Theory of Operations Manual

ESI MD100 Spindle Drill

WCP04
Wyatt Lendle
Ian Doughty
Jared Snapp
Lu Yang
Alexander Marcus

Table of Contents

ESI MD100 Spindle Drill: Function in Industry

Controller Chassis

The controller chassis is made up of ten circuit boards that control different aspects of the Spindle Drill. When this Spindle Drill was used in industry, it had to be capable of incredibly precise drilling of very small holes on printed circuit boards. The controller provided the correct amount of voltage to each component of the spindle drill to achieve this precision. The controller also takes feedback from many drill components in order to know where the drill is and where it needs to go next. How the controller interacts with each component of the Spindle Drill will be outlined in each section of the Theory of Operations Manual A: ESI MD100 Spindle Drill Function in Industry. The controller chassis is shown in the picture below:

Figure 1: Controller Chassis

In order to control the different components and functions of the drill each panel has a different circuit board each with the sole function of communicating inputs from the computer to the components of the axis. A close up of one of the circuit boards is shown below:

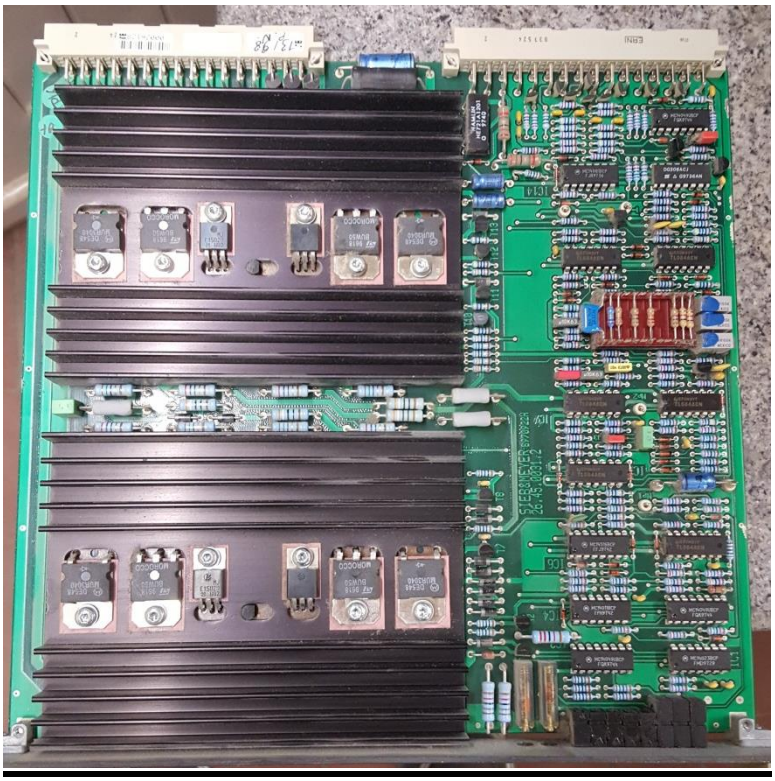

Figure 2: The X axis controller circuit

X-Y Direction Motion

The X-Y movement of the drill is mainly controlled by three different components on the drill:

- DC Motors
- Incremental Linear Encoders
- Limit Switches

DC Motors

The movement of the drill is driven by two Parvex RS430F DC motors on both the X and Y axes of the drill. These motors require 43V of voltage and 8.1A of current to run at operating speed (2000 rpm). In order for the system to use these motors to move the drill, the rotational motion of the motors must be converted into linear motion. To accomplish this, there is a long lead screw attached to each motor. When the motor is running, it spins the lead screw which is connected to the drill head effectively moving the spindle drill to the desired location.

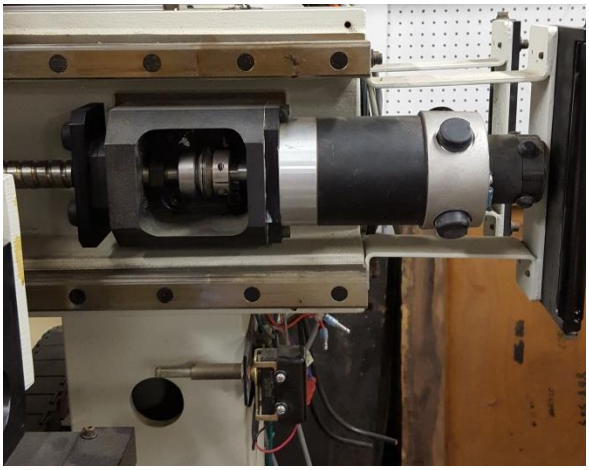

Figure 3: X Axis motor

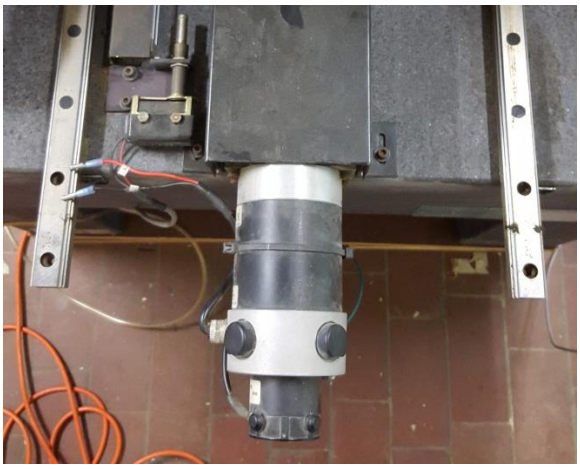

Figure 4: Y Axis Motor

X axis Assembly

Part	Part Information
Servo Motor	Drives the lead screw. The servo motor is position to allow air to circulate freely around the motor for efficient cooling
Tachometer	Attached to the rear of the z-axis servo motor, it monitors the speed of the servo motor
Cable Carrier	Contains and protects the wiring and pneumatic tubing that run to the components mounted on the x-axis assembly
Coupling	Connects the servo motor adapter to the x-axis lead screw
Lead Screw	The x-axis lead screw drives the x-axis table. The end of the lead screw is supported by ball bearing.
Ball Nut	Is mounted around the lead screw and supports the ball bearings that roll over the lead screw when the x-axis is in motion
Free End Bearing Mount	Provides support for the lead screw
Limit Switch	Positioned at either end of the x-axis track, when there is a signal from the limit switch the control responds by stopping carriage movement beyond the programmed travel limits
Limit Switch Flags	Activate the two limit switches when the position of the x-axis table exceeds the programmed travel limits
Hard-stop Bumpers and Mounts	Positioned at either end of the lead screw physically limit the travel of the x-axis able
Linear Guide Rails Scale Head Bracket	Guide bearings mounted to the bottom of the x-axis table move along the linear guide rails at the top and bottom of x-axis assembly
Scale Head Bracket	Holds the scale head in place

Table 1: List of X Axis Parts

Y-axis Assembly

Part	Part Information
Y-axis Carriage Subassembly	Is mounted on two linear rails, and travels on linear rolling bearings. The y-xis is positioned by a ball bearings
Lead Screw and Ball Nut	The end of the lead scre is supported by ball bearings. The ball nut is mounted around the lead screw and holds the ball bearings that roll over the lead screw when the y-axis is in motion
Lead Screw Cover	The lead screw cover houses the lead screw and prevents debris contamination
Servo Motor	Drives the lead screw. The servo motor is position to allow air to circulate freely around the motor for efficient cooling
Tachometer	Attached to the rear of the z-axis servo motor, it monitors the speed of the servo motor
Linear Scale	Provides position feedback to the controller for the y-axis
Limit Switches	Positioned at either end of the y-axis track, when there is a signal from the limit switch the control responds by stopping carriage movement beyond the programmed travel limits
Hard Stops Bearings	Two rubber hard stops on the y-axis bearing mounts provide protection against mechanical damage if there drill travels too far
Linear Rails	Located on each side of the table, allow for the y-axis assembly move back and forth
Coupling	Allows for alignment errors between the y-axis lead screw and the servo motor adapter
Cable Carrier	The cable carrier contains and protects the wiring and pneumatic tubing
Guide	Move along the linear rails and guide the y-axis table

