CHRIST

(DEEMED TO BE UNIVERSITY)

BANGALORE · INDIA

LIVESTOCK MONITORING SYSTEM

by

Bishwash Krishna Parajuli (1741016)

Pitchayut Khumkrongsap (1741042)

Under the guidance of

Dr. Saravanan K. N

&

Dr. Sagaya Aurelia P

A Project report submitted in partial fulfilment of the requirements for the award degree of Bachelor of Computer Applications of CHRIST (Deemed to be University)

March – 2020

CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

CERTIFICATE

This is to certify that the report titled ***Livestock Monitoring System*** *is a bona fide record of work done by* ***Bishwash Krishna Parajuli (1741016) and Pitchayut Khumkrongsap (1741042)*** *of CHRIST (Deemed to be University), Bangalore, in partial fulfilment of the requirements of VI Semester BCA during the year 2020.*

Head of the Department

Valued-by:

1.

2.

Faculty in charge

Name : Bishwash Krishna Parajuli
Pitchayut Khumkrongsap

Register Number(s) : 1741016
1741042

Examination Centre: CHRIST (Deemed to be University)

Date of Exam :

ACKNOWLEDGEMENTS

While working on this project we were accompanied and supported by our peers and the teaching faculty. This is a very pleasant aspect for us as now, we have the opportunity to express our gratitude towards all of them for their constant guidance and encouragement. Our effort would not have been successful if it was not for them.

First and foremost we express our deep sense of gratitude to **Dr. Fr. Abraham V M**, Vice Chancellor of CHRIST (Deemed to be University) for providing the necessary facilities that helped us in many ways to accomplish this project. We feel a deep sense of gratitude to **Dr. Fr. Joseph CC**, Pro-Vice Chancellor of CHRIST (Deemed to be University) for his moral support. We would also like to thank our head of the department **Prof. Joy Paulose,** Department of Computer Science, CHRIST (Deemed to be University), for his invaluable support during the course of this project. We acknowledge our sincere thanks to **Prof. Deepthi Das**, Coordinator of UG, Department of Computer Science, for her valuable guidance throughout the course of the project.

We are most thankful to our project guide, **Dr. Saravanan K. N**, Associate Prof., Department of Computer Science, CHRIST (Deemed to be University), for guiding us with his tremendous knowledge in the field with true spirit throughout the project.

We would like to thank **Dr. Arokia Paul Rajan** and **Dr. Sagaya Aurelia P** and all the faculty members of the Department of Computer Science, CHRIST (Deemed to be University) for providing us with a number of advice and motivation that helped us complete our project on time.

Also, we are thankful to have help received from all our friends and classmates. We would also like to thank each and every one who contributed directly or indirectly towards the successful completion of our project work.

ABSTRACT

India is home to more than 140 million cows and hundreds of millions of other livestock animals such as goats, sheep, pigs, etc. Livestock Monitoring System is a Web based IoT project designed to benefit the farmers of India to effectively monitor their Livestock Animals. The main focus of the project will be on cows as millions of Indians depend on cows as their source of income. Heat stress is one of the major factors affecting milk production, reproduction and general health of a dairy cow.

Many attribute heat stress only to temperature, but in fact it is a combination of air temperature and humidity. Livestock Monitoring System will make effective use of IOT devices such as Microprocessor (NodeMCU) with Wireless Module (ESP8266), Temperature and Humidity Sensor (DHT11) and GPS module (Neo6M) to effectively monitor constraints such as Temperature and Humidity to calculate the Temperature and Humidity Index(THI) which will then be computed inside a Web page and display data along with suggestions to help the farmers to regulate heat stress amongst cows and and maximize production of milk. The project will also be able to deliver the farmers with the location of their cattle animals with the use of GPS module.

This project focuses on offering a cost-efficient livestock monitoring solution to the farmers of India who are operating on a small scale.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Introduction

In India, Millions of people are dependent on Animal husbandry as their source of income. These animals must be kept safe and their health must be monitored by their owners. Use of technology such as the Internet of Things to monitor their location, temperature and humidity has now become a must for the owners. Use of these types of technology will benefit the farmers by increasing production and keeping their cattle healthy for a longer period of time.

Every year thousands of cattle are dying due to heat stress and farmers are facing millions of loss worth. Some of these diseases can be easily prevented by properly monitoring the temperature of livestock animals and the humidity of the environment. Monitoring factors such as their location, body temperature and humidity can help us find an effective solution to control heat stress in cattle and boost the production of milk. The IoT sensors would help us identify whether or not the livestock animals are experiencing heat stress which is one of the major reasons for them to minimize milk production. Heat stress also makes these animals prone to certain viruses such as influenza and others. Using Livestock Monitoring System, we can easily tackle these problems. The system also allows the owner to track their cattle using Global positioning system (GPS) to find their whereabouts. This is important as India is home to more than 500 million livestock animals, more than 140 million of them being cows. Most of these animals are owned by farmers who operate on a very small scale. The system would provide a cost-friendly solution to the daily problems of Indian farmers and help them maximize their profits by boosting milk production.

1.2. Existing system

1.2.1. Cowlar

Cowlar is a wireless neck collar that is used for tracking, detecting illness and it can be used to get actionable recommendations for increased milk yield and improve reproduction rates. Cowlars communicate wirelessly with a solar powered base unit that has a 2 mile range. The base unit of the Cowlar is sold separately from the main device . Cowlar is widely used in the US market and the cost of this device is $99 without its base unit.

Fig 1.1. Cowlar

Features

- Designed for comfort of the animal
- Wireless connection for a range of 2 miles
- Can withstand impact from large animals
- Waterproof (IP68 rated, testing for 2m below water for 90 min)
- Recommendations will be sent on text, phone, email

1.2.2. Cattle Monitoring using Image Processing

Behavioral scientists track animal behavior patterns through the construction of ethograms which detail the activities of cattle over time. To achieve this, scientists currently view video footage from multiple cameras located in and around a pen, which houses the animals, to extract their location and determine their activity. This is a time consuming, laborious task, which could be automated.

Limitations of Existing System (Cattle Monitoring using Image Processing)

- Installation of CCTV cameras in various locations.
- Multiple cameras need to be calibrated on a daily basis.
- Real Time Compressive tracking is not cent percent accurate.
- Faulty processing when the cattle move from one location to another and multiple cameras will have to process the images all over again.

Limitations of Existing System (Cowlar)

- You have to install the Cow Router separately and connect it with the servers.
- Charge for the router is to be paid separately.
- Other charges such as server charges and installation charges.
- Cowlar does not have a GPS tracking system.
- Base unit and Cowlar should be within a mile range to work.

Fig 1.2. Base Unit of Cowlar

1.3 Proposed System

Livestock Monitoring System is a prototype device designed to monitor Livestock animals by tracking their location and monitoring milk production using constraints such as body temperature and relative humidity. LMS is a Web based IoT project where modules such as GPS (Neo6m), Temperature and Humidity Sensor (DHT11) and Wifi Module (ESP8266) will be connected to the NodeMCU. The data obtained from this circuit will be computed and displayed on a WebPage which provides a user-friendly application for the end users. The prototype system works for 5 different breeds of cows for the user to select, monitor and obtain results accordingly. This system provides a cost efficient system which prevents small scale poultry and livestock farmers from spending thousands of dollars on a monitoring system.

1.4. Benefits/Scope of Proposed System

- Users will be able to monitor their cattles.
- The information regarding the livestock animal will be sent directly to the user's device.
- The Customers will save a lot of money such as installation costs.
- GPS location of the livestock animals can be monitored.
- Proposed system is cost-efficient and does not require a lot of resources.
- Easy system installation.
- The system will be connected to Firebase which is an efficient and protected database.
- The IoT devices will be connected to the web and visualisations such as maps and graphs will be shown.
- Data such as heat-stress can be obtained to monitor the health of the cows.
- Using the THI value obtained from the system, the user will be able to maximize the milk production of their cattles.

2. REQUIREMENTS SPECIFICATIONS AND SYSTEM REQUIREMENTS

2.1. Functional Requirements

The system is designed in such a way that it needs to be broken down into three high level modules. Each of these modules are a step by step representation of how the system is going to function. Combination of these modules gives a robust system to monitor livestock animals.

2.1.1. Location Tracking

The location of the livestock animal is one of the major components of the system. The Location of the cattle will be monitored using the Neo 6M GPS module. Neo 6M GPS tracker is considered to be one of the most cost effective and accurate GPS tracking devices. The GPS module gives us the latitude and longitude value of the animal's position which can be used to determine its 2D location. This module will be connected to the NodeMCU microprocessor board with the use of breadboard and resistors, which will then be programmed using the Arduino IDE.

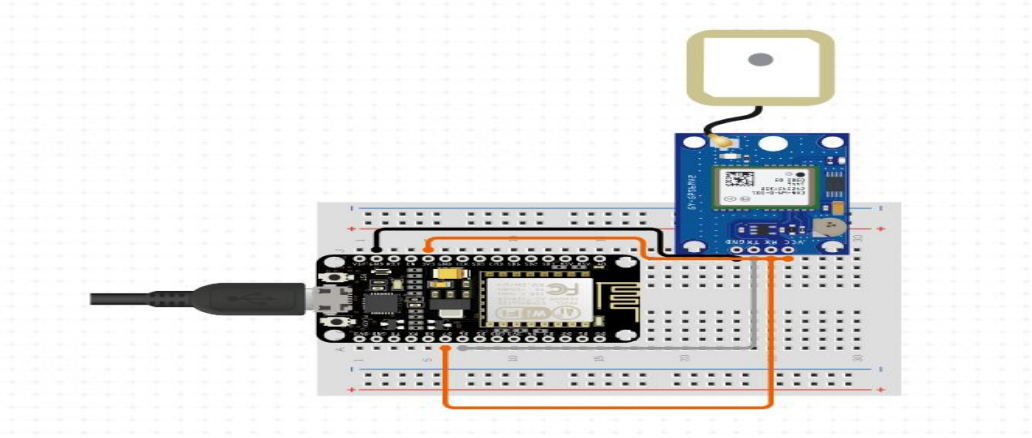

Fig 2.1. Neo-6M GPS module with NodeMCU

2.1.2. Temperature and Humidity Index

Most of the livestock animals, for example cows thrive when it comes to certain environmental factors such as temperature and humidity. Production of milk can increase or decrease significantly with change in temperature and humidity. Using DHT11 Temperature and Humidity Sensor, we can determine the temperature of the environment in degree celsius value and the humidity percentage of the environment. The values obtained from the sensor can be used to calculate Temperature and Humidity Index (THI) which can be used as a metric to monitor the cows' milk production. THI can be used to determine the following effects of environment temperature and humidity:

- When THI exceeds 72, cows are likely to begin experiencing heat stress and their incalf rates will be affected.
- When THI exceeds 78, cows milk production is seriously affected.
- When THI rises above 82, very significant losses in milk production are likely, cows show signs of severe stress and may ultimately die.

