tinyML-01: Next-Gen tinyML Smart Weather Station

ADN Innovators

Abhay Bhosle

abhaybhosle70@gmail.com

ABSTRACT

The creation of a next-generation smart weather station using Tiny Machine Learning (tinyML) technology is described in this project. The station seeks to overcome the shortcomings of conventional weather monitoring systems by being creative, economical, and energy-efficient. A microcontroller unit with tinyML models for on-device data processing will be the central component of the station. Real-time data on temperature, humidity, pressure, rain, and air quality will be recorded by environmental sensors. Through the analysis of this sensor data by the tinyML models, the station will be able to determine local weather trends such as rain intensity. Low-power, low-cost hardware and software solutions are given priority in this design. This guarantees low-energy usage over a lengthy period of autonomous operation. The station is meant to be used by a broad spectrum of users with the goal of encouraging real-world applications.

Figure 1: Core Microcontroller Unit.

1 INTRODUCTION

Using Tiny Machine Learning (tinyML), this research explores the design and construction of a next-generation smart weather station. Our goal is to develop a station that surpasses the limitations of traditional weather monitoring systems by being not only feature-rich and inventive but also economical and energy-efficient. The XIAO ESP32S3 microcontroller unit (MCU) is key to this station. The perfect platform for executing tinyML models directly on the device is this potent yet low-power MCU. Edge Impulse, an intuitive platform that simplifies the entire tinyML creation process, including data preprocessing and feature extraction, will be used to train these models.

The station will make use of a variety of specialized sensors in order to obtain a complete picture of the environmental conditions. Particulate matter levels will be continuously monitored by the Grove PPM 2.5 Sensor, giving important information about air quality. Accurate temperature data are essential for weather analysis, and the MLX90614 temperature sensor will provide them. Together with the XIAO ESP32S3 and Edge Impulse, these sensors form a reliable and effective system for gathering and analyzing data. Station is powered by a dependable 1200mAh 3.7V Li-ion battery. This option allows for long-term deployment without the need for frequent recharging by prioritizing extended operation with low energy usage.

2 DESIGN

The intuition behind the design of the device is to integrate the various mentioned components to work in a synchronous architecture. As the project aims to create an integrated platform that seamlessly combines classification of weather intensities with real-time monitoring of the environment, hence the architecture has to be robust.

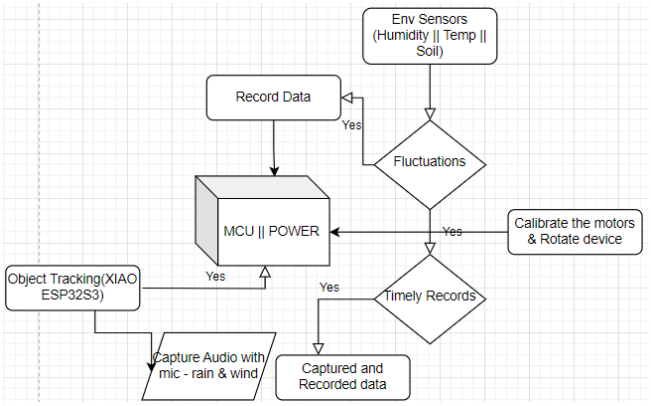


Figure 2: Station Architecture.

2.1 Electronics

2.1.1 XIAO ESP32S3 microcontroller unit (MCU) –

Processing – 240MHz Xtensa 32-bit LX7 dual core processor
Memory – 8MB PSRAM + 8MB Flash
Wireless – 2.4GHz WiFi
The XIAO ESP32S3, a low-power, WiFi-enabled microcontroller unit from Espressif Systems, is the brains of the station.
It is perfect for battery-powered applications like this weather station because of its small size and effective architecture.
With its dual-core processor and built-in Wi-Fi and Bluetooth, the ESP32S3 can communicate with external devices when necessary.
Grove Shield – seamless connectivity and interfacing
Battery - Lithium Battery Charging and Management Function
Protocols – 2xI2C, UART, SPI-Flash .

