

# Leveraging any Microcontrollers & Data Collection at Edge Impulse Studio

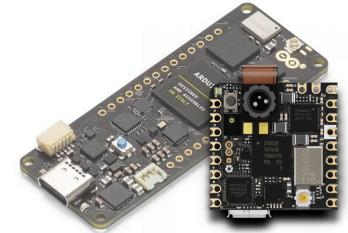
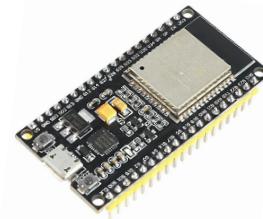
Prof. Marcelo José Rovai  
UNIFEI - Federal University of Itajubá, Brazil  
TinyML4D Academic Network Co-Chair



**UNIFEI**

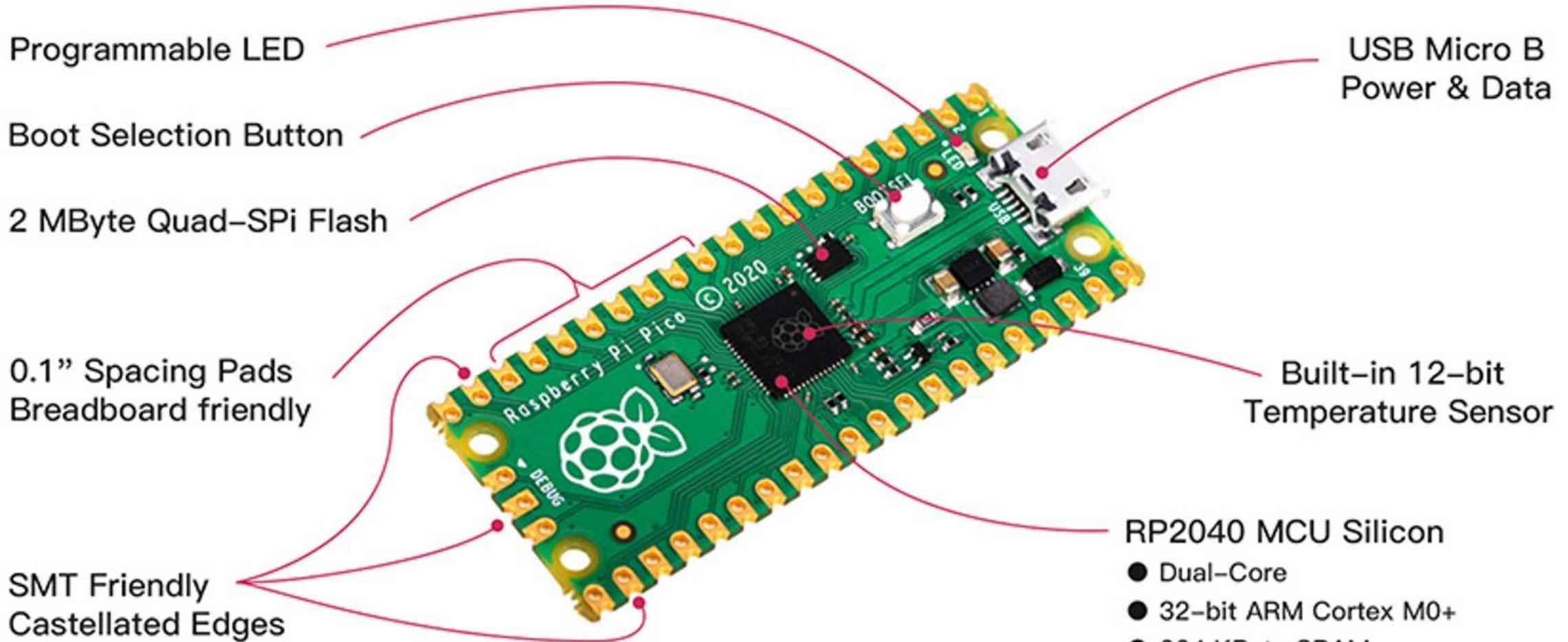
# Hardware

# Hardware



	Raspberry Pico (W)	Arduino Nano Sense	ESP 32	Seeed XIAO Sense / ESP32S3	Arduino Pro
<b>32Bits CPU</b>	Dual-core Arm Cortex-M0+	Arm Cortex-M4F	Xtensa LX6 Dual Core	Arm Cortex-M4F (BLE) Xtensa LX7 Dual Core	Dual Core Arm Cortex M7/M4
<b>CLOCK</b>	133MHz	64MHz	240MHz	64 / 240MHz	480/240MHz
<b>RAM</b>	264KB	256KB	520KB (part available)	256KB / 8MB	1MB
<b>ROM</b>	2MB	1MB	2MB	2MB / 8MB	2MB
<b>Radio</b>	(Yes for W)	BLE	BLE/WiFi	BLE / WiFi (ESP32S3)	BLE/WiFi
<b>Sensors</b>	No	Yes	No	Yes (Sense)	Yes (Nicla)
<b>Bat. Power Manag.</b>	No	No	No	Yes	Yes
<b>Price</b>	\$	\$\$\$	\$	\$\$	\$\$\$\$\$

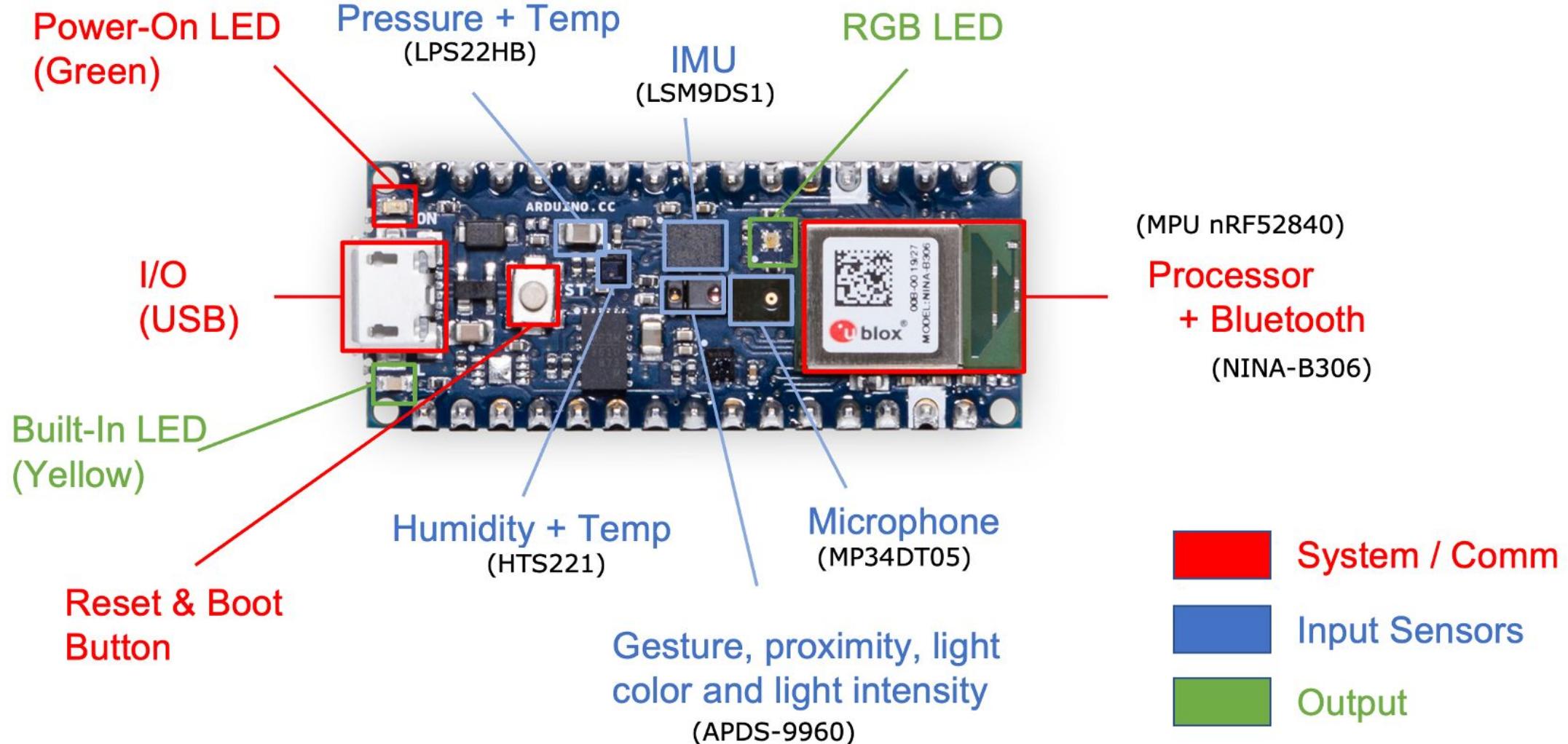
<https://media.digikey.com/Resources/Maker/the-original-guide-to-boards-2022.pdf>



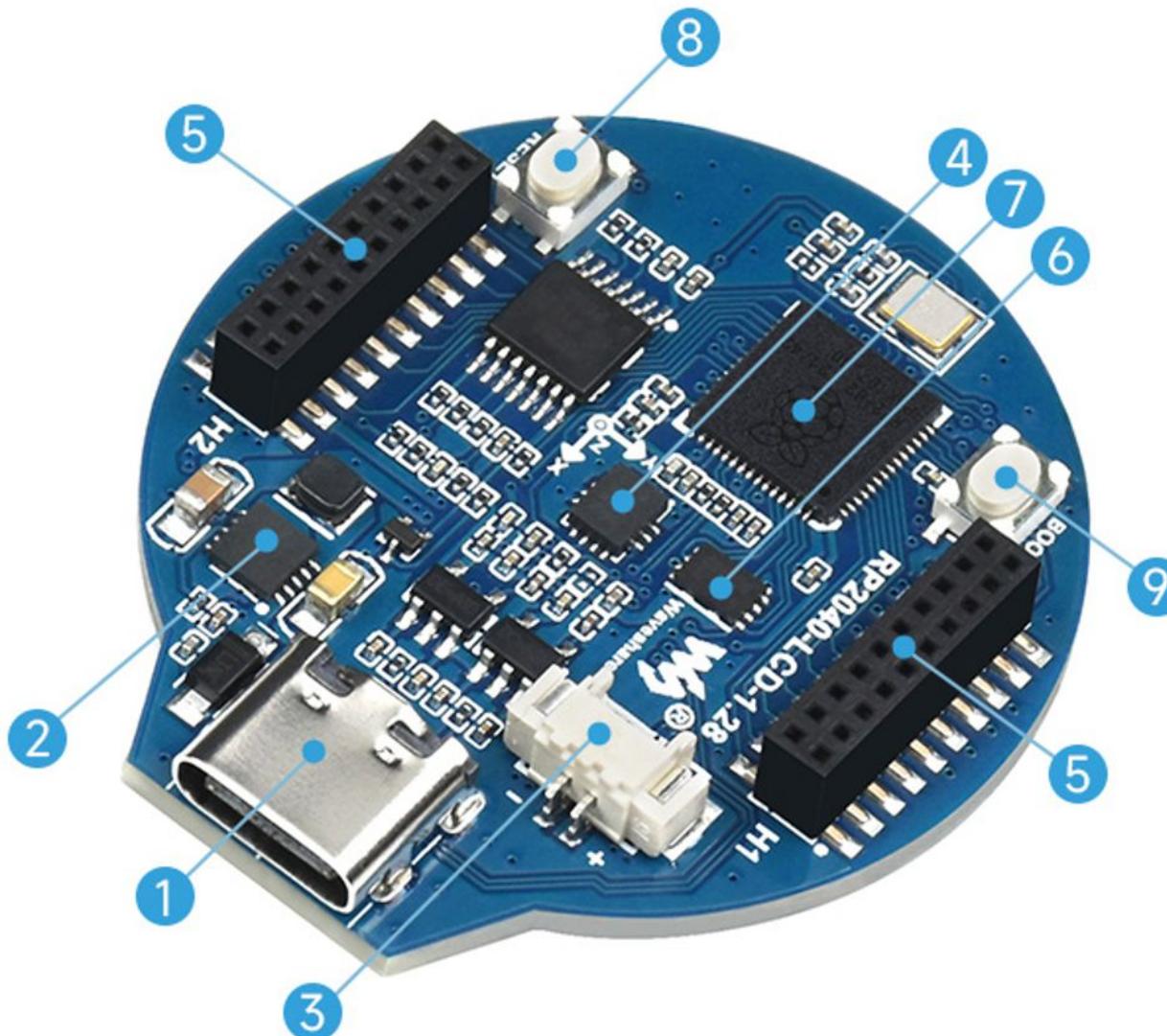
#### RP2040 MCU Silicon

- Dual-Core
- 32-bit ARM Cortex M0+
- 264 KByte SRAM
- Clock @48MHz, Max at 133MHz
- USB 1.1 Host and Device

# Nano 33 BLE Sense (Development board)



# RP2040 MCU Board, with LCD, accelerometer, and gyroscope Sensor



## 1. USB Type-C connector

USB 1.1 with device and host support

## 2. ETA6096

high efficiency Lithium battery recharge manager

## 3. Battery Header

MX1.25 header, for 3.7V Lithium battery, allows recharging the battery and powering the board at the same time

