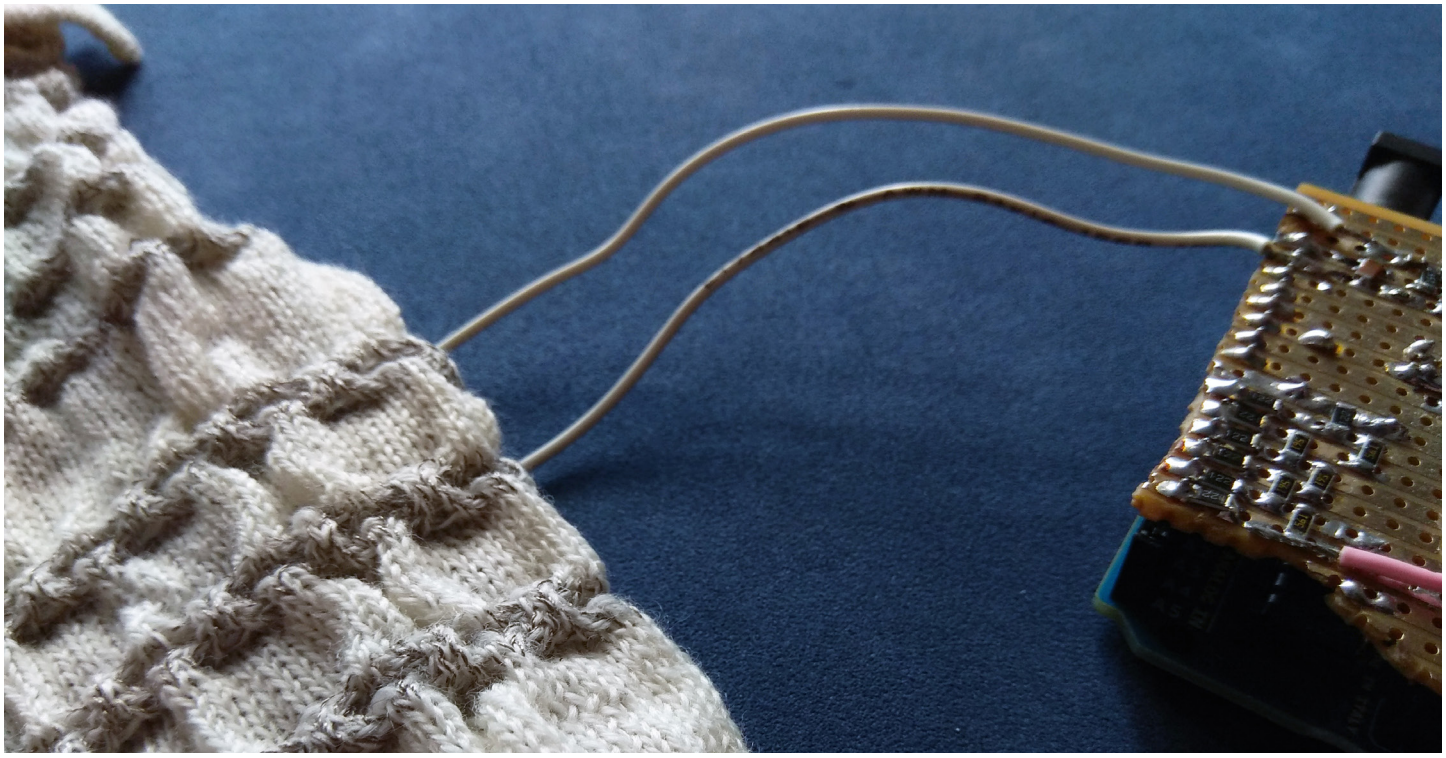
Instructions for building

TEKSIG

using Arduino Uno



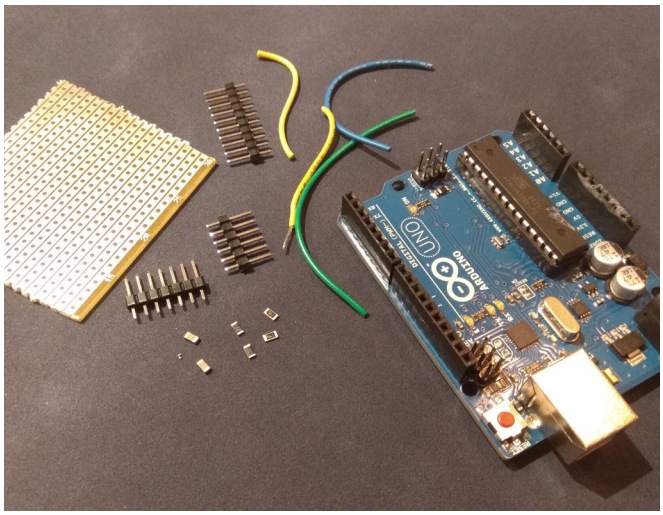
The following pages contain information on how to build a TEKSIG Arduino. While these instructions use surface-mounted components, they can be changed to normal wire-components as well.