Table 2: List of Y Axis Parts

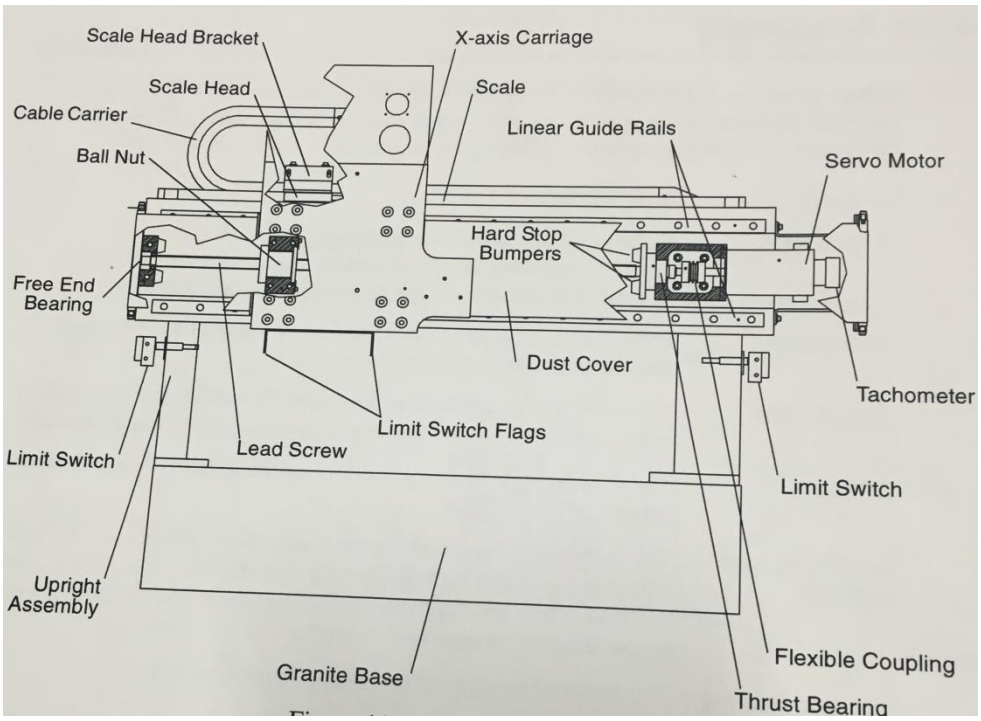

Figure 5: X Axis Assembly

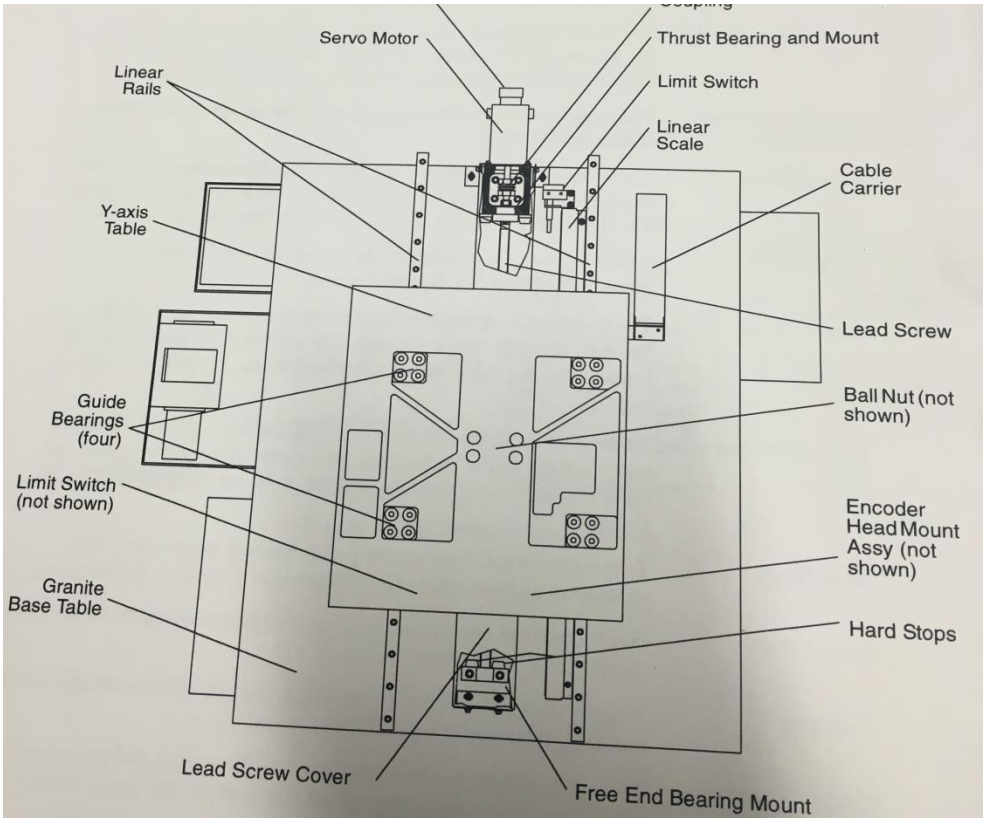

Figure 6: Y Axis Assembly

Incremental Linear Encoders

RFS Elektronik MSA 6706 The incremental linear encoders are used to determine where the drill is along the X and Y axes. When the drill moves in these directions a small component, called the readhead, that is mounted on the drill head moves with the drill. The readhead interacts with the linear scale that is mounted on the body of the drill. The readhead scans the scale and, depending on what direction the drill is moving, increments or decrements the position counter. The encoder that is used on the drill is able to convert the position into an analog signal that is sent to the drill controller to establish a position of the drill on the X-Y axes.

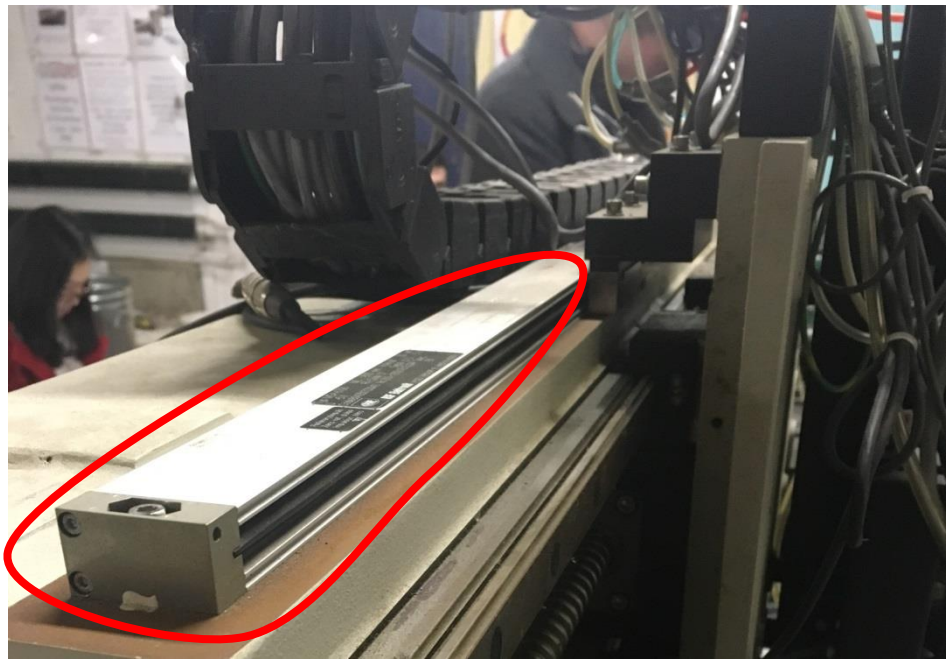

Figure 7: Linear Encoder showing linear scale and readhead

Limit Switches

The final components of the X-Y motion of the spindle drill are the four limit switches, two for both the X and Y directions. The limit switches stop the drill from moving when they are activated. The process is fairly simple; while the drill is moving, if the limit switches are activated a circuit is completed. This sends a signal to the controller which tells the drill to stop moving in that direction. Then it sends the drill back in the other direction in a small increment. These switches are mainly used for if the positioning of the drill is off for some reason. Friction or other outside influences could have altered the position of the drill and the switches stop the drill from running off of the tracks.

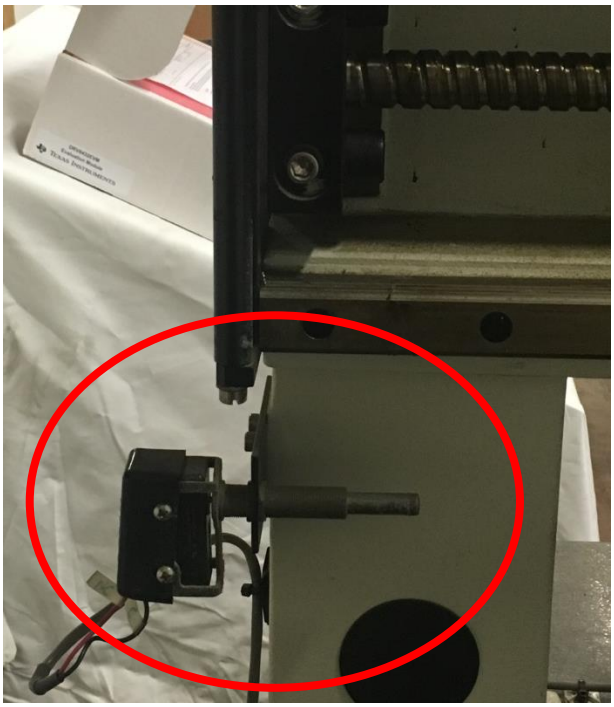

Figure 8: One of the X direction limit switches

Z -Axis

Function

In order to make the holes in the PCB's the spindle drill moves up and down in the z direction, allowing the spindle to cut through the boards as it is spinning. The motion of the z direction is controlled differently compared to the x and y directions. As mentioned in the earlier section, the x and y directions are controlled by DC servo motors while the motion in the z direction is generated by the linear motor actuator. This component is the focal instrument in the drills ability to move up and down when needed to form new holes. The way the linear motion actuator moves the drill vertically is by utilizing a permanent magnetic field and a coil winding, which acts as a conductor, to produce a force which is proportional to the current that is applied to the coil. So as current enters the z-axis from a power source, it uses this current to push the drill down. The electromechanical conversion (current and magnetic field causing motion), operates under the Lorentz Force Principle. This principle states that when a conductor which has a current is placed within an electric and magnetic field and force will act upon it. The Lorentz Force equation is shown below:

$$F=kBLIN$$

Where F is the force, k is a given constant, B is the magnetic flux density, I is the current running through the coil, L is the length of conductor and N is the number of conductors.

