FOREST FIRE DETECTION USING LoRa MESH NETWORK

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE DEGREE
OF

BACHELOR OF TECHNOLOGY
In
ELECTRONICS & COMMUNICATION ENGINEERING

Submitted By

SATYAM SINGH [1705231047]

SHRIKRISHAN BAGHEL [1705231052]

SHUBHAM GUPTA [1705231053]

SHUBHAM JAISWAL [1705231054]

Under the Guidance of

Er. ABHISHEK SRIVASTAVA

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

INSTITUTE OF ENGINEERING & TECHNOLOGY

LUCKNOW – 226021

August 2021

CERTIFICATE

This is to certify that the seminar report entitled **"FOREST FIRE DETECTION USING LoRa MESH NETWORK"** submitted by **Mr. SATYAM SINGH (1705231047), Mr. SHRIKRISHAN BAGHEL (1705231052), Mr. SHUBHAM GUPTA (1705231053) and Mr. SHUBHAM JAISWAL (1705231054),** student of B.Tech. in Electronics and Communication Engineering, is a record of bona-fide work carried out by the candidate under my supervision in the Department of Electronics and Communication Engineering at Institute of Engineering & Technology Lucknow. The seminar report is submitted in partial fulfilment of the requirements for the award of degree of Bachelor of Technology, in Electronics and Communication Engineering from Institute of Engineering and Technology, Lucknow. To the best of my knowledge, the work carried out has not been submitted elsewhere for the award of any degree.

Er. ABHISHEK SRIVASTAVA
Department of Electronics and
Communication Engineering
Institute of Engineering and
Technology, Lucknow

Dr. NEELAM SRIVASTAVA
Head of Department
Department of Electronics and
Communication Engineering
Institute of Engineering and
Technology, Lucknow

DECLARATION

I hereby declare that the work being presented in this dissertation entitled "FOREST FIRE DETECTION USING LoRa MESH NETWORK" towards the partial fulfilment for the award of degree of Bachelor of Technology in Electronics and Instrumentation Engineering, is an authentic record of my own work carried out under the supervision and guidance of Er. ABHISHEK SRIVASTAVA Department of Electronics and Communication Engineering, IET Lucknow. The content of this dissertation, in full or in part, have not been submitted by me to any another university or institute for the award of any degree or diploma.

Date: Satyam Singh [1705231047]

Place: Lucknow Shrikrishan Baghel [1705231052]

Shubham Gupta [1705231053]

Shubham Jaaiswal [1705231054]

CONTENTS

Chapter 3:
TESTING COMPONENTS

Chapter 4:
CONNECTION WITH ESP8266

ABSTRACT

Recently California's largest wildfire destroyed several businesses and homes. The wildfire broke on 14th July this year and it consumed 2,78,000 acres. As of now, over 20,000 firefighters and support personnel are currently battling 97 large, active wildfires in 13 US states. Based on the forest inventory records, ***54.40% of forests in India are exposed to occasional fires***, 7.49% to moderately frequent fires and ***2.405 to high incidence levels*** while 35.71% of India's forests have not yet been exposed to fires of any real significance. **Precious forest resources** including carbon locked in the biomass is **lost due to forest fires** every year, which adversely impact the flow of goods and services from forests.

Forest fire or bushfire can be described as any uncontrolled and non-prescribed combustion or **burning of plants** in a natural setting such as a forest, grassland, brushland, or tundra, which consumes the natural fuels and spreads based on environmental conditions.

This project aims to provide a 24/7 on-site forest fire monitoring and detecting service. If fire is detected at early stage, the spread of fire over large area can be prevented and immediate actions can be taken to reduce the fire. The flame sensor module and CO2 gas detection module are being used to detect fire flames and increase in the concentration of CO2 gas in air due to smoke, respectively. In these mesh networks, LoRa module is being used for its long range reach. For its compact size, and many features like inbuilt Wi-Fi, we are using ESP8266 microcontroller.

ACKNOWLEDGEMENT

I thank the almighty for giving us the courage and perseverance in completing the main-project. This project itself is acknowledgements for all those people who have given us their heartfelt co-operation in making this project a grand success.

With extreme jubilance and deepest gratitude, I put my special thanks to our Project guide for her support and valuable suggestions regarding project work.

I am greatly indebted to our director **Dr. VINEET KANSAL** and our project guide **Er. ABHISHEK SRIVASTAVA**, Electronics and Communications Engineering Department, for providing valuable guidance at every stage of this project work. I am profoundly grateful towards the unmatched services rendered by her.

Our special thanks to all the faculty of Electronics and Communications Engineering and peers for their valuable advises at every stage of this work.

Last but not least, we would like to express our deep sense of gratitude and earnest thanks giving to our dear parents for their moral support and heartfelt cooperation in doing the main project.

Satyam Singh [1705231047]

Shrikrishan Baghel [1705231052]

Shubham Gupta [1705231053]

Shubham Jaiswal [1705231054]

LIST OF FIGURES

List of abbreviation and symbol used

IoT: Internet of things

ESP8266: Micro Controller (NodeMCU)

MQ-135: Smoke sensor

LoRa Module: Long Range module

DAC: Digital to analog converter

CHAPTER 1:
INTRODUCTION

INTRODUCION

The most common hazard in forests is forests fire. Forests fires are as old as the forests themselves. They pose a threat not only to the forest wealth but also to the entire regime to fauna and flora seriously disturbing the bio-diversity and the ecology and environment of a region. During summer, when there is no rain for months, the forests become littered with dry senescent leaves and twinges, which could burst into flames ignited by the slightest spark. The Himalayan forests, particularly, Garhwal Himalayas have been burning regularly during the last few summers, with colossal loss of vegetation cover of that region. Forest fire causes imbalances in nature and endangers biodiversity by reducing faunal and floral wealth. Traditional methods of fire prevention are not proving effective and it is now essential to raise public awareness on the matter, particularly among those people who live close to or in forested areas.

1.1 OBJECTIVE

The objective of this work is to design and implement an **IoT based system of Mesh Network** which is self-sustaining and would predict and detect the forest fires and sends the emergency signal or message to the respective person.

The IoT system aims to provide a 24/7 on-site forest fire monitoring and detecting service. If fire is detected at early stage, the spread of fire over large area can be prevented and immediate actions can be taken to reduce the fire. The microcontroller is connected with an IOT module, **LoRa transmitter and receiver**. This would predict and detect the forest fires and sends the exact location to concerned officials. **The main objective of this method is to detect the forest fire as early as possible and inform the concerned official.**

Every year large areas of forests are affected by fires of varying intensity and extent. Based on the forest inventory records, **54.40% of forests in India are exposed to occasional fires, 7.49% to moderately frequent fires and 2.405 to high incidence levels while 35.71% of India's forests** have not yet been exposed to fires of any real significance.

India Has Already Witnessed 3 Big Forest Fires in 2021

- A massive forest fire ripped through the Simlipal National the Park in Odisha's Mayurbhanj district on march
- In January 2021, Kullu in Himachal Pradesh raged for days before being brought under control. Forest fires were also reported in Shimla and other parts of the state.
- Earlier in January, forest fire was also reported from the Dzukou Valley on Nagaland-Manipur borders.

FIGURE 1.1: Forest Fire

1.2 CAUSE OF FOREST FIRE

Forest fires are caused by Natural causes as well as Man made causes

- Natural causes- Many forest fires start from natural causes such as lightning which set trees on fire. However, rain extinguishes such fires without causing much damage. High atmospheric temperatures and dryness (low humidity) offer favorable circumstance for a fire to start.

- Man made causes- Fire is caused when a source of fire like naked flame, cigarette or bidi, electric spark or any source of ignition comes into contact with inflammable material.

Traditionally Indian forests have been affected by fires. Themenace has been aggravated with rising human and cattle population and the consequent increase in demand for Forest products by individuals and communities. Causes of forest fires can be divided into two broad categories: environmental (which are beyond control) and human related (which are controllable).

Environmental causes are largely related to climatic conditions such as temperature, wind speed and direction, level of moisture in soil and atmosphere and duration of dry spells. Other natural causes are the friction of bamboos swaying due to high wind velocity and rolling stones that result in sparks setting off fires in highly inflammable leaf litter on the forest floor.

Human related causes result from human activity as well as methods of forest management. These can be intentional or unintentional, for example:

- graziers and gatherers of various forest products starting small fires to obtain good grazing grass as well as to facilitate gathering of minor forest produce like flowers of Madhuca indica and leaves of Diospyros melanoxylon
- the centuries old practice of shifting cultivation (especially in the North-Eastern region of India and in parts of the States of Orissa and Andhra Pradesh).
- the use of fires by villagers to ward off wild animals
- fires lit intentionally by people living around forests for recreation
- fires started accidentally by careless visitors to forests who discard cigarette butts.

1.3 EFFECT OF FOREST FIRE

Fires are a major cause of forest degradation and have wide ranging adverse ecological, economic and social impacts, including:

- loss of valuable timber resources
- degradation of catchment areas

- loss of biodiversity and extinction of plants and animals
- loss of wildlife habitat and depletion of wildlife
- loss of natural regeneration and reduction in forest cover
- global warming
- loss of carbon sink resource and increase in percentage of CO2 in atmosphere
- change in the microclimate of the area with unhealthy living conditions
- soil erosion affecting productivity of soils and production
- ozone layer depletion
- health problems leading to diseases
- loss of livelihood for tribal people and the rural poor, as approximately 300 million people are directly dependent upon collection of non-timber forest products from forest areas for their livelihood.