In the last part of this document, there is information on how to install and adjust the programs. Please note that TEKSIG functions only with Arduino UNO. To port it to another board, the code must be changed.

The visualisation program is built with Processing. As such, it should be directly compatible with most environments. However, care must be taken to select the correct serial port.

Bill of Material (BOM)

R_L	= Load varies	
R_1	= 220R	x 3
C_1	= 6,8nF	x 3
R_2	= 150kR	x 2
C_2	= 100nF	x 1
R_3	= 1,2kR	x 7
R_4	= 560R	x 5



Headers:
2 x 6-pin headers
1 x 13-pin header
4 x jumper wires
2 x wires to fabric

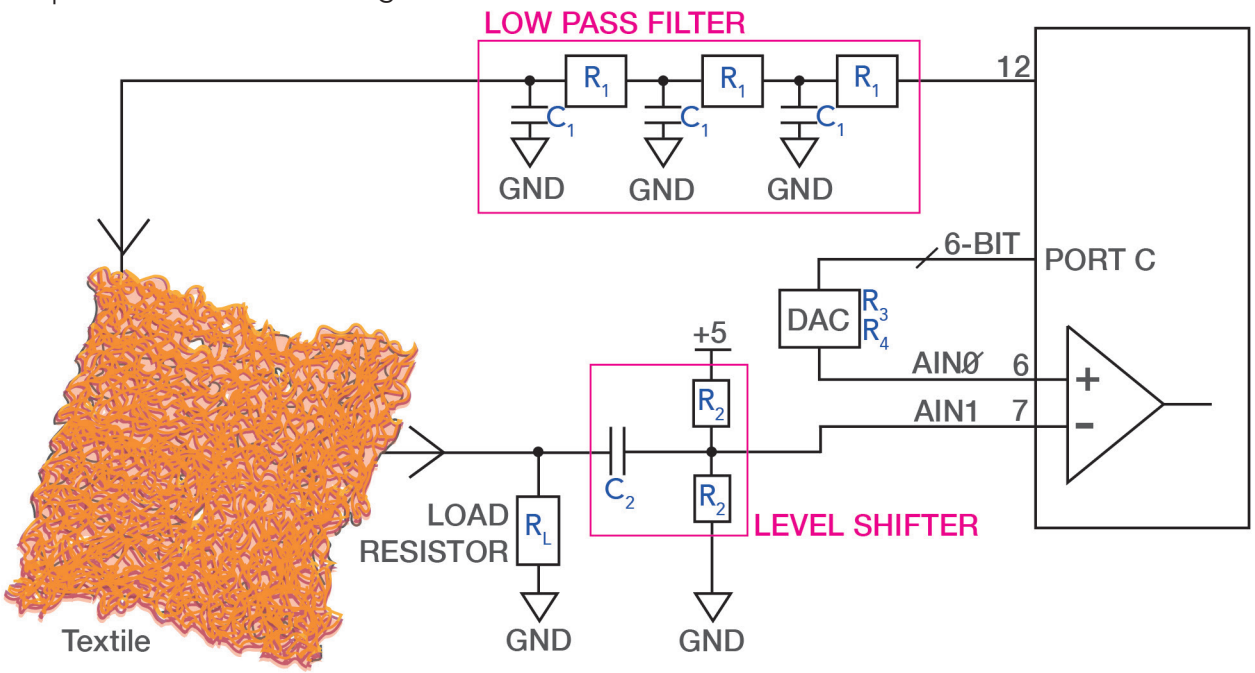
Arduino Uno

Veroboard

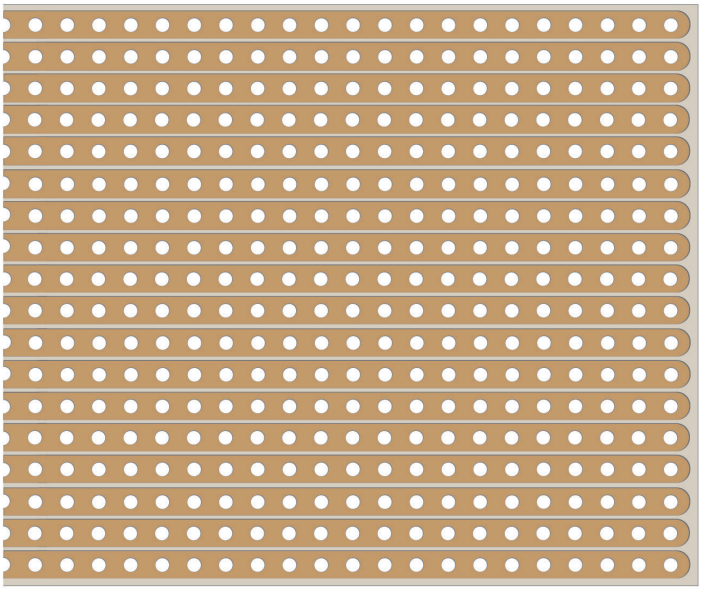
Other supplies:
Soldering station
Soldering tin
Tin scissors

Download the code and other files from github or teksig.net

A simplified schematic of Teksig

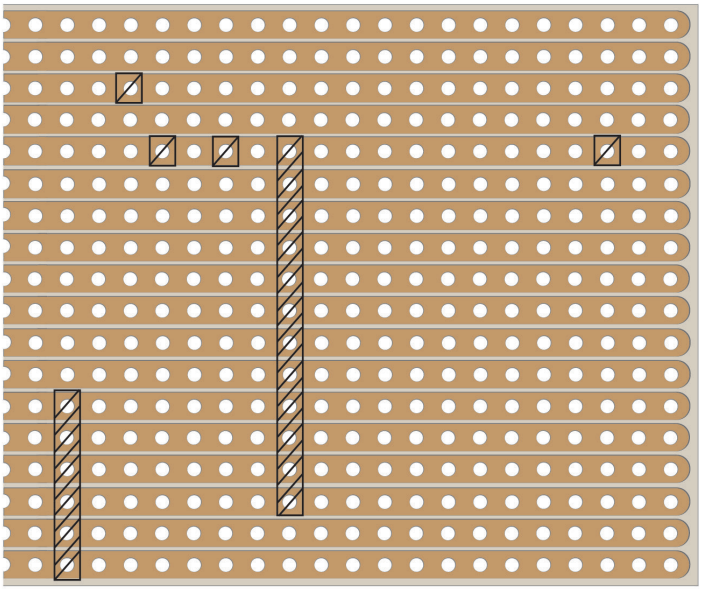


Step-by-step instruction to building a Teksig-shield using Arduino-Uno



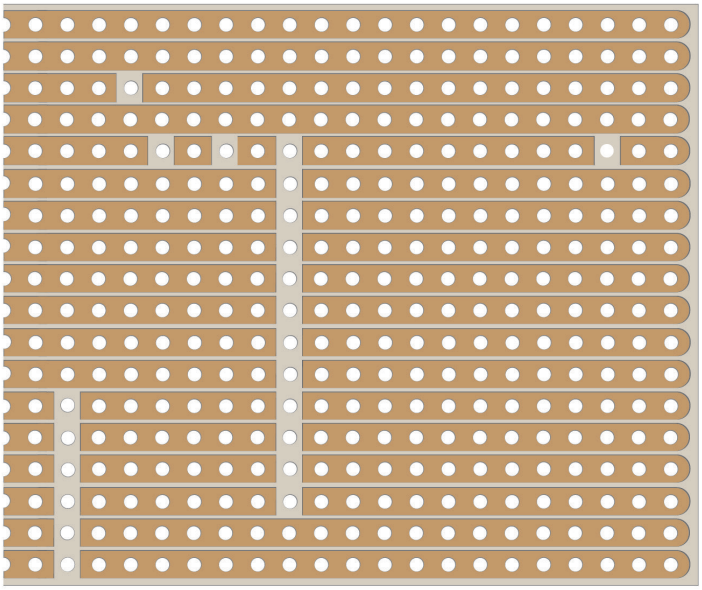
Step 1.
Cut the veroboard to the same size as the Arduino Uno (W21 x H18 dots).

Once the veroboard has been cut to its correct size, the sharp edges can be finished off with sandpaper.



