Mosquito Breeding Sites Prediction System

By

Swarnaditya Maitra
201701006
Email: 201701006@daiict.ac.in

Amit Agarwal
201701060
Email: 201701060@daiict.ac.in

A Project Report - Thesis
Submitted in Partial Fulfilment of the Requirements for the course of
IE407-Internet of Things (IoT)
in
Bachelor of Technology in Information and Communication Technology (ICT)
to
Dhirubhai Ambani Institute of Information and Communication Technology

Assigned and Supervised by **Dr. Sanjay Srivastava**
DA-IICT Gandhinagar
Email: sanjay_srivastava@daiict.ac.in

Declaration

I hereby declare that:

i) The thesis comprises of my original work towards the degree of Bachelor of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree.

ii) Due acknowledgment has been made in the text to all the reference material used.

Swarnaditya Maitra
Amit Agarwal

Certificate

This is to certify that the thesis work titled Mosquito Breeding Sites Prediction System has been carried out by Swarnaditya Maitra and Amit Agarwal in the course IE407 - Internet of Things for the degree of Bachelor of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology under my/our supervision.

Dr. Sanjay Srivastava
Thesis Supervisor

Acknowledgement

In the progression of completing this project successfully, many people have bestowed upon us their blessings and concrete support. We would like to thank all the people who have willingly helped us out with their abilities and advice.

We begin by heartily expressing our sincerest gratitude and respect for our course instructor - Professor Sanjay Srivastava, without whose insights and imperative guidance, this project wouldn’t have come into view. He sparked an interest in the subject of Internet of Things and thanks to his exceptional views, we have grown a lot through this course and through this project. His valuable guidance and support was always there, throughout this journey. He allowed us the freedom to carry out this project in our own way, and at the same time, gave us the right guidance at the right time and driving us in the right direction. His guidance has enhanced our capability in practical manners. We are greatly thankful to him for giving us such freedom to choose the topic that we like along with the freedom to explore the topic at our will and giving us a chance to showcase our creativity.

We would also like to thank the Teaching Assistants - Abhishek Jani, Nikita Joshi, and Maitri Vaghela for guiding us throughout the project, proving to be the backbone without which we could not have overcome the impediments in the assembly of this project. Last but not the least, we would like to thank the institution for providing us with the necessary infrastructure for the completion of this project.

Contents

Declaration 2

Certificate 2

Acknowledgement 3

Abstract 5

Important Background Information 6

1 - Introduction **11**

1-a: Motivation 11

1-b: Problem Statement 11

1-c: Target Users 12

2 - Literature Survey **13**

a) Capturing mosquitoes and analyzing wing-beat frequency 13

b) An ML Approach for Identifying Mosquito Breeding Sites via Drone Images 16

3 - Specific Challenges in our Project **17**

4 - Complete System Design Details **18**

4-a: Components of the System 18

1) Hardware 18

2) Software 18

4-b: System Architecture 19

1) Diagram 19

2) Sensors 19

3) Processing Unit 20

4) Cloud-Services (AWS) 20

5) Web Application - User Interface 21

4-c: Brief Chronology of System Modeling 21

4-d: Methods and Algorithms Used 22

5 - Detailed Experiment Working and Analysis **26**

5-a: Pre-processing 26

5-b: Backend - AWS 29

1) AWS IoT Core 29

2) AWS DynamoDB 30

3) AWS Lambda 31

4) Amazon API Gateway: 34

5-c: FrontEnd UI - Web Application 35

6 - Results 37

7 - Testing plans for correctness and performance **39**

8 - Conclusion **40**

9 - Future Possibilities **41**

10 - References **42**

Abstract

Mosquitoes are one of the deadliest creatures in the world. Their ability to carry and spread diseases to humans causes hundreds of thousands of deaths every year. In 2015 malaria alone caused 438,000 deaths. The worldwide incidence of dengue has risen 30-fold in the past 30 years, and more countries are reporting their first outbreaks of the disease. Zika, dengue, chikungunya, and yellow fever are all transmitted to humans by the Aedes aegypti mosquito. More than half of the world's population live in areas where this mosquito species is present. By 2050, half the world's population could be at risk of mosquito-borne diseases like dengue fever or the Zika virus, new research suggests.

Sustained mosquito control efforts are important to prevent outbreaks from these diseases. The most significant step in this problem is to efficiently and accurately predict the places that have the highest probabilities of emerging as mosquito breeding hotspots. While efforts in the same have been undertaken by WHO and governments all over the world, the status quo largely remains the primitive approach of taking damage-control steps after the outbreak has already begun. By then, it is already late.

We propose a mosquito breeding sites prediction system that leverages IoT technologies, Machine Learning algorithms, and other Information Technology systems, to predict those geographical locations in the area of deployment, that have a high probability to turn into a dangerous mosquito breeding site, owing to a large number of environmental parameters that are most conducive to mosquito breeding. We use a variety of sensors, microcontroller modules, low-cost and low-powered IoT messaging protocols like MQTT, and AWS Cloud services to solve this issue, along with a user-friendly Web Interface providing all the necessary information generated using this system.

Important Background Information

Although the importance of abandoned tires, broken pipes, dumpster containers as a habitat for mosquitoes is well documented, there is surprisingly little information about how environments in and around these containers relate to mosquito populations and communities. There have been several studies that have focused on how environmental factors affect tire-inhabiting mosquitoes. After much research, it was found that a variety of water chemistry and habitat variables affected mosquitoes' breeding. Shading by overhanging vegetation (measured using spherical densiometer) was an important factor affecting mosquito communities, and several measured factors (e.g., turbidity, color) appeared to affect the abundance of some species, but few habitat and water chemistry variables were correlated with mosquito abundances. There have been very few investigations of the relationship between many environmental parameters (e.g., tire size, microorganism abundances, detritus) and mosquito community composition across a wide geographic area, where significant variation in species composition and such environmental factors is expected to occur.

Hence, with the aid of various research papers, we accumulated some results that we seek to use in our project.

Discarded vehicle tires are a common habitat for container mosquito larvae, although the environmental factors that may control their presence or abundance within a tire are largely unknown. The presence of mosquito larvae in containers like broken pipes, dustbins, tires, etc, is the result of two processes:

1) Female oviposition choice.
2) Larval survival.

Female mosquitoes use a variety of sensory cues to locate potential oviposition sites, including olfactory (e.g., volatiles from the aquatic habitat), tactile (e.g., container surfaces), and visual (e.g., color) cues . After hatching, survival of larvae depends on availability of food resources (detritus, microorganisms), severity of intra- and interspecific interactions (competition, predation), and tolerances to physical factors (e.g., pH, temperature, salinity, drying). Thus, the actual number of offspring produced from a given female is likely a function of a female's ability to find oviposition sites and the quality of those locations. How these two forces shape the abundance and prevalence of mosquito larvae in tires is essentially unexplored.

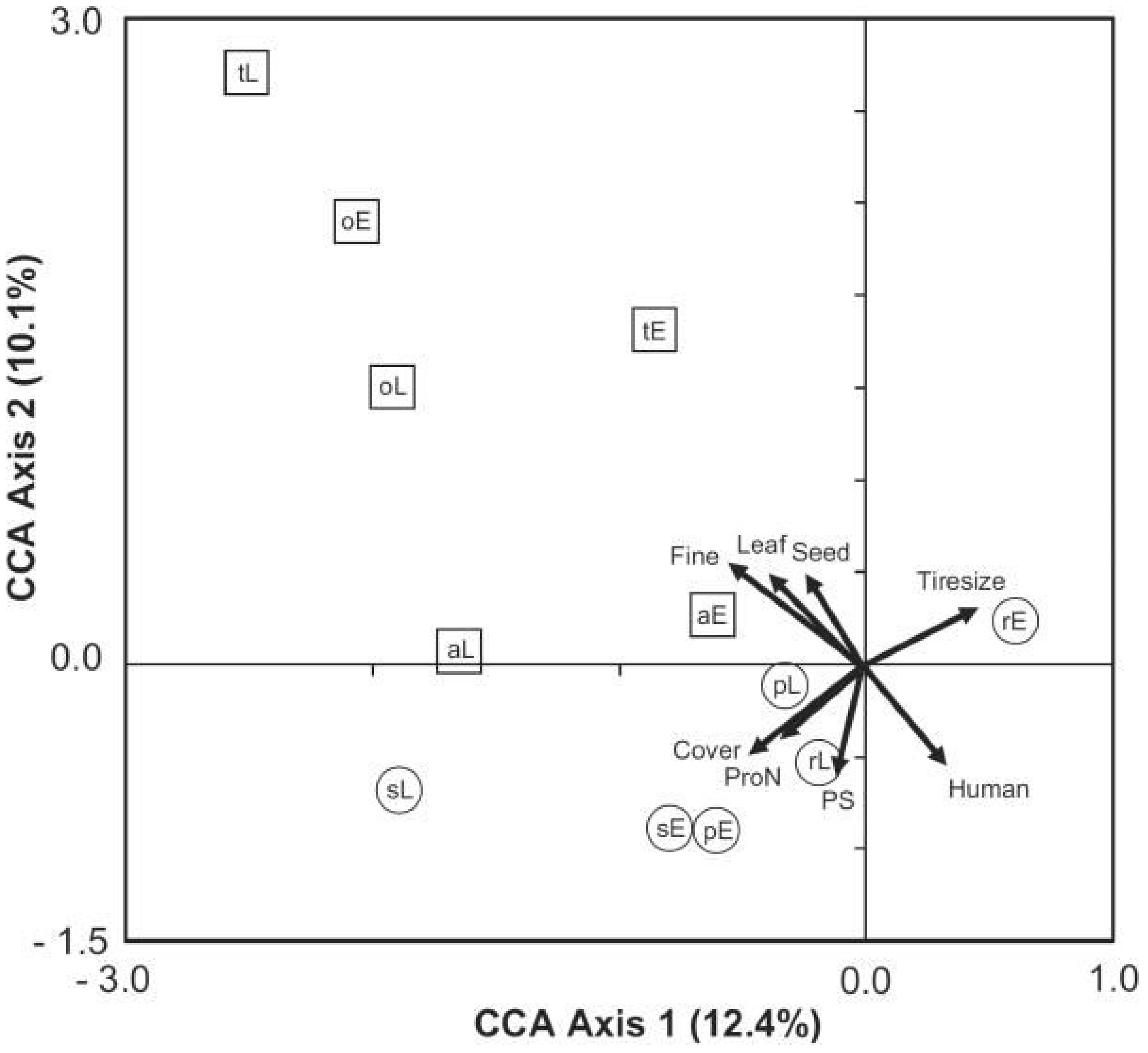
For each container like broken pipe, tires, dumpsters, we can measure a suite of environmental factors in three broad categories that have been seen to affect container mosquito populations :

1) Microorganisms (protozoan richness, protozoan total abundance, bacterial biomass production, algae).
2) Detritus and nutrients (leaves, twigs, seeds, animal detritus, fine detritus, conductivity), which are essential for larvae to grow.
3) Factors related to habitat size or the terrestrial matrix (human density, canopy cover, tire size, tire volume).

CCA: Responses of mosquito breeding stages to the above suite of environmental factors for each season were assessed using **canonical correspondence analysis (CCA)**. CCA is an ordination technique that considers species and environmental variables simultaneously, and is particularly useful when there are underlying unimodal abundance distributions. This technique produces canonical axes that are constrained to optimize their relationship with a set of **environmental variables represented as vectors.**

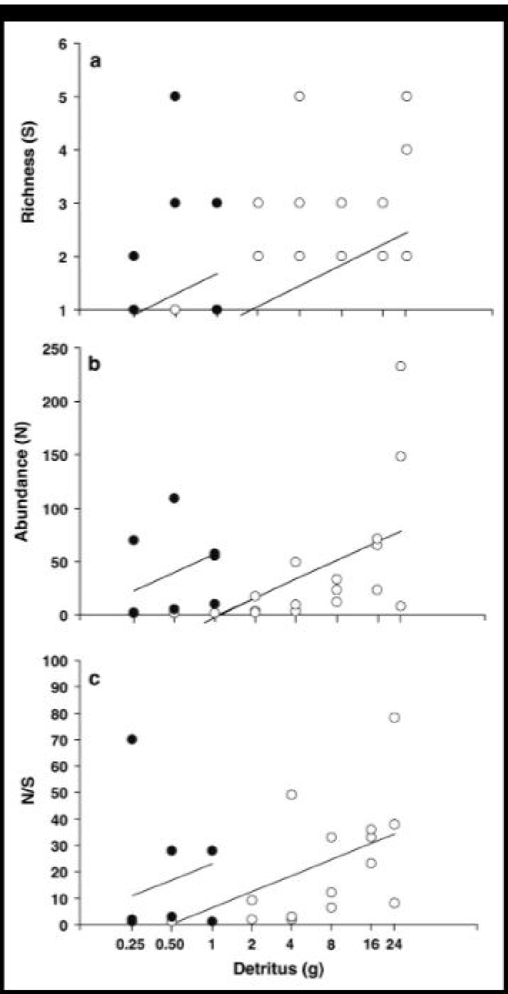
1) The lengths of these vectors indicate the rate of change in that environmental variable; shorter vectors indicate little variation whereas longer arrows reflect more variation among samples.
2) Areas that are perpendicular to the vectors represent no change in that variable. As with other ordination techniques (e.g., principal components analysis) the axes are orthogonal and thus are uncorrelated with one another, although vectors on the same axis that form acute angles are positively correlated, those that form obtuse angles are negatively correlated.
3) Species are represented by points on the diagram that characterize approximate weighted means of the species with respect to environmental variables.
4) The distance between species points and vectors identifies the relative importance of environmental variables in explaining species abundance patterns; points closer to vectors are more influenced by that environmental variable than points farther away.

