TEAM GENSTORM

NEXT-GEN TINYML SMART WEATHER STATION CHALLENGE 2024

PROJECT REPORT

1. Introduction

In the face of escalating climate challenges, precise weather monitoring is crucial for community resilience and adaptation. Enter the "Next-Gen tinyML Smart Weather Station Challenge 2024," a beacon for innovation in environmental monitoring. Team GenStorm rises to this occasion with a groundbreaking solution that harnesses the power of Tiny Machine Learning (tinyML) to revolutionize weather stations. Our team has designed an intelligent model for classifying rainfall intensity, deftly using the Edge Impulse platform. This model, seamlessly deployed as an Arduino library, has been meticulously modified to calculate rainfall amounts while simultaneously recording pressure and temperature. The essence of power optimization lies at the heart of our design, ensuring the station's longevity and efficiency. Leveraging a rich dataset of audio .wav files, generously provided by AI4Africa, our model has been rigorously trained to deliver accurate, real-time environmental data. The dataset was split into 76% for training and 24% for testing. Our impulse creation process included time series data with a window size of 1,000 ms and a frequency of 16,000 Hz. We utilized Mel-filterbank energy (MFE) features, with 40 filters and a frame stride of 0.01, to pre-process the audio files. The rainfall intensities are classified into five categories: HEAVY_RAIN, LOW_RAIN, MODERATE_RAIN, NO_RAIN, and VERY_HEAVY_RAIN.

The neural network was trained over 300 cycles with a learning rate of 0.005, using a 1D convolutional neural network architecture with dropout layers to enhance model robustness. With a peak RAM usage of 20 KB and a processing time of 302 ms, our model achieved an impressive test accuracy of 97.47%. Post-quantization, the model size was reduced to 10 KB, making it highly efficient for deployment on low-power hardware. By embracing low-power, cost-effective hardware, and sophisticated software, our solution is poised to set new standards in weather monitoring. The Next-Gen tinyML Smart Weather Station 2024 is not just a competition but a platform fostering collaboration, creativity, and practical solutions. Team GenStorm is proud to contribute to this vision, enhancing weather monitoring capabilities vital for bolstering community resilience in the face of climate change.

1.1 Problem Statement

Accurate and reliable weather monitoring is essential for agricultural productivity and sustainability. However, existing weather stations are often expensive, complex to maintain, and include mechanical moving parts prone to failure. These challenges are particularly acute in remote or resource-limited settings, such as small farms, especially in developing regions like Africa, where weather stations are scarce due to their high cost and power requirements.

Weather parameters vary spatially and are crucial for predicting future events and aiding planning efforts. Therefore, a dense concentration of weather stations must ensure accurate data collection. This challenge aims to create a low-cost, low-power, reliable, and accurate weather station that can measure all weather conditions, focusing on rain and wind intensities, without mechanical moving parts. This device should utilize ultra-low power machine learning at the edge, ensuring ease of installation and maintenance. The intended application is in agricultural environments, providing farmers with real-time local weather data to make informed crop planting and management decisions. By focusing on the acoustics of rain and wind intensities, this project aims to offer a practical solution to the scarcity of weather stations in developing countries, enhancing agricultural planning and resilience.

2. Data Collection

The dataset used in this project was collected by AI4Africa in September 2022 in Kogi State, Nigeria (7.7337N, 6.6906E). The data collection involved using a galvanized zinc sheet to capture the sound of rainfall, recorded with a Techno Spark 7P mobile phone. The recordings were saved in .aac format at a sampling rate of 22050 Hz. Over 12 days, 7,800 seconds of audio data were collected, capturing a range of rainfall intensities from 0.1 to 6.6 mm per five minutes. A total of 25 samples, each five minutes long, were gathered.

This dataset provides essential audio recordings for training models to classify rainfall intensity based on acoustic signals. The dataset can be accessed via the link: AI4Africa Dataset. The ground truth rain intensity categorization was determined using AI4Africa's provided intensity ranges. These ranges were applied to tag corresponding rainfall intensity measured in millimeters, ensuring accurate labels for the training dataset. This method provided a reliable foundation for classifying rainfall intensity based on the acoustic signals in our model.

Despite the extensive size of the CSEM dataset, we opted not to use it due to time constraints and challenges in classifying its data effectively. In contrast, the AI4Africa dataset was more straightforward and required less time to integrate into our training process. The difficulty in determining appropriate classes within the CSEM dataset led us to prioritize the AI4Africa dataset for its simplicity and ease of use.

Following submitting our final work, our team analyzed our model and identified the need to include noise audio in the "NO_RAIN" class. We are currently working on recording various noise samples to retrain and enhance the model's accuracy.

2.1 Dataset Processing and Feature Extraction

Using the AI4Africa dataset we categorized the .wav files into classes as seen in Table 1. The final dataset with the grouped classes can be accessed via the link: Grouped classes dataset using

AI4Africa original dataset. The input audio data is preprocessed with a window size of 1,000 milliseconds and a window increase of 500 milliseconds. The sampling frequency is set to 16000 Hz, and zero-padding is applied to ensure consistency in data dimensions. MFE parameters, such as frame length, frame stride, and filter number, are configured to extract relevant features from the audio signals. The data is made up of 5 classes as seen in the Table 1, with the respective rainfall intensities.

Table 1: Rainfall Intensity Classes

Category of Rainfall (Classes)	**Rainfall Intensity (mm)**
Very Heavy	Greater than 3.5
Heavy	Between 1.5 - 3.5 exclusive
Moderate	Between 0.5 - 1.5 exclusive
Light / low	Between 0.08 - 0.5 exclusive
No Rain	Below 0.08 - 0

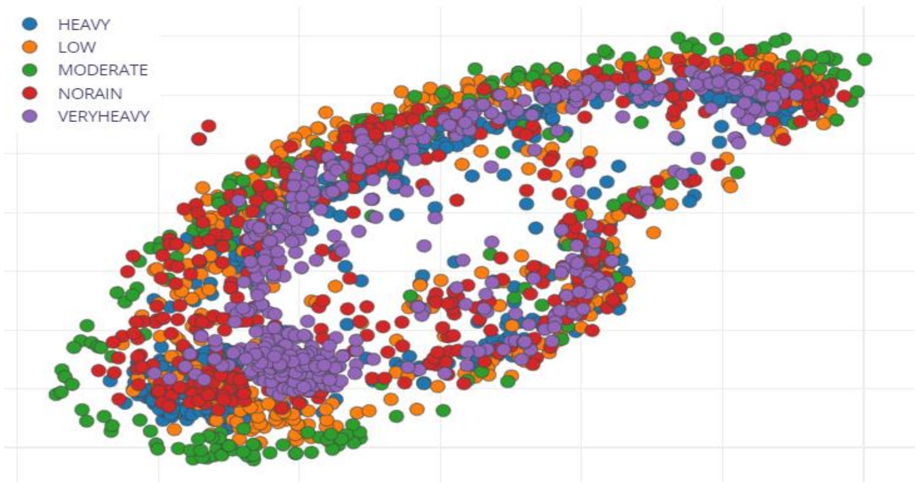


Figure 1: Feature Extraction for the Classes of Rain

Mel-filterbank energy (MFE) parameters are configured to extract relevant features from the audio signals. These parameters include:

- Frame Length: 0.02 seconds
- Frame Stride: 0.01 seconds
- Filter Number: 40
- FFT Length: 256

- Low Frequency: 0 Hz
- High Frequency: Set accordingly
- Normalization: Noise floor set at -52 dB

2.2 Classification and Data Splitting

The data is classified into five categories based on rainfall intensity: HEAVY_RAIN, LOW_RAIN, MODERATE_RAIN, NO_RAIN, and VERY_HEAVY_RAIN. The dataset was split into training and testing sets in a ratio of 76/24 to ensure robust model evaluation and performance validation. This preprocessing and classification setup allows for the development of a reliable and accurate model to classify rainfall intensity using edge machine learning techniques.

3. Proposal Summary and Model Design

The proposed solution leverages a neural network classifier trained on Mel-filterbank energy (MFE) features extracted from audio signals to accurately classify rainfall intensity. The system architecture encompasses several key components: data preprocessing to prepare the audio signals, feature extraction using MFE parameters, model training to develop the classifier, and deployment to implement the solution in real-world settings. This comprehensive approach ensures a robust, efficient, and scalable system for weather monitoring.

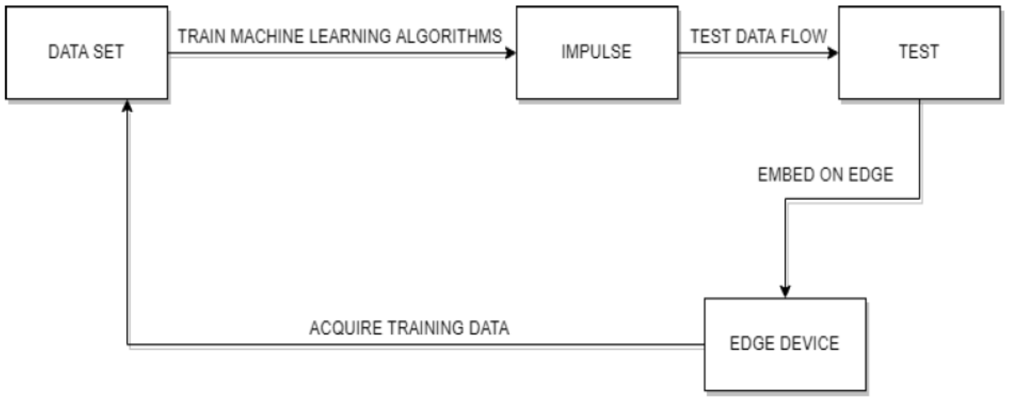


Figure 2: Block Diagram of Edge Impulse

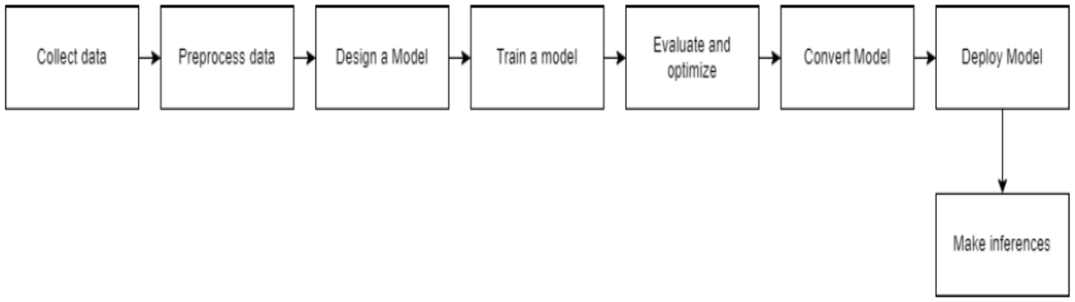


Figure 3: Block diagram of flow of implementation of TinyML

3.1 Model Design and Network Architecture

The neural network architecture for our rainfall classification model is based on a 1D Convolutional Neural Network (CNN) as shown in Figure 4. The input layer consists of 3,960 features, derived from the Mel-filterbank energy (MFE) features of the audio signals. This input layer is reshaped into 20 columns to facilitate convolution operations.

The first convolutional layer employs 8 neurons with a kernel size of 3, followed by a pooling layer to reduce dimensionality. A dropout layer with a rate of 0.25 is added to prevent overfitting. The second convolutional layer increases the neuron count to 16, with the same kernel size of 3, followed by another pooling layer. This is again followed by a dropout layer with a 0.25 rate.

The architecture includes a flatten layer to convert the 2D matrix into a 1D vector, suitable for the fully connected layers. An additional layer is incorporated to enhance the model's capacity to learn complex patterns. Finally, the output layer classifies the input data into one of the five categories: HEAVY_RAIN, LOW_RAIN, MODERATE_RAIN, NO_RAIN, and VERY_HEAVY_RAIN.

Training settings include 300 cycles with a learning rate of 0.005, optimized for the CPU. Advanced training settings and data augmentation techniques are applied to ensure robust model performance. This comprehensive architecture ensures accurate and reliable classification of rainfall intensity.

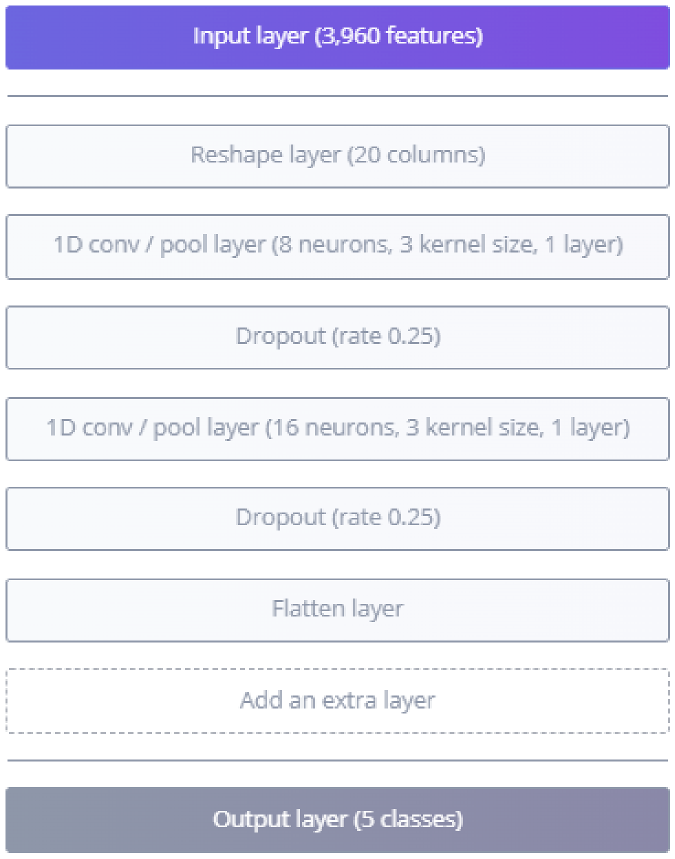

Input layer (3,960 features)
Reshape layer (20 columns)
1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)
Dropout (rate 0.25)
1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)
Dropout (rate 0.25)
Flatten layer
Add an extra layer
Output layer (5 classes)

