Portland State University

Capstone Project Progress Report

Department of Electrical & Computer Engineering

3D Metal Printer

Members:
Cameron Tribe
Branden Driver
Brian Andrews
Ahmad Qazi

Supervisor:
Dr. Marek Perkowski

April 6, 2015

1 Project Overview

The team will interface a CNC machine with a MIG welder to create a 3D 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 will be 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 machine, in this case, the metal deposition process. The purpose for 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. 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 it is planned to use the welding machine with the cost of 400. Another example can be found here:
http://www.wired.co.uk/magazine/archive/2014/08/play/steel-sketch
AKTechnology plan is to manufacture parts for pump and compressors, and R&D part for all use. The goal is to fabricate low cost and highly usable machines.

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 is to upgrade this CNC motion controller – mill into 3D metal deposition printer by adding MIG welder instead of cutting tool spindle. The CNC motion control was reprogrammed. The MIG welder is operational.
The project will also build control to integrate CNC mill and MIG welding machine. Welder has 2 adjustment—- 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 logical 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. This object is chosen to be a cylindrical tube, 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
- And must control when material is being deposited
- Must have user interface
- Should allow for welder thermal shutdown
- May show amount of wire left on spool

Figure 1: General Black Box Diagram of the 3D Metal Printer

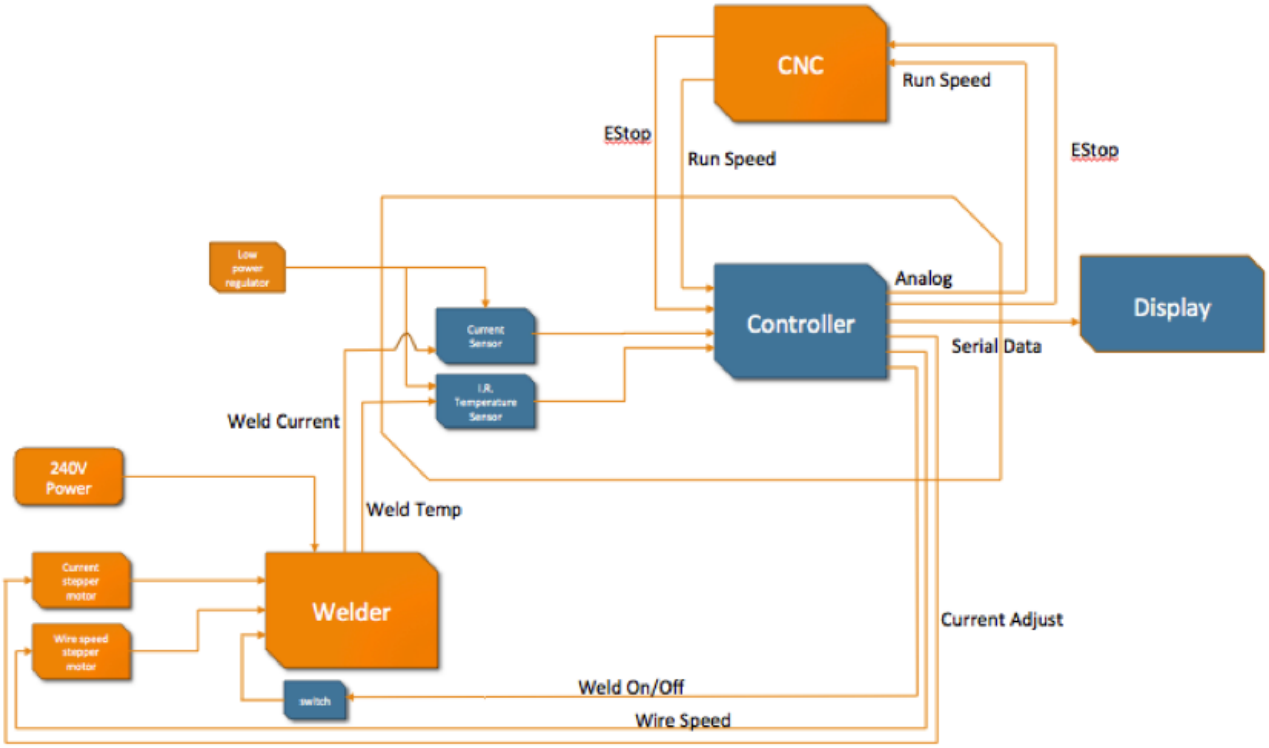


Figure 2: Detailed Black Box Diagram of the 3D Metal Printer

3 Hardware

3.1 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 debate 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.