Beginning with a consideration for 14 factors, CCA shows that the majority of the dependencies are captured by 8 factors. Thus, stepwise selection of environmental variables reduced the factors from fourteen to eight, including three types of detritus (fine, leaf, seed), two microorganism factors (PS, protozoan abundance), and three factors in the habitat category (canopy cover, human density, tire size).



In this project, we directly use the above result, to select the parameters related to **detritus** and nutrients that have the largest influence on mosquito proliferation.

Another graph that represents the variation in mosquito abundance and richness with detritus content in the container, is attached below.



See this image and copyright information in PMC

Fig. 2 Effect of animal (*filled circle*) or leaf (*open circle*) detritus and detritus amount (g) on species (**a**) richness, abundance (**b**), and abundance (**c**) divided by richness in microcosms simulating tree holes during the final month of this study (September 2004). Regression lines are significant ($P < 0.05$) based on ANCOVA. Values on the *x*-axis are presented on a log scale

1 - Introduction

1-a: Motivation

There are on average, over 725,000 reported deaths all over the world, in a year, due to mosquito-borne diseases, making it the officially largest killer in the world. The endemic diseases caused by mosquitoes are perennial. Having said that, what really motivated us to take up this project relates to a very grim situation that we experienced not too long back, in our institute itself.

DA-IICT fell prey to a deadly dengue outbreak in 2019, in which more than 100 students fell ill and had to be hospitalized. The sad part, which motivated us to work on it is that, once mosquito breeding had already occurred in large numbers, and many students had already fallen ill, only then the severity of the situation was realized and the institute called in fumigation. This happens to be the status quo in the majority of the cases in India. Due to the inefficiency of the process, multiple rounds of fumigation were needed to flush out the mosquitoes. Even then, the locations where fumigation needed to be done, i.e., the breeding grounds of the mosquitoes, were determined by government officers after a lot of effort.

This is the part that we wish to work to automate, using IoT concepts and Information Technologies. If the mosquito breeding sites could be predicted and their location obtained, reasonably in advance, this information could be passed on to the relevant authorities. Using this information, necessary measures could be undertaken and all the residents in the vicinity of the areas could be warned. We hope that this will immensely enable the prevention of any disease outbreak.

1-b: Problem Statement

To predict the locations in any region (DA-IICT campus in this project), where the probability of those locations being mosquito breeding grounds is high, using parameters like water stagnation at the location, temperature, humidity, water turbidity, detritus content, and other environmental parameters at the location. The objective is to implement the same using minimal equipment and lightweight protocols, to keep it low-cost.

1-c: Target Users

1) The region/institute administrators - will access the system to know if and when a location is prone to mosquito breeding. This is helpful for warning people.

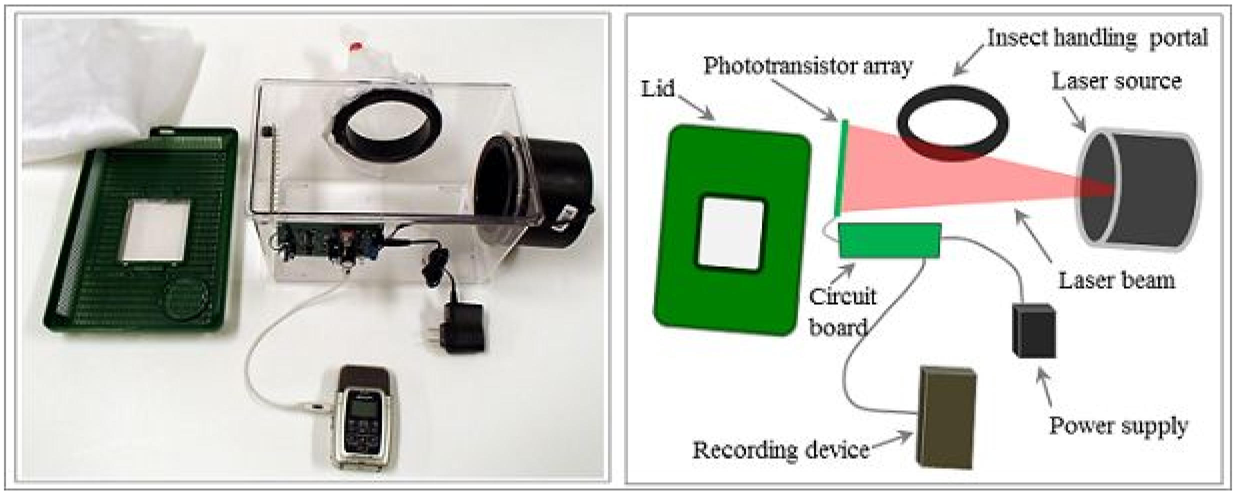
2) The authorities - responsible for disinfecting the area. The information of location, obtained by the admins, is forwarded to the authorities, who take necessary actions.

2 - Literature Survey

The literature survey of projects related to this domain of research is very interesting. We really enjoyed learning about novel ideas that budding researchers came up with. A few of them are briefly summarized below:

a) Capturing mosquitoes and analyzing wing-beat frequency

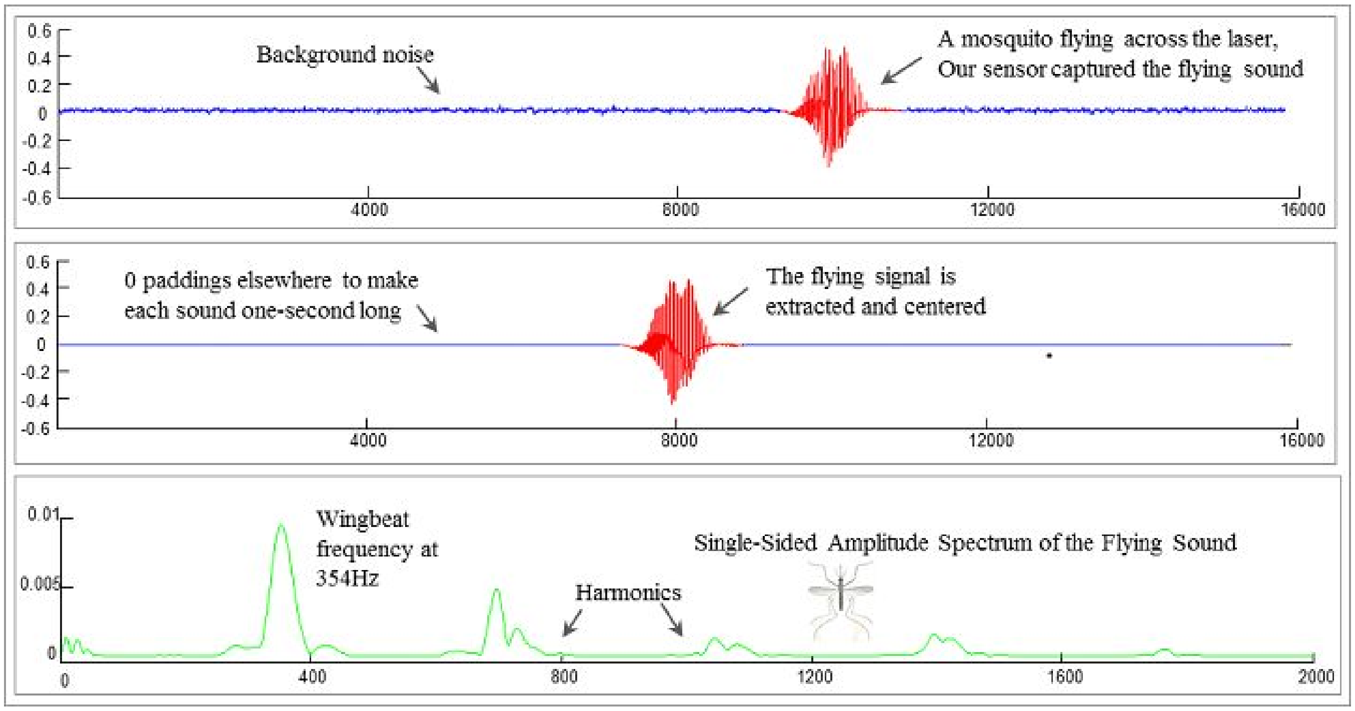
Apparatus:



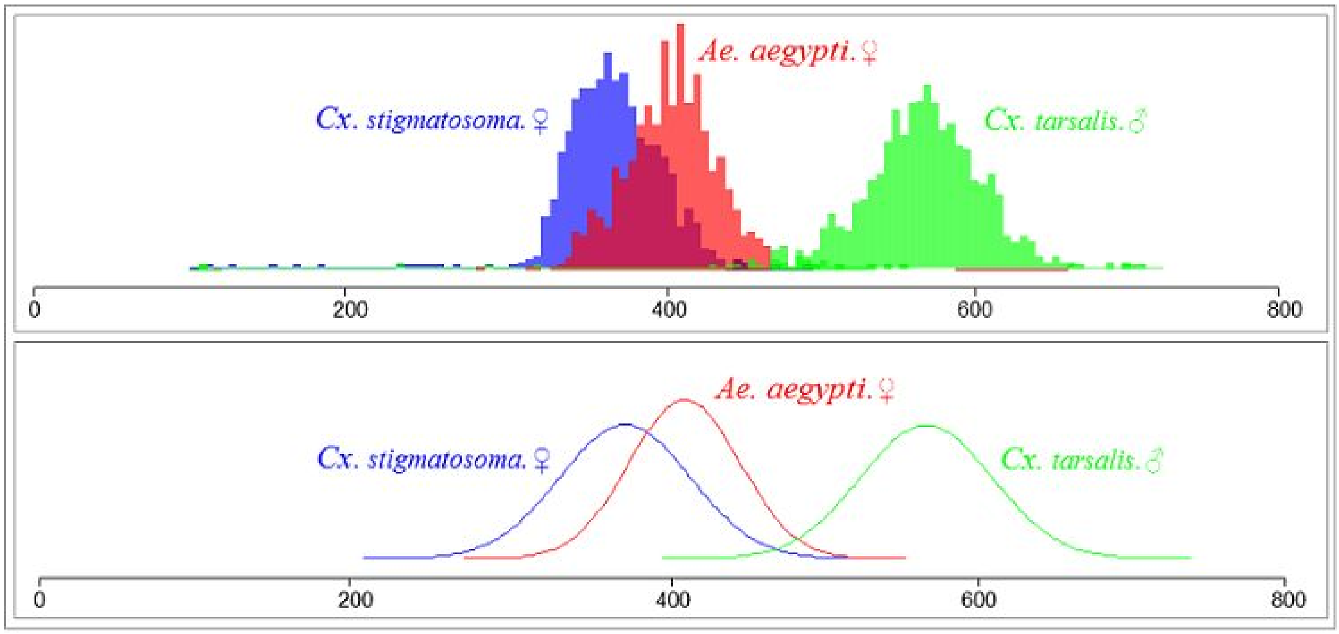
Basic Idea:
The basic idea is that every species of mosquito has a unique wing-beat frequency. Even in the same species, males and females have different wing-beat frequencies. Hence, applying Fourier Transform on these frequencies, we can find the dominant characteristic frequency for each species and gender, and the allowed deviation from this mean. Once we have this info, we can capture a mosquito and get its wing-beat frequency, and then apply classification algo on this frequency to find out the most probable species and gender. Thus finally, we will know if the species found is disease-causing or not, and suitable measures can be taken.

