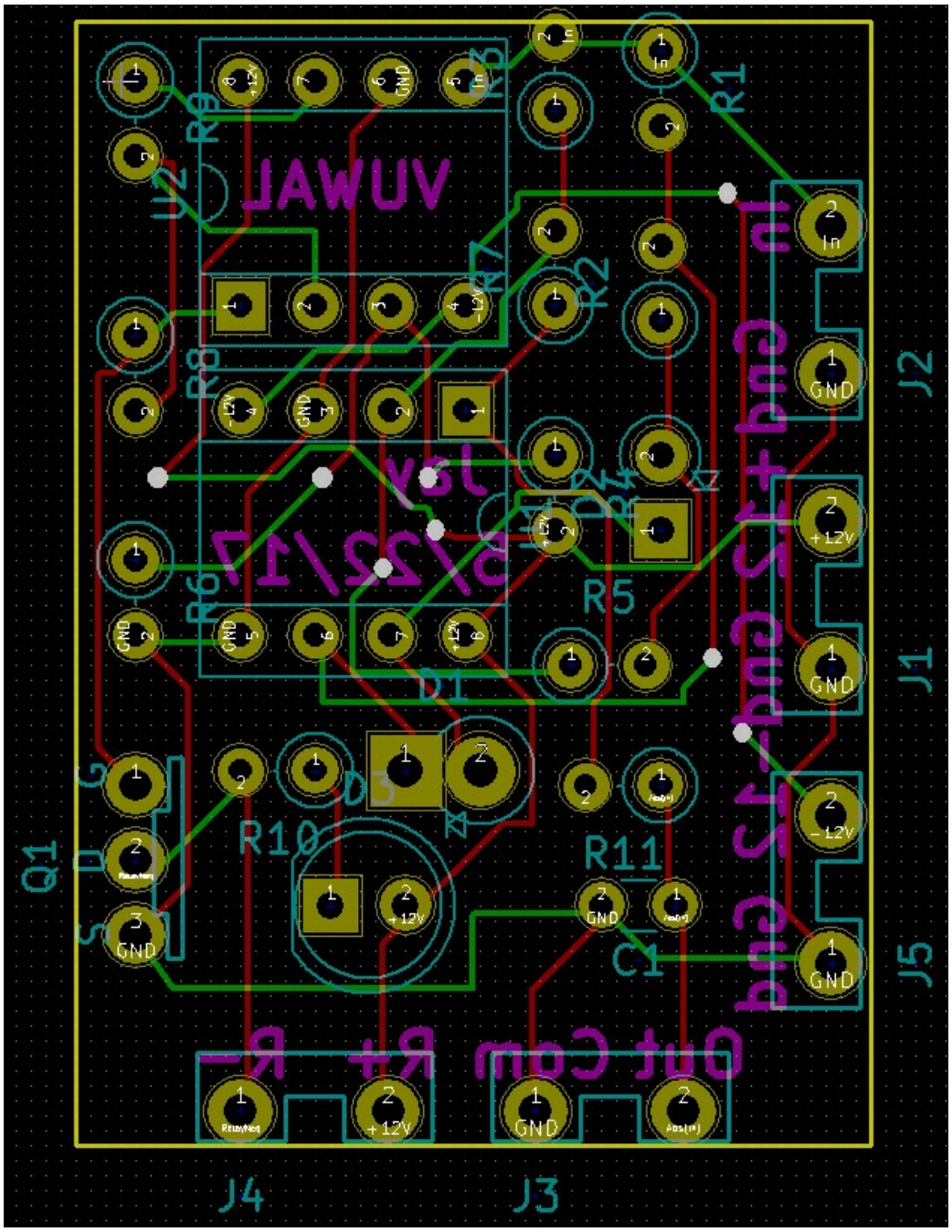


Appendix Figure 4:Schematic Diagram of how analog control circuit is connected

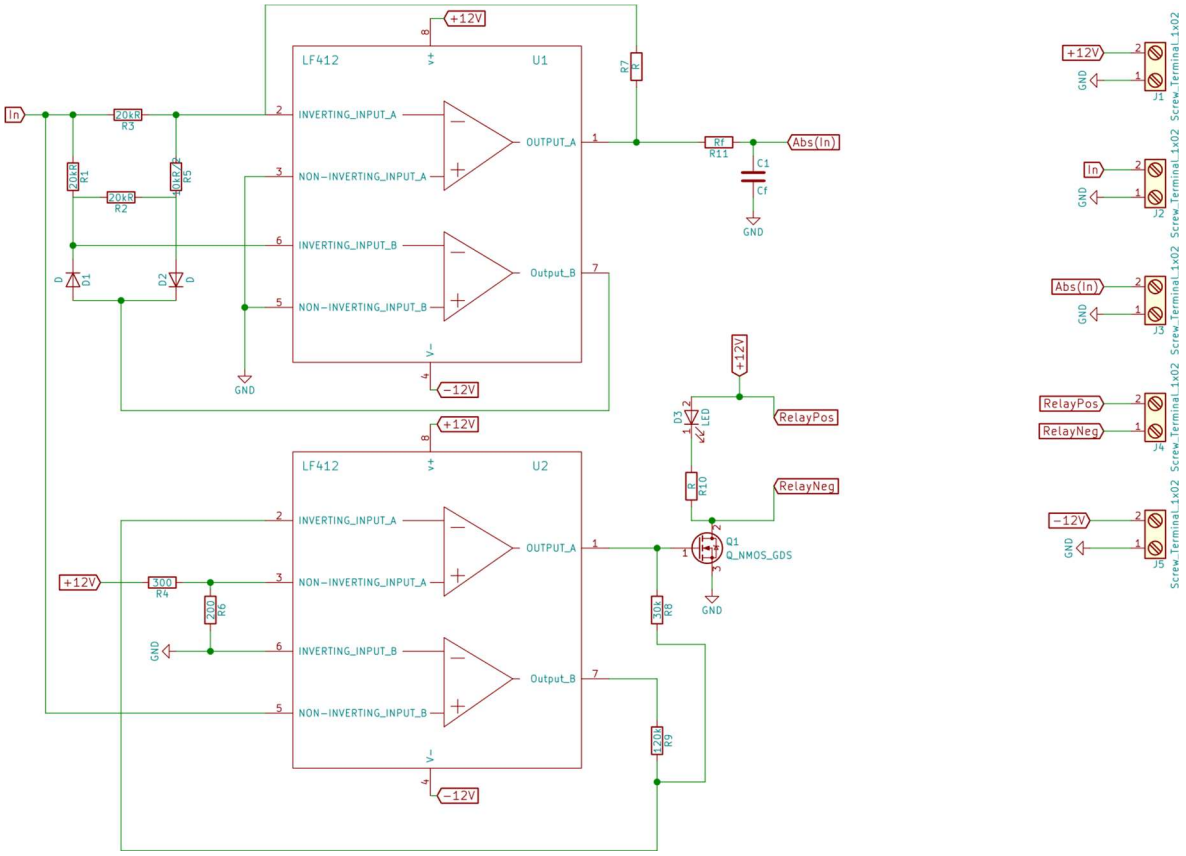
This appendix shows the circuit design that was used to take the analog voltage from the real time computer and then interface with the lateral and traverse motors along with a schematic diagram of how the circuit was integrated.