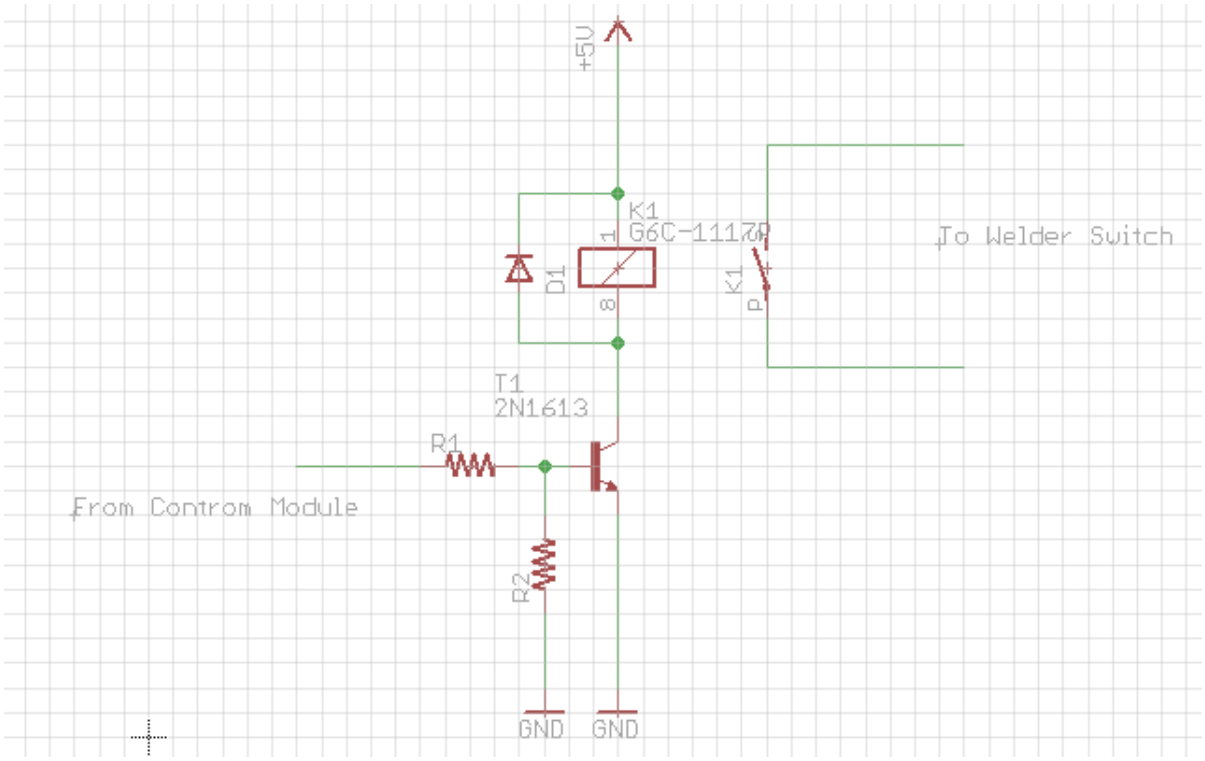


Figure 3: The Relay with the Transistor Driver

The signal to 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, and Linuix 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 5VDC 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 a single 26 pin connector. 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.