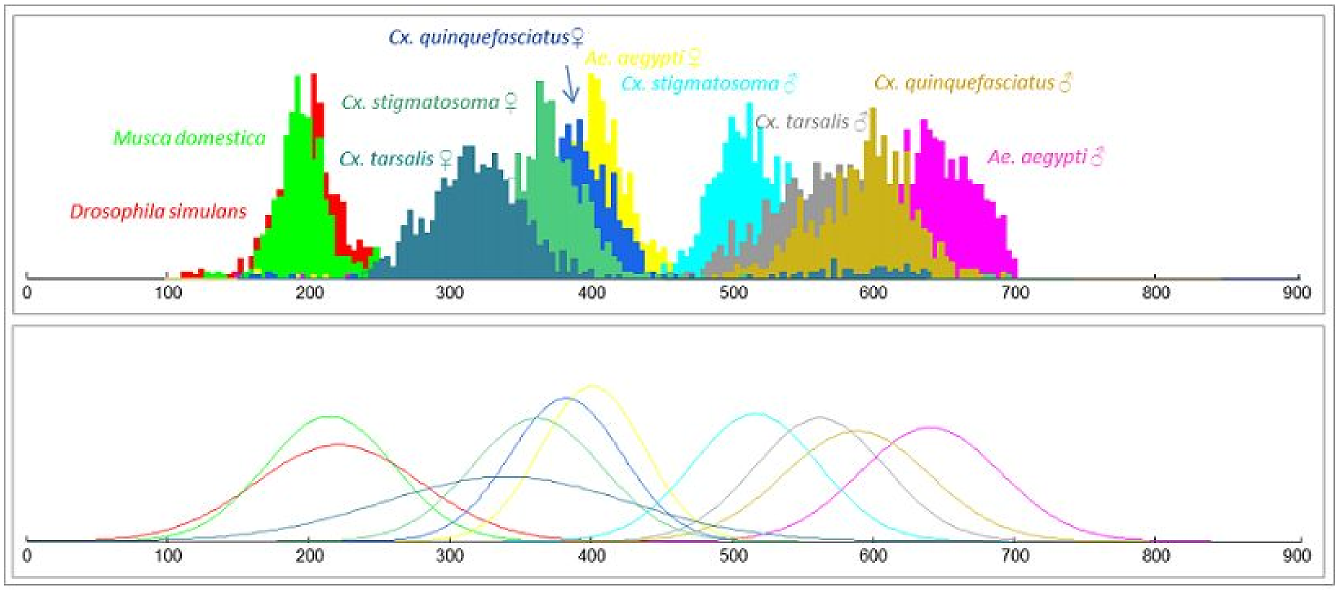
Working:
To prepare the project from scratch, first thousands of wing-beat frequency samples of each of the possible mosquito species, and within the species, the gender is obtained from medical sources. This data is fed into a supervised ML multi-class classifier model to train it and get the decision boundaries.

A sliding window slides through the sensor recording to detect flying insect sound. When an insect flying sound is detected in the sensor recording, the signal corresponding to the flying sound is first extracted from the original recording; then, a noise filter is applied to the signal to remove the background noise; finally, the signal is saved into a one-second long audio clip by centering the insect signal and padding with 0s elsewhere.



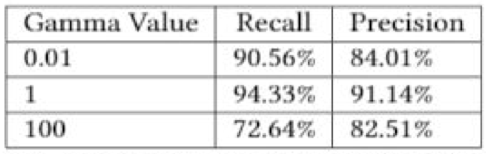
Applying DFT and filtering, we get the dominant frequency. We classify it to get the species. Example:


Ae. aegypti.♀
Cx. stigmatosoma.♀
Cx. tarsalis.♂
0
200
400
600
800


Cx. quinquefasciatus♀
Ae. aegypti ♀
Cx. stigmatosoma ♂
Cx. quinquefasciatus ♂
Cx. stigmatosoma ♀
Musca domestica
Cx. tarsalis ♂
Cx. tarsalis ♀
Ae. aegypti ♂
Drosophila simulans
0
100
200
300
400
500
600
700
800
900

b) An ML Approach for Identifying Mosquito Breeding Sites via Drone Images

This paper presents a novel approach for identifying mosquito breeding areas via drone images through the distinct coloration of those areas by applying the HOG algorithm. Using the HOG algorithm, they detect potential water retention areas using drone images. The female Aedes mosquitoes aim at the places with stagnant water for breeding. However, there can be some places with stagnant water that a man cannot easily reach or identify (i.e. roof gutters, water tanks, inaccessible rooftops and cement materials which are capable of retaining water). Due to the impact of retaining water for a long period of time, those areas are covered with lichens and algaes. According to the observations, these lichens and algaes of those places can be distinguished from the rest, due to their dark color. This color specification is utilized for the identification of mosquito breeding sites in unreachable places. The HOG algorithm is used to extract features from a test dataset, since it is one of the major object detection methods. This gives the positive and negative features.After that, three classifiers referring Support Vector Classification (SVC) were generated and trained, utilizing those positive and negative features under different gamma values (kernel ='rbf'), cost function (C) = 1). Then, those classifiers were analyzed by determining the capability of detecting the possible water retention areas.



Gamma Value	Recall	Precision
0.01	90.56%	84.01%
1	94.33%	91.14%
100	72.64%	82.51%

Table 1: Recall and Precision Values for Three Different Gamma Values for 100 Images

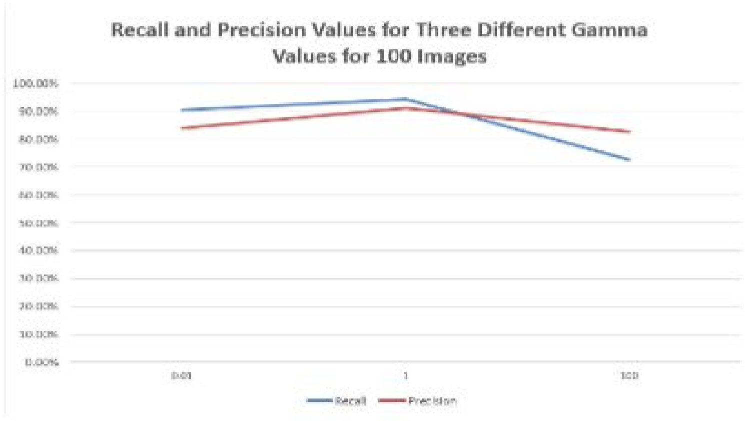


Figure 1: The Line Chart for Recall and Precision Values for Three Different Gamma Values

3 - Specific Challenges in our Project

1) The first challenge was to identify the most suitable breeding ground for mosquitoes, by doing some pre-processing on the region under consideration (the campus of DA-IICT). The rationale was an efficient use of resources, by minimizing the list of probable locations. Stagnant water is a primary condition for a location to be a breeding ground. Hence, the challenge is to identify the most stagnation-prone areas, using some techniques on the topological contour map of the region.

2) One of the main challenges was to make it a low-cost implementation. We can't have cameras, sensors, communication modules, and microcontroller boards all over the region under consideration, and still manage to keep a low cost. However, not using enough sensors would reduce the accuracy of the system. Hence, we needed to strike a balance in this trade-off.

3) Finding out the parameters - both environmental and non-environmental, to formulate the dependencies for an area to become prone to mosquito-breeding, and if certain environmental conditions have a correlation with specific species and gender of mosquitoes. It is generally known that conditions like broken tires, broken pipes, sewage water, marshy environment, water rich in detritus content, etc, are suitable for mosquito proliferation. However, the formulation of mathematical dependencies is still a very active research area.

4) Train appropriate ML models to extract features from images, followed by classification of the extracted features to see if the locations meet the primary criteria of stagnation or not. This requires some form of feature extraction (object detection) algorithm and a good classifier like Support Vector Machine. The models need to be chosen based on the kind and amount of data present.

5) Systematically combine results of 3 and 4 to check if the locations satisfy all criteria for the corresponding location to be labeled as a potential breeding ground, with a reasonably high degree of accuracy.

4 - Complete System Design Details

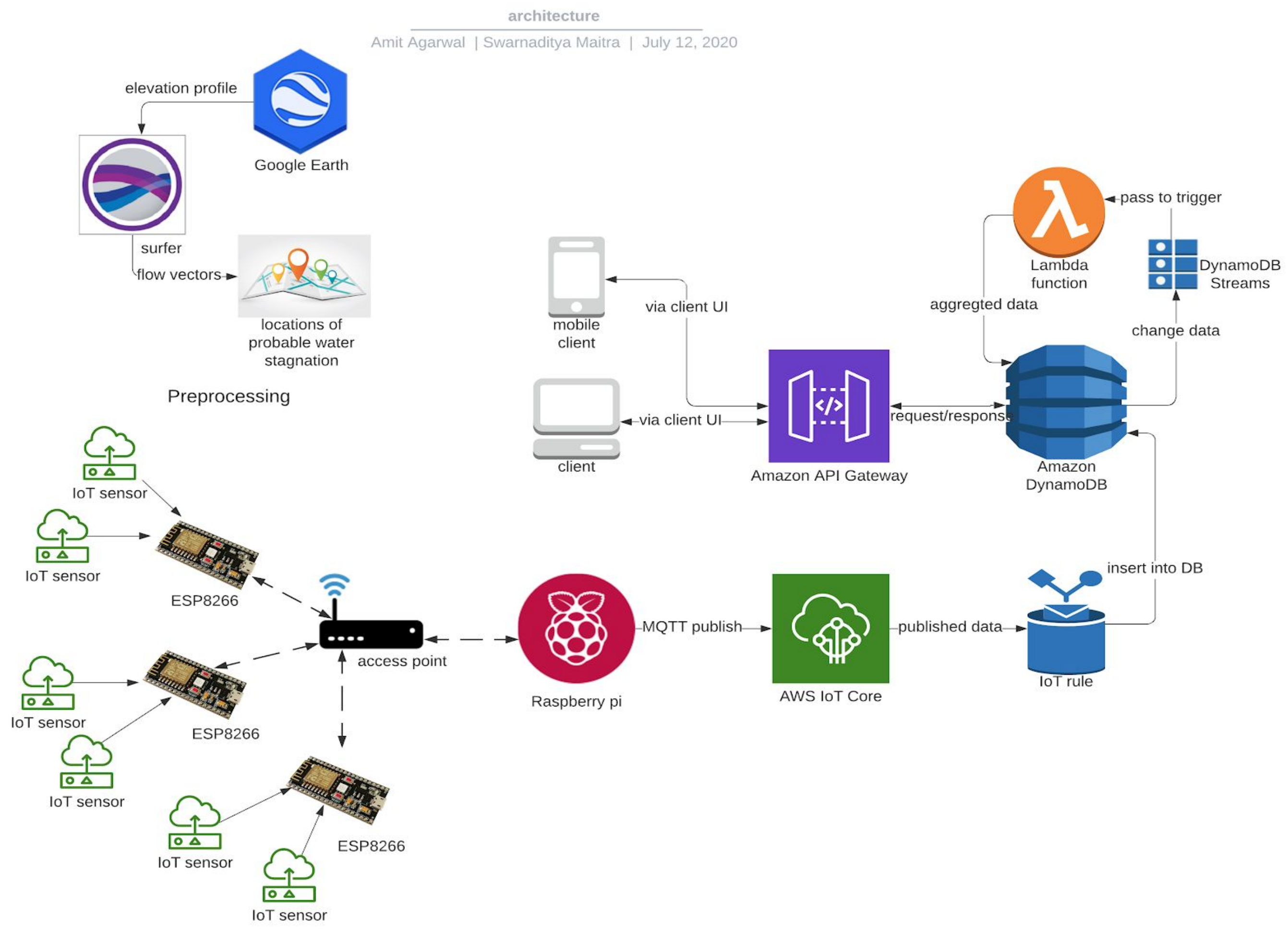
4-a: Components of the System

1) **Hardware**
    a) Camera
    b) LDR
    c) Temperature Sensor
    d) Humidity Sensor
    e) Turbidity Sensor
    f) Detritus Sensor
    g) ESP8266 Chips
    h) Raspberry Pi

2) **Software**
    a) Cloud Services (AWS)
        i) AWS Lambda
        ii) AWS IoT Core
        iii) AWS DynamoDB
        iv) Amazon API Gateway
    b) A Web-User Interface

4-b: System Architecture

1) Diagram



2) Sensors

At determined locations (discussed later), cameras are installed along with all the sensors. The data captured includes:

a) Images of the location. At the locations that are most prone to mosquito breeding to the natural topography of a tendency for water stagnation, cameras will be installed at these locations, which will capture the images periodically.

b) Temperature and Humidity: Sensor like DHT11.

c) Water Turbidity sensor.

d) LDR: Light-dependent Resistor to give us light intensity information.

The sensors are wirelessly connected to a raspberry pi using the ESP8266 chip. At each of the sensor sites, the sensors connect to a chip. The ESP8266 WiFi chip consists of a self-contained

SOC with integrated TCP/IP protocol stack that allows any microcontroller to access a WiFi network. The sensors are connected to a 5V supply either from raspberry pi or an external source. External sources being more preferable.

3) Processing Unit

The processing consists of Raspberry pi which is a processor on-chip. It acts as an intermediate between the sensors and the cloud. All the sensors are wirelessly connected to the processing unit. A single raspberry pi unit comprises 26 GPIO pins, implying that 26 different sensors can be connected to it. This number, however, can be increased by attaching a multiplexer (MUX) to it. It is essential that the ground of raspberry pi and sensors must be connected in order to transfer data using the GPIO pins. There is a python script running on the chip that checks the status of various GPIO pins and updates this information onto the cloud. Since both ESP8266 and the Raspberry Pi chips are connected to a Wifi network, data collected from various sensors is sent to the raspberry pi through the esp8266 chip. The raspberry pi then transmits this data to the MQTT Server through MQTT protocol over a channel. MQTT (Message Queue Telemetry Transport) Protocol is a publish-subscribe based "lightweight" messaging protocol that unlike CoAP, is used on top of the TCP/IP protocol. It is designed to establish connections across remote locations where a limited amount of data needs to be transferred or in cases of low bandwidth availability, making it ideal for this implementation.

