# Classifying mosquito wingbeat sound using TinyML

Moez Altayeb
University of Khartoum, Sudan
ICTP, Trieste, Italy
mohedahmed@hotmail.com

Marcelo Rovai
Universidade Federal de Itajubá
Itajubá, Brazil
rovai@unifei.edu.br

Marco Zennaro
ICTP
Trieste, Italy
mzennaro@ictp.it

## ABSTRACT

Every year more than one billion people are infected and more than one million people die from vector-borne diseases including malaria, dengue, zika and chikungunya. Mosquitoes are the best known disease vector and are geographically spread worldwide. It is important to raise awareness of mosquito proliferation by monitoring their incidence, especially in poor regions. Acoustic detection of mosquitoes has been studied for long and ML can be used to automatically identify mosquito species by their wingbeat. We present a prototype solution based on an openly available dataset, on the Edge Impulse platform and on three commercially-available TinyML devices. The proposed solution is low-power, low-cost and can run without human intervention in resource-constrained areas. This insect monitoring system can reach a global scale.

## KEYWORDS

ML, neural networks, TinyML, mosquito

**ACM Reference Format:**
Moez Altayeb, Marcelo Rovai, and Marco Zennaro. 2022. Classifying mosquito wingbeat sound using TinyML. In *Proceedings of tinyML Research Symposium (tinyML Research Symposium'22).* ACM, New York, NY, USA, 6 pages.

## 1 INTRODUCTION

In 300 B.C., Aristotle referred to mosquitoes as *empis* in his Historia Animalium where he documented their life cycle and metamorphic abilities. Since then, mosquitoes have killed billions of people. According to some studies [2], nearly half of the 108 billion people that lived on Earth have been killed by mosquitoes. The mosquito is a nearly universal animal and we have 110 trillion of them across all the planet.

Environmental changes are causing an increase in the number and geographical spread of vectors. Mosquitoes in particular transmits far more diseases than other insects. According to the World Health Organization (WHO) [7], mosquito bites result in the deaths of more than one million people every year with the majority of these deaths due to malaria. The WHO estimates that between 300 and 500 million cases of malaria occur each year and a child dies from malaria every 30 seconds. Around the world, malaria transmission occurs in 97 countries.

Today the majority of malaria deaths occur in Africa, South of the Sahara, where the malaria parasite is very common. The poorest segments of society and least-developed countries are the most affected. People from poor communities with little access to health care and clean water sources are also at risk. Although anti-malarial drugs exist, there's currently no malaria vaccine.

Vector-borne diseases also exacerbate poverty. Illness prevent people from working and supporting themselves and their families, impeding economic development. Countries with intensive malaria have much lower income levels than those that don't have malaria.

Countries affected by malaria turn to control rather than eradication. Vector control means decreasing contact between humans and disease carriers on an area-by-area basis. It is therefore crucial to be able to detect the presence of mosquitoes in a specific area. This paper presents an approach based on TinyML and on low power embedded devices.

## 2 CLASSIFICATION WITH ACOUSTIC SENSORS

Researchers have followed two main approaches to detect the presence of mosquitoes using ML: the use of acoustic sensors or the use of optical sensors. The use of optical sensors does not go in the direction of a low-cost and low-power solution so we will focus on acoustic sensors in this paper.

Acoustic surveillance has been proposed long ago as mosquitoes can be identified using the species-specific frequencies in their wingbeat sounds. Mosquito sounds have low complexity with a fundamental frequency and some overtones and can be analyzed using a Fourier Transform. Male mosquitoes produce a higher frequency sound than female ones and given the slow flying speed the Doppler shift of frequency can be neglected.

As the wingbeat frequency is between 200 and 700 Hz, microphones normally used to record voice (300 to 3000 Hz) can be used with mosquito sounds.

Researchers in [6] have proposed the use of mobile phones as acoustic sensors for high-throughput mosquito surveillance. The widespread adoption of smartphones make this solution interesting and scalable. Citizens have been involved in the recordings in the framework of a citizen science project. The collection of metadata (time of the day, GPS position, etc) is a plus that can be used to further improve the accuracy of classification. A limitation of this solution is given by the power requirement of a smartphone and by the fact that such an expensive device cannot be left unattanded in the field, especially in poor areas. While the proposed solution is great to gather data, it cannot be used for a regular, automatic surveillence system. A dataset with acoustic recording from 20 different species of mosquitoes has been made openly available.

