Forest Fire Detection using LoRa technology

K. Varshitha,
Dept. of Physics & Electronics
Bhavan's Vivekananda College,
Secunderabad, Telangana, India
varshithareddy464@gmail.com

M.Prasanna
Dept. of Physics & Electronics
Bhavan's Vivekananda College,
Secunderabad, Telangana, India
prasanna.elec@bhavansvc.ac.in

ABSTRACT

Forest fires are common hazards in forests, particularly in remote or unmanaged areas. Despite the investment and evolution in fire fighting techniques and equipment, it is common that forest fires destroy thousands of hectares of forest territory and endanger people's lives, homes and properties due to late detection. Fire detection should be done quickly to avoid damage for large amounts of natural/human resources.
In this article we propose a simple fire detection system using long range (LoRa) based technology. The proposed LoRa based system consists of wireless communication system and different types of sensors, and powered by a low-power battery, with an operating range of 3–12 km distance. This project aims in detection of occurrence of fire in forests without any delay using LoRa communication system.

Keywords—Internet-of-Things (IoT), LoRa technology, Arduino, sensors

INTRODUCTION

When forest fires burn, they emit large volumes of carbon dioxide gas (CO2) and raise the temperature levels; a network of Internet of Things (IoT) sensors like CO2 and temperature sensors can be used for forest fire detection. IoT sensors can operate alongside satellite and optical detection systems or form a standalone network of sensors near key strategic assets. IoT sensors can be deployed in remote areas without the need for internet, cellular/mobile or mains power.

In this project, a temperature, moisture and IR sensors are interfaced to Arduino, to detect the raised temperature and gases produced from the fire. These values are taken by the Microcontroller and are sent through a LoRa transmitter to a distant LoRa receiver connected to a NodeMCU. The system is to detect the fire as fast as possible and its exact localization and early notification to the fire units is vital.[4]

I. SYSTEM ANALYSIS

EXISTING SYSTEMS

- Fire alarm systems with buzzer
- Fire alert systems with no continuous monitoring system
- Highly expensive monitoring systems

DISADVANTAGE

- Current systems are local hence the alert is not conveyed to authorities far away. Also local system could be damaged in the fire.
- It requires man power
- Accuracy of output is less

PROPOSED SYSTEM

- Arduino UNO based fire monitoring system
- Lora based alerting system
- Sensor based monitoring
- LoRa can be used for monitoring and alerting

ADVANTAGES

- Both local and global fire alert is provided

- Very low cost implementation
- Also camera based detection can be provided as failsafe in case sensors malfunction

