

Predicting Volume of Mosquito Borne Diseases Using A.M.E.A. Sensor/Radio Network



Student Researchers: Miguel Jose Bueno, James Ervin, Anushka Jain,
Owen Luo, Arnav Tendulkar

Mentors: Dr. Rusty Low, Om Shastri, Cassie Soeffing

Table of Contents:

Abstract	2
Research Questions	3
Introduction and Review of Literature	4
Methodology	6
Results	15
Discussion	16
Conclusion	16
Acknowledgements	17
Bibliography:	17

Abstract

The A.M.E.A. (AI-Powered Mosquito Environmental Analyzer) Sensor/Radio Network is an effective tool towards predicting volumes of mosquito borne diseases. In this study, data based on light, temperature, and precipitation is fed into a ML model that then processes said data, producing Mosquito Borne Disease, (MBD), predictions for that specific area and climate. Will the data received from the A.M.E.A. Network be sufficient enough in creating accurate predictions in a given location? Using a pair of computers and a transmitter (Raspberry Pi Speaker/Kenwood TS-440) and receiver (Computer microphone/Kenwood R600) unit, each connected to their respective radios, we can demonstrate the ability to transmit data using a novel technique with Slow Scan Television (SSTV) images that can be fed into an ML Model that processes and decodes the data and images to create a prediction on the volume of MBDs in a given area. These MBD predictions may then be compared to GLOBE data in order to assess if our MBD predictions are accurate based on a given location and climate. Our decision forest model had an accuracy of 64.6%, and the neural network Google Teachable Machine image-recognition ML model is able to predict the color of an SSTV image with 100% certainty. Since our AMEA sensor/radio network proved to be extremely accurate, in the future, scientists and public health professionals may better prepare for outbreaks in advance, limiting the amount of cases and casualties from mosquito-borne diseases in communities.

Keywords: citizen science; machine learning model; mosquito prediction; sensor/radio network; engineering

Research Questions

Question 1: Will the A.M.E.A. Network be able to collect quality data in a given location?

Determining whether the A.M.E.A. Network is capable of obtaining quality data is a crucial step in this project. This question focuses on the hardware aspect of the project, ensuring that the sensors and network are able to collect real time climate data in a local environment. This will allow researchers to go forward with the machine learning model in order to make predictions. Previous research provides that “data collection technologies, such as satellite remote sensing and low-cost sensors, promises to provide new advances in data collection and monitoring” [6]. We hope to expand on studies like this in order to prove that this type of data collection with sensors is applicable in the scientific world and can be implemented to acquire data.

Question 2: Can the data provided by the A.M.E.A. Network be used to accurately predict the volume of mosquito borne disease human infection in a given location?

The capability to make these predictions is at the forefront of this project. This question hones in on the software aspect of the project; it will rely on the data collection sources, radio transmission, and the machine learning models in order to run properly. All of these elements of the project will have to be fine-tuned before they can all fit together in the overall project. Being able to predict the volume of humans infected with mosquito borne diseases will ultimately demonstrate success in the overall project. Past research has shown that machine learning

strategies can be implemented for prediction of infectious disease spreading, specifically for the spread of COVID-19 pandemic [7]. We aim to expand on this concept in order to produce a model that can accurately predict infection spread, not only through past trends, but also through realtime data collected with sensors. If an accurate machine learning model is achieved within this project, researchers will confidently be able to use the A.M.E.A. Network in order to make these predictions in the future around the world.

Introduction and Review of Literature

Mosquitos are a type of insect that can be found all over the world, with over 3,000 different species existing on several continents and living in various environments. Mosquito-borne diseases are diseases carried by mosquitoes that can be transmitted through the bite of an affected mosquito. These diseases can negatively affect a variety of species, predominantly humans. Mosquito populations, and thus, mosquito-borne virus rates vary due to many factors, including climate, which vary based on location. Therefore, a location's climate factors are indicative of its volume of mosquito-borne virus infections in humans.

Previous projects have utilized sensor networks in order to collect massive amounts of data. Neural network and logistic regression model are less accurate than the Random Forests Model. Using Random Forests, they were able to predict the threat of mosquitos in a given area as a binary (yes or no). They concluded that humidity and temperature were strong indicators while precipitation, cloud cover, and air pressure were weak indicators. [3]

In this study, we tested our idea of using sensor networks to input more locally-sourced data into our Decision Forests data to increase its accuracy. We were able to prove that such an idea could work by predicting the color of SSTV images with background noise using a Google Teachable image-recognition neural network ML model. Data was then collected from the National Oceanic and Atmospheric Administration (NOAA), which described the temperature (F) and precipitation (in) in relation to the amount of Chikungunya, Dengue, and West Nile Virus cases each month in Austin, Texas. We proceeded with a Decision Forests ML model to predict the amount of total cases given the provided features.

