

Initial Device Assembly Instructions Manual v1.0

Design of an Electromyographic Switch
for Communication System Access
Version 1.0

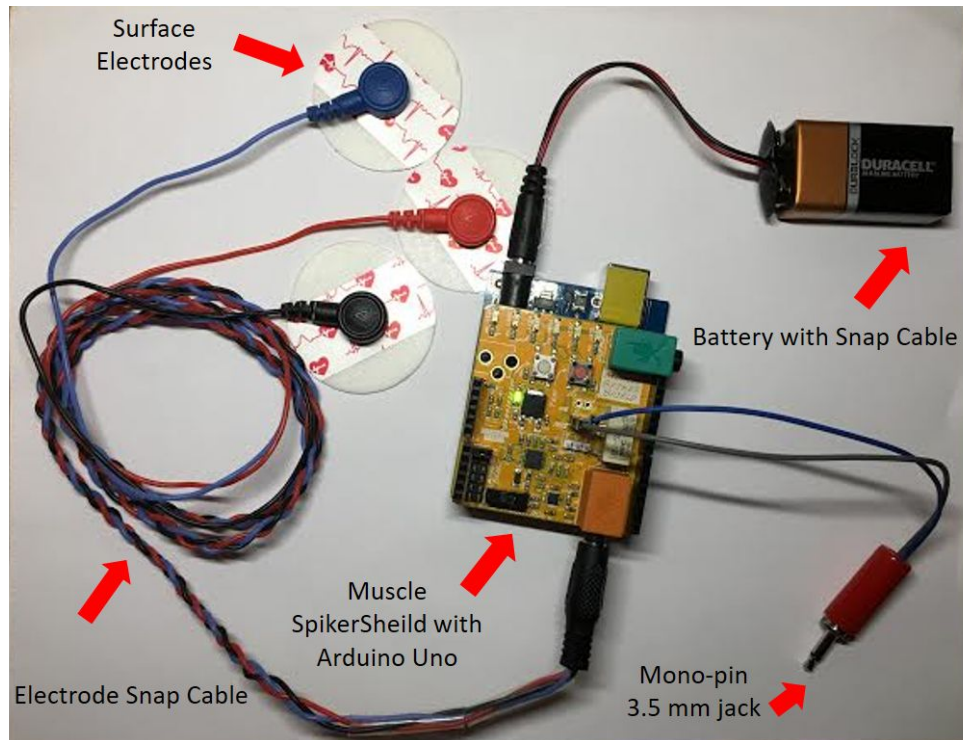


TABLE OF CONTENTS

Contents	Pg. #
Basic Overview	3
Ordering the Parts	9
Hardware Setup	11
Software Setup	20
Housing Setup	29
Device Assembly	31
Appendix	36

BASIC OVERVIEW

This instructions manual will detail the initial assembly of a surface electromyographic (sEMG)-based switch. This device is comprised of 7 basic components which are listed and defined below, and labeled in the figure below.



BASIC OVERVIEW *CONTINUED*

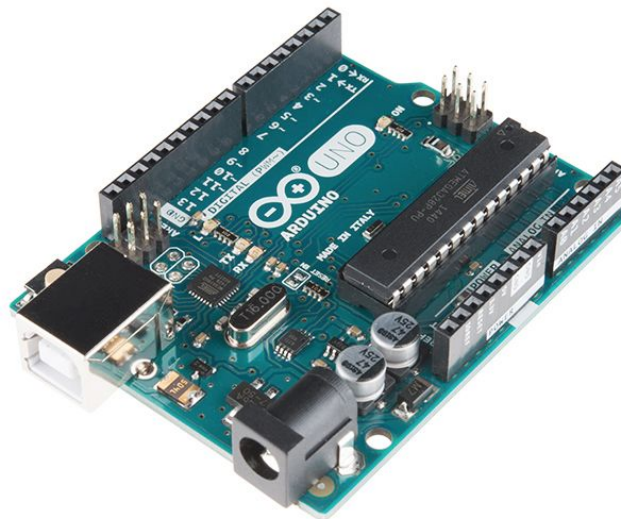
1. Surface electrodes

Surface electrodes are small, conductive devices that are attached to the skin with an adhesive to measure electrical activity in the muscle tissue beneath it.



2. Arduino UNO

Arduino UNO is a small computer that can be easily programmed and used to power and control other pieces of hardware like motors or LED lights. It is generally used for prototyping and open source projects.



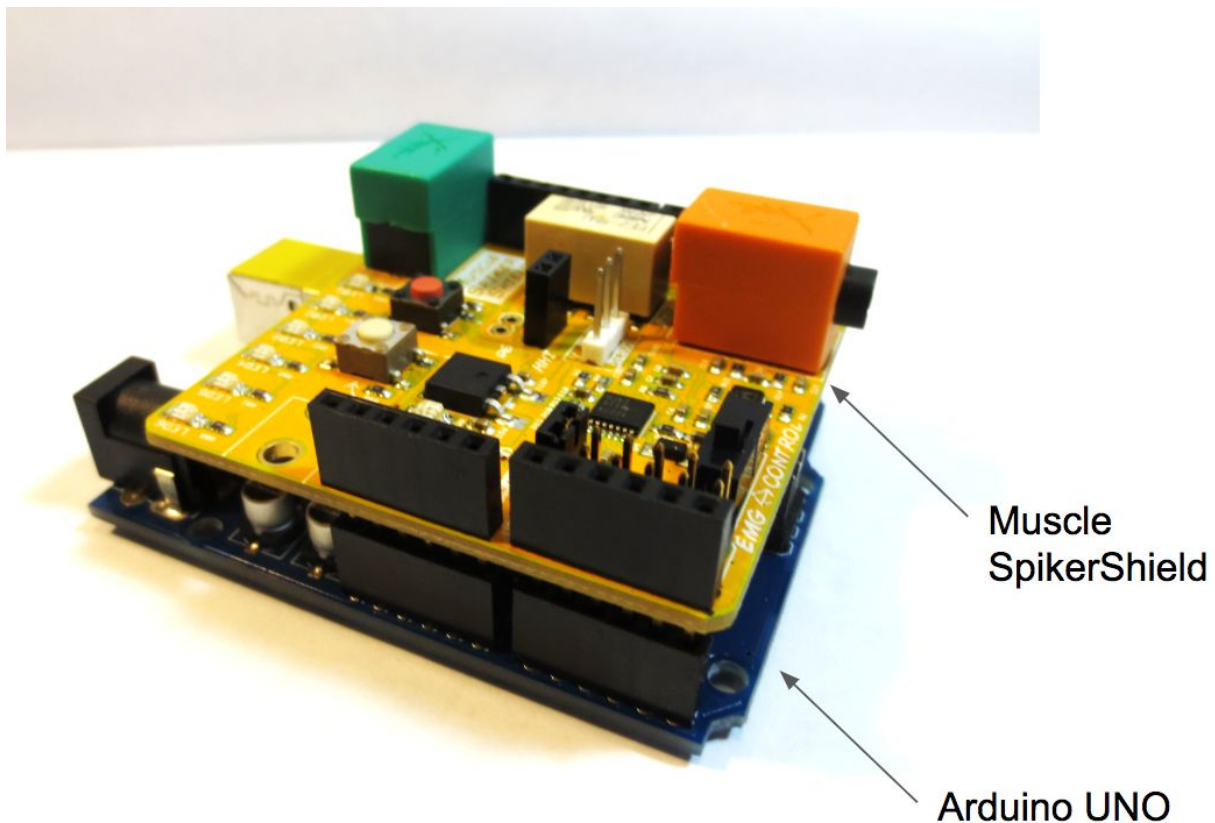
1

¹ Image of Arduino UNO taken from Sparkfun.com

BASIC OVERVIEW *CONTINUED*

3. Muscle SpikerShield

The Muscle SpikerShield is a piece of hardware which is fixed on top of the Arduino UNO (as pictured below). It is used to extend the capabilities of the Arduino UNO so that it can receive the electrode cable and generate the output for a mono-pin 3.5 mm audio jack cable. It also has a row of six colored LED lights--these LEDs are used for calibrating the device and for showing the level of muscle activity during use.

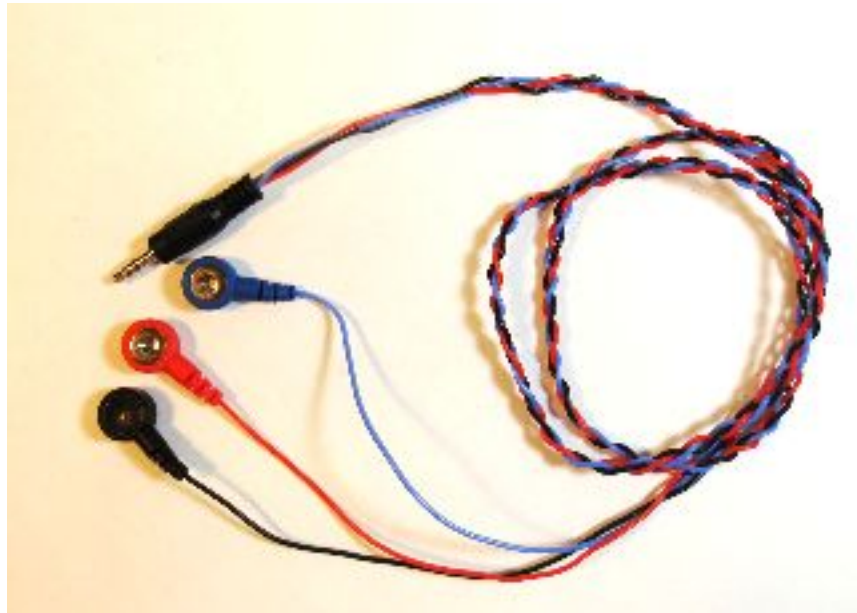


BASIC OVERVIEW *CONTINUED*

4. Electrode snap cable

One end of the electrode snap cable connects to surface electrodes, and the other is a 3.5 mm jack connector to transmit the electrical activity from the surface electrodes to another device. The black cable attaches to the grounded electrode, while the red and blue cable attach to the surface electrodes oriented over the active muscle group.

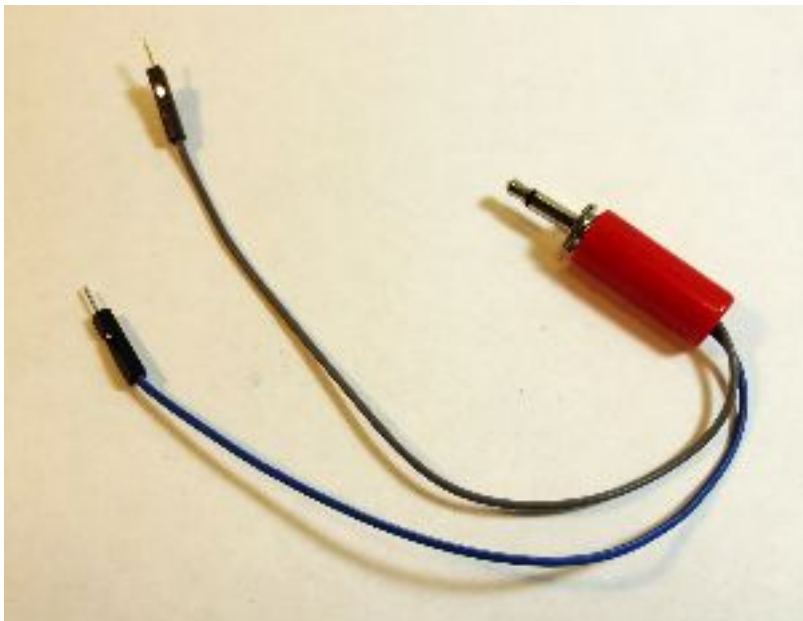
For this device, the 3.5 mm jack connector of the cable will plug into the orange box on the Muscle SpikerShield hardware.



BASIC OVERVIEW *CONTINUED*

5. Mono-pin 3.5 mm audio jack cable

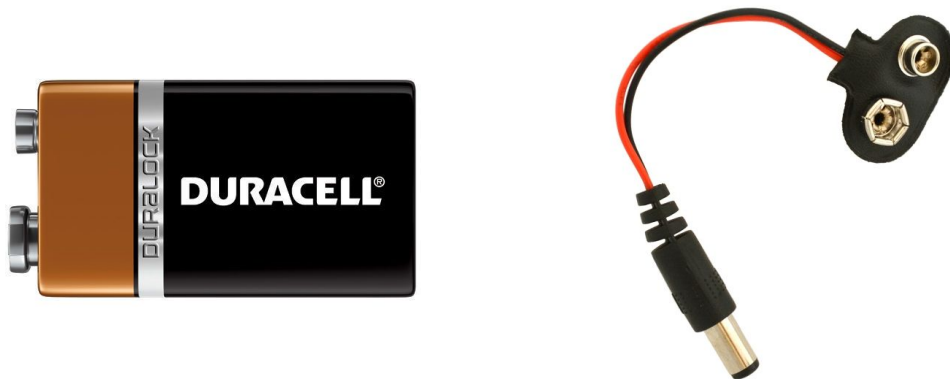
The mono-pin 3.5 mm audio jack cable connects the hardware of the sEMG based switch to the Alternative and Augmentative Communication (AAC) system. The mono-pin 3.5 mm audio jack end connects to the AAC system, and the other end is positive and negative leads which plug into a black relay port on the Muscle SpikerShield hardware.