SYSTEM REQUIREMENTS

Arduino IDE

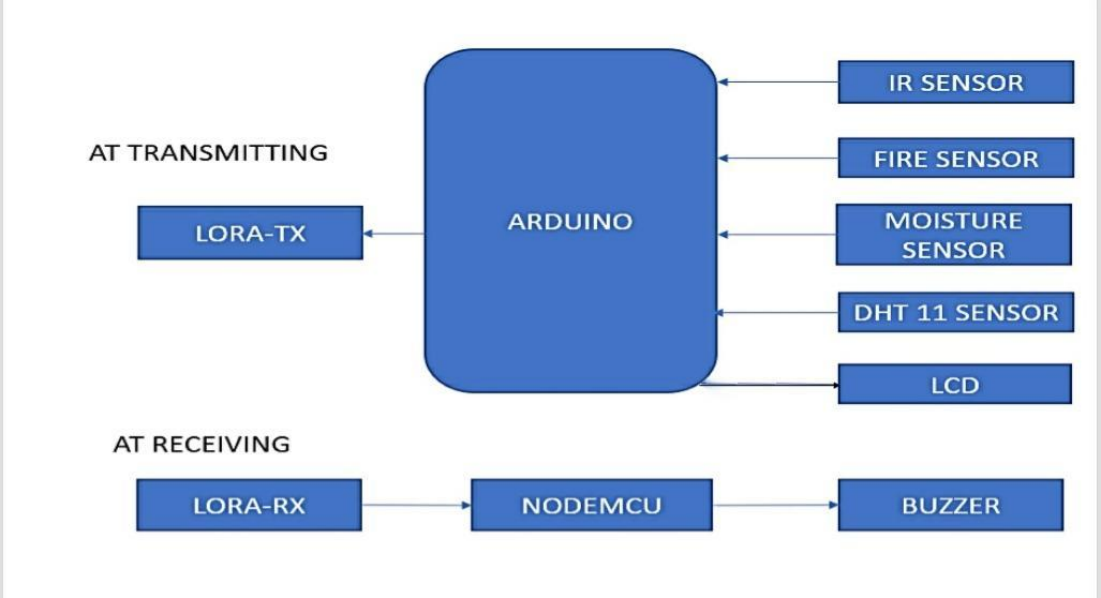


Fig 1. BLOCKDIAGRAM

II. SYSTEM DESIGN AND SOFTWARE

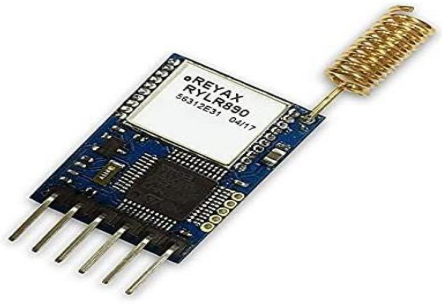
HARDWARE REQUIREMENT

- Arduino Uno
- LoRa Module
- NodeMCU
- Moisture Sensor
- IR Sensor
- Fire Sensor
- Buzzer
- I2C LCD
- DHT 11

HARDWARE DESCRIPTIONS

A. LoRa Module

LoRa is a spread spectrum modulation technique, it was developed by Cycleo (patent 9647718-B2), a company of Grenoble, France, later acquired by Semtech. Semtech's LoRa is a long range, low power wireless platform



that has become the defacto wireless platform of Internet of Things (IoT).

Fig 1: LoRa module

LoRa devices and networks such as the LoRa WAN enable smart IoT applications in many challenging areas such as: energy management, natural resource reduction, pollution control, infrastructure efficiency, and disaster prevention. Semtech's LoRa devices have amassed several hundred known use cases for smart cities, homes and buildings, communities, metering, supply chain and logistics, agriculture, and more. Semtech corporation is the leader in LoRa wireless technology and as such have introduced a number of LoRa RF modules for the market. In particular, the SX127x family of RF transceivers for the IoT/M2M markets. These RF modules operated between 860-1000 MHz and 137- 960MHz. Semtech also offers evaluation and testing devices at 860MHz band. 865 MHz to 867 MHz LoRa frequency band with 865.0625 MHz, 865.4025 MHz, 865.985 MHz frequency channels are used in INDIA.

LoRa wireless technology plays a key role in the IoT market in interconnecting devices to create smart cities, industrial and commercial solutions, whilst reducing the limitations from other wireless technologies such as power and other overheads.

B. Arduino UNO

Fig 2: Arduino Uno

The Arduino Uno is an open-source microcontroller board based on the MicrochipATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo. The board can be programmed for any application through a set of instructions to the microcontroller, using the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

C . DHT11:

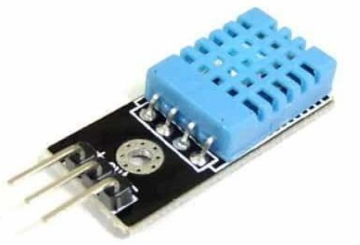


Fig 3:DHT 11 sensor

DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc… to measure humidity and temperature instantaneously.

This sensor is used here to monitor the humidity variation of the environment where the fire is sensed. This

is a digital sensor and measures the humidity value in percentage format. DHT11 humidity and temperature sensor is available as a sensor and as a module. The difference between this sensor and module is the pull-up resistor and a power-on LED. DHT11 is a relative humidity sensor. To measure the surrounding air this sensor uses a thermistor and a capacitive humidity sensor.

