Portland State University

Capstone Project Report

Department of Electrical & Computer Engineering

3D Metal Printer

Members:
Cameron Tribe
Branden Driver
Brian Andrews
Ahmad Qazi

Faculty Supervisor:
Dr. Marek Perkowski
Industry Sponsor:
Aram Kasparov

June 5, 2015

Contents

1 Project Overview

The team interfaced a CNC machine with a MIG welder to create a 3D metal printer.

2 Project Proposal

2.1 Sponsor Proposal

The company is in the process of constructing an innovative 3D metal printer controlled by CNC (Computer Numerical Control). The project was a combination of two machines:

- CNC mill – (3, 4 or 5 axis CNC mill)
- MIG/TIG welding machine.

The purpose for the CNC motion control (CNC mill) is to program and control motion of the machine, and in this case, the metal deposition process. The purpose for the MIG welder is to deposit liquid metal. Many kinds of wire can be used by the welder to form the parts; carbon steel, titanium, stainless steel or aluminum. The idea for metal deposition and an example that uses a laser can be found at:
https://www.youtube.com/watch?feature=player_embedded&v=s9IdZ2pI5dA
A problem with laser use is its high cost. In this project, the welding machine used cost $400. Another example can be found here:
http://www.wired.co.uk/magazine/archive/2014/08/play/steel-sketch
AKTechnology's plan is to manufacture parts for pump and compressors, and research and develop parts for all sorts of use. The goal is to fabricate low cost and highly usable machines.

The company has CNC PC based CNC mill-motion controller.
https://www.youtube.com/watch?v=Plf3t7o951U&list=UUlGufPQeEKdN1-50F89Ejig
https://www.youtube.com/watch?v=G-jokU7v92E&list=UUlGufPQeEKdN1-50F89Ejig
https://www.youtube.com/watch?v=bPQ5UNiGA4c&list=UUlGufPQeEKdN1-50F89Ejig

The project was to upgrade this CNC motion controller – mill into 3D metal deposition printer by adding a MIG welder instead of a cutting tool spindle. The CNC motion control was reprogrammed. The MIG welder was operational.
The project will also build control to integrate CNC mill and MIG welding machine. The Welder has 2 adjustments - feed of wire and current. A stepper motor is planned to be used to control those analog data for wire feed and power current. The PLC, programmable logic controller, will join the CNC motion controller and the MIG welding machine.

2.2 The Goal

The end goal of this project is to fully integrate the MIG welder with the LinuxCNC system. Integration will include a way to control all of the functions of the welder, i.e. wire speed, maximum current output, engaging and disengaging the welder at appropriate times. In order for this to be done, electromechanical devices must be used to manipulate the knobs on the MIG welder. At the very least, the machine must be able to deposit material, reproducing a simple single object from a CAD drawing. Our aim is to produce a 1" cube. However, it is desired that the machine will be able to create complex structures on a single base. Precision of the deposition is not the primary concern, however it will be a requirement that the total amount of material deposited is more than the minimum tolerance of the part being created. This will allow for material to be machined away to a more precise tolerance.

2.3 Our Starting Point

The groundwork of this project has been completed by Aram Kasparov, the project sponsor. The project at its current state consists of a PC controlled CNC machine, a MIG welder, an infrared temperature sensor and a current measuring sensor. The PC controlling the CNC machine is running a Linux operating system. LinuxCNC an open-source software is used for programing and interfacing with the physical machine. Additional hardware is installed onto the PC, consisting of Mesa Electronics 5I20 FPGA based PCI Anything I/O card, 7i33 analog servo interface card and two 7i37-COM isolated I/O cards. The LinuxCNC software communicates the control signals and receives feedback through these cards. The CNC machine is a 3-axis machine-that is it can move in the X, Y and Z directions. Each axis is moved by a servo-motor and each servo motor is driven by a driver which receives its control commands from the PC. The machine is functional, though the motors will require some tuning and limit switches need to be programmed in (they are physically installed on the machine but not included in the program). The MIG/Flux cored welder is rated at 180 Amp-DC, 240 Volt with a duty cycle of 20% at 140 amps. The welder has current and wire feed adjustment capabilities for controlling the weld. These two knobs will be controlled by two stepper motors which have been installed onto the welder already. The current sensor has the ability to measure up to 225A. It has been demonstrated to be functional and will be used to monitor the current of the weld. The infrared non-contact temperature sensor is rated to measure temperatures up to 1800 degrees Celsius, though no tests have been performed yet.

2.4 Requirements

- Must use a wire feed welder
- Welder must have a Control System
- Must measure weld temperature
- Must measure weld current
- Must use both previous parameters to estimate current quality of weld
- Must use "G code" as inputs
- Must control when material is being deposited
- Must have user interface
- Should allow for welder thermal shutdown
- Should Measure Wire Speed from welder

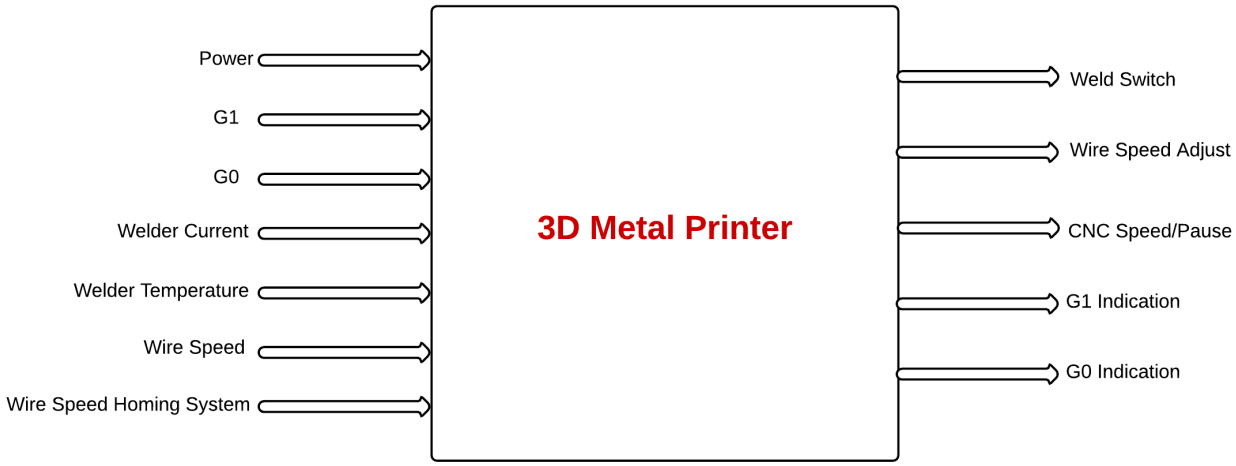


Figure 1: Level-0 Block Diagram of the 3D Metal Printer

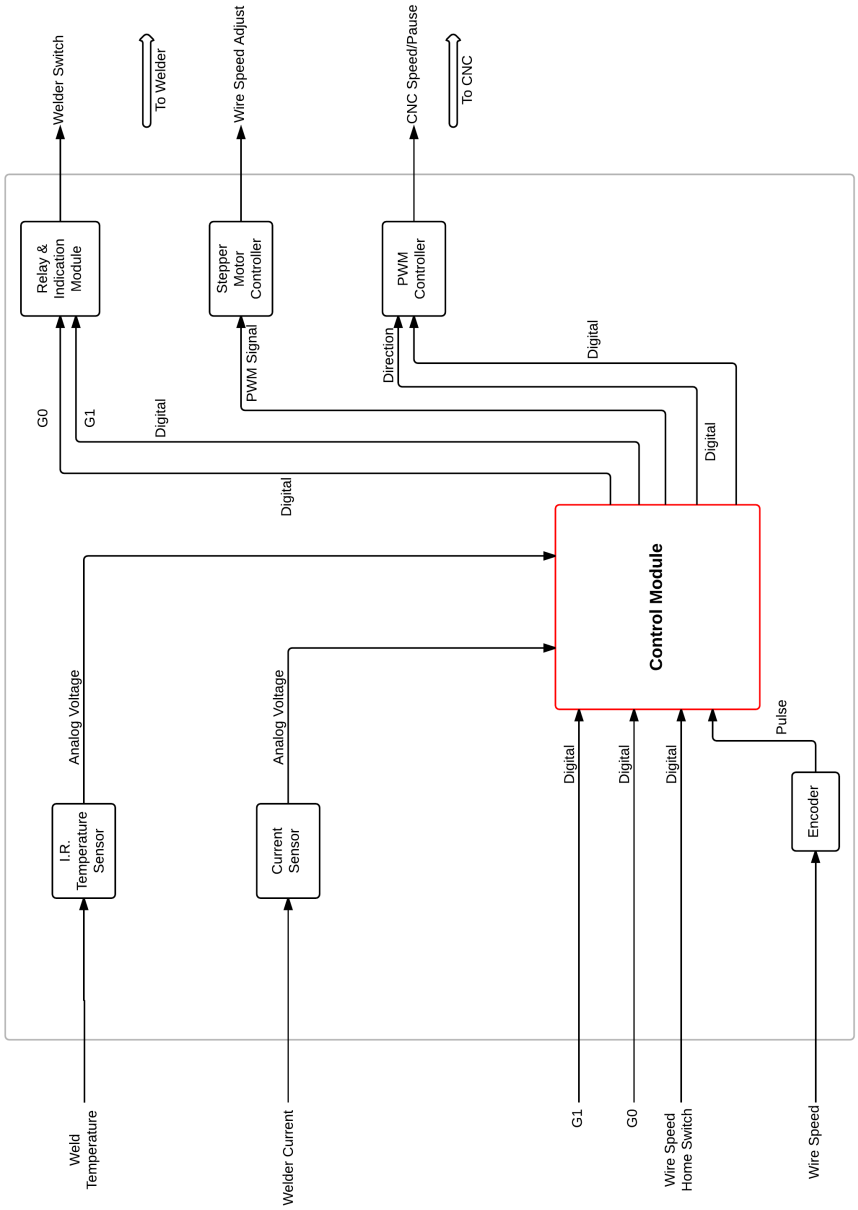
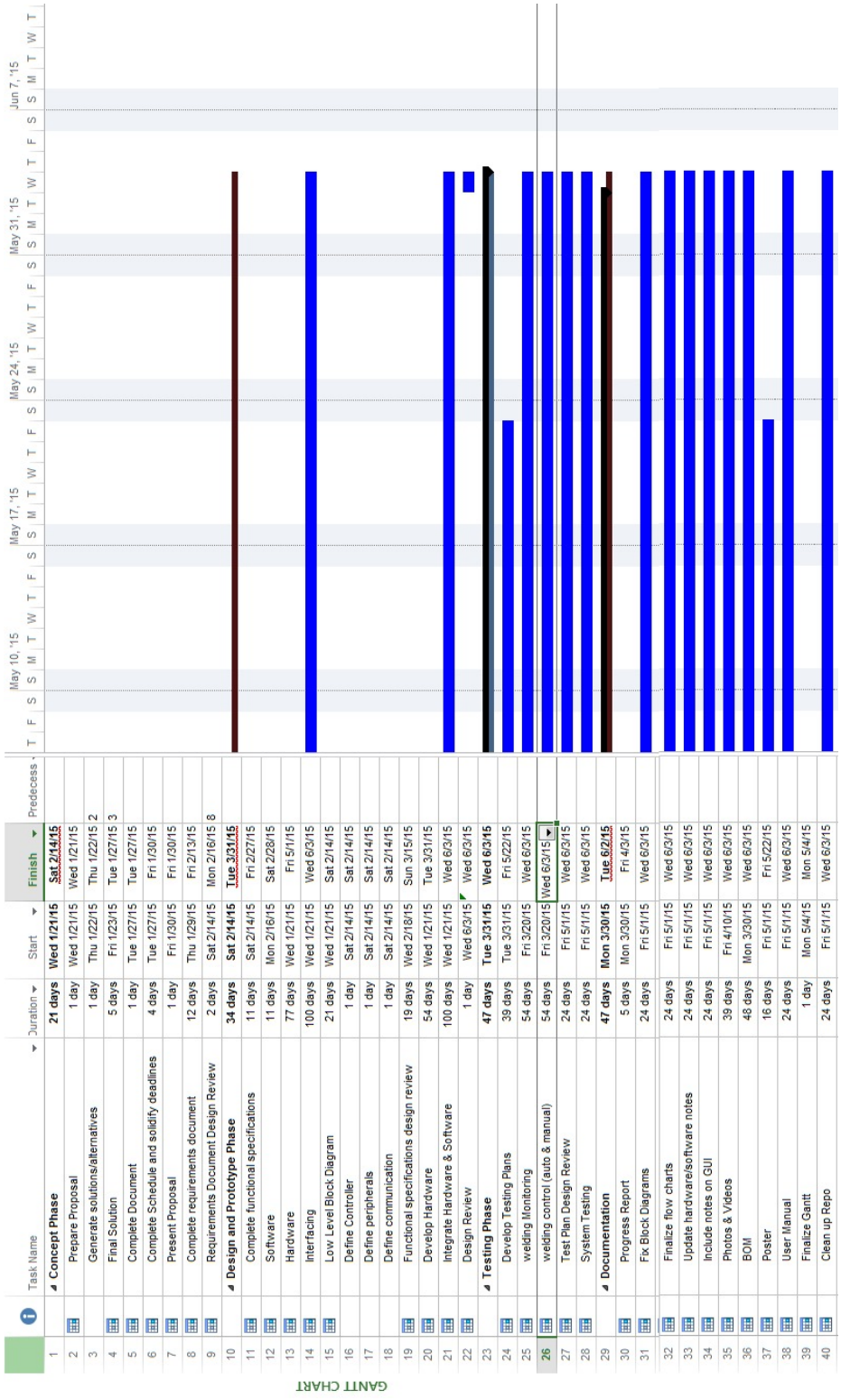


Figure 2: Level-1 Block Diagram of the 3D Metal Printer

3 Schedule



	Task Name	Duration	Start	Finish	Predecess
1	**Concept Phase**	**21 days**	**Wed 1/21/15**	**Sat 2/14/15**	
2	Prepare Proposal	1 day	Wed 1/21/15	Wed 1/21/15	
3	Generate solutions/alternatives	1 day	Thu 1/22/15	Thu 1/22/15	2
4	Final Solution	5 days	Fri 1/23/15	Tue 1/27/15	3
5	Complete Document	1 day	Tue 1/27/15	Tue 1/27/15	
6	Complete Schedule and solidify deadlines	4 days	Tue 1/27/15	Fri 1/30/15	
7	Present Proposal	1 day	Fri 1/30/15	Fri 1/30/15	
8	Complete requirements document	12 days	Thu 1/29/15	Fri 2/13/15	
9	Requirements Document Design Review	2 days	Sat 2/14/15	Mon 2/16/15	8
10	**Design and Prototype Phase**	**34 days**	**Sat 2/14/15**	**Tue 3/31/15**	
11	Complete functional specifications	11 days	Sat 2/14/15	Fri 2/27/15	
12	Software	11 days	Mon 2/16/15	Sat 2/28/15	
13	Hardware	77 days	Wed 1/21/15	Fri 5/1/15	
14	Interfacing	100 days	Wed 1/21/15	Wed 6/3/15	
15	Low Level Block Diagram	21 days	Wed 1/21/15	Sat 2/14/15	
16	Define Controller	1 day	Sat 2/14/15	Sat 2/14/15	
17	Define peripherals	1 day	Sat 2/14/15	Sat 2/14/15	
18	Define communication	1 day	Sat 2/14/15	Sat 2/14/15	
19	Functional specifications design review	19 days	Wed 2/18/15	Sun 3/15/15	
20	Develop Hardware	54 days	Wed 1/21/15	Tue 3/31/15	
21	Integrate Hardware & Software	100 days	Wed 1/21/15	Wed 6/3/15	
22	Design Review	1 day	Wed 6/3/15	Wed 6/3/15	
23	**Testing Phase**	**47 days**	**Tue 3/31/15**	**Wed 6/3/15**	
24	Develop Testing Plans	39 days	Tue 3/31/15	Fri 5/22/15	
25	welding Monitoring	54 days	Fri 3/20/15	Wed 6/3/15	
26	welding control (auto & manual)	54 days	Fri 3/20/15	Wed 6/3/15	
27	Test Plan Design Review	24 days	Fri 5/1/15	Wed 6/3/15	
28	System Testing	24 days	Fri 5/1/15	Wed 6/3/15	
29	**Documentation**	**47 days**	**Mon 3/30/15**	**Tue 6/2/15**	
30	Progress Report	5 days	Mon 3/30/15	Fri 4/3/15	
31	Fix Block Diagrams	24 days	Fri 5/1/15	Wed 6/3/15	
32	Finalize flow charts	24 days	Fri 5/1/15	Wed 6/3/15	
33	Update hardware/software notes	24 days	Fri 5/1/15	Wed 6/3/15	
34	Include notes on GUI	24 days	Fri 5/1/15	Wed 6/3/15	
35	Photos & Videos	39 days	Fri 4/10/15	Wed 6/3/15	
36	BOM	48 days	Mon 3/30/15	Wed 6/3/15	
37	Poster	16 days	Fri 5/1/15	Fri 5/22/15	
38	User Manual	24 days	Fri 5/1/15	Wed 6/3/15	
39	Finalize Gantt	1 day	Mon 5/4/15	Mon 5/4/15	
40	Clean up Repo	24 days	Fri 5/1/15	Wed 6/3/15	

Figure 3: Gantt Chart showing the schedule

4 Hardware

4.1 Welder Control

To control the welder, a central control module will be used. This was a hot topic of debate for several weeks, as the number of choices available for this project are very high. The sponsor's requirements for the project was that all of the control work was done by a separate computer from the one used by Linux CNC, this only narrowed it down to a choice between a PCIe DAC board and a single board computer. Based on the need for both analog and digital control pins and the need for future expansion, we researched and came up with several options.

