Leveraging any Microcontrollers & Data Collection at Edge Impulse Studio

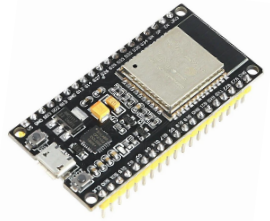
Prof. Marcelo José Rovai
UNIFEI - Federal University of Itajubá, Brazil
TinyML4D Academic Network Co-Chair

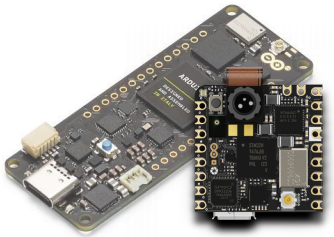
UNIFEI

Hardware

Hardware





	Raspberry Pico (W)	Arduino Nano Sense	ESP 32	Seeed XIAO Sense / ESP32S3	Arduino Pro
32Bits CPU	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
CLOCK	133MHz	64MHz	240MHz	64 / 240MHz	480/240MHz
RAM	264KB	256KB	520KB (part available)	256KB / 8MB	1MB
ROM	2MB	1MB	2MB	2MB / 8MB	2MB
Radio	(Yes for W)	BLE	BLE/WiFi	BLE / WiFi (ESP32S3)	BLE/WiFi
Sensors	No	Yes	No	Yes (Sense)	Yes (Nicla)
Bat. Power Manag.	No	No	No	Yes	Yes
Price	$	$$$	$	$$	$$$$$

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

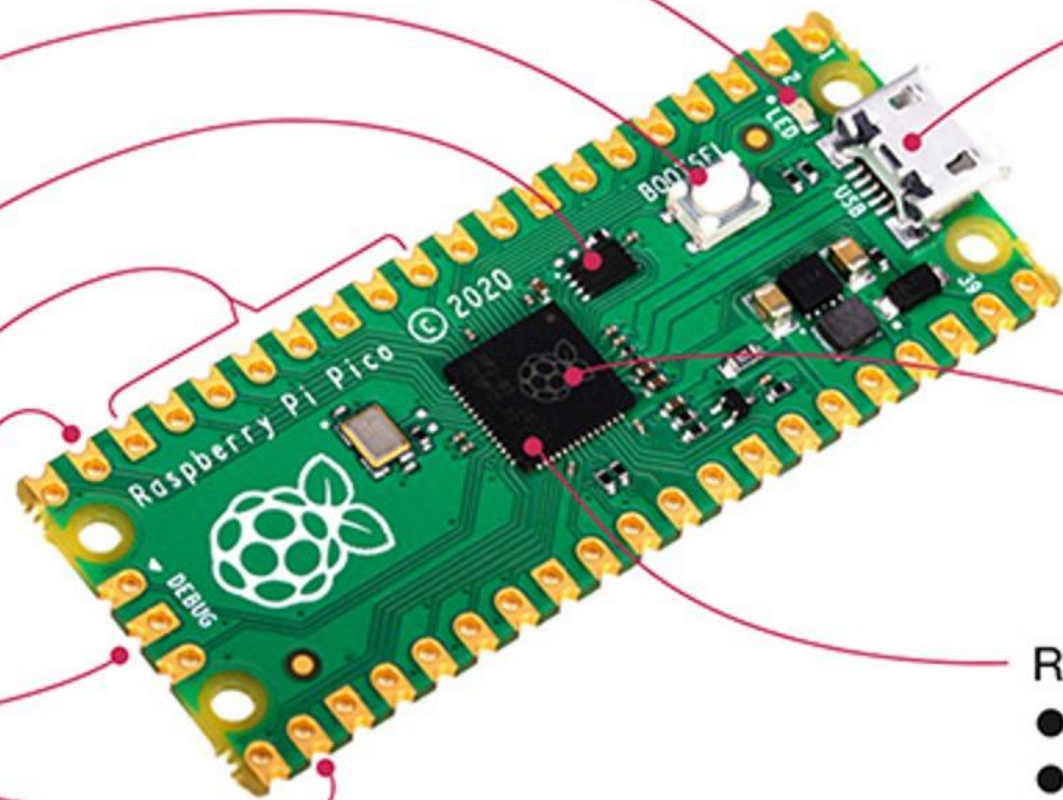
Programmable LED

Boot Selection Button

2 MByte Quad-SPi Flash

0.1” Spacing Pads
Breadboard friendly

SMT Friendly
Castellated Edges



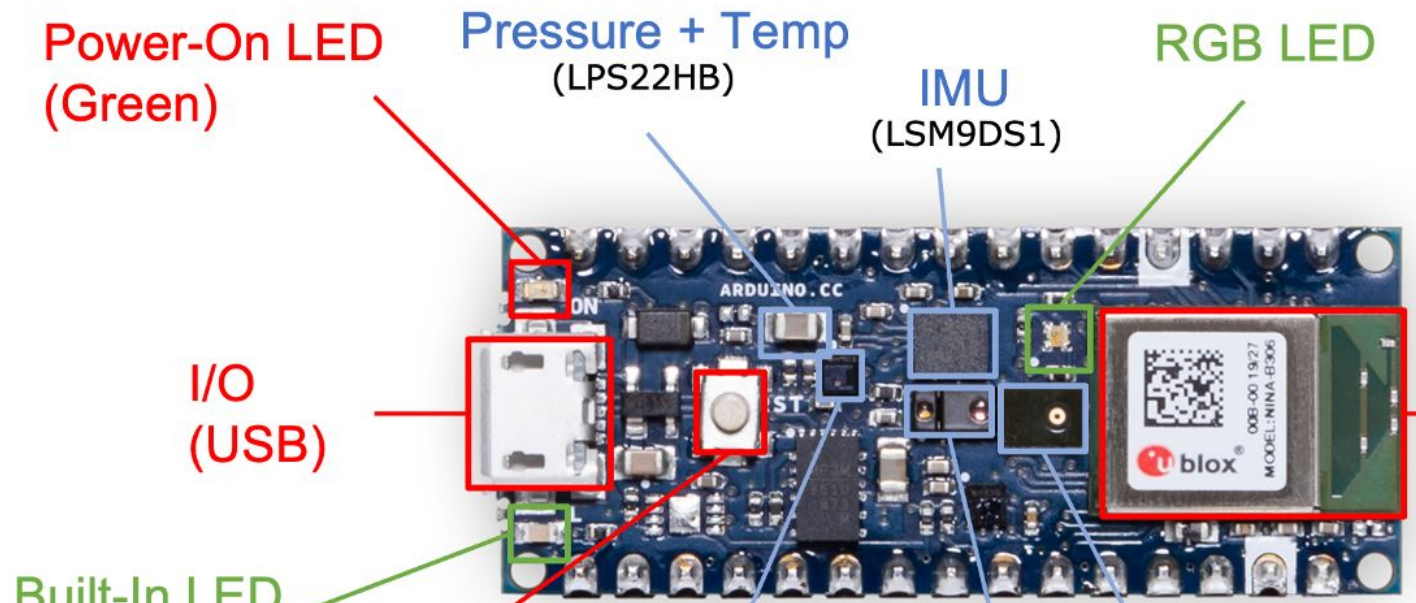
USB Micro B
Power & Data

Built-in 12-bit
Temperature Sensor

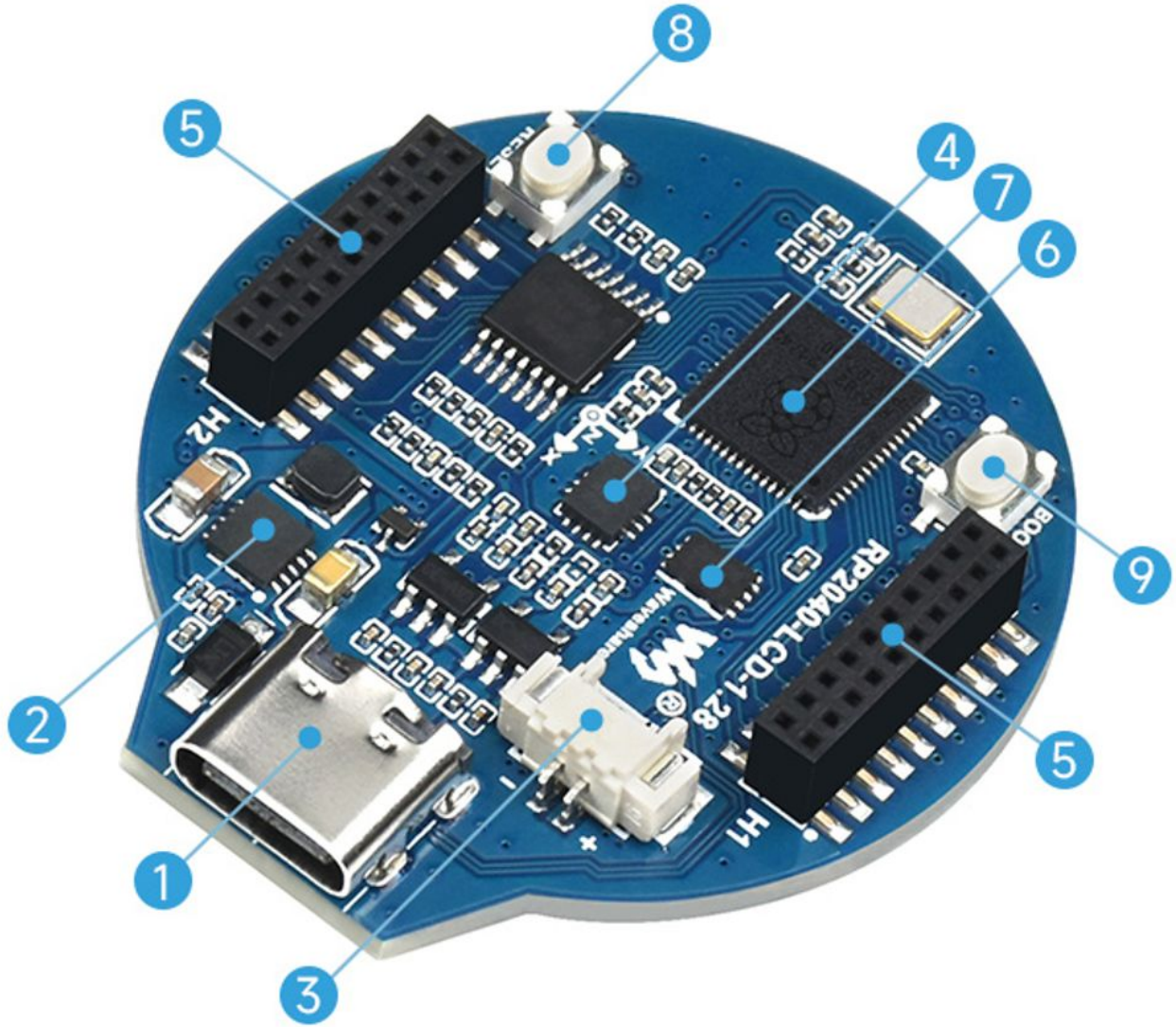
RP2040 MCU Silicon

- 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)

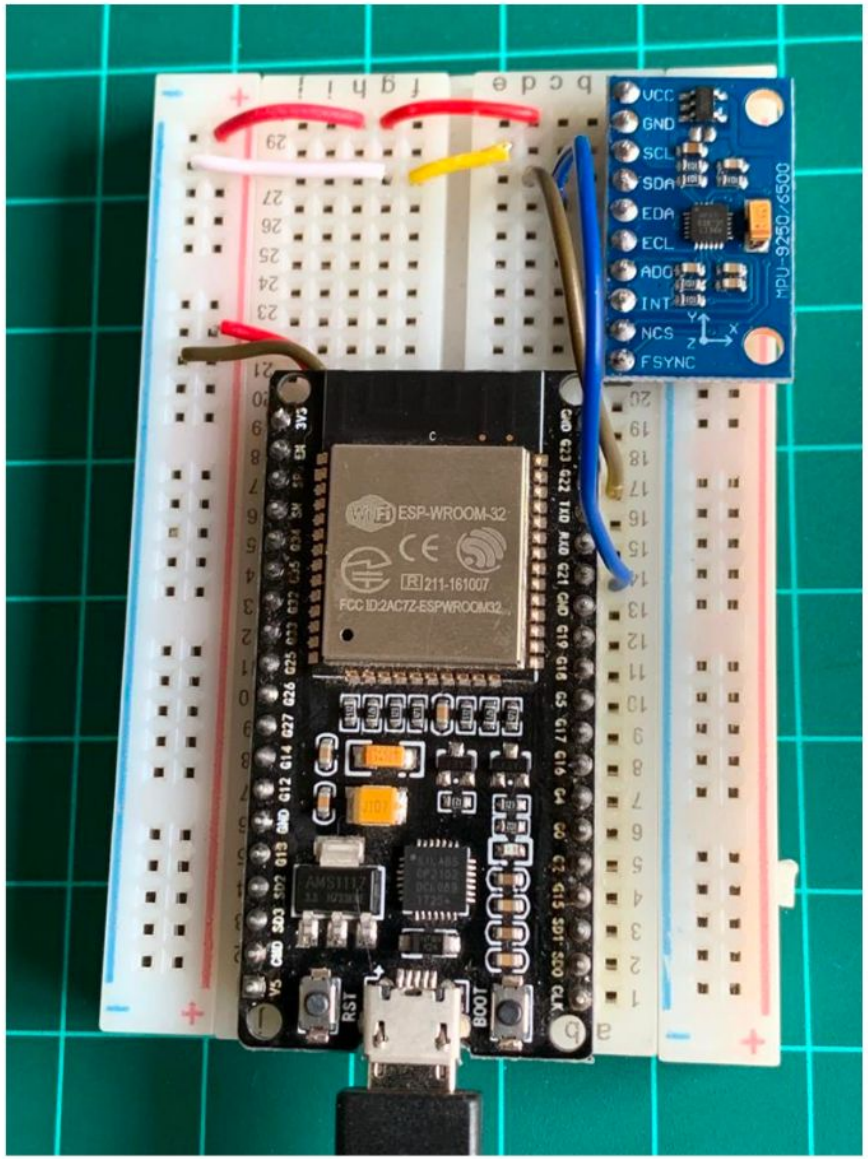


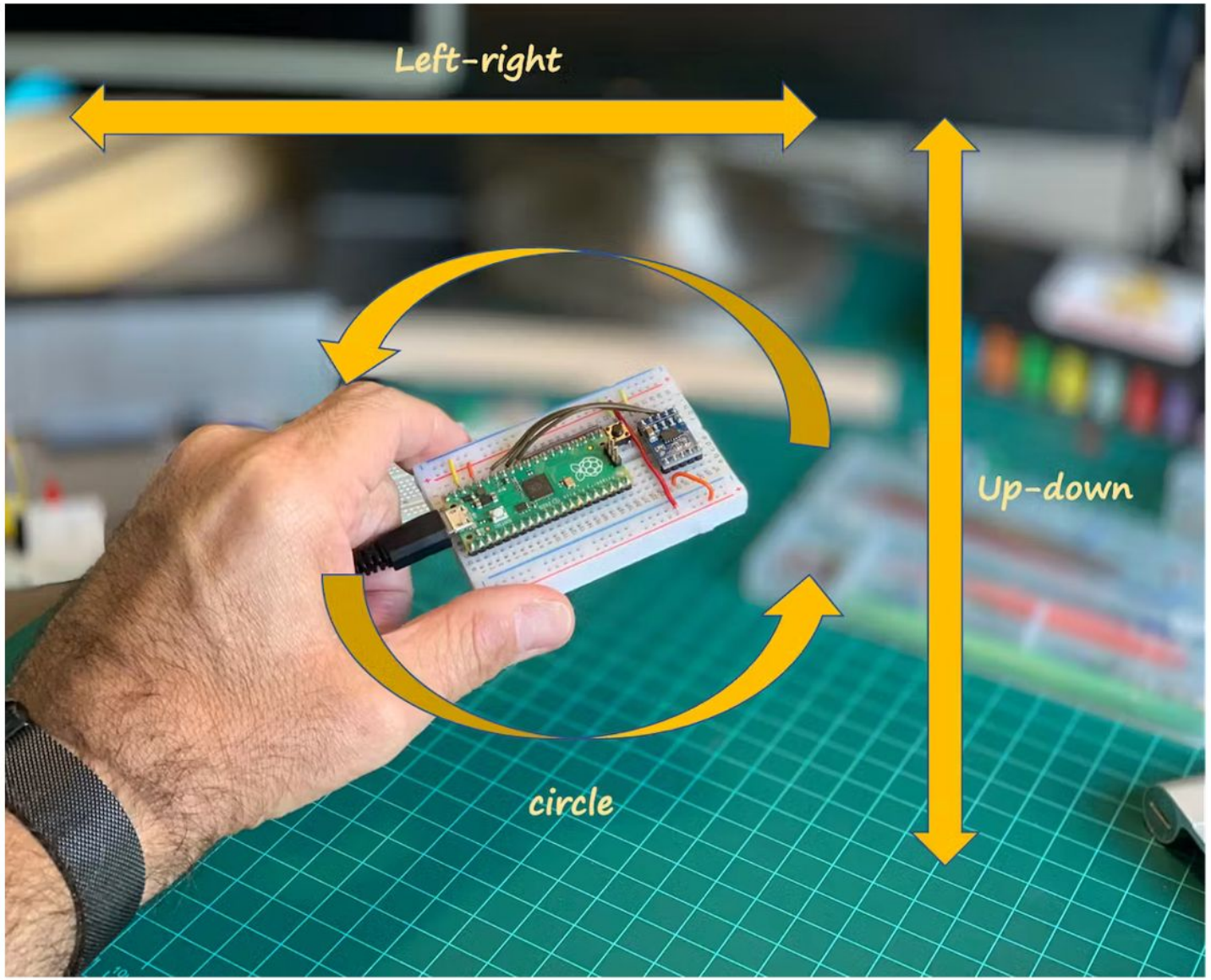
RP2040 MCU Board, with LCD, accelerometer, and gyroscope Sensor



