



Hydrophone Measurement System

By

Sam Clinard

2020

User's Manual

Contents

Part 1: Introduction

Part 2: Getting Started

Part 3: DIY Build **Instructions**

- 1) Hardware**
- 2) Electronics**
- 3) Software**

Part 4: System Details

- 1) Software**
 - a) Versions

- b) MATLAB GUI
- 2) System Components**
- 3) Troubleshooting Connections**
- 4) ReadMe**
- 5) Data Output**
 - a) Storage Matrix
 - b) CSV
- 6) SCPI Commands**
 - a) Oscilloscope
 - b) Function Generator
 - c) Arduino

Part I: Introduction

The Focused Ultrasound Foundation (FUSF) has developed an open-source, cost-effective, automated hydrophone scanning device that fulfills the need to characterize transducers in the ultrasound community. A validation paper has been submitted to Ultrasound in Medicine and Biology. The device is designed to be replicated with minimum time and monetary cost. To obtain this goal, it is composed of readily available parts and is programmed in MATLAB and the Arduino IDE. FUSF has created a GitHub Project containing a User Manual, Bill of Materials, 3D print files, and software. The device has similar functionality to commercial systems and is customizable to each group's applications.

There are currently two options for groups that need to measure acoustic fields. The first is to buy a ready-to-use commercial device, which generally provides good functionality and processing software. However, it is expensive and lacks customizability due to licensing. The other is to design your device, which takes valuable time. Our hydrophone scanner provides a middle ground between the two. It has basic functionality for groups looking to divert resources to other projects. It also provides a template for groups looking to add more advanced features. Overall, this device preserves the advantages of both options.

Our system was designed to be easily replicated and modified. Its advantages follow from its simplicity. The hardware components are readily available from online suppliers or can be 3d printed. The electronics consist of an Arduino, an Arduino Driver Shield, and NEMA 17 stepper motors. The software makes use of the free Arduino IDE and MATLAB, which most groups have licenses too.

The device uses lead screws and linear rails to achieve precision 3d motion. The user manual contains detailed DIY build instructions, which should take about a week to complete. The GitHub project contains a Bill of Materials with parts to order and 3d files for printed components. The total cost for materials is under \$1,000.

The GitHub project also contains a MATLAB Graphical User Interface (GUI), which orchestrates the entire scan. The GUI communicates over USB to an Arduino, function generator, and oscilloscope to move the hydrophone and produce and acquire a signal. It then saves the data in a MATLAB Matrix or a CSV.

We hope that this will become a collaborative project. The system currently has basic functionality. There are several additional features we plan to work on and release in upcoming versions. Any improvements may be shared with us for inclusion in future versions. Please send questions or comments to sam.clinard@utah.edu.

Part II: Getting Started

This section will guide the user through the steps of configuring instrument connections and running a scan. It assumes that the device has been constructed according to Part III: DIY Build Instructions.

Note – The software is programmed for Tektronix oscilloscopes and generators. See Part 4.6 SCPI Commands

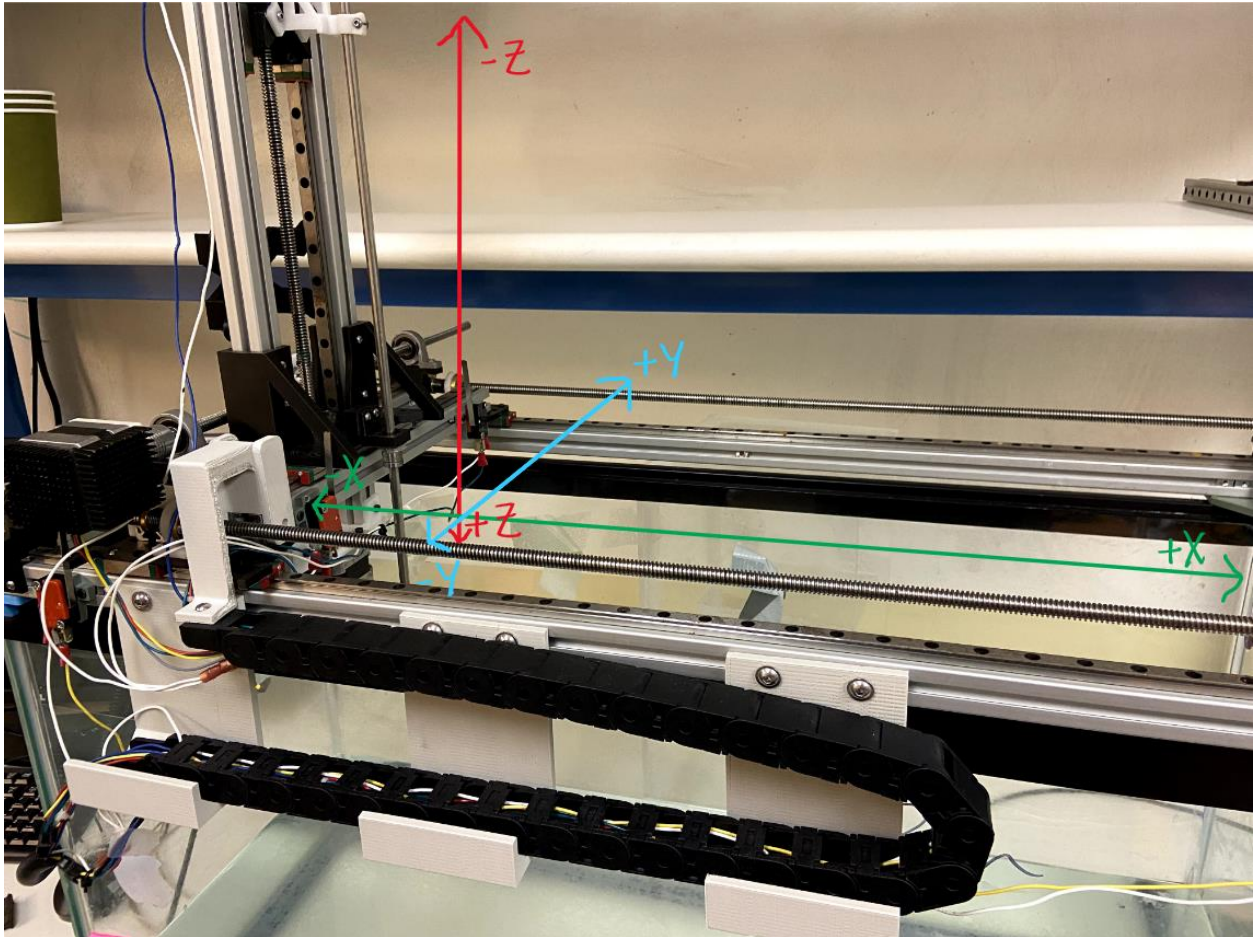


Figure 1: Tank with definitions of axes: the carriage moves right in the positive X direction, forward in the positive Y direction, and downward in the positive Z direction.

1) **Check Hardware**

Step 1: (Optional) Set the limit switches' location to avoid hitting a transducer or other object.

Step 2: Ensure motor couplings are secure.

Step 3: Plug-in power supply to Arduino Shield.

2) **Check Electronics**

(Optional, but suggested if a system has not been used before, or has not been used in a while)

Step 1: Open Arduino file, "FUSF_Hydro_Arduino.ino"

Step 2: Open Serial Monitor: Tools -> Serial Monitor

Step 3: Press limit switches in X, Y, and Z directions.

The Serial Monitor should display:

X X X Pushed / closed, Y Y Y Pushed / closed, Z Z Z Pushed / closed

If these messages appear for all limit switches, then the circuit is connected

3) **Connections Part 1**

*Note – This part is only required to connect instruments that have not been connected to the computer before.

Step 1: Open MATLAB command window

Step 2: Connect oscilloscope, generator, and Arduino with a USB cable to Com Ports

Step 3: Get Com Port address of Arduino

See Start Menu -> Device Manager -> Ports (COM & LPT)

If the Arduino is not listed, try reconnecting the USB cable

Note Arduino's Com Port ("Arduino Uno (COM_)")

Step 4: Find Oscilloscope Address

Method 1: Open Tektronix Visa Resource Manager -> Instrument Manager

Method 2: In the MATLAB Command Window type the following

```
handle = instrhwinfo('visa','tek');  
handle.ObjectConstructorName(1)  
handle.ObjectConstructorName(2)
```

This will return the address in the following format:

```
{'visa('tek', 'USB::0x0699::0x034A::C020626::INSTR');'}
```

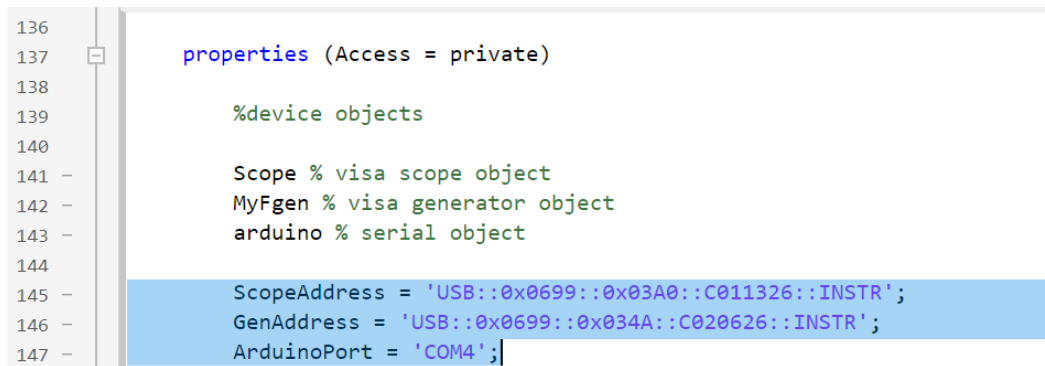
If the address is not visible by either method, try reconnecting the USB Cable. If it still does not appear, restart the computer. Otherwise, see 4.3 Troubleshooting Connections.

Copy both addresses.

Step 5: Open FUSF Hydrophone GUI and navigate to code view

Step 6: Insert Arduino, Oscilloscope, and Generator addresses into code.

Find properties (access = private) and replace strings as highlighted in Figure 2.



The screenshot shows a MATLAB code editor with line numbers 136 to 147 on the left. The code is as follows:

```
136
137 properties (Access = private)
138
139     %device objects
140
141     Scope % visa scope object
142     MyFgen % visa generator object
143     arduino % serial object
144
145     ScopeAddress = 'USB::0x0699::0x03A0::C011326::INSTR';
146     GenAddress = 'USB::0x0699::0x034A::C020626::INSTR';
147     ArduinoPort = 'COM4';
```

Lines 145, 146, and 147 are highlighted in blue in the original image.