Single Board Computers	**DAC**
Raspberry-Pi	Sensoray 826
Intel Galileo	MCC DAS1602/16
SBC 8600B	
Wander Board Solo	
Beagle Bone Black	

In the end we chose the Sensoray 826 board because for the price it outperforms all other boards on the market by having 16 analog inputs, 8 analog outputs, and 48 digital I/O pins. This board was chosen for the high level of future expandability that it has, and because it is a PCIe card which can be packaged into its own desktop as per request from the sponsor. To control the current to the weld and the wire speed of the welder, two stepper motors have been fitted to the manual control knobs, and are connected to a motor driver module. The Sensoray board will be controlling the motor drivers using a sequence of rising and falling edges. To allow the controller board to control at what time the welder is depositing and when it is not depositing, a relay with a transistor driver will be used.

The signal that tells the controller will be coming from the CNC machine's I/O card. It is a switch type signal which means that when the signal is sent, an internal switch will be closed, causing what ever is on the input to be shown on the output. The CNC machine uses G-Code (described below), and Linux CNC allows outputs to be asserted when a particular G-Code instruction is executed. G1 and G0 are going to be used to tell the I/O card to close and open the switch, which will assert 5V DC to the input to the control module. The control module will assert an output high or low which will open or close the welder switch, turning the welder on and off.

The Sensoray 826 I/O card has three 50 pin connectors and two 26 pin connectors. To allow easy access to these pins, a breakout board with screw terminals has been made so that wires can easily be disconnected and switched.

4.2 Breakout Boards

To easily connect to the Sensoray 826 board, several breakout boards with screw terminals were made. There are two types of break out boards, a 50-pin board and a 26-pin board. 50 pin boards are used to connect to the digital and analog pins of the control board, while 26 pin boards are used to connect to the various counter channels. Shown below are mages of the boards them selves. With the empty side of the board facing you, the screw terminals are in order from 1 to 50/26 starting on the right hand side.

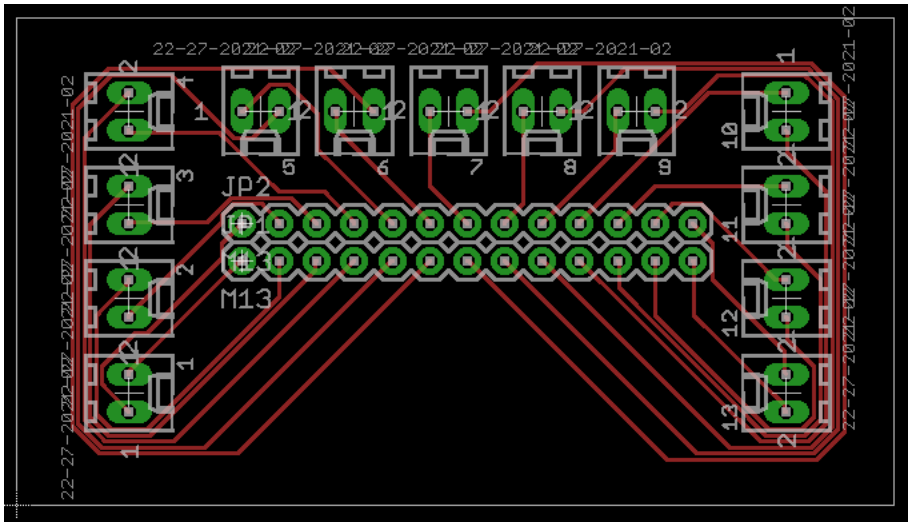


Figure 4: Board Layout of the 26 pin Breakout Board for the Sensoray 826 I/O card

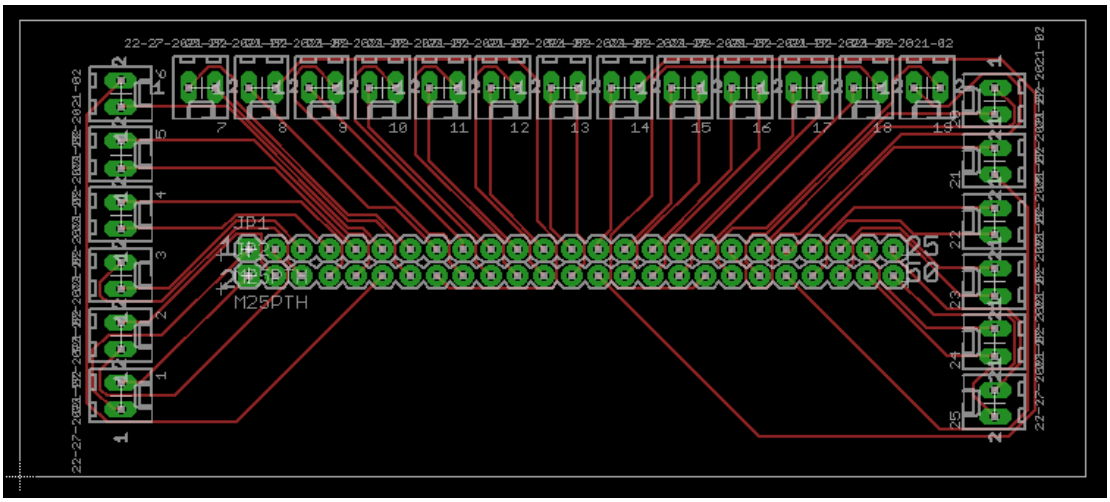


Figure 5: Board Layout of the 50 pin Breakout Board for the Sensoray 826 I/O card

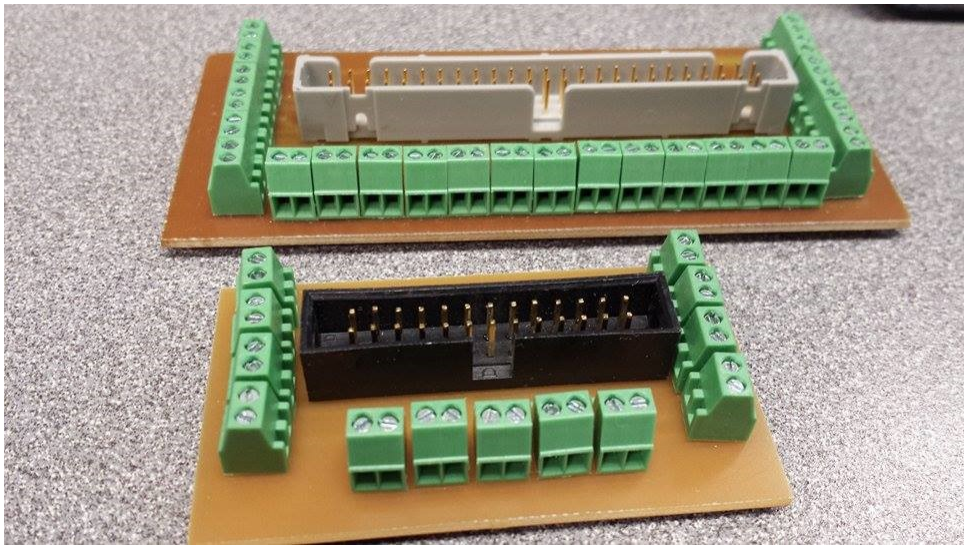


Figure 6: The 26 and 50 pin Breakout Boards with connectors for The Sensoray 826 I/O card

4.3 Temperature Sensor

An infrared temperature sensor will be used to measure the temperature surrounding the weld area. These temperature sensors typically have a higher operating range than other types of temperature sensors. The chosen sensor is the CTLM-1M-1H1-CTL-CF4 from Micro-Epsilon. This sensor was chosen of it's operating range was 800C to 2200C. It has multiple configurable output types, including current output, voltage output, and alarm outputs. This sensor also has a focus point at 450mm ($\tilde{1}8$ in) which gives considerable distance from the weld, and it is not impractically far away. Further documentation can be found (Appendix temp sens). The Sponsor has agreed to take care of mounting the sensor on the machine.

The chosen output type of the temperature sensor is chosen to be a voltage output with a full-scale range of 0V to 10V. This range was chosen because the ADC on the Sensoray 826 are configurable to accept up to +10V and another Analog input needed the input configured to +10V. Setting the output range to 0V-10V on the current sensor removed the need to write additional software to solve an issue that was solved in other means.

Figure 7: The Micro-Epsilon 1MH1-CF4 Temperature Sensor

4.4 Incremental Encoder

The Incremental Encoder is used to measure the actual wire speed of the welder wire. The encoder needed to be incremental because of the need to know speed, and that current position did not matter. The chosen encoder is the U.S. Digital S5-5000-250-IE-D-B. Initially a pulley on a shaft type encoder was going to be used. The pulley be mounted to the frame of the welder and the encoder would be placed under tension underneath the wire that is being fed to the weld. However this was not successful as there could not be any external tension placed on the wire inside the welder.

Fortunately the main drive pulley has a square drive shaft and stuck out past the edge of the pulley. A small plastic "coupler" piece was 3D printed that adapted the round end of the encoder, to the square drive shaft of the drive pulley. The CAD drawing of this part can be seen to the left. This piece allowed an accurate measurement of the actual wire speed of the welder to be interpreted by the Sensoray 826. Here, the calculations for the wire speed can be found.

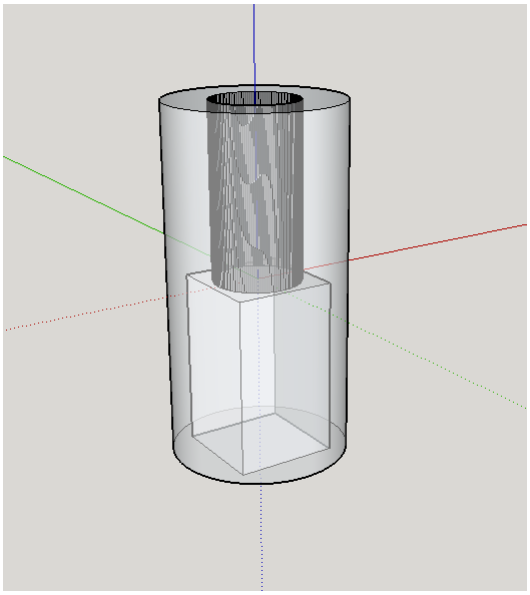


Figure 8: Schematic of the Incremental Encoder

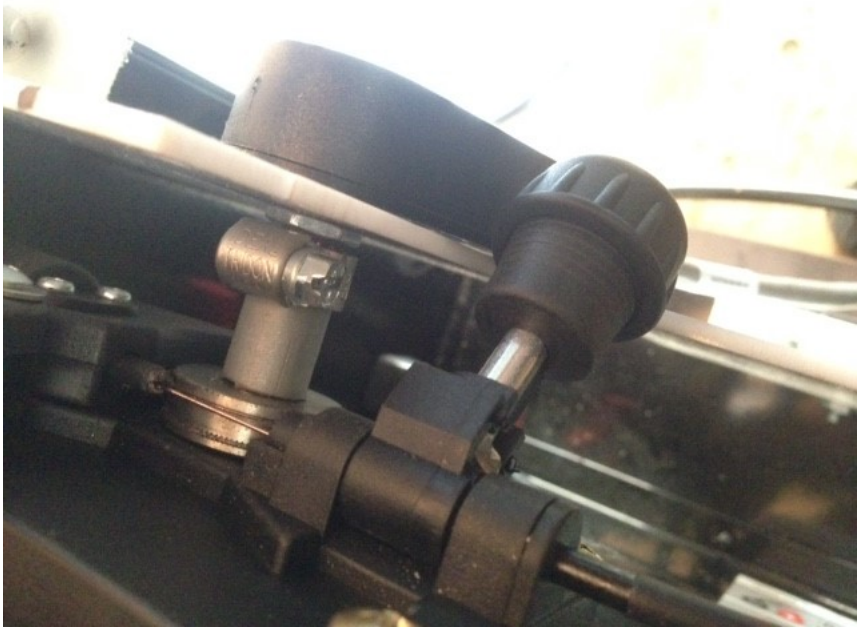


Figure 9: The US Digital S5-5000-250-IE-D-B Incremental Encoder

4.5 Relay and Indication Module

A relay was used to interface the Sensoray 826 with the welder. A relay provides isolation from any harmful voltage spike that occurs on the welder's circuit. It also acts as a mechanical switch, which is the same as the trigger on the welder gun itself. It is unknown what type of signal is passed through the welder switch weather its DC, or AC the relay will on act differently, as a transistor might. The interface circuit includes a transistor drive circuit that switches an 8V supply (which comes from a wall wart power supply) across the coil of the relay on and off. This module also includes LEDs to indicate what state the CNC machine is currently in. A schematic and an image of the final module can be seen below.

There were some issues with the relay module, where were that the Sensoray 826 has internal 10k pull up resistors, and when the machine powers up, the active low DIO pins are initialized to a 0-state, which means that the voltage is high on the pin. When connected to the welder, this meant that when there was not any code being executed the welder would be activated. To remedy this issue, an inverter was placed on the input.