4) Cloud-Services (AWS)

The MQTT server is hosted on AWS IoT Core. Amazon's DynamoDB acts as a database to store all the sensor information that is published into various topics for clients and end-users that have access to the system. Every connected Raspberry Pie can publish messages from sensors to AWS IoT Core. It is due to the flexible nature of cloud which permits the system to add any number of raspberry pi's at any time of the day. Continuous backup is made of the data stored on the cloud in order to ensure easy and quick recovery of data in case of any kind of system failure.

In addition, we use AWS lambda to run algorithms on streams of data and implement the inferencing engine. We use relevant REST APIs, and integrate them to a Web User interface, where the users of the system can see the processed data, the inferences, and predictions in a very informative form.

5) **Web Application - User Interface**

The Web-application for the users of the system has been made using:

a) HTML
b) CSS - Bootstrap, Flexboxes, and Grid Containers
c) JavaScript - jQuery

The application consists of:

a) A Generic Page showing the total number of registered regions and the average danger level (risk percentage of being a mosquito breeding zone) of the respective regions.
b) Pages corresponding to each registered region, which contains the live list of locations that currently have the IoT sensor sets and ESP8266 Chips installed and running.
c) Specific pages that open on clicking each of the above options. The users are redirected to a web-page corresponding to the location of that IoT sensor set, where corresponding to that location, one can see:
    i) The danger level of that location
    ii) The Temperature, Humidity, Water turbidity, Detritus content variation at that location with time
d) Notifications on each page will notify the users in the case of an anomalous event.

4-c: Brief Chronology of System Modeling

1) Generate the contour plot of the region (campus).
2) Find out the end-points (sinks) of the flow vectors.
3) At the sink locations, install a camera, the sensors, and connect them to an ESP8266 chip. This chip connects to a Wifi network and transmits data to a Raspberry Pi. Using the MQTT protocol, the Raspberry Pi publishes all collected data to the MQTT Server. Data is stored in AWS DynamoDB, with triggers programmed for insertion, updation, and deletion.
4) Apply feature extraction (dimensionality reduction) algorithm on images. Then feed the data to a trained classifier that would declare the location, as having water stagnation or not. If yes, then step 5 or else stop.
5) Check environmental parameters for thresholds, ranges, and probabilistic results. AWS Lambda functions are implemented. When appropriate conditions are satisfied, it means mosquito-breeding prone areas. The appropriate notification is sent to the UI, through Amazon API Gateway.

4-d: Methods and Algorithms Used

Vectors like mosquitoes spawn under specific environmental conditions like marshy-swampy areas, areas with stagnated water, scanty sunlight, relatively high water turbidity, sewage water (because it has nutrients necessary for larvae), etc. While environmental parameters act as a catalyst, stagnant water is the 1st primary step for breeding, because mosquitoes can't lay eggs without it. We solve the problem using the following methods and algorithms:

1) Mosquito infestation booms in/immediately after the precipitation season, commencing from stagnant water areas.

   a) We obtain the elevation profile of the campus, called DEM (Digital Elevation Model) using Google Earth.

   b) We generate a contour map of the region along with flow vectors, by applying certain methods on the elevation profile. Flow vector tells us the direction where water would flow, should rainfall occur at that location.

   c) Join together flow vectors to get the streams that form. When rain falls, the water forms streams and then flows. We can find these streams using the above method.

   d) Find the end-points (sinks) of these streams. These are the locations where water is expected to accumulate (stagnate) and serve as ideal breeding grounds.

   e) After getting the locations of potential breeding grounds, we periodically take images of those locations. These images will be transmitted via networking modules to the cloud. This info will be used in step-4.

   f) This method helps in narrowing down the number of macroscopic breeding locations when the total region under consideration is substantially large.

2) We measure environmental parameters like temperature, humidity, water turbidity, etc using sensors.

   a) According to an article published in the health science journal, the optimal breeding temperature for mosquitoes is in the range 28-30 °C. However, in a broad sense temperatures in the range of **24 - 36°C** support the breeding of mosquitoes(more precisely the larval growth).

b) Modest correlation with water turbidity was also found(in the range **451-550 ppm**).

c) **Light/dark cycles and humidity** are two other factors that influence the egg hatching process and not the egg-laying process directly. Relatively higher humidity conditions along with alternating light and dark cycles provide more chance of a successful egg hatch according to (3) and (4). These are important and often necessary conditions for mosquito breeding. This information is also transmitted via the networking modules to the cloud.

d) We model the parameters like temperature, humidity using **sigmoid function,** and the detritus content (explained in next point) **as logarithmic function.**

3) For each container like broken pipe, tires, dumpsters, we can measure a suite of environmental factors in three broad categories that have been seen to affect container mosquito populations :

a) Microorganisms (protozoan richness, protozoan total abundance, bacterial biomass production, algae).

b) Detritus and nutrients, namely:

i) Leaves, twigs, and seeds

ii) Animal detritus

iii) Fine Detritus

iv) Conductivity

(1) As elaborated in the 'Important Background Information', we only consider **leaf, seeds, and fine detritus.** The relationship between detritus (in grams) and mosquito abundance (in absolute numbers) turns out to be of the form of $\mathbf{y = k(\log x)+c}$, as can be seen in the important background information section.

c) Factors related to habitat size or the terrestrial matrix (human density, canopy cover, tire size, tire volume). For the simplicity of the project, we don't consider this factor set.

4) We build an ML model, to classify the site in an image as having water stagnation, and hence having mosquito breeding potential OR not.

a) HOG (Histogram of Oriented Gradients)

i) The first step is to process the images to an appropriate form. To speed up the training and detection, it is necessary to extract only the essential features from the image, without losing the important details. A simple dimensionality reduction algorithm like PCA won't be appropriate. Research shows that for detection of water-retention areas (stagnant water zones), HOG feature extraction algorithm is most suitable.

ii) The **histogram of oriented gradients (HOG)** is a feature descriptor used in computer vision and image processing, that counts occurrences of gradient orientation in localized portions of an image. The basic idea behind the HOG descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms, hence the name - Histogram of Oriented Gradients. This is really useful for water images, because for the images of watery objects, the gradients of the borders with the surroundings are pretty distinct. Hence, rather than reduction by dimensionality reduction, we reduce the size by taking the essential parts from the image that are most suited to the problem at hand.

b) Support Vector Machine (SVM)

i) For the problem of accurate classification of an image as having or not having water stagnation, we use the famous binary classifier - support vector machine. We train the SVM model using feature vectors obtained from the HOG feature extraction algorithm, to increase training speed without compromising accuracy. After training, whenever a feature vector is fed to the SVM classifier, it would classify the site as being stagnated or not. We test for different gamma values for better accuracy. After training on a sufficiently large dataset, we test for the captured images from campus.

5) For the campus locations that are predicted by our ML model to have water stagnation and hence be potential breeding sites for mosquitoes, we test whether the sensed environmental parameters are also conducive to breeding. The data that is being published by MQTT clients (sensors) to the MQTT server hosted using **AWS IoT Core**, is being stored in **AWS DynamoDB** using **rules** defined within AWS IoT Core**.** We use **Lambda Functions** to process any inserted data into the database using **DynamoDB Triggers**. The various thresholds are obtained from domain knowledge. For temperature, turbidity, we use a **sigmoid function with covariance(taken from studies conducted earlier) as the coefficients of temperature and turbidity**, for humidity we have used a simple **linear relationship**, and for the detritus content, we use the **logarithmic relation**. The probability is calculated as a weighted mean from all these functions. When a data record passes all the required criteria for being classified as a mosquito breeding site, it's probability is high. The processed data from lambda functions is again stored in the database. With the help of **Amazon Gateway API,** the data is made available to endpoints from where the client UI can access it. Notifications about these events are triggered that can be seen by the users of the system on the Web User Interface.

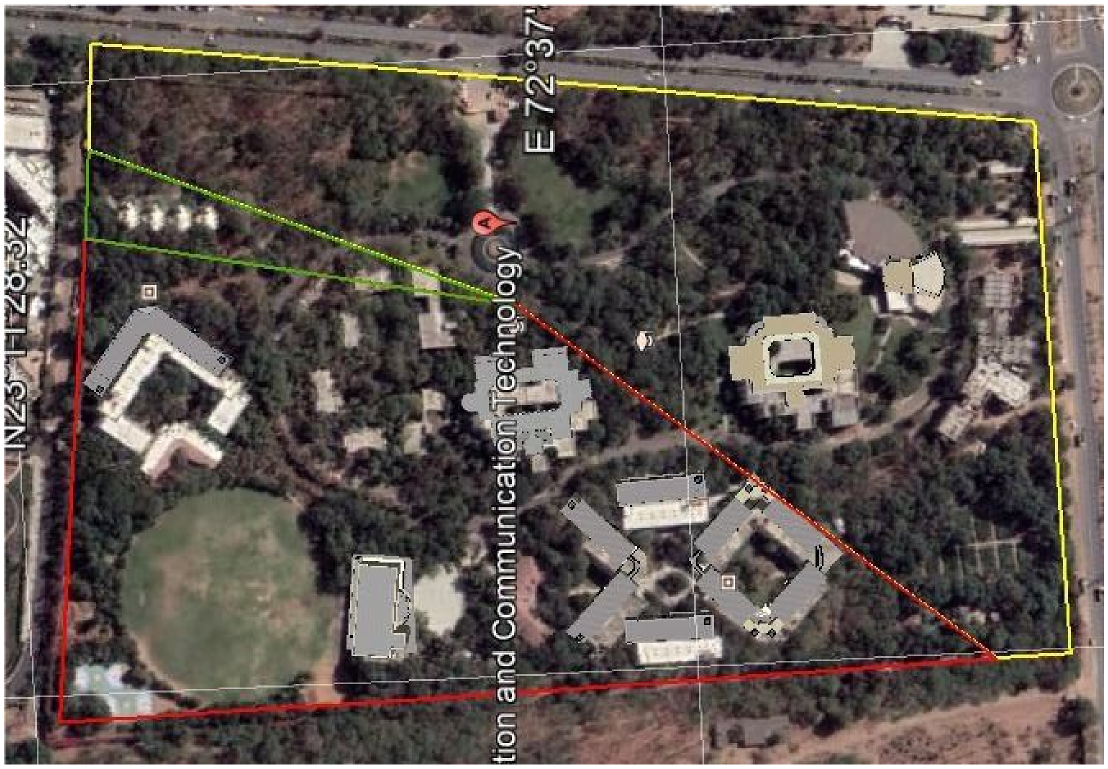
5 - Detailed Experiment Working and Analysis

5-a: Pre-processing

1) For large regions, it is necessary to identify the major macroscopic regions, where water is expected to stagnate. This is necessary, since it is not economical and also very inefficient to randomly go on placing sensors all through the region. This preprocessing to find the most appropriate locations for water stagnation and breeding, is a one-time activity.

2) We have implemented a novel approach to predict the locations on DA-IICT campus, where the water may stagnate.