Appendix Figure 5:Circuit diagram for analog control of lateral and traverse motors

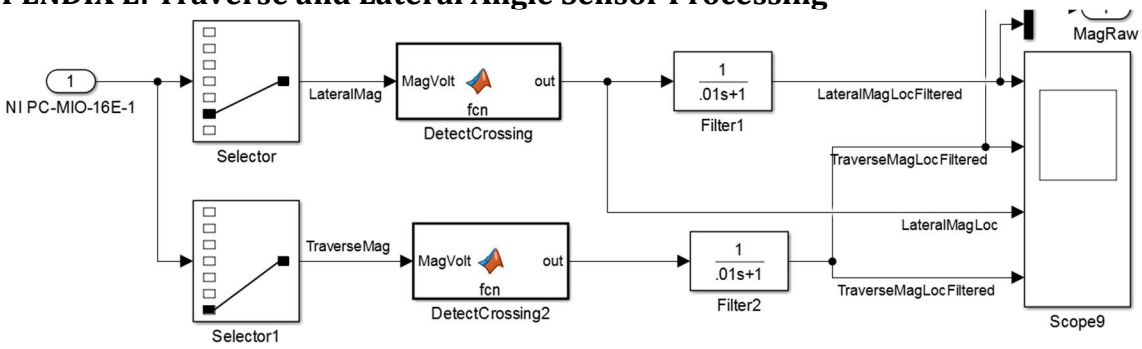


Appendix Figure 6: KiCad PCB Design for analog control of lateral and traverse motors



Appendix Figure 7:KiCad Schematic for analog voltage control of lateral and traverse motors

APPENDIX E: Traverse and Lateral Angle Sensor Processing



Appendix Figure 8:Section of Simulink model for processing signal from magnetic sensor