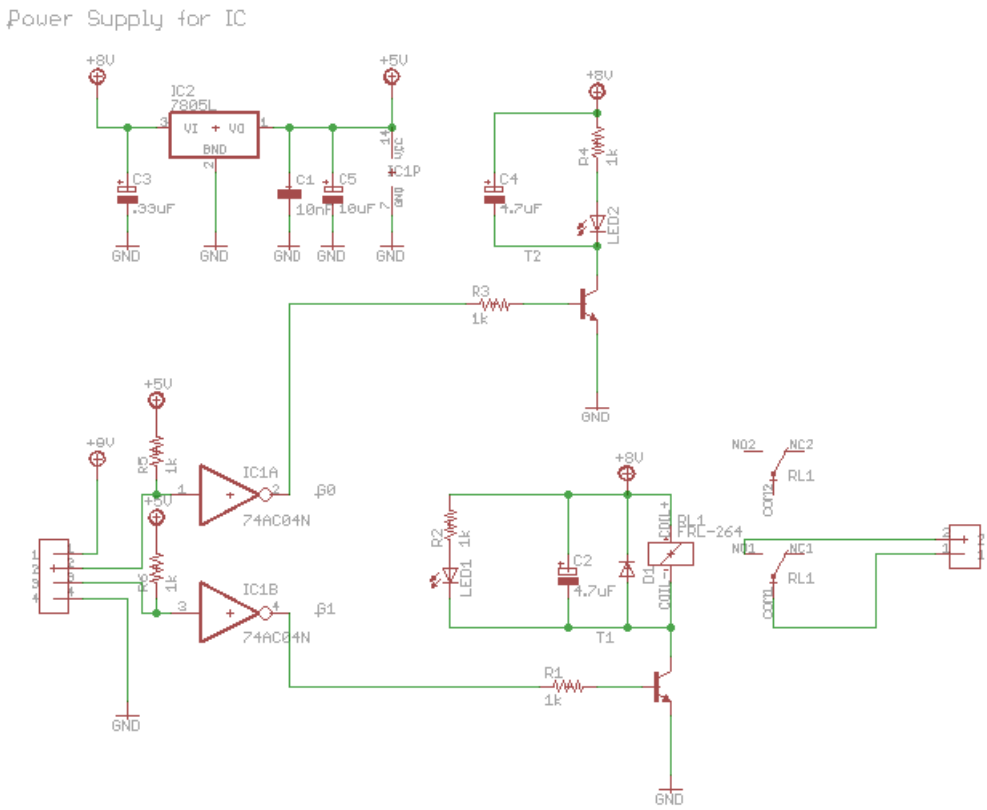


Figure 10: Schematic of the Relay & Indication Module

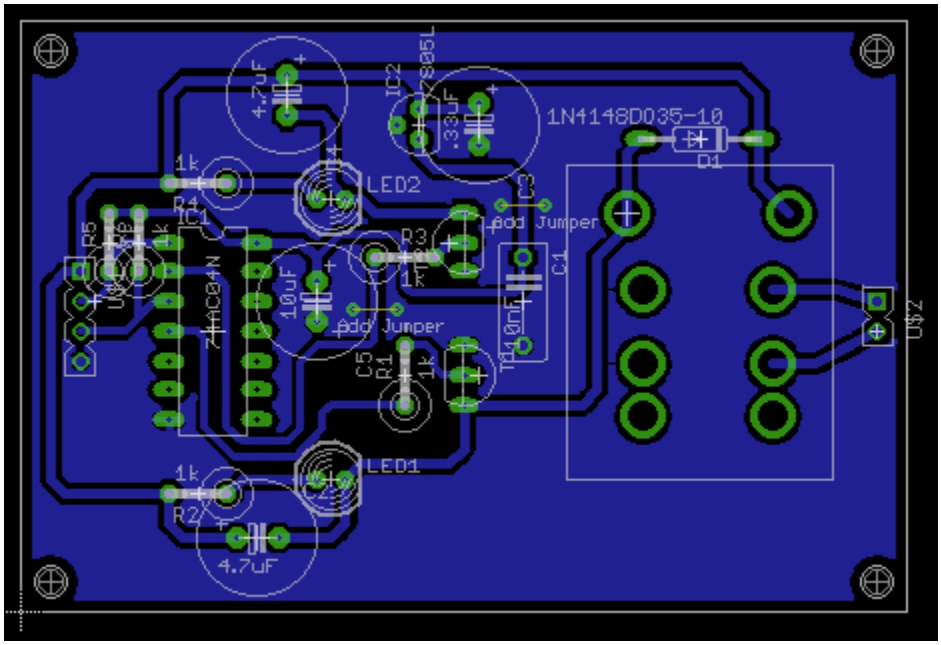


Figure 11: Board Layout of the Relay & Indication Module

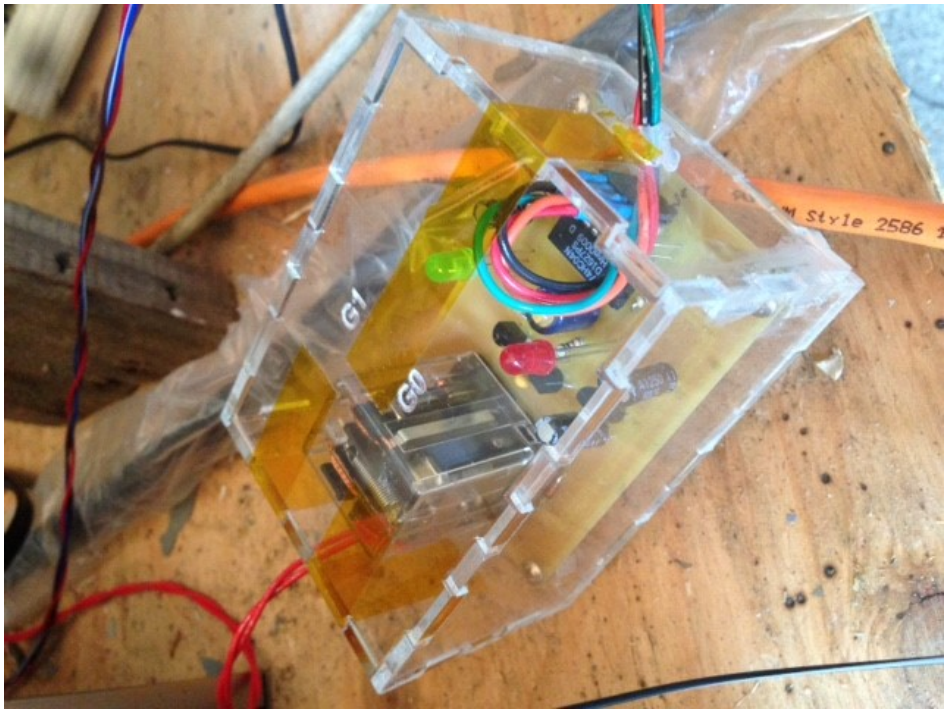


Figure 12: The Relay & Indication Module

4.6 Stepper Motor Controller

P/N:
Controller: KL-5056
Motor: KL23H2100-35-4B

The stepper motor controller is used to control a motor that adjusts the wire speed knob on the welder. From the Sensoray 826, there are two digital signals that control how much and in which direction the motor turns. To tell the motor two turn, a PWM signal with a 50% duty cycle is used. The frequency determines the speed of rotation. The direction signal is either a high or low 5V signal that will turn the motor clockwise or counterclockwise. The controller also has a programmable step resolution, which is either defined in software, or by using the DIP switched located on the side of the controller. Further documentation on the controller can be found here.

The motor is coupled to the wire speed adjustment knob via a PVC cylinder, which have elevated surfaces where the limits of the knob are. Shown below, are switches that get triggered if the motor turns too far. The limit switches are connected to the enable pin of the motor controller, so that if the motor accidentally turns too far, it will disable its self and it will not damage the knob on the welder. An additional switch is added to the PVC cylinder, which is used as a reference position of the knob. Shown below is the wiring diagram of the motor controller.

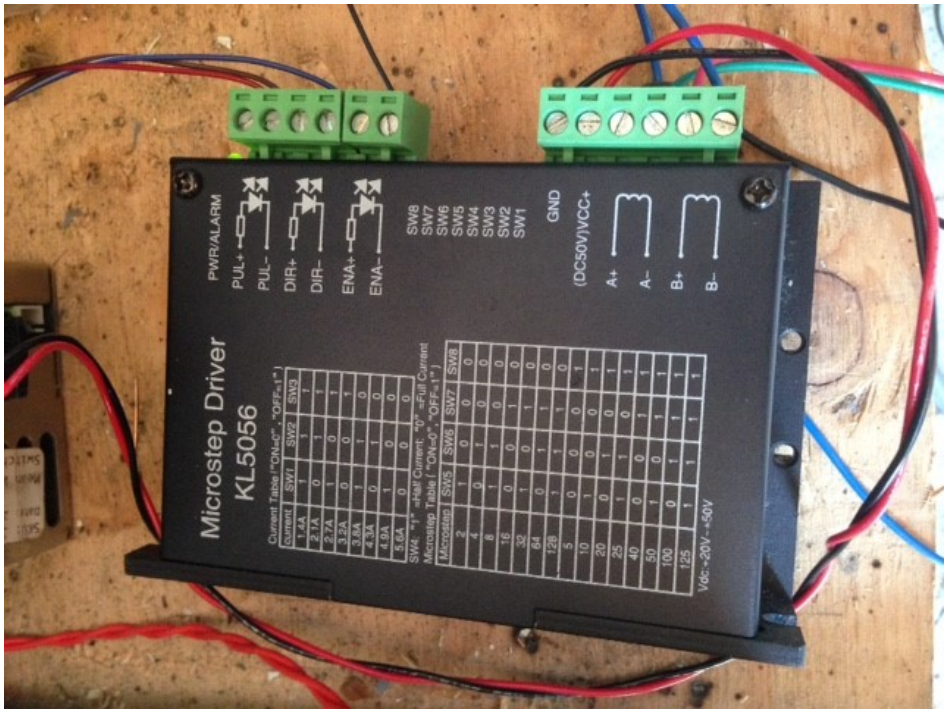


Figure 13: The Stepper Motor Controller

4.7 PWM Controller

P/N: Mesa THC A-D

The PWM controller is a voltage to frequency converter. It is used to externally start and stop the CNC machine. The voltage input of the PWM controller is connected to a digital output of the Sensoray 826. To stop the CNC Machine, a 5V signal is send and to stop it is 0V. The CNC machine is initialized to use this 5V signal in the .HAL file associated with that machine. In fact the voltage on that pin can be anywhere between 0V and 5V for further external control of the welder, however in the scope of this project, only a high or low signal needs to be sent.

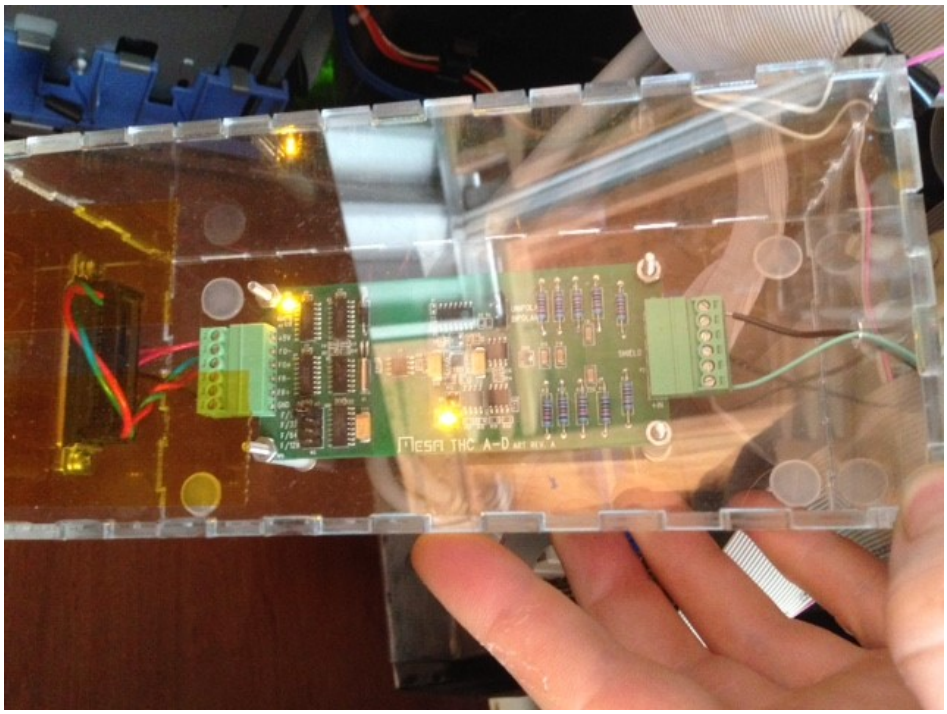


Figure 14: The Mesa THC A-D PWM controller

4.8 Current Sensor

P/N: CSLA1DJ

This current sensor has an operational range of 0 to 225A, which is well over the maximum current of the welder. It is placed in a small plastic housing, which a jumper cable is passed though. The sensor is a Hall effect sensor that is placed inside a ferrite toroid. The output of the sensor is a voltage that sits at half of the supply voltage, and will deviate above or below that level based on he magnitude and direction of the current. There is a metal lug, which the ground connection of the welder connects to. The other end of the module is a large alligator clamp that connects to the plate being welded to.

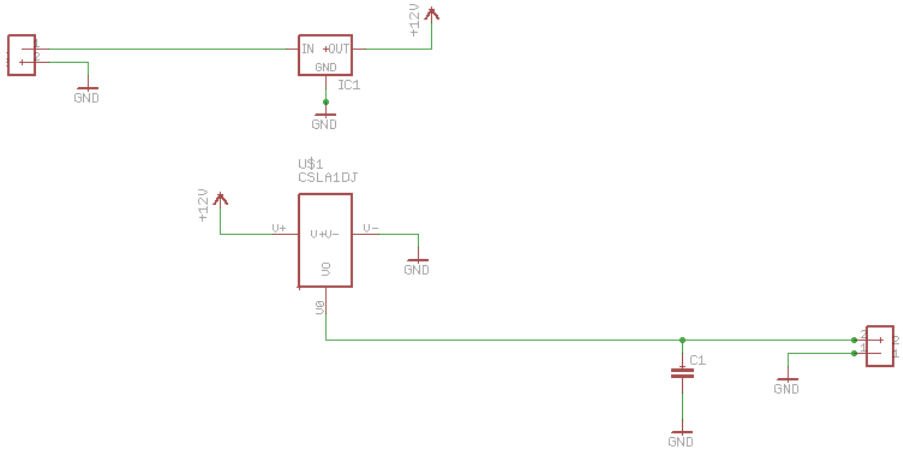


Figure 15: Schematic of the Current Sensor

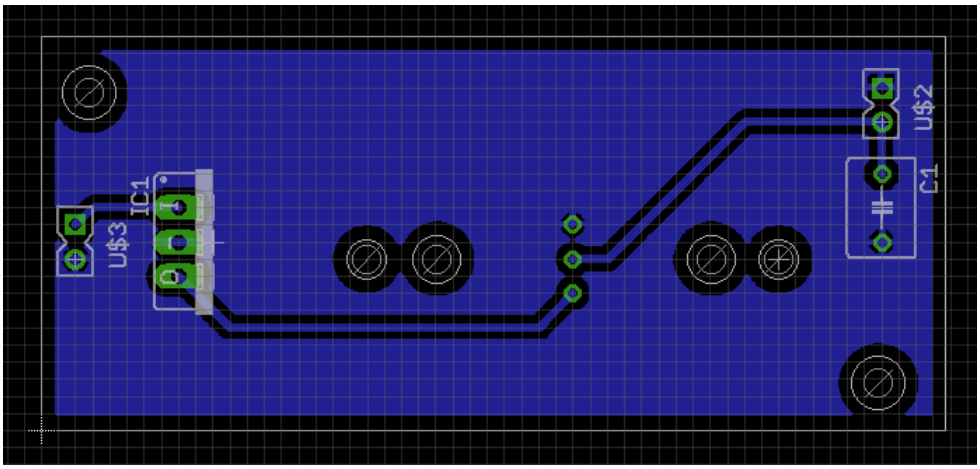
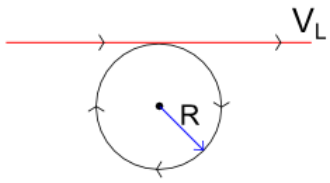


Figure 16: Board Layout of the Current Sensor

Figure 17: The CSLA1DJ Current Sensor

4.9 Wire Speed

For calculating Wire Speed with Rotary Encoder



So Angular Velocity is,

$$\boldsymbol{f = \frac{n}{Nt}}$$

n = number of up/down counts, N = number of up/down counts/rev, T = sampling time.

And Linear Velocity is,

$$v_L = \omega r$$
$$\omega = 2\pi f$$

So,

$$\boldsymbol{V = \frac{\pi nd}{NT}}$$

Connections to 826

J_4	Pin 1	+A0
	Pin 2	-A0
	Pin 3	GND
	Pin 4	+B0
	Pin 5	-B0

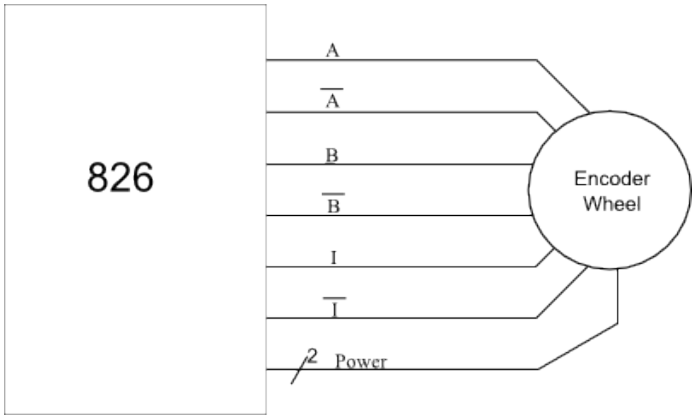


Figure 18: Encoder Connection

4.10 Pin Assignments

DIO PINS	Wire Color	Channel Number	Function
1	Teal	23	CNC Start/Stop
2	Black		GND for CNC Start/Stop
3	Blue with White Stripe	22	Direction of Stepper Motor
4	Black with White Stripe		Return path for Limit Switch
5	Brown	21	Pulse on Motor Controller
6	N/C		GND
7	Purple	20	Wire Speed Home Switch
8	Black		GND for Wire Speed Home Switch
9	Green with White Stripe	19	G1 Input (From CNC Machine)
10	N/C		GND
11	Orange with White Stripe	18	G0 Input (From CNC Machine)
12	N/C		GND
13	Green	17	G1 Output (To Relay and Ind. Module)
14	Black		GND for Relay Module
15	Orange	16	G0 Output (To Relay and Ind. Module)
16	Black		Power Supply GND for Relay Module
17		15	
18			GND
19		14	
20			GND
21		13	
22			GND
23		12	
24			GND
25		11	

DIO PINS	Wire Color	Channel Number	Function
26			GND
27		10	
28			GND
29		9	
30			GND
31		8	
32			GND
33		7	
34			GND
35		6	
36			GND
37		5	
38			GND
39		4	
40			GND
41		3	
42			GND
43		2	
44			GND
45		1	
46			GND
47		0	
48			GND
49			5V to the CNC Switching system and Motor Controller
50			GND

ANALOG PINS	**Wire Color**	**Channel Number**	**Function**
1			
2			
3	Black	0	GND for Current Sensor
4	Yellow	0	Current Sensor Signal
5	Black	1	GND for Temperature Sensor
6	Grey	1	Temperature Sensor Signal
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			

ANALOG PINS	Wire Color	Channel Number	Function
41			
42			
43			
44			
45			
46			
47			
48			
49			
50			

COUNTER PINS	Wire Color	Counter Channel Number	Function
1	Blue with White dots	0	A+ on Encoder
2	White with Blue dots	0	A- on Encoder
3	Green with White dots (note cable shield also connected)	0	GND on Encoder
4	Brown with White dots	0	B+ on Encoder
5	White with Brown dots	0	B- on Encoder
6	White with Green dots	0	POWER on Encoder
7	Orange with White dots	0	I+ on Encoder
8	White with Orange dots	0	I- on Encoder
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			

5 Software

5.1 Software Description

The main program for our control system was written entirely in the programming language "C". This was chosen so as to ensure compatibility with the Sensoray 826 DAQ board used for control. The program begins by asking the user to disconnect the grounding cable of the welder and waiting for an input from the user confirming this task is completed. Upon receiving that input the machine then runs a calibration procedure by running a PWM procedure to turn the stepper motor until the system sees the homing switch is triggered. The program then turns on the welder's gun to feed wire while measuring the wire speed and setting the average of those wire speeds as the new homing offset.