1.4 METHODS USED BY FSI (FOREST SURVEY OF INDIA) FOR FOREST FIRE ALARM:

FIGURE 1.2: Existing method of forest fire alarm system

FSI uses Moderate Resolution Imaging Spectro-radiometer (MODIS) data with 1km X 1km spatial resolution and Suomi-National Polar-orbiting Partnership (SNPP)-Visible Infrared Imaging Radiometer Suite (VIIRS) data with 375m X 375m spatial resolution.

	MODIS (Moderate resolution Imaging spectro-radiometer)	**SNPP-VIIRS Suomi National Polar-orbiting Partnership (NPP) satellite**
Sensor	36 spectral bands (channel 21,22,31)	5 HR Imagery channels (I-bands), 16 moderate resolution channels (M-bands) and a D/N Band (M13 and M15)
Satellite	Aqua & Terra	Suomi National Polar-orbiting Partnership (NPP) satellite
Launch	Dec 99 & May 2002	Oct-11
Algorithm	Contextual	Thresholding and Contextual (Hybrid)
Equatorial Pass	Terra- 10:30 am and 10:30 pm ; Aqua - 1:30 pm and 1:30 am	1:30pm and 1:30am
Resolution	1 km X 1km	375mx 375m &750m x 750m
Night time performance	Poor	Good
Mapping small fires	No (ideally 1000 sq m)	Yes
Accuracy of mapping large fire boundaries	Poor	Good
Under Canopy Fires detection	Poor	Good

1.5 Limitations in the existing Fire Alert System

- The algorithms cannot detect fires through thick cloud, smoke and haze. A large fire may therefore go undetected for several days and then suddenly it reappears later when the cloud cover gets removed. A small fire may burn and even die out without ever being detected.
- Same could be the case for ground or surface fires under very thick, dense canopy which could go undetected, as optical and thermal (like: MODIS and VIIRS) wavelength based sensors cannot penetrate through cloud or canopy cover. In those cases the satellite may miss the existence of fire.
- The forest fire alerts generated by FSI correspond to only 6 satellite overpasses in a day and all fires data active in between these satellite overpasses cannot be detected by the satellite based fire alert systems
- The time lapse between fire detection by the satellite and dissemination of the alert to the user is between 1 to 1.5 hours, depending on the sensor and processing time. This delay limits the utility of satellite detection for tactical fire operations.

1.6 PROPOSED SYSTEM

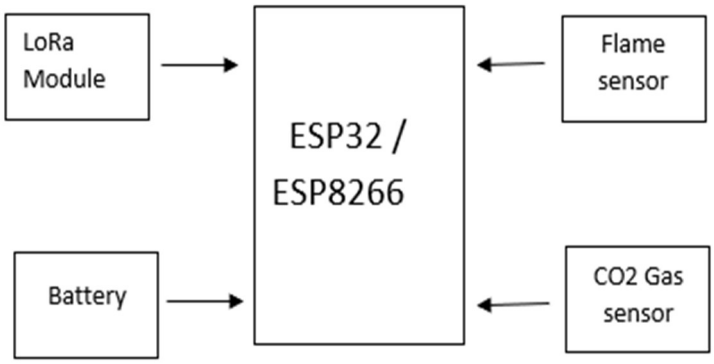

FIGURE 1.3: Transceiver Block diagram

The proposed system (device) is consists of a **flame sensor module** and **CO_2 (Carbon Dioxide)** gas detection sensor module, both are connected with a microcontroller (**ESP8266**). Flame sensor module will detect fire while CO2 gas sensor module will detect rise in the concentration of CO2 gas in the air. A **LoRa (Long Range)** module will be also attached with the microcontroller. LoRa module will help to form a **mesh network using LoRa Technology**. Since a mesh network is used, this system can be implemented in a wide range and can cover a large area of forest. LoRa is a transceiver module which send and receive data at the radio frequency.

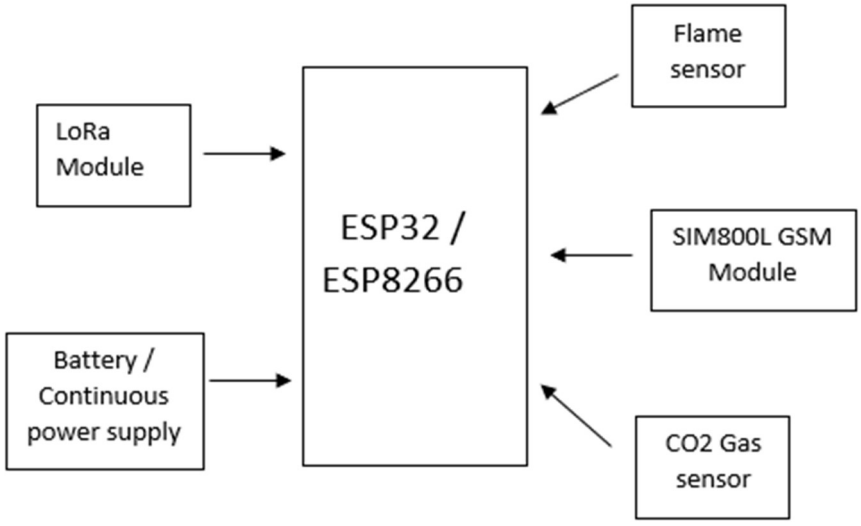

FIGURE 1.4: Master node Block diagram

CHAPTER 2:
ESP8266 AND SENSORS

2.1 Hardware Components

2.1.1 ESP8266 (NodeMCU)

NodeMCU ESP8266 Wifi Module is an open-source Lua based firmware and development board specially targeted for **IoT based applications**. It includes firmware that runs on the **ESP8266 Wi-Fi SoC** from **Espressif Systems**, and hardware which is based on the **ESP-12 module**.

FIGURE 2.1 ESP8266

Features

- Small Sized module
- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Digital I/O Pins (DIO): 16
- Flash Memory: 4 MB, SRAM: 64 KB
- Clock Speed: 80 MHz

2.1.2 Smoke Sensor (MQ-135)

The **MQ-135 Gas sensor** can detect gases like Ammonia (NH3), sulfur (S), Benzene (C6H6), CO2, and other harmful gases and smoke. Similar to other MQ series gas sensor, this sensor also has a digital and analog output pin. When the level of these gases go beyond a threshold limit in the air the digital pin goes high. This threshold value can be set by using the on-board potentiometer. The analog output pin, outputs an analog voltage which can be used to approximate the level of these gases in the atmosphere.

FIGURE 2.2: Smoke sensor

Features

- Wide detecting scope
- Fast response and High sensitivity
- Stable and long life
- Operating voltage 2.5V to 5V
- Detect/Measure NH3. NOx, alcohol, Benzene, Smoke, CO2, etc.
- The sensitivity of digital pin can be varied using the potentiometer.

2.1.3 Flame sensor

A flame-sensor is one **kind of detector** which is mainly designed for detecting as well as responding to the occurrence of a fire or flame. The flame detection response can depend on its fitting. It can be used as an alarm system, a natural gas line, propane & a fire suppression system.

FIGURE 2.3: Flame sensor

Features

- Operating voltage 3.3V-5V
- Fixed bolt holes for easy installation
- Small PCB board size 3.2cm x 1.4cm
- Use an LM393 amplifier as a voltage comparator
- The flame sensor can emit digital or analog signal.
- Detection of fire at an angle of about 60 degree, particularly sensitive to the spectrum of the flame.
- Adjustable sensitivity.
- The comparator output, clean signal, good wave driving capacity more than 15mA.

2.1.4 GSM module

SIM800L is a miniature cellular module which allows for GPRS transmission, sending and receiving SMS and making and receiving voice calls. Low cost and small footprint and quad band frequency support make this module perfect solution for any project that require long range connectivity

FIGURE 2.4: GSM Module SIM800L

Features

- Module model: SIM800L
- Quad-band 850/900/1800/1900MHz
- GPRS mobile station class B
- Supports Real time clock
- Supply voltage range 3.4V-4.4V
- Low power consumption, 1mA in sleep mode
- GPRS multi-slot class 12 connectivity: max. 85.6kbps(down-load/up-load)
- Controlled by AT (3GPP TS 27.007, 27.005 and SIMCOM enhanced AT commands.

2.1.5 LoRa 1276 module

The 868MHz SX1276 RF transceivers feature the LoRaTM long range modem that provides ultra-long range spread spectrum communication and high interference immunity whilst minimizing current consumption.

Using LoRa modulation technique it can achieve a sensitivity of over 148dBm using a low-cost crystal and bill of materials. The high sensitivity combined with the integrated +20dBm power amplifier yields industry leading link budget making it optimal for any application requiring range or robustness.

These device also support high performance (G)FSK modes for systems including WMBus, IEEE802.15.4G, The 868MHz SX1276 RF deliver exceptional phase noise, selectivity, receiver linearity and IIP3 for significantly lower current consumption than competing devices.