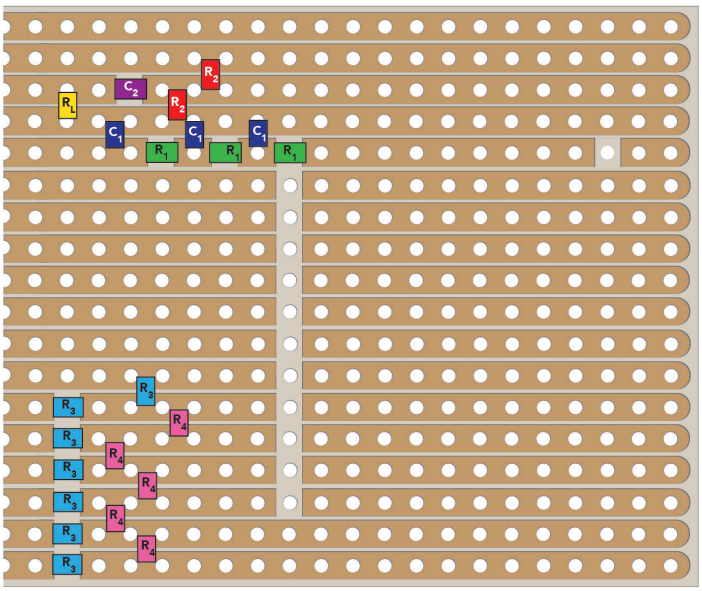
Step 2.
Remove the copper foil from the veroboard carefully with a tool of your choice in the marked area, as seen in the figure on the left.

This step is essential as it ensures no short circuits and the components to function well.



Step 3.
After removing the conductive surface, the veroboard should look like the figure on the left.

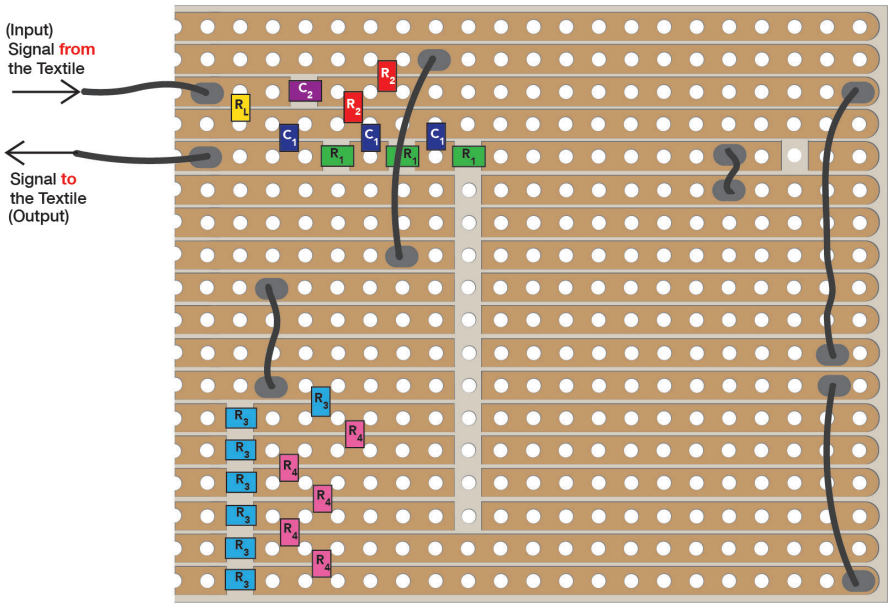
Check that the removed copper foil area is identical to the given instructions, and is clean at the edges.



Step 4.
Solder the components to the veroboard according to the figure on the left.

If you use surface-mount components, solder them first.

Make sure the components are connected to both sides over the gaps.



Step 5.
Cut the wire into suitable length pieces.

Pull the wire ends from the back of the veroboard, through the holes, out to the facing side.

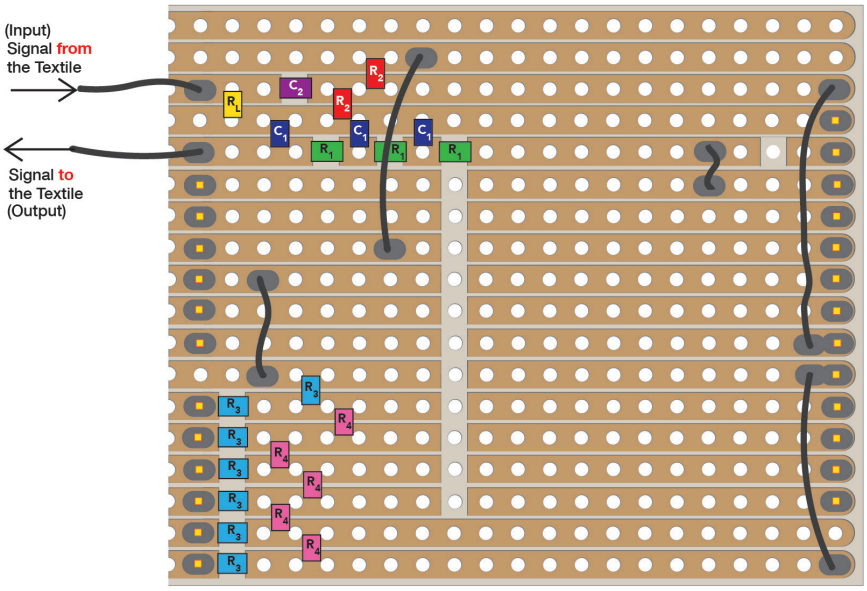
Solder the wire ends to the facing side of the board.

