Mouse fatigue detection training a Neural Network for Arduino in TensorFlow

Introduction

- We hope this presentation helps you understand, how to train, and implement a Neural network to predict forearm muscle pain levels, using Tiny Machine Learning, Arduino, and Tensorflow Lite
- Machine Learning is a branch of artificial intelligence, focused on building applications that learn from data and improve their accuracy over time.
- Deep Learning is a subset of machine learning, that utilizes Deep artificial neural networks for learning from large amounts of data
- Tiny Machine Learning or Tiny ML is an area of development in the AI world. It refers to the use of small, lightweight neural networks that can be deployed on resource-constrained devices like microcontrollers.

Introduction

- GSR stands for galvanic skin response and is a method of measuring the skin's electrical conductance.
- GSR allows you to spot such strong emotions by simply attaching two electrodes to two fingers on one hand. It is interesting to create emotion-related projects like sleep quality monitor.
- EMG detector is a bridge that connects the human body and electrical, the sensor gathers small muscle signals then process with 2th amplify and filter, the output signal can be recognized by Arduino.

What is this Project about?

- A mouse fatigue detection dataset will be trained via TensorFlow.
- Sensors will be connected to the Arduino's Wio terminal to measure forearm muscle soreness levels based on GSR (galvanic skin response) and EMG (Electromyography) measurements.
- The Wio Terminal will display the collected muscle soreness data from its integrated TFT LCD screen. Successfully trained, the screen should display the muscle soreness data in three levels:
  - Relaxed
  - Tense
  - Exhausted

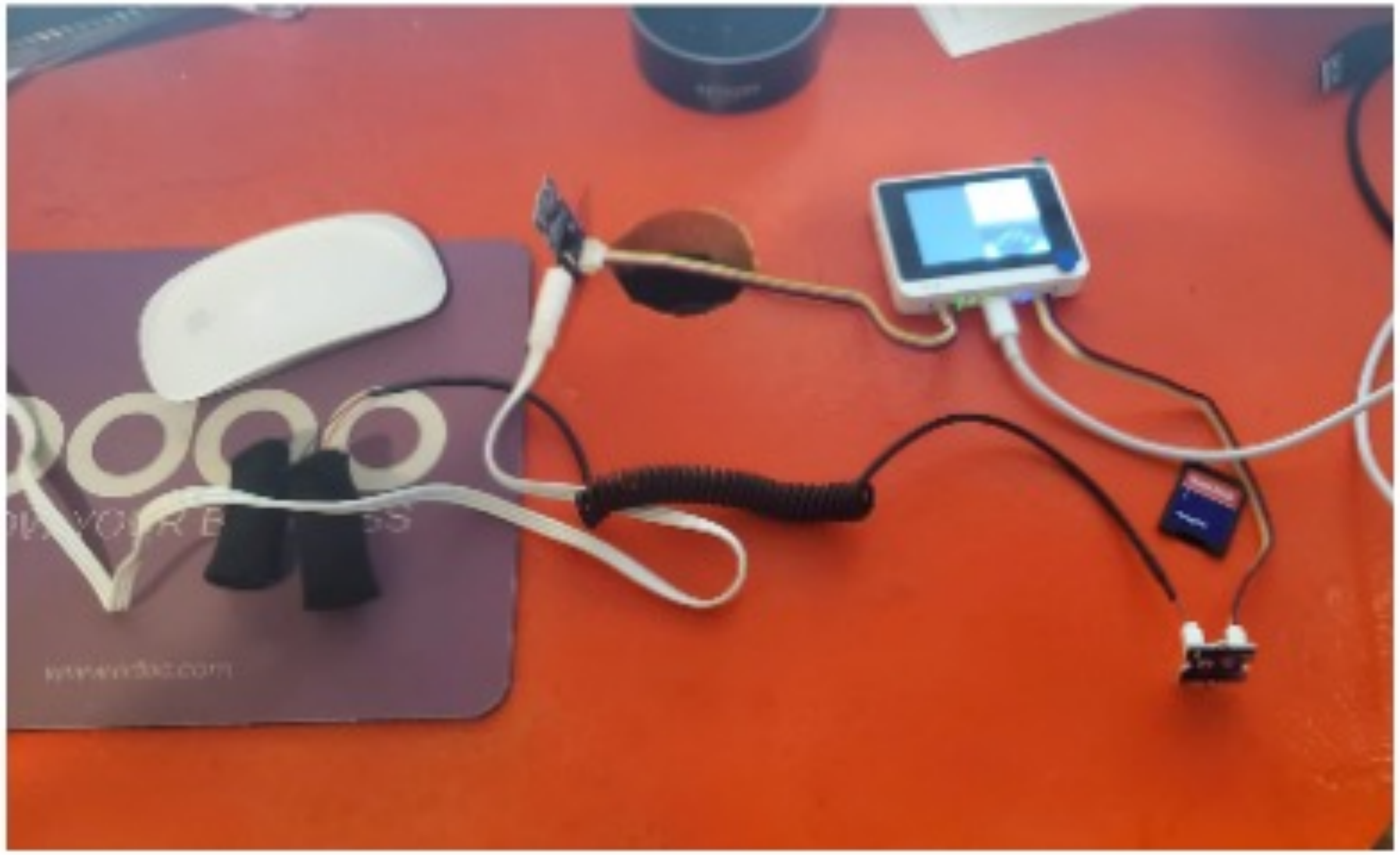
What is this Project about?

- After training and testing the neural network model:
- We converted it from a TensorFlow Keras H5 model to a C array (.h file) to execute the model on Wio Terminal. Therefore, the device is capable of detecting precise forearm muscle soreness levels (classes) by running the model independently.
- After scaling (normalizing) and preprocessing inputs, We obtained two input variables and one label for each data record in the data set.

  GSR,EMG and Soreness Level
- Then, We built an artificial neural network model with TensorFlow and trained it with the training data set to acquire the best results and predictions possible.
- **Focus on Supervised Learning a model uses the input to generate the appropriate output (the model sees the labels in training).**

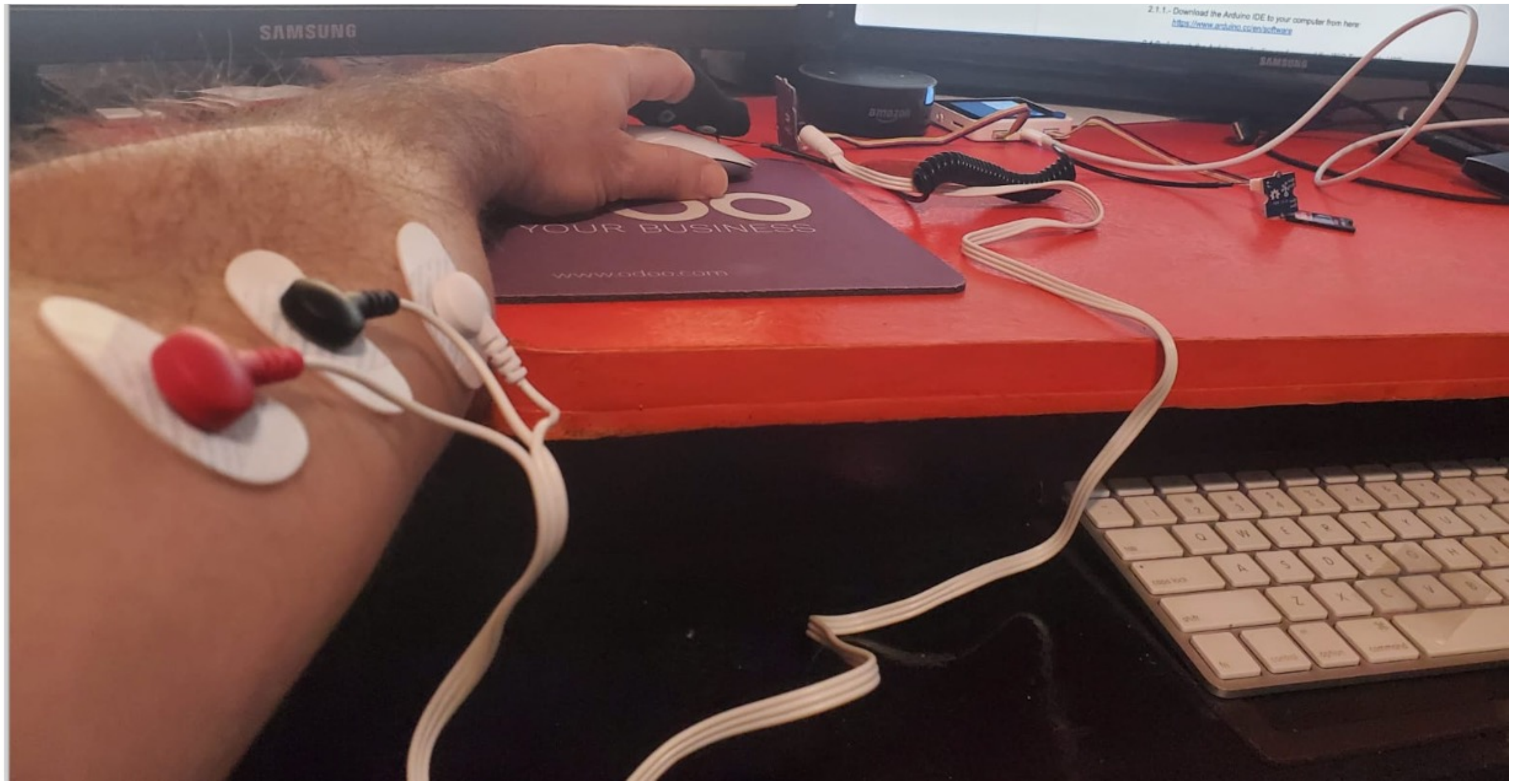
WIO Terminal: https://wiki.seeedstudio.com/Wio-Terminal-Getting-Started/

