

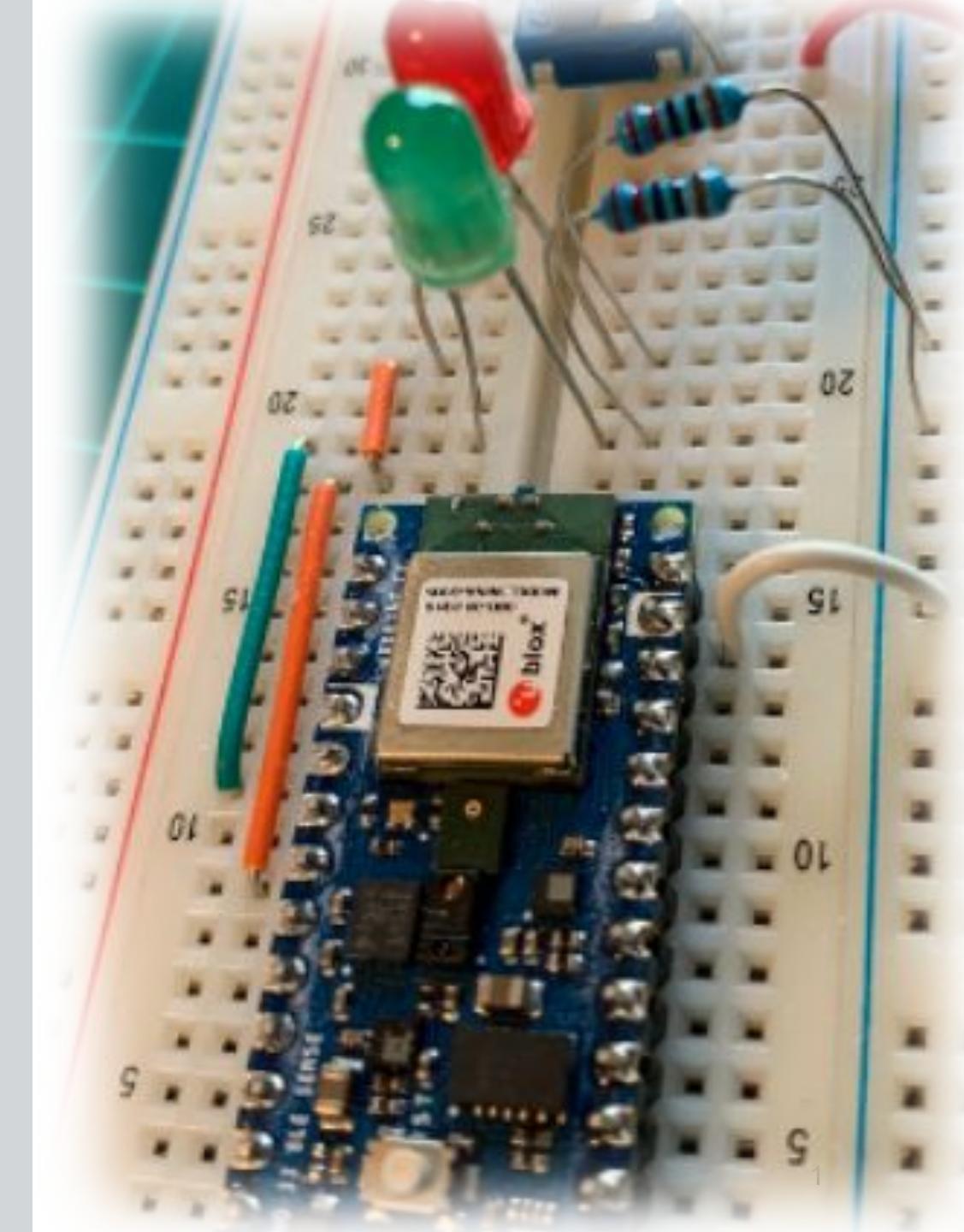
IESTI01 – TinyML

Embedded Machine Learning

27. Collecting Data with Edge Impulse Studio



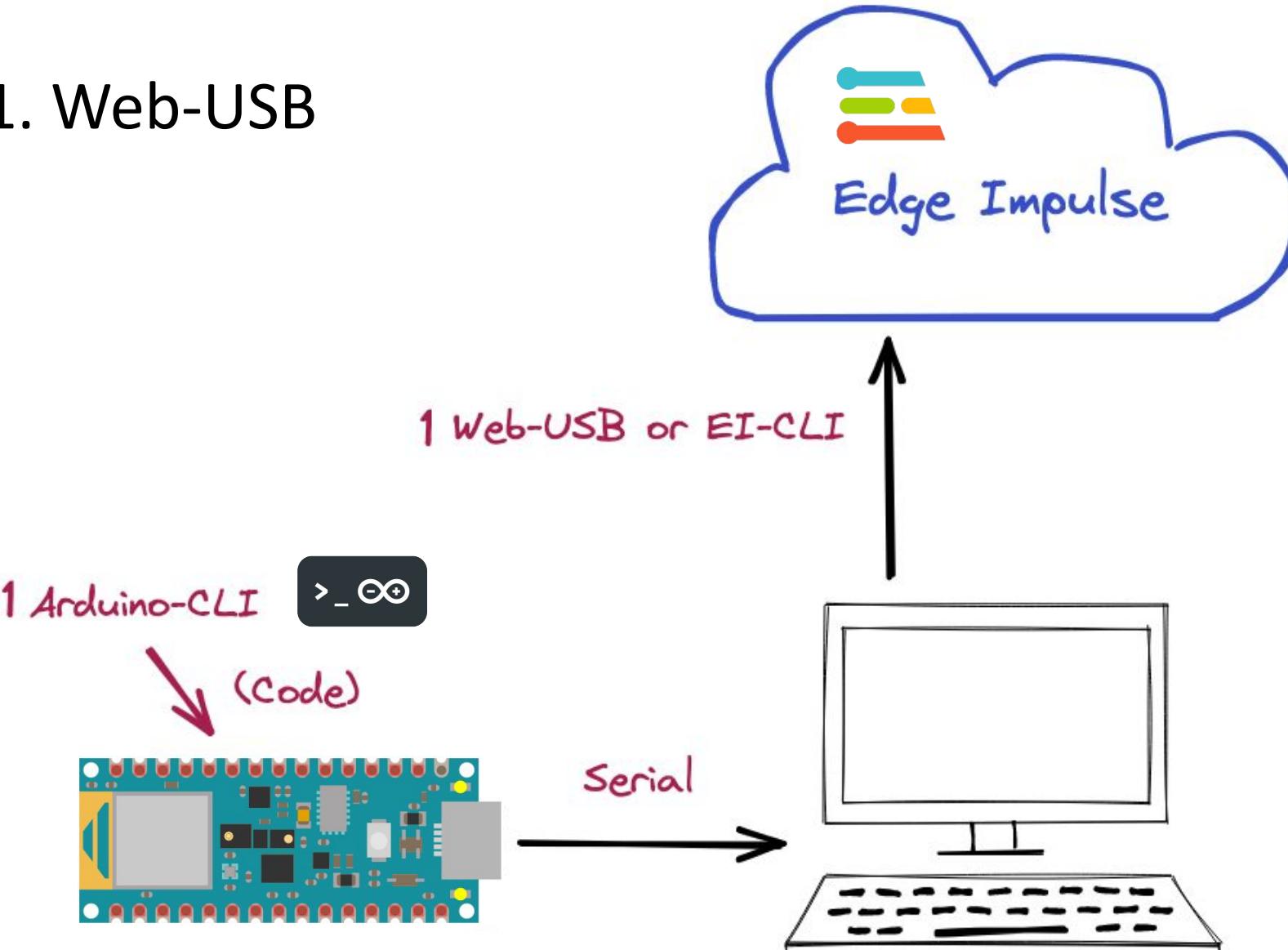
Prof. Marcelo Rovai
UNIFEI



Sensor Data – EI Studio Ingestion

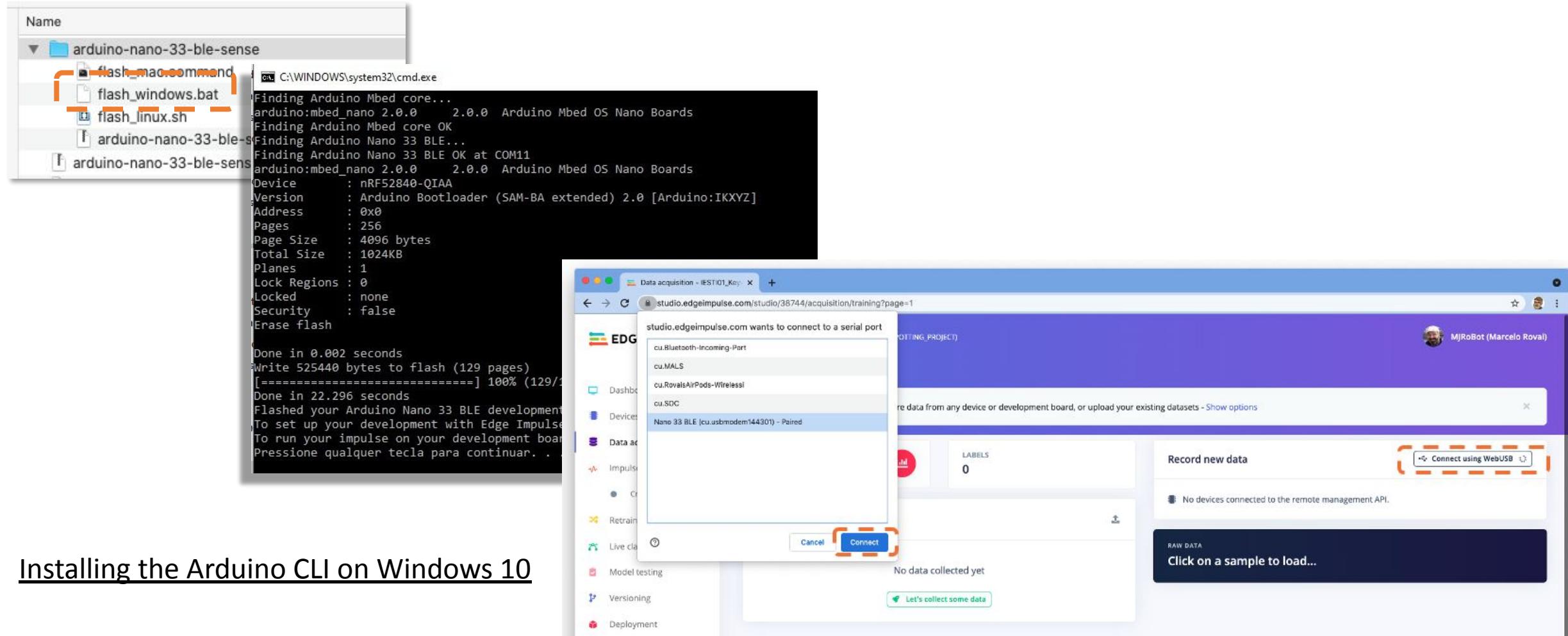
Alternative methods

1. Web-USB



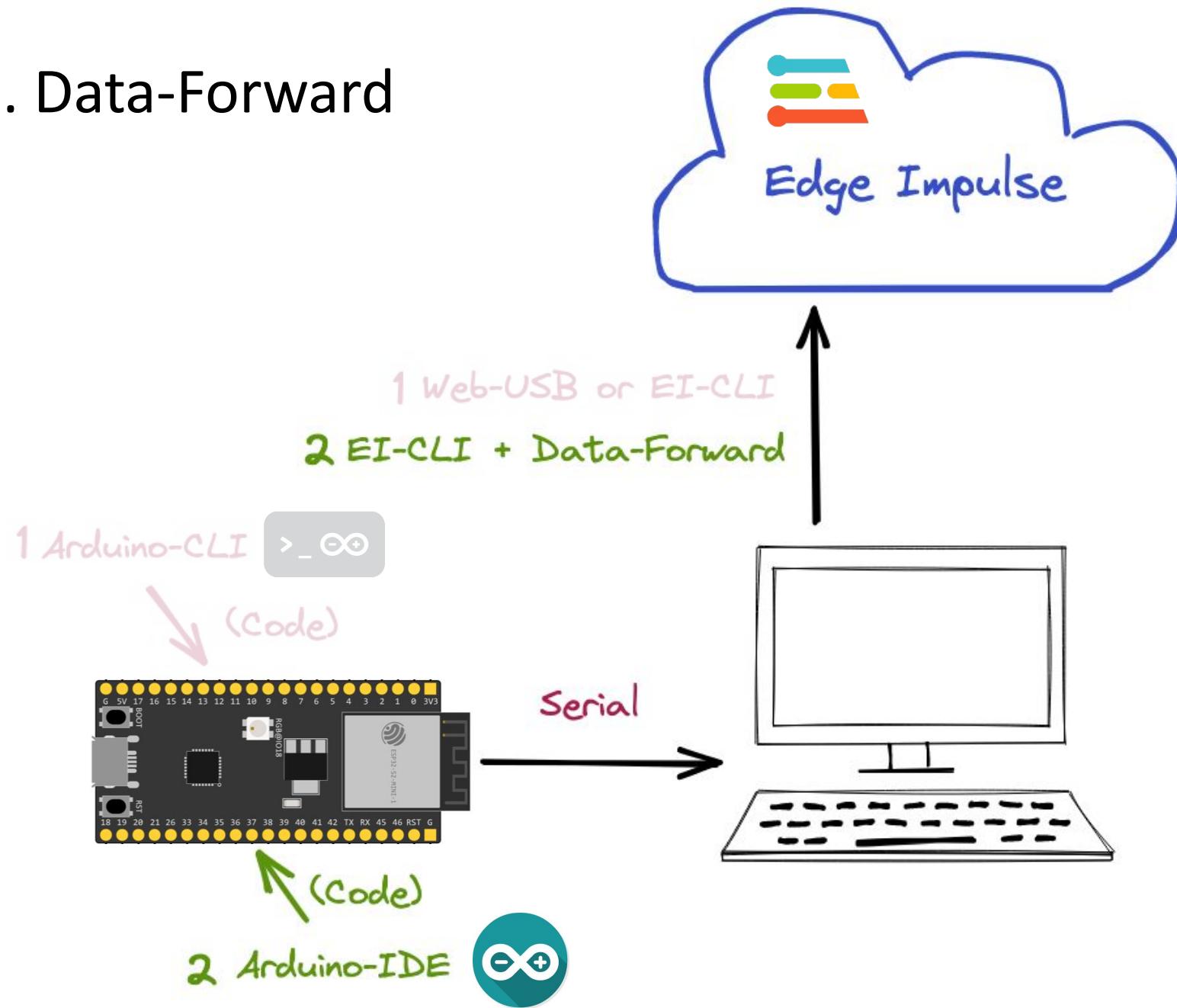
Issue: Limited MCU and sensors

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



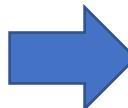
Installing the Arduino CLI on Windows 10

2. Data-Forward



2. Data Ingestion using El-Cli + Data Forward

```
Capture_Ardub33_Sense_IMU_Acc
1 #include <Arduino_LSM9DS1.h>
2
3 #define CONVERT_G_TO_MS2 9.80665f
4 #define FREQUENCY_HZ 50
5 #define INTERVAL_MS (1000 / (FREQUENCY_HZ + 1))
6
7 void setup() {
8     Serial.begin(115200);
9     while (!Serial);
10    Serial.println("Started");
11
12    if (!IMU.begin()) {
13        Serial.println("Failed to initialize IMU!");
14        while (1);
15    }
16 }
17
18 void loop() {
19     static unsigned long last_interval_ms = 0;
20     float x, y, z;
21
22     if (millis() > last_interval_ms + INTERVAL_MS) {
23         last_interval_ms = millis();
24
25         IMU.readAcceleration(x, y, z);
26
27         Serial.print(x * CONVERT_G_TO_MS2);
28         Serial.print(',');
29         Serial.print(y * CONVERT_G_TO_MS2);
30         Serial.print(',');
31         Serial.println(z * CONVERT_G_TO_MS2);
32     }
33 }
```



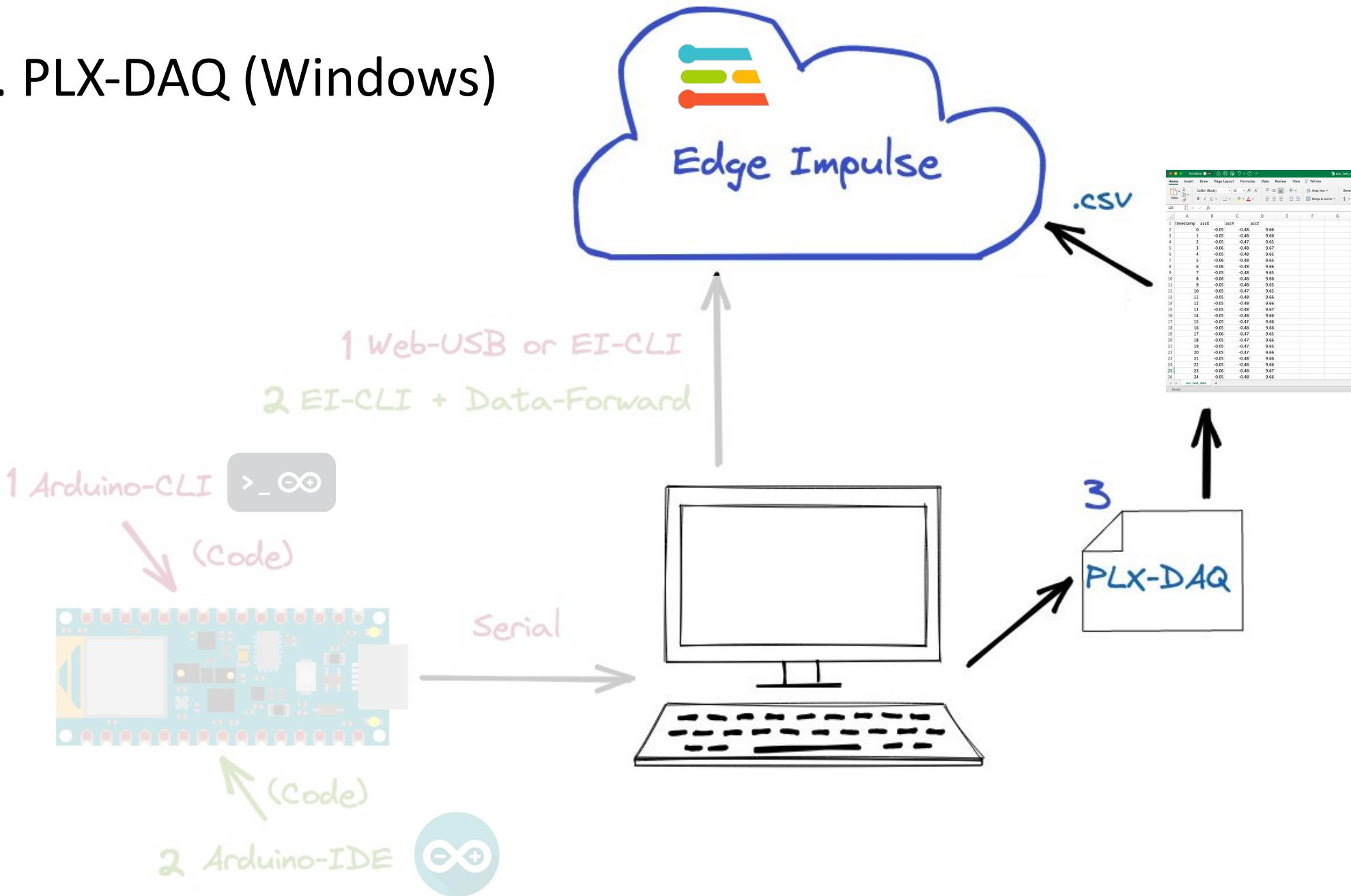
```
$ edge-impulse-data-forwarder --clean
```

```
mjrovai — bash — 80x41
(base) MacBook-Pro-de-Marcelo:~ mjrovai$ edge-impulse-data-forwarder --clean
Edge Impulse data forwarder v1.12.2
[?] What is your user name or e-mail address (edgeimpulse.com)? rovai@mjrobot.org
[?] What is your password? [hidden]
Endpoints:
  WebSocket: wss://remote-mgmt.edgeimpulse.com
  API: https://studio.edgeimpulse.com/v1
  Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to /dev/tty.usbmodem144301
[SER] Serial is connected (4A:5A:36:17:55:F9:70:F7)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com

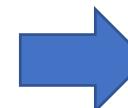
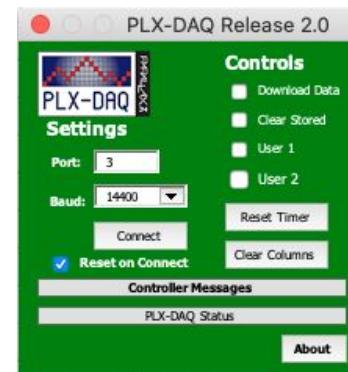
? To which project do you want to connect this device? MJRoBot (Marcelo Rovai) / IESTI01_Input_Data_Test
[SER] Detecting data frequency...
[SER] Detected data frequency: 51Hz
[?] 3 sensor axes detected (example values: [-0.08,-0.34,9.82]). What do you want to call them? Separate the names with ',': accX, accY, accZ
? What name do you want to give this device? nano
[WS ] Device "nano" is now connected to project "IESTI01_Input_Data_Test"
[WS ] Go to https://studio.edgeimpulse.com/studio/39877/acquisition/training to build your machine learning model!