FIGURE 2.5: LoRa Module

2.2 SOFTWARE COMPONENTS

2.2.1 Embedded C

Embedded C is most popular programming language in software field for developing electronic gadgets. Each processor used in electronic system is associated with embedded software. Embedded C programming plays a key role in performing specific function by the processor. In day-to-day life we used many electronic devices such as mobile phone, washing machine, digital camera, etc. These all device working is based on microcontroller that are programmed by embedded C

Embedded C programming typically requires nonstandard extensions to the C language in order to support enhanced microprocessor features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations. In 2008, the C Standards Committee extended the C language to address such capabilities by providing a common standard for all implementations to adhere to. It includes a number of features not available in normal C, such as fixed-point arithmetic, named address spaces and basic I/O hardware addressing. Embedded C uses most of the syntax and semantics of standard C, e.g., main() function, variable definition, datatype declaration, conditional statements (if, switch case), loops (while, for), functions, arrays and strings, structures and union, bit operations, macros, etc.

2.2.2 Arduino IDE

The **Arduino Integrated Development Environment (IDE)** is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.

2.3 WORKING

The nodes in the mesh network placed in forest i.e. slave nodes contains flame sensor and smoke sensor. The smoke sensor is connected with analog pin of microcontroller and continuously gives the analog value of Carbon dioxide present in the air while flame sensor is connected with digital pin of microcontroller and gives digital value (if flame detected then 1 else 0). These values are sent to the master node i.e. the node containing the GSM module in a JSON message format. This message contains reading of both sensors, latitude and longitude of node and node number.

Master node continuously analyse the reading of both sensors. If the value of any sensor is above/below a certain value it means there is smoke or fire detected. Now master node will immediately send a text message to control centre. The message is a google map link containing the latitude and longitude of the node where the fire is detected.

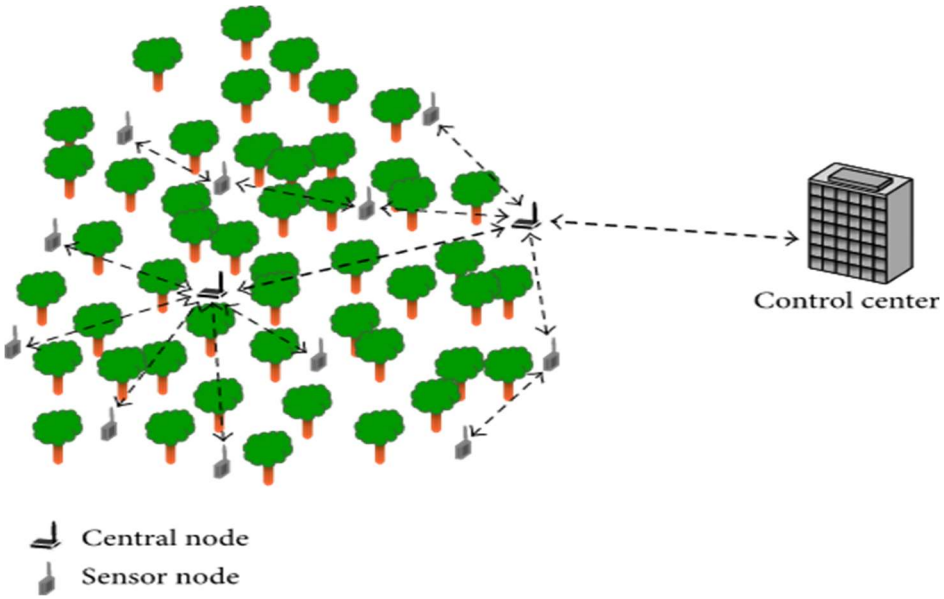

FIGURE 2.6: Working of nodes

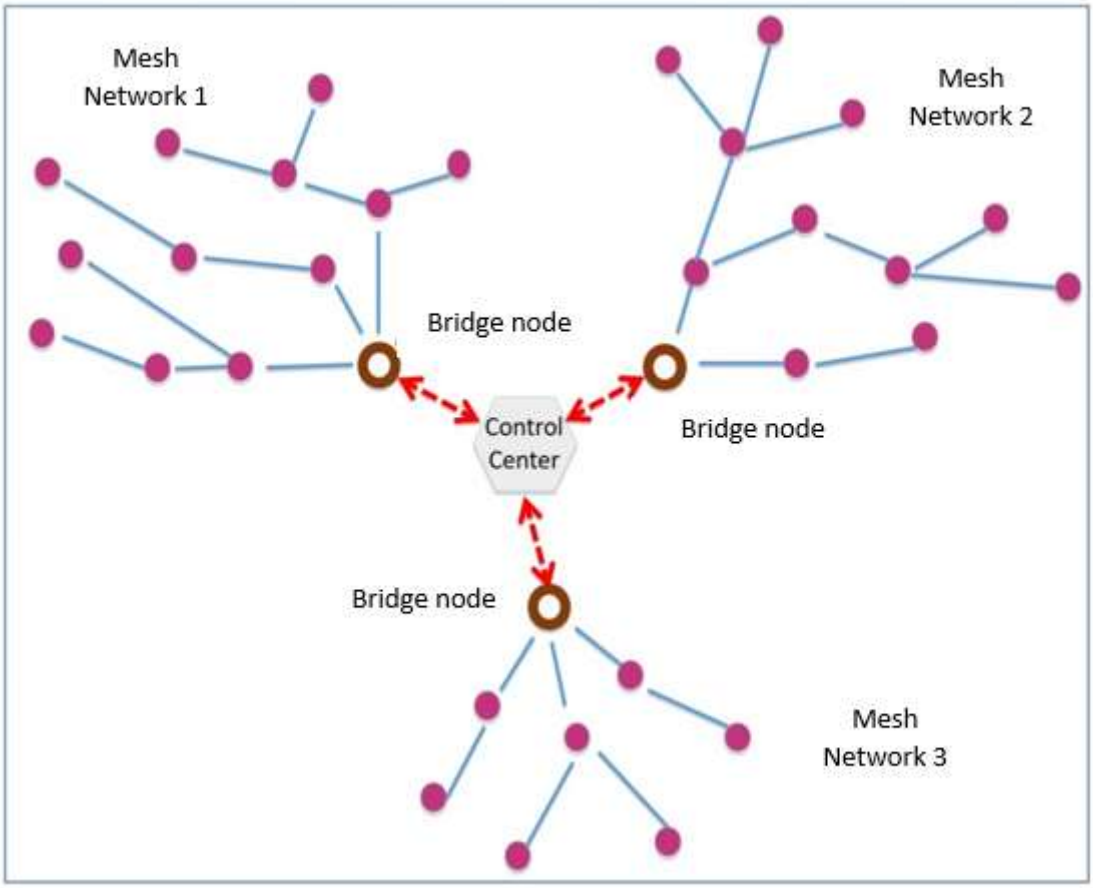

FIGURE 2.7: Mesh Network Structure

CHAPTER 3
TESTING COMPONENTS

3.1 TESTING FLAME SENSOR

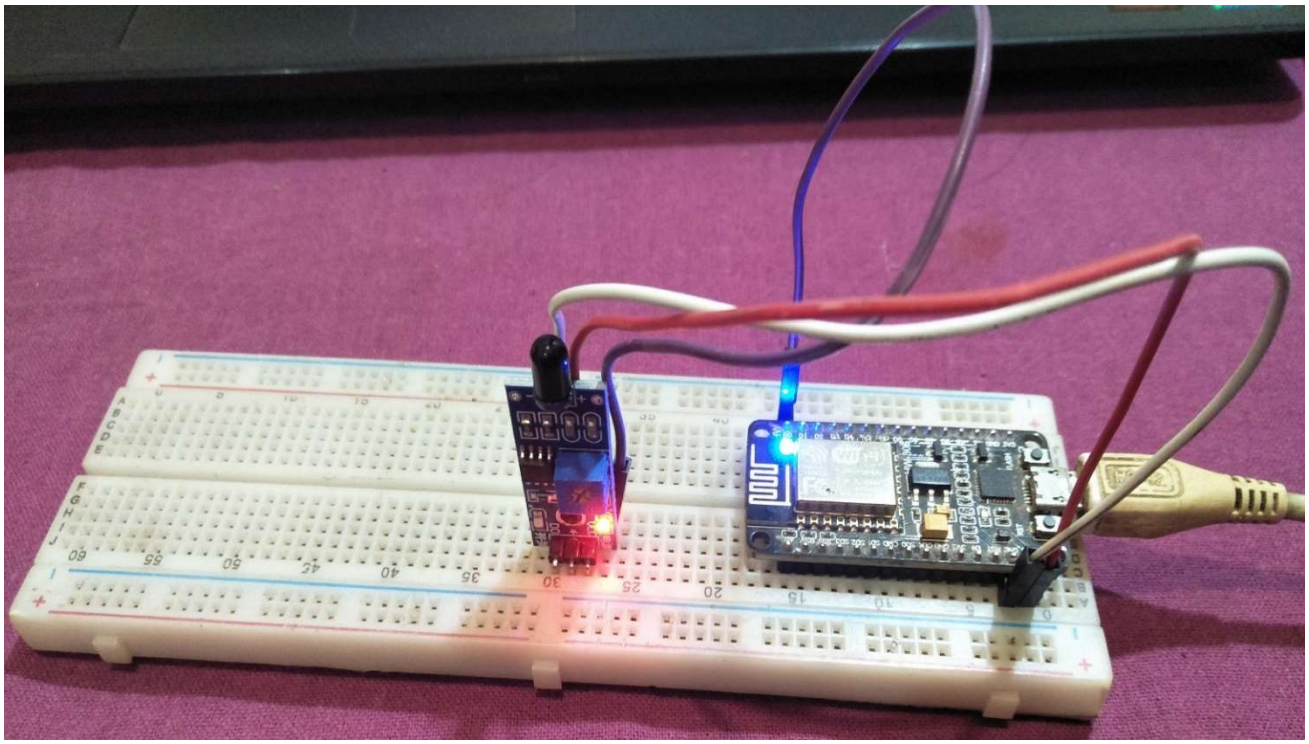

FIGURE 3.1: Testing Flame Sensor module

Flame sensor module was given 5V supply and its digital pin was connected to digital pin of NodeMCU and when a flame is brought near the sensor, the infrared rays emitting from the flame is detected by the flame sensor and state of digital pin become HIGH.