How to connect Galvanic skin response GSR sensor to WIO terminal



ttps://wiki.seeedstudio.com/Grove-GSR_Sensor/

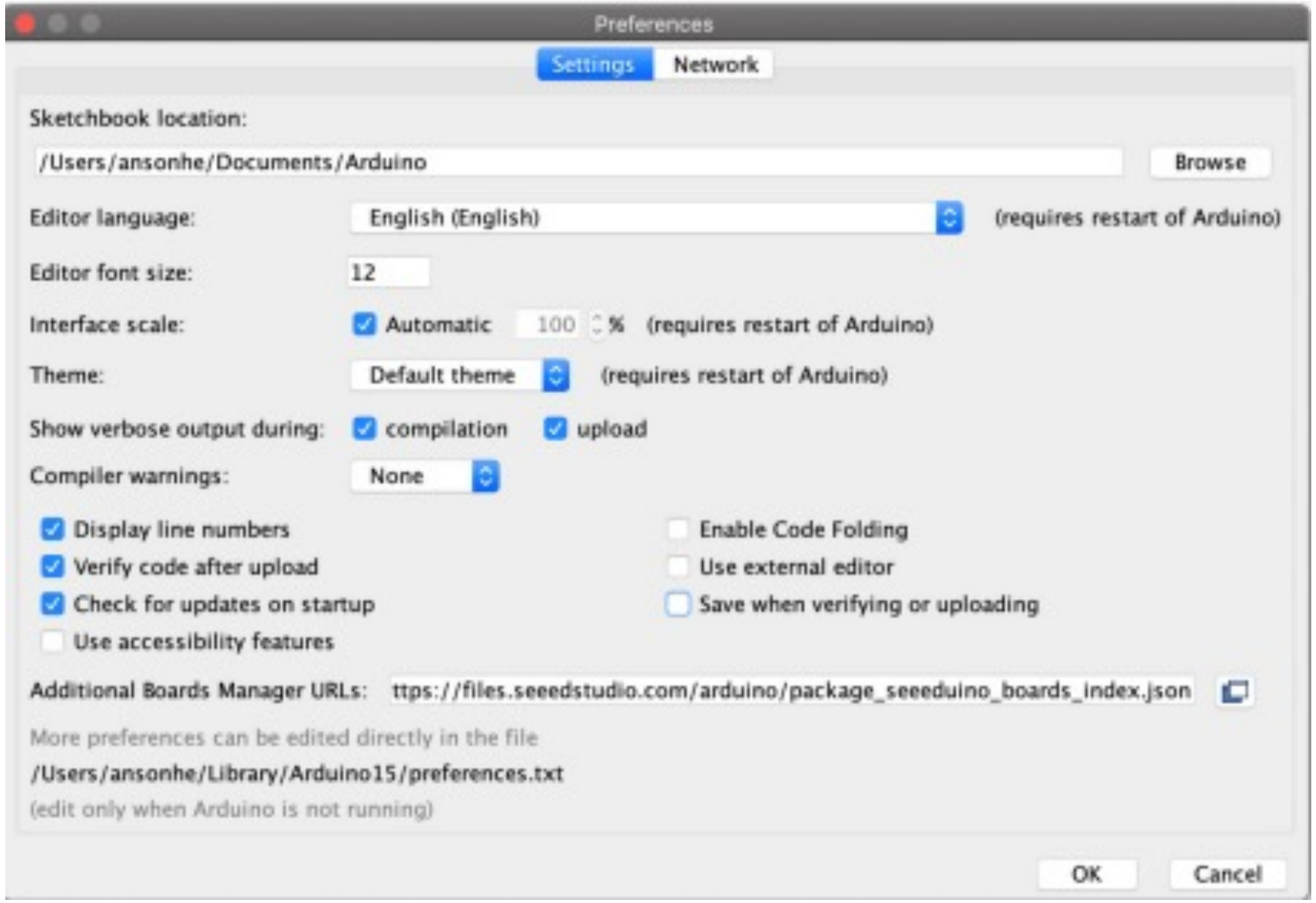
Finally, tack the three electrodes to your muscle, and keep a distance between each electrode.



Arduino IDE 1.8.19

- Download the Arduino IDE to your computer from here: https://www.arduino.cc/en/software
- Launch the Arduino application and connect the WIO Terminal to USB:
- Double-click the Arduino IDE application you have previously downloaded.
- Add WIO Terminal Board Library
- Go to Arduino File | Preferences
- Add Additional Boards Manager URL:
- https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json
-

Add Additional Boards Manager URL:



Sketchbook location:
/Users/ansonhe/Documents/Arduino

Editor language: English (English) (requires restart of Arduino)

Editor font size: 12

Interface scale: ✓ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ✓ compilation ✓ upload

Compiler warnings: None

✓ Display line numbers
✓ Verify code after upload
✓ Check for updates on startup
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
☐ Save when verifying or uploading

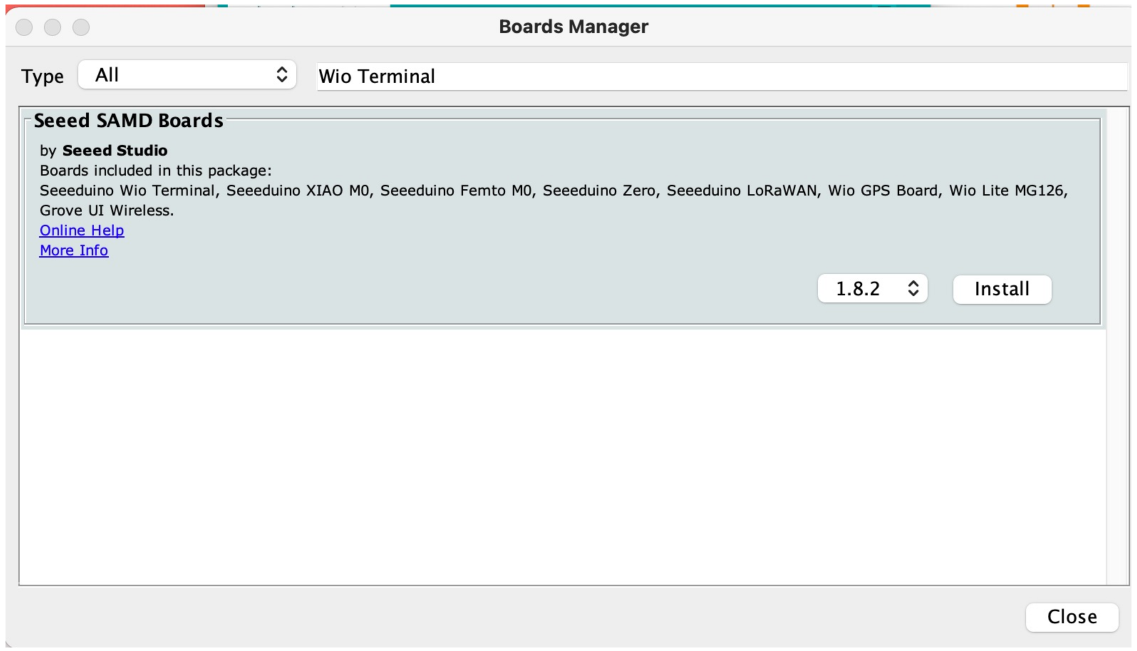
Additional Boards Manager URLs: ttps://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json

More preferences can be edited directly in the file
/Users/ansonhe/Library/Arduino15/preferences.txt
(edit only when Arduino is not running)

Set up the Seeed SAMD Arduino Core

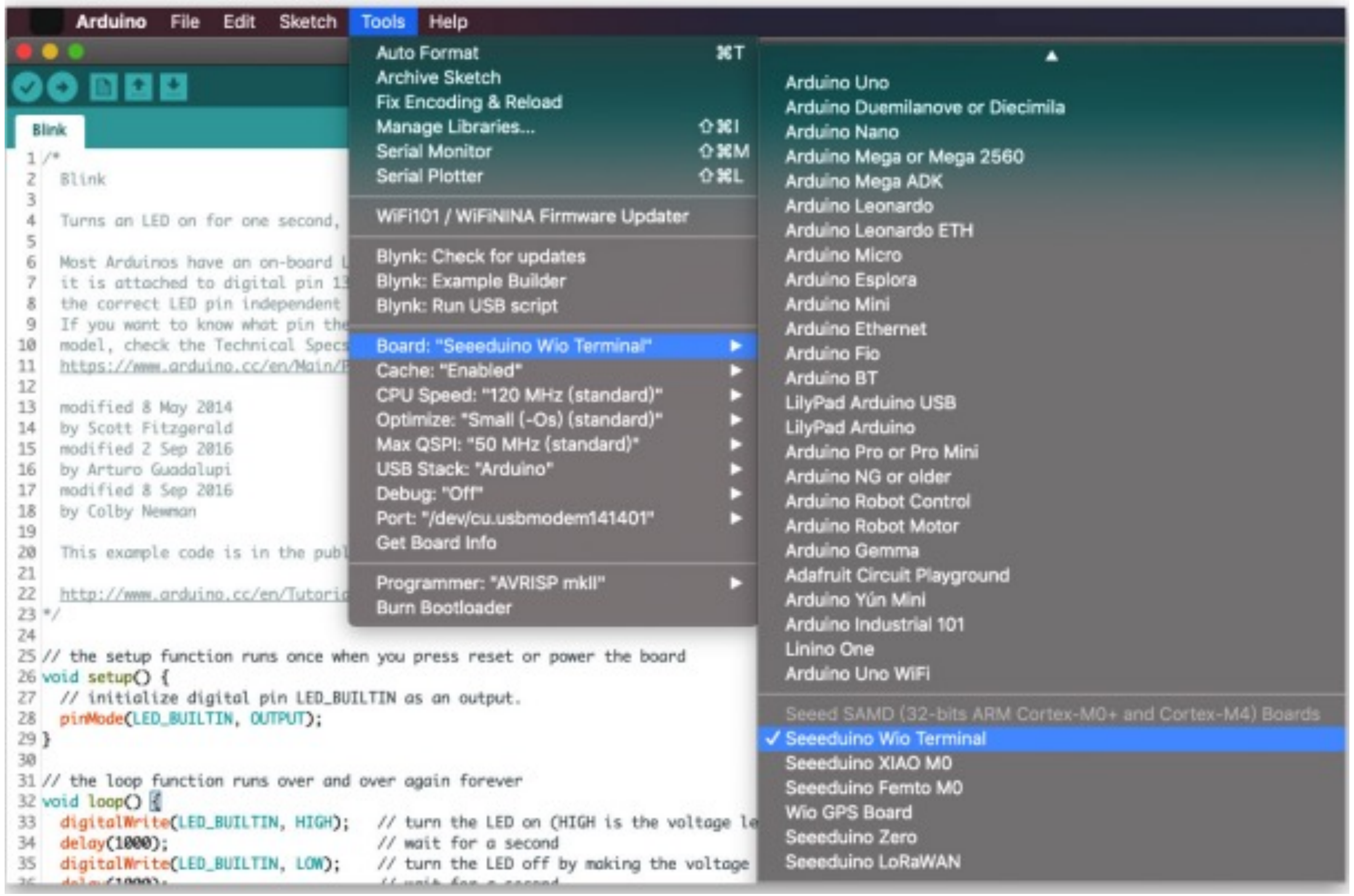
Open the Arduino IDE, click Tools | Board | Boards Manager, and search for Wio Terminal in the search box.