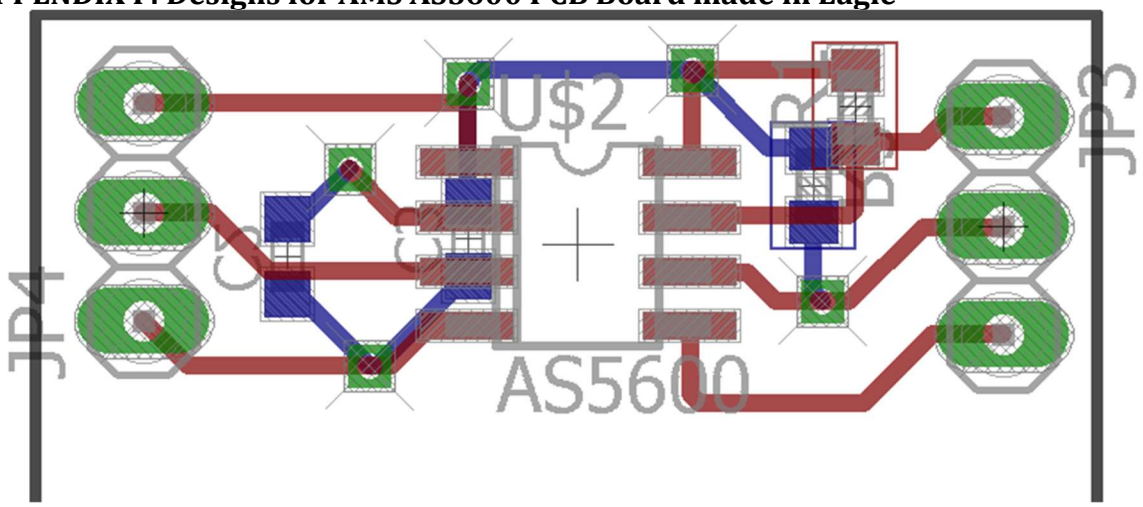
Editor - Process and FilterRaw Mag Data/DetectCrossing

Process and FilterRaw Mag Data/DetectCrossing

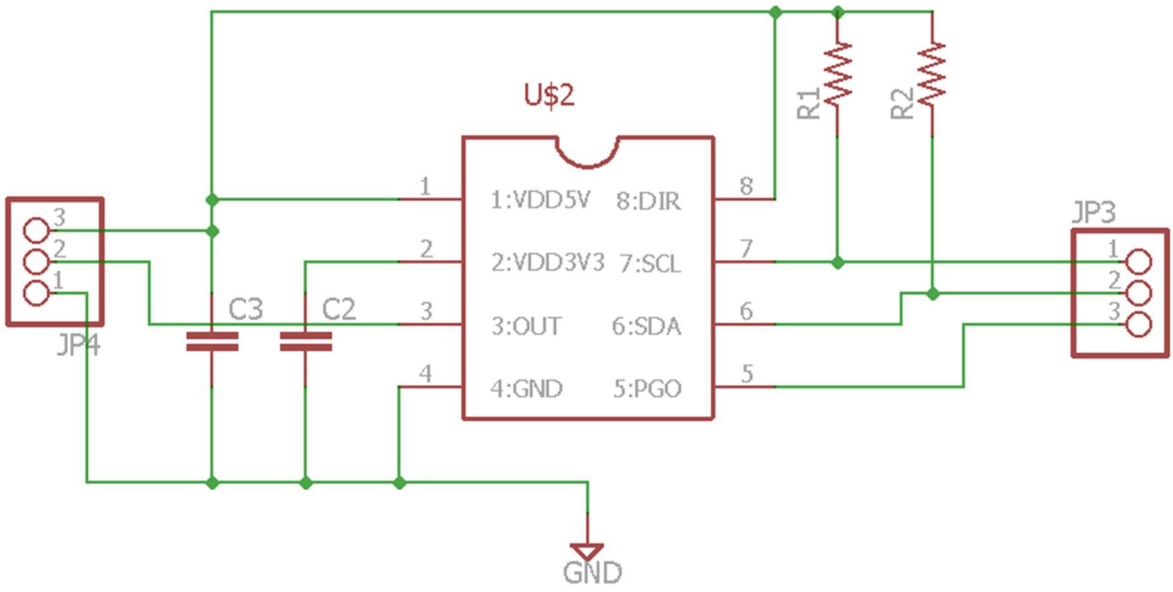
```
function out = fcn(MagVolt)
persistent M0 M1 M2 MagLoc max count
if isempty(M0)
    M0=MagVolt;
end
if isempty(max)
    max=0;
end
if isempty(count)
    count=0;
end
if isempty(M1)
    M1=MagVolt;
end
if isempty(M2)
    M2=MagVolt;
end
if isempty(MagLoc)
    MagLoc=MagVolt;
end
M0=MagVolt;
if((abs(M0-M2)>4)||abs(M0-M1)>4)
    if((abs(M0-M2)>4)&&~count)
        MagLoc = MagLoc - (M1-M2);
        count=1;
    end
else
    MagLoc = MagLoc + M0-M1;
    count=0;
end

M2=M1;
M1=M0;
out=MagLoc;
```

