An IoT based Fire Alarming and Authentication

System for Workhouse using Xdot and AWS Cloud

Ziyao Yang   
*Dept. of ECE, Electrical Engineering*  
*The University of Southern California*CA, US  
ziyaoyan@usc.edu  
  
Xueting Jin   
*Dept. of ECE, Electrical Engineering*  
*The University of Southern California*CA, US  
xuetingj@usc.edu  
  
YU-LUN CHEN   
*Dept. of ECE, Electrical Engineering*  
*The University of Southern California*CA, US  
yulunc@usc.edu  
  
Wensong Xiao   
*Dept. of ECE, Electrical Engineering*  
*The University of Southern California*CA, US  
wensongx@usc.edu

TaHsin Tseng   
*Dept. of ECE, Electrical Engineering*  
*The University of Southern California*CA, US  
tahsints@usc.edu  
Yucong Li   
*Dept. of ECE, Electrical Engineering*  
*The University of Southern California*CA, US

yucongli@usc.edu

***Abstract*—*The new smart fire detector that our group created is used to detect the possible fire risk more efficiently and accurately. The fire detectors in the market or installed in people’s accommodations nowadays could not distinguish the real fire scenario form other living scenarios such as cooking smoke, cigarettes’ smoke, and steam possibly coming from shower. Those situations can all trigger the fire alarm to ring, and people found that unpleasant. We utilized 4 sensors which are gas smoke sensor, carbon monoxide sensor, temperature sensor and infrared rays sensor implemented with the concept of Internet of Things; we have also designed a user interface that could visualize the information to the user, and the user would be able to interact with the UI, for example, user could turn of the fire alarm by pressing one button on the mobile application. We specifically focused on two false alarm scenarios which is cooking smoke and water steam.***

***Keywords—smart fire detectors, AWS cloud, Interface***

# Introduction

In last decades, fire detection has become a very big issue, as it has caused severe damage including resource waste due to false alarm and, more seriously, the loss of human lives. There are many fire accidents happens in California for last 2 years, and it causes the consumptions of a large amount manpower and material resources; at the same time, according to the report from the Police Department in Los Angeles, the vast majority of alarm calls are false, last year over 90 percent of all alarm calls in Los Angeles turned out to be false alarms. In a city where the murder rate is skyrocketing, the LAPD says it wasted 15 percent of its police officers’ time on false alarm. Besides, as a dwelling at home, when the alarm goes on only because of you are cooking or showering is very annoying, and it takes time to turn the alarm down. Therefore, detecting the fire early and accurately is one of the most essential developing direction in respect of fire detector.

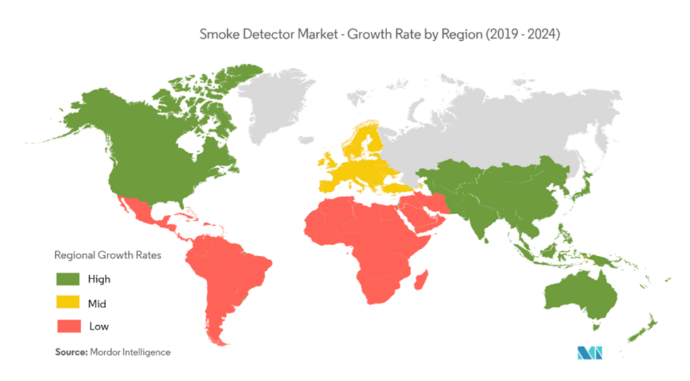
Most of the fire alarm systems use the technology of wireless sensor network (WSN), which could have a variety of uses in different applications, such as target tracking, localization, and healthcare etc.. in Fire detection, each sensor could detects the rising heat, smoke, or gas in a home and then generate alert in the head node in a network, the head node collects reports from various sensors and identifies the presence of a fire. In this kind of fire detection system, the energy could be consumed very fast as the communication between nodes, which would cause the failure of the sensor as it runs out of the battery or the breakdown of the whole network. So that we are proposing a more efficient, IoT based intelligent home smart fire detector using multiple sensors. Moreover, I have added the user interface at the end of our system. It have the information of fire detectors battery level and alarming conditions, so that in a house with more than one fire detector installed, people are able to know which fire detector needs to replace battery, and which alarm is on, then you also can choose to close it.

The rest of the paper is organized as follow: In section II, we talk about market opportunity by comparing with other similar products and introduce the advantages of our design. Section III give a big overview of our proposed architecture. In Section IV, we describe each part of our system in detail. Then, Section V displays experimental results. The last part is conclusion.

# Promises And visons of IOT Fire Alarming

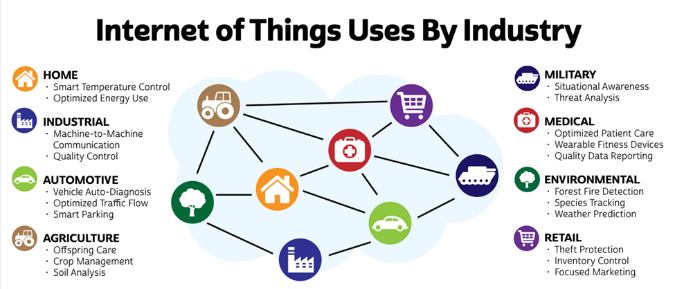
* 1. *Market opportunity*

The need for convenience and strong growth of mobile network infrastructure are driving the popularity of home smart devices, like smart domestic fire detector. In 2018, there are over 46 billion smart home devices were installed in US, as the implementation of smart homes could help to lower the energy cost, strengthen energy saving. And smart fire detector is expected to develop a new generation of smart fire safety equipment during the forecast period. This is needs of all over the world, especially US.



1. Smoke detector market

Furthermore, IoT is one of the trending principle that could be adopted in the fire detection system, The benefits of IoT incorporated smart smoke detectors include the detection of smoke, fire, CO2, and CO on a single device. Customers can turn off the alarm via their IoT enabled devices in case of false alarms and the IoT enabled system notifies the user about the need to change or replace batteries and other wearables. Smart smoke detectors vendors have significant growth opportunities due to the integration of IoT with smoke detecting systems. IoT is penetrating almost every industrial sector at a rapid rate, especially manufacturing, supply chain, and energy. High IoT penetration and industrial automation would have a positive influence on the smoke detector market during the forecast period.



1. Internet of Things Applications

As for our product described in this paper, the sensor costs around 8 dollars per set at retail rate, for the programmable gateway for IoT (conduits), it is around 300-600 dollars could support over thousands connections, which costs around 1 dollar per connections. So the total cost for the hardware setup for single system will be around 10 dollars. Comparing with the traditional fire detection system, in respect of hardware, which is around 15-30 dollars, our product costs are already much lower.

* 1. *Advantages over other products*

Low Cost: Aws Cloud provide SaaS for us. We don’t need to rent many server and make a platform, which is cheap also. Moreover, x-dot and sensors are simple and cheap devices. That’s why we quit the vision of camera and SIM card. It’s power consuming and expensive.

Sensitive: We use advanced algorithm to recognize the fire rather than simple high voltage trigger. Our fire alarm could automatically adjust sensitive to suit for different scenes. For example, traditional fire alarms are not suitable for a factory covered a wide area, the density of smoke cannot reach threshold as fast as small space. In this case, we only need to change some value in our algorithm or choose different model we provide before.

