Firefighter Sensor for CMPE 220

ZHEMIN SU, WENYUEN CHAO, LU TANG

***Abstract*—The Smart Sensing is a mini portable device that can be hung on the firefighters clothes. The device is capable to collect data about device location, fire situation and air quality information include carbon monoxide, methane, etc. Those data will be collected and display through the web to allow fire-station command center to view the real-time fire station. Therefore, the firefighters can save the valuable time to avoid keep reporting the fire station the command center and focus on their jobs at hand.**

***Index Terms*—Smart Sensing, Localization, Fire Detection.**

I. Introduction

THE fires happened frequently in California because of the dry weather. The 2018 wildfire season is the most destructive season in California history. According to the California Department of Forestry and Fire Protection and the National Interagency Fire Center, there are total 8,434 fires burning an area of 765,033 ha. The fires have caused more than 3.5 billion in USD Dollar. The cost of Cal Fire Department is huge.

Currently, most of the firefighters still using the Handheld Two-Way Radios to communicate with the Fire-Department command center. This communication method is not very efficiently. Especially, if the firefighters need to use two hands to carry people or some heavy items, it is conveniently to use Radios to communicate. Meanwhile, the functionality of the Radio is only limited to the communication. Our device can help firefighters to automatically send data about air quality, fire situation and location info to the command center, which can help firefighters to focus on their jobs at hand.

The whole project can be splitted into three parts: hardware, image process and web. For the hardware, it requires a small single-board computer with various sensors and modules to detect the air quality. All the real-time data will be sent to the server. After data collection, the result will be stored into database and display through the web UI. For the image detection, it will detect if there have any fire using the image process with the picture which is captured from the camera module. For the server, it will be deployed on the AWS for the scalability and reliability. This project has several challenges require us to solve. Firstly, we need to connect three or more sensors in the same board, which may lead the compatibility problem Also, different sensors needed the distinct environment, which means our target board have to meet different demand. Secondly, how to determine the fire level based on the cameras image using computer vision will be our biggest challenge. We will use color and motion feature to detect fire, and moreover try to classify the level of fire. Currently, all the computer vision library can only detect the fire, however it cannot distinguish the serious level of the fire. We will focus on this part to build a new library to implement this feature.

II. Related work

A. State-of-the-art solutions, competition, risks

There are three parts in our project, the hardware, web and deep learning with fire detection. The dependency of three parts is not very strong. Therefore, the teammates can focus on their parts at first, then collaborate to connect all different parts together. The risks of our project have many parts. Firstly, since the device is a mini portable device, the require the power to supply the device. The power capacity will be one of our consideration. Meanwhile, the larger the power capacity, the larger of the size. We also wish the size will not too big, otherwise it will be inconvenient. Secondly, we run deep learning in raspberry pi to do fire detection. The raspberry pi will use the camera module to capture the image, then deep learning problem will detect the image, if there have any fire detected from the pictures. The accuracy of deep learning algorithm will be one of our risks. Also, running deep learning with tensorflow in raspberry pi will be quite slow. Since the CPU is not very powerful and RAM is very limited from raspberry pi. The image detection delay and resource limitation may cause the fire detection longer than we expected. Thirdly, for the web part, the web servers are deployed on the aws for convenience access. If the aws is crushed, our web page will be unavailable to view.

B. The innovation design

1) Use cases: About use case, we will focus on the functions of frontend. There are two main basic functions. One is data of smoke and another is position of Fire in Google Map. Once the user sign in the system, he can get all the information of fire status.

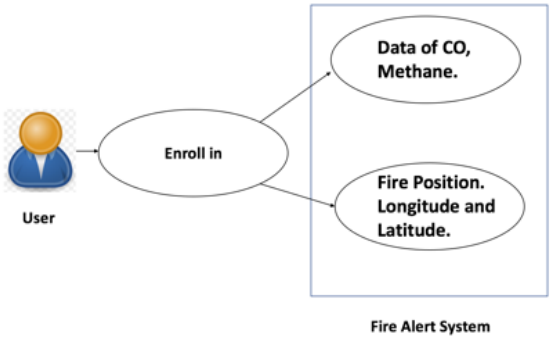


Fig. 1. Use Cases

2) Application: Talking about application, our product is very suitable to be a wearable device. For example, this product can be easy to carry by user and detect the fire situation anytime and anywhere. Also, we provide a very functional user interface (UI), which show all the data environment, like CO, methane, and GPS position, so we can monitor the fire situation in a real time.

3) Key technical challenges: The basic challenge of this project is about the real time. Since our application is used in emergency situation, so the real time data is necessary. However, Raspberry Pi 3 and Arduino dont provide RTOS. Hence, there is a delay time when receiving and sending data. Also, concurrent is also a challenge in this project. When we run the code, we found that data shows in the screen without static order, which is also a issue we need to fix.

