LAMOC: DEVELOPMENT OF AN IOT-BASED ACOUSTIC SIGNAL SENSING DEVICE IN MOSQUITO MONITORING USING RASPBERRY PI

JAROD L. FIANZA, AYEN UNICE M. MANGUAN, AND

JONATHAN CARLOS B. PONIO

Philippine Science High School - Central Luzon Campus

A secondary Science, Technology, Engineering, Mathematics (STEM) research submitted to the Philippine Science High School – Central Luzon Campus in partial fulfillment of the requirements for the subject Research 3.

May 2021

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

Abstract

Mosquito monitoring is essential for limiting the spread of mosquito-borne diseases using vector control. However, existing methods are insufficient due to their unsustainability and risky manual labor. This study develops the Low-cost Acoustic Signal Sensing Mosquito Observation Channel (LAMOC) to identify the presence of mosquitoes (*Culex sp.*, *Anopheles sp., Aedes aegypti*) through bioacoustics and wingbeat frequencies acquired from audio recordings. Internet of Things (IoT) was also used to send the data to a server and display it onto a website. LAMOC utilizes the Raspberry Pi for core processing, Python for program development, and PHP and MySQL for the development of the server and website. Fast Fourier Transform (FFT) and support vector machine (SVM) were used to classify the mosquito species. Cross-validation was used to determine the multi-class classification accuracy, and it was found that *A. aegypti* (male) (99.30%) has the highest accuracy while background noise (90.47%) has the lowest. Overall, LAMOC achieved an accuracy of 94.74% which exceeded existing devices for mosquito monitoring. Further studies can explore other designs for enclosures, utilize higher quality microphones, and examine more machine learning techniques.

Keywords: mosquito monitoring; bioacoustics; machine learning; wingbeat frequency; dengue

Introduction

Background of the Study

Mosquitoes are considered as one of the most lethal animals in the world, carrying numerous diseases that give rise to millions of deaths every year (World Health Organization, 2016). They transmit multiple fatal diseases such as dengue, yellow fever, chikungunya (carried by *Aedes aegypti),* malaria, filariasis (*Anopheles sp.)*, and Japanese encephalitis (*Culex sp.)* (Tolle, 2009; European Centre for Disease Prevention and Control, 2016). Tropical countries, such as the Philippines, are ideal breeding grounds for mosquitoes because of the favorable temperature, rainfall, and humidity levels (Huber et al., 2018).

Mosquito monitoring, the process of tracking and recording changes in mosquito population, is used to analyze if the current control strategy is effective and gives information on the possible diseases mosquitoes may carry to determine what public health measures must be taken (Tolle, 2009). Currently, the most widespread approach is the traditional method of mapping out transition intensities using human-landing catch. Despite its proven effectiveness, this approach is both unsustainable and potentially dangerous for the surveyor due to its repetitiveness and the need for close contact with the insect (Hiscox, 2014).

An alternative solution is the use of acoustic signal sensing, or bioacoustics, wherein flight acoustic recordings of insects (in this case, mosquitoes) are analyzed (Whytock & Christie, 2016). In flight, insects create a sinusoidal tone based on the frequency of their wing movement (Arthur et al., 2013). Mosquitoes, in particular, have varying wing beat frequencies that set them apart from other insects (Robert, 2009); thus, it is possible to identify if a certain animal is a mosquito through acoustic recordings. Despite the potential of bioacoustics in animal monitoring, current

commercially available acoustic sensing devices are expensive even for researchers (Whytock & Christie, 2016).

A viable alternative to bioacoustic devices are single-board computers. These computers contain multiple ports for different microphones, storage options, and environmental sensors needed for a bioacoustic device (Avisoft, 2020). Single-board computers have found applications in the industrial market, smart devices, and even military equipment due to their high processing power, compact size, and high customizability (Johnston et al., 2018). The most popular single-board computer is the Raspberry Pi due to its low price, ease of use, and promising features (Vujović & Maksimović, 2014).

Interconnecting devices can access information across different platforms via an established server (Gubbi et al., 2013). This can be done with the use of Internet of Things (IoT) technology, which allows objects to communicate with one another (Rushikesh & Sivappagari, 2016). Additionally, its wide support from the community and easy application makes it a good choice for multiple projects. Application of this technology will allow remote real-time access of data. This can be done through a website or application connected to the server of the single-board computer.

Statement of the Objectives

The study developed a device, Low-cost Acoustic Signal Sensing Mosquito Observation Channel (LAMOC), to detect the presence of disease-carrying mosquitoes. Specific objectives include:

1. To develop a device that can identify the presence of mosquitoes from their wingbeat frequencies using a single-board computer;
2. To obtain mosquito species classification accuracy using confusion matrices and multi-class classification
3. To send collected data (audio, temperature, humidity, metadata) to a server using Internet of Things (IoT) technology that can be accessed through a developed website

Significance of the Study

Mosquito monitoring provides information on the transmission risks of the diseases carried and the effectiveness of the current method of population control (Hiscox, 2014). Few studies have been conducted on the prevalence of mosquitoes in both rural and urban areas in the Philippines (Sia Su et al., 2014). Instead of performing the traditional method of manual work in mosquito monitoring using mosquito traps, local government units and community health centers can use LAMOC to make the process automated, more sustainable, and efficient. This will inform them on when and where the said disease-carrying mosquito species are increasing in population in order to formulate countermeasures and prepare for a possible mosquito-borne disease outbreak. Moreover, this helps in lowering the number of diseases transmitted by mosquitoes. The user-friendliness and comprehensibility of the device and interface will also make the analysis of the data much easier for the user. Furthermore, the use

of IoT eliminates any possible risk in conducting mosquito monitoring manually in a certain place, as all the data is easily accessible on any device in real-time. Moreover, this study is a significant addition to existing research on mosquito monitoring (leaning towards the traditional methods), and acoustic signal sensing (insufficient research in technology applications, especially in device development and construction).

Scope and Limitations of the Study

The main focus of the study is to develop a mosquito monitoring device; thus, any acoustic recordings that are not identified as coming from mosquitoes are not analyzed. The program will classify these recordings as either belonging to a mosquito or not. Only three mosquito species, namely *Culex sp.*, *Anopheles sp.*, and *Aedes aegypti*, are used for species identification training. These species are chosen because they are the most common species of disease-carrying mosquitoes (Tolle, 2009).

While multiple devices are preferred, a single module is enough to prove that LAMOC can identify the presence of mosquitoes and display information to a website. Unfortunately, the device was not tested under laboratory conditions (breeding mosquitoes for recording audio) due to the associated health risks. Instead, the audio samples needed for program training were acquired from other researchers with similar samples.

Lastly, due to the limited scope of the study, only the chosen feature extraction method and machine learning algorithm are used; there is no comparison between multiple kinds and techniques of each category.

Review of Related Literature

This chapter provides an extensive review of the various variables in the study. It will explore the history of mosquito monitoring, application of bioacoustics in technology, usage of single-board computers (namely, Raspberry Pi) in various researches, emergence of Internet of Things, the programming languages needed to develop the device and its program.

Mosquitoes

Mosquitoes, from the order Diptera and family Culicidae, comprise of approximately 3,500 species worldwide. According to Tolle (2009), among the most common species that carry diseases are *Aedes aegypti* (yellow fever, dengue, chikungunya), *Anopheles sp.* (malaria, filariasis), and *Culex sp.* (West Nile, Japanese encephalitis, filariasis) (see Figure 1). In order to determine if the current method of regulation of mosquito population is adequate, mosquito monitoring is needed (Tolle, 2009).

Figure 1. *Anopheles annulipes* (left), *Aedes aegypti* (middle), *Culex annulirostris* (right)

Noted. Adapted from NSW Arbovirus Surveillance & Vector Monitoring Program, by S.L. Doggett (2003) & Dept. Medical Entomology (2002). Retrieved from http://medent.usyd.edu.au/arbovirus/mosquit/photos/mosquitophotos.htm

Mosquito monitoring. Effective monitoring of the insect's population can provide vital information about the disease transmission risks and the effectiveness of

the vector control. This is done by mapping out the transmission intensity of the disease in a certain area at a particular time (Hiscox, 2014). The data collected are crucial in understanding the epidemic potential brought by the insect and in preparing for effective control strategies for the diseases carried (Kröckel et al., 2006).

Human-landing catch is considered as the most straightforward, dependable, and favored method because it measures the frequency of contact between humans and mosquitoes (Dia et al., 2005). However, it poses a number of risks, such as being dependent on the skills of the mosquito collector and the associated health concern on the direct contact with the insect. Due to its hazards and repetitiveness, other methods of mosquito monitoring had to be explored (Dia et al., 2005).

In 2017, a new approach in mosquito monitoring was explored which utilizes bioacoustics (Mukundarajan et al., 2017). Since there are varying wing beat frequencies for each species of mosquito, acoustic surveillance was proposed as a method of mosquito monitoring. Using mobile phones as audio and location recorders, a crowd-sourced, continuous, and large-scale data acquisition of the sounds of mosquitoes in flight was possible. The study proposed a promising new method for mosquito surveillance.

In this study, the concept of acoustic signal sensing, adapted from the aforementioned study of Mukundarajan et al. (2017), will be used as the method of mosquito monitoring. Instead of mobile phones, however, single-board computers will serve as the main component and recorder of the constructed device. This is due to the unnecessary pre-installed applications and lower power consumption compared to modern devices (Anwaar & Ali Shah, 2015).

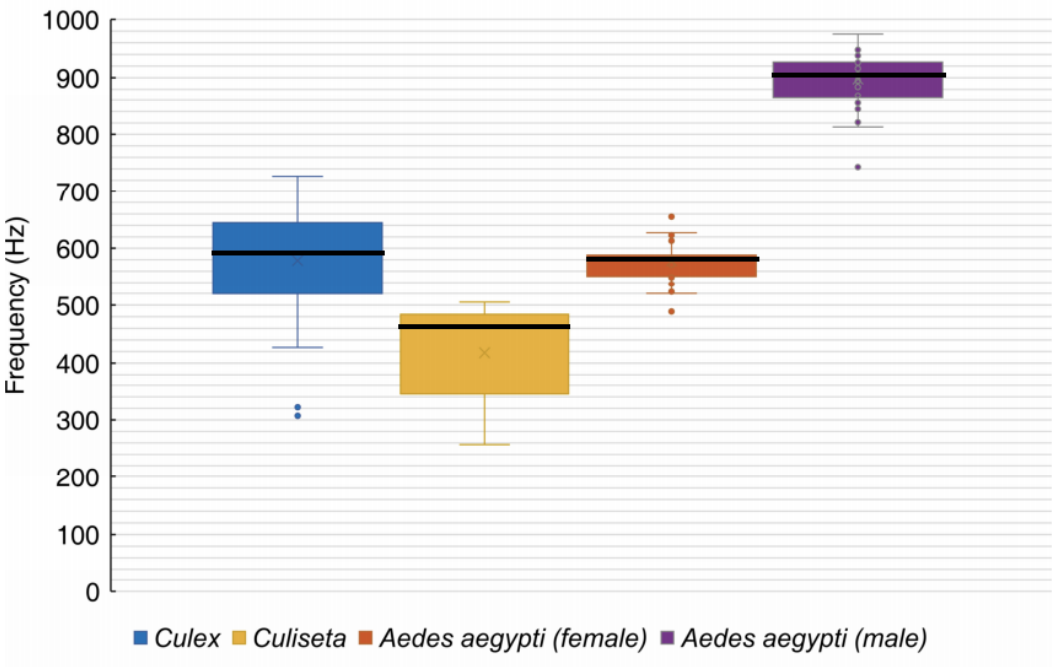

Figure 2. *Wing beat frequencies for selected mosquito species*

Note: Reprinted from "LOCOMOBIS: A Low-cost Acoustic-based Sensing System to Monitor and Classify Mosquitoes" by D. Vasconcelos, N. Nunes, M. Ribeiro, C. Prandi, & A. Rogers, 2019, 2019 16th IEEE Annual Consumer Communications & Networking Conference, 1-6.

