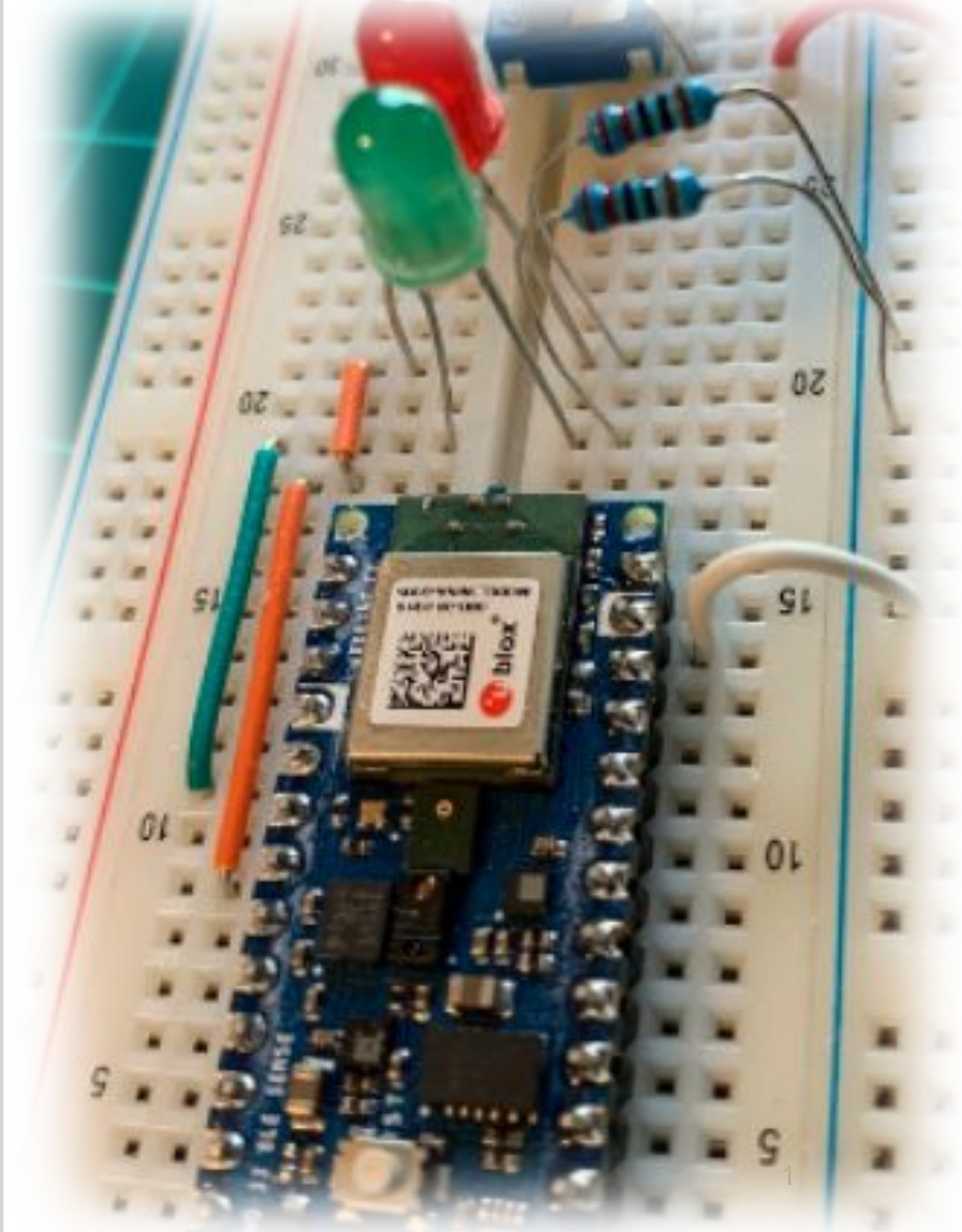
IESTI01 – TinyML

Embedded Machine Learning

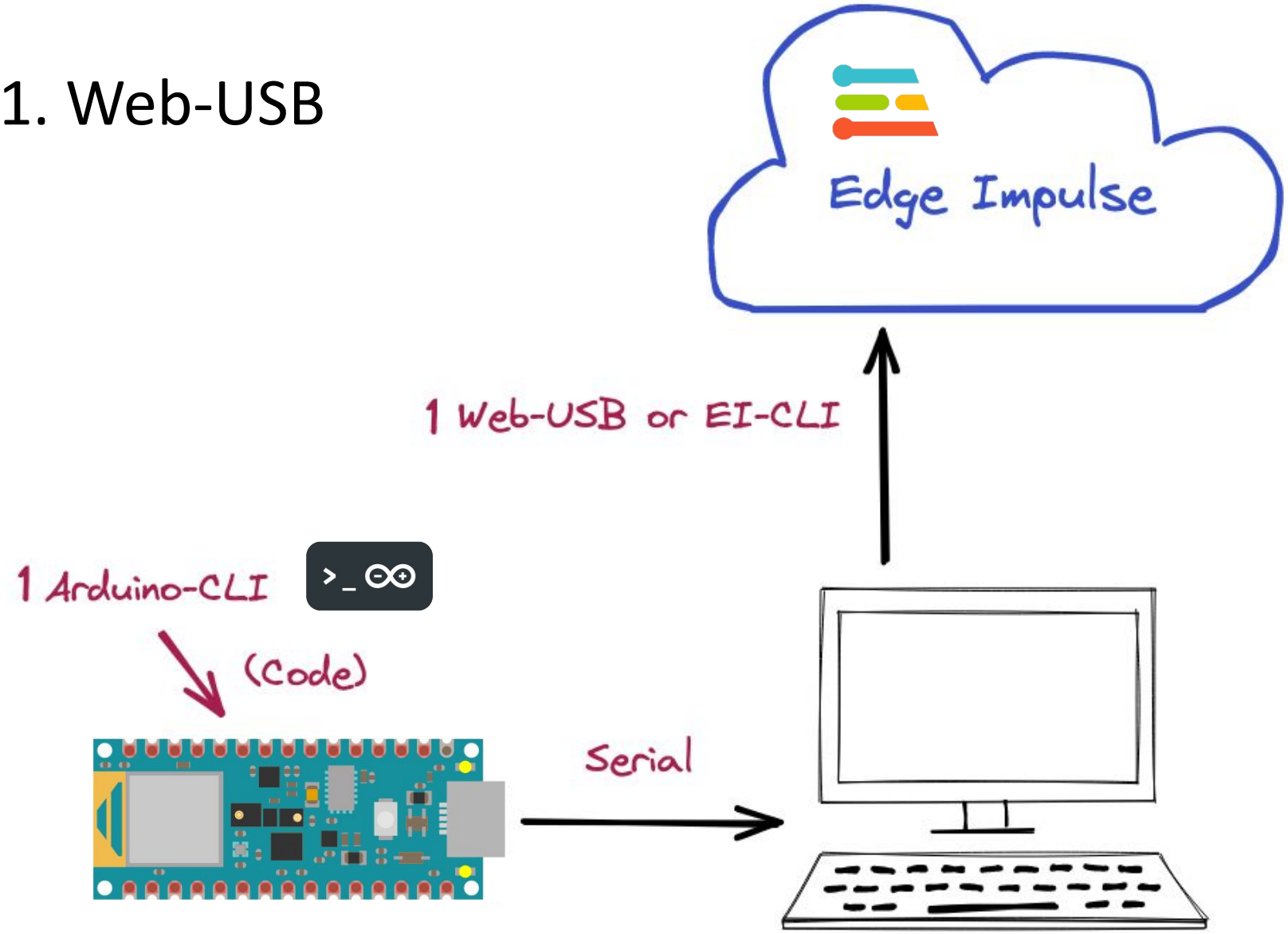
27. Collecting Data with Edge Impulse Studio



Prof. Marcelo Rovai
UNIFEI

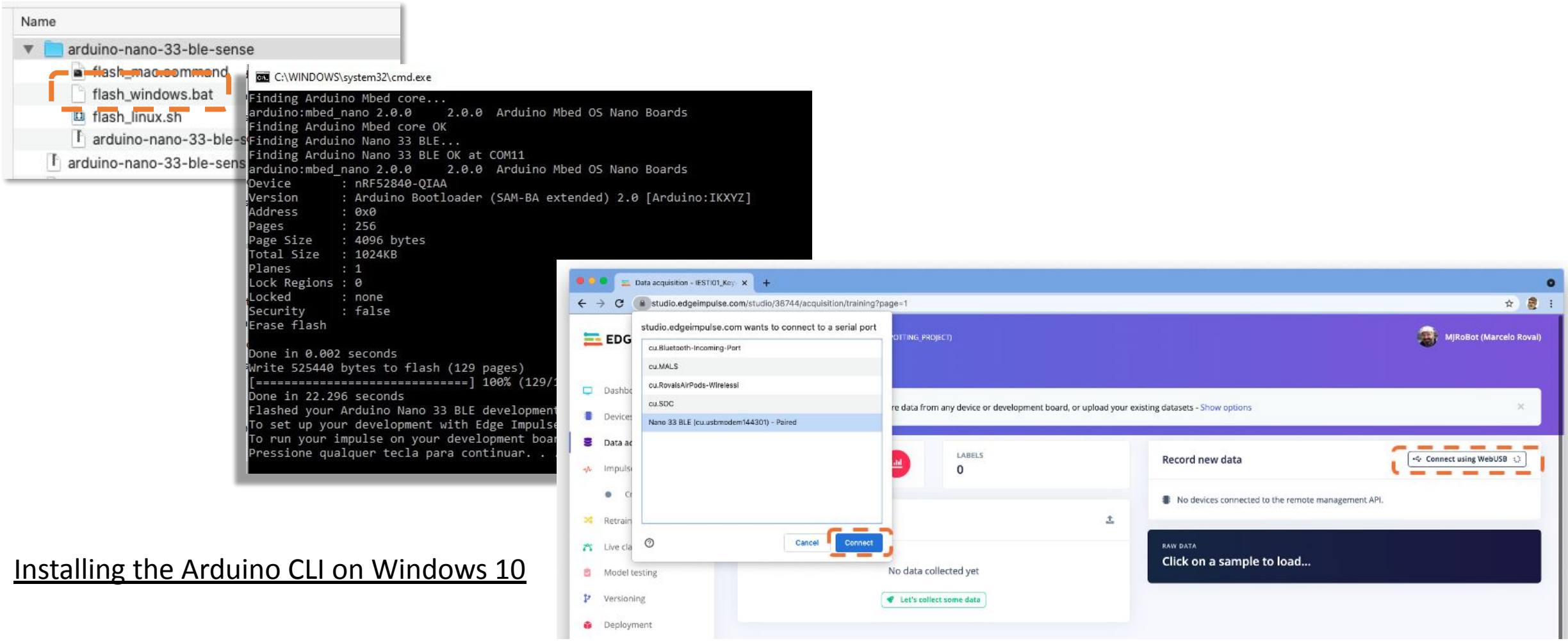
Sensor Data – EI Studio Ingestion
Alternative methods