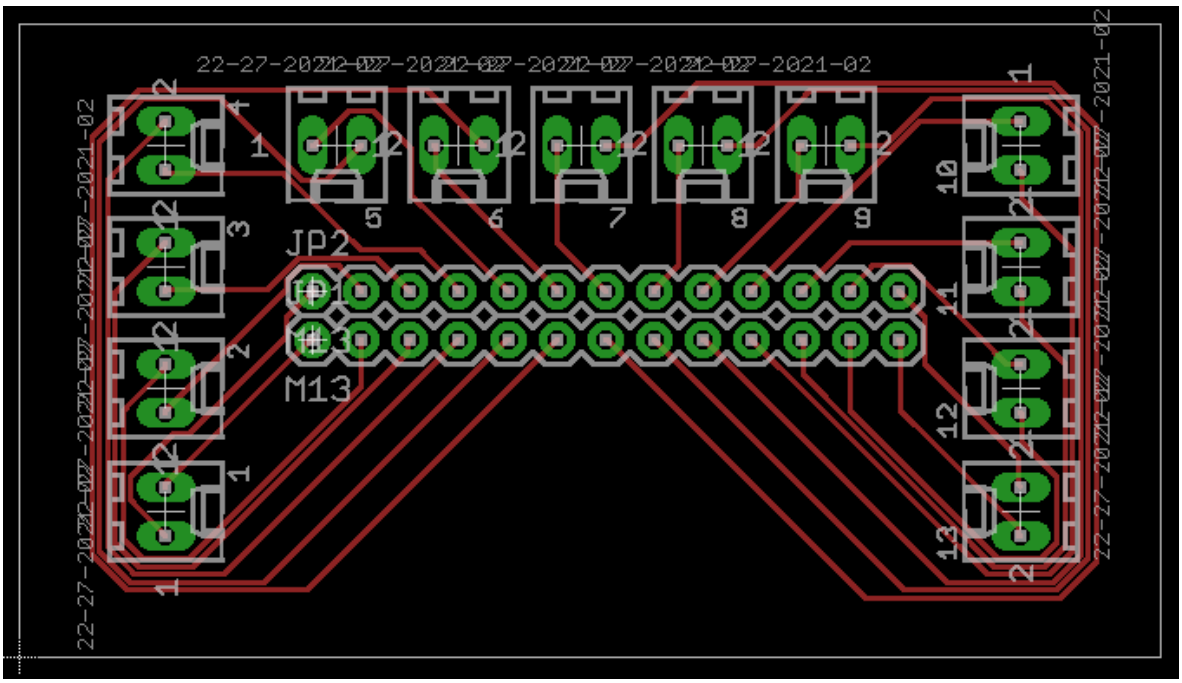


Figure 4: Schematic of the 26 pin Breakout Board for the Sensoray 826 I/O card

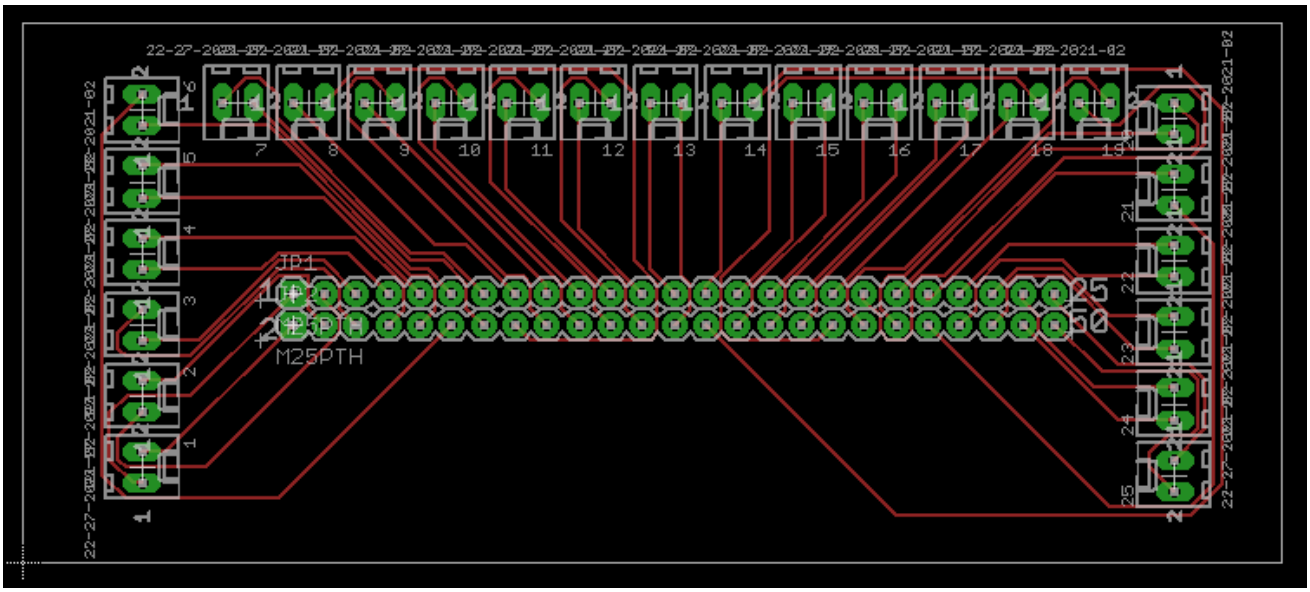


Figure 5: Schematic of the 50 pin Breakout Board