2.2 Sensors

2.1.2 Grove Laser PM2.5 Sensor – The PM2.5 sensor will yield the value of 2.5 um particulate matter. The amount of particulate matter absorbed in the surrounding environment can be ascertained from this data using analytics. This will monitor the effects of industry and seasonality on environmental absorption, since modern woods found in urban areas are thought to be the ones that absorb particulates from automobile and industrial emissions.

2.1.3 Grove TOF Range Sensor – TOF will provide precise results of the detected objects from the device this will help track their movement.

2.1.4 Grove Sunlight Sensor and Grove Colour Sensor – Identify the environmental parameters in real-time scenarios.
Integrating the sensors with XIAO esp32s can be done using the I2C channels

Figure 3: Casing assembly

2.3 Casing - 3D Printing the device covers/mounts

The design of the enclosure shown in Figure 3 was made using a 3D printer. The enclosure is composed entirely of polylactic acid (PLA), with alloys being utilized to construct the other portions. These materials are robust and waterproof if they are sufficiently thick. The thickness of our components varied from 5 to 12 mm. Initially, an open design does not isolate the sensors from the external environment in an effort to prevent physical parameter loss. Second, the environmental sensors are not enclosed. Maintaining the high sensitivity of the PM sensor requires doing this. As seen in Fig. 4, the final enclosure will encase the entire gadget, demonstrating its waterproof nature and ability to survive harsh conditions.

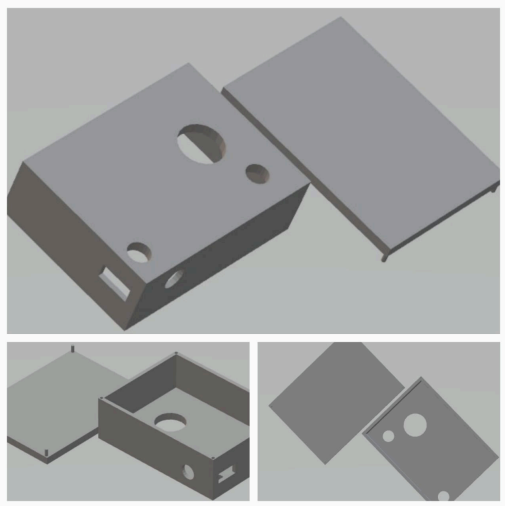


Figure 4: Final 3D-printed casing

3 EMBEDDED APPLICATION

3.1 XIAO ESP32S3

Data Acquisition:
To record sound samples from a microphone at a particular sample rate and bit depth, the code makes use of an I2S (Inter-Integrated Sound) interface.
It buffers the audio data that it continuously reads from the I2S peripheral.
Model Execution:
The system makes use of a tinyML model that has already been trained with Edge Impulse. The purpose of this model is to classify audio.
The model requires audio samples inside a specified frame for input.
The input buffer of the model is continuously filled with recorded acoustic data. The model is prompted to conduct inference on that specific audio frame once the buffer is filled.
Inference and Output:
The three predetermined classifications (high, mid, and low rain intensity) are generated as probabilities by the tinyML model after it has analyzed the audio frame.
For a given audio frame, the predicted classification is the class with the highest probability.
The code shows the classification outcome for each stage of processing (data capture, model inference) together with timing details.

Additional Features:
An LED is included in the code to give a visual representation of the categorization result (perhaps strong rain triggering the LED).
It permits the user to potentially pause or resume audio capture by providing recording status control.

3.2 MFE - Edge Impulse

Although there are two more steps, the features extraction process is comparable to the Spectrogram in that the frame length, frame stride, and FFT length parameters are the same.

Triangular filters are applied on a Mel-scale to isolate frequency bands once the spectrogram has been computed. To choose the frequency band and the quantity of frequency features to be extracted, they are defined with the parameters Filter number, Low frequency, and High frequency. A perceptual scale of pitches that listeners perceive as having equal distances from one another is called the Mel-scale. In order to make it work well on sounds that can be recognized by the human ear, the aim is to extract more features (more filter banks) in the lower frequencies and less in the high frequencies.