Figure 2: Replace highlighted addresses in the GUI code.

4) Connections Part 2: BNC Connections

Step 1: Connect Generator's CH1 Output to transducer matching network.

Step 2: Connect Generator's Trigger Output to Scope's External Trigger.

Step 3: Connect Hydrophone to CH1 of Scope.

5) Launch Application

Step 1: Ensure the Arduino is running 'FUSF_Hydro_Arduino.ino'. If not, upload this script from the Arduino IDE.

Step 2: Open MATLAB command window

Step 2: Connect oscilloscope, generator, and Arduino with USB cables to the computer

Step 3: Ensure the 'Continue_Cancel.mlapp' file is in the Matlab pathway.

Step 4: Run FUSF Hydrophone GUI

Step 5: Pause for one minute while instruments connect. The lamps in the connection panel will turn from red to green once each device connects.

6) Generate Function

Step 1: Navigate to the 'Function Generator' tab in GUI

Step 2: Adjust settings for the desired function

Step 3: Click Generate Button

*Note – The settings can be adjusted before launching the GUI in the 'Design View' of App Designer.

7) Configure Scope

Step 1: Navigate to the 'Oscilloscope' tab in GUI

Step 2: Adjust Settings

Step 3: Click Apply Button

*Note – It is often easier to adjust scope settings manually on the oscilloscope. This may be done as needed up to Step 5 in 2.10 Run Scan.

8) Find Signal (Optional)

Step 1: Navigate to the 'Scan' tab in GUI

Step 2: Click the Hard Home button

Step 3: (Optional) Adjust software limits

Step 4: Move Hydrophone (x-axis for example)

- i) Under the move panel, insert step size [mm] in X edit field.
- ii) Click the X+ button to move in a positive X direction. Click X- button move in a negative direction.

Step 5: Move the hydrophone to the expected signal location.

Step 6: Adjust scope's vertical scaling and horizontal scale to find the signal.

Step 7: Use the "Center" button to find the maximum intensity of the signal.

- i) Input scan dimensions to designate the length of the 1d scans in the X, Y, and Z directions
- ii) Click the Center button
- iii) Note that the Soft Home automatically updates and the "Move X, Y, Z" boxes update to show how much the hydrophone moved to get to the new soft home
- iv) Change scan dimensions and repeat until confident that hydrophone is at maximum voltage

Alternative: Move the hydrophone incrementally to find maximum voltage.

Step 8: Click Set Soft Home Button to save signal location.

9) Read_me.txt (Optional)

Step 1: Navigate to the 'Scan' tab in GUI

Step 2: Enter information under the 'Info recorded in Read_me.txt' panel.

*Note – The scan does not use this information.

10) Scan Settings

*Note – all dimensions are in millimeters.

Step 1: Navigate to the 'Scan' tab in GUI

Step 2: Select the program from the drop-down menu.

Step 3: Select data format in Save As drop-down menu. See 4.5 Data Output

Step 3: (Optional) Enter the Soft Home location to define the center of the scan. This may have been found in 2.8 Find Signal. The center point can be modified again during 2.11 Run Scan step 3)

Step 4: Enter Scan Dimensions

Step 5: Enter Scan Resolution.

*Note – to avoid rounding errors choose a resolution that is a divisor of the scan dimensions.

11) Run Scan

Step 1: Click the Run Scan button

Step 2: Create a new folder and name your data file.

- If "Home Switch" is On, the motors will home each axis then move to soft home.
- Otherwise, the scan will start from Soft Home.

Step 3: Adjust the position of the soft home using the GUI move panel.

Step 4: Adjust scope settings if needed.

Step 5: Click Continue Button in Soft Home GUI.

Part III: DIY Build Instructions

The following build instructions make use of the parts listed in the Bill of Materials (BOM). The design may need to be modified depending on the availability of these parts. Lead times on ordered parts may take up to a month from the suppliers listed in the BOM. 3D printed parts are limited by the group's access to 3d printing. You should acquire all the parts listed in the BOM and 4.2 System Components before beginning this section. In addition to the BOM, a Fusion Assembly has been provided, containing the 2020 aluminum extrusion and 3d printed parts.

1) Hardware

a) Frame

Step 1: Connect 2 x 800 mm and 2 x 300 mm 2020 extrusion using corner brackets.



Step a.1: Connect frame

Step 2: Place frame on the tank and secure with screws in corner brackets.

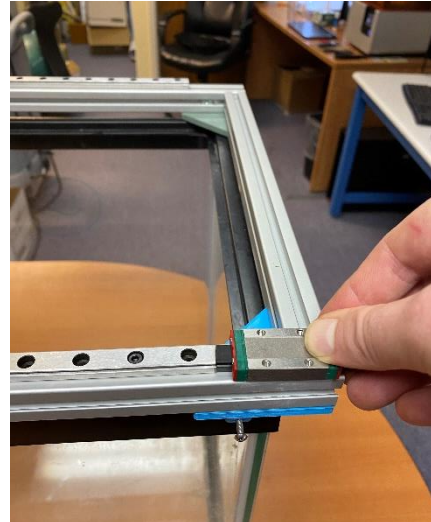
Step 3: Attach 700 mm MGN9H rails to 800 mm extrusion.



Step a.2-3: Place frame on tank and attach rails

b) X-Axis

Step 1: Slide 2 MGN9H carriages onto each of the X-axis rails
*keep loose for now



Step b.1: Slide carriages onto rails.

Step 2: Connect X-axis left motor mount to extrusion

Step 3: Connect X-axis right motor mount to extrusion

*Put two T-nuts in the right side



Steps b.2-3: Connect X-axis motor mounts

Step 4: Connect four rotational bearings to bearing mounts.

Step 5: Connect bearing mounts to four corners of extrusion.



Step b.4: Connect rotational bearings to mounts

c) **Y-Axis (separate from the frame)**

Step 1: Connect 2 x 300 mm 2020 extrusion to X-axis plate left and X-axis plate right.



Step C.2: Connect extrusion to x-axis plates.

Step 2: Hot Glue or screw 2 x nut couple to right axis plate and left axis plate

*Design has been modified from the picture. Nuts and bolts attach the nut couples.

*The X-axis plates will be attached to the extrusion (Step 1).



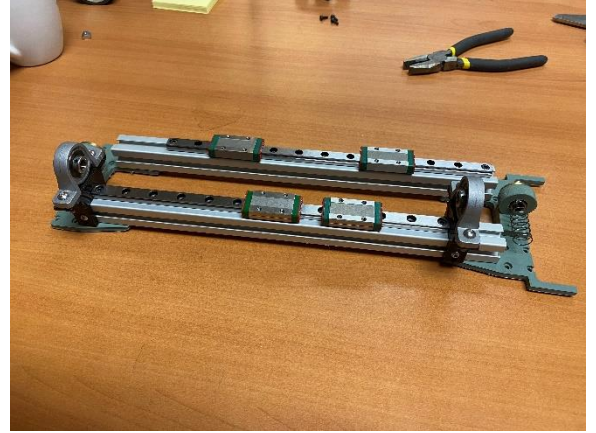
*Step C.2: Attach nut couples to x axis plates. *Note – nut and spring will not be in 3d part yet.*

Step 3: Connect 250 mm MGN9H rails to 300mm extrusion.

Step 4: Slide 2 MGN9H carriages on each rail.

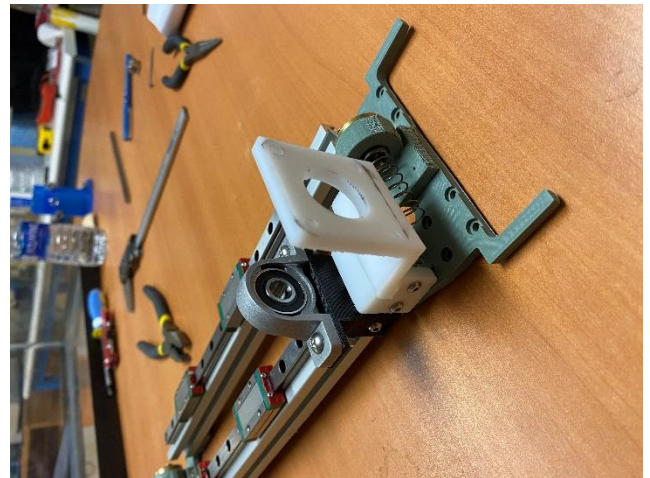
Step 4: Connect two rotational bearings to bearing mounts.

Step 5: Connect bearing mounts to negative side y-axis extrusion.



Step 6: Connect the Y-axis motor mount to the motor side of the y-axis extrusion.

Step C.3-5: Attach rails, bearings, and carriages to y-axis extrusion.



Step C.6: Connect Y axis motor mount.

d) Z-Axis (separate from the frame)

Step 1: Connect 350 mm MGN9H rails to two 400 mm extrusions with rail endings

*leave loose

Step 2: Connect rotational flange bearing to Y carriage.



Step D.2: Connect flanged bearing to y carriage.

Step 3: Place and connect 400 mm extrusion into Y carriage with rails facing in.



Step d.3: Connect 400mm extrusion.

Step 4: With extrusion parallel to the table. Slide two MGN9H carriages onto each of the z-axis rails.

Step 5: Insert z axis nut into the center of Z nut couple/carriage. Secure with either hot glue or screws.



Step d.5: Insert z axis nut

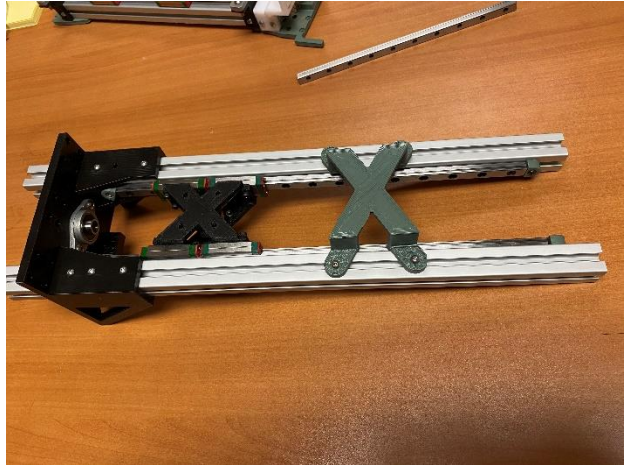
Step 6: Attach Z nut couple/carriage to 4 MGN9H carriages.



Step d.6: Attach z carriage

Step 7: Put rail ends on the rails' top side to prevent carriages from sliding off.

Step 8: Attach at least one and up to three Z-axis back supports.



Step d.8: Attach z axis back support

Step 9: Connect rotational flange bearing to Z cap.

Step 10: Attach Z cap to the top of extrusion.



