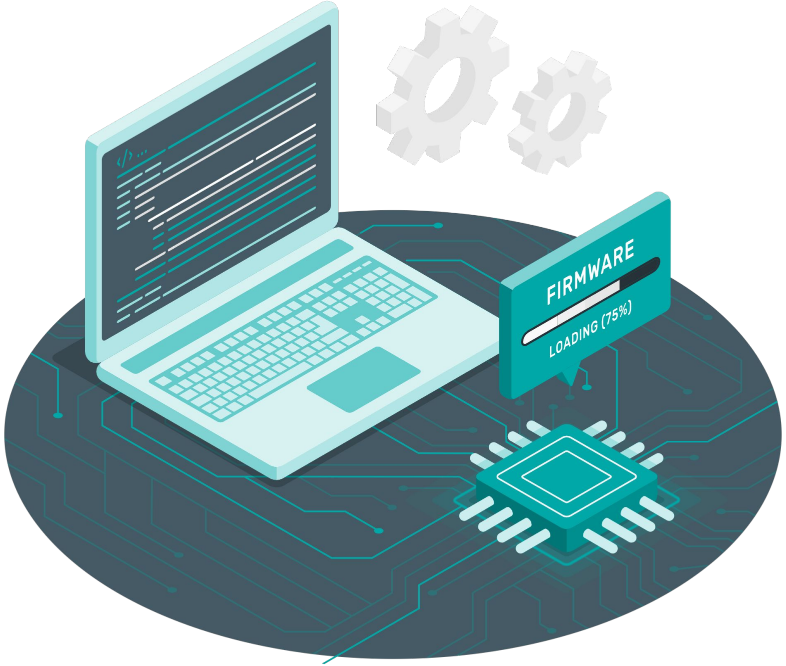
University Politehnica of Bucharest
Computer Science & Engineering Department
Embedded Systems Laboratory
Wireless Sensor Networks, 2022

Air Quality Monitoring Using Edge Impulse

Coordinating Professor: Dan Ștefan Tudose
Student: Alexandra Covor, ACES



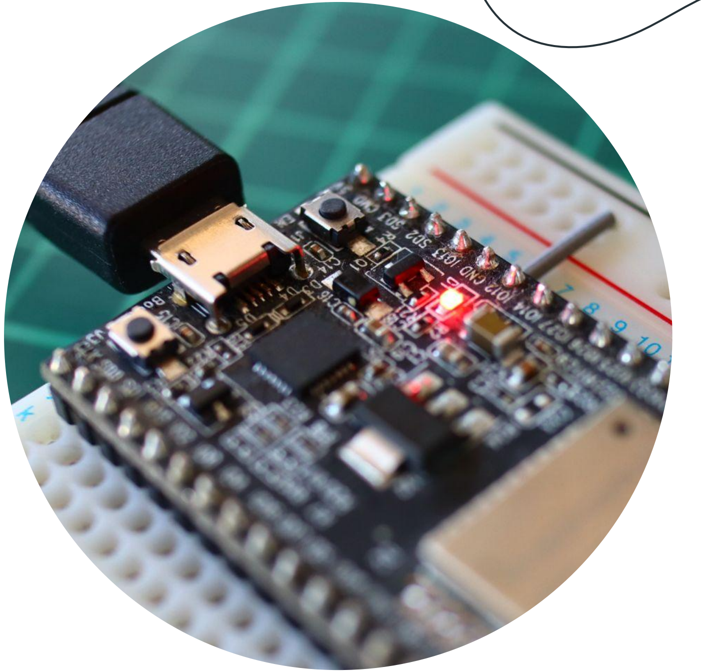
Contents

1 Introduction

2 Hardware Description

3 Software Description

4 Conclusions



Introduction

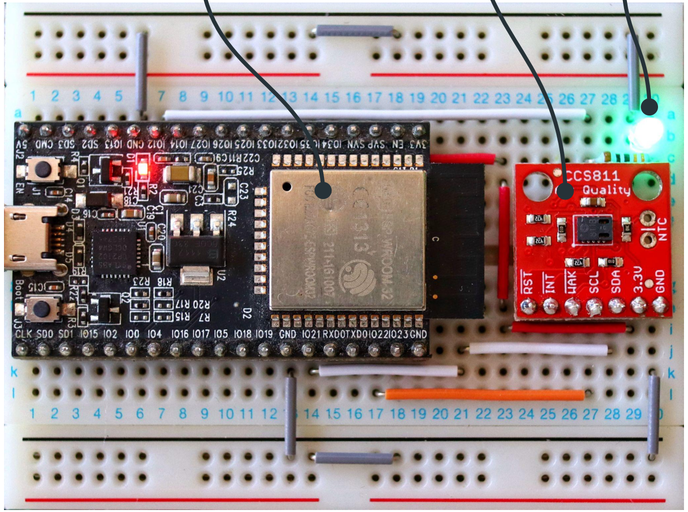
- Identify anomalies in air quality data
- Embedded Machine Learning
- ESP32 development board
- Set up the WiFi connection through a web page
- Dashboard displayed on a local web server