The next step of the program asks the user to input the speed the CNC machine will be running at for the duration of the print in in/sec. It uses this value to find a corresponding nominal wire speed in an array of nominal wire speeds found via testing, and sets the welding machine to that wire speed. The program then waits for a signal from the CNC machine that it has switched from Relocation mode to Deposition mode. Once the system sees that we have reached Deposition mode, it stops the CNC movement and makes sure the welder is not triggered, in order to check the temperature of the base plate being welded to. If the base plate's temperature is too low the system will pause and wait for a torch to heat the base plate up to above a given threshold value.

On completion of the torch routine the system enters the main loop, this begins by reading a timestamp from the 826's onboard timestamp generator, this value in microseconds will be used to keep track of how long the machine has been in deposition mode later. The system then starts the welder and runs a check for spikes in the current seen by the Current Sensor, if it sees none the system will end the program and return an error report, otherwise the program moves on and starts the CNC's movement back up so that we can check if the machine is still in Deposition mode. If the CNC machine is still in Deposition mode at that time it moves on to take measurements of the number of "peaks" seen by the current sensor, the temperature of the weld's base plate, and the incremental encoder used to measure wire speed. Amongst the measurements it checks and updates the mode of movement seen by the CNC in order to avoid issues with our system trying to deposit while in Relocation mode.

At the end of checking all sensor measurements, the system begins comparing the observed value with the pre-programmed threshold values. The first check is to make sure that the current plate temperature is at an acceptable value. If the temperature has fallen or risen too far, the whole system stops, and runs the torch routine before starting the system back at the initial timestamp read. If the temperature is at an acceptable value, the average droplet spacing is checked against a nominal value found through testing. If the error between the two values is greater or less than 20%, the system terminates with an error, asking the user to double check that the entire system is working. The last check is to see if the droplet spacing is greater or less than a 5% tolerance, and if so the system makes an appropriate proportional adjustment to the wire speed before continuing on.

5.2 G-Code

G-code is the commonly used name to refer to a numerical programming language. As is the case with all languages, G-code has its own syntax and semantics. A line of code is referred to as a block, and a program is defined as multiple blocks. All programs start and end with the percent symbol (%). While writing G-code, the backslash (/) can be used to comment out an entire line, whereas if you want to make comments about a block, parentheses are to be used. Anything inside of a set of parentheses will be ignored by the compiler. As is the case with most other languages, G-code ignores white space, so spacing is used to clarify the code for the writer as well as future users.

All points of G code are comprised of words and numbers. A word is simply a letter. For example, the block "X0" is simply the word X and the value 0. The words X, Y, and Z refer to the three axes, while the G words refer to the movement, motion, and location. When first started up, any blocks of code will use the point (0,0,0) as home, however G54 establishes a new temporary "home" point and G52 establishes the point where the temporary reference point is to be set. In other words, the block of code "G54 G52 X100 Y100 Z0" changes the reference point from (0,0,0) to (100,100,0) and the remainder of the code will run from this point, unless a new reference point is defined.

Similarly, other G-code words can be used to define how the space between two points should be interpolated. In some instances you may want two points to be connected via a straight line, while at other times it would be better to have them connected via a circular pattern. When dealing with circular interpolation, you can set it to be done in either a clockwise or counterclockwise manner. Beyond just linear and circular interpolation, there are dozens of G-code words that determine how the code is to be run (Appendix C).