In [8] the authors explored the possibility of using ML to detect Aedes aegypti, a mosquito responsible for transmitting several diseases in poor communities. Their approach is based on the analysis of audio data captured from mosquitoes wingbeats and three distinct formulations for the model using Convolutional Neural Networks (CNN): a single binary classifier (i.e., Aedes aegypti vs non-Aedes aegypti), a 23-class multiclass classifier, and an ensemble classifier that combines the output from 22 base binary classifiers. In their conclusion they say "One challenge is running a simplified but effective version of the model that could operate offline in a smartphone, without posing much overhead in terms of required processing power and battery usage." and this has been a driver for us to design a TinyML solution.

Authors of [9] used Edge Impulse and an Arduino Nano 33 to classify three species of mosquitoes (Anopheles, Aedes and Culex). They use an OLED display to visualize the detected insect. They have not engineered the solution to make it field-ready in terms of power consumption and flexibility, and no communication option is presented. Showing the result on the display is useful for local populations but does not allow for long-term research on mosquito presence.

## 3 PROPOSED SOLUTION

Our contribution is in the development of a TinyML solution that allows the classification of two species of mosquitoes (Aedes aegypti and Aedes albopictus) with the following features:

- Low cost: the proposed solution is cheaper than the existing systems based on acoustic sensors.
- Low power: the solution we developed can live with an external battery for several days, enabling field measurements that are not possible with smartphone-enabled solutions. A small solar panel could extend the system's lifetime.
- Unattended operation: the device can classify mosquitoes without any data transfer to the internet or to a lab PC. There are no humans in the loop.
- Networking: the system can send data via a LoRaWAN network, allowing scientists to analyze the results and to add more sensors (such as temperature and humidity) if needed. LoRaWAN is a low-power data transfer protocol that can reach very long distances, enabling data gathering in isolated regions [3].

## 4 MATERIALS AND METHODS

To develop our solution we followed the workflow shown in Figure 1. This is the typical workflow of ML projects with the addition of the deployment in the embedded device as a final step. The model has to be optimized in terms of memory usage or of latency, as it will run on a device with limited resources.

### 4.1 Data Acquisition

The starting point of our project is the public dataset created in [6]. The dataset includes a collection of 20 different mosquito wingbeat sounds, collected by different people, in different locations, using different mobile phones and having different sample frequencies. From the 20 species, we selected two of the most dangerous mosquitoes, according to WHO, the Aedes aegypti (class: aegypti) and the Aedes albopictus (class: albopictus). Two more classes were included a) mixed samples from the remaining 18 species included on the dataset (class: other) and b) background noise with no mosquito(class: noise). This last class was generated mixing background noise, laboratory background noise collected by the authors, and several one second-long 16 kHz WAV files of various kinds of background noise as near running water or machinery, part of [10].

In summary, we used the following data:

- class: aegypti – 3 files with a total of 11.5 minutes of 44.1KHz/16bits data
- class: albopictus - 2 files with a total of 11.1 minutes of 44.1KHz/16bits data
- class: other – 16 files with a total of 27 minutes of 44.1KHz/16bits data
- class: noise
  Pete Warden's - 243 files with a total of 4.0 minutes of 16KHz/16bits data
  Mukundarajan's – 9 files with a total of 3.1 minutes of 44.1KHz /16bits data
  Autors – 1 file with a total of 4.1 minutes of 16KHz/16bits data

### 4.2 Data Preprocessing

The Data Preprocessing is divided into 2 phases: a) data preparation, where we resampled and splitted all data to end with a uniform selection of .WAV sound files ('Time Series Data') and b) feature generation, where using a sliding window over each one of the raw data samples, we created 'images' of the sound based on their frequency spectrogram. Those 'images' were the input tensor for the model training in the next phase.

### 4.3 Data preparation (resample and split)

Using Librosa[1] and Soundfile[2] both open-source Python packages for audio analysis, we converted all necessary files from the dataset to 16KHz sample rate and cropped them to 1-minute files, to further simplify data split in the training and testing phases. The dataset ended with 297, 1 minute/16Khz/16bits files, split into the 4 classes. The code used for data preprocessing is openly available on GitHub[3].