In order to connect the network of computers and radios, we've chosen to use High Frequency communications for three main reasons, it requires less energy to create a radio wave in the upper half of the HF spectrum compared to the VHF spectrum and the Ionosphere doesn't support skywave communications as readily compared to the lower portions of the HF spectrum.

The equation below, we can find the amount of energy in Joules that a single second of transmission in the 40 Meter Band (7.175 MHz), 20 Meter Band (14.225 MHz), 10 Meter Band (29.600 MHz), 6 Meter Band (50 MHz), and in the 2 Meter Band (144 MHz) would require [8].

$$e = h \cdot f$$

$$e = \frac{c \cdot h}{\lambda}$$

'c' is light velocity 299,792,458 mtrs /sec
'e' is energy in joules
'f' is frequency in cycles per second
'h' is Planck's constant
 $6.6260695729 \times 10^{-34}$ Joule • seconds
 λ is the Greek letter Lambda and it represents wavelength in meters

www.1728.org

(Note: One cycle per second is equivalent to one hertz.)

$$(6.626 * 10^{-34} \text{J} / \text{s}) * (7.175 * 1000000) = 4.754155 * 10^{-27} \text{J}$$

The result for 7.175 MHz in the 40 Meter Band.

$$(6.626 * 10^{-34} \text{J} / \text{s}) * (14.225 * 1000000) = 9.425485 * 10^{-27} \text{J}$$

The result for 14.225 MHz in the 20 Meter Band.

$$(6.626 * 10^{-34} \text{J} / \text{s}) * (29.600 * 1000000) = 1.961296 * 10^{-26} \text{J}$$

The result for 29.600 MHz in the 10 Meter Band.

$$(6.626 * 10^{-34} \text{J} / \text{s}) * (50 * 1000000) = 3.31300 * 10^{-26} \text{J}$$

The result for 50 MHz in the 6 Meter Band.

$$(6.626 * 10^{-34} \text{J} / \text{s}) * (144 * 1000000) = 9.54144 * 10^{-26} \text{J}$$

The result for 144 MHz in the 2 Meter Band.

The Ionosphere is a layer within the Earth's Atmosphere that's comprised of gases that were ionized by solar radiation. This ionization process occurs whenever a UV ray from the Sun strikes a gas particle and frees an electron from the particle. The remainder is a cation and an electron. The density of free electrons is what affects radio waves. [10]

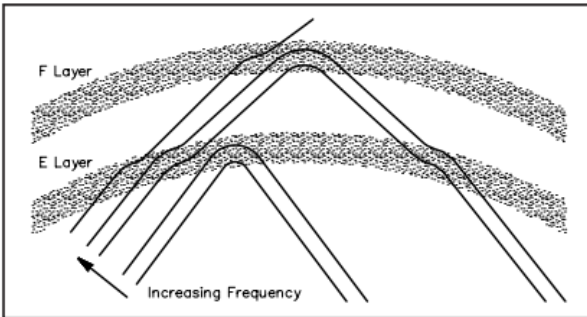


Figure 5—Refraction of a signal as it enters an ionized region.

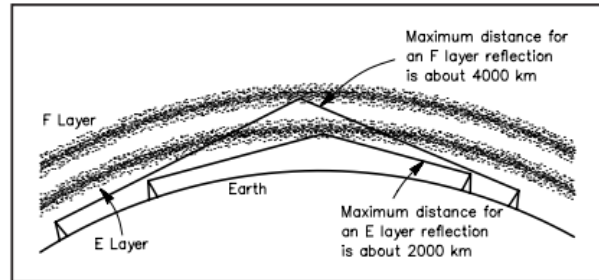


Figure 6—Signals reflected by the E and F layers.

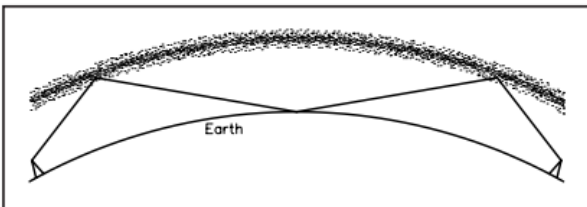


Figure 7—Multiple reflections.

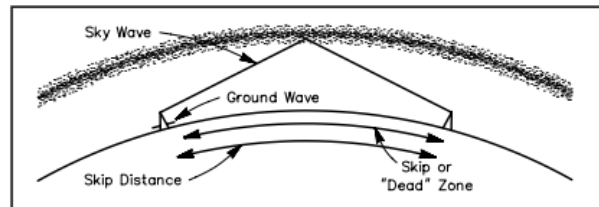


Figure 8—Skip distance and dead zone.