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


VCC
GND
SCL
SDA
EDA
ECL
ADO
INT
NCS
FSYNC
MPU-9250/6500
ESP-WROOM-32
211-161007
FCC ID:2AC7Z-ESPWROOM32
RST
BOOT


Left-right
Up-down
circle

Application Complexity vs. HW

Power

EdgeML

TinyML

Video Classification 2 MB+

Object Detection Complex Voice Processing 1 MB+

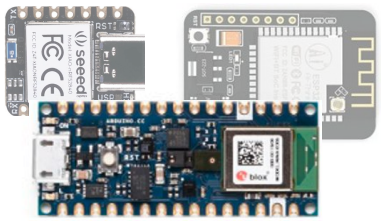
Image Classification 250 KB+

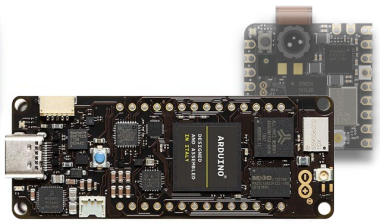
KeyWord Spotting Audio Classification 50 KB

Anomaly Detection Sensor Classification 20 KB

XIAO

ESP32





Rpi-Pico (Cortex-M0+)

Arduino Nano (Cortex-M4)

Arduino Pro (Cortex-M7)

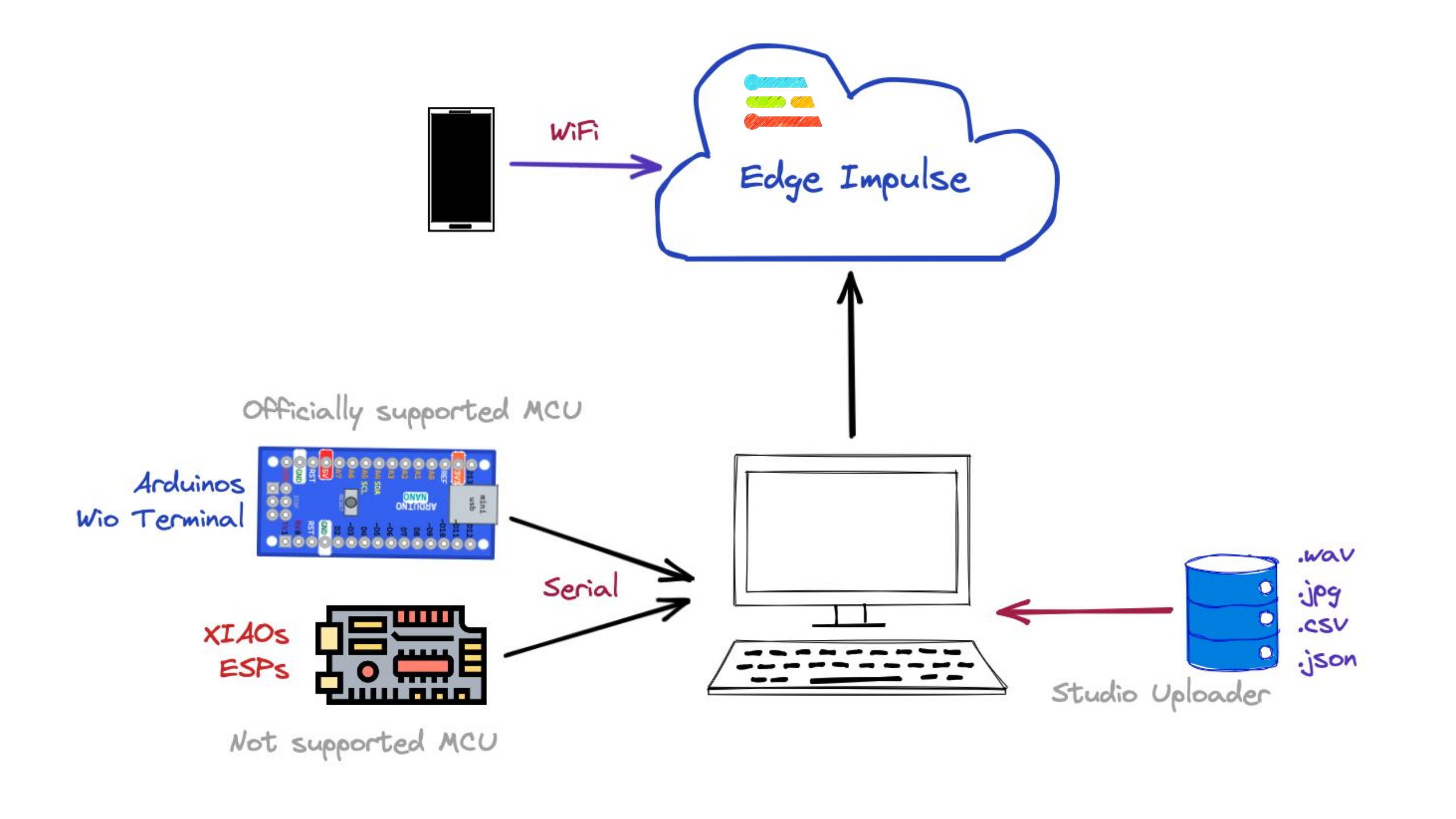
RaspberryPi SmartPhone (Cortex-A)

Jetson Nano (Cortex-A + GPU)

Application Complexity

CPU Power / Memory

EI Studio Data Ingestion
Alternative methods


WiFi
Edge Impulse
Officially supported MCU
Arduinos
Wio Terminal
Serial
XIAOs
ESPs
Not supported MCU
.wav
.jpg
.csv
.json
Studio Uploader

1. SmartPhone

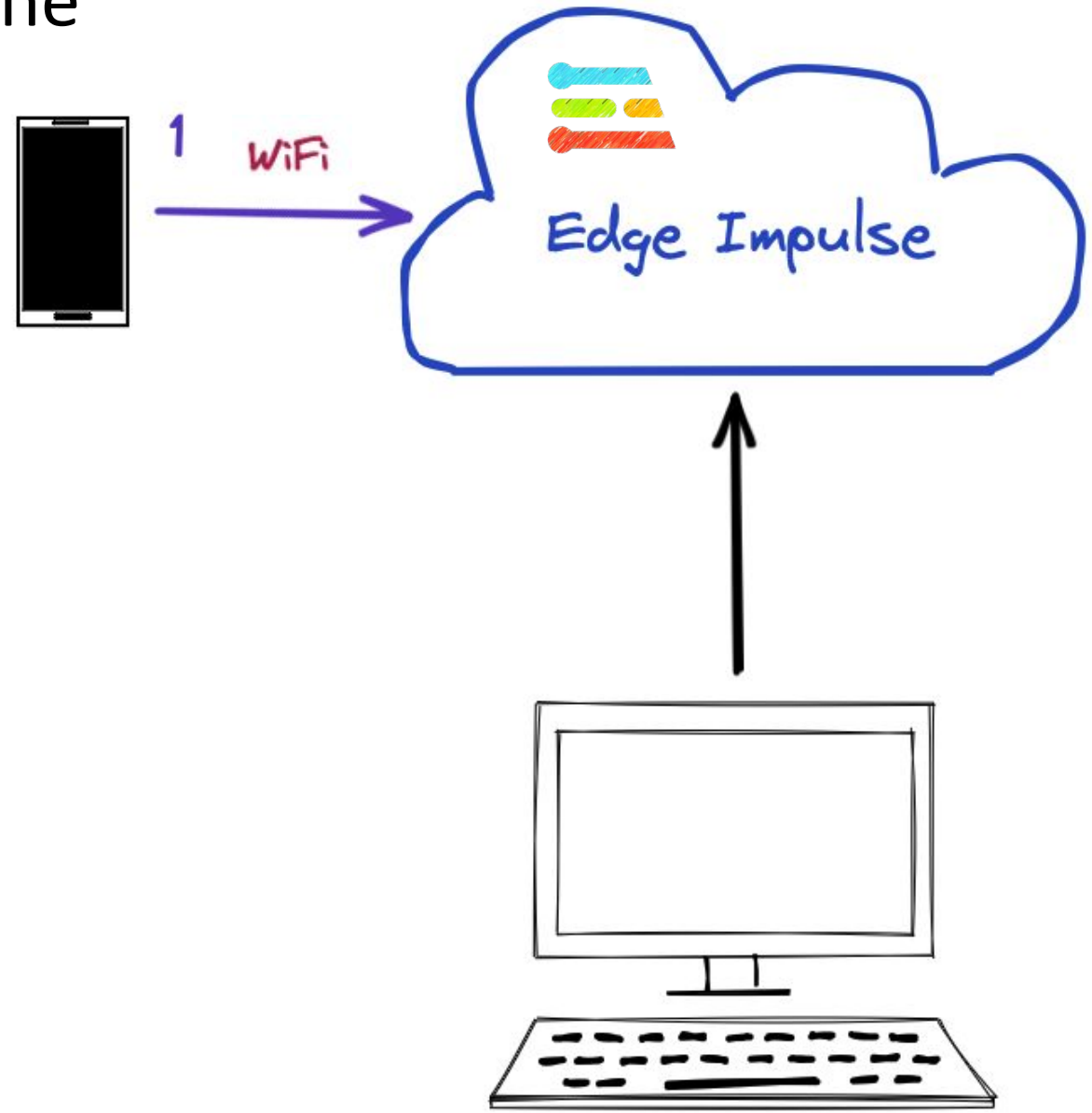


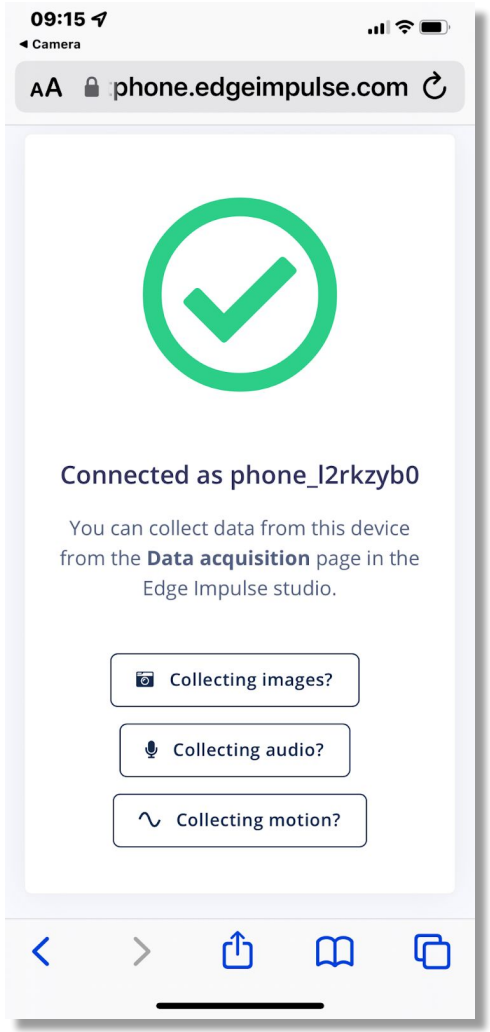
Image Classification using a smartphone and Edge Impulse Studio

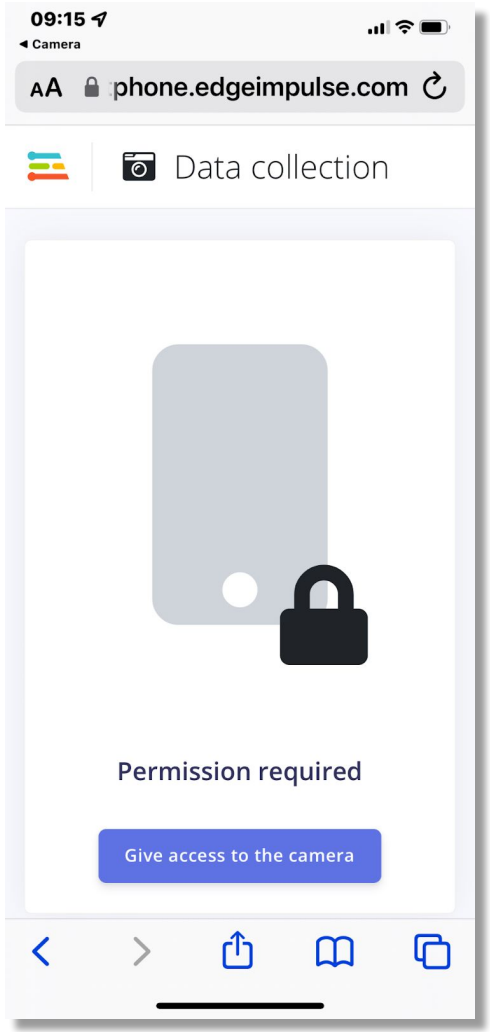
Prof. Marcelo José Rovai
UNIFEI - Federal University of Itajubá, Brazil
TinyML4D Academic Network Co-Chair