1. Web-USB



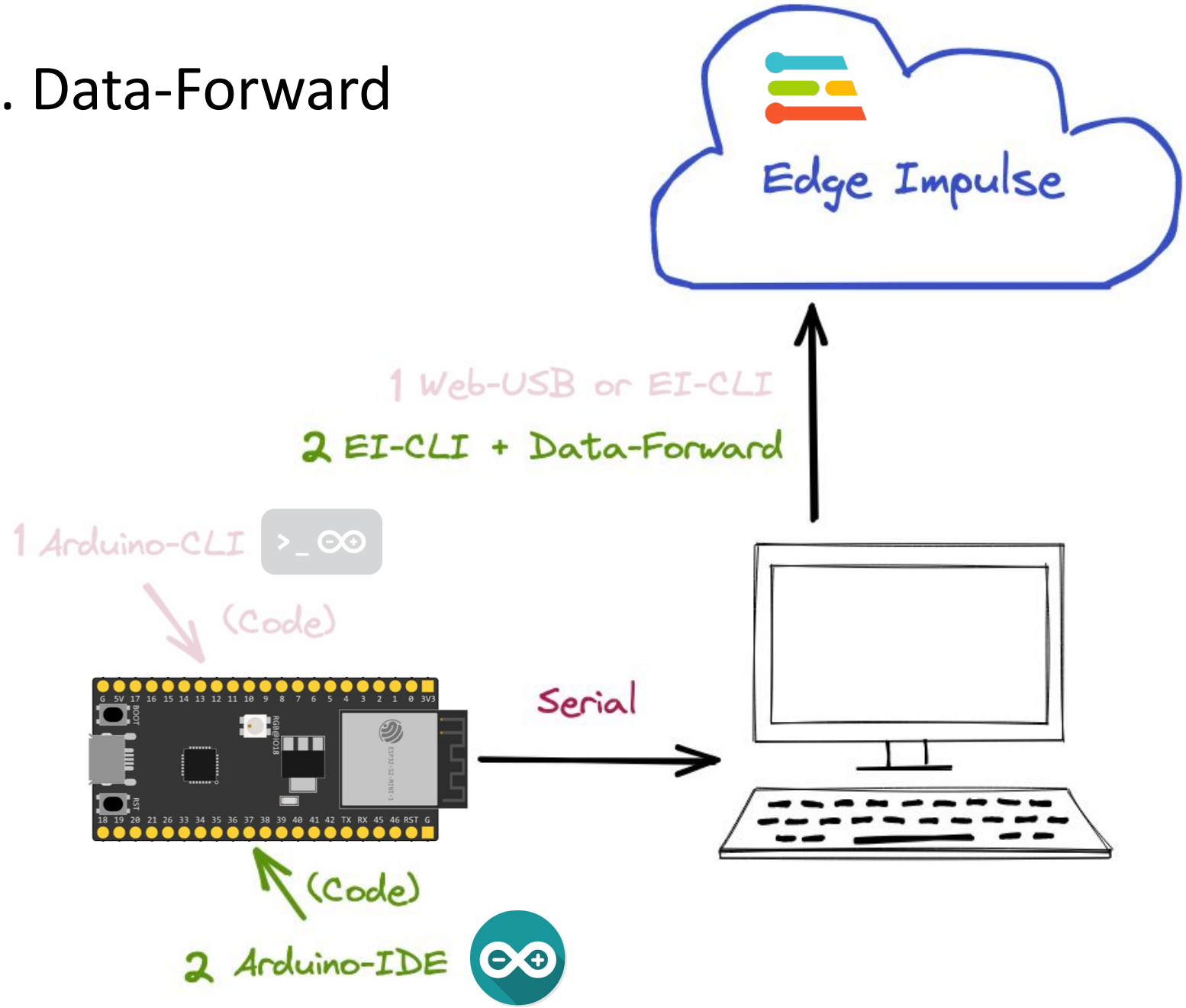
Issue: Limited MCU and sensors

1. Data Ingestion using Arduino-Cli + Web-USB (or EI-CLI)



Installing the Arduino CLI on Windows 10

2. Data-Forward

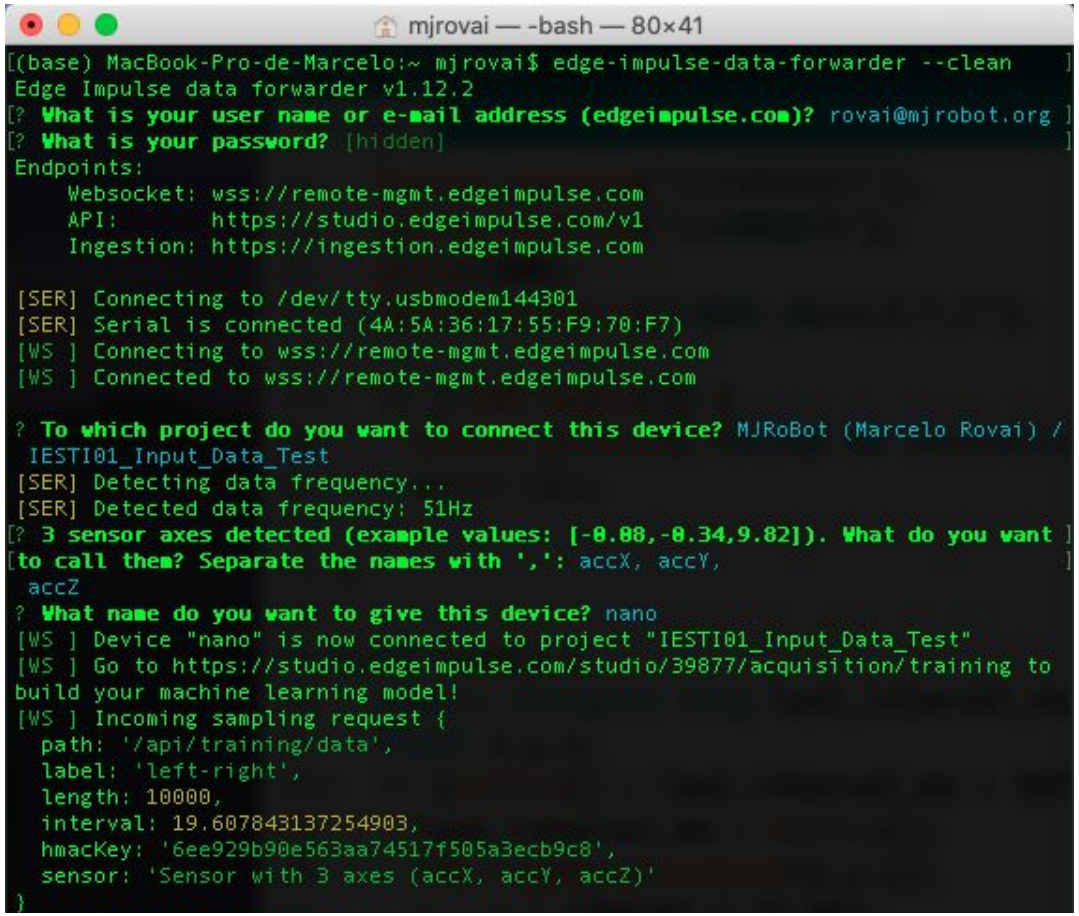


2. Data Ingestion using EI-Cli + Data Forward

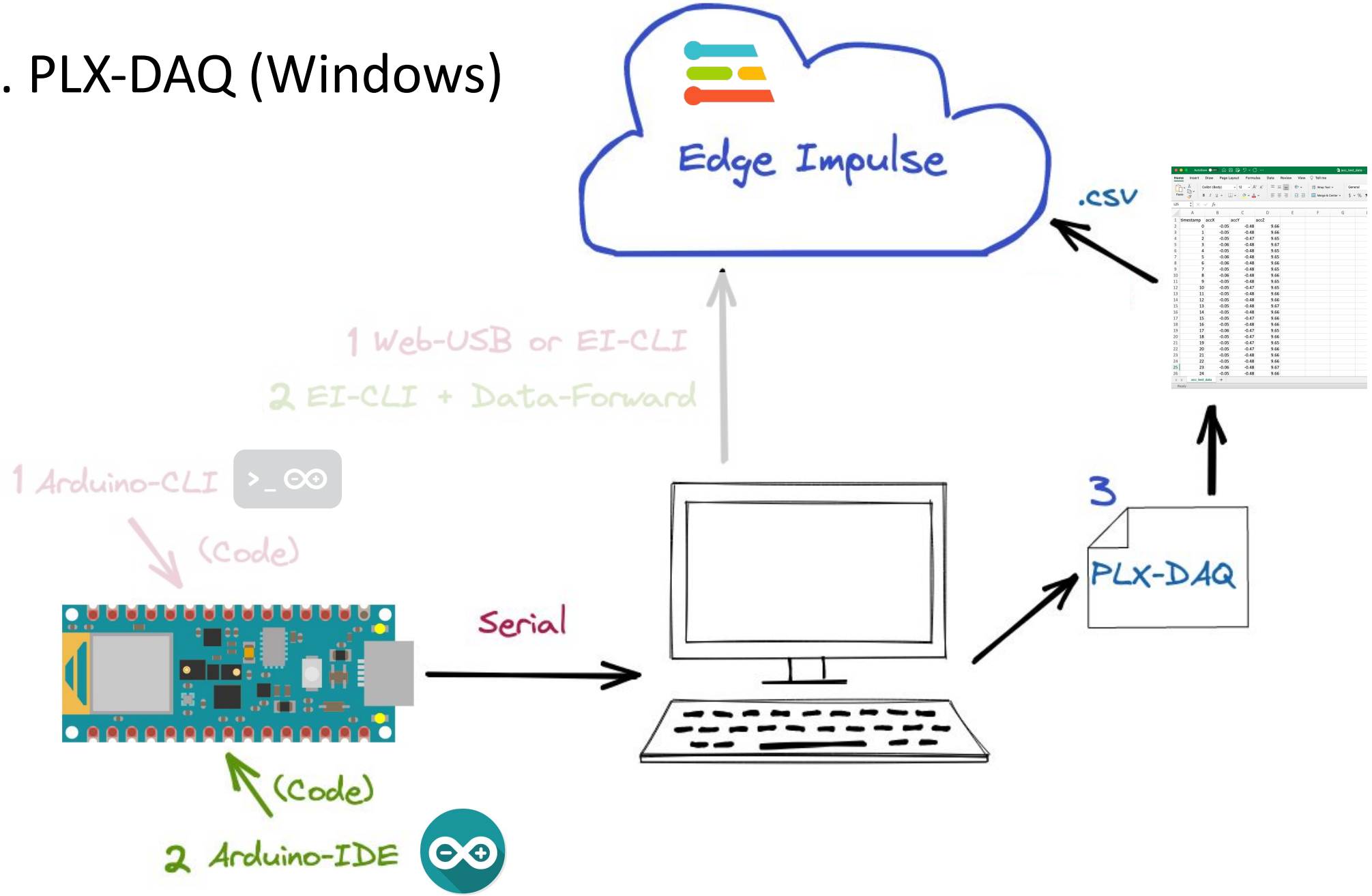
Capture_Ardu33_Sense_IMU_Acc

```ino
#include <Arduino_LSM9DS1.h>

#define CONVERT_G_TO_MS2    9.80665f
#define FREQUENCY_HZ        50
#define INTERVAL_MS         (1000 / (FREQUENCY_HZ + 1))

void setup() {
    Serial.begin(115200);
    while (!Serial);
    Serial.println("Started");

    if (!IMU.begin()) {
        Serial.println("Failed to initialize IMU!");
        while (1);
    }
}

void loop() {
    static unsigned long last_interval_ms = 0;
    float x, y, z;

    if (millis() > last_interval_ms + INTERVAL_MS) {
        last_interval_ms = millis();

        IMU.readAcceleration(x, y, z);

        Serial.print(x * CONVERT_G_TO_MS2);
        Serial.print(',');
        Serial.print(y * CONVERT_G_TO_MS2);
        Serial.print(',');
        Serial.println(z * CONVERT_G_TO_MS2);
    }
}
```

$ edge-impulse-data-forwarder --clean



3. PLX-DAQ (Windows)



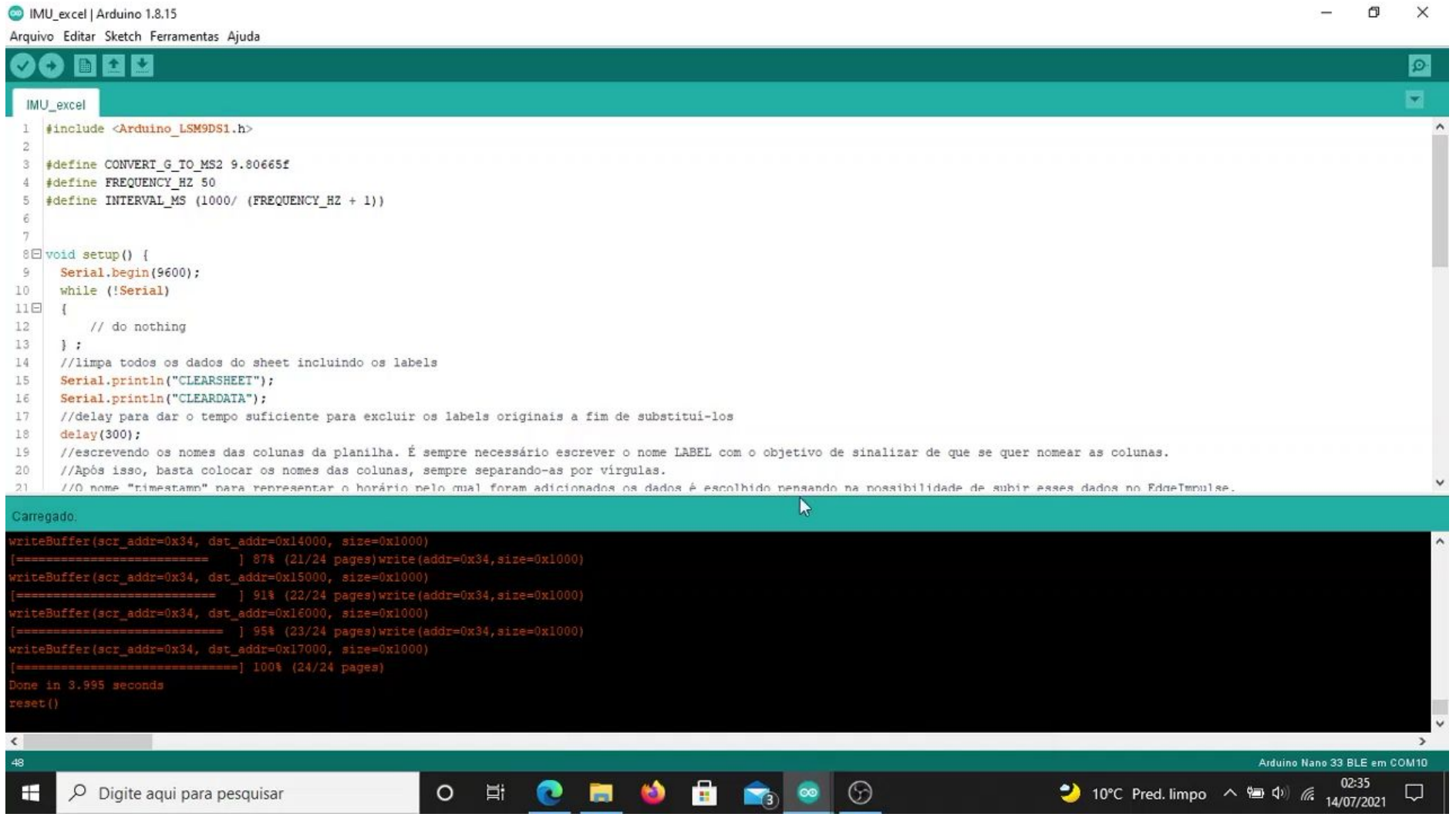
3. Data Ingestion using PLX-DAQ (Windows) => Final Format: .csv

Capture_Ardu33_Sense_IMU_Acc

```ino
#include <Arduino_LSM9DS1.h>

#define CONVERT_G_TO_MS2    9.80665f
#define FREQUENCY_HZ        50
#define INTERVAL_MS         (1000 / (FREQUENCY_HZ + 1))

void setup() {
    Serial.begin(115200);
    while (!Serial);
    Serial.println("Started");

    if (!IMU.begin()) {
        Serial.println("Failed to initialize IMU!");
        while (1);
    }
}

void loop() {
    static unsigned long last_interval_ms = 0;
    float x, y, z;

    if (millis() > last_interval_ms + INTERVAL_MS) {
        last_interval_ms = millis();

        IMU.readAcceleration(x, y, z);

        Serial.print(x * CONVERT_G_TO_MS2);
        Serial.print(',');
        Serial.print(y * CONVERT_G_TO_MS2);
        Serial.print(',');
        Serial.println(z * CONVERT_G_TO_MS2);
    }
}
```

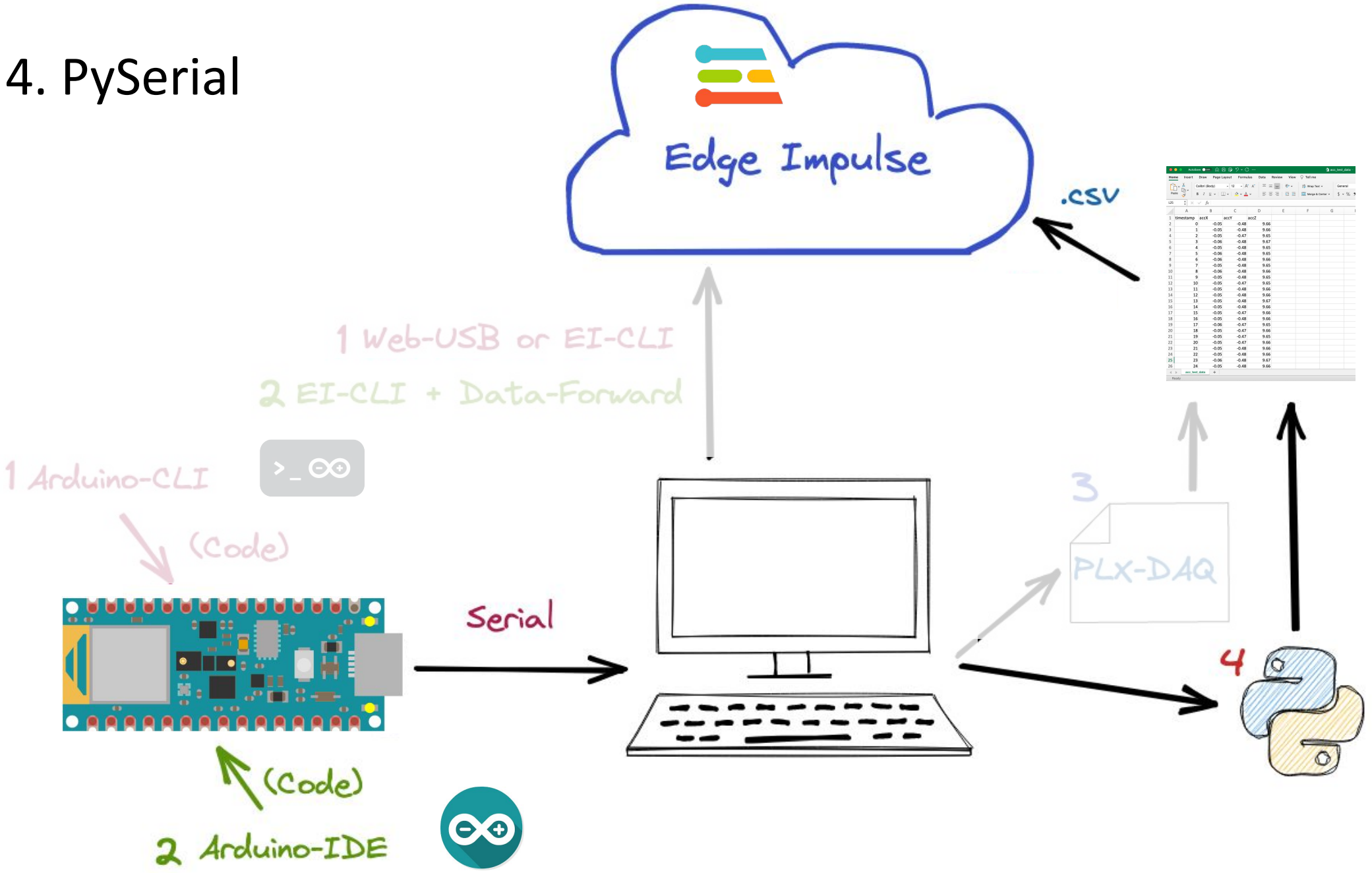
	A	B	C	D
1	timestamp	accX	accY	accZ
2	0	-0.05	-0.48	9.66
3	1	-0.05	-0.48	9.66
4	2	-0.05	-0.47	9.65
5	3	-0.06	-0.48	9.67
6	4	-0.05	-0.48	9.65
7	5	-0.06	-0.48	9.65
8	6	-0.06	-0.48	9.66
9	7	-0.05	-0.48	9.65
10	8	-0.06	-0.48	9.66
11	9	-0.05	-0.48	9.65
12	10	-0.05	-0.47	9.65
13	11	-0.05	-0.48	9.66
14	12	-0.05	-0.48	9.66
15	13	-0.05	-0.48	9.67
16	14	-0.05	-0.48	9.66
17	15	-0.05	-0.47	9.66
18	16	-0.05	-0.48	9.66
19	17	-0.06	-0.47	9.65
20	18	-0.05	-0.47	9.66
21	19	-0.05	-0.47	9.65
22	20	-0.05	-0.47	9.66
23	21	-0.05	-0.48	9.66
24	22	-0.05	-0.48	9.66
25	23	-0.06	-0.48	9.67
26	24	-0.05	-0.48	9.66