[WS ] Incoming sampling request (
  path: '/api/training/data',
  label: 'left-right',
  length: 10000,
  interval: 19.607843137254903,
  hmacKey: '6ee929b90e563aa74517f505a3ecb9c8',
  sensor: 'Sensor with 3 axes (accX, accY, accZ)'
```

3. PLX-DAQ (Windows)



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

```
Capture_Ard33_Sense_IMU_Acc
1 #include <Arduino_LSM9DS1.h>
2
3 #define CONVERT_G_TO_MS2 9.80665f
4 #define FREQUENCY_HZ 50
5 #define INTERVAL_MS (1000 / (FREQUENCY_HZ + 1))
6
7 void setup() {
8     Serial.begin(115200);
9     while (!Serial);
10    Serial.println("Started");
11
12    if (!IMU.begin()) {
13        Serial.println("Failed to initialize IMU!");
14        while (1);
15    }
16 }
17
18 void loop() {
19     static unsigned long last_interval_ms = 0;
20     float x, y, z;
21
22     if (millis() > last_interval_ms + INTERVAL_MS) {
23         last_interval_ms = millis();
24
25         IMU.readAcceleration(x, y, z);
26
27         Serial.print(x * CONVERT_G_TO_MS2);
28         Serial.print(',');
29         Serial.print(y * CONVERT_G_TO_MS2);
30         Serial.print(',');
31         Serial.println(z * CONVERT_G_TO_MS2);
32     }
33 }
```



	A	B	C	D	E	F	G
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

The screenshot shows the Arduino IDE interface. The top menu bar includes 'Arquivo', 'Editar', 'Sketch', 'Ferramentas', and 'Ajuda'. The title bar says 'IMU_excel | Arduino 1.8.15'. The code editor contains the following sketch:

```
#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(9600);
    while (!Serial)
    {
        // do nothing
    }
    //limpa todos os dados do sheet incluindo os labels
    Serial.println("CLEARSHHEET");
    Serial.println("CLEARDATA");
    //delay para dar o tempo suficiente para excluir os labels originais a fim de substitui-los
    delay(300);
    //escrevendo os nomes das colunas da planilha. É sempre necessário escrever o nome LABEL com o objetivo de sinalizar de que se quer nomear as colunas.
    //Após isso, basta colocar os nomes das colunas, sempre separando-as por vírgulas.
    //O nome "timestamP" para representar o horário pelo qual foram adicionados os dados é escolhido pensando na possibilidade de subir esses dados no EdadImpulse.
}

Carregado:
writeBuffer(scr_addr=0x34, dst_addr=0x14000, size=0x1000)
[=====] 87% (21/24 pages) write(addr=0x34, size=0x1000)
writeBuffer(scr_addr=0x34, dst_addr=0x15000, size=0x1000)
[=====] 91% (22/24 pages) write(addr=0x34, size=0x1000)
writeBuffer(scr_addr=0x34, dst_addr=0x16000, size=0x1000)
[=====] 95% (23/24 pages) write(addr=0x34, size=0x1000)
writeBuffer(scr_addr=0x34, dst_addr=0x17000, size=0x1000)
[=====] 100% (24/24 pages)
Done in 3.995 seconds
reset()

48
```