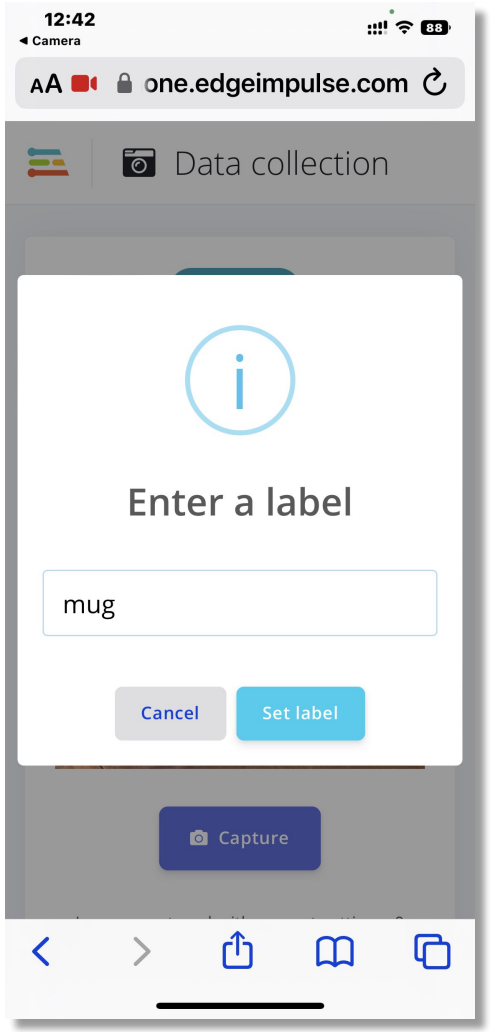
UNIFEI

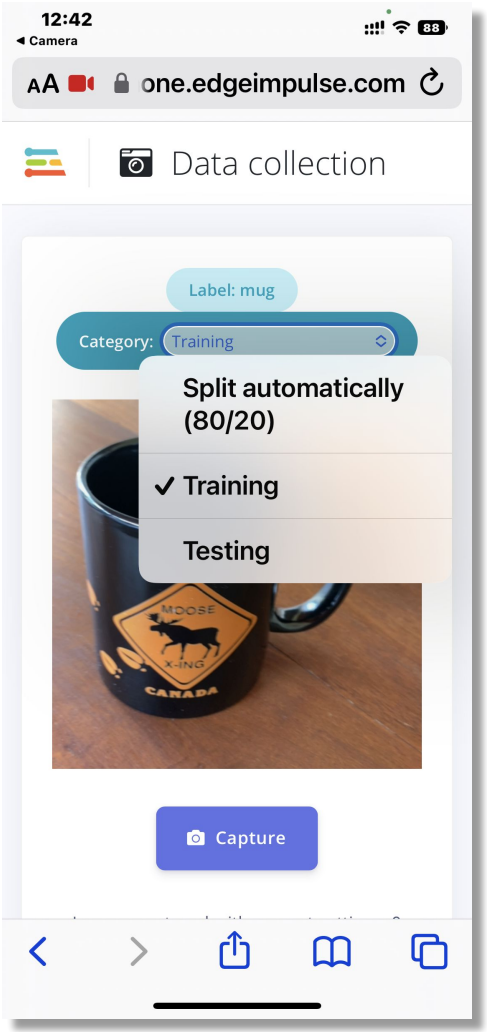
1. Data Ingestion using Smart Phone

1. Data Capture and model Training

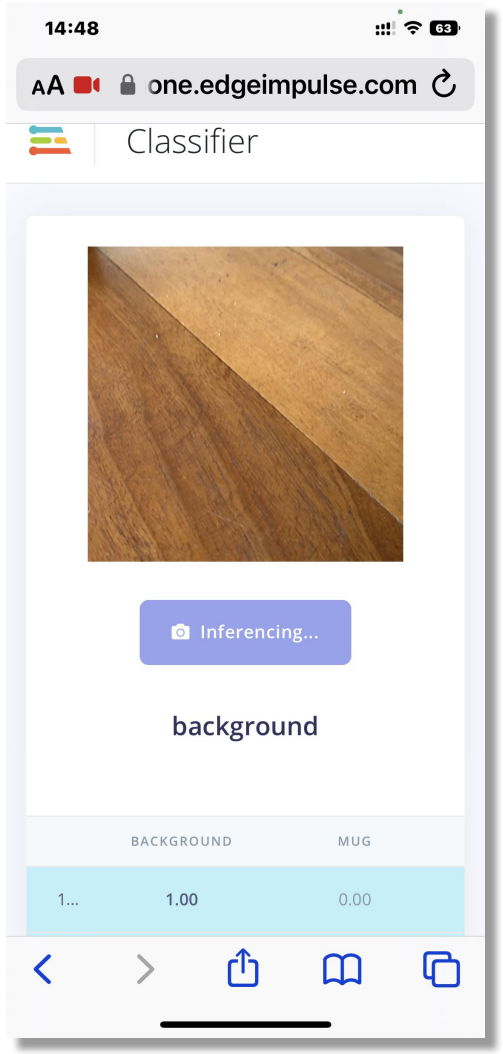


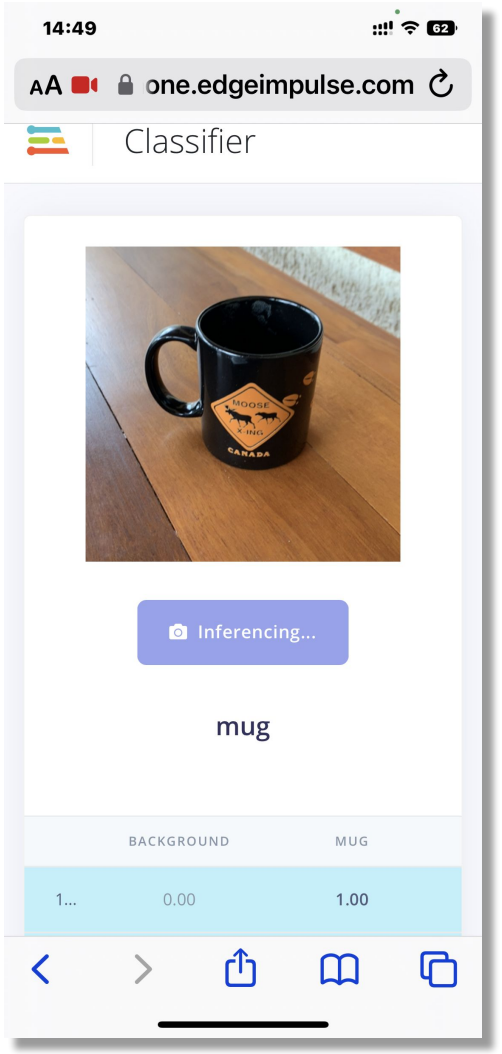


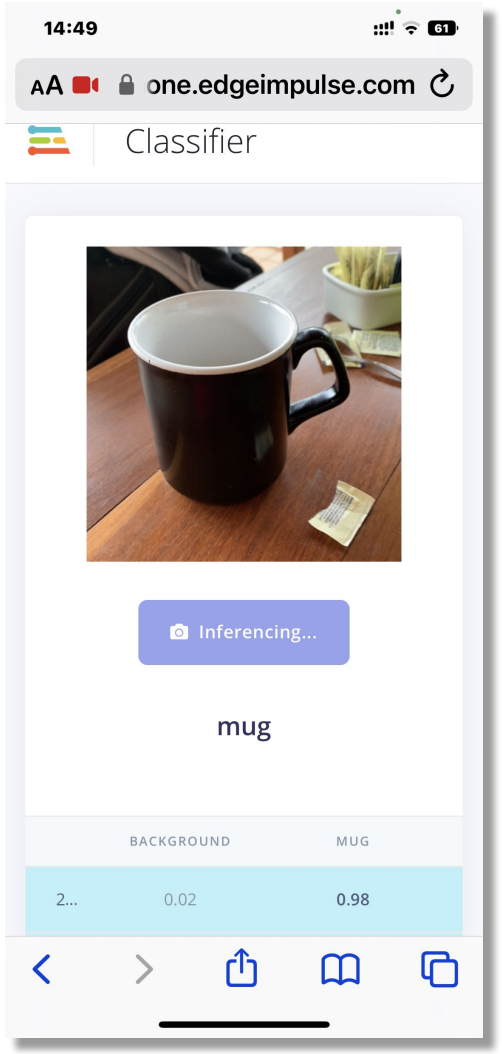


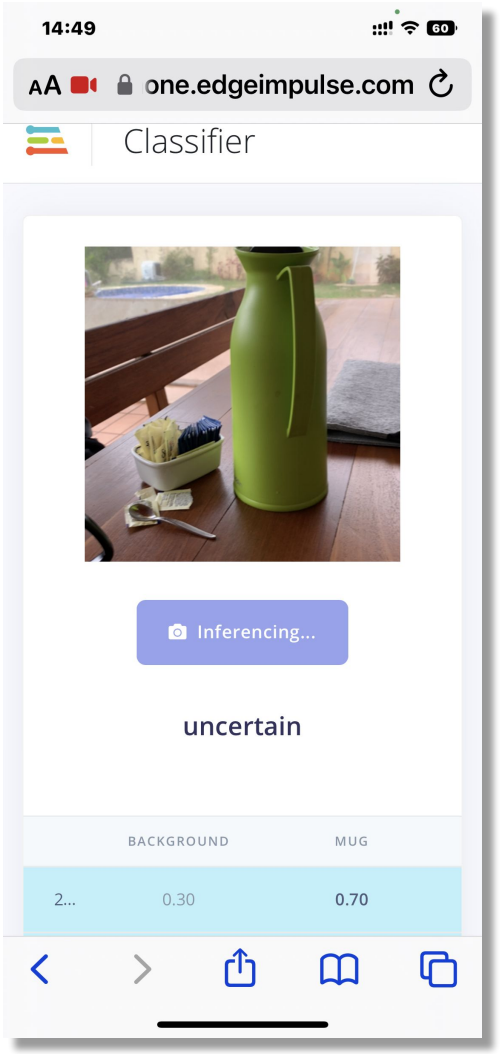


1. Off-Line Inference

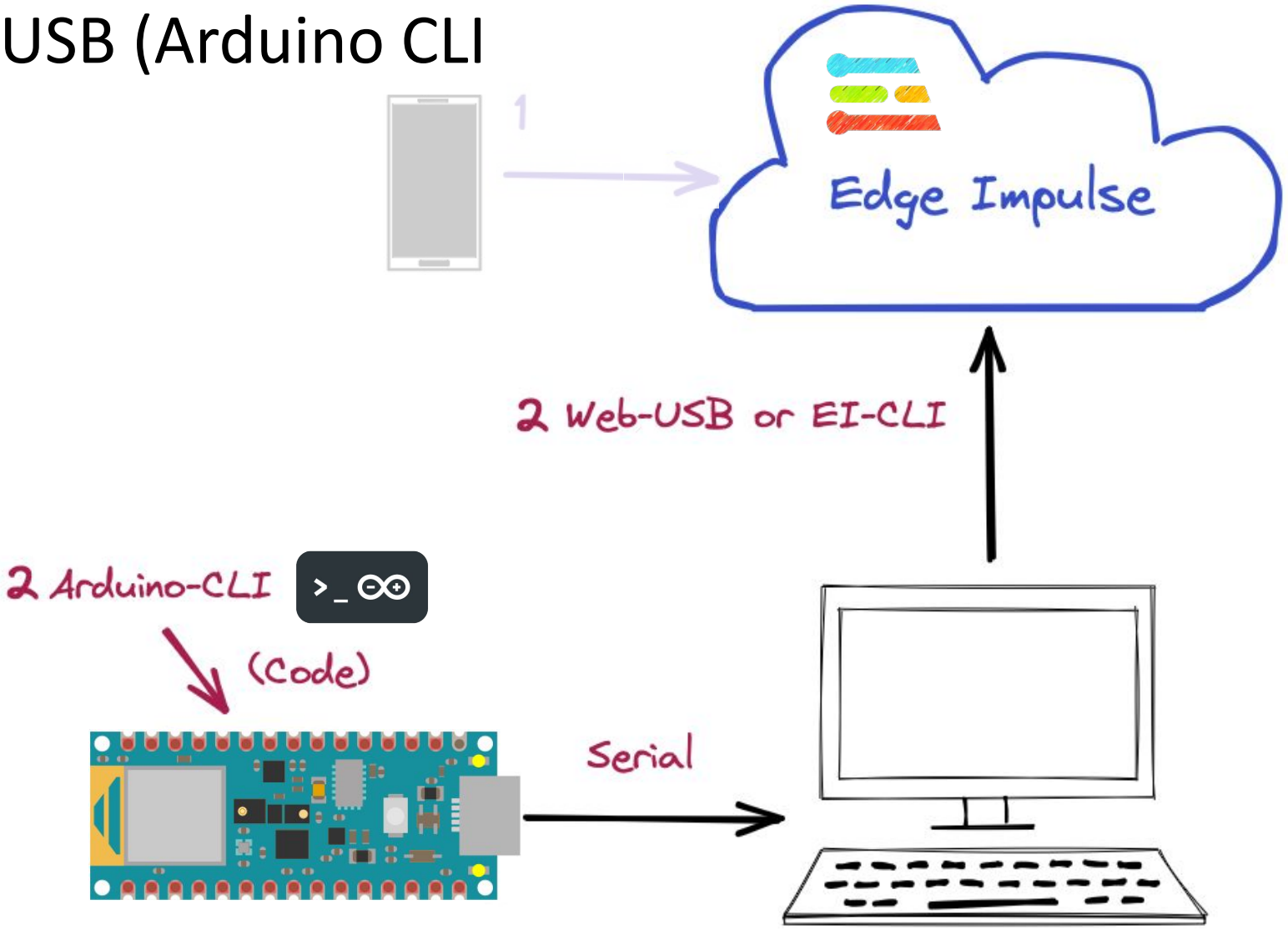






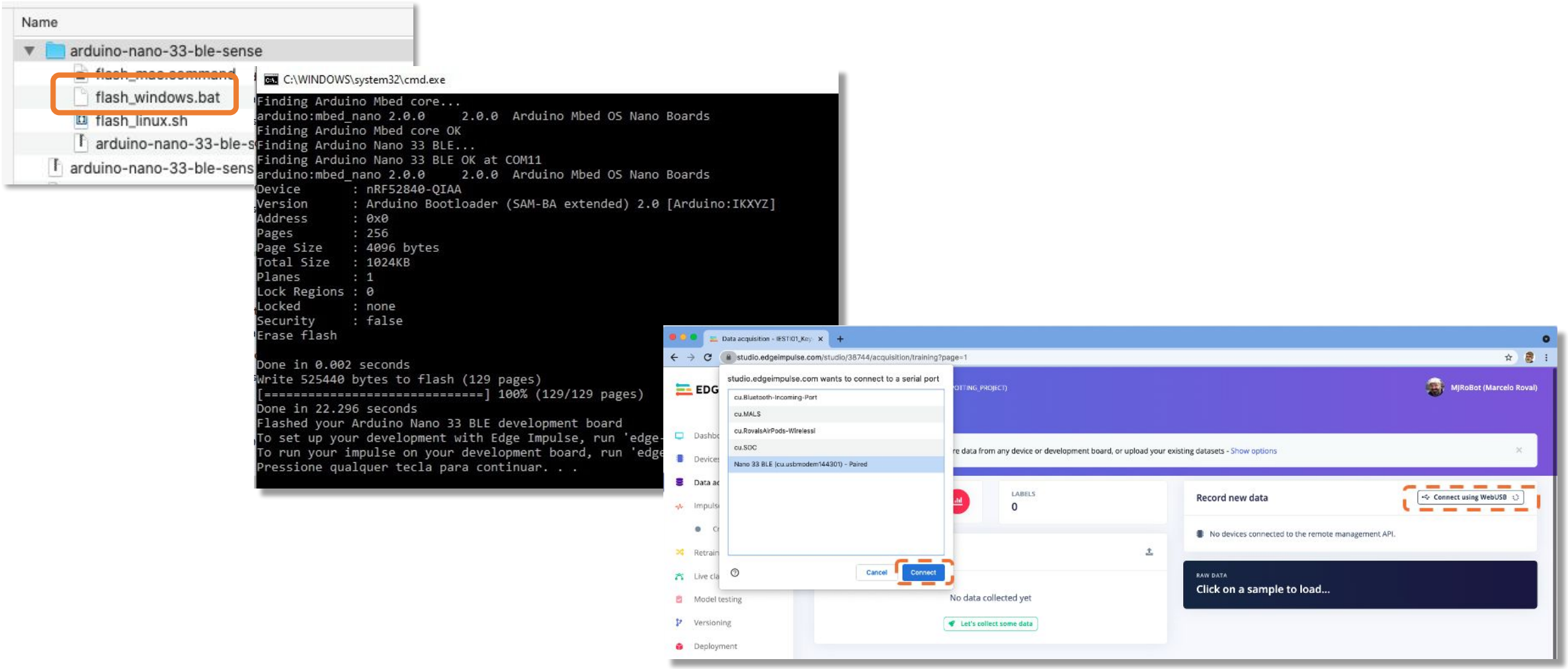


2. Web-USB (Arduino CLI

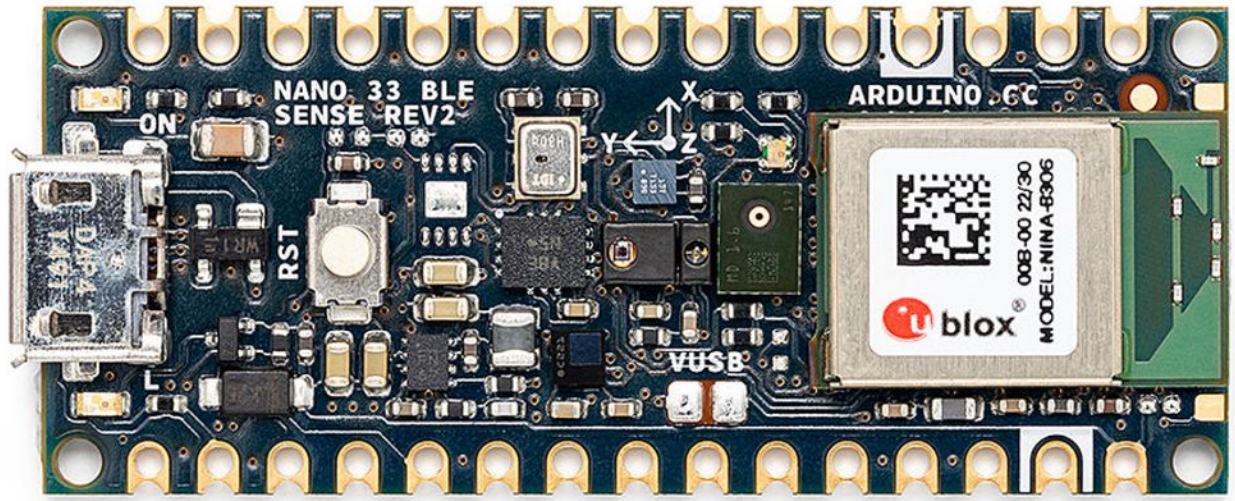


Issue: Limited MCU and sensors

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 ⓘ

36:17:55:F9:70:F7

Label

lift

Sample length (ms.)

