IoT BASED FIRE & SAFETY SYSTEM FOR SMART HOMES

PROJECT FINAL REPORT

Prepared for: Dr. Maher Elshakankiri

Prepared by: Sibdow Abdul-Jalil Iddrisu (200379358)

Date: 26th August, 2019

Course Name: CS890EP – Internet of Things

University of Regina

Abstract – *Home fires have become very rampant and a very disturbing issue. They cause extreme damage including the loss of lives and property. Most home fires are caused by the carelessness or negligence of home owners and are greatly catalysed by other environmental parameters. Gas leakages and indiscriminate smoking are the major causes of such disasters and there has been a lot of sensitization and education by organizations and the fire department as to how to prevent these incidents from occurring. There has also been some technological implementation like the installation of smoke detectors in our homes but these have proven to be quite useful to only an extent. These technologies only alert when the fire has already started and probably spread over a greater perimeter of the home. A system is then required to be able to detect the possibility of this fire disaster before they even start or at least detect them early enough so action can be taken to contain it before it gets out of hand.*

Keywords: *Home Fires, Arduino, Internet of Things (IoT), Sensors, Fire Safety*

1. INTRODUCTION

1.1 What is Internet of Things (IoT)?

The Internet of Things (IoT) has revolutionized the way the internet works. This is the fourth (4th) major evolutionary phase of the internet after Connectivity (Digitized Access), Networked Economy (Digitized Business), and Immersive Experience (Digitized Interaction). It is basically the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment [1]. Apart from the network, an IoT infrastructure typically consists of sensors and actuators that helps the system interact with the environment in which it is deployed. A sensor is a device that converts or changes a physical phenomenon (like temperature, humidity, etc.) into electrical outputs. An actuator is the vice versa, in that it converts an electrical input into a physical output and can directly make changes to the environment in which they reside (like sending a signal to turn on a water pump).

In the earlier days of the internet, it was more of 'Internet of People' as stated above, where individuals connected to the internet with their devices (ex. Mobile phones, computers) to communicate and share data. With the invent of IoT, which was estimated to reach 50 billion devices by 2020, things (such as sensors and/or actuators) can be installed at various locations to monitor and transmit data to a storage autonomously without the intervention of a human agent. Some typical usage of IoT are in healthcare, smart homes, smart cities, smart grids, etc.

1.2 IoT in Smart Home Systems

The Internet of Things have become predominant in our lives and its application in our homes, popularly termed Smart Homes, have become very common and necessary; this is because it tends to increase comfort and quality of life of the people living in such a home. A Smart Home can be viewed as an environment for living that has highly advanced automatic systems [2].

Internet of Things application to Smart Homes can be generally grouped into three (3) sub-categories. Firstly, Fire and Safety; where and IoT system is deployed in a home to monitor the environment and report certain changes in certain defined parameters. A typical application would be to monitor for the presence of gases (ex. Nitrogen, CO) for air quality monitoring and be able to notify the occupants of the home if there is a danger of breathing in bad air and be able to make suggestions as to what is to be done. Also, the deploying of sensors to monitor the presence of flammable/combustible gases (like LPG – which is common in our homes) would fall under this category as they are intended to prevent fires from occurring by notifying home owners of the presence of such a gas in the air before it gets out of hand. In addition to gas monitors, flame sensors could also be deployed to aid in the detection of flame within certain ranges and be able to alert home occupants to evacuate as soon as possible before, maybe being trapped in the flame.

Secondly, Security – this is a major concern for every home owner as we would not like our hard-earned property to be stolen and we want to be safe in our homes as that is one of the main functions of a home; to keep us safe. The IoT technology can help enforce the security we need in our homes. Systems can be designed and developed to detect intrusion and unauthorized breaches to the home perimeter, this can greatly help home owners to be able to see the dangers and act proactively before things get out of hand.

Lastly, we can view IoT in Smart Homes as helping provide Control to home owners in the sense that they are able to have access to equipment in their homes and be able to control them remotely either from their cellphones, tablets or even laptops. For instance, a home owner returning from work and wants their home cool before getting there can get access to the air conditioning system in the house through the internet and switching them on before arriving home

The implementation of IoT in our homes are intended to make the home as comfortable as possible and increasing the quality of life for the better.

1.3 IoT in Home Fire Detection and Prevention

IoT provides us the necessary tools to be able to detect and prevent fire disasters from happening. There are certain parameters and phenonium that can point to the possible occurrence of fire, these parameters can range from the presence of LPG to high room temperatures. With the advent of IoT, we are presented with a myriad of sensors and technologies to help in this regard. We are afforded the opportunity to be able to use sensors to monitor these parameters and be able to see in real-time what is going on in our homes, both when we are home and away.

2. MOTIVATION & RELATED WORK

The motivation of this work is being driven by the need to improve the early detection of home fires. There have been several implementations as well as technological research conducted with the aim of detecting and preventing fire disasters from occurring so as to mitigate the damages that they come along with; loss of lives and property alike.

Ahmed Imteaj et al. [3] proposed a system which was aimed at helping the Ready-Made Garment (RMG) industry to be able to detect and control accidents in factories (like fire disasters). They made use of several sensors like the light intensity sensor for recognizing fires/flames. The sensors are placed in strategic locations within the factory so they can accurately report incidents. They used the Arduino Microcontrollers for the data aggregation from the various sensors and a central node (Raspberry Pi microcontroller) for controlling the various Arduino units. A camera was also deployed to take pictures if a sensor reports an incident, just to have a visual confirmation of the situation. Their proposed system included actuators to turn off gas outlets in case there was a fire and an automated triggering of a GSM module to send an alert to the nearby fire department.

The authors in [4] also proposed a forest fire alerting system that makes use of a GPS module to send exact coordinates. They made use of the temperature and smoke sensors to monitor the environment for changes in these parameters. When the thresholds set are exceeded, the Arduino microcontroller triggers the GPS module to generate the current coordinates of the device's location and it is then being sent to the cloud using a Wi-Fi module for internet connectivity. The data in the cloud is then consumed by a monitoring station where individuals can have access to the data and make decisions based on that.

Sourav et al. [5] designed and developed a fire detection system for smart homes that is based on IoT Data Analytics. They made use of machine learning models such as K-Nearest Neighbor and decision tree to be able to classify whether there is a fire or not and also a maybe fire situation. Their results after tests showed a very high accuracy for both models with the K-NN slightly passing the Decision tree model. An SMS alert is also triggered by their system in case there is a fire using a python module.

Fernandino S. Perilla et al. [6] conducted studies to design and integrate IoT using the Arduino Microcontroller to help detect fire occurrences and be able to give alerts to fire-fighters, inhabitants of the area and other authorities. Their design included sensors to record changes in environmental factors and send alerts to fire fighters and residence when certain thresholds were exceeded. Some sensors employed in the design included: smoke, flame, motion, temperature, humidity and gas sensors; their values recorded are sent along with GPS coordinates to the Arduino for further processing and actions. Their proposed system included an external storage for the data being generated and it stores just the values generated by the sensors so they may be accessed in the future for analytics purposes.

The authors in [10] proposed a fire safety system that could be able to detect fires before they are triggered. They base their study on the fire cracking industry where fire is easy to occur due to the abundant presence of gun powder and other easily flammable and combustible substances.

3. IoT BASED FIRE & SAFETY SYSTEM FOR SMART HOMES

In this section, we provide a description of the design and implementation of the system that would enable us to easily detect and prevent fire disasters from occurring in our homes. The system consists of sensors that are able to monitor changes in the environment in which they are being deployed. Parameters such as temperature, humidity, presence of smoke, presence of LPG and the presence of flame are monitored by the various sensors being deployed, as the presence of these parameters are a strong indication that there may be a fire around or a fire is about to start if nothing is done. These sensors are interfaced with a microcontroller, specifically the Arduino microcontroller which serves as a control and sink node for the sensors to transmit the data they generate. The Arduino microcontroller in it self is unable to send data to the internet as it is originally not built with such capabilities, so we made use of a 3rd party Wi-Fi module to help us connect to the internet gateway and be able to transmit the data being generated by the various sensors to a cloud storage through the internet. As the data is being transmitted from the local source onto the cloud, the data termed “Data in Motion” would be able to be seen by the end-user in real time and be visualized on beautiful and intuitive dashboards. Also, the data is stored in data buckets so that they may be used in the future for investigations and analytics works.

The figure below shows the block diagram for the proposed system, where we have all the sensors and output electronic units interfaced with the Arduino microcontroller, and the Arduino in turn connected to the internet gateway with the help of the Wi-Fi module to transmit data to the cloud.