BASIC OVERVIEW *CONTINUED*

6. 9V battery and battery snap

The 9V battery is used to power the device. It is connected to the hardware with a battery snap. The recommended battery snap plugs into the 2.1 mm black power port on the Arduino UNO. The battery will be connected to the snap upon assembly.



2

7. 28/24 AWG to USB cable

The 28/24 AWG to USB cable connects the Arduino UNO to a computer during the Arduino Software upload required for an initial setup of the device.



3

² Image of 9V battery taken from 1000Bulbs.com, image of battery snap taken from BananaRobotics.com

³ Image of 28/24 AWG to USB cable taken from Adafruit.com

ORDERING THE PARTS

Each component described in the “Basic Overview” section can be ordered from the sources listed below. Follow the links in the “Link to Ordering Site” column. The hospital, clinic or location of care may already have some of the components listed; feel free to use personal or pre-owned components where applicable.

DESCRIPTION	QTY	UNIT COST	LINK TO ORDERING SITE
Muscle SpikerShield and Arduino UNO Bundle	1	\$149.99	https://backyardbrains.com/products/muscleSpikershieldBundle
Disposable Surface Electrodes	1 bag (100 electrodes)	\$19.50	https://www.amazon.com/Universal-ECG-EKG-Electrodes-100pcs/dp/B004S8H4YA/ref=pd_lpo_vtph_121_lpt_4?encoding=UTF8&psc=1&refRID=4GZ4CSSVH736WCRXC2H3
Electrode Cable	1	\$4.95	https://www.sparkfun.com/products/12970
9V Battery	1	\$4.19	https://www.radioshack.com/products/radioshack-9v-alkaline-battery
Battery Snaps	1	\$1.56	https://www.radioshack.com/products/radioshack-fully-insulated-9v-battery-snap-connectors?gclid=CLuxi8WMytMCFUxYDQodzqkGg
3.5 mm Jack	1	\$1.40	https://www.radioshack.com/products/radioshack-1-8-phone-plug-2-pack
Male Pin Jumper Cable Wires for 3.5 mm Jack	1 bag (40 jumper cable wires)	\$6.28	https://www.amazon.com/uxcell-Female-Jumper-Cable-Wires/dp/B00D7SDDLX
Housing	1	\$20 (estimated cost using an in-house 3D printer)	It is recommended that if the hospital has its own in-house 3D printer, it should be used. Follow the instructions in “HOUSING SETUP” for this.

			<p>If the hospital does not have a 3D printer, the housing may be ordered through a 3D printing service. One of the services below may be used:</p> <p>https://www.shapeways.com/</p> <p>https://www.makexyz.com/</p> <p>https://www.ponoko.com/</p> <p>https://www.you3dit.com/</p>
--	--	--	---

HARDWARE SETUP

The hardware setup requires the use of a soldering iron in order to modify components for assembly. Therefore, a basic knowledge of soldering is required. The only hardware component that requires soldering is the mono-pin 3.5mm audio jack. The instructions below are written for those who have soldered, but have little experience nor expertise.

The hardware setup steps do not need to be conducted in the given order. The hardware setup steps are listed below:

1. Mono-Pin 3.5mm Audio Jack Setup
2. Electrode Snap Cable Braiding
3. Raw Mode Setup

HARDWARE SETUP *CONTINUED*

MONO-PIN 3.5MM AUDIO JACK SETUP

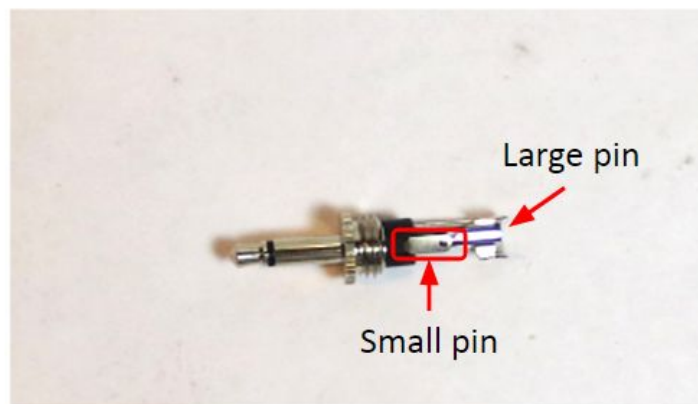
The mono-pin audio jack must be modified such that it can input to any AAC device. This requires:

- **2 x 8 inch wires with male pin connections**
- **Wire strippers**
- **Soldering iron and solder**
- **Mono-Pin 3.5mm audio jack**
- **Clamps for positioning**

Follow the soldering instructions below, or follow along with a video that explains each step:

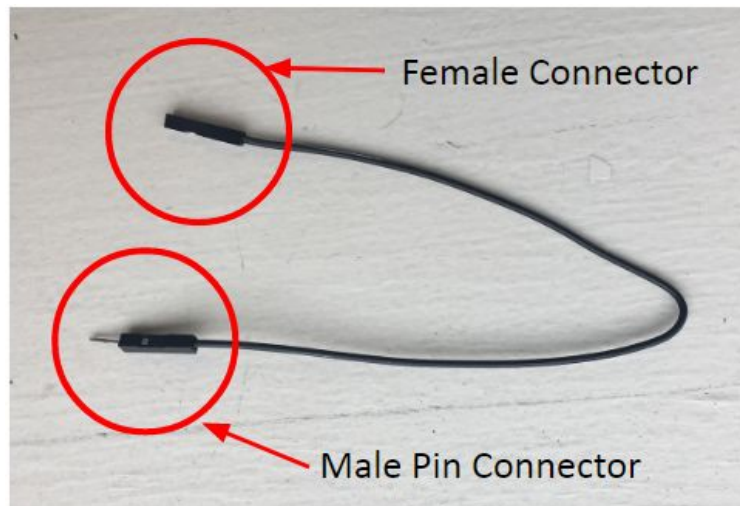
https://youtu.be/lvSt_wtRRs0.

1. Unscrew the plastic coating of the mono-pin jack. Two metal bridges should be exposed--these are called pins. Pins are used to connect electrical hardware, like wires. There is a small pin with a small hole towards the rounded end. There is a large pin that is rounded with a hole in the middle and two wings at the end.



HARDWARE SETUP: MONO-PIN 3.5MM AUDIO JACK SETUP *CONTINUED*

2. The wires with male pin connectors (i.e. a metal pin protruding from the wire) must be cut. Cut the end of the wire right below the other end of the wire (not the male end).



In this image, one end of the wire is a female connector, and the other is a male connector.



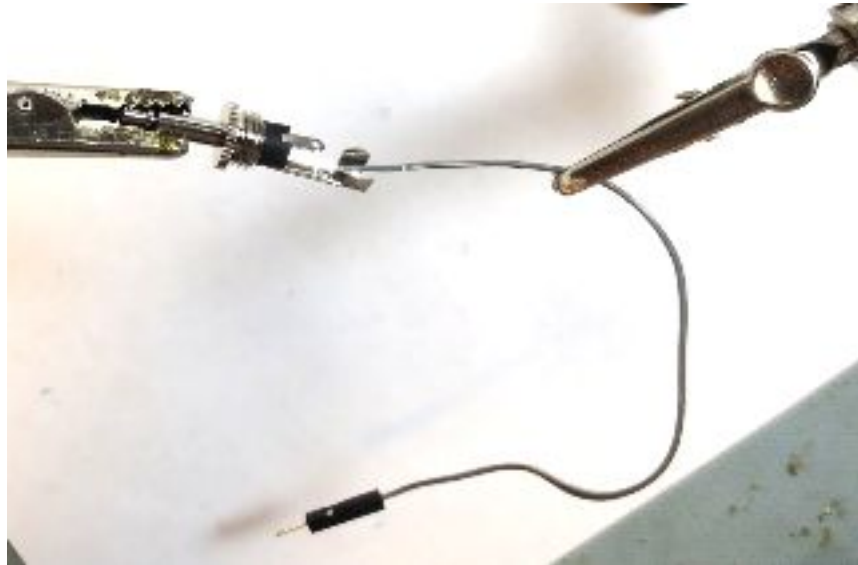
HARDWARE SETUP: MONO-PIN 3.5MM AUDIO JACK SETUP *CONTINUED*

3. Strip the ends of the pin-equipped wires by clamping the wire strippers on the wire and then pull away from the wire. The rubber part should come off, and about 0.5 inches of wire should be exposed. These wires can be any color, as the ordering of the insertion of the pins for device output does not affect the output.



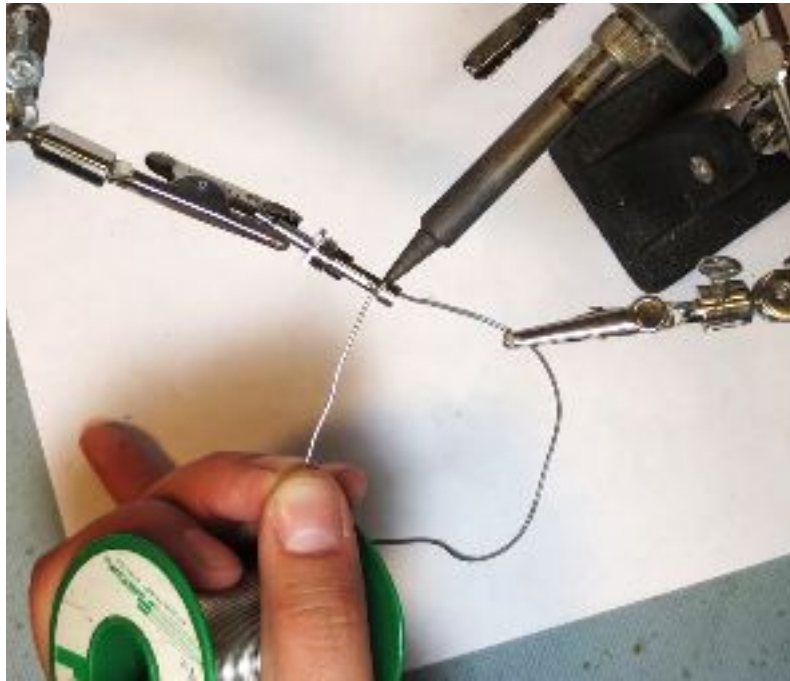
HARDWARE SETUP: MONO-PIN 3.5MM AUDIO JACK SETUP *CONTINUED*

4. Place wires flat on a heat resistant surface, or hold them in place with soldering clamps, as shown. Position the exposed end of the pin-equipped wire with one of the metal pins of the audio jack and loop the wire through the hole. Ensure that the wire does not come in contact with the other pin or other stripped wire.



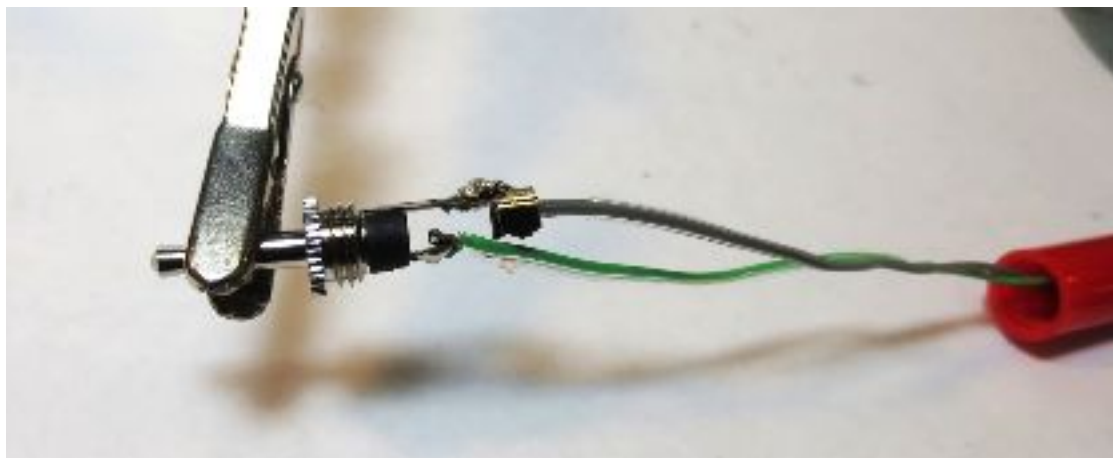
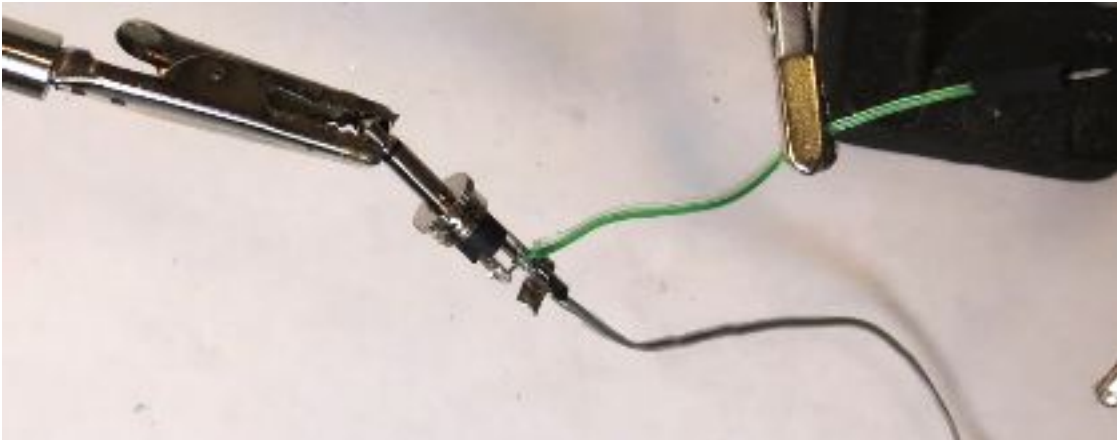
HARDWARE SETUP: MONO-PIN 3.5MM AUDIO JACK SETUP *CONTINUED*