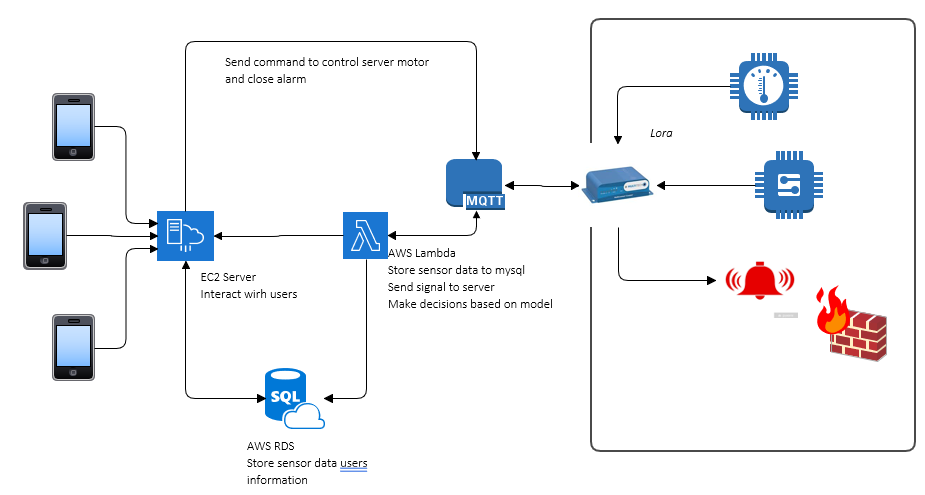
Less False alarm: AWS lambda service is used to process data, that provides great flexibility to program scale up if necessary. We could deploy any model on lambda service. In our design, we choose SVM to detect potential fire hazard. From the result, it could predict fire correctly and filter out the false alarm.

Convenient management: All the X-dot connects to Conduit so that we could use node-red to process and communicate with all the device. Also, all the devices supports peer-to-peer communicate which enable us get access to the device remotely from Conduit. As a result, we could integrate all the sensors in one or serval building.

Long range communication: Lora could cover up to 10 miles range of area and thus we could easily scale up our system.

# SYSTEM DESCRIPTION

The proposed smart fire alarm system is an end-to-end solution which allows users to get the quick notification when there is a fire alarm through the app on their smart phone. Besides, users are able to know which alarm is triggered by the real-time provided sensor device id information and they can shut down the alarm by just pressing a single button. The system can be divided into three stages in general.



1. System architecture



1. System physical photo
   1. *Sensor data collection*

Initially, the sensor will keep measuring the data. When the sensor value exceeds the predefined threshold value (the value comes from pretrained machine learning model), the XDot will send the abnormal data to the Conduit via LoRa. This is the first stage filter mechanism, avoiding transmitting lots of data to the AWS to do the analysis and in turn save the power. After that, the Conduit will send the data via MQTT to AWS lambda serve for the advanced machine learning analysis.

* 1. *AWS based data analysis*

With the incoming collected data, the pretrained machine learning model in AWS lambda will predict the scenario based on the input data. This is the second stage abnormal data filtering process. The possible scenario could be normal cooking, steam or real fire alarm. If the predicted outcome is fire alarm, the lambda will send the signal back to XDot to trigger the buzzer alarm. Besides, in this stage the device id information will be sent in together with the data, which will be compared with the user’s subscription to device table in the SQL database and it will know the list of users who subscribes to this triggered smart fire alarm. Finally, the EC2 server will base on this user’s subscription list to notify the users via the app we provided on smart phone.

* 1. *Users notification and reaction*

In the last stage of this end-to-end solution, the users will get notified via the app if the real fire alarm happens. In the meantime, the user will know the triggered device id information and can just press a button to turn down the alarm. On top of that, when the battery of the system is too low, the AWS will send the low battery warning to notify the user with the information where the system is located at.

# SYSTEM IMPLEMENTATION

* 1. *Sensors*

To collect different types of sensor data as the features for the machine model analysis, we aim to measure like CO, smoke, flame, etc. These are the noticeable index for us to know whether the true fire situation happens. Besides, we also utilize a voltage sensor the measure power for the whole system. It will trigger a low battery alarm when the battery is dying or below certain threshold.

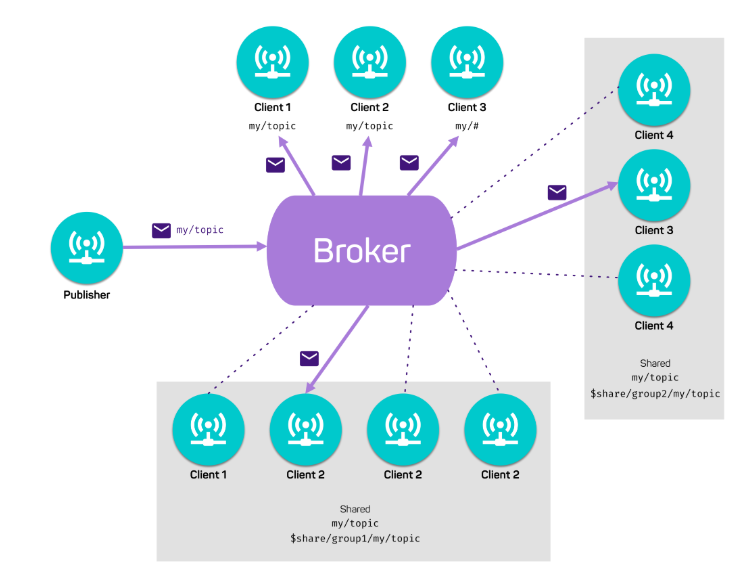
1. Sensors Descriptions

|  |  |  |
| --- | --- | --- |
| ***Sensor*** | ***Utility*** | ***Illustration*** |
| MQ2 | To get a general measurement for gas smoke |  |
| MQ7 | To measure carbon monoxide (CO) |  |
| DHT11 | To measure temperature |  |
| Flame detector | To measure the occurrence of fire with infrared ray |  |
| Voltage | To measure the voltage from the power source to the system |  |
| Buzzer | To beep when the real fire alarm is detected |  |

* 1. *MQTT Protocol*

Due to the particularity of the mobile terminal, the network environment and IP address of it always keep changing, hence, HTTP and simple TCP/IP protocol cannot promise the success of real-time notification push.

MQTT protocol was born to address this problem. MQTT uses publish-subscribe mechanism to perform communication. In MQTT protocol, there is no direct connection between sender and receiver, but both of them are connected to a broker. Client can automatically maintain a long TCP connection with client and if connection lost, it will automatically start re-connection. Sender publish messages with a certain topic to broker, then, broker will send the messages to receiver who subscribe this topic, which decouples the sender and receiver. Besides, every transport of MQTT protocol can specify a quality of message (QoS). MQTT protocol will ensure the arrival of any message with QoS higher than 1. Additionally, MQTT protocol are able to send a message to many different clients. Based on the above characteristics, MQTT protocol is suitable to achieve real-time notification push. Our system also takes MQTT protocol.