Step d.10: Attach Z cap

e) Mount Y-axis to frame

Step 1: Lift y-axis assembly and place on x-axis MGN9H carriages.

Step 2: Connect x-axis plate left and x-axis plate right to carriages.

f) Mount Z-axis to Y-axis extrusion

Step 1: Lift Z-axis and place on the y-axis MGN9H carriages.

Step 2: Connect y carriage to y-axis MGN9H carriages.

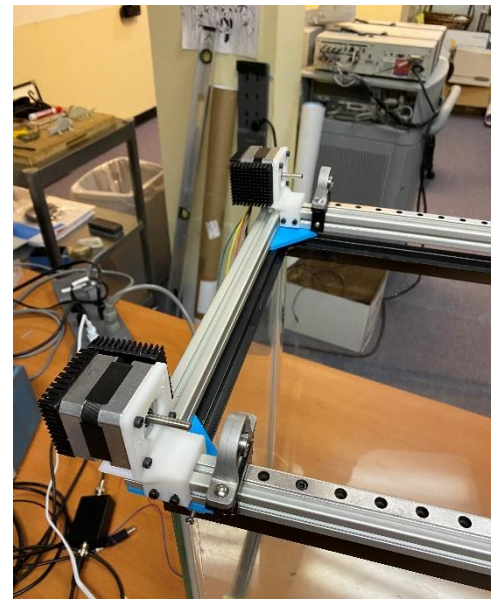
Step 3: Put rail ends on y-axis rail opposite motor side.



Steps f.1-2: Attach z axis to y carriages.

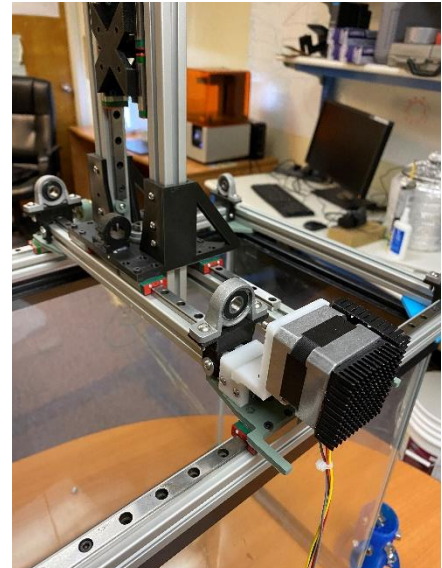
g) Mount all motors

Step 1: Attach NEMA 17 Motors to left and right x-axis motor mounts.



Step g.1: Attach motors to x axis

Step 2: Attach NEMA 17 Motor to y-axis motor mount.

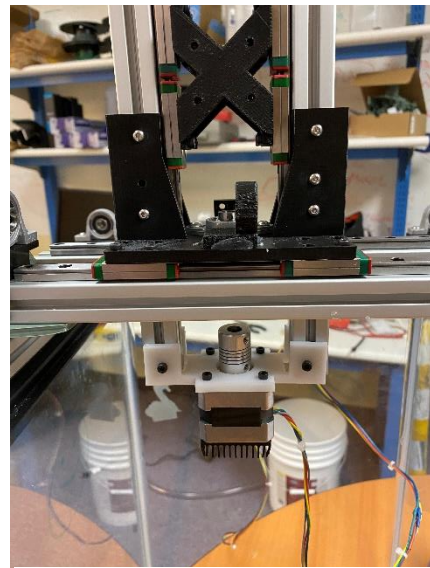


Step g.2: Attach y motor

Step 3: Attach NEMA 17 Motor to z-axis motor mount (currently free of frame)

Step 4: Place a motor coupler on the z-axis motor shaft.

Step 5: Attach z-axis motor mount to the bottom of z-axis extrusion.

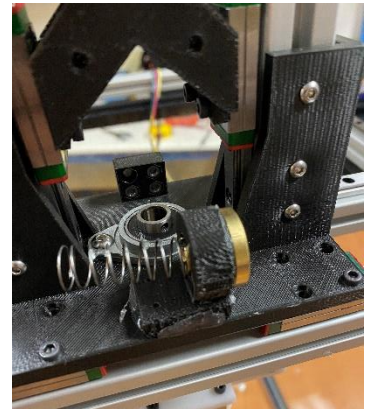


Step g.3-5: Attach z motor

h) Mount all lead screws

*Note – a quick way to thread a lead screw is to grip it with a handheld drill.

Step 1: Hot Glue or screw three anti-backlash nuts into nut couplers on the left and right x-axis plates and y carriage.



Step h.1: Attach nuts to nut couplers.

Step 2: Starting opposite the x-axis motors. Insert the lead screws through the first set of rotational bearings and up to the anti-backlash nuts.

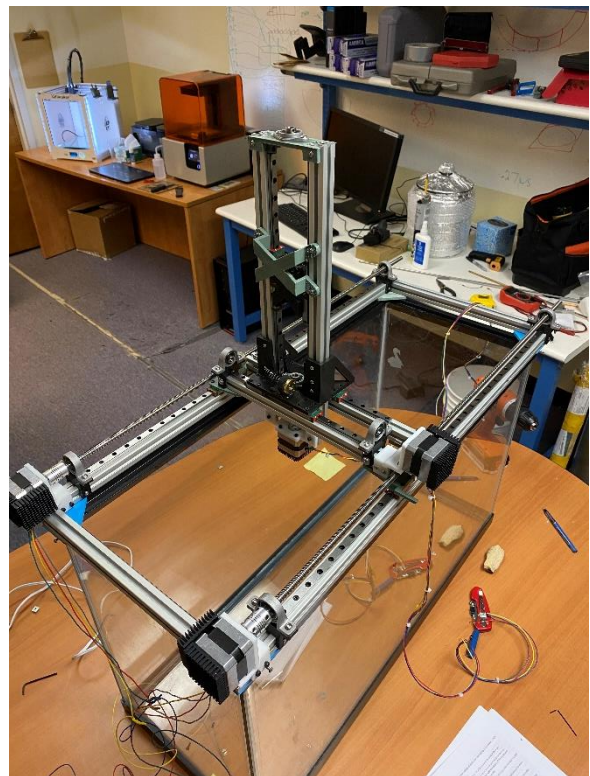
Step 3: While compressing the spring between the two components of the anti-backlash nut, thread the lead screw into the nut. Do this with both sides of the x-axis.

Step 4: Place motor couplers on the x-axis motor shafts

Step 5: Notice that the y and z axes slide along the x-axis without turning the x-axis lead screws.

Step 6: Slide the x-axis lead screws through the next set of rotational bearings up to the motor's shaft.

Step 7: Connect the motor coupler to the motor shaft and lead screw on both sides of the x-axis.



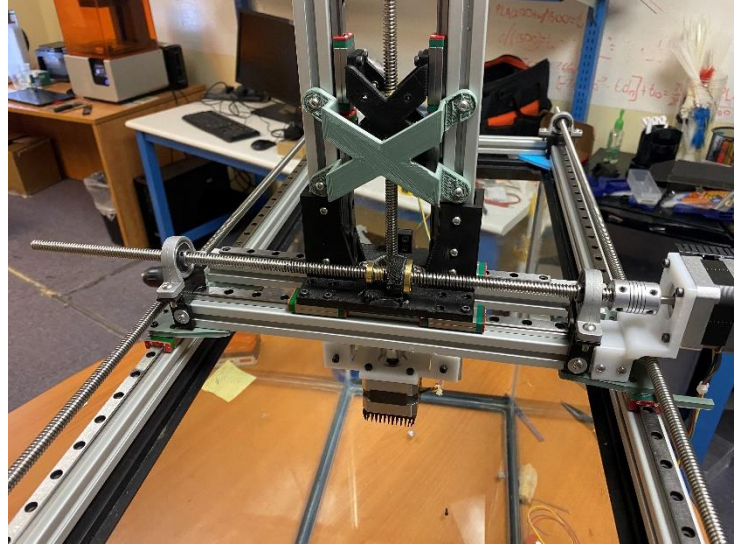
Step h.2-7: X axis screws

*Note – for optimal performance, align the lead screws. The motor mounts and rotational bearing mounts move up and down to enable the user to align the screws with the x-axis anti-backlash nuts' height. This alignment will affect the maximum speed of the x-axis.

Step 8: Starting opposite the y-axis motor. Insert the lead screw through the first rotational bearing and up to the y-axis anti-backlash nut.

Step 9: While compressing the spring between the two components of the anti-backlash nut (as done for the x-axis), thread the lead screw into the nut.

Step 10: Slide the y carriage through the next rotational bearing and connect the screw to the motor shaft using a coupler.



Step h.8-10 y axis screw

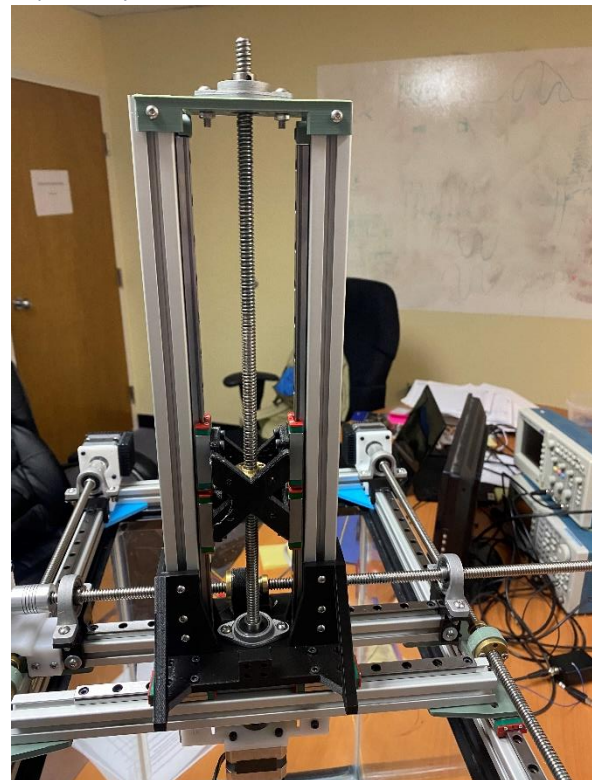
Step 11: Starting from the top of the z-axis, insert the lead screw through the flange rotational bearing and up to the z-axis nut.

Step 12: Thread the screw through the z-axis nut.

Step 13: Slide the z nut couple/carriage towards the y carriage until the screw inserts into the next flange rotational bearing.

Step 14: Connect the z-axis motor shaft and z-axis screw using a coupler.

Step 15: Tighten all set screws in motor couplers and rotational bearings.