The design of the linear voice actuator is that is a tubular coil of wire that is within a magnetic field. The magnetic field is produced by magnets on the inside of the ferromagnetic cylinder. There is also an inner core of ferromagnetic material that is placed along the center of the coil. When the current flows through the coil there is a force that is generated onto the coil which provides the means of the motion between the coil and the cylinder. When voltage is applied across the two coil leads inside the cylinder, a current is then generated in the coil allowing the coil to move along the air gap. This air gap is provided by the pneumatic system which pushes air inside the cylinder which allows the necessary space for the coil to move. The direction that the drill moves is dictated by the direction of the current flow in the wire. The figure below shows the design and the main components of the liner voice actuator:

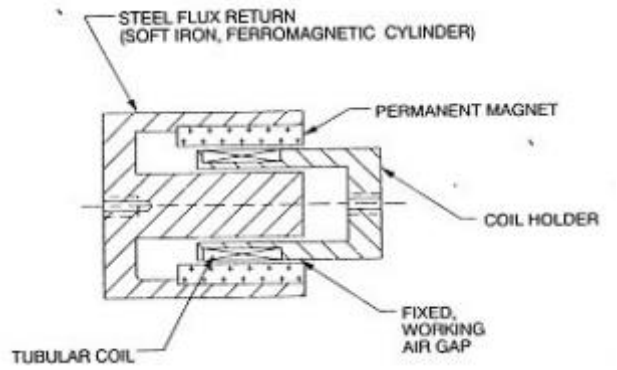

Figure 9: Layout of the linear motor actuator

In order for the spindle drill to be as accurate as it needs to be, there are multiple functions that sense certain elements of the drills functions and motion. While the drill is in operation it is constantly keeping track of the z-axis's position, velocity and temperature of certain parts. It also needs to sense if the drill bit is either broken or not connected properly. The drill is constantly tracking this information in order to ensure that drill is in the exact position that it needs to be at all times. Another function of the z-axis is to vacuum the cooper debris that accumulates once a hole is made. All of these functions play a crucial role in the drill's overall function, and have to be maintained frequently to ensure accuracy.

Components

The z-axis assembly is composed of twenty-five different parts these parts all have a crucial role in the ability for the z-axis to function properly. The table below provides information about all the parts in the z-axis of the drill.

Part Name	Part Information
Linear Velocity Transducer	Used to compare velocity of z-axis to the commanded velocity
Limit Switches and Flag	Used in order to stop the movement of the z-axis if the drill moves in one direction too far. They are electromechanically devises that consists of an actuator that is mechanically linked to a set of contacts. When the drill hits the actuator the device operates the contacts to break an electrical connection
Mechanical Stop	Determines the motion limit of the z-axis moving downward. The limit is when the bracket reaches the rubber bumper
Fixed Side Spindle Guide Bushings	Preloaded ball bushings which support the right spindle guide rods
Spindle Guide Rods	Spindle is attached to the guide rods which allow for smooth motion in the z-axis
Spindle Clamps	Keep the spindle guide rods a specific distance apart and parallel
Adjustable Spindle Ball Bushings	Two preloaded bull bushings that support the left spindle guide rod. They are adjustable to allow the left spindle guide rod to line up parallel to the right

Linear Optical Encoder	Senses the vertical position of the spindle. It is a sensor that is linked to a scale. The sensor reads the scale and converts position into either an analog or digital signal that converts to a digital readout. Movement is determined from changes in position with time.
Flexure	A metal rod that connects the moveable part of the linear actuator to the top of the spindle. It adjusts the height of the spindle
Return Springs	Raise spindle to a safe position when the linear actuator is off. The springs provide resistance to the linear actuator when the spindle moves downward
Motor Range of Z Linear Actuator	Generates the force which moves the spindle vertically. It is the main component of the z-axis providing movement.
Z linear Actuator Housing	Provides support for the permanent magnets in the linear actuator.
Front/Back Tilt Adjust	Changes the front to back spindle tilt when setting the spindle sweep
Spindle Housing	The base of the spindle assembly. It is mounted to the x-axis overhead carriage
Spindle Coolant Lines	The tubes that connect to the spindle motor in order to prevent overheating.
Collet Release Air Line	Opens the collet which is necessary when changing the tool bit
Manual Tool Release Button	Used when manually installing or removing the drill bit
Spindle	The rotating drill. The part which makes the holes in the PCB's
Spindle Collet	Holds the drill bit in place
Air-line for Spindle Air Bearings	Supplies spindle air bearing with clean air
Spindle Electrical Connector	Carries power to the spindle and senses the spindle speed and temperature information to the controller
Air-line	Provides adjustable pressurized air to the pressure foot air cylinders from the pneumatics system
Air Cylinders	Two air cylinders applies a clamping force through the pressure foot to the drill stock during the drilling cycle
Pressure Foot	Clamps the PCB stack ahead of the drill bit. This is crucial to ensure the board doesn't move during the drilling process Also houses the optical and air channels of the Broken Bit Detector. Also there is a vacuum hose at the back of the pressure foot to remove debris
Broken Bit Detector	Uses a through beam, fiber optics cable to detect the presence of absence of the very tip of the drill. If there is an absence that means the tip of the drill is broken, and it needs to be replaced.

Table 3: List of Y Axis Parts

The information in Table 3 describes the twenty-five different parts that make up the z-axis. The Figure below is a diagram of the entire z-axis pointing out the location of each part on the drill.

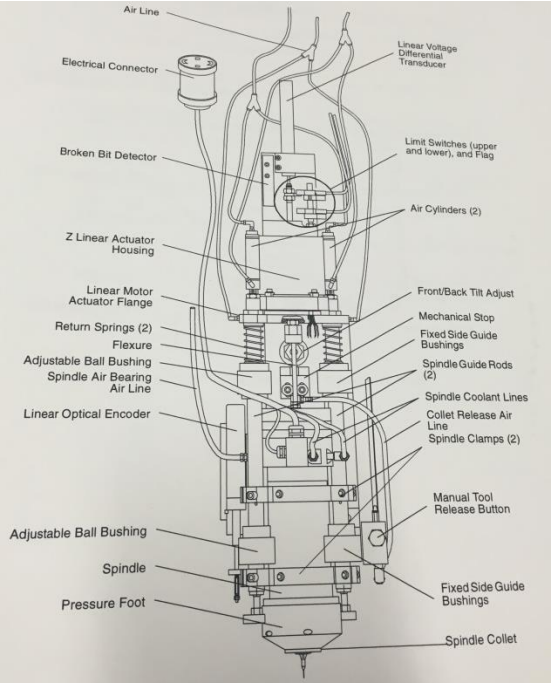

Figure 10: Z Axis Assembly

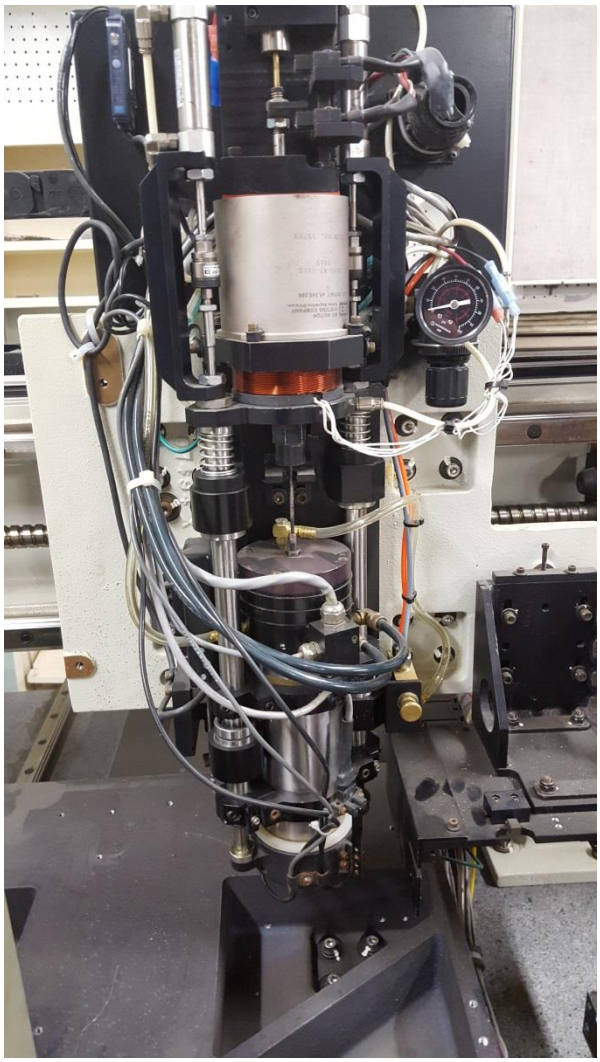

Figure 11: Picture of the Z Axis

Spindle

Gripper

The gripper is the part of the Spindle Drill that changes the drill bit. Pneumatics play a large role in how the gripper switches drill bits. The pneumatic panel provides air to the gripper, which allows the gripper jaws to open and close and allows the gripper to move vertically. When the Broken Bit Detector detects a broken drill bit or the printed circuit board requires a different sized hole, the Spindle Drill will pause drilling and move to the gripper to change the bit.