1. MQTT model
   1. *LORA*

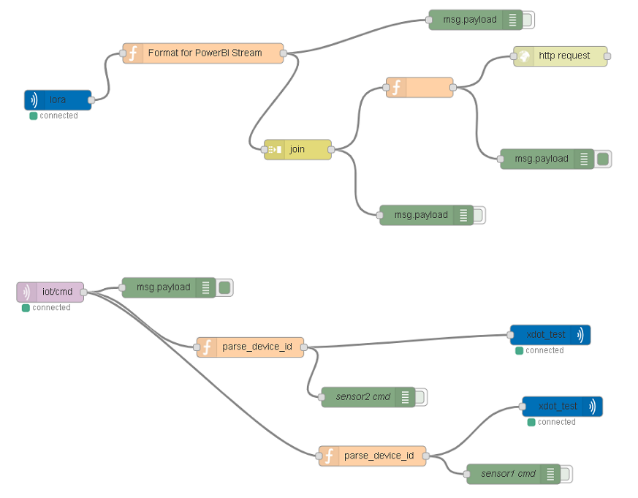
LoRa (Long Range) is a long-range low-power wide-area network (LPWAN) technology. It is based on spread spectrum modulation techniques derived from chirp spread spectrum (CSS) technology. It operates under the 915MHz frequency band in North America and able to transmit up to 10 miles line of sight or 1 to 3 miles into buildings, depending on the positioning and surrounding environment. The reason why we choose LoRa over other wireless technology, for example WiFi or Bluetooth, is that it has longer communication distance with low power consumption and able to support up to 2000 end-devices for a LoRa gateway. The comparison between them is shown in table II. [8] [9] As we can see in table II, the distance and the number of devices a gateway can support has significant difference between LoRa and 2 other technology and those two characteristics are crucial for IoT applications.

1. Wireless Technology Comparison

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***LoRa*** | ***Low-power WiFi*** | ***Bluetooth*** |
| Power consumption during transmission | ~50mA | ~75 mA | ~15mA |
| Distance | 10 miles | ~300 feet | ~50 feet |
| Number of devices supported for a gateway | < 2000 | ~255 | 7 |

* 1. *Conduit and node-red*

The Conduit is the industry’s most configurable, manageable, and scalable LoRa gateway for industrial IoT applications. Network connectivity choices to your preferred data management platform include carrier approved 4G-LTE, 3G, 2G and Ethernet. A diverse range of mCardTM accessory cards provide the local wired or wireless field asset connectivity and plug directly into the rear of the Conduit LoRa gateway. Available options include a LoRaWAN Ready mCard capable of supporting thousands of mDotTM long range RF modules connected to remote sensors or appliances. Development environments to suit all users including IBM’s Node-RED, a graphical, drag and drop interface or mLinuxTM Open Embedded/Yocto. Quick-to-deploy and easy to manage, the Conduit LoRaWAN gateway realizes your IoT application.

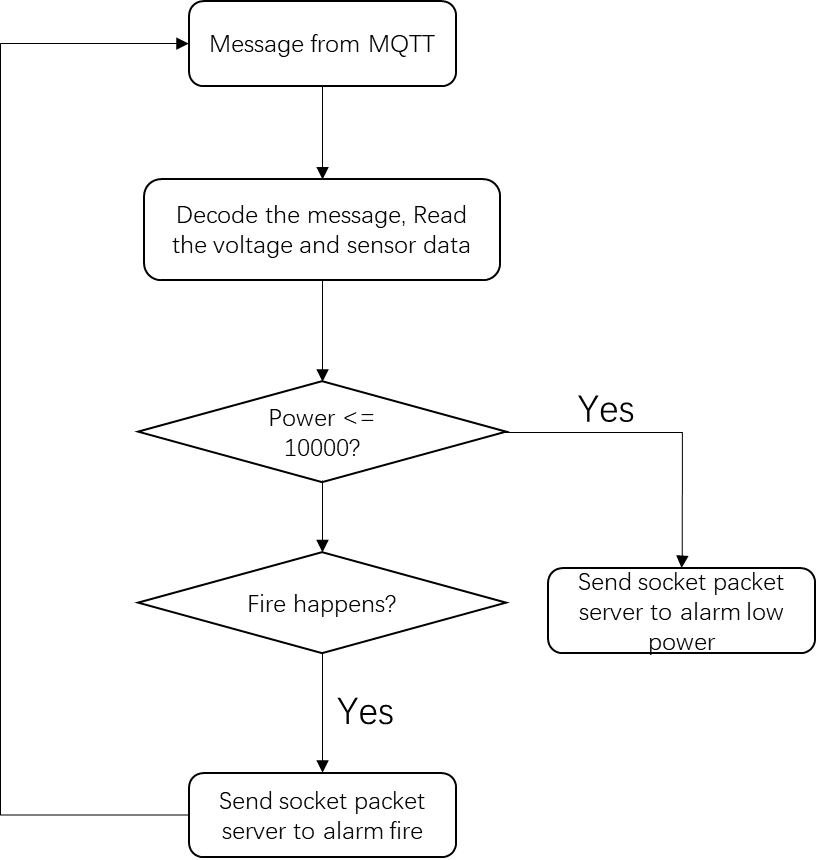


1. Node red data flow control
   1. *AWS Cloud*

AWS Cloud provides us with SAAS as below:

* Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2’s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actuallyy use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.
* AWS IoT Core is a cloud service that ease the complexity of connecting devices with cloud applications and other devices. AWS IoT Core can support billions of devices and trillions of messages, process and route those messages to AWS endpoints and to other devices reliably and securely. With AWS IoT Core, applications can keep track of and communicate with all your devices, all the time, even when they aren’t connected. AWS IoT Core also makes it easy to use AWS and Amazon services like AWS Lambda, Amazon Kinesis, Amazon S3, Amazon SageMaker, Amazon DynamoDB, Amazon CloudWatch, AWS CloudTrail, Amazon QuickSight, and Alexa Voice Service to build IoT applications that gather, process, analyze and act on data generated by connected devices, without having to manage any infrastructure.
* With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.
* Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.
  1. *Aws lambda*

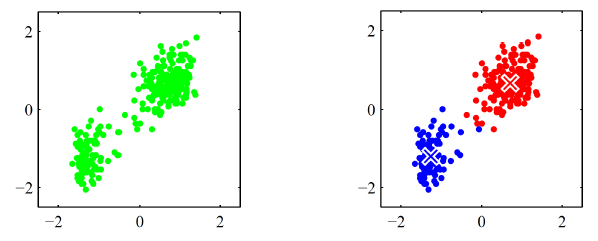
We deploy a tight server in Aws lambda, triggered by messages from MQTT. Before deploy machine model on Lambda, it only stores the data to RDS database.



1. Lambda flow control

As Fig.6 shows lambda function decodes JSON formal message from MQTT and then extract the voltage, CO, Smoke, Temperature and Flame. Then it checks if battery runs up. If so, it will send message to EC2 server packed by socket. Next, it will predict fire and then send message.

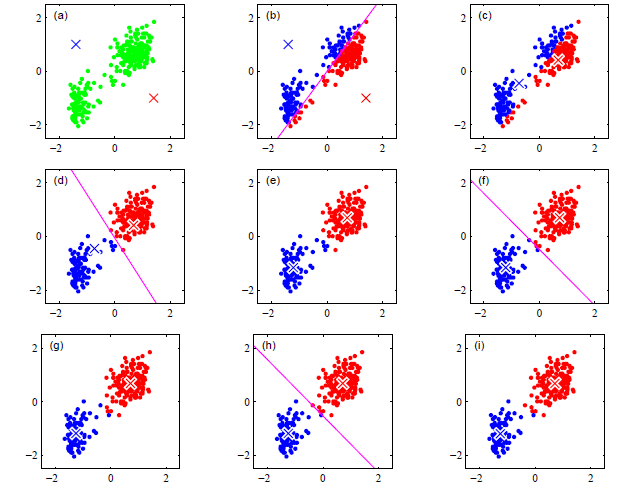
We use K-means cluster algorithm to predict the fire hazard. K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. A cluster refers to a collection of data points aggregated together because of certain similarities. You’ll define a target number k, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. The ‘means’ in the K-means refers to averaging of the data; that is, finding the centroid.