Appendix Figure 9:Logic for detecting jumps in saw wave analog signal to create a linear signal

APPENDIX F: Designs for AMS AS5600 PCB Board made in Eagle



Appendix Figure 10: Eagle PCB design for magnetic rotary sensor



Appendix Figure 11:Eagle circuit schematic for magnetic rotary sensor

APPENDIX G: Leveling Welding Table

Appendix Table 1:Raw height measurements made for leveling the welding table

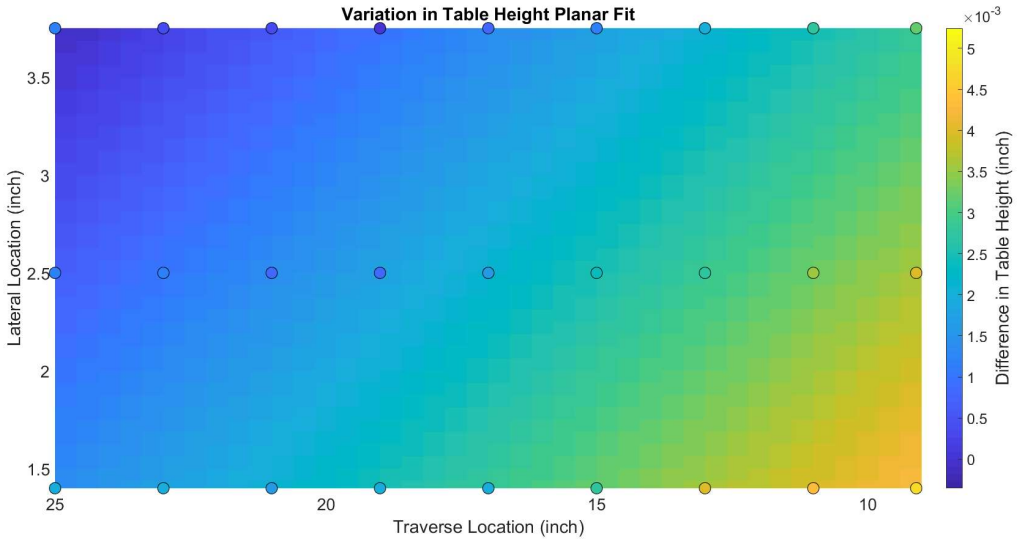
X	Y	No Shims Height	Shim 1 Height	Shim 2 Height	Shim 3 Height	Final Shim Height
9.1	1.4	6.909685035	6.910078736	6.909685035	6.909685035	6.909685035
11	1.4	6.910078736	6.910078736	6.909685035	6.909685035	6.909685035
13	1.4	6.910472437	6.910472437	6.910078736	6.910472437	6.909685035
15	1.4	6.911653539	6.911653539	6.911259838	6.911259838	6.910472437
17	1.4	6.912440941	6.91204724	6.91204724	6.91204724	6.910866138
19	1.4	6.912440941	6.912440941	6.912440941	6.912440941	6.910866138
21	1.4	6.912834641	6.912834641	6.912440941	6.912834641	6.910472437
23	1.4	6.912440941	6.912440941	6.912440941	6.91204724	6.909685035
25	1.4	6.912440941	6.912440941	6.911653539	6.910866138	6.908897634
25	2.5	6.913228342	6.912834641	6.91204724	6.911259838	6.909291334
23	2.5	6.913228342	6.913228342	6.912440941	6.912440941	6.910078736
21	2.5	6.913622043	6.913228342	6.912834641	6.912834641	6.910866138
19	2.5	6.913622043	6.913228342	6.912834641	6.912834641	6.911259838
17	2.5	6.912834641	6.912834641	6.912440941	6.912440941	6.911259838
15	2.5	6.91204724	6.91204724	6.911259838	6.911259838	6.910472437
13	2.5	6.911653539	6.911653539	6.910866138	6.910866138	6.910472437
11	2.5	6.910866138	6.910866138	6.910078736	6.910078736	6.910078736
9.1	2.5	6.910472437	6.910472437	6.910078736	6.910078736	6.909685035
9.1	3.75	6.911259838	6.911259838	6.910078736	6.910078736	6.910078736
11	3.75	6.911653539	6.911259838	6.910078736	6.910472437	6.910078736
13	3.75	6.912440941	6.911653539	6.910866138	6.910866138	6.910078736
15	3.75	6.913228342	6.912440941	6.911653539	6.911653539	6.910866138
17	3.75	6.913622043	6.913622043	6.912440941	6.912440941	6.910866138
19	3.75	6.914409445	6.913622043	6.912834641	6.912834641	6.910866138
21	3.75	6.914015744	6.914015744	6.913228342	6.913228342	6.910866138
23	3.75	6.914015744	6.913228342	6.912440941	6.91204724	6.909685035
25	3.75	6.913228342	6.912440941	6.911259838	6.910472437	6.908503933