3. Data Ingestion using PLX-DAQ (Windows) => Final Format: .csv



https://www.youtube.com/watch?v=BwbmNle2CZo

4. PySerial

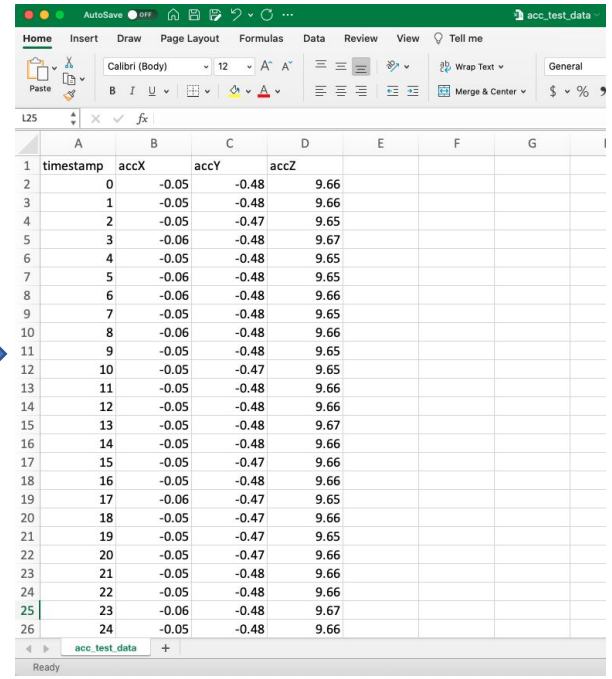

Edge Impulse
.csv
1 Web-USB or EI-CLI
2 EI-CLI + Data-Forward
1 Arduino-CLI
(Code)
Serial
3
PLX-DAQ
4
(Code)
2 Arduino-IDE

4. Data Ingestion using Python (PySerial) => Final Format: .csv

Capture_Ardu33_Sense_IMU_Acc

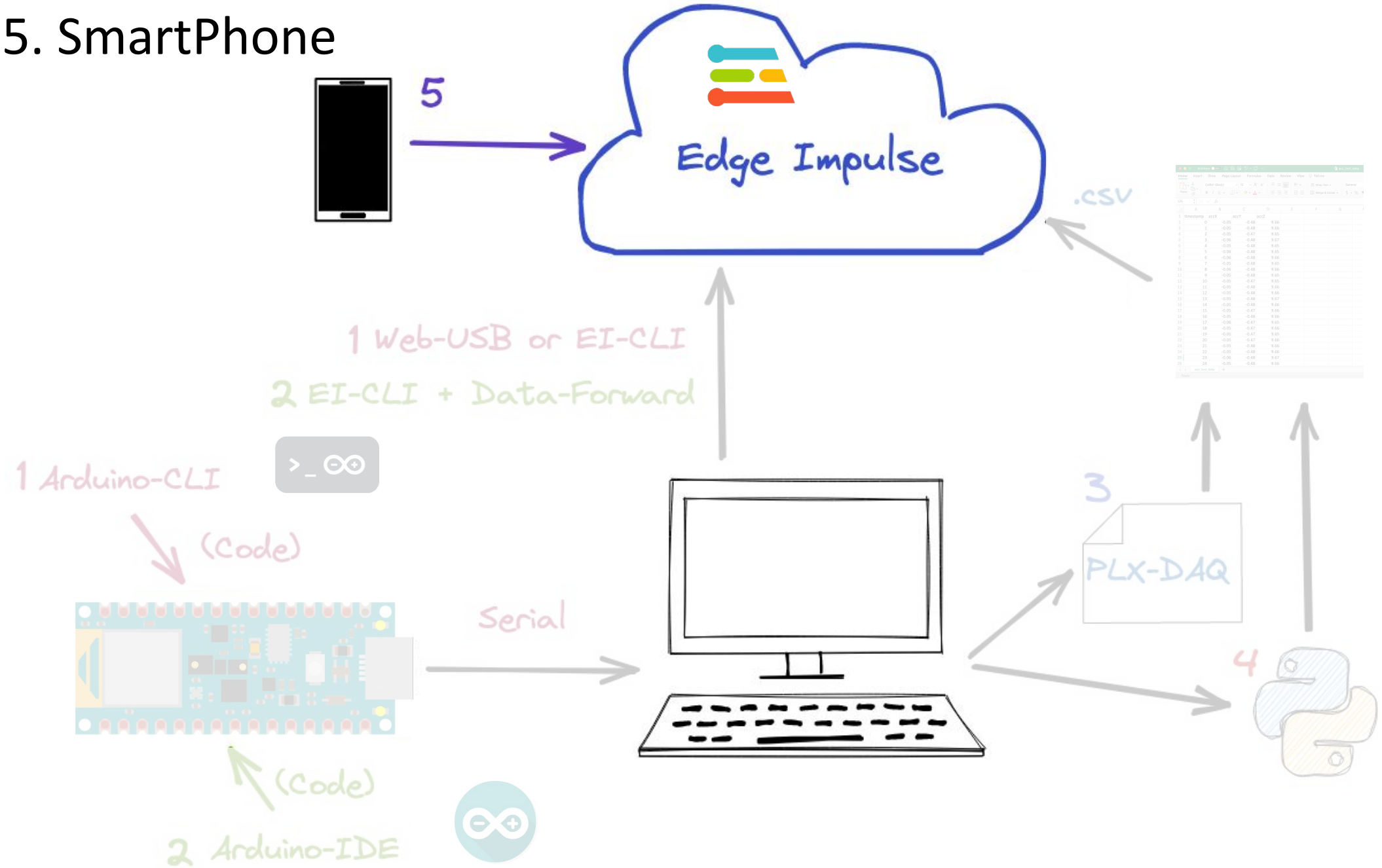
```ino
#include <Arduino_LSM9DS1.h>

#define CONVERT_G_TO_MS2    9.80665f
#define FREQUENCY_HZ        50
#define INTERVAL_MS         (1000 / (FREQUENCY_HZ + 1))

void setup() {
    Serial.begin(115200);
    while (!Serial);
    Serial.println("Started");

    if (!IMU.begin()) {
        Serial.println("Failed to initialize IMU!");
        while (1);
    }
}

void loop() {
    static unsigned long last_interval_ms = 0;
    float x, y, z;

    if (millis() > last_interval_ms + INTERVAL_MS) {
        last_interval_ms = millis();

        IMU.readAcceleration(x, y, z);

        Serial.print(x * CONVERT_G_TO_MS2);
        Serial.print(',');
        Serial.print(y * CONVERT_G_TO_MS2);
        Serial.print(',');
        Serial.println(z * CONVERT_G_TO_MS2);
    }
}
```

```python
# Sensor data Logger (CSV)
# by Marcelo Rovai @ 13July21

import serial

arduino_port = '/dev/tty.usbmodem144301'
baud_rate = 115200
ser = serial.Serial(port=arduino_port, baudrate=baud_rate)

fileName = "acc_test_data.csv"  # name of the CSV file generated

first_line = 'timestamp,accX,accY,accZ'
file = open(fileName, "w")
file.write(first_line + "\n")  # write data with a newline
file.close()

Freq_hz = 50
num_seconds = 10  # number of seconds collecting data
samples = num_seconds * Freq_hz # number of samples to collect

sample = 0
while sample <= samples:
    getData = str(ser.readline())
    data = getData[2:][:-5]
    print(data)

    file = open(fileName, "a")
    file.write(str(sample) + "," + data + "\n")
    sample = sample+1
print("Data collection complete!")
file.close()
```

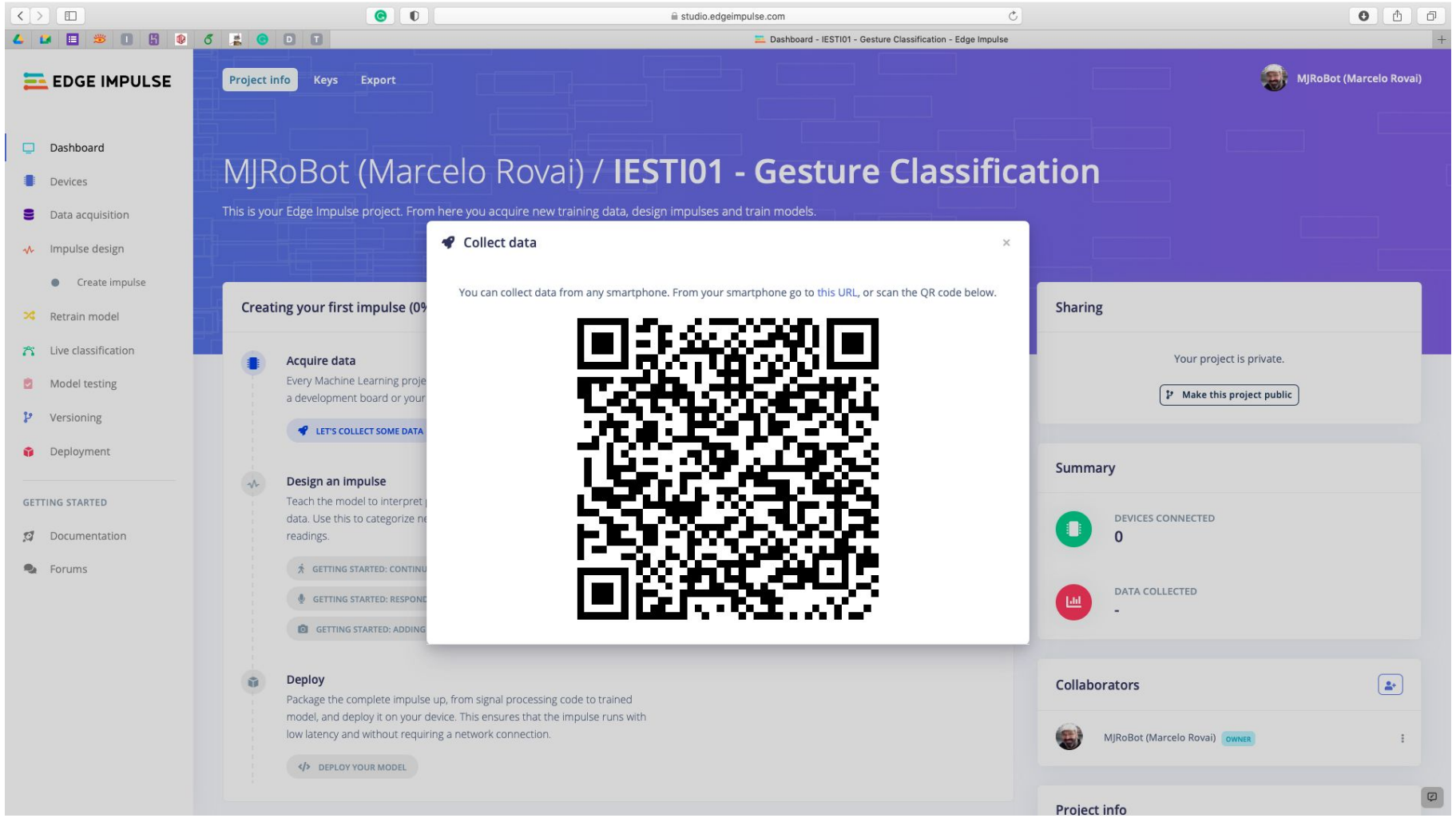


timestamp	accX	accY	accZ
0	-0.05	-0.48	9.66
1	-0.05	-0.48	9.66
2	-0.05	-0.47	9.65
3	-0.06	-0.48	9.67
4	-0.05	-0.48	9.65
5	-0.06	-0.48	9.65
6	-0.06	-0.48	9.66
7	-0.05	-0.48	9.65
8	-0.06	-0.48	9.66
9	-0.05	-0.48	9.65
10	-0.05	-0.47	9.65
11	-0.05	-0.48	9.66
12	-0.05	-0.48	9.66
13	-0.05	-0.48	9.67
14	-0.05	-0.48	9.66
15	-0.05	-0.47	9.66
16	-0.05	-0.48	9.66
17	-0.06	-0.47	9.65
18	-0.05	-0.47	9.66
19	-0.05	-0.47	9.65
20	-0.05	-0.47	9.66
21	-0.05	-0.48	9.66
22	-0.05	-0.48	9.66
23	-0.06	-0.48	9.67
24	-0.05	-0.48	9.66