D. NodeMCU

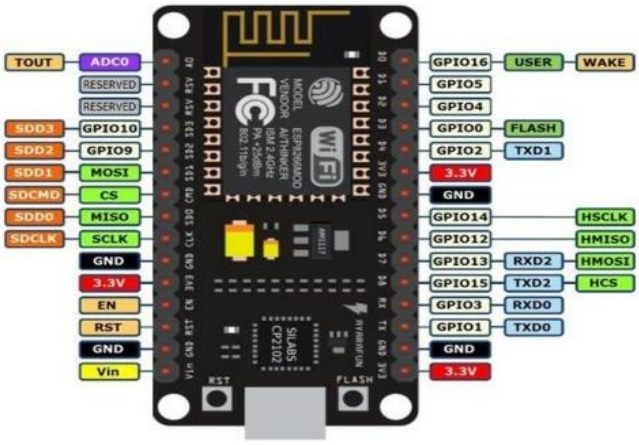
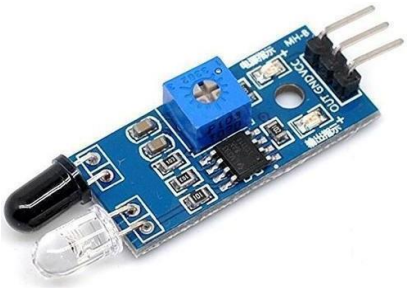


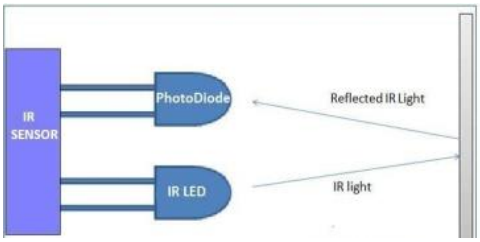
Fig 4:NodeMCU

The NodeMCU (Node MicroController Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

E. IR sensor

An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion





SOFTWARE

A. ARDUINO IDE

Download Arduino Integrated Design Environment (IDE) from : https://www.arduino.cc/en/Main/Software
Once the Arduino IDE is installed, opens into a blank sketch where we can start programming. First, we should configure the board and port settings to allow us to upload code. Connect your Arduino board to the PC via the USB cable, and next follow the below steps

- **Board Setup**
- **COM Port Setup**
- **Uploading of sketch**

Download and install Arduino IDE(https://www.arduino.cc/en/Main/Software)

1. Plug in your Arduino Board
2. Select the proper board in the IDE (Tools>Boards>Arduino Uno)
3. Select the proper COM port (Tools>Port>COMx (Arduino Uno))
4. Open the “Blink” sketch(File>Examples>Basics>01.Blink)
5. Press the Upload button to upload the program to the board
6. Confirm that your board is working as expected by observing LED

B. THINGSPEAK

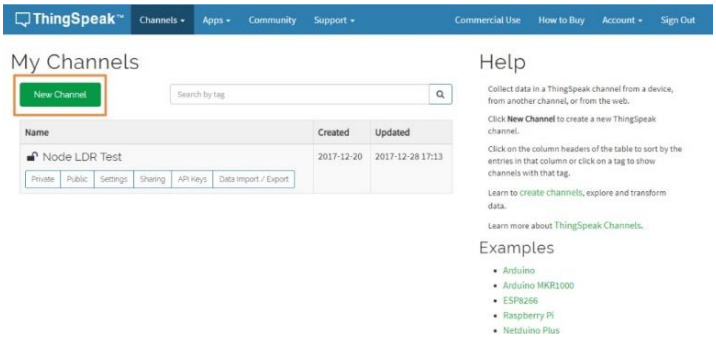


Fig 5 :creating channel

Thing Speak is an Open-Source IoT application and API to store and retrieve data from Hardware devices and Sensors and develop IoT applications. Also, the platform provides apps to analyze and visualize data. It uses HTTP Protocol over the Internet or LAN for its communication. The MATLAB analytics is included to analyze and visualize the data received from Hardware or Sensor Devices. We can create channels for each and every sensor data. These channels can be set as private channels or share the data publically through Public channels.

How to create an Account

Step 1: Go to https://thingspeak.com/and create your ThingSpeak Account if you don’t have. Login to Your Account.

Step 2: Create a Channel by clicking ’New Channel’.

Step 3: Enter the channel details.

Name: Any Name

Description: Optional

Field 1: Light Intensity LDR – This will be displayed on the analytics graph. If you need more than 1 Channels you can create for additional Sensor Data..

Step 4: Now you can see the channels. Click on the ‘API Keys’ tab. Here you will get the Channel ID and API Keys. Note this down.