## 4. QMI8658C

IMU, includes a 3-axis gyroscope and a 3-axis accelerometer

## 5. 1.27mm pitch headers

Adapting all GPIO and Debug pins

## 6. W25Q16JVUXIQ

2MB NOR-Flash

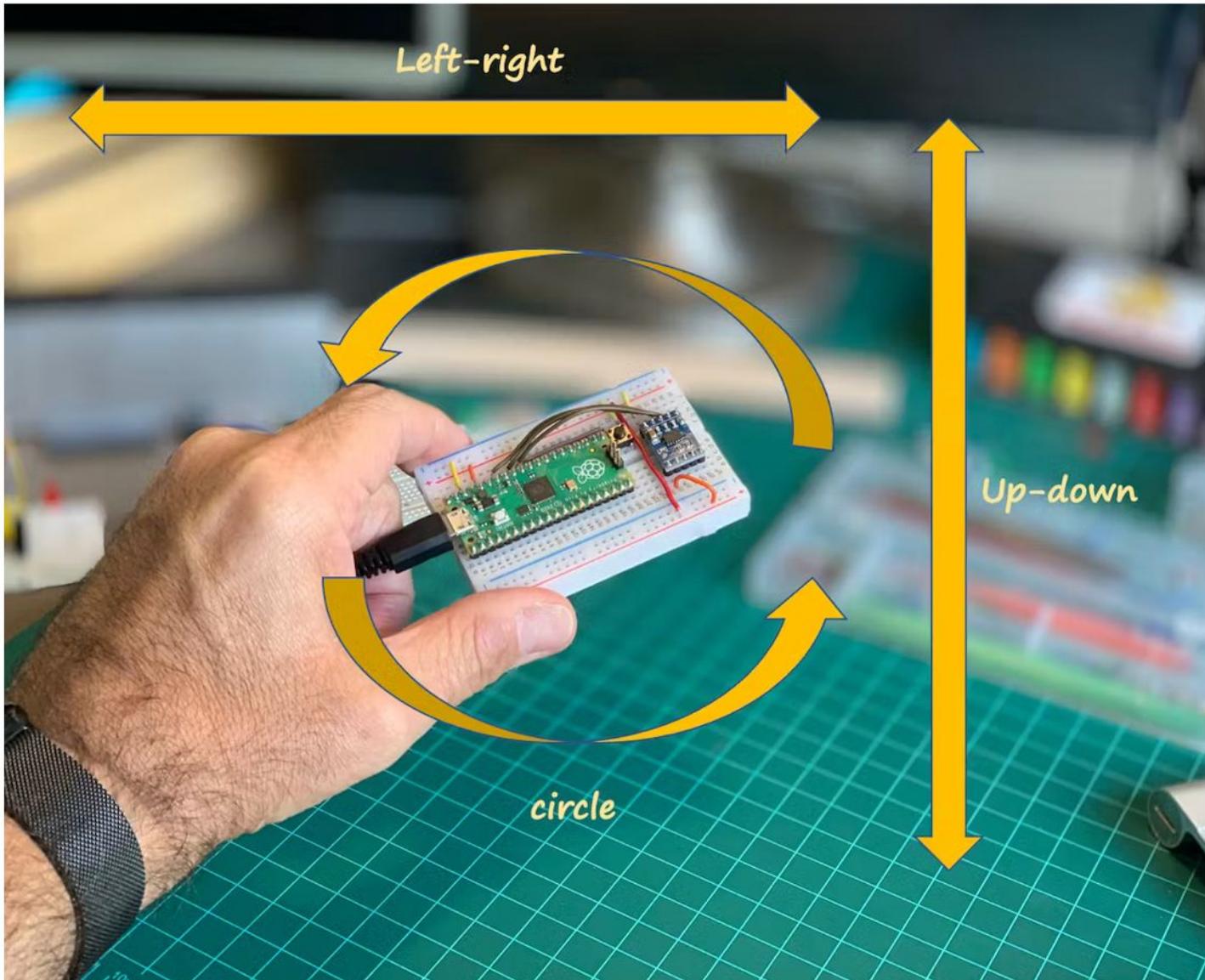
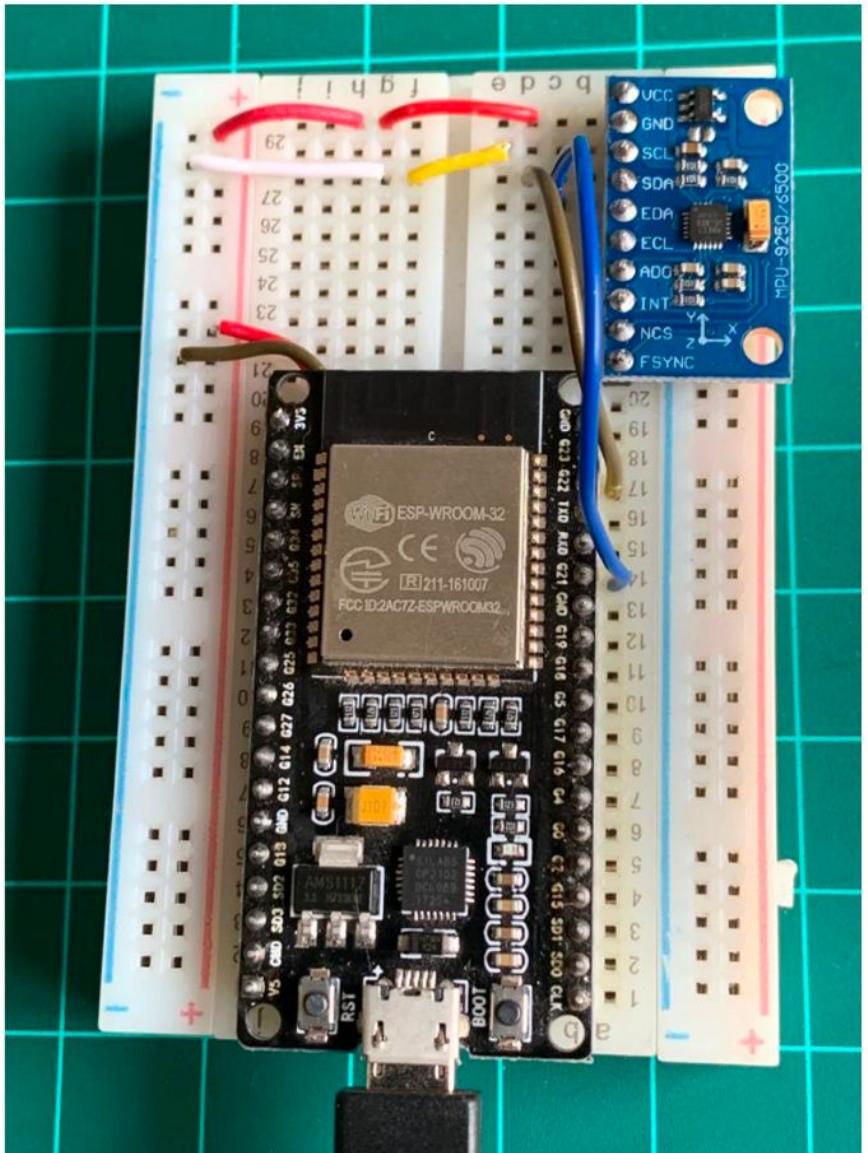
## 7. RP2040

Dual-core processor, up to 133MHz operating frequency

## 8. RESET Button

## 9. BOOT Button

press it when resetting to enter download mode



# Application Complexity vs. HW

Power



# EdgeML

## TinyML



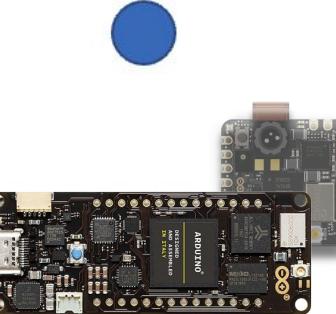
Anomaly Detection  
Sensor Classification  
20 KB



Rpi-Pico  
(Cortex-M0+)



KeyWord Spotting  
Audio Classification  
50 KB



Arduino Pro  
(Cortex-M7)

Image  
Classification  
250 KB+



## TinyML

Object Detection  
Complex Voice  
Processing  
1 MB+



RaspberryPi  
SmartPhone  
(Cortex-A)



Jetson Nano  
(Cortex-A + GPU)