10000

Sensor

Inertial

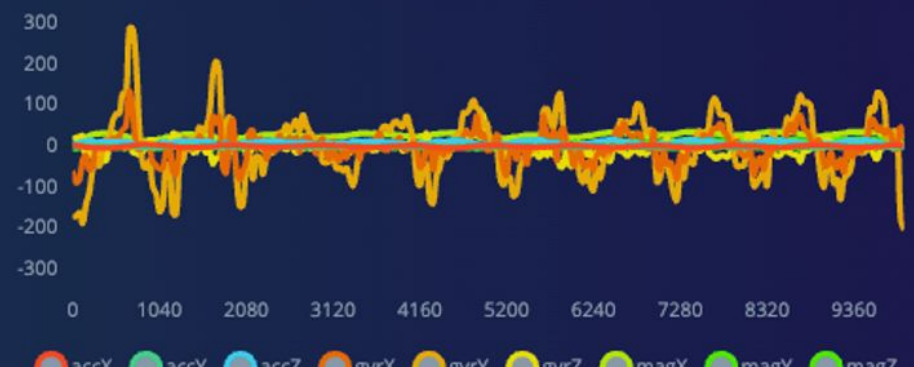
Frequency

62.5Hz

Start sampling

RAW DATA

lift.3soee1ar



- 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

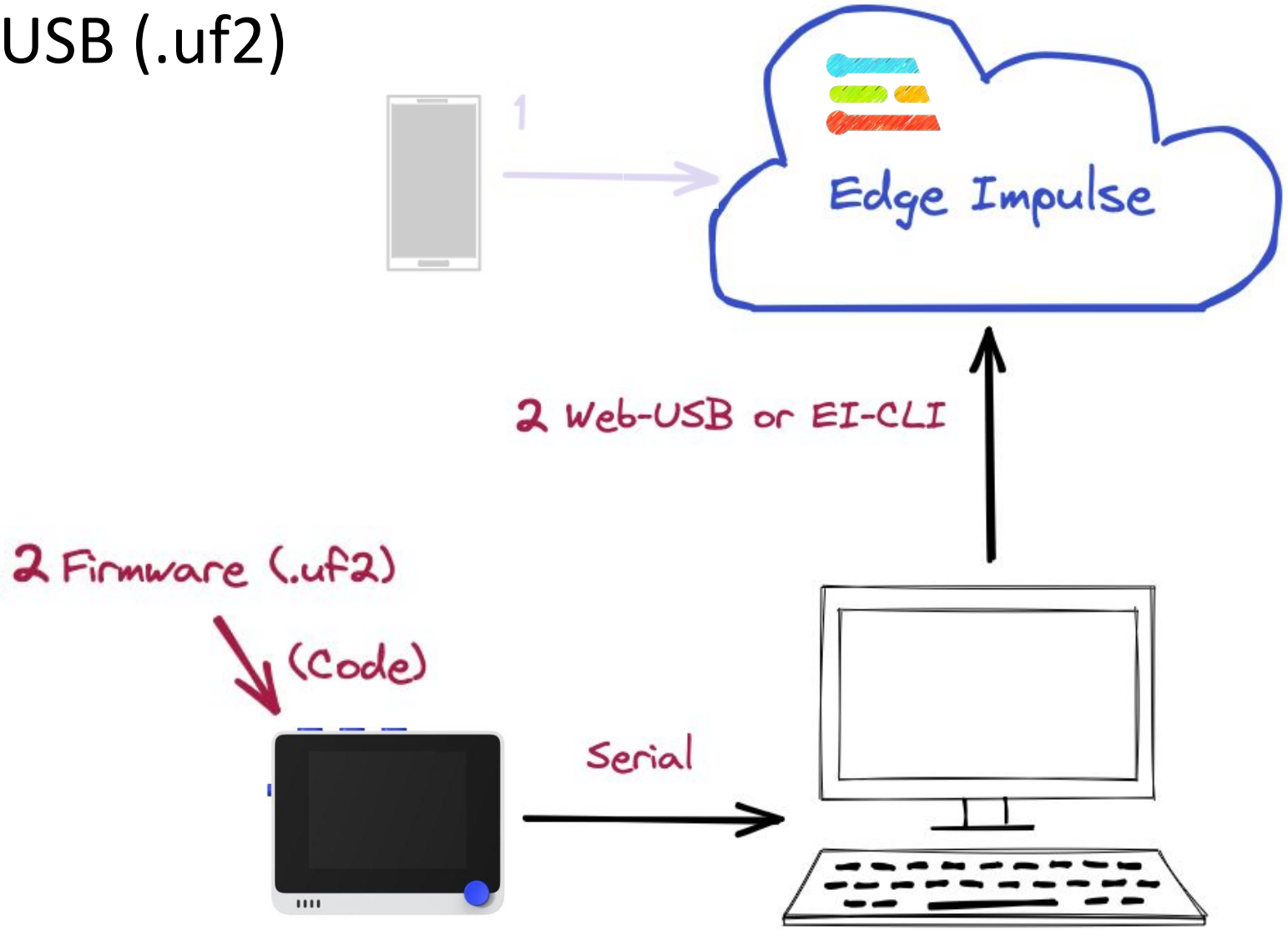
Spectral Analysis

Input axes (3)

- [x] accX
- [x] accY
- [x] accZ
- [ ] gyrX
- [ ] gyrY
- [ ] gyrZ
- [ ] magX
- [ ] magY
- [ ] magZ

Add a processing

2. Web-USB (.uf2)



Issue: Limited MCU and sensors

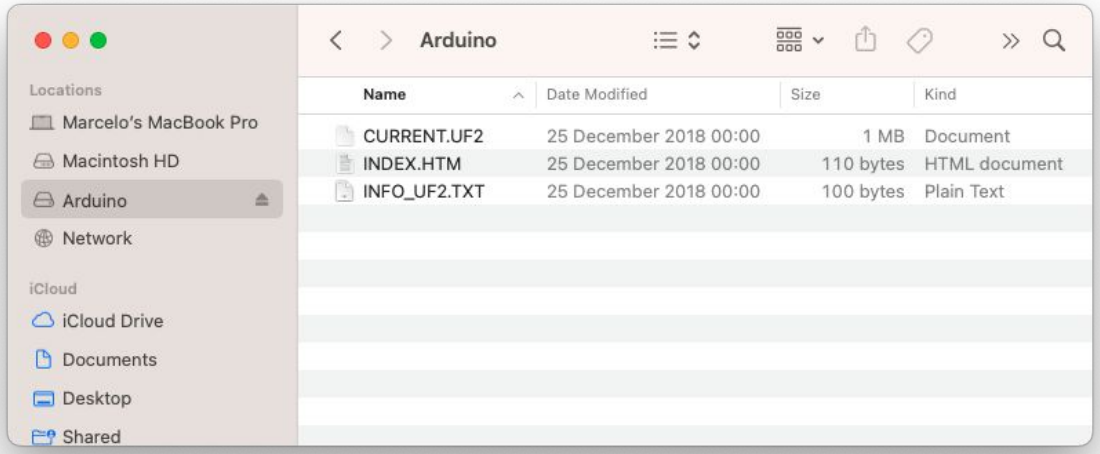
Wio Terminal
Installation & Tests

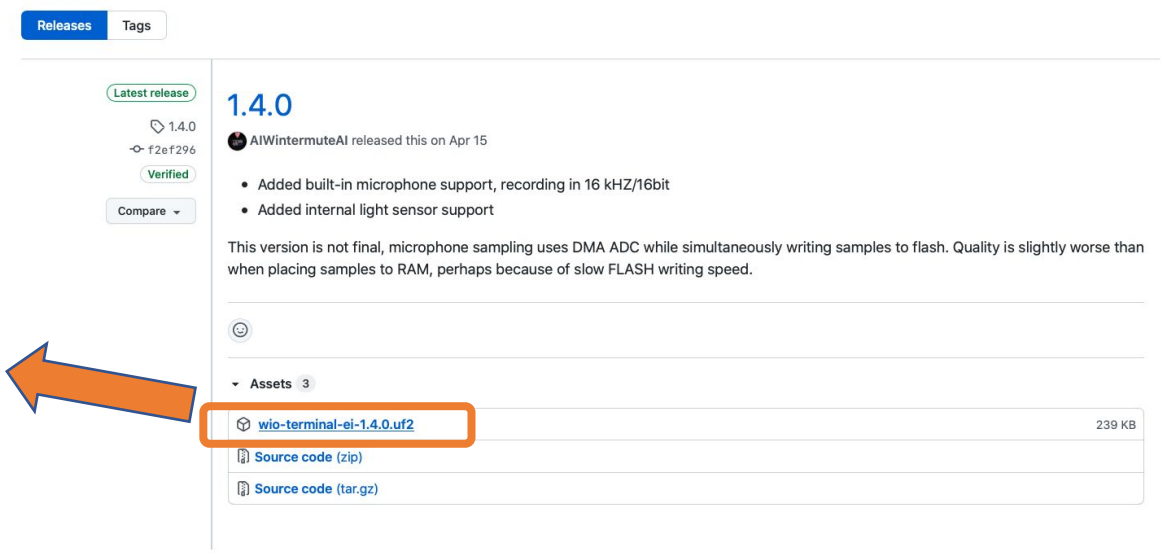
Prof. Marcelo José Rovai
UNIFEI - Federal University of Itajubá, Brazil
TinyML4D Academic Network Co-Chair

UNIFEI

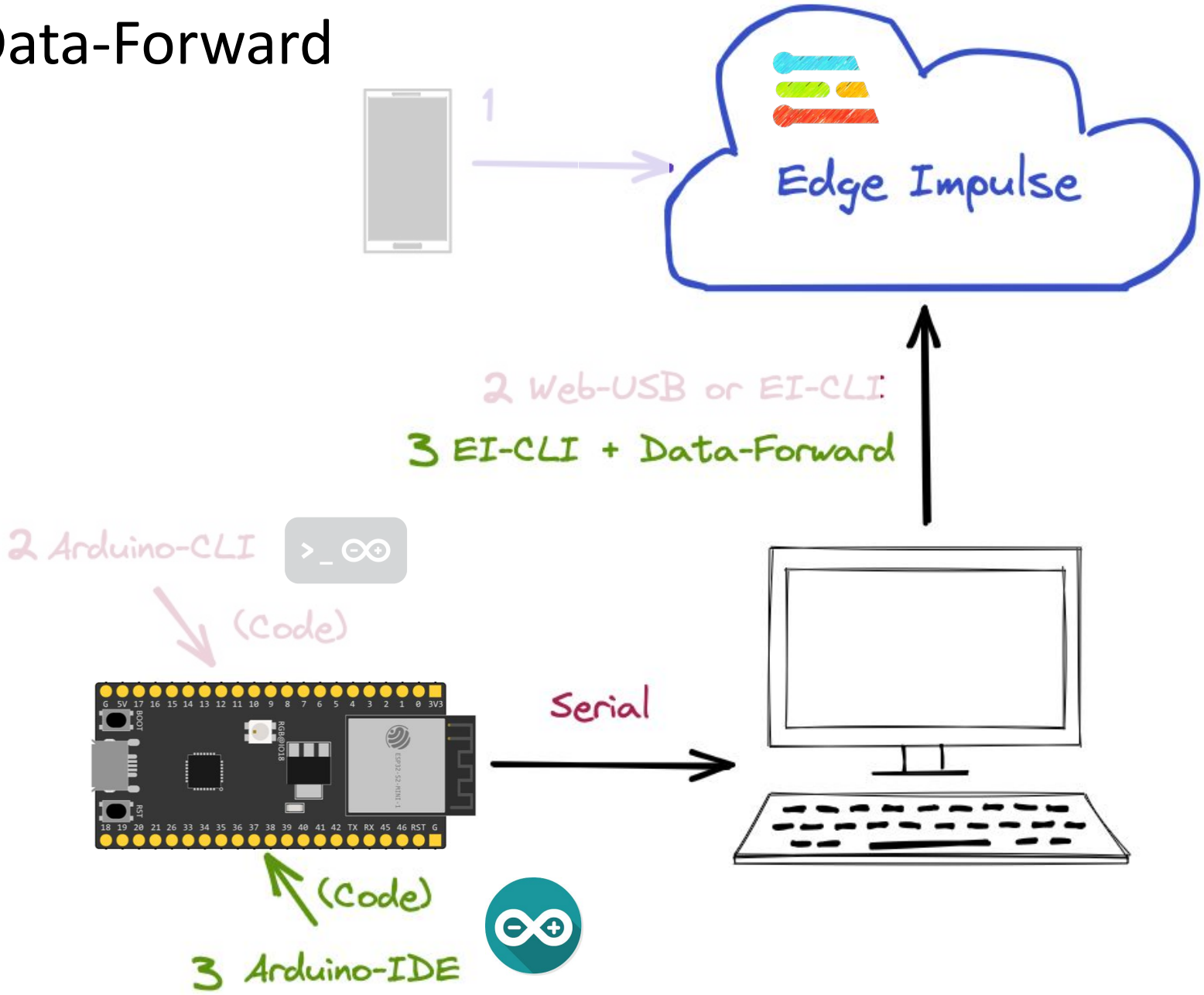
2. (.uf2) Firmware installation

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!