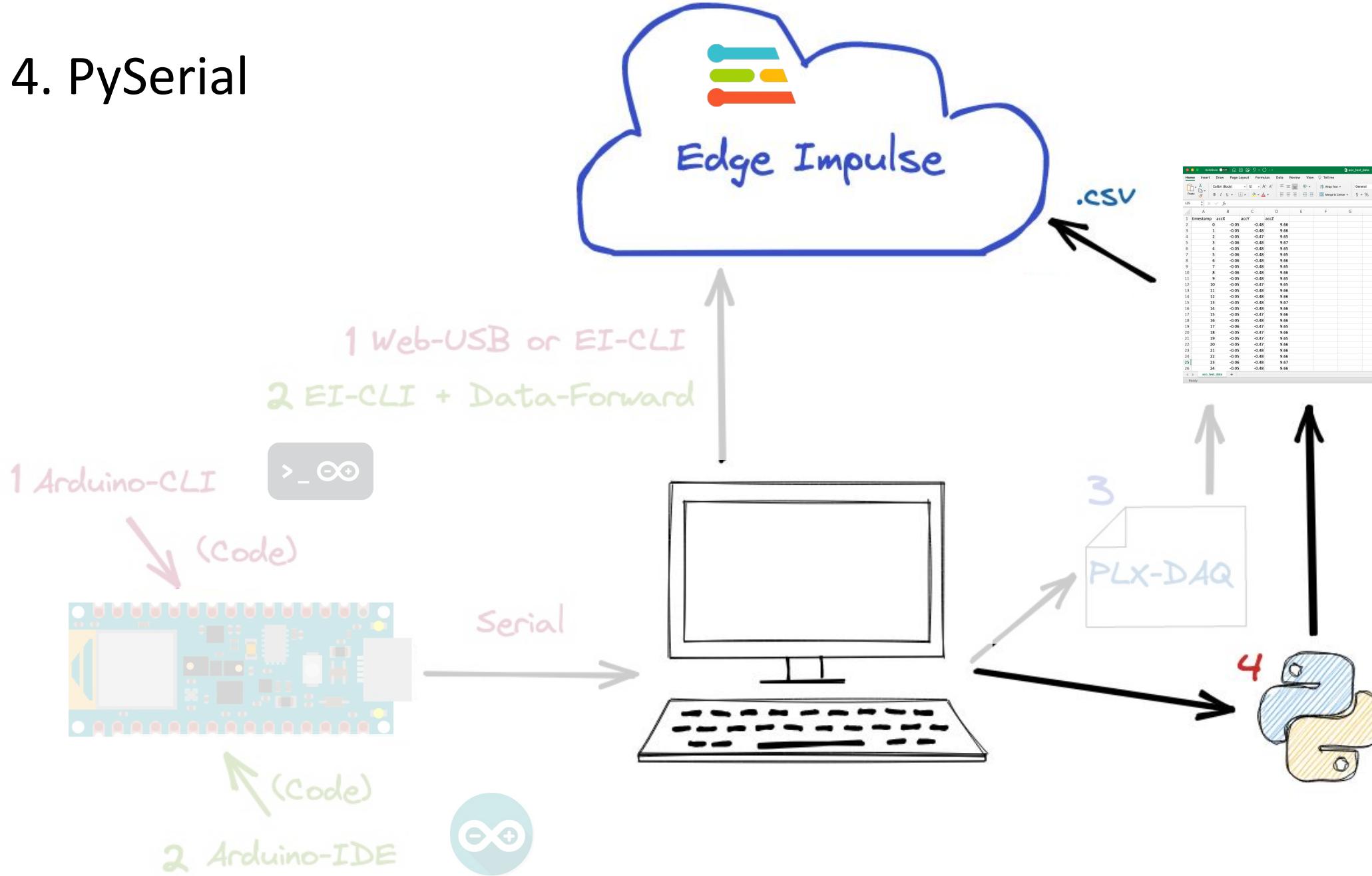
The Serial Monitor window shows the progress of writing data to memory:

```
[=====] 87% (21/24 pages) write(addr=0x34, size=0x1000)
[=====] 91% (22/24 pages) write(addr=0x34, size=0x1000)
[=====] 95% (23/24 pages) write(addr=0x34, size=0x1000)
[=====] 100% (24/24 pages)
```

The status bar at the bottom indicates 'Arduino Nano 33 BLE em COM10'.

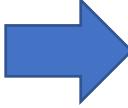
<https://www.youtube.com/watch?v=BwbmNle2CZo>

4. PySerial



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

```
Capture_Ardub33_Sense_IMU_Acc
1 #include <Arduino_LSM9DS1.h>
2
3 #define CONVERT_G_TO_MS2    9.80665f
4 #define FREQUENCY_HZ        50
5 #define INTERVAL_MS          (1000 / (FREQUENCY_HZ + 1))
6
7 void setup() {
8     Serial.begin(115200);
9     while (!Serial);
10    Serial.println("Started");
11
12    if (!IMU.begin()) {
13        Serial.println("Failed to initialize IMU!");
14        while (1);
15    }
16 }
17
18 void loop() {
19     static unsigned long last_interval_ms = 0;
20     float x, y, z;
21
22    if (millis() > last_interval_ms + INTERVAL_MS) {
23        last_interval_ms = millis();
24
25        IMU.readAcceleration(x, y, z);
26
27        Serial.print(x * CONVERT_G_TO_MS2);
28        Serial.print(',');
29        Serial.print(y * CONVERT_G_TO_MS2);
30        Serial.print(',');
31        Serial.println(z * CONVERT_G_TO_MS2);
32    }
33 }
```

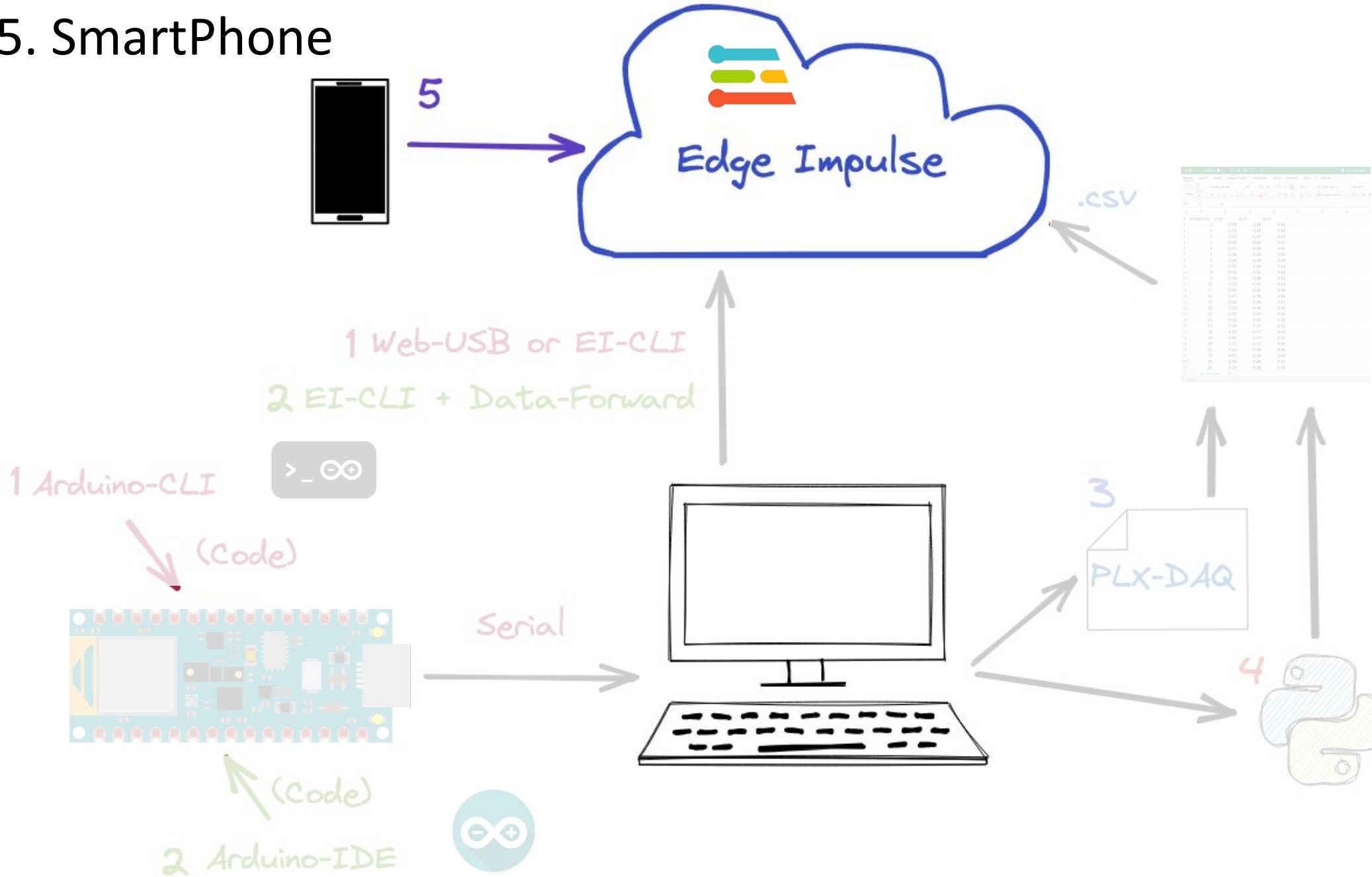


```
1 # Sensor data Logger (CSV)
2 # by Marcelo Rovai @ 13July21
3
4 import serial
5
6 arduino_port = '/dev/tty.usbmodem144301'
7 baud_rate = 115200
8 ser = serial.Serial(port=arduino_port, baudrate=baud_rate)
9
10 fileName = "acc_test_data.csv" # name of the CSV file generated
11
12 first_line = 'timestamp,accX,accY,accZ'
13 file = open(fileName, "w")
14 file.write(first_line + "\n") # write data with a newline
15 file.close()
16
17 Freq_hz = 50
18 num_seconds = 10 # number of seconds collecting data
19 samples = num_seconds * Freq_hz # number of samples to collect
20
21 sample = 0
22 while sample <= samples:
23     getData = str(ser.readline())
24     data = getData[2:][:-5]
25     print(data)
26
27     file = open(fileName, "a")
28     file.write(str(sample) + "," + data + "\n")
29     sample = sample+1
30 print("Data collection complete!")
31 file.close()
```

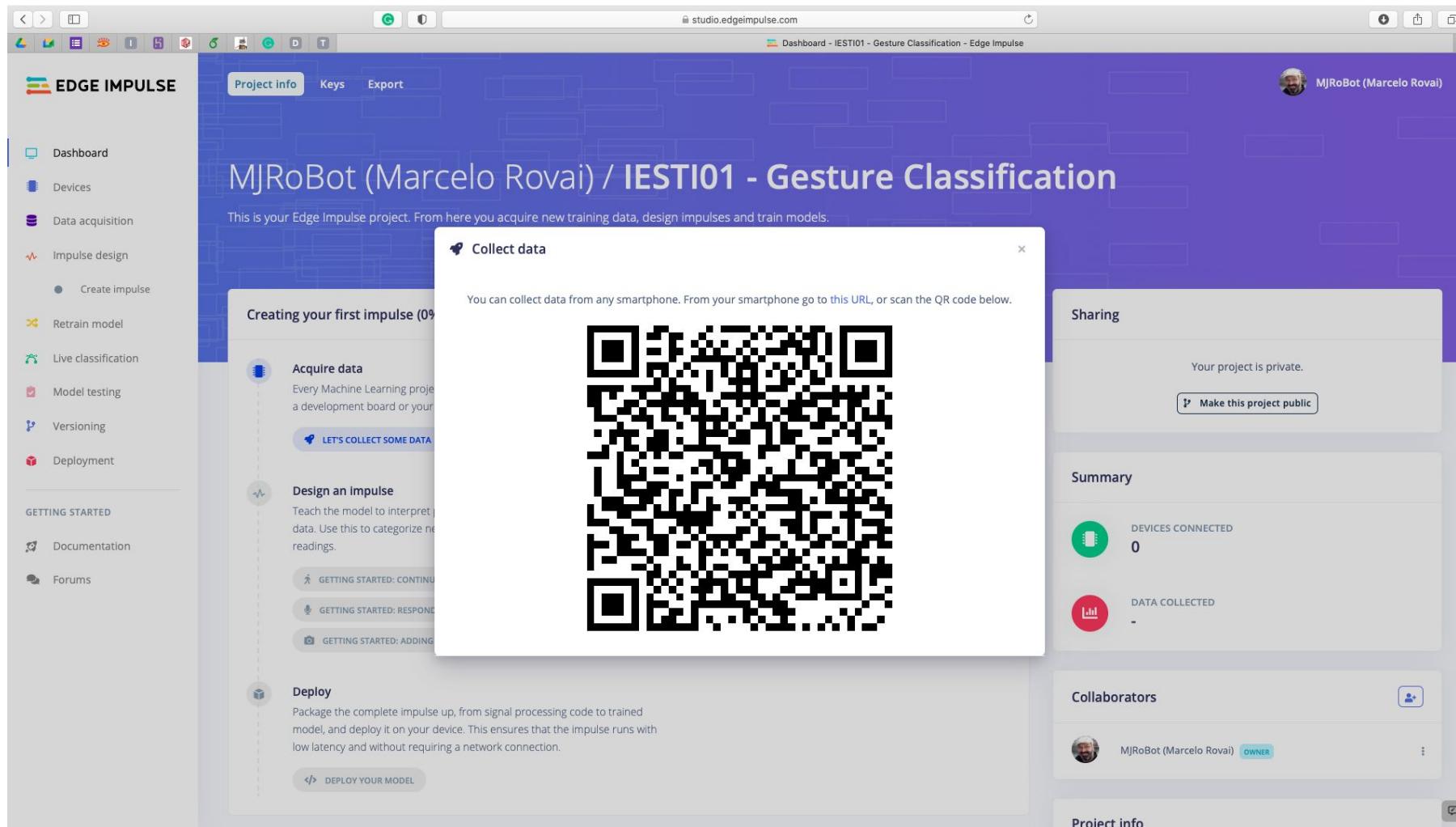


L25	A	B	C	D	E	F	G
1	timestamp	accX	accY	accZ			
2	0	-0.05	-0.48	9.66			
3	1	-0.05	-0.48	9.66			
4	2	-0.05	-0.47	9.65			
5	3	-0.06	-0.48	9.67			
6	4	-0.05	-0.48	9.65			
7	5	-0.06	-0.48	9.65			
8	6	-0.06	-0.48	9.66			
9	7	-0.05	-0.48	9.65			
10	8	-0.06	-0.48	9.66			
11	9	-0.05	-0.48	9.65			
12	10	-0.05	-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			