Steps h.11-14: Z axis screw

2) Electronics

Parts Required:

Arduino Uno

Synthesos gShield

3d printed Arduino home casing

Limit Switches (5)

Extra Wires

Wire connections – heat shrink, crimp connectors, or pin connections

Screwdriver, wire crimper, allen wrench, screws, T-nuts

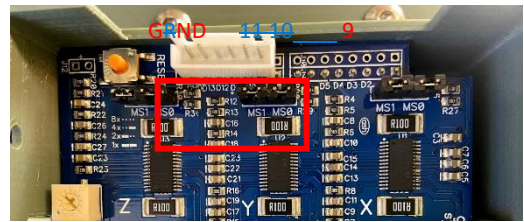
Optional and helpful: Drag chain cable carrier and 3d printed holders to organize wires, wire-to-board header

*If using optional wire organizers, see part d) before beginning

a) Motors

Step1: Place Synthesos gShield onto Arduino Uno

Optional: Solder on wire-to-board header base from GND pin to pin 9



Step a.1: Solder on wire-to-board header

Optional: Print Arduino home case

Screw Arduino board into the home case

Attach to side extrusions for Arduino protection and more accessible access



Step a.1: Arduino home attached to side extrusion

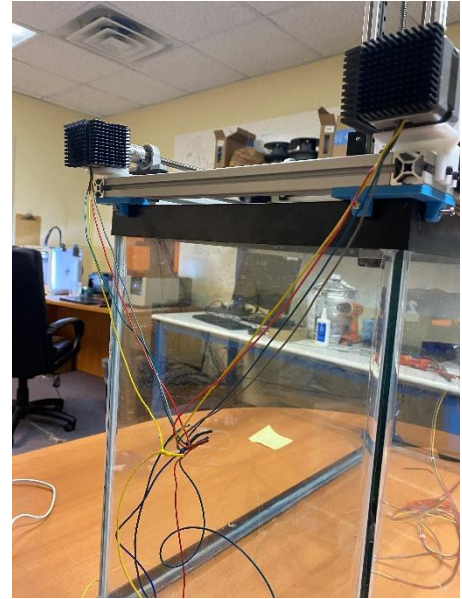
Step 2: Connect and extend the x-axis motor wires.

*Note – When determining the length of wires, keep in mind the electronics' maximum position.

One driver in the shield drives both motors.

Twist each color wire from the motors together with an extension wire to the shield.

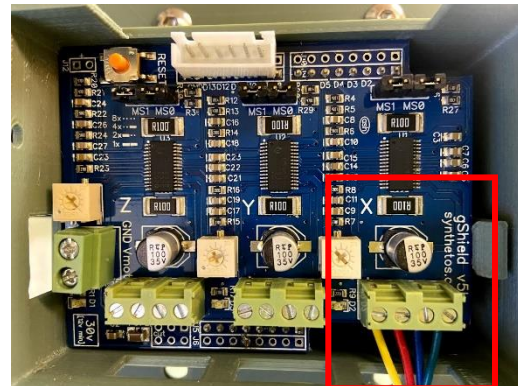
Wrap end in heat shrink – or use crimp connections



Step a.2: Connect x axis motor wires

Step 3: Connect x-axis motor wires to gShield.

The order of wires from left to right is yellow, red, blue, green. (blue = grey on NEMA 17)

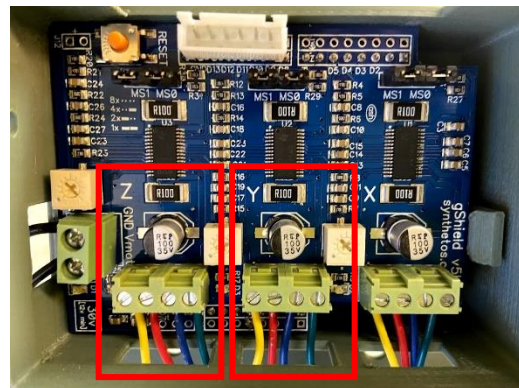


Step ~~a.4~~^{a.3}: Connect x axis motor wires.

Step 4: Extend y-axis motor wires to gShield and connect.

Step 5: Extend z-axis motor wires to gShield and connect.

*Note – datasheets for the NEMA 17 Motor and Synthesos gShield are in the BOM



Step a.4, a.5: Connect y and z axis motor wires

b) Limit Switches

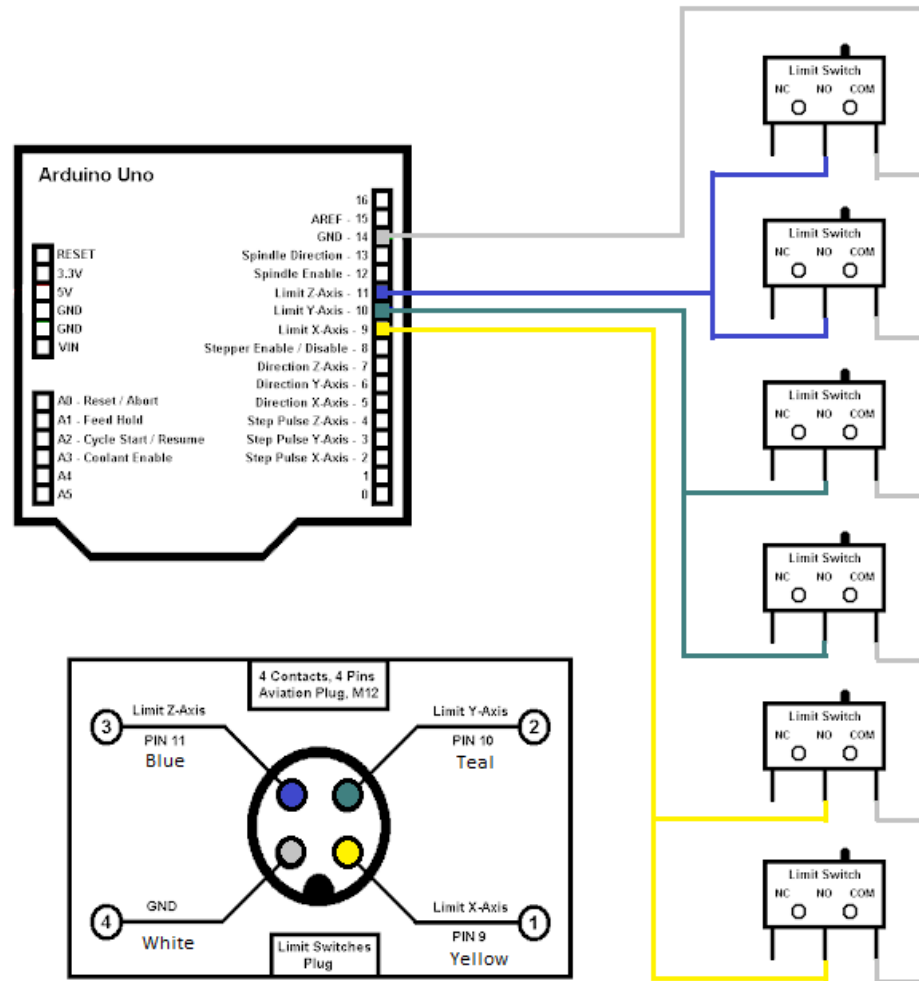
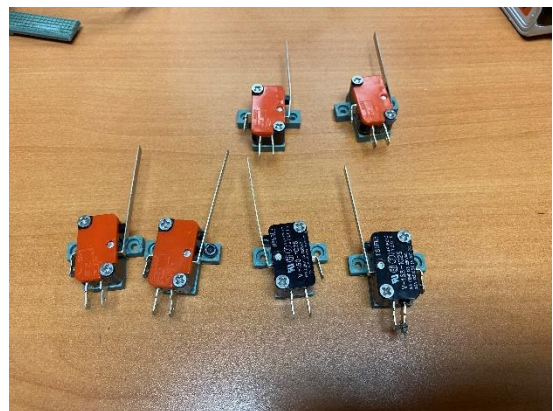


Figure b) Limit Switch Circuit Map

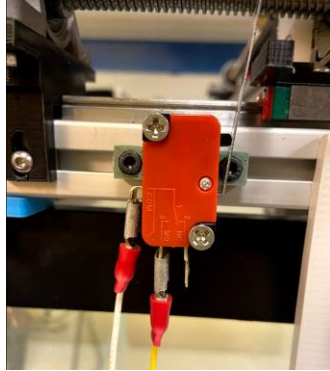
Step 1: Attach limit switches to limit mounts using screws.

*Tip - consider which direction each switch needs to face once mounted.

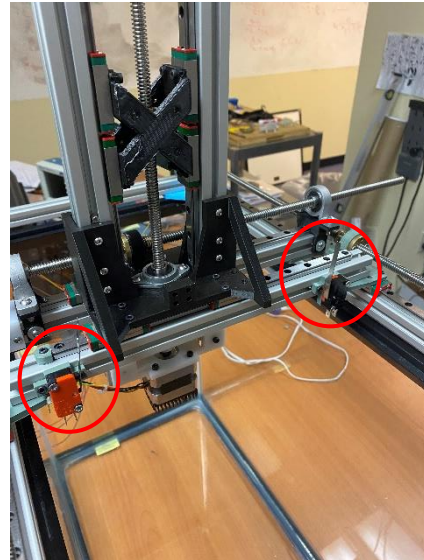


Step b.1: Attach limit switches to limit mounts

Step 2: Mount limit switches on x, y, and z extrusion.



Step b.2: attach mounts to extrusions

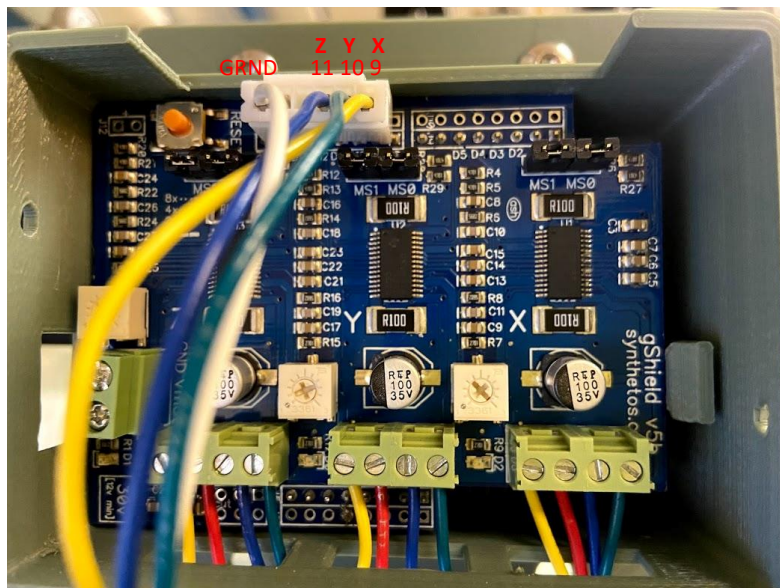


Step b.2: y axis limit switches.