Matlab code used analyzing raw data and creating figures

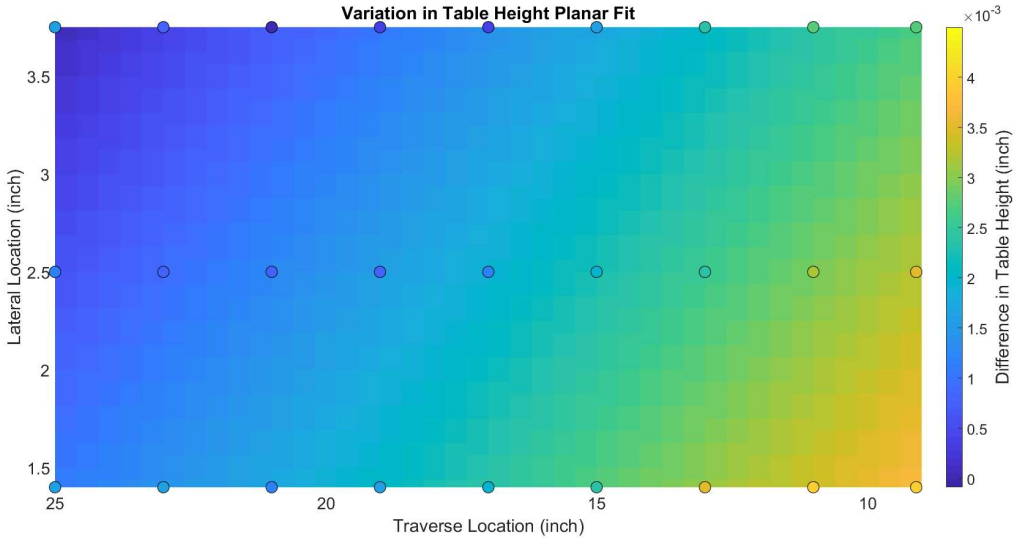
```
clear

clc

fontsize=22;

% name='TableTest2017_Jun_21_9_45_36_A'; %Noshims

% name='TableTest2017_Jun_21_10_20_49_A';%Shim Test 1

% name='TableTest2017_Jun_21_10_55_45_A';%Shim Test 2

% name='TableTest2017_Jun_21_11_23_38_A';%Shim Test 3

name='TableTest2017_Jun_21_12_00_41_P'; %Shim Test 4

fileID = fopen(strcat('D:\Users\Jay\Desktop\New folder (2)\',name,'.txt'));

c= textscan(fileID,'%f %f %f',...

'Delimiter','\t','EmptyValue',-Inf,'headerLines',1);

fclose(fileID);

y=c{1};

x=c{2};

z=-c{3};

z=z-min(z);

dotsize = 200;

figure(2)

sf = fit([x, y],z,'linearinterp');

scatter3(x(:), y(:), z(:)+.003, dotsize, z(:), 'filled');

hold on

scatter3(x(:), y(:), z(:)+.003, dotsize, z(:), 'k');

[X,Y] = meshgrid(1.4:.05:3.75,9.1:.1:25);

Z=sf(X,Y);

s=surf(X,Y,Z)

s.EdgeColor = 'none';

set(gca,'XLim',[1.4 3.75],'YLim',[9 25],'ZLim',[-inf inf])

set(gca, 'FontSize', 20)

% plot(sf)

hold off

title('Variation in Table Height Piecewise Interpolation','FontSize',fontsize)

xlabel('Lateral Location (inch)','FontSize',fontsize)

ylabel('Traverse Location (inch)','FontSize',fontsize) % y-axis label

c = colorbar;
```

```
c.Label.String = 'Difference in Table Height (inch)';
c.Label.FontSize=fontsize;
view(-90,90)
figure(1)
sf = fit([x, y],z,'poly11')
scatter3(x(:), y(:), z(:)+.002, dotsize, z(:), 'filled');
hold on
scatter3(x(:), y(:), z(:)+.002, dotsize, z(:), 'k');
s=plot(sf);
s.EdgeColor = 'none';
set(gca,'XLim',[1.4 3.75],'YLim',[9 25],'ZLim',[-inf inf])
title('Variation in Table Height Planar Fit','FontSize',fontsize)
xlabel('Lateral Location (inch)','FontSize',fontsize) % x-axis label
ylabel('Traverse Location (inch)','FontSize',fontsize) % y-axis label
c = colorbar;
c.Label.String = 'Difference in Table Height (inch)';
c.Label.FontSize=fontsize;
xt = get(gca, 'XTick');
set(gca, 'FontSize', 20)
hold off
view(-90,90)
coe=coeffvalues(sf);
LaterlCoe=coe(2)*1000
TraverseCoe=coe(3)*1000
```