5.3 Control Program

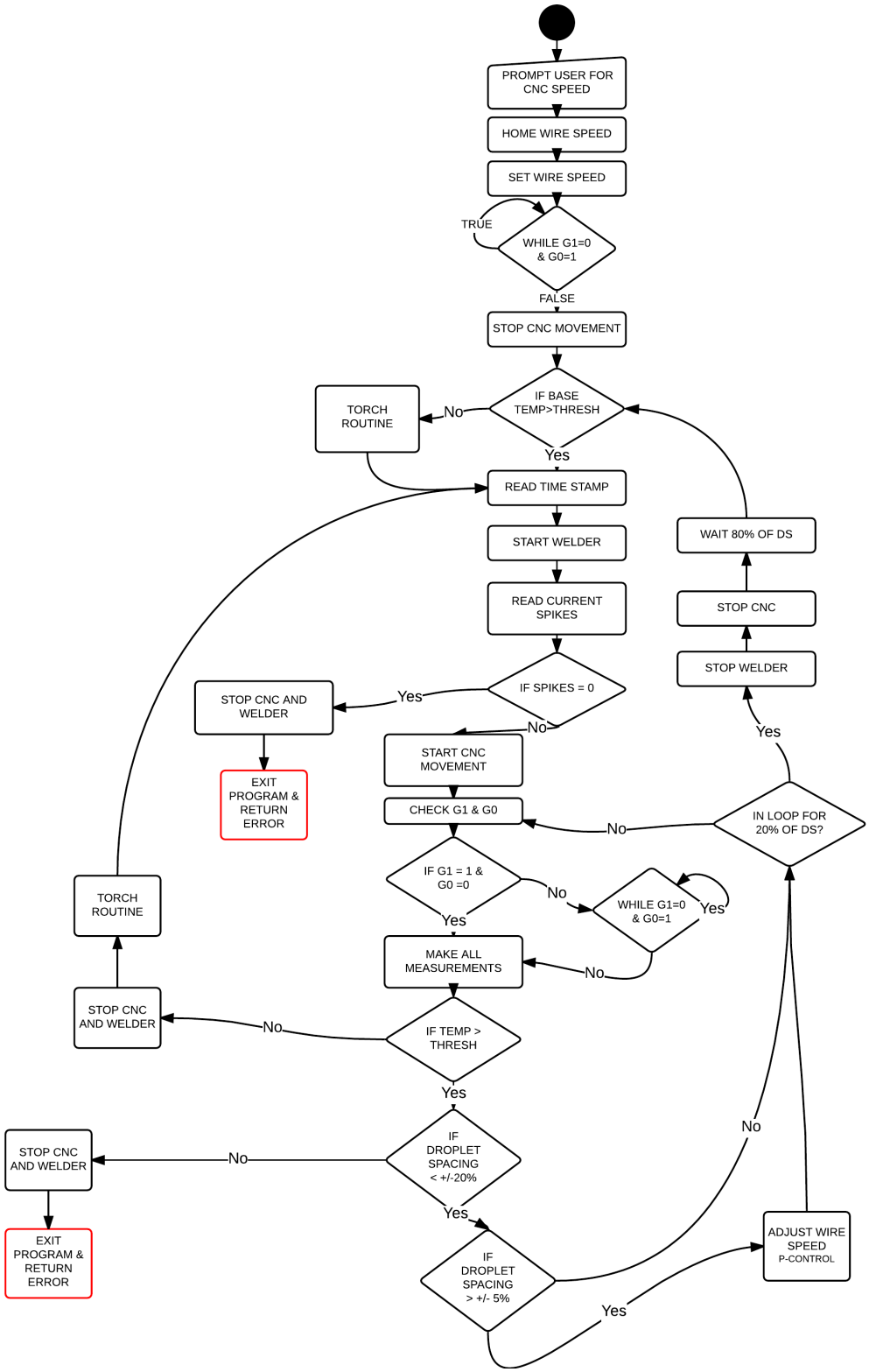


Figure 19: Control Program Flowchart

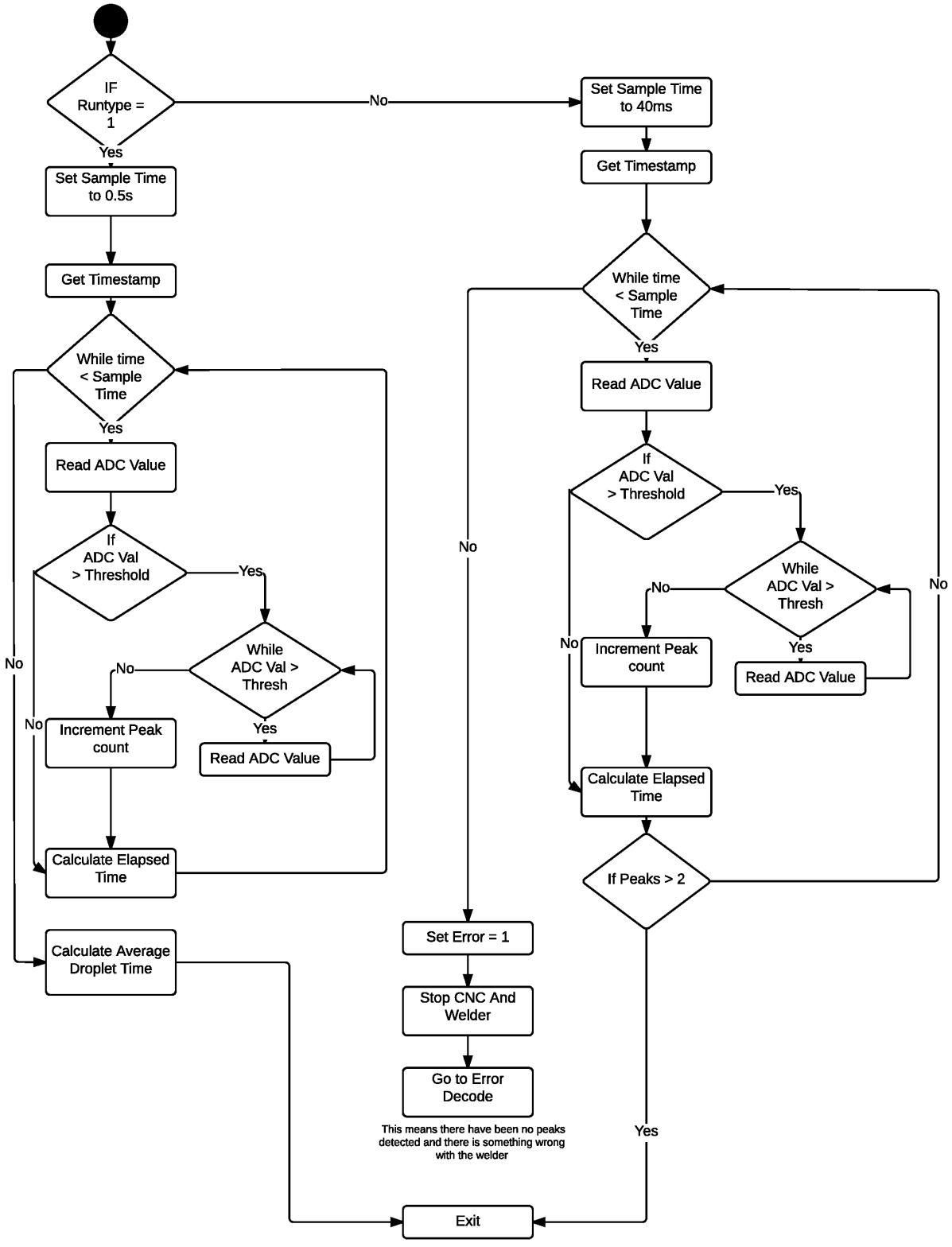


Figure 20: Droplet Spacing Flowchart

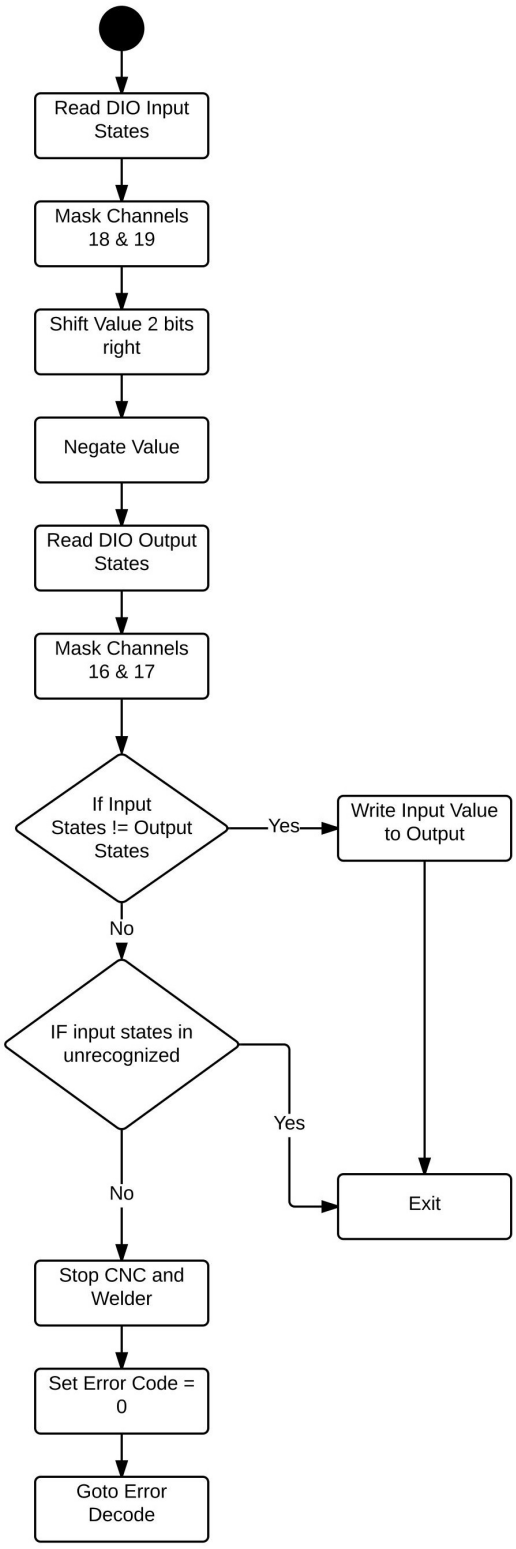


Figure 21: Getting G0 & G1 Flowchart

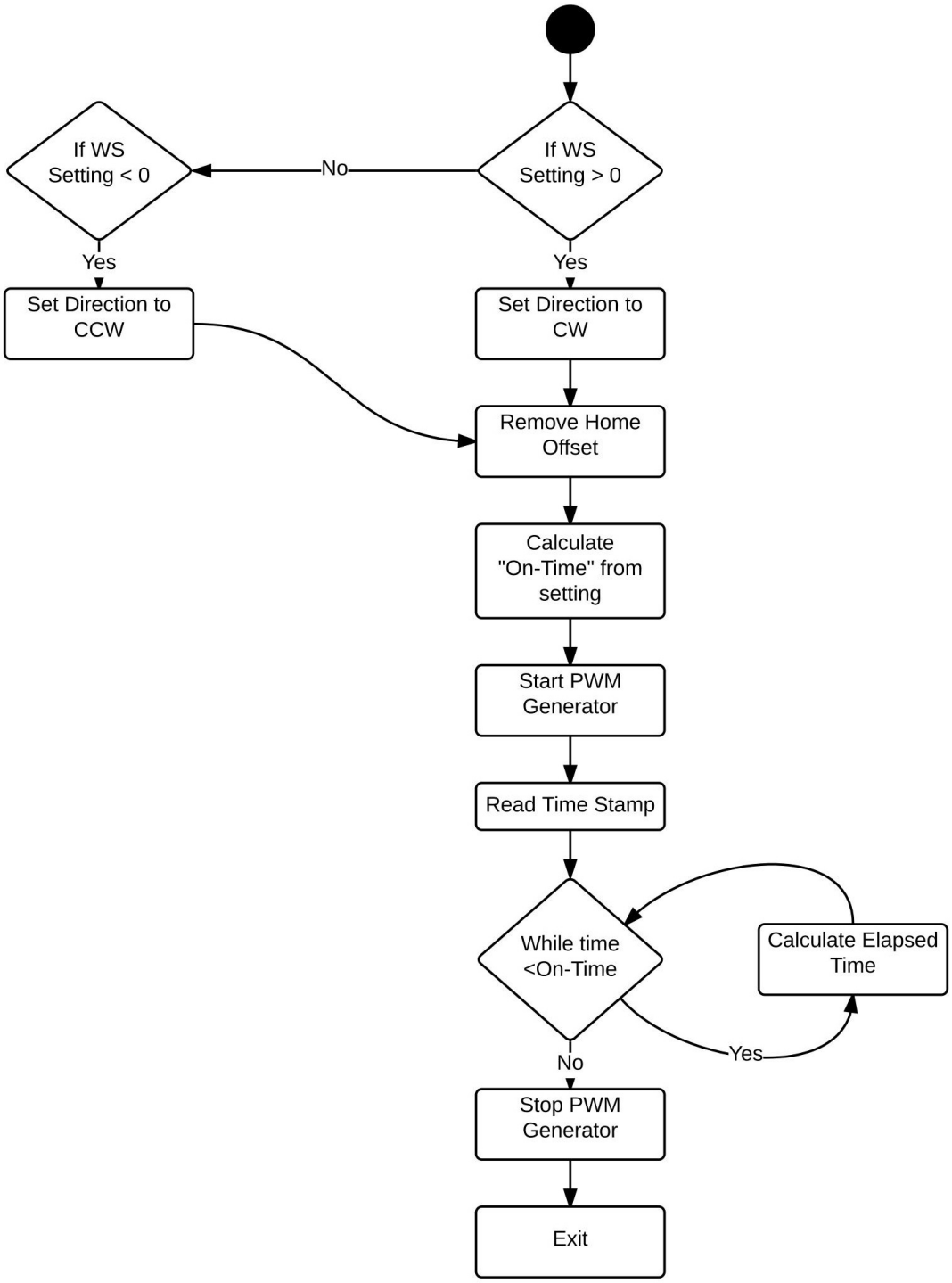


Figure 22: Set Wire Speed Flowchart

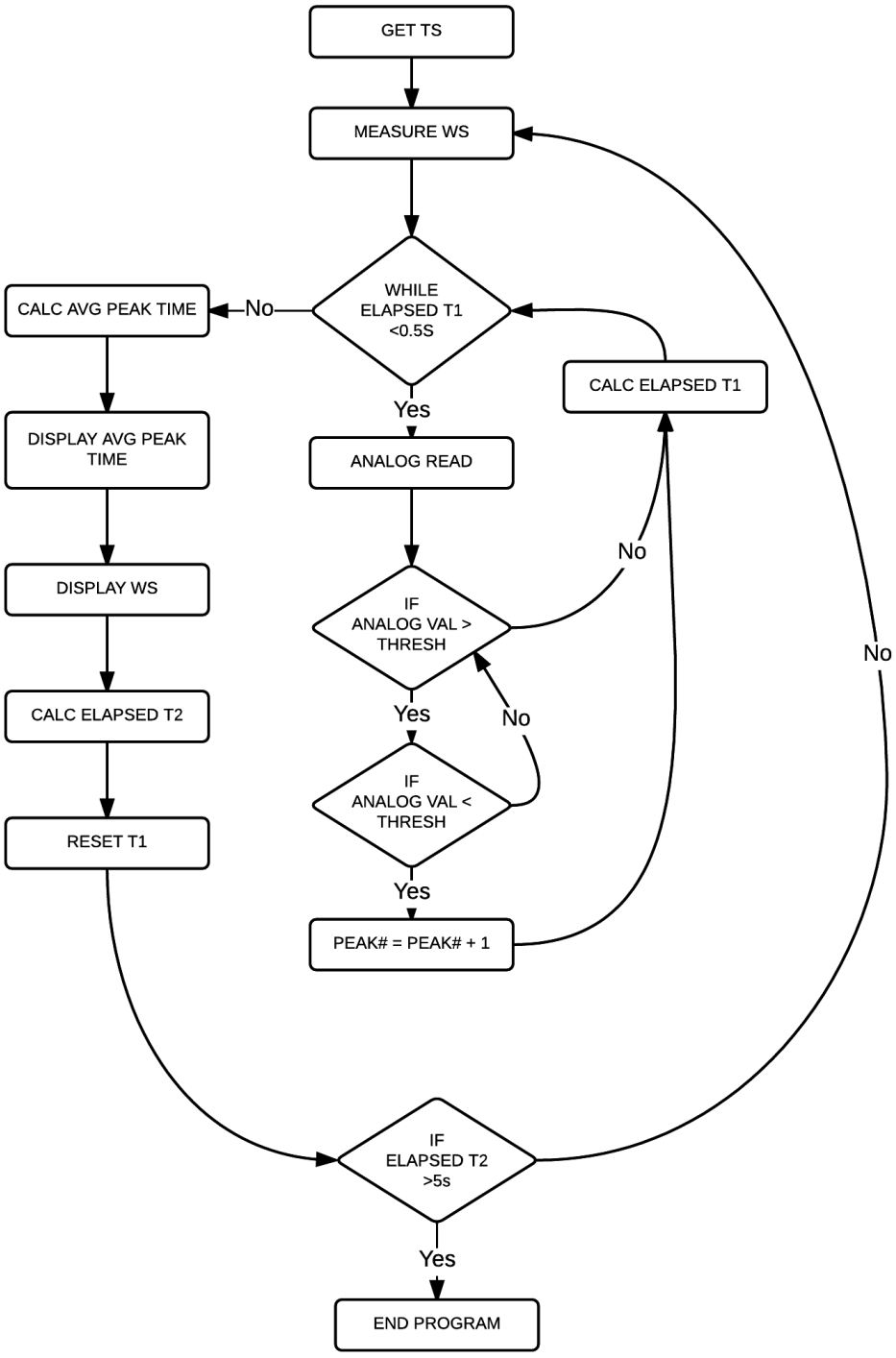


Figure 23: Wire Feed and Droplet Spacing Test Flowchart

5.4 Register Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	IP		IM	0	0	0	TP	NR	UD	BP		OM		OP				TP					TE		TD		K			XS	
0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0		0	1	0	0	0	0	0
0				1				6			8			2				0			2				0						

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0				0				0				0				8				0				0				A			

Table 1: Counter Mode Register Description

where,

- Row 3 - BIN VAL For PWM Mode
- Row 4 - HEX VAL
- Row 5 - BIN VAL For Frequency Measurement Mode
- Row 6 - HEX VAL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47
								1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47
								×	×	×	×	×	×	1	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
								×	×	×	×	0	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
								×	×	×	×	1	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
								×	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
								×	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
								×	×	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
								1	×	×	×	×	×	1	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
								×	×	×	×	1	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

Table 2: DIO[0] Register Description

where,

- Row 1 - Bit
- Row 2 - Channel
- Row 3 - Pin
- Row 4 - G1/G0 Read Mask
- Row 5 - G1/G0 Write Mask
- Row 6 - G1 INPUT ACTIVE
- Row 7 - G0 INPUT ACTIVE
- Row 8 - Motor CCW
- Row 9 - Motor CW
- Row 10 - Wire Speed Home Switch
- Row 11 - Weld and CNC stop
- Row 12 - CNC start

5.5 GUI Description

The initial plan was to have a GUI for our software and but was later not pursued due to time constraints. In GUI the user could see real-time graphed data coming from the current and temperature sensors as well as see the current wire speed. The system would use this feedback to automatically adjust the weld and keep it in a state that can be considered a "good weld". The plan also included manual overrides for the user to adjust the current and wire speed to their own desired result. Fig. 10 shows an initial GUI layout that we were aiming for.

5.6 The Graphical User Interface (GUI)

We were planning on using GTK+ in C programming language to generate the Graphical User Interface in order to view the different data and also control different settings.

The GTK+ is a library for creating graphical user interfaces. The library is created in C programming language. The GTK+ library is also called the GIMP toolkit. Originally, the library was created while developing the GIMP image manipulation program. Since then, the GTK+ became one of the most popular toolkits under Linux and BSD Unix. Today, most of the GUI software in the open source world is created in Qt or in GTK+. The GTK+ is an object oriented application programming interface. The object oriented system is created with the Glib object system, which is a base for the GTK+ library. The GObject also enables to create language bindings for various other programming languages. Language bindings exist for C++, Python, Perl, Java, C#, and other programming languages.

The GTK+ itself depends on the following libraries.

- Glib
- Pango
- ATK
- GDK
- GdkPixbuf
- Cairo

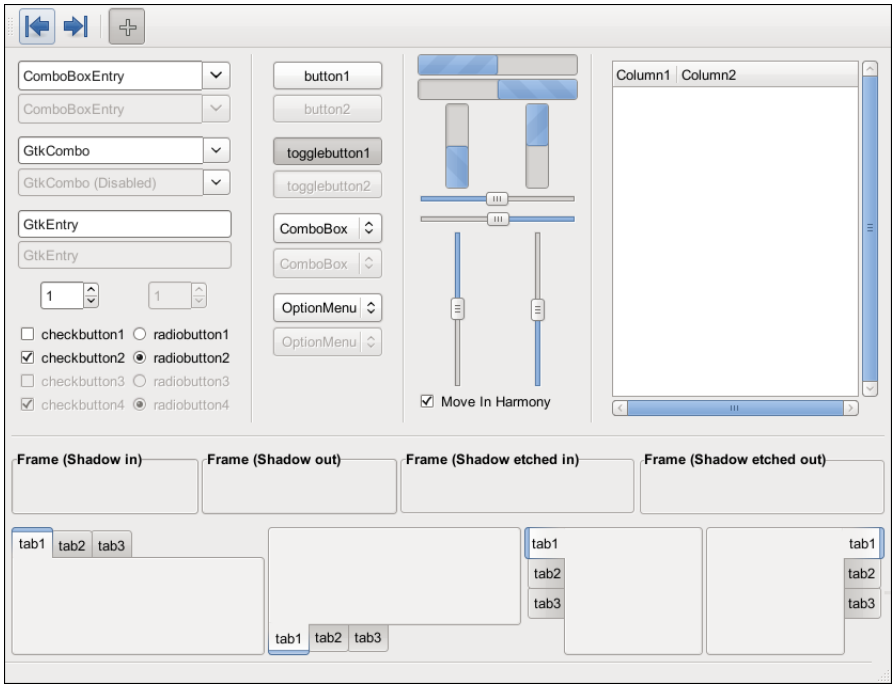
*Note: For detailed functions of each library refer to Appendix A

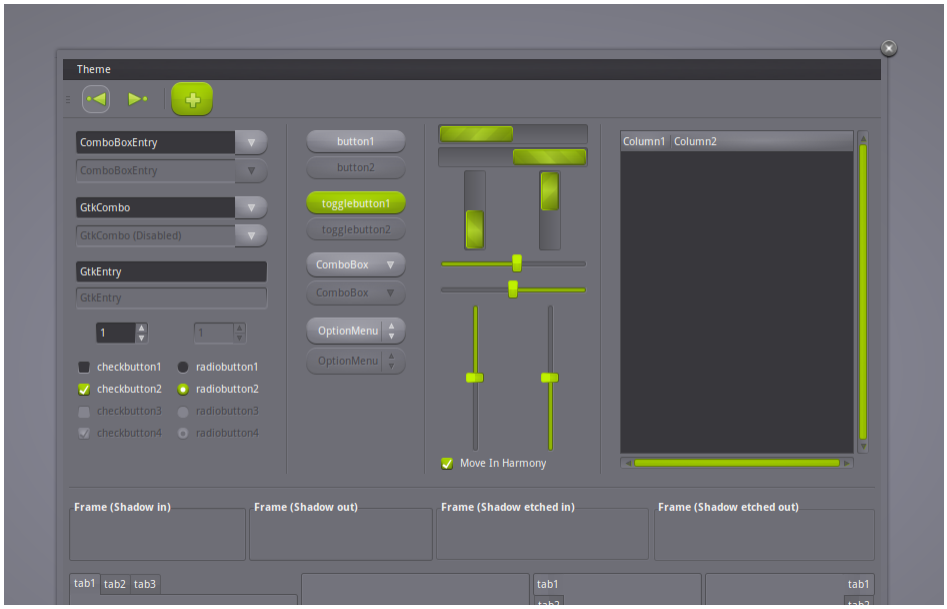
5.7 Reasons for using GTK+

- Language Bindings
  GTK+ is available in many other programming languages thanks to the language bindings available. This makes GTK+ quite an attractive toolkit for application development.

- Interfaces
  GTK+ has a comprehensive collection of core widgets and interfaces for use in your application.
  - Windows (normal window or dialog, about and assistant dialogs)
  - Displays (label, image, progress bar, status bar)
  - Buttons and toggles (check buttons, radio buttons, toggle buttons and link buttons)
  - Numerical (horizontal or vertical scales and spin buttons) and text data entry (with or without completion)
  - Multi-line text editor
  - Tree, list and icon grid viewer (with customizable renderers and model/view separation)
  - Combo box (with or without an entry)
  - Menus (with images, radio buttons and check items)
  - Toolbars (with radio buttons, toggle buttons and menu buttons)
  - GtkBuilder (creates your user interface from XML)
  - Selectors (color selection, file chooser, font selection)
  - Layouts (tabulated widget, table widget, expander widget, frames, separators and more)
  - Status icon (notification area on Linux, tray icon on Windows)
  - Printing widgets
  - Recently used documents (menu, dialog and manager)

5.8 Examples of GUIs created using GTK+





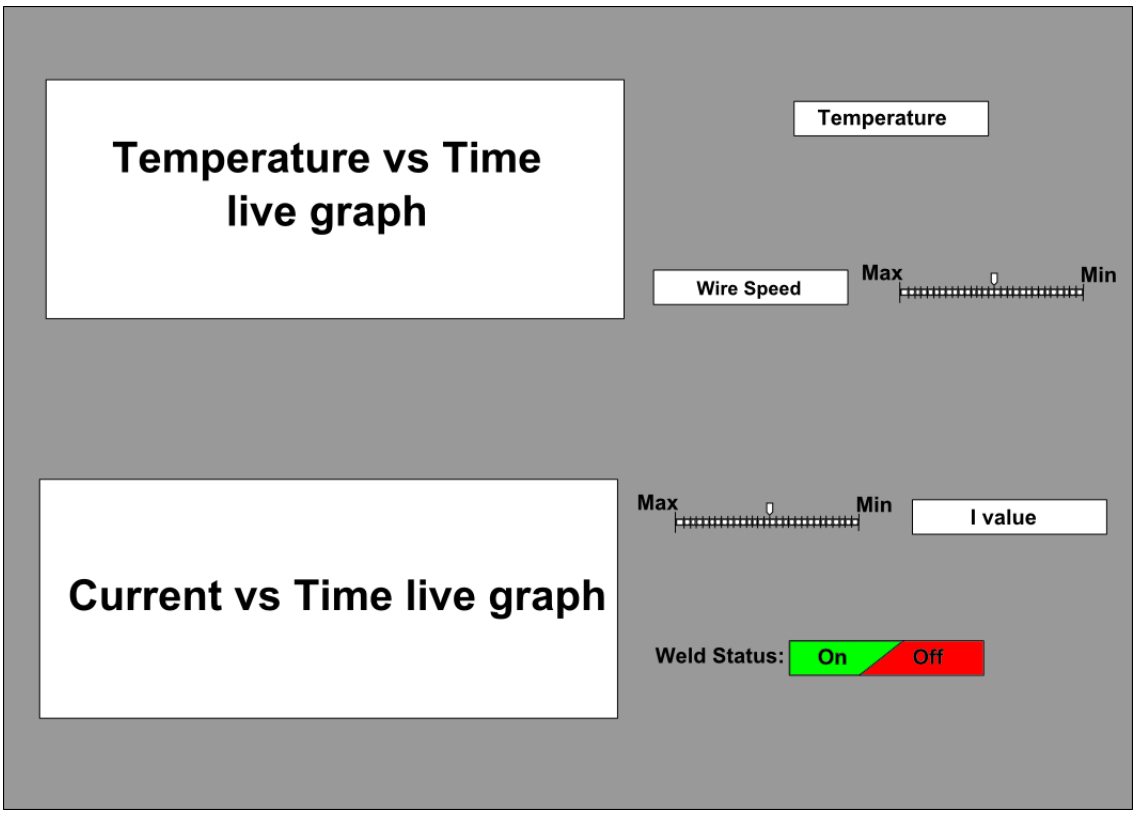


Figure 24: Proposed Rough GUI Layout

6 Testing

Testing was a constant part of this project. Many of the tests were small trouble shooting sessions where we would work on getting a specific piece of hardware or software to work. There were also several tests that were done more methodically, due to their importance of the final outcome. Some of these tests, with their test plans and results, can be seen below.