Figure 4: Model Network Architecture

The classifier utilizes Mel-filterbank energy features (MFE) for feature extraction from the input audio data. These features are then fed into a neural network architecture designed for rain intensity classification. The neural network consists of multiple layers, including convolutional and pooling layers, dropout layers for regularization, and an output layer for class prediction.

Table 2: Network Architecture Parameters

Layer	Description
Input Layer	3,960 features
Reshape Layer	20 columns
1D Conv / Pool Layer	8 neurons, 3 kernel size, 1 layer
Dropout Layer	Rate 0.25
1D Conv / Pool Layer	16 neurons, 3 kernel size, 1 layer
Dropout Layer	Rate 0.25
Flatten Layer	Converts 2D matrix into 1D vector
Additional Layer	Enhances model capacity
Output Layer	5 classes (HEAVY_RAIN, LOW_RAIN, MODERATE_RAIN, NO_RAIN, VERY_HEAVY_RAIN)

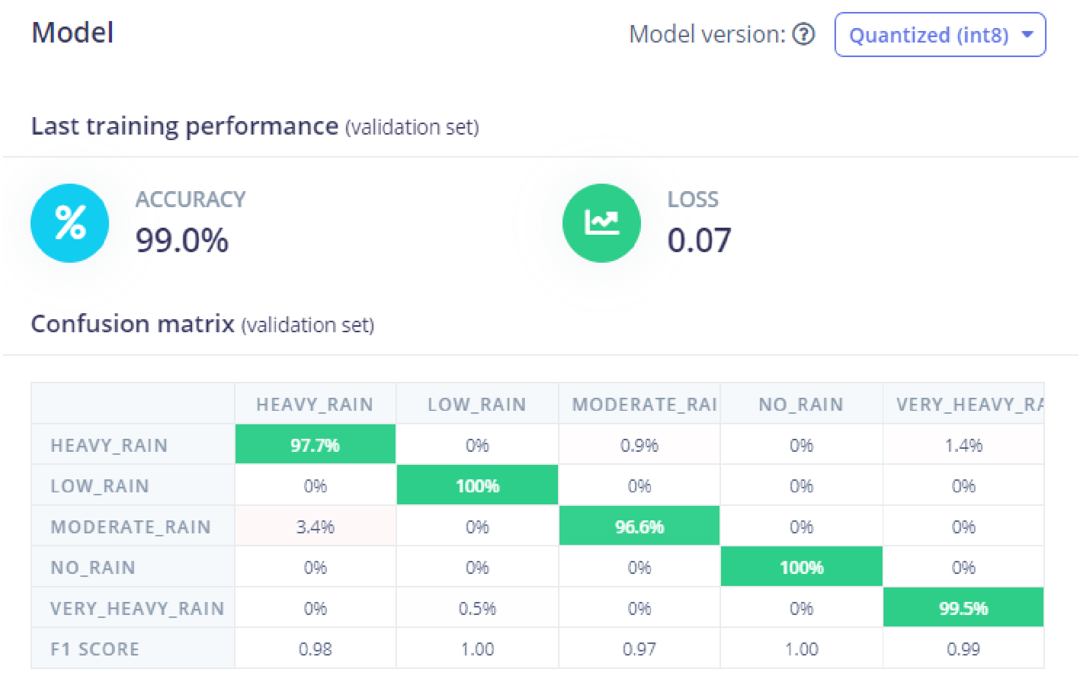
3.1.1 Motivation for Choosing the Model Architecture

The chosen ML model architecture is specifically optimized for classifying rainfall intensity from .wav files, leveraging its robustness and efficiency. Other architectures with fully connected neural networks, may not capture the temporal and spectral features of audio data as effectively. Fully connected networks tend to require significantly more parameters and computational resources, leading to potential overfitting and inefficiency on edge devices. LSTM networks, while good at sequence prediction, may not perform as well with the spectral characteristics of audio signals.

Our architecture begins with 3,960 features, reshaped for effective convolution operations. The 1D convolutional layers, with 8 and 16 neurons respectively, and a kernel size of 3, are adept at capturing temporal patterns in the audio data. Dropout layers with a rate of 0.25 prevent overfitting, ensuring the model generalizes well to new data. The flatten layer then converts the 2D matrix back to a 1D vector, facilitating accurate final classification. This architecture balances complexity and computational efficiency, making it ideal for accurate and real-time rainfall intensity measurement on low-power, resource-constrained edge devices.

3.2 Model Training

The classifier is trained using a dataset of labeled audio recordings representing different levels of rainfall intensity. The training process involves optimizing the neural network parameters through backpropagation and gradient descent. The model is trained for 300 cycles with a learning rate of 0.005, using a CPU for computation. The Results of the model training is seen in Figure 5.



Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 99.0%

LOSS 0.07

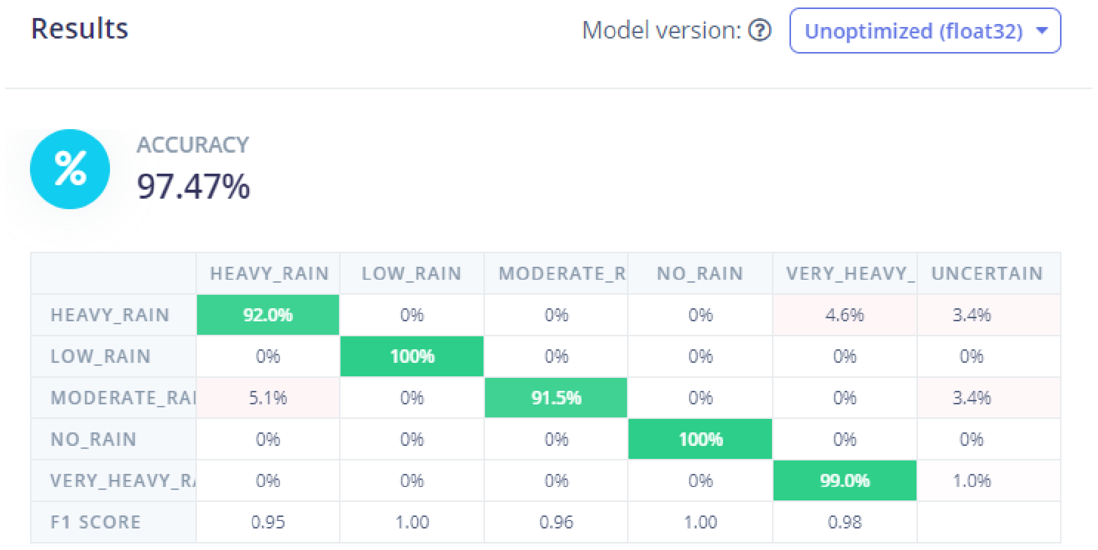
Confusion matrix (validation set)