5. Solder the exposed pin-equipped wire with the pin of the audio jack together. Let cool.

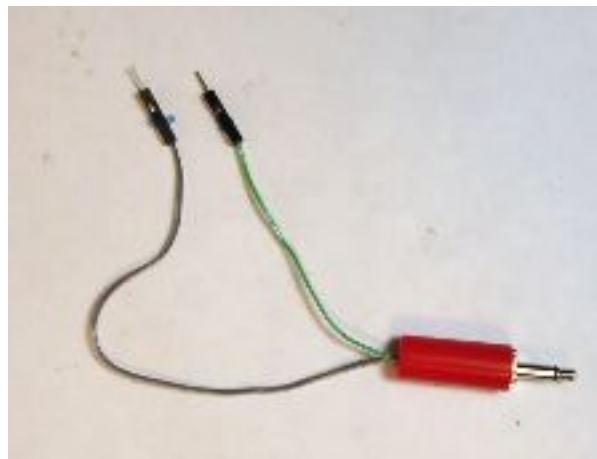


HARDWARE SETUP: MONO-PIN 3.5MM AUDIO JACK SETUP *CONTINUED*

6. Position the exposed end of the pin-equipped wire with the remaining metal pin of the audio jack and loop the wire through the hole. Solder the wire to the remaining pin of the audio jack. Let cool.



7. Re-affix the audio jack cover to the pin.



HARDWARE SETUP *CONTINUED*

ELECTRODE SNAP CABLE BRAIDING

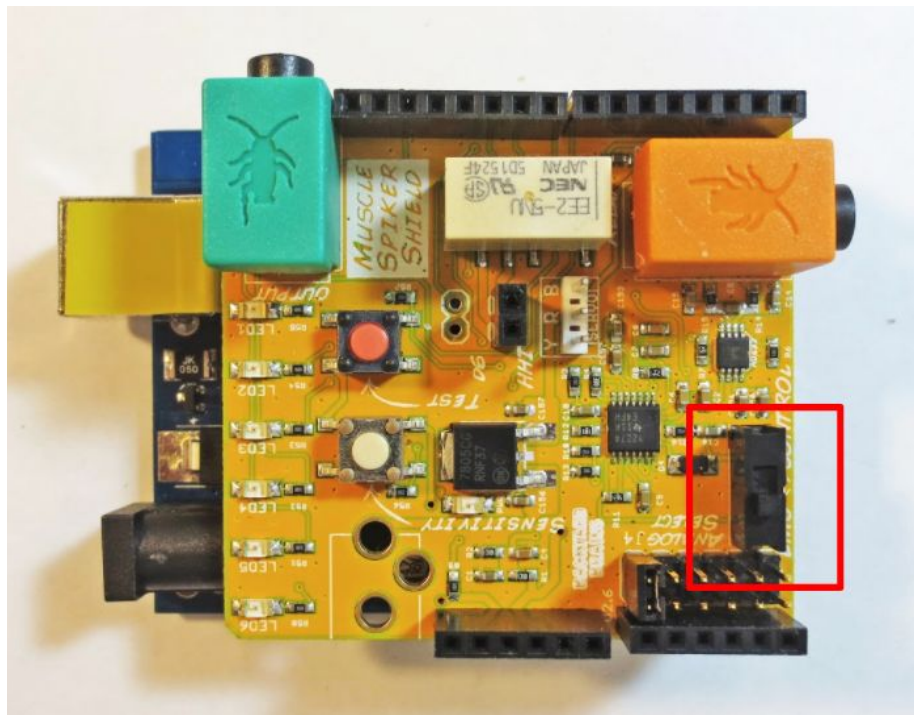
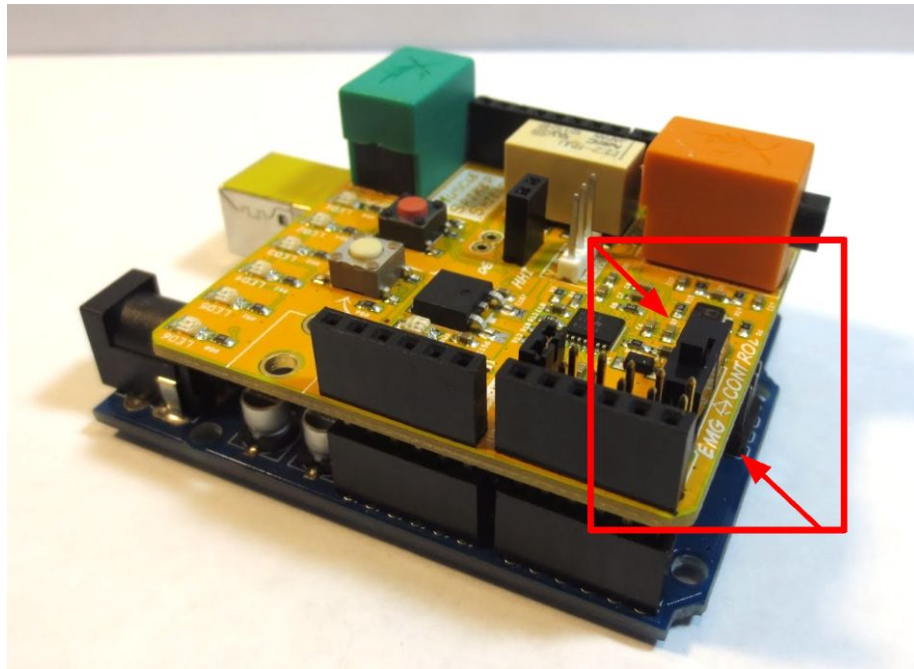
The electrode snap cable must be prepared by braiding the three wires (red, blue and black) together. Braid starting at the base of the wires at the 3.5 mm output jack end, and continue to braid until approximately 8-10 inches from the electrode snap connector ends of the wires. This is done to reduce noise in the signal from the cables.



HARDWARE SETUP *CONTINUED*

RAW MODE SETUP

There is a switch on the Muscle SpikerShield which controls two settings of the signal processing, “EMG” or “CONTROL.” This switch must be set to **EMG** as shown below.



SOFTWARE SETUP

In order to prepare the device to function as a sEMG-based switch for AAC system access, the device's Arduino Software program must be uploaded to the hardware through a series of several computer-dependent steps. These must be performed in order:

1. Download Arduino IDE
2. Download and setup the program
3. Upload to the hardware
4. Troubleshooting

DOWNLOAD ARDUINO IDE

A laptop or desktop computer with Arduino Integrated Development Environment (IDE) is required in order to implement the provided Arduino Software on the microcontroller.

Go to www.arduino.cc/en/Main/Software. Select and download the 'Arduino Desktop IDE' for the operating system and then follow Arduino's instructions for download. Apple computers run Mac OS, PCs run Windows, the IDE must match the user's operating system for success.



SOFTWARE SETUP *CONTINUED*

DOWNLOAD AND SETUP THE PROGRAM

Download the Program

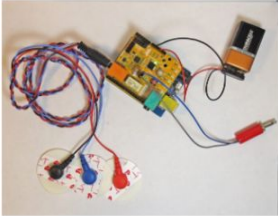
1. Go to: <http://sites.bu.edu/stepplab/research/semg-based-switch-design/>
2. Download the .ZIP file by clicking the embedded link in “Download the Software Setup Files here: sEMG_Switch_Program_Files” and save anywhere on the computer. Extract the files as detailed below.

BU College of Health & Rehabilitation Sciences: Sargent College
STEPP Lab for Sensorimotor Rehabilitation Engineering

RESEARCH VOLUNTEER LAB MEMBERS COLLABORATORS LAB ALUMNI PUBLICATIONS NEWS SUPPORT US CONTACT

sEMG-Based Switch Design

Design of an Electromyographic Switch for Communication System Access



Augmentative and alternative communication (AAC) systems provide patients with specialized methods to communicate when oral communication is not possible. However, some patients have insufficient motor control to access current AAC systems (e.g., via head-tracker, eye-tracker, button-style mechanical switch). This project delivers a device that directly replaces a mechanical switch for AAC operation by capturing muscle activity via surface electromyography (sEMG). It serves the subcategory of patients who possess volitional control over one or more muscle groups but lack the strength or coordination to activate a mechanical switch. The device uses the signal processing capabilities of an Arduino UNO equipped with sEMG-capable hardware and Arduino Software to amplify, filter, and smooth the signal from a patient's voluntary muscle activity. A patient-specific threshold is used to determine the state of the switch and produce a binary output that matches that of an existing mechanical switch. This complete sEMG-based switch is provided with full documentation detailing instructions for sensor placement over any volitionally-controlled muscle group, calibration, operation, and maintenance procedures. The device design is modular and replicable with commercially-available parts and open-source software for use by any hospital or caregiver serving the target population.

The Stepp Lab for Sensorimotor Rehabilitation Engineering at Boston University and the Madonna Rehabilitation Hospital in Lincoln, Nebraska developed and supported this project. The device design and development were completed as a part of the Senior Design Capstone Project for an interdisciplinary team of Biomedical and Mechanical Engineering students: Victoria Frick (BME), Katherine Grouard (BME), and Evi Shairolas (ME).

Click here to access the bill of materials, [setup and maintenance instructions](#), and recommended [procedures for calibration and operation](#) of the switch for communication system access.

Download the Software Setup Files here: [sEMG_Switch_Program_Files](#)

Research

- VoiceTx
- sEMG-Based Switch Design
- Volunteer
- Lab Members
- Collaborators
- Lab Alumni
- Publications
- News
- Support us
- Contact

News

- [2017 Boston Speech Motor Control Mini-Symposium](#)
- [Undergraduate researchers receive Summer 2017 UROP Awards!](#)

Links

- [BU College of Health & Rehabilitation Sciences, Sargent College](#)
- [BU Department Of Biomedical Engineering](#)
- [BU Graduate Program for Neuroscience](#)
- [BU Sargent College Department of Speech, Language & Hearing Sciences](#)

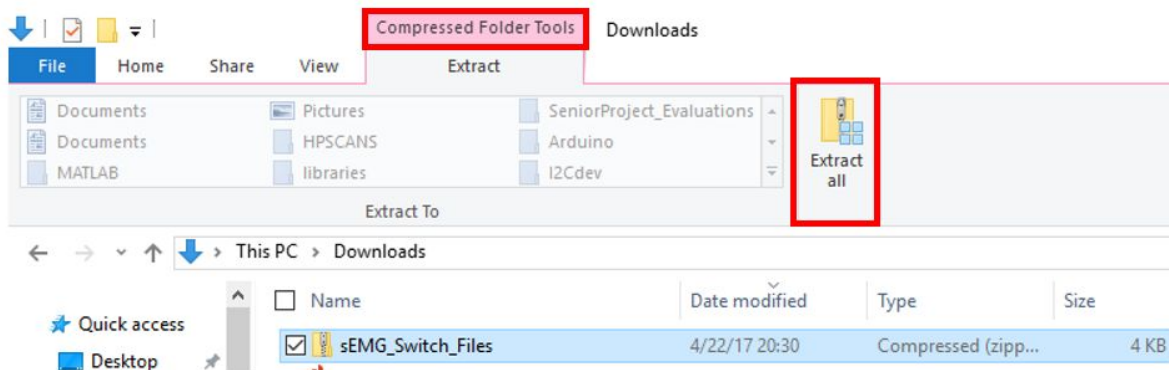
The information to program the switch is available for download in a .ZIP file, a type of compressed folder. To open this folder, extract the files as detailed below for **both** Windows (first) or Mac (second) operating systems.