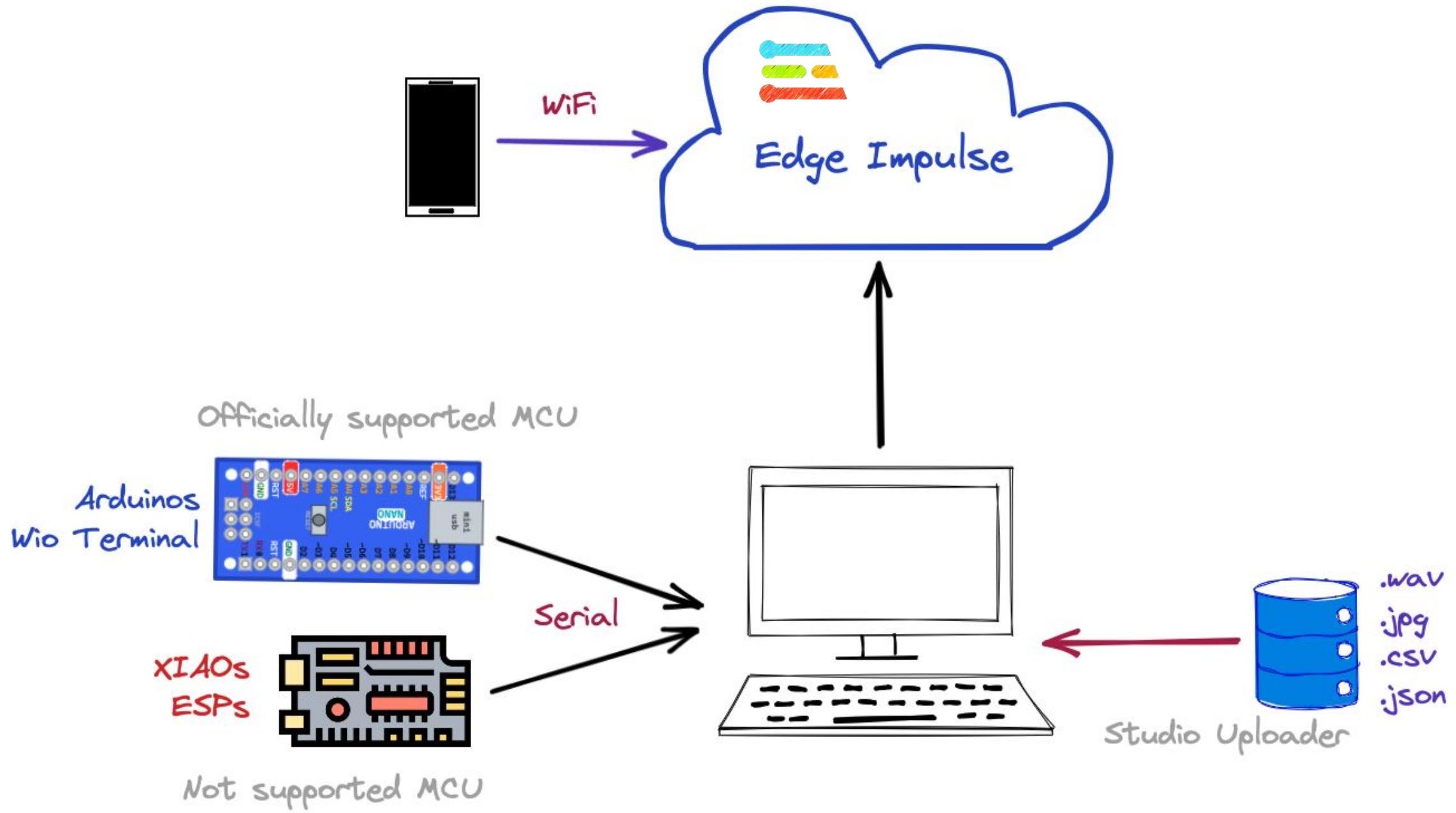
Application Complexity ↑

CPU Power / Memory →

Video  
Classification  
2 MB+

# EI Studio Data Ingestion

## Alternative methods



# 1. SmartPhone

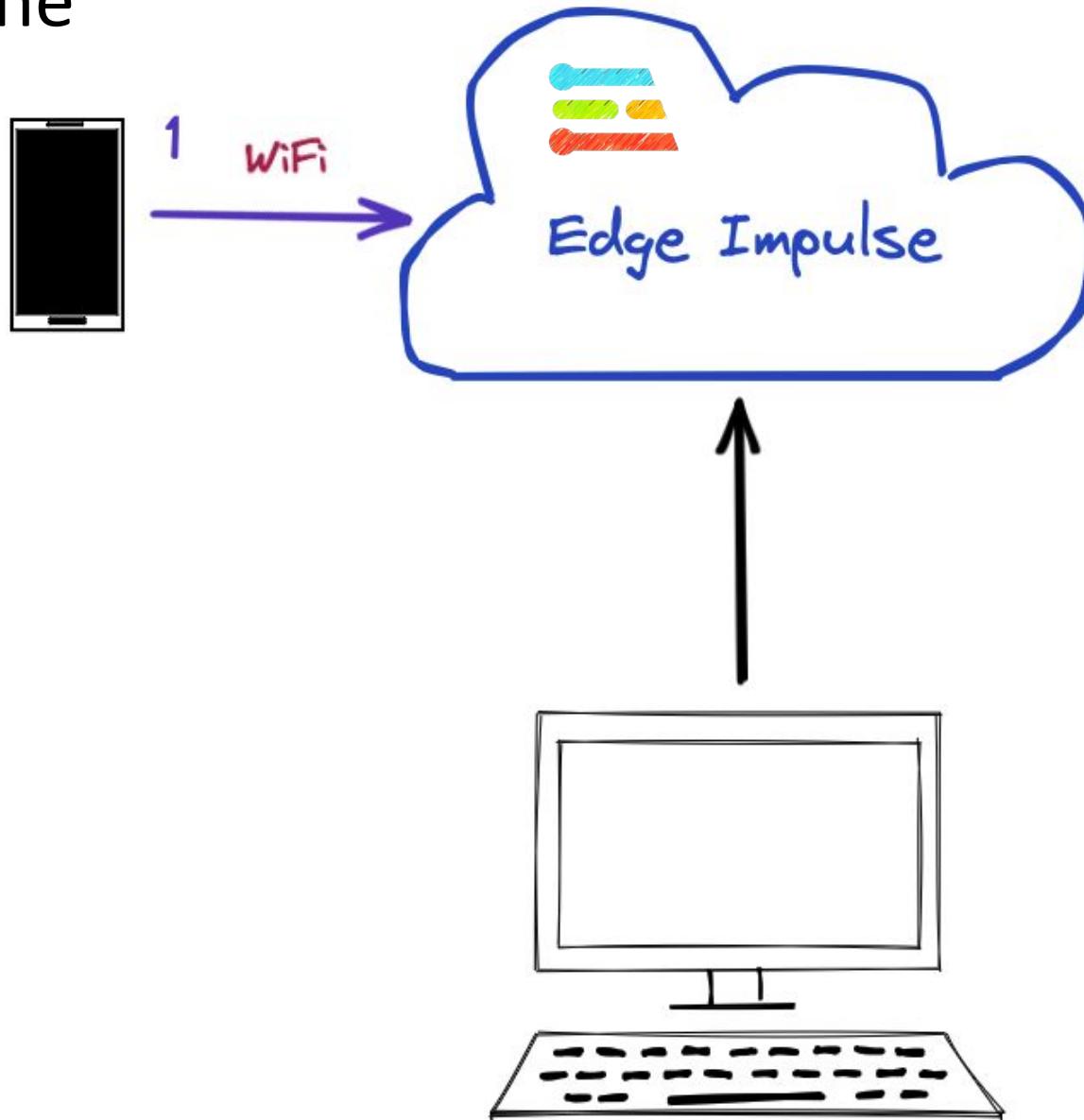
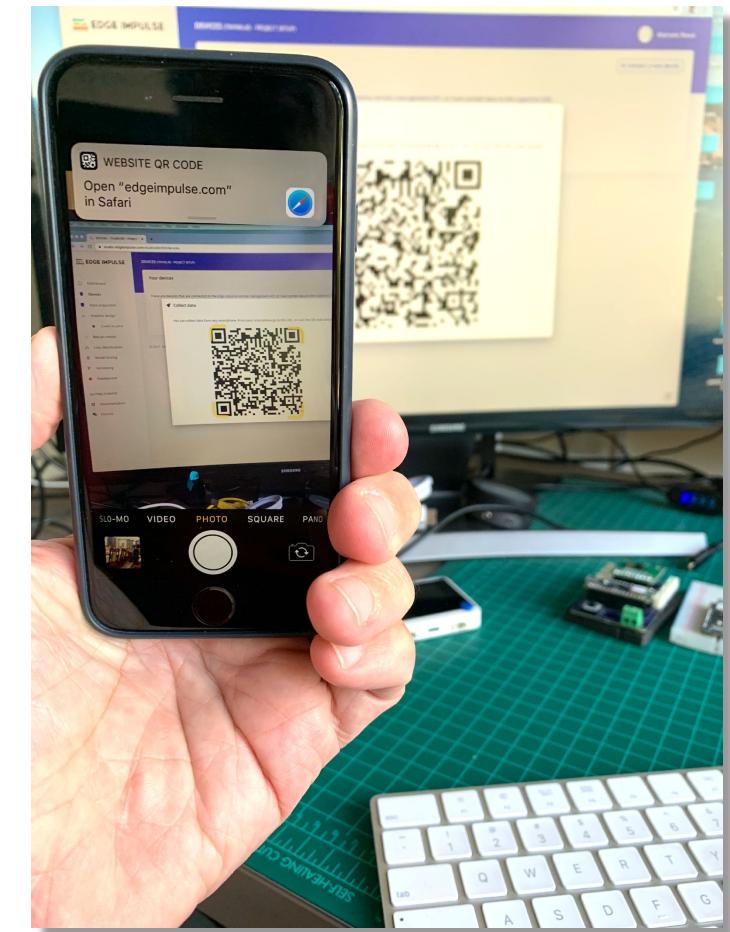
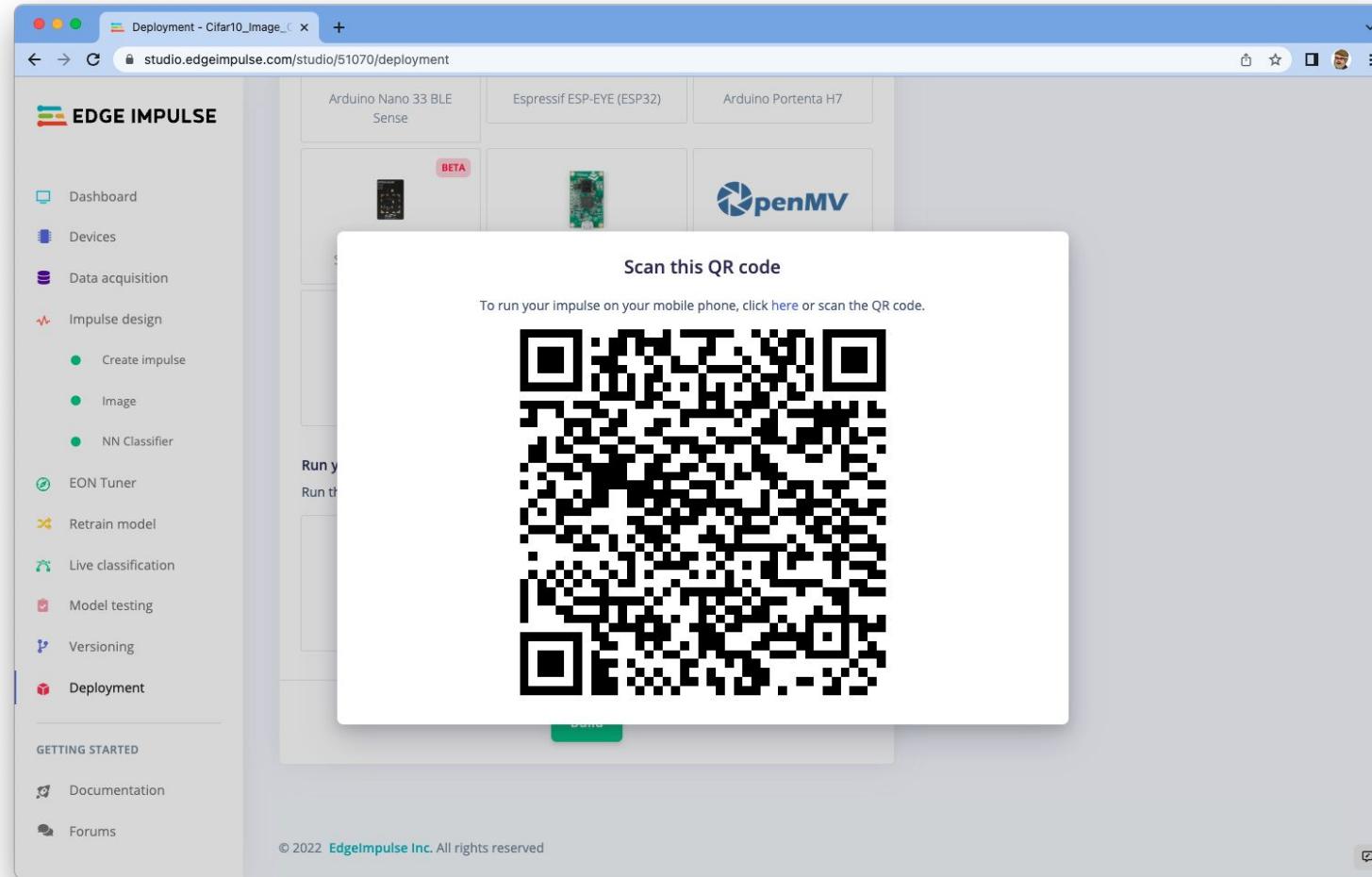


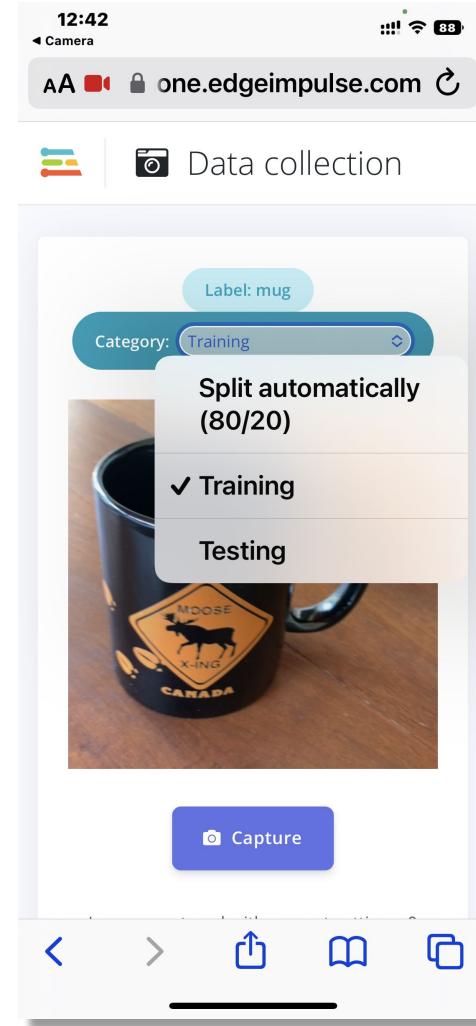
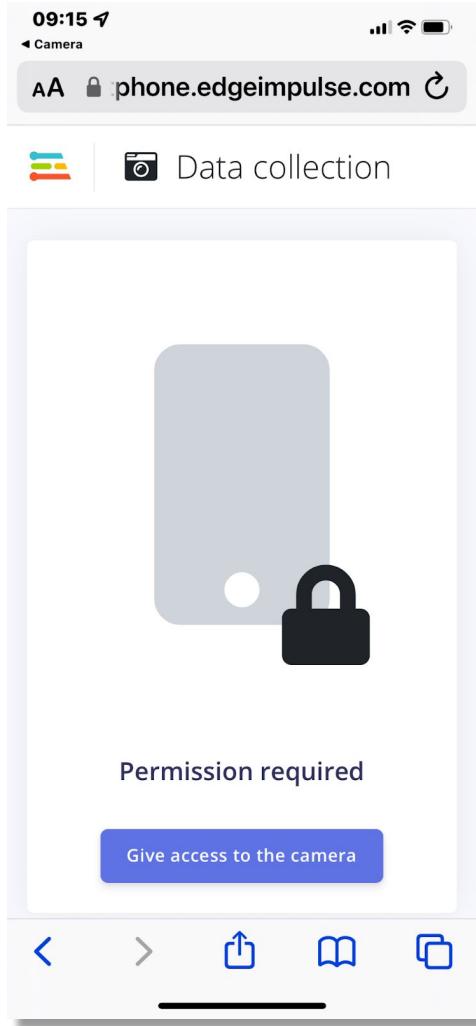
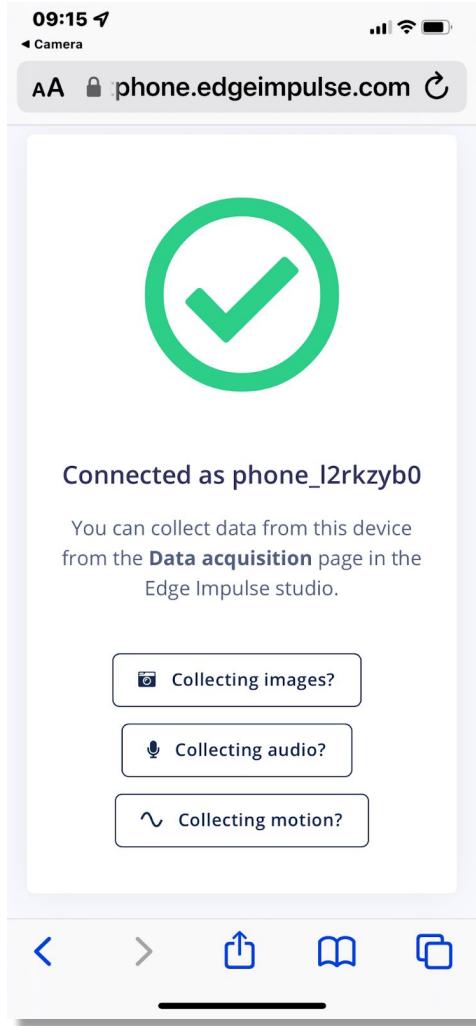
Image Classification using a **smartphone** and  
**Edge Impulse Studio**

Prof. Marcelo José Ribeiro  
UNIFEI - Federal University of Itajubá, Brazil  
TinM4D Academic Network Co-Chair

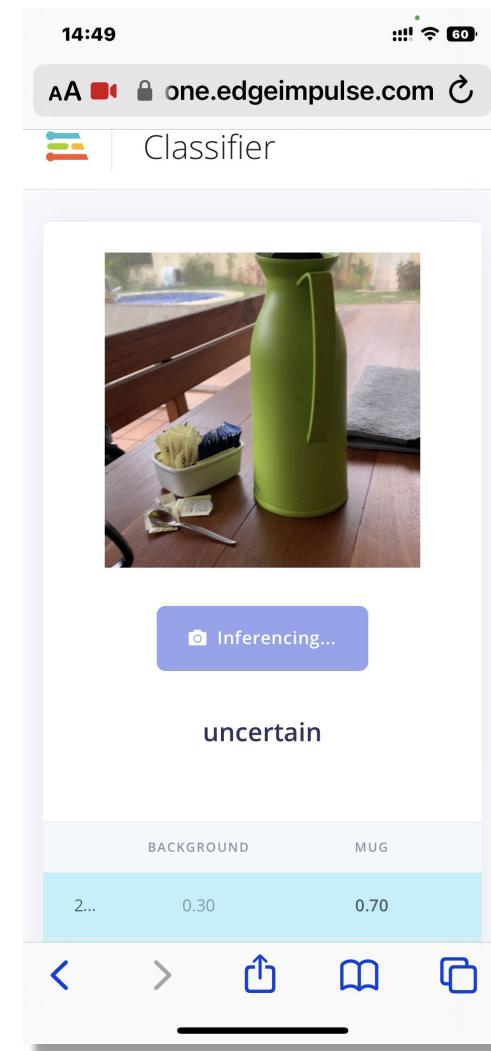
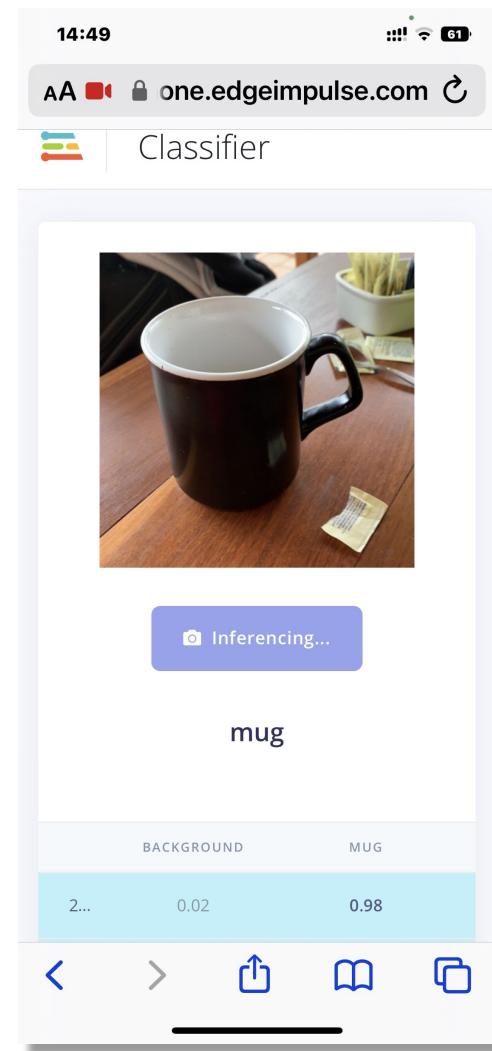
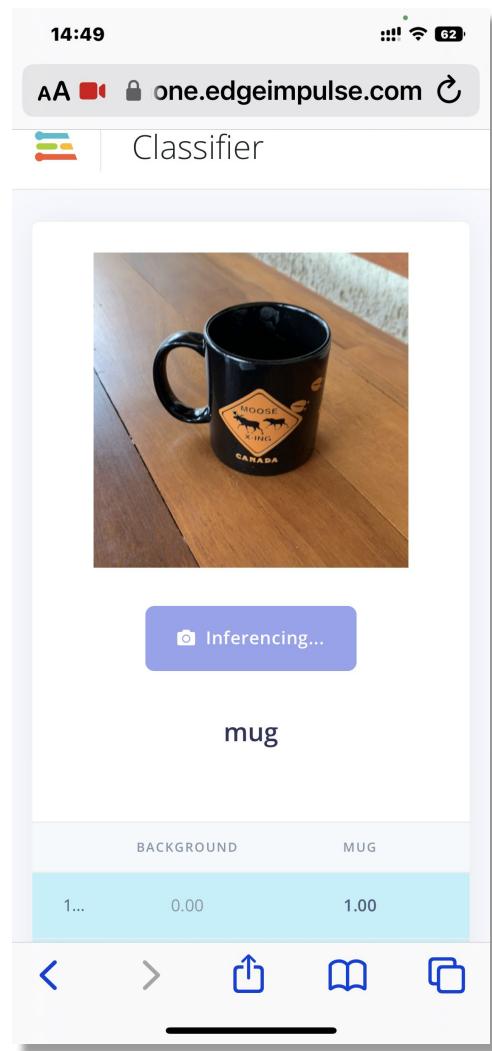
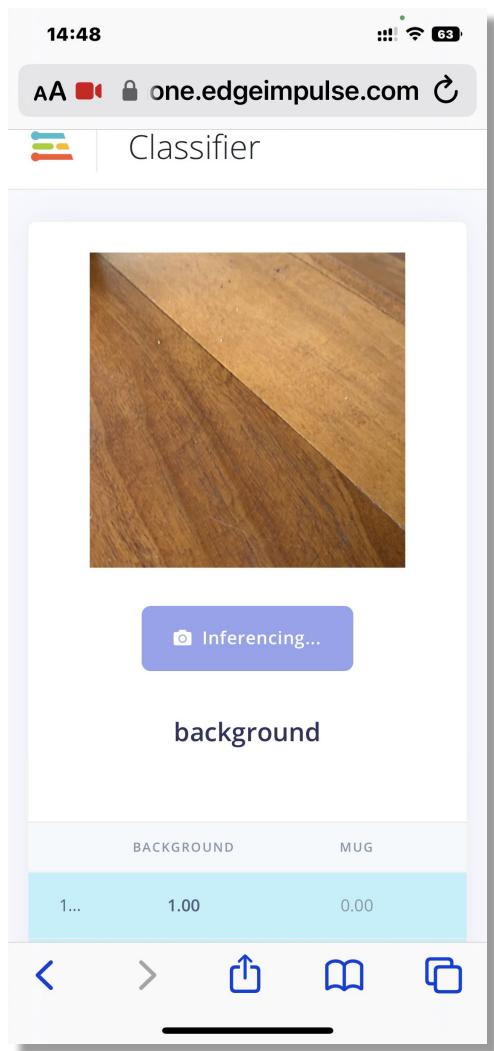
# 1. Data Ingestion using Smart Phone