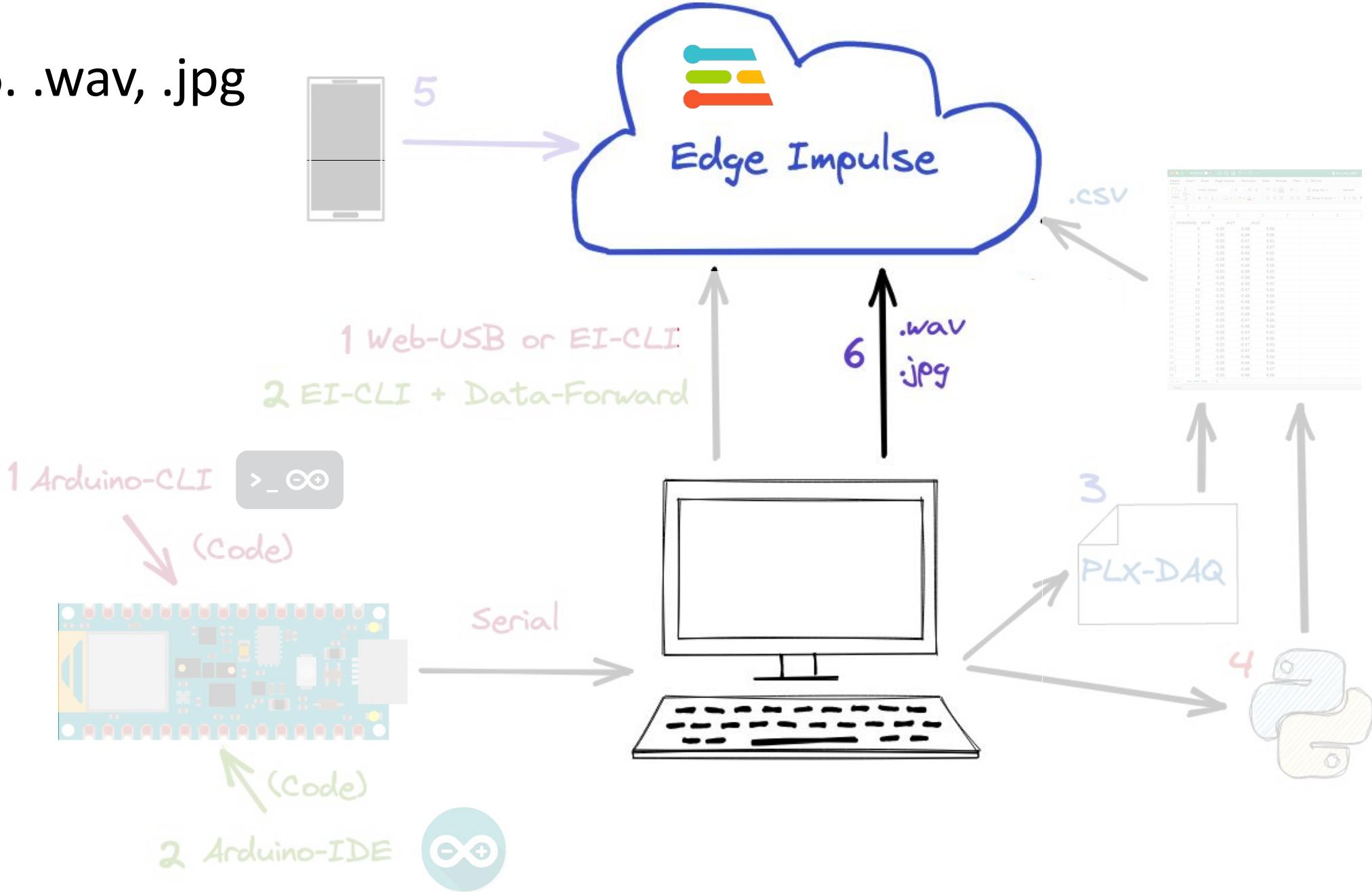
5. SmartPhone



5. Data Ingestion using Smart Phone



6. .wav, .jpg



6. Data Ingestion using Upload existing Data

The screenshot illustrates the process of uploading data to a project. On the left, a file tree shows a 'data' folder containing 'cool' and 'hot' subfolders. A blue arrow points from the 'hot' folder to the 'Upload existing data' button in the center. The central area is a modal titled 'UPLOAD DATA (ICTP_PSYCHOACOUSTICS_TEMPERATURE_DEPENDENCE)'. It contains a sub-modal for 'Upload existing data' with instructions: 'You can upload existing data to your project in the Data Acquisition Format (CBOR, JSON, CSV), or as WAV, JPG or PNG files.' Below this is a 'Select files' section with a 'Choose Files' button and a 'No file chosen' message. Under 'Upload into category', the 'Training' radio button is selected. In the 'Label' section, the 'Enter label:' radio button is selected, and the word 'hot' is typed into the input field. At the bottom is a green 'Begin upload' button. To the right, a progress bar shows 'Uploading 14 files...' with a list of 14 successful uploads. The progress bar ends with 'Done. Files uploaded successful: 14. Files that failed to upload: 0.' and a 'Job completed' message.

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

LABELS
5

Upload existing data

ADDED LENGTH

data cool hot

20210710-125854.wav
20210710-125930.wav
20210710-125956.wav
20210710-130010.wav
20210710-130041.wav
20210710-130100.wav
20210710-130119.wav
20210710-130129.wav
20210710-130142.wav
20210710-130200.wav
20210710-130212.wav
20210710-130224.wav
20210710-130236.wav
20210710-130246.wav
20210710-130256.wav
20210710-130304.wav
20210710-130315.wav
20210710-130329.wav
20210710-130348.wav
20210710-130358.wav
20210710-130408.wav
20210710-130416.wav

Upload existing data

You can upload existing data to your project in the Data Acquisition Format (CBOR, JSON, CSV), or as WAV, JPG or PNG files.

Select files

Choose Files No file chosen

Upload into category

Automatically split between training and testing

Training

Testing

Label

Infer from filename

Enter label:

hot

Begin upload

UPLOAD DATA (ICTP_PSYCHOACOUSTICS_TEMPERATURE_DEPENDENCE)

MJRoBot (Marcelo Roval)

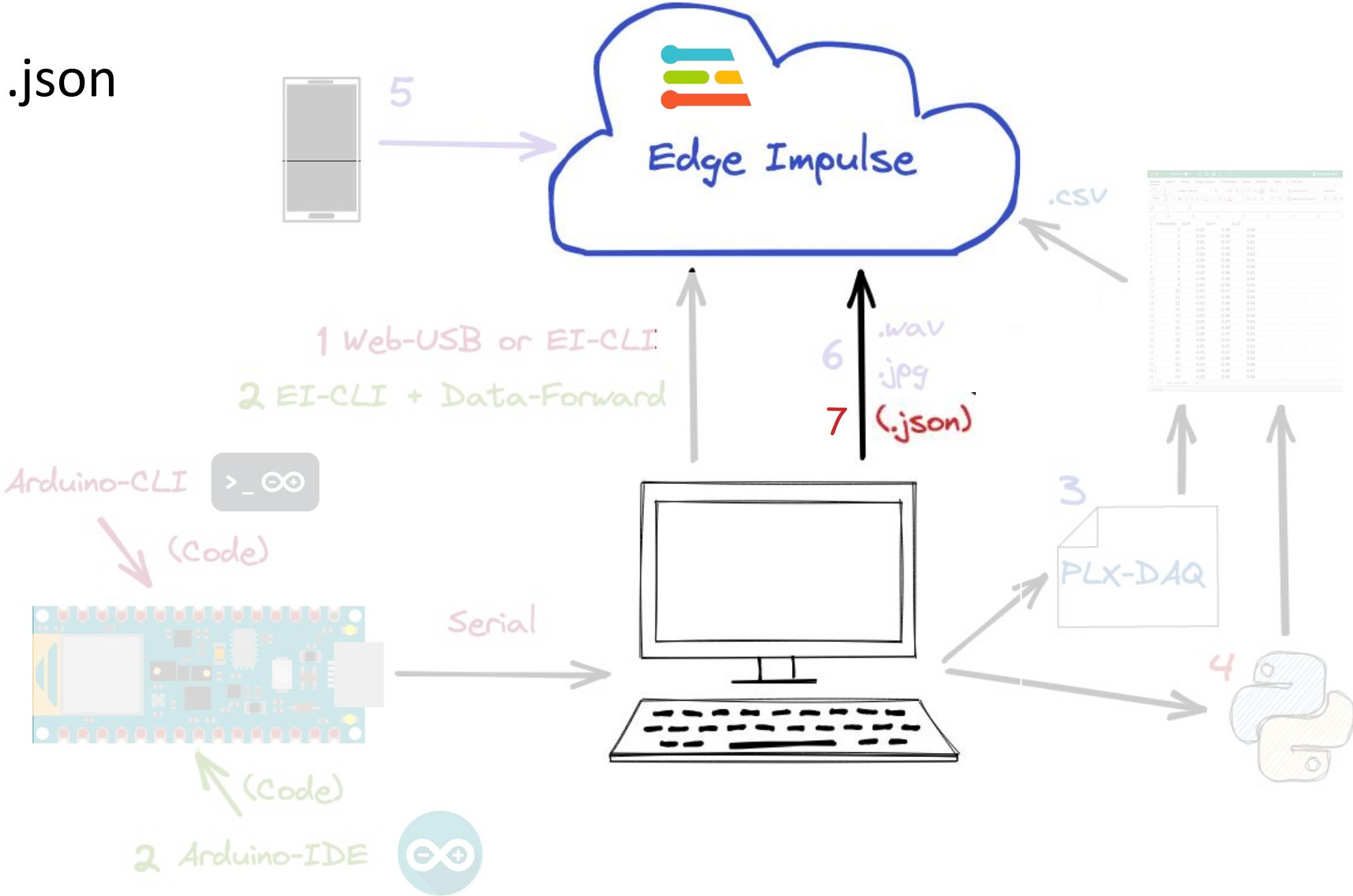
Uploading 14 files...

[1/14] Uploading 20210710-130535.wav OK
[2/14] Uploading 20210710-130603.wav OK
[3/14] Uploading 20210710-130544.wav OK
[4/14] Uploading 20210710-130553.wav OK
[5/14] Uploading 20210710-130738.wav OK
[6/14] Uploading 20210710-130718.wav OK
[7/14] Uploading 20210710-130649.wav OK
[8/14] Uploading 20210710-130700.wav OK
[9/14] Uploading 20210710-130630.wav OK
[10/14] Uploading 20210710-130621.wav OK
[11/14] Uploading 20210710-130709.wav OK
[12/14] Uploading 20210710-130611.wav OK
[13/14] Uploading 20210710-130728.wav OK
[14/14] Uploading 20210710-130639.wav OK

Done. Files uploaded successful: 14. Files that failed to upload: 0.

Job completed

7. .json



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.