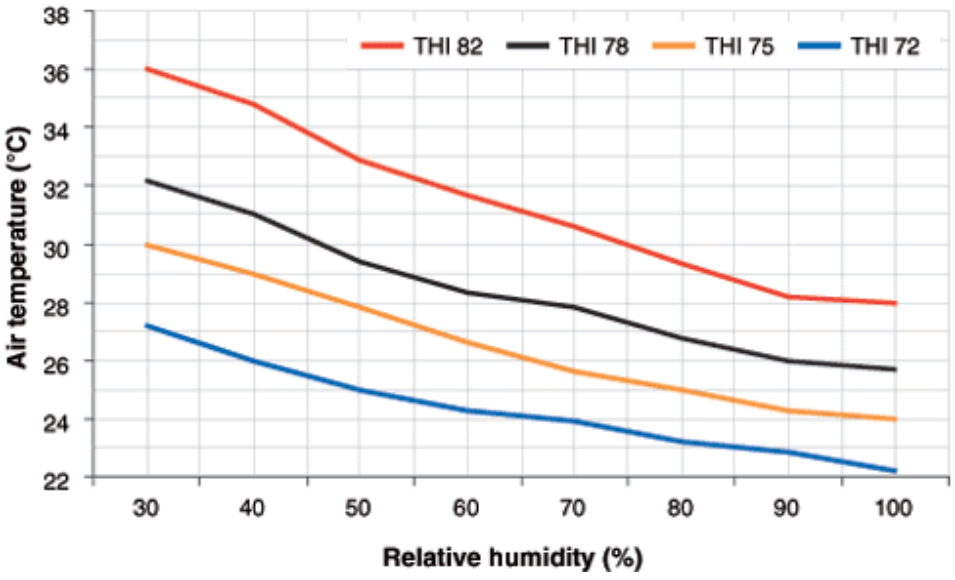

Fig 2.2. Graphical Representation of THI

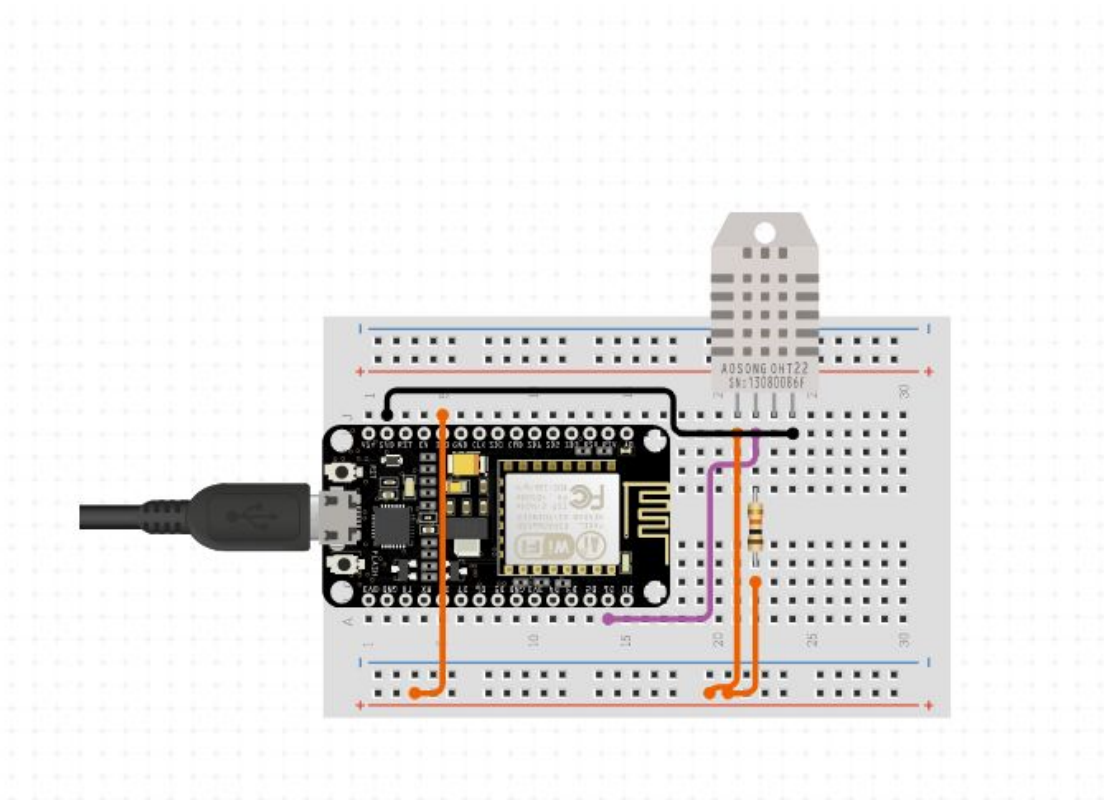

Fig 2.3. DHT11 Temperature and Humidity sensor with NodeMCU

2.1.3. Data Communication

Using the cost effective and efficient ESP8266 wifi module we can obtain the values collected from the NodeMCU Board. The wifi module will be connected to the NodeMCU board directly and it will be programmed in such a way that the user will receive the status of the livestock animals in their system which will be connected to Firebase Database. The Data will then be fetched by the webpage to provide suggestions and values for the user to monitor their livestock animals.presentation such as graphs from the data received from our devices. This module will help us communicate with the primary NodeMCU board. Values such as the 2D location of the animal and THI cannot be obtained without this module.

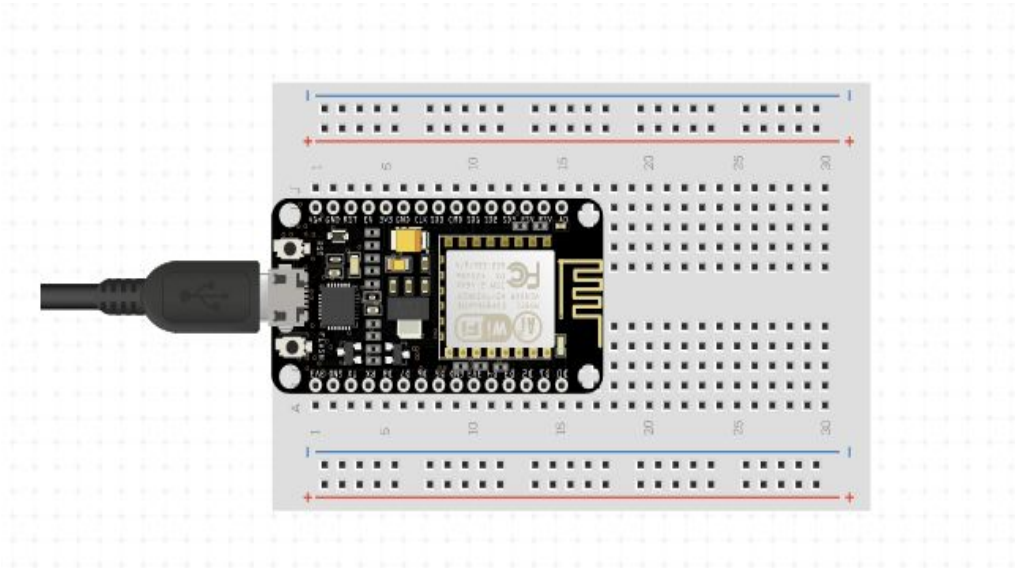

Fig 2.4. NodeMCU (with Inbuilt ESP8266 wifi module)

2.2. Non - Functional Requirements

2.2.1. Operating Environment and Connectivity

Livestock Monitoring System shall operate on various numbers of Livestock animals such as cows, goats, horses, pigs, sheep, et cetera. Like most of the IOT projects, Livestock Monitoring system will be powered by a standard 9v battery which will be directly connected to the standard NodeMCU processor board. All of the devices can connect using a wide variety of other technologies. The connectivity objective is that an IoT platform supports as many modes of connection, wired and wireless. Livestock Monitoring System will use an ESP8266 Wifi module for connectivity purposes. The wifi will bridge a gap between the IOT devices and the user's main device, be it a computer or a phone.

2.2.2. Design and Implementation Constraints

The system shall use the current corporate standard sensors and modules. All the devices such as the Temperature and Humidity sensor, Wifi module, 9v battery connector and the GPS module will be designed and connected to the NodeMCU with the help of a breadboard where a number of wired connections and resistors will be used. The system prototype shall be implemented on a number of livestock animals with cows being the main focus in order to monitor health and productivity.

2.2.3. Platform Scalability and Availability

The system shall support a small platform as it is only a prototype and the scope of the product is scaled to a number of animals but initially it must be implemented only on one livestock animal, for testing purposes. Scaling-up will be done as per the needs of the project requirements. All the modules and sensors, even the test objects(livestock animals) are assumed to be available within the reach and scope of the project.

2.2.4. Assumptions and Dependencies

- The connected devices, IOT devices and the user device are powered on all the time.
- The user's device has internet connectivity all the time which is turned on at the time of monitoring.
- The user has enough knowledge to access ThingSpeak to monitor the data sent by the system.
- The system prototype is directly and physically connected to the livestock animal.

2.3. System Requirements

The Livestock Monitoring System requires the following Hardware and software specifications to be installed by the admin and the user. Such is categorised in the form of client view and admin view.

2.3.1. Hardware Requirements

Hardware requirements for the system are listed below.

Client view:

- PC
- Internet Connection
- Smartphone with Internet connectivity

Admin view:

- Computer with Internet Connectivity
- Processor (NodeMCU)
- DHT-11 Temperature and Humidity Sensor
- ESP 8266 wifi module (Integrated with NodeMCU)
- Neo-6m GPS Module
- 9v Battery connector
- USB data cable
- BreadBoard and copper wires
- 1K Ohm Resistors

2.3.2. Software Requirements

Software requirements for the system are listed below.

Client view:

- Operating System
- Web Browser

Admin view:

- Arduino IDE
- Code editor for HTML, CSS and JavaScript
- ESP8266 Library for Arduino IDE
- TinyGPS Library for Arduino IDE
- DHT11 Library for Arduino IDE
- Firebase Arduino Library for Arduino IDE
- Arduino JSON Library for Arduino IDE
- Software Serial Library for Arduino IDE
- Google Maps API
- Firebase API
- Google Charts API
- Operating System
- Web Browser
- ThingSpeak API

2.4. Block Diagram

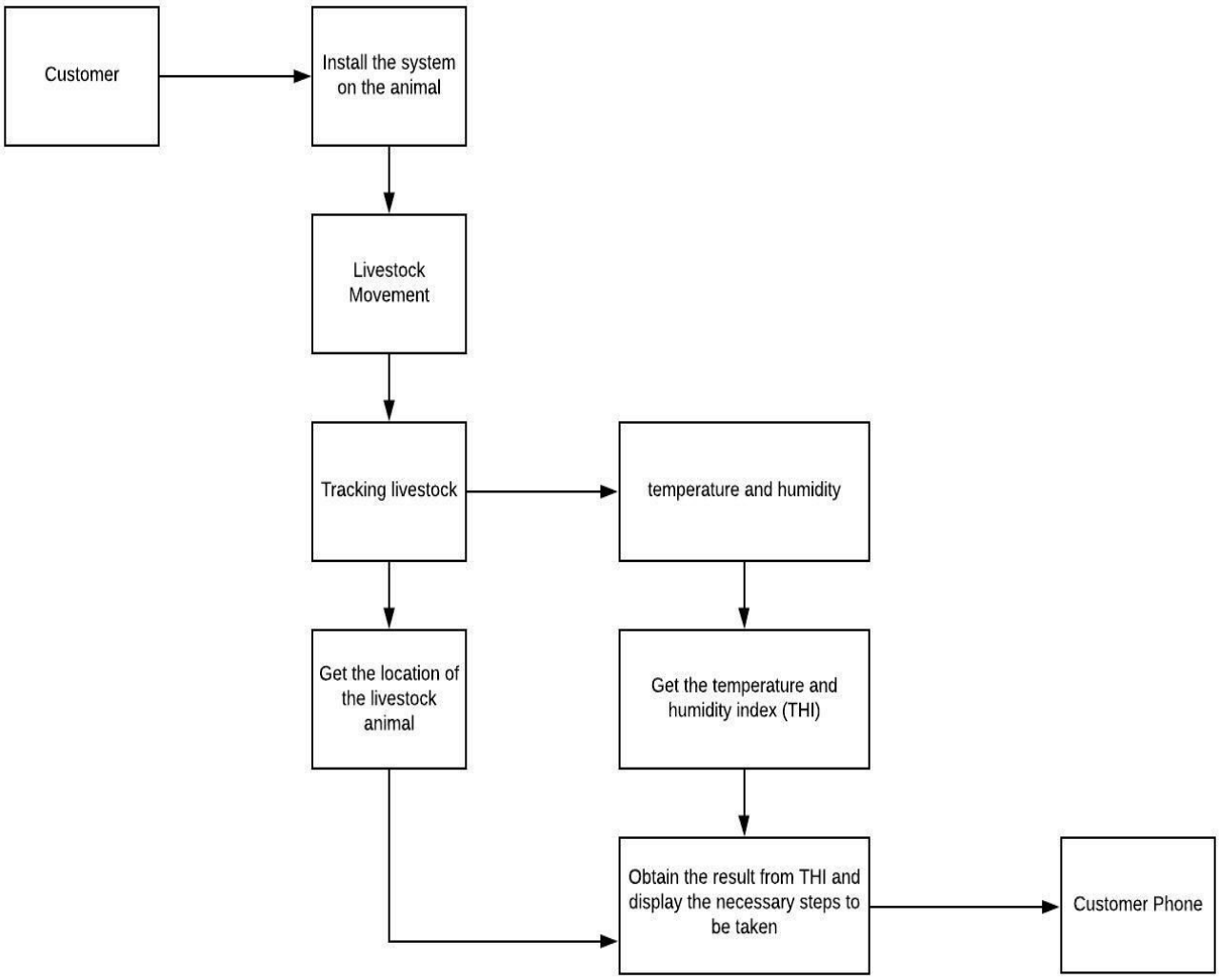

Fig 2.5. Block Diagram of Livestock Monitoring System

This block diagram above represents the step by step procedure of how the system works. Collection of data and communication between different modules and components can be observed in the above diagram.