Then, install Seeed SAMD Boards



Seeed SAMD Boards
by **Seeed Studio**
Boards included in this package:
Seeeduino Wio Terminal, Seeeduino XIAO M0, Seeeduino Femto M0, Seeeduino Zero, Seeeduino LoRaWAN, Wio GPS Board, Wio Lite MG126, Grove UI Wireless.
Online Help
More Info

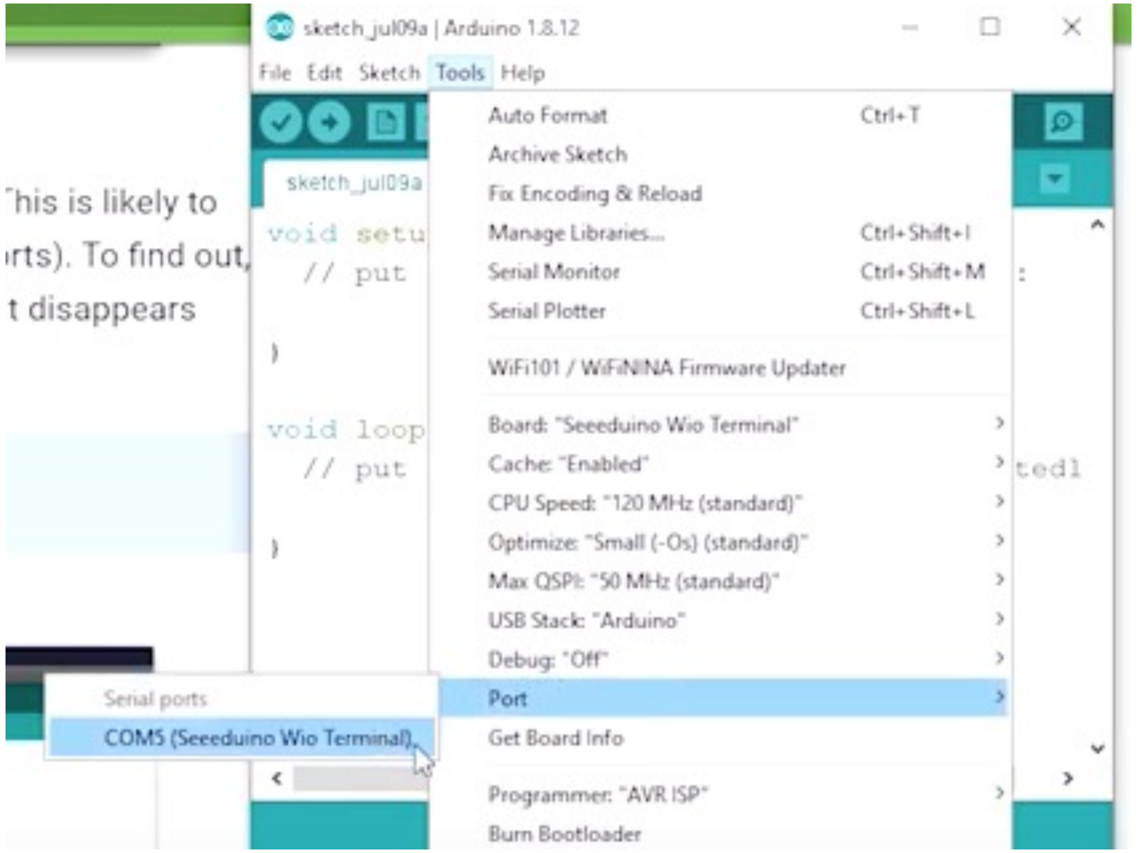
Check the Installed Seeed SAMD Boards



Board: "Seeeduino Wio Terminal"
Cache: "Enabled"
CPU Speed: "120 MHz (standard)"
Optimize: "Small (-Os) (standard)"
Max QSPI: "50 MHz (standard)"
USB Stack: "Arduino"
Debug: "Off"
Port: "/dev/cu.usbmodem141401"
Programmer: "AVRISP mkII"

```
/*
  Blink

  Turns an LED on for one second,

  Most Arduinos have an on-board L
  it is attached to digital pin 13
  the correct LED pin independent
  If you want to know what pin the
  model, check the Technical Specs
  https://www.arduino.cc/en/Main/P

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the publ

  http://www.arduino.cc/en/Tutoria
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage le
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage
```

Selecting board and port


sketch_jul09a | Arduino 1.8.12
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Port
Get Board Info
Burn Bootloader
Serial ports
COM5 (Seeeduino Wio Terminal)

Board: "Seeeduino Wio Terminal"
Cache: "Enabled"
CPU Speed: "120 MHz (standard)"
Optimize: "Small (-Os) (standard)"
Max QSPI: "50 MHz (standard)"
USB Stack: "Arduino"
Debug: "Off"
Programmer: "AVR ISP"

Select the Tools | Board menu entry that corresponds to your Arduino.

Select the Wio Terminal

Installing the SD Card library for Wio Terminal

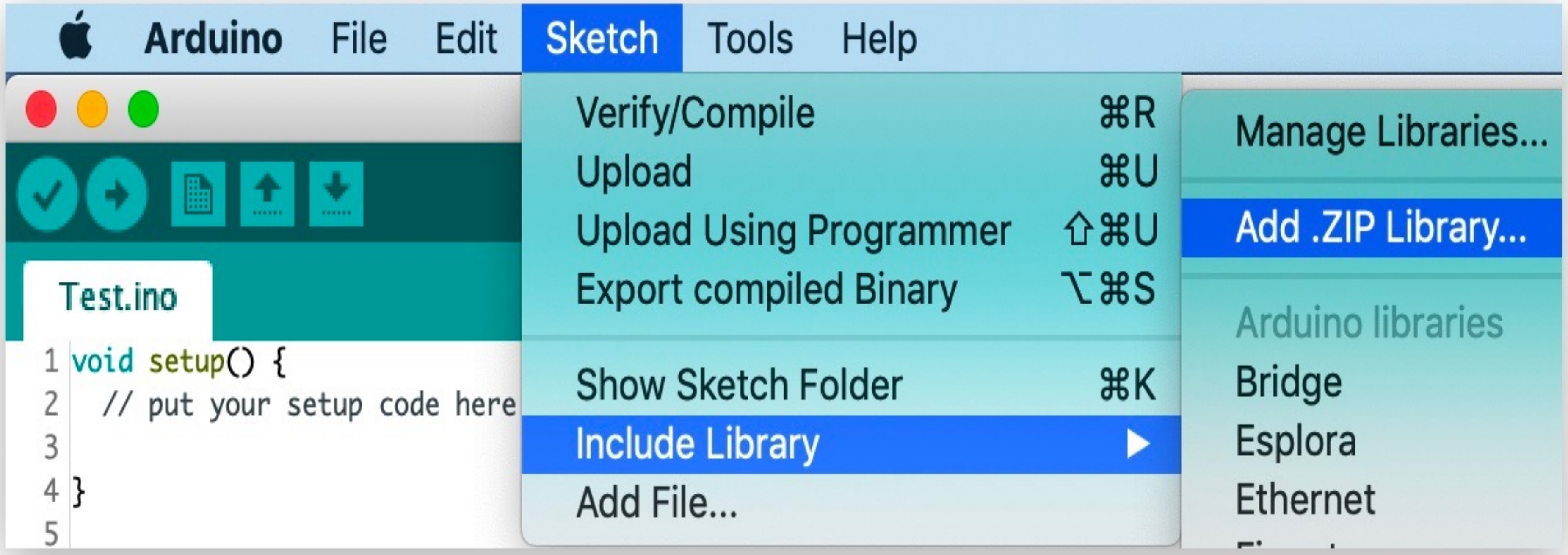
- This presentation introduces how to install the File System library used on Wio Terminal.
- It provides the basic functionality of File operating with the SD card, allowing to Read/Write in or from the SD card using the SPI interface.

Installing the File System Library

Download the entire repo Seeed_Arduino_FS:

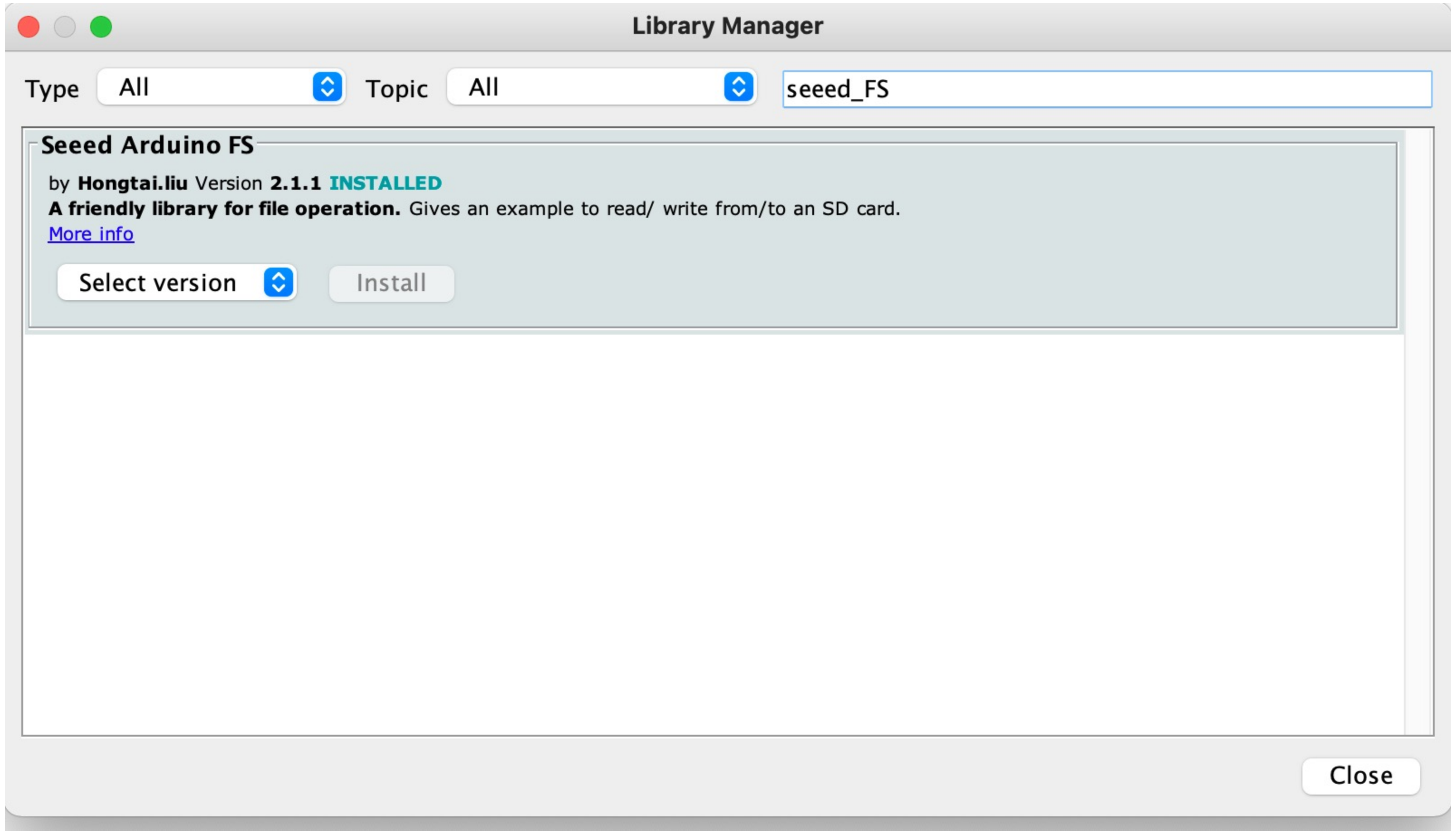
https://github.com/Seeed-Studio/Seeed_Arduino_FS