for the Sensoray 826 I/O card

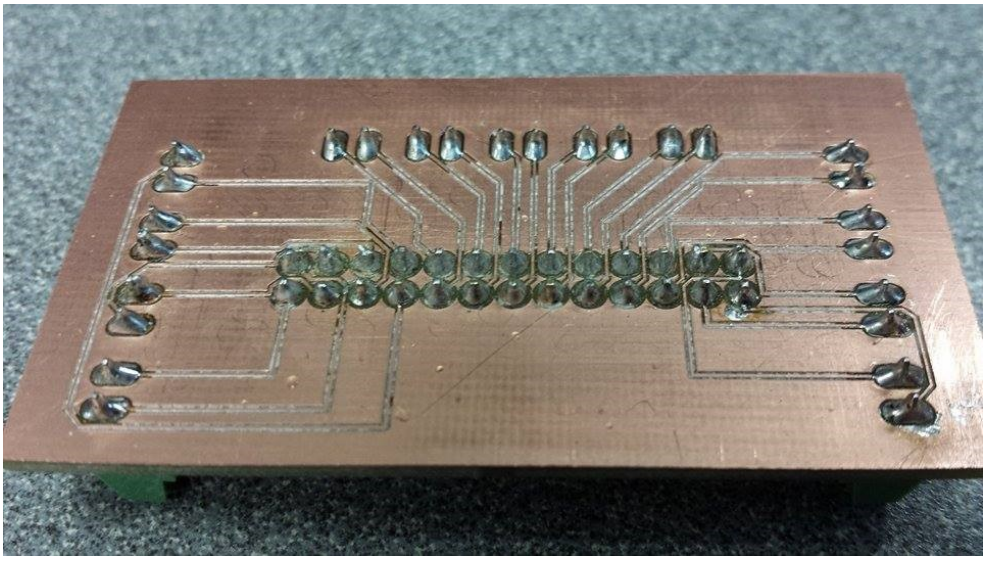


Figure 6: The 26 pin Breakout Board for The Sensoray 826 I/O card

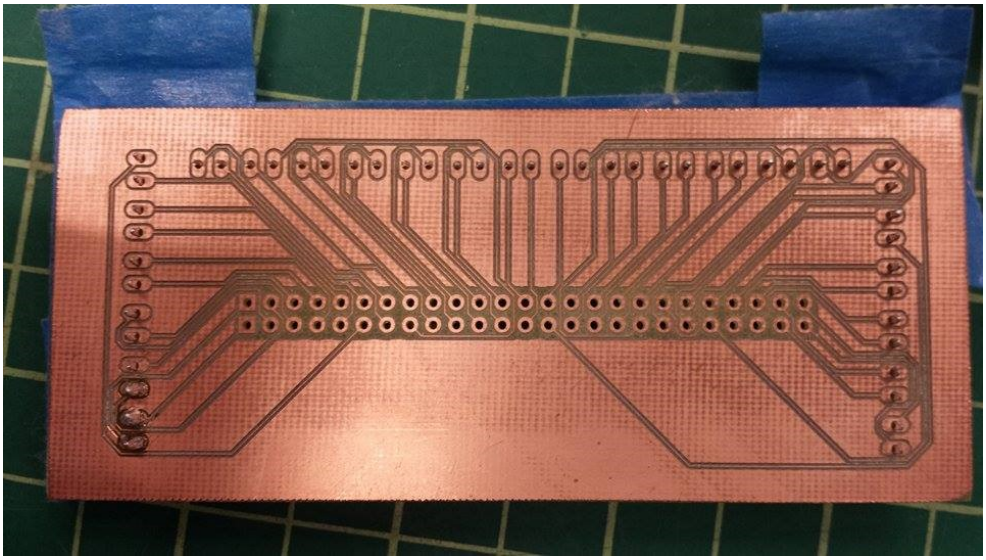