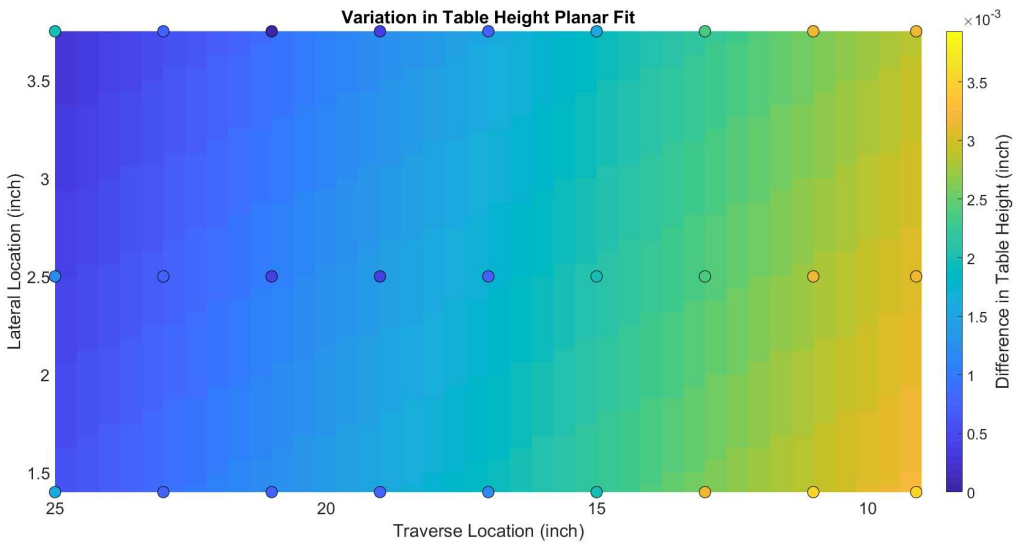
Figures showing the planar fit of all tests. Shows how the table was progressively made more level as more shims were added.