SOFTWARE SETUP: DOWNLOAD AND SETUP THE PROGRAM *CONTINUED*

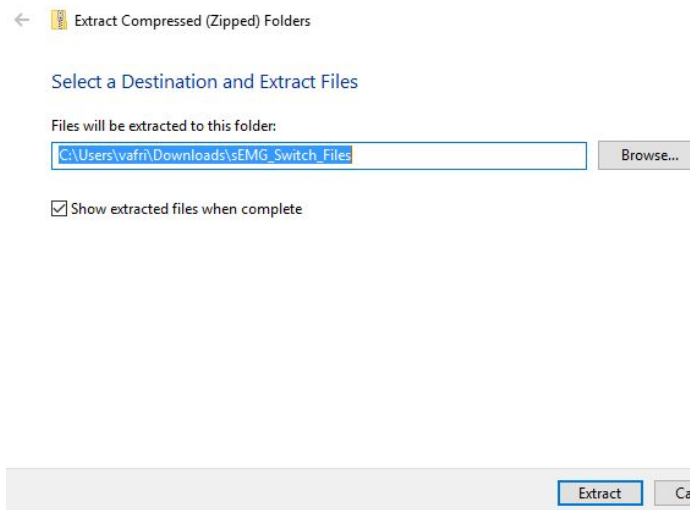
Windows Users:

Follow these instructions if using a Windows operating system. If not, move on to the next section: “Mac OS Users.”

1. Download the .ZIP file. Select the file. The Compressed Folder Tools tab will appear. Click extract all.



The following window will appear. Select “Extract”.



Files can now be selected individually for use.

<input type="checkbox"/> Name	Date modified	Type	Size
sEMG_Switch_Files	4/22/17 20:35	File folder	
sEMG_Switch_Program	4/22/17 20:29	Text Document	13 KB

SOFTWARE SETUP: DOWNLOAD AND SETUP THE PROGRAM *CONTINUED*

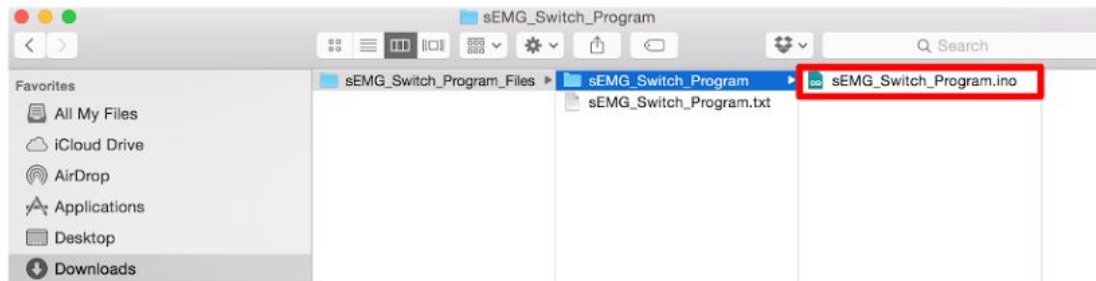
Mac OS Users:

Follow these instructions if using a Mac operating system.

1. Download the .ZIP file. Select the file in the bottom lefthand corner of the browser.



2. The .ZIP file will appear in the "Downloads" folder. Files can now be selected individually for use.



SOFTWARE SETUP *CONTINUED*

UPLOAD TO THE HARDWARE

1. Locate the “sEMG_Switch_Program” folder from the downloaded, unzipped program files, “sEMG_Switch_Files.” It should contain a “sEMG_Switch_Program.ino” file.
2. Open the .ino file--this will open the program in Arduino as shown below. If the .ino file does not open, refer to the Troubleshooting section in the appendix of this document.



```
/*-----  
 * sEMG Switch for Communication System Access  
 *  
 * Written by Katherine Girouard, Evi Shiakolas and Victoria Frick  
 * As a part of Boston University's Senior Design Project in Engineering  
 * In collaboration with the Stepp Lab for Sensorimotor Rehabilitation  
 * and  
 * Madonna Rehabilitation Hospital in Lincoln, NE  
 *  
 * Adapted from Backyard Brains 2015 Muscle SpikerShield Arduino UNO Code  
 *  
 * Code monitors amplitude of EMG, applies a running sum envelope,  
 * displays EMG strength on LED bar, and produces (ON/OFF) signal relay output  
 * on the Muscle SpikerShield board based on strength and thresholding of EMG signal.  
 * Input amplitude can be calibrated to various strengths of users with sensitivity setting.  
 *  
 * Written for and tested with Muscle SpikerShield V2.6  
 * -----*/  
  
#define NUMBER_OF_LEDS 6           //Number of LEDs in LED bar  
#define OUTPUT_PIN 2              //Header output pin  
#define SENSITIVITY_BUTTON_PIN_DEC 7 //Pin for WHITE button that controls sensitivity decrease  
#define SENSITIVITY_BUTTON_PIN_INC 4 //Pin for RED button that controls sensitivity increase  
#define NUMBER_OF_SENS 18         //Number of sensitivity values  
#define INDICATOR_LED_PIN 13       //Pin for LED that indicates state of output  
  
#define RELAY_PIN 2                //PIN FOR RELAY OUTPUT
```


SOFTWARE SETUP: UPLOAD TO THE HARDWARE *CONTINUED*

3. Compile and save the code in Arduino by clicking the checkmark icon in the upper left of corner of the screen.

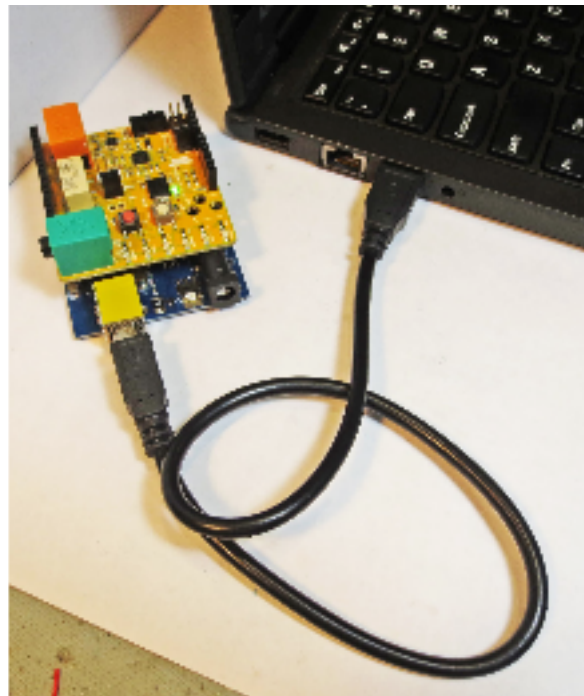


Compiling checks that the code is error-free and ready to be uploaded to the hardware. The compile is successful when the bottom of the Arduino window says “Done compiling.” A message may appear in the black textbox in the bottom of the screen which describes the program memory usage; ignore this. However, if the message is an error or appears in red text, refer to the troubleshooting section of the appendix.

A screenshot of the Arduino IDE interface. The top menu bar shows 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with several icons. The first icon, a green circle with a white checkmark, is highlighted with a red rectangular box. A red arrow points from this icon down to a red box at the bottom of the code editor that contains the text 'Done compiling.'. The code editor displays C++ code for an Arduino sketch. The code includes several preprocessor directives for pin numbers and constants, followed by variable declarations and initialization. At the bottom of the code editor, there is a black status bar with white text that reads: 'Sketch uses 3,436 bytes (10%) of program storage space. Maximum is 32,256 bytes. Global variables use 291 bytes (14%) of dynamic memory, leaving 1,757 bytes for local variables. Maximum is 2,048 bytes.' Below the status bar is a small number '44'.

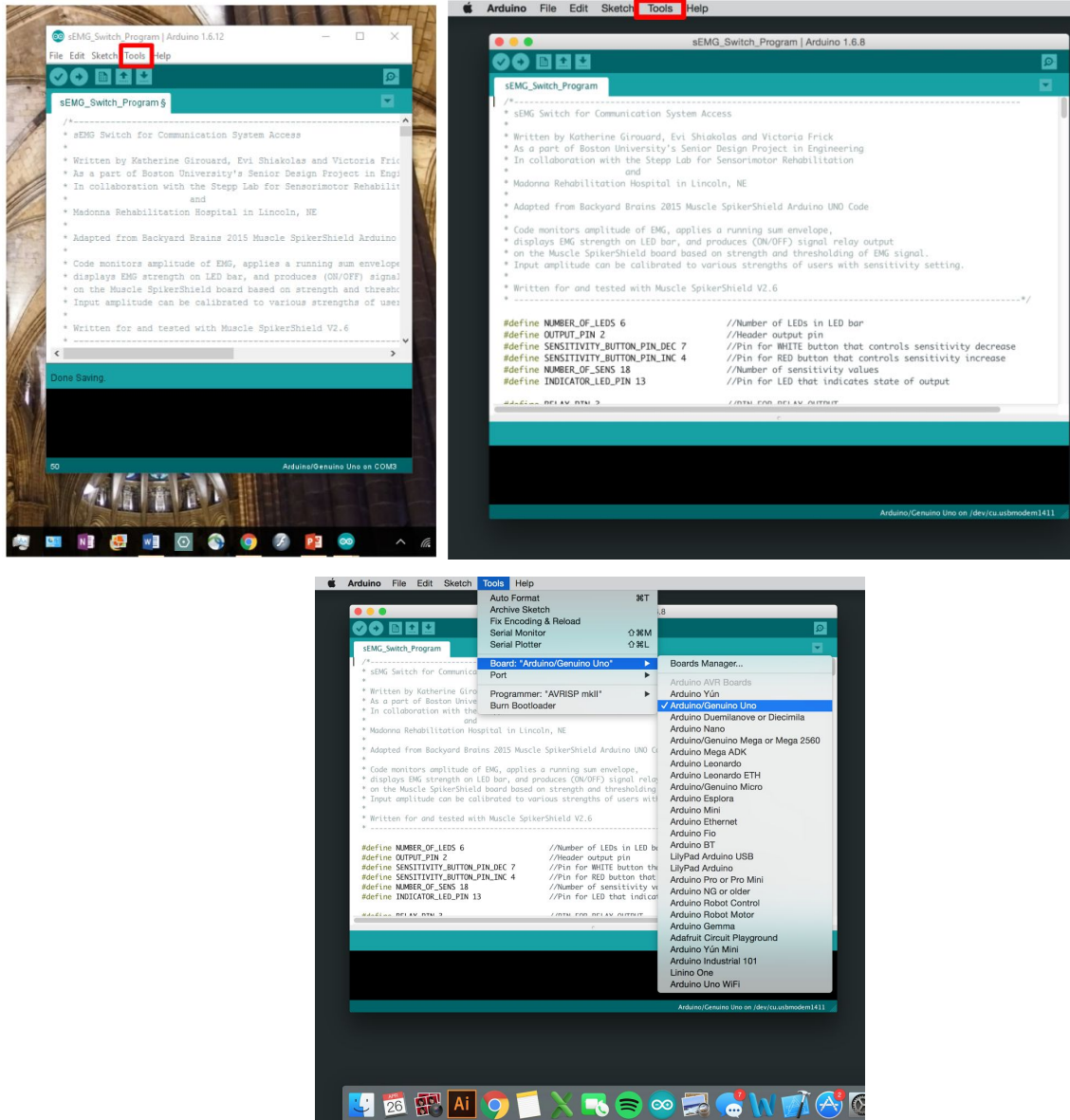
SOFTWARE SETUP: UPLOAD TO THE HARDWARE *CONTINUED*

4. In order to upload the software onto the Arduino hardware, connect the Arduino to the computer with the 28/24 AWG to USB cable. Commonly the USB port needed will feature both the USB symbol and a battery symbol.



SOFTWARE SETUP: UPLOAD TO THE HARDWARE *CONTINUED*

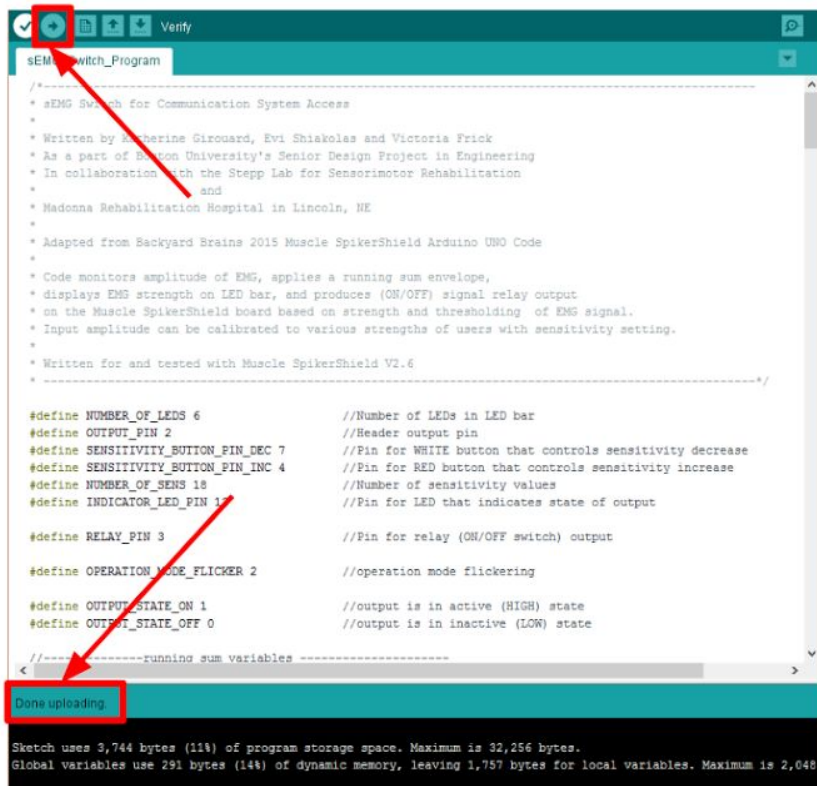
5. Prepare the computer's Arduino Software IDE for successful upload and communication with the microcontroller with the following steps:
 - Select “tools” in the top bar of the Arduino window (Windows), or in the top menu bar (Mac). Check that the board selected reads ‘Board:”Arduino/Genuino Uno”’



- Again under “Tools,” select the proper ‘Port.’ A range of options may be displayed, depending on the number of USB ports with which the computer is equipped.