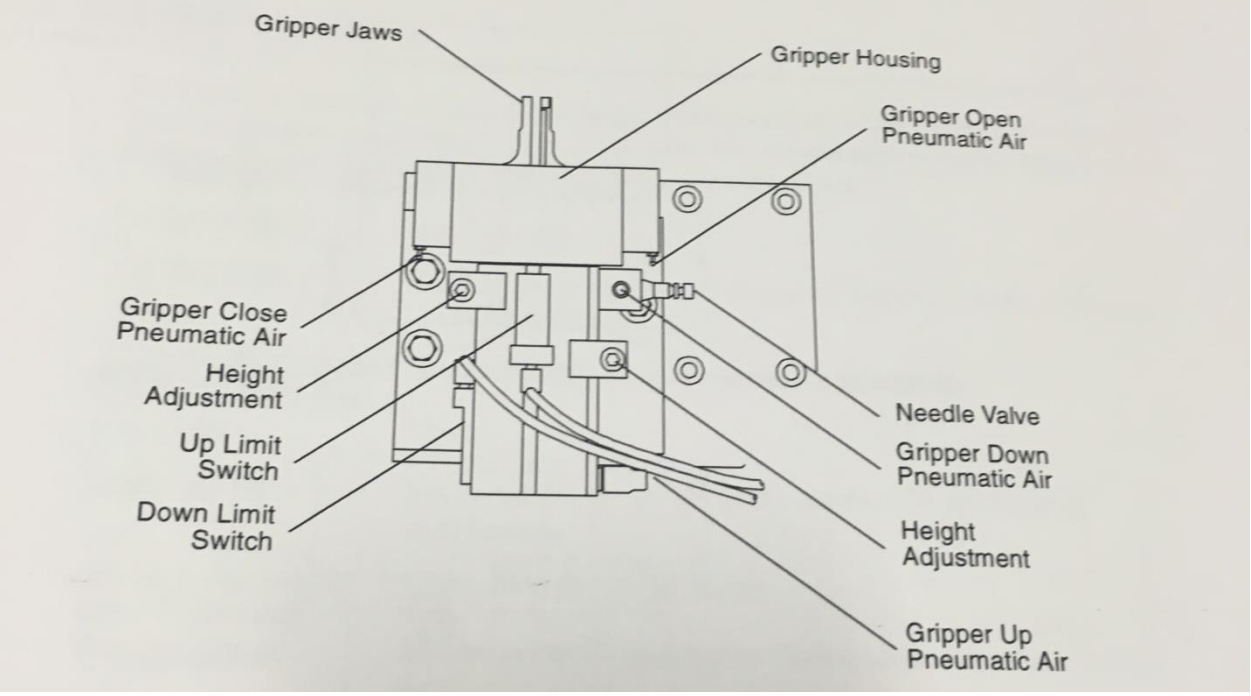

Figure 12: Gripper Assembly

The table below provides information of the eleven parts which compose the gripper:

Part	Part Information
Gripper Jaws	Holds the tool bit
Gripper Housing	Encloses the gripper, completely surrounding it for protection
Gripper Open Pneumatic Air	Opens the Gripper Jaws in order to release the drill bit when the a different drill bit is needed
Needle Valve	Adjusts the velocity for the vertical movement of the gripper
Gripper Down Pneumatic air	During a tool bit change this moves the gripper jaws down to pick up a new bit
Height Adjustment	Adjust the height of the gripper
Gripper Up Pneumatic Air	During a tool bit change this moves the gripper jaws up in order to replace the current drill bit
Down Limit Switch	Provides feedback to the control that the drill has reached the endpoint of the z-axis track
Up Limit Switch	Provides feedback to the control that the drill has reached the endpoint of the z-axis track
Gripper Close Pneumatic Air	Closes the jaws to hold the tool

Table 4: List of Gripper Parts

Cooling and Ventilation

This Spindle Drill was designed to drill incredibly precise holes in PCBs. In order to attain precision, the spindle must spin at very high speeds which generate a large amount of heat. To prevent damage to integral drill components, a cooling system was installed to dissipate some of the heat. The cooling system involves a pump, a heat exchanger, a pressure gauge and a reservoir along with many coolant lines. The working coolant is a mixture of glycol and water and is stored in the reservoir. Then the mixture is pumped to the spindle where the heat generated by the spindle is absorbed. When the hot coolant returns from the spindle, it is outed to a heat exchanger where the excess heat is removed conventionally by a fan and then the cool coolant enters back into the reservoir.

The entire cooling and ventilation system is composed of 10 different parts, each having a specific function to ensure that the drill is not damaged from overheating. The names and a brief description of the parts are listed in the table below:

Part	Part Information
Pump	Circulates the coolant from the reservoir to the spindle. It is the driving force behind the entire system.
Strainer	Filters out any particles in the coolant
Pressure Gauge	Measure the pressure of the coolant. The coolant needs to be in the range of 20 to 40 psi
Coolant Line (to spindle)	A tube that carries the coolant to the spindle
Coolant Return Line (from spindle)	This return line carries the coolant from the spindle back to the heat exchanger
Coolant Line (to reservoir)	This line carries the coolant from the heat exchanger to the reservoir
Water Pump Switch	This switch operates the water pump
Heat Exchanger	Removes heat from the coolant after the coolant has went to the spindle
Coolant Line (from reservoir to pump Reservoir)	This line carries coolant from the reservoir to the pump. The reservoir stores the coolant after it has moved through the heat exchanger and before it is pumped to the spindle
Coolant Line (from reservoir to pump Coolant Return line	This line carries the coolant from the heat exchanger

Table 5: List of Cooling and Ventilation Parts

The diagram below points out these different parts in relation the cooling and ventilation system as a whole.

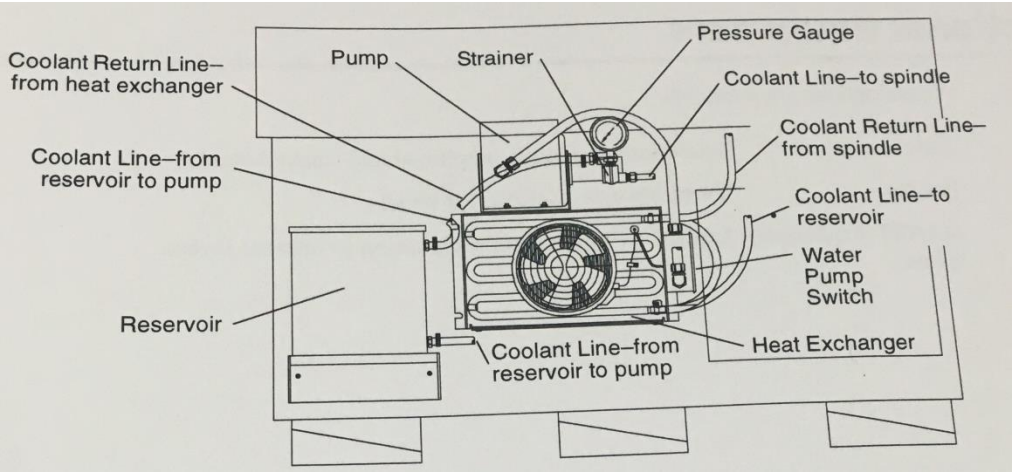

Figure 13: Layout of the cooling and ventilation system

Pneumatic System

Pneumatics Panel

The pneumatic system of the ESI MD100 Spindle Drill controls and aids in many functions of the drill. All of the pneumatics are regulated using the pneumatic panel which contains many valves and gauges. The panel also contains an I/O pressure switch which controls the electrical interface between the controller and the pressure switches on the panel. This panel allows the controller to send air at various pressures to five different parts of the Spindle Drill. The five functions of the drill pneumatics and how much pressure each one needs are as follows:

Function	Desired Air Pressure Range (psi)
Pressure Foot Soft	19-25
Pressure Foot	50-58
Gripper Up/Down	---
Broken Bit Detector (BBD) Air	11-15
Motor Cooling	19-29

Table 6: Functions and Desired Pressure Ranges of the Spindle Drill Pneumatic System

The pneumatic panel that controls the pneumatic system which helps distribute air to the machine is composed of 4 different parts. The table below describes each part and gives some information about the part.

Part	**Part Information**
I/O Pressure Switch	Controls the electrical interface between the controller and the pressure switches on the back of the panel
Pneumatic Valve Stack	Controls the pneumatic air to various components within the machine
Pressure Controls	These knobs are used to set the pressure to a desired value for specific components
Pressure Gauges	Gauges operate a specific air driven component, these components are listed in the previous table