1. Kmeans clustering

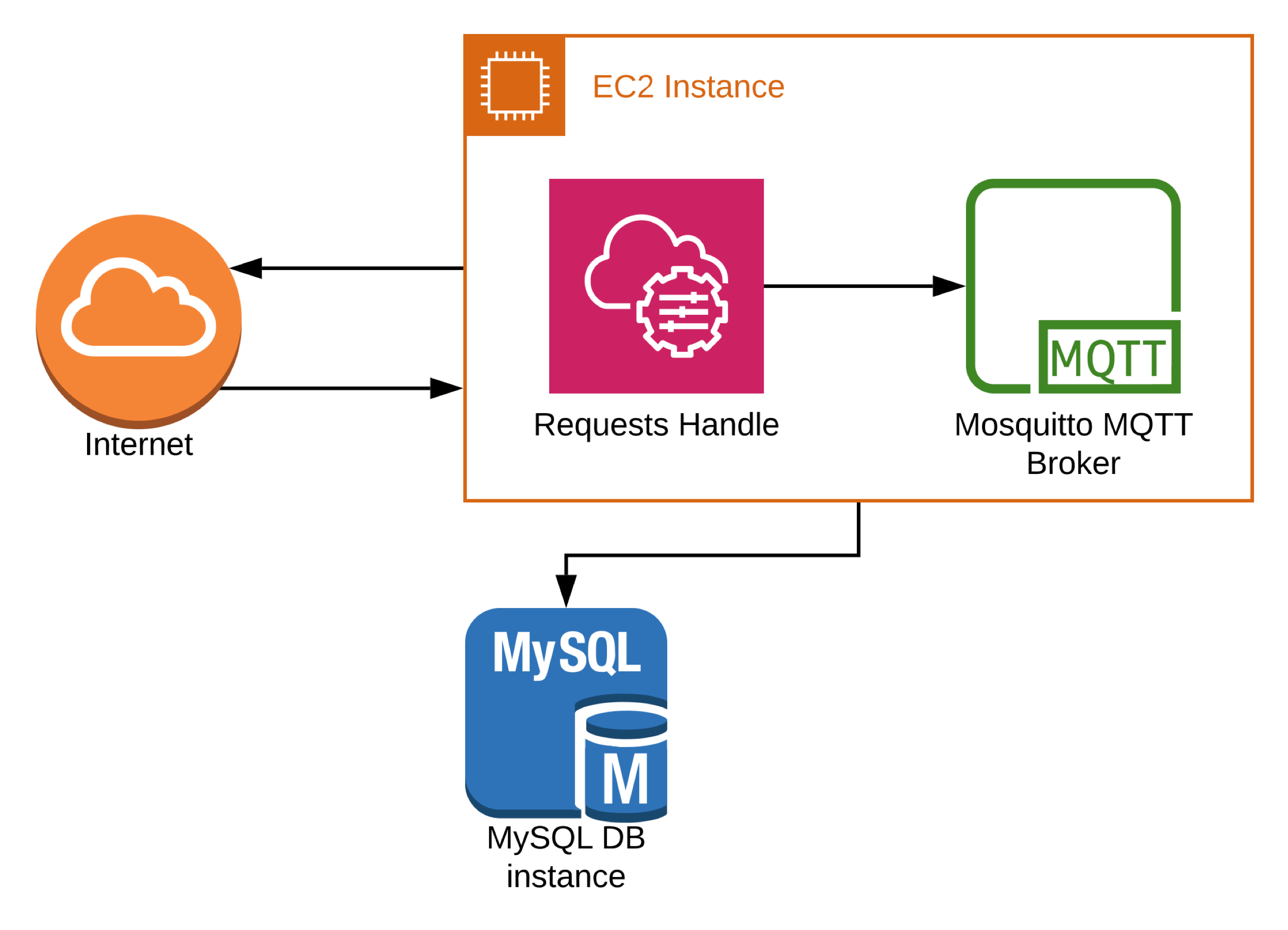
K means algorithm is stepped as below:

1. Initialization (choose K centers)
2. Fix the centers , assign each point to the closest center:
3. Fix the assignment , update the centers：
4. Repeat Steps 1 and 2 until the centers no longer change.



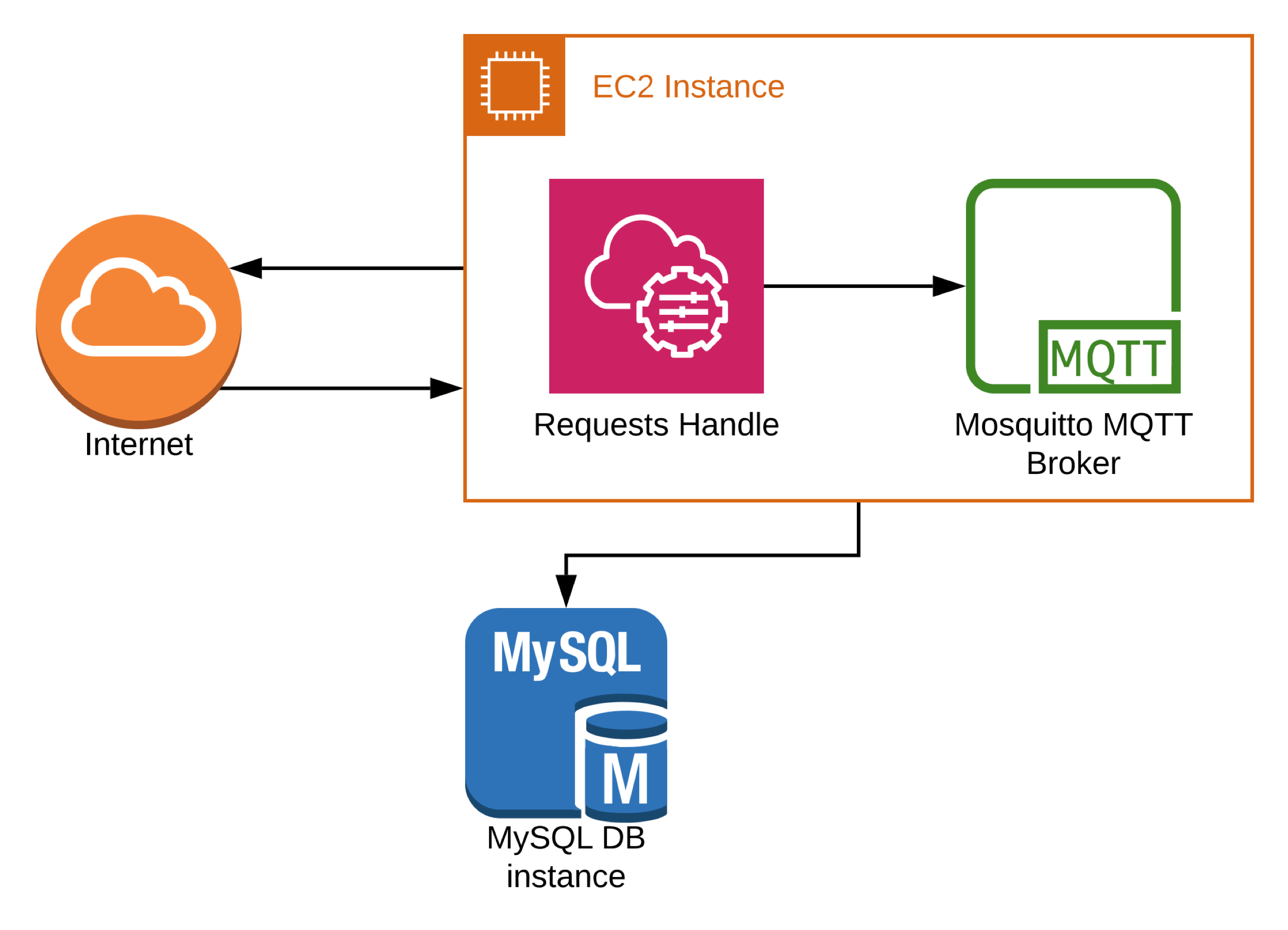
1. A example of K means
   1. *EC2 server*

