IoT for Cook Stove

TEH ZHE KIAN

441170077

Project report submitted as part fulfilment for the degree of Bachelor of Technology (Hons) in Computer Systems and Networks



School of Science and Technology

Wawasan Open University

December 2019

# Introduction

Food is usually prepared with heat. [[1]](#_References) If the cooking is left unattended, it can result in overcooked food or even worst, it can lead a house fire to occur. [[2]](#_References) The purpose of this project is to provide embedded based system for home cook stove monitoring to increase cooking safety and reduce food waste by reminding the user the stove is on. The monitoring system works by detecting the heat from the food in the pan or the cooktop itself. There are smoke detectors in the market now, but when they go off, it may already be too late.

# Objective

The main objective of this project is to identify correctly the status of the cook stove in real time, whether it is on or off by sensing the heat temperature produced from the food in the pan. For this reason, MLX\_90614 (Non-Contact Infrared Temperature Sensor) is used to sense the food temperature in a non-contact way. After getting temperature reading, the system will then fetch the received temperature reading for data processing.

There are some factors that affects the sensor’s accuracy, for example the distance range from the sensor to the object to be read, and the sensor’s field of view is blocked by steam produced by the food. Those factors can result in reading inaccuracy, so I decide to use the averaged data, which it will derive from the last 1minute reading. This will give the system a more accurate picture of the temperature.

# Problem statement

The proposed system Stove sensor is a Raspberry Pi project that analyzes the temperature of your stove, and notifies users if their stove has been left on for an extended period of time. Additionally, it allows users to check on their stove from any smartphone or laptop. This means you never have to ask "Did I leave the gas on?" again.

A motion sensor (Passive Infra-Red (PIR) sensor) is used to detect human presence. It works by measuring changes in the infrared levels emitted by surrounding objects. The system will keep monitoring for human presence when the cook stove is still operating. Depending on the unattended timer threshold setting, which is set by the user, the system will notify the user if the threshold is met. The user will receive a “stove is still on” reminder via the smartphone and the iot gateway (raspberry pi) will produce a sound beep.

# Project Scope

For this project, only one stove top can be monitored. This is because the MLX90614 sensor being used has a limited of field of view (90 degree). There is some suitable sensor, that is the thermal array sensor. But the thermal array sensor is way costly.

The placement of MLX90614 sensor has to be quite near to the object (pan/food, in this project). This is due to lower accuracy of the OEM model of MLX90614 used in this project.

Throughout the monitoring process, the pan lid cannot be used to cover the pan. Or else the sensor not able to tell the stove is still on or off. Because the system is designed to detect the food temperature and process the data from there. If the pan is covered with lid, the sensor will now read the temperature of the pan lid.

# Literature Review

A basic existing solution to handle fire risk at home is installing fire alarms. The main concern of fire alarms is to detect fire occurrence quickly, so fire rescue agents can intervene in time. However, fire alarms have several drawbacks, particularly for ageing people. These people usually forget replacing alarm batteries regularly. In addition, fire alarms generate false alarms (e.g., in the presence of a small quantity of smock generated by regular cooking). This situation disturbs them, which increases their tendency to uninstall fire alarms at their homes. [[3]](#_References)

Few electrical cooking devices equipped with limited safety features are available in the market. For example, Electrolux INSPIRO oven contains programmable cooking modes. According to the selected cooking mode, the oven calculates cooking time and temperature. TMIO society commercializes ovens with tactile screen, and network connection to be remotely controlled. Numerous manufactures integrate LEDs to indicate that an oven surface is hot to prevent burn. However, the concentration of ageing people is on the cooking task itself and she/he may not notice the lightening LED.

Generally speaking, safety measures are partially considered in the existing commercial cooking devices. StoveGuard, SafeCook and HomeSensor propose a timer system to switch off an oven if there is no attendance after certain programmed time. Still, risks may occur within this period of time. To summarize, existing systems propose numerous interesting features to manage risks at home. However, they have several limitations: they focus on aid for only one specific risk situation, they need to be programmed for each type of use and each time they are used, and they provide elevated risks in the case of cognitive deficiencies.

Our proposed system will only be needed to set up for the first time, and is fully automated for next use, requires little or no user interaction.

# Methodology

Waterfall relies on teams following a sequence of steps and never moving forward until the previous phase has been completed. This structure is suited to smaller projects with deliverables that are easy to define from the start.



Figure Waterfall Model Stages

### Stage 1 : Requirement Analysis

During the first stage of waterfall model, I identified the user requirement. I know user wanted a system that can alert them the stove is still on. I surveyed the sensors, choice of iot controller, and the programming language to be used. The reason why I choose raspberry pi is because it is a low cost, small sized, and support multiple OS to be installed, all these reasons make it very perfect for a IoT project.