Hardware Description

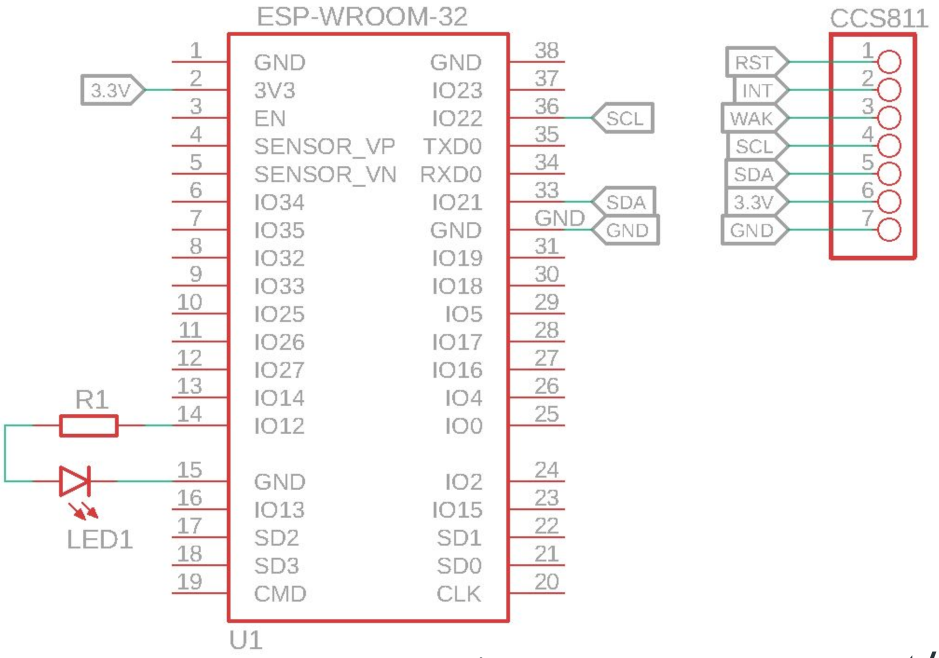
CCS811 Air Quality Sensor

ESP32 Development Board

LED



Circuit schematic drawn in Eagle CAD



Software Description

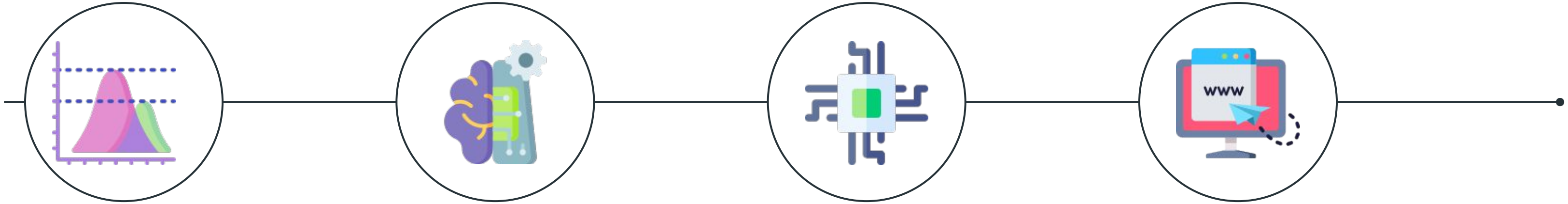
Tools and Platforms Used

ARDUINO™

EDGE IMPULSE

Software Description

Development Steps



1 Collecting the data

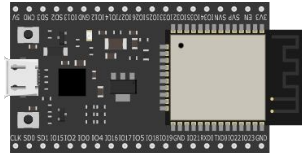
2 Creating & training the ML model

3 Deploying the model

4 Integrating the model with the web server

Software Description

1. Collecting the data



Remote management protocol

Websocket connection:

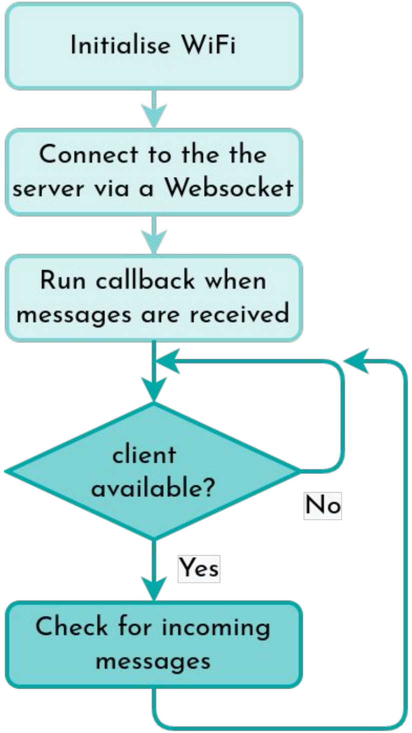
ws://remote-mgmt.edgeimpulse.com

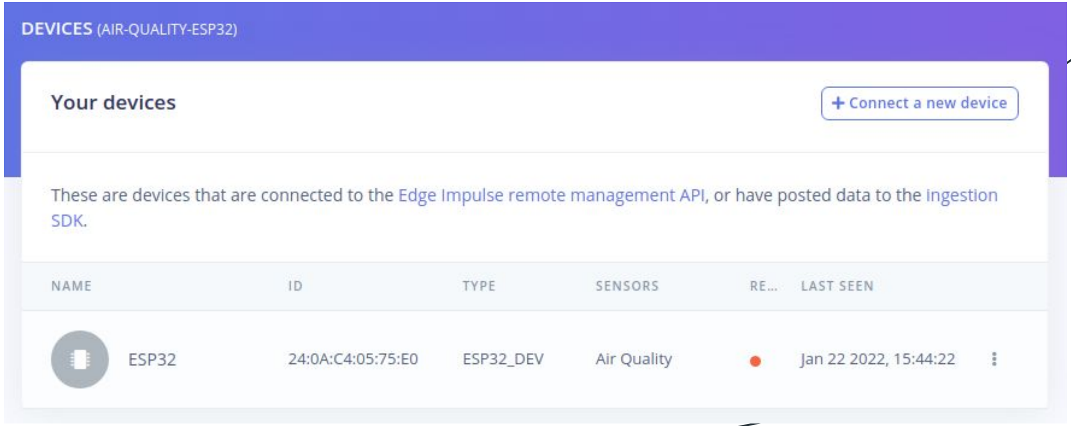
Hello request:

```
{
    "hello": {
        "version": 3,
        "apiKey": "<API_KEY>",
        "deviceId": "<DEVICE_ID>",
        "deviceType": "ESP32_DEV",
        "connection": "ip",
        "sensors": [{
            "name": "Air Quality",
            "frequencies": [0],
            "maxSampleLengthS": 60000
        }],
        "supportsSnapshotStreaming": false
    }
  }
```

Software Description

1. Collecting the data


Initialise WiFi
Connect to the the server via a Websocket
Run callback when messages are received
client available?
No
Yes
Check for incoming messages


DEVICES (AIR-QUALITY-ESP32)

Your devices

These are devices that are connected to the Edge Impulse remote management API, or have posted data to the ingestion SDK.

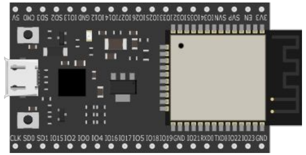
NAME	ID	TYPE	SENSORS	RE...	LAST SEEN
ESP32	24:0A:C4:05:75:E0	ESP32_DEV	Air Quality	●	Jan 22 2022, 15:44:22

Software Description

1. Collecting the data

- Edge Impulse Data acquisition format
- Encoded using CBOR
- Signed with an HMAC key
- Payload header:

```
sensor_aq_payload_info payload = {
    "24:0A:C4:05:75:E0",
    "ESP32_DEV",
    1 / SAMPLE_TIME,
    { { "CO2", "ppm" }, { "TVOC", "ppb" } }
  };
```