Figure 7: The 50 pin Breakout Board for the Sensoray 826 I/O card

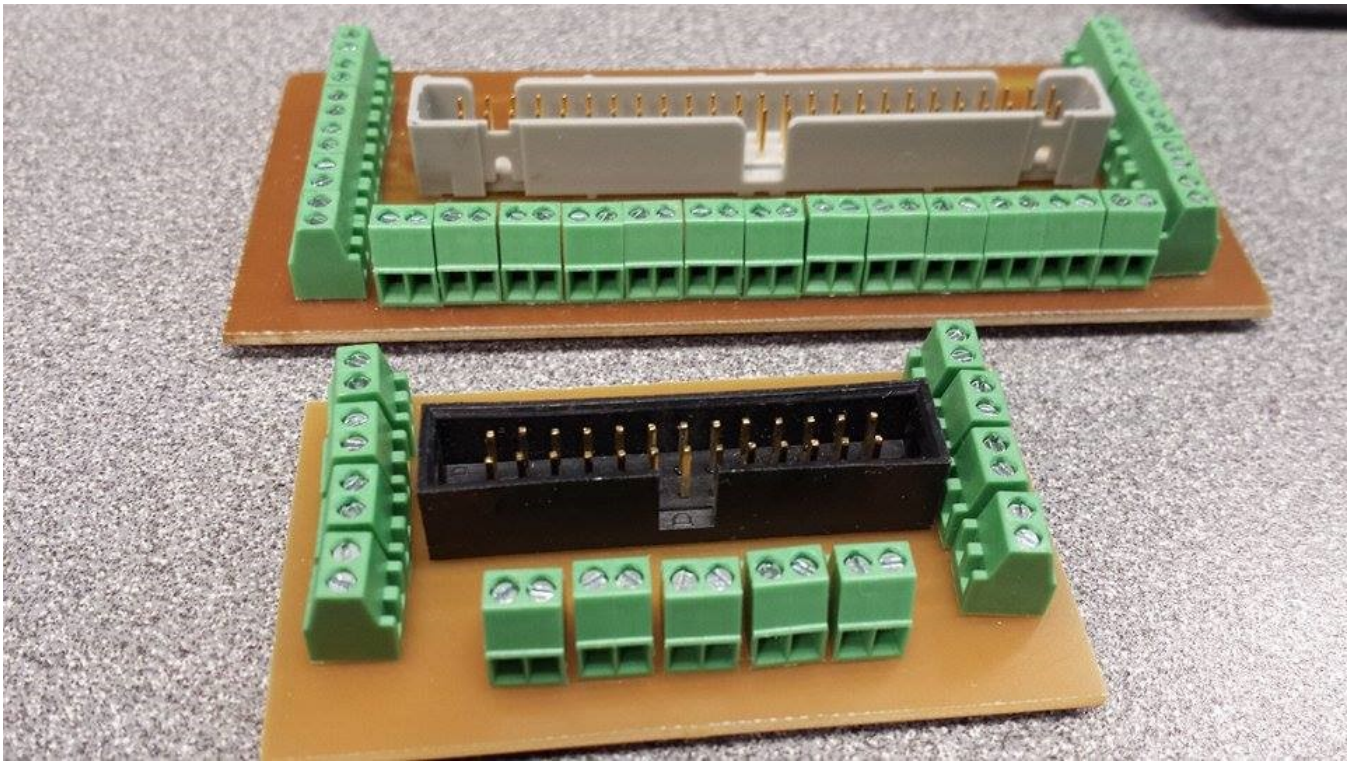
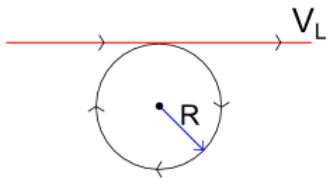


Figure 8: The 26 and 50 pin Breakout Board with connectors for The Sensoray 826 I/O card