Varying wing beat frequencies of four species of mosquitoes from Vasconcelos et al. (2019) are shown in Figure 2. The average wing beat frequencies of the species *Culex* is 520 to 620 Hz, species *Culiseta* is 340 to 580 Hz, *Aedes aegypti* (female) is 540 to 580 Hz, and *Aedes aegypti* (male) is 880 to 920 Hz (Vasconcelos et al., 2019). The average wing beat frequency of species *Anopheles* is 320 to 480 Hz (Caprio et al., 2001). These values will be used as references for the samples and data that will be collected in this study.

Bioacoustics

In the field of bioacoustics, the calls and songs of animals are studied in an effort to discover patterns in their behavior, communication, and mating (McGregor, 2012).

Bioacoustics often finds its applications in the field of technology, specifically in tackling biodiversity. The Automated Remote Biodiversity Monitoring Network (Aide et al. (2013) developed their own hardware and software for automating data acquisition, data management, and species identification based on audio recordings. A website that relays the data in real-time to a project server was also developed. However, the wide scope of animals it can identify makes it difficult to improve on.

Whytock and Christie (2017) developed the Solo audio recorder, an open-source, inexpensive, and customizable system for collecting long-term and high definition audio data. It was proposed as an alternative to the expensive, rarely user-serviceable, and uncustomizable commercial bioacoustic devices. Instead of creating an entirely new program for recording audio from scratch, the software of the Solo audio recorder may be used. The study utilized the Raspberry Pi, one of the most popular single-board computers in the market, as the main component of the device.

The study of Vasconcelos et al. (2019) focused on developing a bioacoustic sensing system that monitors and classifies exclusively mosquitoes, namely *Culex sp.*, *Culiseta sp.*, and *Aedes aegypti*. Instead of using the Raspberry Pi (the device utilized by Whytock and Christie in 2017), this study utilized the Particle Photon microcontroller.

In this study, bioacoustics will be used in analyzing the acoustic recordings acquired from mosquitoes only, much like the system constructed by Vasconcelos (2019). The research will classify the most common mosquitoes in the Philippines, namely *Aedes aegypti*, *Culex*, and *Anopheles* (Tolle, 2009), as opposed to the project of Vasconcelos (2019) which classified *Culiseta* instead of *Anopheles*. The constructed device will only focus on examining recordings obtained in the deployed area.

Single-board computers

Table 1. *Comparison between commercially available single-board computers. Reprinted from "Commodity single board computer clusters and their applications," by S.J. Johnston et al., 2018, Future Generation Computer Systems 89, 203.*

Board	RAM	Price (Php*)	I/O
Raspberry Pi 1 B+	512 MB	1 500	Audio, Ethernet, GPIO, HDMI, microSD, USB2
Raspberry Pi 3 B+	1 GB	1 750	Audio, Bluetooth, Gigabit Ethernet, GPIO, HDMI, microSD, USB2, WiFi
Raspberry Pi Zero W	512 MB	500	Bluetooth, GPIO, HDMI, I2C, I2S, microSD, SPI, USB2, WiFi
Odroid C2	2 GB	2 300	eMMC/microSD, Gigabit Ethernet, GPIO, HDMI, IR, UART, USB
OrangePi Plus 2	2 GB	2 450	Audio, Gigabit Ethernet, GPIO, HDMI, IR, TF, USB, WiFi
BeagleBone Black	512 MB	2 750	ADC, CANbus, Ethernet, GPIO, HDMI, I2C, eMMC/microSD,
UP Squared	< 8 GB	14 450	ADC, Gigabit Ethernet (x2), GPIO, HDMI, mini-PCIe/m-SATA, MIPI (x2), RTC, USB2, USB3
Xilinx Z-turn	1 GB	5 950	CANbus, Gigabit Ethernet, HDMI, TF Card, USB2-OTG, USB_UART

* 50 Php = 1 USD

Single-board computers pack an entire digital computer onto a single circuit board, complete with microprocessor(s), memory, Input/Output (I/O) and other features required of a functional computer (Johnston et al., 2018). Due to their relatively small size paired with high processing power compared to microcontrollers, they have been utilized in environmental applications, smart artificial vision systems, smart appliances, and smart cities (Gómez, 2015).

There have been multiple single-board computers in the market since 2003; however, the Raspberry Pi was the first to be generally available to the public (Johnston et al., 2018). With its low-cost, better features, and widely debugged software, the Raspberry Pi has since led the market (Whytock & Christie, 2017).

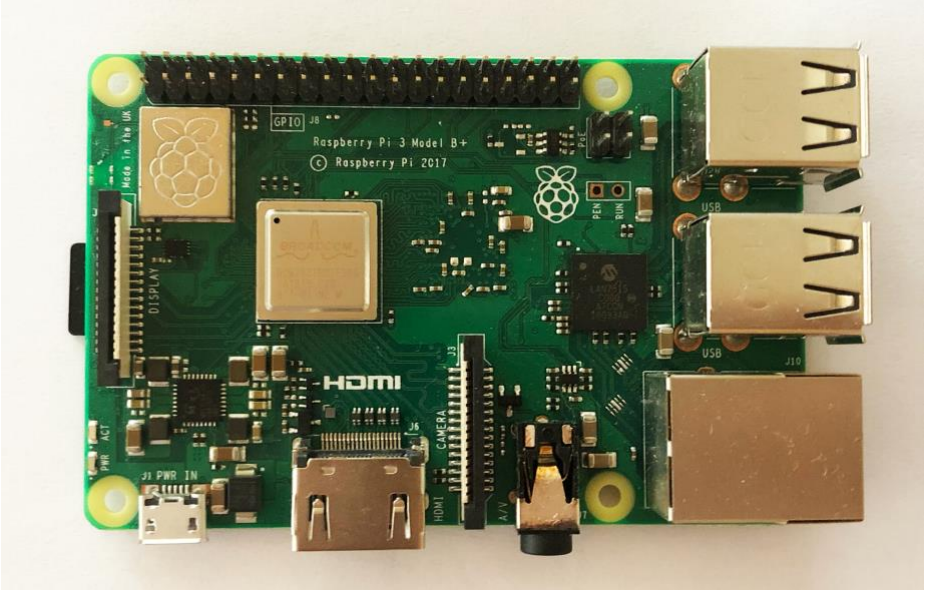

Figure 3. *Raspberry 3B+, a type of single-board computer, released in March 2018.*

Raspberry Pi. The Raspberry Pi, pictured in Figure 3, is a relatively cheap, flexible, and credit-card sized single-board computer that was originally designed for educational purposes. Its features include multiple USB ports for input (external microphone, cooling fan, keyboard, mouse, among others), built-in WiFi, and GPIO pins for other peripherals (clock modules, sensors, etc.) (Vujović & Maksimović, 2014). The main programming language used in development for the Raspberry Pi is Python; however, it also supports other popular languages such as Java, C, C++, among others (Sachdeva & Katchii, 2014). Due to its low price and ease of use, the Raspberry Pi has become one of the most used devices in the development of technology-based studies.

Flores et al. (2016) developed a precision agricultural monitoring system using the Raspberry Pi and deployed the said device in Cagayan de Oro, Philippines. The Raspberry Pi collects the data, sends them to sensor nodes, and stores it in a local database. Whytock and Christie (2017) also utilized the Raspberry Pi as the main component of the Solo audio recorder. The device is a good alternative to expensive

bioacoustic recorders in the market due to the low price of the single-board computer. Due to the features offered by the Raspberry Pi, the system is also highly customizable.

The study will utilize the Raspberry Pi, the single-board computer used by Whytock and Christie in 2017, as the main component of the device. Specifically, the Raspberry Pi 3B+ (see Figure 3) will be used due to its availability and features relative to its price (see Table 1).

Internet of Things (IoT)

Internet of Things (IoT) is an emerging technology which allows objects to communicate with one another, to approach information on the internet, to store and collect data via cloud, and to collaborate with other users (Rushikesh & Sivappagari, 2016). With the help of IoT, there can be an interconnection of multiple devices that report, monitor, or provide other data or services (Atabekov, He, & Bobbie, 2016).

In a study conducted by Koshti and Ganorkar (2016), Arduino and Raspberry Pi were both used to implement a real-time monitoring system which observes a patient's pulse rate. An embedded web server was generated in the Raspbian OS of the Raspberry Pi in order for the user to access the data and the equipment remotely.

A system that controls electrical appliances using a smart phone operated android app was also designed (Shroff et al., 2017). Raspberry Pi 3B+ was used to perform IoT which sends the data captured from the web camera to the user with the android app. Raspberry Pi 3B+ has built-in WiFi unlike Raspberry Pi 2 and Arduino, making it an ideal device for IoT.

For this study, IoT will be used to gather data even without being physically present to check the deployed device. The study will also utilize Raspberry Pi 3B+, due to its built in WiFi, to perform IoT. Raspberry Pi 3B+ will send the collected data from LAMOC to the web server.

Apache. The Apache HTTP Server, most commonly known as simply Apache, is an open-source web platform that is currently dominating the public internet market. The software is also free to use and exhibits great performance (Koshti & Ganorkar, 2016; Fielding & Kaiser, 1997). This study will utilize Apache, but will employ the latest version as of writing (version 2.4) compared to Koshti and Ganorkar (2016) who used version 2.2. It will be used due to its ease of use in the Raspberry Pi.

Programming Languages

Python. Python is an interactive, interpreted, and object-oriented programming language which provides a vast amount of features, such as high-level data structures, dynamic typing, modules, classes, exceptions, among others. Extension modules, also known as packages or libraries, can be installed in order to extend the language with new code (Sanner, 1999). For the past decade, SciPy, a Python library, is extensively used for interactive, exploratory, and computation-driven scientific research (Millman & Aivazis, 2011).

The project of Vaidya et al. (2017) utilized Python in order to create a smart home automation system with Internet of Things (IoT) functionality. It also uses the SciPy of Python for facial recognition with machine learning models.

In this study, Python will be used in order to develop the program for the mosquito monitoring device. This is because Python is the main programming language for Raspberry Pi.

MySQL & PHP. MySQL is a small and compact database management system, while PHP allows the development of dynamic content that can interact with databases. These two platforms are often used in conjunction for developing web-based software applications (Koshti & Ganorkar, 2016).

Rajalakshmi and Mahalkshmi (2016) utilized PHP in a study that developed an IoT-based crop-field monitoring and irrigation system. The language was used to parse the data and display it on an app. MySQL was used to develop the database system.

The study will use these two platforms in order to develop an IoT platform and display it on a website instead of an application. This is due to the popularity and wide support for the two platforms in web development.

Machine Learning

Machine learning is explained as an application of artificial intelligence (AI) which gives the ability to enhance itself without the need of direct programming to systems (Expert System, 2017).

Feature Extraction. Feature extraction can be accomplished by transforming a signal to a simpler parametric representation. It is important in the field of machine learning because it converts a huge amount of raw data to a much simpler form; hence, it eliminates inefficiency and increases the machine learning algorithm's accuracy (Alim & Rashid, 2019).

Frequency Domain Analysis. Frequency domain analysis represents signals in the form of magnitude vs frequency plot. In other words, this analysis shows the process on how the energy of a signal is transmitted over a range of frequencies. Moreover, frequency domain has details on the change of the phase that must be applied in each component of the frequency to retrieve the original time signal combined with all of the individual frequencies (Proakis & Manolakis, 1996; Alim & Rashid, 2019).