	HEAVY_RAIN	LOW_RAIN	MODERATE_RAI	NO_RAIN	VERY_HEAVY_RA
HEAVY_RAIN	97.7%	0%	0.9%	0%	1.4%
LOW_RAIN	0%	100%	0%	0%	0%
MODERATE_RAIN	3.4%	0%	96.6%	0%	0%
NO_RAIN	0%	0%	0%	100%	0%
VERY_HEAVY_RAIN	0%	0.5%	0%	0%	99.5%
F1 SCORE	0.98	1.00	0.97	1.00	0.99

Figure 5: Model Training Confusion Matrix

3.3 Model Evaluation and Testing

The performance of the trained model is evaluated on a validation set using metrics such as accuracy and loss. Additionally, a confusion matrix is generated to analyze the classification performance across different rainfall intensity categories. The model achieves an accuracy of 97.47% and a low loss value of 0.13 as shown in Figure 6. This indicates its high effectiveness in rain intensity classification.



Results

Model version: Unoptimized (float32)

ACCURACY 97.47%

	HEAVY_RAIN	LOW_RAIN	MODERATE_R	NO_RAIN	VERY_HEAVY_	UNCERTAIN
HEAVY_RAIN	92.0%	0%	0%	0%	4.6%	3.4%
LOW_RAIN	0%	100%	0%	0%	0%	0%
MODERATE_RAI	5.1%	0%	91.5%	0%	0%	3.4%
NO_RAIN	0%	0%	0%	100%	0%	0%
VERY_HEAVY_R/	0%	0%	0%	0%	99.0%	1.0%
F1 SCORE	0.95	1.00	0.96	1.00	0.98	

Figure 6: Model Testing Confusion Matrix

3.4 Rainfall Amount Calculations

In this prototype, the amount of rainfall was calculated by using predefined rainfall intensity values corresponding to different rainfall classifications (HEAVY_RAIN, LOW_RAIN, MODERATE_RAIN, NO_RAIN, and VERY_HEAVY_RAIN). During the live testing, the prototype's classification model predicted the type of rainfall based on audio data. Depending on the classification result, the rainfall amount was incremented by a specific value (intensity in 5 minutes) divided by the measurement period (300 seconds), representing the rain gauge's incremental water amount. This approach allowed for continuous accumulation of rainfall data, providing a real-time estimate of the rainfall amount based on the simulated conditions.

The amount of rainfall is calculated using the formula below.

$$Amount\ of\ Rainfall = Amount\ of\ rain + \frac{Rainfall\ Intensity}{Period\ of\ Rain\ Recorded}$$

4. Model Deployment

The model was deployed as an Arduino library and loaded onto the Arduino Nano 33 BLE Sense for real-time inferencing. This deployment enables efficient, on-device rain monitoring, leveraging the board's built-in sensors and processing capabilities.

4.1 Edge Device

The hardware board used in this development is Arduino Nano 33 BLE as shown in Figure 7. It contains a 32-bit ARM Cortex-M4F microcontroller running at 64MHz with 1MB of framework memory and 256KB Smash. This little regulator gives sufficient ability to utilize Tiny ML models. The Arduino Nano 33 BLE Sense contains variety, splendor, closeness, contact, development, vibration and different sensors. This sensor suite will be all that could be needed for most applications. Table 3 lists the key hardware specifications.

Table 3: Arduino Nano 33 BLE Sense hardware specifications

Component	Specification
Microcontroller	nRF52840 (ARM Cortex-M4 CPU)
Operating Voltage	3.3V
Input Voltage	5V (via USB) or 4.5-21V (via VIN)
Digital I/O Pins	14
Analog Input Pins	8
Flash Memory	1 MB
SRAM	256 KB
Clock Speed	64 MHz
Sensors	Microphone (MP34DT05), Barometric Pressure (LPS22HB), Temperature and Humidity (HTS221)

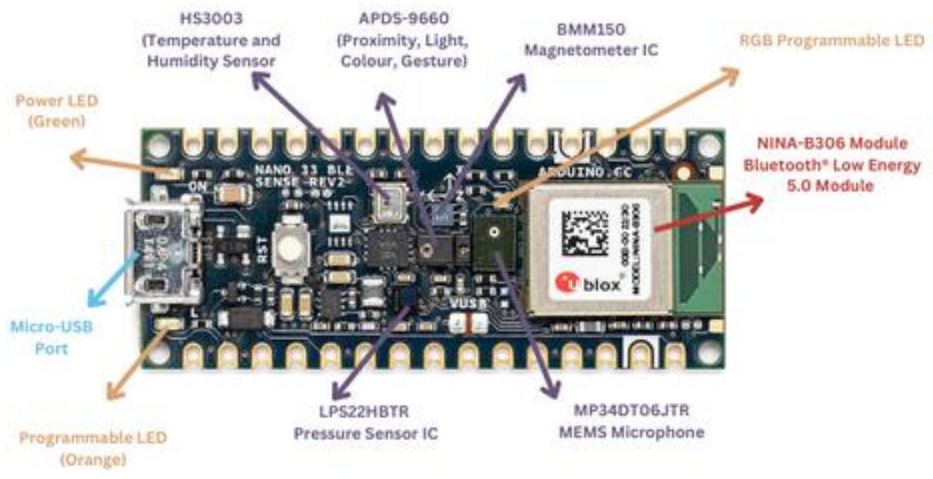

HS3003 (Temperature and Humidity Sensor
APDS-9660 (Proximity, Light, Colour, Gesture)
BMM150 Magnetometer IC
RGB Programmable LED
Power LED (Green)
NINA-B306 Module Bluetooth® Low Energy 5.0 Module
Micro-USB Port
Programmable LED (Orange)
LPS22HBTR Pressure Sensor IC
MP34DT06JTR MEMS Microphone

Figure 7: Arduino Nano 33 BLE Sense

4.2 Sensors

4.2.1 Microphone

The built-in omnidirectional digital microphone (MP34DT05) shown in Figure 8 captures and analyzes sound in real-time, enabling the creation of a voice interface for the project.