2.5. Activity Diagram

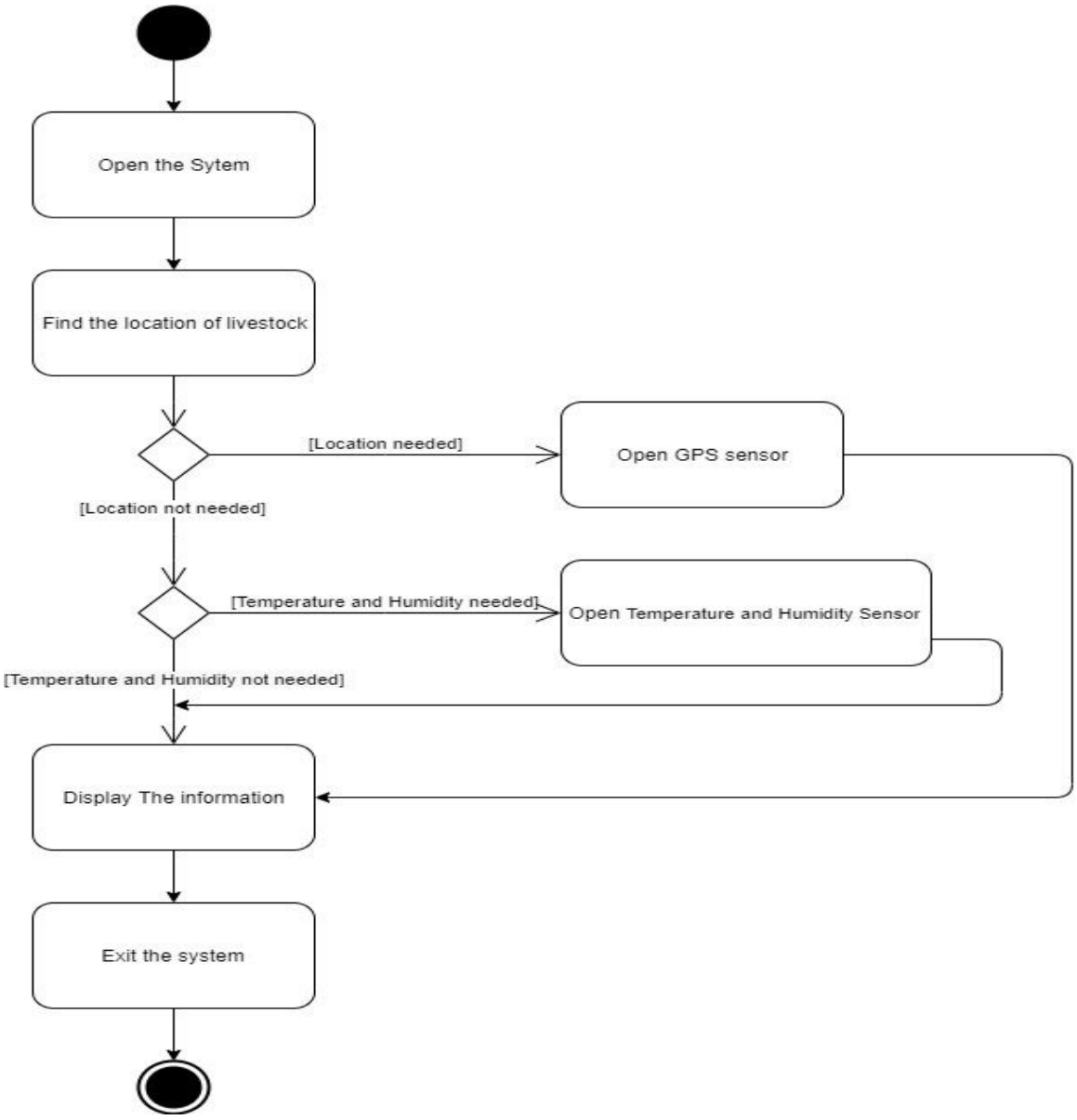

Fig 2.6. Activity Diagram of Livestock Monitoring System

The Activity diagram depicts the various procedures and communication of different module components from the users' point of view. The diagram also shows how the information will be displayed in the users' device to efficiently monitor livestock animals.

2.6. Use Case Diagram

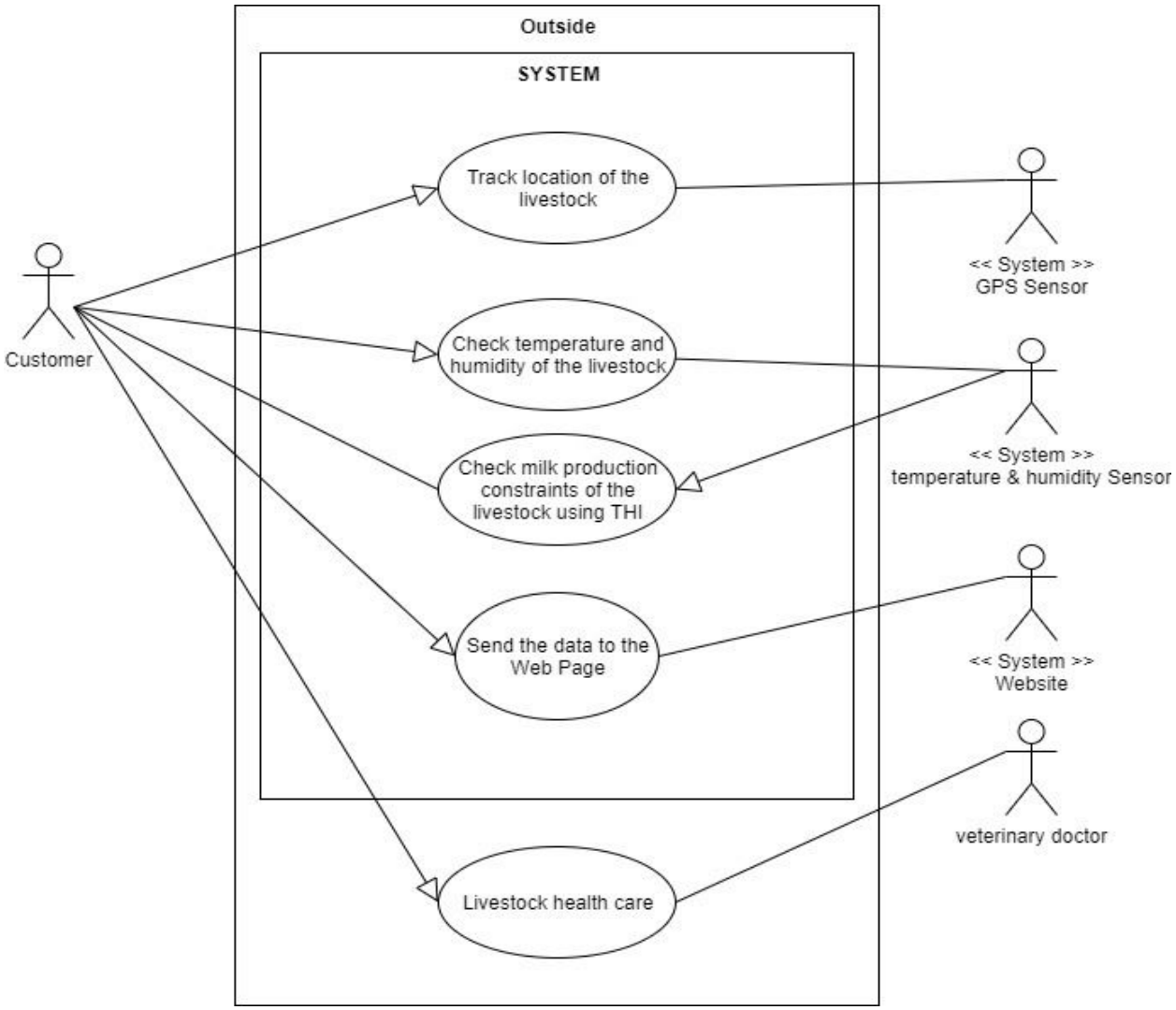

Fig 2.7. Activity Diagram of Livestock Monitoring System

The above Use Case diagram completely depicts the system in the users' point of view. Users' communication and interaction with the system can be clearly obtained from the diagram. The module components are also represented as users' in order to simplify the communication process.

3. SYSTEM ARCHITECTURE

3.1. System Design

Livestock Monitoring System is an IoT project designed for all the small scale farmers who depend on cows and other livestock animals as their source of income. The system is designed to help farmers maximize milk production by regulating heat stress which is a very common factor amongst cows.

The device consists of IoT devices such as DHT11 Temperature and Humidity Sensor, Neo-6m GPS Module and ESP8266 Wifi Module which are all connected to a common processor, NodeMCU. The NodeMCU is connected to Google Firebase platform. Then a Website designed for the system will fetch data from Firebase for further processing and will be displayed for the user to obtain useful information about their cow or any other livestock animal. The system must be tied to the livestock animal using a strap for data to be obtained.

The DHT11 Temperature and Humidity Sensor sends Temperature value in degree centigrade and Humidity value (relative humidity) in percentage to the Processor. The Neo-6m GPS module sends the Latitudinal and Longitudinal position values of the cow to the Processor. All four of these values are stored in float datatype by the processor and sent to the Firebase Platform using ESP8266 WiFi module.

The WiFi module by default is connected to the Processor. Both the wifi module and the Admin device must be connected to the same Wifi network. The Firebase API and the Processor is connected using a secret authentication key obtained from the Admin's Firebase account. The code must be entered in the Arduino IDE file at the time of coding along with the WiFI SSID and Password.

The Firebase API is then called inside the Webpage which is designed using HTML and CSS. The webpage calculates information such as Temperature and Humidity Index for the user which is computed using Javascript and Google Maps API is used to plot latitude and longitude values into Google Maps to obtain the cow's location.

3.2. Database Design

Table 3.1. Database Design - Livestock Data

Livestock Data Table		
Field	**Datatype**	**Description**
Breed (PK)	varchar	Different breed/species of cows
weight	float	Average weight of the breed (in kg).
Lactation_Period	float	Number of days a cow can produce milk.
Age_First_Calving	float	Age of the cow at first parturition.
Calving_Interval	float	Average number of months it takes for a specific breed of cow to give birth after the previous parturition.
Milk_Fat	float	Percentage of fat content in the milk (%).
my_Lactation	float	Total amount of milk a cattle can produce (in kg) in a lactation period. Milk yield per Lactation.
my_Year	float	Total amount of milk a cattle can produce (in kg) in a year. Milk yield per Year.
my_Month	float	Total amount of milk a cattle can produce (in kg) in a month. Milk yield per Month.
my_Day	float	Total amount of milk a cattle can produce (in kg) per day. Milk yield per Day.
Nmpd	int	Number of times you can milk the cow in a day.

Table 3.2. Database Design - Sensor Data

Sensor Data Table		
Field	**Datatype**	**Description**
Temperature	float	Temperature data in Degree Centigrade from Temperature and Humidity Sensor
Humidity	float	Relative Humidity in Percentage from Temperature and Humidity Sensor
Latitude	float	Latitudinal position of the livestock animal from the GPS
Longitude	float	Longitudinal position of livestock animal from the GPS

3.3. Data Flow Diagram

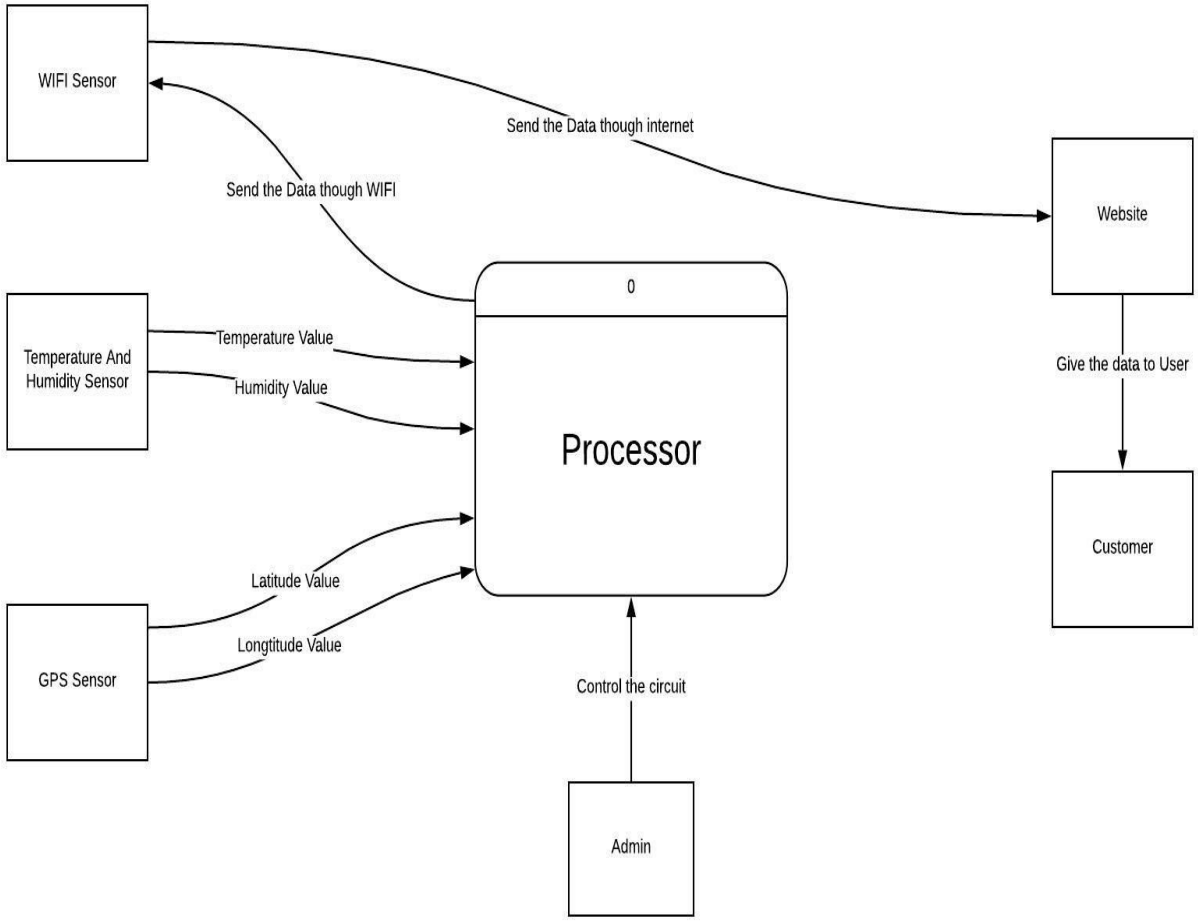

Fig 3.1. Data Flow Diagram (level 0) - Livestock Monitoring System

The DFD represents the flow of data from one IoT device to another. The Processor collects data such as Temperature, Humidity, Latitude and Longitude from Temperature and Humidity Sensor and GPS module which is sent to the Firebase platform using Wifi module. The data is then fetched by the Website for the user to access. It is assumed that the processor and other IoT devices are tied to the Livestock Animal for which the website shows results accordingly.