This study will pattern its frequency domain analysis with the Alim and Rashid's (2019) methodology wherein an audio signal will be the input which will then be divided into frames. Fast Fourier Transform (FFT) will then be administered in each frame to derive the frequency domain of the audio signal.

Support Vector Machine (SVM). Support vector machine, a supervised learning algorithm, is often used for problems regarding categorization. Supervised learning algorithm learns from a labeled training dataset and makes an inferred function that predicts the output.

The goal of SVM is to locate a plane with the maximum margin. This margin can be the largest distance between the data points of various classes. Maximized margin distance allots reinforcement to give a more confident classification for future data points. SVM implements the idea of mapping the input vectors of two classes into some higher N-dimensional space (where N pertains to the total number of features). These vectors can be distinctly classified by the optimal hyperplane, in which it is a linear decision function that has the highest maximum margin among the vectors of two classes (Cortes & Vapnik, 1995; Gandhi, 2018). Consequently, Cortes and Vapnik (1995) noticed that constructing the optimal hyperplane would only require a few training data (these are called support vectors) which ideally creates a line that is on the same line with other support vectors within the same class.

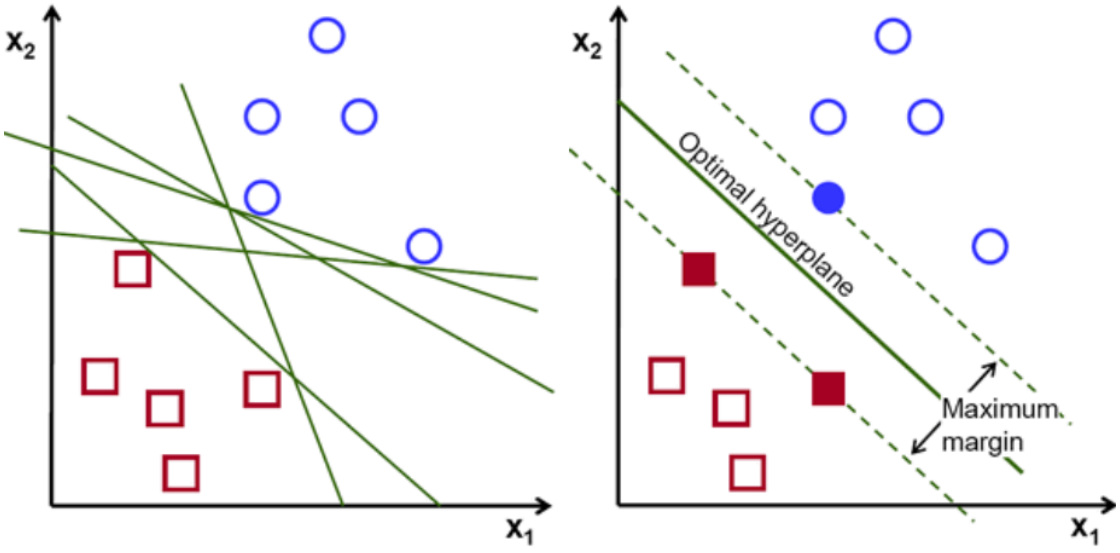

Figure 4. *Training data points which are designated as support vectors*

Note. Adapted from Support Vector Machine - Introduction to Machine Learning Algorithms, by Gandhi, R. (2018). Retrieved from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Even though the nature of an SVM is a binary classifier, there are many ways that can be done for it to categorize a sample from multiple classes. One such case was in the study of Li et al. (2017) where SVM was able to perform a multi-class classification. One-versus-one multi-class strategy was chosen, wherein a sample was tested to all classes, and the class that became positive in all the tests performed was labeled as the sample's class.

In the research conducted by Cao et al. (2019), different machine learning algorithms were compared as to which algorithm is more effective to use. Results showed that SVM is generally used for classification problems. In addition, SVM is advantageous since it is easy to use, can be optimized globally, and only requires a small sample size to perform.

Liu & Cheng (2017) employed SVM to classify the cognitive diagnosis with different attributes. With relatively fewer sample sizes, the use of SVM gave on par results in terms of the classification accuracy rates with those acquired existing researches, both in the attribute and pattern levels.

In this study, SVM will be used in order to identify the mosquito species. Based on the review above, it is the most fit algorithm to use according to the studies mentioned above.

Caruana and Niculescu-Mizil (2004) states that mosquito monitoring can help the community by identifying when and where the said disease-carrying mosquito species are increasing in population. The traditional method of human-landing catch is the most straightforward, dependable, and recommended method, however, conducting

this method poses a lot of risks. With the implementation of bioacoustics, it can eliminate all these risks and ensure the accuracy of the monitoring.

Using a Raspberry Pi 3B+ as the principal component of the device is practical due to its availability and features relative to its price. With its support for different kinds of advance-leveled programming languages (C, C++, Java, and Python) and its powerful CPU and program memory, the implementation of Frequency Domain and Support Vector Machine would be possible.

Methodology

Process Flowchart

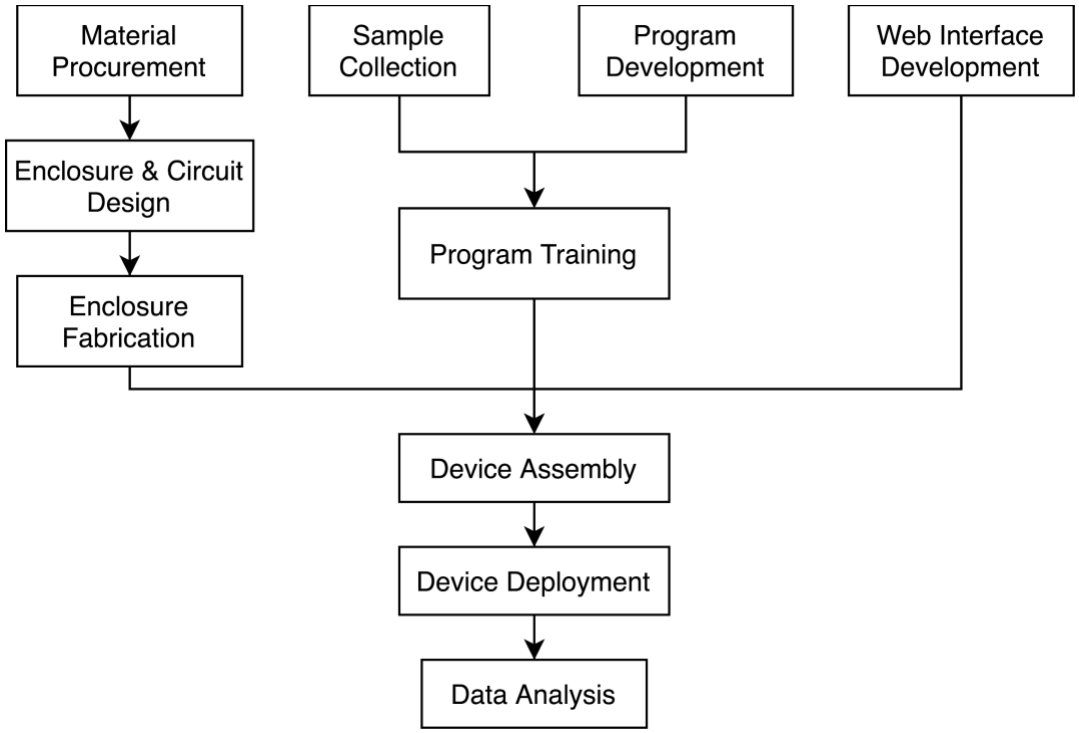

Figure 5. *Process flowchart for the study*

Hardware Design

Material Procurement. The following materials were procured for device construction: Raspberry Pi 3B+ (the single board computer which is the main component of the device), external microphone (for recording audio), 32 GB microSD card (for storing files), powerbank (for supplying the power of the device), DHT22 sensor (for sensing humidity and temperature), clock module (for keeping time), OL pellets (for attracting mosquitoes to the device), and ABS 3D-printer filament (for the enclosure of the device).

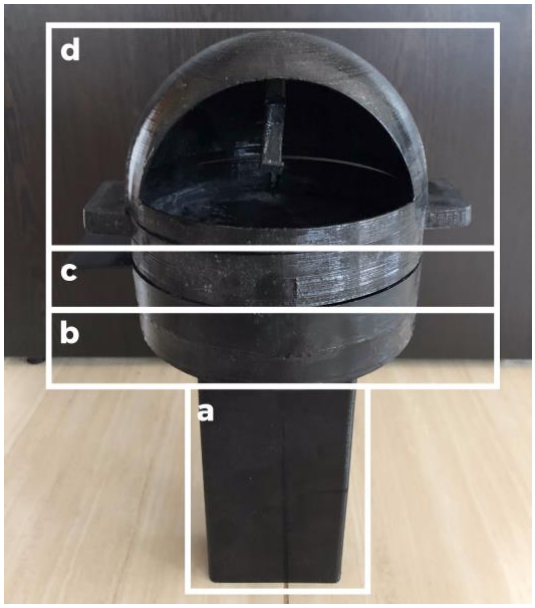

Figure 6. *Device enclosure of LAMOC: (a) power bank layer, (b) RPi layer, (c) OL trap layer, and (d) dome layer*

Enclosure design. Autodesk® Fusion 360™ was used in order to design the enclosure for 3D-printing. The enclosure is divided into four main parts: the powerbank layer, the Raspberry Pi layer, the OL trap layer, and the dome (see Figure 6). The measurements for each layer are described in Table 2.

The powerbank layer (see Figure 6a), with a height of 18 cm, length of 9cm, and width of 4 cm, serves as the stand of the device in order to place the other components on a higher level.

Table 2. *Measurements of the enclosure*

Layer	Height (cm)	Length (cm)	Width (cm)	Radius (cm)
Powerbank	18	9	4	-
Raspberry Pi	4.2	-	-	7.5
OL Trap	2.9	-	-	7.5
Dome	7.5	-	-	7.5

The second layer at the bottom is the RPi layer (see Figure 6b), which has a height of 4.2 cm and radius of 7.5 cm, holds the RPi fitted in the middle, the DHT-22 (temperature and humidity) sensor glued on the roof, and the USB port of the microphone that extends to the dome. A slot for an external fan for the Raspberry Pi is also present in the enclosure. Enough holes are implemented in order to improve ventilation which prevents the microprocessor from overheating.

The OL trap layer (see Figure 6c), with a height of 2.9 cm and radius of 7.5 cm, holds the OL pellets dissolved in water that serves as the bait for the mosquitoes. Sufficient space is also present in the case a *lawanit* paddle is to be placed.

The topmost layer (see Figure 6d) is designed for sound to be recorded; thus, the shape of a dome is used. It has a height and radius of 7.5 cm, Dome-shaped materials are often concave-shaped in order to allow sound to travel across the dome and accumulate at a single focal point, making it an ideal shape for the top layer of the enclosure (Ismail and Eldaly, 2018). An attachment that can be used as a holder for the clip of the microphone is also present.