6.1 CNC Confirmation Test

Test Case # 1: CNC Confirmation Test

Test Writers: Cameron Tribe						
Test Case Name:		CNC Confirmation Test #1			**Test ID #:**	3MP-CNC-01
Description:		This test is to simply confirm that the CNC is operational in all three dimensions.			**Type:**	
Tester Information						
	Name of Tester:				**Date:**	05/30/2015
	Hardware Ver:				**Time:**	
	Setup:	Only Welder will be used, not CNC. Remove Cover of welder to ensure wire feed is properly set up. Welding will not be taking place, so be sure ground wire is not connected.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Load G-Code into Linux CNC and run initial homing procedure					
2	Place paper on base					
3	Fix felt marker onto CNC machine at an appropriate height so that it will draw on the paper.					
4	Run CNC Machine	This test verified that the CNC machine was working properly. Fig 14 shows the reproduced image that was created by the CNC Machine.				
	Overall Test Result:					

6.2 Wire Feed Test

Test Case # 4: Wire Speed Test

Test Writers: Branden Driver						
Test Case Name:		Wire Speed Encoder Test #1			**Test ID #:**	3MP-Wire-01
Description:		Measures the actual wire speed in correlation with the wire speed encoder. This test will allow for the construction of the lookup table to be used with the main program.			**Type:**	
Tester Information						
	Name of Tester:				**Date:**	
	Hardware Ver:	Display V1.0, Main Board V1.5, Sensor V1.0			**Time:**	
	Setup:	Only Welder will be used, not CNC. Remove Cover of welder to ensure wire feed is properly set up. Welding will not be taking place, so be sure ground wire is not connected.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Open wire feed program	Program should open				
2	Set program to run for one second	program should only run for one second				
3	Cut wire extruded from nozzle as short as possible	Very little wire should be showing				
4	Set wire feed speed nozzle to home setting	Wire feed should be at lowest setting				
5	Turn on welder	Welder should turn on				
6	Run test program	Welder should feed wire for exactly one second				
7	Turn off welder	Welder should turn off				
8	Cut wire extruded from nozzle as short as possible	Very little wire should be showing				
9	Measure length of cut wire and record value	Value determined for specified wire feed setting				
10	Repeat steps 1-9 for various values of wire speed	Determine if wire feed is linear. If so, interpolate for all wire feed speeds				
	Overall Test Result:					

Wire Speed (in/s)	Time (sec)	Expected Length (in)	Measured Length (in)
0.88	3	2.64	2.56
0.90	3	2.70	2.46
2.00	3	6.00	6.06
3.20	3	9.60	10.10
5.50	3	16.50	16.80
7.60	3	22.80	23.20
9.50	3	28.50	28.06

Table 3: Linear Velocity Program Test

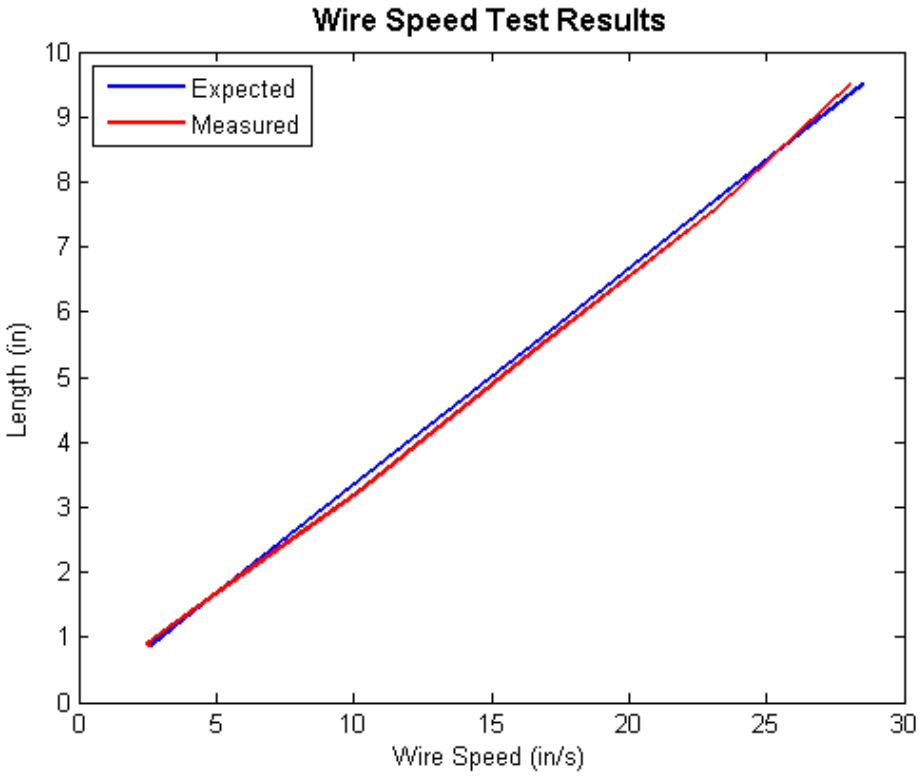


Figure 25: Wire Speed Test Results

6.3 Weld Quality Test

Test Case # 5: Weld Quality Test

Test Writers: Branden Driver						
Test Case Name:		Weld Quality Test #1			**Test ID #:**	3MP-Weld-01
Description:		This test will check the quality of the weld across a variety of currents, wire speeds, and CNC movement speeds			**Type:**	
Tester Information						
	Name of Tester:				**Date:**	
	Hardware Ver:	Display V1.0, Main Board V1.5, Sensor V1.0			**Time:**	
	Setup:	Ensure both welder and CNC are ready to use. Have at least one 1/8" baseplate on hand for each version of test you wish to run. In LinuxCNC, open file "m100". This file is a G-code program that will print seven 1-inch lines, each at a different CNC movement speed.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Set current level	Current level selected				
2	Set wire speed around 2	Wire speed selected				
3	Run program m100	Printing will begin				
4	During weld, run droplet spacing program	Program will output wire speed from encoder and average droplet spacing				
5	Quickly record both wire speed and average droplet spacing	Data acquired				
6	Repeat steps 4-5 for each of the seven welds of the program	First run is complete				
7	Adjust wire speed to 4	Welder ready for next run				
8	Run program m200	Print will continue on same plate, next to previous run				
9	Repeat steps 4-6	Data acquired for all 7 welds, 2nd run is complete				
10	Adjust wire speed to 6	Welder ready for next run				
11	Run program m300	Print will continue on same plate, next to previous run				
12	Repeat steps 4-6	Data acquired for all 7 welds, 3rd run is complete				
13	Adjust wire speed to 8	Welder ready for next run				
13	Run program m400	Print will continue on				

		same plate, next to previous run				
14	Repeat steps 4-6	Data acquired for all 7 welds, 4th run complete				
	Adjust wire speed to 8	Welder ready for next run				
	Run program m400	Print will continue on same plate, next to previous run				
	Run program m400	Data acquired for all 7 welds, 5th run complete				
	Overall Test Result:					

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (μs)
4.21	2	19047
4.04	3	20619
3.88	4	22857
3.98	5	21978
4.14	6	20833
3.88	7	20943
3.95	8	18872
6.02	2	13119
5.81	3	15037
6.19	4	14084
6.06	5	12539
5.71	6	15504
5.6	7	10257
6.54	8	17621
7.5	2	12820
7.2	3	14035
7.61	4	14760
7.67	5	15267
7.61	6	17167
7.45	7	11594
6.95	8	10790
8.99	2	11364
8.82	3	13114
9.58	4	11940
10	5	11765
8.5	6	10106
9.47	7	12751
9.84	8	n/a

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (μs)
11	2	10582
10.86	3	9557
10.65	4	10811
10.03	5	12270
11.05	6	11007
11.49	7	12012
11.85	8	11695

Table 4: Plate 1

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (μs)
3.48	2	20014
3.47	3	23121
3.3	4	23530
3.3	5	27398
2.96	6	26845
3.39	7	29646
3.42	8	30534
4.07	2	24342
4.19	3	19802
3.95	4	17467
3.86	5	18868
4.14	6	14545
4.09	7	13333
3.99	8	19900
6.55	2	11267
6.8	3	11173
7.23	4	12012
6.49	5	9788
6.38	6	9389
6.3	7	9216
6.68	8	9009
9.8	2	9442
9.57	3	8928
10.28	4	9456
10.89	5	9280
9.89	6	9204
9.83	7	8050
10.01	8	9456

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
11.49	2	8677
11.46	3	8340
11.43	4	8752
11.75	5	8798
11.6	6	9302
11.53	7	9029
11.69	8	9132

Table 5: Plate 2

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
5.34	2	12658
4.9	3	14053
4.97	4	12317
5.18	5	16461
4.53	6	11019
4.83	7	79756
n/a	8	n/a
7.12	2	9876
6.93	3	9959
7.22	4	10315
6.82	5	9527
6.74	6	9195
6.8	7	9227
6.87	8	9852
9.3	2	8948
9.19	3	9091
9.42	4	9599
9.26	5	9466
9.31	6	9204
n/a	7	n/a
9.16	8	8180
12.58	2	8421
12.41	3	8445
12.77	4	8465
12.74	5	8792
12.56	6	8620
12.66	7	8602
12.82	8	8714
14.67	2	8477
15.19	3	8585
14.87	4	8756
14.69	5	8547
14.89	6	8677
14.32	7	8639
14.65	8	8403

Table 6: Plate 3

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
3.83	4	9013
3.85	4.5	9828
3.89	5	9780
4	5.5	11662
3.96	6	9552
3.97	6.5	10447
3.8	7	10371
4.66	4	9860
5.07	4.5	9780
4.61	5	9569
4.64	5.5	9238
4.55	6	9501
4.59	6.5	9099
4.75	7	9287
4.77	4	9909
4.63	4.5	10032
4.6	5	9029
5.19	5.5	9784
4.56	6	n/a
4.67	6.5	9756
4.65	7	9434
5.35	4	9645
5.37	4.5	10025
5.34	5	9758
5.35	5.5	10554
5.61	6	9546
5.17	6.5	9961
5.36	7	9307
6.1	4	8877
5.98	4.5	9412
6.09	5	9615
6.03	5.5	9350
6.76	6	9863
6.35	6.5	10474
6.15	7	9198

Table 7: Plate 4

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
4.09	4.8	9195
4.47	5	9737
4.28	5.2	10126
3.39	5.4	9623
4.25	5.6	9740
4.09	5.8	9876
3.98	6	9262
4.61	4.8	9953
4.38	5	9761
4.56	5.2	9389
4.77	5.4	10582
4.56	5.6	9661
4.53	5.8	9625
4.2	6	9183
4.71	4.8	9266
4.61	5	9569
4.94	5.2	9479
4.62	5.4	9376
4.92	5.6	9184
4.89	5.8	9324
4.86	6	9546
5.21	4.8	8994
4.97	5	9111
5.31	5.2	9595
5.21	5.4	9324
5.02	5.6	9117
5.1	5.8	9153
5.26	6	9456
5.37	4.8	9111
5.56	5	9376
5.51	5.2	9456
	5.4	
	5.6	
	5.8	
	6	

Table 8: Plate 5

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
2.57	2	12903
2.55	3	12195
2.59	4	13559
2.48	5	15873
2.79	6	15269
2.62	7	16953
2.63	8	14760
3.38	2	10392
3.27	3	10309
3.72	4	11396
3.7	5	12232
3.39	6	11950
3.45	7	11954
3.52	8	14100
4.12	2	9464
4	3	9662
3.98	4	10000
4.26	5	10256
4.54	6	10816
4.41	7	10816
3.91	8	11665
4.47	2	9117
4.48	3	8960
4.47	4	9284
4.84	5	9756
4.39	6	9112
4.98	7	11628
4.37	8	11338
n/a	2	n/a
4.64	3	8658
4.67	4	8928
4.71	5	8565
5.07	6	11111
4.99	7	10638
5	8	10340

Table 9: Plate 6

Run #	Initial Wire Speed (in/s)	Final Wire Speed (in/s)	Turn Amount (deg)	Ratio (deg/(in/s))	Time/deg (us)	Pulse On Time (us)
Run 1	0.3	1.1	30	37.5	22222	666660
Run 2	1.1	1.8	30	42.85714286	22222	666660
Run 3	1.8	2.5	30	42.85714286	22222	666660
Run 4	2.5	3.3	30	37.5	22222	666660
Run 5	1.9	3	45	40.90909091	22222	999990
Run 6	3	4.2	45	37.5	22222	999990
Run 7	1.2	2.3	45	40.90909091	22222	999990
Run 8	2.3	3.4	45	40.90909091	22222	999990
Run 9	0.9	2.4	60	40	22222	1333320
Run 10	2.4	3.9	60	40	22222	1333320
Run 11	3.9	5.5	60	37.5	22222	1333320
Run 12	3.4	5	60	37.5	22222	1333320
Run 13	1	7	235	39.16666667	22222	5222170
			Average	**39.66179654**		

Table 10: Degree to Turn Measurements

7 Photos and Videos of Progress

Demo of the Printer
https://www.youtube.com/watch?v=Ypetogtn1Iw

One of the fist things done in this project was to confirm the operation of the CNC machine. To do this, G-Code of a 2D image was uploaded to the machine. A felt marker was used to draw the image below.

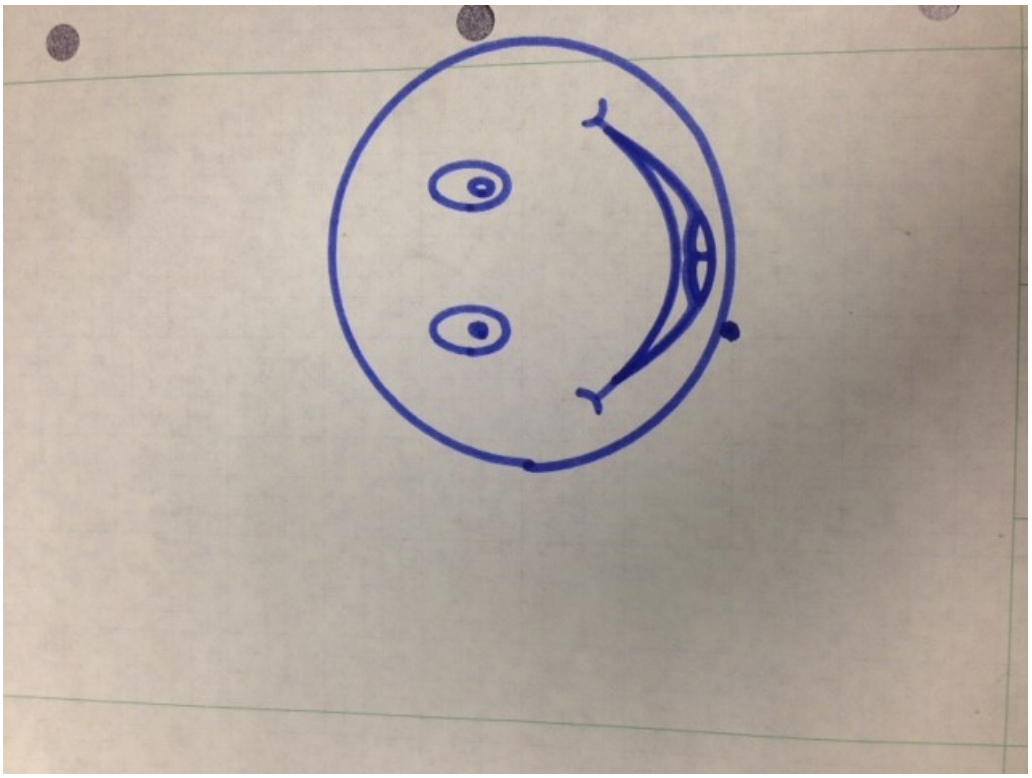


Figure 26: aaaaa

Next, we fitted the welder to the machine, and placed a metal base plate to weld on. To control the welder we just used our hand to active the weld while the machine was moving.

After this, we connected the relay switch circuit in parallel with the manual welder switch, using G-Code to activate the switch. Shown Below is the result of letting the CNC Machine control the weld.