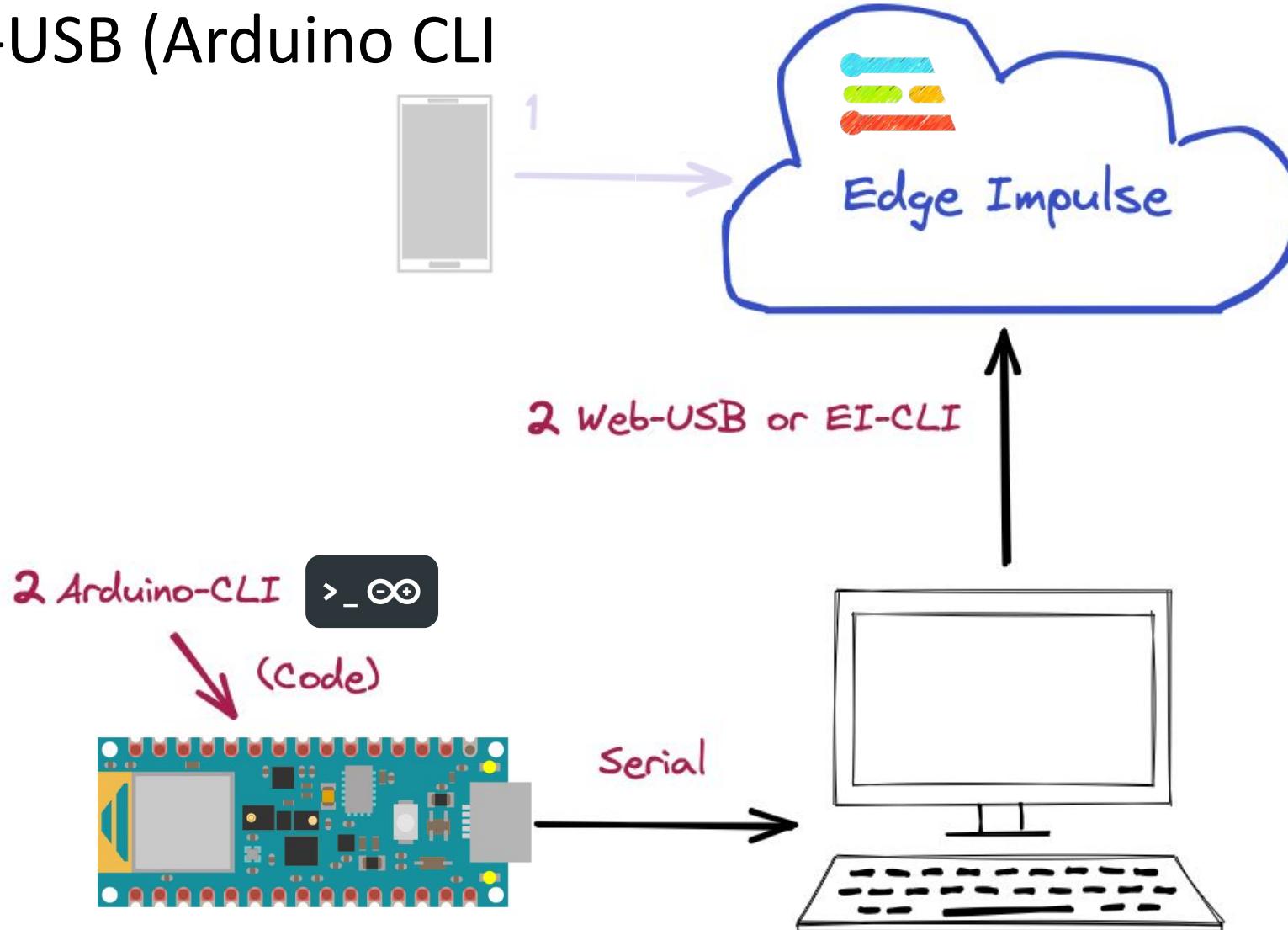
# 1. Data Capture and model Training



# 1. Off-Line Inference



## 2. Web-USB (Arduino CLI)



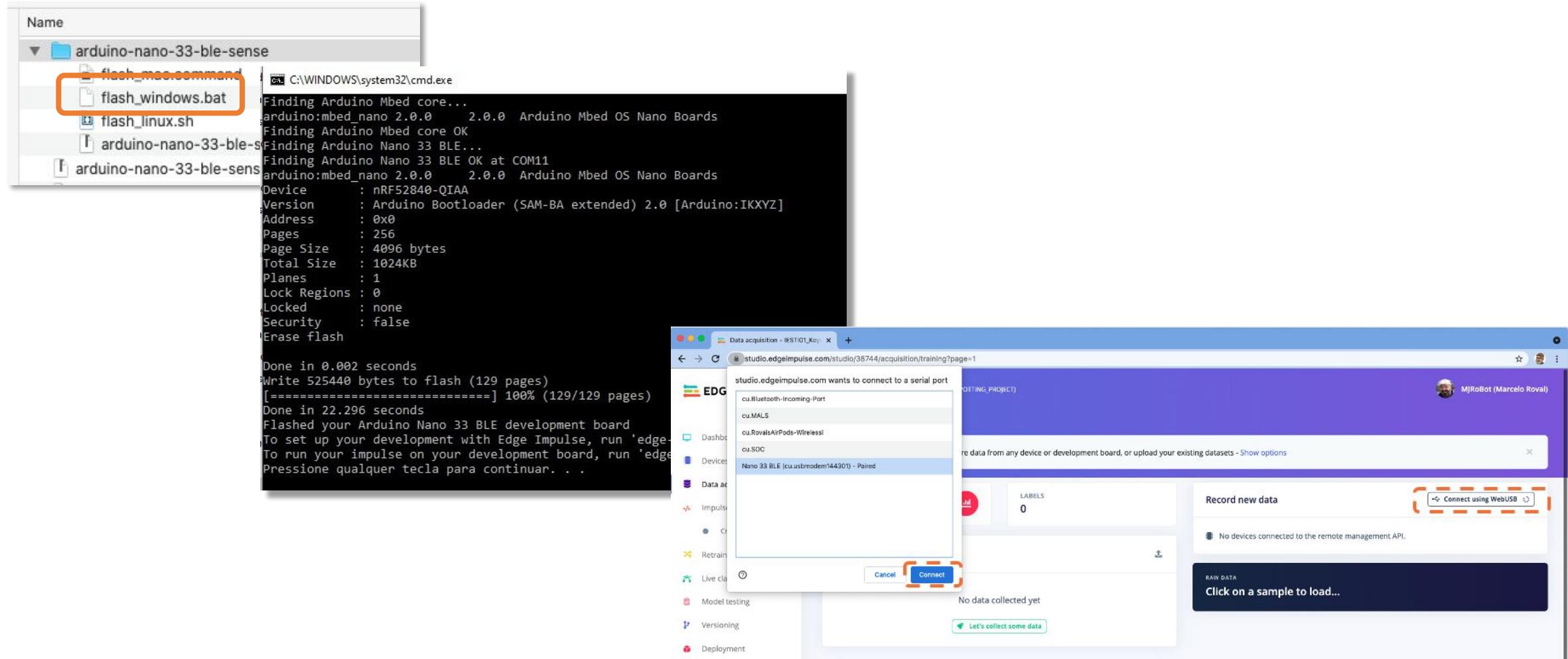
Issue: Limited MCU and sensors



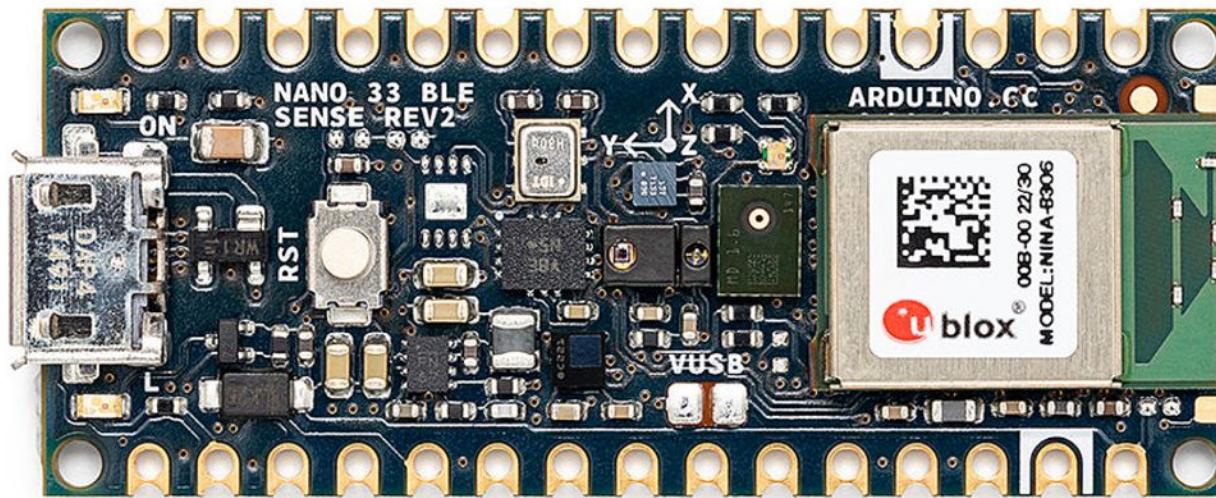
TinyML Arduino Kit  
Connection to Edge Impulse

Prof. Marcelo José Ribeiro  
UNIFEI - Federal University of Itajubá, Brazil  
TinyML4D Academic Network Co-Chair

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



# Arduino Nano 33 BLE Sense **Rev2**



- **IMU** - LSM9DS1 - 9 axis → BMI270 – 6 axis + BMM150 - 3 axis
- **Temperature and humidity sensor** - HTS221 → HS3003
- **Microphone** - MP34DT05 → MP34DT06JTR.

## Record new data

### Device ②

36:17:55:F9:70:F7

Label

lift

Sample length (ms.)

10000

Sensor

Inertial

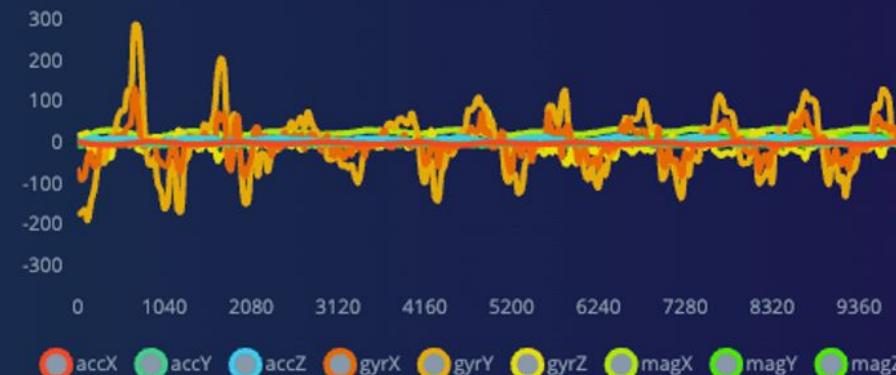
Frequency

62.5Hz

Start sampling

RAW DATA

lift.3soee1ar



### Data acquisition

#### Impulse design

- Create impulse
- Spectral Analysis
- Classifier
- Anomaly detection

#### EON Tuner

#### Retrain model

#### Live classification

#### Model testing

#### Versioning

#### Deployment

### GETTING STARTED

#### Documentation

#### Forums

## Time series data



### Input axes (9)

accX, accY, accZ, gyrX, gyrY, gyrZ, magX, magY, magZ

### Window size

2000 ms.

### Window increase

80 ms.