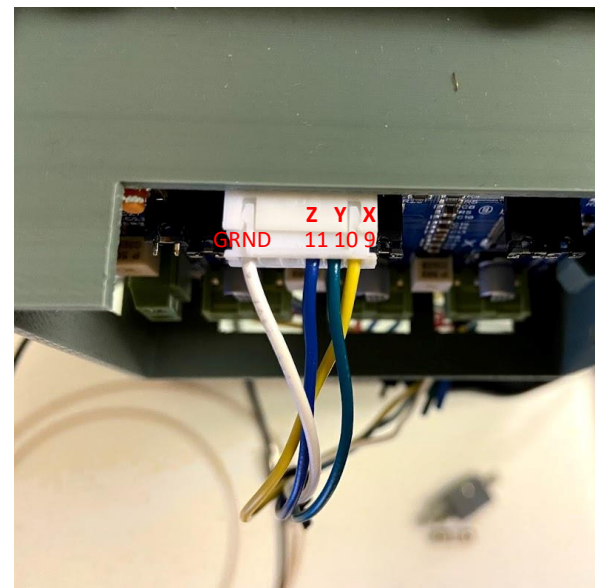
*Note – In the following steps, it helps use crimp connectors or solder wires into place.

Step 3: Connect the limit switches in parallel according to the circuit diagram in Figure b). The pin is high until the limit switch contact closes, causing the input voltage to zero. The Arduino script checks the pins - high for open and low for closed.

*Note – current design does not use two limit switches on the z-axis. The circuit does not change except that there is one Z-limit switch instead of two.



Step b.3 Arduino board connections to limit switches

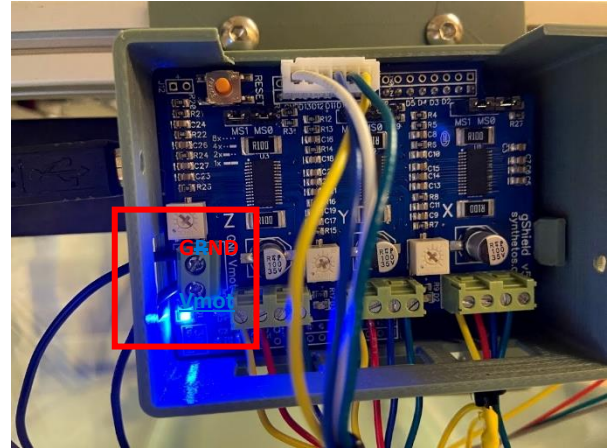


c) Power Supply

Step 1: Connect the male pigtail cable to the power supply.

Step 2: Connect pigtail cable to gShield as labeled.

*Correct connections (GND to GND and V to Vmot) are important here, or else the board will short out.



Step C.2: Connect Power Supply

d) Suggestions for Wire Organization

1) Consider using a drag chain cable carrier to organize wires. The drag chain allows wires to move with the carriage, so points of tension are relieved. This prevents wires from breaking and disrupting the circuit.

Our cable guide hosts the wires for Y and Z axis motors, and Y and Z-axis limit switches

*If using this method, make sure to account for the longer lengths of wire required to loop through the cable carrier.

2) Consider using crimped connections when making branched connections or extending wire paths.

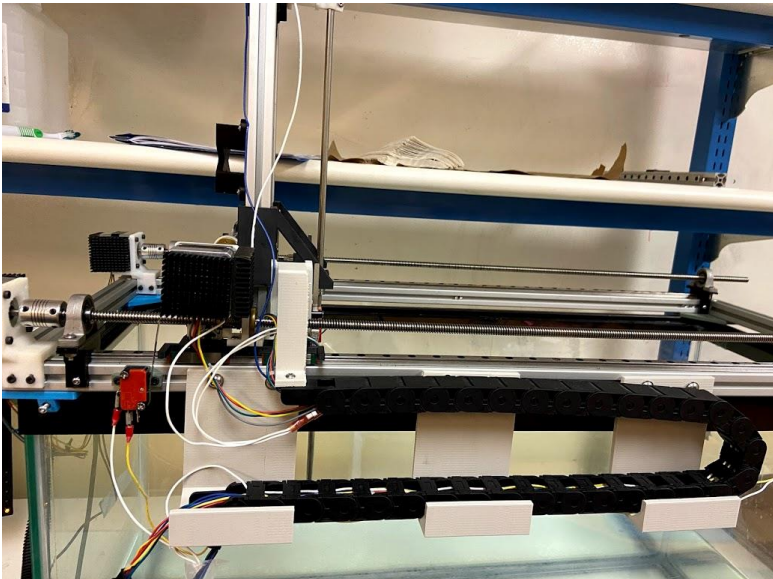


Figure d) Drag chain cable carrier attached to side extrusion and crimped wire connections to connect limit switch GND wires

3) Software

a) **Tektronix Visa**

Step 1: Install the Tektronix Visa - <https://www.tek.com/oscilloscope/tds7054-software/tekvisa-connectivity-software-v420>

Step 2: Set Tek Visa as the primary visa. Other drivers could prevent Tek Visa from connecting to the instruments. See MATLAB note copied below or in full [here](#).

Mathworks - "For 64-bit Tektronix VISA support, it is important to note the following if you have a multi-vendor VISA installation (e.g., you have installed drivers from Tektronix and another vendor such as Keysight). If you are using 64-bit Tektronix VISA on a machine with VISA implementations from multiple vendors, it is required that Tektronix VISA be configured as the primary VISA for it to be usable with Instrument Control Toolbox. Most 64-bit VISA implementations include a utility that allows you to select the primary and preferred VISA implementations. Use the VISA utility to set Tektronix VISA to be the primary VISA implementation on your machine. This step can be accomplished at any time, regardless of the order of installation of the VISA drivers."

*Note – We had connectivity issues running TekVisa with any other visa installed regardless of setting TekVisa as primary.

b) **MATLAB**

Step 1: Install MATLAB with the Instrument Control Toolbox. For version considerations, see 4.1.a Versions.

Step 2: Put 'FUS_Hydrophone_Main.mlapp' and 'Continue_Cancel.mlapp' in MATLAB pathway (i.e same folder).

Step 3: Check if your oscilloscope and function generator are compatible with the SCPI commands used in communications. If not, see Part 4.6 SCPI Commands

Compatible Oscilloscopes (We used Tektronix TDS2001C) - TDS200, TDS1000/TDS2000, TDS1000B/TDS2000B, TDS1000C-EDU/TDS2000C, and TPS2000/TPS2000B Series Digital Oscilloscopes.

Compatible Generators (We used Tektronix AFG3022C) - AFG3000 Series Arbitrary Function Generators

Step 4: Change the resource addresses as described in 2.2 Connections part 1.

c) **Arduino**

Step 1: Install the Arduino IDE - <https://www.arduino.cc/en/Main/Software>

Step 2: Add the AccelStepper Library

In Arduino IDE -> Sketch -> Include Library -> Manage Libraries

Search AccelStepper

Install AccelStepper by Mike McCauley

Step 3: Open "FUS_Hydro_Arduino.ino"

Step 4: Connect Arduino to Computer COM Port

Step 5: Upload script to Arduino

Step 6: Plug in the Power Supply

Step 7: Open the Serial Monitor (Ctrl + Shift + M)

Step 8: Test movement with the following commands. Start with small movements.

*Note – software limits are implemented in MATLAB. Be careful not to run carriages off the rails.

h = home motors

(axis)(number of steps) = move in axis this number of steps. Positive numbers move in the positive direction as defined in getting started (i.e., positive is down in z-axis!)

Examples

x2000 = Move 2000 steps in positive x-direction

y-500 = Move 500 steps in the negative y-direction

z2000 = Move 2000 steps in positive z-direction (down)

Step 9: (Optional) Optimize speed and acceleration.

For each axis, do the following iterative steps

Move several thousand steps and observe if the motor skips or stalls.

If it skips or stalls, reduce speed in the script, upload, and try moving again.

If it does not skip or stall, increase speed in the script, upload, and try moving again.

We found that each axis could move at around 800 steps/s and 6000 steps/s².

```
int MaxxSpeed = 800;  
int MaxzSpeed = 700;  
int MaxySpeed = 700;  
  
int Accel = 6000;
```

The system should be complete now. If you have any questions or suggestions, please email Sam.Clinard@utah.edu. Otherwise, see Part 1: Getting Started to take your first scan.

Part IV: System Details

1) Software

a) Versions

Program	Version	Documentation
MATLAB	R2020a	https://www.mathworks.com/help/
App Designer	R2020a	MATLAB App Designer
Instrument Control Toolbox	Version 3.13	MATLAB Instrument Control Toolbox
Arduino IDE	1.8.10	Arduino Reference
AccelStepper Library	1.59.0	AccelStepper Library
TekVisa	v4.2.0.26	Tek Visa Programmer Manual

b) MATLAB Graphical User Interface

The GUI consists of three tabs – Scan, Function Generator, and oscilloscope. Panels divide each tab into groups of related components. The following is a detailed description of each GUI component in the Scan tab.