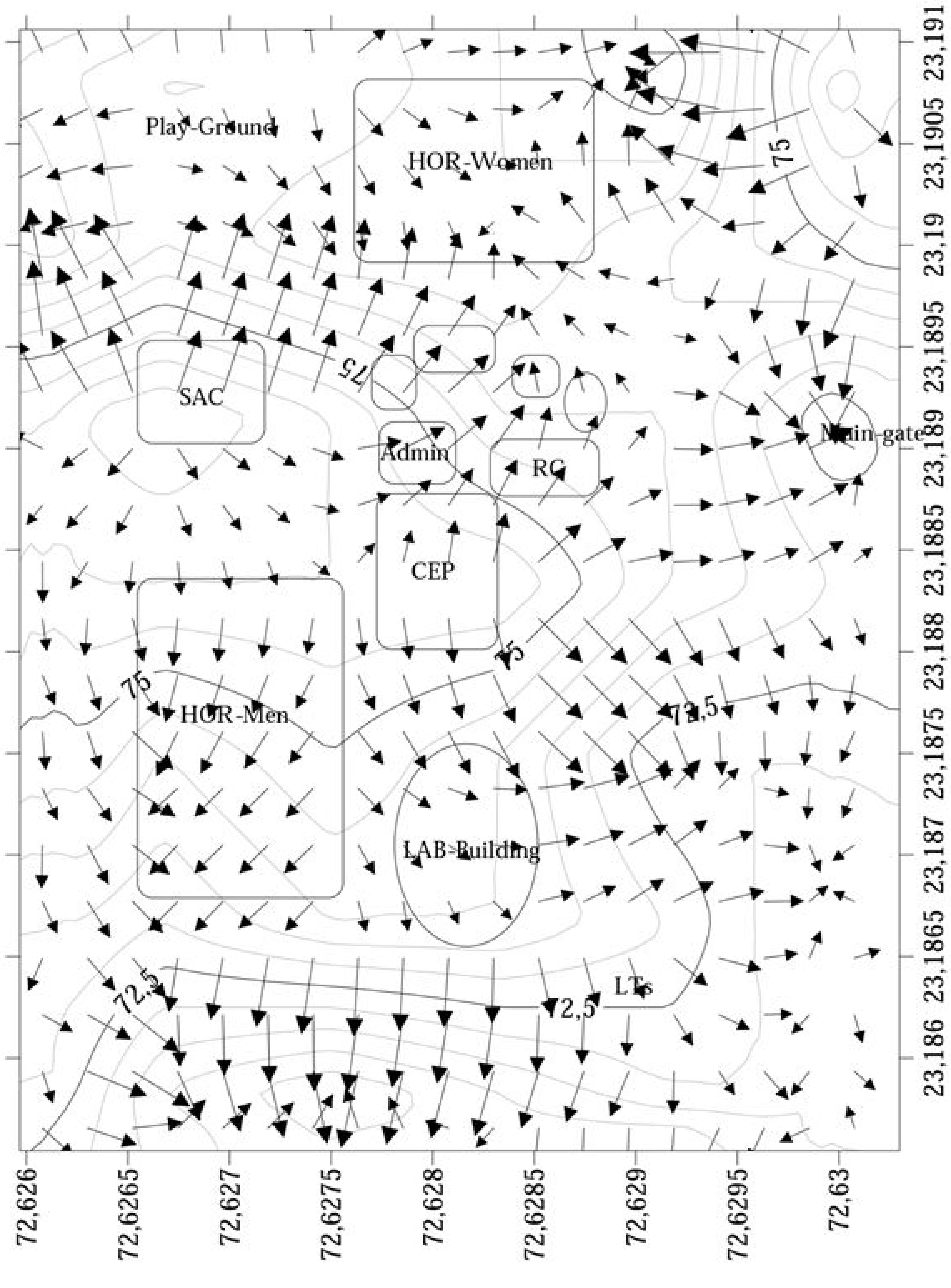
3) Below is a map of DA-IICT obtained from Google Earth.



4) Obtain elevation profile of the region from google earth and save it as .kml file format.

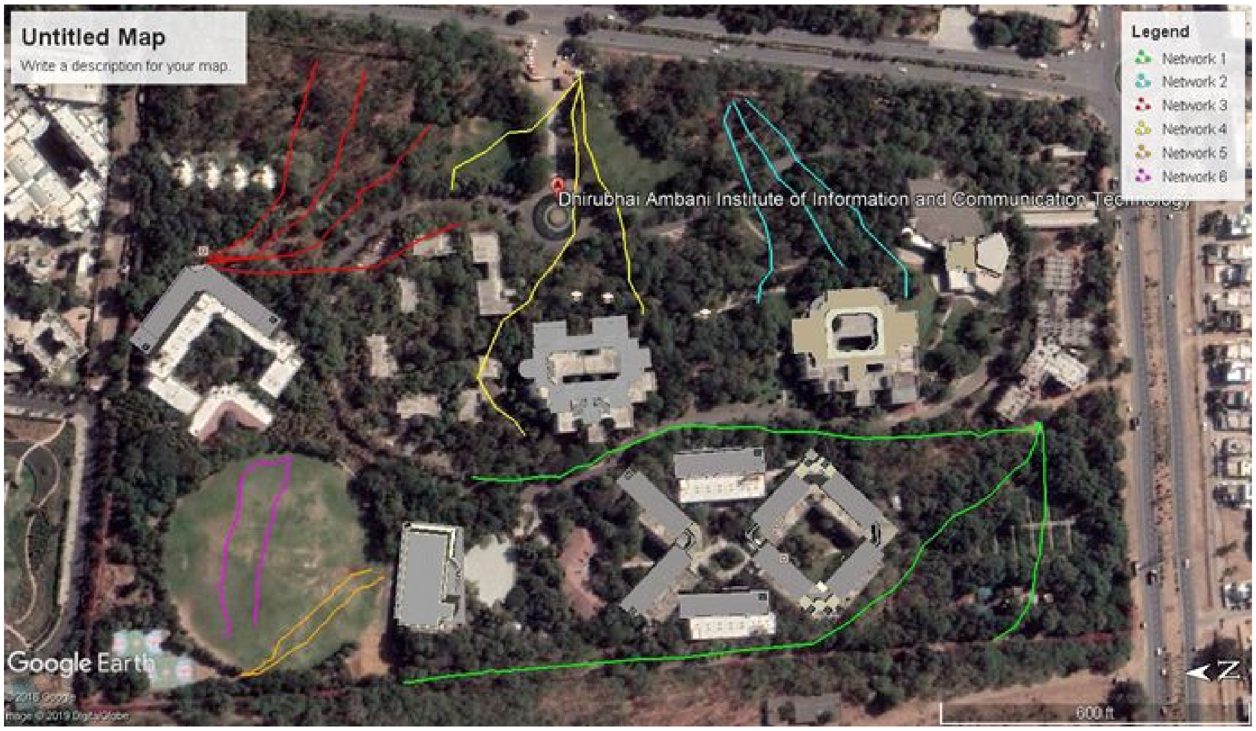
5) Open Surfer software and import the .kml file, and generate contour map, with flow vectors viewmodel. For a simple example, just follow this link.
https://www.youtube.com/watch?v=j5vhxcTBArQ&feature=youtu.be

The resultant image looks like the one shown below:



DA-IICT Campus Contour Plot with Flow Vectors

6) The flow vectors are joined to find the streams and the endpoints of these streams. These are the sinks where water would flow to. Doing for DA-IICT campus, we obtain the following:


Untitled Map
Write a description for your map.
Legend
Network 1
Network 2
Network 3
Network 4
Network 5
Network 6
Dhirubhai Ambani Institute of Information and Communication Technology
Google Earth
600 ft

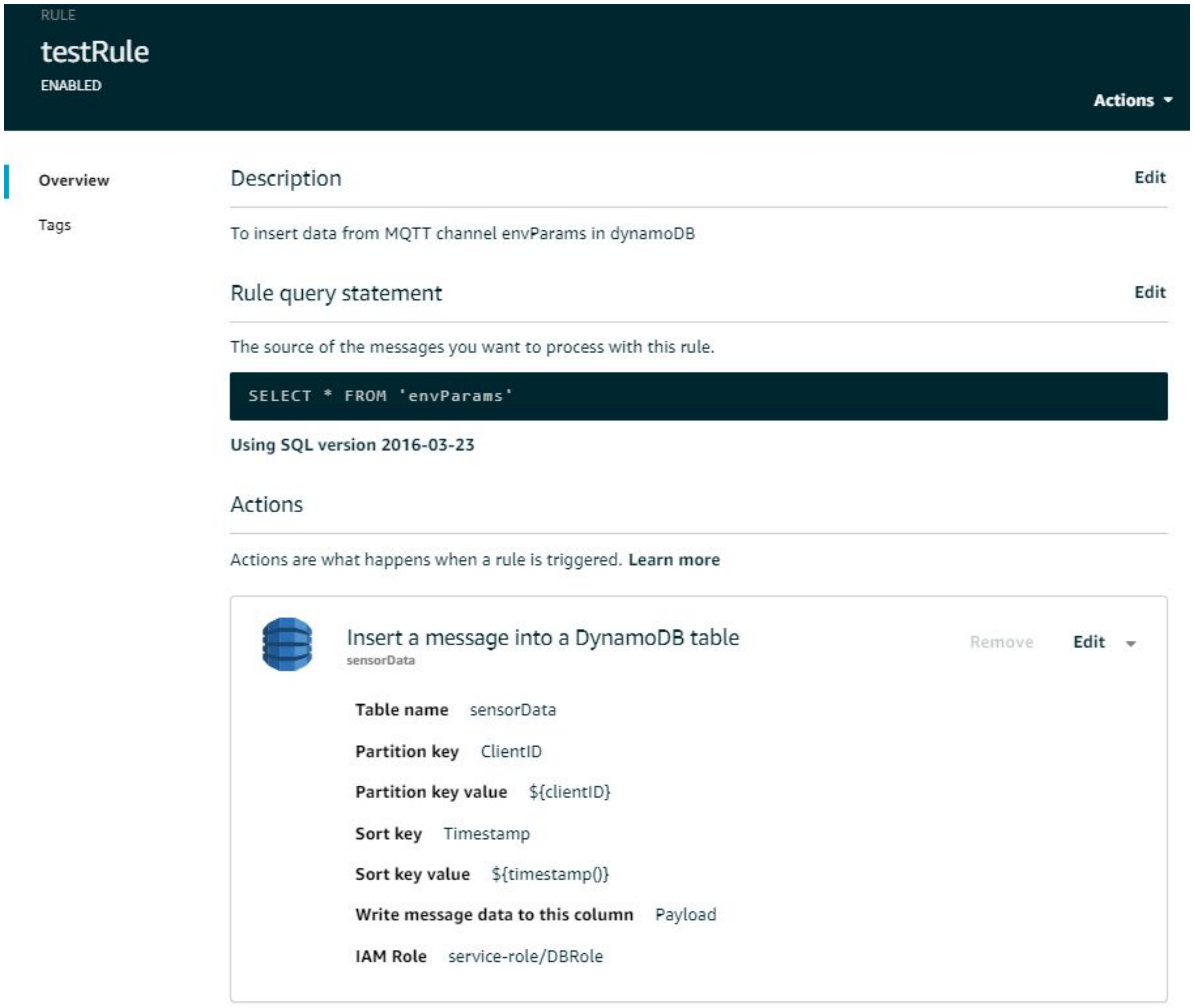
The flow endpoints. The sinks of flows - points of water stagnation

5-b: Backend - AWS

1) AWS IoT Core

Sensor data is published to channels using **MQTT protocol**. AWS IoT Core provides a means to host an MQTT server and configure permissions and rules for various devices that can publish/subscribe to it. Each device has to be configured with the private key, device certificates generated from AWS IoT core.

AWS IoT Rules can be defined which process the data published to a given channel and insert it into **DynamoDB**. Given below is the configuration of such IoT rule:



testRule
ENABLED

Description

To insert data from MQTT channel envParams in dynamoDB

Rule query statement

The source of the messages you want to process with this rule.

```
SELECT * FROM 'envParams'
```

Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. Learn more

Insert a message into a DynamoDB table
sensorData

Table name	sensorData
Partition key	ClientID
Partition key value	${clientID}
Sort key	Timestamp
Sort key value	${timestamp()}
Write message data to this column	Payload
IAM Role	service-role/DBRole

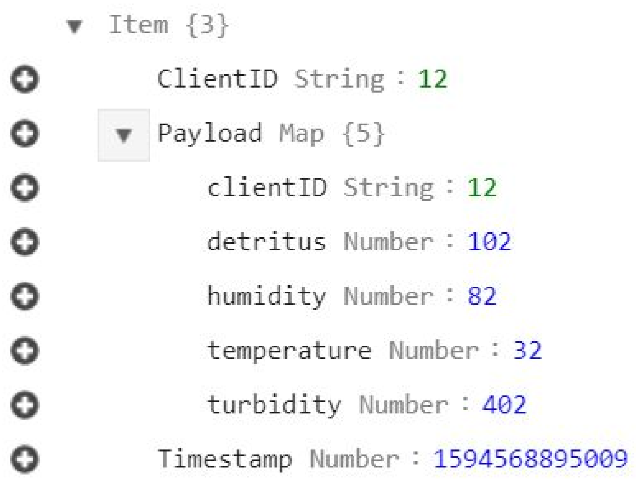
An MQTT message published on to the channel looks as follows:

```
{
"clientID": "1",
"temperature":31,
"turbidity":450,
"detritus": 100,
"humidity":85
}
```

Timestamp is automatically added by the IOT rule before entering into Database.