Table 7: List of Pneumatic Panel Parts

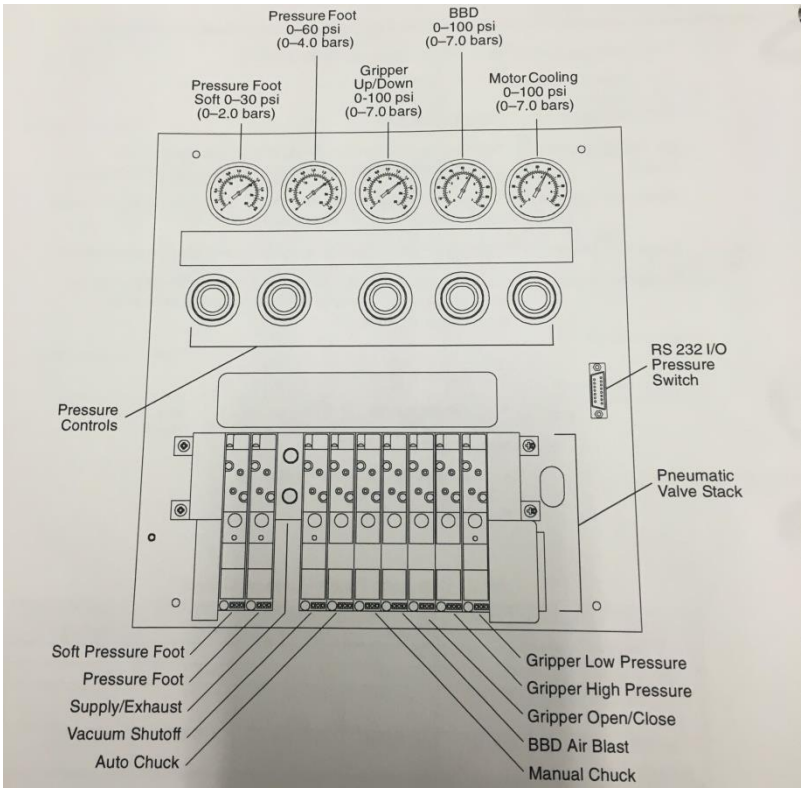

Figure 14: Layout of the Pneumatic Panel

Figure 15: Front of the Pneumatic Panel

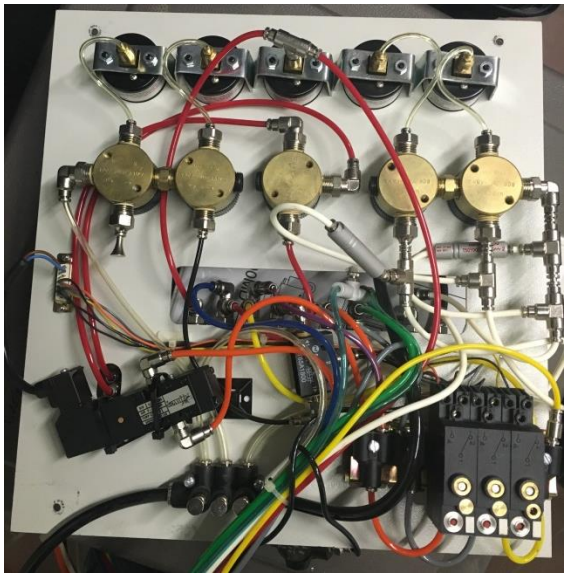

Figure 16: Back of the Pneumatic Panel

Air Filter Regulator

The air filter regulator is located on the front lower left of the machine. The purpose of this part is to provide airflow to the spindle, pneumatic panel and manifold. The air filter regulator is a key component of the pneumatic system ensuring that the air is kept free from containments and that the air is delivered to the necessary parts of the machine. It is composed of eight different components which are listed in the table below along with a brief description of the component.

Part	Part Information
Air Regulator	Regulates the pressure of output air and is set at 90 to 100 psi
Lockable Manual Shut Off Lever	Lockable shut off for the machine air supply
Air Supply Inlet Connection	Measures the air pressure with a range of 0 to 160 psi
Water and Particulate Bowl Filter	Holds the input air filter
Automatic Drains	Automatically drains the fluid from the filters that has accumulated during the drilling cycle
Coalescing Oil Filter Bowl	Holds the coalescing filter in place
Low Pressure Switch	Shuts off power to the servo motors when activated, will also notified the operator with an alarm that it has been activated
Air Supply outlets	Supply air to the spndle manifold and pneumatic panel
Lock Out Valve	Stops air from to the machine, used when doing maintence
Coalescer P Indicator	Indicates when the filter is clogged or not. When green the filter is not clogged and is usable. When red there is some sort of blockage which needs to be removed

Table 8: List of the Air Filter Regulator Parts

Drill Operation

In its original industry setting the MD100 Spindle drill setup looked much different than it currently does. Figure _ below shows the drill as it would have been when it was fully operational.

Figure 17: Spindle Drill with Housing and Control Monitor

The casing that surrounded the drill served several purposes: Holding the different components in a central location, preventing damages to the different components of the machine by limiting exposure to the outside, and preventing injury of the employees by enclosing the moving parts of the drill and providing an emergency stop switch. The operation of the drill was also different than it is now. When making holes in printed circuit boards the spindle drill would operate automatically on a closed loop system. This operation was capable because of the computer that the machine was directly connected to.

Computer Interface

The computer interface allows the CNC machinist or operator to input expressions that result in movement in the drill. To design the PCB the board is originally drawn up in design software, for example Auto CAD. From then the drawing in AutoCAD is converted to G-Code from a Computer Aided Manufacturing. Once it is in G-code, which will be discussed in the next section, the CNC machine is able to make the board. The

AutoCAD process of the design is done before while the CAM process is done on the computer that is directly attached to the machine. This part is crucial because it takes the design into a language that could be understood by the controller chassis. It is possible to also not use the CAM process when making a PCB. A full understanding of G-code would allow the machinist to enter in the exact and full commands that are necessary to make the PCB. Besides the computer the machinist can also interact with the machine through an emergency stop button, which is on the outside of the casing and is used in an emergency to completely stop movement of the drill.

G- code

The way the drill operated in the close loop system was by taking inputs from the computer and these inputs would correspond to a certain amount of holes of a specific size, in specific positions. The inputs that would be entered in the computer were in the form of G-code. The G-code allows the computer to give commands the drill providing all the necessary information for the drill to make all the holes in a PCB. The layout of the PCB would be given before in a schematic, and the CNC programmer would enter in the G-code that would allow the drill to make these holes. The code is the instructions that tell the drill where to move, how fast to move, and what path to follow. These components are all very important and need to be very precise to ensure that the board is cut in the right places. The way the code works is it begins with a letter followed by two numbers, and this would serve as the initial input. The letter could be any in the alphabet, and each letter corresponds to a different command. However G is the most common letter that is used in the code because it controls the position of the dill, which is the reason the code is referred to as G-code. The different letters correspond to many different functions because of the many capabilities of the drill. These functions are referenced in Appendix B, and describe which each input controls. Also specifically for the G input there are many different codes that are commonly used to control the motion of the drill which are listed in Appendix C.

ESI MD100 Spindle Drill: New Controller Design

Controller Hardware

Power/ Emergency Stop

The power to the X-Y motors is supplied by an external 12V power supply since the motors only require 12V at 3.5 amps of typical current. The mains power will first run through an emergency shutoff switch, which enables the power to be shut off entirely in case of equipment failure or danger to people. Next, it will run to the power supply, which will convert the AC power from the outlet to DC power. A 12V power supply will be all that is needed to power the low power components of the controller as well as supply voltage to the motors. The Arduino, H-bridge, and motors will all be powered by this power supply.

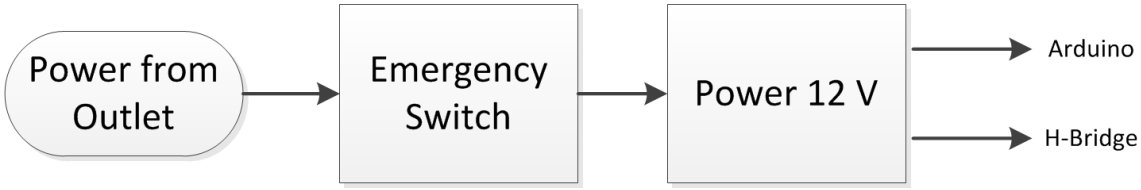

Figure 18: Emergency Switch and Power Schematic

Arduino

The “brain” of this controller design is the Arduino micro controller. An ATMega328p chip made by Atmel powers the Arduino. An Arduino was chosen over a regular microcontroller because Arduino has a plethora of online libraries and help files which make the coding and debugging of the code much easier. The Arduino also has a built in power supply regulator along with other useful features that we would otherwise have to design ourselves. Since the motors are not running and full speed, the 16Mhz clock speed on the Arduino is more than adequate to control the motors. The pin layout for the Arduino is displayed in Figure 19.

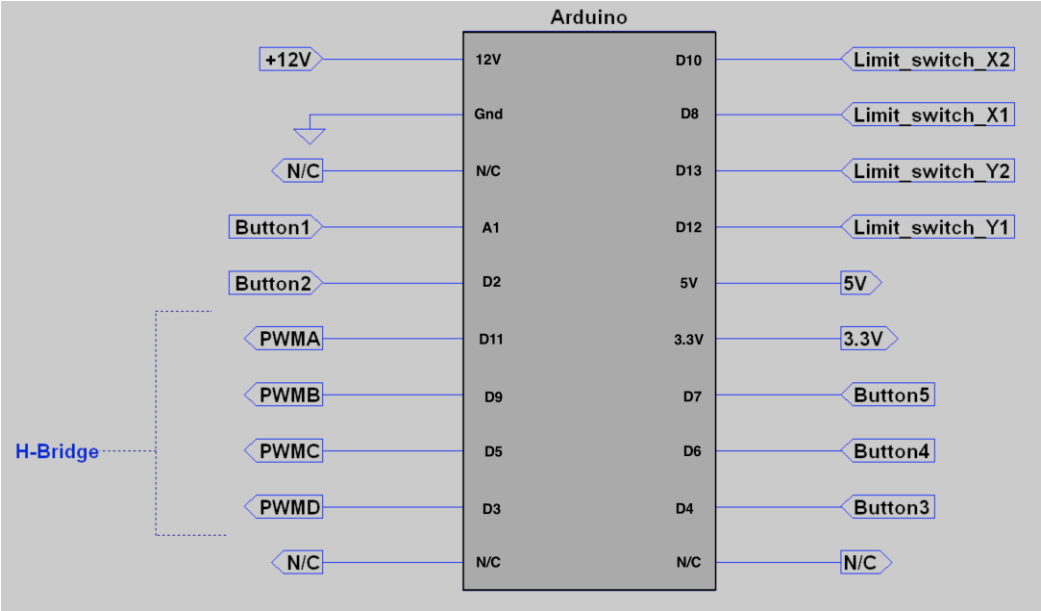

Figure 19: Arduino Connections Layout

Pins D5, D9, D11, and D3 are for controlling the h-bridges. D5 and D11 will be the actual pulse modulated signal that will control the speed and direction of the motors. The four buttons (A1, D2, D4, D6) will be part of the user interface, and will allow the user to control the direction in which the motor moves. D7 will allow the user to alternate between 2 duty cycles, a 98% duty cycle that makes the drill run near top speed and a 50% duty cycle which will make the drill run at half speed. The Arduino will receive feedback from the sensors on the drill on pins D12, D13, D8, and D10.