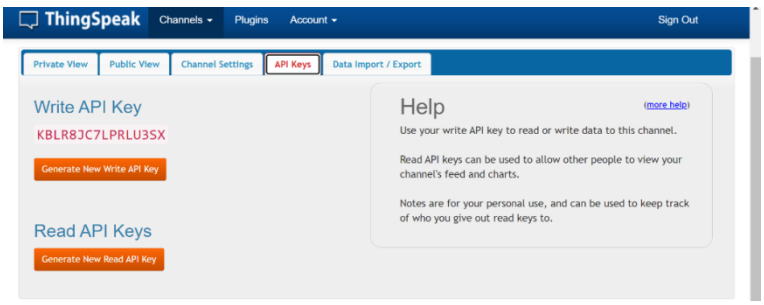


Fig6: API Keys

Step 5: Open Arduino IDE and Install the ThingSpeak Library. To do this go to Sketch>Include Library>Manage Libraries. Search for ThingSpeak and install the library.

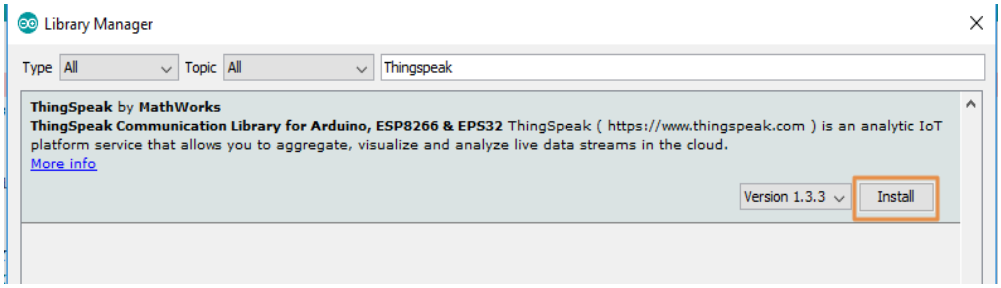


Fig7 : library installing

Step 6: Click **Save Channel** at the bottom of the settings.

III. Source Code

A. Source code to Aurdino

```
#include <SoftwareSerial.h>
#include"DHT.h"
#include<Wire.h>
#include <LiquidCrystal_I2C.h>
// Set the LCD address to 0x27 for a 16 chars and 2 line display LiquidCrystal_I2C lcd(0x27, 16, 2);
#define DHTPIN 6
#define DHTTYPE DHT11 int ir1 = 8;
int fir = 7; SoftwareSeriallora(2,3);
DHT dht (DHTPIN, DHTTYPE);

void setup()
{
Serial.begin(9600); pinMode(7, INPUT);

lora.begin(9600); dht.begin();
lcd.begin();

// Turn on the blacklight and print a message. lcd.backlight();
lcd.print("Hello, world!");
}
void loop()
{
float h = dht.readHumidity();
float t = dht.readTemperature(); float s = h + 40;
float r = t + 63;
float f = (t*1.8+32)-72; if (isnan(h) || isnan(t) ) {
Serial.println(F("Failed to read from DHT sensor!")); return;
}
int moisture = analogRead(A0);
int ir = digitalRead(ir1);
int fire = digitalRead(fir);
Serial.print("fire");
Serial.println(fire);
```

```
lcd.clear();
lcd.print("fire");
lcd.print(fire);
delay(1000);
Serial.print("ir");
Serial.println(ir);
lcd.clear();
lcd.print("ir");
lcd.print(ir);
delay(1000);
Serial.print("moisture");
Serial.println(moisture);
lcd.clear();
lcd.print("moisture");
lcd.print(moisture);
delay(1000);
Serial.print(" Humidity: ");
Serial.print(h);
lcd.clear();
lcd.print(" Humidity: ");
lcd.print(h);
delay(1000);
Serial.print(" Temperature: ");
Serial.print(f);
lcd.clear();
lcd.print(" Temperature: ");
lcd.print(f);
delay(1000);
String data = String(fire) + ":" + String(ir) + ":" + String(moisture) + ":" + String(h) + ":" + String(f);
lora.println(data);
//Serial.println(data);
delay(1000);
```