4) Our Focus: In this project, we focus on two parts, cooperate of sensors and UI. About the sensors part, totally have three modules, smoke sensor, GPS sensor, and SD card reader. Thus, how to combine these sensors into a same driver code becomes very critical. Fortunately, we have developed a driver code, which can show all the data information from each sensor.

5) Solution overview: In our project, we used raspberry pi to handle all the data from sensors and camera. When all the data is ready, we set to server and save on the database. We had smoke sensor, GPS, sim card and camera to capture all the data. And we used JavaScript to develop front end, node.js to implement backend. Also introduced Convolutional Neural Network for fire image detection.

6) Innovation overview: We deployed our website to AWS and use Mongo dB to store our data. As for hardware part, we used Arduino to control all the sensors and use sim card to communication. We cooperated HTTP post request with our CNN fire detection module, so that we can send the result from CNN module to server.

III. TECHNICAL DISCUSSION AND R&D

This project [Figure 2] have 3 parts, the first part is hardware, which is responsible for collecting the data from different sensors and send those data to the server. second part is computer vision, we use computer vision to detect fire situation from camera module. The last part is web, all the data from hardware will be collected by the web server and those data will be display in the web through the google map api.

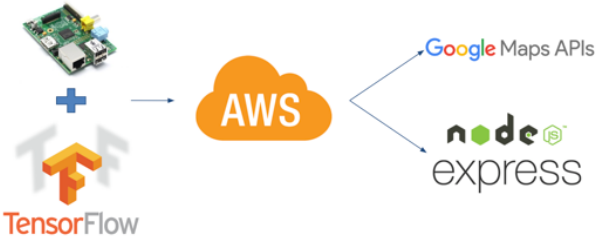


Fig. 2. Project Architecture

A. Fire image detection

To enable raspberry pi to detect fire situation, we used CNN module. First, we setup the environment for raspberry pi. Then, use CNN module to detect fire image. And when we get the result, send it to server.

Setting up environment is one of our challenges due to limited resource on raspberry pi. Moreover, our implementation for CNN need many packages, such as tflearn, tensorflow, computer vision. When we install these packages, every step needs to be very careful and meanful. For example, conda is needed so that using python will be easier, so we install miniconda instead of normal conda.

CNN architecture is in [Figure 3]. This architecture helps us can automatic detection of fire pixel regions in video or image, not influenced by temporal scene information. The main point is breaks down the frame into segments and performs classification on each superpixel segment to provide in-frame localization.

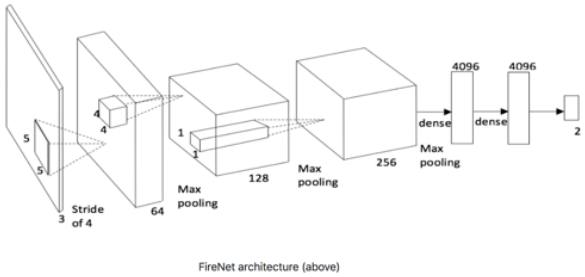


Fig. 3. CNN architecture

Once raspberry pi finished running module, it wound get a result said this image has fire or not. This result would be sent to server and store in MongoDB.

In real world application, firefighter can put raspberry pi with camera and sim card in the fire region. Camera will take the picture, and raspberry pi will run CNN module to detect fire. Then, raspberry pi with sim card can send the result to server. Commander from fire station can see the result and give command depends on result. For example, sometimes firefighter will leave when the fire is taken care of. However, the fire maybe come back and destroy things. In this situation, our device can stay in the region and detect the fire even firefighter is not there.

B. Web

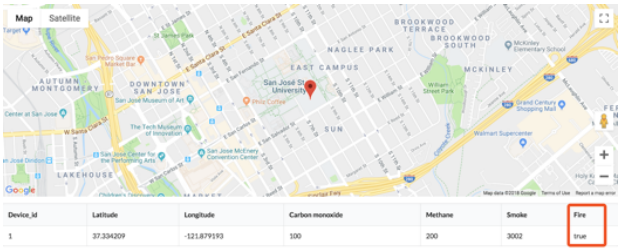


Fig.4. website with table and google map

The web page aims to let command center to view active device information. The website [Figure 4] use HTML5, CSS3, Javascript, ajax and Google Map API to allow user to login or register the account, view each active devices location and air quality information which is collected by that device sensors. Below the google map, there is a table to display each devices detail information which include device id, latitude, longitude, carbon monoxide, methane and fire detection result.