Image from November 1999 QST from Ian Poole's Radio Waves and the Ionosphere.

During the Sun's eleven year solar cycle, its level of electromagnetic radiation varies as it follows its pattern of highs and lows. During the lows, High Frequency Bands above 20 MHz may not be reflected by the Ionosphere while during the highs, Very High Frequency Bands around 50 MHz or above may be propagated by the Ionosphere. [10] It's important to note that the High Frequency Spectrum covers radio frequencies from 3 MHz to 30 MHz while the Very High Frequency Spectrum covers radio frequencies from 30 MHz to 300 MHz [9]. In order for us to take advantage of the lower power requirements and a lowering the chance of Ionospheric skip creating interference with our data reception, we've chosen to use frequencies close to the Amateur 10 Meter Band, and for experimentation, specifically 29.606 MHz.

Methodology

The process of data collection and transmission can be broken into five parts: Data Collection/Encoding, Data Transmission, Data Reception, Data Decoding, and Machine Learning (ML) Processing and Analysis. A significant portion of the project was the development of software that, when paired with the right hardware, is capable of creating a sensor network that has data input around the hour via groundwave radio communications.

The project started with data collection. The data collection for our project was conducted in Travis County, Texas, where the climate factors are optimal for mosquito oviposition and vector transmission.



Map of Travis County

The climate factors used to train the ML model were temperature and precipitation. Researchers accessed National Weather Service/NOAA Online Weather Data from the Austin, Texas area, to obtain temperature and precipitation data [12]. They also used the GLOBE Advanced Data Access Tool with the Air Temperature Monthlies protocol within 300 km of Austin, Texas to obtain more temperature data [13]. Additionally, researchers needed data on mosquito-borne disease infection rates in Travis County, and this was obtained from the Texas Department of State Health Services Arbovirus Weekly Activity Reports, focusing on the Chikungunya Virus, Dengue Virus, and West Nile Virus in Travis County, Texas [5]. All the data was collected on a month-to-month basis, meaning that the temperature and precipitation data represented monthly averages while the infection data represented the number of human cases per month.

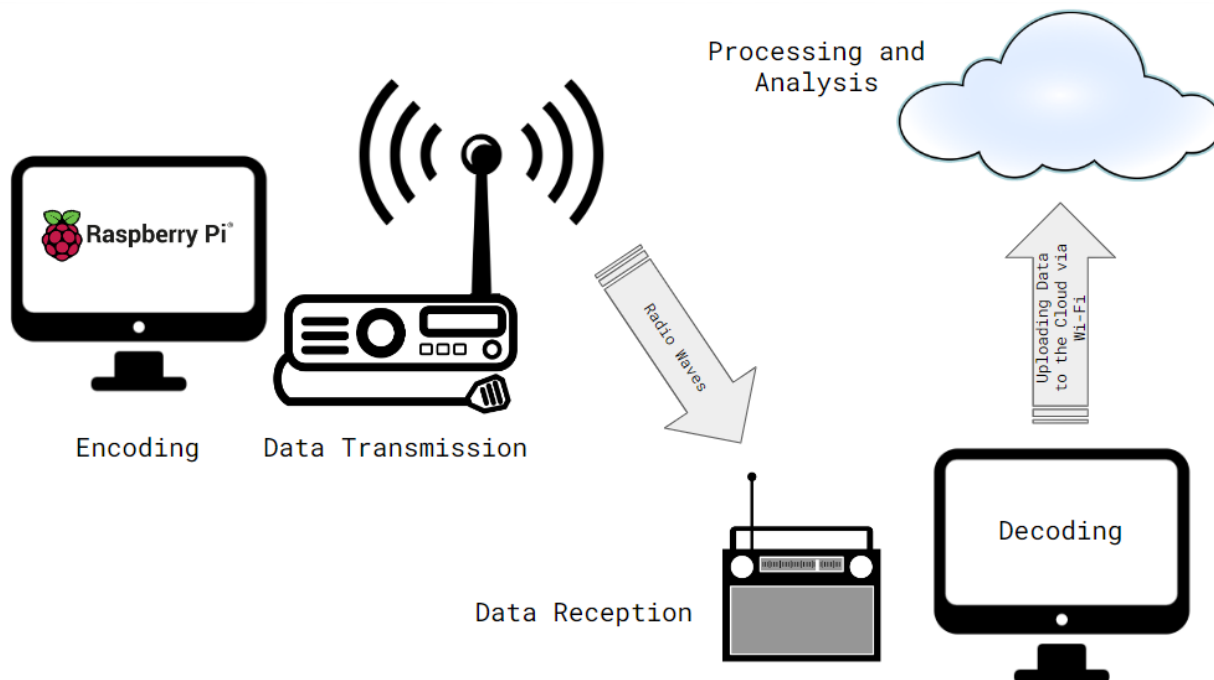
Temperature (F)	Precipitation (in)	Month	Chikungunya Virus Cases	Dengue Virus Case	West Nile Virus Cases	Total
82	0.1	June	0	1	0	1
84.4	0.18	July	2	1	1	4
87.1	0	August	2	1	1	4
80.6	0.23	September	5	3	4	12
74.5	0.06	October	6	3	5	14
55.7	0.19	November	7	3	5	15
55	0.03	December	8	3	6	17
48.7	0.16	January	0	0	0	0
51.3	0.02	February	0	0	0	0
60.2	0.16	March	0	0	0	0
71.2	0.08	April	0	0	0	0
75.2	0.57	May	2	0	0	2
81.1	0.3	June	2	1	0	3
85.6	0.4	July	2	1	0	3
87.4	0.01	August	3	1	0	4
83.4	0.06	September	3	3	0	6
74.6	0.38	October	4	4	0	8
61.5	0.12	November	4	5	0	9
57.1	0.1	December	4	5	0	9
51.9	0.01	January	0	0	0	0
59.8	0.05	February	0	0	0	0
66.1	0.11	March	0	0	0	0
70	0.24	April	0	0	0	0
74.2	0.23	May	1	1	0	2
83.1	0.09	June	1	1	0	2
88	0.06	July	1	3	0	4
84.4	0.22	August	1	3	0	4
82.5	0.07	September	1	3	0	4
75.6	0.01	October	1	4	1	6
65.7	0.1	November	1	4	2	7
55	0.08	December	1	4	3	8