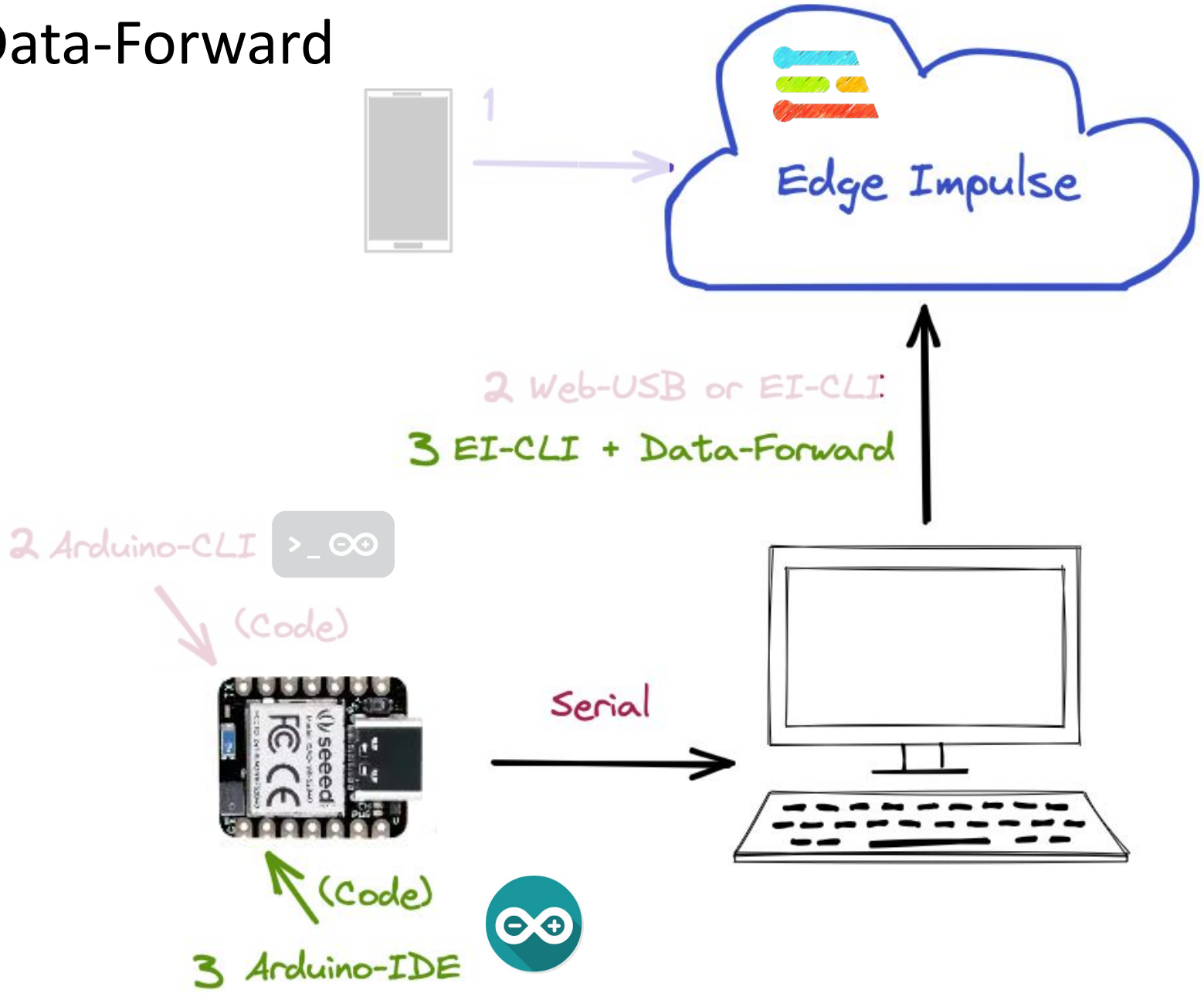




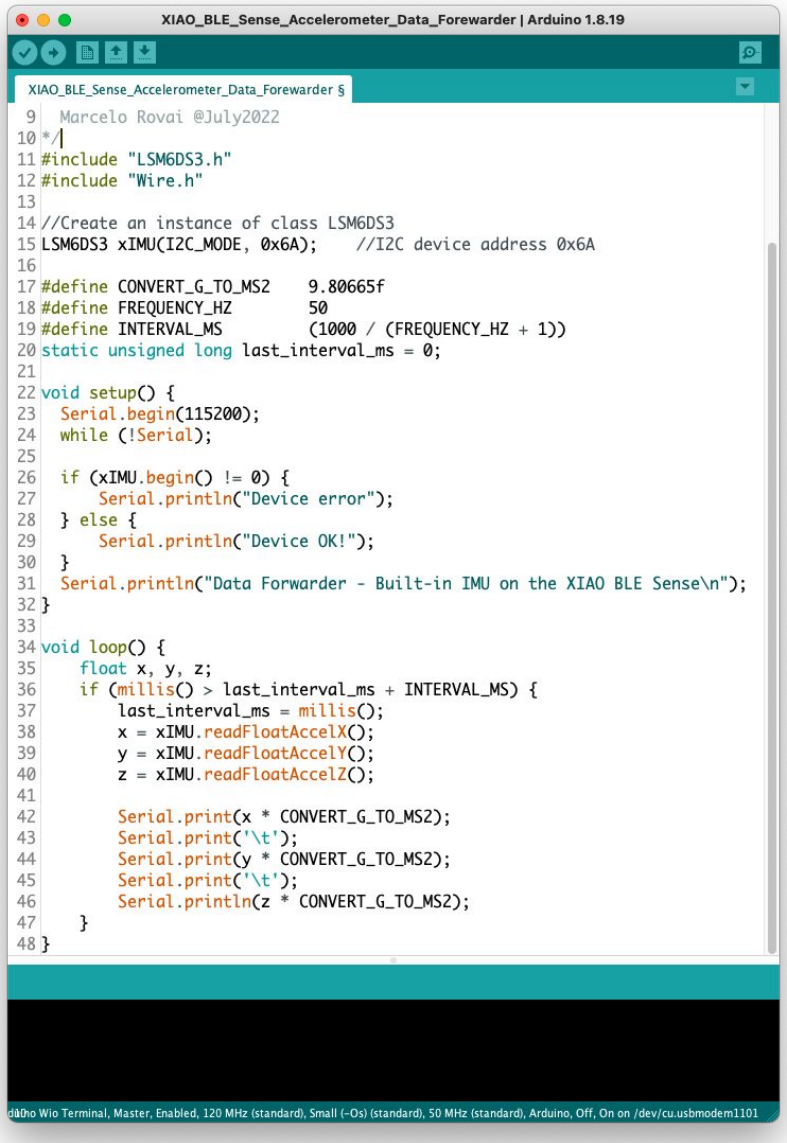
3. Data-Forward



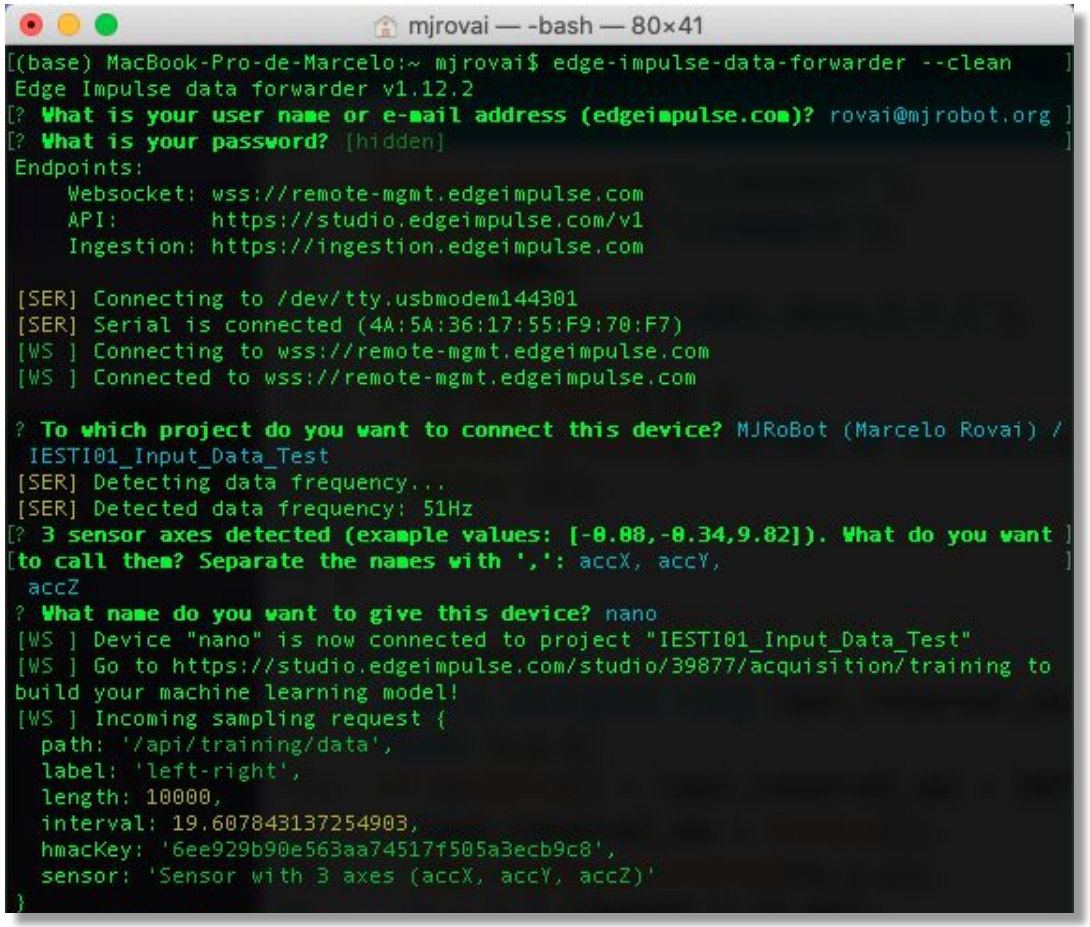
3. Data-Forward



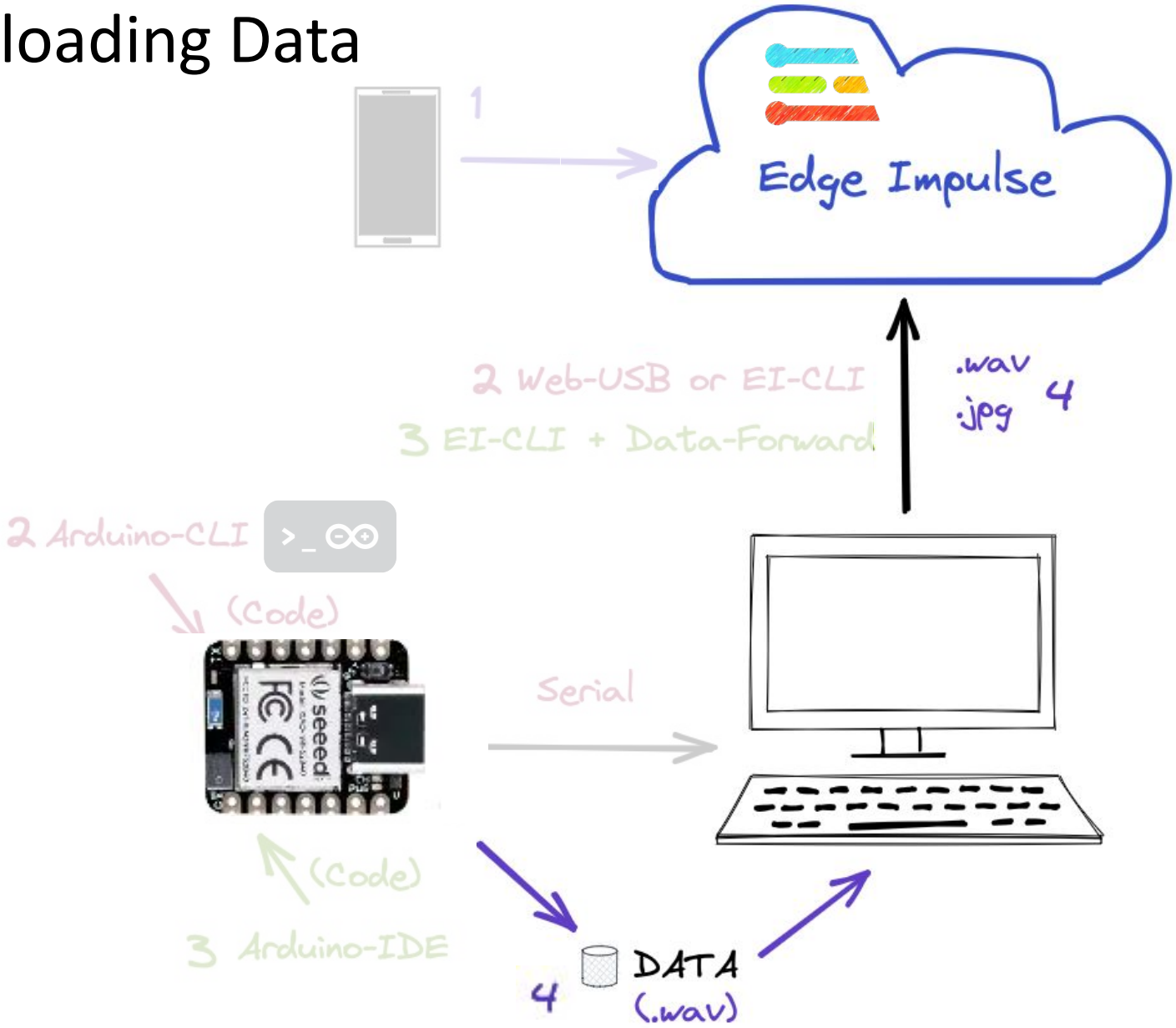
2. Data Ingestion using EI-Cli + Data Forward



$ edge-impulse-data-forwarder --clean

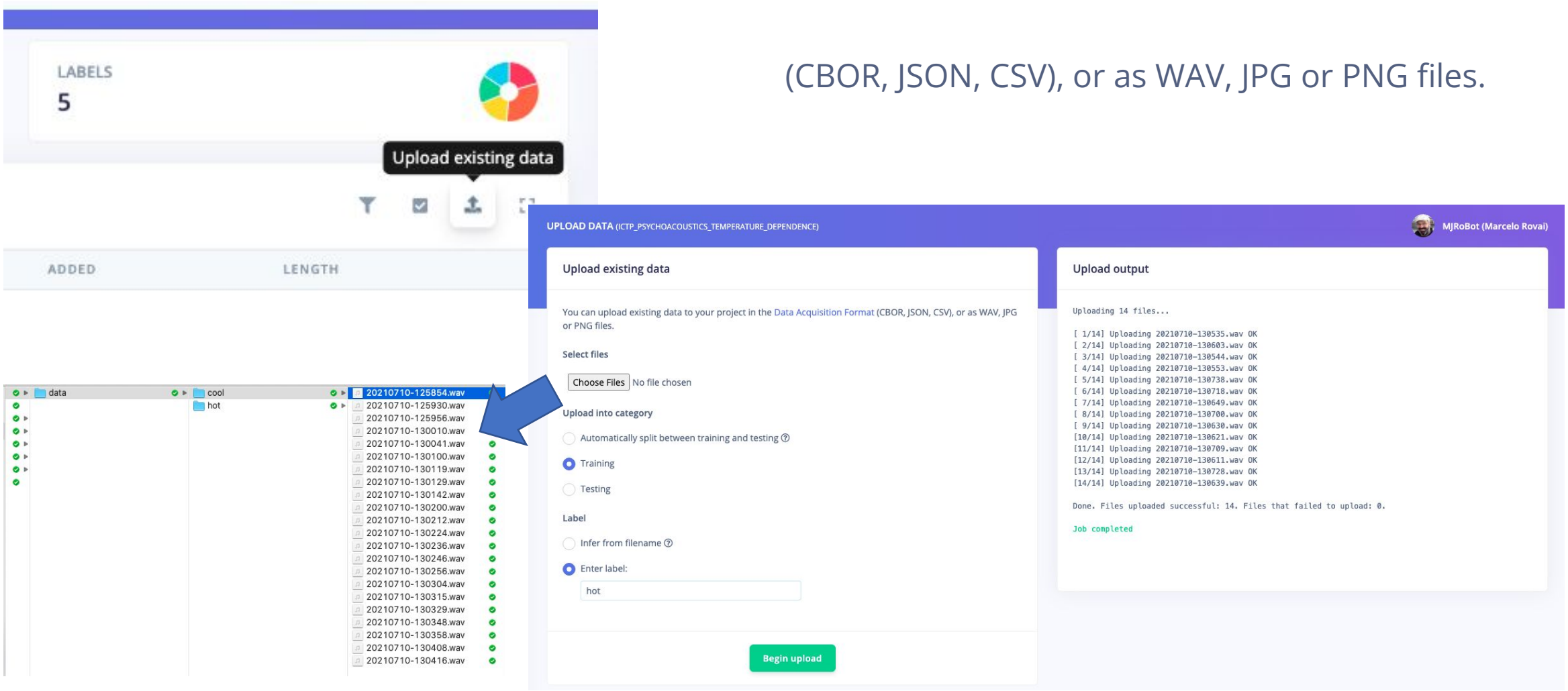


4. Uploading Data

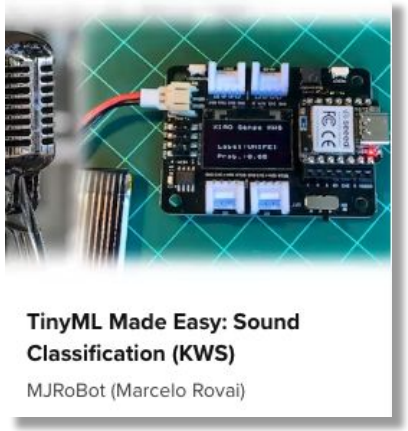


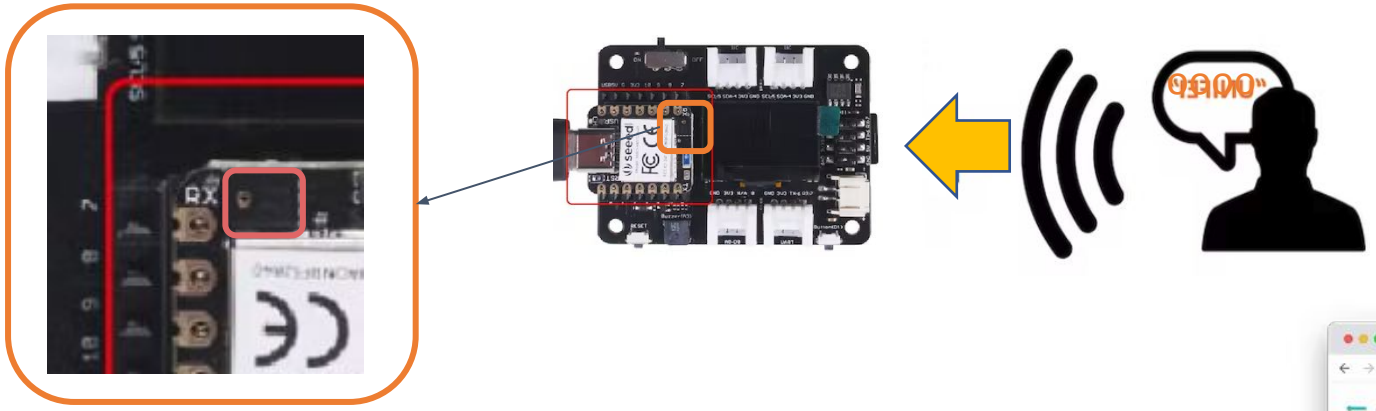
4. Data Ingestion using Upload existing Data