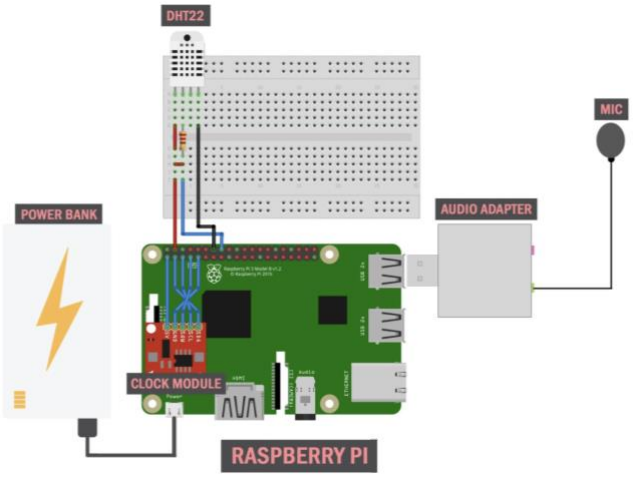

Figure 7. *Circuit and attachments of the device*

Circuit Design. Figure 7 shows the designed circuit for the device. Raspberry Pi 3B+ is the main processor of the whole device. Sensors, specifically an external clock module and DHT-22, are connected to the single-board computer so as to record the time and temperature in real time. A microphone, which is connected to the RPi via an audio adapter, serves as the recording unit of the sound of the mosquitoes. In order for the circuit to work, Xiaomi Mi Power Bank 2C serves as the power source which is connected as well to the RPi. This power bank has an electrical capacity of 20 000 mAh and a 3.0 quick charge.

Enclosure Fabrication. Both the Ultimaker Original+ and the XYZprinting da Vinci 1.0 Pro 3D printers, located in the Fabrication Laboratory in PSHS-CLC, were used in order to produce the enclosure. Two machines were utilized in order to lessen the printing time of the enclosure. The parts were printed using acrylonitrile butadiene styrene (ABS) filament, one of the most commonly used 3D-printer filaments today. As opposed to other commercial filaments, ABS is an extremely durable material that can withstand higher temperatures, flexibility, and strength which makes it suitable for a device that is generally placed outdoors (Kumar et al., 2018).

Software Design

Sample collection. In order to utilize machine learning for species classification, audio samples were acquired. The sources of these audio recordings are shown in Table 3. Recordings with length 51 seconds were retrieved for each of the four classes (*Culex sp., Anopheles sp., Aedes aegypti* male, *Aedes aegypti* female). In addition, 51 seconds of common background noise (e.g. rustling leaves, banging stones, walking on ground, walking on grass, running water) was included as another class to simulate environmental conditions.

Table 3. *Sources of audio recordings for machine learning*

Species	Title	Format
Culex sp.	Mosquito detection with low-cost smartphones: data acquisition for malaria research (Li et al., 2017)	.wav
Anopheles sp.	Using mobile phones as acoustic sensors for high-throughput mosquito surveillance (Mukundarajan et al., 2017)	.wav
Aedes aegypti (male & female)	Lone & Pair Mosquito Auditory Interaction (Homer et al., 2015)	.h5

Program development. Next, the audio recordings were converted to frequency signals by using Fast Fourier Transform (FFT) in a computer. This extraction method was implemented using the NumPy 1.18.1 library for Python. Wingbeat frequencies were determined by analyzing the frequency signals.

Using support vector machine (SVM), the training models for species identification are created. Since LAMOC uses binary SVM, a total of ten training models were created, namely

1. *Aedes aegypti* (male) - *Aedes aegypti* (female)
2. *Aedes aegypti* (male) - *Anopheles sp.*
3. *Aedes aegypti* (male) - *Culex sp.*
4. *Aedes aegypti* (male) - Background Noise
5. *Aedes aegypti* (female) - *Anopheles sp.*
6. *Aedes aegypti* (female) - *Culex sp.*
7. *Aedes aegypti* (female) - Background Noise
8. *Anopheles sp.* - *Culex sp.*
9. *Anopheles sp.* - Background Noise

10. Culex sp. - Background Noise

The 255 seconds recording was further divided to recordings with a length of 46 milliseconds each before going through Fast Fourier Transform (FFT) that was then converted to its frequency domain. The power in the frequency domain was normalized to avoid the problem of inconsistent scale of the power across different recordings. These training models are transferred to the Raspberry Pi for species identification.

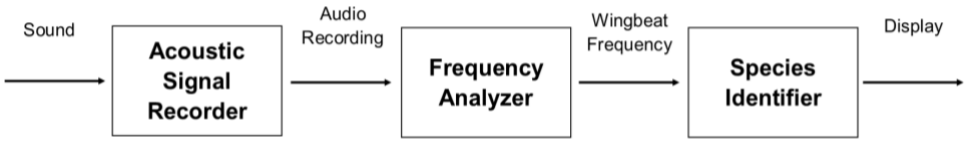

Figure 8. *Block diagram for the program running on the Raspberry Pi*

Acoustic Signal Recorder. Sound was first recorded from the mosquito using an external microphone connected to the Raspberry Pi. Ovicidal-Larvicidal (OL) trap was used to attract mosquitoes to the device. The Raspberry Pi was responsible for recording and storing the audio files for further processing using a developed Python program.

The process of recording audio, classifying the species, and sending data to the server is performed by the Raspberry Pi. Using the wide array of libraries available for development, the program was written entirely in the Python programming language. Table 3 lists the required libraries and each of their specific functions. After installation, the code can be executed using the Raspberry Pi's pre installed Python development software.

Upon running the program for the first time in a new location, the Raspberry Pi will first record an audio sample that will be used for calibration in the environment. This recording has a sampling rate of 44 100 Hz, 2 channels, 16 bits per channel and

length of 51 seconds. The program uses this in order to adjust the models according to the level and type of background noise present in the area.

Table 4. *Python libraries used for file processing and classification*

Library	Function	Description
os	Operating System	Performs operating system dependent functionality, such as renaming and removing files
sys	Runtime	Allows access to variables used by the interpreter
datetime	Data	Provides basic date and time information
pyaudio	Multimedia	Records and plays audio streams
wave	Multimedia	Reads and writes WAV audio files
Adafruit_ DHT	Sensor	Obtains readings from temperature and humidity sensors connected to a Raspberry Pi
mysql.con nector	Database	Inserts data from the Python source code into a mySQL database and table
scipy	Data	Gives access to modules for scientific and technical computing
numpy	Data	Contains fundamental packages for array computing
math	Numerical	Enables the use of mathematical functions
statistics	Numerical	Adds functions for statistical computing
libsvm	Classification	Provides functionality for support vector machine and supports multi-class classification

After calibration, the program proceeds to a continuous loop for processing audio and uploading data. Figure 9 shows a flowchart that illustrates the process. The Raspberry Pi then records audio from the external microphone for two (2) seconds. The audio is saved as a .wav file and is given a temporary file name for further processing.

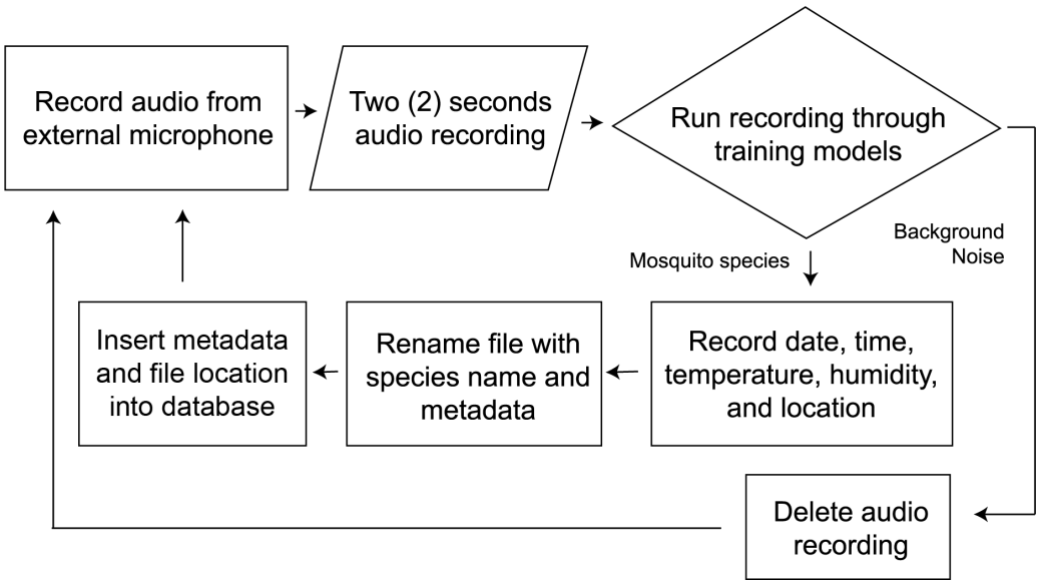

Figure 9. *Flowchart for the Python program*

Frequency Analyzer. Audio recordings were converted to frequency signals using Fast Fourier Transform (FFT). This extraction method was implemented using the NumPy library for Python. Wingbeat frequencies were determined by analyzing the frequency signals.

Species Identifier. The program then feeds the audio recording to the support vector machine training models for multi-class classification. Although SVM is traditionally used as a binary classifier, it can also be used to categorize samples from multiple classes by testing a sample against all classes and finding the class that produced a positive result in all tests, as used by Li et al. (2017) in their study. The program uses the preinstalled training models to predict which of the two species the recording belongs to, and increments the dedicated variable for that class. This is repeated until the recording has been fed into all ten training models.

The variable with the highest value will be deemed the species of the recording. If the variable for background noise is equal to four (4), which is the maximum score a class can have, the program will classify the audio recording as background noise and delete the temporary audio file. However, for accuracy testing purposes, the audio

recordings for background noise are saved in a different folder. Afterwhich, the program repeats its loop by returning to the first process indicated in the flowchart.

If the program classifies the audio file as a recording of a mosquito species, the Raspberry Pi proceeds to collect metadata for both renaming and uploading purposes. Date and time are obtained from the clock module, while temperature and humidity are logged by the DHT 22 sensor. The location is specified in the written code and may be edited if necessary. The temporary audio file is renamed for proper storage and easy access. The file name contains the date, time, species, location, and file format.

Web Interface Development

Lastly, the gathered data are uploaded into the mySQL database for viewing on the website. Apache web server, the most widely used open-source web platform, served as the host server. Meanwhile, PHP and MySQL were utilized to develop and manage the database. HTML and CSS were utilized to create the web interface. The home page of the website consists of the number of audio recorded, the amount of recordings for each species, date and time at access, and the location of the device (see Figure 10). On the “More” tab, all the recordings together with the metadata are displayed on a table. Each file contains a unique identification number, a button for audio playback, its identified species, specific location, date and time during recording, temperature, and humidity. There is also a “Print Page” function which allows the user to print the table or save it as a PDF file. Moreover, a survey was sent out to thirty-five respondents in order to test the usability and effectiveness of the constructed website.