SOFTWARE SETUP: UPLOAD TO THE HARDWARE *CONTINUED*

6. After the correct board and port have been selected, upload the code by selecting the rightward-facing arrow. Wait a moment for the upload to complete; the upload was successful if the bottom of the window reads **“Done uploading.”**



If the upload does not complete within a minute or if the Arduino Software throws an error, refer to the Troubleshooting section of the appendix. If the upload completed with the “Done uploading.” message, there is no need to consult the Troubleshooting section.

7. Unplug the 28/24 cable from the Arduino board and the computer.

If the set up detailed above was completed successfully (the bottom of the window reads “Done Uploading”), proceed to housing setup. If there were errors with the completion of the Software Setup, refer to the Troubleshooting section of the appendix.

HOUSING SETUP

The housing for the hardware of the device will be 3D printed. It is assumed that the user has access to an in-house 3D printer. However, if this is not the case, a contingency plan is outlined in the section titled “Alternate Housing Build Instructions.”

1. Download the CAD files
2. 3D print the housing
3. Attach the fixing mechanism
4. Alternate housing build instructions

DOWNLOAD THE CAD FILES

1. Go to: <http://sites.bu.edu/stepplab/research/semg-based-switch-design/>.
2. Download the .ZIP file and extract the files.

3D PRINT THE HOUSING

1. Locate the “Device Housing Base.STL” and “Device Housing Top.STL” files from the downloaded program files, “sEMG_Switch_Files” zip file. Make sure they are the “.STL” files.
2. Upload both .STL files to the 3D printer and print
 - a. Materials Recommendation: Acrylonite Butadiene Styren (ABS) P430
 - b. Part Density: Solid

ATTACH THE FIXING MECHANISM

1. The fixing mechanism may be velcro, a snap buckle, or whatever the hospital may choose to use for this device, as the housing may be best suited to be fixed to a wheelchair arm, hospital bed, or to the patient themselves. The housing design includes a slot for a 1”x0.12” strap.
2. Thread the chosen strap through the housing.

HOUSING SETUP *CONTINUED*

ALTERNATIVE HOUSING BUILD INSTRUCTIONS

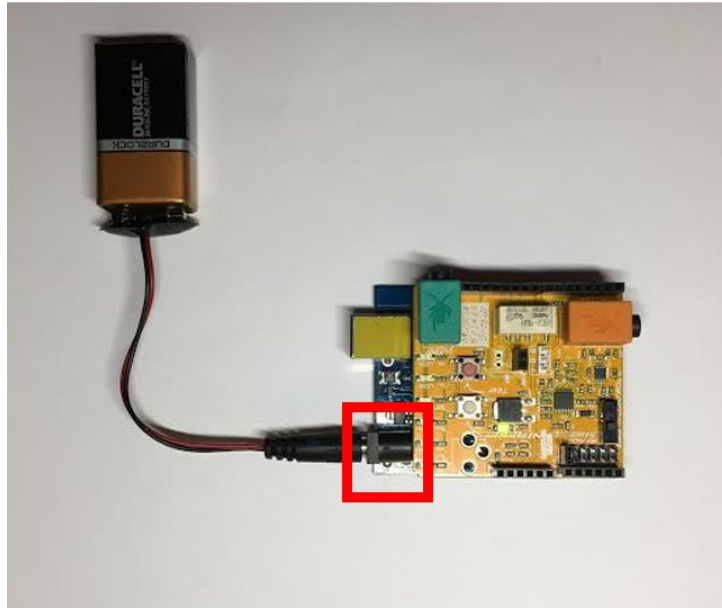
If using a 3D printer is not possible nor cost effective for the patient, an alternate approach to the housing design would be to use any kind of plastic box which can hold the electronic components. The purchased box would require several holes for the ports of the hardware to exit the box:

1. Electrode snap cable input
2. Mono-pin 3.5 mm audio jack output
3. Arduino 28/24 AWG to USB cable port

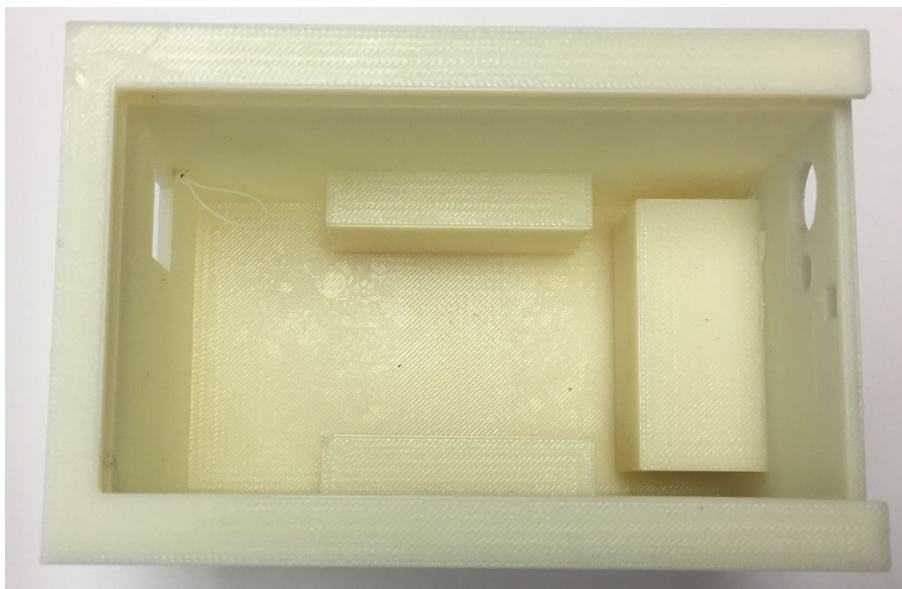
It is suggested that styrofoam blocks are used in place of the supports in the base of the housing to create a space for the 9V battery hold. The drawings of the device housing are included in the Device Housing Drawings section of the appendix in order to aid with the dimensions needed for drilling the holes in the correct locations. Refer to the drawings as well to size the styrofoam supports.

DEVICE ASSEMBLY

1. Plug the battery component into the 2.1mm black power port as shown on the diagram below. Make sure the plug is fully inserted into the port, but not forced. The 9V battery must be fully fastened, the setup will snap when in place. A green light will illuminate on the Arduino when the battery is correctly attached and providing power to the device.

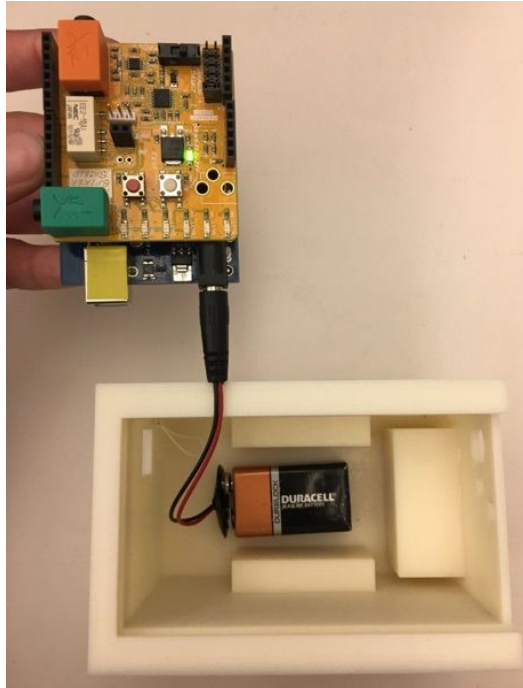


2. Orient the open housing lengthwise such that the two circular holes are to the right.

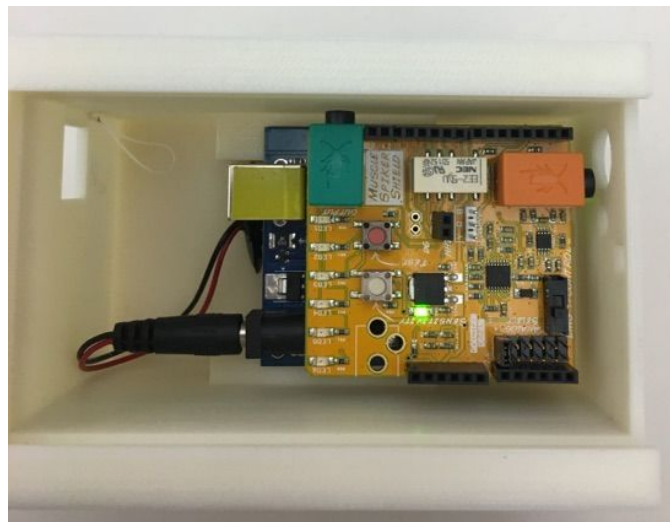


DEVICE ASSEMBLY *CONTINUED*

3. Place the battery into the housing as depicted in the photo below. The battery should lay lengthwise in the housing.

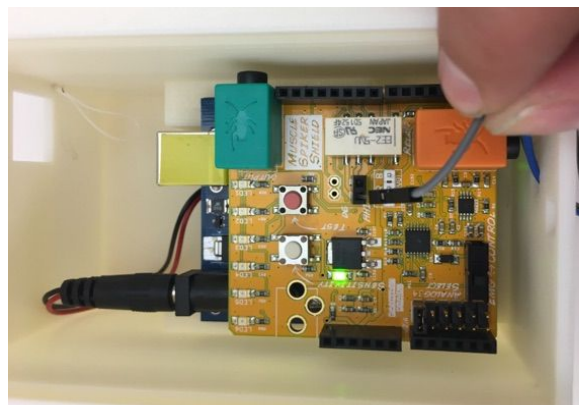
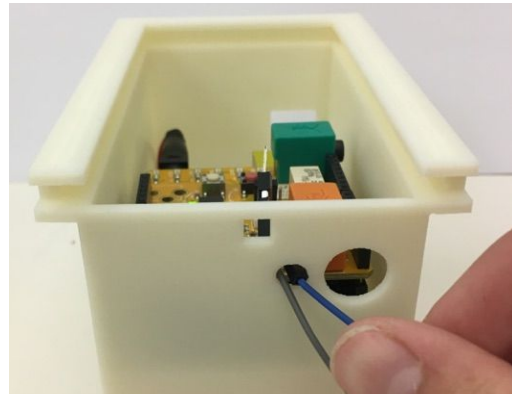
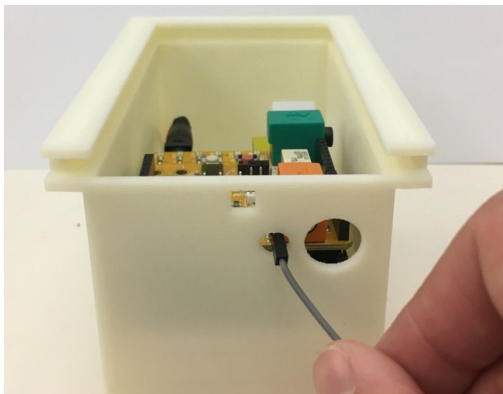
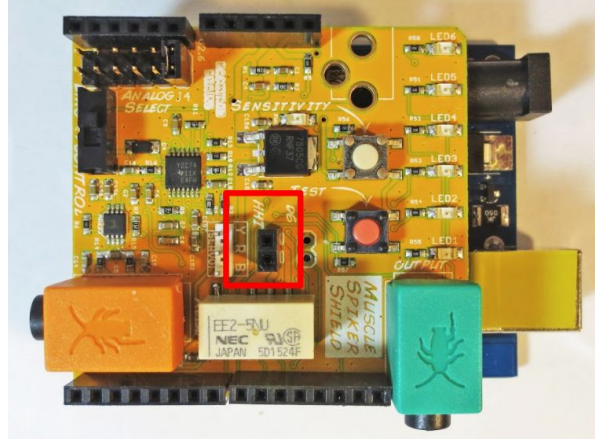


4. The battery sits between the supports for the Arduino. Orient the Arduino such that the orange port is to the right and aligns with the larger hole. Set the Arduino above the battery on the support as shown.