By analyzing the complete dataset and suggesting a set of parameters that are tailored for your dataset, the autotuning tool streamlines this procedure.

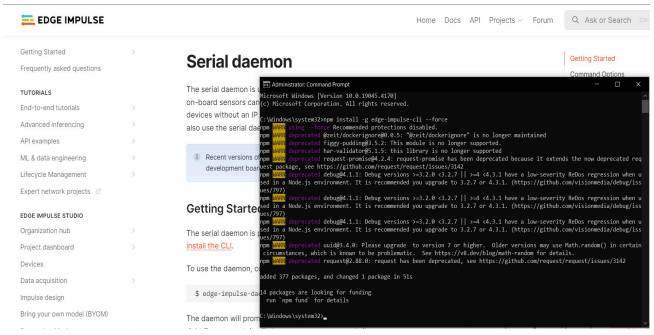


Figure 6: Installing edge-impulse-cli for data-forwarder

4 DATASET

Dataset Acquisition and Preprocessing:
Audio Dataset: The challenge refers to a sound dataset with three types of sounds: "high," "low," and "mid." Regarding the particular audio content and its format, no information is given.
MFE Features: The system makes use of MFE features, a class of spectral feature that records the audio signal's energy distribution within Mel-frequency bands. **Center Frequency - 8220 Hz**. Data collected within the bandwidth of the C.F. i.e. **BW = 200 Hz (app.)**
Feature Extraction Configuration: Variables like "Filter number," which probably regulates how many Mel-frequency bands are employed in the study, are mentioned in the prompt. There is no explicit provision for further configuration information such as frame length, frame stride, etc.

4.1. Dataset Selection:
ITU's recommendations guided the selection process, emphasizing the importance of datasets that align with the device and model parameters. Acquiring the data from multiple use cases to diversify it provides distinctive features for proper classification. Utilizing the edge-impulse -data-forwarder from the edge-impulse-cli which has support from all edge devices proved efficient in accurate data acquisition.
4.2 Feature Analysis and Processing Block Selection: Each **column** represents the features for a single time frame in your audio data. The columns progress from **low frequency to high**.
The **number of rows** in the array is equal to the number of Mel filters used in the analysis (set by the "Filter number" parameter).
Each **value** in the array represents the **energy** captured within a specific Mel-frequency band for a particular time frame in the audio.
4.3. DSP Performance Metrics:
Signal-to-Noise Ratio (SNR): This gauges how much of the features' background noise is replaced by the desired signal. Features with a higher SNR are quieter.
Mel-Cepstral Distortion (MFCC): This is a measure of separation between two Mel-frequency cepstral coefficients (which are obtained from MFE characteristics) that is employed in the comparison of audio sources' similarity.
Computational Efficiency: Your feature extraction algorithm's processing time and resource consumption are measured here.
Feature Discrimination: This metric evaluates how well the extracted features separate different audio classes.
4.4. Rain Intensities as per Class boundaries:
High Intensity - 19.09 mm and above rainfall
Mid Intensity - 6.36 - 19.09 mm rainfall
Low Intensity - below 6.36 mm rainfall