3.4. Entity Relationship (ER) Diagram

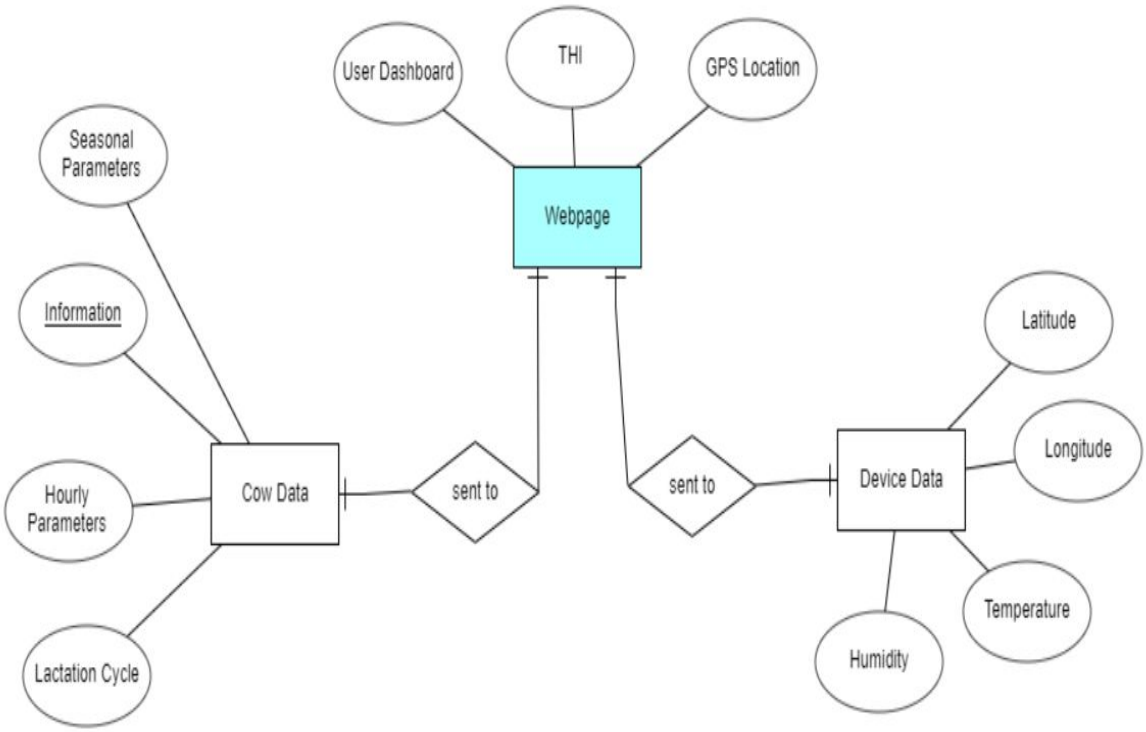

Fig 3.2. Entity Relationship Diagram of Livestock Monitoring System

In this ER diagram, The Entities Cow Data and Device Data is being integrated to the Entity Webpage of the System where the user will obtain information regarding the location of the livestock animal, hourly updates, Seasonal Update and update regarding it's Lactation cycle to help the farmer to maximize milk production.

3.5. Circuit Diagram

3.5.1. Phase I Circuit

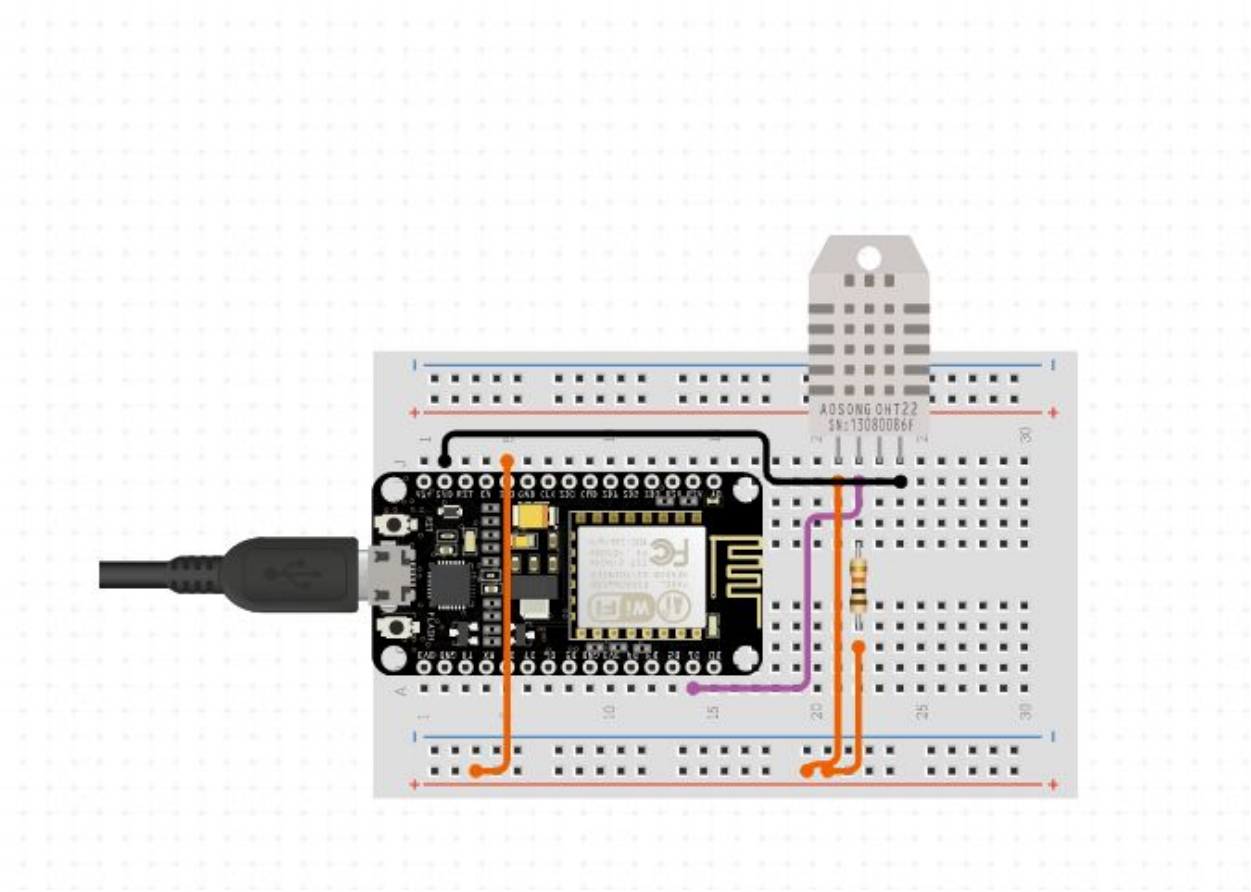

Fig 3.3. Phase 1 Circuit Diagram - Livestock Monitoring System

In this phase, DHT11 Temperature and Humidity Sensor is connected to NodeMCU and data is sent to ThingSpeak, an IOT cloud platform through ESP8266 Wi-Fi module, to calculate and analyze the Temperature and Humidity Index (THI). All the sensor data and calculated values were successfully obtained and analyzed on the ThingSpeak Server using tools such as graphs, buttons, gauges and other indicators.

3.5.2. Complete Circuit Diagram

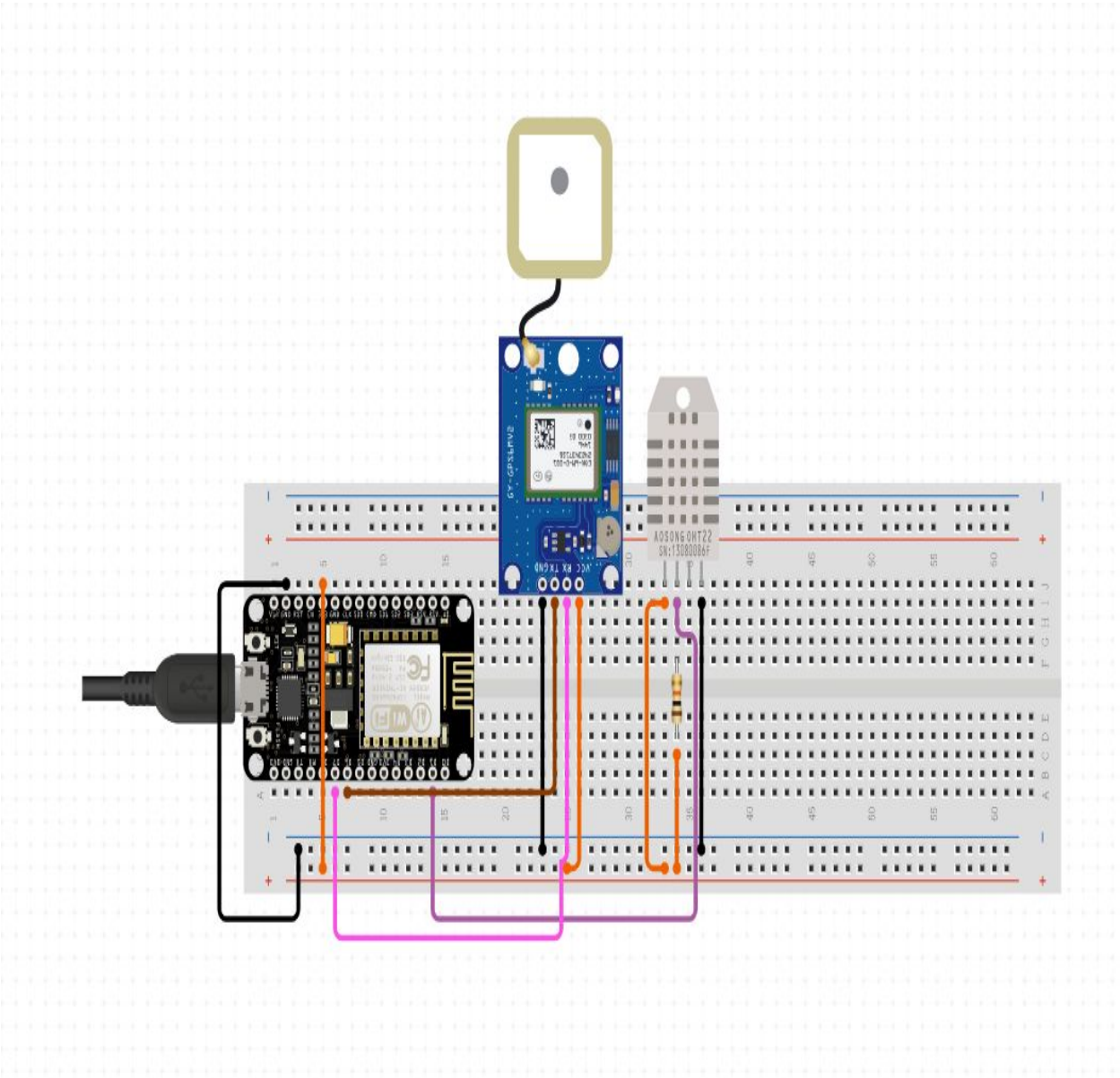

Fig 3.4. Final Circuit Diagram - Livestock Monitoring System

The Final Circuit diagram shows how all the IoT sensors and modules are connected to the processor. Each of the components are connected to the Ground pin of the NodeMCU. They are also connected to the Voltage collector to collector (VCC) pin which either regulates voltage of 3.3V or 5V according to the requirement of the device.

These sensors are then connected to Transmitter and Receiver pins of the processor for sending and receiving data and they are also connected to the digital pins for data processing. Since different IoT devices are connected to the same pins of the processor, we use breadboards and resistors to make that happen.