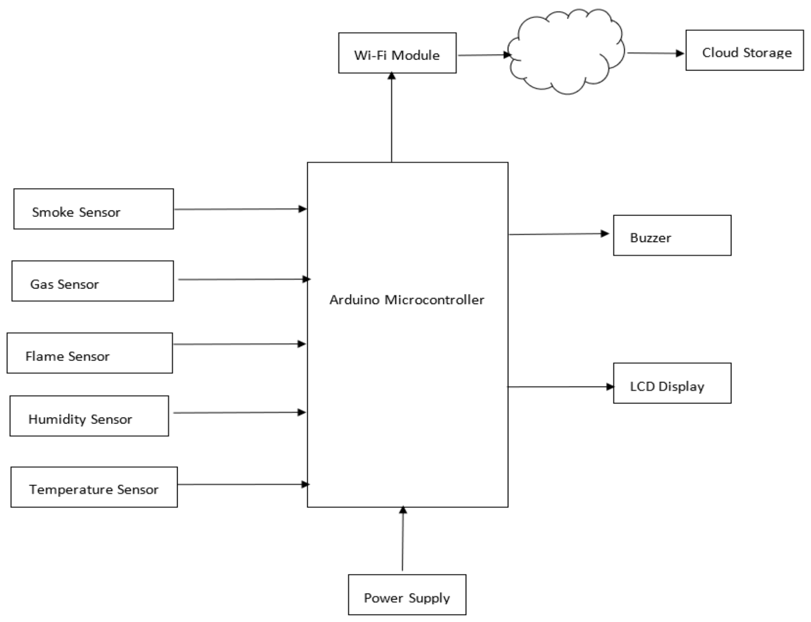


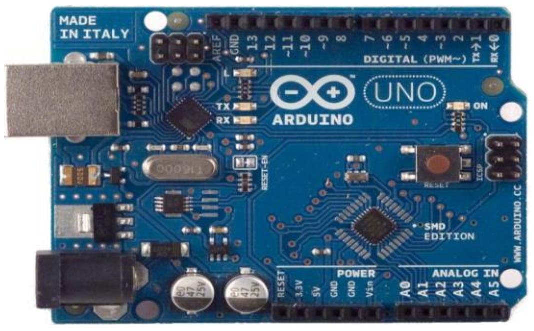
Fig. 3.1: Block Diagram of Proposed System

We made use of some hardware components and supplies in the implementation of the proposed system as well as some applications and online service. We shall use the next few paragraphs to describe and explain in details these parts and services being made use of in the project and how they were used.

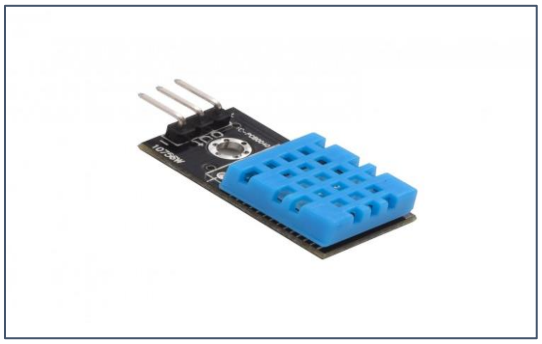
3.1 Description of Individual Hardware Components and Supplies of the Proposed System

The proposed system consists of several sensors and electronic output devices. It is also composed of a microcontroller for the control of the sensors and output units, a breadboard to help us connect the jumper wires in a clean and effective manner, a Wi-Fi module to help connect to the internet and an external cloud storage. Below is a list and description of the various tools/hardware used in our project:

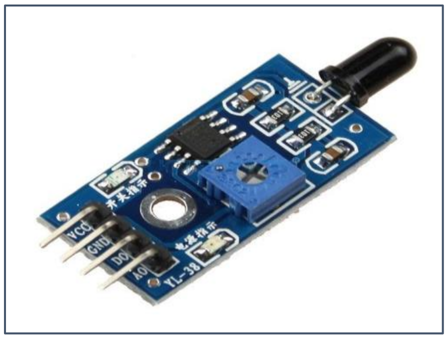
- **Arduino Uno Microcontroller Unit**: This is the main processing unit used in the project that provides a mechanism for the sensors to be interfaced with it and be able to transmit data. It is a widely used microcontroller and we choose to work with it in this project because it is a proven technology and most importantly cheap and easy to acquire. It comes with a power supply unit, memory, processor and a lot of ports for connecting and communicating with sensors and other external hardware equipment.



- **DHT11 Sensor**: This is the sensor we are using to measure the temperature and humidity values of the environment we intend to monitor. It comes with three (3) pins to connect to the Arduino. We have the VCC pin for the 5V power supply and the ground pin; along with these, we also have the data pin that is able to transmit the data recorded by the sensor to the Arduino and also be able to receive control signals from the Arduino as well.

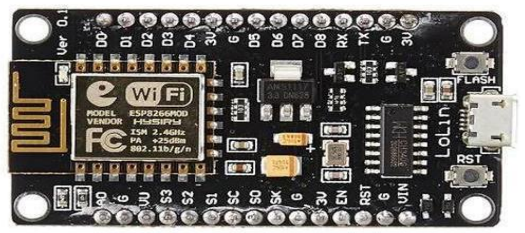


- **Flame Sensor**: This sensor is responsible for detecting the presence of flame within a certain distance. It is a digital sensor and for that matter sends 0 to the Arduino for the absence of flame and 1 for the presence of flame.

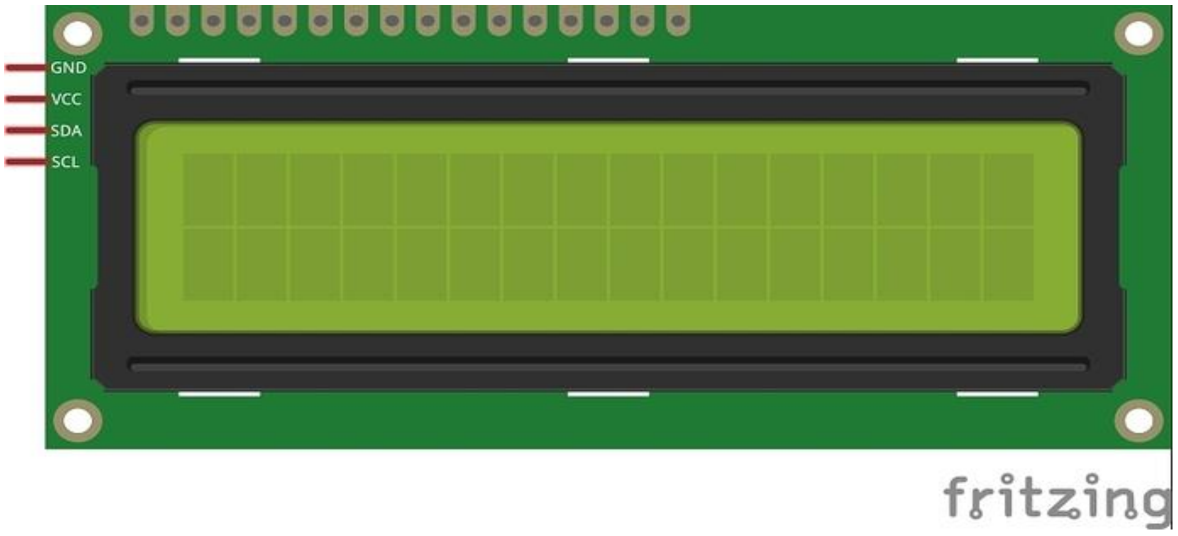


- **MQ2 Gas Sensor**: This sensor is typically used in gas leakage detection. It is suitable for the detection of various parameters such as: H2, LPG, CH4, CO, Alcohol, Smoke or Propane.

- **NodeMCU Wi-Fi Module**: This is the module that is used to interface the Arduino microprocessor with the internet gateway. The data generated by the sensors and sent to the Arduino is able to reach the cloud by the help of this module.

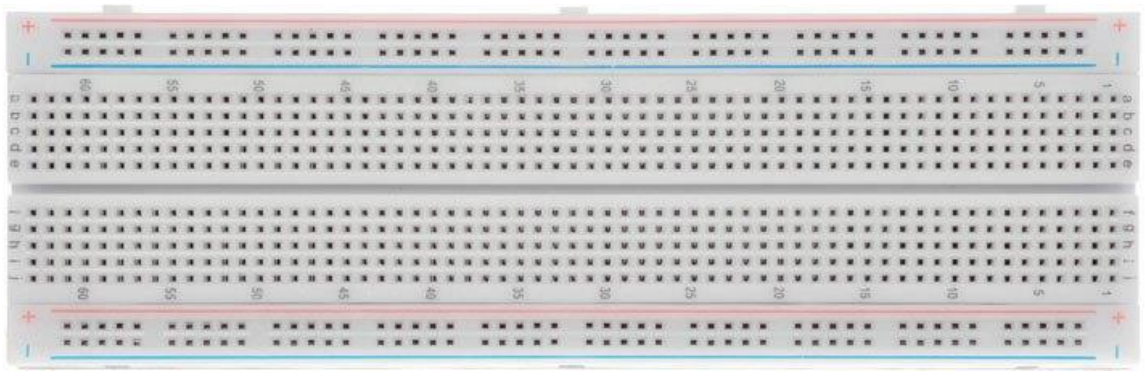


- **LCD Display Screen**: This module helps us display the readings from the various sensors to the user. It is also responsible for the display of alert messages should there be a danger to the home occupants.