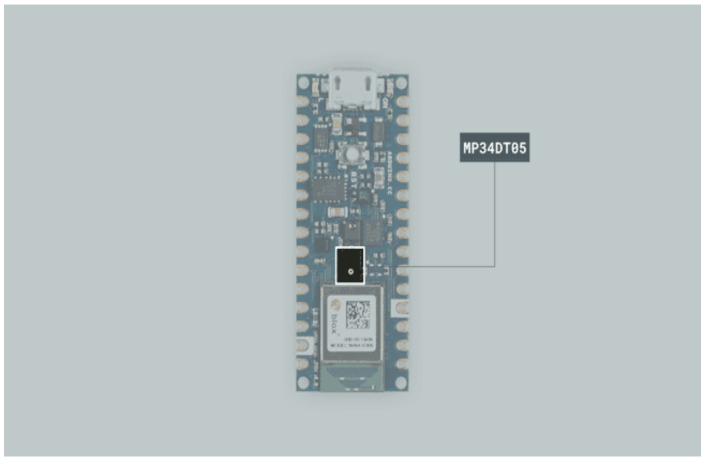


Figure 8: Microphone Sensor

4.2.2 Barometric Pressure Sensor

The LPS22HB sensor shown in Figure 9 detects barometric pressure, providing a 24-bit pressure data output range of 260 to 1260 hPa, essential for accurate weather monitoring.

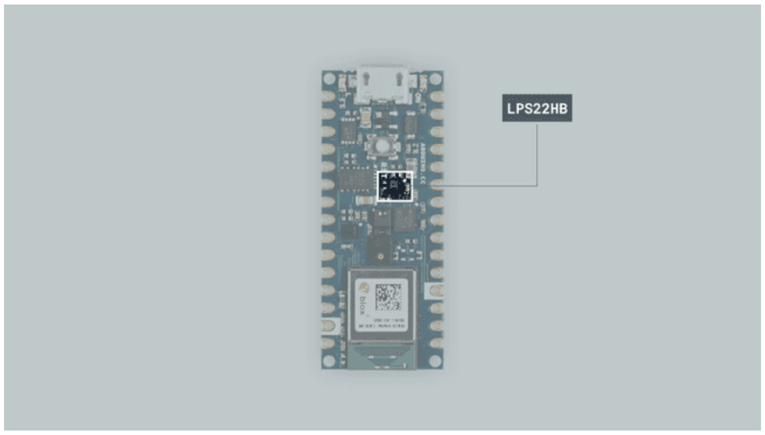


Figure 9: Barometric Pressure Sensor

4.2.3 Temperature and Humidity Sensor

The HTS221 capacitive digital sensor shown in Figure 10 measures relative humidity and temperature with an accuracy of ±0.5 °C, making it ideal for detecting ambient temperature.

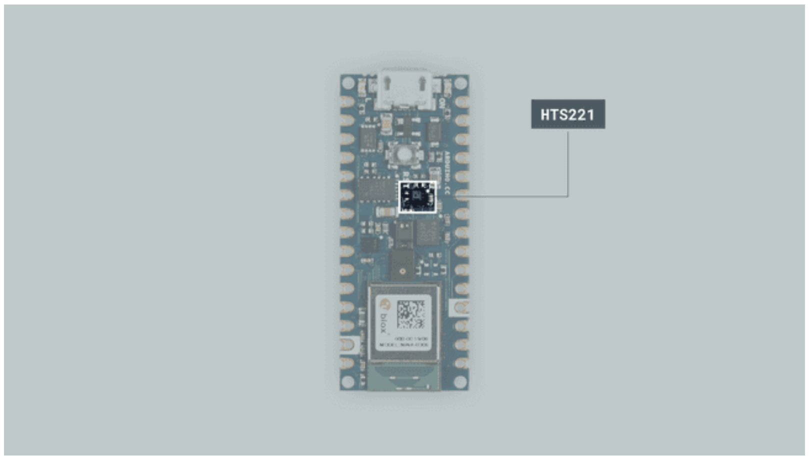
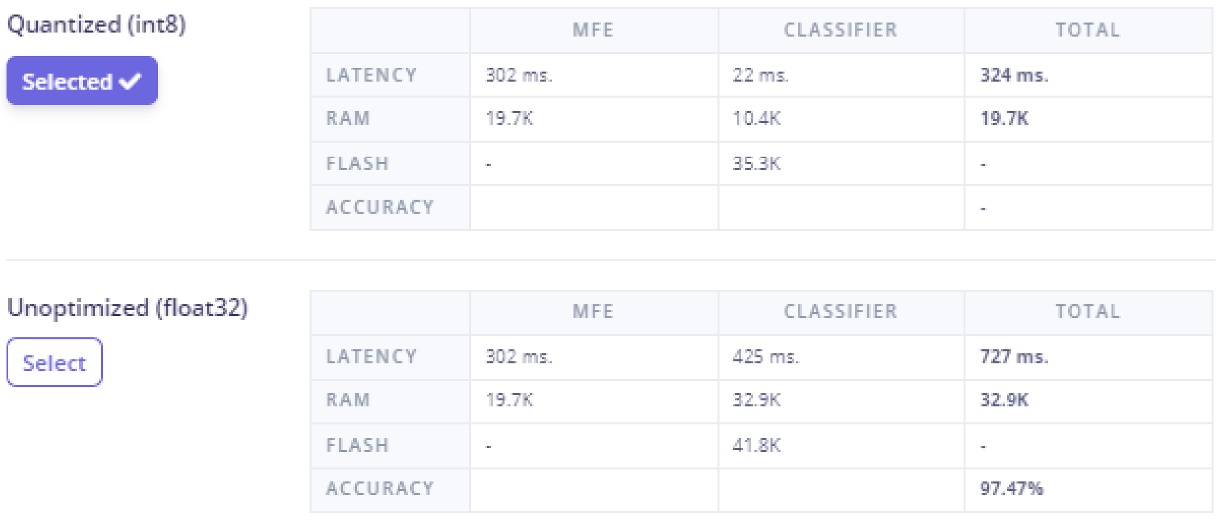


Figure 10: Temperature and Humidity Sensor

4.3 Arduino Library

The trained neural network model is packaged as an Arduino library in ZIP format. This library is added to the Arduino IDE by selecting “Sketch” -> “Include Library” -> “Add .ZIP Library…” and then uploading it to the Arduino Nano 33 BLE Sense. This integration allows the model to run on the board, enabling real-time inferencing. Figure 11 highlights the efficiency improvements achieved by using the EON™ Compiler with quantization, maintaining the same accuracy while significantly reducing both RAM and ROM usage, and improving latency for the classifier with 27% Less RAM, and 39% Less ROM as compared to the specification of the Arduino Nano 33 BLE Sense.



