CHAPTER 4
ANALYSIS AND DESIGN

4.1 General Explanation

This chapter discusses the five main parts of the Fire Detection project in detail:

1. Fire detection part.
2. The location determination part (GPS receiver).
3. Wireless data connection to the server.
4. Section SMS server Gammu.
5. The nearest fire search section.

4.1.1 Fire detection part

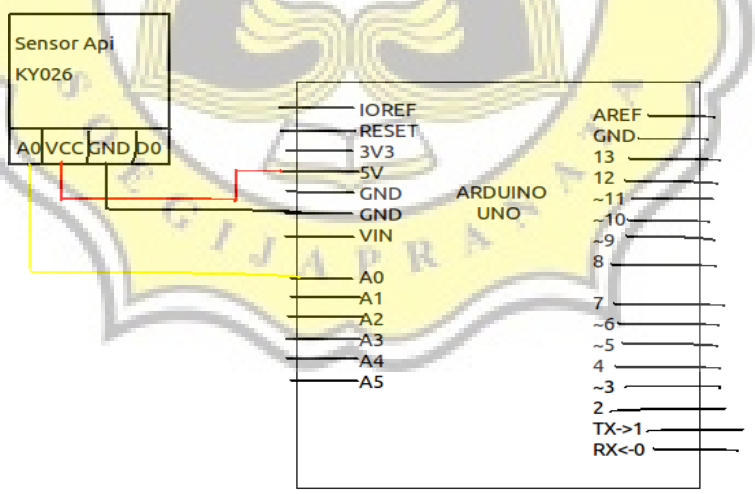

Illustration 4.1: Fire Sensor KY-026

The fire sensor KY026 in section A0 aka analog is connected to A0 Arduino Uno to extract the data value from 0 to 1023. Using analogue because it can set the desired value, here use value 70 on the flame sensor to detect the

flame. VCC is connected to a 5V voltage because the KY026 fire sensor uses 5V voltage to be used properly. GND is connected to GND Arduino uno to provide negative voltage.

Here's a flowchart of how KY026 works:

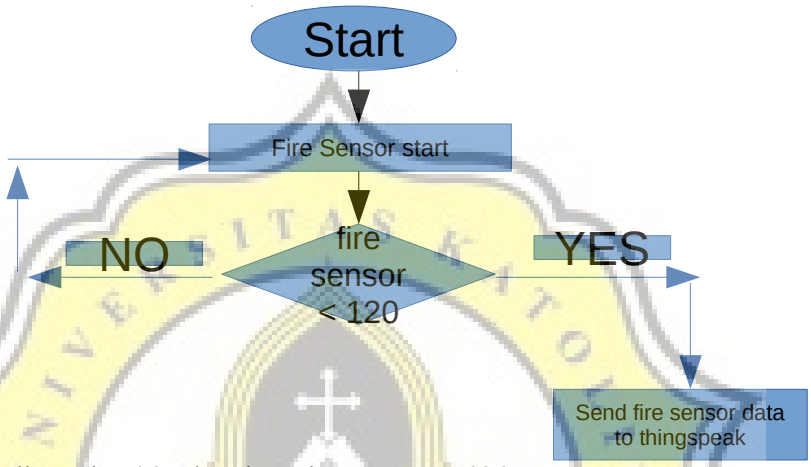

Illustration 4.2: Flowchart Fire Sensor KY-026

Here when the Arduino is run, the fire sensor will work to detect the presence of fire. If the fire sensor valuenya less than 70, it will send the fire sensor data on thingspeak by using ESP8266.