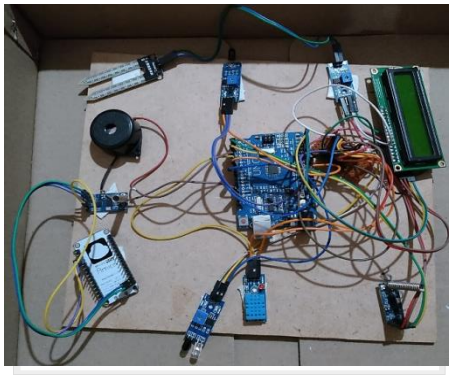
B. SOURCE CODE FOR NODEMCU

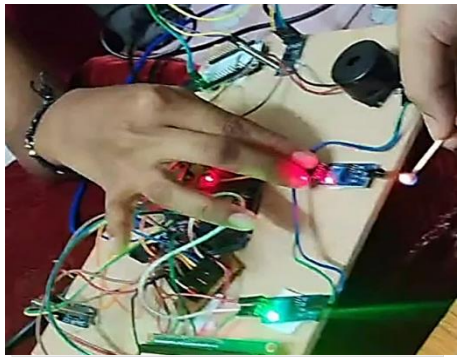
```
#define SW_VERSION " ThinkSpeak.com" // SW version will appears at innitial LCD Display

#include <SoftwareSerial.h> #include <ESP8266WiFi.h>
// The TinyGPS++ object

SoftwareSeriallora(D2, D3); // The serial connection to the GPS device const char* ssid = "Social Network";
const char* password = "PanProEdHyd&";
const char* TS_SERVER ="api.thingspeak.com";
String TS_API_KEY ="KBLR8JC7LPRLU3SX";//

WiFiClient client1;
String fire,ir,moisture,h,f;
String getStringPartByNr(String data, char separator, int index)
{
// spliting a string and return the part nr index
```

```
// split by separator
int stringData= 0;                    //variable to count data part nr
String dataPart="";                    //variable to hole the return text

for(int i = 0; i<data.length()-1;i++){

//Walk through the text one letter at a time if(data[i]==separator){
//Count the number of times separator character appears in the text stringData++;
}

else if(stringData==index) {

//get the text when separator is the rignt one dataPart.concat(data[i]);
}else if(stringData>index) {

//return text and stop if the next separator appears - to save CPU-time return dataPart;
break;
}
}
//return text if this is the last part return dataPart;
}

void sendDataTS(void)
{
if (client1.connect(TS_SERVER, 80))
{
String postStr = TS_API_KEY; postStr +="&field1=";
postStr += String(f); postStr += "&field2="; postStr += String(ir); postStr +="&field3=";
postStr += String(moisture); postStr += "&field4="; postStr += String(fire); postStr += "&field5="; postStr +=
String(h); postStr +="\r\n\r\n";
client1.print("POST /update HTTP/1.1\n"); client1.print("Host: api.thingspeak.com\n");
client1.print("Connection: close\n");
client1.print("X-THINGSPEAKAPIKEY: " + TS_API_KEY + "\n");
client1.print("Content-Type: application/x-www-form-urlencoded\n");

client1.print("Content-Length: "); client1.print(postStr.length()); client1.print("\n\n"); client1.print(postStr);
delay(1000);
}

client1.stop();

}
void setup()
{
pinMode(D5,OUTPUT); Serial.begin(9600); lora.begin(9600); Serial.println(); Serial.print("Connecting to ");
Serial.println(ssid); WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED)
{
delay(500); Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected"); Serial.println("Server started");
// Print the IP address Serial.println(WiFi.localIP()); Serial.print("Connected! IP address: ");
}
```

```
void loop()
{
if(lora.available()>0)
{
String rcv=lora.readString(); Serial.println(rcv); fire=(getStringPartByNr(rcv,':',0));
ir=(getStringPartByNr(rcv,':',1));
moisture=(getStringPartByNr(rcv,':',2));
h=(getStringPartByNr(rcv,':',3));
f=(getStringPartByNr(rcv,':',4));
}
Serial.print((" fire Value: ")); Serial.println(fire); Serial.print((" hum: ")); Serial.println(h); Serial.print((" moisture: ")); Serial.println(moisture);
Serial.print(("object: ")); Serial.println(ir);
// Serial.println(("C ")); Serial.print(("temperature: ")); Serial.println(f);
float f1 =f.toInt();

int h = moisture.toInt(); int g =fire.toInt();
int t = ir.toInt();
 if(f1>30){
Serial.println(F("::::::temperature Detected::::::::")); digitalWrite(D5,HIGH);
delay(1000);
}
elseif(g==0){

Serial.println(F("::::::fire Detected::::::::")); digitalWrite(D5,HIGH);
delay(1000);
}
else{
digitalWrite(D5,LOW); delay(1000);
}
sendDataTS();
delay(1000);
```