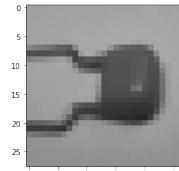
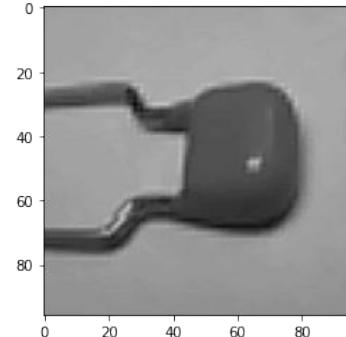
Create a folder named "dataset" in the /content directory and upload your images there. The images should be divided into their respective classes, where each class has its own folder with the name of the class. For example:

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

Author: EdgImpulse, Inc.

Date: June 6, 2021

License: [Apache-2.0](#)

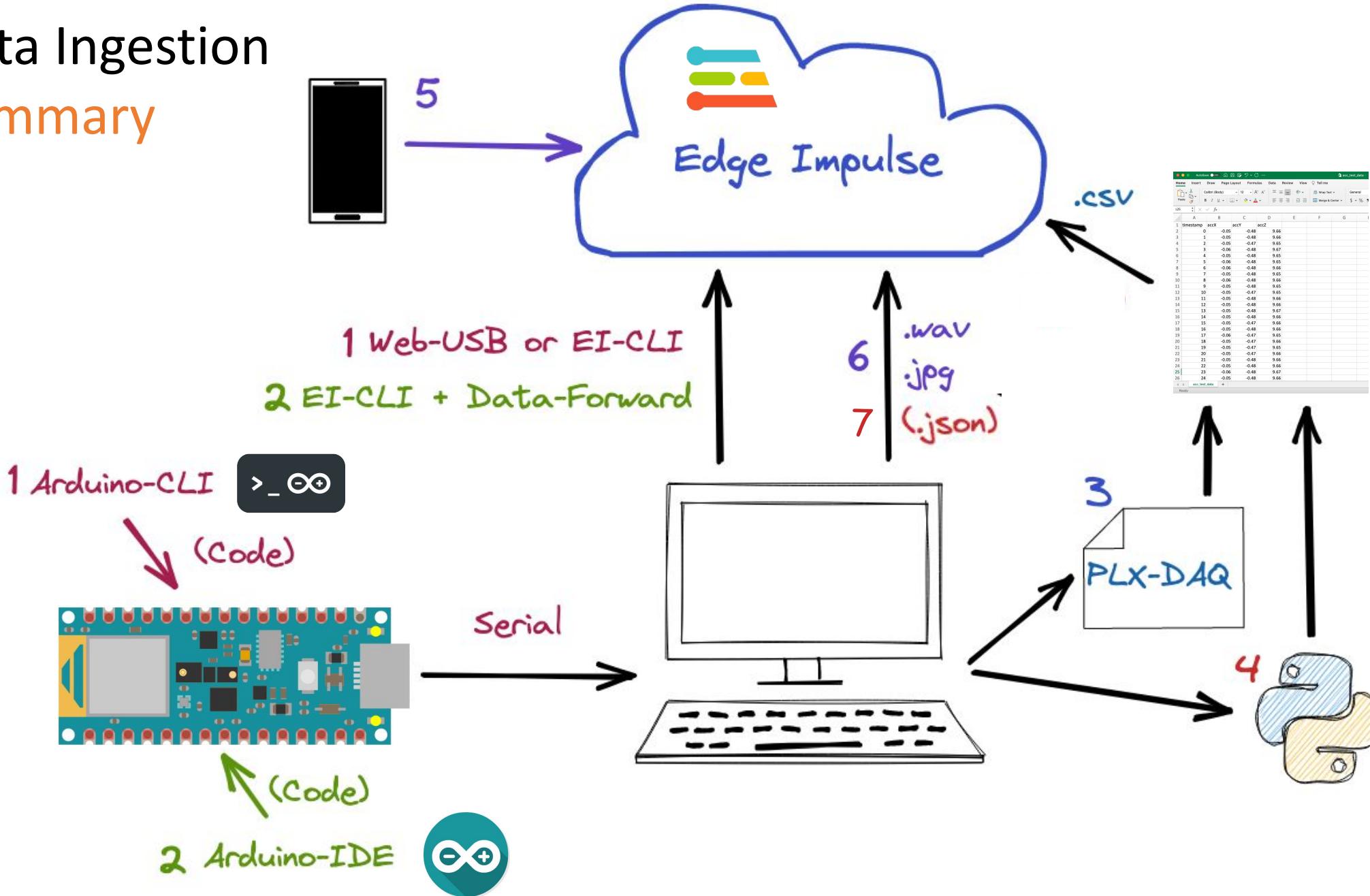


.json 



EDGE IMPULSE
(Training as DNN)

Data Ingestion Summary



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

This Engineering Brief was selected on the basis of a submitted synopsis. The author is solely responsible for its presentation, and the AES takes no responsibility for the contents. All rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

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

He Peng¹ and Joshua D. Reiss²

¹Tianjin University

²Queen 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](http://www.eecs.qmul.ac.uk/~josh/documents/2018/19737.pdf)

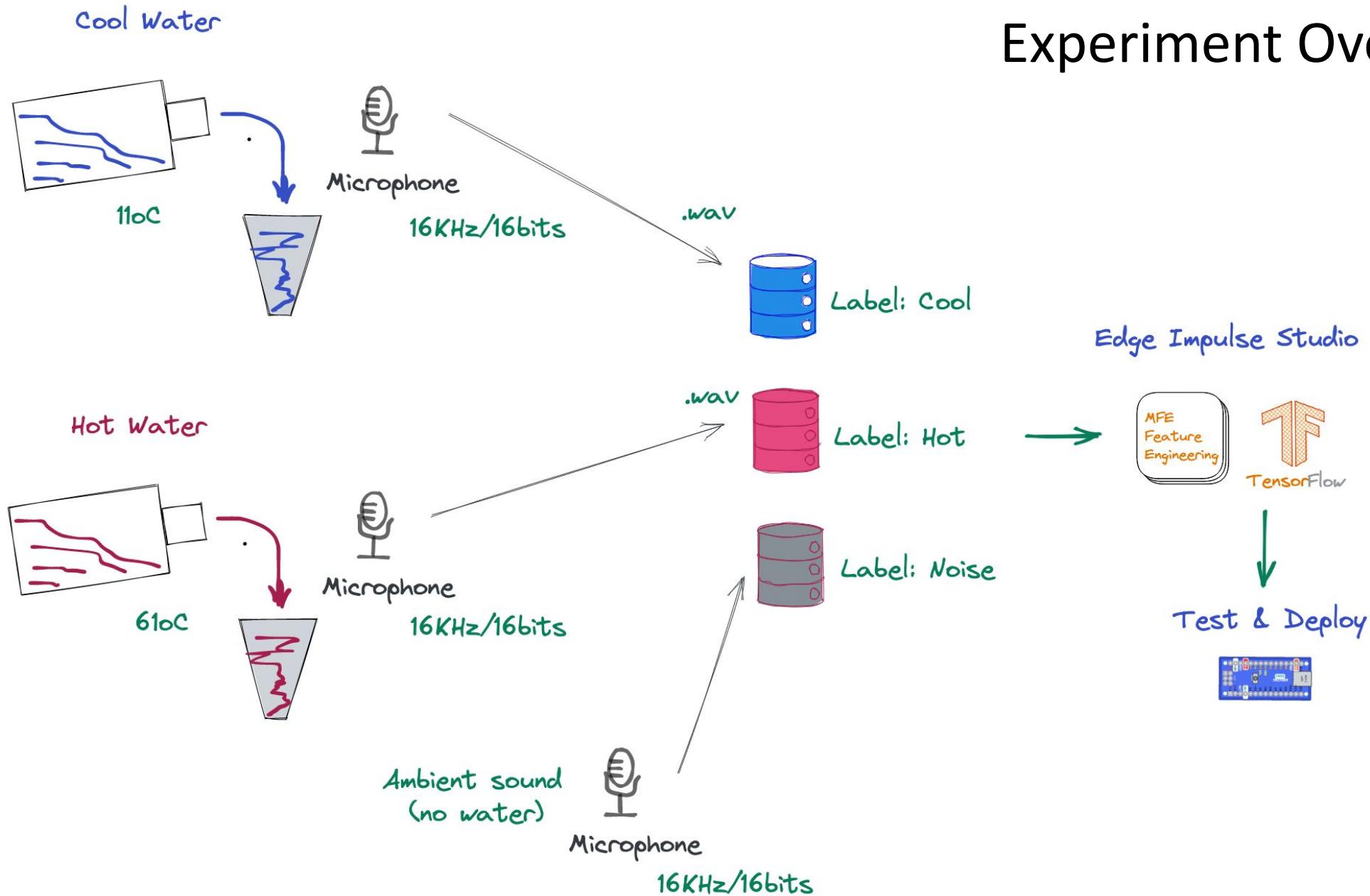


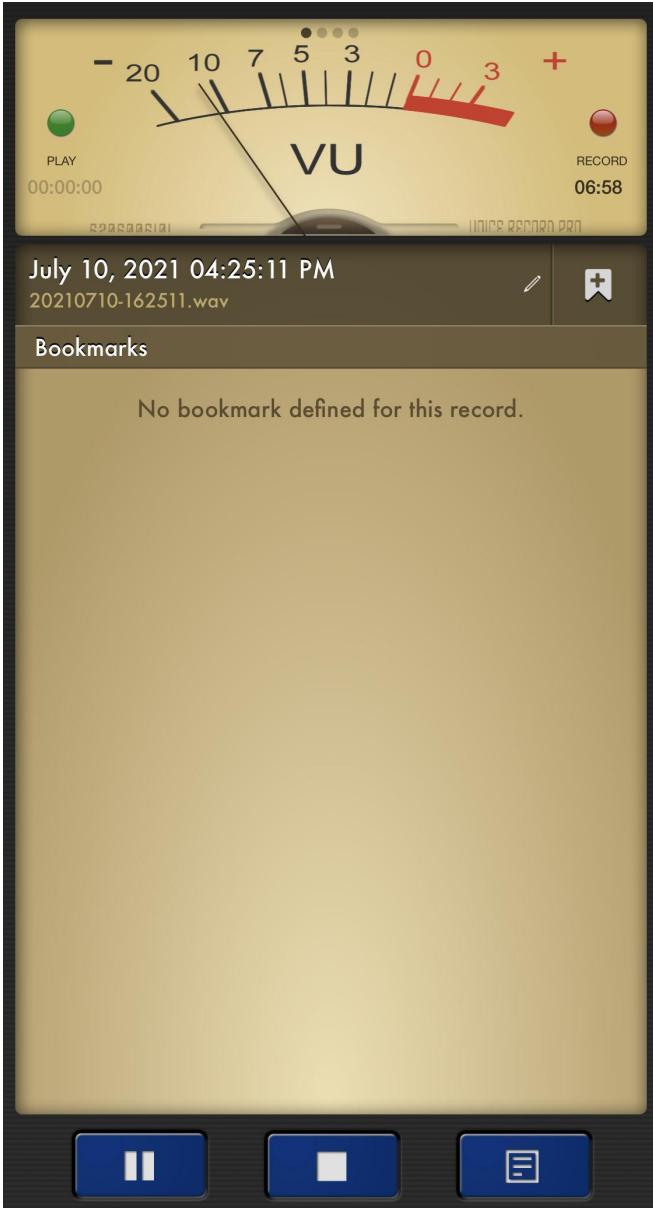
You Can Hear The Difference Between Hot and Cold W...