- **Buzzer**: There is the need to sound an alarm and bring to the attention of home owners an impending danger, the buzzer helps us achieve this functionality

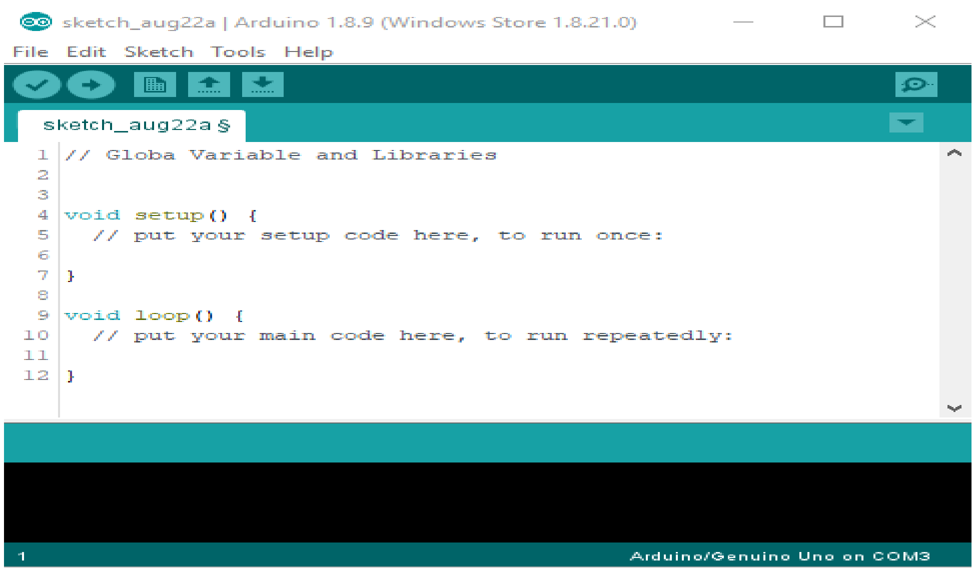
- **Breadboard:** This is a solderless device that is mostly used to temporarily prototype electronic and circuit designs [7]. It makes it easier for us to connect our components in a quick and effective manner as well as keeping our jumper wire connection clean and easy to trace.



3.2 Description of Application and Online Service used in the Proposed System

We basically made use of only two (2) components with regards to this heading:

- **Arduino IDE:** The Arduino Integrated Development Environment; a cross-platform development environment written with the java programming language [8], is used to write the code required by our proposed system. It makes it easy for us to quickly design and deploy software onto our hardware as it supports the code structure of the popular C and C++ hence does not present a very steep learning curve. It also comes with inbuilt software libraries that makes development even easier and faster. A typical program written with the Arduino IDE comes with only two (2) functions and maybe some optional global declarations; which may include variables and library references. The classical functions are the **setup()** and **loop()** functions. As the names imply, the **setup()** basically initializes whatever resources are needed by the program and hardware and the **loop()** contains the instruction that will be run over and over again by the program and hardware. In our system, we made use of two separate software file (sketches) developed using the IDE; one for the **Arduino UNO Microcontroller** and the other for the **NodeMCU Wi-Fi Module**. The individual files (sketches) were pushed onto their respective hardware components through specific board and port definitions from the IDE.