IV. Results





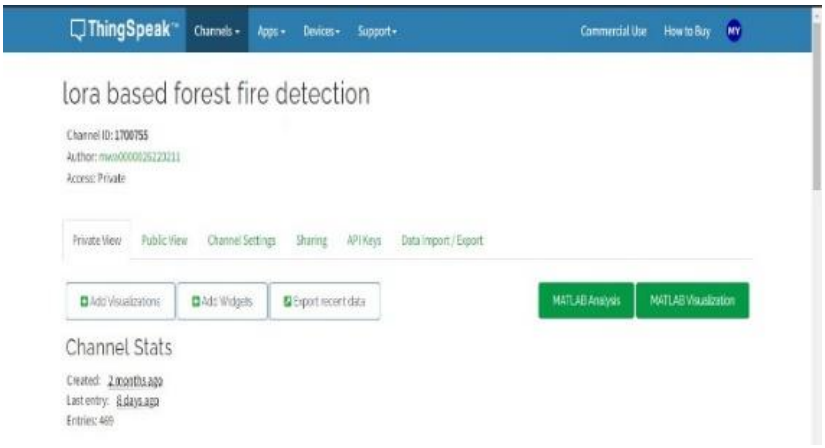


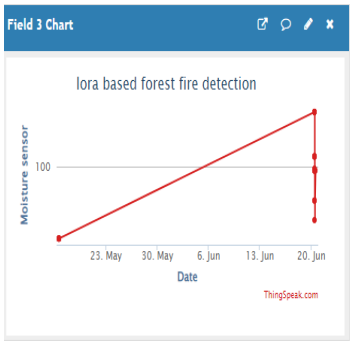
Fig 8

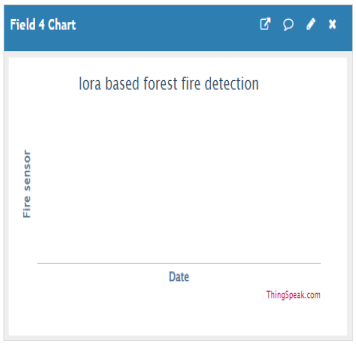
V. CONCLUSION & FUTURE SCOPE

This study described as fire monitoring system uses a wireless sensor network to inform the user remotely. This system was successfully created and implemented. The technology has been tested in a simulated fire disaster environment and has proven to be quite responsive.

Efficiency can be improved by using more accurate sensors and GPS receivers and to predict the disaster .

This system can be used in schools, colleges, offices and industries for any fire, gas leakage.





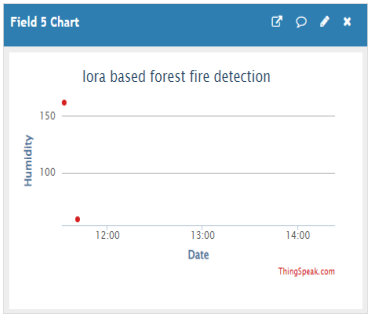


Fig 9 **Fig 10**

VI . REFERENCES

[1] K Ram Prasanna, J.M. Mathana, T. Anne Ramya et al., LoRa network based high performance forest fire detection system, Materials Today: Proceedings, https://doi.org/10.1016/j.matpr.2021.05.656

[2] FOREST FIRE DETECTION USING Lora K.Mahesh Babu, R. Priyakanth, G. Roshini, V. Saisri, BH. Keerhi priya, N. Srujana, M. Taruni Dept. of ECE 2020 JETIR May 2020, Volume 7, Issue 5

[3]Aksamovic, M. Hebibovic, and D. Boskovic, "Forest fire early detection system design utilising the WSN simulator," in *2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT)*, 2017, pp. 1-5.

[4]Y. Liu, Y. Liu, H. Xu, and K. L. Teo, "Forest fire monitoring, detection and decision making systems by wireless sensor network," in *2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 5482-5486.

[4]D. Pant, S. Verma, and P. Dhuliya, "A study on disaster detection and management using WSN in Himalayan region of Uttarakhand," in *2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA) (Fall)*, 2017, pp.1-6.

[5]https://circuitdigest.com/microcontroller-projects/iot-based-forest-fire-detection-using-arduino-and-gsm-module

[6]https://www.make-it.ca/nodemcu-details-specifications/

[7] www.jetir.org