Tom Scott

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

Experiment Overview





Voice Recorder



Sample Sound:

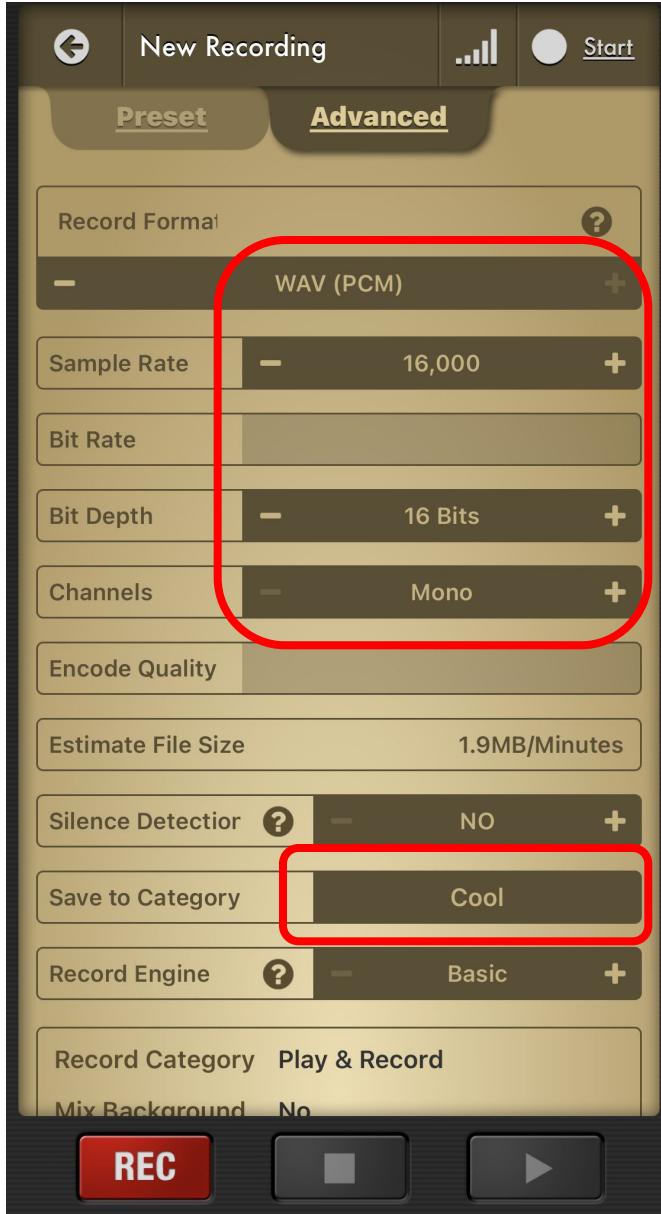
- 16KHz
- PCM – 16bits
- Mono

Classes:

- Hot
- Cool
- Noise



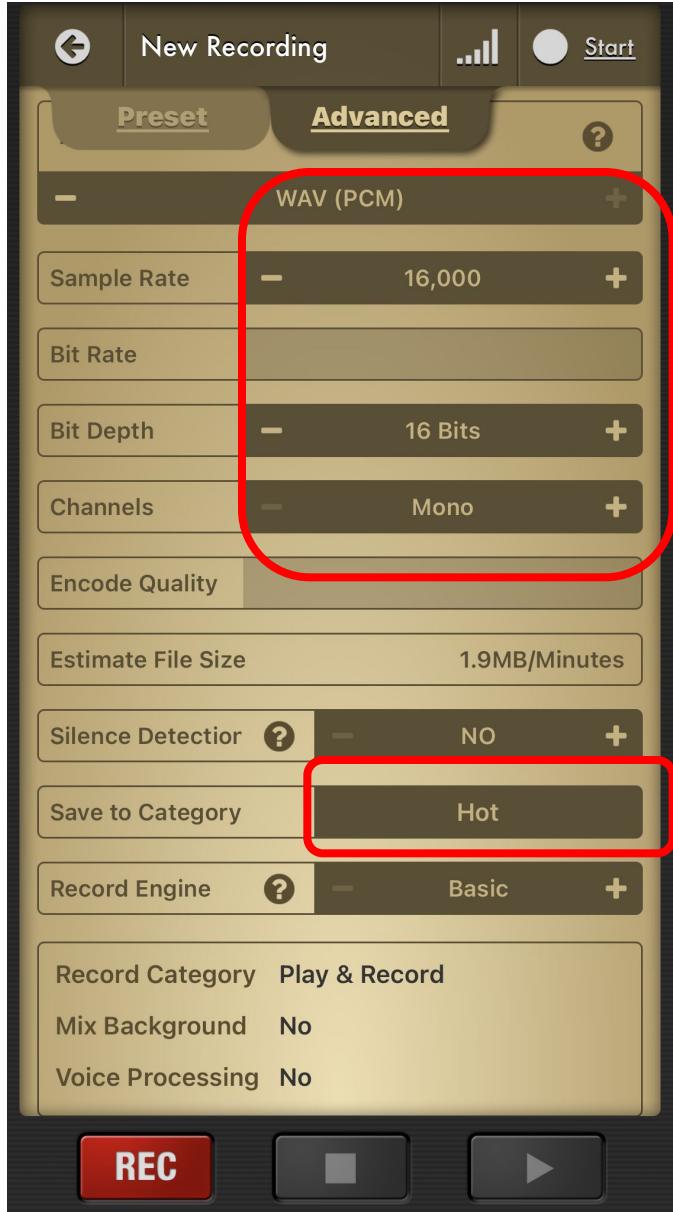
Ambient Temperature: 19°C



Class: Cool



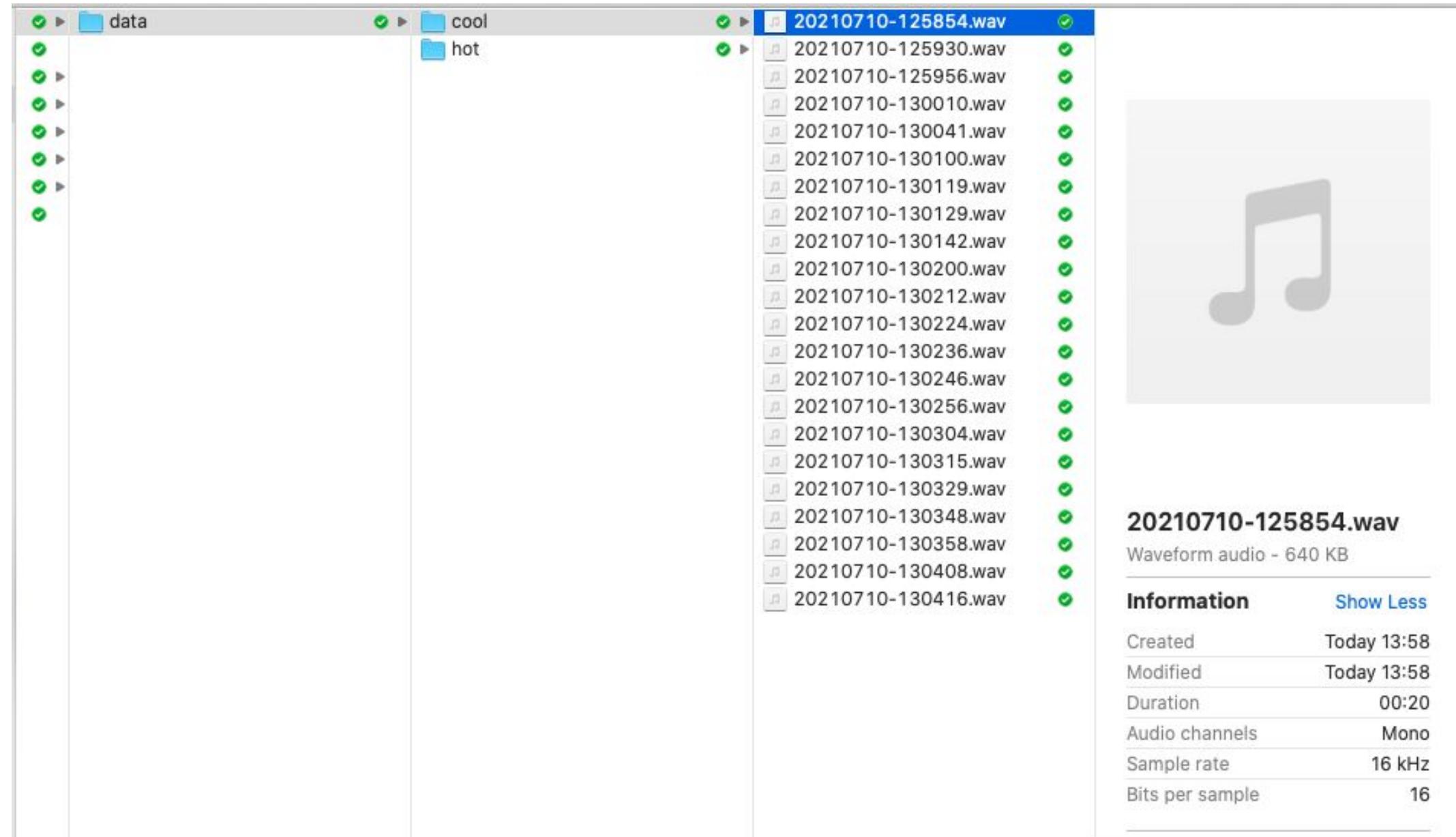
Cool Water Temperature: 11°C



Class: Hot



Hot Water Temperature: 61°C



The screenshot shows a file explorer window with three main directories: 'data', 'cool', and 'hot'. The 'cool' directory is expanded, revealing a list of 24 wav files, each with a green checkmark icon. The files are named sequentially from '20210710-125854.wav' to '20210710-130416.wav'. To the right of the file list, there is a detailed view for the selected file, '20210710-125854.wav'. This view includes a large gray musical note icon, the file name, its type ('Waveform audio - 640 KB'), and an 'Information' table with the following details:

Information	Show Less
Created	Today 13:58
Modified	Today 13:58
Duration	00:20
Audio channels	Mono
Sample rate	16 kHz
Bits per sample	16

Data captured using app Voice Recorder and uploaded to Computer



Upload existing data

You can upload existing data to your project in the [Data Acquisition Format](#) (CBOR, JSON, CSV), or as WAV, JPG or PNG files.

Select files

No file chosen

Upload into category

- Automatically split between training and testing ?
- Training
- Testing

Label

- Infer from filename ?
- Enter label:

hot

Upload output

Uploading 14 files...

```
[ 1/14] Uploading 20210710-130535.wav OK
[ 2/14] Uploading 20210710-130603.wav OK
[ 3/14] Uploading 20210710-130544.wav OK
[ 4/14] Uploading 20210710-130553.wav OK
[ 5/14] Uploading 20210710-130738.wav OK
[ 6/14] Uploading 20210710-130718.wav OK
[ 7/14] Uploading 20210710-130649.wav OK
[ 8/14] Uploading 20210710-130700.wav OK
[ 9/14] Uploading 20210710-130630.wav OK
[10/14] Uploading 20210710-130621.wav OK
[11/14] Uploading 20210710-130709.wav OK
[12/14] Uploading 20210710-130611.wav OK
[13/14] Uploading 20210710-130728.wav OK
[14/14] Uploading 20210710-130639.wav OK
```

Done. Files uploaded successful: 14. Files that failed to upload: 0.

Job completed

Raw Data uploaded to Edge Impulse Studio as .wav

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

DATA COLLECTED

2m 49s



LABEL

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 cleaned as split in 1 second samples

Split sample '20210710-130621.wav.2a5e0r33'

x

+ Add Segment**Remove segment**

is): 1000

Apply15000
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



⚡ An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Time series data

Axes
audio

Window size 1000 ms.

Window increase 500 ms.

Zero-pad data

Audio (MFE)

Name

Input axes audio

Neural Network (Keras)

Name

Input features MFE

Output features
3 (cool, hot, noise)

Output features

3 (cool, hot, noise)