Alternatively, the wires can remain on the facing side of the veroboard.

Solder the wires, used for signal transmission to and from the textiles, onto the veroboard surface, not through the holes.

It is advisable to consider the length of the signal transmission wires to and from the textile according to the intended interaction measurement activities.

Alternatively, you can use a connector of your choice.



Step 6.

Attach the pin headers to the back side of the veroboard so that the shorter pins come through the holes to the facing side.
(The longer pins remain on the back side pointing away from the veroboard.)

Check that the placement of the pin headers matches the instructions seen in the left figure.

Solder the pin headers to the veroboard on the facing side. Note, that the right-side 13-pin header does not directly fit Arduino due to small offset. Some pins need to be bent.

Arduino-TEKSIG and Arduino.cc

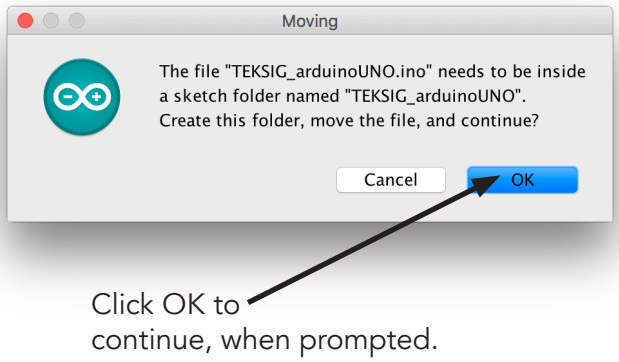
NOTE *before programming Arduino*: Make sure you have Arduino UNO. TEKSIG needs a fast board and expects to have a 20MHz clock. Furthermore, TEKSIG has only been tested with Arduino UNO, and it contains inline assembler for the ATMEGA328-processor. It may not work with any other Arduino without modifications. If none of the above is familiar for you, just use Arduino UNO for now.

The Arduino-TEKSIG is tailored for Arduino UNO. If you have not used Arduino before, it is recommended to familiarise yourself with the examples that come with the Arduino- program.
If you do not have the Arduino Software (IDE) installed in your computer, install it first.

Download Arduino Software from here: https://www.arduino.cc/en/Main/Software

Download the TEKSIG_arduinoUNO.ino file from Github and open it using Arduino app. If you need drivers, refer to instructions available on Arduino-website.

If the If the TEKSIG_arduinoUNO.ino file is not in a correct folder, Arduino will prompt you to create one. Click OK (as seen in the figure on the right).



Click OK to continue, when prompted.

After you have opened the Arduino, and the TEKSIG_arduinoUNO has been loaded, you can set the BOARD and the TOOLS.

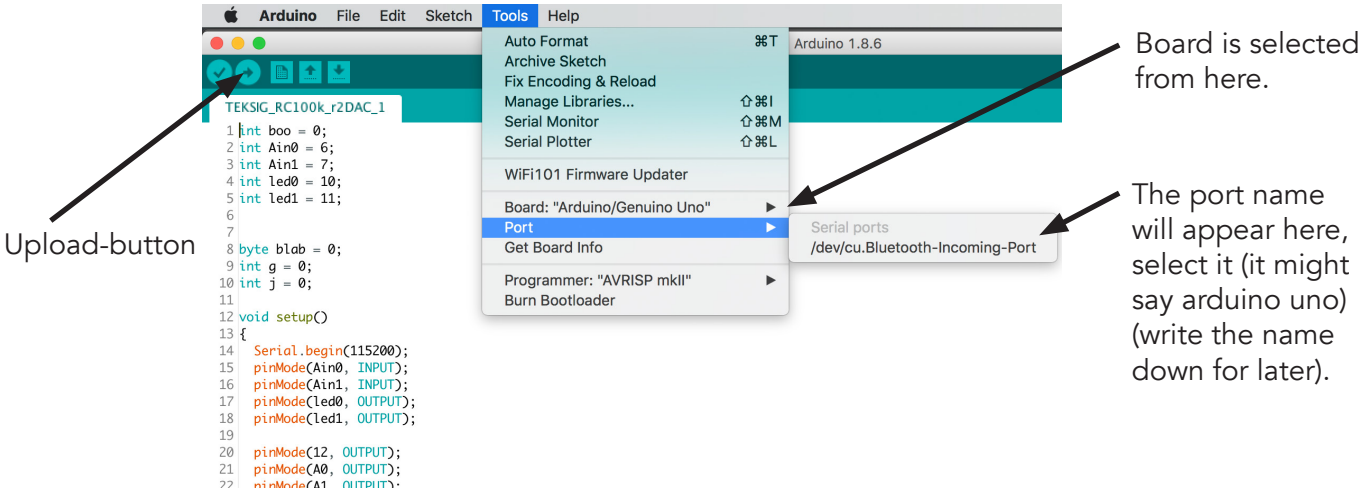
1) Plug in your arduino.