Next, firebase is also my choice because, is a cloud-hosted NoSQL database that lets you store and sync data between your users in realtime. NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale. In other words, I do not need to worry about creating the dataset, or retrieving the data using SQL such as joins. It also provide me architecture to host web app, which I do not need to worry about the limited resources of raspberry pi.

### Stage 2 : System Design

This is the high level architecture. The sensor is attached to the raspberry pi and it will produce some data reading. Raspberry pi and sensor will become part of the sensor node. And from the sensor node, the data will be sent to online database via the internet to the Firebase. And the web app (interface) will then display the dashboard to the user. From the dashboard, the user can view the temperature changes either in graph and text form. In future, they will be able to set the alert threshold in timer.

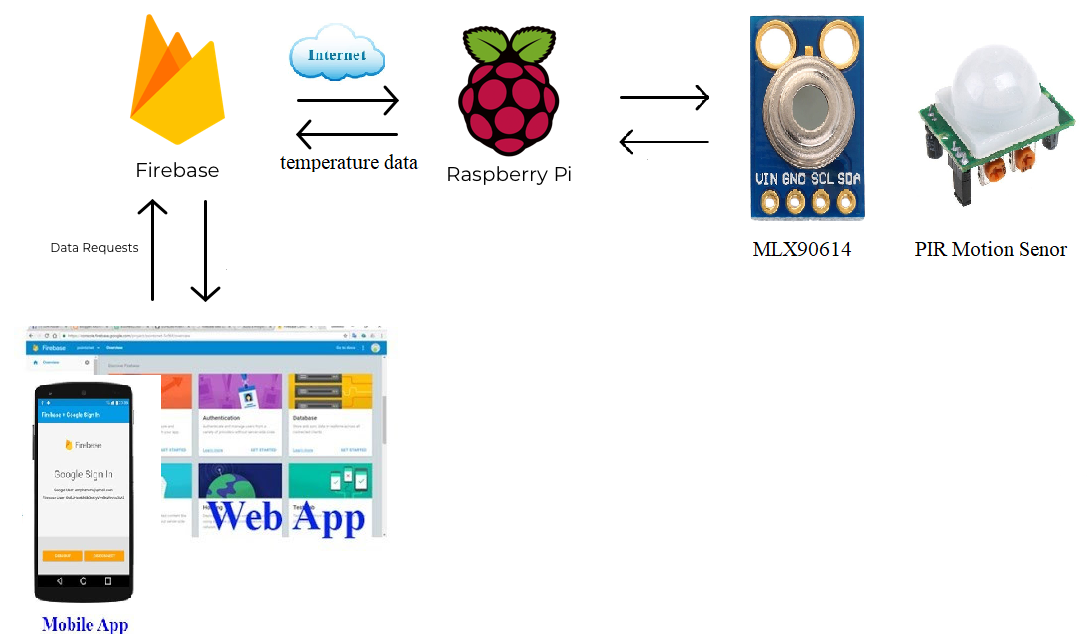


Figure High Level Architecture

For the current stage, this project is only able to do for local storage and access of information. Everything is done locally, for now. The web is hosted in the raspberry pi using Flask, a micro web framework based on Python. And used sqlite3 as the database storage.

In future, firebase will be used, which is the project initially intend to use. Some workaround need to be done to find a way to link local (sqlite3) and internet (firebase) together. Which make the hardware more flexible to use, not only depending on the internet.