sketch_aug22a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
sketch_aug22a §
1
Arduino/Genuino Uno on COM3

```
// Globa Variable and Libraries


void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig 3.2: Sample Code for an Arduino Project

➢ **Thinger.io Cloud Services:** Data being generated by sensors are transmitted through the internet to a cloud storage so they may be accessed in the future for data analysis and maybe investigations. We are using a 3rd party cloud service, called thinger.io, because it will help us in the quick prototyping and testing of the system. We choose this platform because it is open source and free and the most robust platform to build our data driven system; its agnostic nature makes it possible to connect any device with internet connectivity [9] and hence does not limit our future possibilities of adding different devices from other vendors. It supports bi-directional communications in real-time [9], and gives us the possibilities of visualizing our data in real-time, with beautiful dashboards, as they come in from the sensors. It comes with several other features and below is a diagram that depicts typically the general overview of the platform:

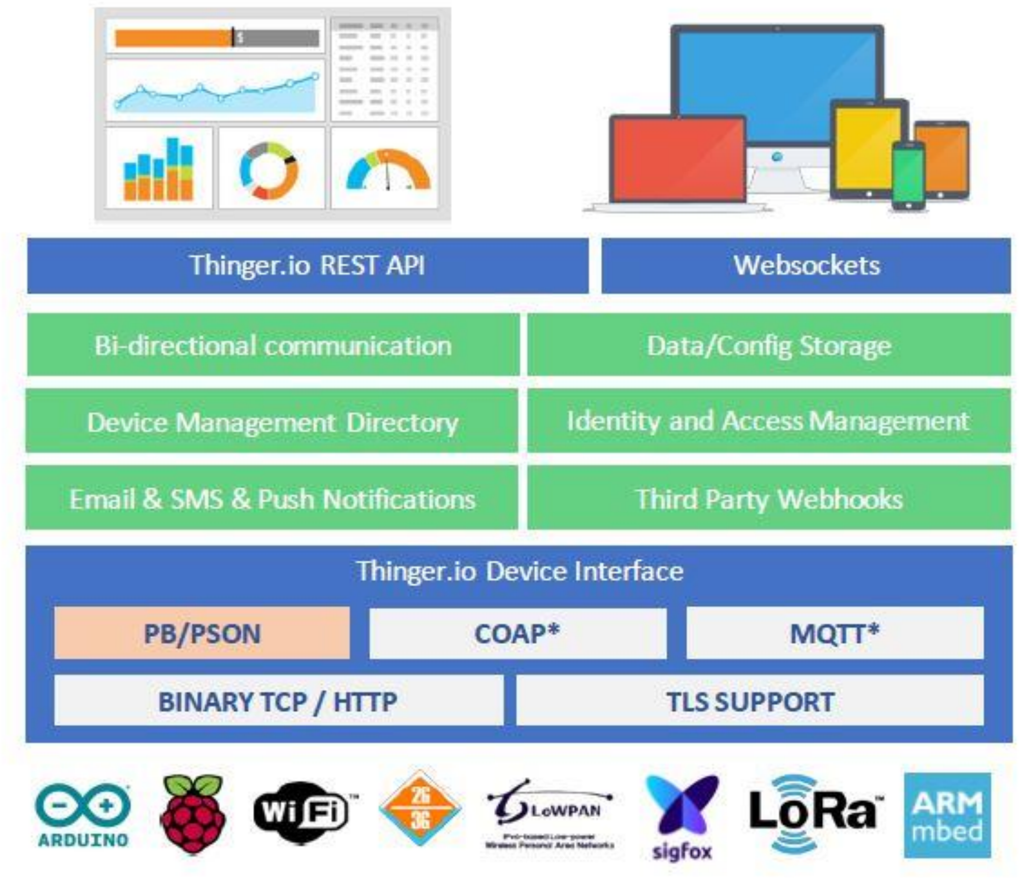


Fig. 3.3: General Overview of the Thinger.io Platform [9]

3.3 Technical Implementation of the Various Hardware Parts of the Proposed System

Below is the breadboard and schematic representation of the connections between the Arduino Uno Microcontroller, the various sensors/output units and the NodeMCU. Here, the breadboard is used to arrange the jumper wire connections to the various components. The red and black jumper wires represent the 5V power supply and Ground connection respectively. All the sensors and output components get their power supply and ground connection from the Arduino through the 5V pin and GND pin respectively, connected with the breadboard. The Arduino and NodeMCU in turn get their power supply from an external supply source which can range between 9V and 12V. In our proposed system, the buzzer is connected to the digital pin 9 of the Arduino. Also, the MQ2 gas sensor is connected to the A0 analogue pin, the DHT11 data pin is connected to digital pin 7 and the flame sensor connected to digital pin 4. The IC2 LCD display makes use of the analogue pins A4 and A5. The NodeMCU pins D5 and D6 are serially connected to the Arduino digital pins 5 and 6 respectively, so they can be able to communicate. They are connected like: (NodeMCU [D5, D6] → Arduino [5, 6]) TX → RX and RX → TX.

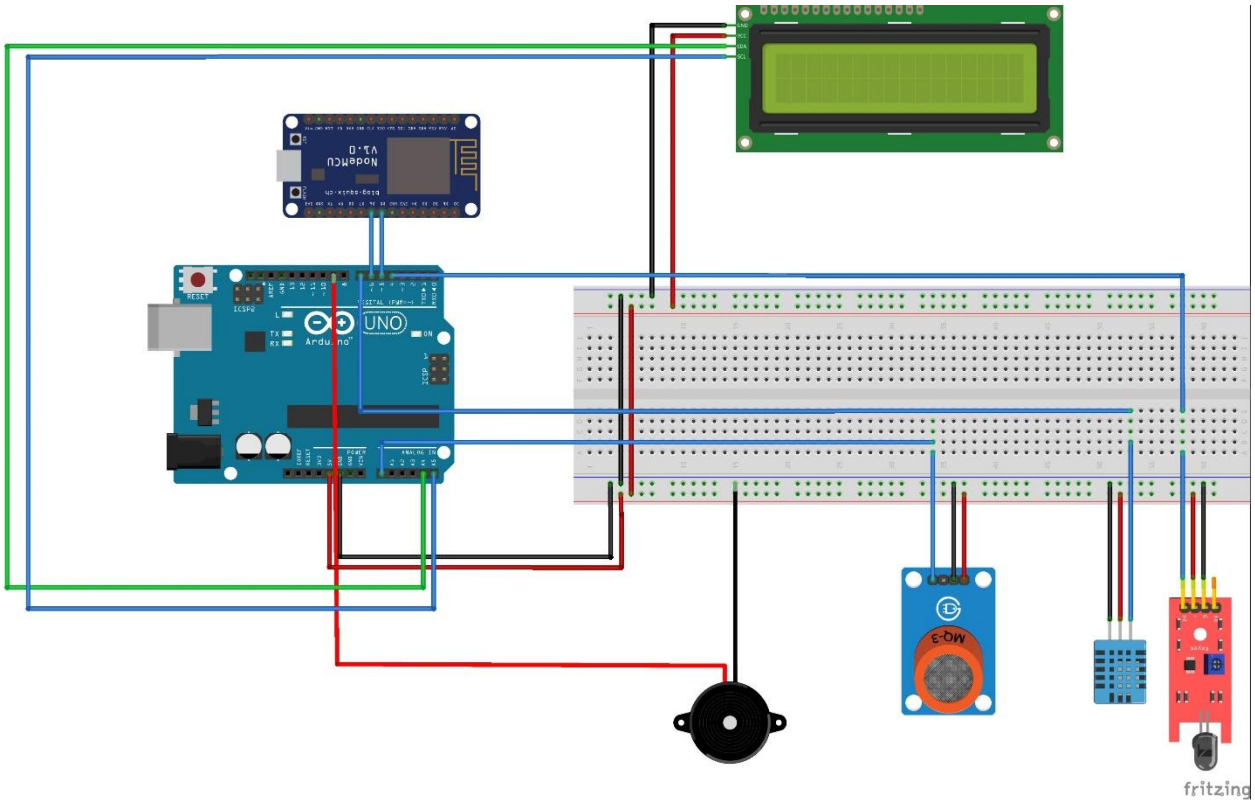


Fig 3.4: Breadboard Representation of Hardware Components

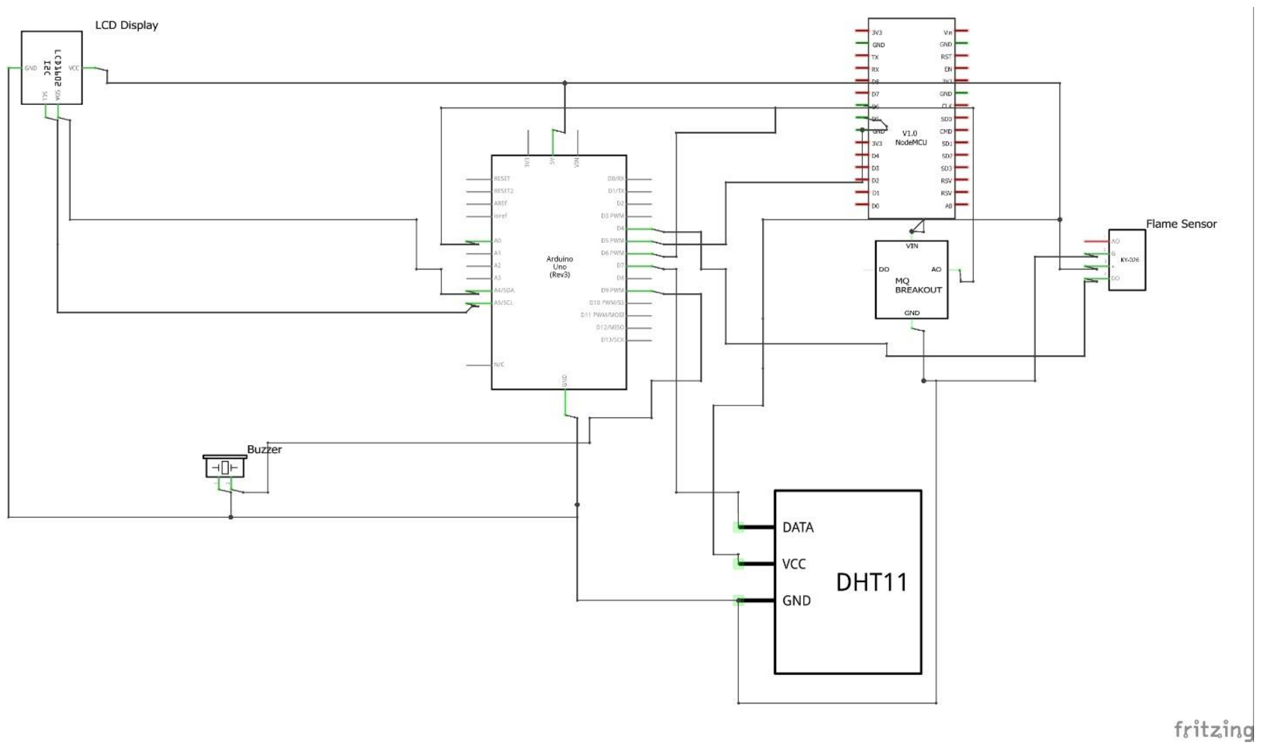
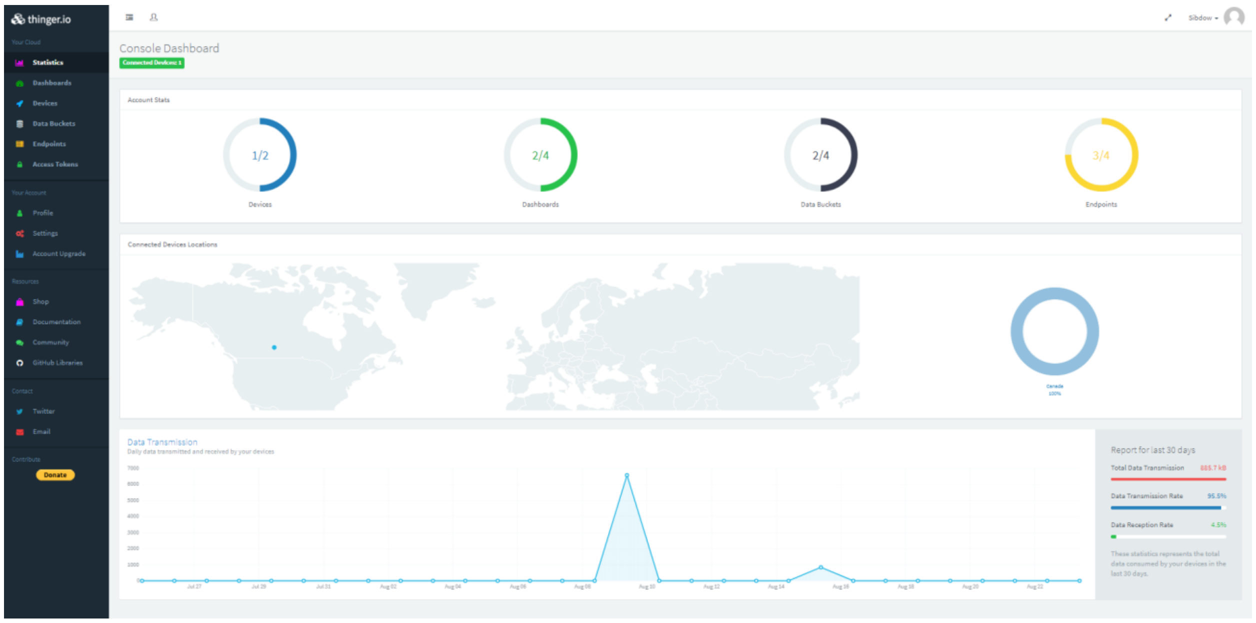


Fig 3.5: Schematic Representation of Hardware Components

3.4 Implemented Cloud based Service

As mentioned earlier, we employed the services of a 3rd party cloud system named **thinger.io**. This platform is designed to be intuitive and you can get a project up and running on it within the shortest possible time. Here, we describe the various features we implemented and made use of from the cloud platform:

- **Console Dashboard:** This provides us with a summary of the various parts of the platform and their current statuses. It gives an overview of the devices connected, dashboards configured, data buckets connected and endpoint configuration summaries. It also points to the locations of all the connected devices as well as gives a summary of the data transmission rates.

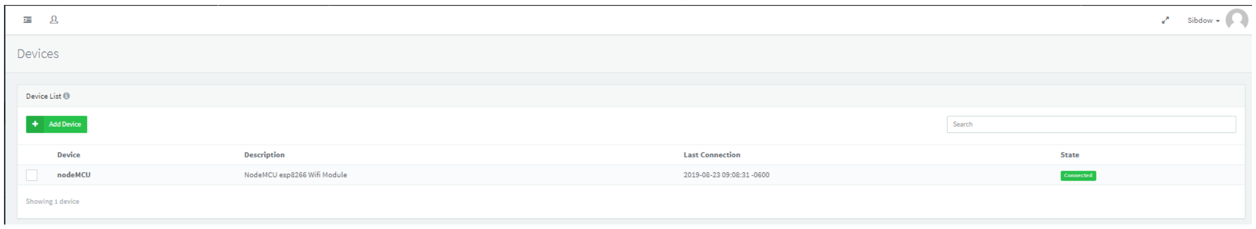


Report for last 30 days

Total Data Transmission	885.7 kB
Data Transmission Rate	95.5%
Data Reception Rate	4.5%

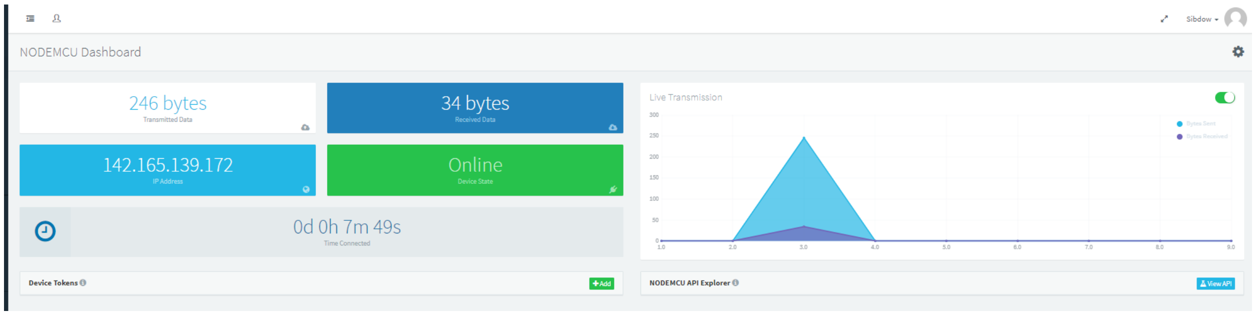
These statistics represents the total data consumed by your devices in the last 30 days.