Open the Arduino IDE: click sketch | Include Library | Add .ZIP Library, and choose the Seeed Arduino FS file that you have just downloaded.



```
void setup() {
  // put your setup code here

}
```

Check on sketch | Library Manager | Search for seeed_FS



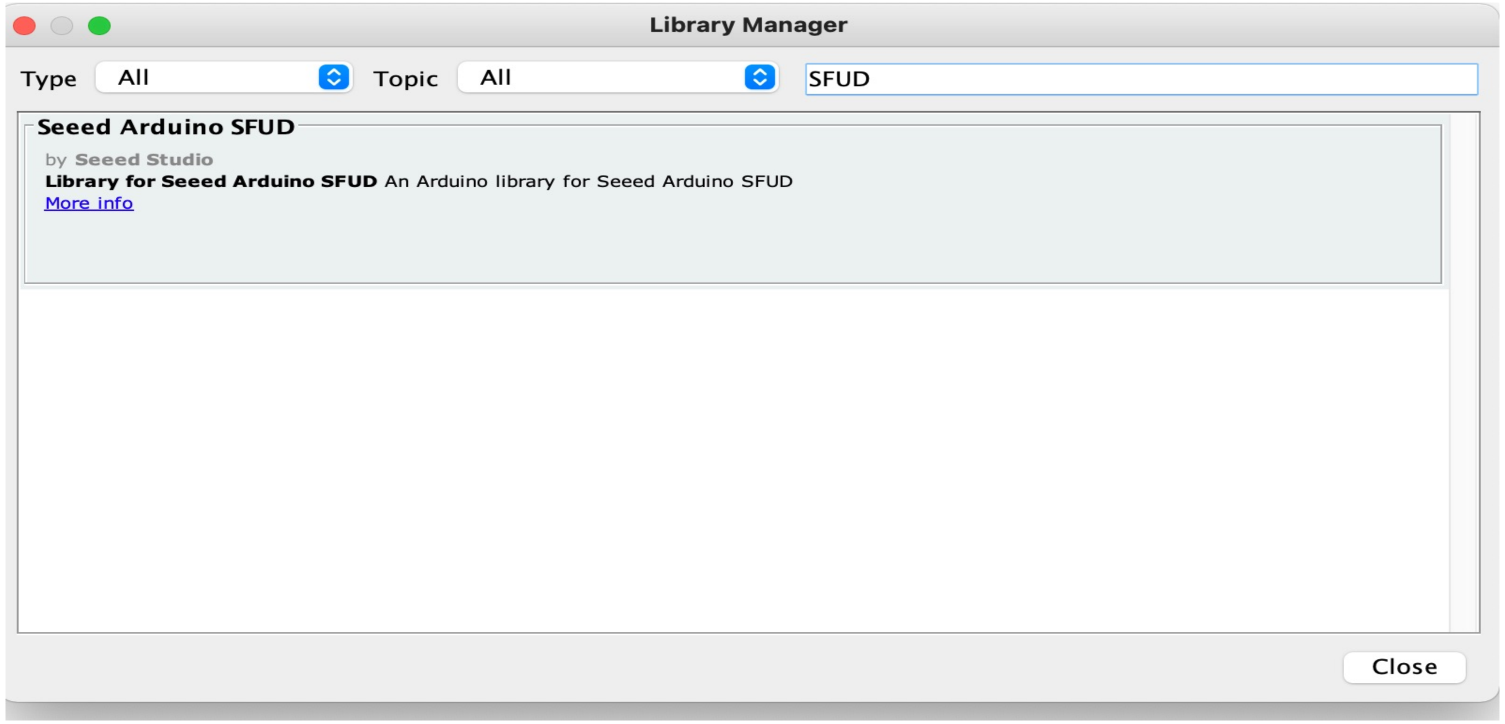
Seeed Arduino FS

by **Hongtai.liu** Version **2.1.1** **INSTALLED**
A friendly library for file operation. Gives an example to read/ write from/to an SD card.
More info

Install the Dependent SFUD Libraries

Download and Install the Library
https://github.com/Seeed-Studio/Seeed_Arduino_SFUD

Check on sketch | Library Manager | Search for SFUD



SFUD

Seeed Arduino SFUD
by **Seeed Studio**
Library for Seeed Arduino SFUD An Arduino library for Seeed Arduino SFUD
More info

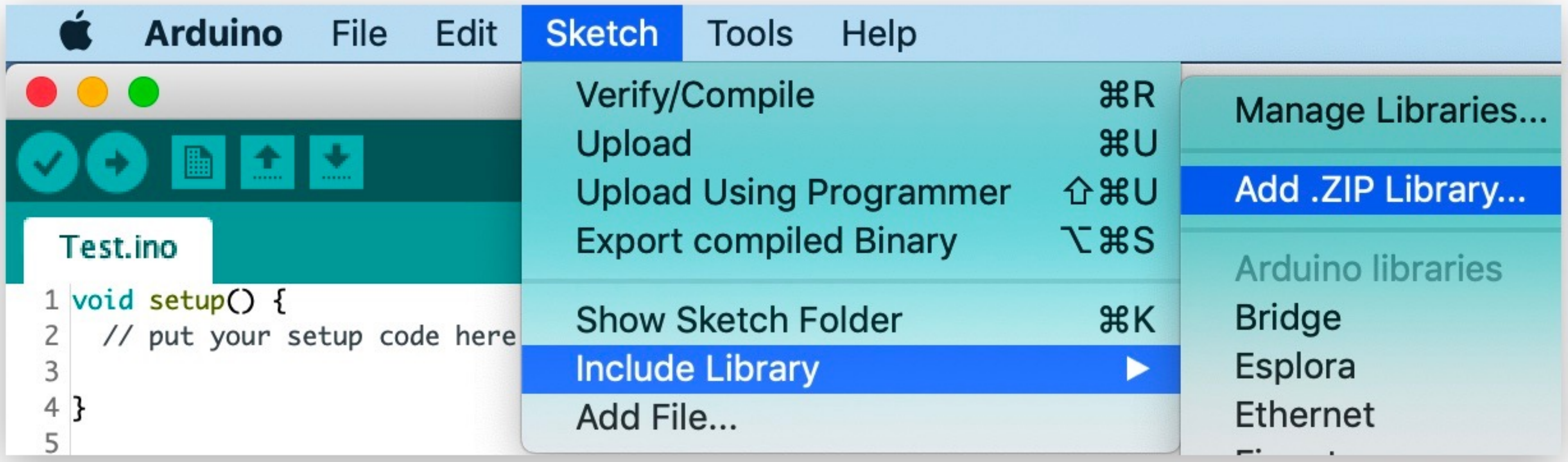
Install Seeed_Arduino_Linechart and TFT LCD Library Separately

- Visit the Seeed_Arduino_LCD repositories and download the entire repo to your local drive.

https://github.com/Seeed-Studio/Seeed_Arduino_Linechart

https://github.com/Seeed-Studio/Seeed_Arduino_LCD

TFT LCD library can be installed to the Arduino IDE. Open the Arduino IDE, and click sketch | Include Library | Add .ZIP Library, and choose the Seeed_Arduino_Linechart, then Seeed_Arduino_LCD file that you've just downloaded



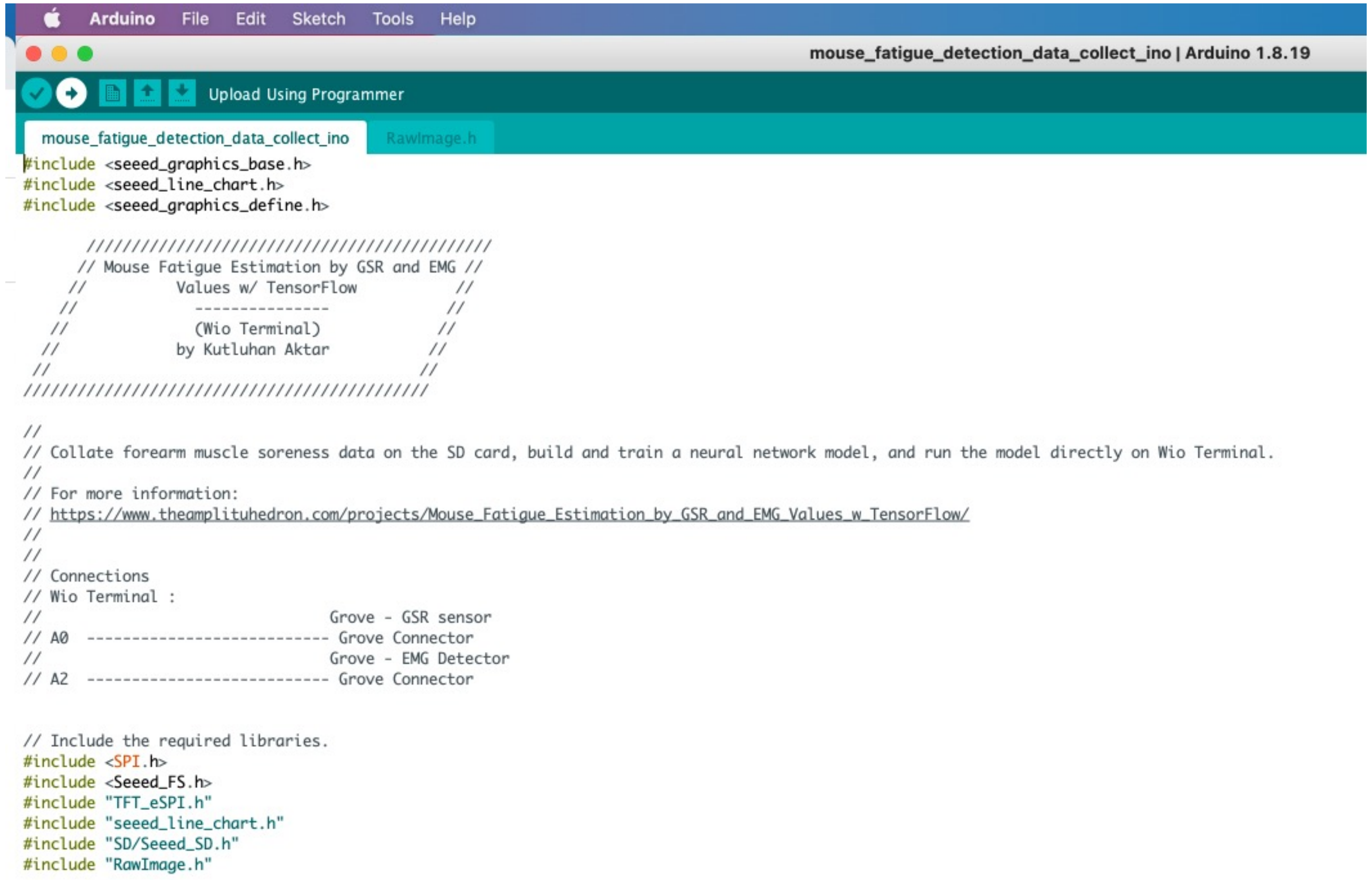
```
void setup() {
  // put your setup code here

}
```