Analyzing the data of the two mosquitos (Aedes aegypti and Aedes albopictus) confirmed that their main wingbeat frequency is inside a range of 500 and 700Hz as stated in [5] and shown in Figure 2. This also confirmed that we could use a microphone normally used for voice measurements.

The final dataset was uploaded as .WAV files to Edge Impulse Studio, being split as below:

- Training Data (86%):
  class: aegypti – 10m 15s
  class: albopictus – 8m 53s
  class: other – 10m 27s
  class: noise - 9m 54s
- Test Data (14%):
  class: aegypti – 2m 35s
  class: albopictus – 2m 13s
  class: other – 1m 0s
  class: noise - 0m 27s

[1]https://librosa.org
[2]https://github.com/bastibe/python-soundfile
[3]https://github.com/Mjrovai/wingbeat-mosquito-tinyml

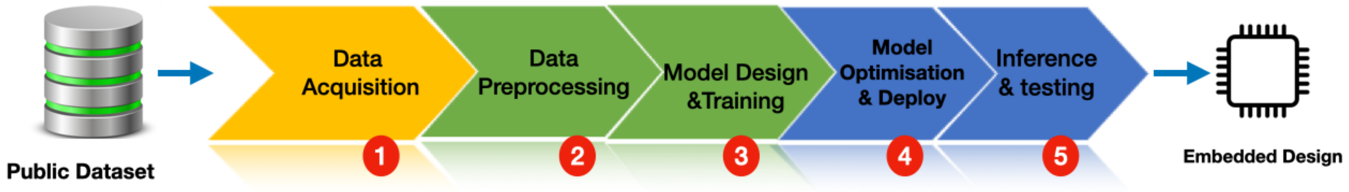

**Figure 1: Workflow diagram of a TinyML project, including the deployment to an embedded device**

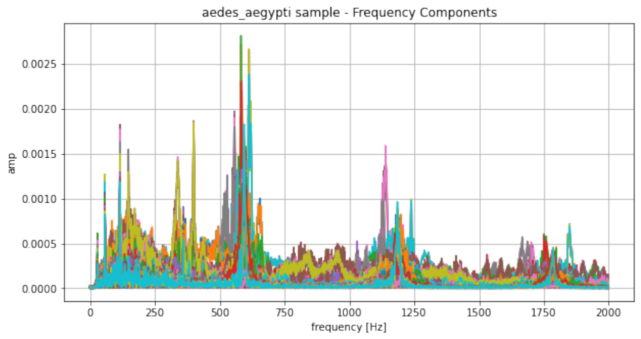

**Figure 2: Frequency components of the Aedes Aegypti**

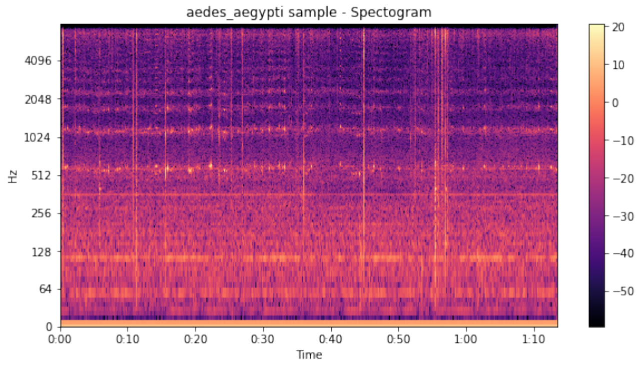

**Figure 3: Spectrogram of a Aedes Aegypti sound measurement**

## 4.4 Feature Generation

Starting with the 1-minute raw data samples, windows of 1, 2, and 4 seconds were tested. We concluded that cropping data with 1 second length ended with better accuracy. We also applied data augmentation, going over the data with a sliding window of 250ms (we also tested with 500ms and 1s, but 250ms ended with higher accuracy).

With the raw features, we proceeded with the Spectrogram extraction. The chosen parameters were:

- Frame length: 25 ms
- Frame Stride: 12.5 ms
- Number of Frequency bands: 128
- Noise Floor Filtering: -52db

Data used in the training ended with around 8,400 instances (or samples).