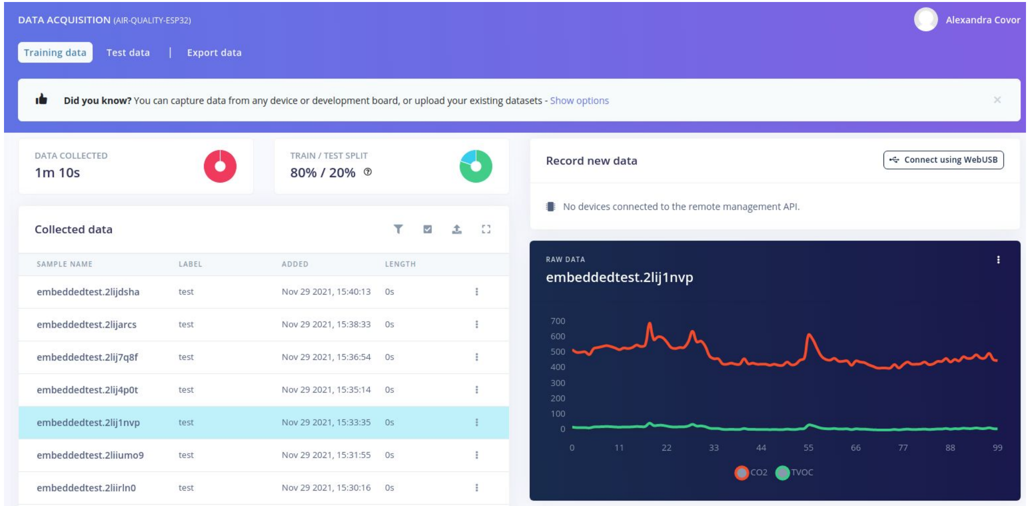


HTTP POST request to:

http://ingestion.edgeimpulse.com/api/training/data

Software Description

1. Collecting the data



DATA ACQUISITION (AIR-QUALITY-ESP32)

Alexandra Covor

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

DATA COLLECTED
1m 10s

TRAIN / TEST SPLIT
80% / 20%

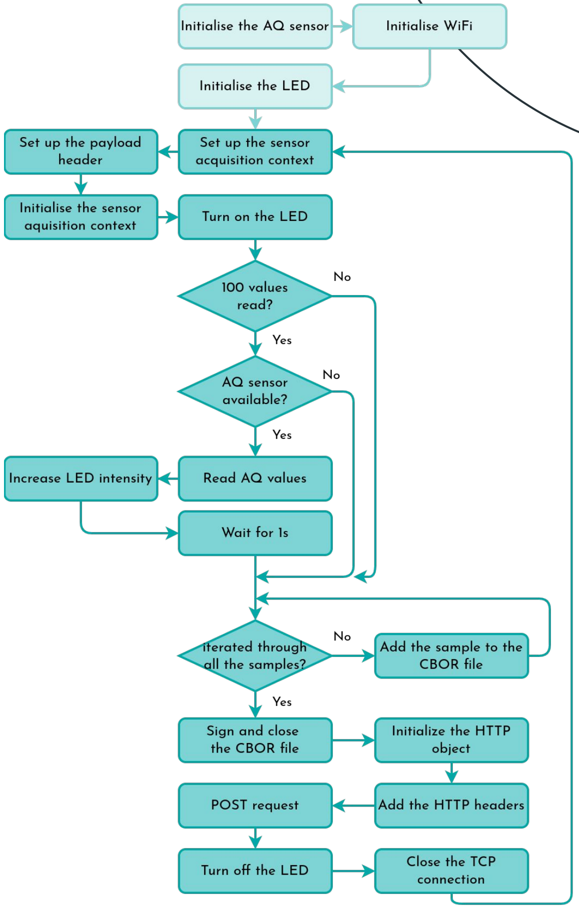
Record new data

No devices connected to the remote management API.

Collected data

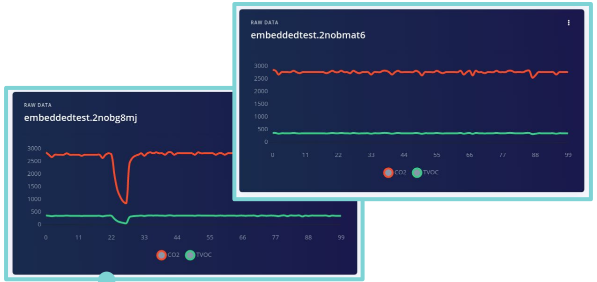
SAMPLE NAME	LABEL	ADDED	LENGTH
embeddedtest.2lijdsha	test	Nov 29 2021, 15:40:13	0s
embeddedtest.2lijarcs	test	Nov 29 2021, 15:38:33	0s
embeddedtest.2lij7q8f	test	Nov 29 2021, 15:36:54	0s
embeddedtest.2lij4p0t	test	Nov 29 2021, 15:35:14	0s
embeddedtest.2lij1nvp	test	Nov 29 2021, 15:33:35	0s
embeddedtest.2liiumo9	test	Nov 29 2021, 15:31:55	0s
embeddedtest.2liirln0	test	Nov 29 2021, 15:30:16	0s