H-Bridge

In order to control the DC motors, an H-bridge is also needed to pulse modulate the signals sent from the Arduino. An H-bridge is a chip comprised of switches used to control motors. The H-bridge we will be using will take a 3.3-volt signal from the Arduino to switch on the 12 volts, from the power supply, to the motors. This signal can then be pulse modulated at 500-1000 Hz to adjust the speed at which the motors move.

8.2.1 Full Bridge Mode Operation

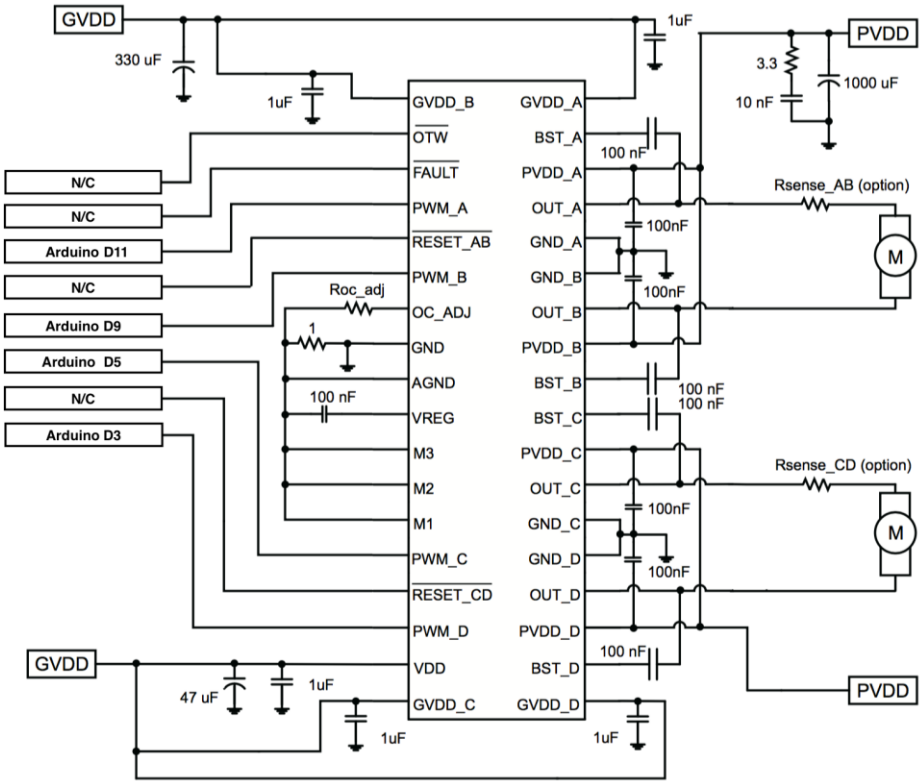

Figure 8. Application Diagram Example for Full Bridge Mode Operation Schematic

Figure 20: DRV8432 H-Bridge (Full Bridge Mode Configuration) Schematic

The chip used is a Texas Instruments DRV8432 which consists of two H-bridges, both being able to operate at a maximum constant current of 7 amps. Since this design, will only be operating at 12 volts, the current will never exceed 4 amps. Also by only moving 1 motor at a time via the full bridge mode configuration, we are never in danger of exceeding the 7-amp threshold.

Prototype User Interface and Controller Software

Software

Appendix C shows the code used to control the Arduino. The first ~15 lines of code initializes which pins correspond to which drill functions. Then the pins are set to either input or output and all of the pins are set to low (or 0). Then the code enters and endless while loop, which allows the program to run continuously. After the first while loop, there are 5 parts of the code, which correspond to each button on the user interface. The first part sets the duty cycle of the PWM signals while the other four control the directional motion of the drill. The drill can move continuously while any directional button is held down. There is also a significant amount of delays in the code. This allows the buttons on the user interface to re-initialize to 0 or low. If these delays were not present, the Arduino may read incorrect values from the buttons, which would result in an inaccurate and somewhat erratic controller.

User Interface

The user interface that was designed and implemented is a prototype and was designed only to demonstrate the functionality of the Spindle Drill. The UI consists of five buttons soldered onto a proto-board. The board consists of four directional buttons along with a fifth button that controls the duty cycle of the PWM signal. While any single directional button is held down, the drill will move in that direction until the button is released, stopping the drill, or until a limit switch is activate, moving the drill back a predetermined length. The amount the drill moves back after hitting a limit switch can be changed simply by changing a couple of lines in the Arduino code.

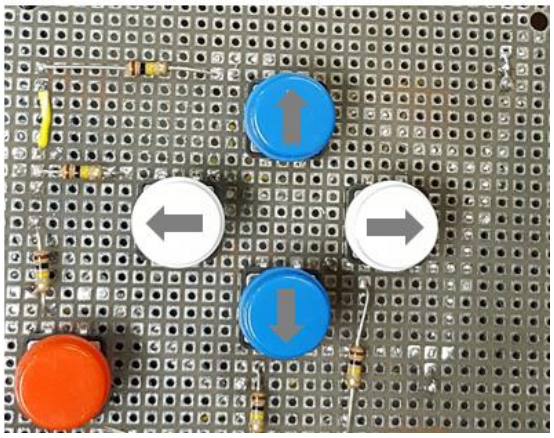

Figure 21: User Interface

Interaction of Controller Components

Our design includes four main components; button controller, Arduino, H-bridge, and power supply. These parts interact with each other and also with the x & y motors and the four limit switches on the drill. The basic layout begins with an input from the user in the form of pressing one of the five buttons on the controller pad. This sends a signal to the Arduino which takes power from the power supply and sends a PWM to the H-bridge. The H-bridge also is connected to the power supply and selects which motor to send the PWM signal to. The H-bridge allows the Arduino to control the high power coming from the power supply as the Arduino cannot interface with such high voltages and currents directly. The motors move when the PWM is sent to them. An external power supply provides power for the motors which run at 12 volts and 6 amps. When the motors hit the limit switch a signal is sent to the Arduino to send the motor in the opposite direction. This interaction is shown in the Figure 22.

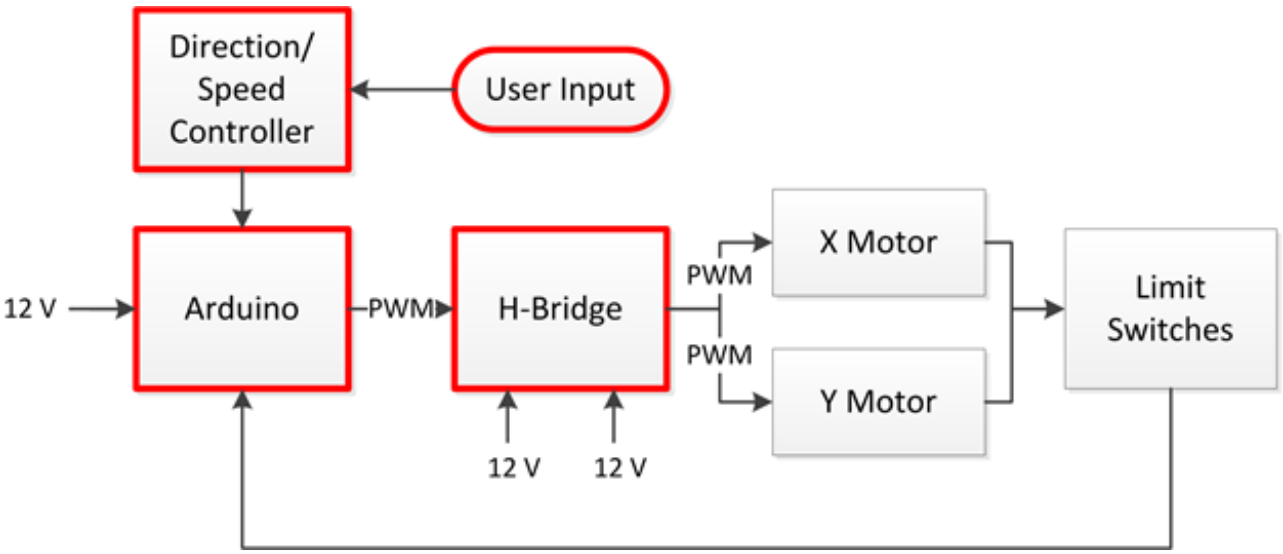

Figure 22: Proof of Concept Controller

The 5-button controller is the equipment that the user needs in order to move the drill. If the user wants to move the drill in a specific direction they will press one of the blue or white buttons. Each of these buttons is connected directly to a pin on the Arduino. The pins being utilized for the four different directions are A1, D2, D4 and D6. The user interface layout is shown in Figure 23.