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

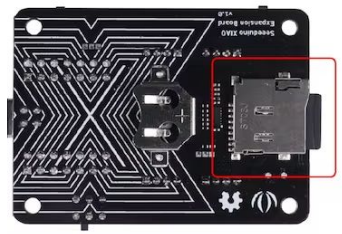


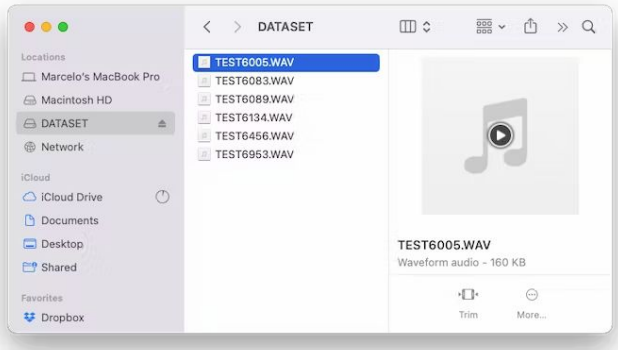
4. Uploading .wav data

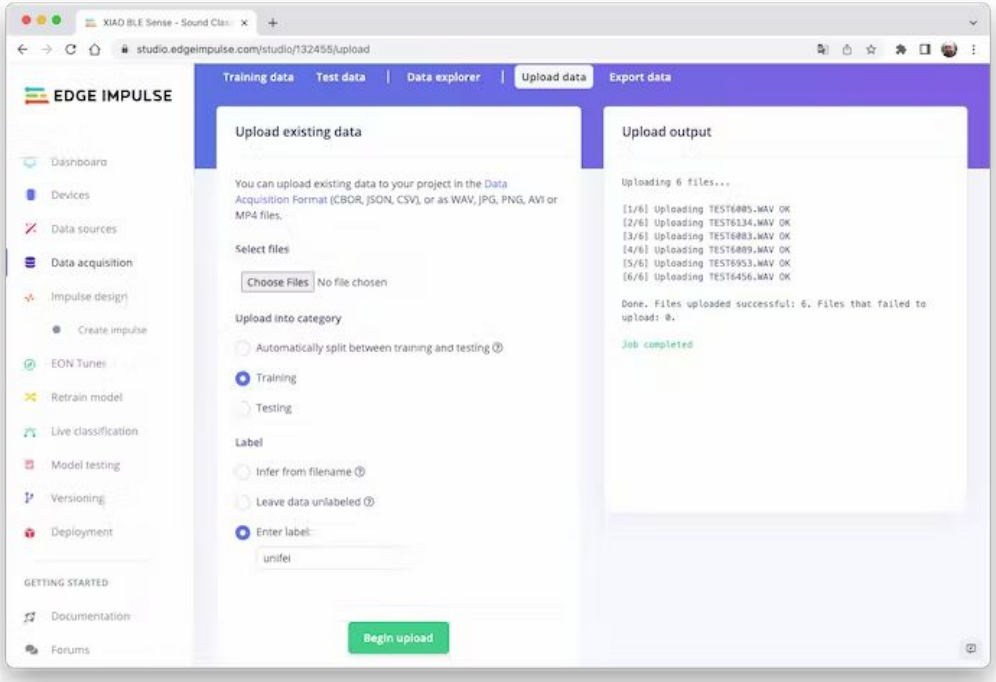




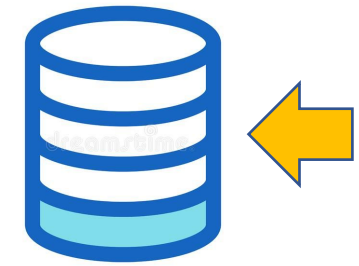
.wav

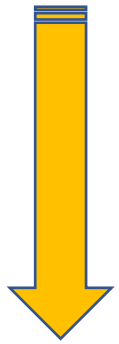




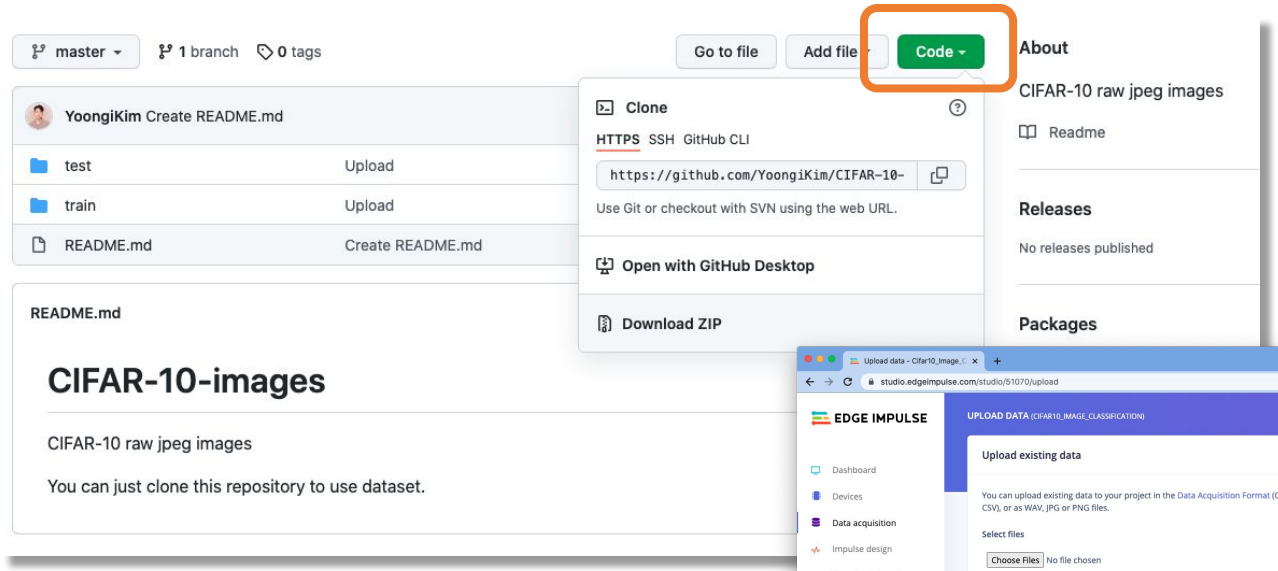


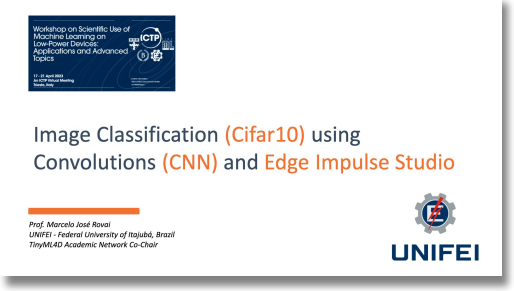
4. Uploading .jpg data

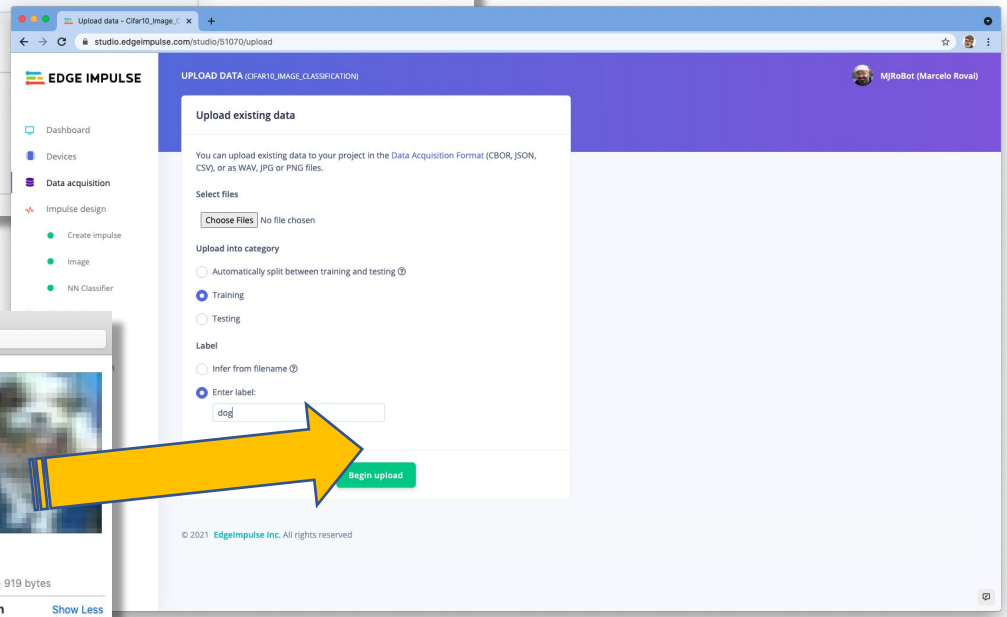


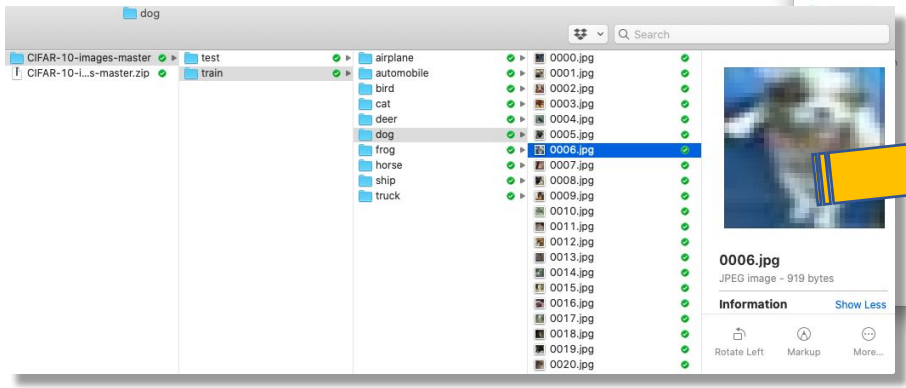


https://github.com/YoongiKim/CIFAR-10-images

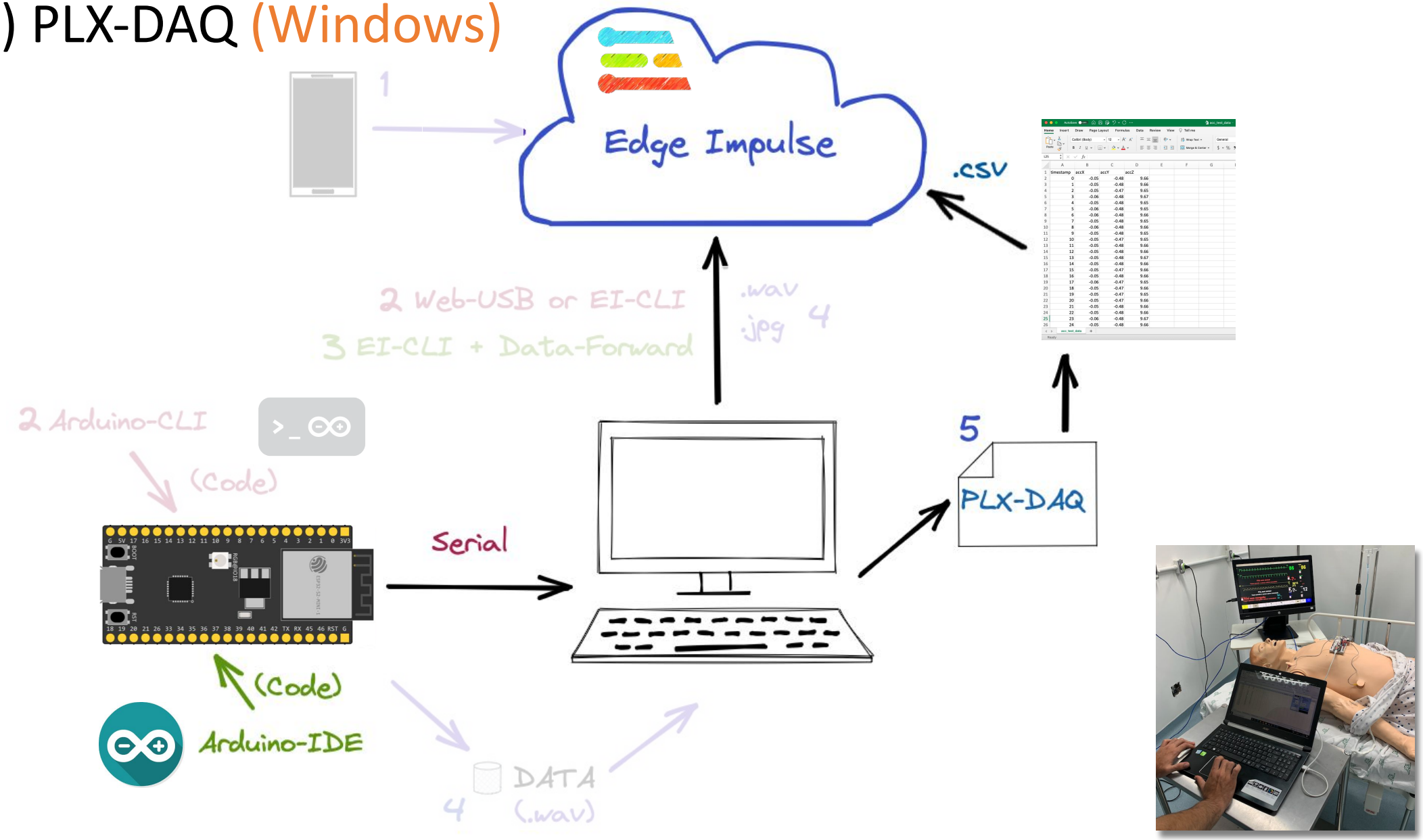




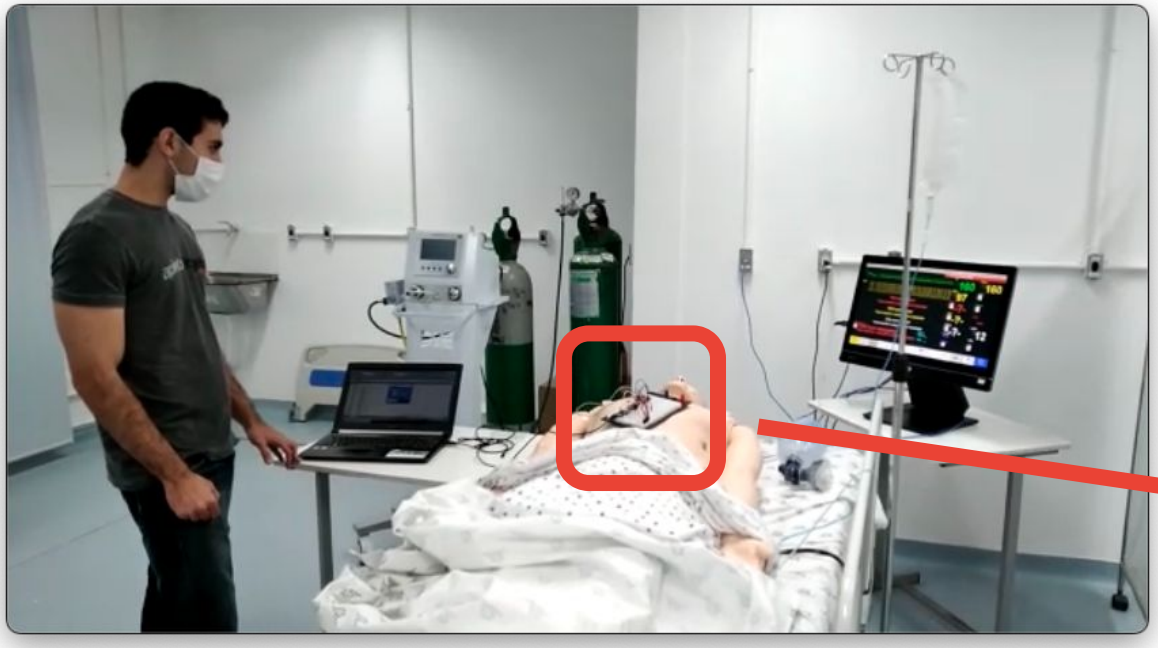


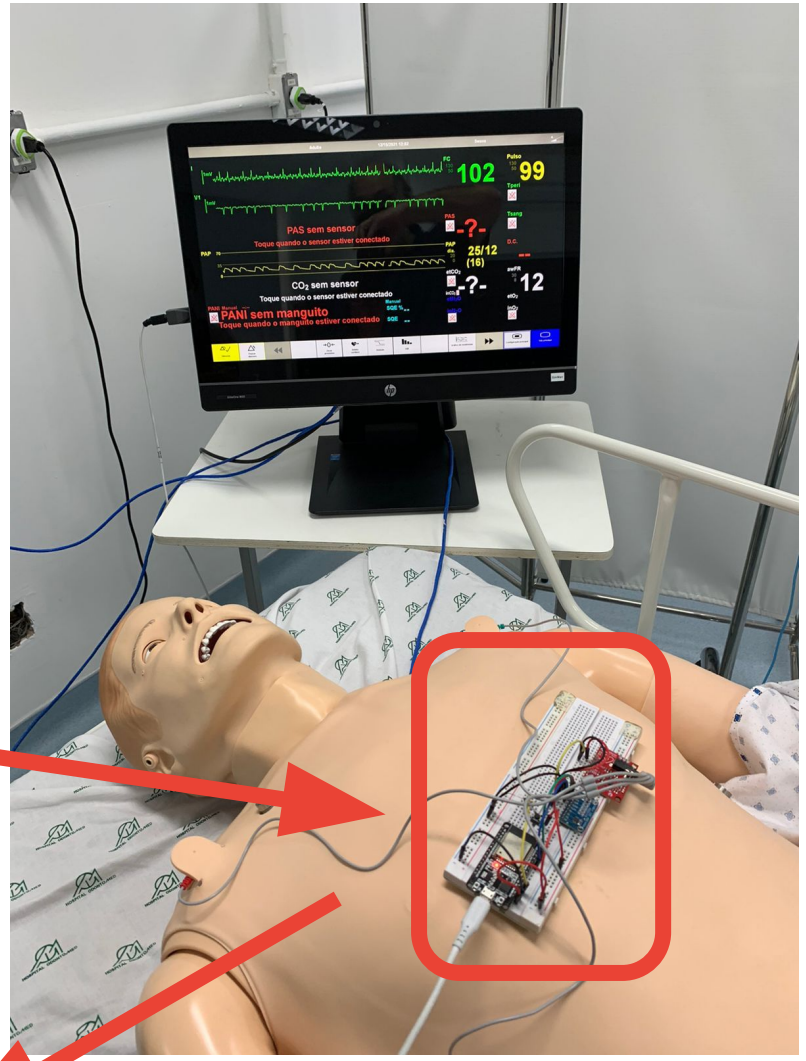