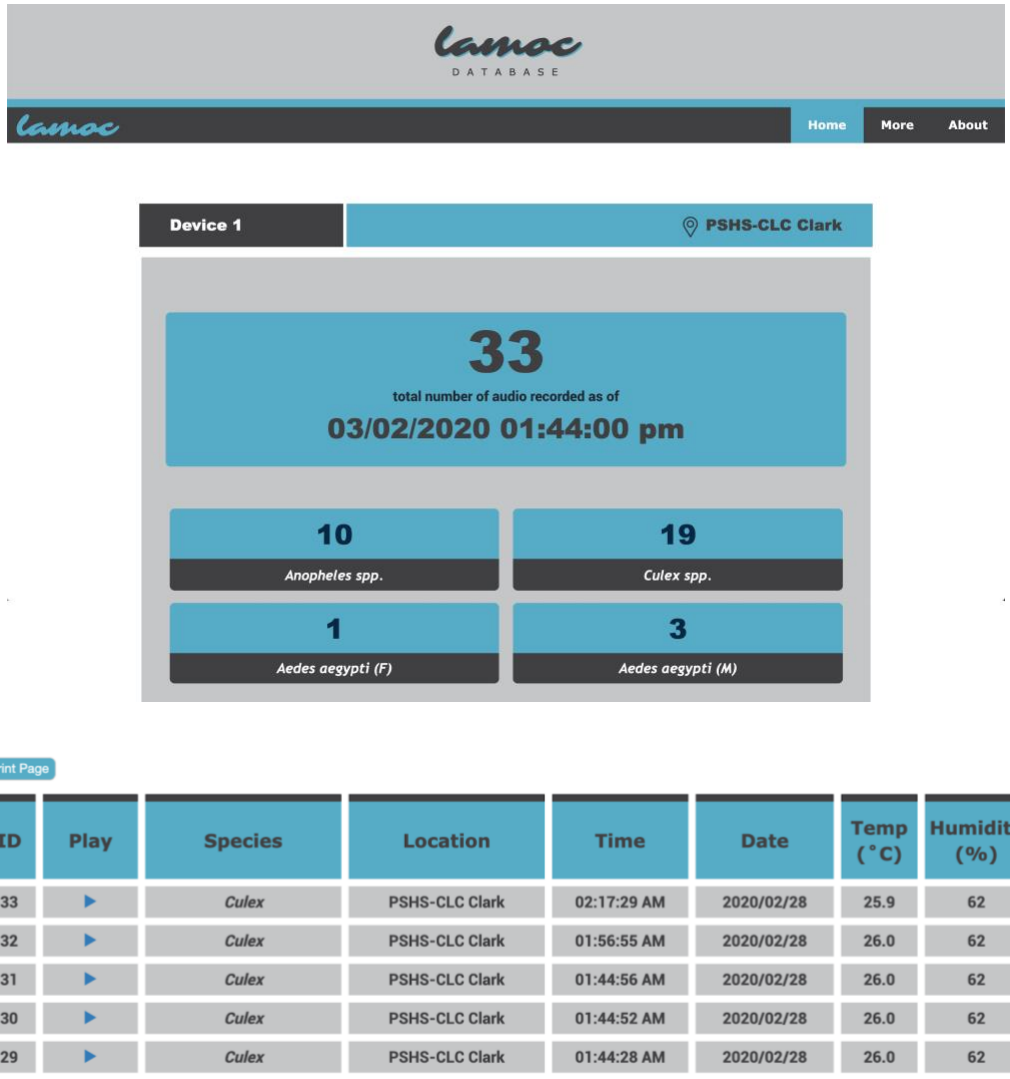

Figure 10. *Home page of the website that contains summary of the data and additional page that shows more information about each entry*

Device Assembly. In this part, the developed program was uploaded to the tested prototype in which the data collected can be accessed through the developed web interface. Since this study is a developmental research, only one device was assembled and tested.

Device Deployment. For field testing, the criteria of the location for the device deployment is its accessibility to the researcher and its projected mosquito population. The device was deployed in a residential area in Capas, Tarlac to collect data (audio recording, temperature, and humidity) for three nights. The selected area is well-

populated with greenery, making it a suitable testing area for recording mosquito sounds. The data recorded from the device was accessed via the server for LAMOC.

Data Analysis. The collected audio recordings were listened to one by one in order to ensure that the predicted categorization matches the real category. From this, confusion matrices were constructed and the accuracy rate of the device was derived. Three separate confusion matrices for each day and a specific confusion matrix for mosquito classification were constructed. Moreover, cross-validation was also used to identify the multi-class classification accuracy of the device.

Results and Discussion

LAMOC is a device that aims to automate the mosquito monitoring process using the Raspberry Pi, a 3D-printed enclosure, custom-built code, and classification training models. This section details the specifics behind the design of both the hardware and software of the device. Moreover, the process of improving the accuracy rate of the classification program is described.

Device Accuracy

Table 5. *Confusion matrix for 3 days of field deployment (February 26-28, 2020)*

n = 10 542			Predicted		Accuracy
			Mosquito	Background	
Day 1 (n=657)	Actual	Mosquito	0	0	86.91%
		Background	86	571	
Day 2 (n=4989)		Mosquito	0	0	99.34%
		Background	33	4956	
Day 3 (n=4896)		Mosquito	13	3	98.04%
		Background	93	4787	

On the first day of field testing of the device, 22 minutes worth of sound was recorded. It was then manually listened to and it was determined that no mosquito sound was present. In this case, the device should classify all the recorded sounds as background noise, however, only 86.91% was classified as background noise as shown in Table 5. An 86.91% accuracy in classifying a background noise as background noise is worrying because it means that eight minutes worth of audio is classified wrongly every hour.

On the second day of field testing, the location where the device was tested was in a residential area in Capas, Tarlac. In order to solve the problem of low accuracy

when it comes to classifying background noise, the device was designed to first record the background noise in the location it will be deployed before starting the loop of audio recording and species classification. After implementing the adjustment, a clear improvement in accuracy (99.34%) was observed compared to the first day (86.91%) of testing as shown in Table 5. A total of 166 minutes of recording was listened to, and it was determined that no mosquito sound was present.

On the third day of testing, a total of 170 minutes was listened to by the researchers, and it was determined that 32 seconds (0.33%) of it belonged to a mosquito. The device, however, managed to correctly classify only 26 seconds (0.27%) of the entire recording as mosquito sound, and the remaining 6 seconds (0.06%) was wrongly classified as background noise. On other hand, 160 minutes (97.77%) of the whole recording was correctly classified as background noise, but 3 minutes and 6 seconds (1.90%) of the entire recording was wrongly classified as mosquito sound as shown in Table 5. Overall, the accuracy of the device in classifying between a mosquito sound and background noise was 98.04%.

For further improvement of the accuracy, it was decided that the mosquito sound recorded by the device on the third day of field testing would be integrated to the training pool of data. To avoid bias, the number of samples (n) was limited up to 1100 only so that the training data set of each class would be balanced. In addition, instead of classifying the class of a 2-second audio recording, the program of the device was adjusted to check the class of the 50-millisecond sound clip.

Table 6. *Summary of confusion matrix for mosquito detection after integrating the mosquito sound procured on Day 3 of field testing.*

n = 5500		Predicted	
		Positive	**Negative**
Actual	**Positive**	TP: 4200	FN: 200
	Negative	FP: 105	TN: 995

To test the accuracy of the device with the addition of the 32-second *Aedes aegypti* (male) recording, a partially nested cross-validation (in which the parameter tuning was applied in a non-nested fashion) was used. A partially nested cross-validation is suitable to use when the data procured is limited and when all the data available will be used in training the machine learning algorithm (Vabalas et al., 2019).

The overall accuracy of the device was calculated using the confusion matrix given in Table 6. In terms of the device's ability to differentiate a mosquito sound to a background noise, an accuracy of 90.47% was determined. Comparing this to a study's (Li et al., 2017) overall accuracy of 80.25% that dealt with the identification of mosquito species using smartphones, LAMOC got a higher accuracy with 94.74%.

Table 7. *Accuracy rates for the five classes using cross-validation*

Class	*Aedes aegypti* (male)	*Aedes aegypti* (female)	*Anopheles sp.*	*Culex sp.*	Background Noise	**Overall Accuracy**
Mean	99.30%	96.74%	92.09%	93.72%	90.47%	**94.74%**
n	1100	1100	1100	1100	1100	**5500**

For multiclass classification of the device, several adjustments were implemented. As shown in Table 7, *Aedes aegypti* (male) obtained the highest accuracy of 99.30% with a standard deviation of 1.12%, while the background noise got the lowest accuracy rate of 90.47% with a relatively high standard deviation of 9.83%. The

high accuracy of the program in correctly classifying *Aedes aegypti* (male) may be attributed to the fact that new sound files were integrated into the training pool of data, while the relatively low accuracy of the program in correctly classifying the background noise may be attributed to the fact that the background noise in the training pool data did not have a more distinctive features compared to the other classes.

The program of this study obtained a higher accuracy compared to the multi-class classification accuracy of the study of Li et al. (2017), wherein they also used support vector machine (SVM). Both studies had a class of *Aedes aegypti*, but their study only managed to acquire an 82% accuracy while this study managed to have an accuracy of 99.30% and 96.74% in *Aedes aegypti* (male) and *Aedes aegypti* (female), respectively. The difference in result may be attributed to the fact that the classes in their study may not have significant differences compared to this study. Another possible reason is their usage of mel frequency cepstral coefficient (MFCC) as the feature extraction method while this study used the frequency domain of the sound clip.

Web Design

Results of the survey show that the developed web site is user-friendly with the highest average rating of 4.83 (Figure 11). It is followed by satisfaction rating (4.77), efficiency and overall appearance (4.74), color scheme (4.71), and layout and usability (4.69). The rating of respondents on the layout and usability of the website has a higher variation compared to the other criteria.

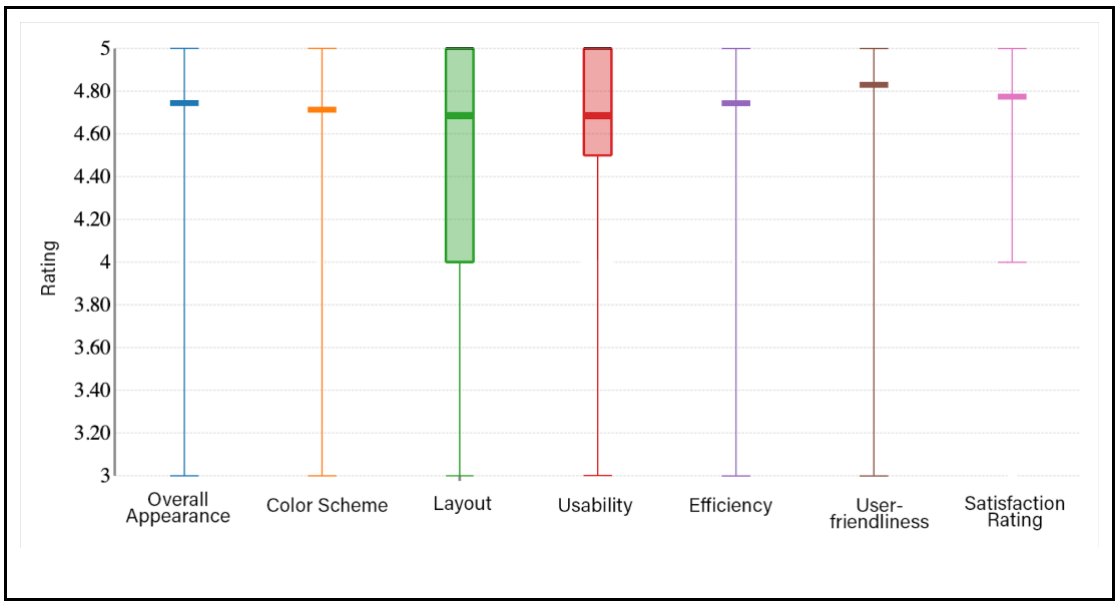

Figure 11. *Ratings of 35 respondents in a survey on the website*

Based on the rating scale used, the website is in between “excellent” and “very good”. With this, the overall average rating of 4.74 of the website indicates that it is a good platform for viewing the device’s collected data. Further editing of the parameters with the lowest scores, namely Layout and Usability, can be performed in order to improve the website.

Conclusions & Recommendations

LAMOC was developed as an alternative to the expensive bioacoustic devices in the market and a solution to the unsustainable method of mosquito monitoring in the Philippines, where multiple species of mosquitoes are especially prevalent. Using existing methodologies for machine learning and acoustic signal processing, the study succeeded in developing an acoustic signal monitoring device that detects the presence of mosquitoes. With an overall classification accuracy of 94.74%, LAMOC has shown to be a promising alternative to existing mosquito monitoring methods. Moreover, the device enclosure has been proven sturdy enough for deployment in a real-life setting. There were also no issues in accessing the collected data using the developed website. Thus, the development of LAMOC using the Raspberry Pi can be considered successful.