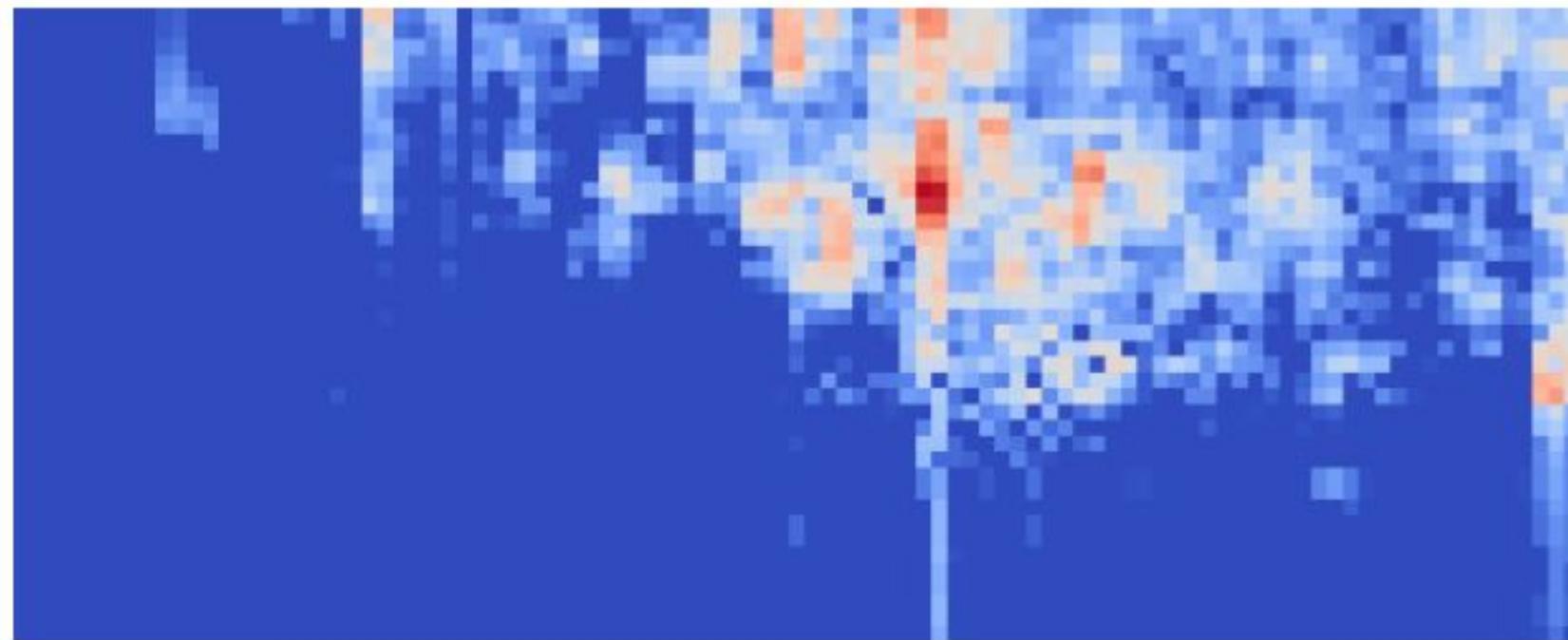
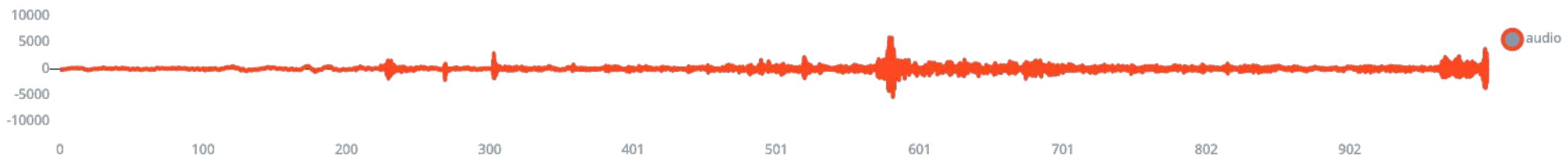
Save Impulse

Add a processing block

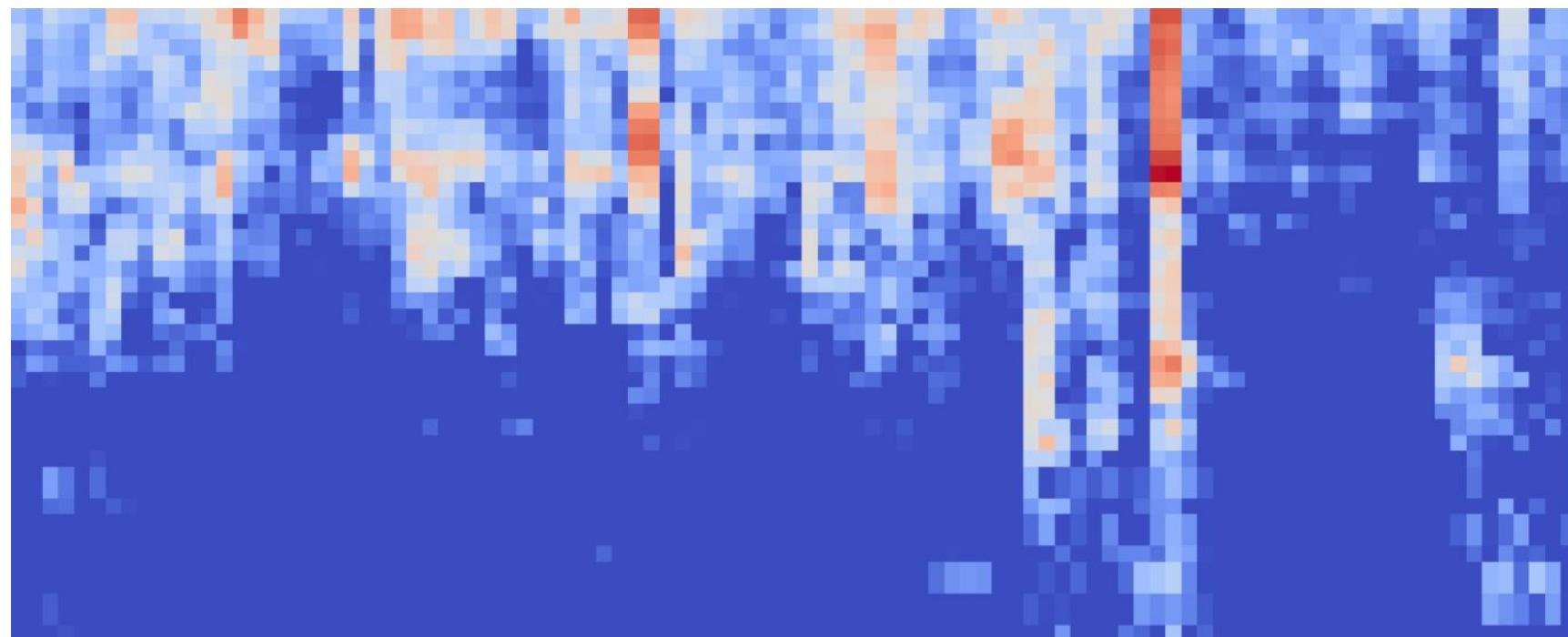
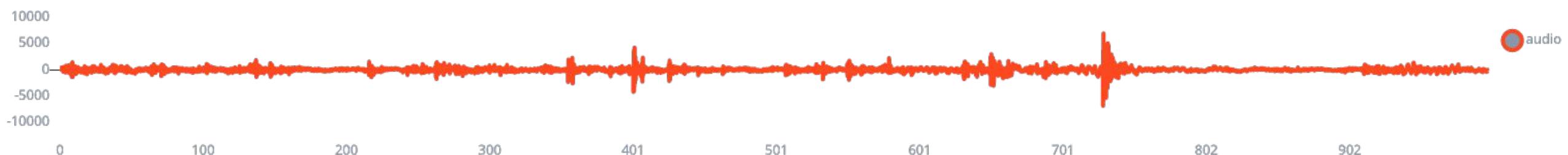
Add a learning block

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

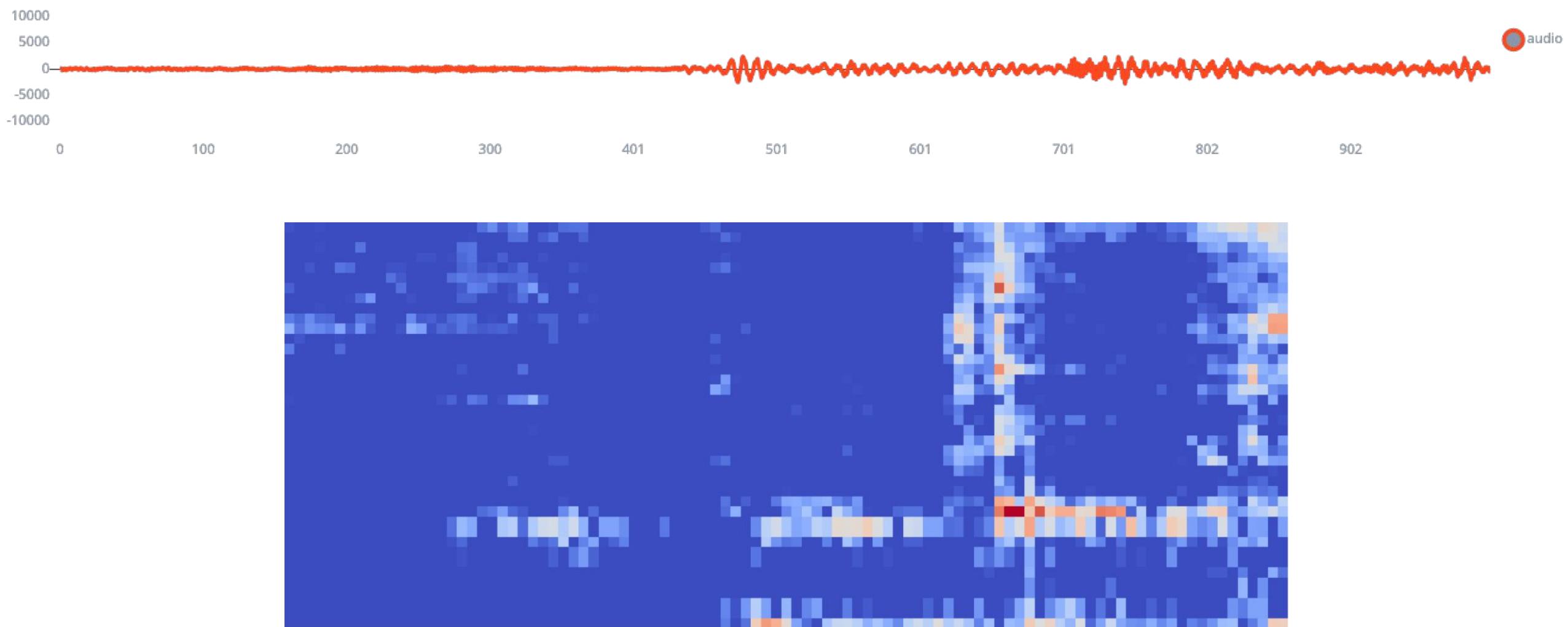
Cool Water 1 second sample



Hot Water 1 second sample



Noise 1 second sample



Parameters [Generate features](#)

Training set

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

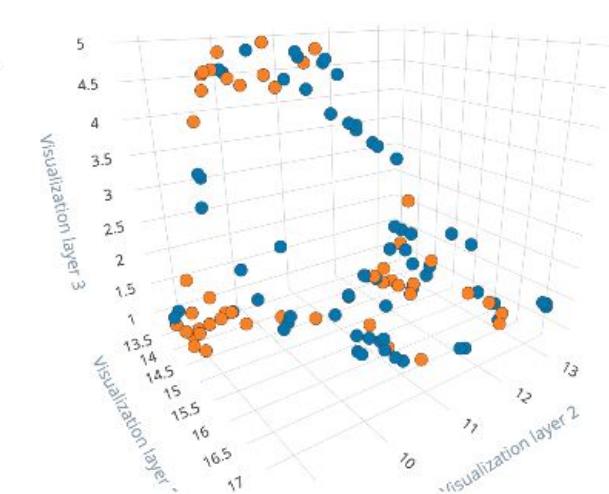
Generate feature

Feature explorer (155 samples)

?

Y Axis

Z Axis



20210710-125930.wav.2a5dy05j.s

Label: co

[View sample](#)



On-device performance

PROCESSING TIME
250 ms.

PEAK RAM USAGE
25 KB

#1 ▾ Click to set a description for this version

Neural Network settings

Training settings

Number of training cycles

Learning rate

Minimum confidence rating

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

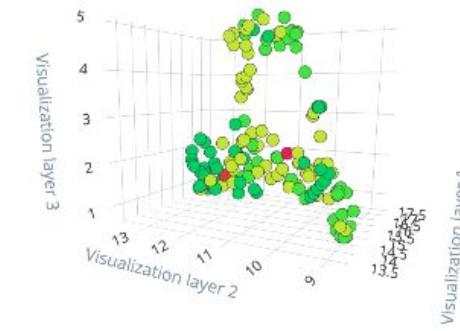
Model version:

Last training performance (validation set)

	COOL	HOT	NOISE
ACCURACY	93.5%	7.1%	0%
LOSS	0.13	83.3%	0%
CONFUSION MATRIX			
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)

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



On-device performance

	INFERENCING TIME	PEAK RAM USAGE	FLASH USAGE
	17 ms.		10.9K
			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.wav	hot	1s	100%	1 hot	...
20210710-130639.wav	hot	1s	100%	1 hot	...
20210710-130553.wav	hot	1s	100%	1 hot	...
20210710-130535.wav	hot	1s	100%	1 hot	...
20210710-130535.wav	hot	1s	100%	1 hot	...
20210710-130224.wav	cool	1s	0%	1 noise	...
20210710-130304.wav	cool	1s	100%	1 cool	...
20210710-130236.wav	cool	1s	100%	1 cool	...
20210710-130256.wav	cool	1s	100%	1 cool	...
20210710-130224.wav	cool	1s	100%	1 cool	...
20210710-130142.wav	cool	1s	0%	1 noise	...
20210710-130100.wav	noise	1s	100%	1 noise	...
20210710-130041.wav	noise	1s	100%	1 noise	...
20210710-125854.wav	noise	1s	100%	1 noise	...
20210710-125854.wav	noise	1s	100%	1 noise	...