- **Device Configuration Management:** This functionality allows us to easily configure and add the various devices we would like to make visible on the cloud. It presents us with the status of the device, the IP address, amount of transmitted and received data as well as how long device has been online. Also, we are able to view the datapoints sent to the device by the various sensors through the device API.

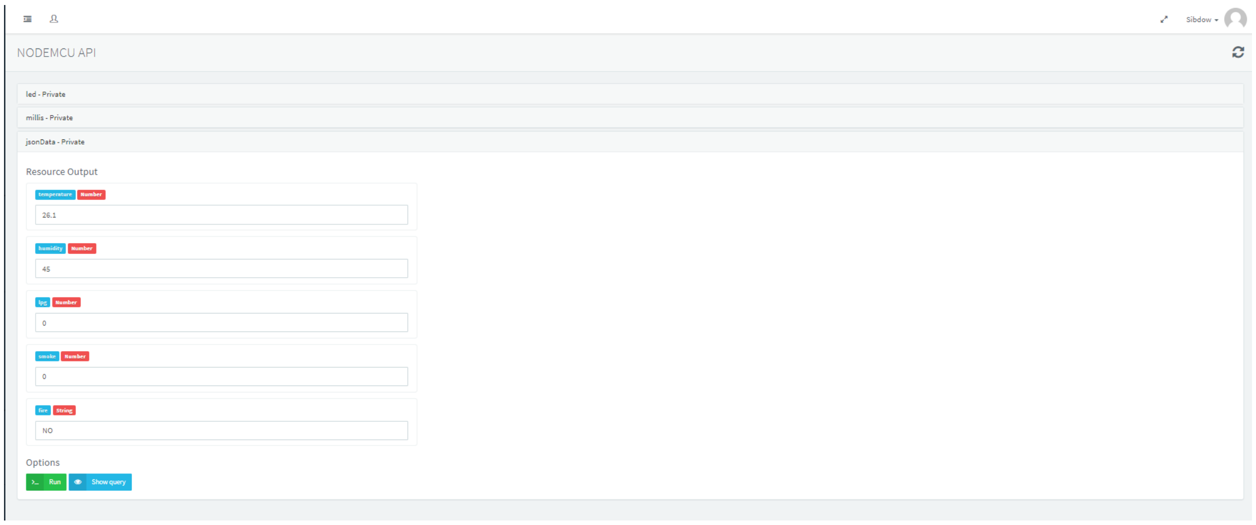


Device	Description	Last Connection	State
nodeMCU	NodeMCU esp8266 Wifi Module	2019-08-23 09:08:31 -0600	Connected

Showing 1 device



246 bytes	Transmitted Data
34 bytes	Received Data
142.165.139.172	IP Address
Online	Device State
0d 0h 7m 49s	Time Connected


Sibdow

NODEMCU API

led - Private

millis - Private

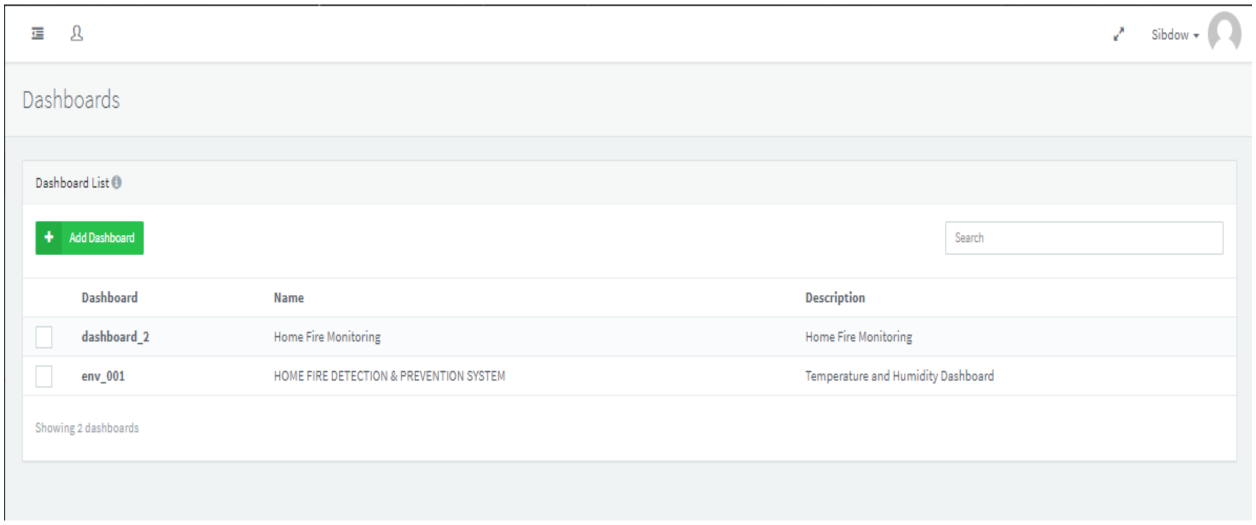
jsonData - Private

Resource Output

temperature	Number	26.1
humidity	Number	45
lpg	Number	0
smoke	Number	0
fire	String	NO

Options

- **Dashboards:** Thinger.io provides us with the functionality to design very beautiful and attractive dashboards which does not only convey the information from the sensors but provides users with interactive features so they could interact with the graphs to better understand what is going on. The feature allows us to add various dashboard widgets ranging from time-series graphs to text and many more. The dashboards can be shared with anyone who has an interest in knowing what is going on; the data displayed on the dashboard can be gotten directly from the sensors and/or from the data buckets on the cloud storage.

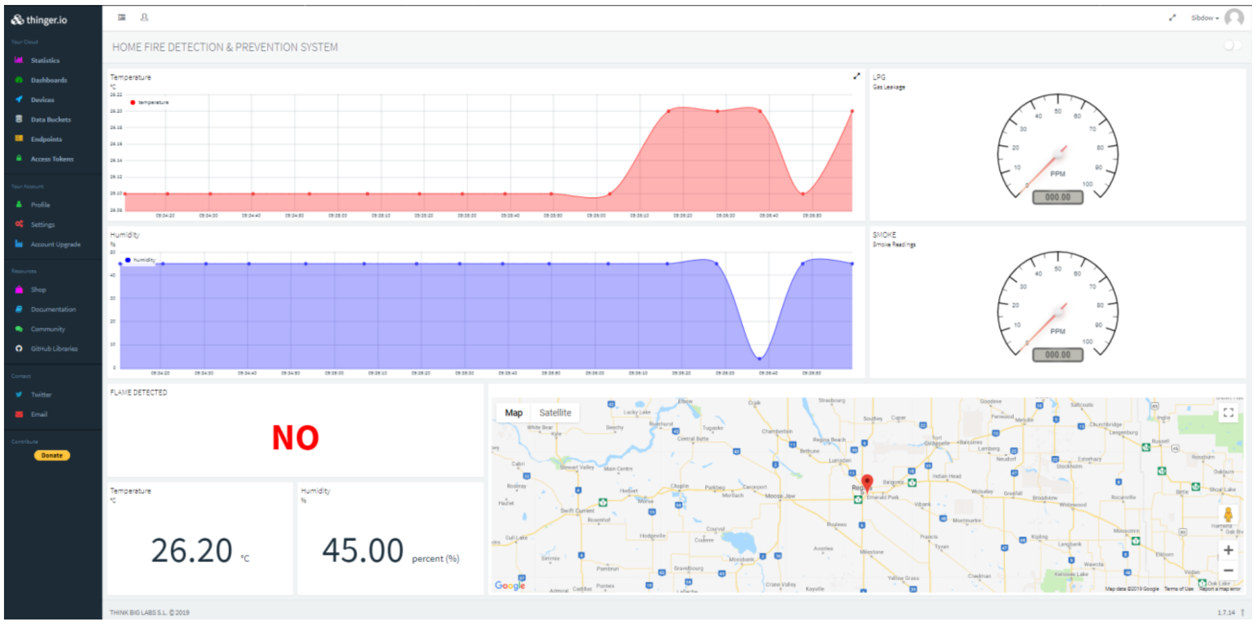

Sibdow

Dashboards

Dashboard List

Dashboard	Name	Description
dashboard_2	Home Fire Monitoring	Home Fire Monitoring
env_001	HOME FIRE DETECTION & PREVENTION SYSTEM	Temperature and Humidity Dashboard

Showing 2 dashboards



HOME FIRE DETECTION & PREVENTION SYSTEM

Temperature °C

Humidity %

LPG Gas Leakage

SMOKE Smoke Readings

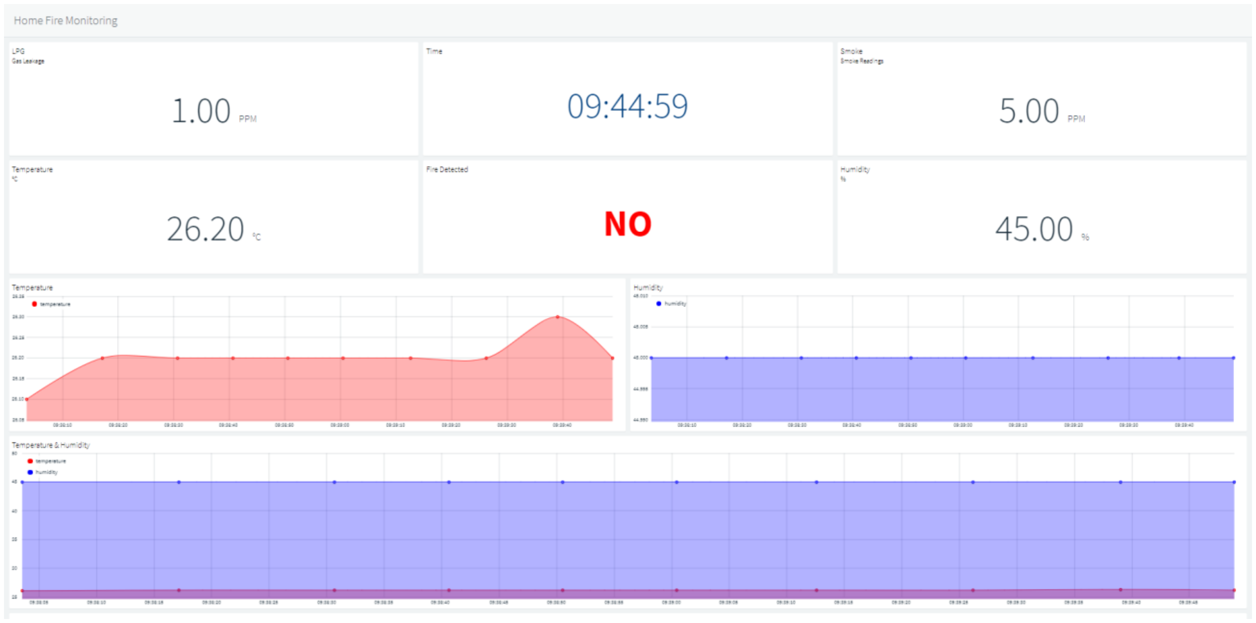
FLAME DETECTED

NO

Temperature °C 26.20 °C

Humidity % 45.00 percent (%)

THINK BIG LABS S.L. © 2019



Home Fire Monitoring

LPG Gas Leakage 1.00 PPM

Time 09:44:59

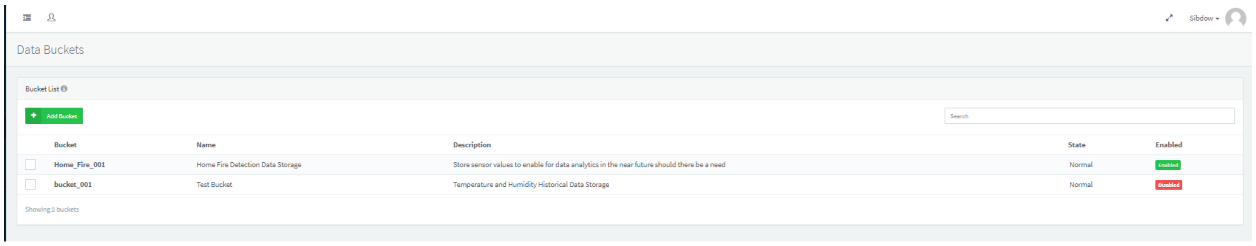
Smoke Smoke Readings 5.00 PPM

Temperature °C 26.20 °C

Fire Detected NO

Humidity % 45.00 %

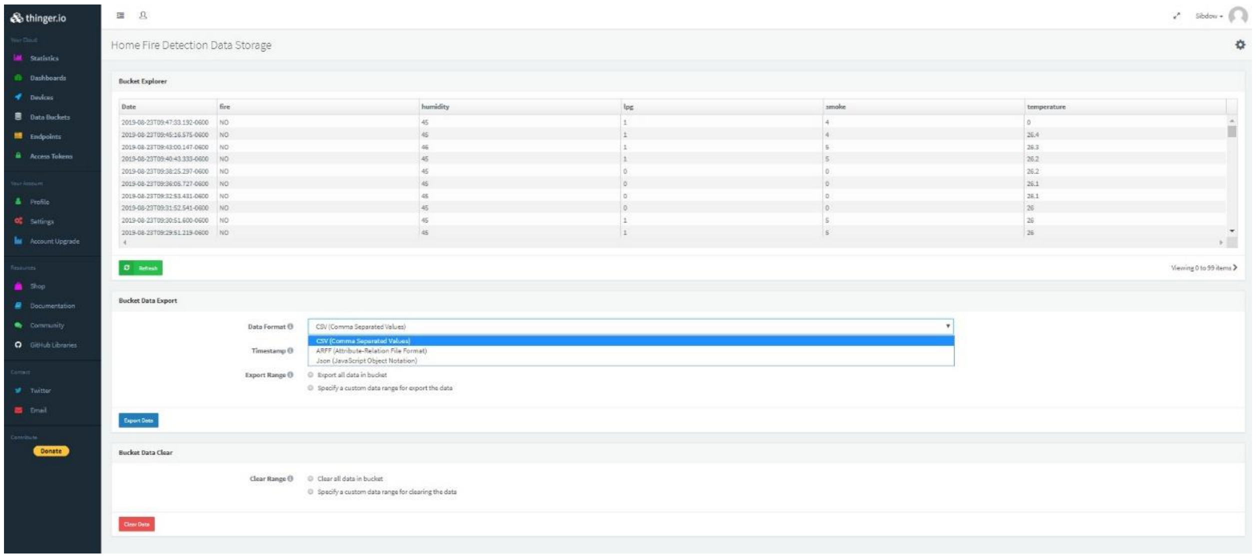
➢ **Data Buckets:** This a feature that is able to save the data sent by the sensors into a storage so it can be retrieved for future use. The data in the buckets can be exported in various data formats including Comma Separated Values (CSV), Attribute-Relation File Format (ARFF) and JavaScript Object Notation (JSON)


Data Buckets

Bucket List

Bucket	Name	Description	State	Enabled
Home_Fire_001	Home Fire Detection Data Storage	Store sensor values to enable for data analytics in the near future should there be a need	Normal	Enabled
bucket_001	Test Bucket	Temperature and Humidity Historical Data Storage	Normal	Disabled

Showing 2 buckets



Home Fire Detection Data Storage

Bucket Explorer

Date	fire	humidity	lpg	smoke	temperature
2019-08-23T09:47:33.192-0600	NO	45	1	4	0
2019-08-23T09:45:16.575-0600	NO	45	1	4	26.4
2019-08-23T09:43:00.147-0600	NO	46	1	5	26.3
2019-08-23T09:40:43.333-0600	NO	45	1	5	26.2
2019-08-23T09:38:25.297-0600	NO	45	0	0	26.2
2019-08-23T09:36:05.727-0600	NO	45	0	0	26.1
2019-08-23T09:32:53.431-0600	NO	45	0	0	26.1
2019-08-23T09:31:52.541-0600	NO	45	0	0	26
2019-08-23T09:30:51.600-0600	NO	45	1	5	26
2019-08-23T09:29:51.219-0600	NO	45	1	5	26

Refresh

Viewing 0 to 99 items

Bucket Data Export

Data Format: CSV (Comma Separated Values)
- CSV (Comma Separated Values)
- ARFF (Attribute-Relation File Format)
- Json (JavaScript Object Notation)

Timestamp

Export Range:
- Export all data in bucket
- Specify a custom data range for export the data