2) **AWS DynamoDB**

DynamoDB is amazon's fully managed NoSQL service. Data published onto the MQTT channel is stored into DynamoDB. This data can be used to run analytics and also to provide it directly to the client. We store the sensor data in table 'sensorData' as follows:



```
▼ Item {3}
    ClientID String : 12
  ▼ Payload Map {5}
      clientID String : 12
      detritus Number : 102
      humidity Number : 82
      temperature Number : 32
      turbidity Number : 402
    Timestamp Number : 1594568895009
```

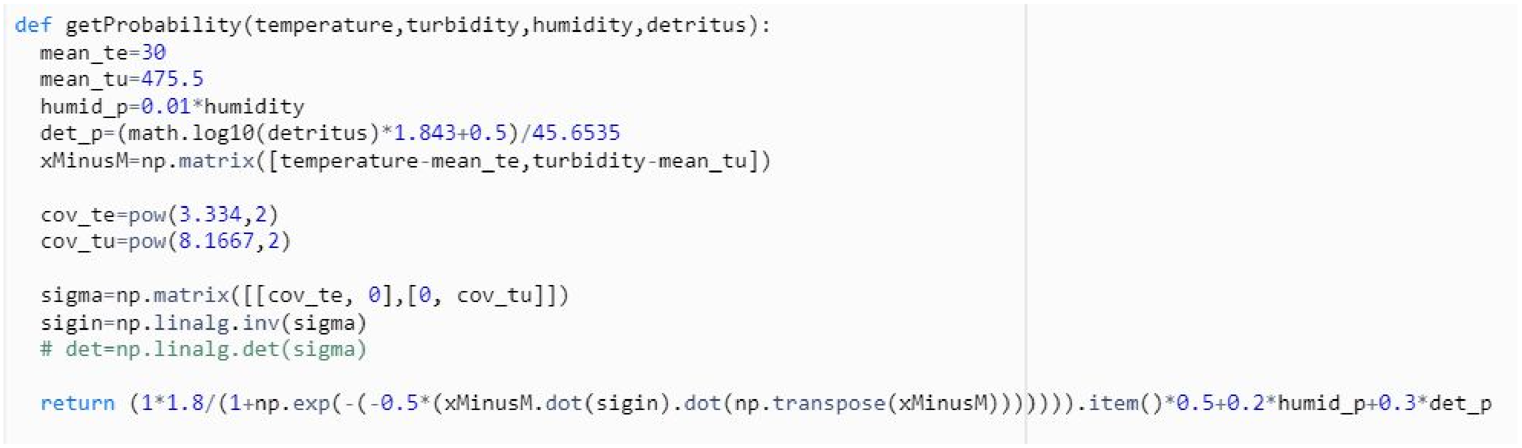
A. **DynamoDB Streams:** Tables in DynamoDB can be configured to produce streams of changed data upon insertion, modification and deletion. The streams can be picked up by any service to perform actions accordingly. These streams are guaranteed to be in order.
B. **DynamoDB Triggers**: AWS Lambda functions can be defined as triggers to a DynamoDB table which can make use of the data streams generated.

3) AWS Lambda

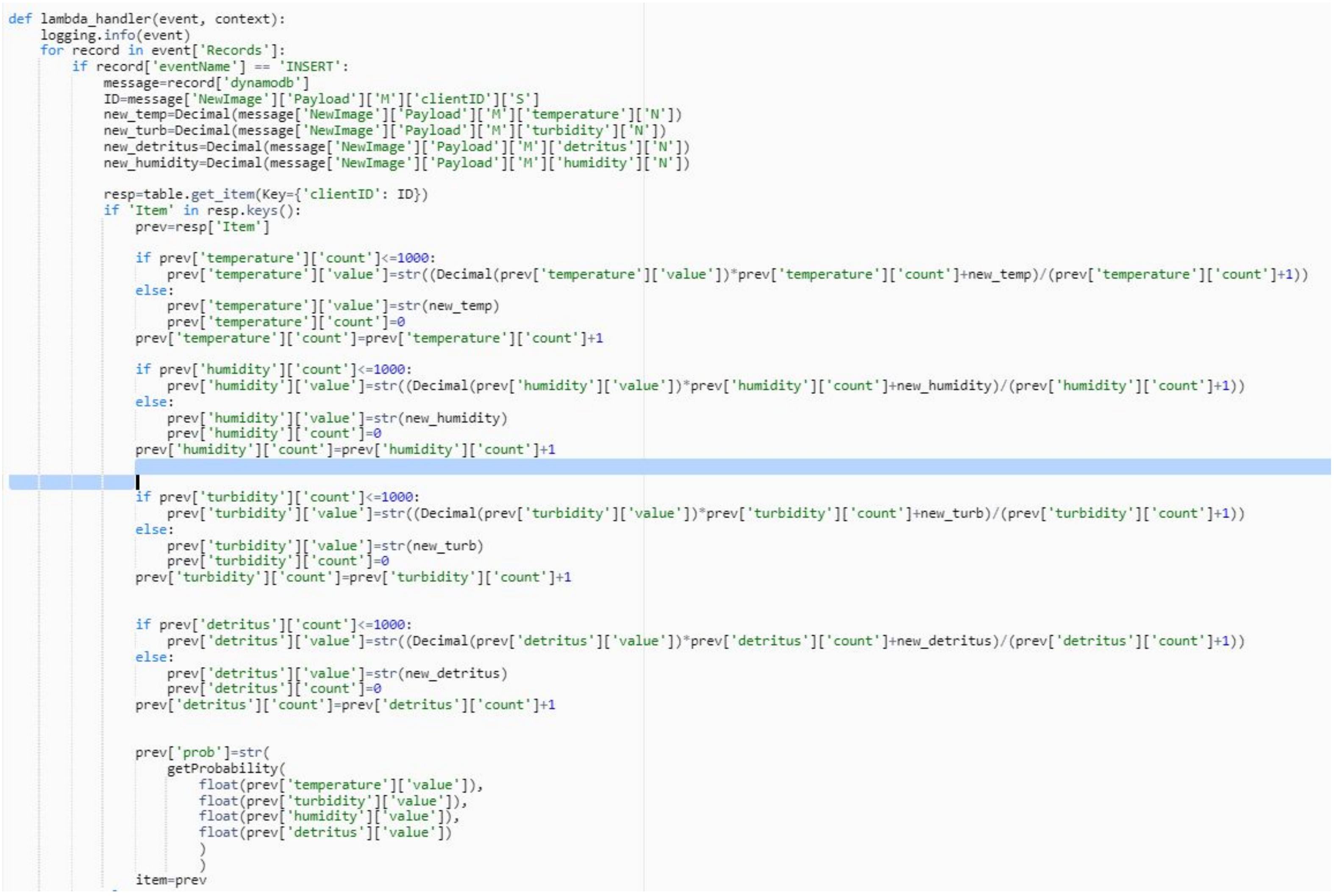
AWS Lambda provides means to run code on the cloud without the hassle of configuring the environment and not really worrying about the hardware required to do so. We are using AWS Lambda function as a DynamoDB trigger to aggregate the sensor data stored in the 'sensorData' table. The aggregate data is again stored in another table called 'summary' within the DynamoDB instance. Given below is the **pseudocode** for the lambda function being used:

```
1. For all the records in the data stream:
    a. If the record is generated due to insertion in the
       database
         i.  Extract environmental parameters from the newly
             inserted data in the record.
        ii.  Extract clientID from the data record
       iii.  For the obtained clientID, get the aggregated
             data from the Database (summary table)
        iv.  If aggregated data is present for that
             clientID:
               1. Compute moving average of temperature,
                  humidity, turbidity, detritus.
               2. Compute probability of mosquito breeding
                  for the new average temperature, humidity,
                  turbidity, detritus
               3. Update the average values of the
                  parameters and the probability in the
                  database.
         v.  Else:
               1. Compute probability of mosquito breeding
                  using parameters obtained in step (i)
               2. Add new entry in the summary table for the
                  given clientID and set the values for
                  temperature, humidity, detritus, turbidity
                  and the probability
```

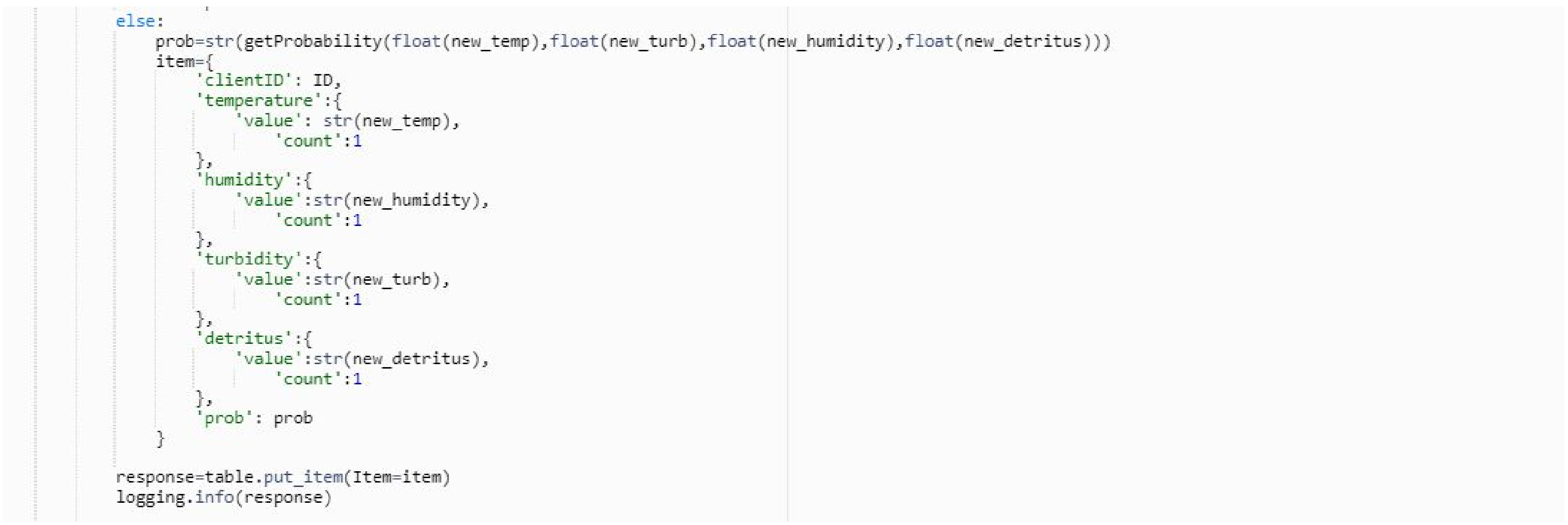
Some python script - code snippets for the above pseudocode are as follows:



```python
def getProbability(temperature,turbidity,humidity,detritus):
  mean_te=30
  mean_tu=475.5
  humid_p=0.01*humidity
  det_p=(math.log10(detritus)*1.843+0.5)/45.6535
  xMinusM=np.matrix([temperature-mean_te,turbidity-mean_tu])

  cov_te=pow(3.334,2)
  cov_tu=pow(8.1667,2)

  sigma=np.matrix([[cov_te, 0],[0, cov_tu]])
  sigin=np.linalg.inv(sigma)
  # det=np.linalg.det(sigma)

  return (1*1.8/(1+np.exp(-(-0.5*(xMinusM.dot(sigin).dot(np.transpose(xMinusM))))))).item()*0.5+0.2*humid_p+0.3*det_p
```