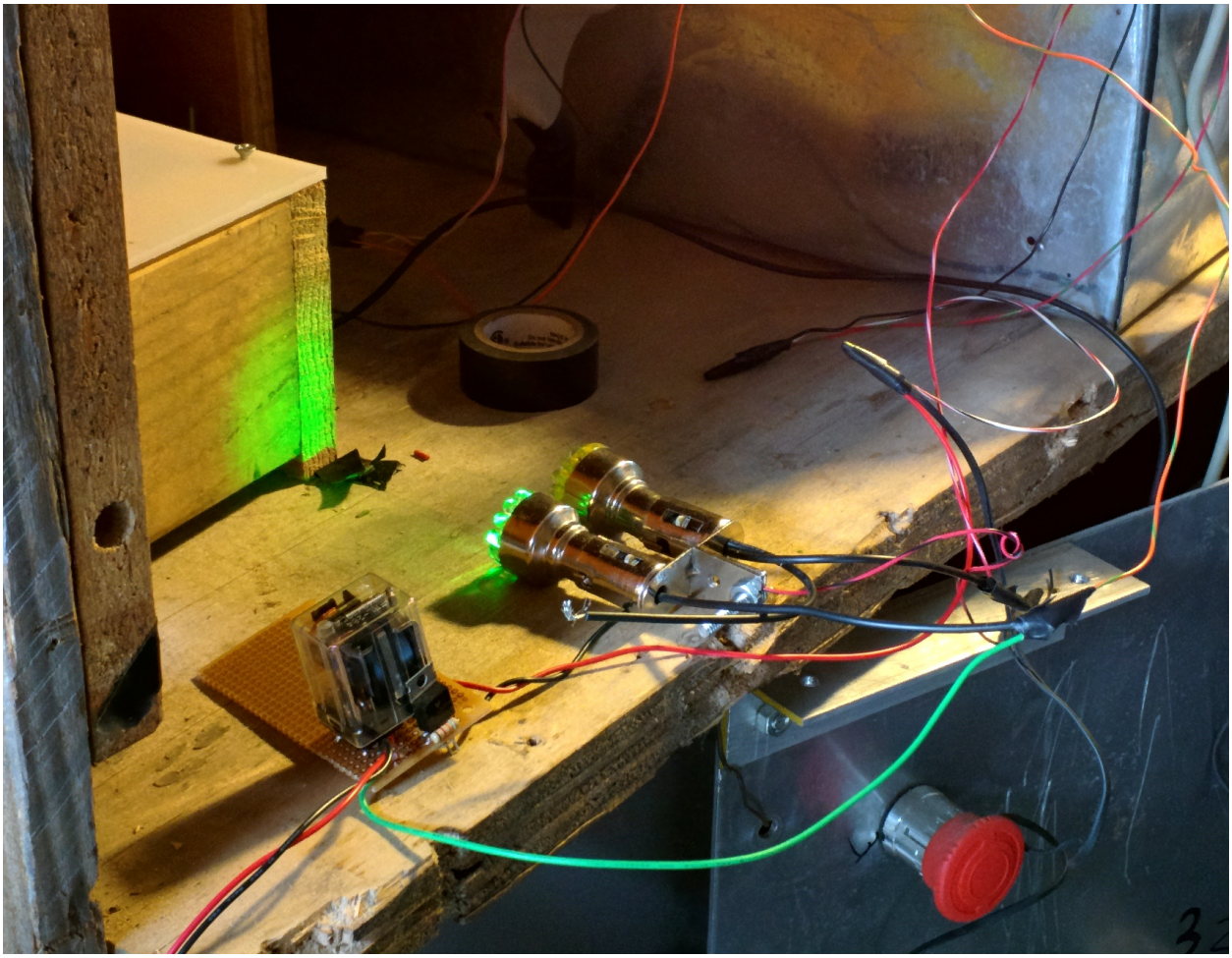


Figure 27: Proposed Rough GUI Layout

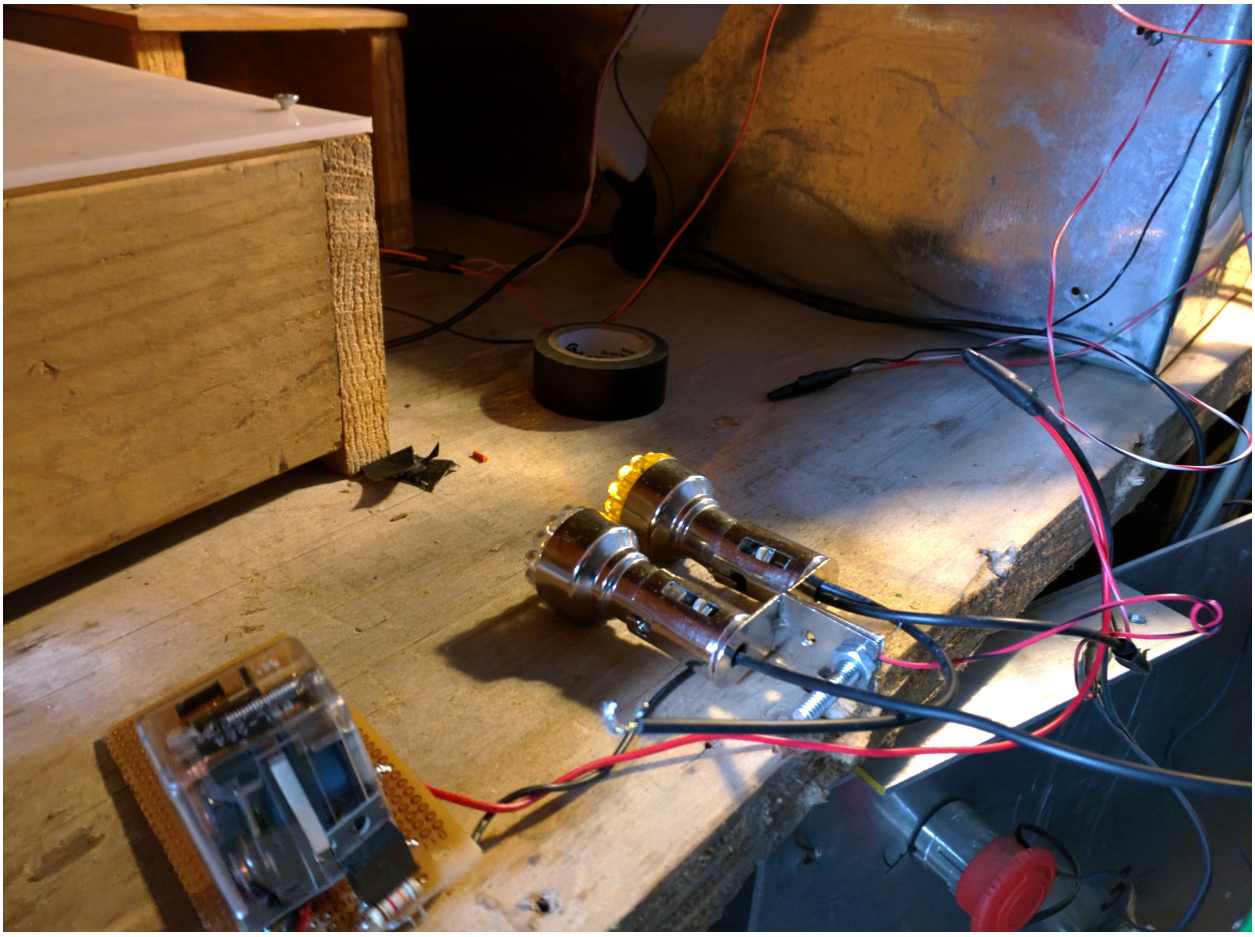


Figure 28: Proposed Rough GUI Layout

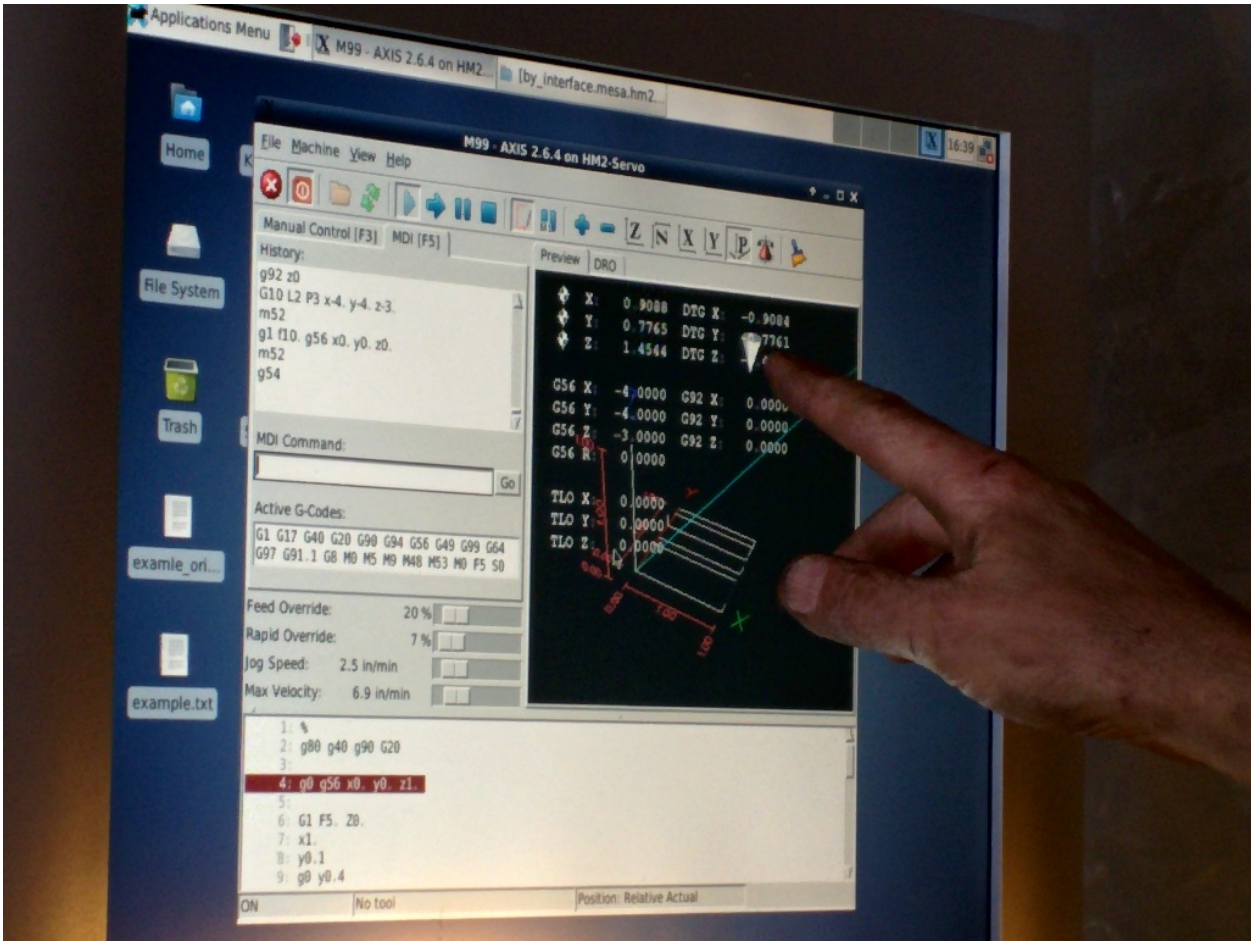


Figure 29: Proposed Rough GUI Layout

Figure 30: Proposed Rough GUI Layout

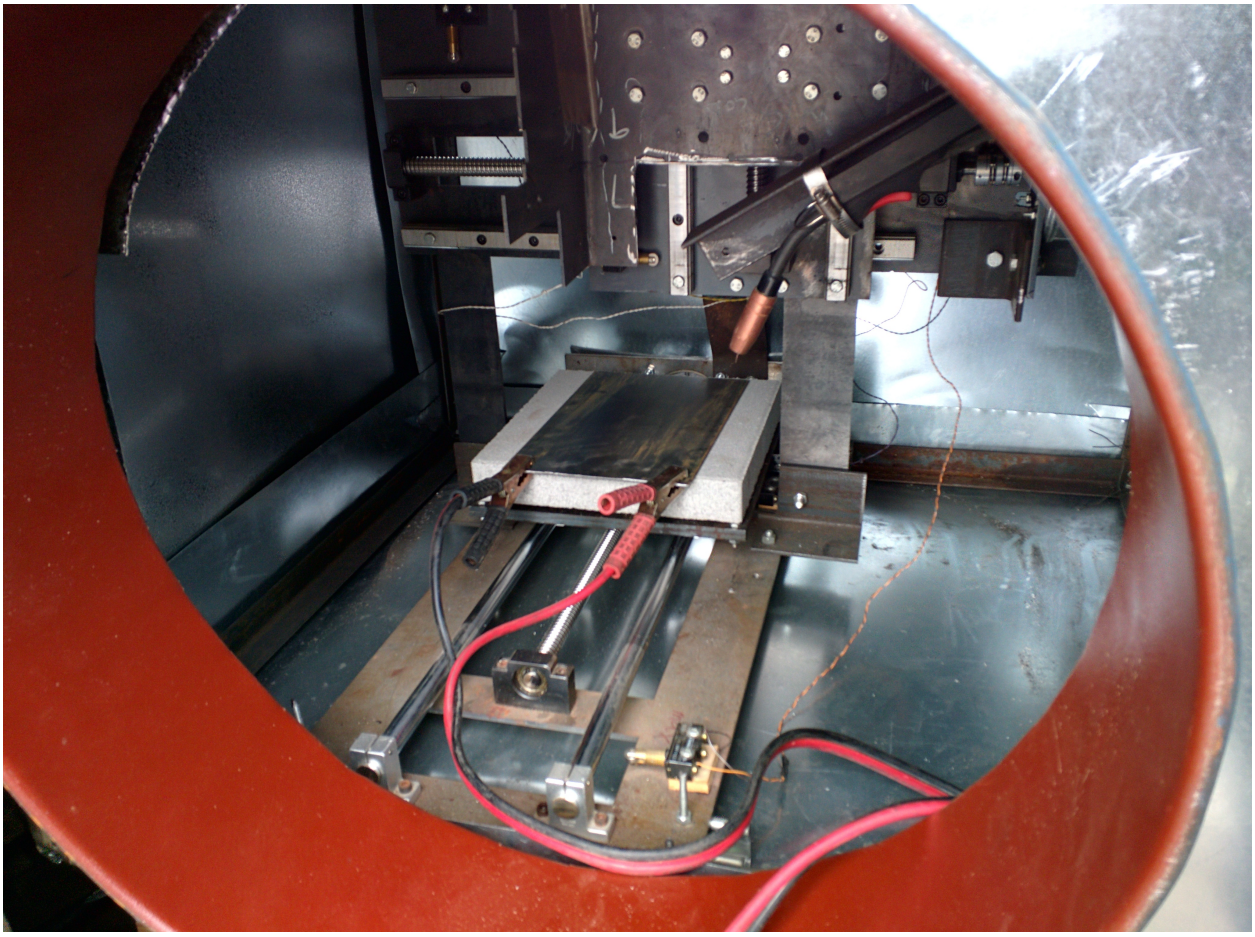


Figure 31: Proposed Rough GUI Layout

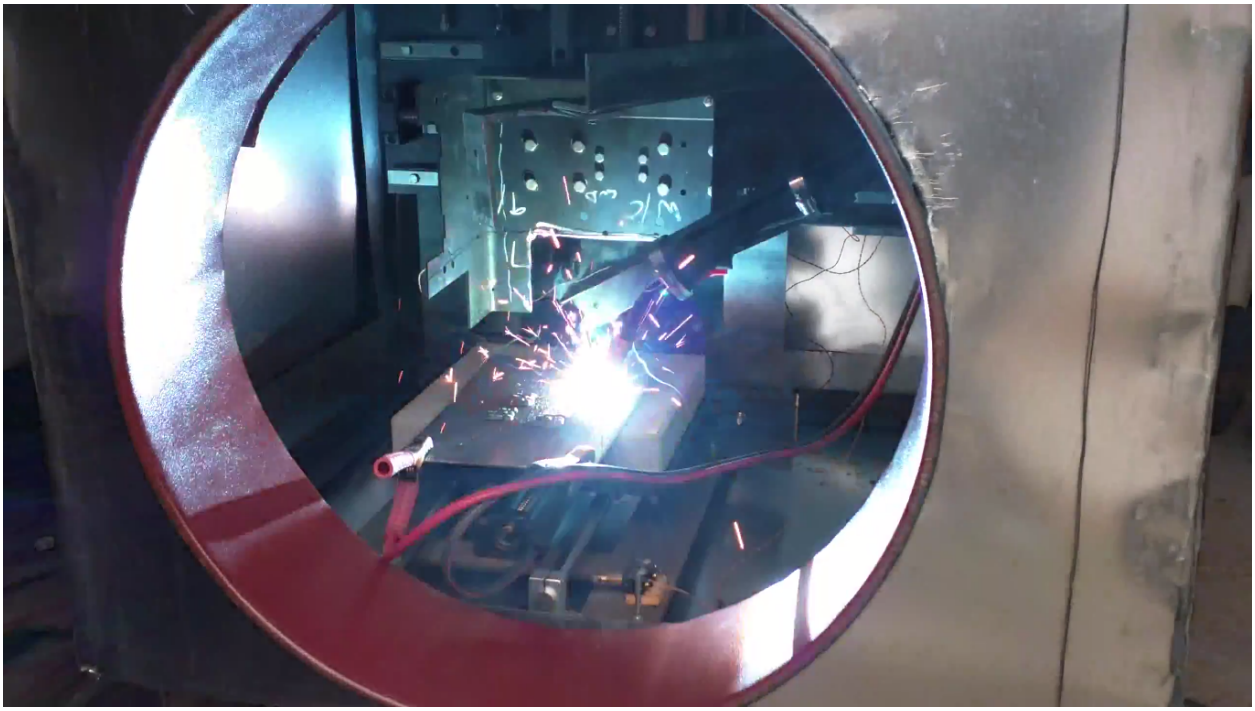


Figure 32: Proposed Rough GUI Layout

8 Bill of Materials (BOM)

Item	Quantity	Part Number	Mfg	Price	Description
CNC Machine	1	NPN	Aram Kasparov	~$500	
- X-Stepper Motor	1	IG34CK-32-IE8192-SB213	Servo Dynamics	$400	Stepper Motor for the X axis
- Y-Stepper Motor	1	HJ130E8-1308	Servo Dynamics	$800	Stepper Motor for the Y axis
- Z-Stepper Motor	1	IG34CK-32-IE8192-SB213	Servo Dynamics	$450	Stepper Motor for the Z axis
- Motor Driver	3	SD94	Servo Dynamics	$550	Motor driver for XYZ axis motors.
Limit and Home Switches	11	Z15G1308	HIGHLY	$10	Any limit and home switch used on the machine
PCI I/O Card	1	5I20	Mesa Electronics	$250	Part of the CNC system
Servo Interface Card	1	7i33	Mesa Electronics	$100	Interface Card for communicating with servo motors
Isolated I/O Card	2	7i37	Mesa Electronics	$100	I/O card used for limit and home switches
50 Pin Breakout Board	3	NPN		~$10	To connect to analog & digital channels of the Sensoray
Screw Terminals		NPN	Capstone 2015 Team		
50 Pin Connector		NPN	Capstone 2015 Team		
26 Pin Breakout Board	2	NPN		~$10	To connect to the counter channels of the Sensoray
Screw Terminals					
26 Pin Connector					
Incremental Encoder	1	S5-5000-250-IE-D-B	US Digital		Used for measuring the wire speed of the welder

Item	Quantity	Part Number	Mfg	Price	Description
MIG Welder	1	MIG 180	Chicgao Electric	$79.95	Wire Feed Welder that uses inert shielding gas
Motor Controller	2	KL-5056D	Keling Technology Inc		To control the motors on the welder control knobs
Stepper Motors	1				To control the knobs on the welder
PWM Module	1				To externally set CNC speed

Appendix A: GTK+

- **Glib**
  GLib provides the core application building blocks for libraries and applications written in C. It provides the core object system used in GNOME, the main loop implementation, and a large set of utility functions for strings and common data structures.