Main server can respond to the different requests from both detectors and clients at the same time. It The core function of server is to allow one user to monitor and control multiple sensors and allow multiple users to monitor and control the same sensor simultaneously. Additionally, the server should ensure the reliable arrival of message push. Figure 1 shows the architecture of the server. The main server is deployed on the EC2 instance. It includes a request handler and a Mosquitto MQTT broker. MySQL database stores the information of clients and detectors.



1. Architecture of server

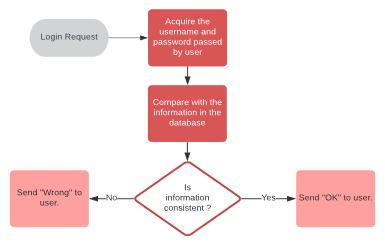
After user login into the system, they will connect to the MQTT broker followed by subscription of a topic with their username. If server need to send messages to some specific clients, it would just publish those messages to topics of username of the clients on MQTT broker. Then, specific clients would receive messages.



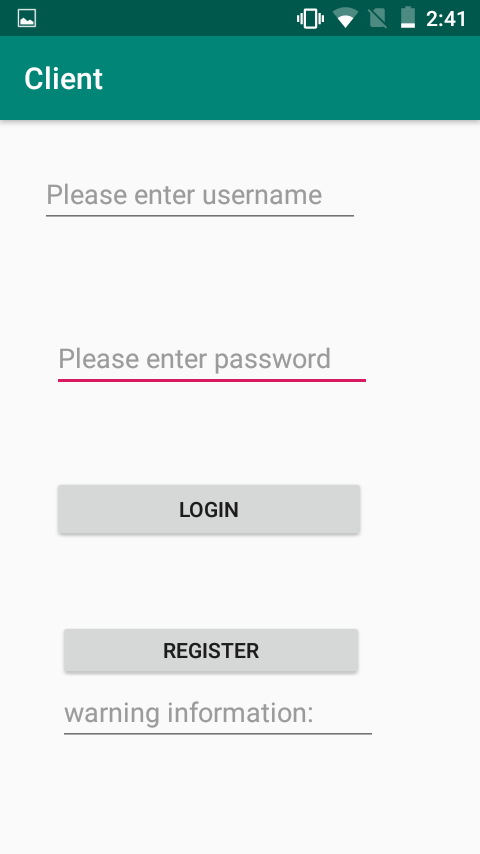
1. notification push flow of system

In our system, one user is able to monitor multiple detectors and one detector can be accessed by multiple users. Hence, the relationship between users and detectors is many-to-many. In order to store this relationship, we design three tables sensor\_Info, user\_Info, relation\_table in the MySQL database. The device ID is unique to distinguish different detectors. Sensor\_Info table record the installation position of each detector, which is responsible to locate the position of fire. Relation\_table stores the relationship between users and detectors. In the query process, server can get the all detectors concerned by one user by setting username as filter. Similarly, if device ID is filter, all users who concerns this device can be acquired. User\_Info table manage the user information, and in business applications, this list can be extended to store more.

The request handler is the main part of the entire server. Request sent to server through TCP short link is json data in which type field represents the request type. Handler firstly acquire the types of request and then submit it to different process program according to the types. Those requests are “Login”, “Register”, “Add Device” and “Close Alarm” from clients and “Fire Warning” and “Low Battery” from detector. Invalid request type will be simply discarded without any feedback.



1. Logic flow of processing login



1. Screenshot of login page on Android APP

“Login” and “Register” services are used to manage user information. This module left room for future commercial expansion, such as only users who meet certain conditions can use the system.

The “login” data sent by user is: {‘username’: ,’password’: , ‘type’: ’login’} As Fig.8 shows, when processing login request, the username and password are read from the login data firstly. Then querying the database according to username to get correct password. Finally, if those two passwords are same, server send “OK” to user as feedback to show success of login, otherwise “Wrong”.



1. Logic flow of processing add device

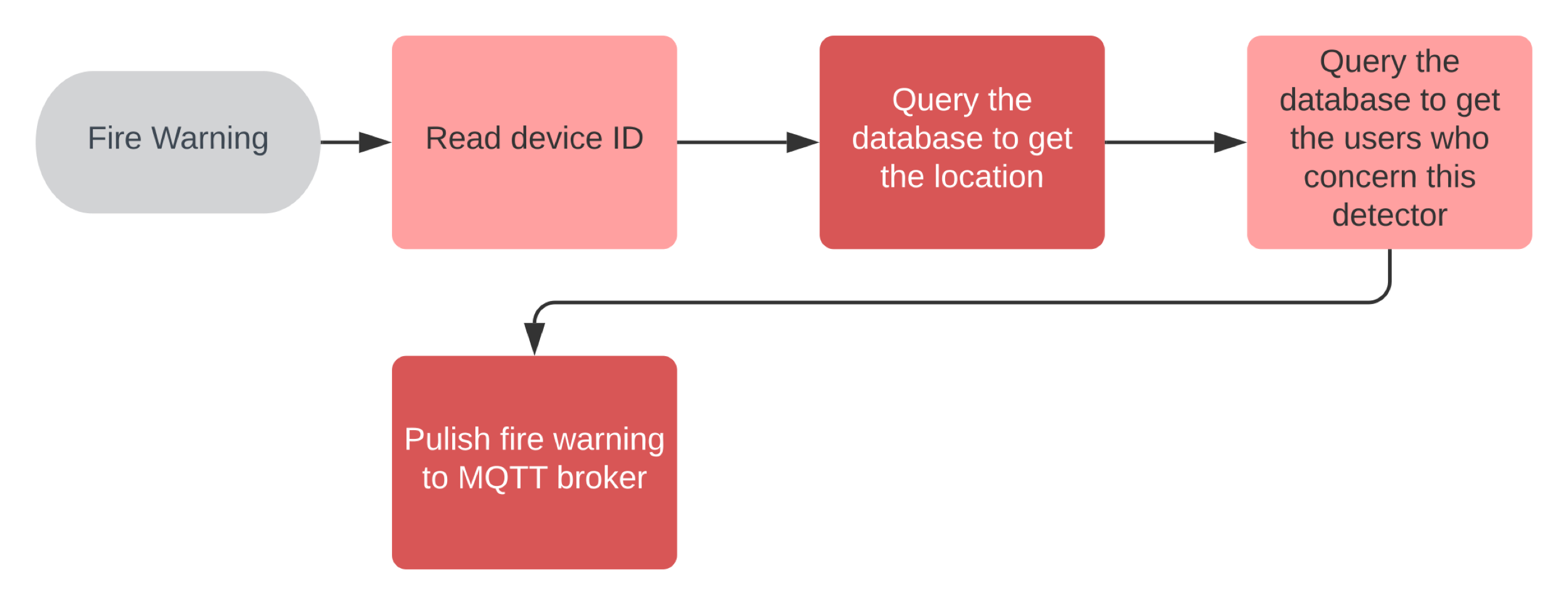
“Add device” service allows user to monitor and control multiple detector at same time. The “add device” data sent by user is: {“username”: “yang”, “device ID”: “xxxxx”, “type”: “add Device”, “location”: “room”}. The device ID of each detector is unique. Location represents the installation position. Server searches the database to see whether this detector is a new device. If yes, server would insert the device ID and location of this detector into database. Next, this device is added into list of devices that this user subscribes. Finally, “OK” would be sent to user as feedback. If any error occurs, no feedback would be sent.