RAW DATA
embeddedtest.2lij1nvp

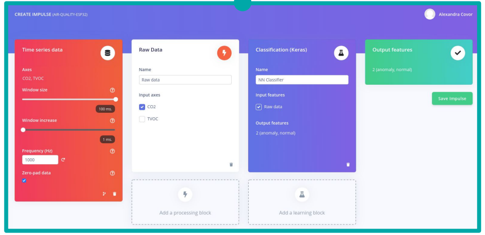


Software Description

2. Creating & training the model



Labelling the data



CREATE IMPULSE (AIR-QUALITY-ESP32)

Alexandra Covor

Time series data

Axes: CO2, TVOC

Window size: 100 ms.

Window increase: 1 ms.

Frequency (Hz): 1000

Zero-pad data: ☑

Raw Data

Name: Raw data

Input axes: ☑ CO2 ☐ TVOC

Classification (Keras)

Name: NN Classifier

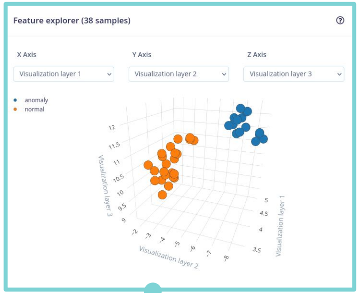
Input features: ☑ Raw data

Output features: 2 (anomaly, normal)

Output features

2 (anomaly, normal)

Creating the model



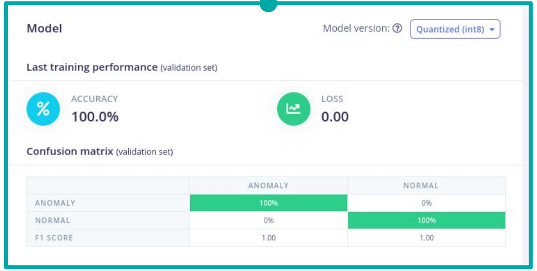
Feature explorer (38 samples)

X Axis: Visualization layer 1

Y Axis: Visualization layer 2

Z Axis: Visualization layer 3

Generating features



Model

Model version: Quantized (int8)

Last training performance (validation set)

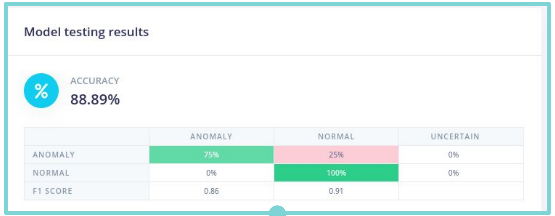
ACCURACY 100.0%

LOSS 0.00

Confusion matrix (validation set)