### Frequency (Hz)

62,5

### Zero-pad data



## Spectral Analysis

Name

Spectral Analysis

### Input axes (3)

accX

accY

accZ

gyrX

gyrY

gyrZ

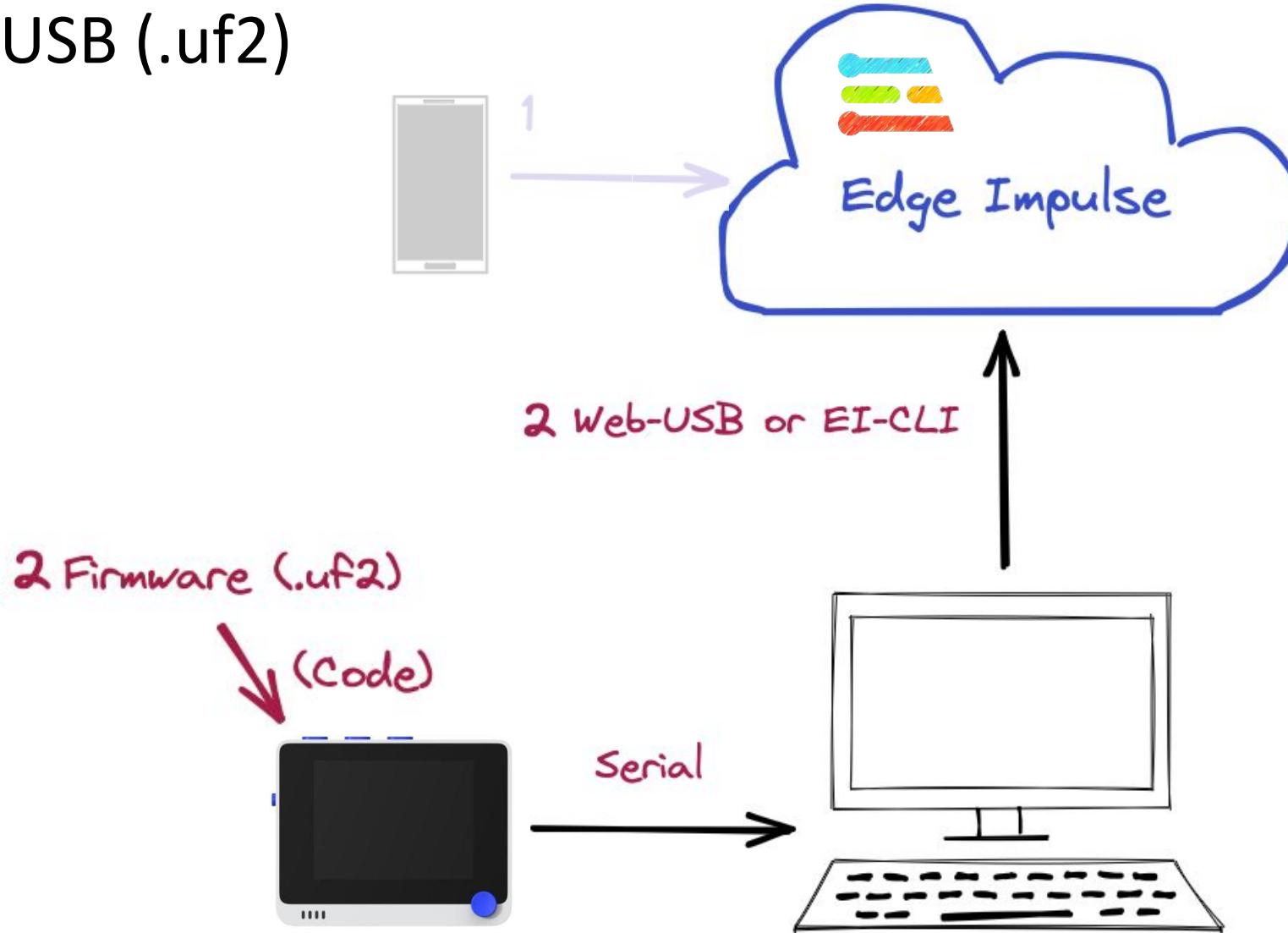
magX

magY

magZ

Add a processing

## 2. Web-USB (.uf2)



Issue: Limited MCU and sensors

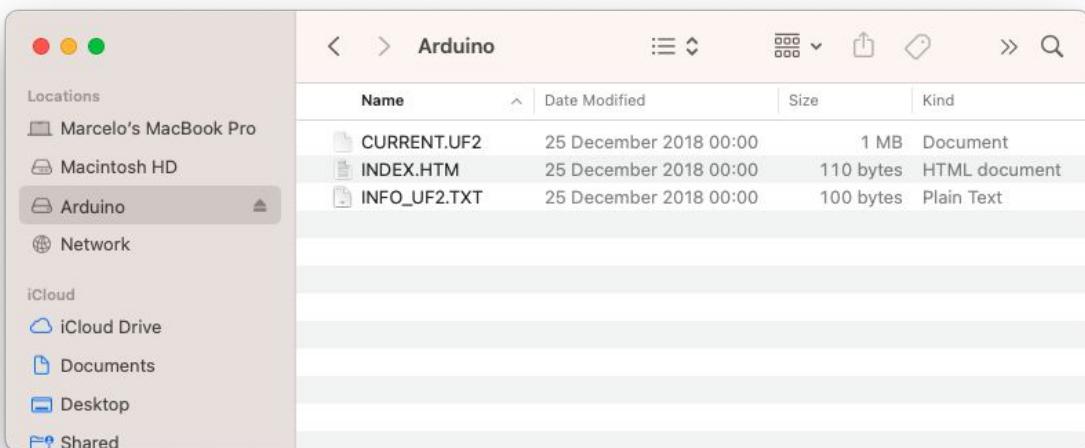


Wio Terminal  
Installation & Tests

Prof. Marcelo José Rovai  
UNIFEI - Federal University of Itajubá, Brazil  
TinyML4D Academics Network Co-Chair

## 2. (.uf2) Firmware installation

1. Connect Wio Terminal to your computer.
2. Entering the bootloader mode by sliding the power switch twice quickly.
3. An external drive named Arduino should appear in your PC.
4. Drag the the downloaded [Edge Impulse uf2 firmware files](#) to the Arduino drive. Now, Edge Impulse is loaded on Seeeduino Wio Terminal!



Releases Tags

Latest release

1.4.0  
AIWintermuteAI released this on Apr 15  
• Added built-in microphone support, recording in 16 kHz/16bit  
• Added internal light sensor support

This version is not final, microphone sampling uses DMA ADC while simultaneously writing samples to flash. Quality is slightly worse than when placing samples to RAM, perhaps because of slow FLASH writing speed.

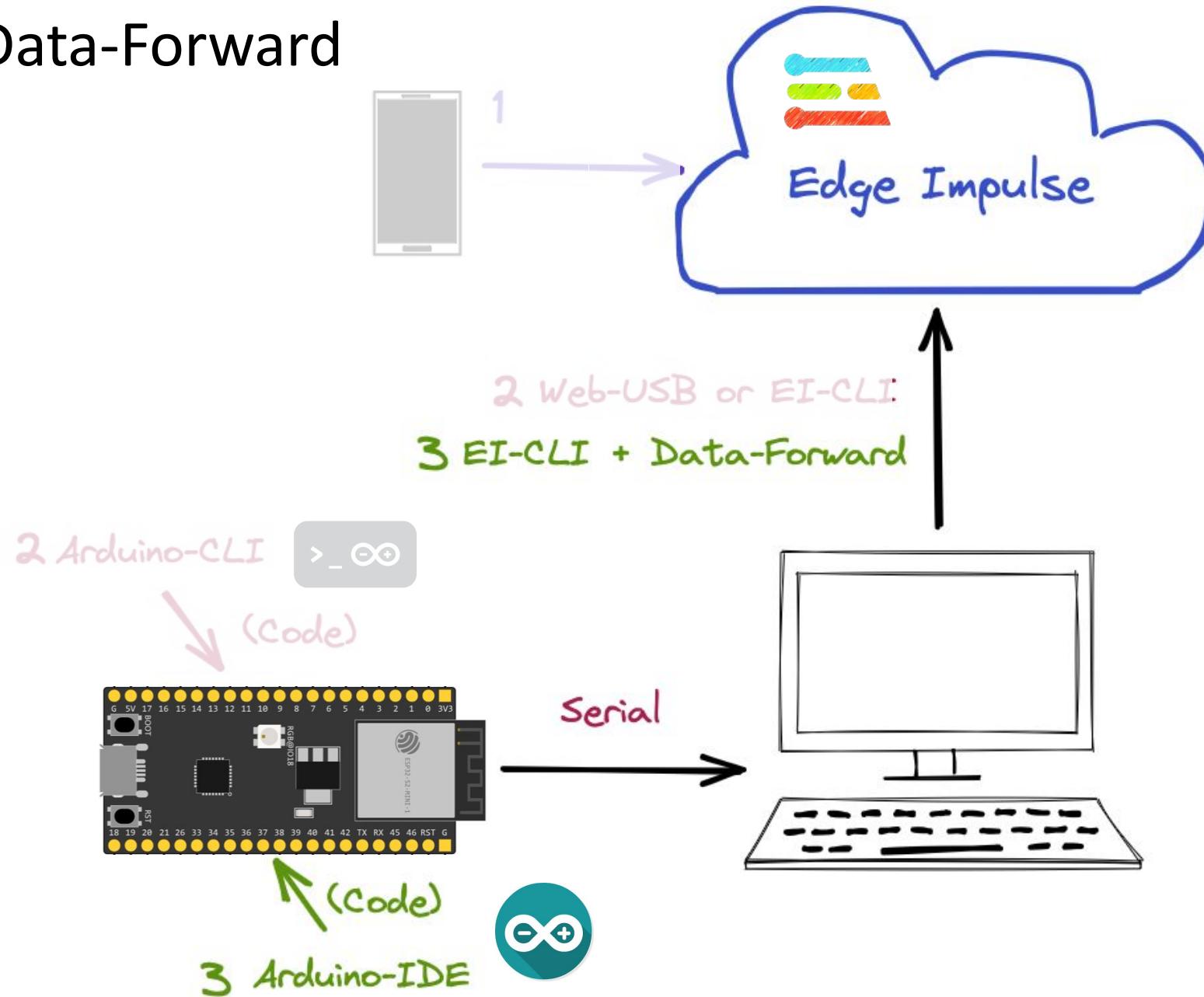
Assets 3

wio-terminal-ei-1.4.0.uf2 (239 KB)

Source code (zip)

Source code (tar.gz)

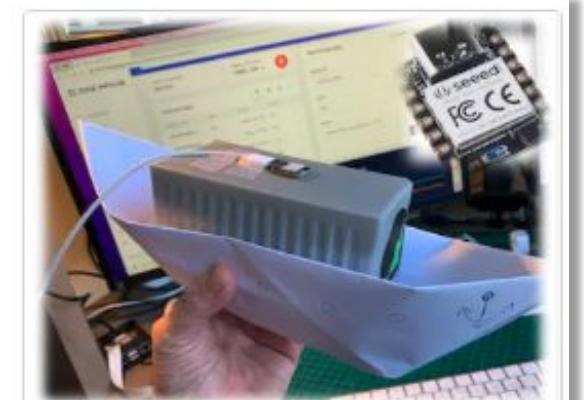
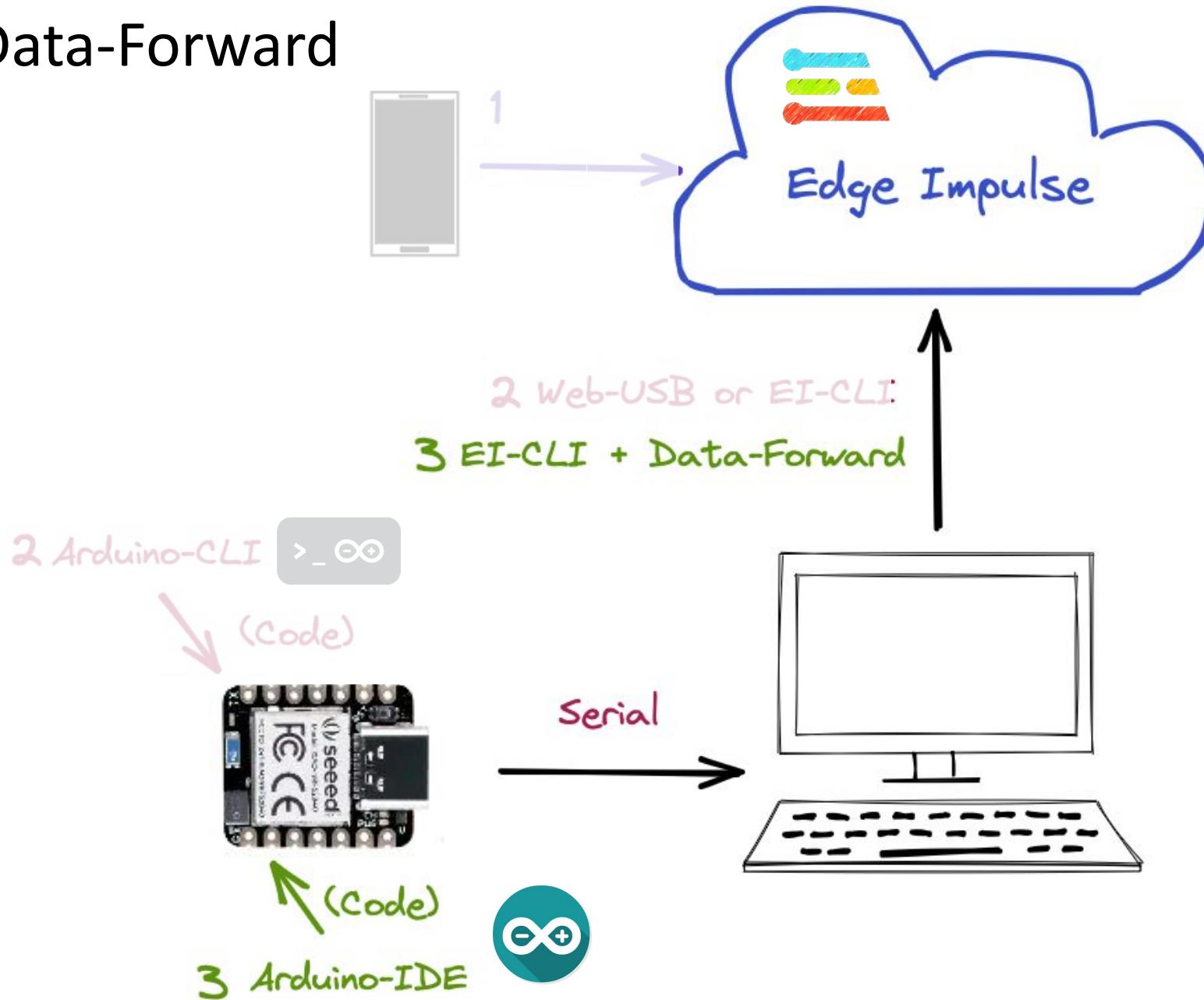
### 3. Data-Forward



ESP32 - Motion Classification

Prof. Morello Izaid Rovai  
UNIFEI - Federal University of Itajubá, Brazil  
TinyML4D Academic Network Co-Chair

### 3. Data-Forward



TinyML Made Easy: Anomaly  
Detection & Motion Classification  
MJRoBot (Marcelo Rovai)

## 2. Data Ingestion using El-Cli + Data Forward

```
XIAO_BLE_Sense_Accelerometer_Data_Forewarder | Arduino 1.8.19
XIAO_BLE_Sense_Accelerometer_Data_Forewarder $ 

9  Marcelo Rovai @July2022
10 */
11 #include "LSM6DS3.h"
12 #include "Wire.h"
13
14 //Create an instance of class LSM6DS3
15 LSM6DS3 xIMU(I2C_MODE, 0x6A); //I2C device address 0x6A
16
17 #define CONVERT_G_TO_MS2 9.80665f
18 #define FREQUENCY_HZ 50
19 #define INTERVAL_MS (1000 / (FREQUENCY_HZ + 1))
20 static unsigned long last_interval_ms = 0;
21
22 void setup() {
23   Serial.begin(115200);
24   while (!Serial);
25
26   if (xIMU.begin() != 0) {
27     Serial.println("Device error");
28   } else {
29     Serial.println("Device OK!");
30   }
31   Serial.println("Data Forwarder - Built-in IMU on the XIAO BLE Sense\n");
32 }
33
34 void loop() {
35   float x, y, z;
36   if (millis() > last_interval_ms + INTERVAL_MS) {
37     last_interval_ms = millis();
38     x = xIMU.readFloatAccelX();
39     y = xIMU.readFloatAccelY();
40     z = xIMU.readFloatAccelZ();
41
42     Serial.print(x * CONVERT_G_TO_MS2);
43     Serial.print('\t');
44     Serial.print(y * CONVERT_G_TO_MS2);
45     Serial.print('\t');
46     Serial.println(z * CONVERT_G_TO_MS2);
47   }
48 }
```

dúlio Wio Terminal, Master, Enabled, 120 MHz (standard), Small (-Os) (standard), 50 MHz (standard), Arduino, Off, On on /dev/cu.usbmodem1101