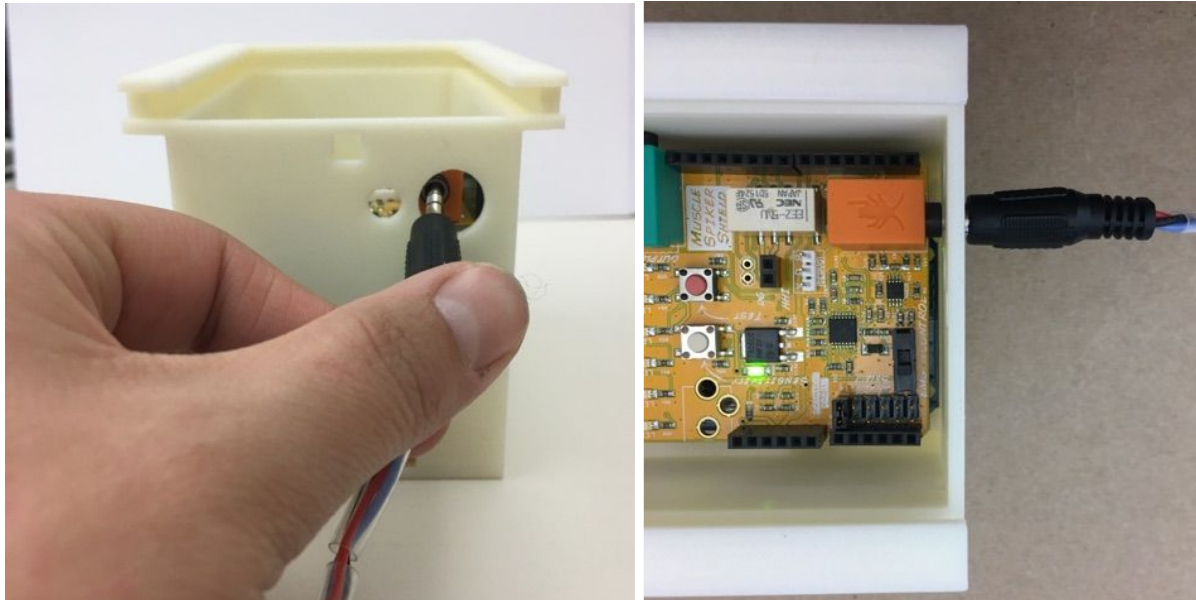
DEVICE ASSEMBLY CONTINUED

5. Plug the 3.5 mm male audio jack into the black relay pins as shown in the sequence below. Thread the pins of the plug, one at a time, through the smaller hole of the housing. Then insert the pins into the relay highlighted on the diagram below. It is labeled as “HHI” on the Muscle SpikerShield. This connection is what allows the switch to interface with existing AAC systems. It does not matter which colored wire plugs into each port of the relay pin.

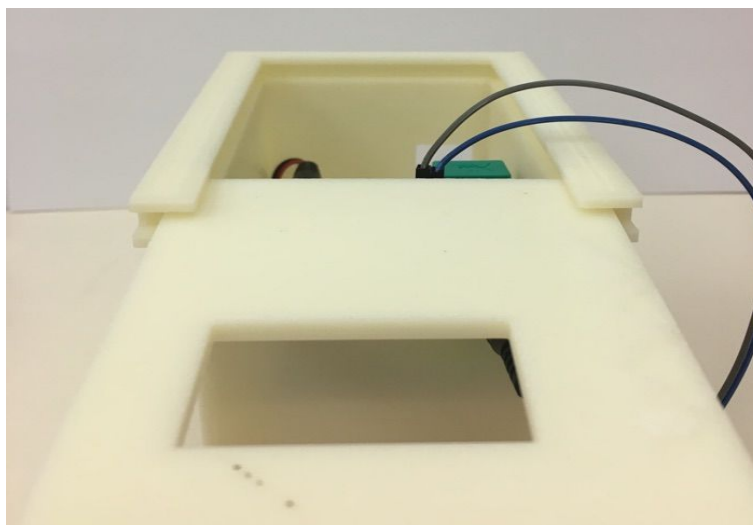


DEVICE ASSEMBLY CONTINUED

6. The Arduino may have shifted. It is alright to push it back into place such that the orange port is flush with the large hole. Insert the electrode snap plug into the orange port **through** the housing.

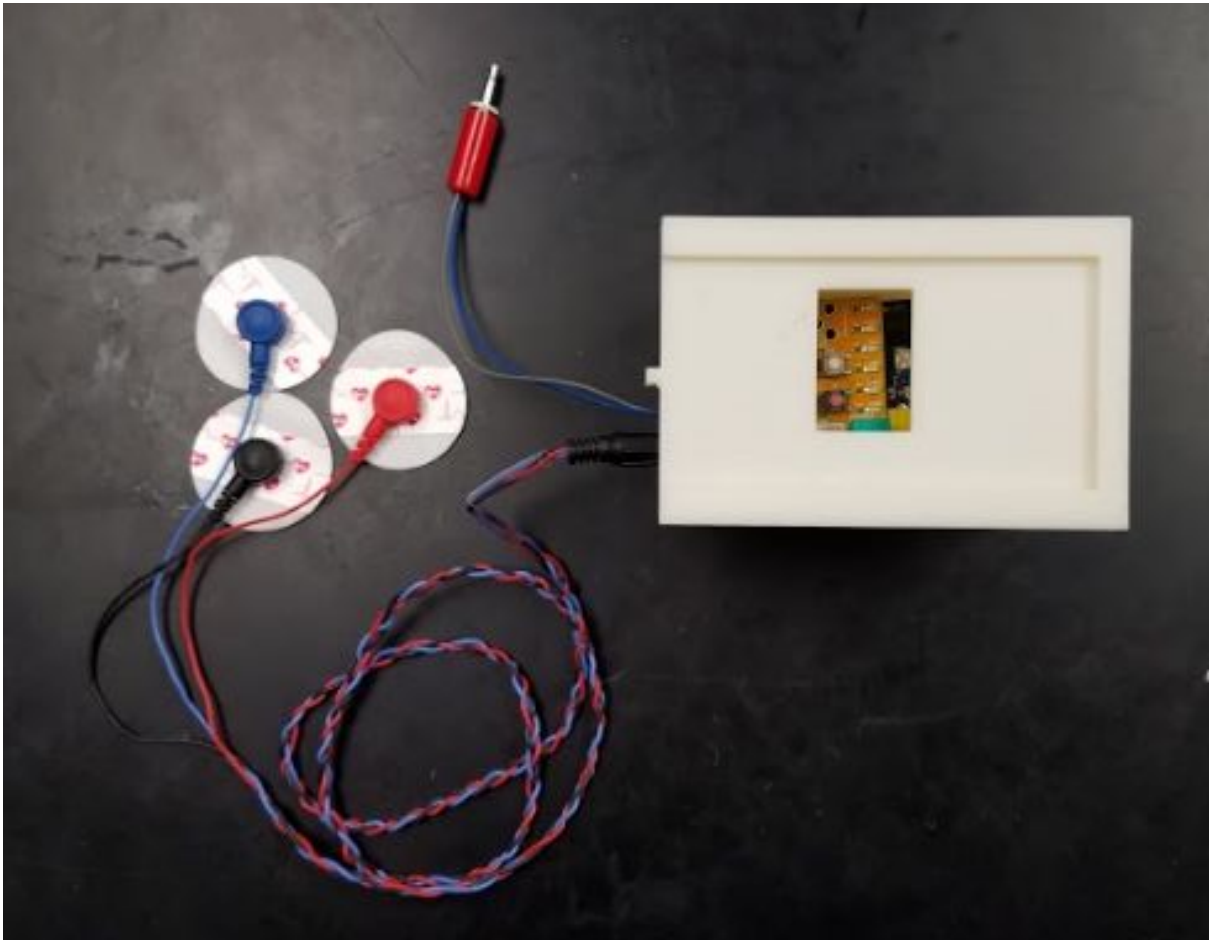


7. Slide the housing lid into the the grooves at the top of the base. Be sure to set the hook down so it catches the base of the housing.



DEVICE ASSEMBLY *CONTINUED*

8. Now the device is completely assembled and ready to be calibrated for patient use. Move on to “Calibration Instructions.”

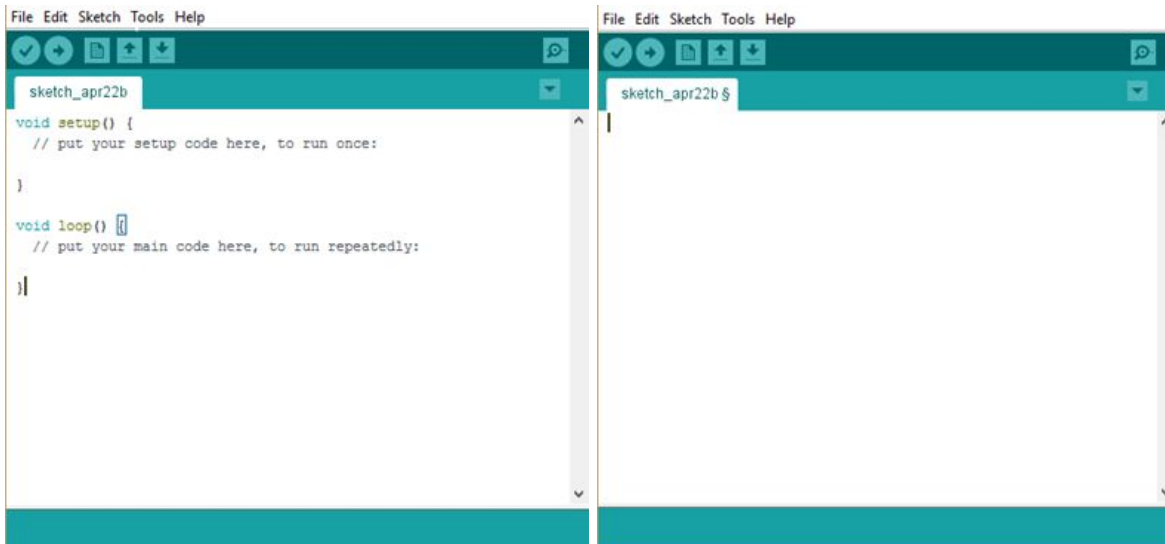


APPENDIX: TROUBLESHOOTING

If there were no problems setting up the device, it is not necessary to read the Troubleshooting section.

If the sEMG_Switch_Program.ino **does not open a program in Arduino IDE**, perform the following:

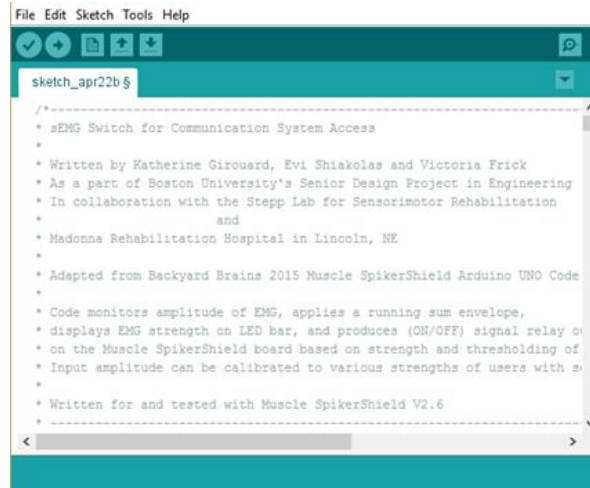
1. Open the file “sEMG_Switch_Program.txt” text document from the zip file “sEMG_Switch_Files.” This is a text file; it will open in Notepad for Windows users and TextEdit for Mac OS users.
2. Open the Arduino IDE. Select “File: New” to open a new sketch window if one does not appear. This will have a basic template already in the sketch; delete all of the text. Press CTRL + A (Windows), or CMND + A (Mac OS) and delete. Alternatively highlight all of the text and delete it. Delete **everything** in the sketch window.



“File: New” produces this window.