As the device is still in its early development stage, there are many aspects that can be improved before mass deployment. Other machine learning algorithms that maximize both power and accuracy can be tested. On the physical aspect, different enclosure designs and high quality microphones that can further enhance sound quality can be used. Finally, further testing and debugging in large-scale platforms is highly recommended in order to refine and perfect the system of the device.

References

Aide, T. M., Corrada-Bravo, C., Campos-Cerqueira, M., Milan, C., Vega, G., & Alvarez, R. (2013). Real-time bioacoustics monitoring and automated species identification. *PeerJ*, *1*, e103. https://doi.org/10.7717/peerj.103

Arthur, B. J., Emr, K. S., Wyttenbach, R. A., & Hoy, R. R. (2014). Mosquito (*Aedes aegypti*) flight tones: Frequency, harmonicity, spherical spreading, and phase relationships. *The Journal of the Acoustical Society of America*, *135*(2), 933–941. https://doi.org/10.1121/1.4861233

Atabekov, A., He, J., & Bobbie, P. O. (2016). Internet of Things-Based Framework to Facilitate Indoor Localization Education. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 269–274. https://doi.org/10.1109/COMPSAC.2016.143

Avisoft. (2020). *UltraSoundGate - Avisoft Bioacoustics*. http://www.avisoft.com/ultrasoundgate/

Cao, J., Wang, M., Li, Y., & Zhang, Q. (2019). Improved support vector machine classification algorithm based on adaptive feature weight updating in the Hadoop cluster environment. *PLOS ONE*, *14*(4), e0215136. https://doi.org/10.1371/journal.pone.0215136

Caprio MA, Huang JX, Faver MK, & Moore A. (2001). Characterization of male and female wingbeat frequencies in the *Anopheles* quadrimaculatus complex in Mississippi. *Journal of the American Mosquito Control Association*, *17*(3), 186–189.

Caruana, R., & Niculescu-Mizil, A. (2004). Data mining in metric space. *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '04*, 69. https://doi.org/10.1145/1014052.1014063

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Dia, I., Diallo, D., Duchemin, J., Ba, Y., Konate, L., Costantini, C., & Diallo, M. (2005). Comparisons of Human-Landing Catches and Odor-Baited Entry Traps for Sampling Malaria Vectors in Senegal. *Journal of Medical Entomology*, *42*(2), 104–109. https://doi.org/10.1093/jmedent/42.2.104

European Centre for Disease Prevention and Control. (2016, December 20). *Aedes aegypti - Factsheet for experts*. https://www.ecdc.europa.eu/en/disease-vectors/facts/mosquito-factsheets/aedes-aegypti

Fielding, R. T., & Kaiser, G. (1997). The Apache HTTP Server Project. *IEEE Internet Computing*, *1*(4), 88–90. https://doi.org/10.1109/4236.612229

Flores, K. O., Butaslac, I. M., Gonzales, J. E. M., Dumlao, S. M. G., & Reyes, R. S. J. (2016). Precision agriculture monitoring system using wireless sensor network and Raspberry Pi local server. *2016 IEEE Region 10 Conference (TENCON)*, 3018–3021. https://doi.org/10.1109/TENCON.2016.7848600

Gómez, A., Cuiñas, D., Catalá, P., Xin, L., Li, W., Conway, S., & Lack, D. (2015). Use of Single Board Computers as Smart Sensors in the Manufacturing Industry. *Procedia Engineering*, *132*, 153–159. https://doi.org/10.1016/j.proeng.2015.12.461

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

Harbach, R., & Besansky, N. (2014). Mosquitoes. *Current Biology*, *24*(1), R14–R15.

Hiscox, A., Otieno, B., Kibet, A., Mweresa, C. K., Omusula, P., Geier, M., … Takken, W. (2014). Development and optimization of the Suna trap as a tool for mosquito monitoring and control. *Malaria Journal*, *13*(1), 257. https://doi.org/10.1186/1475-2875-13-257

Homer, M., Champneys, A., Aldersley, A., & Robert, D. (2015). Lone & Pair Mosquito Auditory Interaction. *Engineering and Physical Sciences Research Council*.

Huber, J. H., Childs, M. L., Caldwell, J. M., & Mordecai, E. A. (2018). Seasonal temperature variation influences climate suitability for dengue, chikungunya, and Zika transmission. *PLOS Neglected Tropical Diseases*, *12*(5), e0006451. https://doi.org/10.1371/journal.pntd.0006451

Ismail, M. R., & Eldaly, H. (2018). Acoustic of monolithic dome structures. *Frontiers of Architectural Research, 7*(1), 56–66. https://doi.org/10.1016/j.foar.2017.11.002

Johnston, S. J., Basford, P. J., Perkins, C. S., Herry, H., Tso, F. P., Pezaros, D., … Singer, J. (2018). Commodity single board computer clusters and their applications. *Future Generation Computer Systems*, *89*, 201–212. https://doi.org/10.1016/j.future.2018.06.048

Koshti, M., & Ganorkar, S. (2016). IoT Based Health Monitoring System by Using Raspberry Pi and ECG Signal. *International Journal of Innovative Research in Science, Engineering and Technology*, *5*(5).

Krockel, U., Rose, A., Eiras, A., & Geier, M. (2006). New tools for surveillance of adult yellow fever mosquitoes: Comparison of trap catches with human landing rates in an urban environment. *Journal of the American Mosquito Control Association*, *22*(2), 229–238.

Kumar, R., Singh, K., & Farina, I. (2018). On the 3D printing of recycled ABS, PLA and HIPS thermoplastics for structural applications. *PSU Research Review, 2*(2), 115-137. https://doi.org/10.1108/PRR-07-2018-0018

Li, Y., Zilli, D., Chan, H., Kiskin, I., Sinka, M., Roberts, S., & Willis, K. (2017). Mosquito detection with low-cost smartphones: data acquisition for malaria research. *NIPS Workshop on Machine Learning for the Developing World (ML4D)*, 1–5.

Liu, C., & Cheng, Y. (2018). An Application of the Support Vector Machine for Attribute-By-Attribute Classification in Cognitive Diagnosis. *Applied Psychological Measurement*, *42*(1), 58–72. https://doi.org/10.1177/0146621617712246

McGregor, P. (2012). Applying bioacoustics: What can be learned from pure and applied aspects? *Bioacoustics*, *21*(1), 13–15. https://doi.org/doi:10.1080/09524622.2011.647526

Mukundarajan, H., Hol, F. J. H., Castillo, E. A., Newby, C., & Prakash, M. (2017). Using mobile phones as acoustic sensors for high-throughput mosquito surveillance. *ELife*, *6*. https://doi.org/10.7554/eLife.27854

Proakis, J. G., & Manolakis, D. G. (1996). *Digital Signal Processing. Principles, Algorithms, and Applications*. Prentice Hall.

Rajalakshmi, P., & Devi Mahalakshmi, S. (2016). IOT based crop-field monitoring and irrigation automation. *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 1–6. https://doi.org/10.1109/ISCO.2016.7726900

Robert, D. (2009). Insect Bioacoustics: Mosquitoes Make an Effort to Listen to Each Other. *Current Biology*, *19*(11), R446–R449. https://doi.org/10.1016/j.cub.2009.04.021

Rushikesh, R., & Sivappagari, C. M. R. (2015). Development of IoT based vehicular pollution monitoring system. *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 779–783. https://doi.org/10.1109/ICGCIoT.2015.7380568

Sachdeva, P., & Katchii, S. (2014). A Review Paper on Raspberry Pi. *International Journal of Current Engineering and Technology*, *4*(6), 3818–3819.

Sanner, M. F. (1999). Python: a programming language for software integration and development. *Journal of Molecular Graphics & Modelling*, *17*(1), 57–61.

Sia Su, G. (2014). Larval Mosquito Diversity and Distribution in Rice Field Agro-Ecosystems in Sariaya, Quezon Province, Philippines. *Annual Research &*

Review in Biology, *4*(18), 2884–2891. https://doi.org/10.9734/ARRB/2014/10128

Tolle, M. A. (2009). Mosquito-borne Diseases. *Current Problems in Pediatric and Adolescent Health Care*, *39*(4), 97–140. https://doi.org/10.1016/j.cppeds.2009.01.001

Vabalas, A., Gowen, E., Poliakoff, E., & Casson, A. J. (2019). Machine learning algorithm validation with a limited sample size. *PLOS ONE, 14*(11). https://doi.org/10.1371/journal.pone.0224365

Vaidya, B., Patel, A., Panchal, A., Mehta, R., Mehta, K., & Vaghasiya, P. (2017). Smart home automation with a unique door monitoring system for old age people using Python, OpenCV, Android and Raspberry Pi. *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*. doi:10.1109/iccons.2017.8250582

Vasconcelos, D., Nunes, N., Ribeiro, M., Prandi, C., & Rogers, A. (2019). LOCOMOBIS: a low-cost acoustic-based sensing system to monitor and classify mosquitoes. *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 1–6. https://doi.org/10.1109/CCNC.2019.8651767

Vujovic, V., & Maksimovic, M. (2014). Raspberry Pi as a Wireless Sensor node: Performances and constraints. *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1013–1018. https://doi.org/10.1109/MIPRO.2014.6859717

World Health Organization. (2016, October 17). *Mosquito-borne diseases*. https://www.who.int/neglected_diseases/vector_ecology/mosquito-borne-diseases/en/

Appendix A

Raw Data

Audio Files

Access raw audio files with this link https://tinyurl.com/lamocrawdata or https://drive.google.com/drive/folders/1LMuqLcniLh3Rqr3adVt1PSaMgXJpPEaw?usp=sharing

List of Folders and Files

Aedes aegypti female (Homer et al., 2015)

AedF1.h5
AedF2.h5
AedF3.h5
AedF4.h5
AedF5.h5
AedF6.h5
AedF7.h5
AedF8.h5
AedF9.h5
AedF10.h5
AedF11.h5
AedF12.h5
AedF13.h5
AedF14.h5
AedF15.h5
AedF16.h5
AedF17.h5
AedF18.h5
AedF19.h5

Aedes aegypti male (Homer et al., 2015)

AedM1.h5
AedM2.h5
AedM3.h5
AedM4.h5
AedM5.h5
AedM6.h5

AedM7.h5

AedM8.h5

AedM9.h5

AedM10.h5

AedM11.h5

AedM12.h5

AedM13.h5

AedM14.h5

AedM15.h5

AedM16.h5

AedM17.h5

AedM18.h5

AedM19.h5

AedM20.h5

AedM21.h5

AedM22.h5

AedM23.h5

AedM24.h5

AedM25.h5

AedM26.h5

AedM27.h5

Anopheles sp. (Mukundarajan et al., 2017)

Anopheles albimanus

Anopheles quadriannulatus

Anopheles stephensi

Background Noise

Bamboo Rustling.wav

Basketball.wav

Blender.wav

Crushing Leaves.wav

Garden Ambience.wav

Garden Entrance.wav

Stone Banging 1.wav

Stone Banging 2.wav

Walking Garden.wav

Walking Grass.wav

Walking Ground.wav

Walking Stones.wav

Water Running.wav

Culex sp. (Li et al., 2017)