3.1.1 Test result –

Flame sensor module is working fine.

Flame detection Range – 1 meter

3.1.2 Code

```
// lowest and highest sensor readings:
#include<ESP8266WiFi.h>

void setup() {
```

```
  // initialize serial communication @ 9600 baud:
  Serial.begin(9600);
  pinMode(A0, INPUT);


}
void loop() {
  // read the sensor on analog A0:
            int sensorReading = analogRead(A0);
Serial.print("Fire sensor value is : "):
Serial.println(sensorReading);
// Read data in every one second
delay(1000);
}
```

3.2 TESTING SMOKE SENSOR

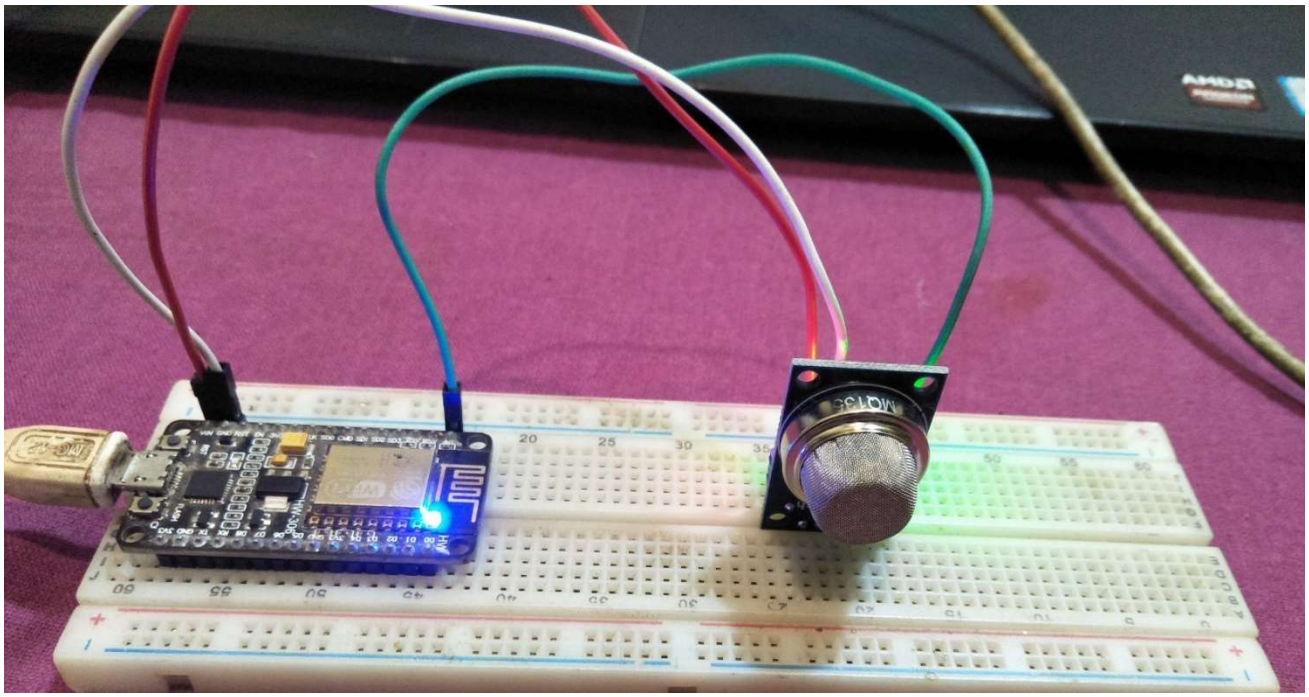

FIGURE 3.2: Testing Smoke sensor module

Smoke sensor module was given 5V supply and its Analog pin was connected to Analog pin of NodeMCU and when smoke enters in the sensor, the analog value increases.

3.2.1 Test result –

Smoke detection is successful.

3.2.2 Code

```
#include <ESP8266WiFi.h>
int smokeA0 = A0;
void setup() {
// setting baud rate to 9600bps
  Serial.begin(9600);
```

```
  pinMode(smokeA0, INPUT);
}
void loop() {
// Reading analog input
  int analogSensor = analogRead(smokeA0);
  Serial.print("Pin A0: ");
  Serial.println(analogSensor);
// Giving 1 second delay in every reading
  delay(1000);
}
```

CHAPTER 4:
CONNECTION WITH ESP8266

4.1 Connection of slave node

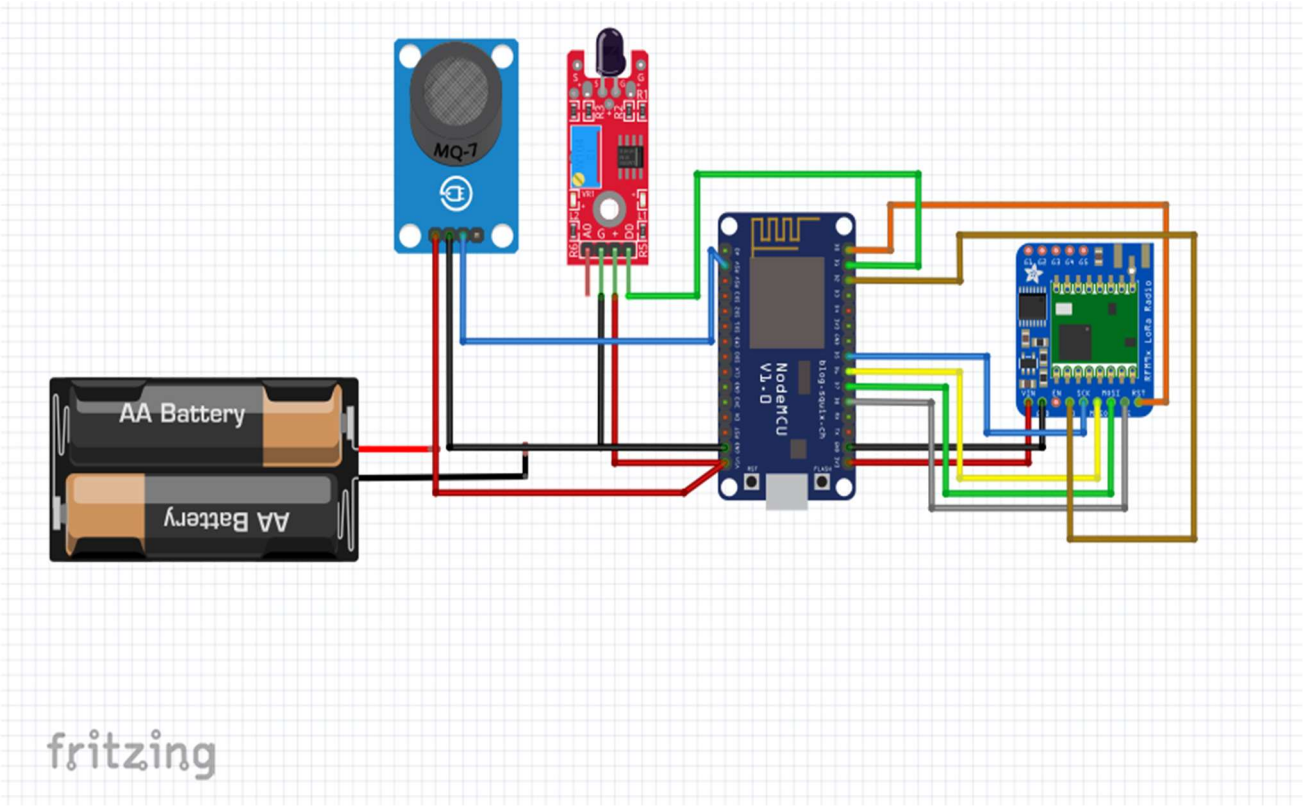

FIGURE 4.1: Pin connecting diagram of slave node

Slave nodes are the nodes which will be distributed across the forest and form a mesh network with the other slave nodes. Slave node will consist of flame sensor, smoke sensor, ESP8266 microcontroller, LoRa module and a rechargeable battery. Slave node will send data continuously to the master node through master node. Master node will consist of GSM module. Master node will analyze the slave node data and if the value of given data is more than a certain value then it means that the fire is detected and master node will send text SMS to control centre that contain the location of slave node where the fire is detected.