2) Select the correct board, by choosing Arduino UNO:
menu -> TOOLS -> BOARD -> Arduino/Genuino UNO.

3) Select the correct port:
menu -> TOOLS -> PORT -> arduino

4) Press the UPLOAD -button. This will compile and upload the TEKSIG to Arduino. If everything is ok, after a while, TEKSIG is programmed to your arduino UNO.

5) You can now close Arduino and start the Processing-program.



Make a note of the name shown in the menu -> TOOLS -> PORT -> [selected Arduino], as you will have to match this name with a correct number in the visualisation program.

Visualisation program and Processing.org

NOTE *before the first try of TEKSIG*: if you RUN the program, it will most probably crash. This means you will be looking at a grey screen. You cannot close it by pressing ESC-key, but you must move the window aside, and press STOP. This is the only way to close the crashing app. Do not panic, it is ok.

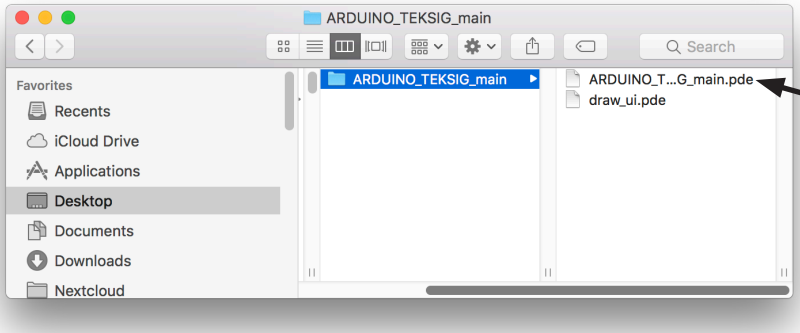
The visualisation program is written with processing.org. If you have not used Processing before, it's reasonably similar to Arduino. Download and install it. Familiarise yourself with it by running a few simple example-programs that are included, so that you have a rough idea where everything is. There are some differences between Mac, Windows and other OS versions. Those are mainly related to the serial communication, i.e. how the data travels in the USB-cable between your computer and Arduino.

Download Processing from here: https://www.processing.org/download/

Modifying the visualisation-program to read correct Arduino-port

When you have downloaded the Processing-files, move them to a folder. Your files should look like the figure below. Before opening the Processing-files, ensure that your Arduino Uno is connected to your computer.

Open the main file named "ARDUINO_TEKSIG_main.pde". This will also open additional files which are part of the program. You can ignore those for now. The part you may need to change is in the main file.



Once Processing is installed, you can open the visualisation program code by double-clicking this file.

To make the program communicate with your Arduino, scroll down the code to the Setup() -function in the main file (found in the white section of the window).