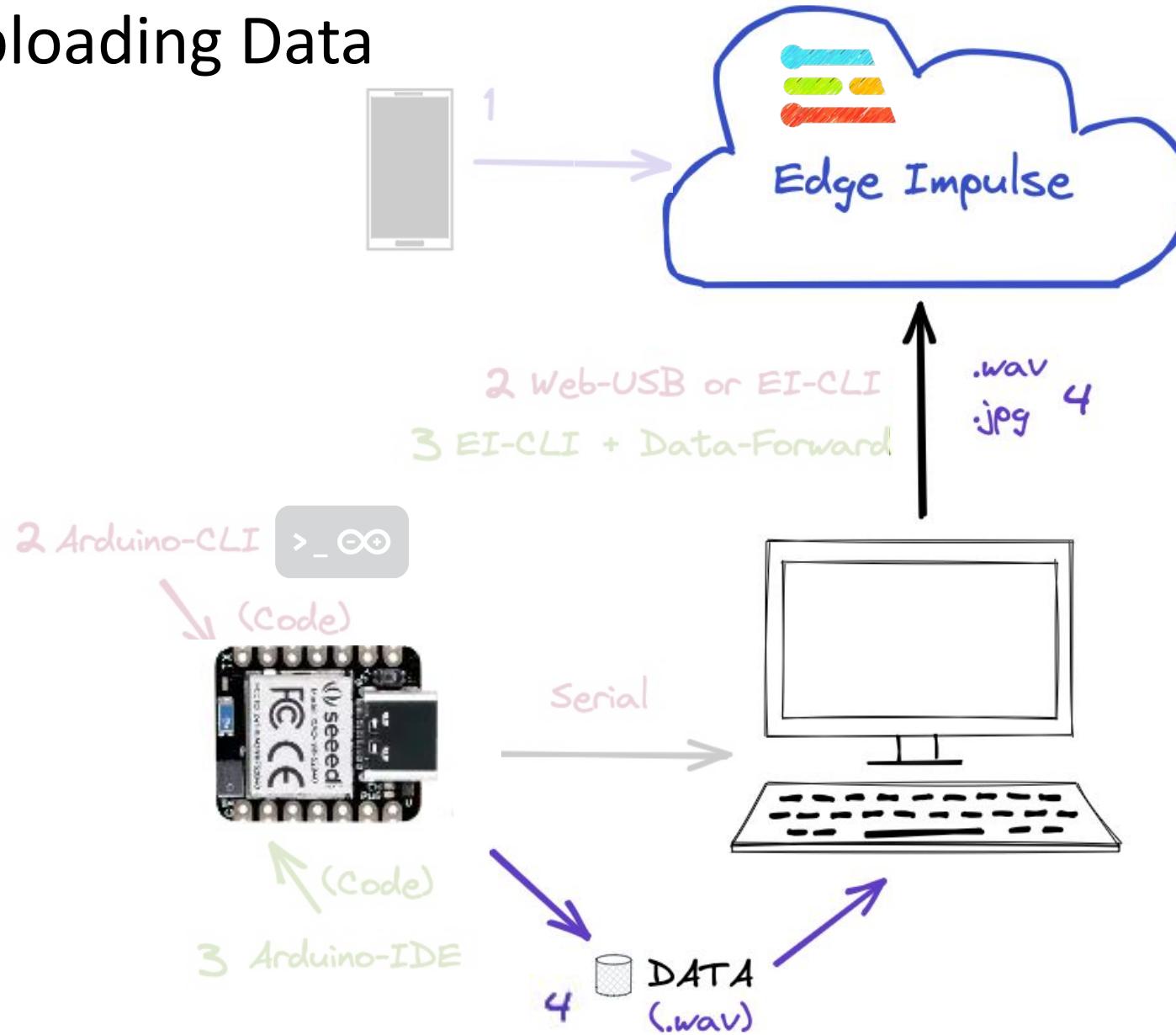
\$ edge-impulse-data-forwarder --clean

```
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)'
)
```

## 4. Uploading Data



## 4. Data Ingestion using Upload existing Data

The screenshot illustrates the process of uploading data to a project. On the left, the main interface shows a tree view of data categories: 'data' (containing 'cool' and 'hot') and a list of wav files. A blue arrow points from the 'hot' category in the tree view to the 'hot' label input field in the upload dialog. The central dialog is titled 'UPLOAD DATA (ICTP\_PSYCHOACOUSTICS\_TEMPERATURE\_DEPENDENCE)' and contains the following fields:

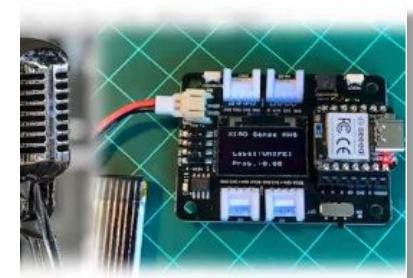
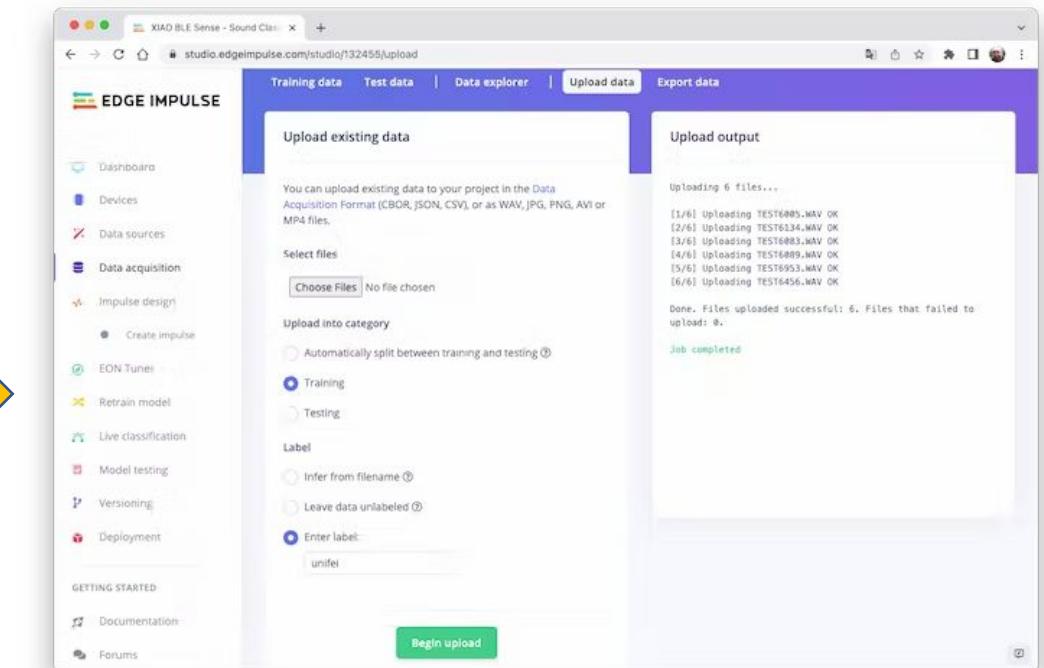
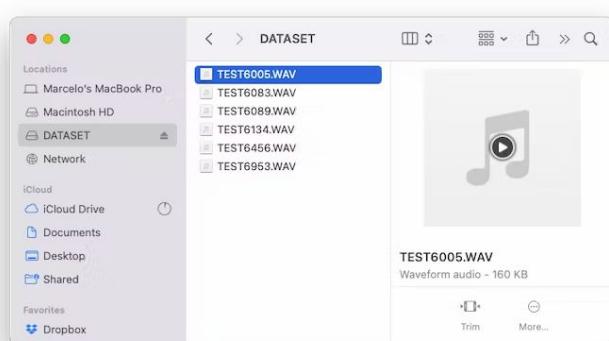
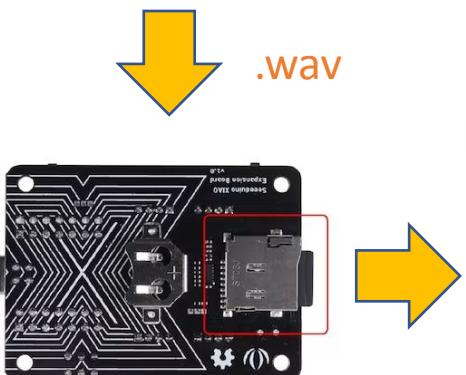
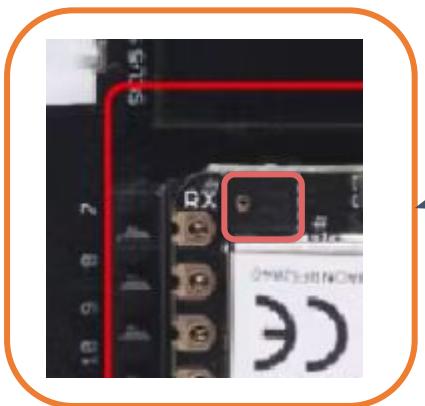
- Upload existing data:** A text area with placeholder text: "You can upload existing data to your project in the Data Acquisition Format (CBOR, JSON, CSV), or as WAV, JPG or PNG files."
- Select files:** A 'Choose Files' button with the text "No file chosen".
- Upload into category:** A radio button group:
  - Automatically split between training and testing (unselected)
  - Training (selected)
  - Testing (unselected)
- Label:** A radio button group:
  - Infer from filename (unselected)
  - Enter label: (selected) with an input field containing the value "hot".
- Begin upload** (a green button at the bottom right of the dialog).

To the right, a separate window titled "Upload output" shows the progress of uploading 14 files, with a log of successful uploads:

```
[ 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
```

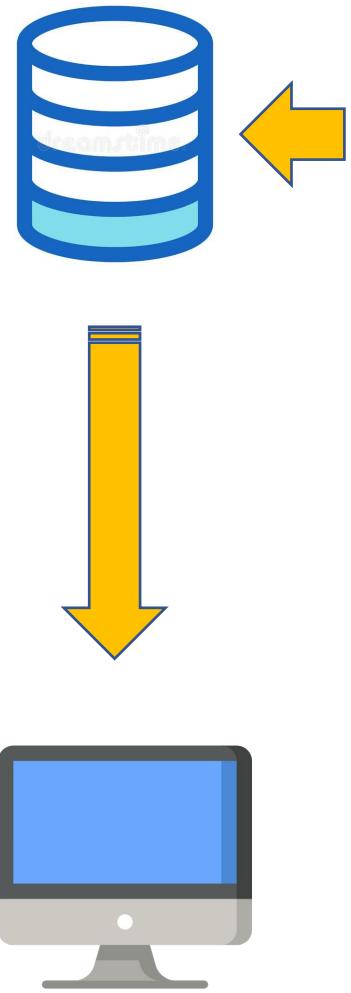
The status message at the bottom of the output window reads: "Done. Files uploaded successful: 14. Files that failed to upload: 0." and "Job completed".

# 4. Uploading .wav data



TinyML Made Easy: Sound Classification (KWS)  
MJRoBot (Marcelo Rovai)

# 4. Uploading .jpg data



<https://github.com/YoongiKim/CIFAR-10-images>

The screenshot shows two main windows. On the left is a GitHub repository page for 'CIFAR-10-images' with branches 'master', '1 branch', and '0 tags'. It lists files: 'test', 'train', and 'README.md'. A green 'Code' button is highlighted with a red box. On the right is the 'EDGE IMPULSE' studio interface. It displays a file browser with a tree view of 'CIFAR-10-images-master' containing 'test', 'train', and 'airplane' folders. A specific file, '0006.jpg', is selected and shown in a preview window. To the right of the file browser is an 'UPLOAD DATA (CIFAR10\_IMAGE\_CLASSIFICATION)' form. It has sections for 'Select files' (with a 'Choose Files' button), 'Upload into category' (radio buttons for 'Training' and 'Testing'), 'Label' (radio buttons for 'Infer from filename' and 'Enter label' with a text input field containing 'dog'), and a 'Begin upload' button. A large yellow arrow points from the '0006.jpg' preview area towards the 'Begin upload' button.

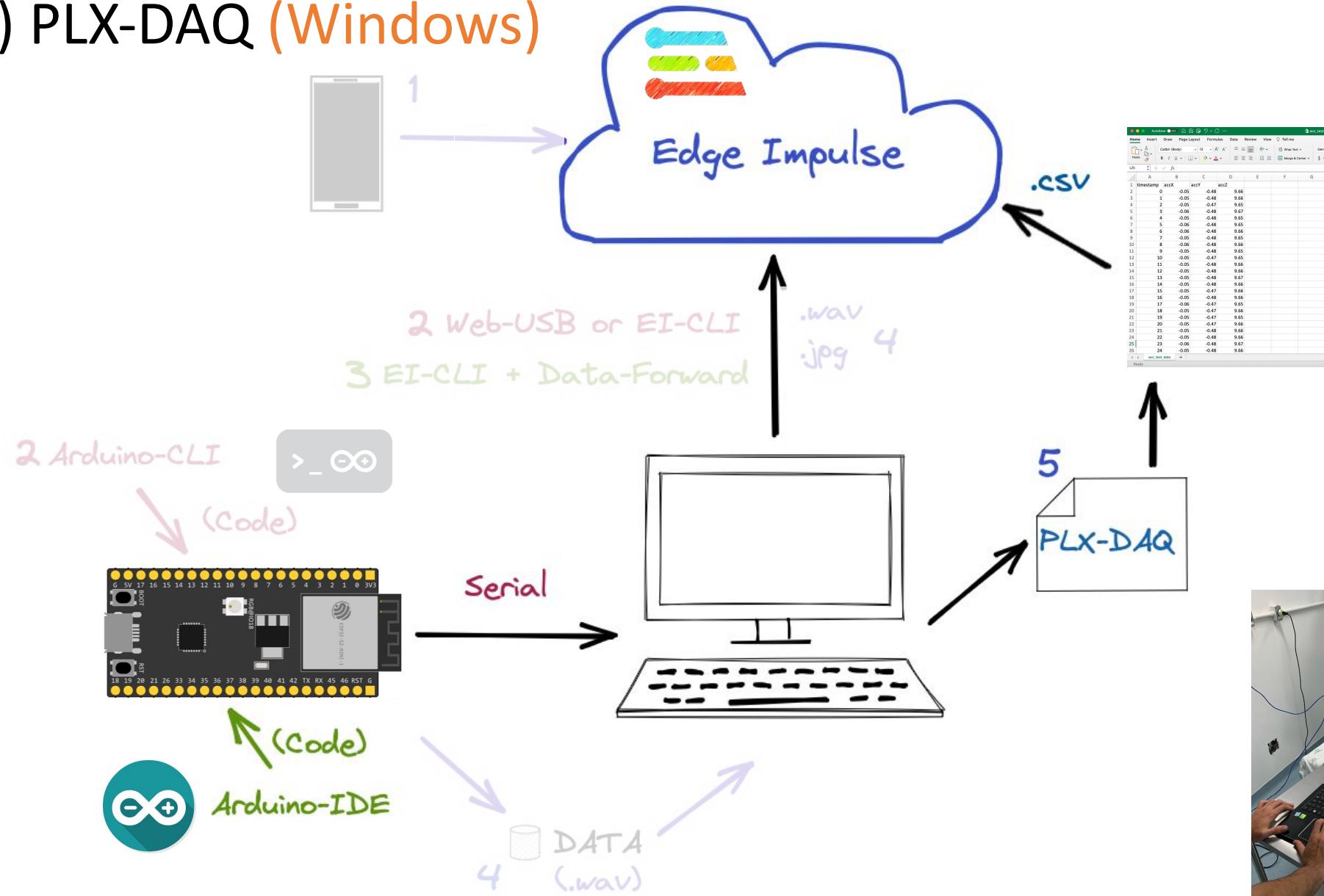


Image Classification (Cifar10) using  
Convolutions (CNN) and Edge Impulse Studio