We also explored extracting the spectrogram from the audio signal using MFE (Mel-Filterbank Energy features), but a simple spectrogram showed better results and lower latency, which is critical for further deployment on embedded devices. Figure 3 shows an example of spectrogram of a Aedes Aegypti sound measurement of one minute.

## 4.5 Model Design

Each one of the 8,400 instances is a 1D image, obtained through the spectrogram creation of a 1-second window of raw data. Each instance or 1D Image has 5,135 features (model input tensor), given by its length (65 columns) times its height (79 lines), as shown in Figure 4.

The model is trained with all 8,403 instances resulting in an output tensor that includes 4 values, each one being the probability

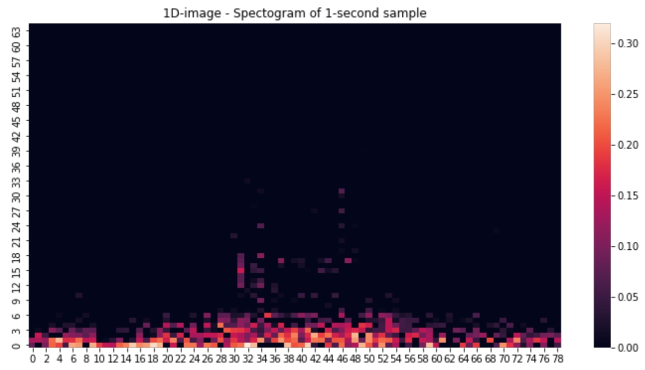

**Figure 4: 1D image, obtained through the spectrogram creation of a 1-second window of raw data**

of that specific class. The vector index is related with one of the classified classes:

$$classes\_values = [\text{"}aegypti\text{"}, \text{"}albopictus\text{"}, \text{"}noise\text{"}, \text{"}other\text{"}]$$

As we are working with images a Convolutional Neural Network (CNN) is usually the right choice for model design. A 2D-CNN can be used for complex image classifications, while for audio a 1D convolution brings a smaller and acceptable solution [1].

The correct architecture including the number of 1-D layers, the pooling layers configuration, the drop-out rate for overfitting prevention, among other hyperparameter definition was obtained on a trail-test basis and also using the EON-Tuner[4], an Edge Impulse Auto-ML tool that helps to find and select the best hyperparameters to be applied to model training. Good results with embedded machine learning models should be understood not only in terms of model accuracy, but also for low latency and limited memory consumption.

The simplest but yet very effective model design, was reached with two 1D-CNN layers with respectively 32 and 64 neurons and a Re-Lu activation function, with each 1D-CNN followed by a Max-Pooling layer of size 2, strides 2 and padding 'same'. After the convolution blocks, a flatten block, a Dropout layer with 50% rate of neuron disconnection and a final Dense layer with 4 neurons followed by a Softmax activation function for final classification.

## 4.6 Model Training

For training, the best result (trail/error) was reached with the following hyperparameters:

- Learning Rate (LR)= 0.001
- Batch Size (BS) = 32
- Initial Epochs (EPOCHS) = 100
- Training early stop patience = 5, monitoring 'val_loss'

We also split the train data, separating 20% of it for validation during training. After 60 epochs, the validation accuracy was 98% as shown in Figure 5. Table 1 resumes the training results.

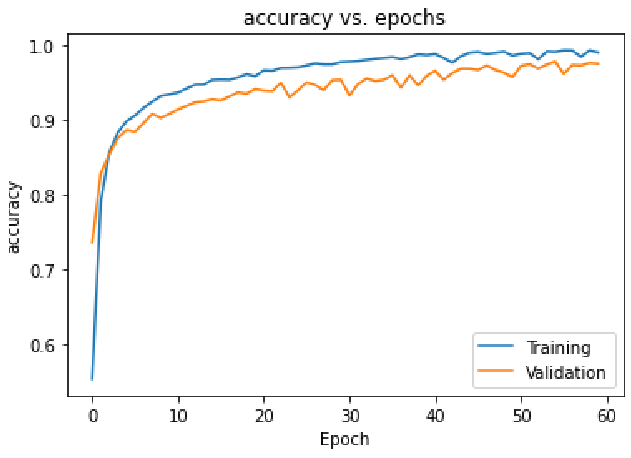

**Figure 5: Accuracy vs epochs**

## 4.7 Model Testing

Applying the test dataset we got around 94% of general accuracy. Table 1 resumes the overall result. The general accuracy with the test dataset was around 4% lower than we got during training, which is expected, and the slight difference seems to confirm that the model is generalizing well. Regarding the F1 score, the albopiticus class gets a higher drop than the aegypti class. This result is given by the the samples coming from 2 distinct smartphones in the case of albopiticus and from 3 smarphones in the case of aegypti.