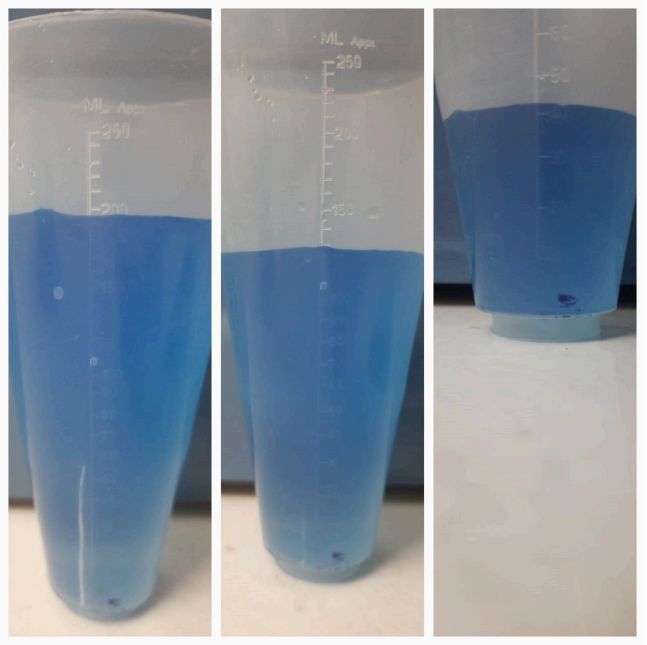


Figure 8: High Mid Low

To determine the ground rain intensities -
Collect the rain into a gauge of diameter 10 cm = 100mm
Rainfall = (Gauge surface Area * Height of water) / 1000
Volume of water = (Gauge surface Area * Height of water)
At intensity = high
measured volume of water = 195 ml = 195000 mm3
Height of water = 195000/(50*50*pi) = 24.828 mm
At intensity = Mid
measured volume of water = 130 ml = 130000 mm3
Height of water = 130000/(50*50*pi) =16.55 mm
At intensity = Low
measured volume of water = 43 ml = 43000 mm3
Height of water = 43000/(50*50*pi) = 5.47 mm

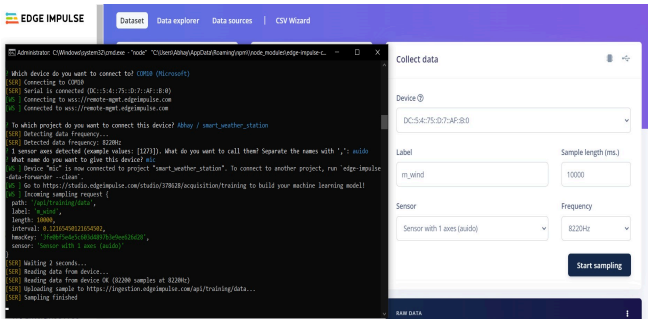


Figure 7: Data Acquisition - One Axis - Audio - 8220Hz

5 TinyML MODEL – Selection, Training, and Validation

5.1. Model Selection:
Input Layer:
Shape: (3920) - This signifies a 1D array with 3920 features. These features likely represent Mel-filterbank energy (MFE) extracted from audio samples.
Convolutional Layers:
1st Convolutional Layer:
Number of filters: 8
Kernel size: 3
Activation function: ReLU (Rectified Linear Unit)
Padding: Same (preserves the input dimensions)
2nd Convolutional Layer:
Number of filters: 16
Kernel size: 3
Activation function: ReLU
Max Pooling Layers: After each convolutional layer, there's a Max Pooling layer with a pool size of 2 that performs downsampling, reducing the dimensionality of the data.
Dropout Layers: Each convolutional layer is followed by a Dropout layer with a rate of 0.5. This helps prevent overfitting by randomly dropping out a certain percentage of activations during training.
Flatten Layer: This layer reshapes the output from the final convolutional layer into a 1D vector suitable for feeding into the fully-connected layer.
Dense Layer:
Number of units: 3 (matches the number of classes: "high", "low", "mid")
Activation function: Softmax - This layer outputs a probability distribution for each class.

5.2. Hyperparameter Tuning:
Number of training cycles (epochs): You've set this to 30 (param: epochs). This controls how many times the entire training dataset is passed through the network for learning.
Learning rate: Set this to 0.001 (param: learning-rate). This determines how much the network updates its weights based on the errors during training.