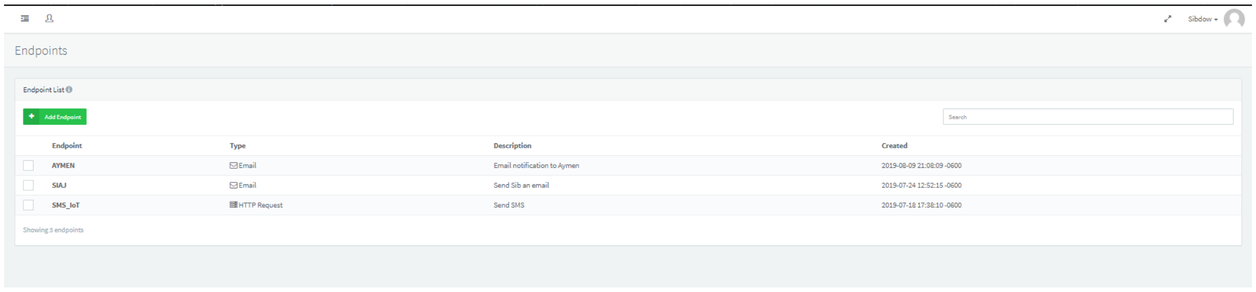
Export Data

Bucket Data Clear

Clear Range:
- Clear all data in bucket
- Specify a custom data range for clearing the data

Clear Data

- **Endpoints:** This is a functionality that allows the cloud system to be able to trigger an event when certain set conditions or thresholds are reached. These events can range from email, HTTP requests, IFTTT Webhook Trigger, Keen IO Analytics, Initial State Event and even a Thinger.io Device call (where an action is sent to another connected device).

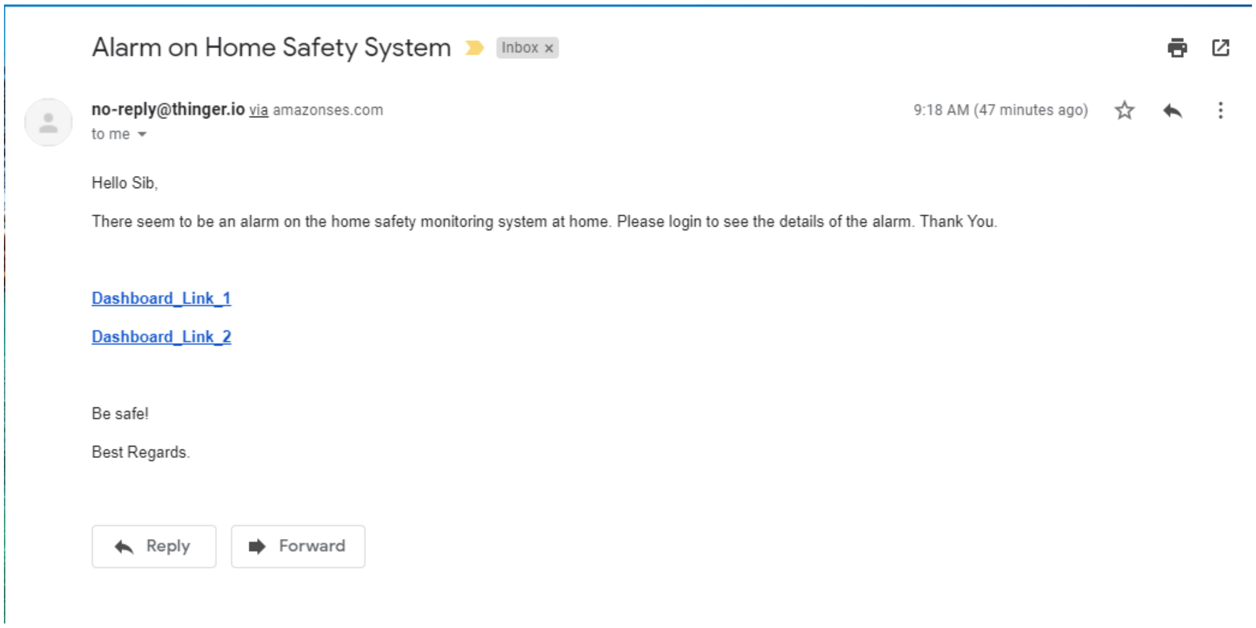

Endpoints

Endpoint List

Endpoint	Type	Description	Created
AYMEN	Email	Email notification to Aymen	2019-08-09 21:08:09 -0600
SIAJ	Email	Send Sib an email	2019-07-24 12:52:15 -0600
SMS_IoT	HTTP Request	Send SMS	2019-07-18 17:38:10 -0600

Showing 3 endpoints

In our system, we made use of the email endpoint where an email is sent to a registered email address when a fire is detected. Shared dashboard links are included in the email so a user can use it to navigate directly to the dashboards displaying the sensor readings.


Alarm on Home Safety System Inbox ×

no-reply@thinger.io via amazonses.com
to me
9:18 AM (47 minutes ago)

Hello Sib,

There seem to be an alarm on the home safety monitoring system at home. Please login to see the details of the alarm. Thank You.

Dashboard_Link_1

Dashboard_Link_2

Be safe!

Best Regards.

The data transmission efficiency and real-time bi-directional communication our chosen cloud system provides makes our proposed solution different from the other existing fire and safety systems. With respect to transmission efficiency, most IoT platforms use the traditional HTTP (Hypertext Transfer Protocol). In HTTP, every device sending data to the cloud issues the HTTP request each time and this way of transmission is inefficient in terms of bandwidth, latency and power consumption which are already limited for IoT devices. Thinger.io provides us with an approach similar to MQTT (Message Queuing Telemetry Transport); which is based on the publisher-subscriber messaging protocol, but defines a custom payload encoding scheme called Protoson (PSON) [11] that is much better in terms of transmission efficiency. The encoding scheme is used to improve memory footprint, save bandwidth, and reduce power consumption on devices [9]. We are also able to implement bi-directional communication, in that when it becomes necessary to control a device over the internet in real time, the platform provides us with the functionality of bidirectional communication between devices and the cloud server.

4. CONCLUSION AND FUTURE WORKS

Home fires are indeed devastating when they strike and with the advent of IoT, it has become easier and necessary to design, develop and deploy systems to help prevent these incidents from occurring. By the use of various sensors and technologies at our disposal, home fires can be prevented or at least detected quickly so as to prevent tremendous damages. The system can also be deployed in Industrial fire monitoring (Manufacturing and Coal mines), forest fire monitoring, oil rigs safety monitoring and many other fields.

This proposed system does not fully solve the problem of having a fire monitoring system in the home which is robust, effective and efficient. If this current system were required to be deployed in a very large environment; it may be impractical at certain locations (like tunnels) or extremely expensive to do so; as we would need to deploy the module repeatedly across the area. A mobile fire detection system, where a rover is deployed to carry the current system around a certain range in order to be able to record and report conditions, is being planned for future works. We also intend to include actuators to the current system so they may be controlled to start a water pump to curb any occurrence of fires before the fire department gets there. Also, there may be alarms of flames which may not necessarily be the case, hence, a camera would be included to the current system to be able to capture images for confirmation by a monitoring station before any actions are taken. Finally, an automatic notification of a nearby fire station through SMS is also intended for future works of this project.