Install the project

- To display the converted BMP files on the TFT LCD screen, move them to the SD card.
- Then, copy the RawImage.h file to the sketches on the Arduino IDE.
- https://files.seeedstudio.com/wiki/Wio-Terminal/res/RawImage.h

- Now, simply click the Upload button in the environment. Wait a few seconds and if the upload is successful, the message "Done uploading." will appear in the status bar.

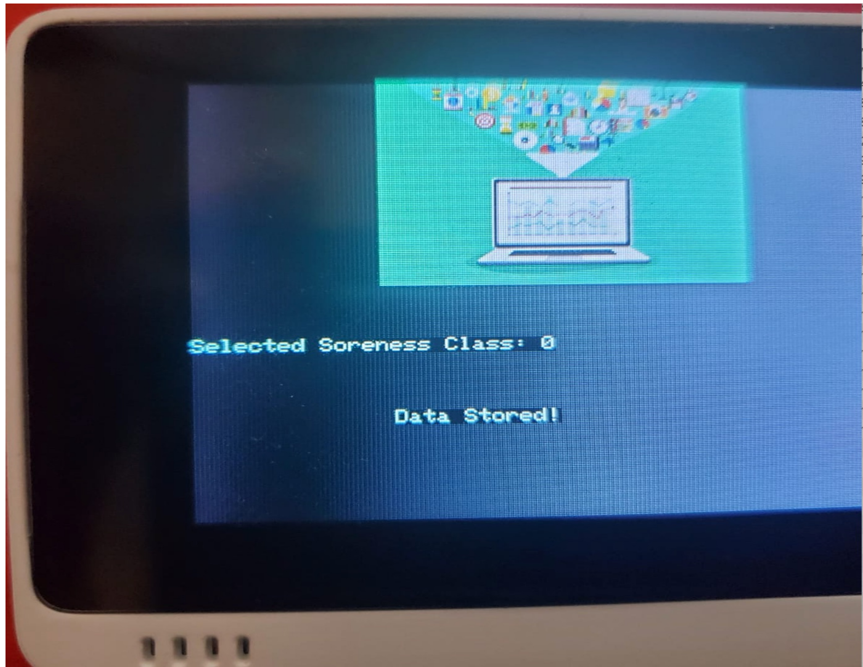
Uploading the program to the WIO Terminal


Arduino
File
Edit
Sketch
Tools
Help
mouse_fatigue_detection_data_collect_ino | Arduino 1.8.19
Upload Using Programmer
mouse_fatigue_detection_data_collect_ino
RawImage.h

```
#include <seeed_graphics_base.h>
#include <seeed_line_chart.h>
#include <seeed_graphics_define.h>

       /////////////////////////////////////////////
      // Mouse Fatigue Estimation by GSR and EMG //
     //          Values w/ TensorFlow           //
    //            ---------------             //
   //              (Wio Terminal)            //
  //             by Kutluhan Aktar          //
 //                                        //
/////////////////////////////////////////////

//
// Collate forearm muscle soreness data on the SD card, build and train a neural network model, and run the model directly on Wio Terminal.
//
// For more information:
// https://www.theamplituhedron.com/projects/Mouse_Fatigue_Estimation_by_GSR_and_EMG_Values_w_TensorFlow/
//
//
// Connections
// Wio Terminal :
//                                  Grove - GSR sensor
// A0  --------------------------- Grove Connector
//                                  Grove - EMG Detector
// A2  --------------------------- Grove Connector


// Include the required libraries.
#include <SPI.h>
#include <Seeed_FS.h>
#include "TFT_eSPI.h"
#include "seeed_line_chart.h"
#include "SD/Seeed_SD.h"
#include "RawImage.h"
```

The device displays real-time GSR and EMG measurements as line charts on the TFT screen

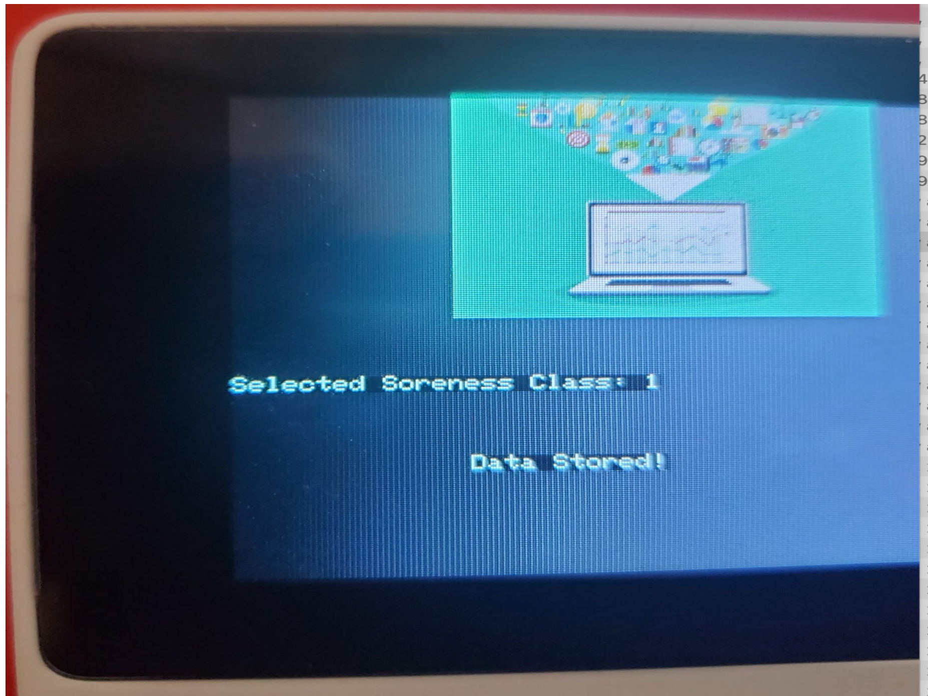
If Button A (configurable button) is pressed, the device appends a new row (data record) to the mouse_fatigue_data_set.csv file on the SD card by adding the Relaxed [0] muscle soreness class under the Soreness data field. Then, if the device saves the data record successfully to the given CSV file on the SD card


Selected Soreness Class: 0

Data Stored!

The device displays real-time GSR and EMG measurements as line charts on the TFT screen

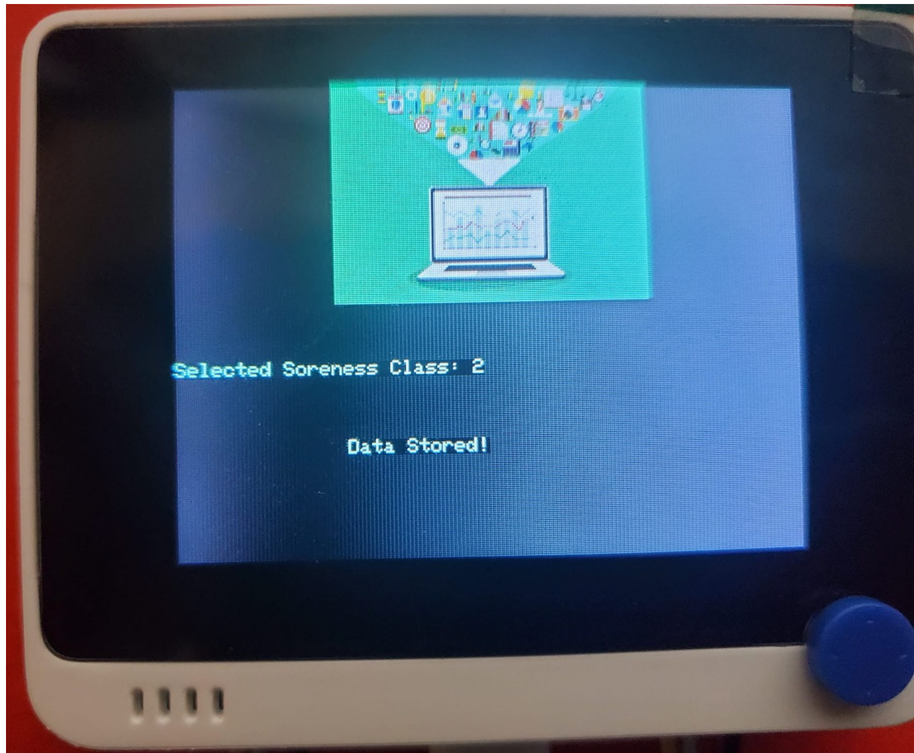
If Button B (configurable button) is pressed, the device appends a new row (data record) to the mouse_fatigue_data_set.csv file on the SD card by adding the Tense [1] muscle soreness class under the Soreness data field. Then, if the device saves the data record successfully to the given CSV file on the SD card


Selected Soreness Class: 1

Data Stored!