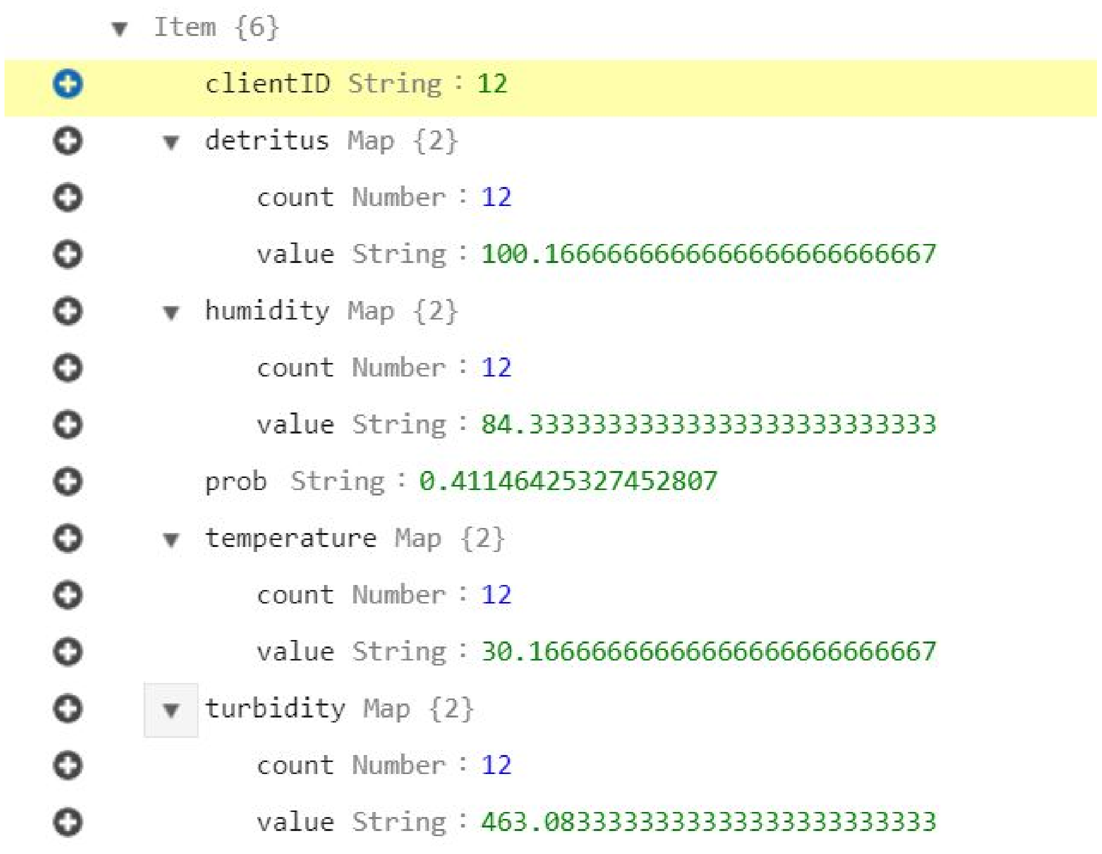


```python
def lambda_handler(event, context):
    logging.info(event)
    for record in event['Records']:
        if record['eventName'] == 'INSERT':
            message=record['dynamodb']
            ID=message['NewImage']['Payload']['M']['clientID']['S']
            new_temp=Decimal(message['NewImage']['Payload']['M']['temperature']['N'])
            new_turb=Decimal(message['NewImage']['Payload']['M']['turbidity']['N'])
            new_detritus=Decimal(message['NewImage']['Payload']['M']['detritus']['N'])
            new_humidity=Decimal(message['NewImage']['Payload']['M']['humidity']['N'])

            resp=table.get_item(Key={'clientID': ID})
            if 'Item' in resp.keys():
                prev=resp['Item']

                if prev['temperature']['count']<=1000:
                    prev['temperature']['value']=str((Decimal(prev['temperature']['value'])*prev['temperature']['count']+new_temp)/(prev['temperature']['count']+1))
                else:
                    prev['temperature']['value']=str(new_temp)
                    prev['temperature']['count']=0
                prev['temperature']['count']=prev['temperature']['count']+1

                if prev['humidity']['count']<=1000:
                    prev['humidity']['value']=str((Decimal(prev['humidity']['value'])*prev['humidity']['count']+new_humidity)/(prev['humidity']['count']+1))
                else:
                    prev['humidity']['value']=str(new_humidity)
                    prev['humidity']['count']=0
                prev['humidity']['count']=prev['humidity']['count']+1


                if prev['turbidity']['count']<=1000:
                    prev['turbidity']['value']=str((Decimal(prev['turbidity']['value'])*prev['turbidity']['count']+new_turb)/(prev['turbidity']['count']+1))
                else:
                    prev['turbidity']['value']=str(new_turb)
                    prev['turbidity']['count']=0
                prev['turbidity']['count']=prev['turbidity']['count']+1


                if prev['detritus']['count']<=1000:
                    prev['detritus']['value']=str((Decimal(prev['detritus']['value'])*prev['detritus']['count']+new_detritus)/(prev['detritus']['count']+1))
                else:
                    prev['detritus']['value']=str(new_detritus)
                    prev['detritus']['count']=0
                prev['detritus']['count']=prev['detritus']['count']+1


                prev['prob']=str(
                    getProbability(
                        float(prev['temperature']['value']),
                        float(prev['turbidity']['value']),
                        float(prev['humidity']['value']),
                        float(prev['detritus']['value'])
                        )
                        )
                item=prev
```



```
else:
    prob=str(getProbability(float(new_temp),float(new_turb),float(new_humidity),float(new_detritus)))
    item={
        'clientID': ID,
        'temperature':{
            'value': str(new_temp),
                'count':1
        },
        'humidity':{
            'value':str(new_humidity),
                'count':1
        },
        'turbidity':{
            'value':str(new_turb),
                'count':1
        },
        'detritus':{
            'value':str(new_detritus),
                'count':1
        },
        'prob': prob
    }

response=table.put_item(Item=item)
logging.info(response)
```

The data in the 'summary' table is as follows:



```
▼ Item {6}
    clientID String : 12
  ▼ detritus Map {2}
      count Number : 12
      value String : 100.16666666666666666666666667
  ▼ humidity Map {2}
      count Number : 12
      value String : 84.333333333333333333333333333
    prob String : 0.41146425327452807
  ▼ temperature Map {2}
      count Number : 12
      value String : 30.166666666666666666666666667
  ▼ turbidity Map {2}
      count Number : 12
      value String : 463.08333333333333333333333333
```

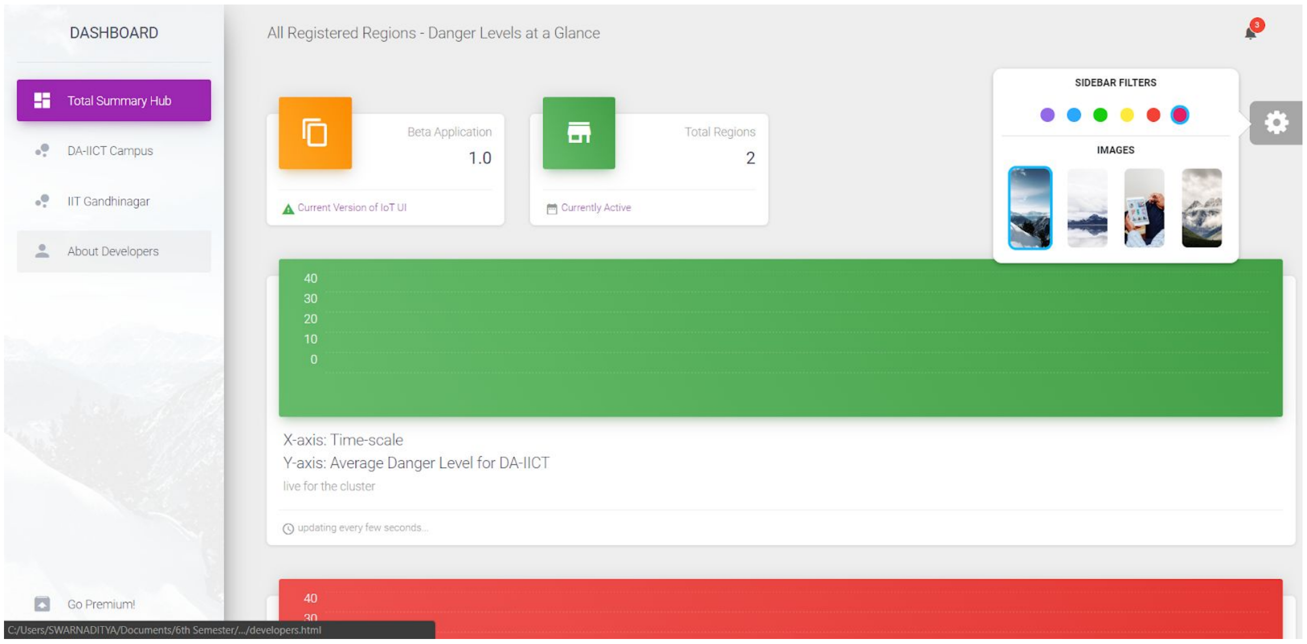
4) Amazon API Gateway:

It is a fully managed service to build, deploy, monitor **REST APIs.** We utilise this service as a proxy for DynamoDB. This is done so that any client application can make use of this service to get data from DynamoDB instances indirectly. In this case, the sensor data such as temperature, turbidity, humidity etc. along with the aggregated data so that the consumer can easily monitor the breeding site environmental parameters and the analytics on top operating on the data, using the UI developed. The service **URIs** for different resources are as follows:

5-c: FrontEnd UI - Web Application

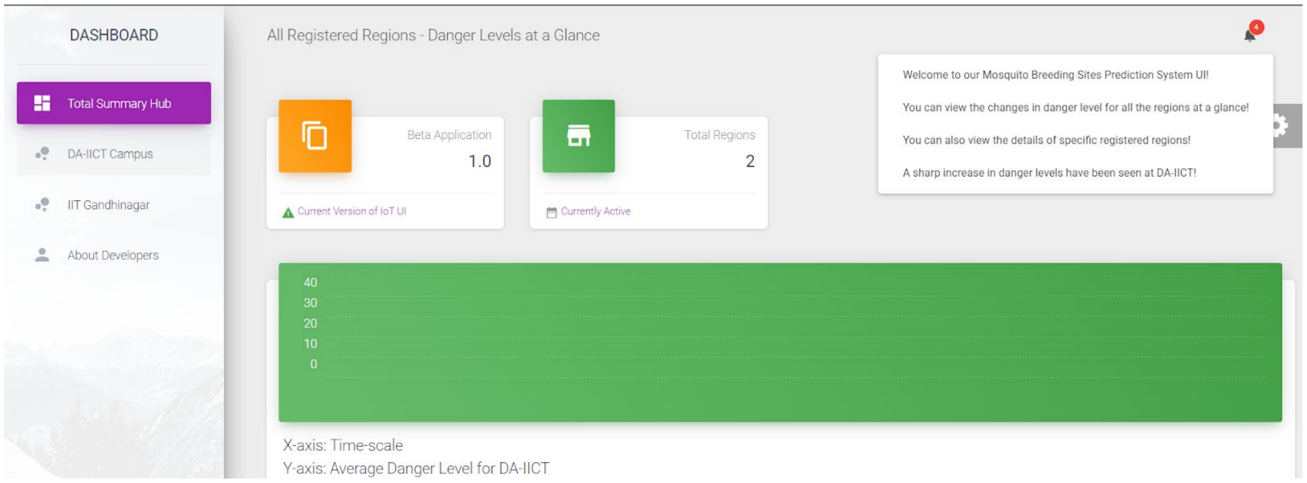
The Web Application is meant to display the measurements, summary, and predictions for the regions and corresponding sites registered in the system. It consists of:

1) **The main dashboard page:** shows the total number of registered regions and the variation in average danger levels (risk percentage of becoming a breeding ground) of each, as graphs.



X-axis: Time-scale
Y-axis: Average Danger Level for DA-IICT

2) All the registered regions will appear on the dashboard. The sidebar filters can be used to modify the look of the page. The bell icon on the top right is for notifications. The expected behaviour is that, in case some anomaly is detected at a region, it would immediately show up in notifications.



Welcome to our Mosquito Breeding Sites Prediction System UI!

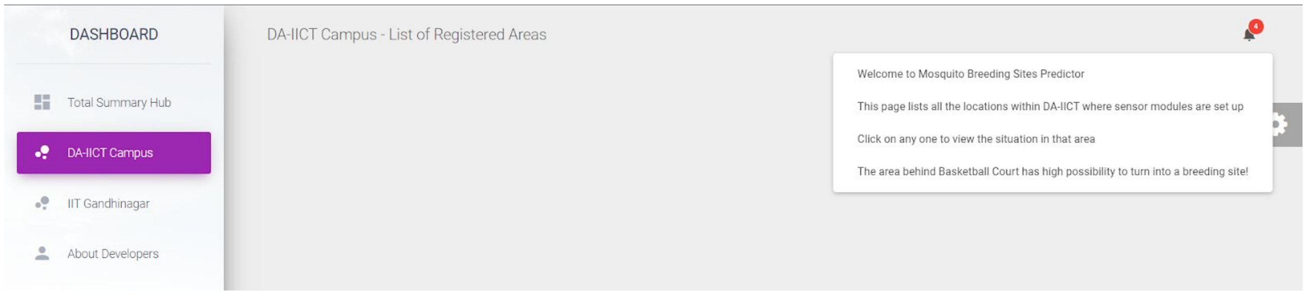
You can view the changes in danger level for all the regions at a glance!

You can also view the details of specific registered regions!

A sharp increase in danger levels have been seen at DA-IICT!

X-axis: Time-scale
Y-axis: Average Danger Level for DA-IICT

3) Clicking on a tab, eg: DA-IICT Campus, will redirect to a new page, where a list of all the locations where IoT sensors are installed and running live, will show up (currently there are none) as icons. Since there can be many locations in a region where sensors are installed, it doesn't make sense to check up each individually. Hence, if any anomaly is detected at a location, it shows up immediately in notifications.