5. CHALLENGES AND LIMITATION

Through the development of our system, there were a few challenges and limitations that were encountered:

1. The connection of the various sensors and output units to the Arduino microcontroller requires a lot of jumper wires, even with the help of the breadboard; this makes it sometimes difficult to trace and manage each connection properly

2. The current design does not protect the sensors enough from fire and when sensors get too close to a fire, it may be damaged.
3. The current design can only be able to detect readings at only the location they were deployed and up to a certain radius.
4. The calibration of the gas sensor was an issue, as proper gas sensor calibration tools were very expensive, so we had to go with the software calibration method which may once a while not provide desired values.

REFERENCES

[1] "Internet of Things Defined - Tech Definitions by Gartner", *Gartner IT Glossary*. [Online]. Available: https://www.gartner.com/it-glossary/internet-of-things/

[2] T. Malche and P. Maheshwary, "Internet of Things (IoT) for building smart home system," *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, 2017, pp. 65-70. doi: 10.1109/I-SMAC.2017.8058258

[3] A. Imteaj, T. Rahman, M. K. Hossain, M. S. Alam and S. A. Rahat, "An IoT based fire alarming and authentication system for workhouse using Raspberry Pi 3," *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Cox's Bazar, 2017, pp. 899-904.
doi: 10.1109/ECACE.2017.7913031

[4] J. K, J. K, J. R, K. R and M. N, "Forest Fire Alerting System With GPS Co-ordinates Using IoT," *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, Coimbatore, India, 2019, pp. 488-491.
doi: 10.1109/ICACCS.2019.8728383

[5] S. K. Bhoi *et al*., "FireDS-IoT: A Fire Detection System for Smart Home Based on IoT Data Analytics," *2018 International Conference on Information Technology (ICIT)*, Bhubaneswar, India, 2018, pp. 161-165.
doi: 10.1109/ICIT.2018.00042

[6] Fernandino S. Perilla, George R. Villanueva, Jr., Napoleon M. Cacanindin, and Thelma D. Palaoag. 2018. Fire Safety and Alert System Using Arduino Sensors with IoT Integration. In Proceedings of the 2018 7th International Conference on Software and Computer Applications (ICSCA 2018). ACM, New York, NY, USA, 199-203. DOI: https://doi.org/10.1145/3185089.3185121

[7] Wiring.org.co. (2019). *Breadboard \ Wiring*. [online] Available at: http://wiring.org.co/learning/tutorials/breadboard/ [Accessed 22 Aug. 2019].

[8] Bush, S. (2019). *Updated: Arduino announces FPGA board, ATmega4809 in Uno Wi-Fi mk2, cloud-based IDE and IoT hardware*. [online] Electronics Weekly. Available at: https://www.electronicsweekly.com/news/products/bus-systems-sbcs/arduino-announced-fpga-board-new-atmega-uno-wi-fi-2018-05/ [Accessed 22 Aug. 2019].

[9] Bustamante, A.L., Patricio, M.A., & López, J.M. (2019). Thinger.io: An Open Source Platform for Deploying Data Fusion Applications in IoT Environments. *Sensors*.

[10] N. Savitha and S. Malathi, "A Survey on Fire Safety Measures for Industry Safety Using IOT," *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, 2018, pp. 1199-1205. doi: 10.1109/CESYS.2018.8723930

[11] https://github.com/thinger-io/Protoson

APPENDIX A

Complete project code can also be found on the link below:

https://github.com/Siaj/Arduino-Fire-Safety-System-with-NodeMCU

→ Arduino Code