The device displays real-time GSR and EMG measurements as line charts on the TFT screen

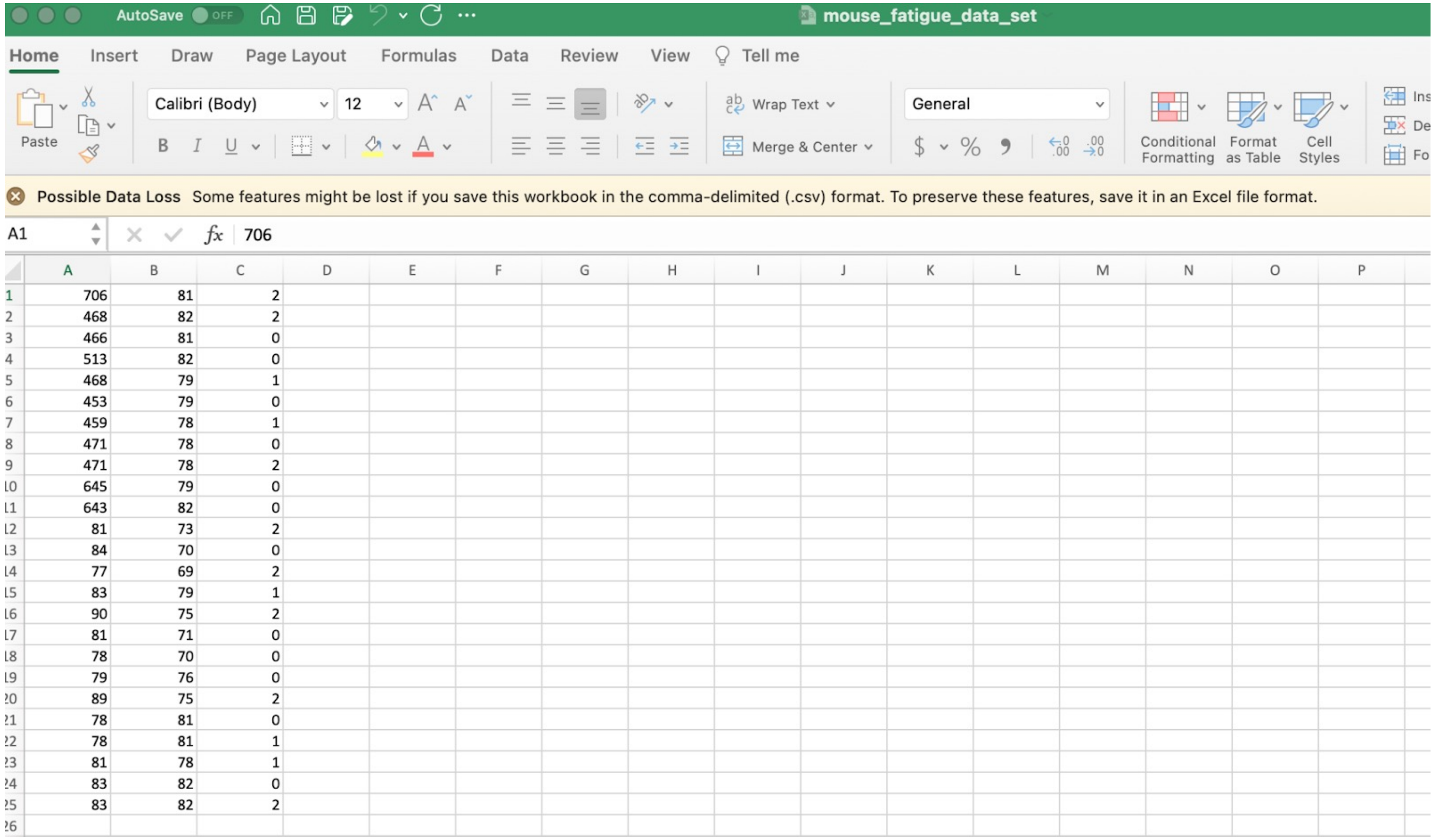
If Button C (configurable button) is pressed, the device appends a new row (data record) to the mouse_fatigue_data_set.csv file on the SD card by adding the Exhausted [2] muscle soreness class under the Soreness data field. Then, if the device saves the data record successfully to the given CSV file on the SD card



Selected Soreness Class: 2

Data Stored!

After logging the collected forearm muscle soreness data in a CSV file on the SD card



Possible Data Loss Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

A1 fx 706

	A	B	C
1	706	81	2
2	468	82	2
3	466	81	0
4	513	82	0
5	468	79	1
6	453	79	0
7	459	78	1
8	471	78	0
9	471	78	2
10	645	79	0
11	643	82	0
12	81	73	2
13	84	70	0
14	77	69	2
15	83	79	1
16	90	75	2
17	81	71	0
18	78	70	0
19	79	76	0
20	89	75	2
21	78	81	0
22	78	81	1
23	81	78	1
24	83	82	0
25	83	82	2
26			

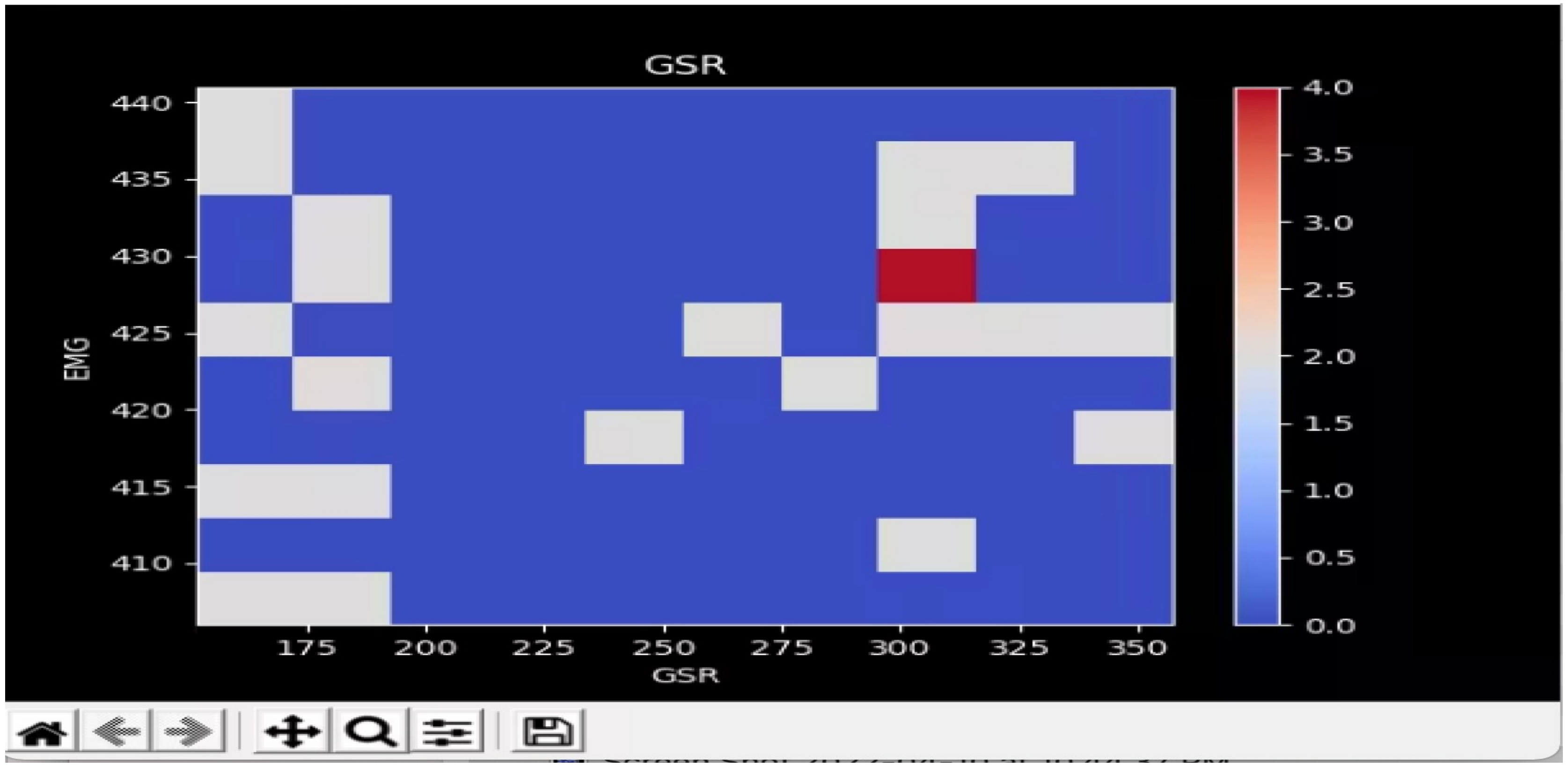
Run the Python Application

To convert the model from a TensorFlow Keras H5 model to a C array (.h file) so as to run it successfully on Wio Terminal, Run the application in Python. As shown below, the application consists of three code files and three folders:

- main.py
- test_data.py
- tflite_to_c_array.py
- /data
- /model
- /bmp

Visualizing the forearm muscle soreness data set

- After extracting it from the *mouse_fatigue_data_set.csv* file saved in the *data* folder