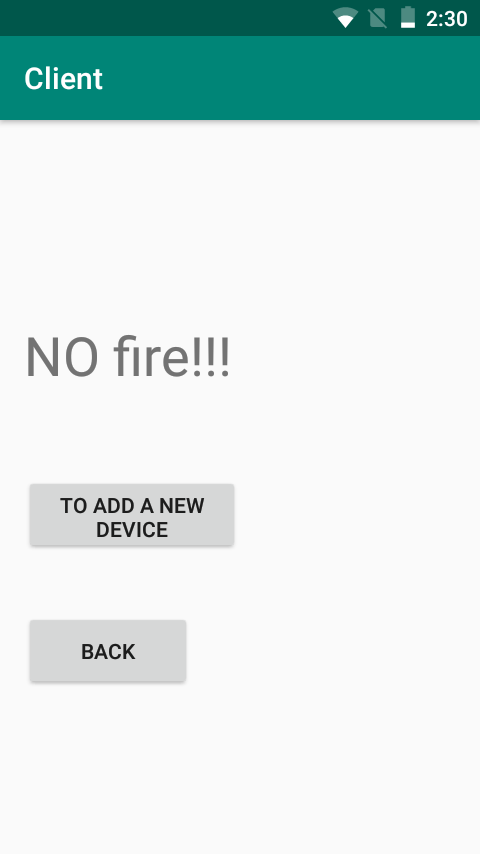
1. Logic flow of processing close Alarm

“Close Alarm” allows users to turn off the alarm detector remotely. The request sent by users is: {“type”: “Close Alarm”,

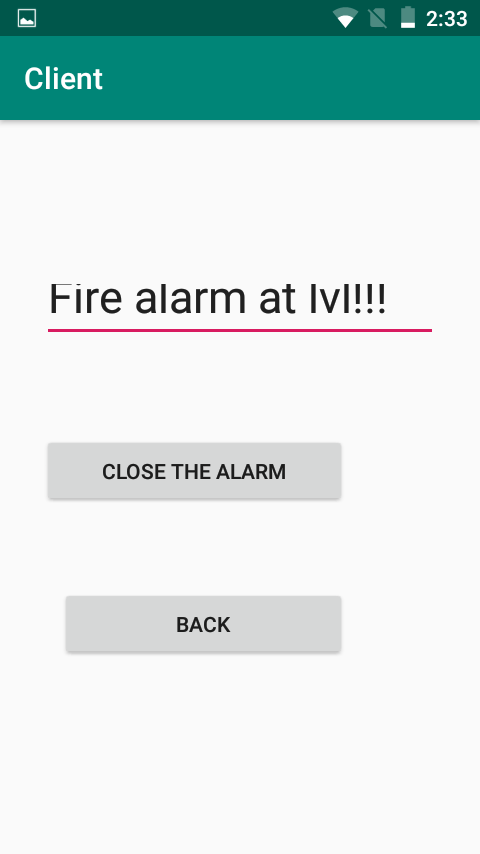
“device ID”: “xxxx”}.When server receives “Close Alarm” request, it will forward this message to the AWS IoT core MQTT broker “iot/cmd” topic. Encrypted SSL communication is used when forwarding to ensure detector will not be controlled by others.



1. Logic flow of processing Fire warning

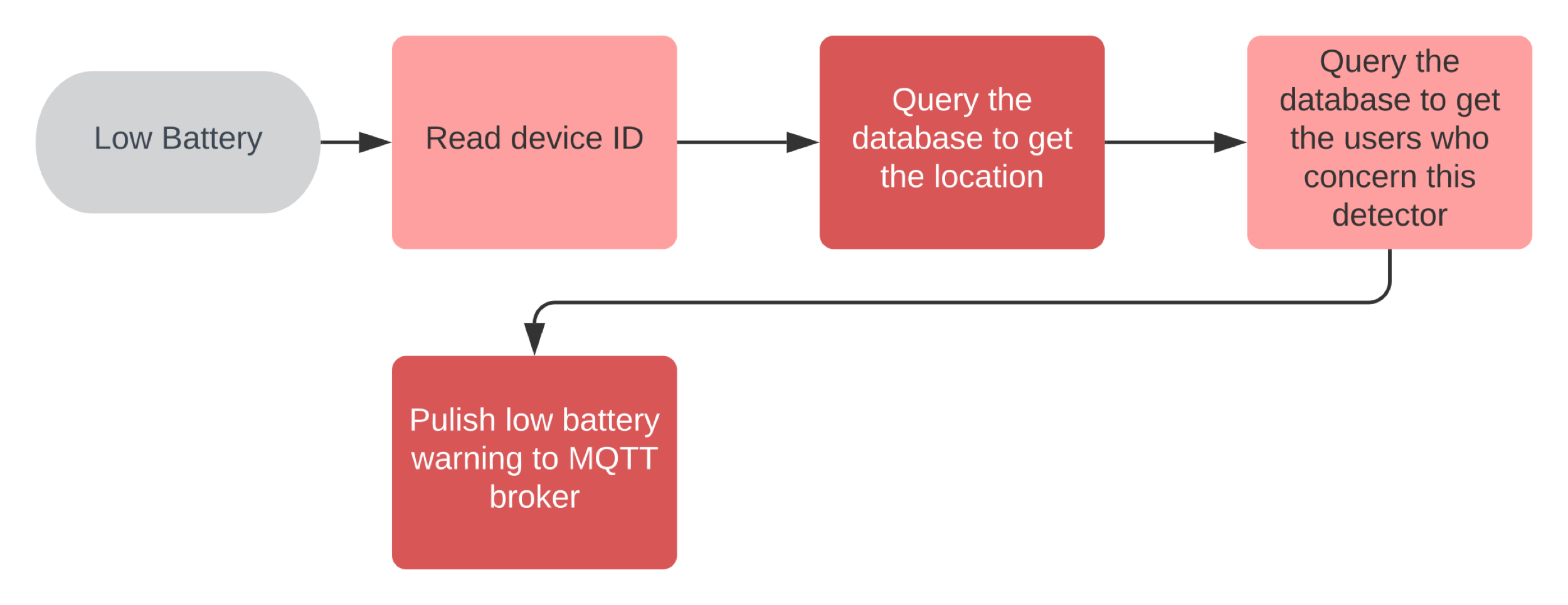


1. Screenshot of normal condition on Android APP



1. Screenshot of fire warning on Android APP

Fig.11 shows Logic flow of processing Fire warning If fire is detected, detector will send fire warning to server. The “Fire warning” data sent by detector is: {‘type’: ‘Fire Warning’, ‘device ID’: } Server query the database to get the fire location and the users who concern this detector according to device ID. Subsequently, server would publish fire information {‘type’: ‘Fire Warning’ ,‘location’: } with username as topic to MQTT broker. Then, users can receive notifications via MQTT broker.