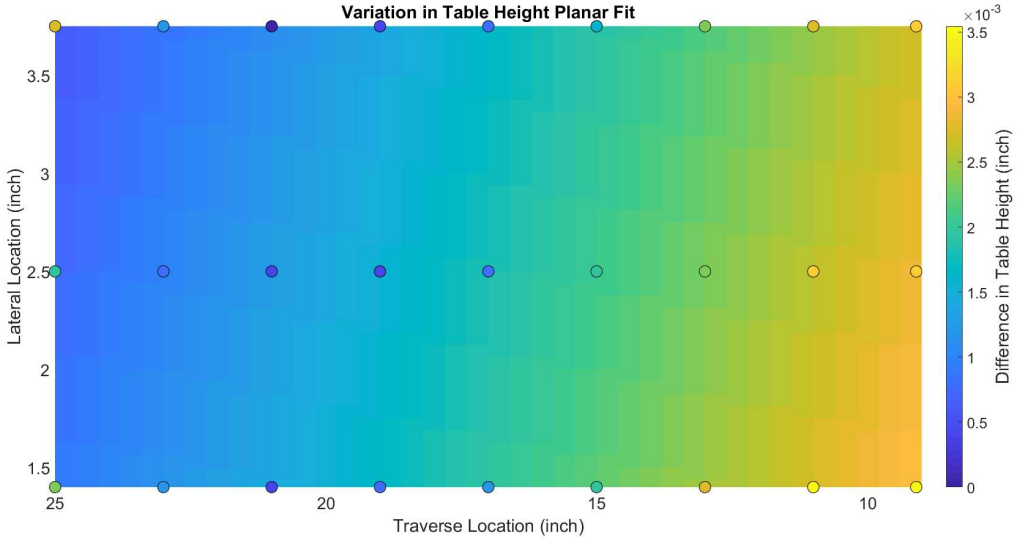
Appendix Figure 12: No shims added



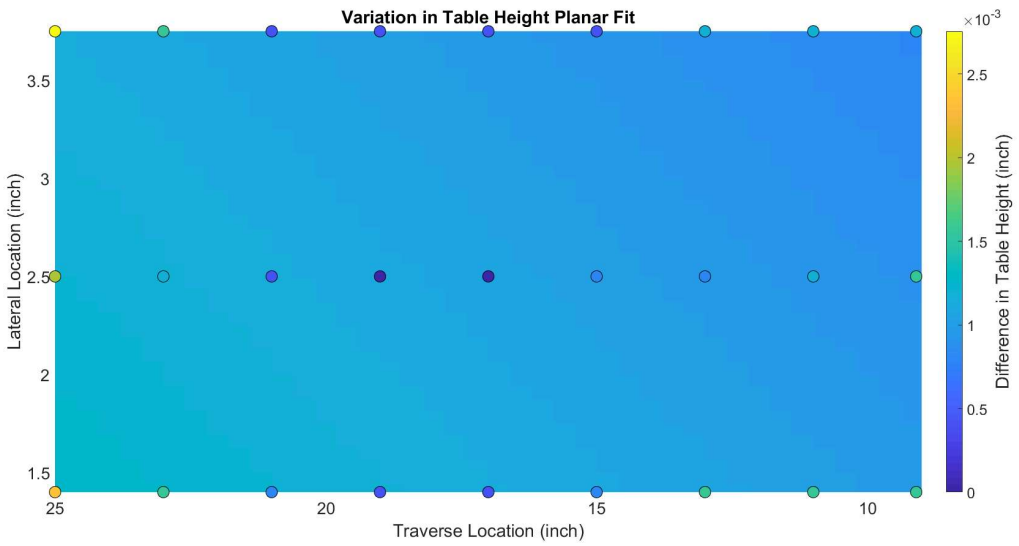
Appendix Figure 13: First test after adding shims



Appendix Figure 14: Second test after adding more shims



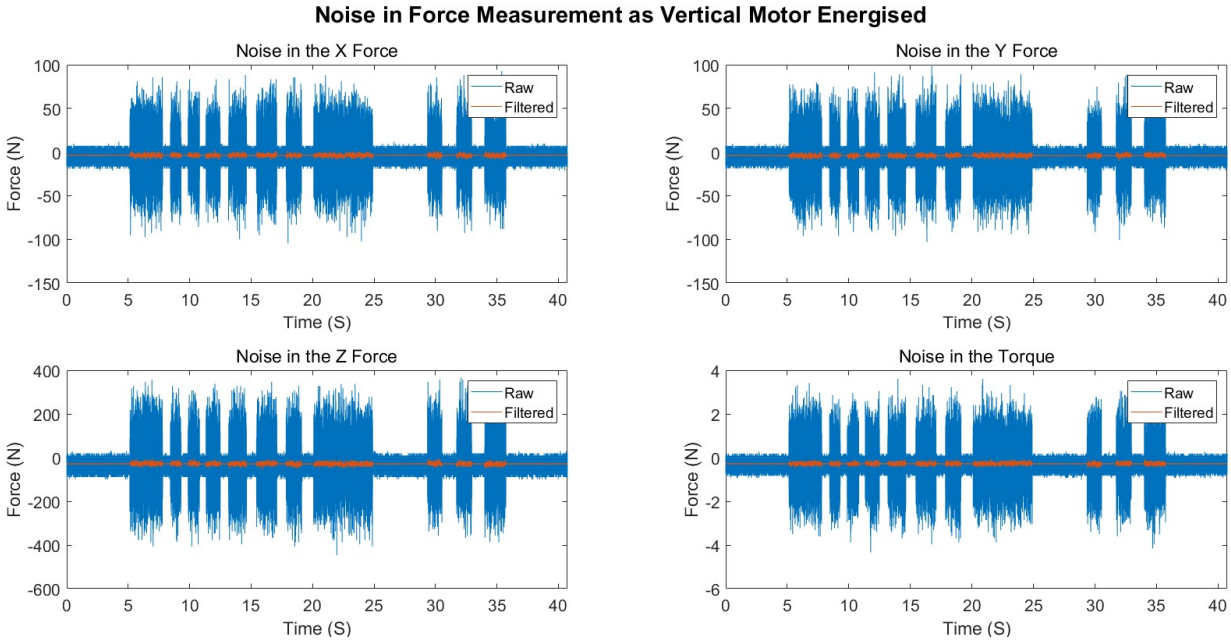
Appendix Figure 15: Third test after adding more shims