During the encoding process, a Raspberry Pi computer captures a picture with its camera and runs a brief script to analyze the amount of brightness in the picture. The Raspberry Pi then resizes and rescales the image from the camera in order to prepare it for encoding. Once the image is prepared, the PySSTV library turns the image into a 320×240 grid and then converts each image in the grid into a sound that the decoder can recognize and process into an image. After processing all 76,800 pixels into sounds, it converts the sounds into .wav files, ready to be transmitted into the airwaves.

The number that comes from the analysis of the light levels is also similarly processed but instead of using a webcam picture, each digit/character of the number is converted into a solid color picture that's then converted to a .wav file and stored for transmission.

Zero (0)	One (1)	Two (2)	Three (3)	Four (4)	Five (5)	Six (6)
Seven (7)	Eight (8)	Nine (9)	Positive (+)	Negative (-)	Decimal (.)	

During the transmission process, the stored .wav files are transferred to the transceiver via a sound card. Essentially, the computer treats the radio like a pair of headphones, except instead of converting the output from the computer into audio that a human could understand, it converts it into radio waves that are sent out to be received by any radio listening/receiving on the frequency that the transceiver is transmitting on.



During the reception process, the computer treats the receiver radio as a microphone. Instead of the radio putting the audio out to its speaker, instead it diverts it into the computer where the converted radio waves can be analyzed, processed, and used. The decoding program records a four minute segment directly from the radio's input to the computer.

During the decoding process, the decoder software separates the recording into six different chunks. The first chunk is the encoded SSTV image. The second through sixth chunks are the encoded characters of the data. The decoder then runs each one of these files through colaclanth's SSTV Decoder and reverses the process that the encoder did to prepare the images for transmission resulting in usable images. Once all the images have been decoded and stored, the computer then uploads the data to the cloud where our ML model can process it and make predictions of the volume of Mosquito-Borne Diseases in the area where the A.M.E.A. Network has been deployed.

[KI5UXW/SSTV-and-Data-Transmission](#) is a Python Program that we've written that serves as the data collection and transmission functionality when connected to a Kenwood TS-440S or any other similar radio that supports AFSK. Using a TRRS to RCA adapter then an RCA Stereo to Mono connector allows for the computer's audio input and output to match perfectly with the TS-440S's AFSK ports.