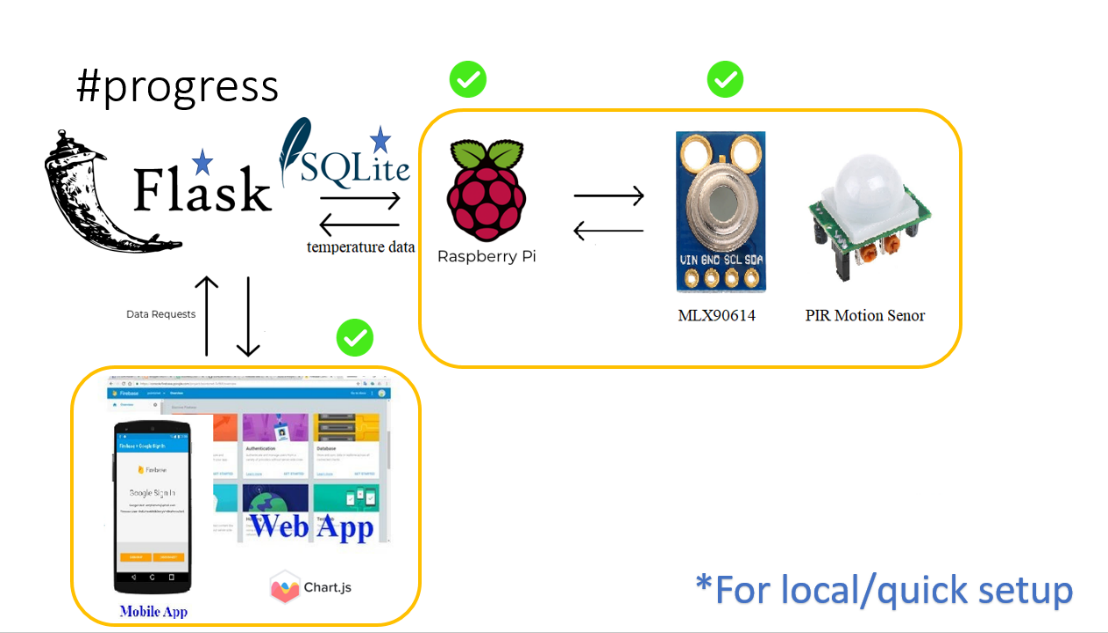


Figure Current progress in a diagram

The database schema is developed using one table and four columns. Sqlite3 is used, because it is very small and light weight, which is very suitable for low resources environment, such as raspberry pi.

Below is the ER diagram and the sample sql statement used to create the table in sqlite3.  
*CREATE TABLE mlx\_data (timestamp DATETIME, ds18b20 NUMERIC, mlx\_ambient NUMERIC, mlx\_object);*

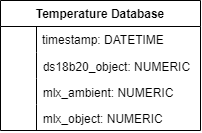


Figure Database schema

When the system starts up, logging application is run in loop to get new data from the sensor in 3 second interval. And the data received will the saved in the sqlite3 database. Website is hosted with Flask and ready to serve the client with website. The website will then display the collected data in a graph with chart.js. And the data processing will run to tell the user whether the temperature is rising/falling and stove on/off.

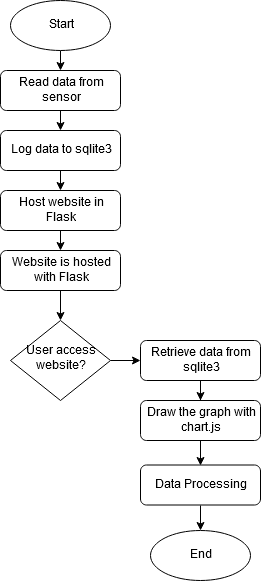
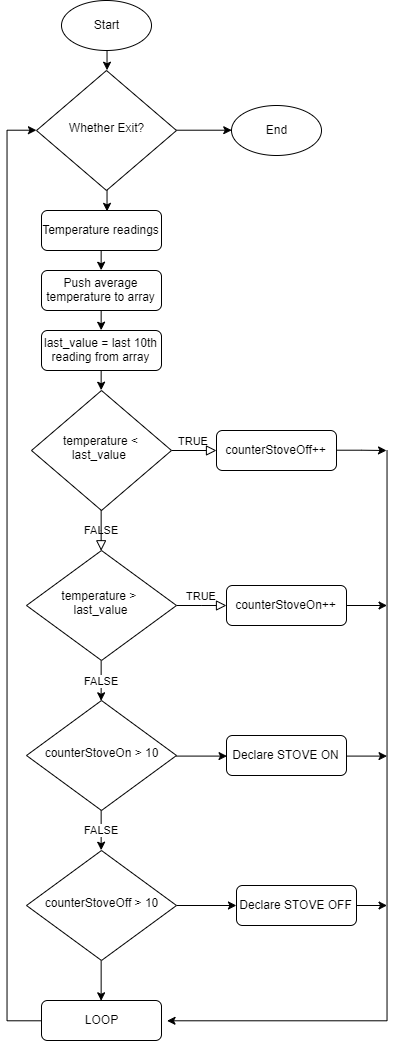


Figure Overall idea of data processing show in a flowchart

Figure Overall system process in a flowchart

### Stage 3 : Implementation

Initially, the ds18b20 temperature sensor is chosen, which is a contact based temperature sensor. This type of sensor is designed to be waterproof, which is what I initially wanted, because the sensor will be needed to place near the gas burner. It can provide accurate reading in wet environment, for example, the water temperature, provided the sensor must be submerged inside the water.

### Stage 4 : Testing

However, due to the contact based design, it do not produce a good experiment result for my project. For example, if the gas burner has been turned down, the reading from the sensor will decrease and near to ambient temperature, which will makes the system thinks that the stove has been turned off. Next, the overall sensor hardware design is made of material that will not endure high temperature such as gas burner in a long period of time. After 1 hour of experiment testing, the cable feels soft and wanted to melt.



Figure Above is the DS18B20 being used, some foil tape is used to fix it near the gas burner.

For that reason, another suitable sensor is here to get the job done. Which is the MLX90614, it is a Non-Contact Infrared Temperature Sensor. MLX90614 is available in either 3.3 volt or 5 volt versions. The breakout board used in this project is the 3.3 volt MLX90614ESF-BAA-000-TU-ND.