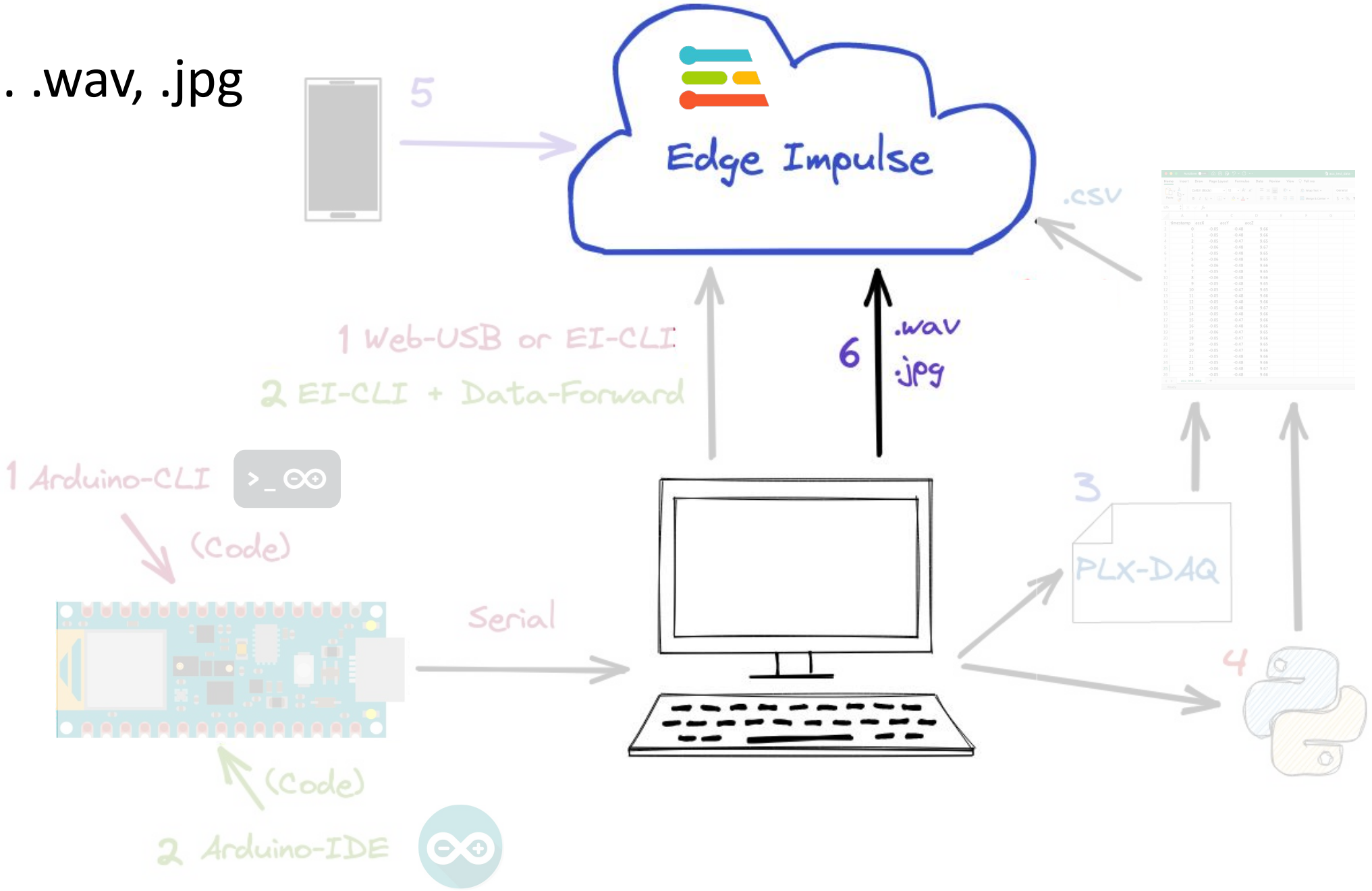
5. SmartPhone



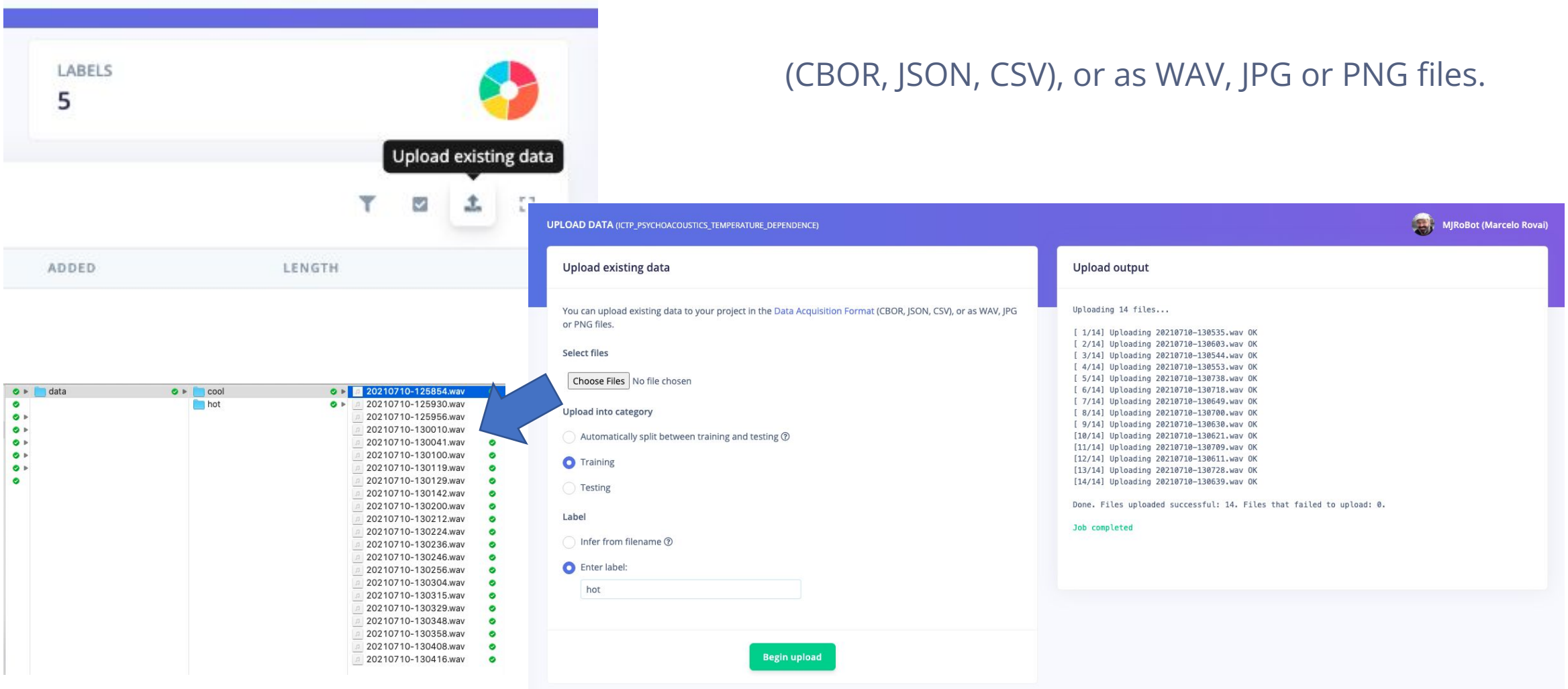
5. Data Ingestion using Smart Phone



6. .wav, .jpg

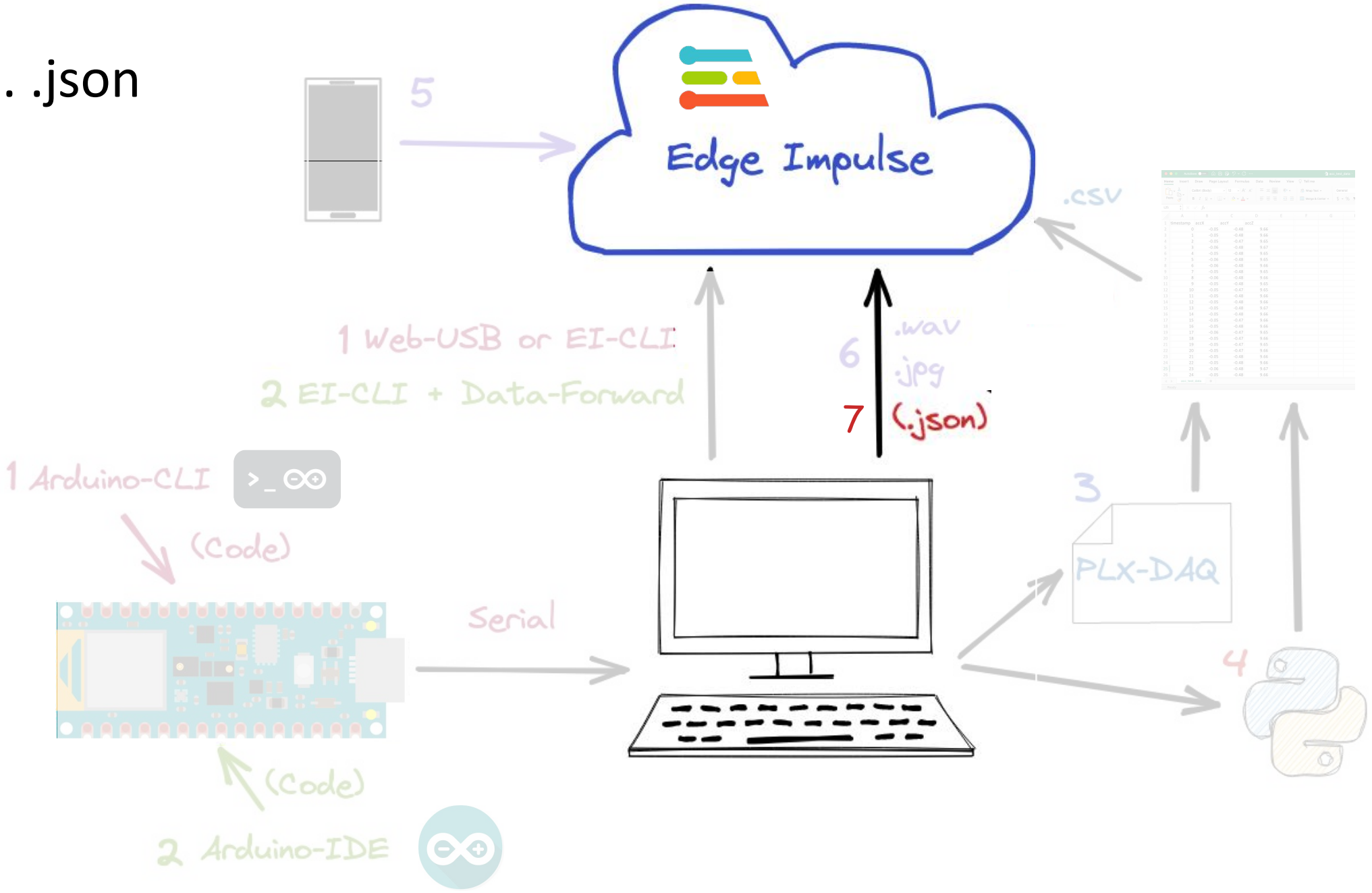

5
Edge Impulse
.csv
1 Web-USB or EI-CLI
2 EI-CLI + Data-Forward
6
.wav
.jpg
1 Arduino-CLI
(Code)
3
PLX-DAQ
Serial
4
(Code)
2 Arduino-IDE

6. Data Ingestion using Upload existing Data



(CBOR, JSON, CSV), or as WAV, JPG or PNG files.

7. .json


5
Edge Impulse
.csv
1 Web-USB or EI-CLI
2 EI-CLI + Data-Forward
6
.wav
.jpg
7
(.json)
1 Arduino-CLI
(Code)
3
PLX-DAQ
Serial
4
(Code)
2 Arduino-IDE

7. Raw Uploader (.json files)

Image Classification: Raw Uploader

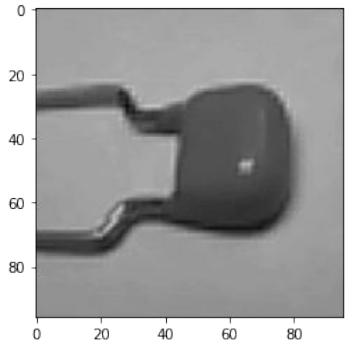
Open in Colab

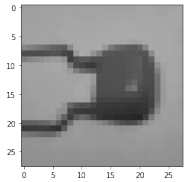
Run this notebook to convert images to a single row of raw, normalized values (between 0 and 1) and upload them to Edge Impulse as raw samples. Note that pixel values will be normalized to be between 0 and 1.

Create a folder named "dataset" in the /content directory and upload your images there. The images should be divided into their respective classes, where each class has its own folder with the name of the class. For example:

```
/content
    |- dataset
        |- background
        |- capacitor
        |- diode
        |- led
        |- resistor
```

Author: EdgeImpulse, Inc.
Date: June 6, 2021
License: Apache-2.0





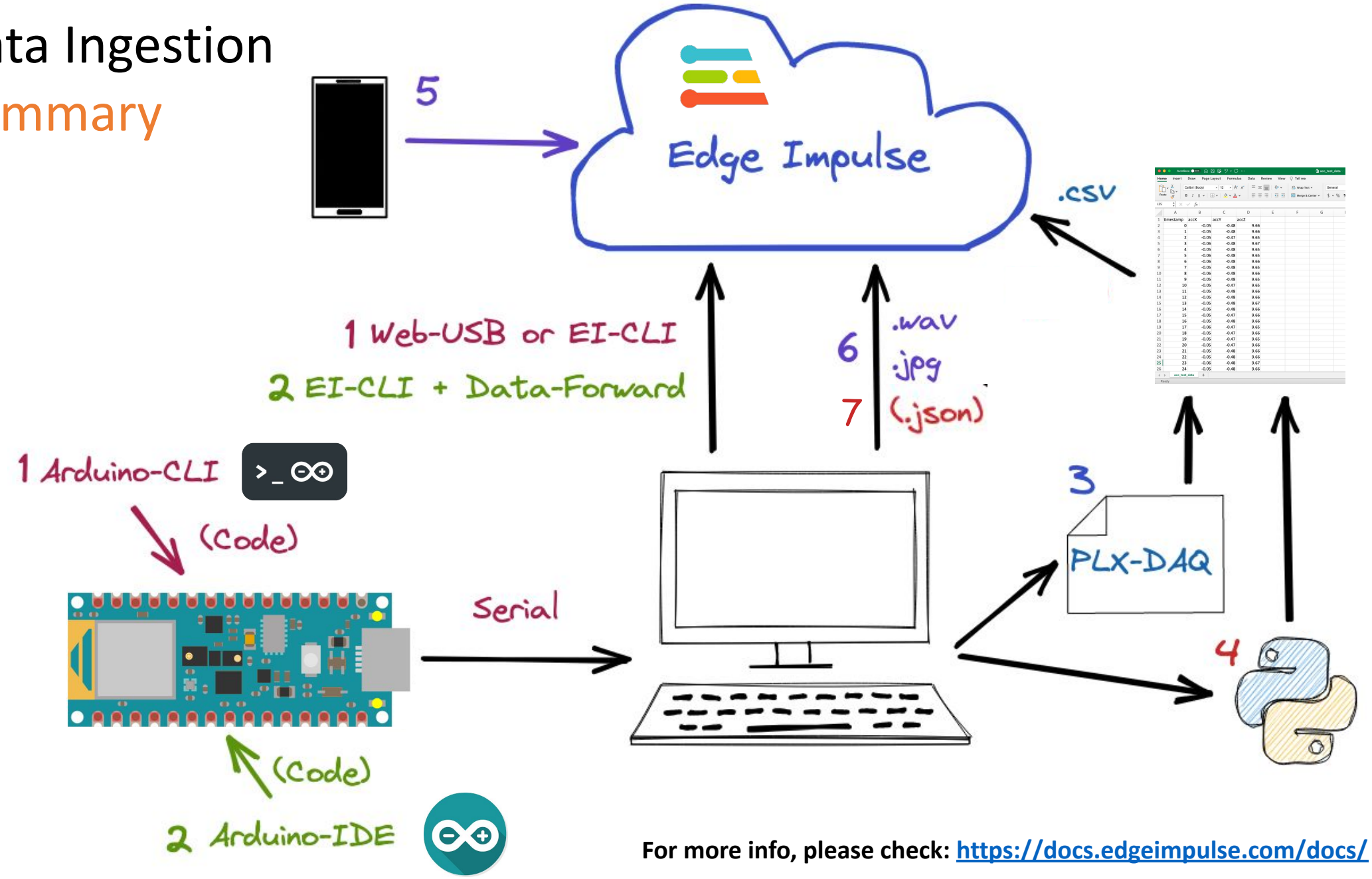
.json

EDGE IMPULSE
(Training as DNN)

Also see: [TinyML Made Easy: Exploring Regression - White Wine Quality]

Data Ingestion

Summary



For more info, please check: https://docs.edgeimpulse.com/docs/

Temperature Dependence

Psychoacoustics

Simple TinyML Proof-of-concept

https://www.hackster.io/mjrobot/listening-temperature-with-tinyml-7e1325

Audio Engineering Society

Convention e-Brief 473

Presented at the 145th Convention
2018 October 17 – 20, New York, NY, USA

Why can you hear a difference between pouring hot and cold water? An investigation of temperature dependence in psychoacoustics.

He Peng[1] and Joshua D. Reiss[2]

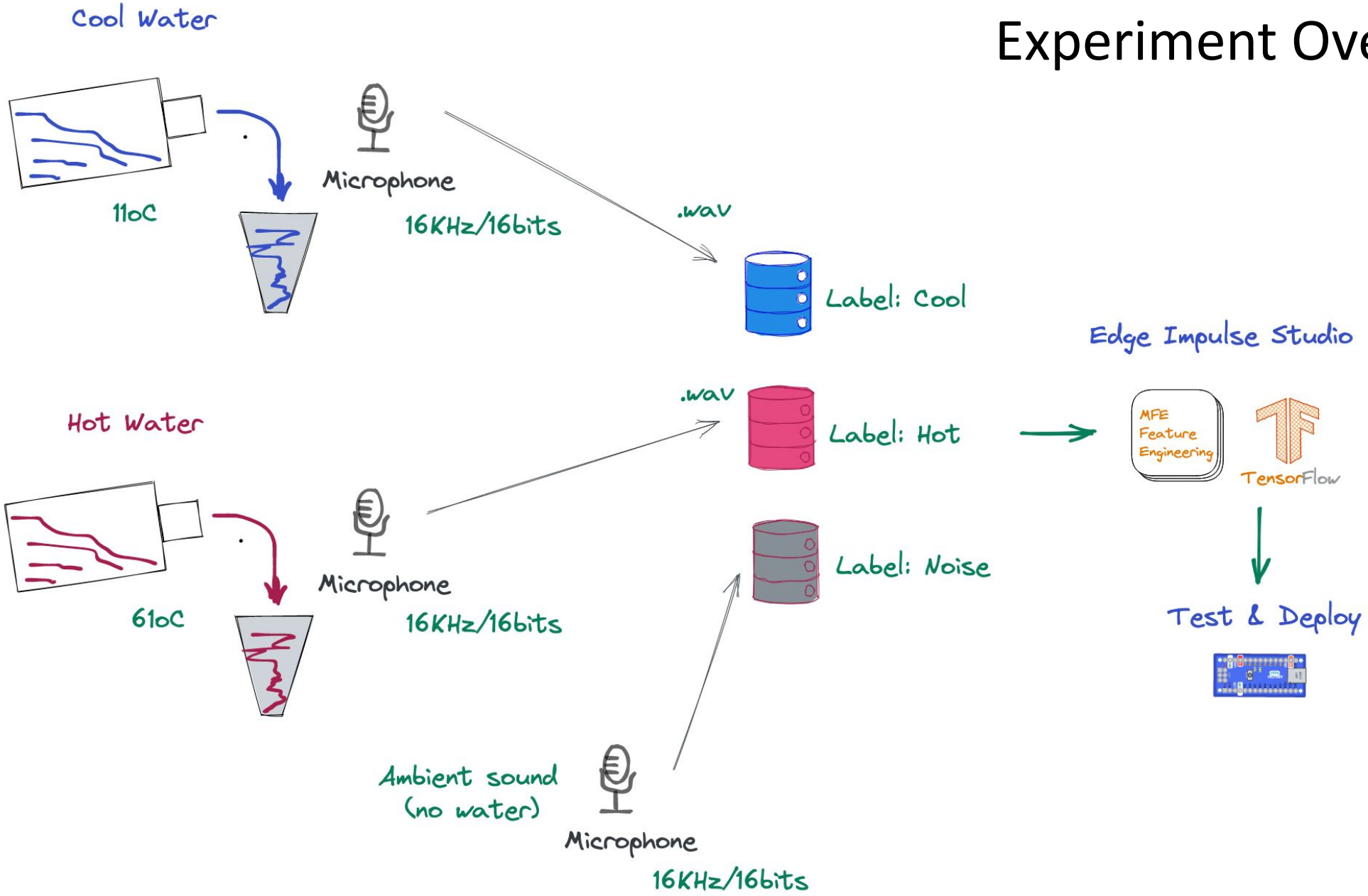
[1] *Tianjin University*
[2] *Queen Mary University of London*