Figure 8: Testing the device withAudio feed of classes

5.3. Training and Validation:
The training phase involved multiple iterations to ascertain the optimal balance between accuracy and computational efficiency. Two iterations were conducted, each utilizing datasets with equal numbers of audio frames per class. The careful orchestration of training parameters, such as epoch scheduling, ensured the absence of overfitting, with the model accuracy progressively improving throughout the training process.
5.4 Results: The Training iteration, comprising 54 samples of 10 sec each totaling 540 sec i.e. 9 mins of data, yielded a training accuracy of 97.03% . The Testing iteration, encompassing 9 samples of 10 sec each totaling 90 sec i.e. 1.5 mins of data, demonstrated a testing accuracy of 98.57%. Notably, the model showcased resilience to an increase in classes, with accuracy maintained and loss error reduced, emphasizing its scalability.
Inferencing Metrics: The model, deployed on Edge Impulse as Project ID - 376828, exhibited an inferencing time of 271 ms, with a peak RAM utilization of 40.6KB and Flash storage occupancy of 43KB. These metrics underscore the model's efficiency in

real-time rain intensity classification on resource-constrained edge devices.

This thorough method of choosing and optimizing models lays the foundation for a strong and effective tinyML solution made specifically for the special problems faced by smart weather stations. The model's adaptability is enhanced by the iterative training approach and subtle configuration choices, which make it a reliable tool for practical applications.

6 Project Results

6.1 Neural Network Operations

The following table outlines the data types utilized in various neural network operations during the project. These operations play a crucial role in the inference process, where input data undergoes specific transformations to produce output data. The types indicated are U8 (Unsigned 8-bit), I16 (16-bit Integer), F32 (32-bit Floating Point), and BOOL (Boolean).

Category	Description	Details
Model Selection	NN Classifier	3920 input features generated by MFE
Hyperparameter Tuning	Parameters for training the CNN	Learning Rate set on EI 0.005
Hyperparameter Tuning	Number of training cycles (epochs)	30
Hyperparameter Tuning	Learning rate	0.005
Final Layer Configuration	Output layer of the CNN	Softmax layer with 3 units (matches class labels)
Training and Validation	Data preparation and training process	Training Acc - 97.03%
Data Pre-processing	Data split	80% training, 20% validation (from train_input.json)
Data Pre-processing	Batch size	32 (default, can be overridden)
Results	Train and Test Scores	Run the script (train.py) to obtain accuracy, loss etc.
Model Architecture	Layers and connections in the CNN	6 - Layered Architecture
Layer - 1	Input layer	1D array with 3920 features (likely MFE)
Layer - 2	Convolutional layers	2 layers with ReLU activation, (8 filters, kernel size 3) and (16 filters, kernel size 3)
Layer - 3	Pooling layers	Max pooling layers after each convolutional layer (pool size 2)
Layer - 4	Dropout layers	Dropout rate of 0.5 after each convolutional layer
Layer - 5	Flatten layer	Reshapes output from convolutional layers
Layer - 6	Output layer	Dense layer with 3 units and Softmax activation

Table 1: Dataframe processing functions

6.1 Interpretation

Softmax, Fully Connected, Convolution 2D: These operations accept various data types as input, such as unsigned 8-bit integers (U8), 16-bit integers (I16), 32-bit floating-point numbers (F32), and booleans (BOOL). The output maintains the same flexibility in data types.

Max Pooling 2D: This operation involves pooling the maximum value from a set of values. It accepts a range of data types for input and produces output with the same diversity of data types.

Depthwise Conv 2D, Average Pooling 2D, Strided Slice: These operations, similar to others, exhibit versatility in accepting different data types as input and generating output with comparable flexibility.

6.2 Implications for the Project

The adaptability of these operations to diverse data types underscores the project's commitment to efficiency and resource optimization. It allows the neural network to handle a wide range of input data, contributing to the project's success in achieving high accuracy, minimal loss, and low power and memory requirements during inferencing.

Frame_Length	0.02
Frame_Sride	0.01
Filter Number	40
FFT_Length	256
Noise_floor(dB)	-52 dB
Training Window	1395
input_frames	10
interval_ms	1000 ms
frequency_low	300
frequency_high	sample_rate/2

Table2: Frame Details for DSP and Classifier

6.2 ML accuracy

The machine learning (ML) model developed for the Smart weather station project has demonstrated commendable accuracy, reflecting its robust performance in identifying and classifying rain intensities. Through meticulous training, validation, and fine-tuning processes, the model has achieved consistent and reliable results.