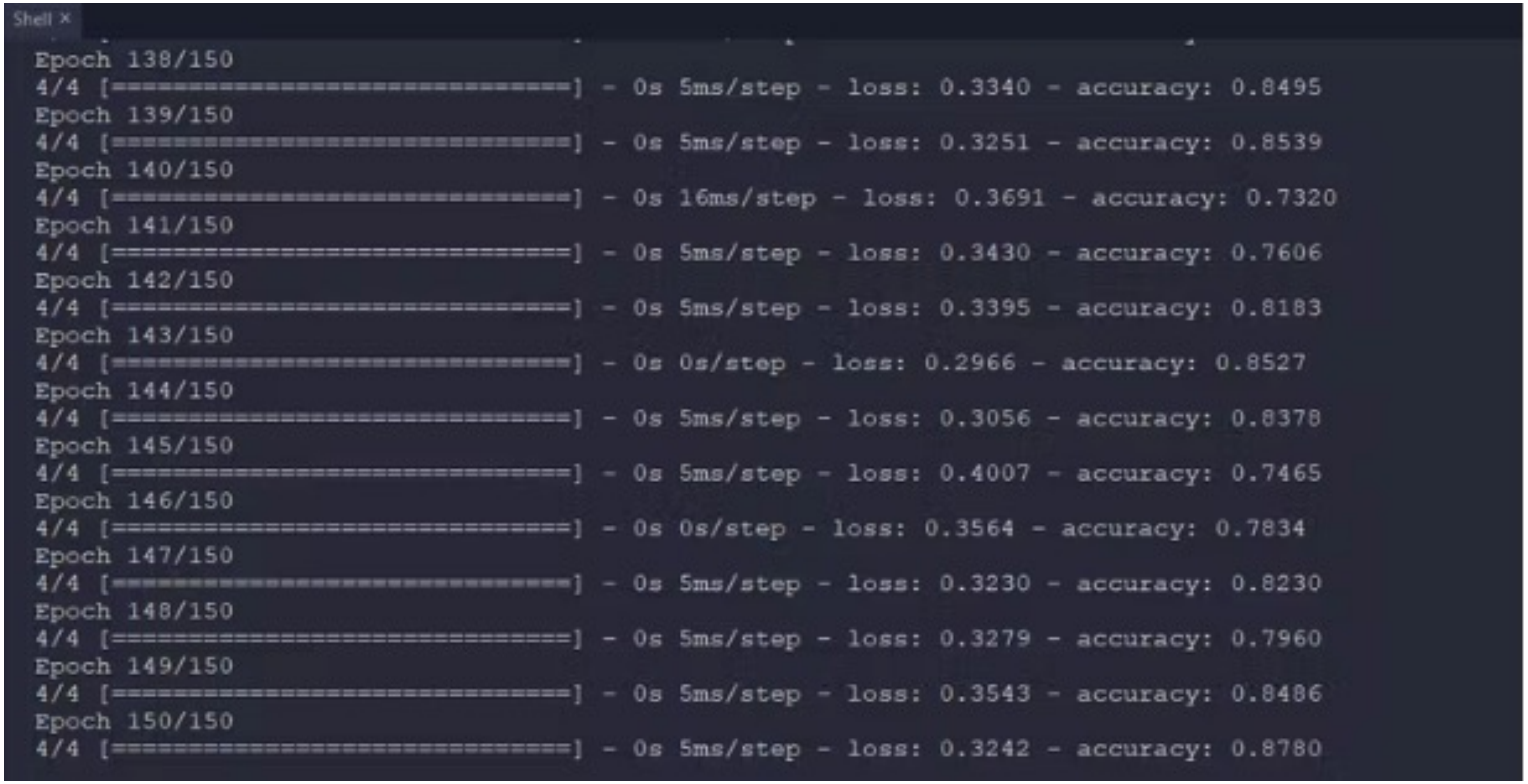
Assigning labels and scaling (normalizing) data records to create inputs

- We create inputs from data records to train the neural network model. Therefore, we utilized these two data elements to create inputs:
  - GSR
  - EMG
- Then, we scaled (normalized) each data element to format them properly and thus extracted these scaled data elements from the data set for each data record:
  - scaled_GSR
  - scaled_EMG
- In the *scale_data_and_define_inputs* function, divide every data element into their required values so as to make them smaller than or equal to 1.
- Then, create inputs with the scaled data elements, append them to the *inputs* array, and convert this array to a NumPy array by using the *asarray* function.
- Each input includes two parameters [shape=(2,)]:
- *[0.152, 0.407]*

Training the model (ANN) on muscle soreness levels (classes)

- Since the forearm muscle soreness data set is already limited, We decided to utilize all of the data set as the training data set instead of splitting it into training and testing data sets. Thus, We created a separate testing data set in the *test_data.py* file.
- After defining the training data set, we scaled (normalized) the testing data set inputs to format them appropriately.
- Then, We built my artificial neural network model (ANN) by utilizing Keras and trained it with the training set for 150 epochs.
- After training with the training set (inputs and labels), the accuracy of the neural network model was between *0.83* and *0.88*.

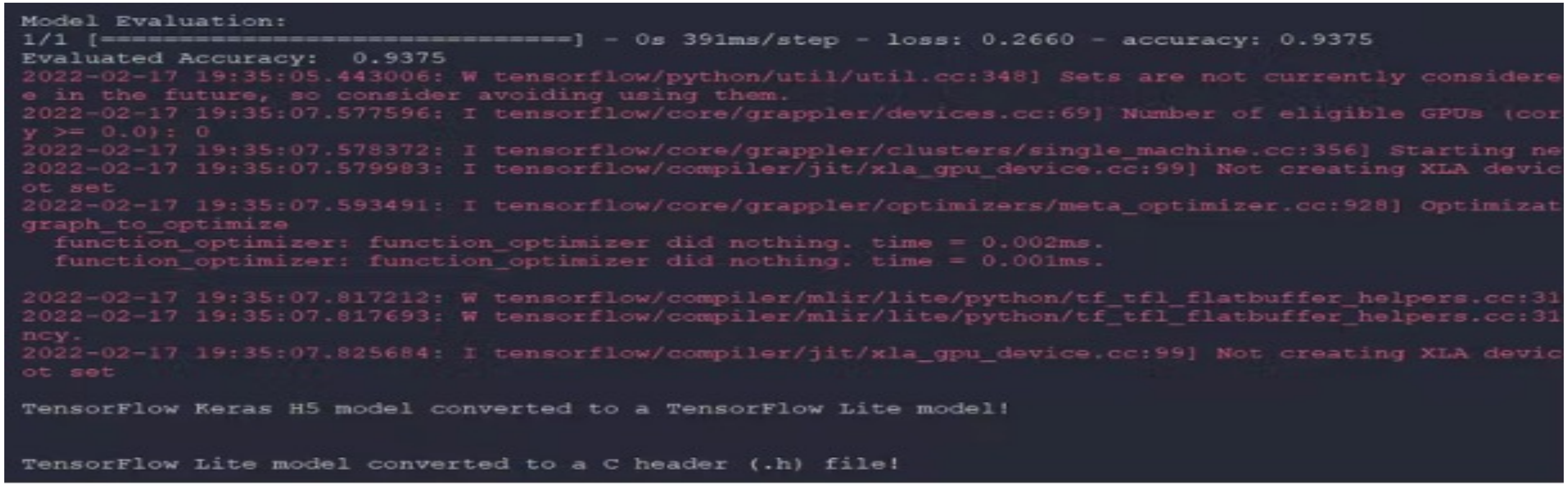
Accuracy of the neural network model was between *0.83* and *0.88*.



```
Epoch 138/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3340 - accuracy: 0.8495
Epoch 139/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3251 - accuracy: 0.8539
Epoch 140/150
4/4 [==============================] - 0s 16ms/step - loss: 0.3691 - accuracy: 0.7320
Epoch 141/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3430 - accuracy: 0.7606
Epoch 142/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3395 - accuracy: 0.8183
Epoch 143/150
4/4 [==============================] - 0s 0s/step - loss: 0.2966 - accuracy: 0.8527
Epoch 144/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3056 - accuracy: 0.8378
Epoch 145/150
4/4 [==============================] - 0s 5ms/step - loss: 0.4007 - accuracy: 0.7465
Epoch 146/150
4/4 [==============================] - 0s 0s/step - loss: 0.3564 - accuracy: 0.7834
Epoch 147/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3230 - accuracy: 0.8230
Epoch 148/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3279 - accuracy: 0.7960
Epoch 149/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3543 - accuracy: 0.8486
Epoch 150/150
4/4 [==============================] - 0s 5ms/step - loss: 0.3242 - accuracy: 0.8780
```

Evaluating the model accuracy and converting the model to a C array

- After building and training my artificial neural network model, We tested its accuracy and validity by utilizing the testing data set (inputs and labels).
- The evaluated accuracy of the model was *0.9375*.



```
Model Evaluation:
1/1 [==============================] - 0s 391ms/step - loss: 0.2660 - accuracy: 0.9375
Evaluated Accuracy:  0.9375
2022-02-17 19:35:05.443006: W tensorflow/python/util/util.cc:348] Sets are not currently considere
e in the future, so consider avoiding using them.
2022-02-17 19:35:07.577596: I tensorflow/core/grappler/devices.cc:69] Number of eligible GPUs (cor
y >= 0.0): 0
2022-02-17 19:35:07.578372: I tensorflow/core/grappler/clusters/single_machine.cc:356] Starting ne
2022-02-17 19:35:07.579983: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devic
ot set
2022-02-17 19:35:07.593491: I tensorflow/core/grappler/optimizers/meta_optimizer.cc:928] Optimizat
graph_to_optimize
  function_optimizer: function_optimizer did nothing. time = 0.002ms.
  function_optimizer: function_optimizer did nothing. time = 0.001ms.

2022-02-17 19:35:07.817212: W tensorflow/compiler/mlir/lite/python/tf_tfl_flatbuffer_helpers.cc:31
2022-02-17 19:35:07.817693: W tensorflow/compiler/mlir/lite/python/tf_tfl_flatbuffer_helpers.cc:31
ncy.
2022-02-17 19:35:07.825684: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devic
ot set

TensorFlow Keras H5 model converted to a TensorFlow Lite model!


TensorFlow Lite model converted to a C header (.h) file!
```

Saved it as a TensorFlow Keras H5 model

- After evaluating the neural network model, We saved it as a TensorFlow Keras H5 model *(mouse_fatigue_level.h5)* to the *model* folder.
- However, running a TensorFlow Keras H5 model on Wio Terminal to make predictions on muscle soreness levels is not eligible and efficient considering size, latency, and power consumption.
- Thus, We converted the neural network model from a TensorFlow Keras H5 model (.h5) to a TensorFlow Lite model (.tflite). Then, We modified the TensorFlow Lite model to create a C array (.h file) to run the model on Wio Terminal successfully.