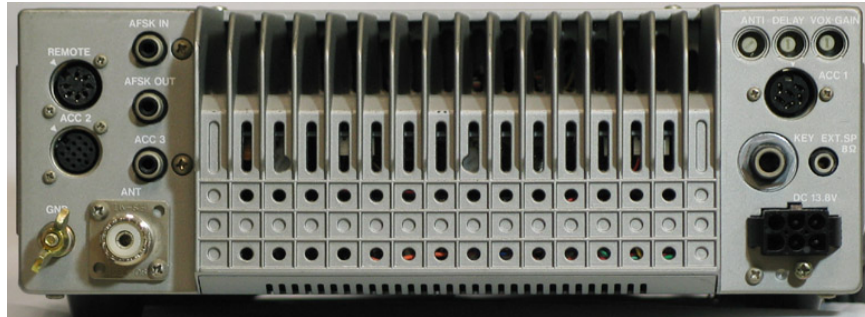
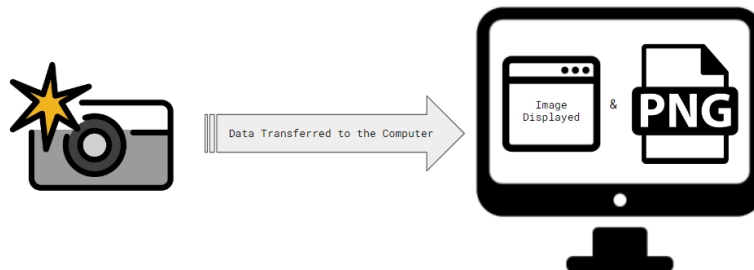


Image Courtesy of Universal Radio Inc.

The major functions of the program can be broken into two parts, the image transmission and the data transmission portions. The output from the SSTV picture analysis of the light levels is used as data to be transmitted back to the base station/reception program.

In order to increase legibility of the program, each of the major functions of the SSTV transmission have been organized into individual functions that can be reused and repurposed later.

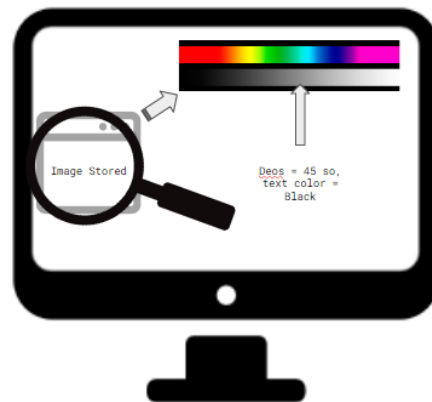
Python's OpenCV library is used to take images from the webcam. It's a simple and straightforward script that was borrowed from GeeksForGeeks and lightly repurposed to work with the code. [1]



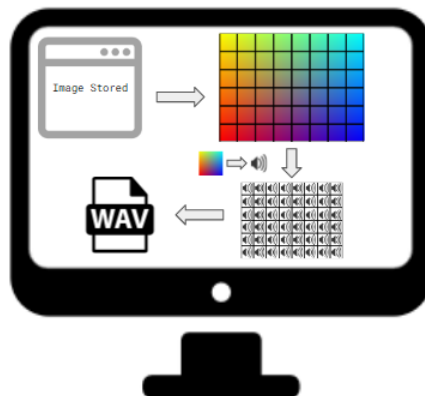
Another script from Stack Overflow was then found, that makes use of the PIL/Pillow Library to analyze the level of brightness that the webcam captured within the image which could be used alongside a photometer to get an accurate idea of the amount of light within a given area. This program originally gave ranges from 0-250 but at the time when it was being developed, a scale from 0-100 was decided to be better. It was then decided that we'd name a new unit of measurement called the Deo, based off of the the range of a solid black image would be 0 Deo and a solid white image being 100 Deos.



The collection of the data of the amount of light in a picture isn't completely useless by itself. It's quite useful in creating an image that is still useful and readable to the human eye. Using the PIL library again, the light level in Deos is taken and used to choose the color on the text that is added to the image taken. After adding the text, the program overwrites the original file in order to save space. The Raspberry Pi font base contrasted with the computer font base, in that 'Arial.tiff' was unavailable in the Raspberry Pi font library, which resulted in said font being replaced with an already present font.



Then, PySSTV converts the image created into a usable SSTV transmission stored in a .wav file, after resizing the image to the resolution that the Robot36 Encoding protocol requires.



Finally, a header is added, which allows any Robot36 decoder to begin processing the image, then use the winsound module to play both the header and the transmission file previously created.



For reference, here are all the import statements used in order to ensure the functionality of the transmission script. When working with the Raspberry Pi 3, there were many adjustments needing to be made to said code, including installing the pillow, opencv, and pygame packages instead of the PIL, cv2, and winsound packages.