The DHT11 Temperature and Humidity Sensor sends Temperature and Humidity Data to the NodeMCU processor, The Neo-6m GPS module sends Latitude and Longitude value to the NodeMCU processor and then we use ESP8266 WiFi module to send these values to the Firebase platform for further data processing.

3.6. User Interface Design

3.6.1. LMS Home Page

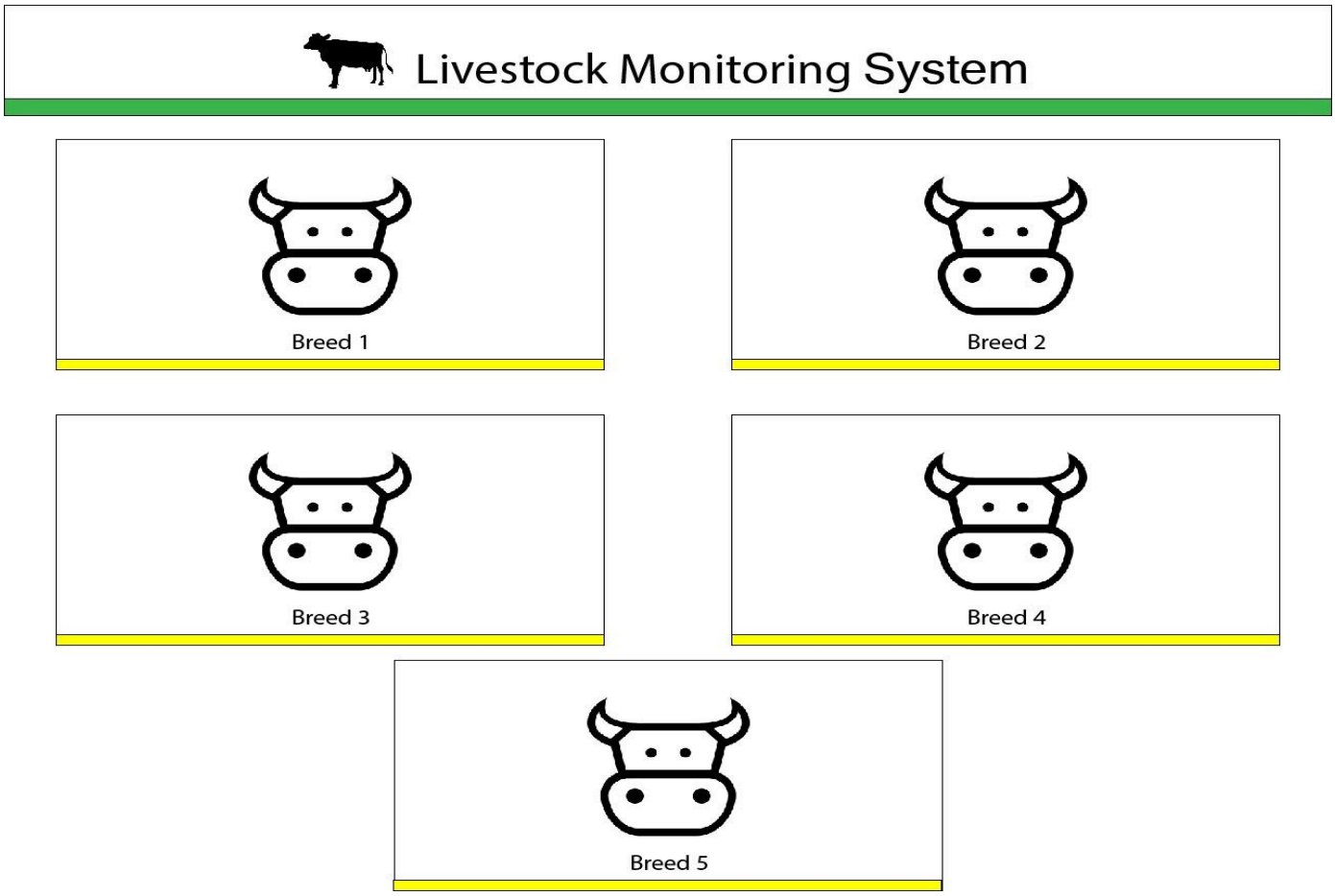

Fig 3.5. User Interface Design Page 1 - Livestock Monitoring System

This diagram represents the Home page of the Livestock Monitoring System Website which gives the user an option to choose a Cow according to their breeds. The system will give the user an option to select one of five different breeds/species of cows which will then redirect to another page where further information will be given about the specific breed.

3.6.2. LMS User Dashboard Page

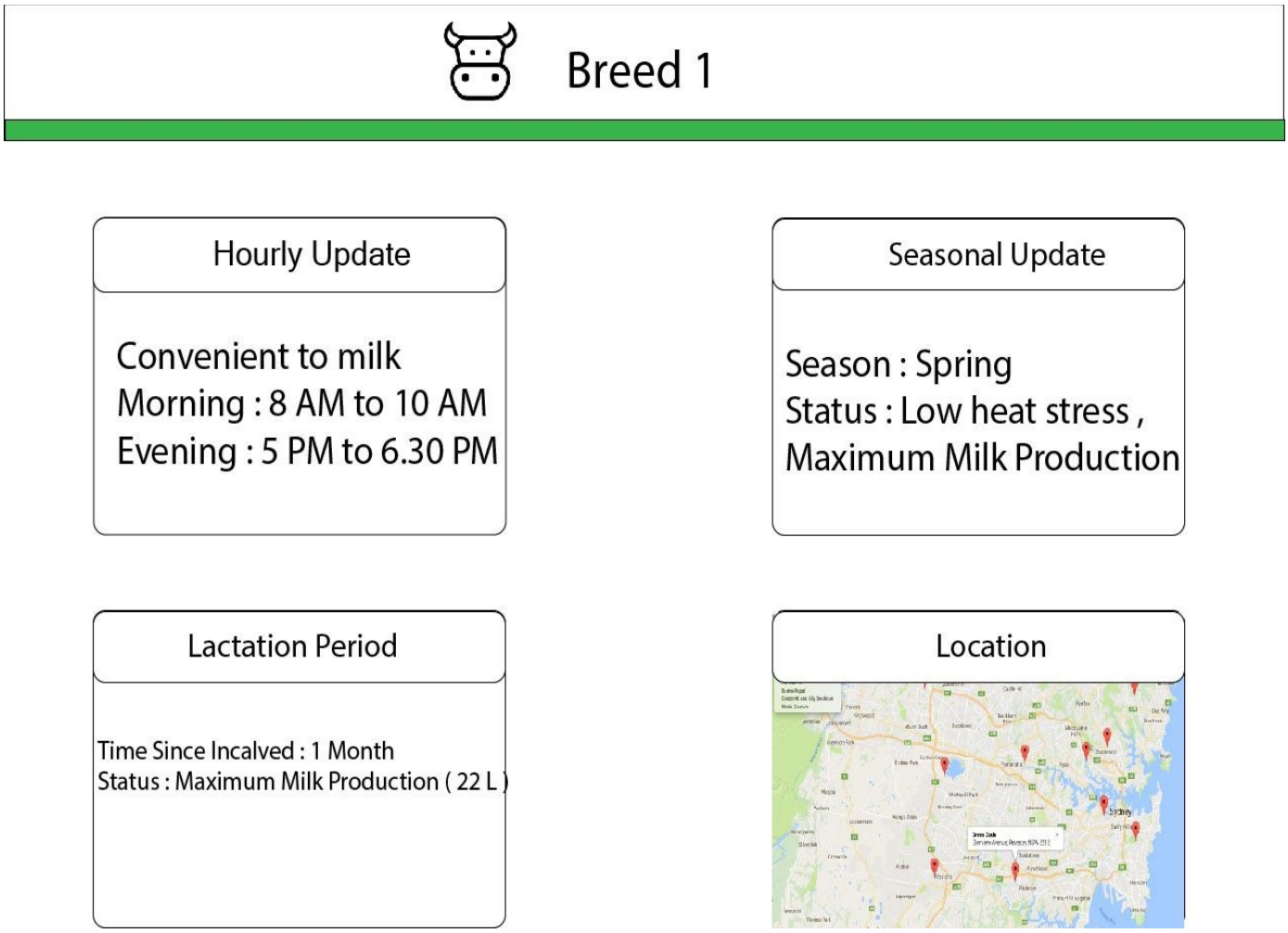

Fig 3.6. User Interface Design Page 2 - Livestock Monitoring System

The diagram above shows the UI design of the Dashboard page of Livestock Monitoring System. Information such as a Cow's (Breed specific) convenient time to give milk, seasonal milk production details and its lactation cycle will be shown to the user along with its current location which will be displayed on Google Maps.

4. IMPLEMENTATION

4.1. Implementation Approaches

Livestock Monitoring System provides a platform which supports five different breeds of cow or species of cow owned by the user. The System is built in such a way that IoT devices are connected and integrated on a web based platform which is easy for the user to access. This user-friendly web based platform will not ask the user with information such as age of the cow in month and date of last parturition. Using these two information along with the information provided by the IoT device, we can compute results such as number of milking per day and provide users with suggestions to provide a steady Environment to maximize milk production.

The Processor along with the Wifi, Temperature and GPS Module will send the data to the Firebase database where the data is stored in realtime. This is done using the Arduino IDE where multiple libraries are used to send the data to firebase. After successfully sending data to firebase, our webpage will ask the user to provide some information about the cow and accordingly compute the results to provide effective solutions or suggestions to the user. The Webpage is designed using HTML and CSS whereas JavaScript queries are used to fetch the realtime database to the web and compute the data for the user.

4.2. Coding Standards

The Coding standards for the project is divided into three parts. All of them are mentioned below and explained briefly.

4.2.1. Front End Web Page

HTML is used to design the webpage whereas CSS is used to style and provide an interactive webpage for the users. The validations and functionality of the coding is completely done using HTML whereas the CSS is used to insert and align various images and colours to make the website as user-friendly as possible

Javascript Queries are used to make the website interactive and give it a more dynamic approach. The algorithms and computation done on the website is also done by using Javascript.

4.2.2. Back End Web Page

Firebase Realtime database is used to store the data from the devices as well as the data entered by the user and additional information related to the livestock animal. The firebase is connected to the web using Javascript where the variables are stored inside Firebase and called when needed. Google Maps API is used using the latitude and longitude coordinates provided by the GPS Module. Google Charts API is also used to represent the temperature, humidity and Temperature and Humidity Index in a diagrammatic form

4.2.3. Back End Arduino

Arduino IDE is used to code the processor which is connected to the sensors. The language used to code is C programming where a number of existing libraries are called in the Arduino IDE. Libraries for the processor, devices and modules are fetched by the arduino and then sent to Firebase to provide the Web Page with realtime Database. The data is stored in a JSON form and sent to the firebase.

4.3. Coding Details (Source Code)

4.3.1. Front End Web Page

//Homepage

```
<!DOCTYPE html>
<html lang="en">

<head>
   <link rel="stylesheet" type="text/css" href="Vendors/CSS/normalize.css">
    <link rel="stylesheet" type="text/css" href="Vendors/CSS/038%20Grid.css">
   <link rel="stylesheet" type="text/css" href="Resources/CSS/style_h.css">
   <link href="https://fonts.googleapis.com/css?family=Montserrat&display=swap" rel="stylesheet">
   <title>Home Page</title>
</head>
   <body>
      <header>

         <h1>Livestock Monitoring System</h1>

      </header>

      <section class="section-photo">
         <ul class="Breed-selection">
            <li>
               <figure class="cow-photo">
                  <img src="Resources/Image/1.jpg" alt="Ongole">
               </figure>
            </li>
            <li>
               <figure class="cow-photo">
                  <img src="Resources/Image/2.jpg" alt="Kankrej">
               </figure>
            </li>
            <li>
               <figure class="cow-photo">
```

```
                <img src="Resources/Image/3.jpg" alt="Kenkathal">
              </figure>
            </li>
            <li>
              <figure class="cow-photo">
                <img src="Resources/Image/4.jpg" alt="Kherigarh">
              </figure>
            </li>
            <li>
              <figure class="cow-photo">
                <img src="Resources/Image/5.jpg" alt="Rathi">
              </figure>
            </li>
          </ul>
        </section>

        <section class="section-breeds">
        <div class="row">
          <div class="col span-1-of-5 box ">
          <h3>Ongole</h3>
            <p>Cow's Info</p>
             <a class="btn btn-full"
href="file:///C:/Users/Admin/Documents/Livestock/Breed_1_Page.html">Get Info</a>
          </div>

            <div class="col span-1-of-5 box">
          <h3>Kankrej</h3>
            <p>Cow's Info
              </p>
           <a class="btn btn-full" href="#">Get Info</a>


          </div>
            <div class="col span-1-of-5 box">
          <h3>Kenkatha</h3>
            <p>Cow's Info</p>
           <a class="btn btn-full" href="#">Get Info</a>
```

```
        </div>
         <div class="col span-1-of-5 box">
        <h3>Kherigarh</h3>
           <p>Cow's Info</p>
         <a class="btn btn-full" href="#">Get Info</a>

        </div>
         <div class="col span-1-of-5 box">
        <h3>Rathi</h3>
           <p> Cow's Info</p>

         <a class="btn btn-full" href="#">Get Info</a>
        </div>
       </div>
  </section>

  </body>

</html>
```