Culex.wav

Appendix B

Hardware Design

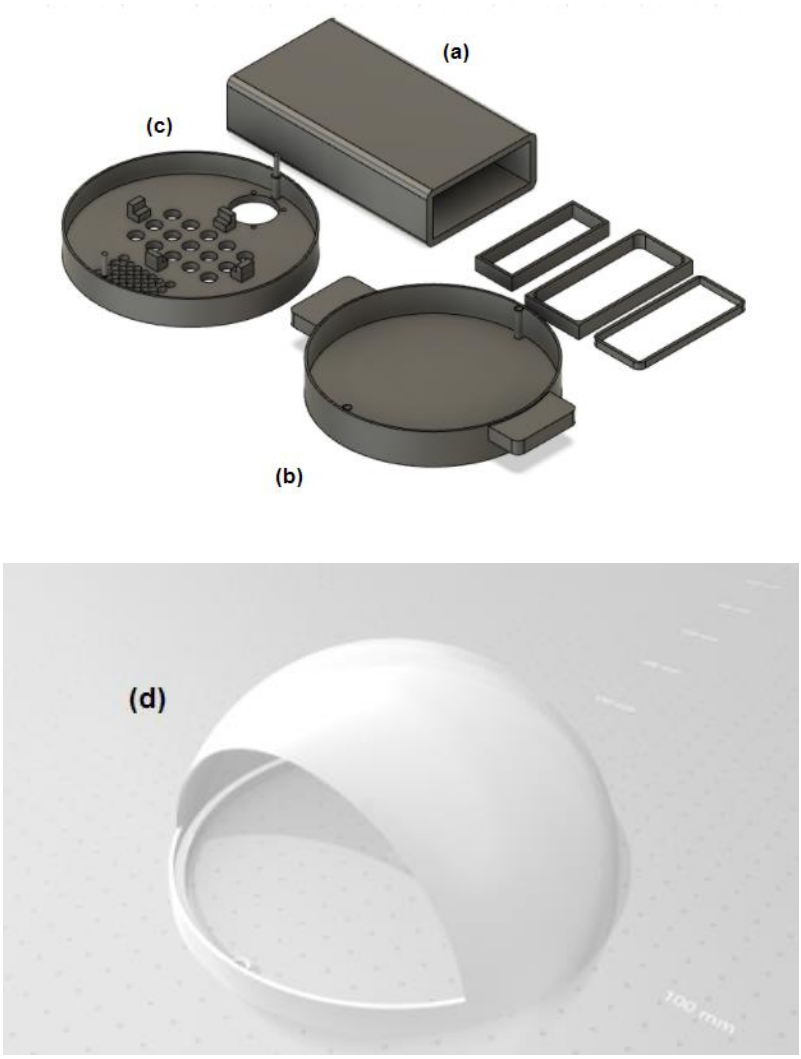

Figure 12. *3D model of the hardware design via Autodesk Fusion 360.*

Appendix C

Code

Cross Validation Code

```
import math
from scipy.fftpack import fft
from scipy.io import wavfile
import numpy as np
from scipy import stats
from svmutil import *
import scipy.io
import statistics


def main():
   y = myCalibration("D:/ResearchData/Model/October/jarod.wav")
   sample_AeM = y[:430, :]
   #converting matlab matrix to numpy array of python
   sample_AeF, sample_BG =
matToNumpyArray("D:/ResearchData/Model/Matlab/model_AeF_BG.mat")
   sample_C, sample_An =
matToNumpyArray("D:/ResearchData/Model/Matlab/model_C_An.mat")

   i = 1
   final = list()
   while i <= 10:
       model_AeM_BG, test_AeM, test_BG, featurePos_AeM_BG =
featureSelection(sample_AeM, sample_BG, i,toggle1=False)
       model_AeM_AeF, featurePos_AeM_AeF =
featureSelection(sample_AeM, sample_AeF, i)
       model_AeM_C, featurePos_AeM_C = featureSelection(sample_AeM,
sample_C, i)
       model_AeM_An, featurePos_AeM_An = featureSelection(sample_AeM,
sample_An, i)
       model_AeF_BG, featurePos_AeF_BG = featureSelection(sample_AeF,
sample_BG, i)
       model_AeF_C, test_AeF, test_C, featurePos_AeF_C =
featureSelection(sample_AeF, sample_C, i, toggle1=False)
       model_AeF_An, featurePos_AeF_An = featureSelection(sample_AeF,
sample_An, i)
       model_C_An, featurePos_C_An = featureSelection(sample_C,
sample_An, i)
       model_C_BG, featurePos_C_BG = featureSelection(sample_C,
sample_BG, i)
       model_An_BG, test_An, featurePos_An_BG =
featureSelection(sample_An, sample_BG, i,toggle=True)

       key =
{"AeM_BG":featurePos_AeM_BG,"AeM_AeF":featurePos_AeM_AeF,"AeM_C":feat
urePos_AeM_C,
                "AeM_An":featurePos_AeM_An,"AeF_BG":featurePos_AeF_BG,"
AeF_C":featurePos_AeF_C,
                "AeF_An":featurePos_AeF_An,"C_An":featurePos_C_An,"C_BG
":featurePos_C_BG,"An_BG":featurePos_An_BG}
```

```
        results = multiClassClassification(test_AeM, test_AeF, test_C,
test_An, test_BG, model_AeM_BG, model_AeM_AeF,
                                           model_AeM_C, model_AeM_An,
model_AeF_BG,
                                           model_AeF_C, model_AeF_An,
model_C_An, model_C_BG,model_An_BG,key)

        final.append(results)
        i += 1
    print(final)
    aem = 0
    aef = 0
    c = 0
    an = 0
    bg = 0
    for i in range(10):
        aem += final[i][0]/43
        aef += final[i][1]/43
        c += final[i][2]/43
        an += final[i][3]/43
        bg += final[i][4]/43
    print("AeM",aem*10,"%")
    print("AeF",aef * 10, "%")
    print("C",c * 10, "%")
    print("An",an * 10, "%")
    print("BG",bg * 10, "%")
    print("done")

def myCalibration(path):
    """
    name: myCalibration. this function will convert wav audio file to
its frequency domain
    :param path: the directory of the wav file in which the fft will
be applied
    :return: cnt: a matrix where the row is the sample and the column
is the feature of the sample
    """
    frame_length = 2048
    rate, data = scipy.io.wavfile.read(path)
    cell_no = math.floor(len(data) / frame_length)
    data = np.array(data[:cell_no * frame_length])  ##cutting the data
so that it will be divisible to 2048

    b = 0
    while (b + 1) * frame_length < len(data):
        x_n = data[(b * frame_length):(b + 1) * frame_length]
        fft_out = fft(x_n)
        y = np.fft.fftshift(np.abs(fft_out))

        y_new = y[round(len(y) / 2):]

        max_amplitude = np.max(y_new)
        y_new = y_new / max_amplitude
        y_new = y_new[:362]

        if b == 0:
            cnt = y_new
        else:
            cnt = np.vstack((cnt, y_new))
        b += 1
```

```
    ##the result variable is cnt
    ##cnt[x,y], x: different samples with a size of 362 ; y: features
    # return cnt[39,:] --> this is a 1D array horizontal
    return cnt


def featureSelection(mat1, mat2, i,toggle=False,
toggle1=True):      ##if toggle is True, it will return
    """
    name: featureSelection. this function will create a binary
classification svm model that can be used to predict a sample's
class.
    :param mat1: numpy.array (should have the same size with mat2)
that will be used to create a model
    :param mat2: numpy.array (should have the same size with mat1)
that will be used to create a model
    :param i: the iteration of the cross-validation currently being
performed
    :param toggle: (default: False). a param to indicate what to
return
    :param toggle1: (default: True). a param to indicate what to
return
    :return: depends on boolean value of the toggle and toggle1.
    model (always): svm model created that will be used to predict
another sample
    mat1_test (when toggle=True or toggle=toggle1=False): test sample
that will be used for validation
    mat2_test (when toggle=False and toggle1=True or
toggle=toggle1=False): test sample that will be used for validation
    featureRankPosition (always): numpy.array that contains the index
of the features that are important
    """
    delList = list()
    for j in range(43):
        delList.append(j + ((i - 1) * 43))  ##delList that will delete
the test data in train data

    mat1_train = np.delete(mat1, delList, axis=0)
    mat1_test = mat1[(i - 1) * 43:i * 43, :]

    mat2_train = np.delete(mat2, delList, axis=0)
    mat2_test = mat2[(i - 1) * 43:i * 43, :]

    featureRank, p_value = np.abs(stats.ttest_ind(mat1_train,
mat2_train, axis=0, equal_var=False))
    featureRankPosition = np.zeros(NUM_FEATURES)
    resultingPos = np.zeros(2)
    for k in range(NUM_FEATURES):
        resultingPos = np.where(featureRank ==
np.amax(featureRank))  # finds position of max value
        if np.shape(resultingPos[0]) != 1:
            featureRankPosition[k] = resultingPos[0][0]
            featureRank[resultingPos[0][0]] = 0
        else:
            featureRankPosition[k] = resultingPos[0]
            featureRank[resultingPos[0]] = 0


    mat1_train_featured = np.zeros((430 - 43, NUM_FEATURES))
    mat2_train_featured = np.zeros((430 - 43, NUM_FEATURES))
```

```
    for l in range(NUM_FEATURES):
        mat1_train_featured[:, l] = mat1_train[:,
int(featureRankPosition[l])]
        mat2_train_featured[:, l] = mat2_train[:,
int(featureRankPosition[l])]

    SIZE = 430 - 43
    Y = np.zeros(SIZE * 2)
    Y[:SIZE] = -1
    Y[SIZE:] = 1

    finalMat = np.vstack((mat1_train_featured, mat2_train_featured))
    prob = svm_problem(Y, finalMat)
    param = svm_parameter('-s 0 -t 0 -c 30')

    model = svm_train(prob, param)
    if toggle:
        return model, mat1_test, featureRankPosition
    elif toggle1:
        return model, featureRankPosition
    else:
        return model, mat1_test, mat2_test, featureRankPosition


def multiClassClassification(test_AeM, test_AeF, test_C, test_An,
test_BG, model_AeM_BG, model_AeM_AeF,
                             model_AeM_C, model_AeM_An, model_AeF_BG,
                             model_AeF_C, model_AeF_An, model_C_An,
model_C_BG,
                             model_An_BG,key):
    """
    name: multiClassClassification. this is function will guess what
class does a particular sample belongs to. For now, it was modified
to return the accuracy of the multi-classification.
    :param test_AeM: numpy.array of the AeM to be tested to determine
the accuracy of the model (for validation)
    :param test_AeF: numpy.array of the AeF to be tested to determine
the accuracy of the model (for validation)
    :param test_C: numpy.array of the C to be tested to determine the
accuracy of the model (for validation)
    :param test_An: numpy.array of the An to be tested to determine
the accuracy of the model (for validation)
    :param test_BG: numpy.array of the BG to be tested to determine
the accuracy of the model (for validation)
    :param model_AeM_BG: model to be used when comparing
    :param model_AeM_AeF: model to be used when comparing
    :param model_AeM_C: model to be used when comparing
    :param model_AeM_An: model to be used when comparing
    :param model_AeF_BG: model to be used when comparing
    :param model_AeF_C: model to be used when comparing
    :param model_AeF_An: model to be used when comparing
    :param model_C_An: model to be used when comparing
    :param model_C_BG: model to be used when comparing
    :param model_An_BG: model to be used when comparing
    :param key: dictionary that contains the index of the features to
be retained on a specific binary classification
    :return: an array that contains the accuracy of the multi-
classification in guessing the correct class of the test
```

```
    sample. the order is acc_aem,acc_aef,acc_c,acc_an,acc_bg
respectively.
    """
    test = np.vstack((test_AeM, test_AeF, test_C, test_An,test_BG))
    a = np.shape(test)
    results = list()
    acc_aef = 0
    acc_aem = 0
    acc_an = 0
    acc_c = 0
    acc_bg = 0
    j = 0
    x = np.zeros((1,NUM_FEATURES))
    for i in range(a[0]):

        y = np.array([test[i, :]])

        AeF = 0
        AeM = 0
        An = 0
        C = 0
        BG = 0

        try:
            x = keyDecoder(y,key["AeF_BG"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_AeF_BG)
            if statistics.mode(p_labels) == -1:
                AeF += 1
            else:
                BG += 1
            x = keyDecoder(y, key["AeM_BG"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_AeM_BG)
            if statistics.mode(p_labels) == -1:
                AeM += 1
            else:
                BG += 1
            x = keyDecoder(y, key["An_BG"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_An_BG)
            if statistics.mode(p_labels) == -1:
                An += 1
            else:
                BG += 1
            x = keyDecoder(y, key["C_BG"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_C_BG)
            if statistics.mode(p_labels) == -1:
                C += 1
            else:
                BG += 1
            x = keyDecoder(y, key["AeM_AeF"])
            p_labels, p_acc, p_vals = svm_predict([], x,
model_AeM_AeF)
            if statistics.mode(p_labels) == -1:
                AeM += 1
            else:
                AeF += 1
            x = keyDecoder(y, key["AeF_An"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_AeF_An)
            if statistics.mode(p_labels) == -1:
                AeF += 1
            else:
```

```
                An += 1
            x = keyDecoder(y, key["AeM_An"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_AeM_An)
            if statistics.mode(p_labels) == -1:
                AeM += 1
            else:
                An += 1
            x = keyDecoder(y, key["AeF_C"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_AeF_C)
            if statistics.mode(p_labels) == -1:
                AeF += 1
            else:
                C += 1
            x = keyDecoder(y, key["AeM_C"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_AeM_C)
            if statistics.mode(p_labels) == -1:
                AeM += 1
            else:
                C += 1
            x = keyDecoder(y, key["C_An"])
            p_labels, p_acc, p_vals = svm_predict([], x, model_C_An)
            if statistics.mode(p_labels) == -1:
                C += 1
            else:
                An += 1

            var = {'Aedes Female': AeF, 'Aedes Male': AeM,
'Anopheles': An, 'Culex': C, 'Background': BG}
            results.append(max(var, key=var.get))

        except:
            results.append('Undetermined')

        j = math.floor(i/43)

        if results[i] == "Aedes Male" and j == 0:
            acc_aem += 1
        if results[i] == 'Aedes Female' and j == 1:
            acc_aef += 1
        if results[i] == "Culex" and j == 2:
            acc_c += 1
        if results[i] == 'Anopheles' and j == 3:
            acc_an += 1
        if results[i] == 'Background' and j == 4:
            acc_bg += 1

    acc = [acc_aem,acc_aef,acc_c,acc_an,acc_bg]
    return acc

def matToNumpyArray(loadMatY):
    """
    name: matToNumpyArray. This function will seperate the two sample
inside the .mat file and it will also convert it to a numpy array
    :param loadMatY: directory of the .mat file
    :return: 2 numpy.array.
    resultMatA: a matrix where the row is the sample and the column is
the feature of the sample.
    resultMatB: a matrix where the row is the sample and the column is
the feature of the sample.
    """
```

```
    matY = scipy.io.loadmat(loadMatY)
    finalMatY = np.array(matY['train'])

    delList = list()
    for i in range(1100):
        delList.append(i+1100)
    MatA = np.delete(finalMatY,delList,axis=0)

    delList = list()
    for i in range(1100):
        delList.append(i)
    MatB = np.delete(finalMatY,delList,axis=0)

    resultMatA = np.zeros((430,362))
    resultMatB = np.zeros((430, 362))
    for i in range(430):
        resultMatA[i,:] = MatA[math.floor(1100/430*i), :]
        resultMatB[i, :] = MatB[math.floor(1100 / 430 * i), :]

    return resultMatA, resultMatB

def keyDecoder(mat1_test,key):
    """
    name: keyDecoder. this function will remove the features of the
sample that are not important
    :param mat1_test: numpy.array that contains the sample to be
classified by the model
    :param key: numpy.array that contains the index of the features to
be retained
    :return: numpy.array that only contains the important features
    """
    mat1_test_featured = np.zeros((1,NUM_FEATURES))
    for l in range(NUM_FEATURES):
        mat1_test_featured[0,l] = mat1_test[0,int(key[l])]

    return mat1_test_featured

NUM_FEATURES = 100
if __name__ == "__main__":
    main()
```