[4]https://docs.edgeimpulse.com/docs/eon-tuner

| | Train Result | | Test Result | |
|---|---|---|---|---|
| Evaluation / Metrics | Accuracy | F1-Score | Accuracy | F1-Score |
| General Test Result | 98.2% | | 93.8% | |
| aegypti | 98.1% | 0.99 | 97.2% | 0.97 |
| albopiticus | 98.3% | 0.98 | 86.9% | 0.93 |
| noise | 98.3% | 0.99 | 100% | 0.95 |
| other | 98.2% | 0.98 | 99.6% | 0.94 |

**Table 1: Model Train and Test performance**

## 5 HARDWARE SOLUTION

For the prototyping we decided to focus on TinyML devices that have an embedded microphone. Table 2 shows the main characteristics of the three devices: the Arduino Nano 33 TinyML Kit, the Arduino Portenta and the Wio Terminal. We opted for LoRaWAN as communications protocol as it is low power and long distance and it allows for the deployment of devices in remote areas [3]. The Arduino Nano 33 and the Wio Terminal require an external Grove device to be able to send data via LoRaWAN. The Portenta has a LoRa chip included in the Portanta Vision Shield. The microphone technology is quite different in the three solutions. While the Arduino Nano 33 has a single microphone, the Arduino Portenta has two microphones of the same type. The Wio uses a Electret Condenser Microphone. The Wio Terminal has a 2.4" LCD that can be used to visualize text or images.

*5.0.1 Model Optimization.* For the three devices, the trained and tested model was optimized, mainly quantizing all model parameters (from four bytes float32 to one byte int8). The optimization was carried on in two steps, using:

- TensorFlow Lite (TF-Lite) - quantization
- Edge Impulse EON Compiler- to reduce memory use

Table 3 summarizes the estimated on-device performance after optimizations. It is clear that the Portenta, with the faster CPU clock has a shorter latency compared to the other devices.

*5.0.2 Model Deployment.* The optimized model (quantized and optimized with the EON Compiler) was deployed as an Arduino Library that runs on all three ARM-Based devices analysed in this work. The generated .zip library has a size of 4.3Mb , including all needed code to run inference on embedded devices. The Arduino Library was generated using Edge Impulse Studio, based on the Google Framework [4]

## 6 EVALUATION

We started our evaluation by measuring the power consumption of the three devices while operating in inference mode and in data-transmission mode. The Arduino code produced by Edge Impulse was modified to allow the maximum flexibility: the user can decide not to send any data and to come up with a visualization system (LED/display/other), or to send data via a LoRaWAN network. In the latter case, additional information can be sent together with the inference result. For example the temperature and humidity levels could be of interest to scientists working on modelling the presence of mosquitoes in a specific area.

| Name | MCU | Microphone | Memory | Clock speed | LoRa | Price |
|---|---|---|---|---|---|---|
| Arduino Nano 33 TinyML kit | Cortex-M0+ | MP34DT05 | 1MB | 64MHz | External Grove sensor RFM95 module | USD 70 |
| Arduino Portenta H7 | Cortex M7 and Cortex M4 | 2 x MP34DT05 | 16MB | M7 at 480 MHz and M4 at 240 MHz. | External with Arduino Portenta Vision Shield ABZ-093 LoRa Module with ARM Cortex-M0 | USD 153 |
| Wio Terminal | Cortex-M4F | Electret Condenser | 4MB | 120MHz | External Grove sensor RFM95 module | USD 60 |

**Table 2: Technical characteristics of the three TinyML devices**

| Optimation Method | RAM [KB] | ROM Flash [KB] | Latency [ms] Nano | Latency [ms] Portenta | Latency [ms] Wio |
|---|---|---|---|---|---|
| No Optimization | 53.2 | 116.7 | 626 | 123 | 334 |
| Quantized (int8) | 17.1 | 69.3 | 133 | 27 | 71 |
| Quantized + EON | 15.2 | 51.6 | 133 | 27 | 71 |