Quantized (int8)

	MFE	CLASSIFIER	TOTAL
LATENCY	302 ms.	22 ms.	324 ms.
RAM	19.7K	10.4K	19.7K
FLASH	-	35.3K	-
ACCURACY			-

Unoptimized (float32)

	MFE	CLASSIFIER	TOTAL
LATENCY	302 ms.	425 ms.	727 ms.
RAM	19.7K	32.9K	32.9K
FLASH	-	41.8K	-
ACCURACY			97.47%

Figure 11: Metrics of Deployment

The results of the model deployment metrics on quantized (int8) and unoptimized (float32) models reveal crucial insights into their performance metrics. Notably, the latency for Mel-Frequency Cepstral Coefficients (MFE) extraction remains constant at 302 ms for both configurations, showcasing that this preprocessing step is independent of model quantization. However, in the quantized model, the classifier latency notably decreases to 22 ms compared to 425 ms in the unoptimized model. This reduction in latency can be attributed to the simplification of arithmetic operations and reduced memory access times facilitated by model quantization.

Regarding RAM usage, both configurations share identical MFE RAM requirements (19.7K), while the classifier RAM usage significantly decreases in the quantized model (10.4K) compared to the unoptimized model (32.9K). This reduction is due to int8's lower memory footprint compared to float32. Similarly, flash memory usage is lower in the quantized model (35.3K) compared to the unoptimized model (41.8K), further emphasizing the efficiency gains of model quantization.

While accuracy is specified only for the unoptimized model (97.47%), quantized models typically experience a slight drop in accuracy. However, the quantized model's benefits in terms of latency, RAM, and flash memory usage make it highly suitable for resource-constrained environments such as embedded systems. Overall, the quantized model demonstrates superior efficiency and is well-suited for deployment in real-world applications where computational resources are limited.

4.4 Inference

Once deployed, the neural network model operates on the Arduino Nano 33 BLE Sense for real-time rain monitoring. The system processes incoming audio signals through the model to classify rainfall intensity. The deployment also includes power optimization measures to ensure the device operates efficiently, extending its runtime.

4.5 Power Optimization

Power consumption optimization is crucial for prolonged operation of the monitoring system without frequent battery replacements. In this project, two boards are used: the microcontroller and environmental sensors. The BLE Sense board is powered by a USB cord providing 5V from a laptop. Initially, an empty code is loaded onto the board to measure baseline voltage and current readings from the VIN (Voltage Input) and GND pins. After loading the model, the board runs inferencing tasks, and voltage and current readings are measured again to calculate power consumption. The key power specifications are tabulated below in Table 5. The Arduino Nano 33 BLE Sense has specific current ratings depending on its operational mode and power source. Here are the key current ratings:

- Idle Mode: The board consumes around 10 mA when idle (not performing significant tasks).
- Active Mode: When running tasks, the current consumption can increase significantly. For typical operations, it can range from 15 mA to 20 mA.
- Peak Usage: During peak usage, such as when performing heavy computations or using the BLE radio extensively, the current can go up to around 25 mA to 30 mA.
- Sleep Modes: In sleep modes, the current consumption drops significantly to around 1 mA or lower, depending on the specific sleep mode and peripherals disabled.

Table 4: Power Specification

State	Voltage (V)	Current (mA)	Power (mW)
Baseline (No model running)	5.0	10	50
Model Running and inferencing	5.0	20	100

If we were to use a battery, we might need to calculate the expected battery life based on the increased consumption. For instance, with a 500mAh battery, we could expect approximately 25 hours of operation (500mAh / 20mA = 40 hours).

In a realistic scenario where the device is not active all the time, the energy consumption can be estimated based on the duty cycle of the device. Assuming intermittent activity, such as periodic data collection and inference tasks, the average power consumption over time can be calculated. For example, if the device operates at full power for 10% of the time and remains idle for the remaining 90%, the average power consumption would be significantly lower than the maximum power consumption stated. By considering the duty cycle and adjusting for periods of inactivity, a more accurate estimate of energy consumption can be obtained for practical deployment scenarios.

4.6 Live Testing

The live testing of the prototype was conducted by placing the device under a shower to simulate rainfall conditions as shown in Figure 12. This setup allowed for realistic testing of the rain intensity classification model. The prototype, connected to a computer via USB, enabled continuous monitoring and data collection as shown in Figures 13 and 14. Using the Arduino IDE's serial monitor as shown in Figure 15, we read the results and the environmental data from the BME sensors in real time. The testing was performed for each rainfall class-HEAVY_RAIN, LOW_RAIN, MODERATE_RAIN, NO_RAIN, and VERY_HEAVY_RAIN—by regulating the water flow from the shower to simulate various rainfall intensities. The rainfall intensities, amounts of rainfall, and BME data were accurately measured and displayed on the Arduino IDE serial monitor. To ensure comprehensive testing, we carefully adjusted the shower settings to replicate different rainfall conditions. For NO_RAIN, the shower was turned off to simulate dry conditions, and the prototype correctly identified the absence of rain. For LOW_RAIN, the shower was set to a gentle drizzle, and the prototype accurately detected this lower intensity. For MODERATE_RAIN, the water flow was increased to a steady stream, and the device successfully recognized this intermediate level of rainfall. For HEAVY_RAIN and VERY_HEAVY_RAIN, the shower was turned to high and very high settings, respectively, allowing us to verify the prototype's ability to distinguish between these higher intensities.

Throughout the testing process, the prototype's sensors recorded additional environmental parameters such as barometric pressure, temperature, and humidity, which were also displayed on the serial monitor. This real-time data collection and display validated the system's functionality, demonstrating its capability to provide accurate and detailed weather information under controlled conditions. The successful testing of the prototype under various simulated rainfall conditions confirms its potential for practical deployment in real-world scenarios.