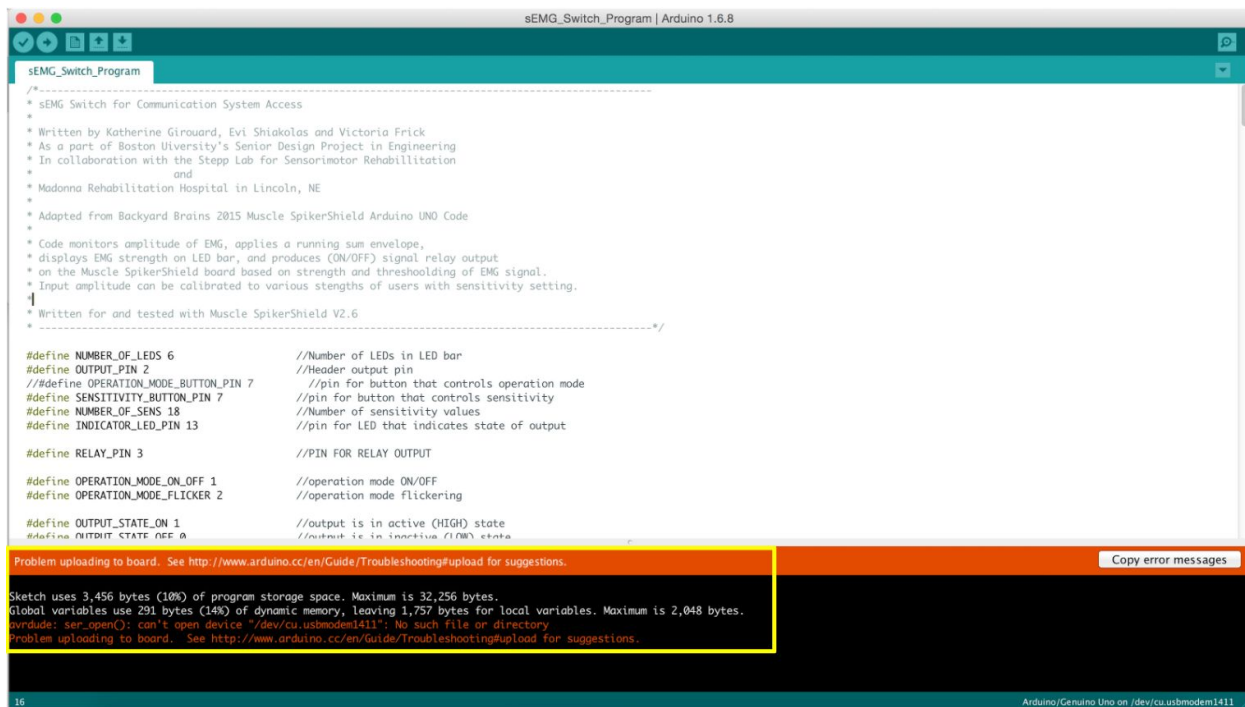
The blank sketch window. **Be sure there is nothing in the window.** Only the cursor is shown in the image above

3. Copy and paste the program into the Arduino IDE (CTRL+A on a PC or CMND+A on a Mac to select all of the contents of the .txt file).



The program should appear as above. Refer back to the Upload to the Hardware section of Software Setup and continue with step 3.

The Arduino Software **may not correctly communicate with the microcontroller**. This is the problem if the uploading takes longer than a minute, or if the bottom of the window does not read “Done uploading.” In the case that the bottom of the window does not read “Done Uploading,” it will show an error message and red text in the dialogue window, as shown below.



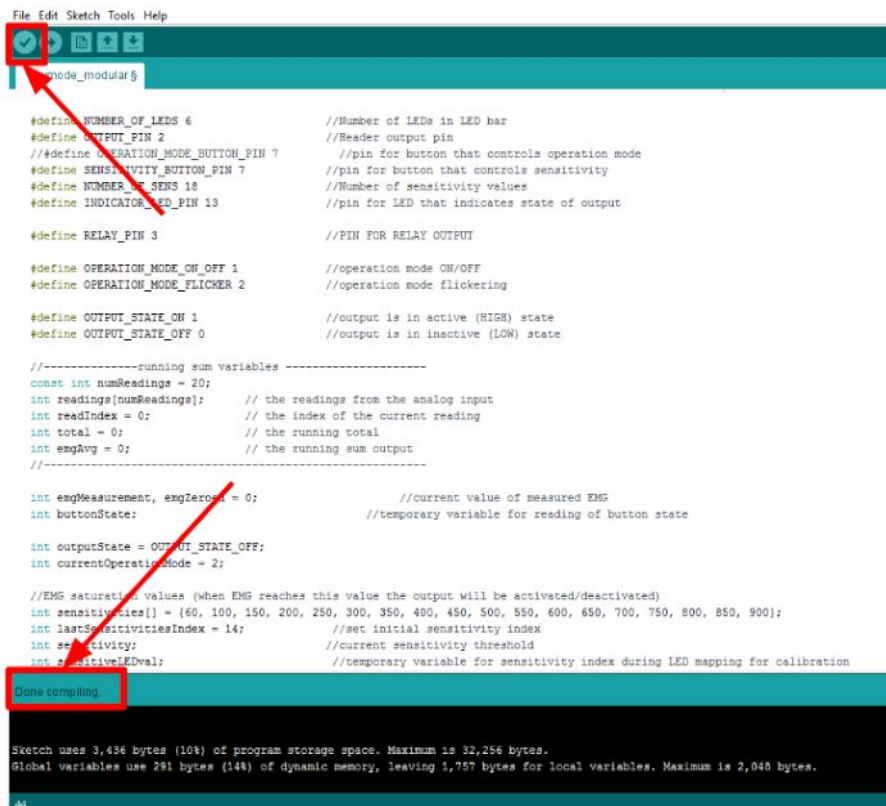
The error shown in the image above reads: “Problem uploading to board. See

<http://www.arduino.cc/en/Guide/Troubleshooting#upload> for suggestions.”

Attempt troubleshooting common errors with the solutions listed below:

1. Compile the code again.

- Click the checkmark icon in the upper left of corner of the screen. Compiling checks that the code is error-free and ready to be uploaded to the hardware. The compile is successful when the bottom of the Arduino window says “Done compiling.” This is shown below.



```
File Edit Sketch Tools Help
[Checkmark] [Upload] [New] [Open] [Save] [Find] [Help]

arduino_modular$

#define NUMBER_OF_LEDS 6           //Number of LEDs in LED bar
#define OUTPUT_PIN 2              //Header output pin
//define OPERATION_MODE_BUTTON_PIN 7 //pin for button that controls operation mode
#define SENSITIVITY_BUTTON_PIN 7  //pin for button that controls sensitivity
#define NUMBER_OF_SENS 18         //Number of sensitivity values
#define INDICATOR_LED_PIN 13      //pin for LED that indicates state of output

#define RELAY_PIN 3               //PIN FOR RELAY OUTPUT

#define OPERATION_MODE_ON_OFF 1   //operation mode ON/OFF
#define OPERATION_MODE_FLICKER 2 //operation mode flickering

#define OUTPUT_STATE_ON 1         //output is in active (HIGH) state
#define OUTPUT_STATE_OFF 0       //output is in inactive (LOW) state

//-----running sum variables-----
const int numReadings = 20;
int readings[numReadings]; // the readings from the analog input
int readIndex = 0;         // the index of the current reading
int total = 0;             // the running total
int emgAvg = 0;            // the running sum output
//-----

int emgMeasurement, emgZero = 0; //current value of measured EMS
int buttonState;                //temporary variable for reading of button state

int outputState = OUTPUT_STATE_OFF;
int currentOperationMode = 2;

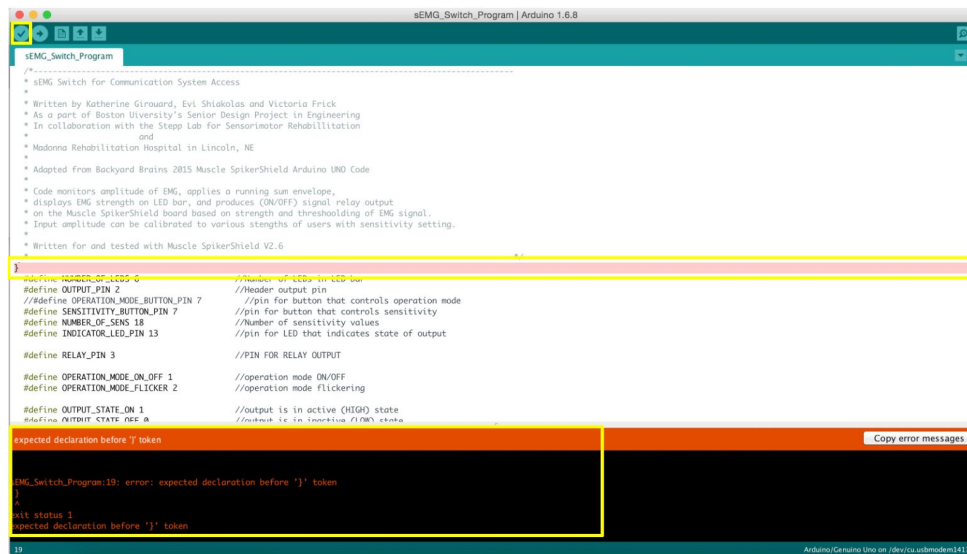
//EMS saturation values (when EMS reaches this value the output will be activated/deactivated)
int sensitivities[] = {60, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900};
int lastSensitivitiesIndex = 14; //set initial sensitivity index
int sensitivity;                //current sensitivity threshold
int sensitivityLevel;           //temporary variable for sensitivity index during LED mapping for calibration

Done compiling

Sketch uses 3,436 bytes (10%) of program storage space. Maximum is 32,256 bytes.
Global variables use 291 bytes (14%) of dynamic memory, leaving 1,757 bytes for local variables. Maximum is 2,048 bytes.

44
```

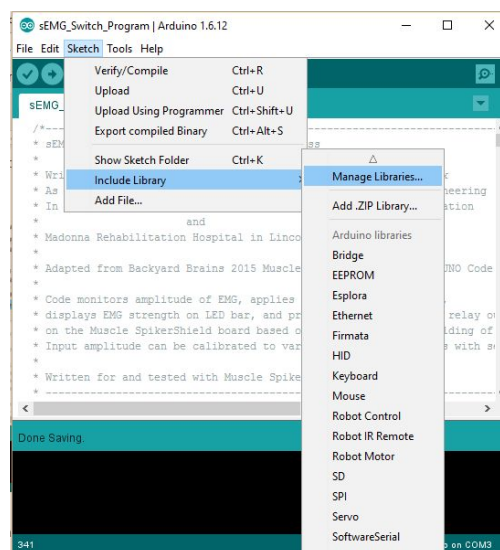
- If it is not successful, the bottom of the Arduino will show an error message. Most likely, it will be a syntax error, which means that the text in the program was altered in some way between download and compiling.
- For example, the error message shown below reads “expected declaration before “}” token.” This shows that the user accidentally typed an extra “}” in the code, or did not completely delete the text from the basic template in Arduino, leaving an extra “}” in the code.



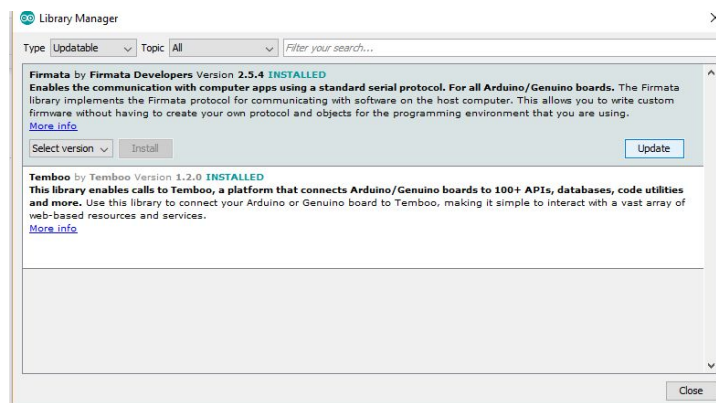
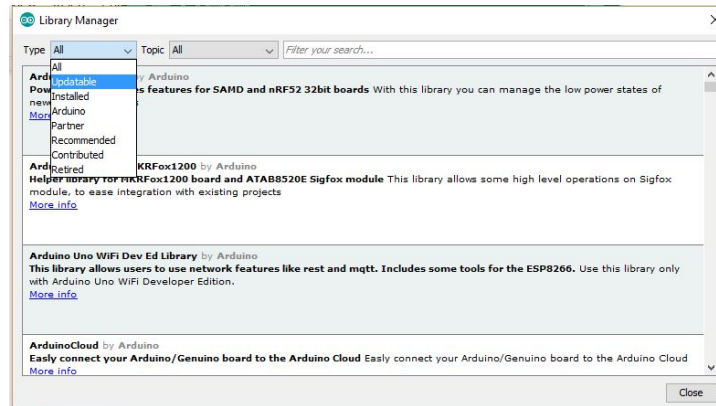
- If the user originally opened the sEMG_Switch_Program.ino file, try downloading and opening the file again.
- If the user originally opened the sEMG_Switch_Program.txt file, try removing all of the text from the Arduino window, and then copy and paste all of the text from the sEMG_Switch_Program.txt file into the Arduino window again.
 - Be sure that the Arduino window is completely blank when adding the text. Do not type or delete any characters while copying and pasting the text.