Figure MLX90614 Module

The infrared sensor (MLX90614) can read the temperature of an object without touching it by sensing the thermal infrared radiation emitted from it.

The infrared sensor has great potential for producing temperature results without being in contact with the object of interest. If the object fills the field of view of the sensor (90°), the accuracy of the IR sensor seems to be similar to a contact sensor. Which requires the sensor to be very near to the radiating object.

Which turns out very suitable for the project, the sensor is set with the position above my pan and stove. Which is safe from high heat.



Figure Sensor position in setup

The DS18B20 sensor is used as a reference point (red legend) to sense the food temperature and compare with the MLX90614 (blue legend). Next, populate the reading into chart.js. And it turns out the reading not accurate enough, this is because the MLX90614 used in this project is OEM brand. But in overall, the temperature trend is still there, the system still can know the temperature is still rising or falling after some data processing and viewing from the data graph itself. So, MLX90614 will continue to use this sensor in the project.

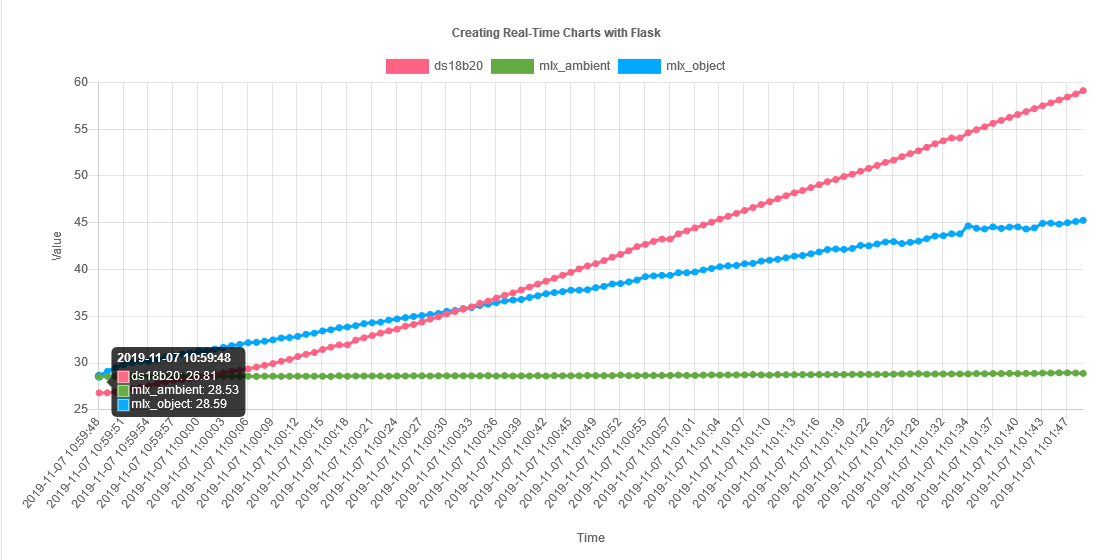


Figure DS18B20 vs MLX90614 Accuracy

### Stage 5 : Deployment

When the testing above is done. The project is then ready to deploy and ready to be used by the user.

Below is the Proposed working of temperature condition and the reported status

|  |  |  |
| --- | --- | --- |
| No. | Conditions of temperature readings | Stove Status |
| 1 | Temperature keep increasing for at least 10 readings | Stove ON |
| 2 | Temperature keep decreasing for at least 10 readings | Stove OFF |
| 3 | Stove left on for unattended period (no motion detected) | System send alert reminder |
| 4 | No action taken by user (still no motion detected) after the reminder sent. | System shut down the stove automatically |
| 5 | Motion detected while the stove is operating | Reset the timer for sending reminder |

### Stage 6 : Maintenance

This is the last stage of waterfall model. If the user found any issue with the project, the problem will be hot-fixed with the next planned release.

# Progress

Since the initial report, various achievement has been done. It may divide into 3 main part, reading data from sensor, storing the data into database, retrieve the data from database to the client interface.

## Reading data from sensor

The sensor has been successfully connected to the raspberry pi physically and it is ready to be used. Searching through the raspberry pi for raspberry pi’s 40 GPIO pins is tedious job. Luckily, there is some solution in the market, which is the breakout board. The board has the list of label number, which makes the connecting sensor to the raspberry pi job easier.