**Table 3: Estimated on-device performance after optimizations**

## 6.1 Power consumption for inference

We used the OTII device to measure the power consumption of the devices, as shown in Figure 6. OTII is a source measure unit (SMU) with a constant voltage or constant current source, replacing the battery of the embedded devices for a test run. The user can check the statistics of the accumulated energy consumption continuously while recording. Table 4 shows the power consumption of the devices while in inference mode. In this mode the embedded device is continuously listening for the wingbeat sound and will categorize the mosquito type according to the model.

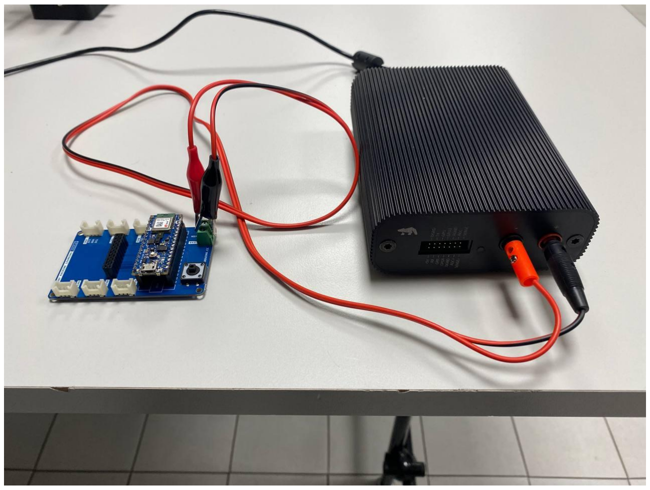

**Figure 6: OTII device measuing power consumption of an Arduino Nano 33**

|  | Arduino Nano 33 | Arduino Portenta | Wio Terminal |
|---|---|---|---|
| Power consumption Inference mode | 13.4mA | 154mA | 91.9mA |

**Table 4: Power consumption while in Inference mode**

## 6.2 Power consumption with data transmission

To measure power consumption with LoRaWAN data transmission we had to analyze the consumption in the two phases of transmission and of reception of LoRa frames, as shown in Figure 7 where we graph the power consumption of the Arduino Nano 33 and of the Wio Terminal. To make the measurements comparable we made sure that the same spreading factor was used during transmission (set to SF7), that the packet size was the same (2 bytes, necessary to send the 4 categories of classification output) and that the transmission sampling rate was set to 60 seconds. The overall power consumption of the three devices if shown in Table 5. As it clearly visible, the Portenta is the most power hungry device of the three, while the Nano 33 is the most low power one.

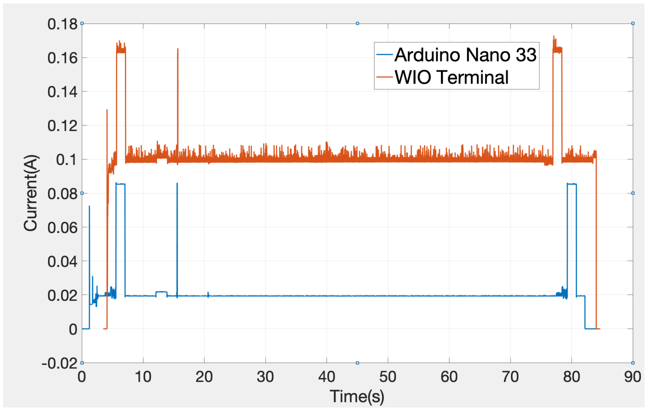

**Figure 7: Power consumption of an Arduino Nano 33 and Wio Terminal during LoRa transmission**

| | Arduino Nano 33 | Arduino Portenta | Wio Terminal |
|---|---|---|---|
| Power consumption LoRa transmission mode | 1.86mWh | 10.1mWh | 8.9mWh |

**Table 5: Power consumption while LoRa transmission mode**

| | Arduino Nano 33 | Arduino Portenta | Wio Terminal |
|---|---|---|---|
| Battery time LoRa transmission mode | 4 days | 13h | 20.1h |

**Table 6: Battery life time**

### 6.3 Estimated battery life

Using the OTII's battery estimation tool we simulated the battery life of the three devices under test by using a 2000mAh LiPo battery and by setting the transmission every 60 seconds. The results of the simulation are summarized in Table 6. A battery life of 4 days would allow the user to leave the device in the field, measuring the presence of mosquitoes and sending the result of the inference via LoRaWAN transmission. A small solar panel would make the system completely sustainable from the energy point of view. The features of sustainability and of communication flexibility has not been provided by other solutions presented in the literature and is, in our view, extremely useful to monitor potential vectors in remote areas.