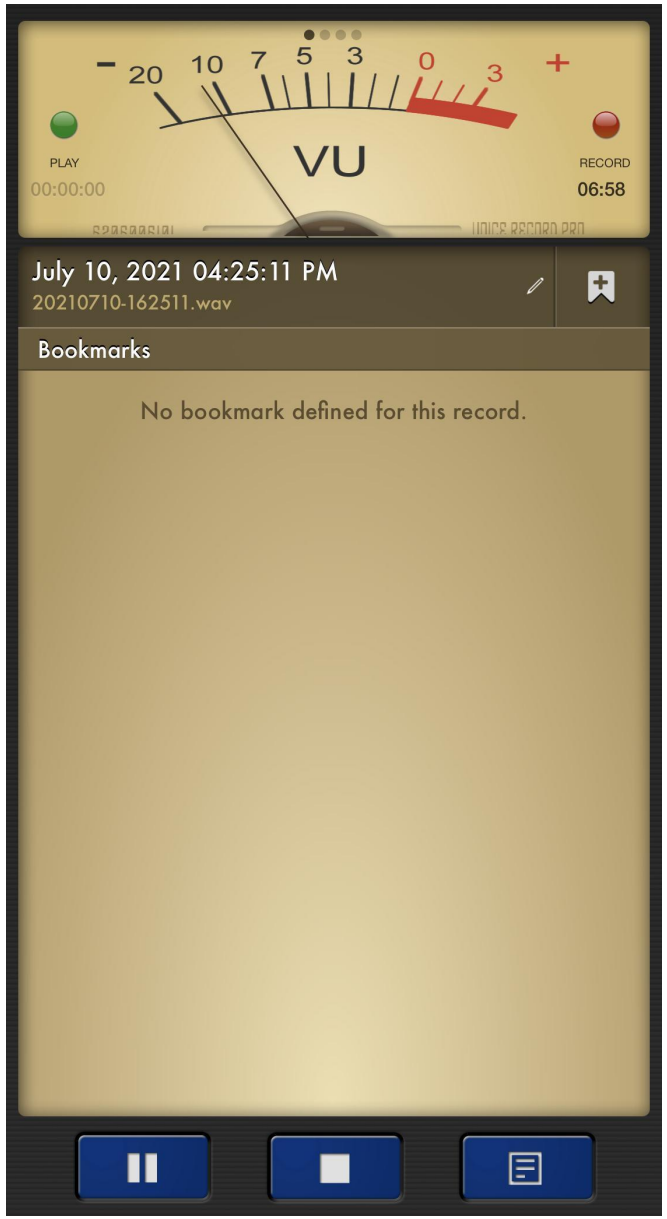
Correspondence should be addressed to He Peng, Joshua Reiss (hepeng2018@hotmail.com, joshua.reiss@qmul.ac.uk)

http://www.eecs.qmul.ac.uk/~josh/documents/2018/19737.pdf

https://www.youtube.com/watch?v=Ri_4dDvcZeM (min: 0.17 => min 2:37)

Experiment Overview





Voice Recorder

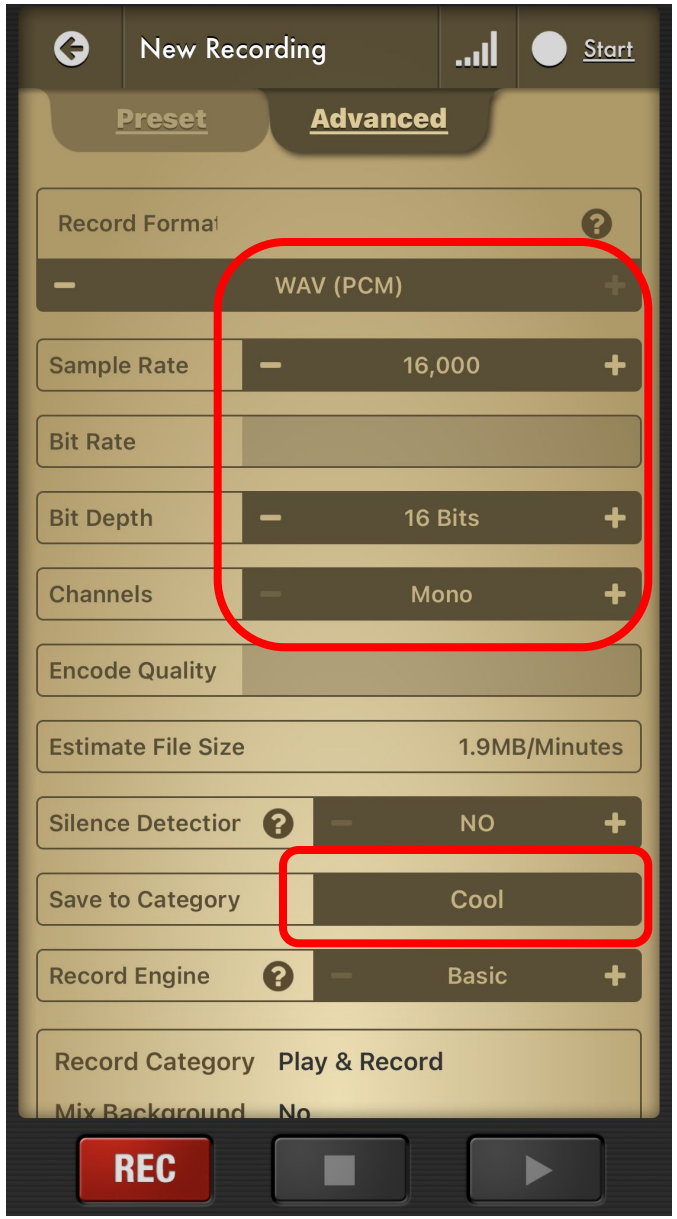
Sample Sound:

- 16KHz
- PCM – 16bits
- Mono

Classes:

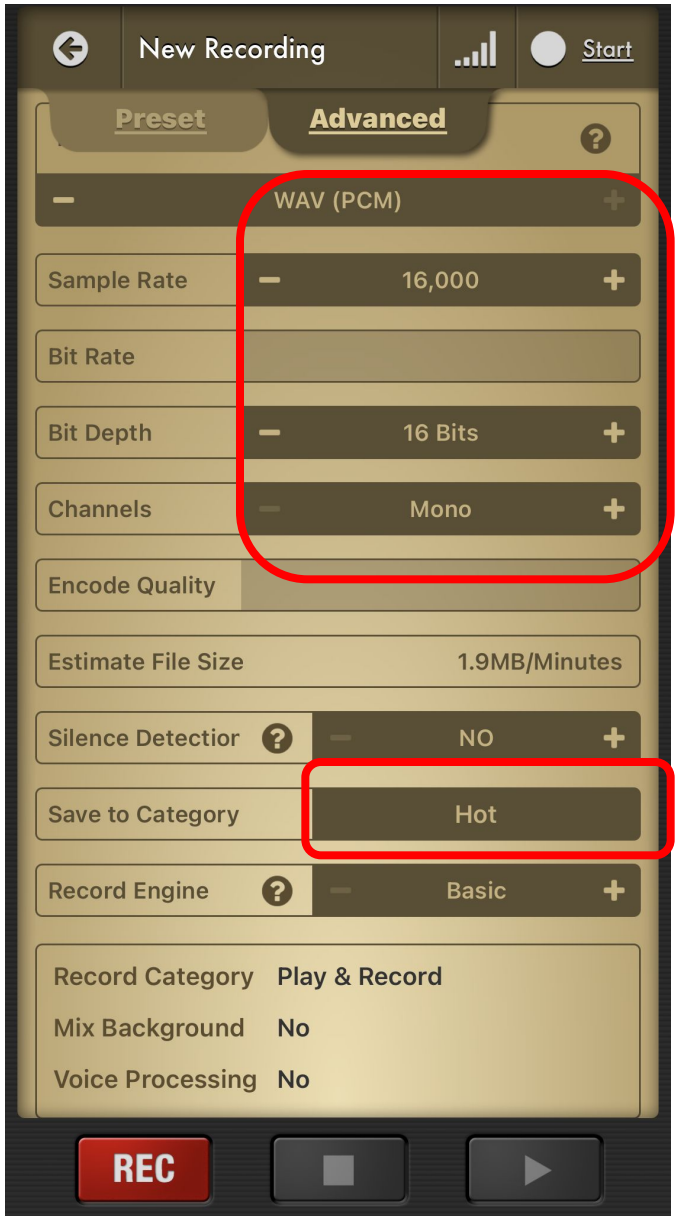
- Hot
- Cool
- Noise

Ambient Temperature: 19°C



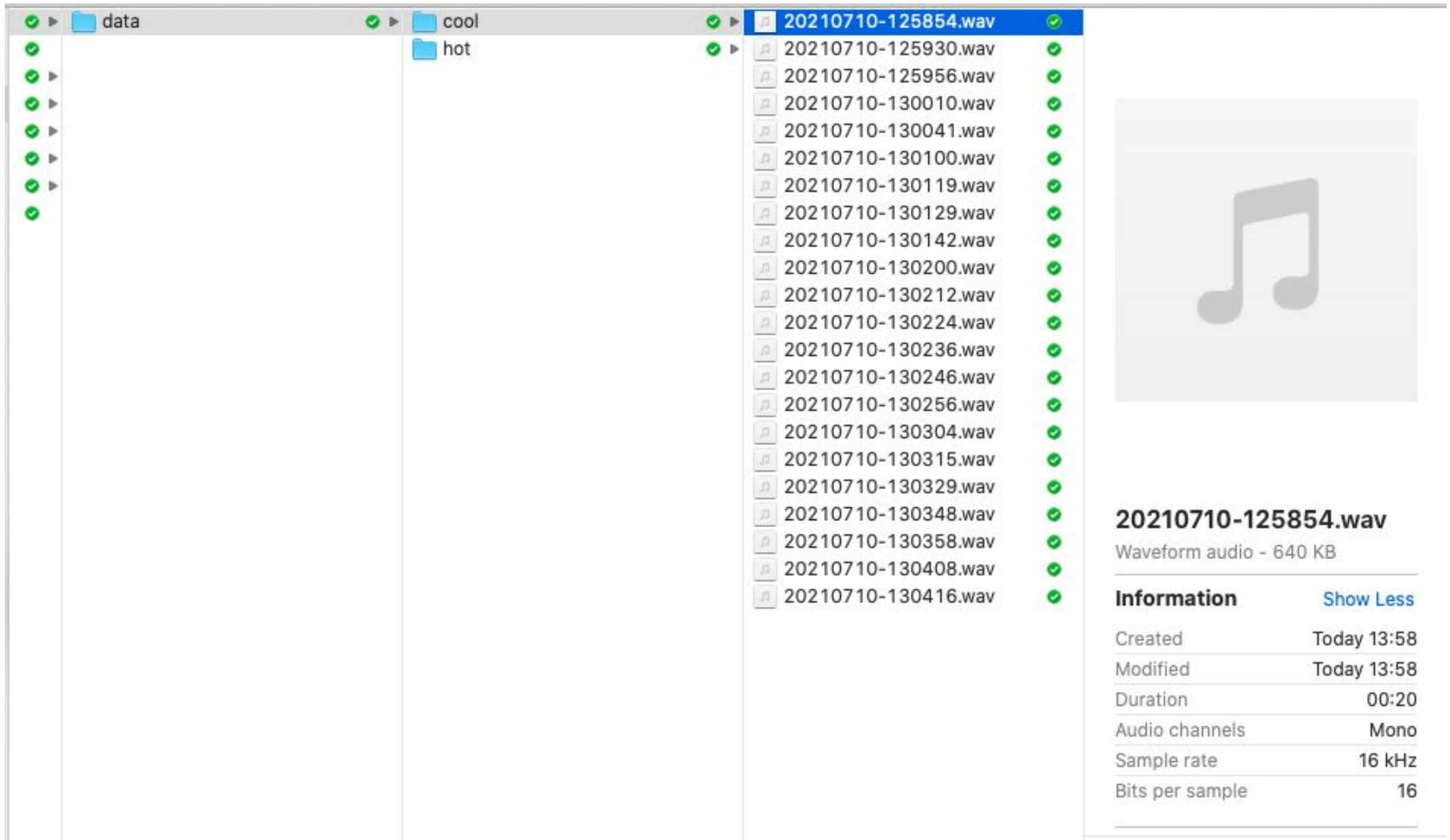
Class: Cool

Cool Water Temperature: 11°C

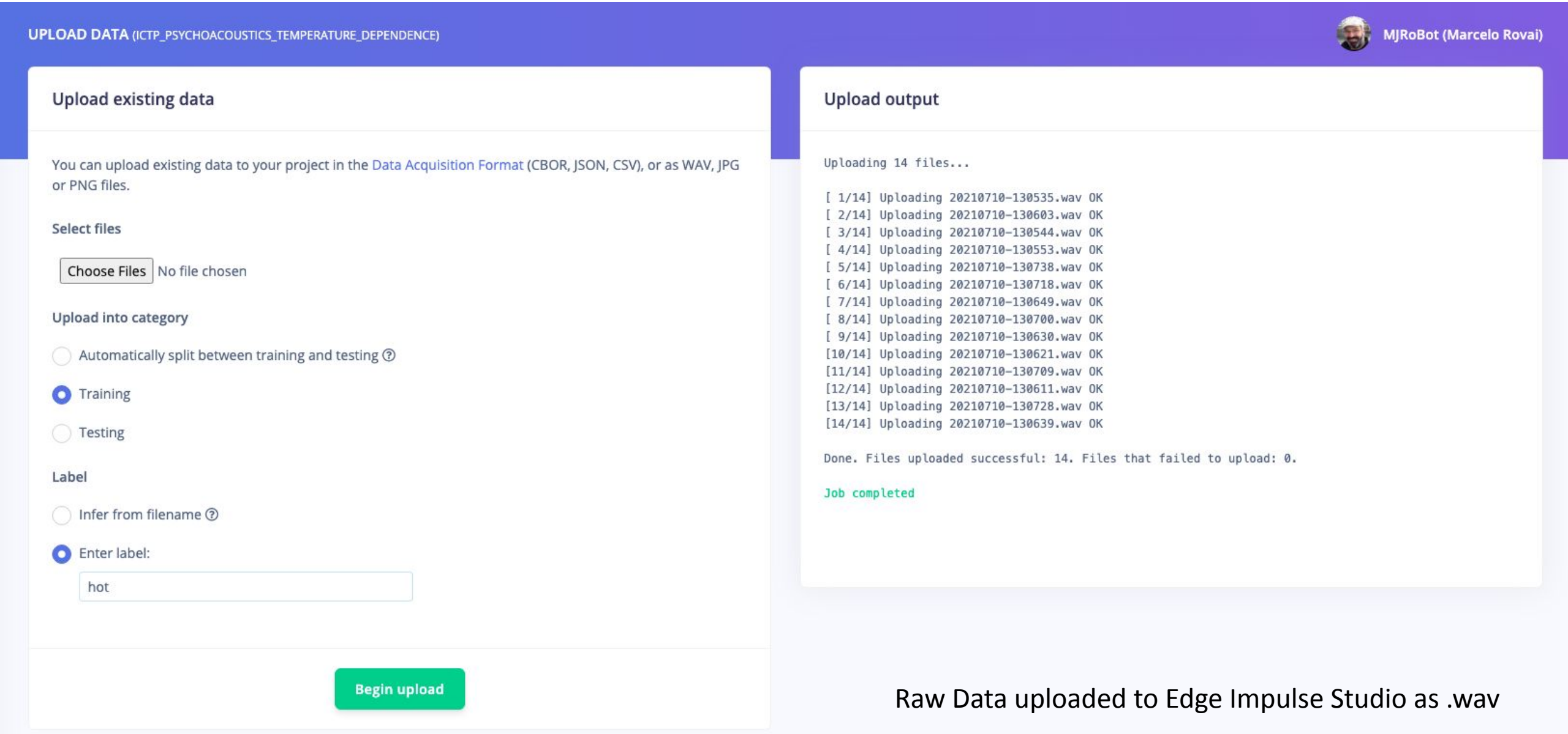


Class: Hot

Hot Water Temperature: 61°C



Data captured using app Voice Recorder and uploaded to Computer



Raw Data uploaded to Edge Impulse Studio as .wav

Training data | Test data

Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

DATA COLLECTED
2m 49s

LABELS
2

Collected data

SAMPLE NAME	LABEL	ADDED	LENGTH
20210710-130621.wav.2a5e0...	hot	Today, 14:02:55	4s
20210710-130630.wav.2a5e0...	hot	Today, 14:02:54	3s
20210710-130700.wav.2a5e0...	hot	Today, 14:02:52	4s
20210710-130649.wav.2a5e0...	hot	Today, 14:02:52	5s
20210710-130718.wav.2a5e0...	hot	Today, 14:02:52	5s
20210710-130738.wav.2a5e0...	hot	Today, 14:02:51	5s
20210710-130553.wav.2a5e0...	hot	Today, 14:02:51	4s
20210710-130544.wav.2a5e0...	hot	Today, 14:02:51	4s
20210710-130603.wav.2a5e0...	hot	Today, 14:02:51	3s
20210710-130535.wav.2a5e0...	hot	Today, 14:02:48	5s
20210710-130416.wav.2a5dv...	cool	Today, 14:02:12	4s
20210710-130408.wav.2a5dv...	cool	Today, 14:02:11	4s