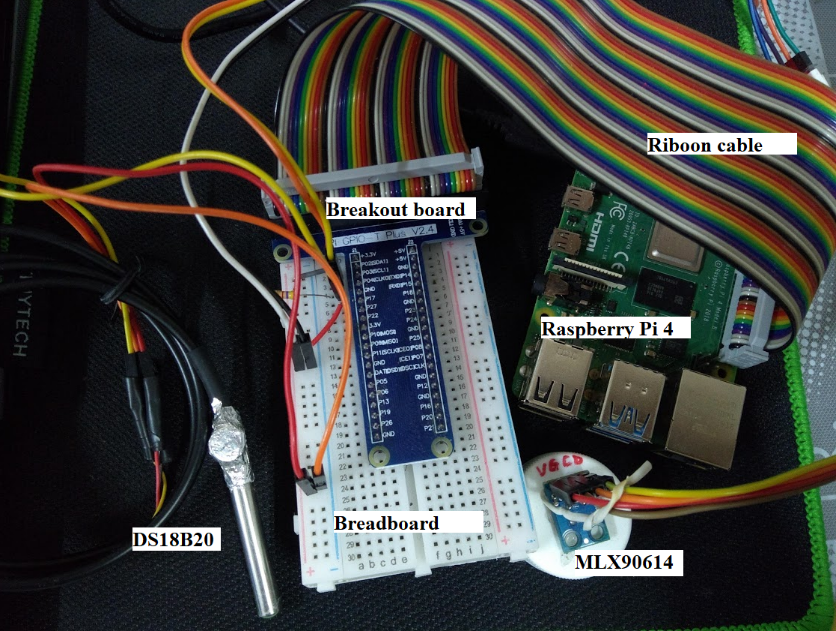


Figure Sensor connection

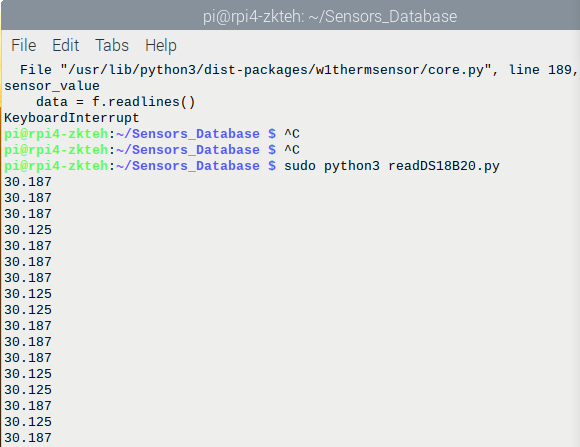


Figure Using library to read the sensor

Above is the result of temperature reading output from the sensor using the library called w1thermsensor. The library is available online and it’s free to use. With the library, system can just initialize the library with a variable, and call the function in a simple way. For example, sensor.get\_temperature().

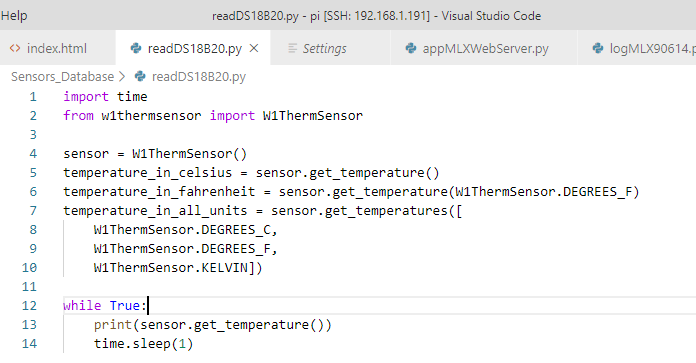


Figure Soure code of a simple sensor reading

## Storing the data into database.

After knowing how to read data from the sensor. The data can now be stored in a database. For now, the data is now stored into sqlite3, which a lightweight database. Lightweight is important in a resources limited environment, in this case, the raspberry pi.

The variable, such as ds18b20, mlx\_object is feeded with data, and these data will be store inside sqlite3 in a loop and with a 3 second on hold. Which means the data will only be store in database in a 3 second interval.