The web server use node.js with express framework. The server has Three exposed apis, the first api is use to receive data from hardware about latitude, longitude, carbon monoxide, methane. Those data are collected from the Raspberry Pi 3 sensors. The second api is use to receive data from image detection result. Because running deep learning in Raspberry Pi 3 with tensorflow may have some delays according to the

limited hardware resources from Raspberry Pi and the delay from capture the image from camera module, therefore it requires a extra api to receive the result of image detection. The third api is using to retrieve information from DB.

For the database, we use dynamoDB from AWS to store all the data which we received from hardware device. The Amazon DynamoDB is a fully managed NoSQL database that supports key-vale and document data structures. Which is very flexible and scalable for large data processing. Inside the database have one table called DEVICE LIST with sevens fields: $Device\_id$, $Latitude$, $Longitude$, $Carbonmonoxide$, $Methane$, $Smoke$ and $Fire$. The primary key for the database is $Device\_id$ because of each device have unique device id.

C. Hardware

In our hardware part, our main processor is Raspberry Pi 3 model b and the microcontroller is Arduino Uno. Also, there are three sensor module are controlled by Arduino, which are smoke sensor, GPS sensor and SD card reader. First, we will collect data from these sensor every 30 seconds. Second, we send the data to Raspberry Pi, because Raspberry Pi has WIFI. Last, sending data to AWS server for analyzing by HTTP protocol. The benefit of this architecture is that we divide the hardware part very clearly. To explain, we seperate it as collecting data part and sending data part, which is easier to manage and maintain. Moreover, we have a webcam, which can take pictures every 30 seconds automatically and be analyzed by machine learning, so our device is very smart.

There are many communication protocols are applied in this system, like SPI, UART, and I2C. First, lets talk about SPI, there are four lines in SPI, which are SCK, MOSI, MISO, SS. To explain, master can control the SS to choose a slave he want and then send the data to the slave. SCK can help us do the synchronization transmission. The advantage of SPI is that it has a higher speed than I2C. Also, we can send and receive at same time.

Next, I2C has two lines, SDA and SCL. SDA is a line for sending and receiving data. SCK is a line for clock signal. In I2C, one master can connect with multiple slaves by using two lines. Each message is limited to 8 bits. The advantage of I2C is that its very simple to implement and support multiple masters and multiple slaves. However, the speed is slower than SPI.

Last, UART has two lines, TX and RX. UART transmits data asynchronously, because it has no clock. For solving this problem, UART uses start bit and stop bit to define the start and end of packet. The advantage of UART is that it is very easy to implement, because it only has two lines. Also, it no needs clock.

IV. RESULTS, EXPERIMENT & EVALUATION

A. Fire image detection

When we give an input to our deep learning module, we will get result about it is fire or not. Just like the image following, it will grid our input image and get the result on the corner of image. Considering image transmission is expensive, we decided not send the image to server, send only true or false result to server instead. So instead of sending the image, we decode the result, in this example fire means true, clear means false, send true or false to back end.

Fig.5. Fire Image result

We also test our CNN module on Data compiled from Chenebert et al. F1 score for this module is 0.90.

B. Web

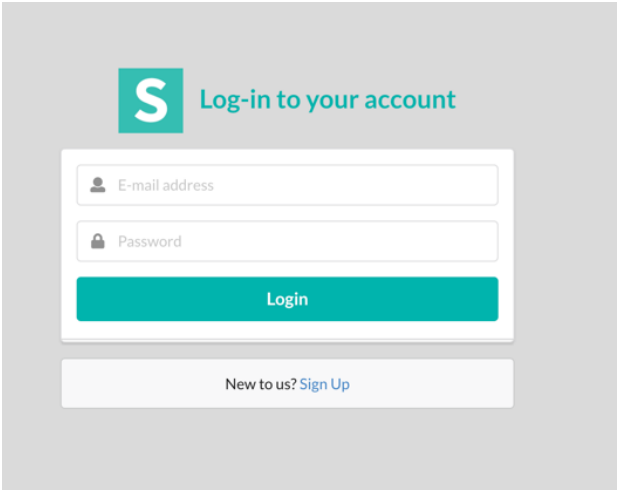


Fig.6. login webpage

The server is deployed into AWS, the AWS have load-balance to handle the different amount of traffic. Currently, the webpage is only under HTTP protocol, for the security purpose. Those data should not be visible for the other people, it is required to build the secure connection with https protocol. The page will do the authentication first to ask user to login [Figure 6], then they can able to view the webpage contents. For the web page contents, the web servers are continuously receiving the data from Raspberry Pi 3, the web page also require to update those real-time data on time. The fronted web page is using ajax with long poll to keep send http request to the server ask for the most recently data every 30 seconds. Meanwhile, using ajax will only refresh small part of the webpage instead of the whole page to save the time to load new content of webpage. We have not have change to test the web servers performance with big data and very high load. Since those problem will be took care of by AWS, and AWS have very good reputation in cloud computing area, is will not be a problem.