```
// MQ-2 sensor working
// Temperature and Humidity sensor working
// LCD displaying values okay
// Buzzer triggered on event(s)
// Potentiometer works fine in tuning the brightness of the LCD

// DHT11 sensor

#include <SoftwareSerial.h>
#include <ArduinoJson.h>
SoftwareSerial s(5,6); //RX, TX
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


#include "DHT.h"
#include <MQ2.h>

#define DHTPIN 7 // what digital pin we're connected to
#define DHTTYPE DHT11  //DHT 11
#define GAS_SENSOR A0
#define FIRE_SENSE 4
int flame_detected;
const char* flame;

```

```
#define BUZZER 9

#define mq2_threshold 100
#define temp_threshold 40

int lpg, co, smoke;
MQ2 mq2(GAS_SENSOR);

DHT dht (DHTPIN, DHTTYPE); //Initialize DHT sensor.

// Set the LCD address to 0x27 for a 20 chars and 4 line display
LiquidCrystal_I2C lcd(0x27, 20, 4);

void setup() {

s.begin(115200);
Serial.begin(9600);
pinMode(BUZZER, OUTPUT);
pinMode(FIRE_SENSE, INPUT);

dht.begin();
mq2.begin();


lcd.begin();
lcd.backlight();

lcd.setCursor(0,0);
lcd.print("******WELCOME!******");
lcd.setCursor(0,1);
lcd.print("IoT PROJECT ~CS890EP");
lcd.setCursor(0,2);
lcd.print("***Sibdow Iddrisu***");
lcd.setCursor(0,3);
lcd.print("***Aymen Ben-Said***");

//delay(5000);

}

StaticJsonBuffer<200> jsonBuffer;
JsonObject& root = jsonBuffer.createObject();

void loop() {

float* values = mq2.read(true);
lpg = mq2.readLPG();
co = mq2.readCO();
smoke = mq2.readSmoke();

float t = dht.readTemperature(); // Read temperature in *C (default)
float h = dht.readHumidity(); // Read humidity %

float hic = dht.computeHeatIndex(t, h);

flame_detected = digitalRead(FIRE_SENSE);
if(flame_detected == 1){
  flame = "NO";
  }else{
    flame = "YES";
    }
```

```

Serial.print("FIRE: ");
Serial.print(flame_detected);
Serial.print(";\t");
Serial.println(flame);


if (isnan(t) || isnan(h) || isnan(hic)) {
    return;
  }

//-----------------------------------------------------------------------------------------------//
  root["temperature"] = t;
         root["humidity"] = h;
         root["lpg"] = lpg;
         root["smoke"] = smoke;
         root["fire"] = flame_detected;

         if(s.available()>0)
           {
             root.printTo(s);
          }
       //  delay(3000);

       //-----------------------------------------------------------------------------------------------//

       if (lpg > mq2_threshold || smoke > mq2_threshold || t > temp_threshold || flame_detected == 0) { // If the current readings exceed this limit, then turn on buzzer
         tone(BUZZER, 1000);
       } else {
           noTone(BUZZER);
           }

       Serial.print("Temerature: ");
       Serial.print(t);
       Serial.print("*C\t");
       Serial.print("Humidity: ");
       Serial.print(h);
       Serial.print("%\t");
       Serial.print("Heat index: ");
       Serial.println(hic);




       // LCD display

       lcd.begin(); // display diamensions
       lcd.backlight();
       lcd.setCursor(0,0);
       lcd.print("Temp=");
       lcd.setCursor(5,0);
       lcd.print(t);
       lcd.setCursor(7,0);
       lcd.print("*C ");
       lcd.setCursor(11,0);
       lcd.print("Hum=");
       lcd.setCursor(15,0);
       lcd.print(h);
```

```
      lcd.setCursor(17,0);
      lcd.print("%  ");

      //===============================================//

      lcd.setCursor(0,1);
      lcd.print("LPG=");
      lcd.setCursor(4,1);
      lcd.print(lpg);
      lcd.setCursor(13,1);
      lcd.print("PPM");
      lcd.setCursor(0,2);
      lcd.print("SMK=");
      lcd.setCursor(4,2);
      lcd.print(smoke);
      lcd.setCursor(13,2);
      lcd.print("PPM");

      if(lpg > mq2_threshold || smoke > mq2_threshold || flame_detected == 0) { // Current thresholds set at only 100 PPM for testing purposes
        if(flame_detected == 0){
          lcd.setCursor(0,3);
          lcd.print("Possible Fire >Flame");
          }else {
            lcd.setCursor(0,3);
            lcd.print("Possible Fire");
            }

        }
        else{
          lcd.setCursor(0,3);
          lcd.print("No Fire Detected");
          }

      }
```

→NodeMCU Code

```
//#include <ESP8266WiFi.h>
// Thnger.io Specifications
#include <ThingerESP8266.h>

#include <SoftwareSerial.h>
SoftwareSerial s(D6,D5); //RX, TX
#include <ArduinoJson.h>



#define USERNAME "Sibdow"
#define DEVICE_ID "nodeMCU"
#define DEVICE_CREDENTIAL "siaj_IoT"

#define SSID "Siaj"
#define SSID_PASSWORD ""

```

```

float t = 0;
float h = 0;
float lpg = 0;
float smk = 0;
int fire;
const char* flame_detected;

// Setup device details
ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);

  // Setup WiFi
  thing.add_wifi(SSID, SSID_PASSWORD);


  Serial.begin(115200);
  s.begin(115200);
  while (!Serial) continue;

  // digital pin control example (i.e. turning on/off a light, a relay, configuring a p
arameter, etc)
  thing["led"] << digitalPin(LED_BUILTIN);

  // resource output example (i.e. reading a sensor value)
  thing["millis"] >> outputValue(millis());

  thing["jsonData"] >> [](pson& out) {
     out["temperature"] = t;
     out["humidity"] = h;
     out["lpg"] = lpg;
     out["smoke"] = smk;
     out["fire"] = flame_detected;
   };

}

unsigned long lastCheck = 0;
void loop() {
  // put your main code here, to run repeatedly:
  thing.handle();

  StaticJsonBuffer<1000> jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(s);

  if (root == JsonObject::invalid())
  {
    return;
  }



  t = root["temperature"];
  h = root["humidity"];
  lpg = root["lpg"];
  smk = root["smoke"];

  fire = root["fire"];
  if(fire == 1) {
    flame_detected = "NO";
    }else {
      flame_detected = "YES";
      }

  unsigned long currentTS = millis();
  Serial.println(lastCheck);
  Serial.println(currentTS);
  Serial.println(currentTS-lastCheck);
  if(currentTS-lastCheck>=1*60*1000) {
    lastCheck=currentTS;
    if (fire == 0) {
      // Call thinger.io email endpoint to send an email to address
      thing.call_endpoint("SIAJ");
      thing.call_endpoint("AYMEN");
      Serial.println(fire);
      Serial.println("Endpoint called");
      }
    }
}
```