Model Link - Project ID : 378628

6.2.1 Key Metrics:

Training Accuracy: The model has been trained on a dataset which is personally acquired by recording audio data of total 9 mins with frame size of 10 sec each. The errors in confusion matrix indicate faults between high & low are more than high & mid - Reasons for this spectral similarities which have resulted in similar feature generation for some frames of the two separated classes and distinctive unavoidable sounds while toggling the intensities over the setup. Out of the three classes each class is distinctly classified in training as is evident by the results:

Training Accuracy - 97.03% **Loss - 0.13**

Last training performance (validation set)

ACCURACY 97.0%

LOSS 0.13

Confusion matrix (validation set)

	HIGH	LOW	MID
HIGH	95.2%	4.8%	0%
LOW	0%	99.1%	0.9%
MID	0.9%	2.6%	96.6%
F1 SCORE	0.97	0.96	0.98

Figure 9: Confusion Matrix ALong with F1-Scores - Training

Testing Accuracy: The model's performance was rigorously assessed through testing on distinct datasets. Across multiple iterations, the testing accuracy consistently proved for each individual class as well as for the entire test dataset.

Testing Accuracy - 98.57%

ACCURACY 98.57%

	HIGH	LOW	MID	UNCERTAIN
HIGH	100%	0%	0%	0%
LOW	0%	95.7%	2.2%	2.2%
MID	0%	0%	100%	0%
F1 SCORE	1.00	0.98	0.99	

Figure 10: Confusion Matrix ALong with F1-Scores - Testing

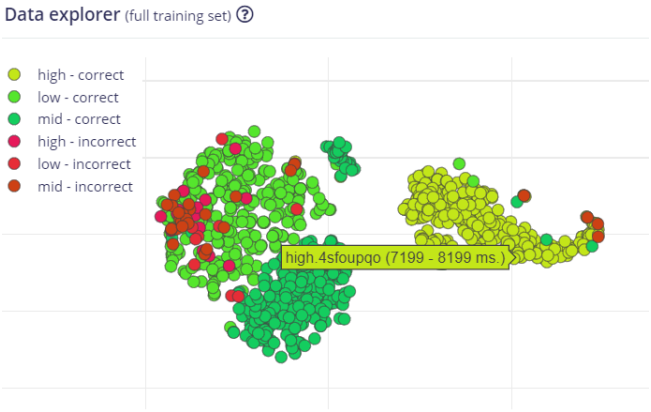


Figure 11: Classified Data Points - 3 Classes

Resource Utilization:

Inferencing the model on the XIAO ESP32S3 has demonstrated impressive efficiency. The model operates within the constraints of the microcontroller's memory and processing capabilities, showcasing a harmonious balance between accuracy and resource optimization.

Dataset Utilization:

The 9 min + 1.5 min = 10 min of data collected was with regard to actual real-time testing of the prototype. Provided dataset was efficient but bulky and training time exceeded the limit. Model trained on the dataset performed efficiently to the respective setup environment. Test set is strictly separated from the training dataset. Noise class is not added as the device does not need to run the model unless a trigger indicates rain - this feature will save energy and memory. A rain sensor which detects the presence of rainfall will work as an interrupt for classification inference.