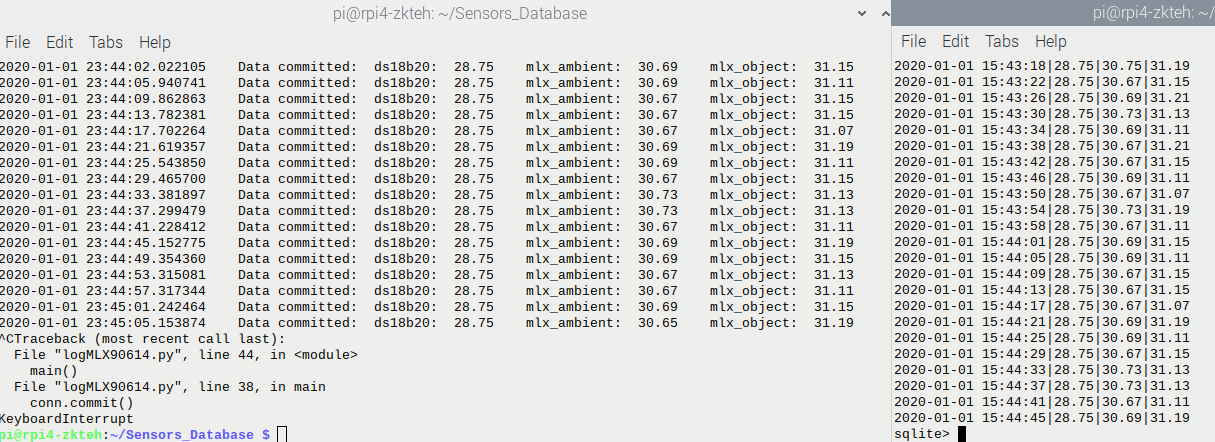


Figure Left one is the logging process. Right one is the sql query being run

## Retrieve the data from database

JSON is used to transfer the data from database to the web app, and eventually to the user interface. JSON is useful, because it is a lightweight format that is used for data interchanging. Below is some code from the app logic hosted with Flask (micro python) web framework. The first two line is to retrieve data from database using sql query. Then, jsonify to convert the variable into a JSON representation



Figure Part of the source of the web app.py for Flask

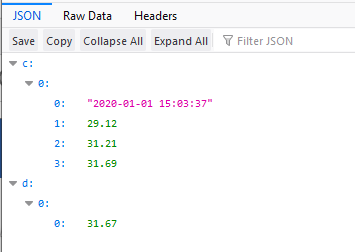


Figure Aboveis using Firefox to view the structured JSON data instead of the RAW data dump.

## Data processing

Because data processing need at least two value to compare, and if the two variable keeps getting the same value, then the comparison cannot be done. Thus, I solved it with delay the reading for *last\_value* by inserting zero value to it during the first loop. Now, the *temperature* will get the newer data than the *last\_value.* And the comparison of data can be done.



Figure last\_value will now hold the previous value of data

A simple data processing has been done. This is the core of the device, which it will detect for rise and fall of the temperature being read by the sensor node (raspberry pi + sensors).

First, I will use average temperature as my temperature variable. Then push temperature into an array. Last\_value will be the last 10th value in the array. Which means now the *temperature* is the latest reading while *last\_value* is the old data. To make the data processing more accurate, I will pick the *min* and *max* from the previous array.

Using *max* and *min* is very important to conclude that from the last 10 reading of array, the temperature is indeed increasing or decreasing. Because if only use the *last\_value* which is only one reading, as the comparison with *temperature* might have some influences such as reading fluctuating.

Below is some part of the data processing algorithm.



Figure Data processing is done using the code above

# Conclusions

In my opinion, the wrong choice of sensor (DS18B20) makes me feel under motivated. But luckily with the help of SST lecturer, Dr.Tan Choo Jun, Dr Mohd Hezri Marzaki and Mr. Muhammad Norhadri Md Hilmi. I finally tried a different sensor, and it turns out it is performing well than I expected. The choice of sensor in IoT project is very important as it will stuck me for quite a while before I able to proceed with the data processing and more things to do.

In the next progress, I will try to add firebase connection. Internet connection will the heart of any IoT project. And after that will try to enhance the data processing. Because the data processing algorithm is still in baby stage and need some fine tuning.

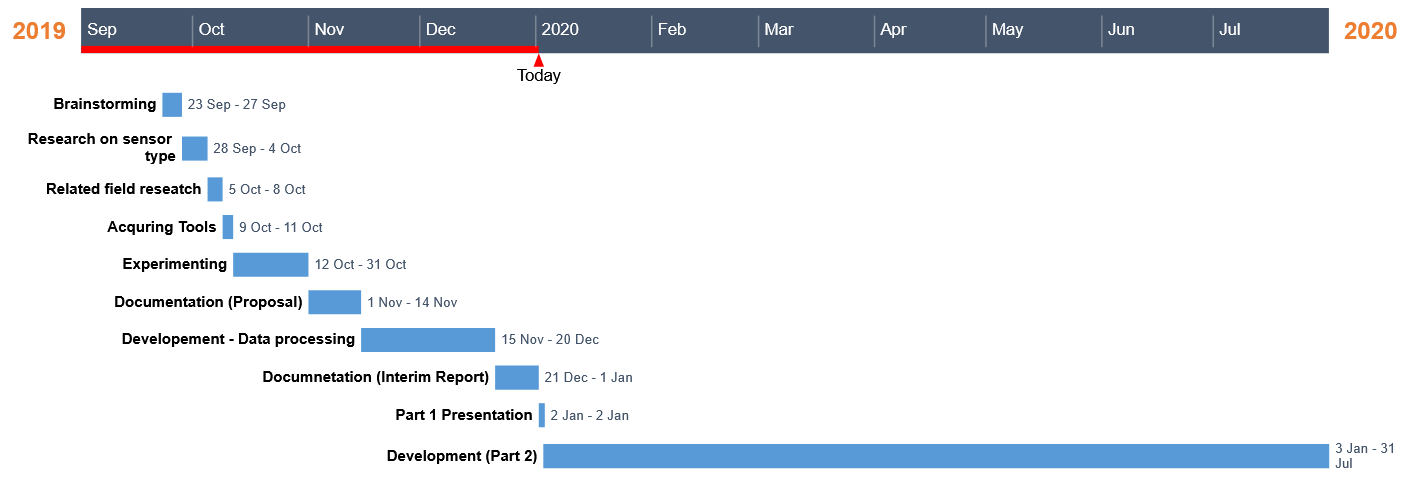


Figure Overall timeline

Below are my proposed list of chapter headings for my final report.