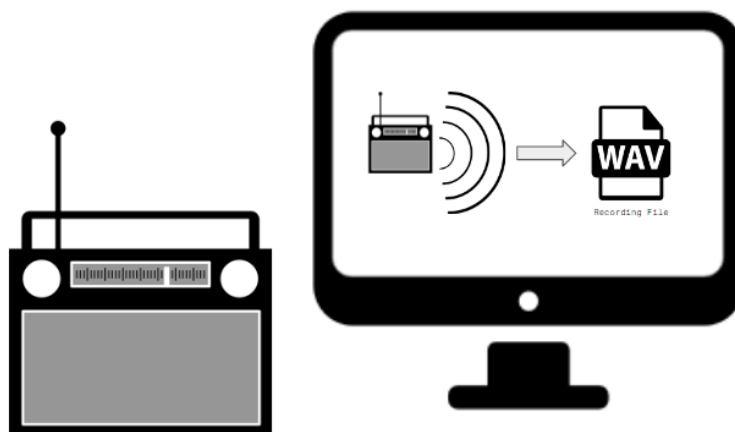
```
import cv2
import time
import PIL
import pysstv
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
from PIL import ImageStat
from pysstv import sstv
from pysstv.color import Robot36
import winsound
import math
```

The decoder works in a very similar manner. It takes the light level in Deos, saved from the previous function, and converts it into a five character list. The list starts with either a positive or a negative symbol, then is followed by a number with three digits on the left side of the decimal point, a number with two digits on the left side of the decimal point with another on the right, or a number with one number on the left side of the decimal point and two on the right. Then, the program converts this number to a string and splits its contents into a list. This list is then read from and, using a very similar method to the SSTV Image Conversion, it draws from its database of solid colors that've been chosen to represent numbers, converts the images to a .wav file, adds a header, and then transmits them.

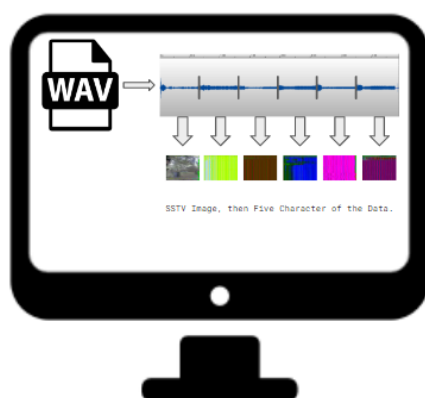
Zero (0)	One (1)	Two (2)	Three (3)	Four (4)	Five (5)	Six (6)
Seven (7)	Eight (8)	Nine (9)	Positive (+)	Negative (-)	Decimal (.)	

While data is being transmitted, another program is running at the same time recording and decoding the output from the first program. [KI5UXW/SSTV-Decoder](#) is a Python Program built off of [colaclanth/sstv](#) that takes a four minute recording during a window of time that matches up with the reception of the transmission. The code is divided into three main parts, the recording section, the audio division section, and the SSTV decoding section.

Using Python's sounddevice library, a four minute recording is created from the Kenwood R600's output and is saved to a .wav file for later processing.



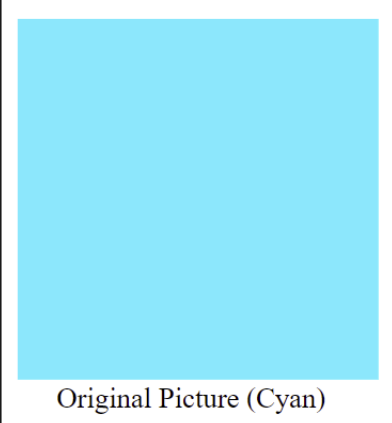
Then Python's powerful pydub library is used to analyze and segment the recording. The library is set so that whenever the audio level on the recording dips below -50 db and lasts longer than 250 ms, it splits the audio file. After all the splits have been made, the program then exports the chunks it created into .wav files that're then processed by the next step of the program. In order to save time and complexity, the program inputs its requests to use colaclanth's SSTV Decoder library via the command prompt. The program then saves the decoded images as .png files with the date and name.



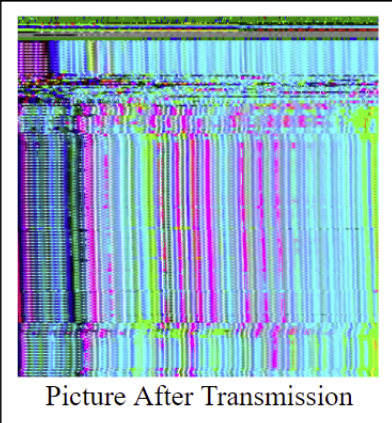
Following image decoding, the images are run through the neural network Google Teachable Machine image-recognition ML model which works to identify colors. As discussed before, a unique color code was set up in which colors correspond to specific digits, as seen below.

Zero (0)	One (1)	Two (2)	Three (3)	Four (4)	Five (5)	Six (6)
Seven (7)	Eight (8)	Nine (9)	Positive (+)	Negative (-)	Decimal (.)	