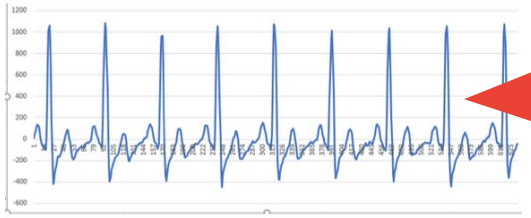
5. (.CSV) PLX-DAQ (Windows)

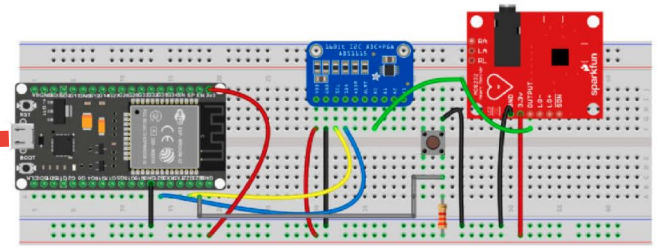


5. (.CSV) PLX-DAQ (Windows)





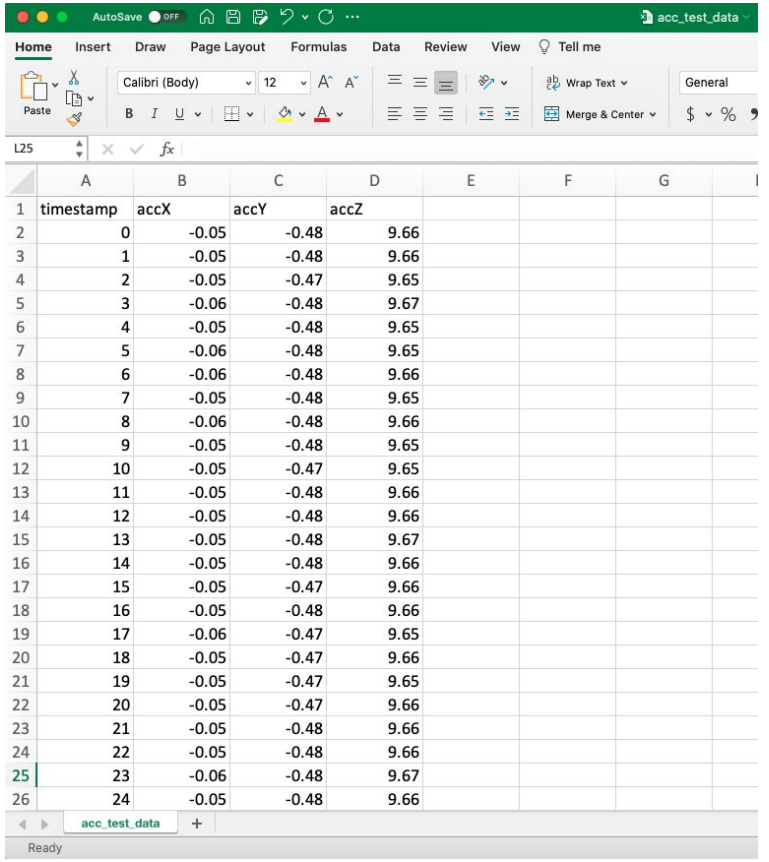




5. Data Ingestion using PLX-DAQ (Windows) => Final Format: .csv

Capture_Ardu33_Sense_IMU_Acc

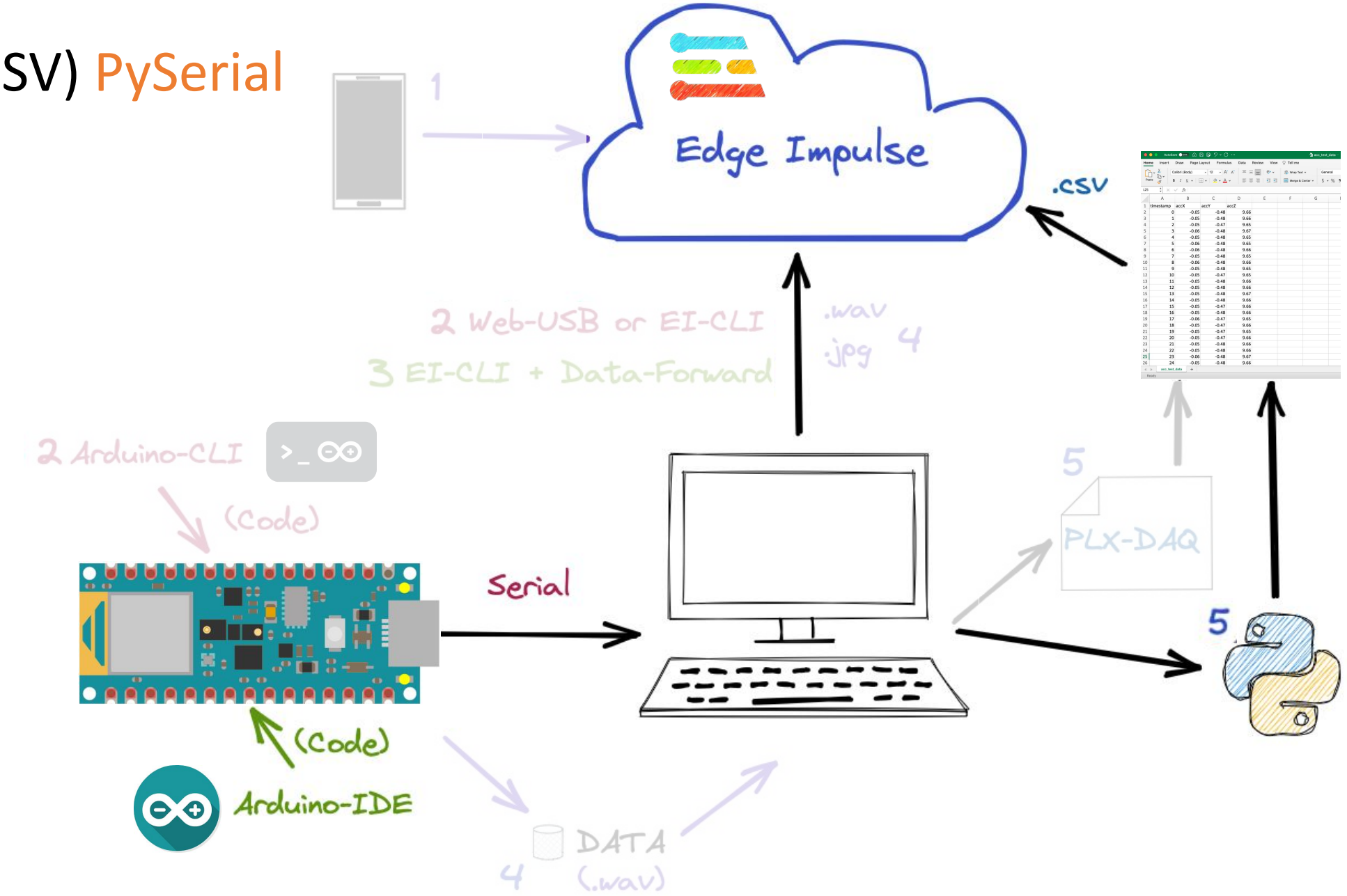
```ino
#include <Arduino_LSM9DS1.h>

#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

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

5. (.CSV) PySerial

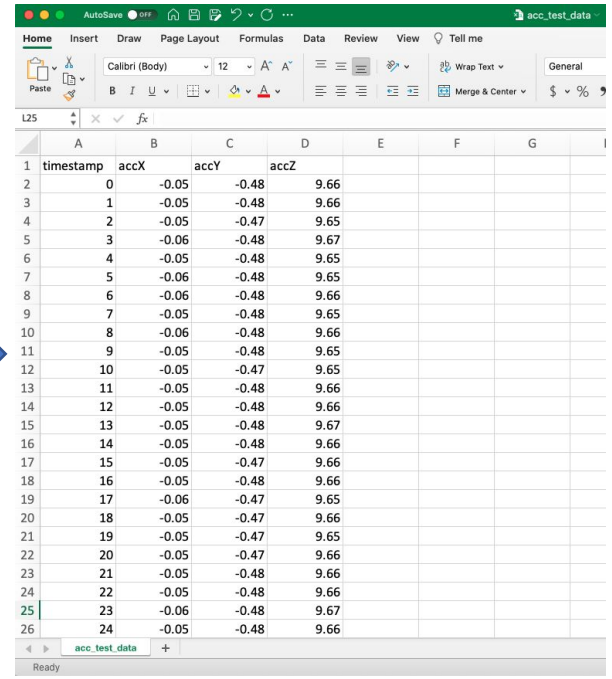


5. Data Ingestion using Python (PySerial) => Final Format: .csv

Capture_Ardu33_Sense_IMU_Acc

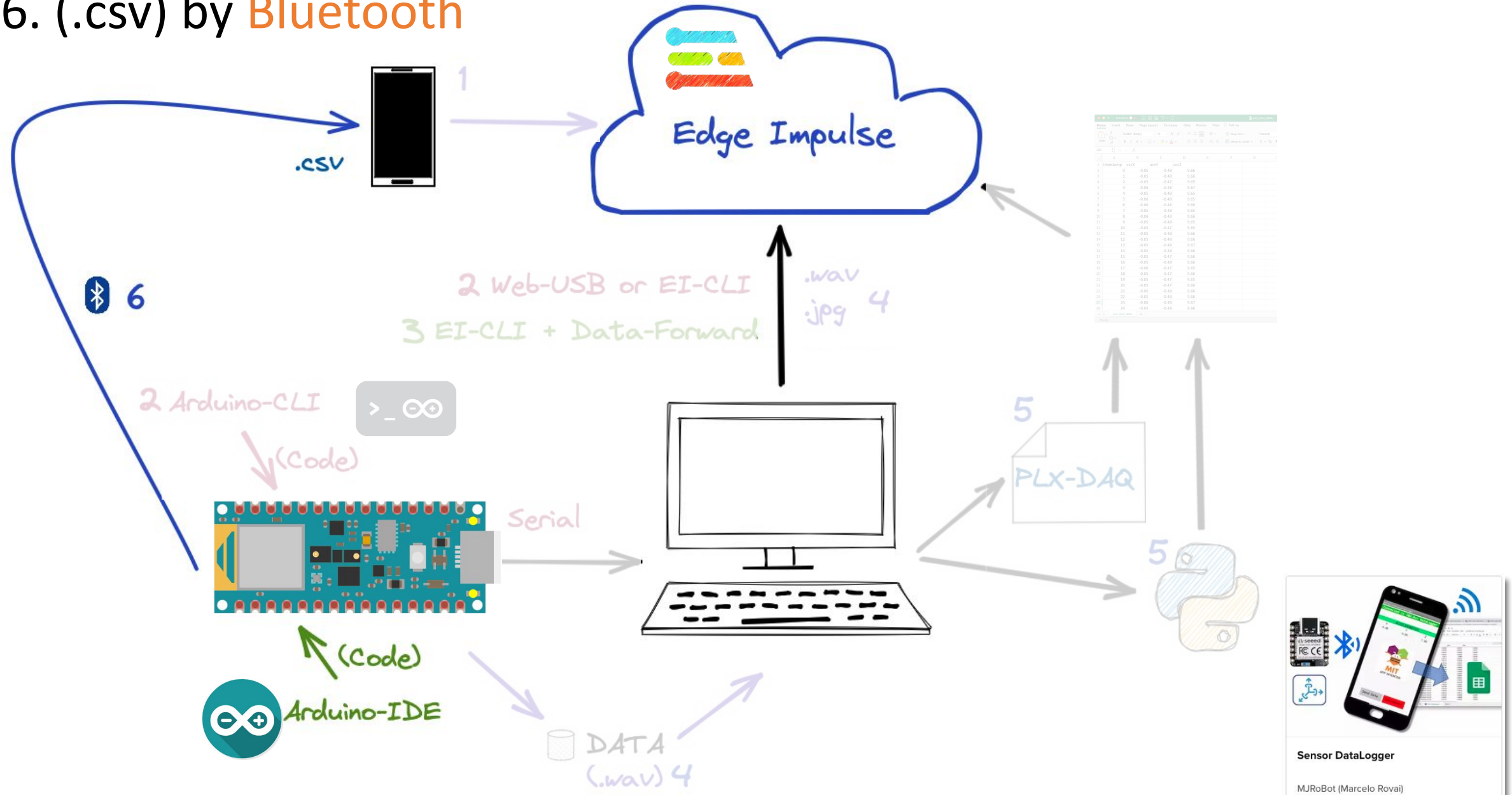
```ino
#include <Arduino_LSM9DS1.h>