DA-IICT Campus - List of Registered Areas

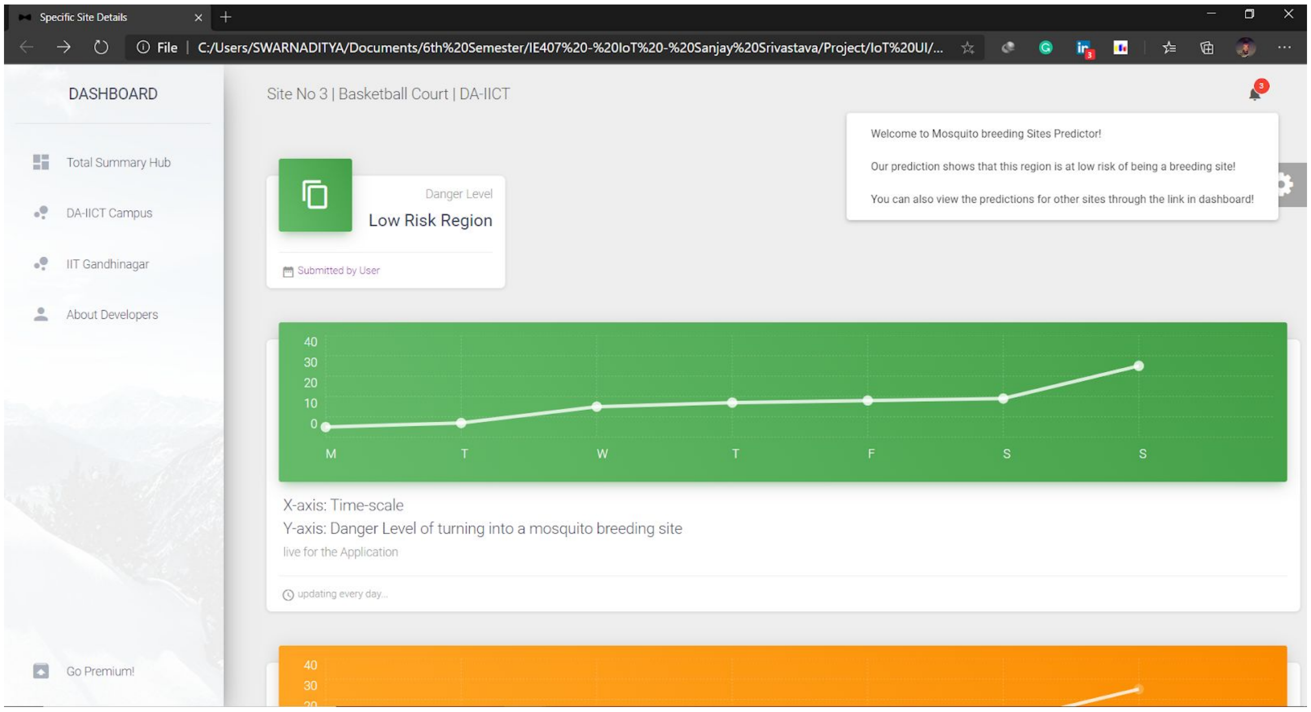
Welcome to Mosquito Breeding Sites Predictor

This page lists all the locations within DA-IICT where sensor modules are set up

Click on any one to view the situation in that area

The area behind Basketball Court has high possibility to turn into a breeding site!

4) On clicking any icon of a location, the user will be redirected to the page with specific details of that location.



Site No 3 | Basketball Court | DA-IICT

Welcome to Mosquito breeding Sites Predictor!

Our prediction shows that this region is at low risk of being a breeding site!

You can also view the predictions for other sites through the link in dashboard!

Danger Level

Low Risk Region

Submitted by User

X-axis: Time-scale

Y-axis: Danger Level of turning into a mosquito breeding site

live for the Application

updating every day...

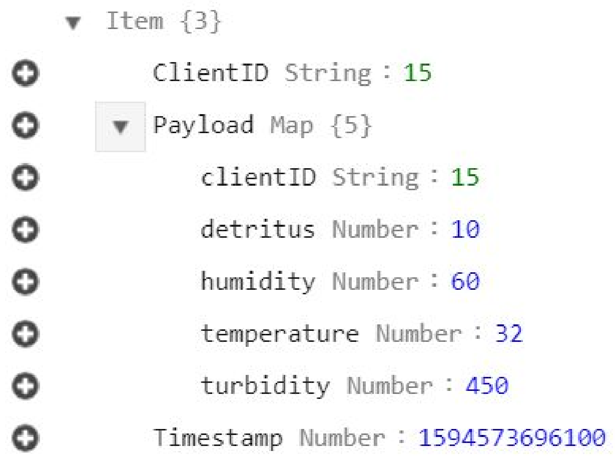
5) This Web Application has not yet integrated the REST APIs to access the URIs provided by the AWS API Gateway. Hence, the live updation of data can't be seen.

6 - Results

This section includes results about the working of various components of the project. Due to lack of access to the actual sensor data and sensors, to simulate MQTT publisher which publishes the data to MQTT server, a third party software **MQTT.fx** is used to publish messages manually. These messages carry the sensor data. An Example message structure is as follows:

```
{
"clientID": "15",
"temperature":32,
"turbidity":450,
"detritus":10,
"humidity":60
}
```

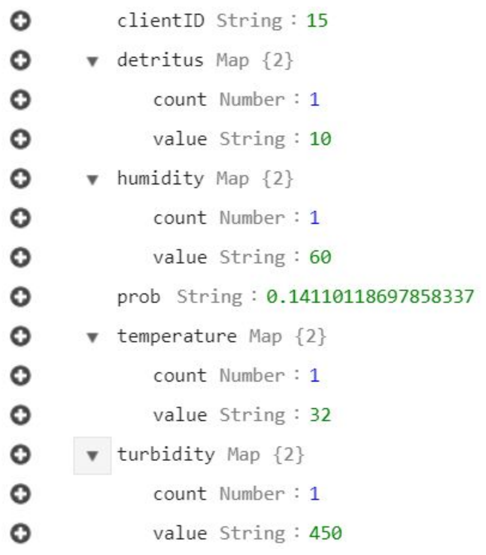
When this data is published to the MQTT channel, **the data is correctly populated in the database** in the sensorData table.



```
▼ Item {3}
    ClientID String : 15
  ▼ Payload Map {5}
      clientID String : 15
      detritus Number : 10
      humidity Number : 60
      temperature Number : 32
      turbidity Number : 450
    Timestamp Number : 1594573696100
```

AWS Lambda function(Trigger) is called upon insertion of this data and the **aggregated data is correctly inserted into the summary table.** The probability of mosquito breeding (stored in the field 'prob') is low because the environmental parameters are not ideal. The 'value' field will hold the average value for each parameter and the 'count' field will hold the count of the number of samples, which is needed for computing average.

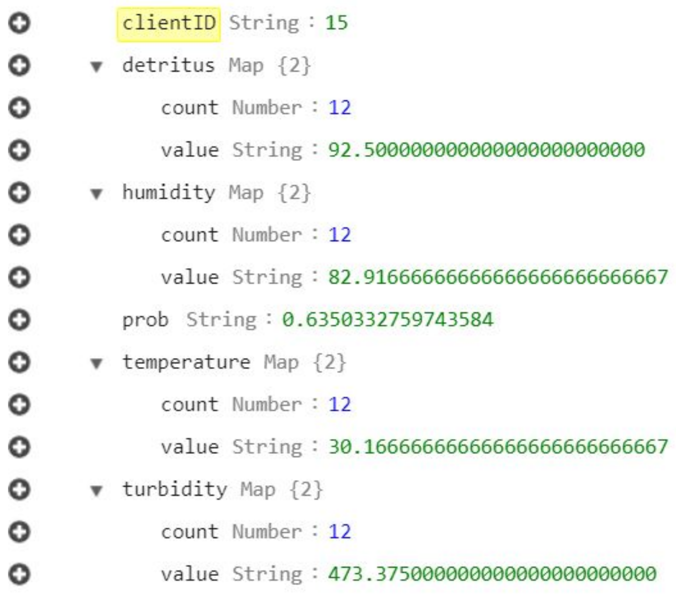
The inserted data is as follows:



```
clientID String : 15
▼ detritus Map {2}
    count Number : 1
    value String : 10
▼ humidity Map {2}
    count Number : 1
    value String : 60
prob String : 0.14110118697858337
▼ temperature Map {2}
    count Number : 1
    value String : 32
▼ turbidity Map {2}
    count Number : 1
    value String : 450
```

If data with environment parameters close to optimal values are published, the probability rises steadily. For example, upon publishing the following message for about 12 times, the probability of mosquito breeding rose to 0.635:

```
{
"clientID": "15",
"temperature":30,
"turbidity":475.5,
"detritus":100,
"humidity":85
}
```



```
clientID String : 15
▼ detritus Map {2}
    count Number : 12
    value String : 92.50000000000000000000000
▼ humidity Map {2}
    count Number : 12
    value String : 82.91666666666666666666667
prob String : 0.6350332759743584
▼ temperature Map {2}
    count Number : 12
    value String : 30.16666666666666666666667
▼ turbidity Map {2}
    count Number : 12
    value String : 473.37500000000000000000000
```

Note the change in count values for each parameter and the change in the average values. This data is made available as a resource and the client UI can get this aggregated data for a given clientID.

There are further experiments to be done with regards to the accuracy of the model, testing with actual sensor data and dealing with fluctuations due to environments, seasons and the accuracy of sensors.

7 - Testing plans for correctness and performance

Due to unavailability of resources, we couldn't set up the hardware. However, whenever the model is ready, we can test with real images and environmental data, the result of which we already know, whether it is a mosquito breeding ground or not. We can take images of stagnant drains, areas at the back of the campus, etc.
There are 4 types of possible results:

i) It is classified as a potential breeding ground, and it's actually a breeding ground. (Called true positive - TP)

ii) It is classified as a potential breeding ground, but it's not actually a breeding ground. (Called false positive - FP)

iii) It is not classified as a potential breeding ground, but it's actually a breeding ground. (Called false negative - FN)

iv) It is not classified as a potential breeding ground, and it's actually not a breeding ground. (Called true negative - TN)

We can do the above multiple times, and get the counts of TP, TN, FP and FN. We can calculate the percentage of TP and TN combined. This is the percentage of the number of times our model gives the correct answer. This method should give us a quantitative method of estimating the correctness of our model.

8 - Conclusion

The project began with an aim to identify the most probable mosquito breeding sites of mosquitoes. For the same, we have developed and deployed multiple approaches that include:

1) A **geographical pre-processing** to narrow down the number of macroscopic regions of water stagnation, with the aim of better allocation of resources. We accomplished this using **Google Earth, Microsoft Excel, and Surfer Software.**

2) We considered many important environmental parameters, identified using **CCA**, including **temperature, humidity, water turbidity, and detritus content.** We found out many dependencies between mosquito abundance and environmental parameters, and formulated other relations using **sigmoid function** by using the mean, covariance values from the studies conducted earlier**.**

3) With the sensors, we plan to transfer the data captured by the same, through **ESP8266** module to **Raspberry Pi** over a wireless network. An **MQTT** broker software runs on Raspberry Pi. We mimicked this behaviour using a software on our laptops, that publishes data through MQTT channels using MQTT IoT protocol, to an MQTT server on **AWS.**

4) We stored the data in different tables on **AWS DynamoDB**, and created python scripts that trigger on entry, updation, and deletion of data into the tables.

5) For the images which would be captured by the camera, we would train an **SVM** model, after passing all the images through the **HOG** feature extraction algorithm. With the model ready, we would be able to identify the water stagnation areas.

6) The summary information obtained after complete data processing, is obtained through **URIs** that are supplied by **AWS API gateway**. Once the URI becomes functional, **HTTP requests** on the same will fetch the data to the FrontEnd Web Application, in **JSON** format.

7) The Web Application was built using the standard trio of **HTML, CSS, and JavaScript.** The UI features have been completely implemented, however a few functionalities implementations are still in progress. The Users of the system are expected to get all the important information about all the regions where the system is implemented, using this application.

9 - Future Possibilities

This project has focused on determining the most probable locations of mosquito breeding sites, based on a plethora of environmental parameters. We haven't incorporated species identification algorithms in this project. However, using the same kind of data that has been used, combined with much more advanced forms of analysis and algorithms, can give us specific information about the mosquito species itself.

Our study of CCA ordination of tires in spring season, yielded two general environmental gradients: a detritus (fine, leaf, seed) versus human density gradient, and a tire size versus canopy cover and microorganism (protozoan abundance, bacterial productivity) gradient. In general, these gradients pointed to two mosquito assemblages, with Aedes being more abundant in tires surrounded by low human population densities but with high detritus amounts (i.e., fine, leaf, seeds), whereas Culex were more likely found in tires with greater protozoan abundance and lower canopy cover.

Regardless of season, a typical Aedes tire would most likely contain high amounts of detritus, be located in areas with low human density, and be occupied by both early and late instars of the same species. Conversely, a typical Culex tire would be relatively large and contain high concentrations of protozoans and bacteria, be found in association with high human densities, and be less likely than a typical Aedes tire to contain both early and late instars. The data for these is not much different from what has been already used in this project.

Hence, in the future, **we would like to expand the project to not only predict the locations of breeding, but also predict the mosquito species that is most likely to spawn.** This is invaluable information, since we would then be able to tell if the impending mosquito outbreak is just a simple menace or a potential pandemic like Zika Virus.

10 - References

1) https://sites.google.com/site/insectclassification/

2) https://www.researchgate.net/publication/323787037_A_Machine_Learning_Approach_for_Identifying_Mosquito_Breeding_Sites_via_Drone_Images

3) http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0085-56262010000300021

4) http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0074-02762018000400301

5) https://www.hsj.gr/medicine/influence-of-environmental-factors-on-breeding-habitats-of-mosquito-species-in-kosti-city-white-nile-state-sudan.php?aid=23140

6) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3377495/

7) https://www.youtube.com/watch?v=j5vhxcTBArQ&feature=youtu.be

8) https://pubmed.ncbi.nlm.nih.gov/17401581/

9) Beier JC, Patricoski C, Kranzfelder J. Habitat segregation among larval mosquitoes in tire yards in Indiana, USA. J. Med. Entomol.

10) Yee DA, Juliano SA. Consequences of detritus type in an aquatic microsystem: assessing water quality, microorganisms, and the performance of the dominant consumer.