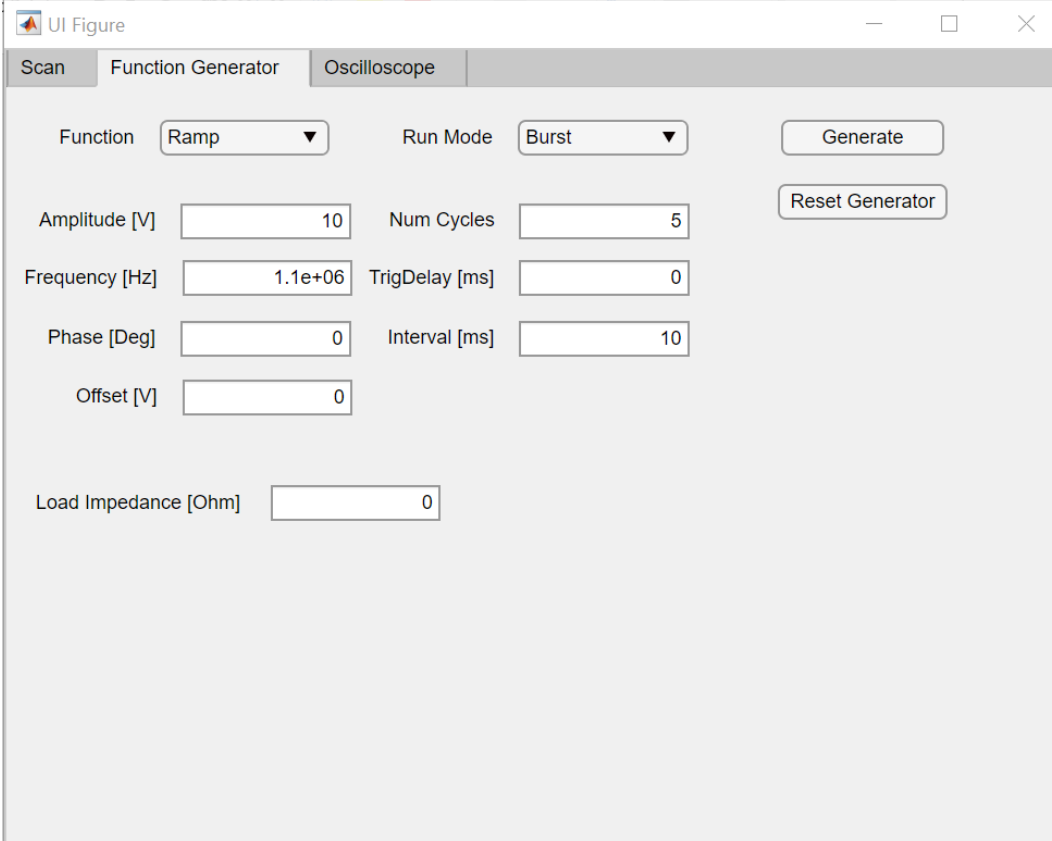
The Function Generator and Oscilloscope Tabs components have the same function as the corresponding device's physical components. You may use the MATLAB or device controls; however, the program currently records the MATLAB GUI settings. The use of the device's controls may lead to incorrect information in the Read_Me.txt.

i. Scan Tab

Panel	Component(s)	Type	Function
Upper Row	Program	Drop Down	Selects the type of program to run.
	Save As	Drop Down	Selects the save format of the data. See 4.5 Data Output
	Run Scan	Button	Starts the scan process. See 2.10 Run Scan
	Center	Button	Finds the maximum signal position by running linear 1D scans in the X, Y, and Z direction based on the user's scan dimensions. Automatically updates Soft Home and the Move X, Y, Z values based on how much it moved to get to its new position.
	Hard Home	Button	Homes the device and enables the Limit and Position Panels.
	Set Soft Home	Button	Copies the coordinates from the Position panel to the Soft Home panel.
Soft Home	X, Y, Z	Edit Fields	The location of soft home [mm] – defined as the center of the scan.
Scan Dimensions	X, Y, Z	Edit Fields	The dimension of the scan [mm] in each axis.
Limit	X, Y, Z	Edit Fields	Software Limits [mm] enabled after hard home.
Move	X, Y, Z	Edit Fields	The length [mm] of each movement.

	X+, Y+, Z+	Buttons	Move respective axis in the positive direction
	X-, Y-, Z-	Buttons	Move respective axis in the negative direction
Position	X, Y, Z	Display	Displays current position of hydrophone
n/a	Resolution [mm]	Edit Field	The spatial resolution of the scan. Should be a divisor of the scan dimensions.
	Save Wave in Matrix	Switch	Save the waveform in the n+1 dimension. See 4.5. i Storage Matrix
	Home Switch	Switch	When switched On will return to the Hard Home before starting a scan.
	Hydrophone Calibration Value	Edit Field	Converts voltage values to pressure in final image plots; unique to each transducer.
	Stop	Button	Breaks out of the scan. Matrix data will be lost. CSV will save up to break.
	Pause	Button	Pauses program until pushed again.
Connections	Arduino, Generator, Oscilloscope	Buttons	Connects to respective devices. The GUI attempts to connect when run.
	n/a	Lamps	Buttons mentioned above serve as titles. Turn green when the device connects. *Note – will not turn back to red if device disconnects.
Read_Me.txt	Several	Edit Fields	A Read_Me.txt stores this information in the folder selected at the beginning of step 10 run scan in part 2. The scan does not actively use this information.

ii. Function Generator Tab



The image shows a software window titled "UI Figure" with three tabs: "Scan", "Function Generator", and "Oscilloscope". The "Function Generator" tab is active. It contains several controls for generating a signal:

- Function:** A dropdown menu set to "Ramp".
- Run Mode:** A dropdown menu set to "Burst".
- Buttons:** "Generate" and "Reset Generator" buttons are located on the right side.
- Amplitude [V]:** A text input field containing "10".
- Num Cycles:** A text input field containing "5".
- Frequency [Hz]:** A text input field containing "1.1e+06".
- TrigDelay [ms]:** A text input field containing "0".
- Phase [Deg]:** A text input field containing "0".
- Interval [ms]:** A text input field containing "10".
- Offset [V]:** A text input field containing "0".
- Load Impedance [Ohm]:** A text input field containing "0".

iii. Oscilloscope Tab

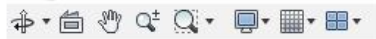
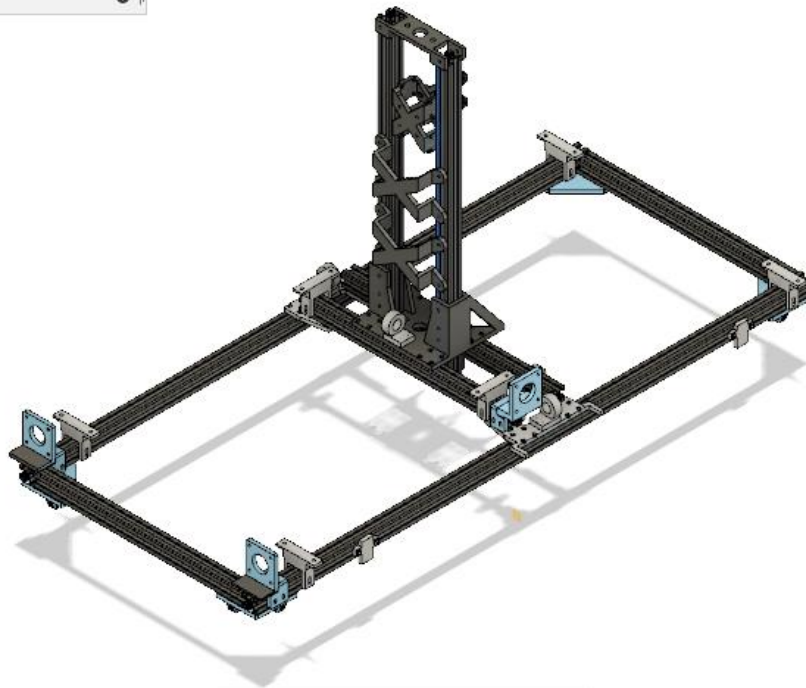
The screenshot shows a software window titled "UI Figure" with three tabs: "Scan", "Function Generator", and "Oscilloscope". The "Oscilloscope" tab is active. It contains several configuration controls:

- Coupling:** A dropdown menu set to "AC".
- Attenuation:** A dropdown menu set to "1".
- Vertical Scale [V]:** A text input field containing "4".
- Acquire Mode:** A dropdown menu set to "Average".
- # Average:** A dropdown menu set to "4".
- Horizontal Scale [s]:** A text input field containing "2.5e-6".
- Horizontal Position [s]:** A text input field containing "0".
- Trig Source:** A dropdown menu set to "Ext".
- Trig Couple:** A dropdown menu set to "DC".
- Slope:** A dropdown menu set to "Rise".
- Drop Down:** A dropdown menu set to "Option 1".

On the right side of the tab, there are two buttons: "Apply" and "Reset Scope". Below these buttons is a "Notes" section with a text box containing the message: "Scope connection is configured for Channel 1."

2) System Components

The provided Bill of Materials contains a detailed list of the electronics and hardware. It includes suppliers, cost, and datasheets when applicable. The following is a list of the 3d printed parts. We used an Ultimaker 2+ with PLA to make most of the parts. The motor mounts, however, require a material that can withstand heat. We used the Formlabs Form 2 with Rigid Resin for these parts. Each 3d part is provided in Autodesk Fusion 360 format (.f3d) and STL. Also, there is a Fusion Assembly provided.



1 Face | Area : 2144.80 mm^2

Part	Material	Number
Corner_Bracket	PLA	4
Bearing_mount	PLA	6
X_axis_plate_left	PLA	1
X_axis_plate_right	PLA	1
X_axis_left_motor_mount	Rigid	1
X_axis_right_motor_mount	Rigid	1
Y_axis_motor_mount	Rigid	1
Y_carriage	PLA 40% infill	1
Nut_Couple	PLA	3
z_back_support	PLA	2
Z_cap	PLA	1
Z_motor_mount	Rigid	1
Z_nut_carriage	PLA	1
Z_rod_holder	RIGID	1
Limit_mount	PLA	6

3) Troubleshooting Connections

Problem - Oscilloscope or function generator does not connect on MATLAB launch.

- i) Make sure the oscilloscope is compatible with the software. See 4.6 SCPI Commands


*Note – after each step in 2 through 4, try to connect by hitting the device's button in the connect panel or relaunching the GUI.

- ii) Make sure the USB cable is connected and not damaged.
- iii) Unplug/plug-in USB cable.
- iv) Make sure the device address is correct in MATLAB code. See 2.2 Connections Part 1
- v) Restart computer

Check the visa through the MATLAB command line.

- vi) Enter "instrhwinfo('visa') in the MATLAB command line
- vii) Enter handle = "instrhwinfo('visa','tek') in MATLAB command line

The result should look like this. There should be two elements (one for each instrument) in the ObjectConstructorName cell. If this is not the case, try reconnecting the USB of both instruments.



```
Command Window
>> instrhwinfo('visa')

ans =

    HardwareInfo with properties:

        InstalledAdaptors: {'tek'}
        JarFileVersion: 'Version 3.13'

Access to your hardware may be provided by a support package. Go to the Support Package Installer to learn more.

>> handle = instrhwinfo('visa','tek')

handle =

    HardwareInfo with properties:

        AdaptorDllName: 'C:\Program Files\MATLAB\R2018a\toolbox\instrument\instrumentadaptors\win64\mwtekvisa.dll'
        AdaptorDllVersion: 'Version 3.13'
        AdaptorName: 'TEK'
        AvailableChassis: []
        AvailableSerialPorts: ''
        InstalledBoardIds: []
        ObjectConstructorName: {2x1 cell}
        SerialPorts: ''
        VendorDllName: 'visa32.dll'
        VendorDriverDescription: 'Tektronix VISA Driver'
        VendorDriverVersion: 4.2000

Access to your hardware may be provided by a support package. Go to the Support Package Installer to learn more.
```


- viii) Enter `handle.ObjectConstructorName(1)` and `handle.ObjectConstructorName(2)` to check resource names again.