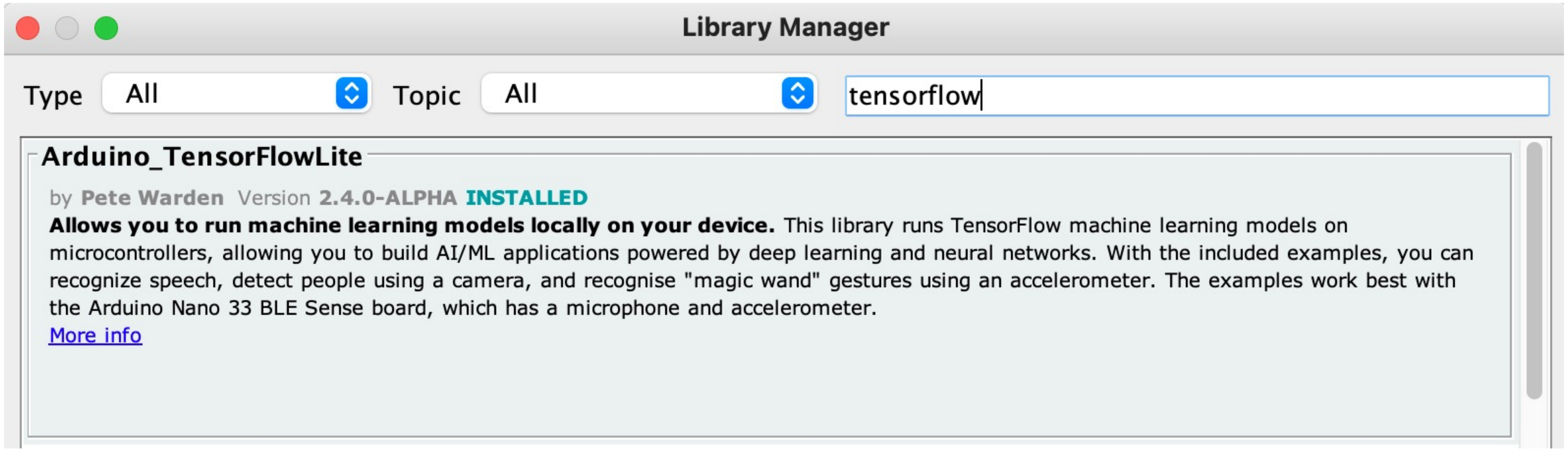
Convert TF model

- In the *convert_TF_model* function, convert the recently trained and evaluated model to a TensorFlow Lite model by applying the TensorFlow Lite converter *(tf.lite.TFLiteConverter.from_keras_model)*.
- Then, save the generated TensorFlow Lite model to the *model* folder *(mouse_fatigue_level.tflite)*.
- Modify the saved TensorFlow Lite model to a C array (.h file) by executing the *hex_to_c_array* function.
- Finally, save the generated C array to the *model* folder *(mouse_fatigue_level.h)*.

Setting up the model on Wio Terminal

- After building, training, and converting the neural network model to a C array (.h file), We needed to upload and run the model directly on Wio Terminal so as to create an easy-to-use and capable device without any dependencies.
- To download the official TensorFlow library on the Arduino IDE, go to *Sketch | Include Library |Manage Libraries…* and search for *TensorFlow*. Then, install the latest version of the *Arduino_TensorFlowLite* library.

Arduino Tensorflow Lite



tensorflow

Arduino_TensorFlowLite

by **Pete Warden** Version **2.4.0-ALPHA** **INSTALLED**

Allows you to run machine learning models locally on your device. This library runs TensorFlow machine learning models on microcontrollers, allowing you to build AI/ML applications powered by deep learning and neural networks. With the included examples, you can recognize speech, detect people using a camera, and recognise "magic wand" gestures using an accelerometer. The examples work best with the Arduino Nano 33 BLE Sense board, which has a microphone and accelerometer.

More info

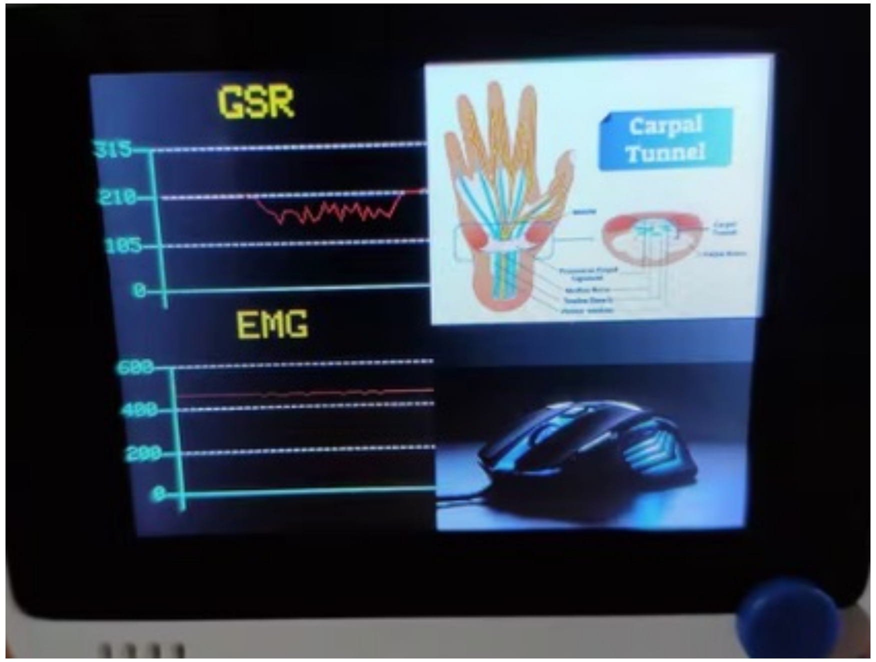
Import the neural network model modified as a C array *(mouse_fatigue_level.h)* to run inferences.

- After installing the TensorFlow library on the Arduino IDE, We needed to import the neural network model modified as a C array *(mouse_fatigue_level.h)* to run inferences.
- To import the TensorFlow Lite model converted as a C array (.h file), go to *Sketch | Add File...*
- After importing the converted model (.h file) successfully to the Arduino IDE, We modified the code in to run the neural network model. Then, We employed the 5-way switch integrated into Wio Terminal to run inferences so as to forecast forearm muscle soreness levels

Running the model on Wio Terminal to make predictions on muscle soreness levels

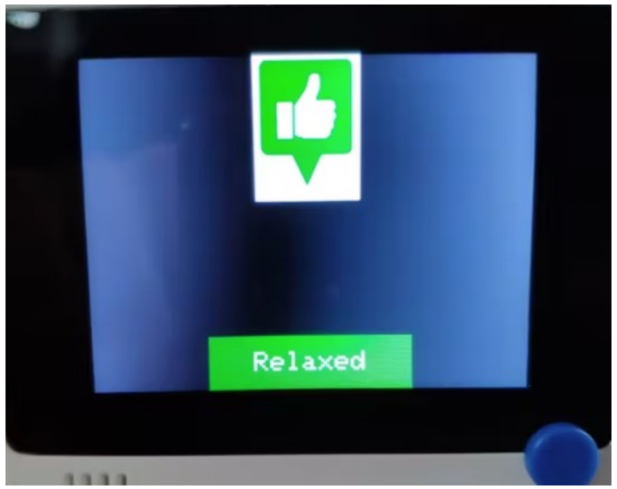
- The neural network model predicts possibilities of labels (muscle soreness classes) for each given input as an array of 3 numbers. They represent the model's *"confidence"* that the given input array corresponds to each of the three different muscle soreness classes based on GSR and EMG measurements [0 - 2], as shown in Step 4:
- 0 — Relaxed
- 1 — Tense
- 2 — Exhausted
- After importing and setting up the neural network model as a C array on Wio Terminal successfully, I utilized the model to run inferences to forecast muscle soreness levels.

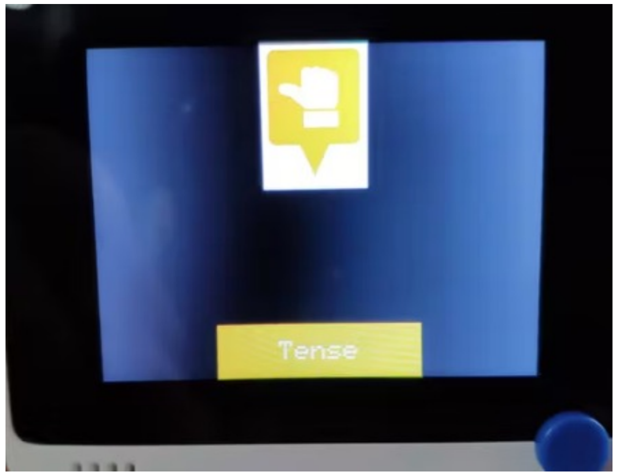
The device displays real-time GSR and EMG measurements as line charts on the TFT screen

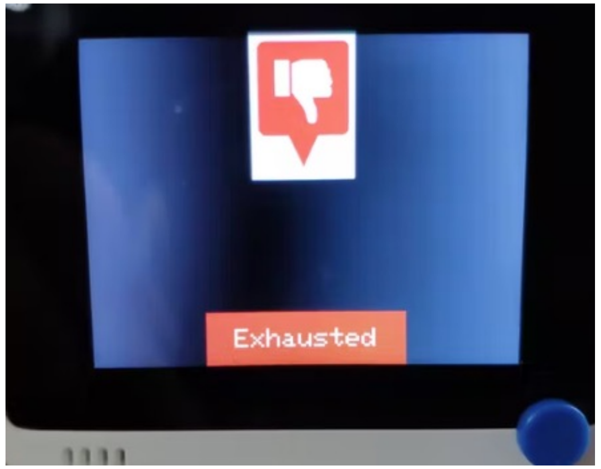


Predicting forearm muscle soreness levels (classes)

- the device displays the output, which represents the most accurate label (muscle soreness class) predicted by the model.
- Each muscle soreness level (class) has a unique image (16-bit BMP file) and color code to be shown on the TFT screen when being detected as the output:







Conclusion

In summary, our project and research in coordination with Artificial Intelligence combined presented a supervised neural network model with TensorFlow in Python. Furthermore, in combination with Tiny ML Wio Terminal compatible with Arduino, helping to understand how to train and implement a Neural network to predict forearm muscle pain levels using Tiny Machine Learning, Arduino, and Tensorflow Lite.

Accessibility

- This simulation will be accessible to the general public at:
- https://github.com/lmarinve/mouse-fatigue-detection.git
-

Citation

- Getting Started with TinyMl, wiki.seeedstudio.com/Wio-Terminal-TinyML-Kit-Course/
- Get started with microcontrollers, www.tensorflow.org/lite/microcontrollers/get_started_low_level
- Grove - GSR Sensor, wiki.seeedstudio.com/Grove-GSR_Sensor/
- Intro to TinyML Part 1: Training a Model for Arduino in TensorFlow www.digikey.com/en/maker/projects/intro-to-tinyml-part-1-training-a-model-for-arduino-in-tensorflow/8f1fc8c0b83d417ab521c48864d2a8ec
- TensorFlow Lite for Microcontrollers, www.tensorflow.org/lite/microcontrollers
- TensorFlow Lite converter, www.tensorflow.org/lite/convert
- Machine Learning vs Neural Networks: Why It's Not One or the Other
- www.verypossible.com/insights/machine-learning-vs.-neural-networks#:~:text=Strictly%20speaking%2C%20a%20neural%20network,usually%20used%20in%20supervised%20learning.