3.2 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,

$$\boldsymbol{V = \frac{2\pi nr}{NT}}$$

**Note: For Sensoray 826, Max $f = 25MHz$*

Connections to 826

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

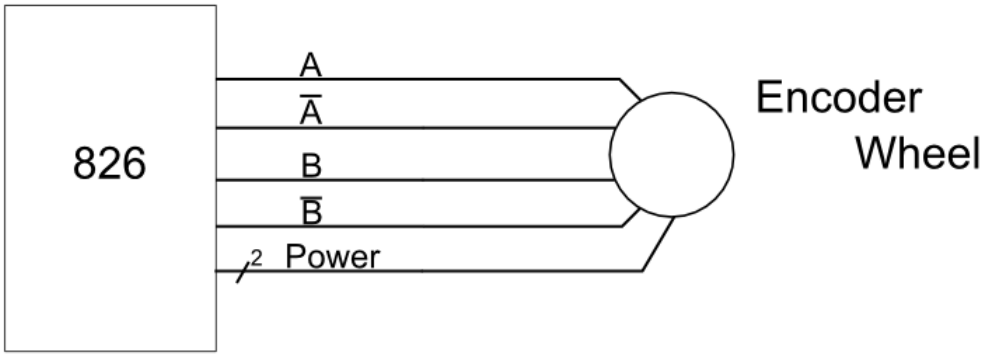


Figure 9: Encoder Connection

4 Software

4.1 GUI Description

Our software will have a GUI in which the user can see real-time graphed data coming from the current and temperature sensors as well as see the current wire speed. The system should use this feedback to automatically adjust the weld and keep it in a state that can be considered a "good weld". Also included here will be manual overrides for the user to adjust the current and wire speed to their own desired result. Below is an initial GUI layout that we will be aiming for.

4.2 The Graphical User Interface (GUI)

We are 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**
  GLib provides the core application building blocks for libraries and applications written