1. Logic flow of processing Low Battery

When the detector battery power drops below the threshold, detector would send low battery notification to the server. The “Low Battery” data is {‘type’: ‘Low Battery ‘device ID:} Similar with “Fire Warning”, Server query the database to get the fire location and the users who concern this detector according to device ID. Subsequently, server would publish fire information {‘type’: ‘Fire Warning’ ‘device ID’: ‘location’: } with username as topic to MQTT broker. This message can notice user which detector need to replace battery to avoid stop.



1. Screenshot of Low Battery on Android APP
   1. *User Interface*

In general, our UI has 6 activities including Main Activity, Login, Register, Scan, Alarm and Low Battery, respectively.

A screenshot of a cell phone

Description automatically generated

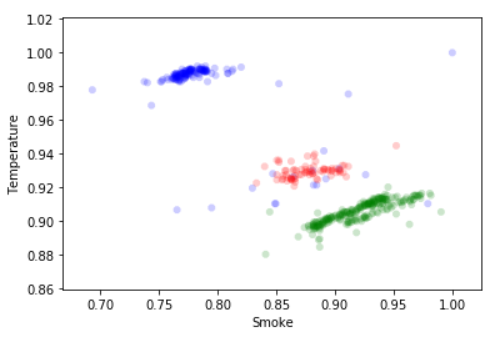
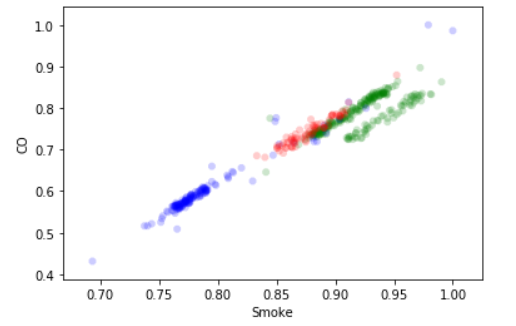
1. Flow chart of all activities

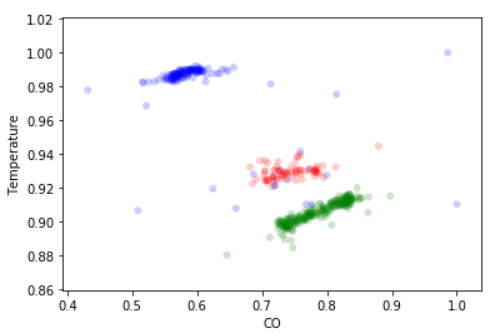
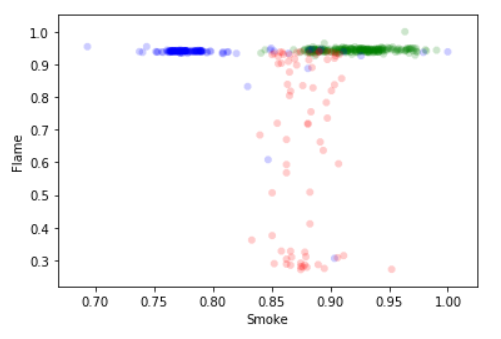
Main Activity is the main menu and it supports the function of Log in with username and password to authenticate the account. However, if user input the wrong username or wrong password, it will show the wrong message. Meanwhile, it supports the transfer from this activity to Register activity by clicking the button of Register. Register activity supports the user writing down username and password to apply for an account in the back-end server. Login is the activity after authenticating the account of the user. In this activity, user can add device by scanning to the server to accept the information from the associated sensors and specify the location of that device. Besides, MQTT connection is kept to back-end server to accept fire alarm information or low battery information. Scan is the activity after user clicking the button of “To add a new device” button in the login activity. It supports user writing the device ID or scanning QR code to get the device ID automatically and specify the location of that device. Then, submit this bundled information to the server to require the data from that device. After logging in, user will keep MQTT connection with server. If receiving the fire alarm message, the UI will transfer to alarm activity to show the fire message and its location. It supports closing the fire ring from app to click “close the alarm” button. Meanwhile, if the user receives the low battery message, the Ui will transfer to low battery activity and show the device ID and location of that device.

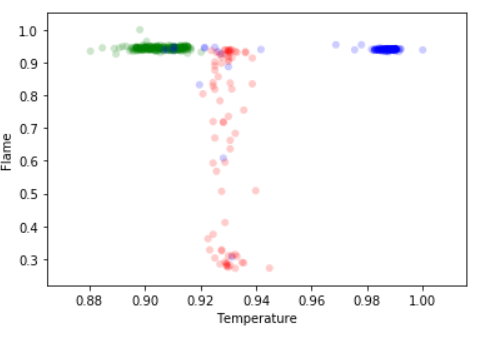
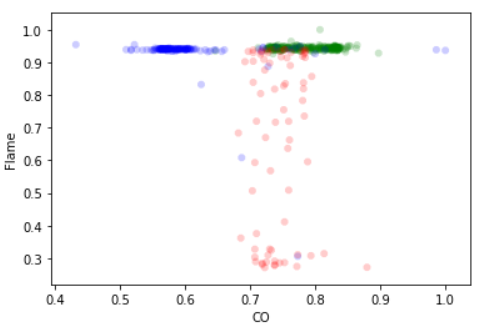
# EXPERIMENTED RESULT

After constructing the whole system, we collect related data through sensors in the artificial scenario, fire, cook and normal. In order to eliminate the effect of size, we normalize the data first by dividing the maximum. All the data collected are below:

Green denotes there are no fire happen, Red means fire occurs, Blue represents data collected on cooking.