Initialization Code

```
import mysql.connector
import Adafruit_DHT
import datetime
import pyaudio
import wave
import os
import sys

while True:
    now = datetime.datetime.now() # Records date and time from clock
module

    # Parameters for audio recording
```

```
    CHUNK = 1024
    FORMAT = pyaudio.paInt16
    CHANNELS = 2
    RATE = 44100
    RECORD_SECONDS = 2
    WAVE_OUTPUT_FILENAME = "/var/www/html/aud/recordTemp.wav"

    p = pyaudio.PyAudio()

    # Start recording

    stream = p.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

    print("* recording")

    frames = []

    for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
        data = stream.read(CHUNK)
        frames.append(data)

    print("* done recording")

    stream.stop_stream() # End recording
    stream.close()
    p.terminate()

    # Save recording as .wav file
    wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
    wf.setnchannels(CHANNELS)
    wf.setsampwidth(p.get_sample_size(FORMAT))
    wf.setframerate(RATE)
    wf.writeframes(b''.join(frames))
    wf.close()

    # Run audio recording to training model
    from trainingModel import trainingModel
    specI = trainingModel('/var/www/html/aud/recordTemp.wav')
    print(specI) # Print identified species

    if (specI == 'Background'):
        os.remove("/var/www/html/aud/recordTemp.wav") # Delete
recording if it is background noise
    else: # Set identified species to variable for file name
        if (specI == "Aedes aegypti (female)"):
            specF = "AeF"
        elif (specI == "Aedes aegypti (male)"):
            specF = "AeM"
        elif (specI == "Anopheles"):
            specF = "Ano"
        elif (specI == "Culex"):
            specF = "Cul"
```

```
        # Record current temperature and humidity
        DHT_SENSOR = Adafruit_DHT.DHT22
        DHT_PIN = 17
        humidityTemp, temperatureTemp =
Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
        humi = round(humidityTemp,2)
        temp = round(temperatureTemp,1)
        print(temp)
        print(humi)

        # Save metadata as variables
        dateI = now.strftime("%Y/%m/%d")
        timeI = now.strftime("%I:%M:%S %p")
        locI = "PSHS-CLC Clark"
        datetimeF=now.strftime("%Y-%m-%d-%H:%M:%S")
        locF = "PSHSCLCClark"

        # Rename file to name with metadata
        fileName = datetimeF + "-" + specF + "-" + locF + ".wav"
        print(fileName)
        os.rename("/var/www/html/aud/recordTemp.wav",
"/var/www/html/aud/" + fileName)
        linkI = "/aud/" + fileName

        # Access server
        mydb =
mysql.connector.connect(host="localhost",user="username",password="",
database="lamoc")
        mycursor = mydb.cursor()

        # Insert data into server
        sql = "INSERT INTO display (species, location,time,date,temp,
humi,link) VALUES (%s,%s,%s,%s,%s,%s,%s)"
        val = (specI,locI,timeI, dateI,temp, humi, linkI)
        mycursor.execute(sql, val)

        mydb.commit()

        print(mycursor.rowcount,"inserted")
```

trainingModel.py Code

```
import math
from scipy.fftpack import fft
import numpy as np
from svmutil import *
import scipy.io
import statistics

def trainingModel(path):
    """
    name: trainingModel. this function will convert a wav file to its
frequency domain using fft
    :param path: the directory of the wav file in which the fft will
be applied
    :return: myFunction(cnt): a string (the class that won out of 1-
vs-many comparison)
    """
```

```
    frame_length = 2048
    rate, data = scipy.io.wavfile.read(path)
    #cutting the audio so that it will be divisible to 2048
    cell_no = math.floor(len(data)/frame_length)
    data = np.array(data[:cell_no*frame_length])

    b = 0
    while (b+1)*frame_length < len(data):
        x_n = data[(b*frame_length):(b+1)*frame_length]
        fft_out = fft(x_n)
        y = np.fft.fftshift(np.abs(fft_out))

        #Cutting the negative data due to fft
        y_new = y[round(len(y)/2):]

        #normalizing the y-axis
        max_amplitude = np.max(y_new)
        y_new = y_new / max_amplitude

        #362 features only was included so that 0 - 8000Hz frequency
would only be included, instead of 0-22050Hz
        y_new = y_new[:362]

        if b == 0:
            cnt = y_new
        else:
            cnt = np.vstack((cnt,y_new))
        b += 1

    return myFunction(cnt)

def myFunction(x):
    #smv model were loaded for the comparison
    model_AeF_BG = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_AeF_BG')
    model_AeM_AeF = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_AeM_AeF')
    model_AeM_BG = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_AeM_BG')
    model_An_AeF = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_An_AeF')
    model_An_AeM = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_An_AeM')
    model_An_BG = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_An_BG')
    model_C_AeF = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_C_AeF')
    model_C_AeM = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_C_AeM')
    model_C_An = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_C_An')
    model_C_BG = svm_load_model('/usr/local/lib/python3.7/dist-
packages/libsvm/models/model_C_BG')

    AeF = 0
    AeM = 0
    An = 0
    C = 0
```

```
BG = 0

#try-catch code where 1-vs-many comparison would be done
try:
    p_labels, p_acc, p_vals = svm_predict([], x, model_AeF_BG)
    if statistics.mode(p_labels) == -1:
        AeF += 1
    else:
        BG += 1

    p_labels, p_acc, p_vals = svm_predict([], x, model_AeM_BG)
    if statistics.mode(p_labels) == -1:
        AeM += 1
    else:
        BG += 1

    p_labels, p_acc, p_vals = svm_predict([], x, model_An_BG)
    if statistics.mode(p_labels) == -1:
        An += 1
    else:
        BG += 1

    p_labels, p_acc, p_vals = svm_predict([], x, model_C_BG)
    if statistics.mode(p_labels) == -1:
        C += 1
    else:
        BG += 1

    if BG == 4:
        return 'Background'

    p_labels, p_acc, p_vals = svm_predict([], x, model_AeM_AeF)
    if statistics.mode(p_labels) == -1:
        AeM += 1
    else:
        AeF += 1

    p_labels, p_acc, p_vals = svm_predict([], x, model_An_AeF)
    if statistics.mode(p_labels) == -1:
        An += 1
    else:
        AeF += 1

    p_labels, p_acc, p_vals = svm_predict([], x, model_An_AeM)
    if statistics.mode(p_labels) == -1:
        An += 1
    else:
        AeM += 1

    p_labels, p_acc, p_vals = svm_predict([], x, model_C_AeF)
    if statistics.mode(p_labels) == -1:
        C += 1
    else:
        AeF += 1

    p_labels, p_acc, p_vals = svm_predict([], x, model_C_AeM)
    if statistics.mode(p_labels) == -1:
        C += 1
    else:
```

```
            AeM += 1

        p_labels, p_acc, p_vals = svm_predict([], x, model_C_An)
        if statistics.mode(p_labels) == -1:
            C += 1
        else:
            An += 1
        #print(AeF)
        #print(AeM)
        #print(C)
        #print(BG)
        var = {'Aedes aegypti (female)':AeF,'Aedes aegypti
(male)':AeM,'Anopheles':An,'Culex':C,'Background':BG}
        return max(var, key = var.get)

    except:
        return 'Undetermined'
```