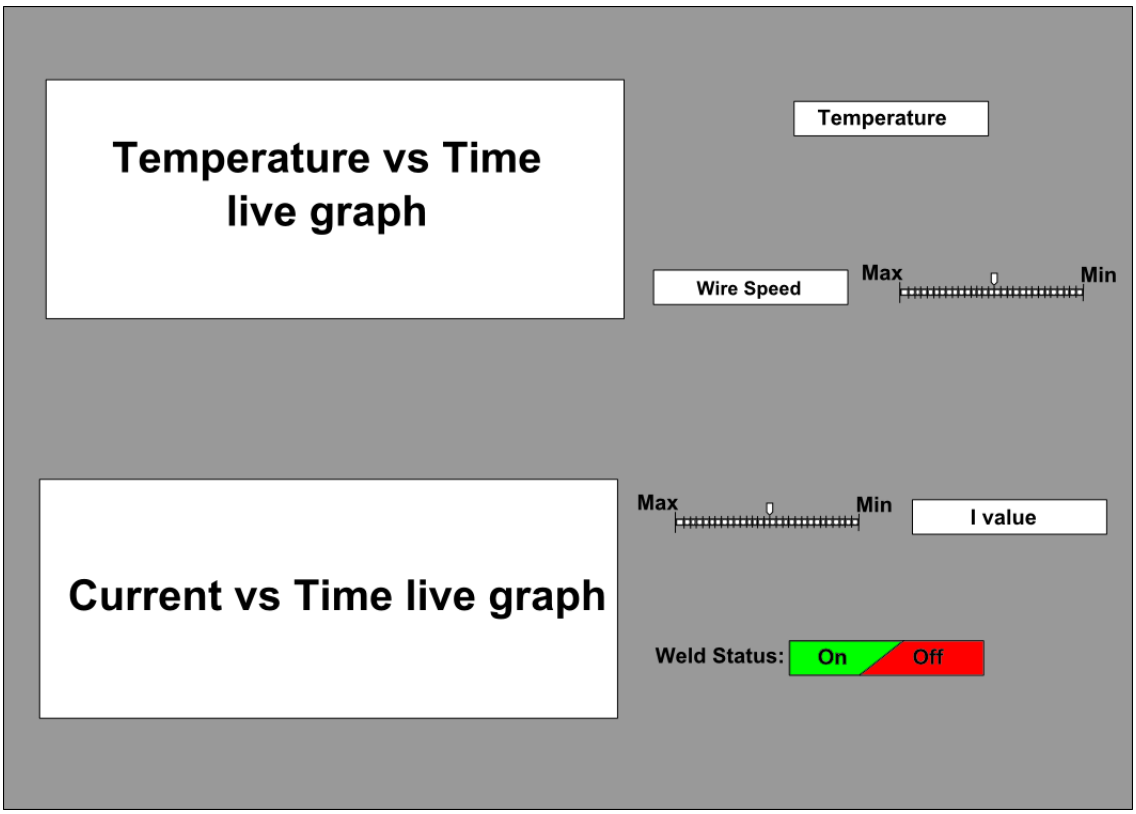


Figure 10: Proposed GUI Layout

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

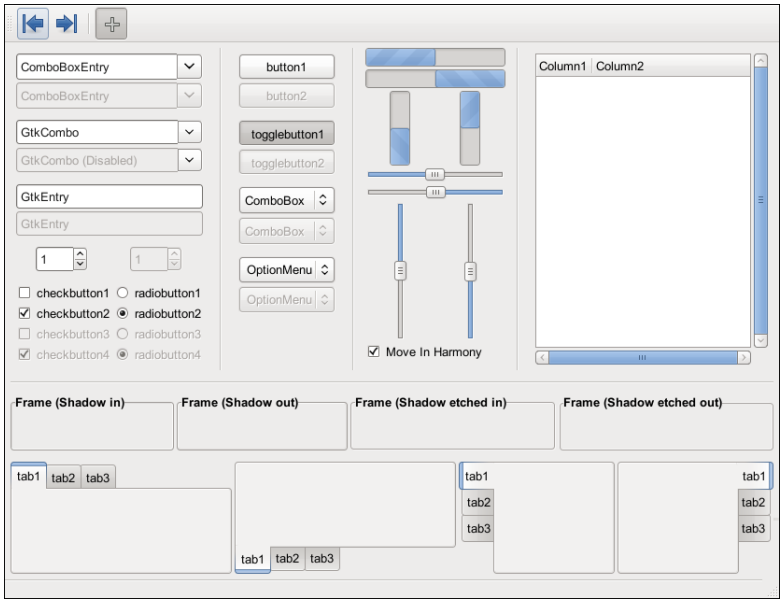
4.3 Reasons for using GTK+

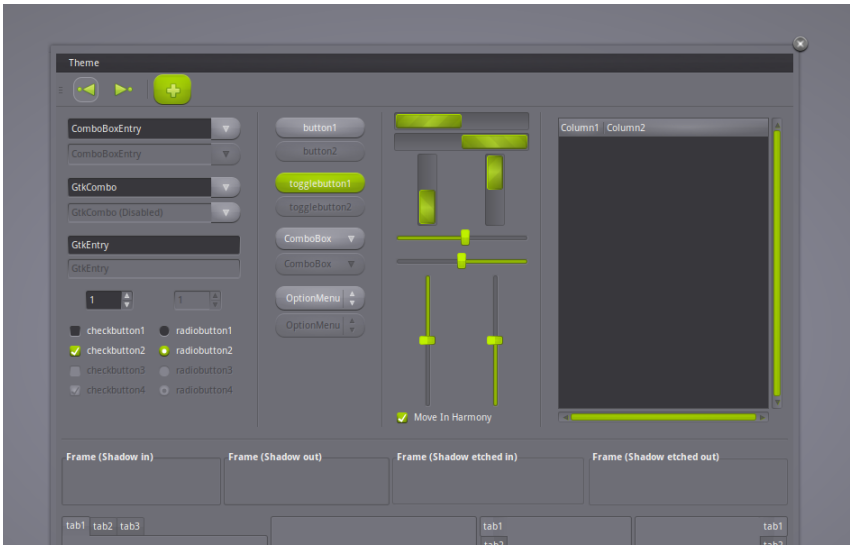
- Language Bindings
  GTK+ is available in many other programming languages thanks to the language bindings available. This makes GTK+ quite an attractive toolkit for application development.