All the source code and related files for this project can be accessed at the following link:

<https://github.com/zkteh/myStove>

# References

[1] Rupp, Rebecca (2 September 2015). "A Brief History of Cooking With Fire". National Geographic. Retrieved 31 December 2019.

<https://www.nationalgeographic.com/culture/food/the-plate/2015/09/02/a-brief-history-of-cooking-with-fire/>

[2] Ahrens, Marty (November 2018). “Home Cooking Fires”. National Fire Protection Association. Retrieved 31 December 2019.

<https://www.nfpa.org/-/media/Files/News-and-Research/Fire-statistics-and-reports/US-Fire-Problem/Fire-causes/2018-Home-Cooking-FIres--Report_FINAL.ashx>

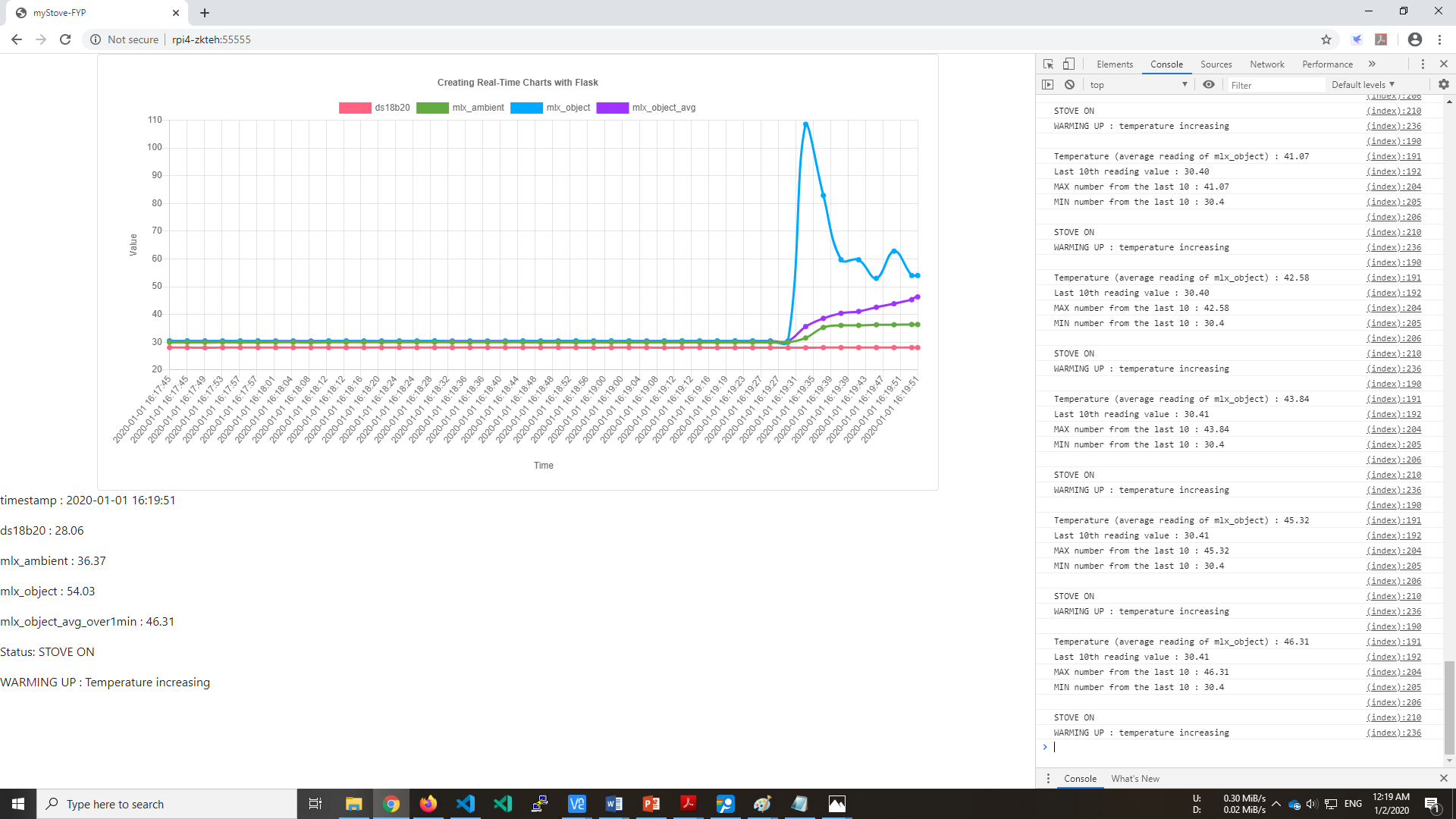
[3] Ahrens, Marty (Jan 2019). “Smoke Alarms in U.S. Home Fires”. National Fire Protection Association. Retrieved 31 December 2019.