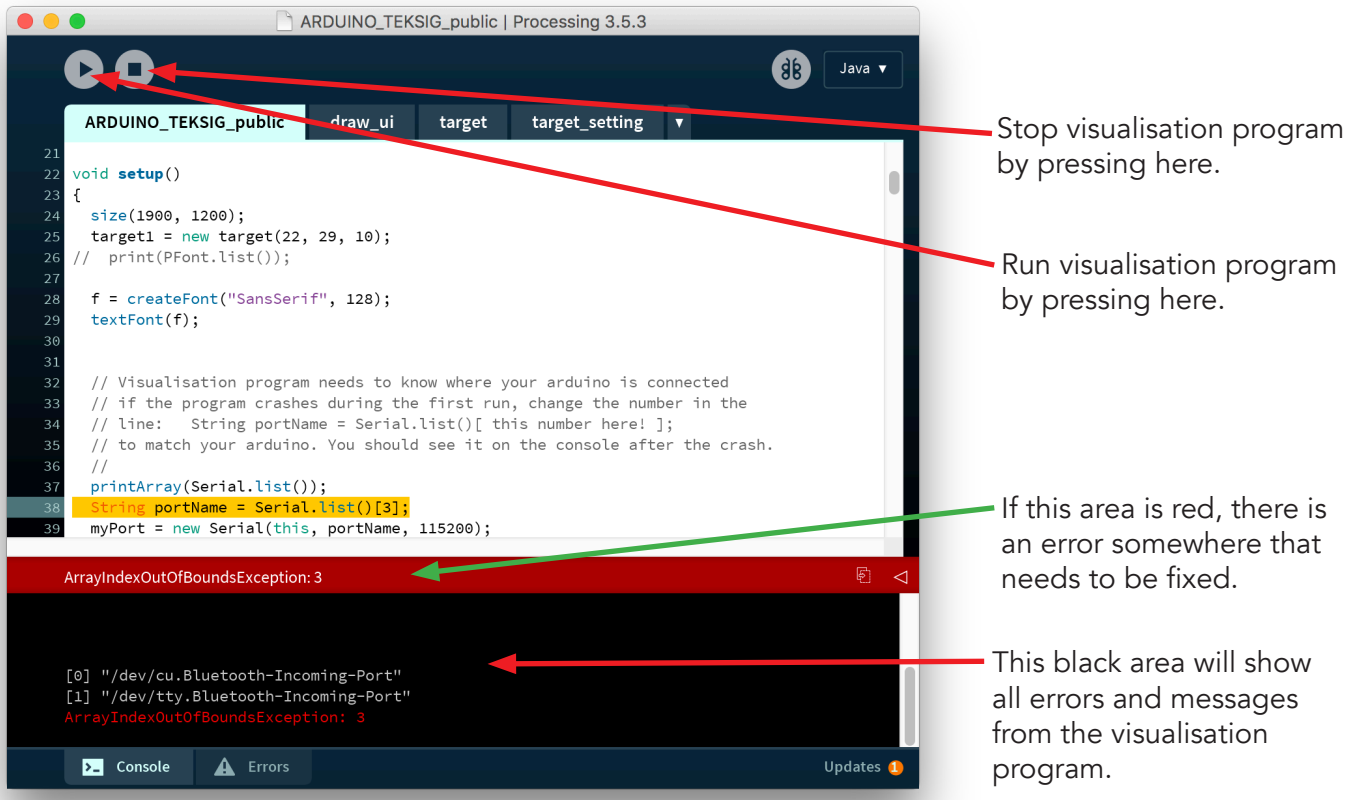
Change the "Serial.list()[3]" number to the correct number.

```
31
32  // Visualisation program needs to know where your arduino is connected
33  // if the program crashes during the first run, change the number in the
34  // line:   String portName = Serial.list()[ this number here! ];
35  // to match your arduino. You should see it on the console after the crash.
36  //
37  printArray(Serial.list());
38  String portName = Serial.list()[3];
39  myPort = new Serial(this, portName, 115200);
```

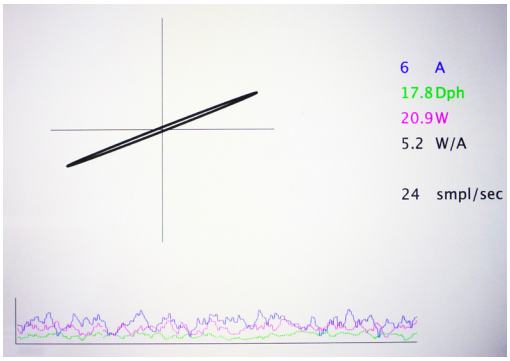
Change the number 3 to the number indicating the correct port name.

Instead of writing the name of the port, the program will select one from the list that your system generates. The correct number is found in brackets, in the black output-area below the code, in conjunction with the same port name given after uploading TEKSIG_arduinoUNO. (see p.5)

If the black output-area is empty, press the Run icon, as seen in the figure below. A separate window displaying the Lissajous-figure will appear. Then, press the Stop icon, which will cause the window to disappear, and the required information to determine the correct number should appear in the black output area.