	ANOMALY	NORMAL
ANOMALY	100%	0%
NORMAL	0%	100%
F1 SCORE	1.00	1.00

Training



Model testing results

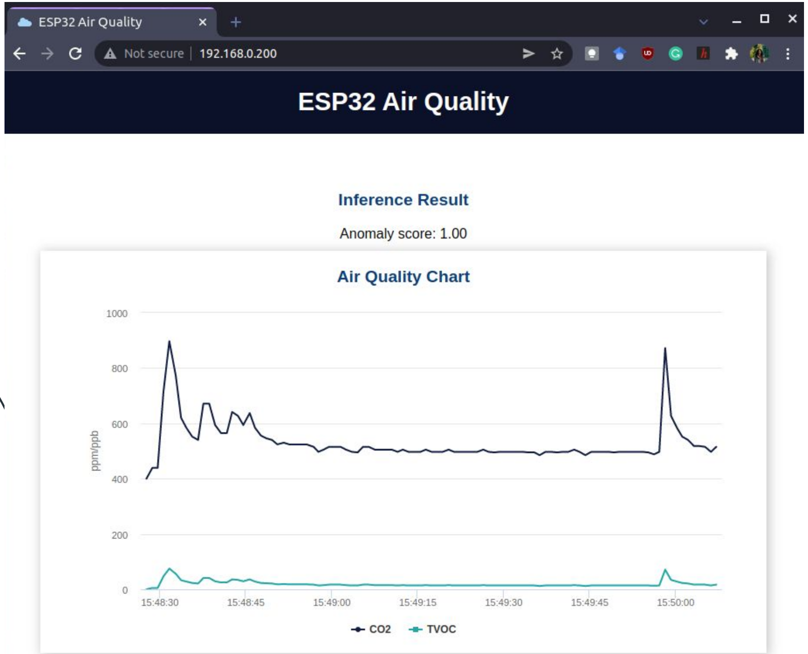
ACCURACY 88.89%

	ANOMALY	NORMAL	UNCERTAIN
ANOMALY	75%	25%	0%
NORMAL	0%	100%	0%
F1 SCORE	0.86	0.91	

Testing the model

Software Description

3. & 4. Deploying the model & Integrating it with the web server



ESP32 Air Quality

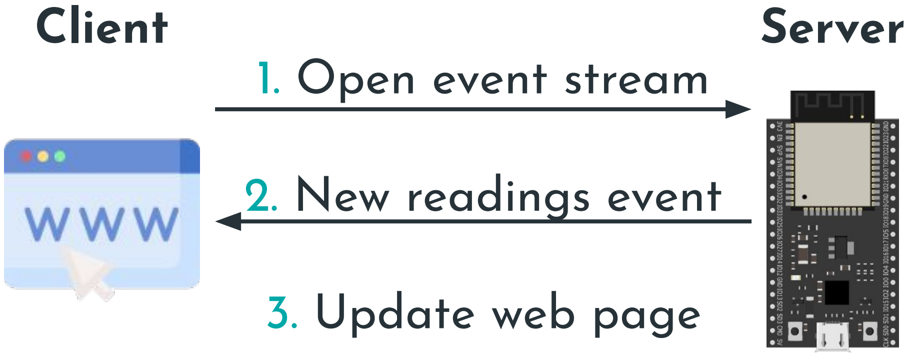
Not secure | 192.168.0.200

ESP32 Air Quality

Inference Result

Anomaly score: 1.00

Air Quality Chart



Client

Server

1. Open event stream

2. New readings event

3. Update web page

Server-Sent Events
HTTP Protocol
SPIFFS

Software Description

3. & 4. Deploying the model & Integrating it with the web page

Arduino Code

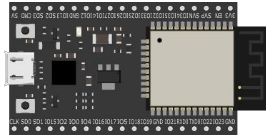
```
String getSensorReadings() {
  airQualitySensor.readAlgorithmResults();
  readings["co2"] = String(airQualitySensor.getCO2());
  readings["tvoc"] = String(airQualitySensor.getTVOC());
  readings["result"] = String(anomalyScore);

  String jsonString = JSON.stringify(readings);
  Serial.println(jsonString);
  return jsonString;
}
```

JavaScript Code

```
function getReadings(){
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var myObj = JSON.parse(this.responseText);
      console.log(myObj);
      plotAirQuality(myObj);
      inferenceElement.innerHTML = myObj.result;
    }
  };
  xhr.open("GET", "/readings", true);
  xhr.send();
}
```