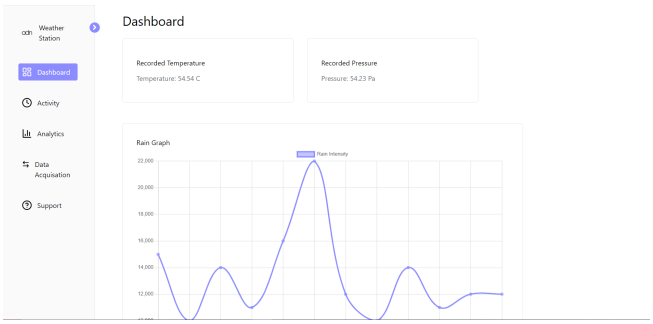
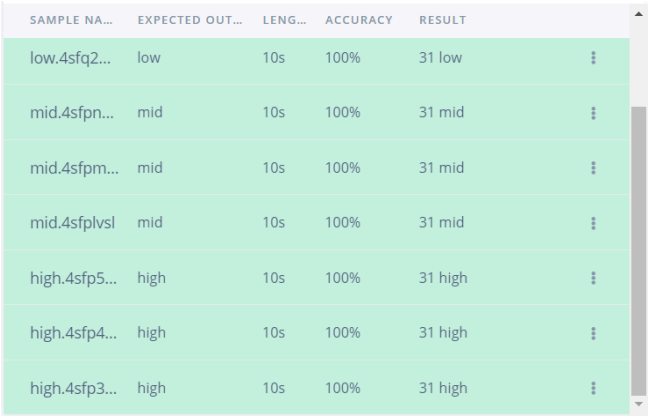


Fig 12: Dashboard Made with React & Firebase

Dashboard Link - Hosted on Netlify



SAMPLE NA...	EXPECTED OUT...	LENG...	ACCURACY	RESULT
low.4sfq2...	low	10s	100%	31 low
mid.4sfpn...	mid	10s	100%	31 mid
mid.4sfpm...	mid	10s	100%	31 mid
mid.4sfplvsl	mid	10s	100%	31 mid
high.4sfp5...	high	10s	100%	31 high
high.4sfp4...	high	10s	100%	31 high
high.4sfp3...	high	10s	100%	31 high

Figure 13: Classified Data Points - 3 Classes

6.3Memory Consumption - Model

Figure 14: Memory Requirement - 3 Classes

Details on Prototype:

Battery Life Calculations:

Battery Life = Battery Capacity / Average Power Consumption

BLavg = 1,200 mAh / 46.5mA = 25.8 = 25 hr 48 min (Considering Average Working Conditions)

BLmax = 1,200 mAh / 89.6mA = 13.392 = 13 hr 24 min (Considering Max Power Utilization)

Device ON time per day = 24 – (Duration between considerable weather change * No. of such Cycles)

DON = 24 – (3*6)

DON = 6 hrs (approx.)

Battery Life (on single charge) = ((25*60)+48) / (6*60) [considering avg power consumption]

Battery Life (on single charge) = 4.3 days ~ 4 Days (approx..) [considering avg power consumption]

Battery Life (on single charge) = ((13*60)+24) / (6*60) [considering max power consumption]

Battery Life (on single charge) = 2.23 days ~ 2 Days (approx..) [considering max power consumption]

Average Battery Life ranges from 2 – 4 Days

Cost Estimation = $ 75 (includes MCU - Sensors - Shield) + $ 10 (3D printing & Miscellaneous costs)

Total Cost = $85

REFERENCES

[1] Seeed Studio XIAO Series: https://www.seeedstudio.com/xiao-series-page

[2] Seeed Studio camera usage article: https://wiki.seeedstudio.com/xiao_esp32s3_camera_usage/

[3] Git-Hub repo for XIAO ESP32 : https://github.com/Mjrovai/XIAO-ESP32S3-Sense

[4] Mobile Application Development and Firebase Interaction: https://firebase.google.com/docs/functions
Digital Signal Processing (DSP) for Image Datasets:https://arxiv.org/abs/1710.04043

[5] TinyML Model Selection and Optimization:
Title: "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"
Authors: Mingxing Tan, Quoc V. Le
https://arxiv.org/abs/1905.11946

Flutter Mobile Application Development:
Title: "Flutter: A Framework for Building Native Apps"
Authors: Eric Seidel, Larwan Berke, et al.

[6] *https://flutter.dev/*

[7] Seeed Studio – Getting Started

[8] Camera Usage

[9] Github for XIAO

[10] https://smart-weather-station-ab.netlify.app/

[11] https://tailwindui.com/components#pricing

[12] https://react.dev/reference/react