2. Update the libraries.

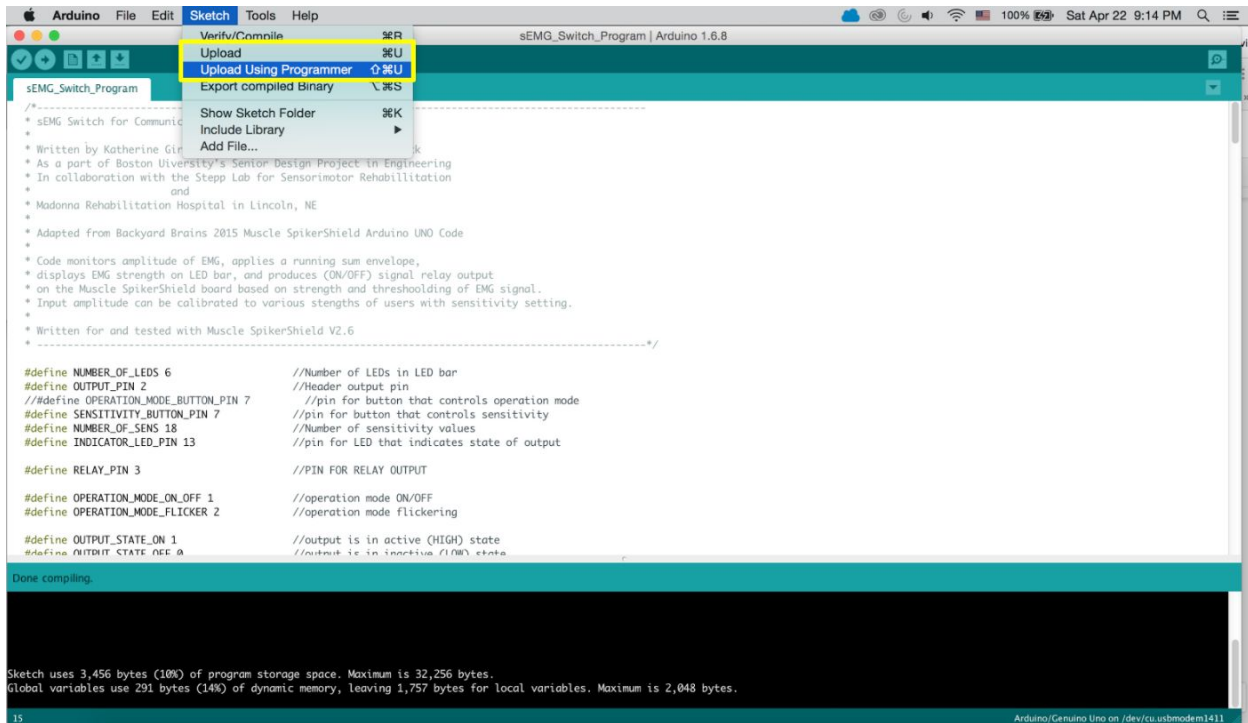
- Update the Arduino Software libraries by clicking “Sketch” in the header buttons, then “Include Library” and next “Manage Libraries...”



- The subsequent pop up window will allow filtering of existing libraries by “Type” (top left corner); filter by “Updateable” libraries. Click each one, and an update button will appear. Update all that are available.

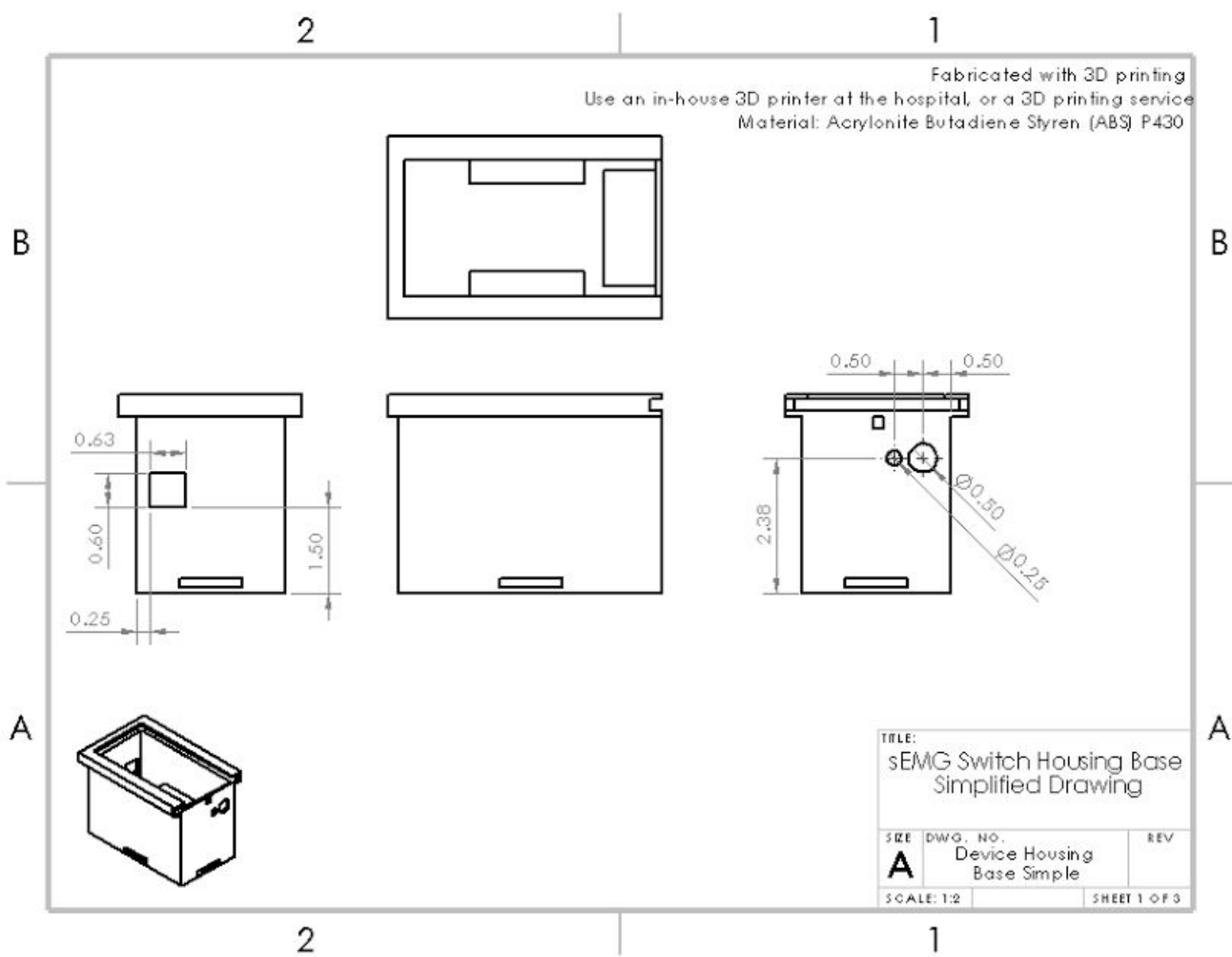


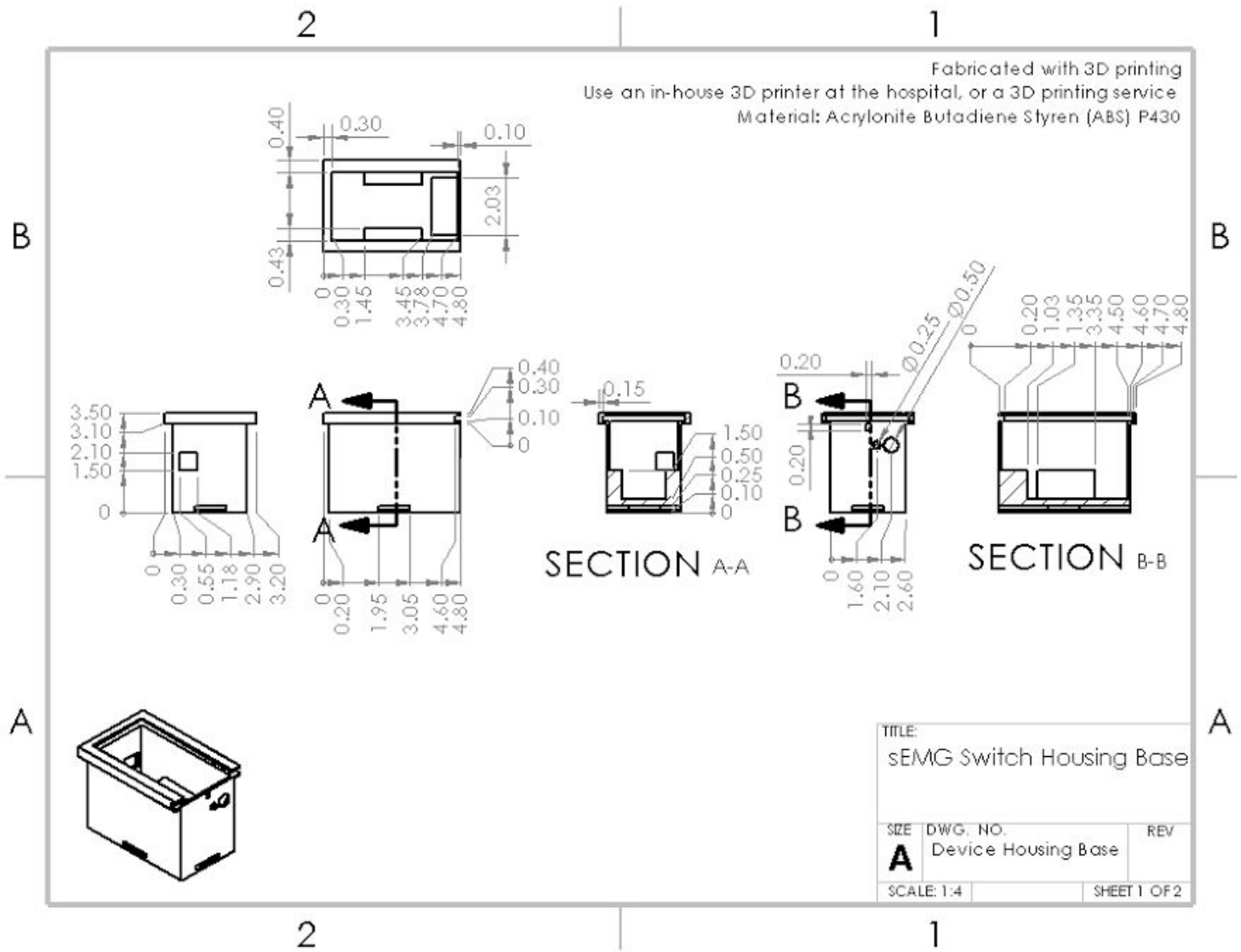
3. Complete restart of Arduino hardware and software.
 - Unplug the Arduino from the computer using the 28/24 AWG to USB cable
 - Quit the Arduino software
 - For Windows users, click “File” then “Quit” or press “CNTL+Q”
 - For Mac OS users, click “Arduino” then “Quit Arduino” or press “Command+Q”
 - Plug the Arduino back into the computer using the 28/24 AWG to USB cable
 - Reopen the Arduino software
4. Try uploading with programmer
 - Click “Sketch” in the header bar and select “Upload Using Programmer.” Wait as the system attempts an upload.



- If this does not work, try a normal upload by clicking “Sketch” in the header bar and selecting “Upload,” or click the rightward-facing arrow, as described step 5 of “Uploading the Hardware.”
5. Verify that the correct board and serial port are selected, as described in step 4 of Upload to the Hardware.
 6. Most computers are equipped with multiple USB ports. If the proper board and port have been selected a green light will illuminate on the Arduino. If not, another USB port may need to be used. Port selection is detailed in step 4 of “Upload to the Hardware”
 7. Restart the computer
 - Plug the Arduino into the computer using the 28/24 AWG to USB cable *before* opening the Arduino Software application.
 8. Research the errors: copy the error messages that appear at the bottom of the Arduino Software screen after an upload attempt and paste them into a search engine. Common errors often have simple solutions defined by online resources, specifically at <http://www.arduino.cc/en/Guide/Troubleshooting>.

APPENDIX: HOUSING DESIGN DRAWINGS





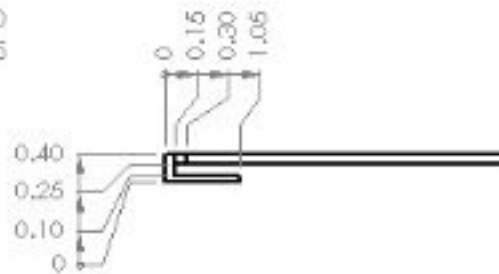
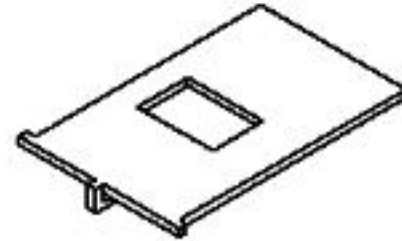
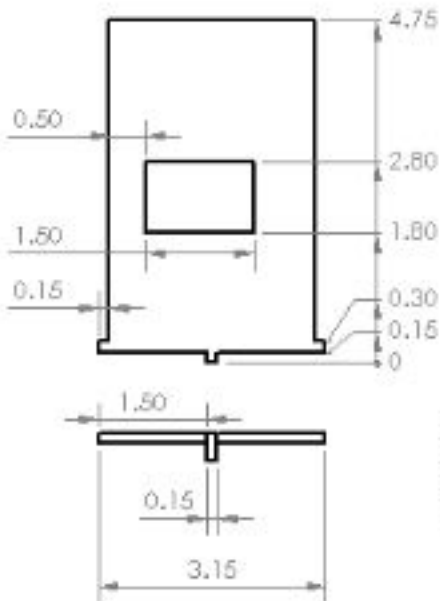
2

1

Fabricated with 3D printing
 Use an in-house 3D printer at the hospital or a 3D printing service
 Material: Acrylonitrile Butadiene Styren (ABS) P430

B

B



A

A

TITLE:		
sEMG Switch Housing Top		
SEE	DWG. NO.	REV
A	Device Housing Top	
SCALE: 1:2		SHEET 2 OF 2

2

1