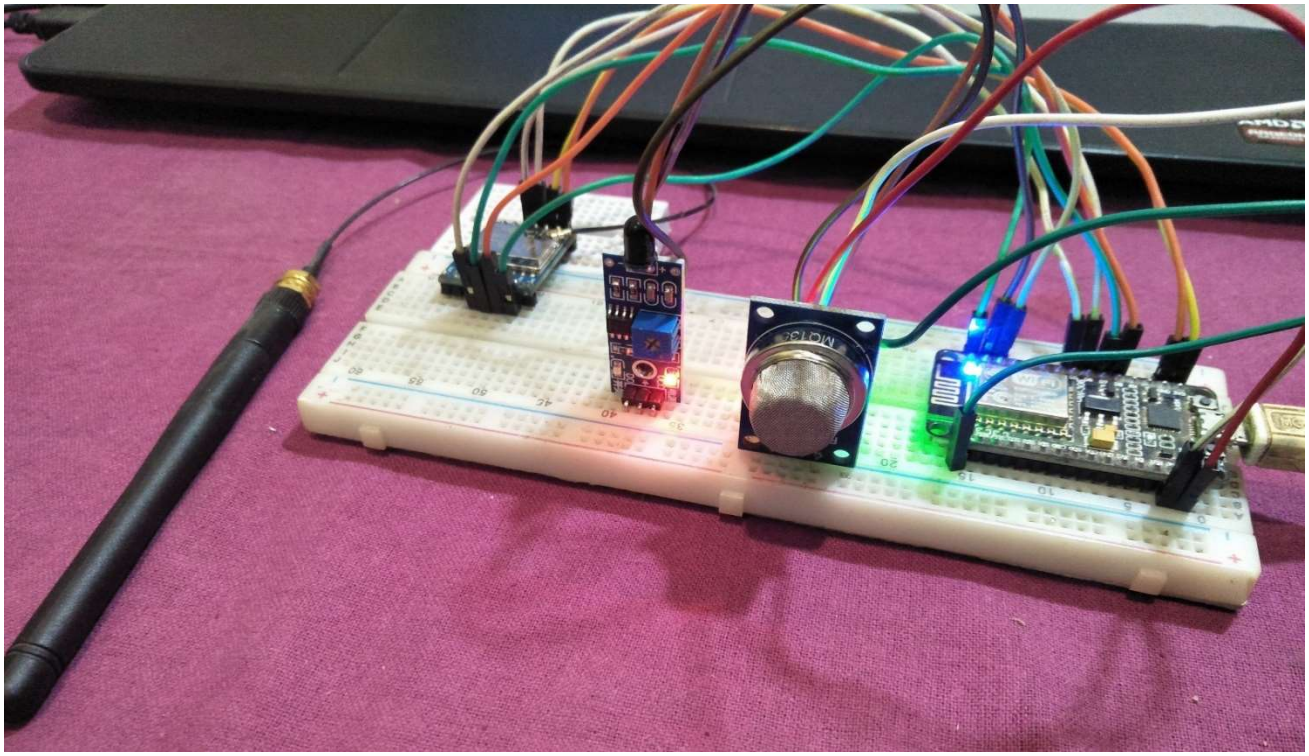

FIGURE 4.2: Testing Of slave node

4.1.1 Test Result –

- Gas sensor is detecting smoke.
- Flame sensor is detecting fire.
- LoRa Module is communicating with ESP8266 microcontroller.
- Sensor data can be seen on serial monitor of Laptop.

4.1.2 C code for slave node:

```
#include <ESP8266WiFi.h>
#include <SPI.h>      // include libraries
#include <LoRa.h>
#include <ArduinoJson.h>
//Pinout! Customized for TTGO LoRa32 V2.0 Oled Board!
#define SX1278_SCK  14    // D5, GPIO5  -- SX1278's SCK
#define SX1278_MISO 12   //D6 GPIO19 -- SX1278's MISO
#define SX1278_MOSI 13   //D7 GPIO27 -- SX1278's MOSI
#define SX1278_CS   15   //D8 GPIO18 -- SX1278's CS
#define SX1278_RST  16   //D0 GPIO14 -- SX1278's RESET
#define SX1278_DI0  4   //D2 GPIO26 -- SX1278's IRQ(Interrupt Request)

#define LORA_BAND   433E6 // LoRa Band (Europe)
#define PABOOST true
```

```
// LoRaWAN Parameters
#define TXPOWER 14
#define SPREADING_FACTOR 11
#define BANDWIDTH 125000
#define CODING_RATE 5
#define PREAMBLE_LENGTH 8
#define SYNC_WORD 0x34
#define LEDPIN 2
uint64_t chipid;
String MAC;
String Values;
char Node[4] = "002";
char Latitude[10] = "80.142654";
char Longitude[10] = "12.110365";
int Flame_value = 2;
char data[150] = "";

//////////////////////CONFIG 1//////////////////////////
byte isServer = 1;
String nodeFunction[4] = {"SINK","ESTRADA","CAMINHO","SOLTEIRO"};

byte const maxTableArrayNeighbours = 32; // number of neighbors can be increased according to available
memory
byte myNeighbours[maxTableArrayNeighbours] = {}; // address of vizinhos(neighbours) directos
byte const maxTableArrayServers = 4; // number of servers to which I have access can be increased
byte myServers[maxTableArrayServers]    = {}; // address of the servers I found
byte localAddress = 002;    // This is me!!!!! 65
byte destination = 0x1;    // Destino original broadcast (0xFF broad)

int interval = 1000;      // interval between sends
String message = "Hello";   // send a message
//////////////////////////////////////////////////////
byte msgCount     = 0;       // count of outgoing messages
long lastSendTime = 1;       // last send time
String otherValues= "";
String otherNodesMsg(String value){
 otherValues = value;
 // String JSON = "{\"T\":"+randomTemp+",\"H\":"+randomHum+"}";
  return String(value);
}
 boolean arrayIncludeElement(byte array[], byte element, byte max) {
 for (int i = 0; i < max; i++) {
     if (array[i] == element) {
        return true;
     }
   }
  return false;
```

```
 }
void arrayAddElement(byte array[], byte element, byte max) {
 for (int i = 0; i < max; i++) {
     if (array[i] == 0) {
        array[i] = element;
        return;
     }
   }
 }
void printNeighbours(){
  Serial.print("Neighbours: {");
  for (int i = 0; i < sizeof(myNeighbours); i++) {
     Serial.print(String(myNeighbours[i]));  Serial.print(" ");
    }  Serial.println("}");
 Serial.print("Sink: {");
  for (int i = 0; i < sizeof(myServers); i++) {
     Serial.print(String(myServers[i]));     Serial.print(" ");
    }  Serial.println("}");
}
void sendMessage(String outgoing, byte destination) {
  LoRa.beginPacket();                 // start packet
  LoRa.write(destination);            // add destination address
  LoRa.write(localAddress);           // add sender address
  LoRa.write(isServer);               // add server ID
  LoRa.write(msgCount);               // add message ID
 // LoRa.write(outgoing.length());       // add payload length
  LoRa.print(outgoing);               // add payload
  LoRa.endPacket();                   // finish packet and send it
  //printScreen();
   Serial.println("Send Message " + String(msgCount) + " for Node: "+ String(destination));
  Serial.println("Message: " + message);
  Serial.println();
  delay(1000);
  msgCount++;                         // increment message ID
}
void onReceive(int packetSize) {
 // Serial.println("error");
  if (packetSize == 0) return;         // if there's no packet, return
// read packet header bytes:
  int recipient = LoRa.read();         // recipient address
  byte sender = LoRa.read();           // sender address
  byte incomingMsgHand = LoRa.read();    // incoming msg ID
String incoming = "";                 // payload of packet
while (LoRa.available()) {            // can't use readString() in callback, so
   incoming += LoRa.readString();     // add bytes one by one
  }
// if the recipient isn't this device or broadcast,
```

```
  if (recipient != localAddress && recipient != 0xFF) {
    Serial.println("This message is not for me.");
    incoming = "message is not for me";
    message= incoming;
//  printScreen();
    delay(150);
    return;                       // skip rest of function
  }
// if message is for this device, or broadcast, print details:
  Serial.println("Handshake: "+String(incomingMsgHand));

  Serial.println("Received from: 0x" + String(sender, HEX));
  Serial.println("Send to: 0x" + String(recipient, HEX));
// Serial.println("Message length: " + String(incomingLength));
  Serial.println("Msg received is: " + incoming);
  //--->> Save Values--->>
  otherValues += incoming;
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
  delay(1000);
  // Positioning of servers on the mesh
   switch (incomingMsgHand) {
 case 0:
      // statements
       if(!arrayIncludeElement(myServers,sender,maxTableArrayServers)){
          Serial.println("I found a SINK! "+sender);
          arrayAddElement(myServers,sender,maxTableArrayServers);
         // display.drawString(0, 32, "NOVO: " + String(sender));
       }
      destination = sender;
      break;
   case 1:
      // statements
       if(!arrayIncludeElement(myNeighbours,sender,maxTableArrayNeighbours)){
          Serial.println("I found the freeway to SINK! "+sender);
          arrayAddElement(myNeighbours,sender,maxTableArrayNeighbours);
          //display.drawString(0, 32, "NOVO: " + String(sender));
       }
       if(isServer!=0){
         destination = sender;
       }
      break;
   case 2:
      // statements
      Serial.println("I found PATH to SINK!");
      break;
```

```
  default:
    // statements
    break;
  }
}
void configForLoRaWAN()
{
 LoRa.setTxPower(TXPOWER);
 LoRa.setSpreadingFactor(SPREADING_FACTOR);
 LoRa.setSignalBandwidth(BANDWIDTH);
 LoRa.setCodingRate4(CODING_RATE);
 LoRa.setPreambleLength(PREAMBLE_LENGTH);
 LoRa.setSyncWord(SYNC_WORD);
 LoRa.crc();
}
void makeData(){
// add some values
const size_t capacity = JSON_OBJECT_SIZE(6)+100;
   DynamicJsonDocument json1(capacity);
   Flame_value = digitalRead(5);
json1["Node"] = Node;
   json1["MAC"] = MAC;
   json1["Latitude"] = Latitude;
   json1["Longitude"] = Longitude;
   json1["Flame_Value"] = String(Flame_value);
 json1["Smoke_Value"] = String(analogRead(A0));
// serialize the array and send the result to Serial
// serialize the array and send the result to Serial
 serializeJson(json1, data);
 serializeJson(json1, Serial);
 Serial.println("");
}
void setup() {
 delay(1000);
 //makeData();
  // Use the Blue pin to signal transmission.
 pinMode(LEDPIN,OUTPUT);
 pinMode(A0, INPUT);
 pinMode(5, INPUT);
Serial.begin(9600);               // initialize serial
 while (!Serial);
 Serial.println("IPG SFarm LoRa V0.5");
LoRa.setPins(SX1278_CS, SX1278_RST, SX1278_DI0);   // set CS, reset, IRQ pin]52qw2ws
// should be done before LoRa.begin
 configForLoRaWAN();
if (!LoRa.begin(LORA_BAND))
 {          // initialize ratio at 868 MHz
```

```
   Serial.println("LoRa init failed. Check your connections.");
while (true);                    // if failed, do nothing
 }
//LoRa.onReceive(onReceive);
 LoRa.receive();
Serial.println("LoRa init succeeded.");
delay(1500);
// MAC do LoRa
chipid=ESP.getChipId();
MAC = String((uint16_t)(chipid>>32), HEX);
MAC += String((uint32_t)chipid, HEX);
Serial.print("MAC address is: ");
Serial.println(MAC);
}
String sendTable(){
 const size_t CAPACITY = JSON_ARRAY_SIZE(6);
    StaticJsonDocument<CAPACITY> doc;
    JsonArray array = doc.to<JsonArray>();
     for (int i = 0; i < sizeof(myServers); i++) {
        array.add(myServers[i]);
     }
    String Values;
    serializeJson(doc, Values);
    return Values;
}
void loop() {
  if (millis() - lastSendTime > interval) {
Serial.print("Message count is: ");
   Serial.println(msgCount);
   if(msgCount>10)
   {
    message = sendTable();
    sendMessage(message, 255);
    //<<<--- send All values of all Nodes
    //SendValues(otherValues);
    otherValues="";
    msgCount = 0;    // increment message ID
}else{
    message = data;
     if (isServer==0){
       // enviar para TTN
        digitalWrite(LEDPIN, HIGH);
       // printSensor();
       otherValues = Values;
       Serial.print("Other values are: ");
       Serial.println(otherValues);
}else{
```

```
        // send to nearest TTN at random
          makeData();
         digitalWrite(LEDPIN, LOW);
 destination = myNeighbours[0];
        sendMessage(message, 1);
        Serial.print("Message sent is: ");
        Serial.println(message);
       //  printSensor();
      }
 }
msgCount++;                    // increment message ID
 lastSendTime = millis();          // timestamp the message
   interval = random(interval) + 10000;    // 20-30 seconds
   LoRa.receive();                // go back into receive mode
  // makeData();
 }
nt packetSize = LoRa.parsePacket();
 if (packetSize) { onReceive(packetSize);  }
}
```