Response -

```
>> handle.ObjectConstructorName(1)

ans =

1x1 cell array

{'visa('tek', 'USB::0x0699::0x03A0::C011326::INSTR');'}

>> handle.ObjectConstructorName(2)

ans =

1x1 cell array

{'visa('tek', 'USB::0x0699::0x034A::C020626::INSTR');'}
```

- ix) If you do not get these responses or they are inconsistent, it could be that TekVisa is not set as the primary visa. See note in 3.3. a Tektronix Visa
- a) Set Tek Visa as primary and try all this again.
 - b) Remove all other Visas

Problem – Arduino does not connect on MATLAB app launch

- i) Make sure the COM port is correct
 - a) Open device manager
 - b) Find Ports
 - c) Note which port the Arduino is on. i.e., "COM6."
 - d) Open MATLAB App Designer Code View and change the "ArduinoPort" property.
- ii) Upload "FUSF_Hydro_Arduino.ino" to Arduino again.

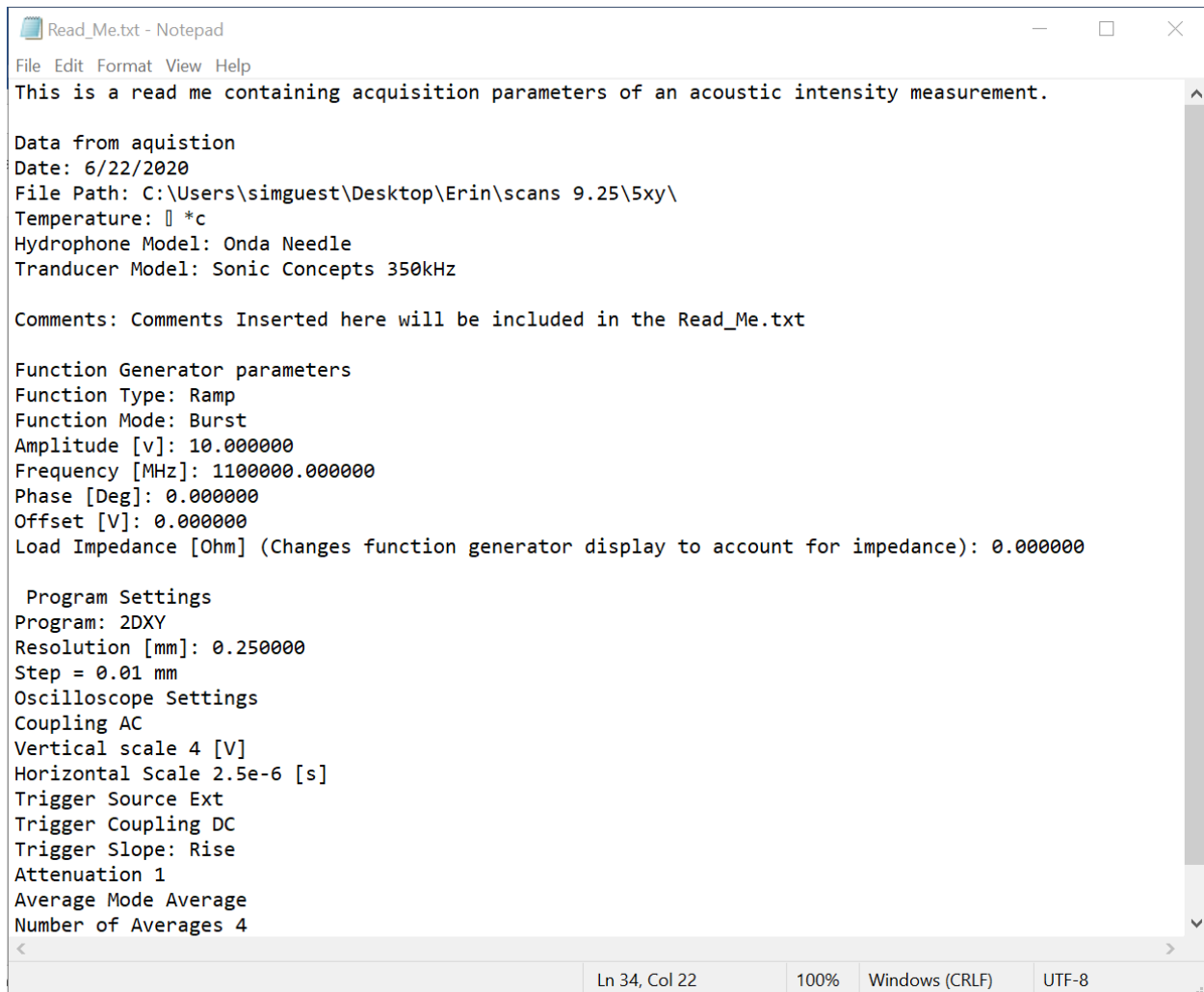
Problem – Oscilloscope visa timeout

- i) If at the start of a scan, make sure the hydrophone is connected to Channel 1.
- ii) During a scan, there is a temporary fix that throws an error if it occurs. We are working on debugging this.

4) ReadMe

A ReadMe.txt containing the scan parameters and information in the ReadMe.txt panel generates at the start of a scan. It is stored in the folder selected for the data. The ReadMe records the settings

displayed in the GUI. If the controls on the function generator and oscilloscope are used instead of the GUI, the ReadMe may record incorrect information.



```
Read_Me.txt - Notepad
File Edit Format View Help
This is a read me containing acquisition parameters of an acoustic intensity measurement.

Data from aquistion
Date: 6/22/2020
File Path: C:\Users\simquest\Desktop\Erin\scans 9.25\5xy\
Temperature:  *c
Hydrophone Model: Onda Needle
Tranducer Model: Sonic Concepts 350kHz

Comments: Comments Inserted here will be included in the Read_Me.txt

Function Generator parameters
Function Type: Ramp
Function Mode: Burst
Amplitude [v]: 10.000000
Frequency [MHz]: 110000.000000
Phase [Deg]: 0.000000
Offset [V]: 0.000000
Load Impedance [Ohm] (Changes function generator display to account for impedance): 0.000000

Program Settings
Program: 2DXY
Resolution [mm]: 0.250000
Step = 0.01 mm
Oscilloscope Settings
Coupling AC
Vertical scale 4 [V]
Horizontal Scale 2.5e-6 [s]
Trigger Source Ext
Trigger Coupling DC
Trigger Slope: Rise
Attenuation 1
Average Mode Average
Number of Averages 4

Ln 34, Col 22    100%    Windows (CRLF)    UTF-8
```

5) Data Output

There are two options for saving the data acquired in a scan—the first stores the data in a MATLAB matrix. The second creates a CSV. The GUI can do one or both. In general, if the scan has a large volume or high resolution, the CSV will be less likely to encounter memory warnings. We provide a MATLAB script to go from CSV back into MATLAB Matrix.

a) MATLAB Matrix

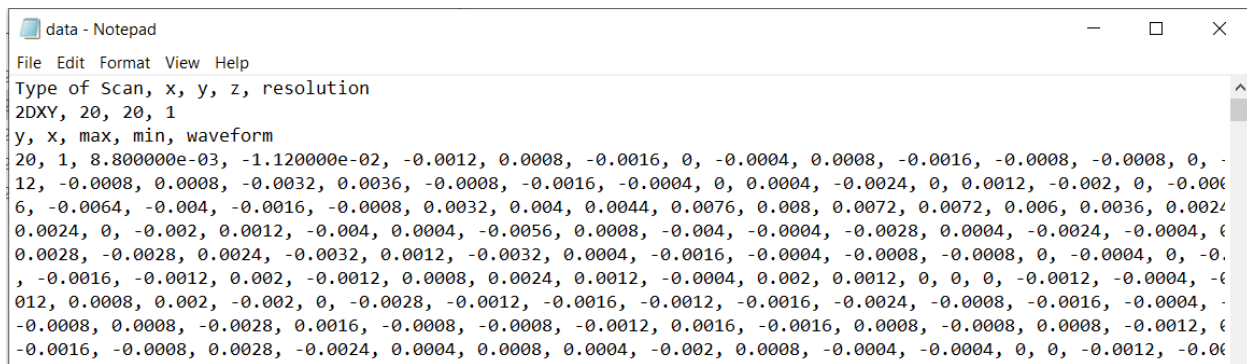
If the scan has n spatial dimensions, then the data saves in the $n+1$ dimension at each pixel. The first element of the $n+1$ dimension is the maximum voltage. The second element is the minimum voltage. The

proceeding 2500 elements are the voltages composing the waveform displayed on the scope. Including the waveform is optional.

For example, a 2d scan will have three dimensions of data. The first plane is a maximum voltage image. The second plane is a minimum voltage image. The remaining 2500 planes store the waveforms.

b) CSV

The CSV has a header that includes the program and scan dimensions. Each line of data starts with the matrix index (as defined in MATLAB), then the maximum and minimum voltages, followed by the waveform. The data can be reconstructed into a MATLAB matrix or other data format. The advantages of saving it in a CSV are that it takes less memory and saves in real-time.



```
data - Notepad
File Edit Format View Help
Type of Scan, x, y, z, resolution
2DXY, 20, 20, 1
y, x, max, min, waveform
20, 1, 8.800000e-03, -1.120000e-02, -0.0012, 0.0008, -0.0016, 0, -0.0004, 0.0008, -0.0016, -0.0008, -0.0008, 0, -
12, -0.0008, 0.0008, -0.0032, 0.0036, -0.0008, -0.0016, -0.0004, 0, 0.0004, -0.0024, 0, 0.0012, -0.002, 0, -0.000
6, -0.0064, -0.004, -0.0016, -0.0008, 0.0032, 0.004, 0.0044, 0.0076, 0.008, 0.0072, 0.0072, 0.006, 0.0036, 0.0024
0.0024, 0, -0.002, 0.0012, -0.004, 0.0004, -0.0056, 0.0008, -0.004, -0.0004, -0.0028, 0.0004, -0.0024, -0.0004, 0
0.0028, -0.0028, 0.0024, -0.0032, 0.0012, -0.0032, 0.0004, -0.0016, -0.0004, -0.0008, -0.0008, 0, -0.0004, 0, -0.
, -0.0016, -0.0012, 0.002, -0.0012, 0.0008, 0.0024, 0.0012, -0.0004, 0.002, 0.0012, 0, 0, 0, -0.0012, -0.0004, -0
012, 0.0008, 0.002, -0.002, 0, -0.0028, -0.0012, -0.0016, -0.0012, -0.0016, -0.0024, -0.0008, -0.0016, -0.0004, -
-0.0008, 0.0008, -0.0028, 0.0016, -0.0008, -0.0008, -0.0012, 0.0016, -0.0016, 0.0008, -0.0008, 0.0008, -0.0012, 0
-0.0016, -0.0008, 0.0028, -0.0024, 0.0004, 0.0008, 0.0004, -0.002, 0.0008, -0.0004, -0.0004, 0, 0, -0.0012, -0.000
```

6) SCPI Commands

MATLAB communicates with the instrumentation through SCPI Commands (Standard Commands for Programmable Instruments), which are ASCII textual strings. The instrument understands specific keywords and parameters defined by the device manufacturer. These are in the programming manuals. The GitHub project includes the relevant programming manuals for the Tektronix TDS 2001C and Tektronix AFG3022C.