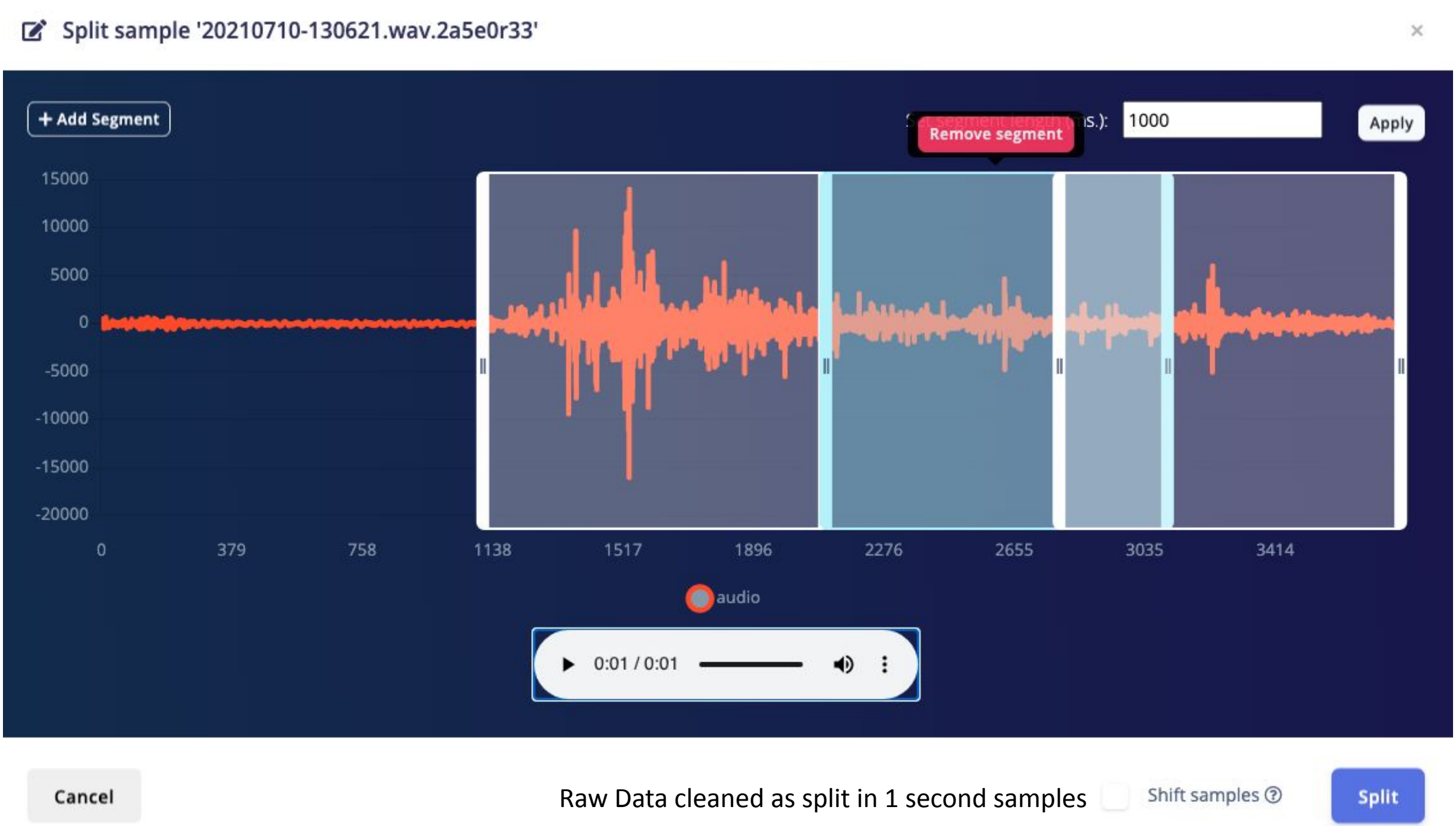
Record new data

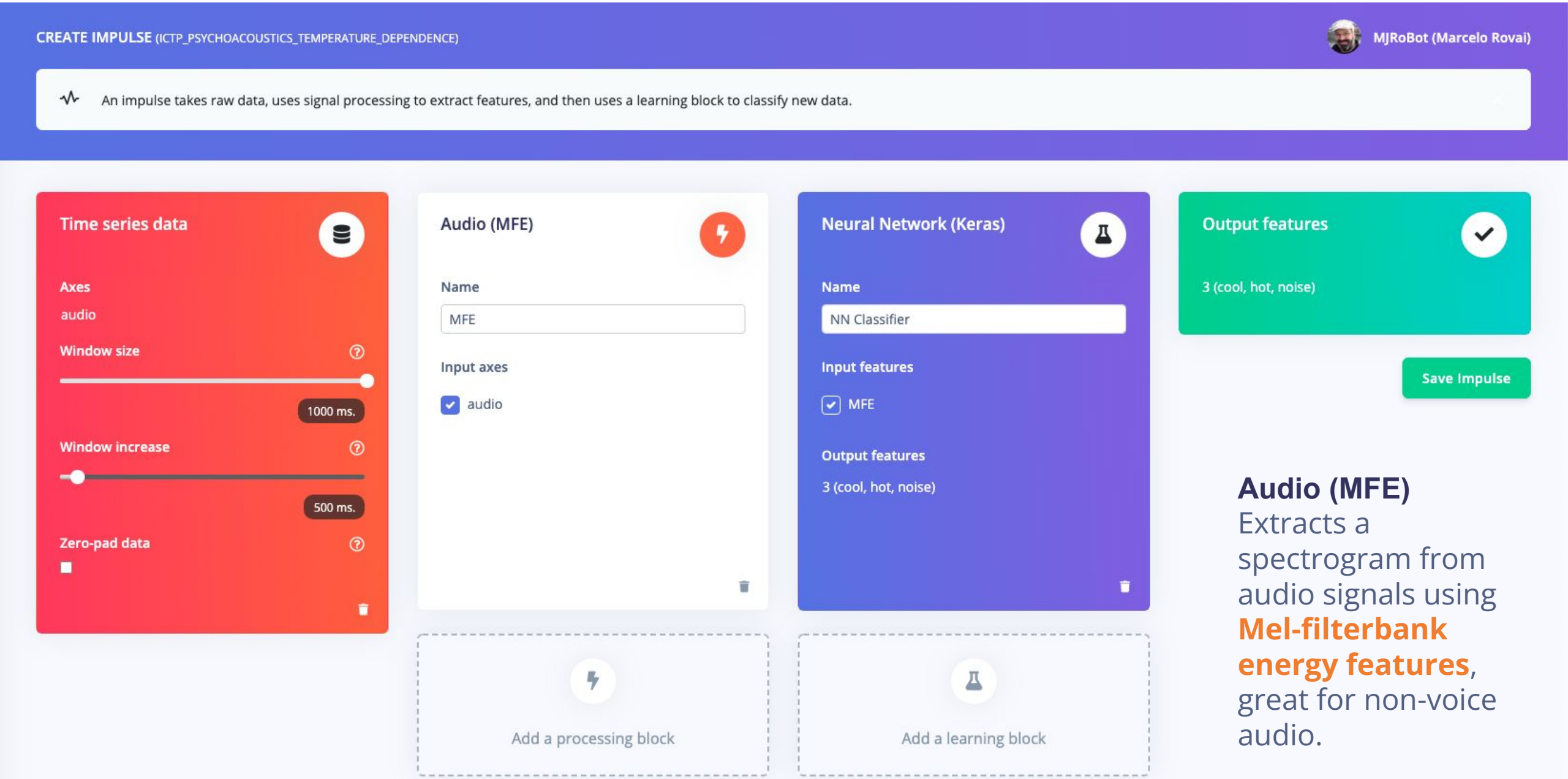
Connect using WebUSB

No devices connected to the remote management API.

RAW DATA
20210710-130621.wav.2a5e0r33

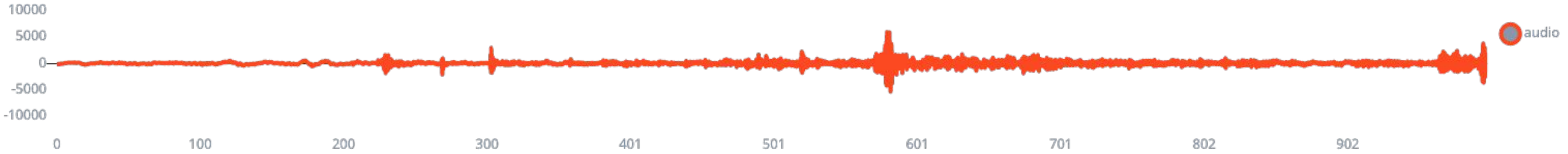
Raw Data cleaned as split in 1 second samples

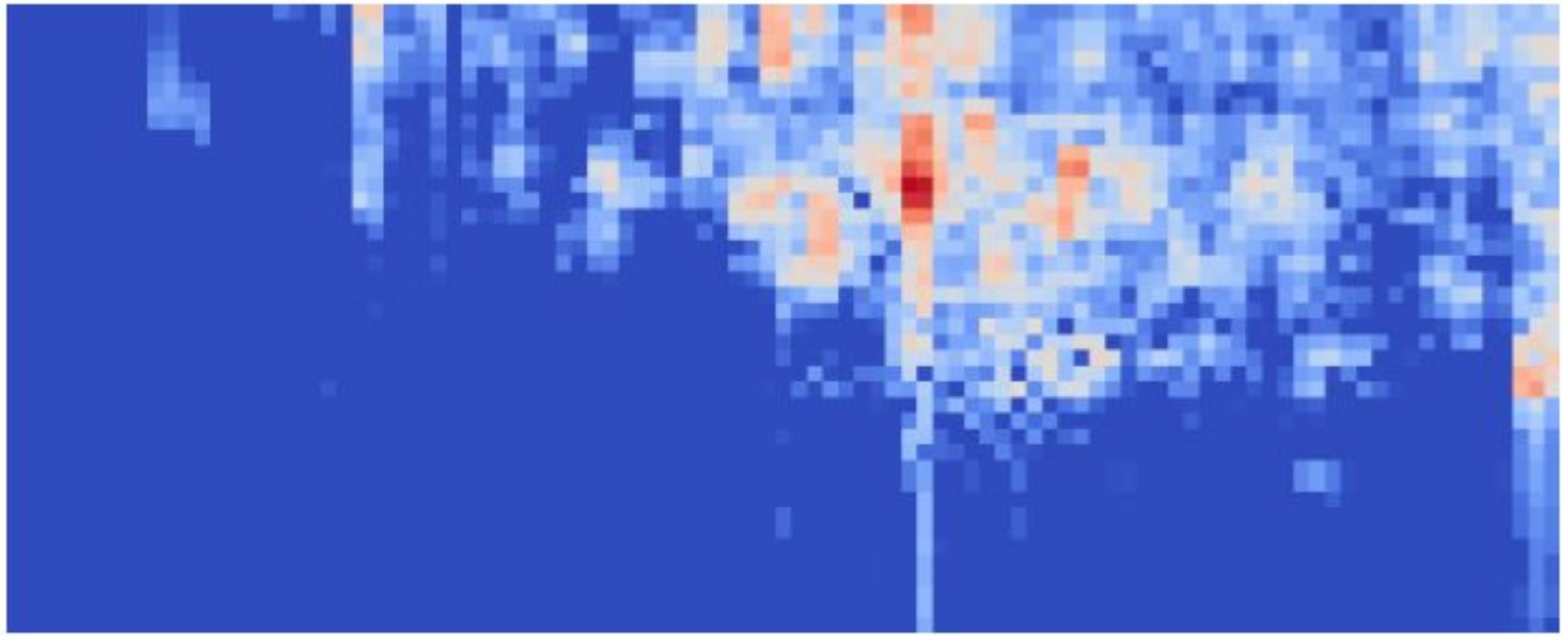

Split sample '20210710-130621.wav.2a5e0r33'
+ Add Segment
Remove segment
1000
Apply
15000
10000
5000
0
-5000
-10000
-15000
-20000
0
379
758
1138
1517
1896
2276
2655
3035
3414
audio
0:01 / 0:01
Cancel
Raw Data cleaned as split in 1 second samples
Shift samples
Split



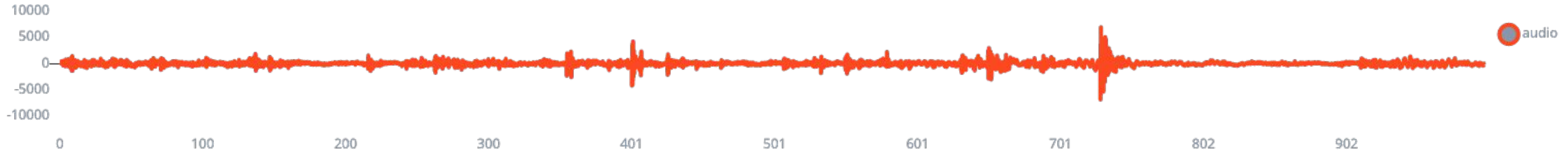
Audio (MFE)
Extracts a spectrogram from audio signals using **Mel-filterbank energy features**, great for non-voice audio.