- 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)

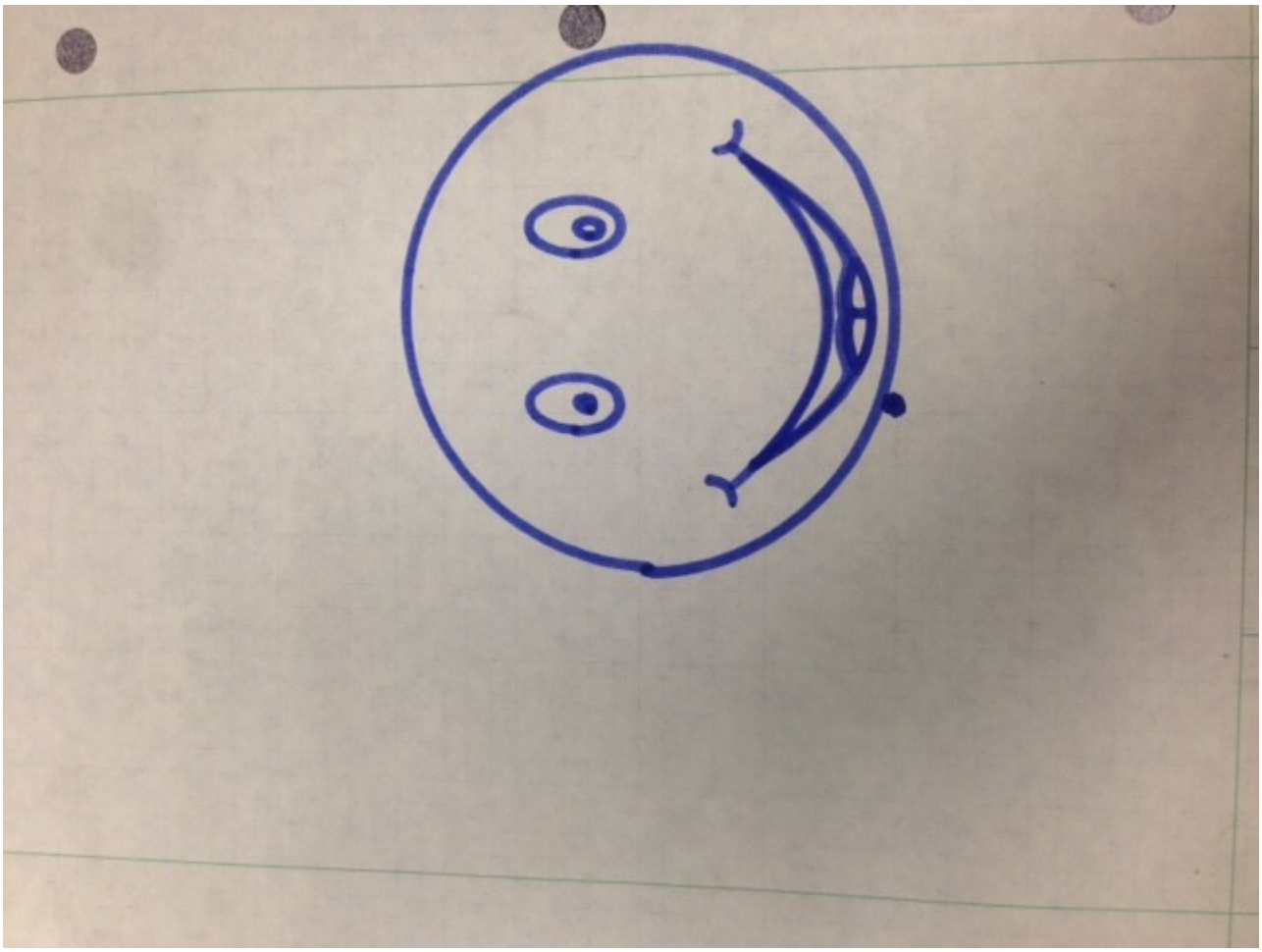
4.4 Examples of GUIs created using GTK+





5 Photos of Progress

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.



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.

6 Bill of Materials (BOM)

Item	Quantity	Part Number	Mfg	Price	Description
CNC Machine					
X-Stepper Motor					
Y-Stepper Motor					
Z-Stepper Motor					
PCIe DAQ	1	826	Sensoray	$677	
Limit Switches	9				
PCI I/O Card		5I20	Mesa Electronics		
Servo Interface Card		7i33			
Isolated I/O Card	2	7i37			
Temperature Sensor					
Current Sensor					
MIG Welder			Chicago Electric		
Motor Controller					