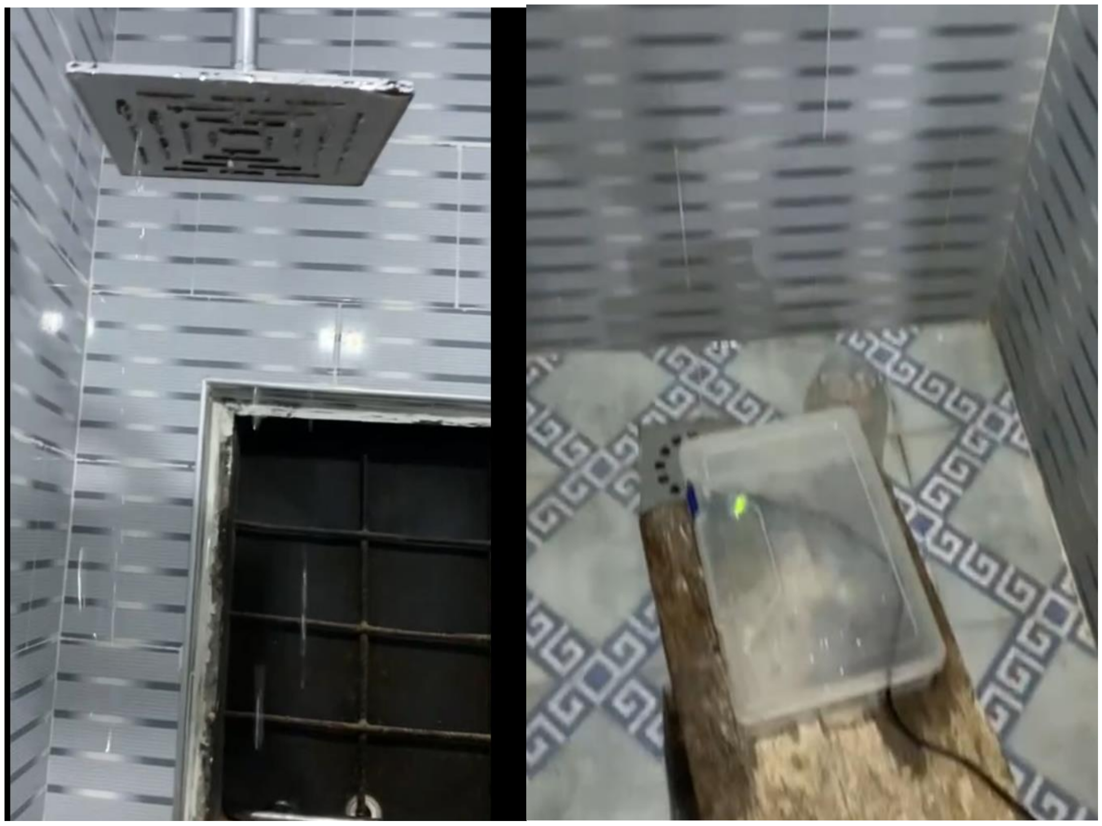


Figure 12: Testing Site

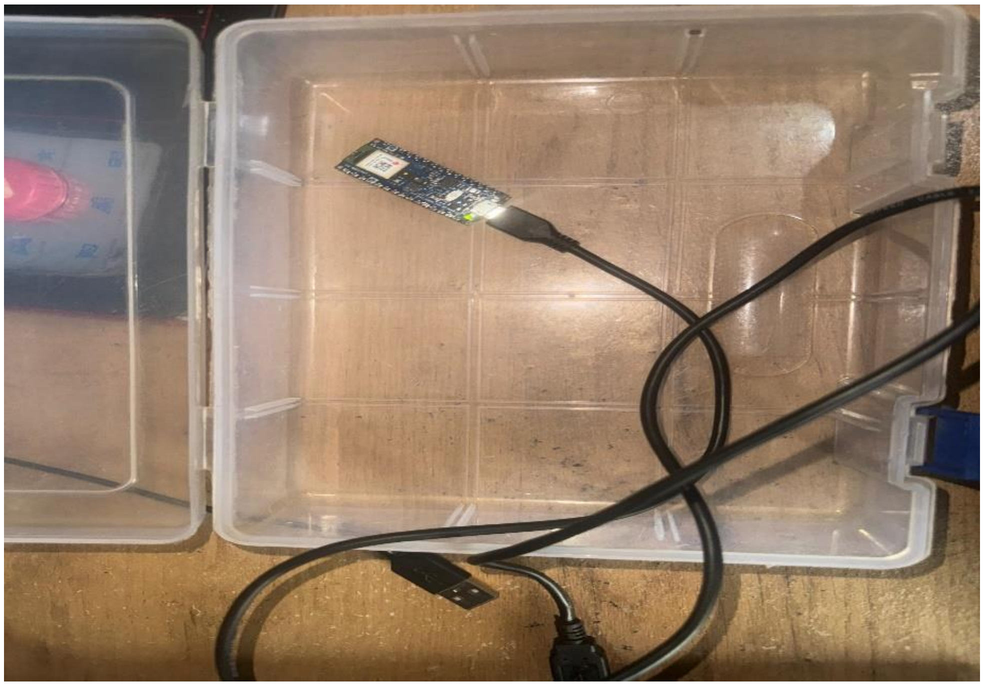


Figure 13: Prototype with Cover Opened

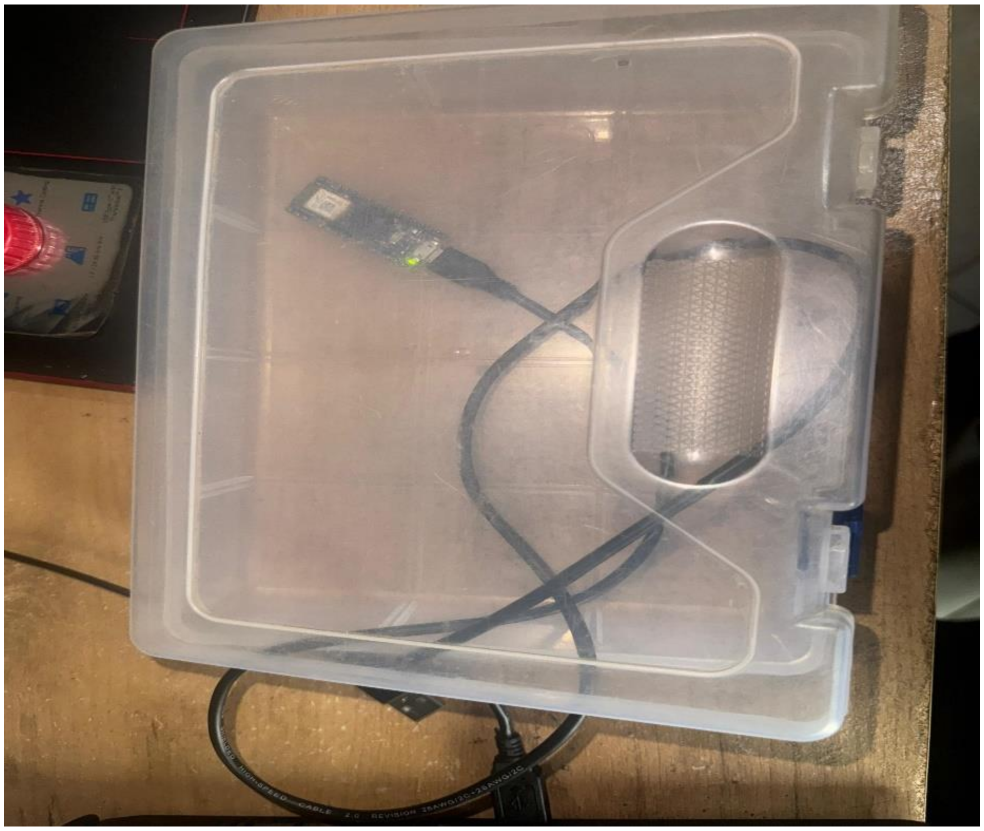
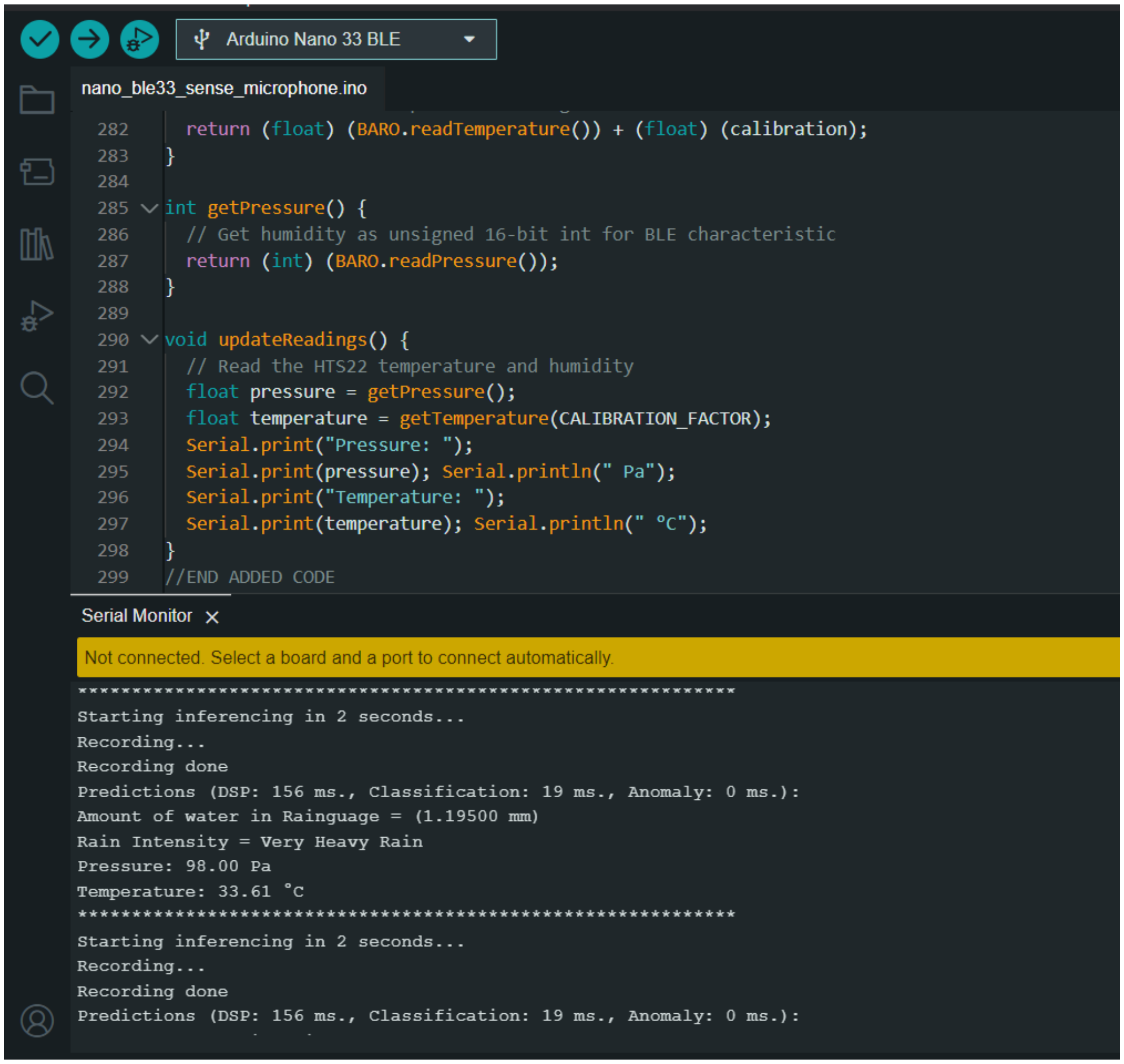


Figure 14: Prototype with Cover Closed



```
  return (float) (BARO.readTemperature()) + (float) (calibration);
}

int getPressure() {
  // Get humidity as unsigned 16-bit int for BLE characteristic
  return (int) (BARO.readPressure());
}

void updateReadings() {
  // Read the HTS22 temperature and humidity
  float pressure = getPressure();
  float temperature = getTemperature(CALIBRATION_FACTOR);
  Serial.print("Pressure: ");
  Serial.print(pressure); Serial.println(" Pa");
  Serial.print("Temperature: ");
  Serial.print(temperature); Serial.println(" °C");
}
//END ADDED CODE
```

Not connected. Select a board and a port to connect automatically.

```
*************************************************************
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 156 ms., Classification: 19 ms., Anomaly: 0 ms.):
Amount of water in Rainguage = (1.19500 mm)
Rain Intensity = Very Heavy Rain
Pressure: 98.00 Pa
Temperature: 33.61 °C
*************************************************************
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 156 ms., Classification: 19 ms., Anomaly: 0 ms.):
```

Figure 15: Arduino IDE Interface with Serial Monitor Displaying Inferenced Result.

5. Conclusion

The Next-Gen tinyML Smart Weather Station Challenge 2024 has provided a platform to innovate and address the crucial need for precise and reliable weather monitoring in agricultural settings. Team GenStorm has risen to the challenge by developing a cutting-edge solution that leverages Tiny Machine Learning (tinyML) to revolutionize weather stations. Our intelligent model for classifying rainfall intensity, meticulously trained using the Edge Impulse platform, has demonstrated exceptional accuracy and efficiency. By deploying the model as an Arduino library on the Arduino Nano 33 BLE Sense, we ensure real-time inferencing capabilities that are both low-cost and low-power, making the solution ideal for remote or resource-limited environments. Our approach includes robust data preprocessing, feature extraction using Mel-filterbank energy (MFE) parameters, and a well-designed 1D convolutional neural network architecture. These components work in harmony to deliver accurate classifications of rainfall intensity, essential for informed agricultural decision-making. The model's high accuracy of 97.47% and its optimized deployment size of 10 KB post-quantization highlight its effectiveness and efficiency. Our focus on power optimization ensures the weather station's prolonged operation without frequent battery replacements, crucial for sustainable deployment in developing regions. By providing real-time local weather data, our solution empowers farmers to make data-driven decisions, enhancing productivity and resilience against climate variability.