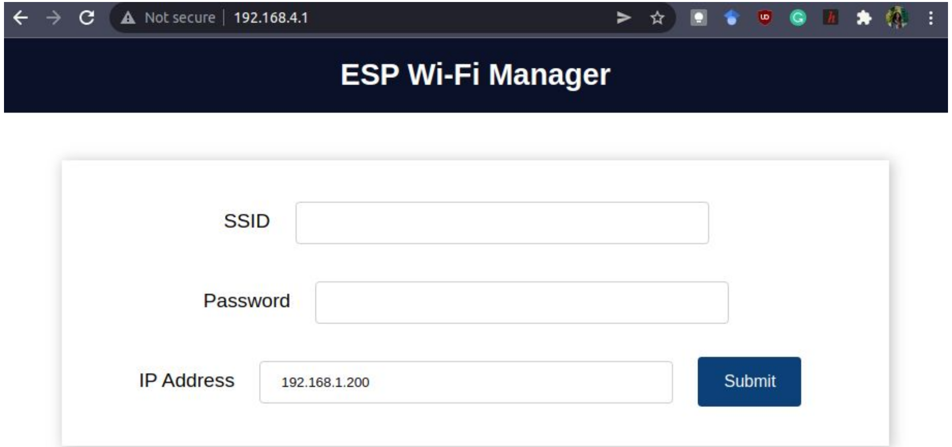
Software Description

3. & 4. Deploying the model & Integrating it with the web page



Access Point: ESP-WIFI-MANAGER

IP: 192.168.4.1



Not secure | 192.168.4.1

ESP Wi-Fi Manager

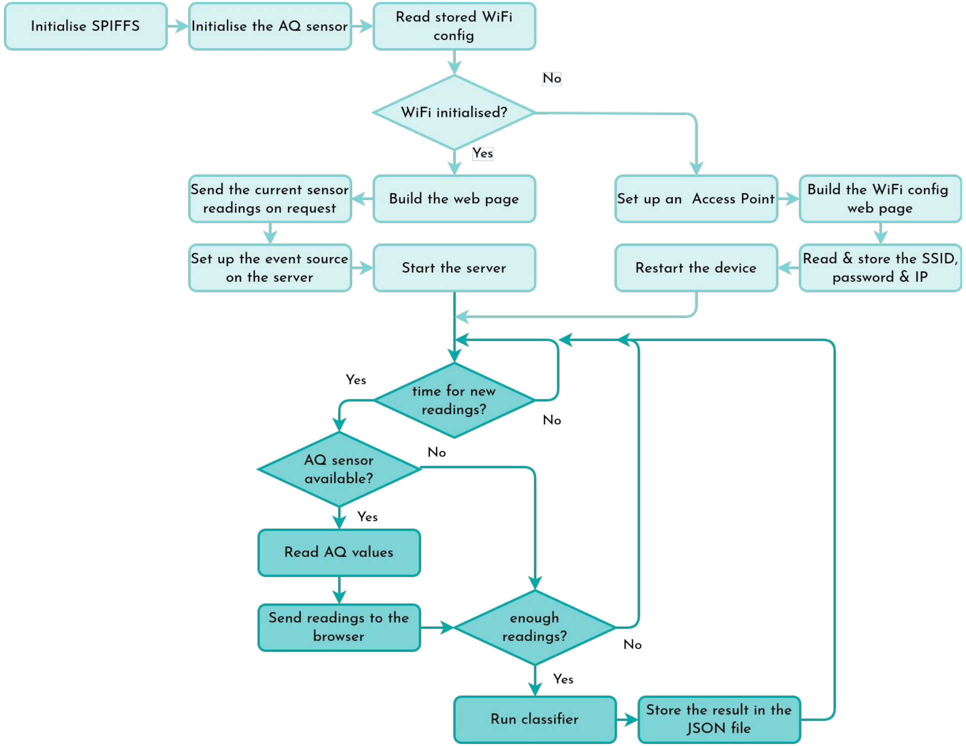
SSID

Password

IP Address 192.168.1.200

Software Description

3. & 4. Deploying the model & Integrating it with the web page



Conclusions

Future improvements:

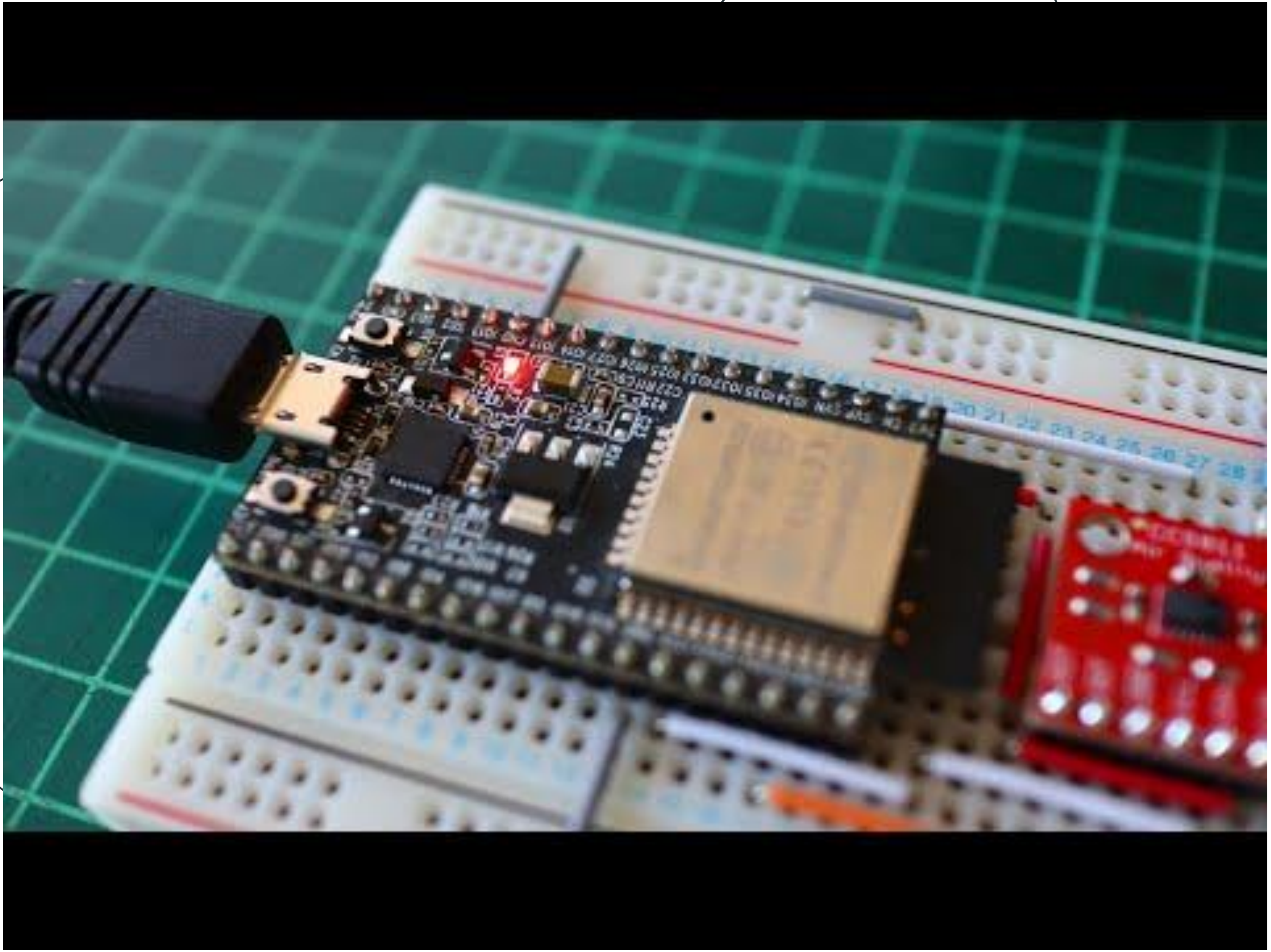
Gathering more data in order to better train the model