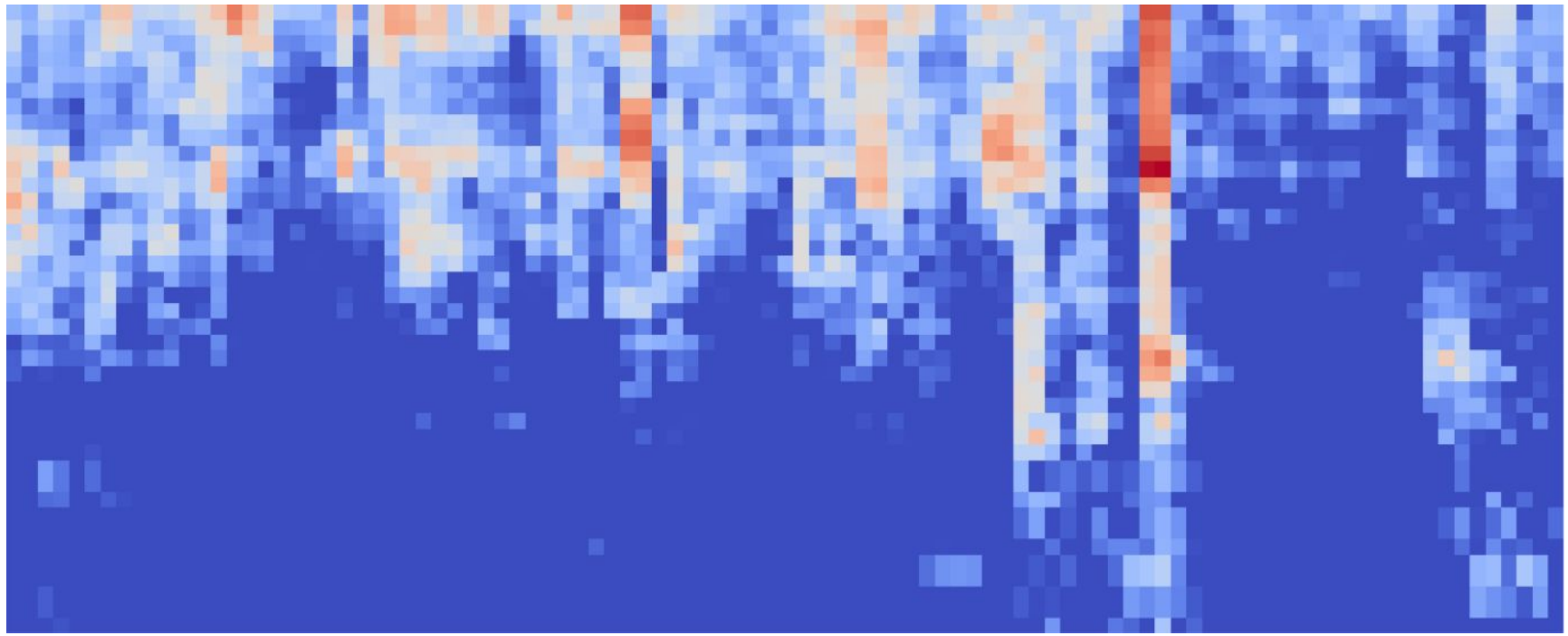
Cool Water 1 second sample



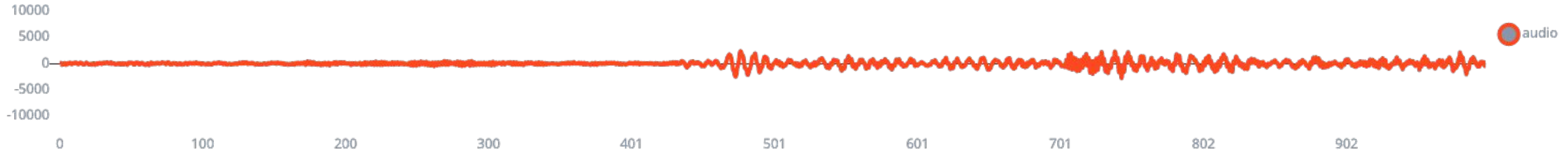


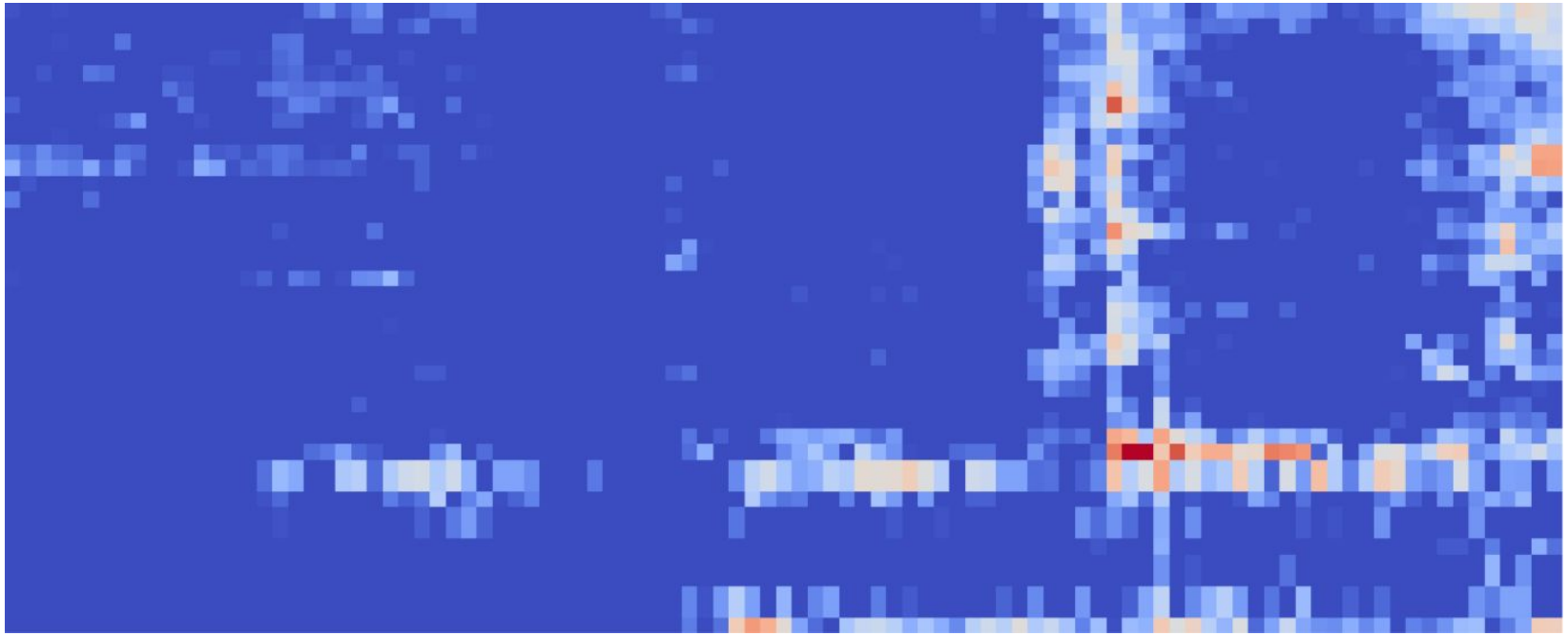
Hot Water 1 second sample





Noise 1 second sample





Training set

Data in training set	2m 35s
Classes	3 (cool, hot, noise)
Window length	1000 ms.
Window increase	500 ms.
Training windows	155

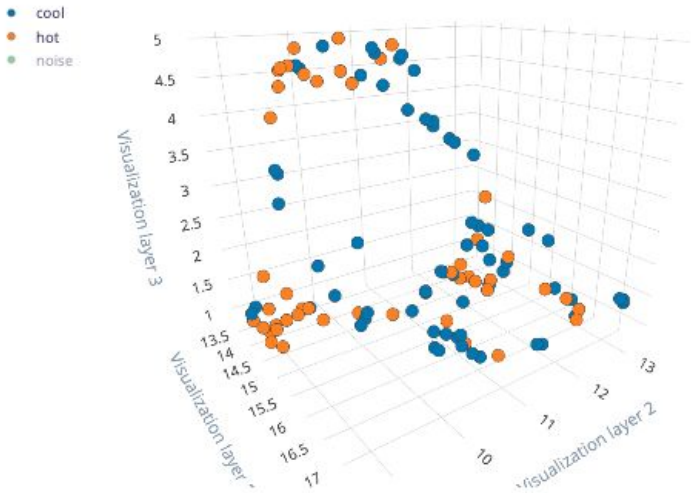
Generate features

Feature explorer (155 samples)

X Axis: Visualization layer 1

Y Axis: Visualization layer 2

Z Axis: Visualization layer 3

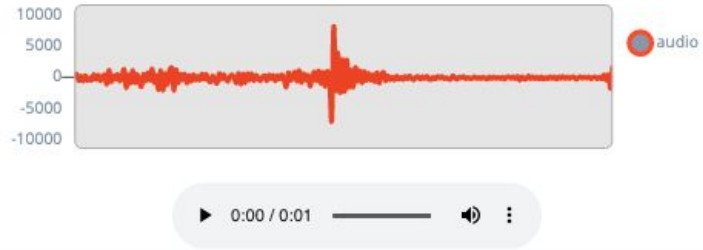


20210710-125930.wav.2a5dv05j.s2

Label: cool

View sample

View features



0:00 / 0:01

On-device performance

PROCESSING TIME
250 ms.

PEAK RAM USAGE
25 KB

NN CLASSIFIER (ICTP_PSYCHOACOUSTICS_TEMPERATURE_DEPENDENCE)

#1 Click to set a description for this version

MJRoBot (Marcelo Rovai)

Neural Network settings

Training settings

Number of training cycles: 100

Learning rate: 0.005

Minimum confidence rating: 0.60

Audio training options

Data augmentation

Neural network architecture

Architecture presets: 1D Convolutional (Default) 2D Convolutional

Input layer (3,960 features)

Reshape layer (40 columns)

1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

Flatten layer

Add an extra layer

Output layer (3 features)

Training output

Model

Model version: Quantized (int8)

Last training performance (validation set)

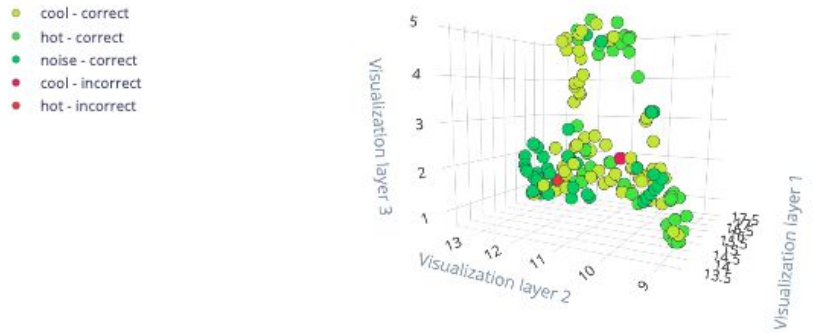
ACCURACY 93.5%

LOSS 0.13

Confusion matrix (validation set)

	COOL	HOT	NOISE
COOL	92.9%	7.1%	0%
HOT	16.7%	83.3%	0%
NOISE	0%	0%	100%
F1 SCORE	0.93	0.83	1.00

Feature explorer (full training set)



On-device performance

INFERENCING TIME 17 ms.

PEAK RAM USAGE 10.9K

FLASH USAGE 31.4K

This lists all test data. You can manage this data through Data acquisition.

Test data

Classify all

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTCOME	LENGTH	ACCURACY	RESULT
20210710-130728.wa...	hot	1s	100%	1 hot
20210710-130639.wa...	hot	1s	100%	1 hot
20210710-130553.wa...	hot	1s	100%	1 hot
20210710-130535.wa...	hot	1s	100%	1 hot
20210710-130535.wa...	hot	1s	100%	1 hot
20210710-130224.wa...	cool	1s	0%	1 noise
20210710-130304.wa...	cool	1s	100%	1 cool
20210710-130236.wa...	cool	1s	100%	1 cool
20210710-130256.wa...	cool	1s	100%	1 cool
20210710-130224.wa...	cool	1s	100%	1 cool
20210710-130142.wa...	cool	1s	0%	1 noise
20210710-130100.wa...	noise	1s	100%	1 noise
20210710-130041.wa...	noise	1s	100%	1 noise
20210710-125854.wa...	noise	1s	100%	1 noise
20210710-125854.wa...	noise	1s	100%	1 noise

Model testing output

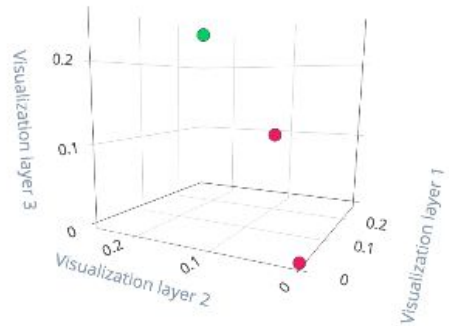
Model testing results

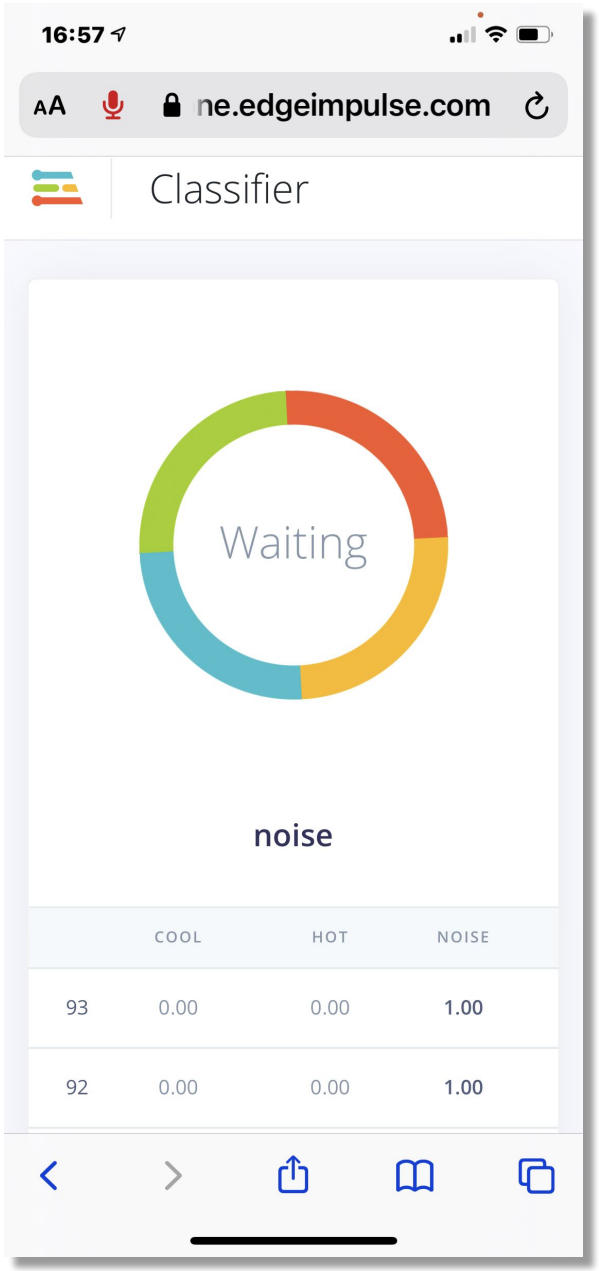
ACCURACY
86.67%

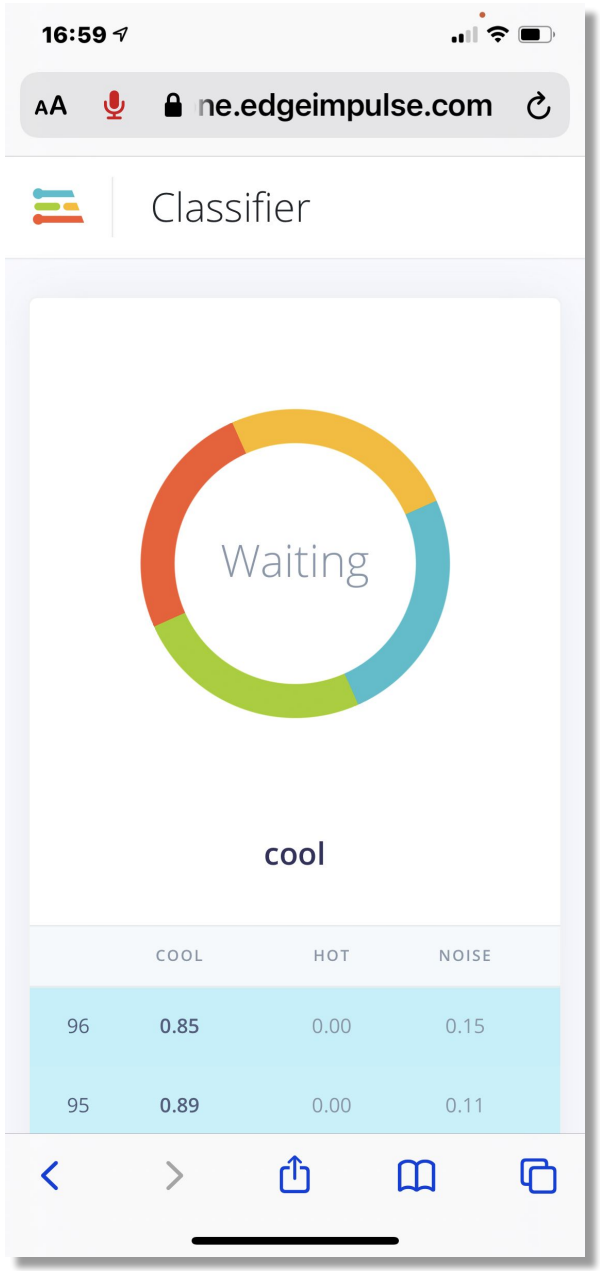
	COOL	HOT	NOISE	UNCERTAIN
COOL	66.7%	0%	33.3%	0%
HOT	0%	100%	0%	0%
NOISE	0%	0%	100%	0%

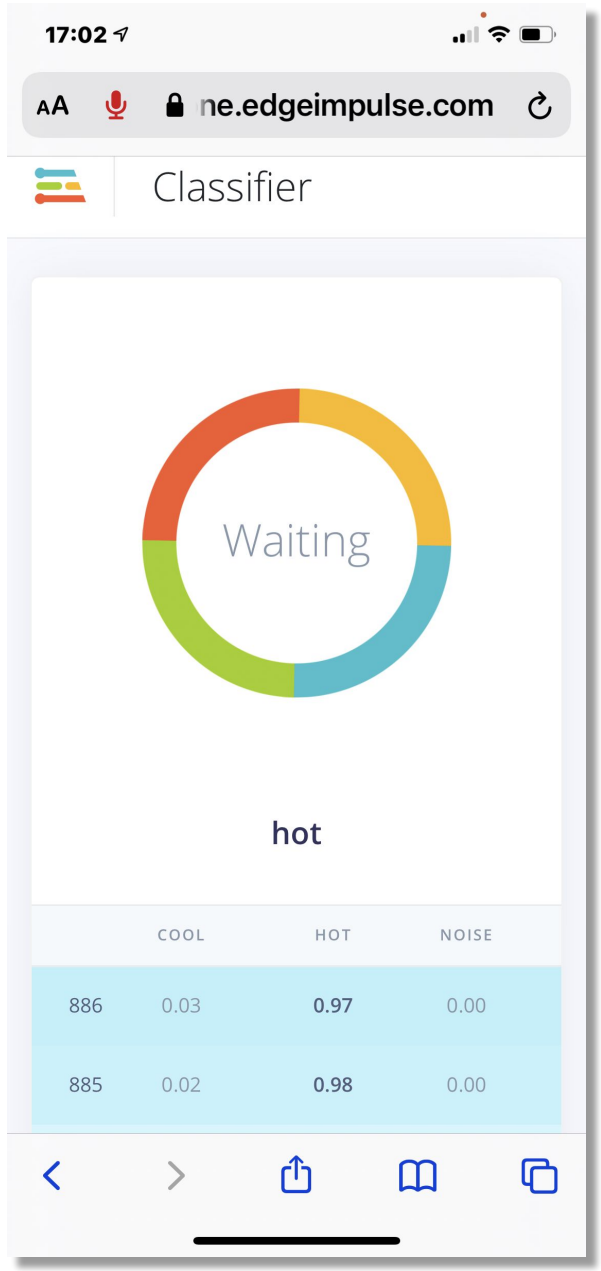
Feature explorer

- cool - correct
- hot - correct
- noise - correct
- cool - incorrect









Live Classifier (Off line) using iphone

Select optimizations *(optional)*

Model optimizations can increase on-device performance but may reduce accuracy. Click below to analyze optimizations and see the recommended choices for your target. Or, just click Build to use the currently selected options.