  Description

  - New types which are not part of standard C (but are defined in various C standard library header files) - gboolean, gsize, gssize,goffset, gintptr, guintptr.
  - Integer types which are guaranteed to be the same size across all platforms - gint8, guint8, gint16, guint16, gint32, guint32,gint64, guint64.
  - Types which are easier to use than their standard C counterparts - gpointer, gconstpointer, guchar, guint, gushort, gulong.

  GLib also defines macros for the limits of some of the standard integer and floating point types, as well as macros for suitableprintf() formats for these types.

  Standard Macros — commonly-used macros
  Type Conversion Macros — portably storing integers in pointer variables
  Byte Order Macros — a portable way to convert between different byte orders
  Numerical Definitions — mathematical constants, and floating point decomposition
  Miscellaneous Macros — specialized macros which are not used often
  Atomic Operations — basic atomic integer and pointer operations

  GLib Core Application Support

  - **The Main Event Loop** — manages all available sources of events
  - **Threads** — portable support for threads, mutexes, locks, conditions and thread private data
  - **Thread Pools** — pools of threads to execute work concurrently
  - **Asynchronous Queues** — asynchronous communication between threads
  - **Dynamic Loading of Modules** — portable method for dynamically loading 'plug-ins'
  - **Memory Allocation** — general memory-handling
  - **Memory Slices** — efficient way to allocate groups of equal-sized chunks of memory
  - **IO Channels** — portable support for using files, pipes and sockets
  - **Error Reporting** — a system for reporting errors
  - **Message Output and Debugging Functions** — functions to output messages and help debug applications
  - **Message Logging** — versatile support for logging messages with different levels of importance

GLib Utilities

- **String Utility Functions** — various string-related functions
- **Character Set Conversion** — convert strings between different character sets
- **Unicode Manipulation** — functions operating on Unicode characters and UTF-8 strings
- **Base64 Encoding** — encodes and decodes data in Base64 format
- **Data Checksums** — computes the checksum for data
- **Secure HMAC Digests** — computes the HMAC for data
- **Internationalization** — gettext support macros
- **Date and Time Functions** — calendrical calculations and miscellaneous time stuff
- **GTimeZone** — a structure representing a time zone
- **GDateTime** — a structure representing Date and Time
- **Random Numbers** — pseudo-random number generator
- **Hook Functions** — support for manipulating lists of hook functions
- **Miscellaneous Utility Functions** — a selection of portable utility functions
- **Lexical Scanner** — a general purpose lexical scanner
- **Timers** — keep track of elapsed time
- **Spawning Processes** — process launching
- **File Utilities** — various file-related functions
- **URI Functions** — manipulating URIs
- **Hostname Utilities** — Internet hostname utilities
- **Shell-related Utilities** — shell-like commandline handling
- **Commandline option parser** — parses commandline options
- **Glob-style pattern matching** — matches strings against patterns containing '*' (wildcard) and '?' (joker)
- **Perl-compatible regular expressions** — matches strings against regular expressions
- **Regular expression syntax** — syntax and semantics of regular expressions supported by GRegex
- **Simple XML Subset Parser** — parses a subset of XML
- **Key-value file parser** — parses .ini-like config files
- **Bookmark file parser** — parses files containing bookmarks
- **Testing** — a test framework
- **UNIX-specific utilities and integration** — pipes, signal handling

- **Windows Compatibility Functions** — UNIX emulation on Windows

GLib Data Types

- **Doubly-Linked Lists** — linked lists that can be iterated over in both directions
- **Singly-Linked Lists** — linked lists that can be iterated in one direction
- **Double-ended Queues** — double-ended queue data structure
- **Sequences** — scalable lists
- **Trash Stacks** — maintain a stack of unused allocated memory chunks
- **Hash Tables** — associations between keys and values so that given a key the value can be found quickly
- **Strings** — text buffers which grow automatically as text is added
- **String Chunks** — efficient storage of groups of strings
- **Arrays** — arrays of arbitrary elements which grow automatically as elements are added
- **Pointer Arrays** — arrays of pointers to any type of data, which grow automatically as new elements are added
- **Byte Arrays** — arrays of bytes
- **Balanced Binary Trees** — a sorted collection of key/value pairs optimized for searching and traversing in order
- **N-ary Trees**— trees of data with any number of branches
- **Quarks** — a 2-way association between a string and a unique integer identifier
- **Keyed Data Lists** — lists of data elements which are accessible by a string or GQuark identifier
- **Datasets** — associate groups of data elements with particular memory locations
- **GVariantType** — introduction to the GVariant type system
- **GVariant** — strongly typed value datatype
- **GVariant Format Strings** — varargs conversion of GVariants
- **GVariant Text Format** — textual representation of GVariants

Deprecated APIs

- **Deprecated thread API** — old thread APIs (for reference only)
- **Caches** — caches allow sharing of complex data structures to save resources
- **Relations and Tuples** — tables of data which can be indexed on any number of fields
- **Automatic String Completion** — support for automatic completion using a group of target strings

GLib Tools

- **glib-gettextize** — gettext internationalization utility
- **gtester** — test running utility
- **gtester-report** — test report formatting utility

- **Pango**

  Pango is a library for laying out and rendering of text, with an emphasis on internationalization. Pango can be used anywhere that text layout is needed, though most of the work on Pango so far has been done in the context of the GTK+ widget toolkit. Pango forms the core of text and font handling for GTK+-2.x.

  Pango is designed to be modular; the core Pango layout engine can be used with different font backends. There are three basic backends, with multiple options for rendering with each.

  - Client side fonts using the FreeType and fontconfig libraries, using HarfBuzz for complex-text handling. Rendering can be with with Cairo or Xft libraries, or directly to an in-memory buffer with no additional libraries.
  - Native fonts on Microsoft Windows using Uniscribe for complex-text handling. Rendering can be done via Cairo or directly using the native Win32 API.
  - Native fonts on MacOS X using CoreText for complex-text handling, rendering via Cairo.

  The integration of Pango with Cairo provides a complete solution with high quality text handling and graphics rendering.

  Dynamically loaded modules then handle text layout for particular combinations of script and font backend. Pango ships with a wide selection of modules, including modules for Hebrew, Arabic, Hangul, Thai, and a number of Indic scripts. Virtually all of the world's major scripts are supported. As well as the low level layout rendering routines, Pango includes PangoLayout, a high level driver for laying out entire blocks of text, and routines to assist in editing internationalized text. Pango depends on 2.x series of the GLib library.

- **ATK**

  ATK provides the set of accessibility interfaces that are implemented by other toolkits and applications. Using the ATK interfaces, accessibility tools have full access to view and control running applications.

  Base accessibility object

  - **AtkObject** — The base object class for the Accessibility Toolkit API.

Event and Toolkit Support

- **AtkUtil** — A set of ATK utility functions for event and toolkit support

ATK Interfaces

- **AtkAction** — The ATK interface provided by UI components which the user can activate/interact with.
- **AtkComponent** — The ATK interface provided by UI components which occupy a physical area on the screen. which the user can activate/interact with.
- **AtkDocument** — The ATK interface which represents the toplevel container for document content.
- **AtkEditableText** — The ATK interface implemented by components containing user-editable text content.
- **AtkHyperlinImpl** — An interface from which the AtkHyperlink associated with an AtkObject may be obtained.
- **AtkHypertext** — The ATK interface which provides standard mechanism for manipulating hyperlinks.
- **AtkImage** — The ATK Interface implemented by components which expose image or pixmap content on-screen.
- **AtkSelection** - The ATK interface implemented by container objects whose AtkObject children can be selected.
- **AtkStreamableContent** — The ATK interface which provides access to streamable content.
- **AtkTable** — The ATK interface implemented for UI components which contain tabular or row/column information.
- **AtkTableCell** - The ATK interface implemented for a cell inside a two-dimentional AtkTable
- **AtkText** — The ATK interface implemented by components with text content.
- **AtkValue** — The ATK interface implemented by valuators and components which display or select a value from a bounded range of values.
- **AtkWindow** — The ATK Interface provided by UI components that represent a top-level window.

Basic accessible data types

- **AtkRange** - A given range or subrange, to be used with AtkValue
- **AtkRelation** — An object used to describe a relation between a object and one or more other objects.
- **AtkRelationSet** — A set of AtkRelations, normally the set of AtkRelations which an AtkObject has.
- **AtkState** — An AtkState describes a single state of an object.
- **AtkStateSet** — An AtkStateSet contains the states of an object.

Custom accessible objects

- **AtkGObjectAccessible** — This object class is derived from AtkObject and can be used as a basis implementing accessible objects.
- **AtkHyperlink** — An ATK object which encapsulates a link or set of links in a hypertext document.
- **AtkNoOpObject** — An AtkObject which purports to implement all ATK interfaces.
- **AtkPlug** — Toplevel for embedding into other processes
- **AtkSocket** — Container for AtkPlug objects from other processes

Utilities

- **AtkNoOpObjectFactory** — The AtkObjectFactory which creates an AtkNoOpObject.
- **AtkObjectFactory** — The base object class for a factory used to create accessible objects for objects of a specific GType.
- **AtkRegistry** — An object used to store the GType of the factories used to create an accessible object for an object of a particular GType.
- **Versioning macros** — Variables and functions to check the ATK version

Deprecated Interfaces

- **AtkMisc** — A set of ATK utility functions for thread locking

- **GDK**

  I **API Reference**

  - **General** — Library initialization and miscellaneous functions
  - **GdkDisplayManager** — Maintains a list of all open GdkDisplays
  - **GdkDisplay** — Controls a set of GdkScreens and their associated input devices
  - **GdkScreen** — Object representing a physical screen
  - **GdkDeviceManager** — Functions for handling input devices
  - **GdkDevice** — Object representing an input device
  - **Points and Rectangles** — Simple graphical data types
  - **Pixbufs** — Functions for obtaining pixbufs
  - **RGBA Colors** — RGBA colors
  - **Visuals** — Low-level display hardware information
  - **Cursors** — Standard and pixmap cursors
  - **Windows** — Onscreen display areas in the target window system
  - **Frame clock** — Frame clock syncs painting to a window or display

- **Frame timings** — Object holding timing information for a single frame
- **GdkGLContext** — OpenGL context
- **Events** — Functions for handling events from the window system
- **Event Structures** — Data structures specific to each type of event
- **Key Values** — Functions for manipulating keyboard codes
- **Selections** — Functions for transfering data via the X selection mechanism
- **Drag And Drop** — Functions for controlling drag and drop handling
- **Properties and Atoms** — Functions to manipulate properties on windows
- **Threads** — Functions for using GDK in multi-threaded programs
- **Pango Interaction** — Using Pango in GDK
- **Cairo Interaction** — Functions to support using cairo
- **X Window System Interaction** — X backend-specific functions
- **Wayland Interaction** — Wayland backend-specific functions
- **Application launching** — Startup notification for applications
- **Testing** — Test utilities

II **Deprecated**

- **Colors** — Manipulation of colors

- **GdkPixbuf**

I **API Reference**

- **Initialization and Versions** — Library version numbers.
- **The GdkPixbuf Structure** — Information that describes an image.
- **Reference Counting and Memory Management** — Functions for reference counting and memory management on pixbufs.
- **File Loading** — Loading a pixbuf from a file.
- **File saving** — Saving a pixbuf to a file.
- **Image Data in Memory** — Creating a pixbuf from image data that is already in memory.
- **Inline data** — Functions for inlined pixbuf handling.
- **Scaling** — Scaling pixbufs and scaling and compositing pixbufs
- **Rendering** — Rendering a pixbuf to a GDK drawable.
- **Drawables to Pixbufs** — Getting parts of a GDK drawable's image data into a pixbuf.
- **Utilities** — Utility and miscellaneous convenience functions.
- **Animations** — Animated images.
- **GdkPixbufLoader** — Application-driven progressive image loading.
- **Module Interface** — Extending GdkPixBuf
- **gdk-pixbuf Xlib initialization** — Initializing the gdk-pixbuf Xlib library.

- **Xlib Rendering** — Rendering a pixbuf to an X drawable.
- **X Drawables to Pixbufs** — Getting parts of an X drawable's image data into a pixbuf.
- **XlibRGB** — Rendering RGB buffers to X drawables.

II **Tools Reference**

- **gdk-pixbuf-csource** — C code generation utility for GdkPixbuf images
- **gdk-pixbuf-query-loaders** — GdkPixbuf loader registration utility

- **Cairo**

  A Vector Graphics Library.

  Drawing

  - **cairo_t** — The cairo drawing context
  - **Paths** — Creating paths and manipulating path data
  - **cairo_pattern_t** — Sources for drawing
  - **Regions** — Representing a pixel-aligned area
  - **Transformations** — Manipulating the current transformation matrix
  - **text** — Rendering text and glyphs
  - **Raster Sources** — Supplying arbitrary image data

  Fonts

  - **cairo_font_face_t** — Base class for font faces
  - **cairo_scaled_font_t** — Font face at particular size and options
  - **cairo_font_options_t** — How a font should be rendered
  - **FreeType Fonts** — Font support for FreeType
  - **Win32 Fonts** — Font support for Microsoft Windows
  - **Quartz (CGFont) Fonts** — Font support via CGFont on OS X
  - **User Fonts** — Font support with font data provided by the user

  Surfaces

  - **cairo_device_t** — interface to underlying rendering system
  - **cairo_surface_t** — Base class for surfaces
  - **Image Surfaces** — Rendering to memory buffers
  - **PDF Surfaces** — Rendering PDF documents
  - **PNG Support** — Reading and writing PNG images

- **PostScript Surfaces** — Rendering PostScript documents
- **Recording Surfaces** — Records all drawing operations
- **Win32 Surfaces** — Microsoft Windows surface support
- **SVG Surfaces** — Rendering SVG documents
- **Quartz Surfaces** — Rendering to Quartz surfaces
- **XCB Surfaces** — X Window System rendering using the XCB library
- **XLib Surfaces** — X Window System rendering using XLib
- **XLib-XRender Backend** — X Window System rendering using XLib and the X Render extension
- **Script Surfaces** — Rendering to replayable scripts

Utilities

- **cairo_matrix_t** — Generic matrix operations
- **Error handling** — Decoding cairo's status
- **Version Information** — Compile-time and run-time version checks.
- **Types** — Generic data types

Appendix B: Sensoray 826i Manual

Appendix C: Linux CNC

Appendix D: Visual Studio 2013 on the 826i

Appendix E: Stepper Motor and Driver

Appendix F: Optical Shaft Encoder