The images started off as solid blocks of color before transmission. However, transmission through the airwaves gets obstructed by interference which causes discoloration and choppiness to appear in the images. Because the images are not exact replicas after transmission, a Google Teachable Machine image-recognition machine learning model was created to recognize and distinguish the transmitted color. In order to train this model, many images for each color had to be uploaded to the training dataset. To create these images, they were transmitted through the radio network multiple times, each time with a different distance between the transmitter and the receiver with varying levels of volume. The resulting images all had unique amounts of interference and were varying examples of what transmitted images could look like in the final project, which is why they made up a good training dataset. These images were then uploaded into the Google Teachable Machine program and linked up to their corresponding digits. For example, the transmitted cyan pictures were uploaded together under the “0” digit. After the model was trained with this dataset, it was able to predict the color of a transmitted image, and effectively convert the images back into a number code.



Original Picture (Cyan)



Picture After Transmission



Machine Learning Model Output

Output

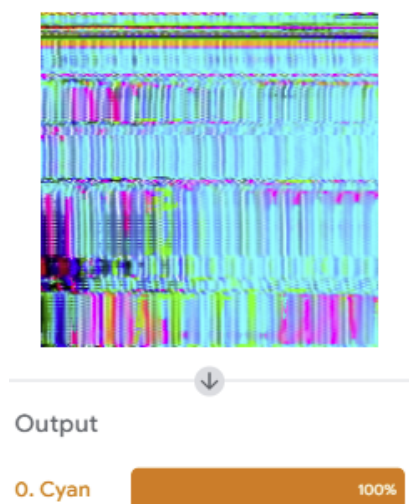
0. Cyan	100%
1. Red	
2. Yellow	
3. Blue	
4. Purple	
5. Orange	
6. Brown	
7. Black	
8. White	
9. Gray	
Positi... Green	
Negat... Aqu...	
Deci... Pink	

In the example above of a cyan SSTV image, the model is 100% certain that the color displayed onscreen is cyan because it has compared the received image to other images of cyan that have also been transmitted and put into the database. This procedure, now that it has been proven to work, can be utilized with more resources and time in the future to identify more climate data, such as ground moisture.

Results

In order to test the transmission of data and the reception quality, we ran 100 watts of power into a center-fed, five meter dipole using Single Sideband Modulation, or SSB. Testing the reception at 780 ft, we used a Kenwood R600 Receiver with a RadioShack 20-161B Telescoping Scanner Antenna and determined the reception quality to be significantly higher than a 5/9. For reference, the RST is a signal classification system developed by Arthur W. Braaten in 1938. RST stands for Readability, Strength, and Tone and serves as a sort of rubric that Amateur Radio Operators use to grade each other's signals and reception. Most of the time, RST is shortened to RS due to tone being only used by Morse Code operators. Readability is on a scale from one, being unreadable, to five, being perfectly readable. Strength is measured on a scale from one, being faint—signals barely perceptible, to nine, being extremely strong signals. So our result that was deemed to be higher than a 5/9, is a signal that is easily readable, where a computer could easily distinguish the different tones being transmitted, and with an extremely strong signal, where a computer can easily make a high quality image with very little to no interference appearing in its result.

We were able to plug in data from the National Oceanic and Atmospheric Administration from Austin, Texas into two ML models designed to predict the volume of mosquito-borne diseases. We used two different ML models. The first one being a linear regression model which is the process of creating a line that best fits the data points given to the trained model and this is the most simple form of a machine learning model. The second model we made is a random forest model that creates a forest full of decision trees that creates different predictions and shows us which decision tree is the most accurate. The Decision Forests model had an accuracy of 64.6%, while the linear regression model only had an accuracy of 37.5%. Decision Forests is far more accurate, so we kept the Decision Forests model and left the linear regression model out. In the future, we would like to add a column for light level from our sensor network. We would like for our own sensor network to feed additional features and respective data values into our current Decision Forests ML model. For now, this portion is unutilized, but it has lots of potential seeing that the neural network Google Teachable Machine image-recognition ML model is able to predict the color of an SSTV image with 100% certainty.



Accuracy of Color ID ML Model	
Color	Accuracy (%)
Cyan	100
Red	100
Yellow	100
Blue	100
Purple	100
Orange	100
Brown	100
Black	100
White	100
Gray	100
Green	100
Aquamarine	100
Pink	100

Discussion

Our project was able to track and predict the amount of Chikungunya, Dengue, and West Nile Virus cases. These cases are vector diseases, which means that they are carried by mosquitoes. This can be a big problem for tropical regions as well as areas in the world with poor health infrastructure. In fact, such developing regions have less access to air conditioning, which prompts people to seek the coolness of shaded, well-ventilated areas, precisely the sites where *Ae. aegypti* prefers to feed, directly causing the increase of Dengue cases. [4]

A 100%-certain image-recognition ML model proves that our sensor network would be able to acquire accurate data in the future to predict the amount of mosquito-borne diseases. With sensor-created data, we would be able to obtain more localized data opposed to the ones in city-wide datasets. Moreover, the readability of data at an elevation of 780 ft proves that our sensor network device works at all altitudes. Our 64.6% accuracy level for the NOAA Decision Forests ML Model is still not ideal though, as we don't have much data in the datasource for Austin, Texas. Some possible sources for the low accuracy has to do with the ML model being overfitted with data which made the model accurate only to that data.