Enable EON™ Compiler

Same accuracy, up to 50% less memory. Open source.

Available optimizations for NN Classifier

Quantized (int8) ⭐

Currently selected

This optimization is recommended for best performance.

RAM USAGE	LATENCY
10.9K	17 ms
FLASH USAGE	**ACCURACY**
31.4K	86.67%

CONFUSION MATRIX

66.7	0	33.3	0
0	100	0	0
0	0	100	0

Unoptimized (float32)

Click to select

RAM USAGE	LATENCY
33.9K	78 ms
FLASH USAGE	**ACCURACY**
38.0K	86.67%

CONFUSION MATRIX

66.7	0	33.3	0
0	100	0	0
0	0	100	0

Estimate for Cortex-M4F 80MHz (ST IoT Discovery Kit)

Arduino File Edit Sketch Tools Help

Sun 16:45

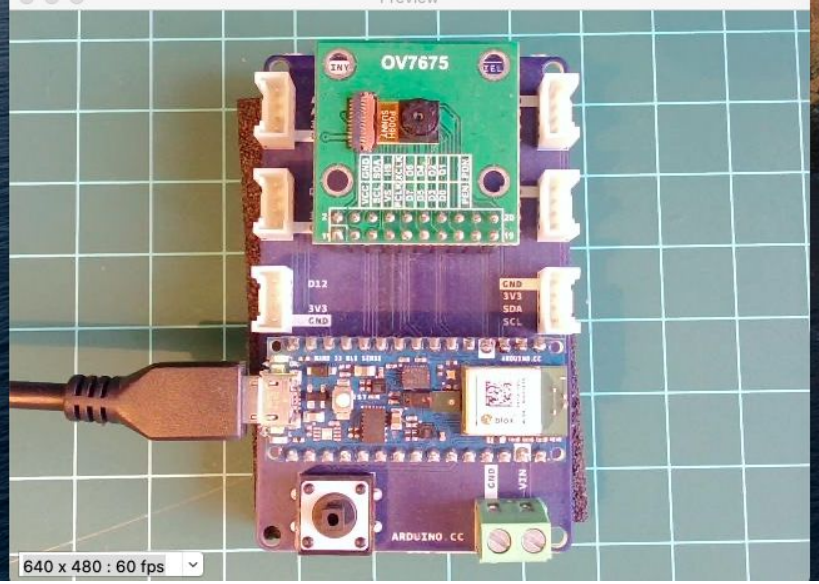
/dev/cu.usbmodem144301

Send

```
ICTP - PSYCOACOUSTICS TEMPERATURE Project
Inferencing settings:
        Interval: 0.06 ms.
        Frame size: 16000
        Sample length: 1000 ms.
        No. of classes: 3
Predictions (DSP: 126 ms., Classification: 21 ms., Anomaly: 0 ms.):
:
  PREDICTION: ==> noise with probability 1.00
:
Predictions (DSP: 126 ms., Classification: 21 ms., Anomaly: 0 ms.):
:
  PREDICTION: ==> noise with probability 1.00
:
Predictions (DSP: 126 ms., Classification: 20 ms., Anomaly: 0 ms.):
:
  PREDICTION: ==> noise with probability 1.00
```

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

Preview



nano_ble33_sense_microphone_continuous_LED | Arduino 1.8.15

nano_ble33_sense_microphone_continuous_LED

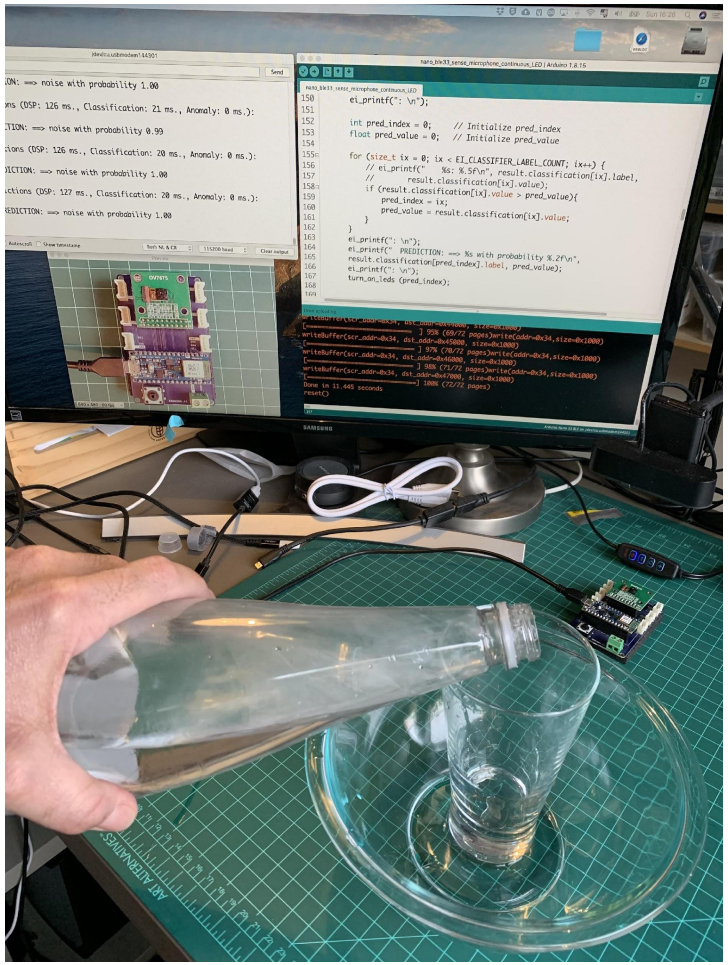
```
/**
 * @brief      Special Postprocess function for RGB LEDs
 */

void turn_off_leds(){
    digitalWrite(LEDR, HIGH);
    digitalWrite(LEDG, HIGH);
    digitalWrite(LEDB, HIGH);
}

/*
 * cool:  [0] ==> Blue ON
 * hot:   [1] ==> Red ON
 * noise: [2] ==> ALL OFF
 */

void turn_on_leds(int pred_index) {
  switch (pred_index)
  {
    case 0:
      turn_off_leds();
      digitalWrite(LEDB, LOW);
      break;

    case 1:
      turn_off_leds();
      digitalWrite(LEDR, LOW);
      break;

    case 2:
      turn_off_leds();
      break;
  }
}

```

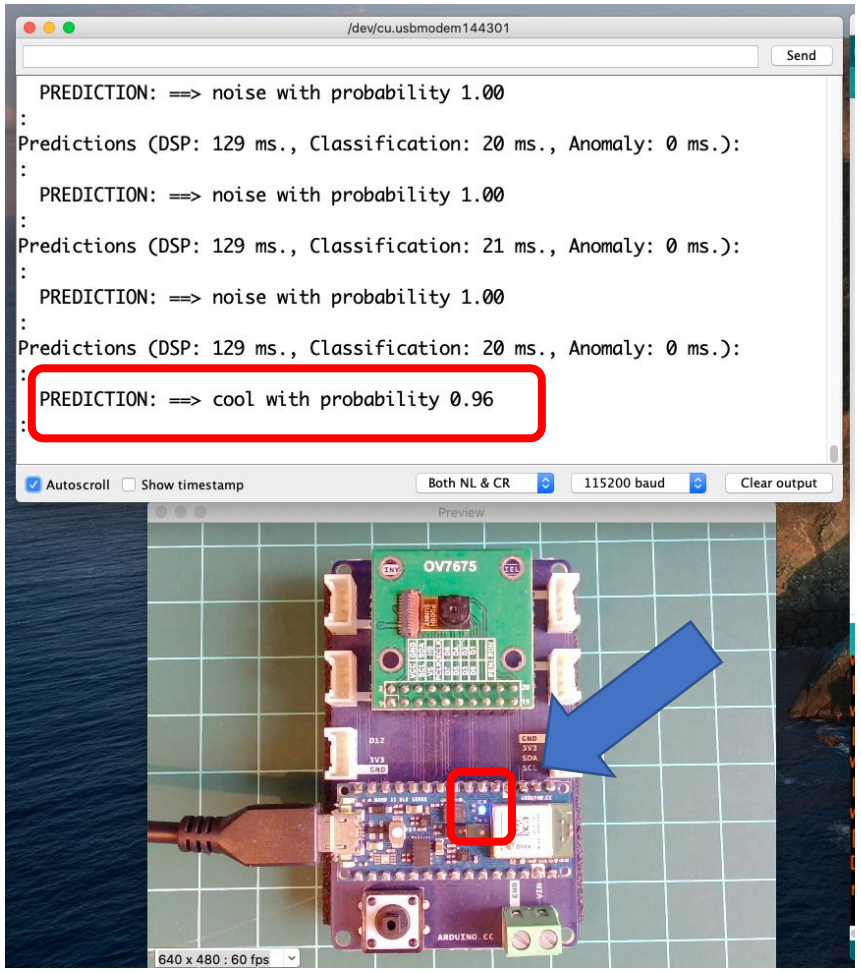
Done uploading.

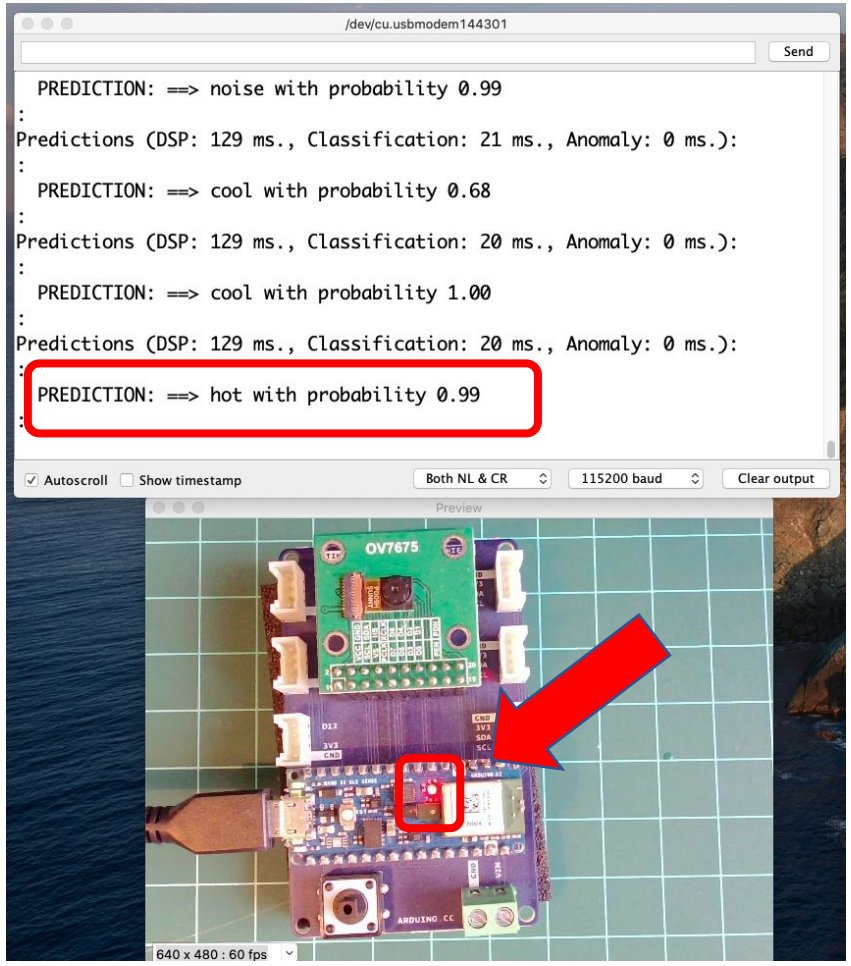
```
writeBuffer(scr_addr=0x34, dst_addr=0x44000, size=0x1000)
[========================  ] 95% (69/72 pages)write(addr=0x34,size=0x1000)
writeBuffer(scr_addr=0x34, dst_addr=0x45000, size=0x1000)
```

87

Arduino Nano 33 BLE on /dev/cu.usbmodem144301




/dev/cu.usbmodem144301
Send
PREDICTION: ==> noise with probability 1.00
:
Predictions (DSP: 129 ms., Classification: 20 ms., Anomaly: 0 ms.):
:
PREDICTION: ==> noise with probability 1.00
:
Predictions (DSP: 129 ms., Classification: 21 ms., Anomaly: 0 ms.):
:
PREDICTION: ==> noise with probability 1.00
:
Predictions (DSP: 129 ms., Classification: 20 ms., Anomaly: 0 ms.):
:
PREDICTION: ==> cool with probability 0.96
:
Autoscroll
Show timestamp
Both NL & CR
115200 baud
Clear output
Preview
OV7675
640 x 480 : 60 fps


/dev/cu.usbmodem144301
Send
PREDICTION: ==> noise with probability 0.99
:
Predictions (DSP: 129 ms., Classification: 21 ms., Anomaly: 0 ms.):
:
PREDICTION: ==> cool with probability 0.68
:
Predictions (DSP: 129 ms., Classification: 20 ms., Anomaly: 0 ms.):
:
PREDICTION: ==> cool with probability 1.00
:
Predictions (DSP: 129 ms., Classification: 20 ms., Anomaly: 0 ms.):
:
PREDICTION: ==> hot with probability 0.99
:
Autoscroll
Show timestamp
Both NL & CR
115200 baud
Clear output
Preview
OV7675
640 x 480 : 60 fps

Reading Material

Main references

- Harvard School of Engineering and Applied Sciences - CS249r: Tiny Machine Learning
- Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard
- Introduction to Embedded Machine Learning - Coursera/Edge Impulse
- Computer Vision with Embedded Machine Learning - Coursera/Edge Impulse
- Fundamentals textbook: “Deep Learning with Python” by François Chollet
- Applications & Deploy textbook: “TinyML” by Pete Warden, Daniel Situnayake
- Deploy textbook “TinyML Cookbook” by Gian Marco Iodice

I want to thank Shawn Hymel and Edge Impulse, Pete Warden and Laurence Moroney from Google, Professor Vijay Janapa Reddi and Brian Plancher from Harvard, and the rest of the TinyMLedu team for preparing the excellent material on TinyML that is the basis of this course at UNIFEI.

The IESTI01 course is part of the **TinyML4D**, an initiative to make TinyML education available to everyone globally.

Thanks

UNIFEI