// Breed_Info

```
<!DOCTYPE html>
<html lang="en">

  <head>
  <meta name="viewport" content="width=device-width , initial-scale=1.0">

  <link rel="stylesheet" type="text/css" href="Vendors/CSS/normalize.css">
   <link rel="stylesheet" type="text/css" href="Vendors/CSS/038%20Grid.css">
  <link rel="stylesheet" type="text/css" href="Resources/CSS/quaries.css">
  <link rel="stylesheet" type="text/css" href="Resources/CSS/style_breed.css">
  <link href="https://fonts.googleapis.com/css?family=Montserrat&display=swap" rel="stylesheet">
  <style>
     /* Set the size of the div element that contains the map */
```

```
      #map {
        height: 400px;  /* The height is 400 pixels */
        width: 100%;  /* The width is the width of the web page */
       }
    </style>

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
  google.charts.load('current', {'packages':['gauge']});
  google.charts.setOnLoadCallback(drawChart);

  function drawChart() {

    var data = google.visualization.arrayToDataTable([
      ['Label', 'Value'],
      ['Temperature', 25.2],
      ['Humidity', 49],
      ['THI', 68.5]
    ]);

    var options = {
      width: 400, height: 120,
      redFrom: 88, redTo: 100,
      yellowFrom:72, yellowTo: 88,
      minorTicks: 5
    };

    var chart = new google.visualization.Gauge(document.getElementById('chart_div'));

    chart.draw(data, options);

    setInterval(function() {
      data.setValue(0, 1, 40 + Math.round(60 * Math.random()));
      chart.draw(data, options);
    }, 13000);
    setInterval(function() {
      data.setValue(1, 1, 40 + Math.round(60 * Math.random()));
```

```
      chart.draw(data, options);
    }, 5000);
    setInterval(function() {
      data.setValue(2, 1, 60 + Math.round(20 * Math.random()));
      chart.draw(data, options);
    }, 26000);
  }
</script>

      <title>Breed 1</title>
</head>

   <body>

   <header>

      <div class="Header">
         <a href="homepage.html"><h2>Livestock Monitoring System</h2></a>
      </div>
   </header>

   <h3>Ongole</h3><br>
   <h4 alignment = center;>Ongole is a prominent dual purpose breed of Andhra Pradesh. The breed got its name from its geographical area of origin, i.e. Ongole. The breed is also known as “Nellore” as the Ongole area was earlier in Nellore district.The breed is known for hardiness, disease resistance and capacity to thrive on scanty resources.</h4>
   <br>
   <div id="chart_div" style="width: 400px; height: 120px;"></div>
   <br>
   <div class="First_row">
      <div class="head_box">
       <h4>Updates</h4>
      </div>
      <div class="content_box">
         <p>THI Value:<span id="thiValue">Loading...</span><br><br><br>
         <b>Status:</b><span id='condition'> status of cattle with reference to THI</span></b></p>
      </div>
```

```
    <div class="head_box">
      <h4>Information</h4>
    </div>
    <div class="content_box">
      <p>

      </p>
    </div>
  </div>
  <div class="Second_row">
      <div class="head_box_r">
      <h4>Food</h4>
      </div>
      <div class="content_box_r">
      <p>Grain: 2.7 kg pounds<br>Hay: 4.08 kg<br>Protein supplement: 0.23 kg<br>Mineral block: as much as needed<br>Water: 20-24 gal per day</p>
      </div>
      <div class="head_box_r">
      <h4>Statistics</h4>
      </div>
      <div class="content_box_r_2">
        <h4>Stats</h4>
        <p>Age while calving: <span id="ageCalv1"></span> months</p>
        <p> Calving Interval: <span id="calvInter1"></span> months</p>
        <p>lactation Period: <span id="lactPeriod1"></span> days</p>
        <p>Milk Fat: <span id="milkFat1"></span> %</p>
        <p>Milk Yield per day: <span id="myDay1"></span> kg</p>
        <p>Milk Yield per Lactation: <span id="myLactation1"></span> kg</p>
        <p>Milk Yield per month: <span id="myMonth1"></span> kg</p>
        <p>Milk Yield per year: <span id="myYear1"></span> kg</p>
        <p>Number of milking per day: <span id="nmpd1"></span> times</p>
        <p>Weight(kg):<span id="weight1"></span> kg</p>
      </div>
  </div>
    <div class="Map_box">
      <div id="map"></div>
      </p>
```

```
    </div>




<script src="https://www.gstatic.com/firebasejs/7.8.1/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
    https://firebase.google.com/docs/web/setup#available-libraries -->
<script src="https://www.gstatic.com/firebasejs/7.8.1/firebase-database.js"></script>
<script>
    // Your web app's Firebase configuration
    var firebaseConfig = {
        apiKey: "AIzaSyBeeWcGeSlkQEcjKOFpPnVzn6bbYFsOXyc",
        authDomain: "livestock-monitoring-system.firebaseapp.com",
        databaseURL: "https://livestock-monitoring-system.firebaseio.com",
        projectId: "livestock-monitoring-system",
        storageBucket: "livestock-monitoring-system.appspot.com",
        messagingSenderId: "31103750254",
        appId: "1:31103750254:web:e12b16e888c308cef1567f",
        measurementId: "G-3E7V8NDEWF"
    };
    // Initialize Firebase
    firebase.initializeApp(firebaseConfig);
</script>
<script>
    // Initialize and add the map
    function initMap() {


        firebase.database().ref('GPS').once("value", snapshot => {
            if(snapshot && snapshot.val()) {

                let GPS = snapshot.val();

                // The location of Uluru
```

```
                var uluru = {lat: parseFloat(GPS.Latitude), lng: parseFloat(GPS.Longitude)};
                // The map, centered at Uluru
                var map = new google.maps.Map(
                document.getElementById('map'), {zoom: 4, center: uluru});
                // The marker, positioned at Uluru
                var marker = new google.maps.Marker({position: uluru, map: map});
            }
        });

    }
    </script>
    <!--Load the API from the specified URL
    * The async attribute allows the browser to render the page while the API loads
    * The key parameter will contain your own API key (which is not needed for this tutorial)
    * The callback parameter executes the initMap() function
    -->
    <script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyC_DSnUPH4VHzO0Wv3pdRFOvLQ-zdTHRYQ
&callback=initMap">
    </script>

  <script src="./app.js"></script>
</body>

</html>
```

Breed_Page Styling Sheet

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

```
html {

   background: #fff;
   color: #555;
   font-family: 'Montserrat', sans-serif , 'Arial';
   font-size: 20px;
   font-weight: 300;
   text-rendering: optimizeLegibility;
}
h1,h2,h3
{
   text-transform: uppercase;

}
h2{
   font-size: 180%;
   word-spacing: 2px;
   text-align: center;
   margin-bottom: 30px;
   margin-top: 30px;
   letter-spacing: 1px;

}

h3 {
    word-spacing: 2px;
   text-align: center;
   margin-top: 30px;
   letter-spacing: 1px;
}

h3:after {
   display: block;
   height: 2px;
   background-color: #27ae60;
   content: " ";
   width: 120px;
```

```
  margin:  0 auto;
  margin-top: 10px;
}

.Header {
 border-bottom: 15px solid #27ae60;
}

.head_box {
  width: 500px;
 border: 1px solid  #555;
 background-color: #555;
 color : #fff;
 padding: 20px;
 margin-bottom: 0;
  margin-top: 5%;
  margin-left: 150px;
  text-align: center;
}

.content_box {
  width: 500px;
 border: 1px solid  #555;
 padding: 100px 10px;
 margin-top: 0;
  margin-left: 150px;
  margin-bottom: 30px;
  text-align: center;



}

.head_box_r {
  width: 500px;
 border: 1px solid  #555;
 background-color: #555;
 color : #fff;
```

```
  padding: 20px;
  margin-bottom: 0;
    margin-top: 5%;
    margin-right: 150px;
    text-align: center;
}

.content_box_r {
    width: 500px;
  border: 1px solid  #555;
  padding: 100px 10px;
  margin-top: 0;
    margin-right: 150px;
    margin-bottom: 30px;
    text-align: center;
    padding-bottom: 22.5%;

}
.content_box_r_2 {
  width: 500px;
border: 1px solid  #555;
padding: 50px 5px;
margin-top: 0;
  margin-right: 150px;
  margin-bottom: 30px;
  text-align: center;
  padding-bottom: 8.9%;

}

.Map_box {
  height: 500px;  /* The height is 400 pixels */
  width: 90%;  /* The width is the width of the web page */
  margin-left: 5%;


 }
```

```
.head_box_map {
    width: 90%;
    height: 100px;
  border: 1px solid  #555;
  padding: 20px;
    margin-bottom: 0;
    margin-top: 45%;
    margin-left: 150px;
    margin-right: 150px;
    text-align: center;
}

.content_box_map {
    width: 1360px;
    height: 600px;
  border: 1px solid  #555;
  padding: 120px 120px;
  margin-top: 0;
    margin-left: 150px;
    margin-bottom: 30px;
    text-align: center;

}


.First_row {
    float: left;
}

.Second_row {
    float: right;
}
```

4.3.2. Back End Web Page

// Fetching Firebase data to Webpage using JavaScript

```
console.log("App.js included");
var thi = null;
firebase.database().ref('DHT').on("value", function(snapshot) {

  if(snapshot) {
    let DHT = snapshot.val();
    let t = DHT.Temperature;
    let h = DHT.Humidity;
    console.log(t);
    console.log(h);
    thi = (0.8*t) + ((h/100)*(t-14.4))+ 46.4;
    thi = thi.toFixed(2);
    document.getElementById('thiValue').innerHTML = thi.toString();

    if (thi<70)
    {
      let condition = "No heat stress, milk production normal. Can milk 3 times a day at Morning, Afternoon and Evening."
      document.getElementById('condition').innerHTML = condition.toString();
    }
    else if (thi>72 || thi<=78)
    {
      let condition = "Mild heat stress, milk production Likely to be reduced by 5-10%. Milking should be done only at morning and at evening. Offer lots of water and use a cooler"
      document.getElementById('condition').innerHTML = condition.toString();
    }
    else if (thi>78 || thi<=88)
    {
      let condition = "Severe heat stress. Cattle in no condition to produce Milk. Consult a doctor immediately to avoid serious health problems"
      document.getElementById('condition').innerHTML = condition.toString();
    }
```

```
      else if (thi>88)
      {
         let condition = "Cattle can die at any moment due to heat stroke! "
         document.getElementById('condition').innerHTML = condition.toString();
      }
      else
      {
         let condition = "No heat stress, milk production normal. Can milk 3 times a day in Morning, Afternoon and Evening."
         document.getElementById('condition').innerHTML = condition.toString();
      }


   }
});
// Database Breed1 Ongole
firebase.database().ref('Ongole').once("value", snapshot => {
   if(snapshot && snapshot.val()) {

      let Ongole = snapshot.val();
      if(window.thi) console.log(thi);
      document.getElementById('ageCalv1').innerHTML = Ongole.ageCalv;
      document.getElementById('calvInter1').innerHTML = Ongole.calvInter;
      document.getElementById('lactPeriod1').innerHTML = Ongole.lactPeriod;
      document.getElementById('milkFat1').innerHTML = Ongole.milkFat;
      document.getElementById('myDay1').innerHTML = Ongole.myDay;
      document.getElementById('myLactation1').innerHTML = Ongole.myLactation;
      document.getElementById('myMonth1').innerHTML = Ongole.myMonth;
      document.getElementById('myYear1').innerHTML = Ongole.myYear;
      document.getElementById('nmpd1').innerHTML = Ongole.nmpd;
      document.getElementById('weight1').innerHTML = Ongole.weight;
   }
});
// Database Breed2 Gir
firebase.database().ref('Ongole').once("value", snapshot => {
   if(snapshot && snapshot.val()) {
```

```
    let Gir = snapshot.val();
    if(window.thi) console.log(thi);
    document.getElementById('ageCalv2').innerHTML = Gir.ageCalv;
    document.getElementById('calvInter2').innerHTML = Gir.calvInter;
    document.getElementById('lactPeriod2').innerHTML = Gir.lactPeriod;
    document.getElementById('milkFat2').innerHTML = Gir.milkFat;
    document.getElementById('myDay2').innerHTML = Gir.myDay;
    document.getElementById('myLactation2').innerHTML = Gir.myLactation;
    document.getElementById('myMonth2').innerHTML = Gir.myMonth;
    document.getElementById('myYear2').innerHTML = Gir.myYear;
    document.getElementById('nmpd2').innerHTML = Gir.nmpd;
    document.getElementById('weight2').innerHTML = Gir.weight;
  }
});
// Database Breed3 Rathi
firebase.database().ref('Rathi').once("value", snapshot => {
  if(snapshot && snapshot.val()) {

    let Rathi = snapshot.val();
    if(window.thi) console.log(thi);
    document.getElementById('ageCalv3').innerHTML = Rathi.ageCalv;
    document.getElementById('calvInter3').innerHTML = Rathi.calvInter;
    document.getElementById('lactPeriod3').innerHTML = Rathi.lactPeriod;
    document.getElementById('milkFat3').innerHTML = Rathi.milkFat;
    document.getElementById('myDay3').innerHTML = Rathi.myDay;
    document.getElementById('myLactation3').innerHTML = Rathi.myLactation;
    document.getElementById('myMonth3').innerHTML = Rathi.myMonth;
    document.getElementById('myYear3').innerHTML = Rathi.myYear;
    document.getElementById('nmpd3').innerHTML = Rathi.nmpd;
    document.getElementById('weight3').innerHTML = Rathi.weight;
  }
});
// Database Breed4 RedSidhi
firebase.database().ref('RedSindhi').once("value", snapshot => {
  if(snapshot && snapshot.val()) {

    let RedSindhi = snapshot.val();
```

```
    if(window.thi) console.log(thi);
    document.getElementById('ageCalv4').innerHTML = RedSindhi.ageCalv;
    document.getElementById('calvInter4').innerHTML = RedSindhi.calvInter;
    document.getElementById('lactPeriod4').innerHTML = RedSindhi.lactPeriod;
    document.getElementById('milkFat4').innerHTML = RedSindhi.milkFat;
    document.getElementById('myDay4').innerHTML = RedSindhi.myDay;
    document.getElementById('myLactation4').innerHTML = RedSindhi.myLactation;
    document.getElementById('myMonth4').innerHTML = RedSindhi.myMonth;
    document.getElementById('myYear4').innerHTML = RedSindhi.myYear;
    document.getElementById('nmpd4').innerHTML = RedSindhi.nmpd;
    document.getElementById('weight4').innerHTML = RedSindhi.weight;
  }
});
// Database Breed5 Sahiwal
firebase.database().ref('Sahiwal').once("value", snapshot => {
  if(snapshot && snapshot.val()) {

    let Sahiwal = snapshot.val();
    if(window.thi) console.log(thi);
    document.getElementById('ageCalv5').innerHTML = Sahiwal.ageCalv;
    document.getElementById('calvInter5').innerHTML = Sahiwal.calvInter;
    document.getElementById('lactPeriod5').innerHTML = Sahiwal.lactPeriod;
    document.getElementById('milkFat5').innerHTML = Sahiwal.milkFat;
    document.getElementById('myDay5').innerHTML = Sahiwal.myDay;
    document.getElementById('myLactation5').innerHTML = Sahiwal.myLactation;
    document.getElementById('myMonth5').innerHTML = Sahiwal.myMonth;
    document.getElementById('myYear5').innerHTML = Sahiwal.myYear;
    document.getElementById('nmpd5').innerHTML = Sahiwal.nmpd;
    document.getElementById('weight5').innerHTML = Sahiwal.weight;
  }
});
```