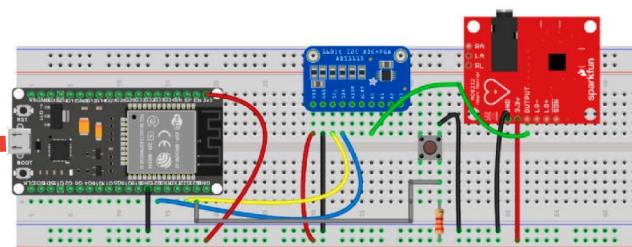
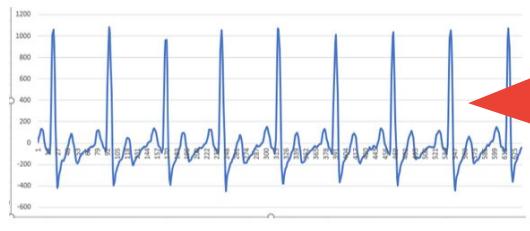
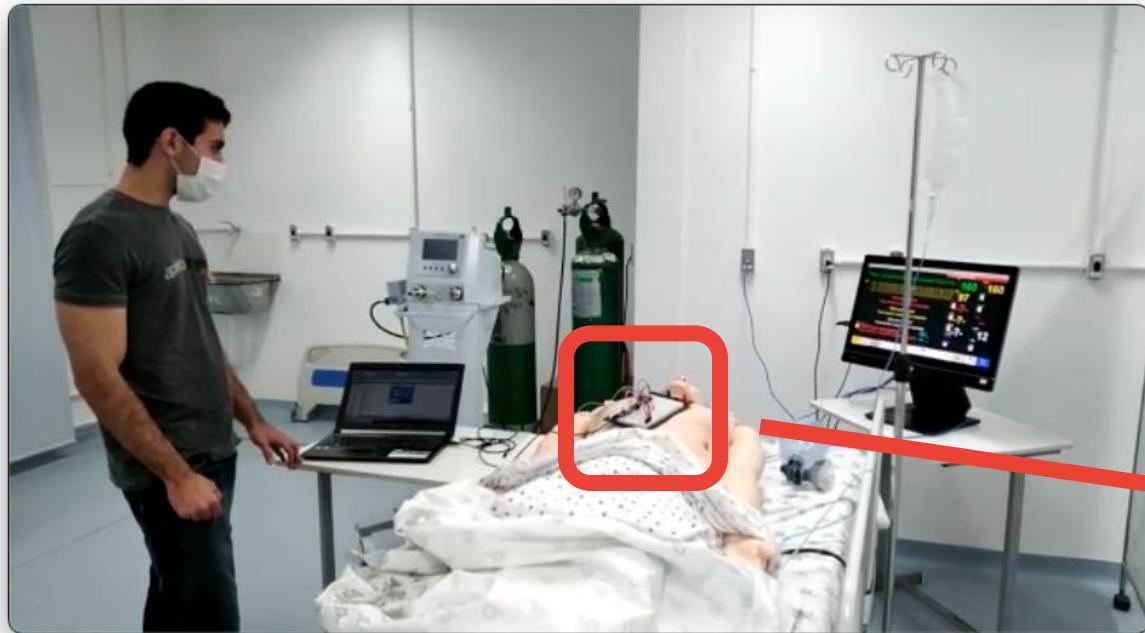
Prof. Marcelo José Roval  
UNIFEI - Federal University of Itajubá, Brazil  
TinyML40 Academic Network Co-Chair



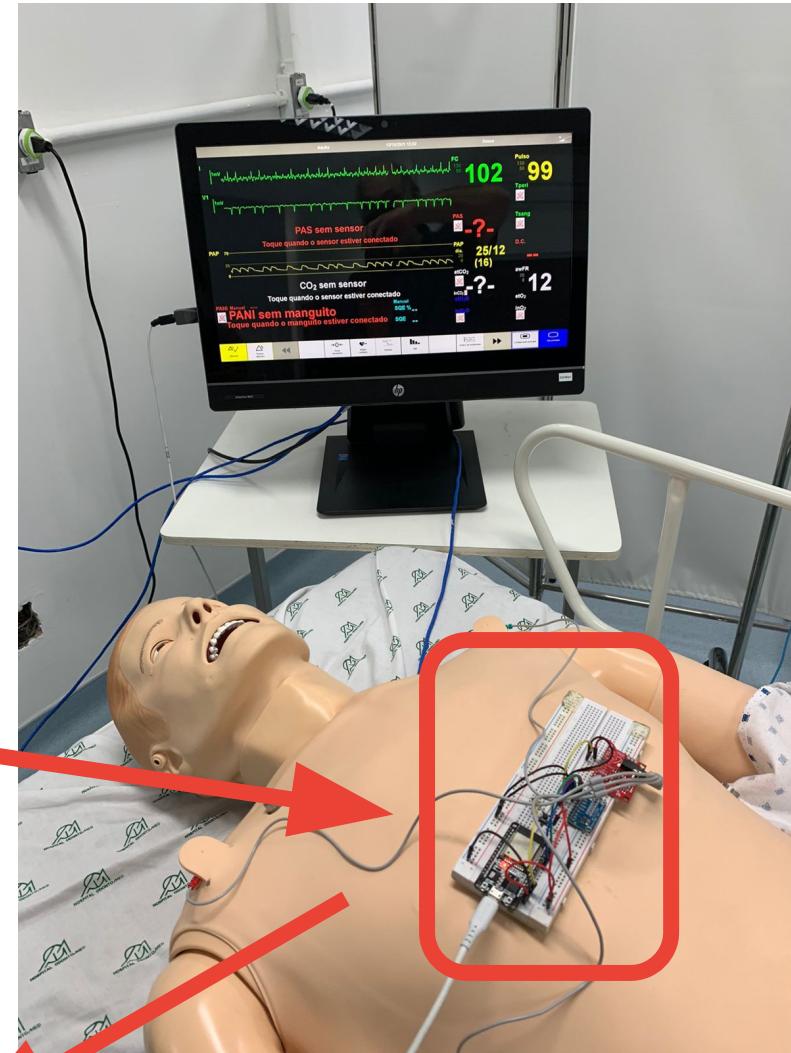
## 5. (.CSV) PLX-DAQ (Windows)



## 5. (.CSV) PLX-DAQ (Windows)

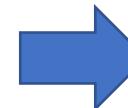
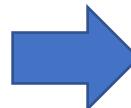


fritzing



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

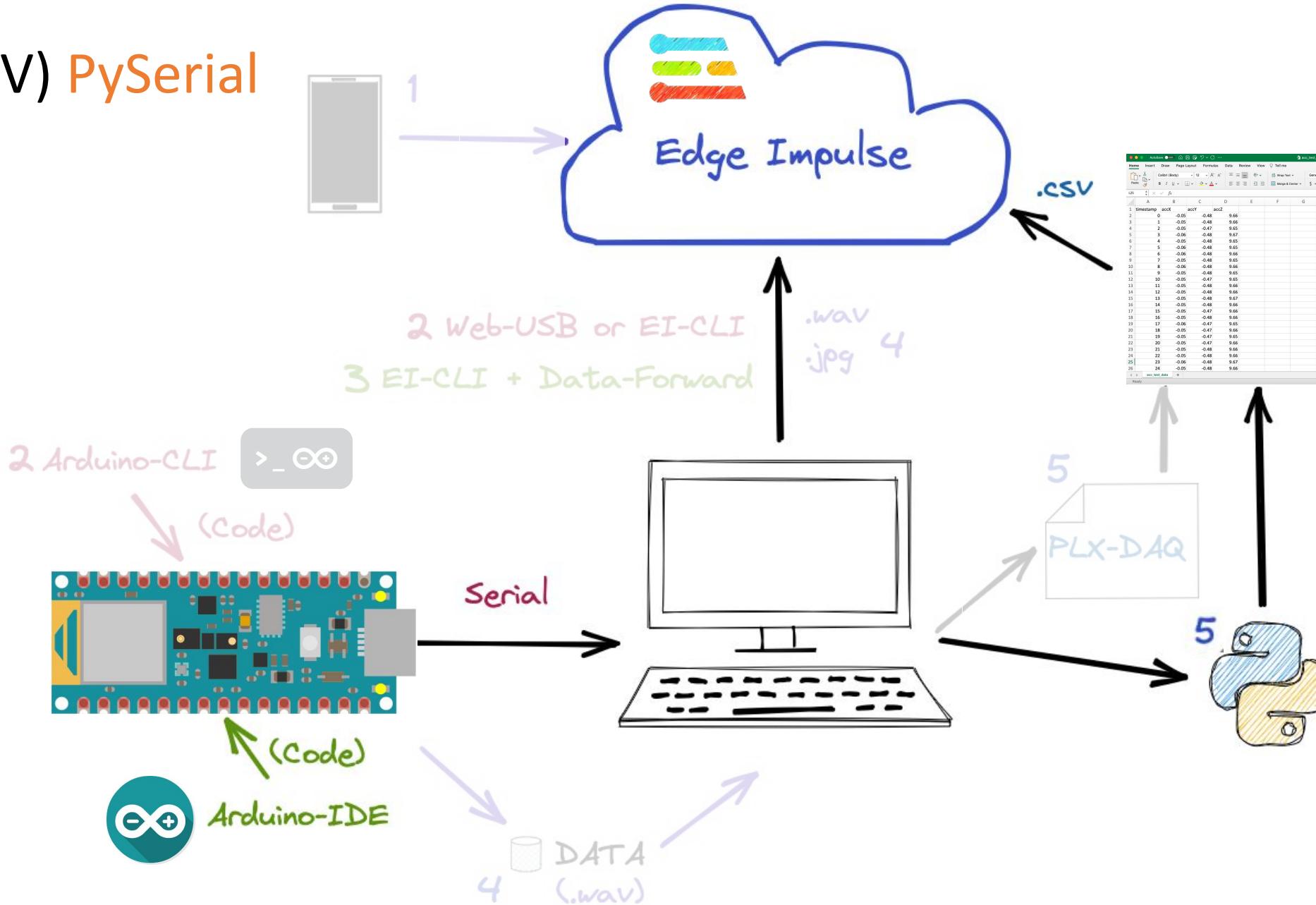
```
Capture_Ard33_Sense_IMU_Acc
1 #include <Arduino_LSM9DS1.h>
2
3 #define CONVERT_G_TO_MS2 9.80665f
4 #define FREQUENCY_HZ 50
5 #define INTERVAL_MS (1000 / (FREQUENCY_HZ + 1))
6
7 void setup() {
8     Serial.begin(115200);
9     while (!Serial);
10    Serial.println("Started");
11
12    if (!IMU.begin()) {
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		

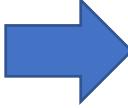
<https://www.youtube.com/watch?v=BwbmNle2CZo>

## 5. (.CSV) PySerial



# 5. 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 }
```

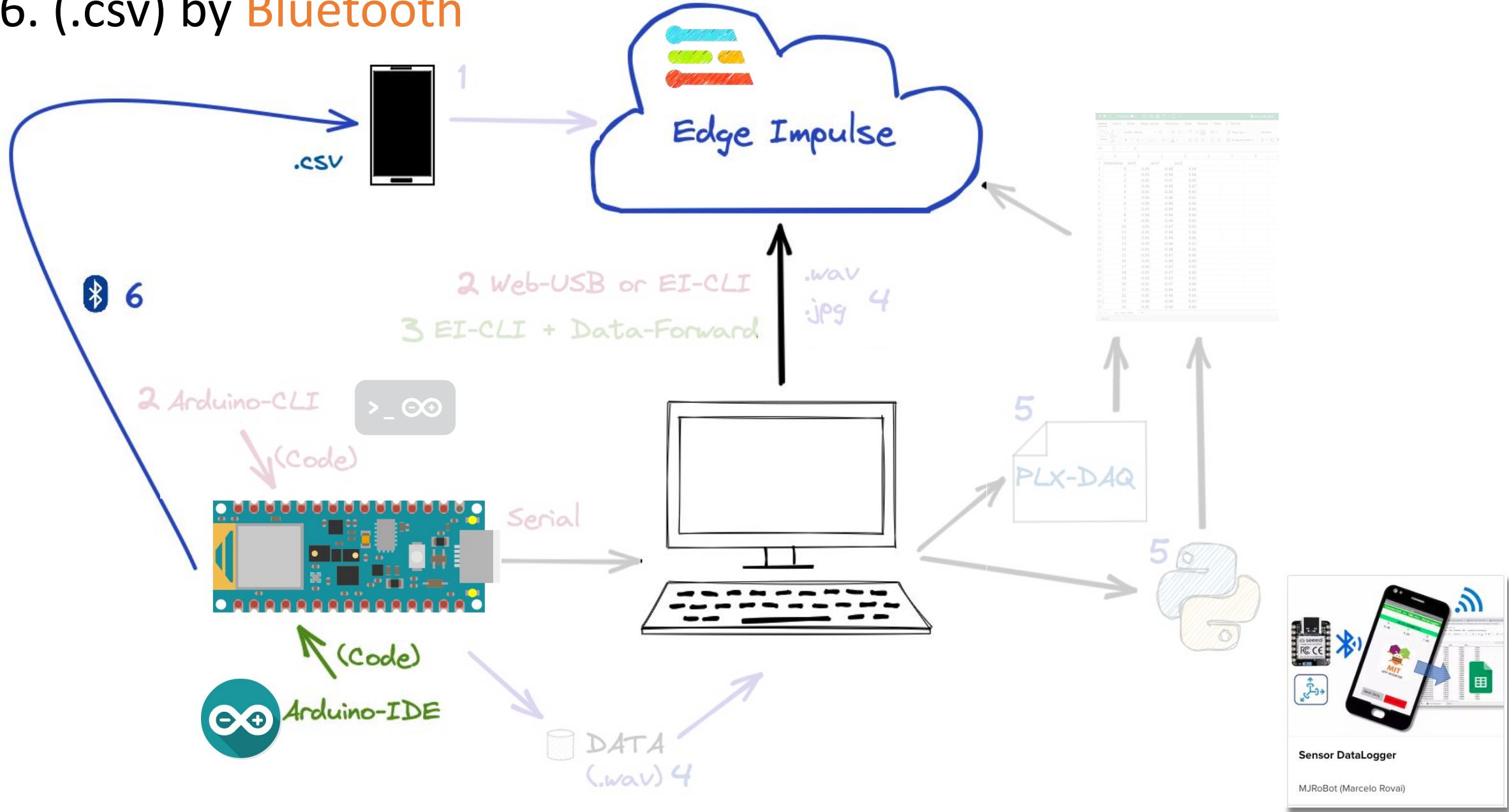


```
1 # Sensor data Logger (CSV)
2 # by Marcelo Rovai @ 13July21
3
4 import serial
5
6 arduino_port = '/dev/tty.usbmodem144301'
7 baud_rate = 115200
8 ser = serial.Serial(port=arduino_port, baudrate=baud_rate)
9
10 fileName = "acc_test_data.csv" # name of the CSV file generated
11
12 first_line = 'timestamp,accX,accY,accZ'
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()
```