4.1.3 Result on Serial Monitor

These are the readings shown on serial monitor of computer

COM36

```
12:39:51.316 -> IPG SFarm LoRa V0.5
12:39:51.316 -> LoRa init succeeded.
12:39:52.811 -> MAC address is: 06a3517
12:39:52.858 -> Message count is: 0
12:39:52.858 -> {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"1"}
12:39:53.000 -> Send Message 0 for Node: 1
12:39:53.045 -> Message:
12:39:53.045 ->
12:39:53.931 -> Message sent is:
12:40:04.119 -> Message count is: 2
12:40:04.119 -> {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"3"}
12:40:04.305 -> Send Message 2 for Node: 1
12:40:04.353 -> Message: {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"1"}
12:40:04.493 ->
12:40:05.337 -> Message sent is: {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"1"}
12:40:23.336 -> Message count is: 4
12:40:23.336 -> {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"3"}
12:40:23.523 -> Send Message 4 for Node: 1
12:40:23.570 -> Message: {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"3"}
12:40:23.712 ->
12:40:24.552 -> Message sent is: {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"3"}
12:40:36.622 -> Message count is: 6
12:40:36.668 -> {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"2"}
12:40:36.856 -> Send Message 6 for Node: 1
12:40:36.856 -> Message: {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"3"}
12:40:36.996 ->
12:40:37.883 -> Message sent is: {"Node":"002","MAC":"06a3517","Latitude":"80.142654","Longitude":"12.110365","Flame_Value":"1","Smoke_Value":"3"}
```