// Script.js (Basic JS)

```
jQuery(document).ready(function($){
$(".main").onepage_scroll({
  sectionContainer: "section",     // sectionContainer accepts any kind of selector in case you don't want to use section
  easing: "ease",                  // Easing options accepts the CSS3 easing animation such "ease", "linear", "ease-in",
                                   // "ease-out", "ease-in-out", or even cubic bezier value such as "cubic-bezier(0.175, 0.885, 0.420, 1.310)"
  animationTime: 1000,             // AnimationTime let you define how long each section takes to animate
  pagination: true,                // You can either show or hide the pagination. Toggle true for show, false for hide.
  updateURL: false,                // Toggle this true if you want the URL to be updated automatically when the user scroll to each page.
  beforeMove: function(index) {},  // This option accepts a callback function. The function will be called before the page moves.
  afterMove: function(index) {},   // This option accepts a callback function. The function will be called after the page moves.
  loop: false,                     // You can have the page loop back to the top/bottom when the user navigates at up/down on the first/last page.
  keyboard: true,                  // You can activate the keyboard controls
  responsiveFallback: false,       // You can fallback to normal page scroll by defining the width of the browser in which
                                   // you want the responsive fallback to be triggered. For example, set this to 600 and whenever
                                   // the browser's width is less than 600, the fallback will kick in.
  direction: "vertical"            // You can now define the direction of the One Page Scroll animation. Options available are "vertical" and "horizontal". The default value is "vertical".
});
});
```

4.3.2. Back End Arduino

```
#include <FirebaseArduino.h>
#include <DHT.h>                        //DHT Library
#include <ESP8266WiFi.h>                // ESP8266 WiFi library
#define FIREBASE_HOST "livestock-monitoring-system.firebaseio.com"  //host
```

```
#define FIREBASE_AUTH "91ImeJ2JNjFXMXxAj0FhxTYdZf1cu75wA38sJGJ9"    //auth
#define FIREBASE_HOST "livestock-monitoring-system.firebaseio.com"  //firebase_host

#define WIFI_SSID "K"                    // wifi name
#define WIFI_PASSWORD "kangsan123"       //password of wifi

#define DHTPIN D4              // what digital pin we're connected to
#define DHTTYPE DHT11          // select dht type as DHT 11 or DHT22
#define relay D4
#define relay 0
DHT dht(DHTPIN, DHTTYPE);

      void setup()
      {
       Serial.begin(9600);
       delay(1000);
       WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
       Serial.print("Connecting to ");
       Serial.print(WIFI_SSID);
       while (WiFi.status() != WL_CONNECTED)
       {
        Serial.print(".");
        delay(500);
       }
       Serial.println();
       Serial.print("Connected to ");
       Serial.println(WIFI_SSID);
       Serial.print("IP Address is : ");
       Serial.println(WiFi.localIP());       //print local IP address
       Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); // connect to firebase
       dht.begin();                          //Start reading dht sensor
      }
      void loop()
      {
       float h = dht.readHumidity();       // Reading temperature or humidity
       float t = dht.readTemperature();   // Read temperature as Celsius (the default
       if (isnan(h) || isnan(t))
```

```
  {  // Check if any reads failed and exit early (to try again).
   Serial.println(F("Failed to read from DHT sensor!"));
   return;
   }

  Serial.print("Humidity: ");  Serial.print(h);
  String fireHumid = String(h) + String("%");    //convert integer humidity to string humidity
  Serial.print("%  Temperature: ");  Serial.print(t);  Serial.println("°C ");
  String fireTemp = String(t) + String("°C");    //convert integer temperature to string temperature
  delay(4000);
 Firebase.setFloat("/DHT/Humidity", h);        //setup path and send readings
 Firebase.setFloat("/DHT/Temperature", t);       //setup path and send readings

}
```