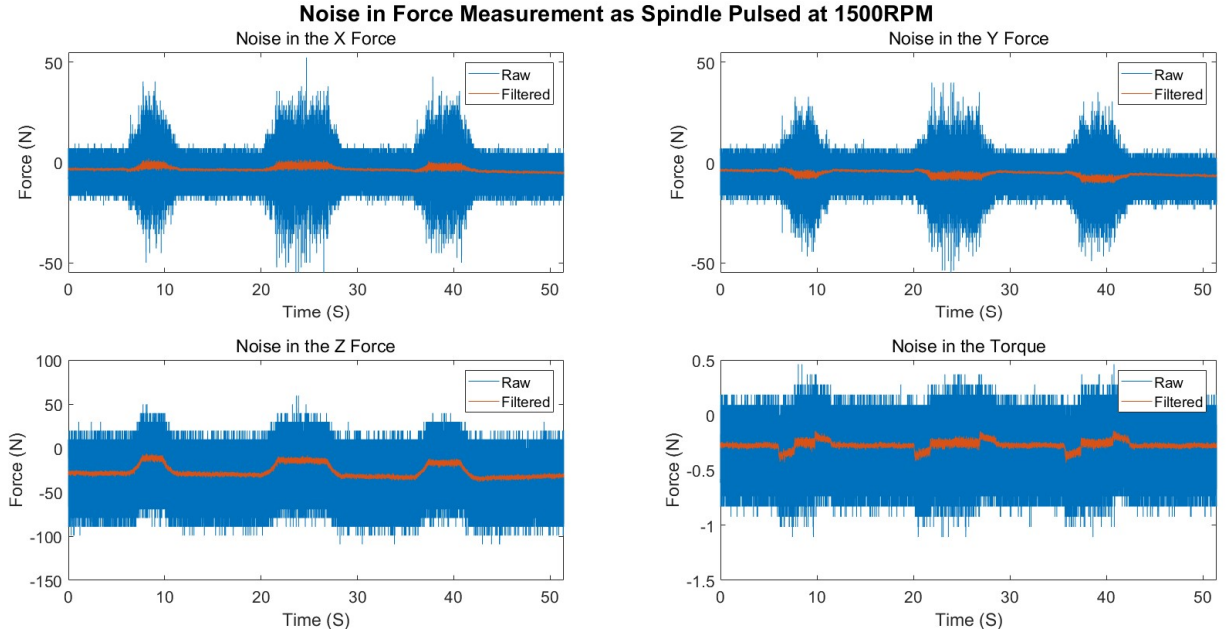


Appendix Figure 16:Fourth and final test of table height

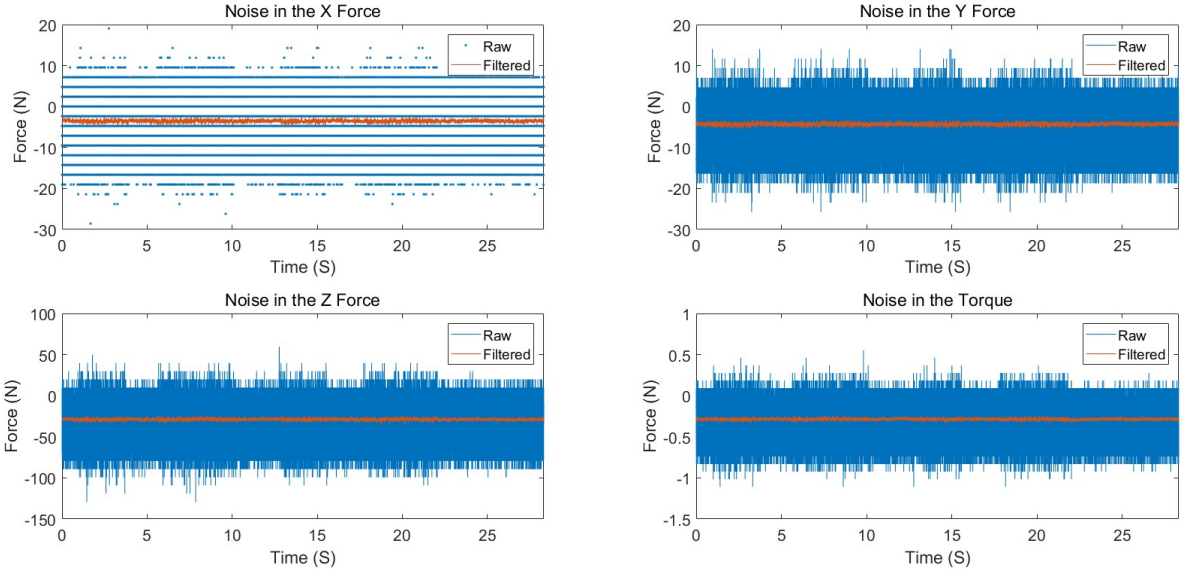
APPENDIX H: Results of Motors Effect on Dynamometer's Measurement

These figures show the result of the tests to get an idea of how much electromagnetic noise each motor causes on the measured forces from the dynamometer.





Noise in Force Measurement as Traverse Motor Moved 10 IPM



Noise in Force Measurement as Lateral Motor Moved 6 IPM