## 7 DISCUSSION

We have shown that TinyML is capable of categorizing mosquito species by listening to the sound of wingbeats. When considering different hardware solutions, some parameters have to be considered.

The first is cost. While the Nano 33 and the Wio have similar costs when considering the LoRaWAN option, the Portenta is twice as expensive. As in our application latency is not an important parameter, it is not worth choosing this expensive solution.

Another important parameter is power consumption. We have shown that power consumption for both inference and inference plus transmission allow for battery-operated devices for long periods of time. This means that the solutions are field-ready for our application.

The third parameter to be considered is flexibility. The Nano and the Wio have Grove connectors, meaning that additional sensors can be used. This is an important feature as local scientists might want to use sensors for their research on mosquito proliferation. One example could be measuring light, temperature and humidity to try to model the presence of a specific type of mosquito in an area. By using external Grove sensors together with the Grove LoRa module, they can send the data wirelessly. The Grove connector can also be used to add an external display or a buzzer, if the local population has to be alerted about the presence of a dangerous species.

## 8 CONCLUSIONS

We have developed a sustainable solution to categorize mosquitoes in the field using TinyML. Several other solutions based on ML and audio sensors have been presented but the uniqueness of the proposed system is that is adds field-readiness to accurate ML-based categorization. We have shown that commercially available TinyML devices can be used for a week in the field to monitor the proliferation of mosquitoes. As future work we plan to optimize power consumption by developing our bare-bone embedded deice to only measure audio signals and send results via LoRaWAN.

## ACKNOWLEDGMENTS

We want to thank the TinyML4D Academic Network working group for its continuous support and vibrant exchange of ideas.

## REFERENCES

[1] Sajjad Abdoli, Patrick Cardinal, and Alessandro Lameiras Koerich. 2019. End-to-end environmental sound classification using a 1D convolutional neural network. *Expert Systems with Applications* 136 (2019), 252–263.

[2] Timothy C. Winegard. 2019. *The Mosquito: A Human History of Our Deadliest Predator.* Dutton.

[3] Bharat S Chaudhari, Marco Zennaro, and Suresh Borkar. 2020. LPWAN technologies: Emerging application characteristics, requirements, and design considerations. *Future Internet* 12, 3 (2020), 46.

[4] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, et al. 2020. Tensorflow lite micro: Embedded machine learning on tinyml systems. *arXiv preprint arXiv:2010.08678* (2020).

[5] Ivan Kiskin, Adam D Cobb, Lawrence Wang, and Stephen Roberts. 2020. Humbug zooniverse: A crowd-sourced acoustic mosquito dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 916–920.

[6] Haripriya Mukundarajan, Felix Jan Hein Hol, Erica Araceli Castillo, Cooper Newby, and Manu Prakash. 2017. Using mobile phones as acoustic sensors for high-throughput mosquito surveillance. *Elife* 6 (2017), e27854.

[7] World Health Organization et al. 2014. *A global brief on vector-borne diseases.* Technical Report. World Health Organization.

[8] Marcelo Schreiber Fernandes, Weverton Cordeiro, and Mariana Recamonde-Mendoza. 2020. Detecting Aedes Aegypti Mosquitoes through Audio Classification with Convolutional Neural Networks. *arXiv e-prints* (2020), arXiv–2008.

[9] Kirankumar Trivedi and Harsh Shroff. 2021. Identification of deadliest mosquitoes using wing beats sound classification on tiny embedded system using machine learning and edge impulse platform. *Kaleidoscope Academic Conference Proceedings 2021* (2021), 117–122.

[10] Pete Warden. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. arXiv:1804.03209 [cs.CL]