C. Hardware

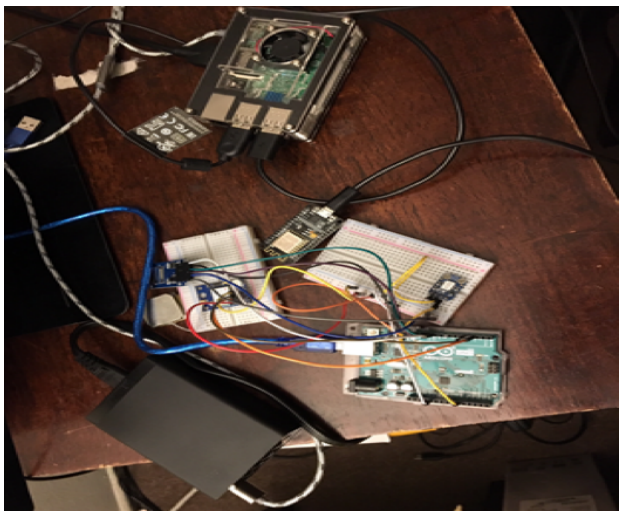


Fig.7. Fire Alert Devices

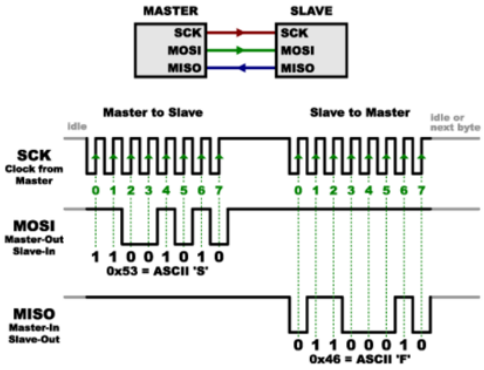


Fig.8. SPI Protocol

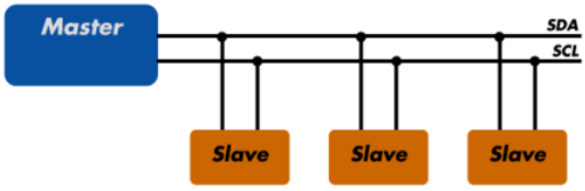


Fig.9. I2C protocol

In this hardware part, there are three tasks we already have finished. First, we can show the data of smoke and GPS in the UI very successfully. To explain, smoke sensor can show the concentration of CO and Methane. Also, GPS module can show the very accurate longitude and latitude. Last, SD card reader can save our history data for doing big data analyzing. Second, the data can be sent from Arduino to Raspberry Pi by UART. The reason we send data to Raspberry Pi is that it provides us WIFI, so we can upload the data to AWS server. Third, after Raspberry Pi receives the data, it will upload this data to AWS server by HTTP protocol. Additionally, we set up a webcam, which can take a picture every 30 seconds, so we can detect the environment in time. In this project, I learn many communication protocols, like SPI, UART, I2C and HTTP. These protocols are very popular and be used in embedded system widely. Each protocol has its own pro and con. To explain, SPI has a higher speed than I2C. However, it is more complicated and less efficiency when we want to connect many devices at the same time. Next, HTTP is a communication protocol, which can help us send data to server.

V. Conclusion

In this project, we already construct a complete model of fire alert system from backend to frontend. However, there are some challenges we still need to conquer. In hardware part, we need to add the RTOS into our system for handle the emergency situation. Also, concurrent issue is a key problem we need to deal with. For the fire image detection, we want to improve the module for detecting the level of fire in future. Furthermore, if we can classify the level of fire with all the data we get from sensors and the images from camera, it will be more useful for firefighters.

REFERENCES

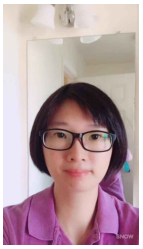
[1] Dunnings, Andrew J., and Toby P. Breckon. *Experimentally Defined Convolutional Neural Network Architecture Variants for Non-Temporal Real-Time Fire Detection.* 2018 25th IEEE International Conference on Image Processing (ICIP). IEEE, 2018.

[2] A. Chenebert, T. P.Breckon, and A. Gaszczak, *A non-temporal texture driven approach to real-time fire detection* in Proc. International Conference on Image Processing. September 2011, pp. 17811784, IEEE.

[3] https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all/

[4] https://www.totalphase.com/support/articles/200349156-I2C-Background

ZHEMIN SU



WENYUEN CHAO

LU TANG