<https://www.nfpa.org/-/media/Files/News-and-Research/Fire-statistics-and-reports/Detection-and-signaling/ossmokealarms.pdf>

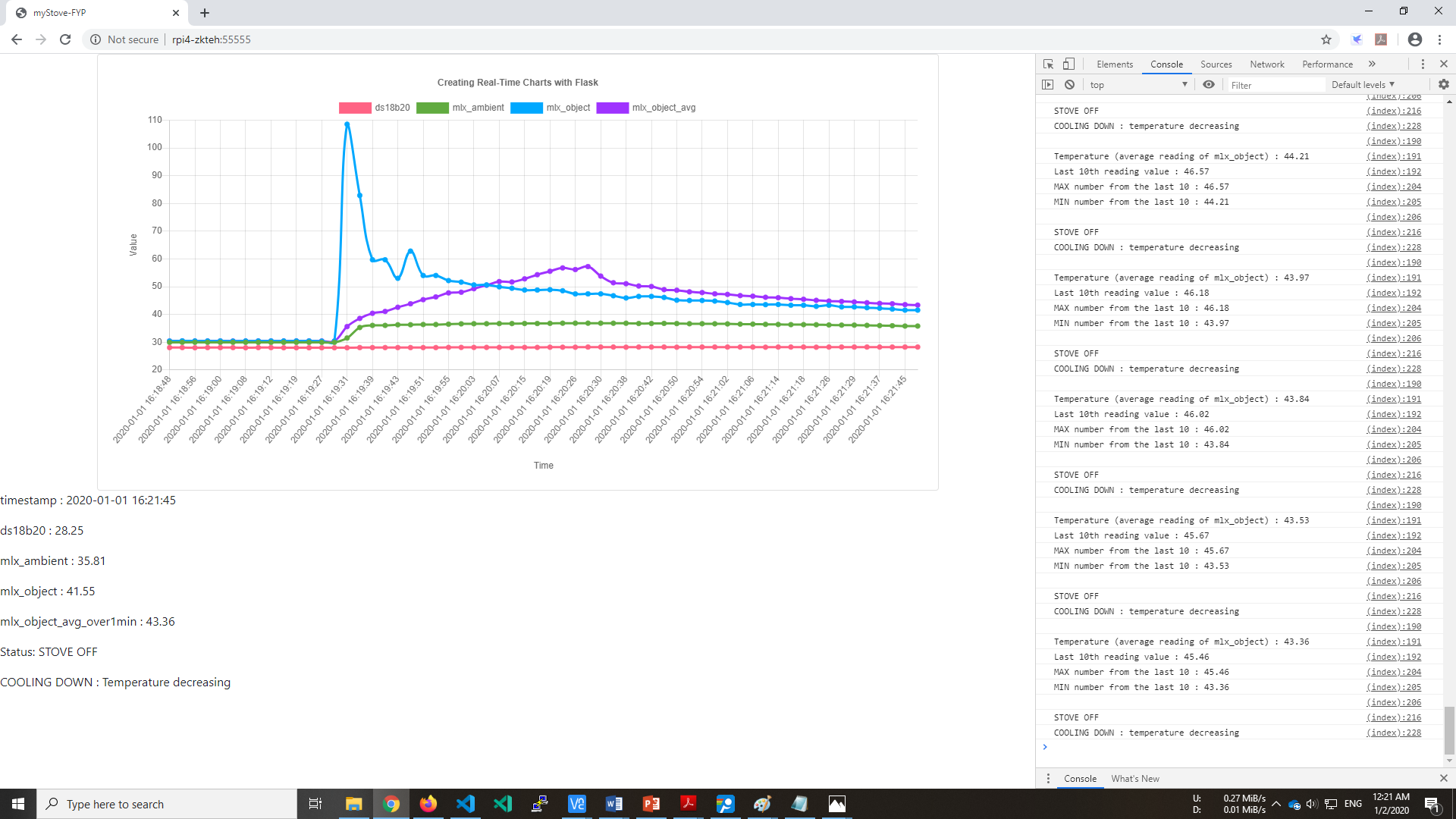
# Appendices



Figure Above Is some debugging message for identify some bug



This is the dashboard for the user. The grah is done using chartjs. Website is hosted with Flask. The data is fetch from sqlite3. And data is processed in the client side. The result show the temperature is increasing.



The result show the temperature is decreasing.