4.4. Screenshots

Fig 4.1. Screenshot of the Firebase Database

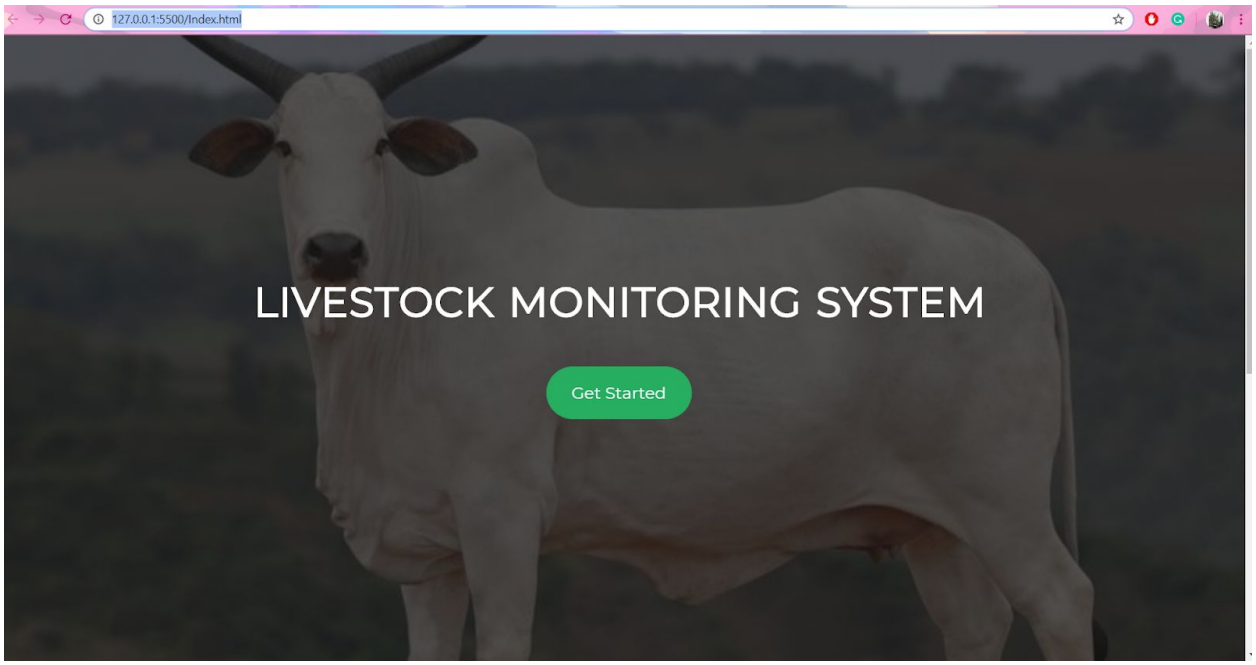

Fig 4.2. Screenshot of Index Page 1

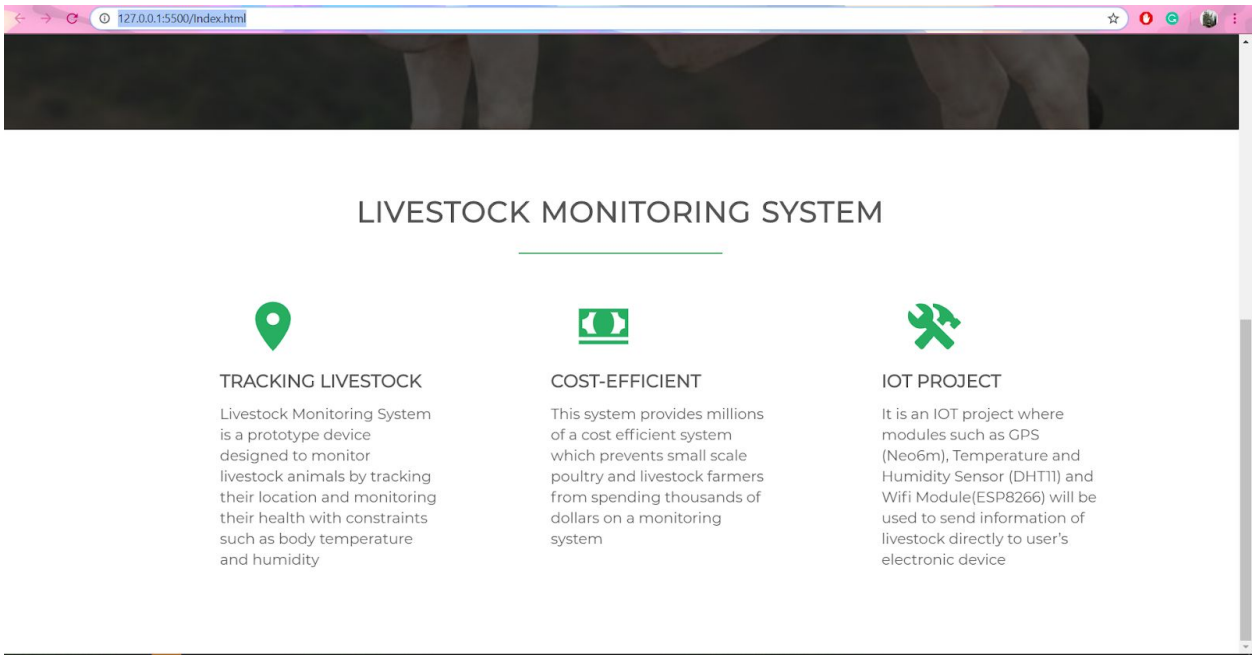

Fig 4.3. Screenshot of Index Page 2

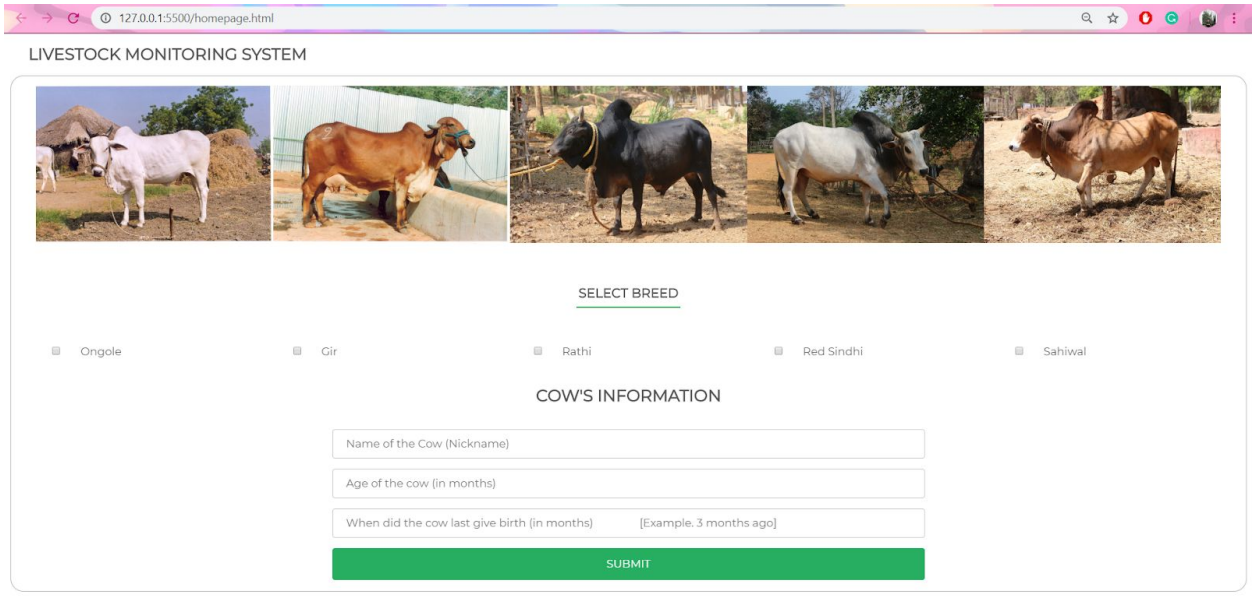

Fig 4.4. Screenshot of Homepage

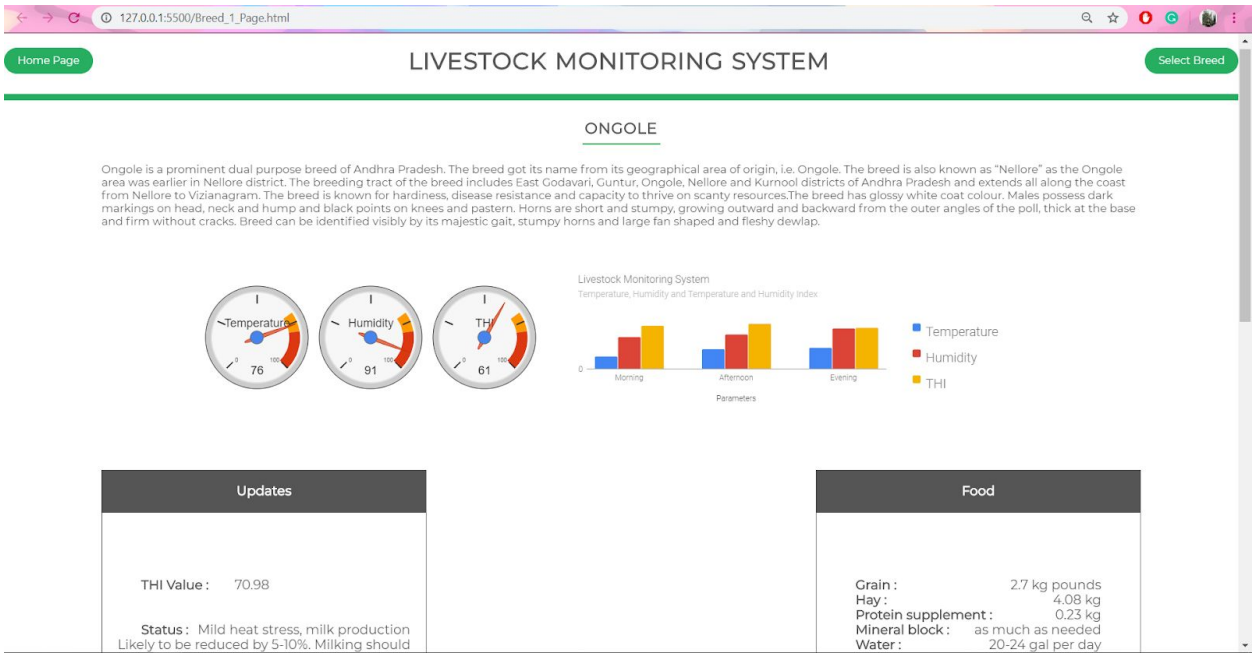

Fig 4.5. Screenshot of Main Page 1 (Breed 1)

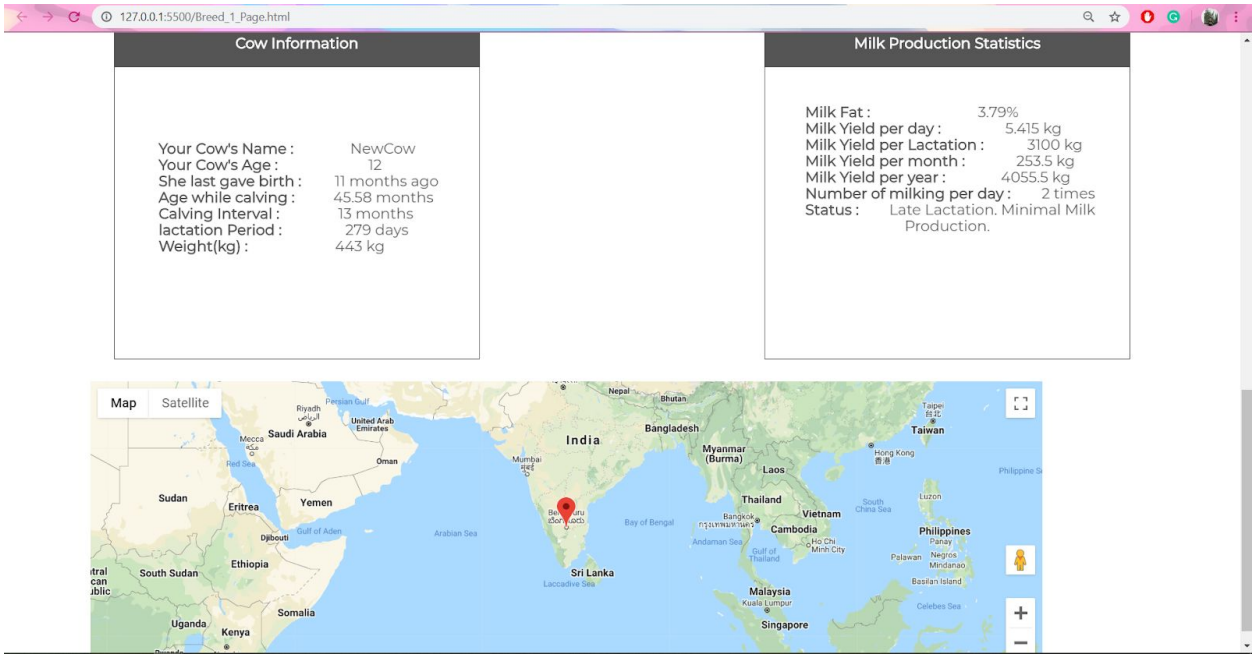

Fig 4.6. Screenshot of Main Page 2 (Breed 1)

5. TESTING

5.1. Test cases

5.1.1. Unit Testing

Unit Testing is a level of software testing where individual units/ components as a hardware or components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any hardware or software. It usually has one or a few inputs and usually a single output or in case of IoT projects can be individual devices as well.

5.1.2. Integration Testing

Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

5.1.3. Validation Testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meets the client's needs.

5.2. Test Approach

Table 5.2. Test Approach Table

SITUATION	EXPECTED	RESULT
All the IoT devices must work individually and as a complete circuit.	All the devices should successfully send data to the serial monitor of Arduino IDE.	Successfully obtained meaningful results on the serial monitor.
Data such as Temperature, Humidity, Latitude and Longitude sent to the Firebase Database through IoT Devices	Successful Transmission of Data from the IoT devices to the Firebase Realtime Database	Successful transmission of data.
Data inside the Firebase Database, Realtime or not to be fetched on webpage	Data inside the Firebase Database, Realtime and all the other data should be obtained on the console log of the web browser and then used inside the webpage.	Successfully obtained data on the console log of Google Chrome and Firefox browsers.
Selection of breeds and sending data from the Homepage to the Firebase server with proper validation.	Properly validated data such as Name, age and last parturition should be sent to the Firebase Database.	Values entered by the user were sent successfully to the Firebase Database.

Fetching Data such as Temperature, Humidity, Latitude and Longitude to the webpage from the Firebase Database to obtain Temperature and Humidity Index and Obtain Graphs and other visualizations using the values.	Google Maps and Google Charts API should be functional and values from the firebase database must be obtained to calculate other parameters. Latitude and Longitude values should be correctly coordinated on Google Maps and THI values must be shown in a Gauge meter as a form of data representation. THI values must also be computed to obtain other parameters such as number of milking and milk yield.	Successfully obtained Google Maps and Google Charts and all the other values were also successfully computed.

Multiple test cases were tried where all three (unit, integration and validation) testing strategies were used according to the context and the website displayed. As a result, all four team members got their expected output.

5.3. Test reports

5.3.1. Unit Testing

All the devices were tested individually under unit testing by both the team members and the bugs were removed to obtain successful working of data.

5.3.2. Integration Testing

After that, we undertook the Integration testing under which we integrated all the forms and checked the flow of the circuit, connectivity to the database and functionality inside the webpage to obtain flow of the software.

5.3.3. Validation testing

After that we undertook Validation testing in which the results were positive, which shows that the developed software satisfies the user's basic requirement.

6. CONCLUSION

6.1. Design and Implementation Issues

Initially the Livestock Monitoring System was designed using HC05 Bluetooth module instead of the ESP8266 WiFi module. The range of bluetooth connectivity was too low to connect the IoT Devices with the user's device. Therefore, The WiFi module offered a longer range of connectivity and also the efficiency of the module was superior to the bluetooth module and hence, the Bluetooth Module was replaced by the Wifi Module.

6.2. Future Scope of the Project

Livestock Monitoring System is designed to help the small scale dairy farmers in India. The cost of the system will not be more than Rs.650 per device, which is a small price to pay for a system which effectively helps the farmers to maximize and monitor the milk production of their cattle and a system which also prevents the cattle from getting lost by using GPS tracker. The system will also advise the farmer about the fodder and water consumption required when the cows experience heat stress and provides a sustainable income for the farmers by tackling these problems. The target audience of the project will be rural areas of India where any literate individual will be able to use this user-friendly product.

REFERENCES

1. Sommerville, Ian, Software Engineering, Addison Wesley, 9th Edition, 2010.
2. Rumbaugh, James, Object Oriented Modeling and Design, Pearson Education, New Delhi, 2005.
3. Pressman S Roger, Software Engineering A Practitioners Approach, McGraw Hill, International Editions, 7th edition, 2010.
4. 5 Functional Requirements of an IoT Platform, 5 Dec 2019, Available online at: https://medium.com/ibm-journal/5-functional-requirements-of-an-iot-platform-81eaab46712a
5. Arduino Genuino Revision 3, 10 Dec 2019, Available online at: https://www.wikidata.org/wiki/Q2581415
6. NodeMCU ESP8266, 13 Dec 2019, Available at: https://www.wikidata.org/wiki/Q21094865
7. Cowlar: How it works, 27 Dec 2019, Available at: https://www.cowlar.com/howItWorks
8. Image Processing for Location Tracking, 3 Jan 2020, Available at: https://www.eurasip.org/Proceedings/Eusipco/Eusipco2015/papers/1570102933.pdf
9. GPS Navigation Device, 7 Jan 2020, Available at: https://www.wikidata.org/wiki/Q1486497
10. THI for Cows, 11 Jan 2020, Available at: https://www.wikidata.org/wiki/Q1486497
11. Need for LMS Integrated Systems, 15 Jan 2020, Available at: https://www.sciencedirect.com/science/article/pii/S0168169996013014
12. Heat Stress in Beef Cattle, 18 Jan 2020, Available at: https://vetmed.iastate.edu/vdpam/about/production-animal-medicine/beef/bovine-disease-topics/heat-stress-beef-cattle

13. DHT11 on Arduino, 18 Jan 2020, Available at:
How to Set Up the DHT11 Humidity Sensor on an Arduino
14. Arduino to Firebase, 20 Jan 2020, Available at:
https://circuitdigest.com/microcontroller-projects/sending-temperature-and-humidity-data-to-firebase-database-using-esp8266
15. Circuit Diagram IOT, 22 Jan 2020, Available at:
https://www.circuito.io/app?components=513,10167,360216,975601