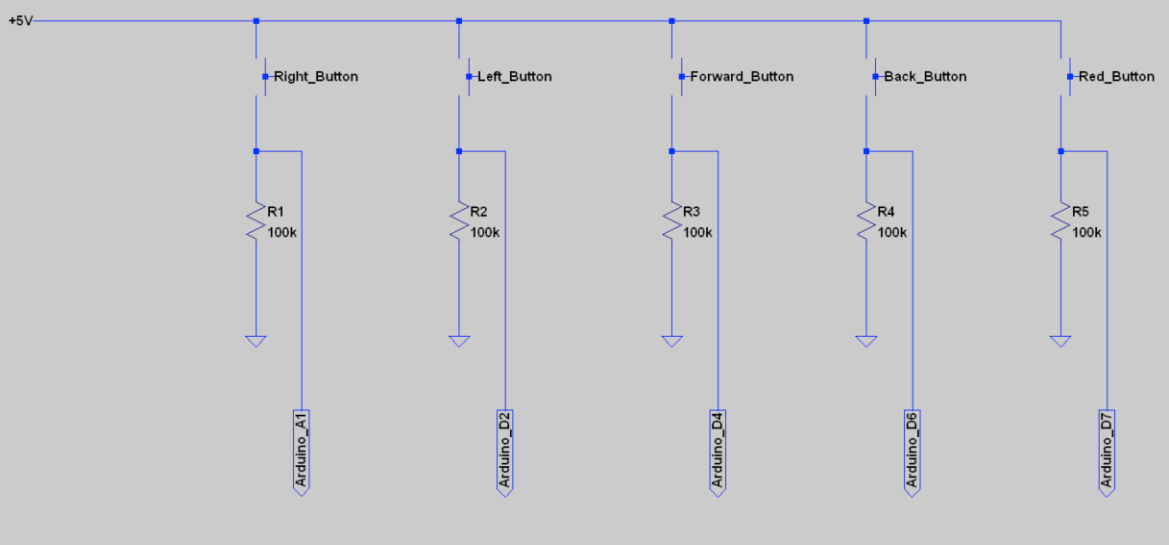

Figure 23: User Interface Layout

The button to change between the two different speeds is D7. In the code each of these buttons are set at an initial value of zero. When the button is pressed the value given to the button is changed to 1. This begins the while loop that controls the movement of the motors. The while loop in the code begins because it is constantly checking if any of the buttons have been changed to a value of 1. The while loop continues to run until the button is no longer being held down. While the button is being held down the Arduino sends the 3.3-volt signal to the H-bridge which will use this signal to switch on the 12 volts from the power supply to the motors. This signal can be pulse modulated at up to 500 Hz to adjust the speed at which the motors move. The H-bridge is directly wired to the Arduino, connected at pins D11, D5, D9 and D3. These are for the four different PWM's. Once the user is no longer pressing the button, the while loop is stopped. Then there is a delay in the code to allow for a transition between movements. Then the code will go back to checking if any of the buttons are pressed and begin the process all over again. The delay is very minimal and will not be noticed by the users; to them it will seem like a smooth transition. The same process occurs for each of the 4 buttons, with the same interactions between the code, Arduino, H-bridge, power supply and motors.

To change speeds the initial sequence is similar to move in any direction, the code includes while loop checking for a change in the button value in the pin that the specific button is connected to. When the change speed button is pressed, the length PWM is changed and the duty cycle will switch between 50% and 98%, alternating between low and high speeds. Once the change of the pulse modulated signal I is changed the 4-way directional pad acts the same as before.

When the controller is turned on, the limit switches are continuously on. Once the drill hits them then they are turned off. Their given value in the code goes from HIGH to LOW. While the controller is high the code is also running a while loop checking for a

value change in the limit switches. Once the right limit switch is hit, the code runs a delay function, and then runs the function to move the motor left. This action prevents the drill from moving dangerously past the limit switches. The code includes a while loop for each limit switch which are constantly being checked. The limit switches are also connected to the Arduino using pins D12, D13, D8, and D10. And the schematic for their layout is shown in Figure 24.

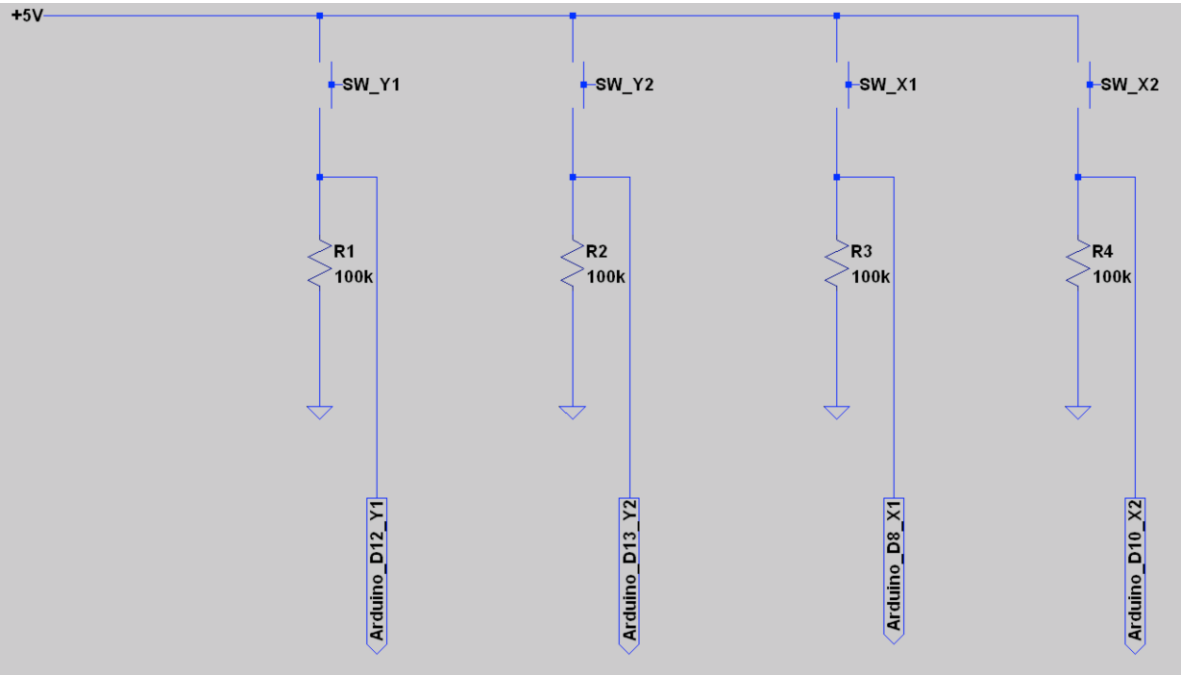

Figure 24: Limit Switches Schematic

In the code each pin being used is initially given a label. The buttons for the controller pad, limit switches and motor pins are then initialized.

Appendix A: G Code Inputs

Variable	Description	Corollary info
A	Absolute or incremental position of A axis (rotational axis around X axis)	
B	Absolute or incremental position of B axis (rotational axis around Y axis)	
C	Absolute or incremental position of C axis (rotational axis around Z axis)	
D	Defines diameter or radial offset used for cutter compensation. D is used for depth of cut on lathes. It is used for aperture selection and commands on photoplotters.	
E	Precision feedrate for	

	threading on lathes	
F	Defines feed rate	Common units are distance per time for mills (inches per minute, IPM, or millimeters per minute, mm/min) and distance per revolution for lathes (inches per revolution, IPR, or millimeters per revolution, mm/rev)
G	Address for preparatory commands	G commands often tell the control what kind of motion is wanted (e.g., rapid positioning, linear feed, circular feed, fixed cycle) or what offset value to use.
H	Defines tool length offset; Incremental axis corresponding to C axis (e.g., on a turn-mill)	
I	Defines arc center in X axis for G02 or G03 arc commands. Also used as a parameter within some fixed cycles.	
J	Defines arc center in Y axis for G02 or G03 arc commands. Also used as a parameter within	

	some fixed cycles.	
K	Defines arc center in Z axis for G02 or G03 arc commands. Also used as a parameter within some fixed cycles, equal to L address.	
L	Fixed cycle loop count; Specification of what register to edit using G10	*Fixed cycle loop count:* Defines number of repetitions ("loops") of a fixed cycle at *each* position. Assumed to be 1 unless programmed with another integer. Sometimes the K address is used instead of L. With incremental positioning (G91), a series of equally spaced holes can be programmed as a loop rather than as individual positions. *G10 use:* Specification of what register to edit (work offsets, tool radius offsets, tool length offsets, etc.).
M	Miscellaneous function	Action code, auxiliary command; descriptions vary. Many M-codes call for machine functions, which is why people often say that the "M" stands for "machine", although it was not intended to.
N	Line (block) number in program; System parameter number to be changed using G10	*Line (block) numbers:* Optional, so often omitted. Necessary for certain tasks, such as M99 P address (to tell the control which block of the program to return to if not the default one) or GoTo statements (if the control supports those). N numbering need not increment by 1 (for example, it can increment by 10, 20, or 1000) and can be used on

		every block or only in certain spots throughout a program. *System parameter number:* G10 allows changing of system parameters under program control.
O	Program name	For example, O4501. For many years it was common for CNC control displays to use slashed zero glyphs to ensure effortless distinction of letter "O" from digit "0". Today's GUI controls often have a choice of fonts, like a PC does.
P	Serves as parameter address for various G and M codes	• With G04, defines dwell time value. • Also serves as a parameter in some canned cycles, representing dwell times or other variables. • Also used in the calling and termination of subprograms. (With M98, it specifies which subprogram to call; with M99, it specifies which block number of the main program to return to.)
Q	Peck increment in canned cycles	For example, G73, G83 (peck drilling cycles)
R	Defines size of arc radius, or defines retract height in milling canned cycles	For radii, not all controls support the R address for G02 and G03, in which case IJK vectors are used. For retract height, the "R level", as it's called, is returned to if G99 is programmed.
S	Defines speed, either spindle speed or surface speed depending on mode	Data type = integer. In G97 mode (which is usually the default), an integer after S is interpreted as a number of rev/min (rpm). In G96 mode (CSS), an integer after S is interpreted as surface speed—sfm (G20) or m/min (G21). See also Speeds and feeds. On multifunction (turn-mill or mill-turn) machines, which spindle gets the input (main spindle or subspindles) is determined by other M codes.