FIGURE 4.3 Reading on serial monitor

4.2 Connection of Master node

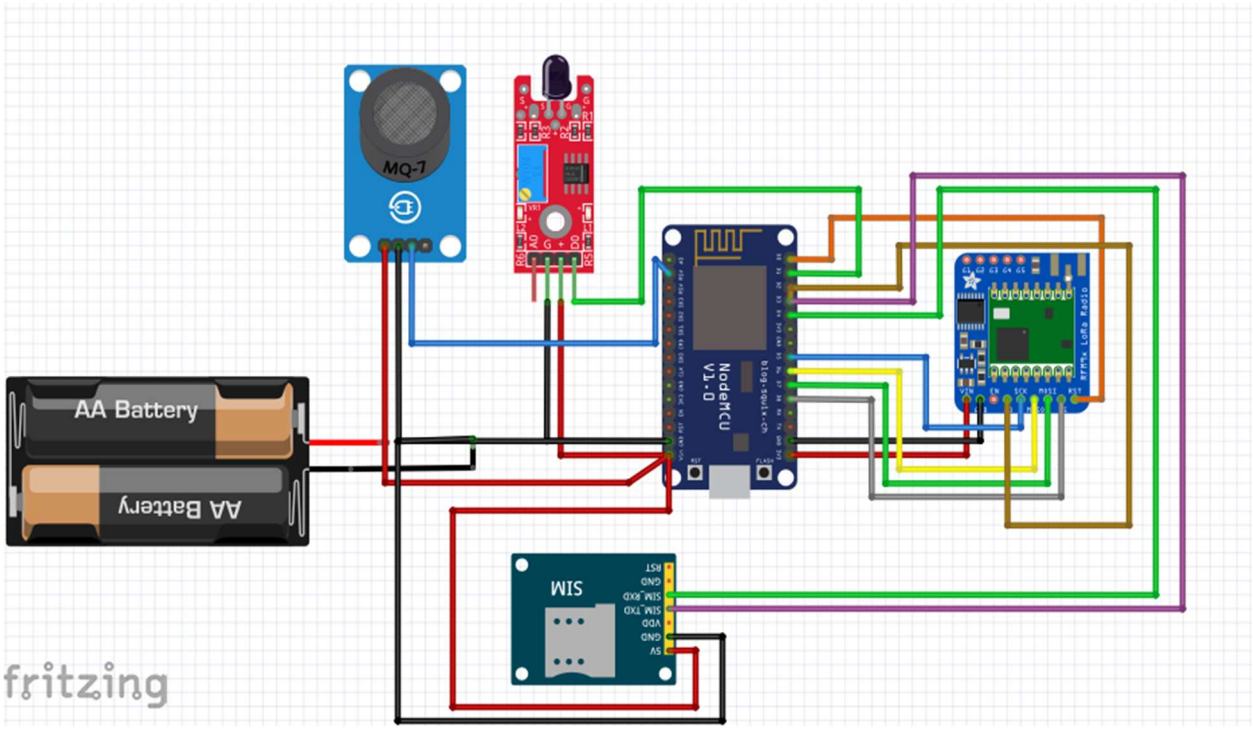

FIGURE 4.4: PIN connection Diagram of master node

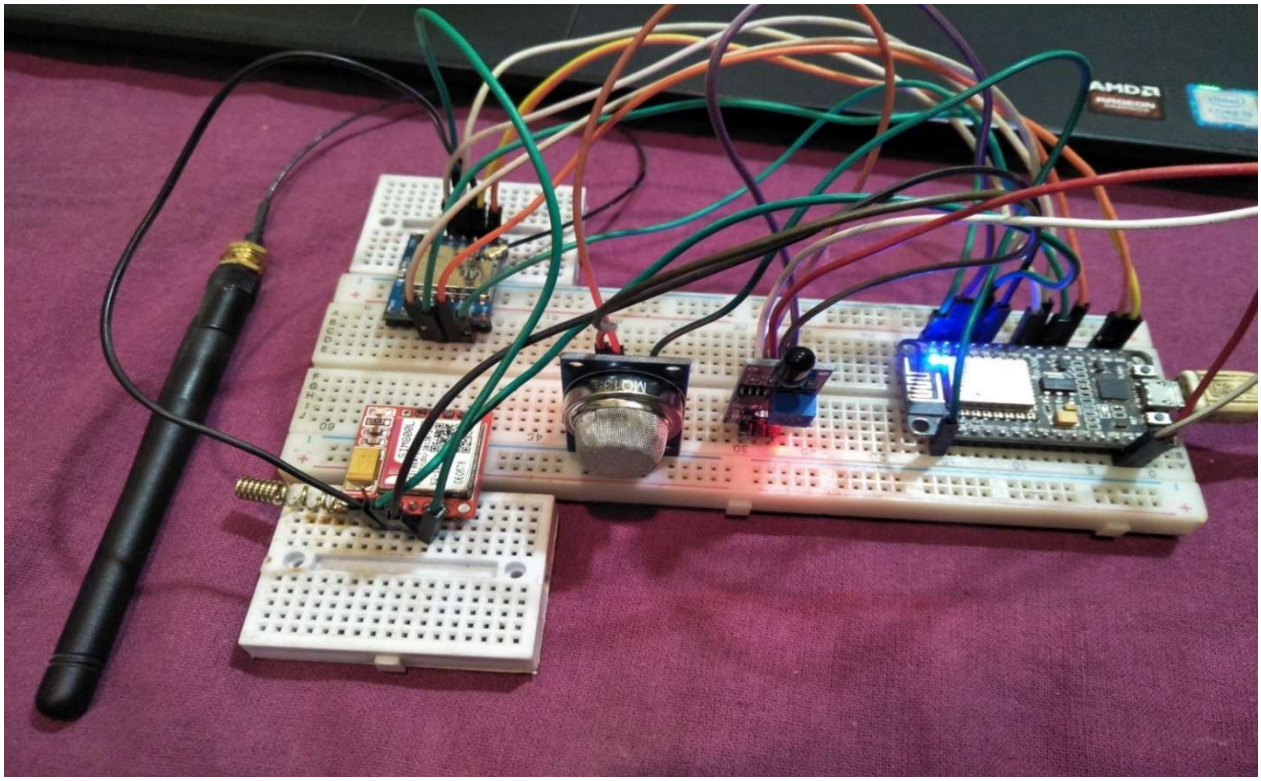

FIGURE 4.5: Testing of master node

4.2.1 Test Result-

- LoRa Module is communicating with ESP8266 microcontroller.
- GSM module is communicating with ESP8266 microcontroller.
- SIM Card is connected to network.
- Sending of text SMS through GSM module is successful.

4.2.2 C code for master node

```
#include <ESP8266WiFi.h>
#include <SPI.h>      // include libraries
#include <LoRa.h>
#include "SSD1306.h"
#include <ArduinoJson.h>
#include <string.h>
//Pinout! Between LoRa Ra-02 and NodeMCU
#define SX1278_SCK  14    // D5, GPIO5  -- SX1278's SCK
#define SX1278_MISO 12   //D6 GPIO19 -- SX1278's MISO
#define SX1278_MOSI 13   //D7 GPIO27 -- SX1278's MOSI
#define SX1278_CS   15   //D8 GPIO18 -- SX1278's CS
#define SX1278_RST  16   //D0 GPIO14 -- SX1278's RESET
#define SX1278_DI0  4   //D2 GPIO26 -- SX1278's IRQ(Interrupt Request)
#define LORA_BAND   433E6 // LoRa Band (Asia)
#define PABOOST true
// LoRaWAN Parameters
#define TXPOWER 14
#define SPREADING_FACTOR 12
#define BANDWIDTH 125000
#define CODING_RATE 5
#define PREAMBLE_LENGTH 8
#define SYNC_WORD 0x34
#define LEDPIN 2
String MAC;
uint64_t chipid;
char mac[10]="";
String incoming = "";
char Node[4] = "";
char Latitude[10] = "";
char Longitude[10] = "";
char Smoke_value[10] = "";
char Flame_value[10] ="";
//////////////////////CONFIG 1//////////////////////////
// This node and server
// 0 = Internet server
// 1 = Neighbor of internet server
```

```
// 2 = Neighbor with a Neighbor of an internet server
byte isServer = 0;
String nodeFunction[4] = {"SINK","ESTRADA","CAMINHO","SOLTEIRO"};
byte const maxTableArrayVizinhos = 32; // number of neighbors can be increased according to available
memory
byte myNeighbours[maxTableArrayVizinhos] = {}; // address of neighbour nodes
byte const maxTableArrayServers = 4; // number of servers to which I have access can be increased
byte myServers[maxTableArrayServers]    = {}; // address of the servers I found
byte localAddress = 001;    // This is me!!!!! 65
byte destination = 0xFF;    // Destino original broadcast (0xFF broad)
int interval = 1000;      // interval between sends
String message = "Hello";   // send a message
//////////////////////////////////////////////////////
byte msgCount     = 0;      // count of outgoing messages
long lastSendTime = 1;      // last send time
boolean arrayIncludeElement(byte array[], byte element, byte max) {
 for (int i = 0; i < max; i++) {
     if (array[i] == element) {
        return true;
     }
   }
 return false;
 }
void arrayAddElement(byte array[], byte element, byte max) {
 for (int i = 0; i < max; i++) {
     if (array[i] == 0) {
        array[i] = element;
        return;
     }
   }
 }
void printNeighbours(){
  Serial.print("Vizinhos: {");
  for (int i = 0; i < sizeof(myNeighbours); i++) {
     Serial.print(String(myNeighbours[i]));  Serial.print(" ");
   }  Serial.println("}");
Serial.print("Sink: {");
  for (int i = 0; i < sizeof(myServers); i++) {
     Serial.print(String(myServers[i]));    Serial.print(" ");
   }  Serial.println("}");
}
void sendMessage(String outgoing, byte destination) {
  LoRa.beginPacket();               // start packet
  LoRa.write(destination);            // add destination address
  LoRa.write(localAddress);           // add sender address
  LoRa.write(isServer);             // add server ID
  LoRa.write(msgCount);               // add message ID
```

```
  LoRa.write(outgoing.length());       // add payload length
  LoRa.print(outgoing);                // add payload
  LoRa.endPacket();                    // finish packet and send it
  //printScreen();
  Serial.println("Send Message " + String(msgCount) + " for Node: "+ String(destination));
  Serial.println("Message: " + message);
  Serial.println();
  delay(1000);
  msgCount++;                          // increment message ID
}
void extract(){
int opening_bracket = incoming.indexOf("{");
String extracted_string = incoming.substring(opening_bracket);
Serial.print("Extracted data is: ");
Serial.println(extracted_string);
StaticJsonDocument<256> doc1;
// Reading message arrived
     DeserializationError err = deserializeJson(doc1, extracted_string);
     if (err) {
       Serial.print(F("deserializeJson() failed with code "));
       Serial.println(err.c_str());
     }
     else {
       strcpy(Node , doc1["Node"]);
       strcpy(mac , doc1["MAC"]);
    strcpy(Latitude, doc1["Latitude"]);
       strcpy(Longitude, doc1["Longitude"]);
       strcpy(Smoke_value, doc1["Smoke_Value"]);
       strcpy(Flame_value, doc1["Flame_Value"]);
       Serial.print("Node is: ");
       Serial.print(Node);
       Serial.print(" MAC is: ");
       Serial.print(mac);
       Serial.print(" Latitude is: ");
       Serial.println(Latitude);
// write here the condition for sensing the SMS.
     }
}
void onReceive(int packetSize) {
 // Serial.println("error");
  if (packetSize == 0) return;         // if there's no packet, return
// read packet header bytes:
  int recipient = LoRa.read();         // recipient address
  byte sender = LoRa.read();           // sender address
  byte incomingMsgHand = LoRa.read();    // incoming msg ID
//  byte incomingMsgId = LoRa.read();    // incoming msg ID
 // byte incomingLength = LoRa.read();    // incoming msg length
```

```
while (LoRa.available()) {          // can't use readString() in callback, so
   //incoming += LoRa.readString();     // add bytes one by one
    incoming += (char)LoRa.read();     // add bytes one by one
 }
// if the recipient isn't this device or broadcast,
 if (recipient != localAddress && recipient != 0xFF) {
   Serial.println("This message is not for me.");
   incoming = "message is not for me";
   message = incoming;
//    printScreen();
   delay(150);
   return;                       // skip rest of function
 }


 // if message is for this device, or broadcast, print details:
Serial.println("Handshake: "+String(incomingMsgHand));
Serial.println("Received from: 0x" + String(sender, HEX));
 Serial.println("Send to: 0x" + String(recipient, HEX));
// Serial.println("Message ID: " + String(incomingMsgId));
// Serial.println("Message length: " + String(incomingLength));
 Serial.print("Msg received is: ");
 Serial.println(incoming);
//  otherValues += incoming;
 Serial.println("RSSI: " + String(LoRa.packetRssi()));
 Serial.println("Snr: " + String(LoRa.packetSnr()));
 Serial.println();
extract();
incoming="";
// Positioning of servers on the mesh
  switch (incomingMsgHand) {
  case 0:
     // statements
      if(!arrayIncludeElement(myServers,sender,maxTableArrayServers)){
        Serial.println("I found a SINK! "+sender);
        arrayAddElement(myServers,sender,maxTableArrayServers);
       // display.drawString(0, 32, "NOVO: " + String(sender));
      }
     destination = sender;
     break;
  case 1:
     // statements
      if(!arrayIncludeElement(myNeighbours,sender,maxTableArrayVizinhos)){
        Serial.println("I found the freeway to SINK! "+sender);
        arrayAddElement(myNeighbours,sender,maxTableArrayVizinhos);
        //display.drawString(0, 32, "NOVO: " + String(sender));
      }
```

```
      if(isServer!=0){
        destination = sender;
      }
      break;
  case 2:
     // statements
     Serial.println("I found PATH to SINK!");
     break;
  default:
     // statements
     break;
  }
}
void configForLoRaWAN()
{
 LoRa.setTxPower(TXPOWER);
 LoRa.setSpreadingFactor(SPREADING_FACTOR);
 LoRa.setSignalBandwidth(BANDWIDTH);
 LoRa.setCodingRate4(CODING_RATE);
 LoRa.setPreambleLength(PREAMBLE_LENGTH);
 LoRa.setSyncWord(SYNC_WORD);
 LoRa.crc();
}
const size_t CAPACITY = JSON_ARRAY_SIZE(12);
StaticJsonDocument<CAPACITY> doc;
JsonArray array = doc.to<JsonArray>();
String Values;
void makeData(){
// add some values
array.add(MAC);                          //<- Lora MAC
array.add(156);                          //<-- Timestamp
array.add(0);
 // serialize the array and send the result to Serial
// serialize the array and send the result to Serial
 serializeJson(doc, Values);
 serializeJson(doc, Serial);
 Serial.println("");
}

void setup() {
 delay(1000);
 //makeData();
 // Use the Blue pin to signal transmission.
 pinMode(LEDPIN,OUTPUT);
Serial.begin(9600);                      // initialize serial
 while (!Serial);
 Serial.println("IPG SFarm LoRa V0.5");
```

```
LoRa.setPins(SX1278_CS, SX1278_RST, SX1278_DI0);// set CS, reset, IRQ pin]52qw2ws
 // should be done before LoRa.begin
 configForLoRaWAN();
if (!LoRa.begin(LORA_BAND))
 {          // initialize ratio at 868 MHz
   Serial.println("LoRa init failed. Check your connections.");
while (true);                    // if failed, do nothing
 }
//LoRa.onReceive(onReceive);
 LoRa.receive();
Serial.println("LoRa init succeeded.");
delay(1500);
// MAC do LoRa
chipid=ESP.getChipId();
//Serial.printf("ESP32 Chip ID = %04X",(uint16_t)(chipid>>32));//print High 2 bytes
//Serial.printf("%08X\n",(uint32_t)chipid);//print Low 4bytes.
MAC = String((uint16_t)(chipid>>32), HEX);
MAC += String((uint32_t)chipid, HEX);
Serial.print("MAC address is: ");
Serial.println(MAC);
}
String sendTable(){
 const size_t CAPACITY = JSON_ARRAY_SIZE(6);
    StaticJsonDocument<CAPACITY> doc;
    JsonArray array = doc.to<JsonArray>();
      for (int i = 0; i < sizeof(myServers); i++) {
         array.add(myServers[i]);
     }
    String Values;
    serializeJson(doc, Values);
    return Values;
}
void loop() {
 if (millis() - lastSendTime > interval) {
Serial.print("Message count is: ");
    Serial.println(msgCount);
   if(msgCount>10)
   {
     message = sendTable();
     sendMessage(message, 255);
     msgCount = 0;    // increment message ID
 }else{
     message = "Hello from master";
      if (isServer==0){
        // enviar para TTN
        digitalWrite(LEDPIN, HIGH);
       // printSensor();
```

```
}else{
        // send to nearest TTN at random
         digitalWrite(LEDPIN, LOW);
        destination = myNeighbours[0];
        sendMessage(message, destination);
        Serial.println("Message sent is: ");
        Serial.print(message);
       //  printSensor();
      }
}
   msgCount++;                    // increment message ID
lastSendTime = millis();          // timestamp the message
   interval = random(interval) + 10000;    // 20-30 seconds
   LoRa.receive();                // go back into receive mode
  // makeData();
  }

  int packetSize = LoRa.parsePacket();
  if (packetSize) { onReceive(packetSize);  }
}
```