Your oscilloscope or function generator will likely not be fully compatible with the SCPI commands implemented. However, modifying the MATLAB code should be a relatively simple process. The Tektronix commands and functions used by the GUI are below.

To modify the program, find corresponding commands in the programming manual for your device and replace the App Designer code view's strings. We would appreciate each group sharing their work such that the project can be compatible with as many devices as possible.

The next edition of the software will transition towards a more object-oriented structure. The MATLAB GUI will create a scope/generator object that inherits a class containing the relevant SCPI commands. Then each group can create a class for their specific instrument.

a) Oscilloscope – Tektronix TDS2001C

Compatible with – Tektronix TDS200, TDS1000/TDS2000, TDS1000B/TDS2000B, TDS1000C-EDU/TDS2000C, and TPS2000/TPS2000B Series Digital Oscilloscopes.

Command	Function
*ESR? (Query Only)	Returns the contents of the Standard Event Status Register (SESR). *ESR? also clears the SESR (since reading the SESR clears it). (See page 3-1, <i>Status and Events</i> .)
*RST	(Reset) Returns the oscilloscope to a known set of oscilloscope settings, but does not purge any stored settings. This command executes a subset of the FACTory command.
*CLS	The *CLS command clears the following oscilloscope status data structures: The Event Queue, The Standard Event Status Register (SESR), The Status Byte Register (except the MAV bit)
CURVE?	<p>Transfers oscilloscope waveform data to and from the oscilloscope in binary or ASCII format. Each waveform that is transferred has an associated waveform preamble that contains information such as data format and scale. For information about the waveform preamble, refer to WFMPre?. The data format is specified by the DATA:ENCdg and DATA:WIDth commands.</p> <p>The CURVe? query sends data from the oscilloscope to an external device. The data source is specified by the DATA:SOUrce command. The first and last data points that are transferred are specified by the DATA:STARt and DATA:STOP commands.</p>
CH1:VOLTs %s	Sets or queries the vertical gain of the specified channel. The value of <x> can vary from 1 through 4 for 4-channel instruments or 1 through 2 for 2-channel instruments.
WFMPre:YOFF?	<p>YOff is a value, expressed in digitizer levels, used to convert waveform record values to YUNit values using the following formula (where dl is digitizer levels): $\text{value_in_YUNits} = ((\text{curve_in_dl} - \text{YOFF_in_dl}) * \text{YMuLt}) + \text{YZERO_in_YUNits}$</p> <p>The set form of this command stores a value for the reference waveform specified by the DATA:DESTination command. This value does not affect how the oscilloscope displays the waveform but does affect the cursor readouts.</p> <p>The query form returns a value for the waveform specified by the DATA:SOUrce command in digitizer levels if that waveform is active or displayed. If that waveform is not active or displayed, the query fails, and the oscilloscope generates an execution error with event code 2244 (waveform requested is not active).</p>
WFMPRE:YMuLT?	YMuLt is a value, expressed in YUNits per digitizer level, used to convert waveform record values to YUNit values using the following formula (where dl is digitizer levels):

	$\text{value_in_YUNits} = ((\text{curve_in_dl} - \text{YOFF_in_dl}) * \text{YMULt}) + \text{YZERO_in_YUNits}$ <p>The set form of this command sets the vertical scale factor of the reference waveform specified by the DATA:DESTination command, expressed in YUNits per digitizing level.</p> <p>The query form returns a value for the waveform specified by the DATA:SOURce command if that waveform is active or displayed. If that waveform is not active or displayed, the query fails, and the oscilloscope generates an execution error with event code 2244 (waveform requested is not active).</p>
CH1:Scale?	Sets or queries the vertical gain of the specified oscilloscope channel. The value of <x> can vary from 1 through 4 for 4-channel instruments or 1 through 2 for 2-channel instruments.
DATA:ENCDG RIBINARY;WIDTH 1	<p>Two commands separated by ';'.</p> <p>First - Sets or queries the format of the waveform data. This command is equivalent to setting WFMPRe:ENCdg, WFMPRe:BN_Fmt, and WFMPRe:BYT_Or. (See Table 2-31.)</p> <p>Setting the DATA:ENCdg value causes the corresponding WFMPRe values to update. Setting the WFMPRe value causes the corresponding DATA:ENCdg values to update.</p> <p>Second - Sets the number of bytes per waveform data point to be transferred when executing the CURVe command. (Changing DATA:WIDth may change the following WFMPRe parameters: BIT_Nr, BYT_Nr, YMULt, YOFF, and YZEro.)</p>
*IDN?	Returns the oscilloscope identification code in IEEE 488.2 notation.
CH1:Coupling %s	<p>Sets or queries the input attenuator coupling setting of the specified oscilloscope channel. The value of <x> can vary from 1 through 4 for 4-channel instruments or 1 through 2 for 2-channel instruments.</p> <p>This command is equivalent to setting the Coupling option in the Vertical menu.</p>
CH1:Position 0	<p>Sets or queries the vertical position of the specified oscilloscope channel. The value of <x> can vary from 1 through 4 for 4-channel instruments or 1 through 2 for 2-channel instruments.</p> <p>The position voltage value is applied to the signal before digitization. This command is equivalent to adjusting the front-panel VERTICAL POSITION knob.</p>
DATA:Source CH1	Sets or queries which waveform will be transferred from the oscilloscope by the CURVe, WFMPRe, or WAVFrm? Queries. You can transfer only one waveform at a time.
HOR:SCALE %s	Sets the time per division for the main time base and is identical to the HORizontal:MAIn:SCALE command. It is included for compatibility purposes.

Hor:Scale?	Query form of HOR:Scale
HORizontal:MAIn:POSition %s	Sets or queries the main timebase horizontal position. This command is equivalent to adjusting the Horizontal Position when Main is selected from the Horizontal menu.
MEASU:Immed:Source CH1	Sets or queries the source for single-source immediate measurements.
ACQUIRE:MODE %s	<p>Sets or queries the oscilloscope acquisition mode. This affects all live waveforms and is equivalent to setting the Mode option in the Acquire menu.</p> <p>Waveforms are the displayed data point values taken from acquisition intervals. Each acquisition interval represents a time duration determined by the horizontal scale (time per division).</p> <p>The oscilloscope sampling system can operate at a rate greater than that indicated by the horizontal scale. Therefore, an acquisition interval can include more than one sample.</p> <p>The acquisition mode set using the ACQUIRE:MODE command determines how the acquisition interval's final value is generated from the many data samples.</p>
ACQUIRE:NUMAVg %s	Sets the number of oscilloscope waveform acquisitions that make up an averaged waveform. This command is equivalent to setting the Averages option in the Acquire menu.
TRIG:MAIN:EDGE:SOURCE %s	Sets or queries the source for the edge trigger. This is equivalent to setting the Source option in the Trigger menu.
TRIG:MAIN:EDGE:COUPLING %s	Sets or queries the type of coupling for the edge trigger. This is equivalent to setting the Coupling option in the Trigger menu.
TRIG:MAIN:EDGE:Slope %s	Selects a rising or falling slope for the edge trigger. This is equivalent to setting the Slope option in the Trigger menu.

**Function descriptions are taken directly from the Tektronix Programming Manual found in the GitHub project.*

b) Function Generator – Tektronix AFG3000 Series

Compatible with AFG3000 Series Arbitrary Function Generators.

Command	Function
'OUTPUT OFF'	Returns the contents of the Standard Event Status Register (SESR). *ESR? also clears the SESR (since reading the SESR clears it). (See page 3-1, <i>Status and Events</i> .)
*IDN?	This query-only command returns identification information on the arbitrary function generator.
*RST	This command resets the instrument to the factory default settings. This command is equivalent to pushing the Default button on the front panel. The default values are listed in Default Settings.
*CLS	This command clears all the event registers and queues used in the arbitrary function generator status and event reporting system.
BURST:MODE TRIG	This command sets or queries the burst mode for the specified channel.
BURST:NCycles %f	This command sets or queries the number of cycles (burst count) to be output in burst mode for the specified channel. The query command returns 9.9E+37 if the burst count is set to INFINITY.
BURST:TDelay %f ms	This command sets or queries delay time in the burst mode for the specified channel. It specifies a time delay between the trigger and the signal output. This command is available only in the Triggered burst mode.
BURST:STAT ON	This command enables or disables the burst mode for the specified channel. The query command returns the state of burst mode.
TRIGGER:TIMER %f ms	This command sets or queries an internal clock period when selecting the internal clock as the trigger source with the TRIGger[:SEquence]:SOURce command. The setting range is 1 μ s to 500.0 s.
FUNCTION %s	This command sets the function type.
VOLT %d	This command sets or queries the output amplitude for the specified channel.
VOLTAGE:OFFSET %f v	This command sets or queries the offset level for the specified channel.
Voltage:Offset?	Query form of voltage:offset
FREQUENCY %f	This command sets the function frequency.
Freq?	Query form of 'Frequency.'
PHASE %f	This command sets the function phase
OUTPUT ON	This command turns the channel on.
SYST:ERR?	This query-only command returns the contents of the Error/Event queue.

Function Descriptions are taken from Tektronix AFG3022C programming manual found in the GitHub project.

c) Arduino

The Arduino acts as a secondary to the MATLAB primary and receives ASCII commands. In MATLAB, that is just a string. Sending 'H' runs the home function in the Arduino script. Sending an 'x-

10' moves the X-Axis in the negative direction for ten steps. A third example, 'y100', moves the y axis 100 steps in the positive direction. Note that one step is 0.01 mm if the same stepper motors and leadscrews are used.

*Note – these commands may be modified in 'FUSF_Hydro_Arduino.ino'

Command	Function
H	Hard homes the device
x(direction)(number of steps)	Move device (number of steps) in the x-axis (direction) where the direction is '-' for negative and blank for positive.
y(direction)(number of steps)	Move device (number of steps) in y-axis in (direction) where the direction is '-' for negative and blank for positive.
z(direction)(number of steps)	Move device (number of steps) in z-axis in (direction) where the direction is '-' for negative and blank for positive.