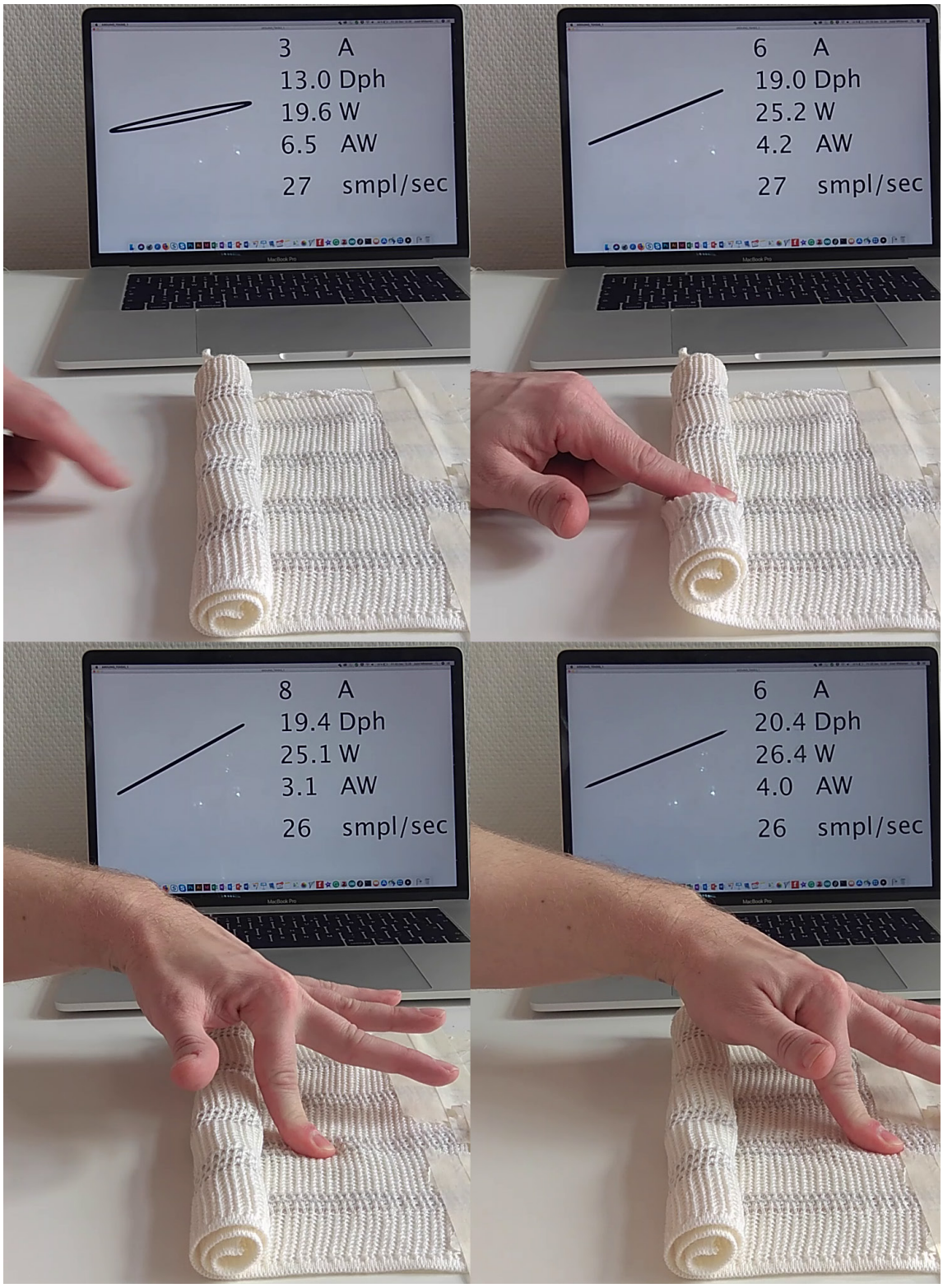


If you have a wrong number, the program will print out the available ports and numbers and crash. Change to the number correctly, and restart the program. (If your Arduino is not on the list, then you might be missing drivers, etc. which is beyond the scope of these instructions.)



Once the visualisation-program is running properly, you will see this kind of screen in the figure above. Smpl/sec indicates how many data-points per second the Arduino is sending. If it is 0, then the Arduino is not sending any data, or the selected port number is incorrect.

The program is slightly improved from the version presented at the CHI-paper, and also has a small history-graph screen below the Lissajous-reproduction, as seen in the figure above. This can be used to learn to see the differences between the Lissajous-reproduction and the values over time.



An example of testing the electrical signal behaviour of a conductive knit, while rolling and pressing the knit sample.

(Images taken from: Jussi Mikkonen and Riikka Townsend. 2019. Frequency-Based Design of Smart Textiles. In CHI Conference in Human Factors in Computing Systems Proceedings (CHI 2019), May 4-9, 2019, Glasgow, Scotland, UK. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300524)