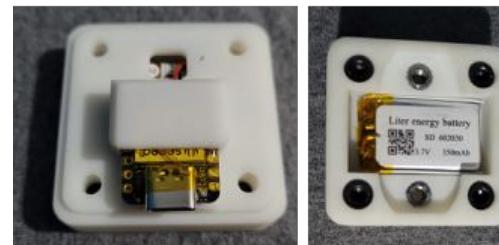
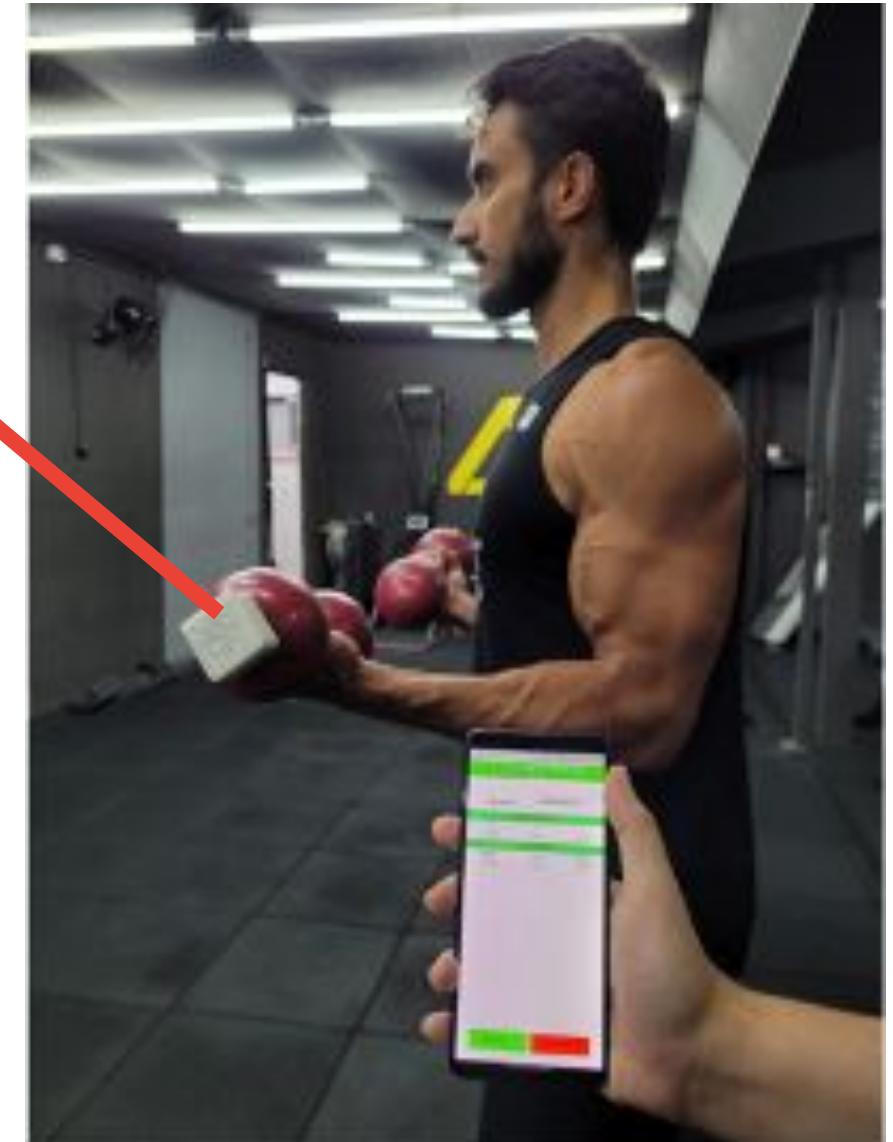
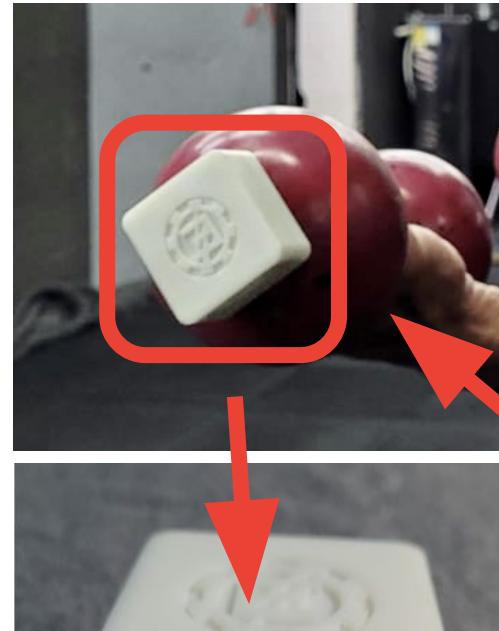


L25	A	B	C	D	E	F	G
1	timestamp	accX	accY	accZ			
2	0	-0.05	-0.48	9.66			
3	1	-0.05	-0.48	9.66			
4	2	-0.05	-0.47	9.65			
5	3	-0.06	-0.48	9.67			
6	4	-0.05	-0.48	9.65			
7	5	-0.06	-0.48	9.65			
8	6	-0.06	-0.48	9.66			
9	7	-0.05	-0.48	9.65			
10	8	-0.06	-0.48	9.66			
11	9	-0.05	-0.48	9.65			
12	10	-0.05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

## 6. (.csv) by Bluetooth



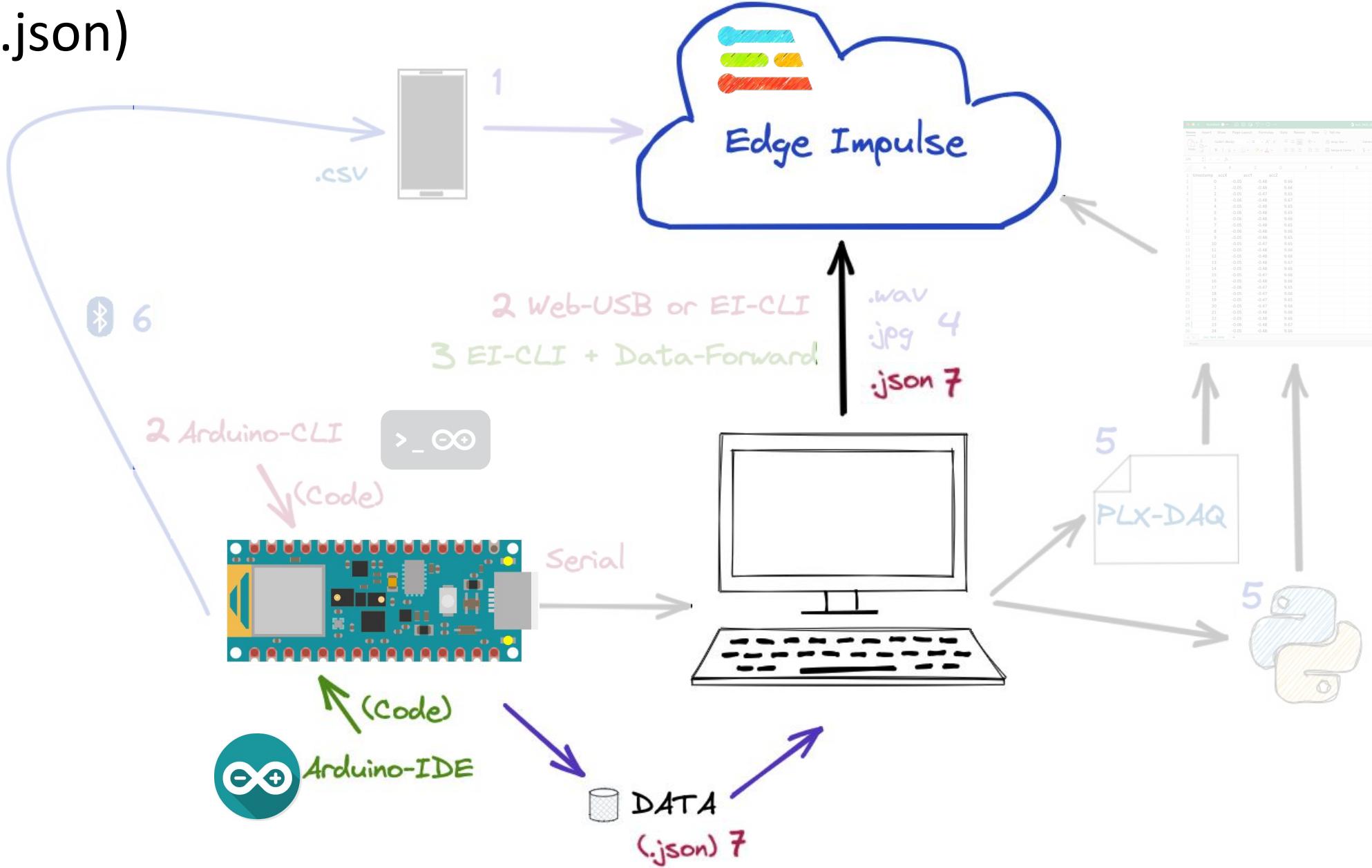
## 6. (.csv) by Bluetooth



## 6. (.csv) by Bluetooth



## 7. (.json)



# 7. Raw Uploader (.json files)

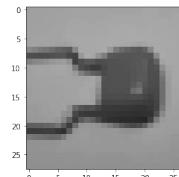
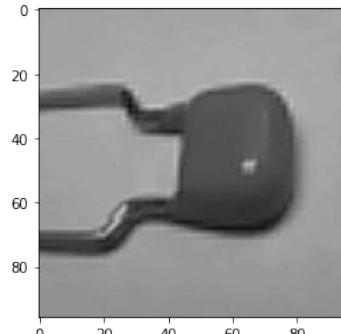
## Image Classification: Raw Uploader

 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
```



.json →



**EDGE IMPULSE**  
(Training as DNN)

Author: Edgelimpulse, Inc.

Date: June 6, 2021

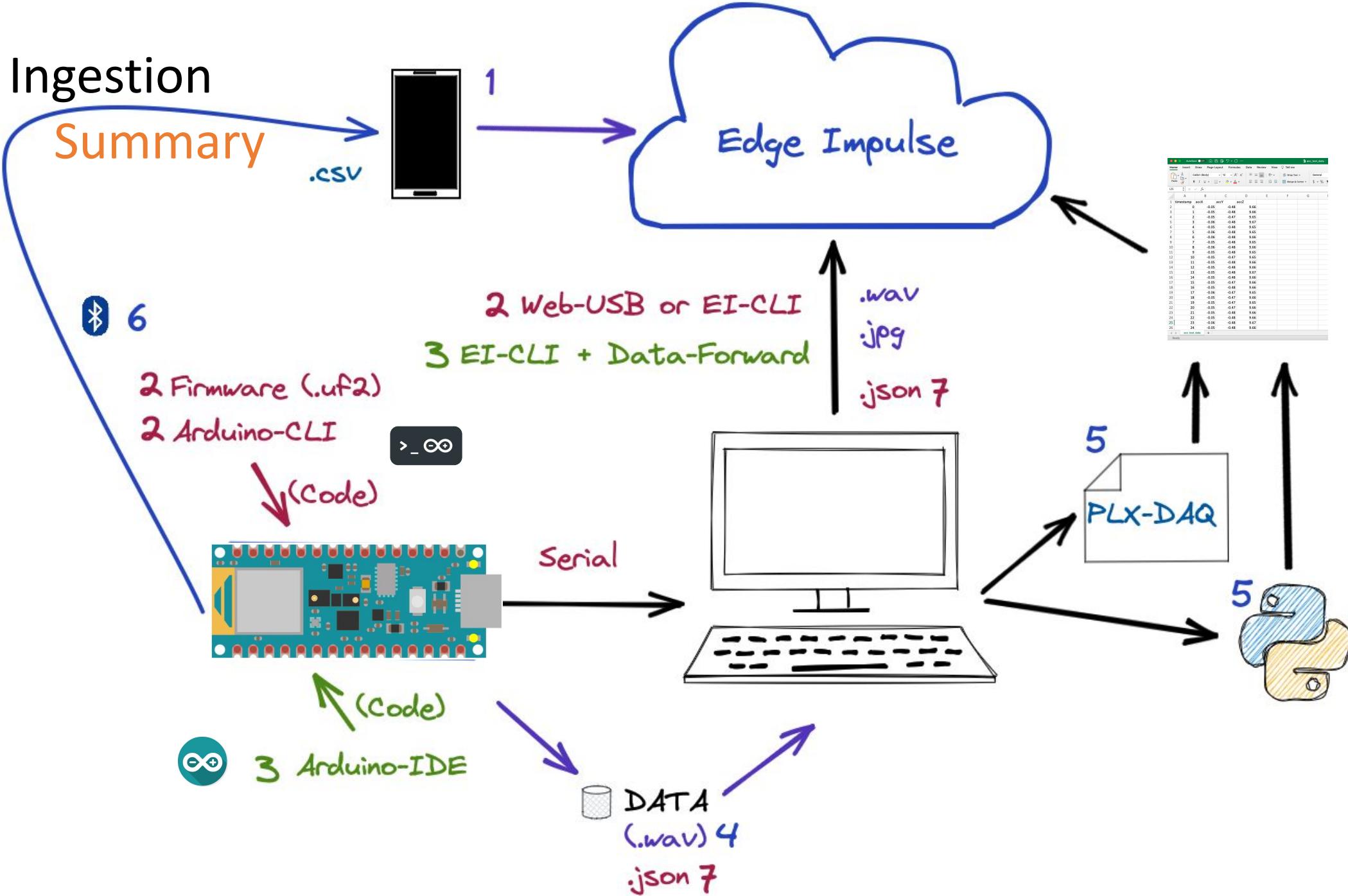
License: [Apache-2.0](#)



TinyML Made Easy: Exploring  
Regression - White Wine Quality  
MJRoBot (Marcelo Rovai)

# Data Ingestion

## Summary



**Thanks**



**UNIFEI**