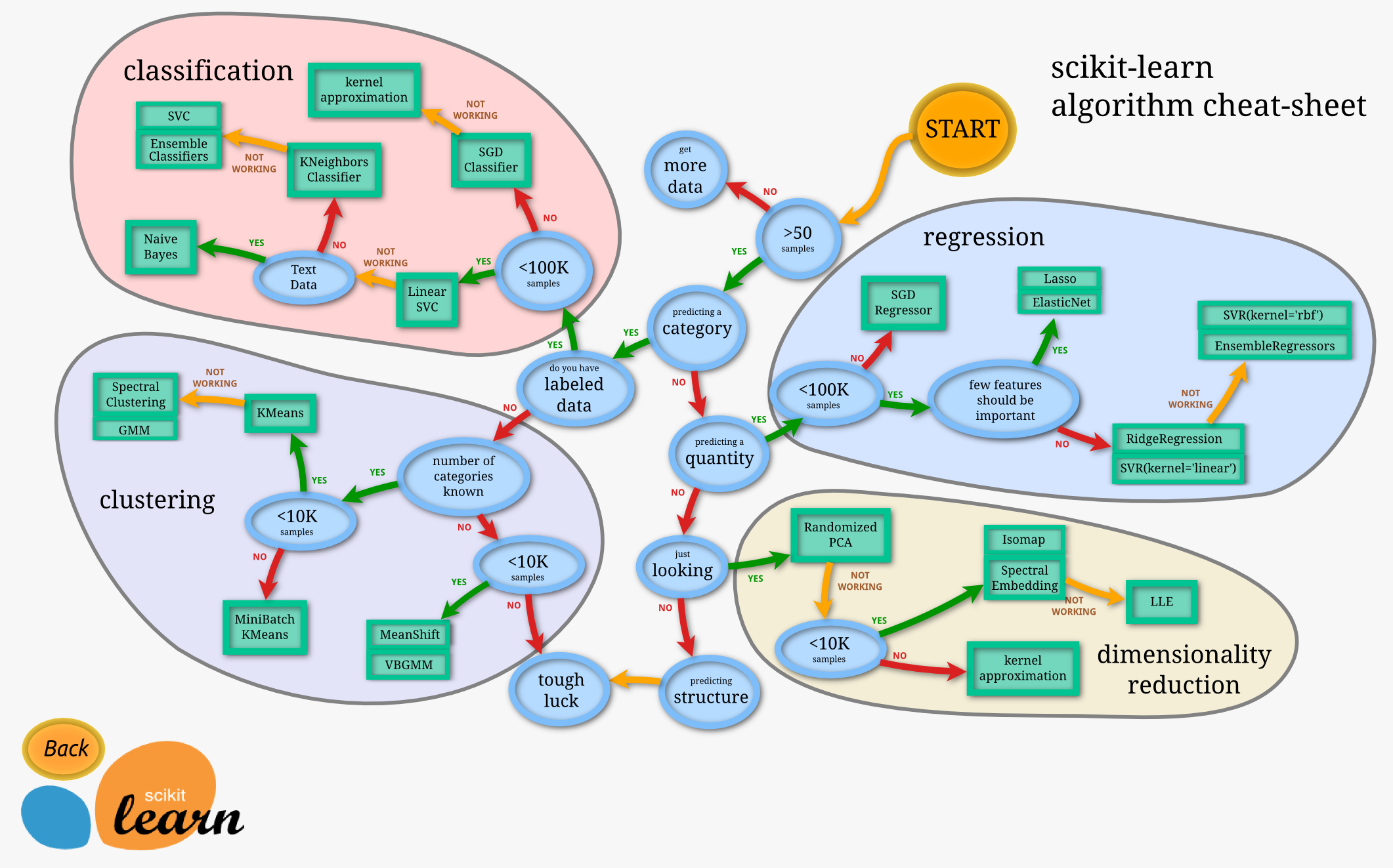




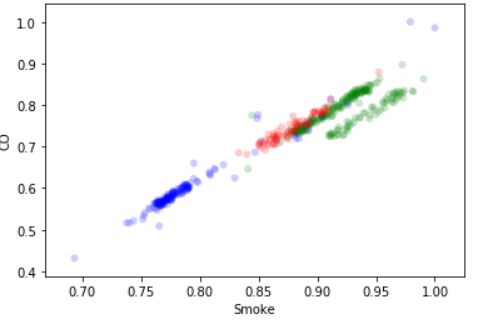


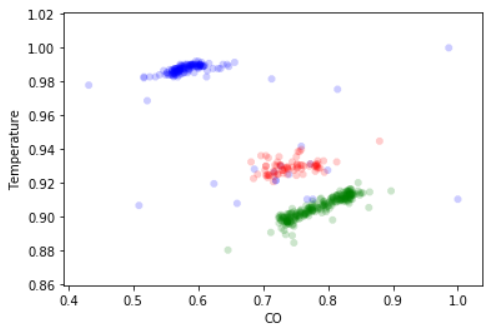
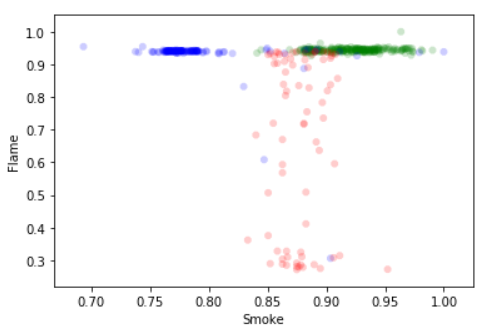
1. Normalize data

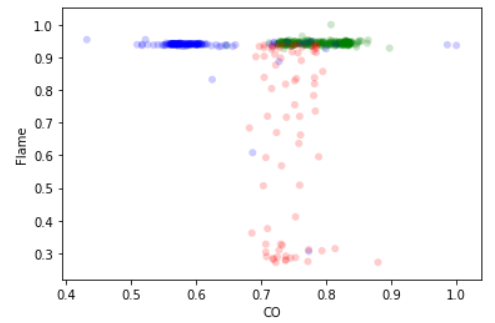
Because of the limited data we collect, SVM is most suitable algorithm for our model as the Fig. shows. From the scatter plot, we know that CO and Smoke are linearly correlated, and thus one of which is redundant data. Obviously, it’s linearly separated.



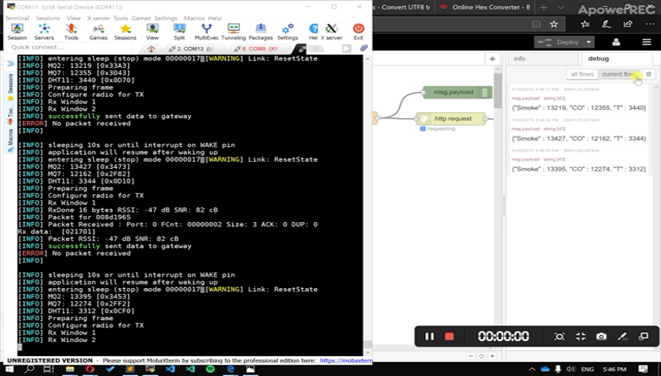
In the offline mode, we train the data based on SVM and then deploy the model on the AWS lambda function. The result is below:







1. Prediction



1. Running log of Serial port communication with x-dot

A screenshot of a social media post

Description automatically generated

1. Watch cloud of AWS lambda function

A screenshot of a cell phone

Description automatically generated

1. Log of AWS MQTT broker

# Conclusion

In this paper, we firstly do the survey about market opportunity. From the survey, there are still increasing demand in fire detector. However, the majority of existing products are not programmable. People want to easily open and close the alarm and check if the battery runs up. Therefore, we build a smart IOT fire detector. Taking cost, accuracy, management, we design three layer system: Physical layer, Cloud layer and Interface layer. In physical layer, sensors collect data and sends to conduit. Then conduit encode and decode data then send data to cloud layer. In the cloud layer, we use AWS cloud service to process data them transmit it to Interface server. Users could control the sensors in interface layer. For example, user could close the alarm remotely or they could get the battery information in mobile app. From the result, the system become more efficient and useful.

References

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
5. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
6. Perez, Ugaitz Amozarrain. "Low power wi-fi: A study on power consumption for internet of things." Master's Thesis, Facultat d’Informàtica de Barcelona (FIB), Universitat Politècnica de Catalunya (UPC), BarcelonaTech (2015).
7. <https://www.bluetooth.com/specifications/bluetooth-core-specification>.
8. R. Angeline, Adithya S, Abishek Narayanan, “Fire Alarm System Using IOT,” International journal of innovative technology and exploring Engineering vol.8, April, 2019.
9. Majid.B, Nirvana.M,etc., “Automatic Fire Detection,” Pervasive System Group, University of Twente.
10. Jia.J, Zhe.G, etc., “Research on Fire Warning Program of Cotton Warehousing Based on IoT Technology”, Institute of Economics and Management, Hebei University of Science and Technology.
11. S.R. Vijayalakshmi, S.Muruganand, “Internet of Things technology for fire monitorsystem,” International Research journal of Engineering and Technology vol.04, Issue: 06 June, 2017.