Overall, our A.M.E.A sensor network as well as an improved ML model would give scientists and citizens around the world a heads up before a major mosquito-borne outbreak occurs. Based on these findings, in the future, scientists and public health professionals will be better prepared for outbreaks in advance, limiting the amount of cases and casualties from mosquito-borne diseases in communities.

Conclusion

Our project was able to track and predict the amount of Chikungunya, Dengue, and West Nile Virus cases. These cases being vector diseases, meaning they were carried by mosquitos, can be a big problem for tropical regions as well as areas in the world with poor health infrastructure. Our model and sensor network gives scientists and citizens around the world a

heads up before a major mosquito-borne outbreak occurs. This enables citizens around the world to protect themselves from possible mosquito-borne viruses by knowing general data trends around their area.

Based on the current 100% certainty of the color-identifying model and the 64.6% accuracy of the prediction machine learning model, we are confident that we can continue working on this project to make it more accurate and precise. Next steps include adding more data to the training datasets and adding more climate factors such as humidity. If these are implemented, the A.M.E.A. network will be much more functional and advantageous. Based on these findings, scientists and public health professionals in the future may better prepare for outbreaks in advance, limiting the amount of cases and casualties from mosquito-borne diseases in communities.

Working with project mentors has allowed us to have multiple points of view regarding our research, authorizing us to make sensible and required changes to our research methods and experiment protocols.

Acknowledgements

We thank SEES Earth System Explorers mentors Dr. Rusty Low, Om Shastri, and Cassie Soeffing for providing their assistance for this research project.

Bibliography:

- [1] “How to capture a image from webcam in python?” GeeksforGeeks. (2021, December 21). Retrieved July 25, 2022, from <https://www.geeksforgeeks.org/how-to-capture-a-image-from-webcam-in-python/>
- [2] “Kenwood TS-440S Rear Panel.” Kenwood TS-440s Rear Panel, Universal Radio Inc., <https://www.universal-radio.com/catalog/hamhf/ts440sr.html>.
- [3] Polineni S, Shastri O, Bagchi A, Gnanakumar G, Rasamsetti S, Sundaravadivel P. MOSQUITO EDGE: An Edge-Intelligent Real-Time Mosquito Threat Prediction Using an IoT-Enabled Hardware System. *Sensors*. 2022; 22(2):695. <https://doi.org/10.3390/s22020695>
- [4] Reiter P. Climate change and mosquito-borne disease. *Environ Health Perspect*. 2001 Mar;109 Suppl 1(Suppl 1):141-61. doi: 10.1289/ehp.01109s1141. PMID: 11250812; PMCID: PMC1240549.
- [5] Texas Department of State Health Services. “DSHS Arbovirus Weekly Activity Reports.” *Texas Department of State Health Services*, <https://dshs.texas.gov/idcu/disease/arboviral/westNile/reports/weekly/>.
- [6] Hsu, Angel, et al. “Next-Generation Digital Ecosystem for Climate Data Mining and Knowledge Discovery: A Review of Digital Data Collection Technologies.” *Frontiers*, *Frontiers*, 1 Jan. 1AD, <https://www.frontiersin.org/articles/10.3389/fdata.2020.00029/full>.

- [7] Ghafouri-Fard, Soudeh, et al. "Application of Machine Learning in the Prediction of COVID-19 Daily New Cases: A Scoping Review." *Heliyon*, Elsevier, Oct. 2021, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8503968/>.
- [8] "Electromagnetic Frequency, Wavelength and Energy Ultra Calculator." *1728.Org*, <https://www.1728.org/freqwave.htm>.
- [9] "HF Frequency Band." *Encyclopædia Britannica*, Encyclopædia Britannica, Inc., <https://www.britannica.com/technology/HF>.
- [10] Poole, I. (1999, November). Radio waves and the ionosphere - ARRL. ARRL. Retrieved July 28, 2022, from <https://www.arrl.org/files/file/Technology/pdf/119962.pdf>
- [11] Mbaabu, Onesmus. "Introduction to Random Forest in Machine Learning." Section.io, <https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/>
- [12] US Department of Commerce, NOAA. *Climate*, NOAA's National Weather Service, 3 Mar. 2022, <https://www.weather.gov/wrh/climate?wfo=ewx>.
- [13] "Globe Advanced Data Access Tools." *GLOBE Advanced Data Access Tools*, <https://datasearch.globe.gov/>.