Implementing alerts

Potential application:

Artificial Nose

Image source: Benjamin Cabé, makezine.com/projects/second-sense-build-an-ai-smart-nose


20 21 22 23 24 25 26 27 28

References

Edge Impulse Documentation

- Edge Impulse Porting Guide
- Edge Impulse Remote Management Protocol
- Edge Impulse Ingestion Service
- Edge Impulse C SDK Usage Guide
- Edge Impulse Data Acquisition Format
- Running your impulse locally
- Classifying data (Arduino)

Software Libraries

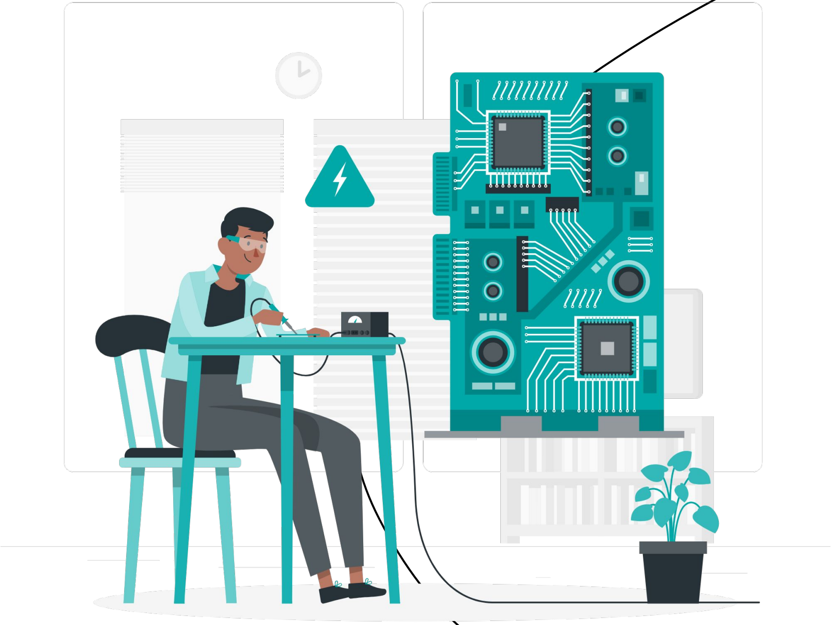
- Arduino Websockets
- SparkFun CCS811 Air Quality Breakout
- Edge Impulse C Ingestion SDK
- Edge Impulse C Ingestion SDK Samples
- ESPAsyncWebServer
- AsyncTCP

Others

- ESP32 Plot Sensor Readings in Charts (Multiple Series)
- ESP32: Create a Wi-Fi Manager (AsyncWebServer library)

Images

This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik** and illustrations by **Stories**.



Thank you!