T	Tool selection	To understand how the T address works and how it interacts (or not) with M06, one must study the various methods, such as lathe turret programming, ATC fixed tool selection, ATC random memory tool selection, the concept of "next tool waiting", and empty tools. Programming on any particular machine tool requires knowing which method that machine uses. Ways of obtaining this training are mentioned in the comments forM06.
U	Incremental axis corresponding to X axis (typically only lathe group A controls) Also defines dwell time on some machines (instead of "P" or "X").	In these controls, X and U obviate G90 and G91, respectively. On these lathes, G90 is instead a fixed cycle address for roughing.
V	Incremental axis corresponding to Y axis	Until the 2000s, the V address was very rarely used, because most lathes that used U and W didn't have a Y-axis, so they didn't use V. (Green *et al.* 1996[5] did not even list V in their table of addresses.) That is still often the case, although the proliferation of live lathe tooling and turn-mill machining has made V address usage less rare than it used to be (Smid 2008[3] shows an example). See also G18.
W	Incremental axis corresponding to Z	In these controls, Z and W obviate G90 and G91, respectively. On these lathes, G90 is instead a fixed cycle

	axis (typically only lathe group A controls)	address for roughing.
X	Absolute or incremental position of X axis. Also defines dwell time on some machines (instead of "P" or "U").	
Y	Absolute or incremental position of Y axis	
Z	Absolute or incremental position of Z axis	The main spindle's axis of rotation often determines which axis of a machine tool is labeled as Z.

Appendix B: DC Motor Data-Sheet

DC-SERVOMOTOR

RS430F

PARVEX

8 avenue du Lac
BP249
F-21007 DIJON Cedex

Low speed torque	***1.3***	*N.m*	*Mo*
Permanent current at low speed	***8.1***	*A*	*Io*
Supply voltage with loaded motor	***43***	*V*	*U*
Definition speed	***2000***	*rpm*	*N*
Maximum supply voltage	***90***	*V*	*Umax*
Maximum speed	***5100***	*rpm*	*Nmax*
Peak current	***28***	*A*	*Imax*
*Back emf constant at 1000 rpm (25°C)**	***17.5***	*V*	*Ke*
Torque constant	***0.167***	*N.m/A*	*Kt*
Static friction torque	***5.7***	*N.cm*	*Tf*
Viscous damping for 1000 rpm	***0.94***	*N.cm*	*Kd*
Winding resistance(25°C)	***0.59***	*Ω*	*Rb*
Winding inductance	***1.33***	*mH*	*L*
Rotor inertia	***0.00031***	*kg.m²*	*J*
Thermal time constant	***11.5***	*min*	*Tth*
Motor mass	***2.8***	*kg*	*M*

All data are given in typical values under standard conditions

Appendix C: Arduino Code

```
//WCP04 Spindle Drill Controller Software
//spindle pins
  const int right_spindle_pin = 11;
  const int left_spindle_pin = 9;
  const int forward_base_pin = 5;
  const int backward_base_pin = 3;

  //user input buttons:
  const int right_button = A1;
  const int left_button = 2;
  const int forward_button = 4;
  const int back_button = 6;
  const int red_button = 7;

  //limit switches
  const int left_limit_switch = 8;
  const int right_limit_switch = 10;
  const int back_limit_switch = 12;
  const int forward_limit_switch = 13;

  int speed;


void setup() {

  //functions
  void move_spindle_left(int speed);
  void move_spindle_right(int speed);
  void move_base_backward(int speed);
  void move_base_forward(int speed);

  //initialize buttons
  pinMode(forward_button, INPUT);
  pinMode(back_button, INPUT);
  pinMode(right_button, INPUT);
  pinMode(left_button, INPUT);
  pinMode(red_button, INPUT);

  pinMode(13, OUTPUT); //onboard led

  //initialize switches
  pinMode(right_limit_switch , INPUT);
  pinMode(left_limit_switch , INPUT);
  pinMode(forward_limit_switch , INPUT);
  pinMode(back_limit_switch , INPUT);

  //initialize motor pins
  pinMode(right_spindle_pin, OUTPUT);
  pinMode(left_spindle_pin, OUTPUT);
  pinMode(forward_base_pin, OUTPUT);
  pinMode(backward_base_pin, OUTPUT);
```

```
  //set pins low
  stop_spindle();
  stop_base();


}

void loop() {


        stop_spindle();
        stop_base();
        speed = 100;

        while(1){


          stop_base();
          stop_spindle();
          delay(40);

          //speed adjust button
          if(digitalRead(red_button) == HIGH){
            if (speed > 120){
              speed = 100;
            }
            else if (speed <= 120){
              speed = 250;
            }
            delay(400);
          }



          //=============================================================

          //left button
          while(digitalRead(left_button) == HIGH){
                    if(digitalRead(left_limit_switch) == LOW){
                      stop_spindle();
                      delay(200);
                  move_spindle_right(speed);
                  delay(2000);
                      stop_spindle();
                      delay(1200);
                  }
                    move_spindle_left(speed);
               delay(50);
                    //stop_spindle();
                    //delay(0);
        }
          stop_spindle();

          //right button
```

```
          while(digitalRead(right_button) == HIGH){
                    if(digitalRead(right_limit_switch) == LOW){
                      stop_spindle();
                      delay(200);
                move_spindle_left(speed);
                delay(2000);
                     stop_spindle();
                     delay(1200);
                 }
                   move_spindle_right(speed);
               delay(50);
                     //stop_spindle();
                     //delay(1200);
        }
          stop_spindle();

          //=============================================================

          //forward button
          while(digitalRead(forward_button) == HIGH){
                    if(digitalRead(forward_limit_switch) == LOW){
                      stop_base();
                      delay(200);
                move_base_backward(speed);
                delay(2000);
                     stop_base();
                     delay(1200);
                 }
                   move_base_forward(speed);
               delay(50);
                     //stop_base();
                     //delay(1200);
        }
          stop_base();

          //back button
          while(digitalRead(back_button) == HIGH){
                    if(digitalRead(back_limit_switch) == LOW){
                      stop_base();
                      delay(200);
                move_base_forward(speed);
                delay(2000);
                     stop_base();
                     delay(1200);
                 }
                   move_base_backward(speed);
               delay(50);
                     //stop_base();
                     //delay(1200);
        }
          stop_base();

         }
}// end main
```

```
void stop_spindle(){
        analogWrite(left_spindle_pin, 0);
        analogWrite(right_spindle_pin, 0);
        digitalWrite(right_spindle_pin, LOW);
      digitalWrite(left_spindle_pin, LOW);
        delay(10);
}

void move_spindle_left(int speed){
  digitalWrite(right_spindle_pin, LOW);
  analogWrite(left_spindle_pin, speed);
}

void move_spindle_right(int speed){
   digitalWrite(left_spindle_pin, LOW);
   analogWrite(right_spindle_pin, speed);
}



void stop_base(){
  digitalWrite(forward_base_pin, LOW);
  digitalWrite(backward_base_pin, LOW);
  analogWrite(backward_base_pin, 0);
  analogWrite(forward_base_pin, 0);
  delay(10);
}

void move_base_backward(int speed){
  digitalWrite(forward_base_pin, LOW);
  analogWrite(backward_base_pin, speed);
}

void move_base_forward(int speed){
   digitalWrite(backward_base_pin, LOW);
   analogWrite(forward_base_pin, speed);
}
```

Appendix D: Bill of Materials/Rebuild Implementation

1	2 Separate 12V Power Supplies
2	1 Texas Instruments DRV8432 H-Bridge
3	1 Arduino Mega
4	1 5V to 3.3V Logic Convertor
5	1 Fully Assembled 30 Pin Ribbon Cable
6	5 Push Buttons
7	1 Emergency Stop Button
8	20 Male/Female Wire Connectors
9	2 Breadboards/ProtoBoards

Instructions to Replicate Design

1) Connect all limit switches to the Arduino using a breadboard and by following the schematics in this report.
2) Connect all 4 drill motors to the H-Bridge outputs as depicted in the schematics.
3) Solder all 5 buttons to a separate board and configure them with the Arduino as shown in the button schematic.
4) Connect all PWM signals from the Arduino to a 5V to 3.3V logic convertor.
5) Connect the Ribbon Cable to the H-Bridge.
6) Take the converted PWM signals and input them into the Ribbon Cable in the correct slots as shown in the schematics in this report as well as the schematics of the H-Bridge.
7) Connect the first 12V power supply to both the H-Bridge internal power and the Arduino Vin pin.
8) Connect the other 12V power supply to the PWM power switch of the H-Bridge.
9) Upload provided machine code into the Arduino