4.2.3 Received Data Packets on serial monitor from Slave nodes

These readings were shown on serial monitor of computer.

FIGURE 4.6: Reading on serial monitor

4.3 Connections and power supply

- We are providing 5V to NodeMCU via the USB cable.
- We are taking 5V for Smoke sensor module and flame sensor module from “Vin” pin of NodeMCU and 3.3V for Lora module from 3.3V pin of NodeMCU.
- SPI (Serial Peripheral Interface) pins of LoRa module are connected with SPI pins of NodeMCU.
- SIM800L GSM module is connected with digital pins of NodeMCU and supply voltage is 3.3V.
- Analog pin of smoke sensor module is connected with Analog pin(A0) of NodeMCU. DAC (Digital to analog converter) of NodeMCU provides value in 10 bits. So value of Smoke sensor reading can vary from 0 to 1023.
- Digital pin of flame sensor module is connected with digital pin of NodeMCU. It will give value either HIGH (1) or LOW (0).

4.4 Result

- LoRa module-
    - Initialization - Successful
    - Operating Frequency – 433MHz
    - Bandwidth – 125KHz

- Smoke Sensor Module Reading
    - In fresh air (No smoke) – 256 to 267
    - In presence of smoke – 290 to 360

- Flame Sensor Module Reading
    - In presence of fire - 1
    - In absence of fire – 0

- A mesh network is also creating in presence of more than 2 nodes. Every node in mesh has a sync word (0x34). With help of this sync work any node can join this mesh network working on a defined bandwidth.

- Message from every node is reaching to master node.

- The first C program will be used for slave nodes i.e. the nodes that will send data to master node. Master node is a node which will send information to control sensor when any node in the mesh detect fire or smoke.
- Master node is given a number i.e. 001 and slave nodes has numbers 002, 003, 004 and so on. These numbers are written in C program.
- Slave node will read data from smoke sensor module and flame sensor module and arrange it into JSON format so that it may be read easily by master node.
- JSON message will be sent to master node from every slave node through the mesh network at an interval of 10sec.-20sec.
- JSON message is containing the following data-
    - Node number
    - MAC address of node (In HEX form)
    - Latitude and longitude of node where node is placed
    - Smoke sensor module reading
    - Flame sensor module reading
- The JSON format message is reaching to the master node.
- Also, every node has information table of their neighbor nodes and this table is updated in every 3-4 min. so that nodes can know that if any new node is connected in mesh or any node is disconnected from the mesh.
- Master node check the readings of both the sensor modules and if the smoke sensor module reading or flame sensor module reading is greater than a certain value (this value will be given in program), it means fire or smoke is detected.
- Now master node will send a text message to control center through the GSM module.

Future Scope

- A communication system can also be established for the people who are living near forest so that they may get alert notification when fire is detected.
- All the components can be integrated in one small embedded system, it will help in reducing the cost and size of the system.
- Animal tracking system can also be connected with mesh network so that live location of animal will be known.

Conclusion

This final year project which is carried out through two semester needs a lot learning and hand on practices with microcontrollers and sensors also the knowledge of long-range communication. In the situation like covid fetching components was itself a challenge however we managed to do so. We manage to adapt ourself in these environments with the support from the our guide and department members. We had finally known concept of long-range communication with LoRa and to establish a mesh network up to first hopping using LoRa Modules.

Our project will help to reduce the number of forest fire taking across the world. It could be deployed with a little embedded system on board and will effectively response and help in reducing the frequent forest fire and will also help to save environment and animal lives.
This design requires minimal cost and embedded devices, it is relatively easy to install, manage and build.

Reference

[1] Forest fire
https://fsi.nic.in/focus-areas?pgID=focus-areas

[2] ESP8266
https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/

[3] Smoke sensor
https://components101.com/sensors/mq135-gas-sensor-for-air-quality

[4] Flame sensor
https://robu.in/product/flame-sensor-module/

[5] GSM 800L
https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/

[6] LoRa module
https://www.semtech.com/products/wireless-rf/lora-core/sx1276

[7] LoRa mesh network
https://nootropicdesign.com/projectlab/2018/10/20/lora-mesh-networking/

[8] News and updates
Inshorts