#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()
```

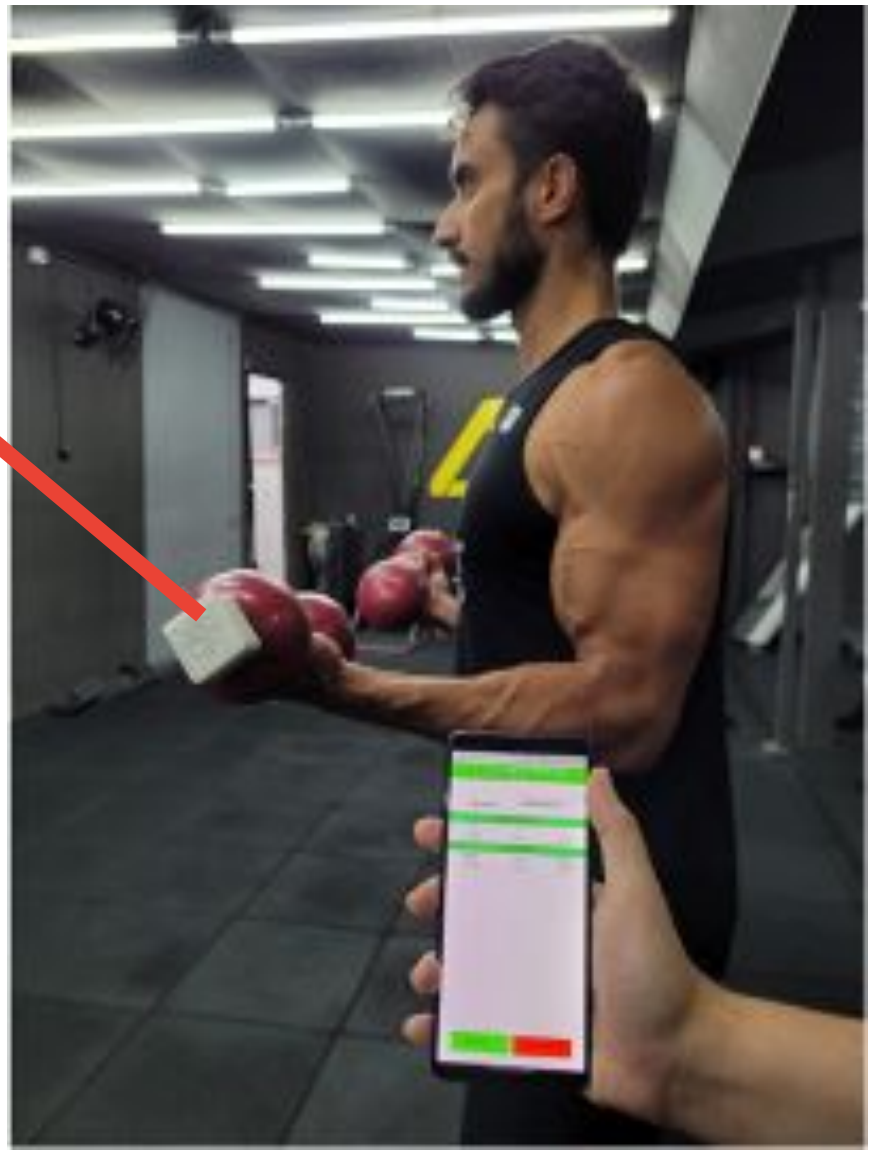


timestamp	accX	accY	accZ
0	-0.05	-0.48	9.66
1	-0.05	-0.48	9.66
2	-0.05	-0.47	9.65
3	-0.06	-0.48	9.67
4	-0.05	-0.48	9.65
5	-0.06	-0.48	9.65
6	-0.06	-0.48	9.66
7	-0.05	-0.48	9.65
8	-0.06	-0.48	9.66
9	-0.05	-0.48	9.65
10	-0.05	-0.47	9.65
11	-0.05	-0.48	9.66
12	-0.05	-0.48	9.66
13	-0.05	-0.48	9.67
14	-0.05	-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

6. (.csv) by Bluetooth



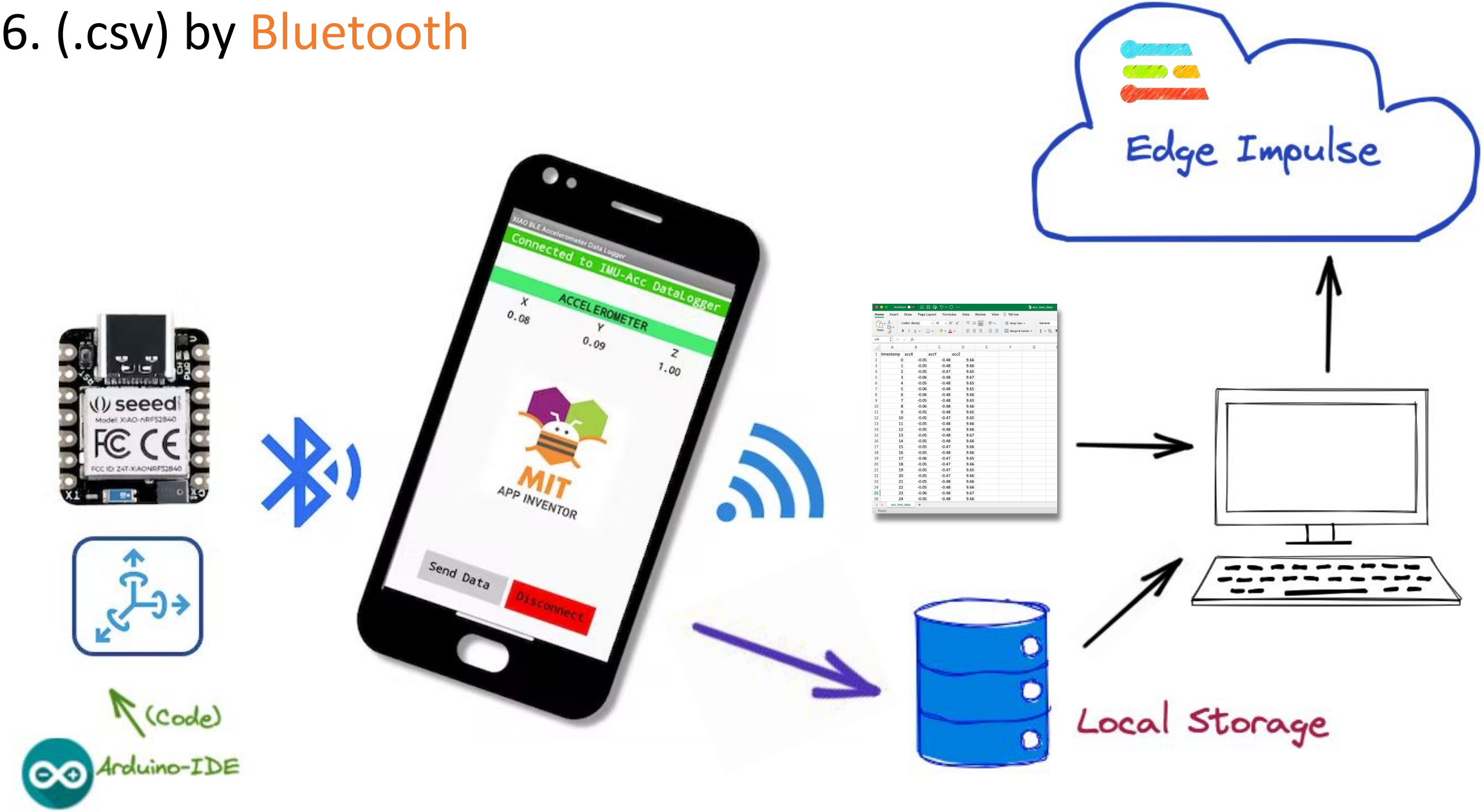
6. (.csv) by Bluetooth



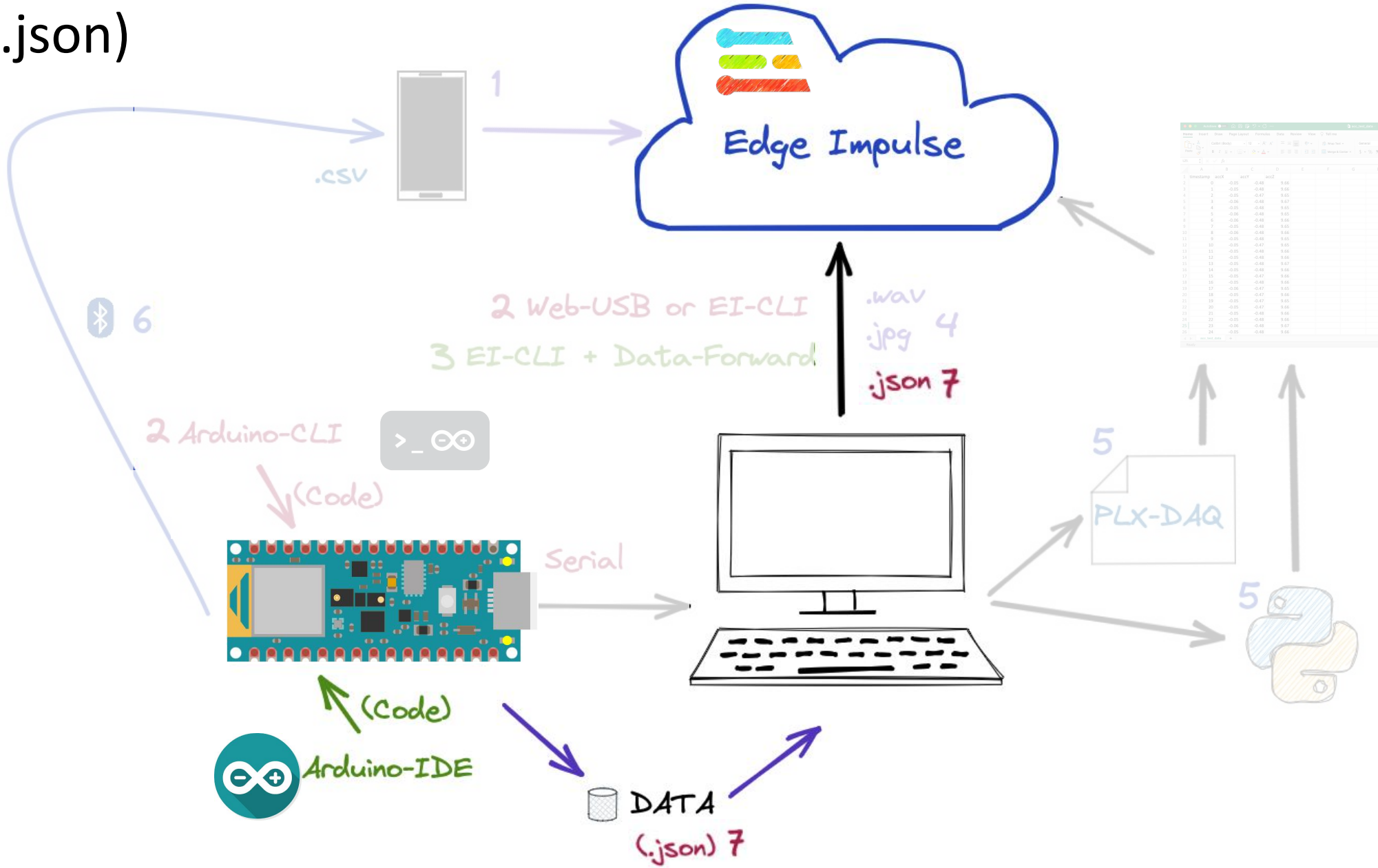
CNPq

UNIFEI

6. (.csv) by Bluetooth



7. (.json)


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

7. Raw Uploader (.json files)

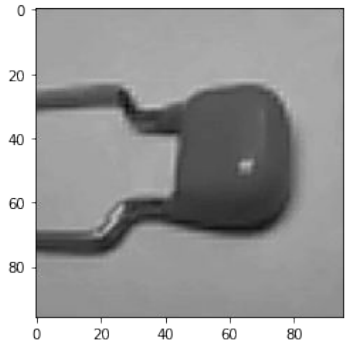
Image Classification: Raw Uploader

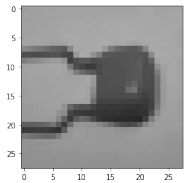
Open in Colab

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

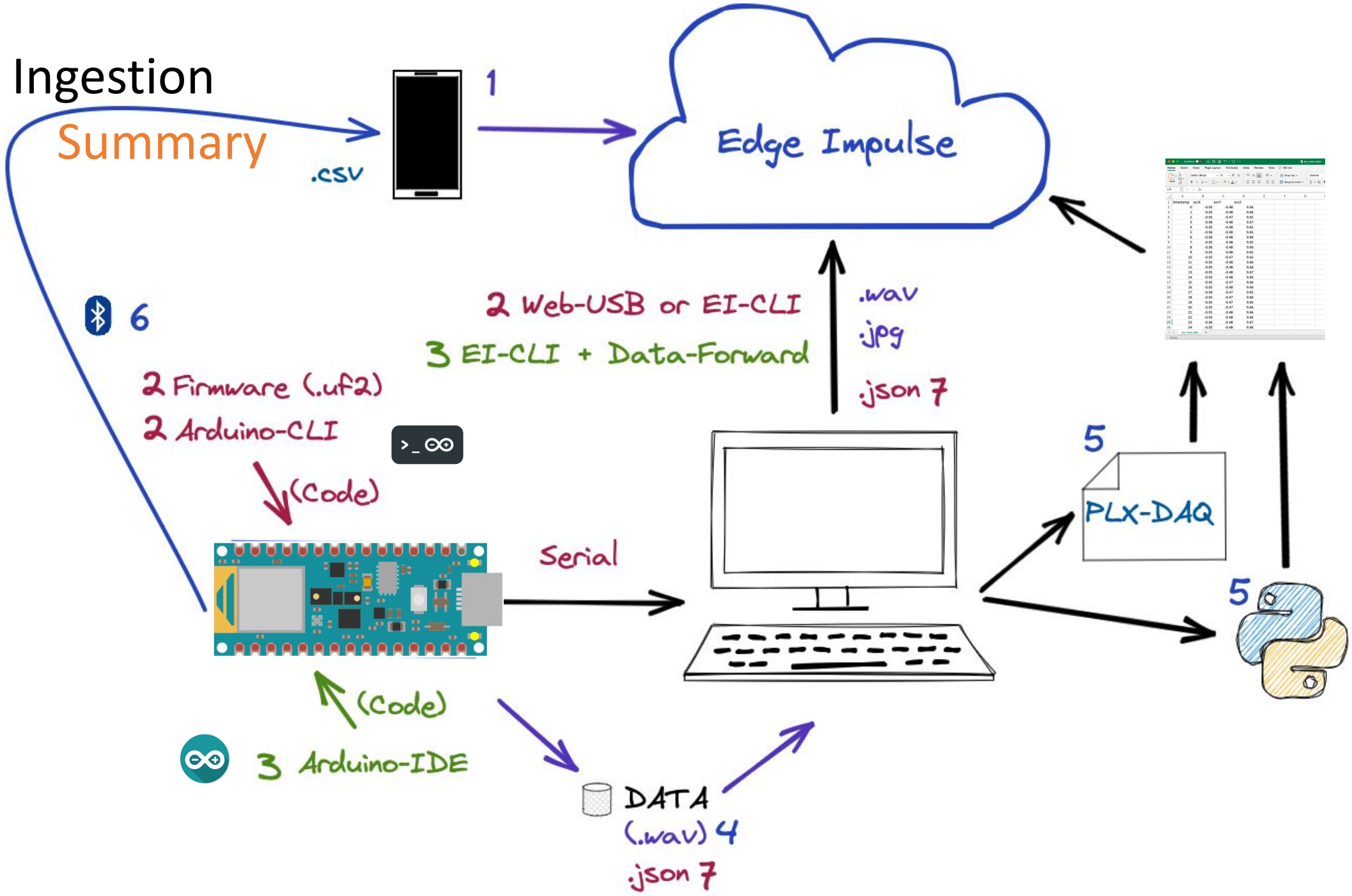
EDGE IMPULSE

(Training as DNN)

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

Data Ingestion

Summary



Thanks

UNIFEI