Model testing output

Model testing results

ACCURACY

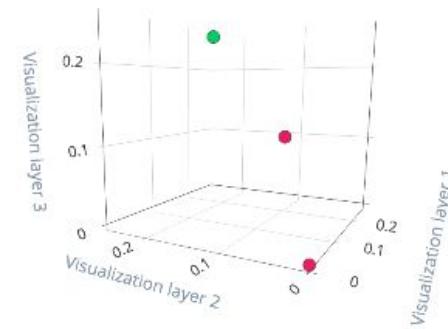
86.67%

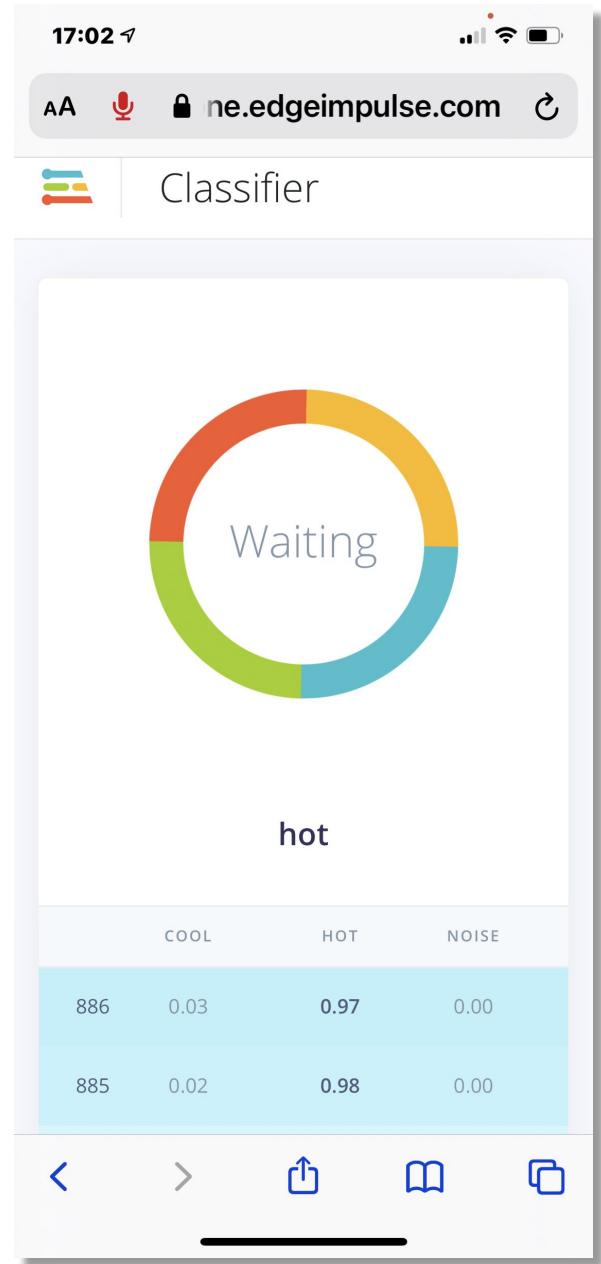
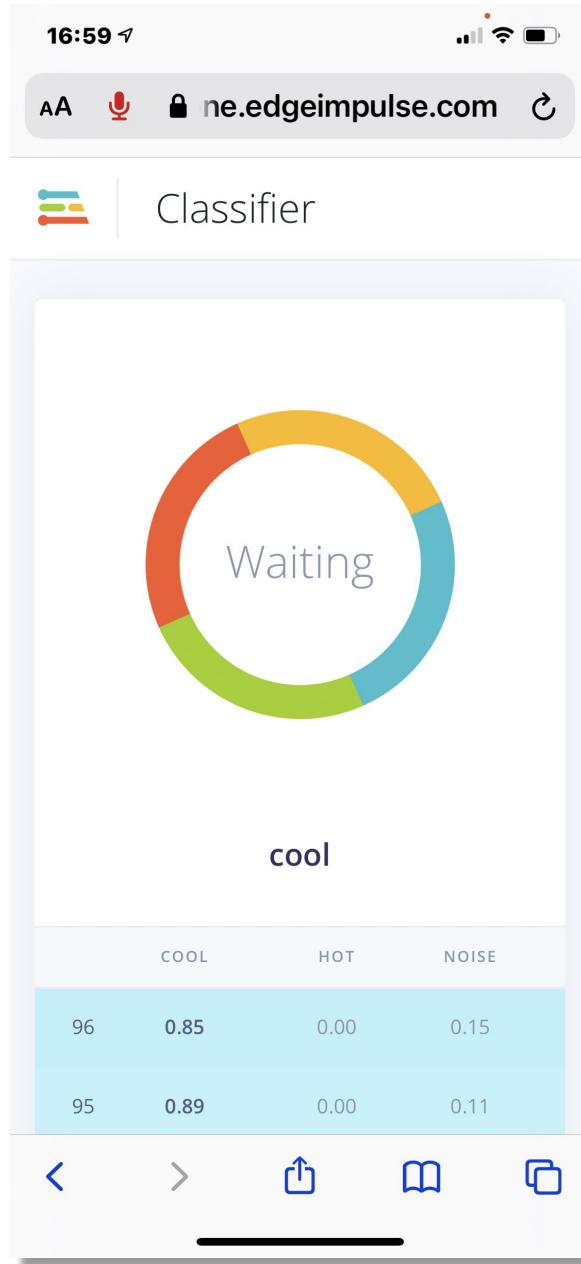
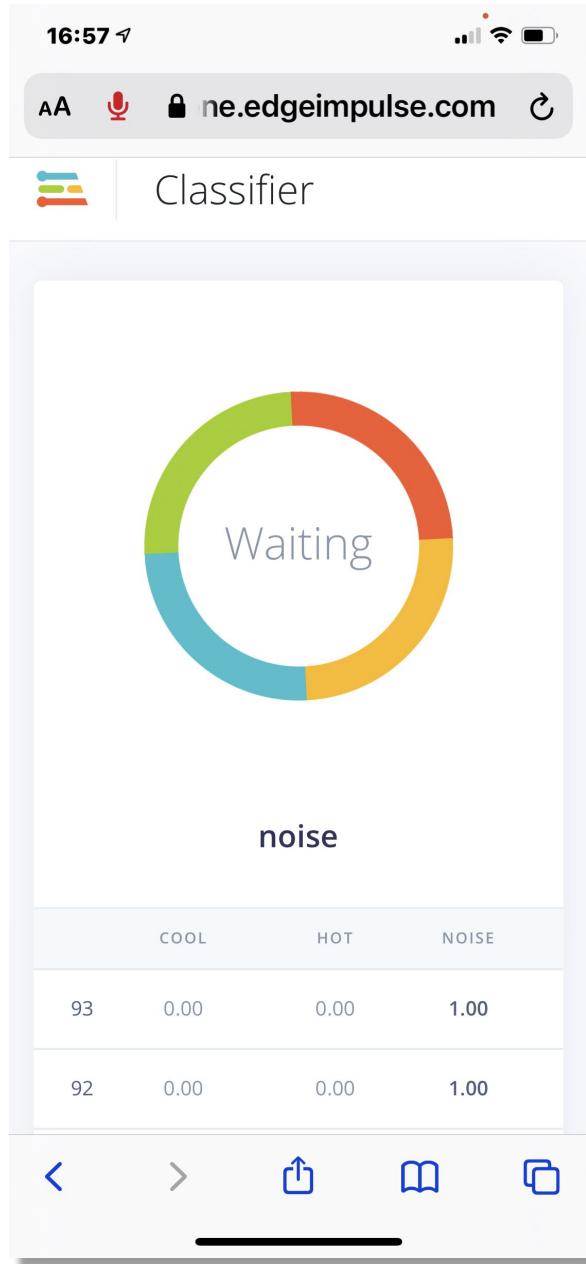


	COOL	HOT	NOISE	UNCERTAIN
COOL	66.7%	0%	33.3%	0%
HOT	0%	100%	0%	0%
NOISE	0%	0%	100%	0%

Feature explorer

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





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)	RAM USAGE	LATENCY	CONFUSION MATRIX			
	10.9K	17 ms	66.7	0	33.3	0
Currently selected	FLASH USAGE	ACCURACY	0	100	0	0
	31.4K	86.67%	0	0	100	0
Unoptimized (float32)	RAM USAGE	LATENCY	CONFUSION MATRIX			
	33.9K	78 ms	66.7	0	33.3	0
Click to select	FLASH USAGE	ACCURACY	0	100	0	0
	38.0K	86.67%	0	0	100	0

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

Arduino File Edit Sketch Tools Help

nano_ble33_sense_microphone_continuous_LED | Arduino 1.8.15

/dev/cu.usbmodem144301

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

640 x 480 : 60 fps

```

88 /**
89 * @brief      Special Postprocess function for RGB LEDs
90 */
91
92 void turn_off_leds(){
93     digitalWrite(LED_R, HIGH);
94     digitalWrite(LED_G, HIGH);
95     digitalWrite(LED_B, HIGH);
96 }
97
98 /*
99 * cool: [0] ==> Blue ON
100 * hot: [1] ==> Red ON
101 * noise: [2] ==> ALL OFF
102 */
103
104 void turn_on_leds(int pred_index) {
105     switch (pred_index)
106     {
107     case 0:
108         turn_off_leds();
109         digitalWrite(LED_B, LOW);
110         break;
111
112     case 1:
113         turn_off_leds();
114         digitalWrite(LED_R, LOW);
115         break;
116
117     case 2:
118         turn_off_leds();
119         break;
120     }
121 }
122 
```

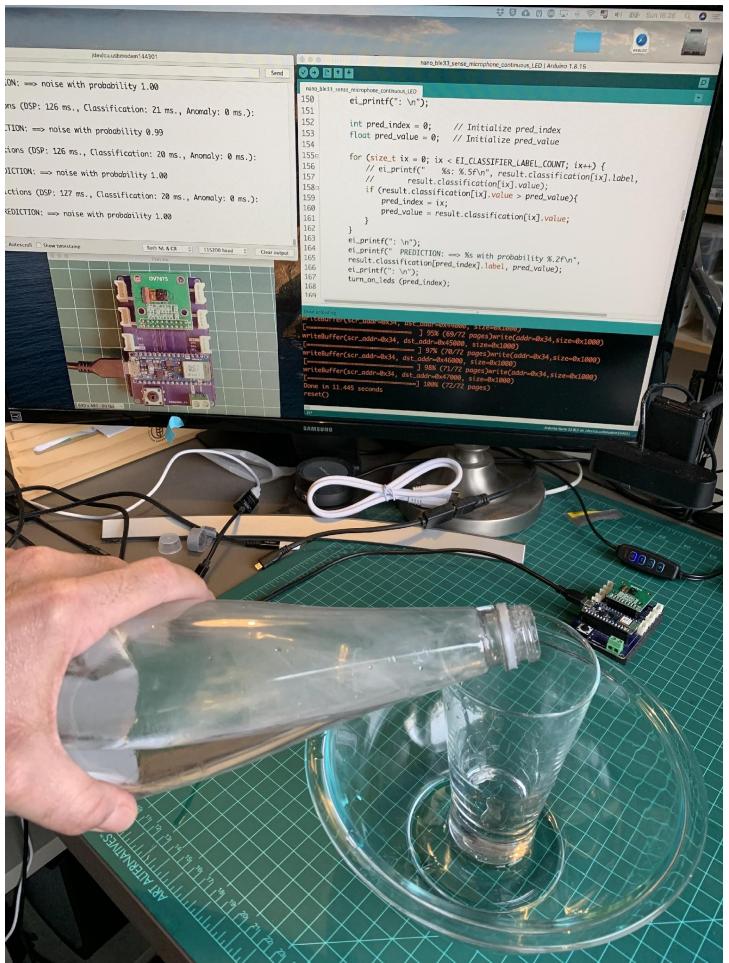
Done uploading.

```

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)
[===== ] 97% (70/72 pages) write(addr=0x34, size=0x1000)

```

Arduino Nano 33 BLE on /dev/cu.usbmodem144301



The terminal window displays a continuous loop of predictions. The last prediction, "PREDICTION: ==> cool with probability 0.96", is highlighted with a red box. Below the terminal is a photograph of an Arduino Uno microcontroller. An OV7675 camera module is mounted on top of the Arduino. A blue arrow points from the terminal's prediction towards the camera module. A red box on the Arduino Uno highlights the digital pins used for communication.

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

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