4.1.2 Location determination section (GPS Receiver)

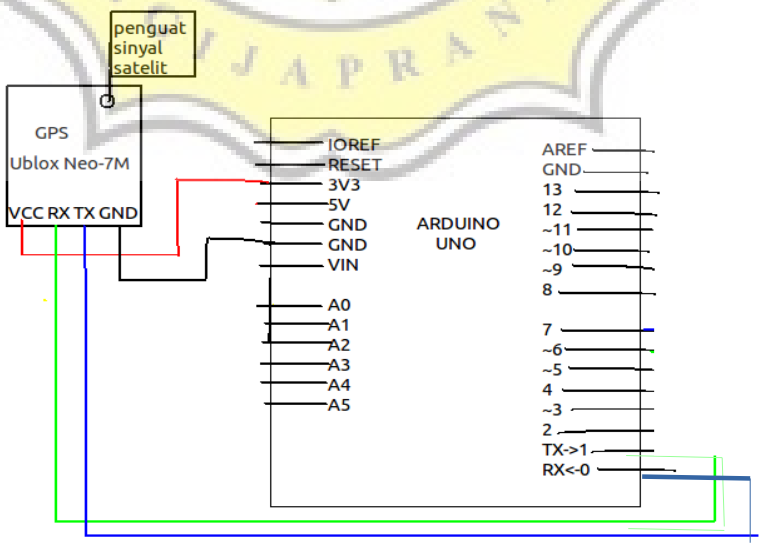

Illustration 4.3: GPS Ublox Neo-7m

This GPS takes approximately 1 - 3 minutes to get a satellite signal so it is better if not blocked by many walls.

Here's a flowchart of how GPS works Ublox Neo-7M:

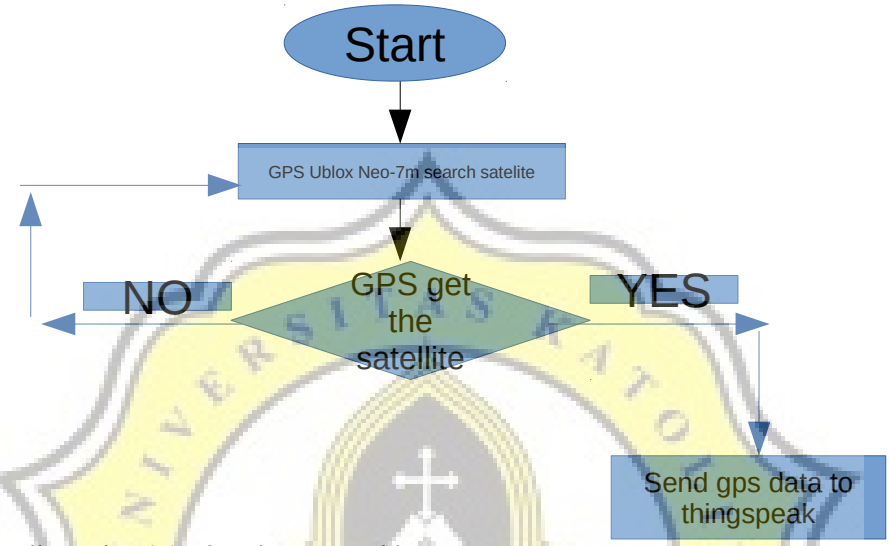

Illustration 4.4: Flowchart GPS Ublox Neo-7m

When the Arduino is turned on, wait for approximately 1 to 3 minutes for the Neo-7m GPS receiver to get the satellite signal. When you get a satellite signal, the GPS will display latitude and longitude.

4.1.3 Wireless data connection to the server

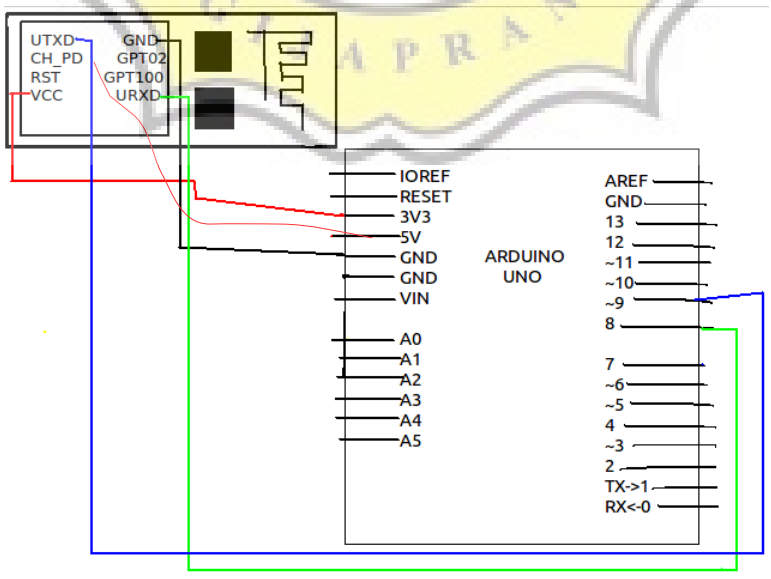

Illustration 4.5: ESP8266

ESP8266 is used to connect arduino to the internet so it can transmit data. The data will be sent to the web server Thingspeak.com and then stored on the web server. The data stored in this project in the form of fire sensor data, data latitude, and data longitude.

Here is the ESP8266 flowchart:

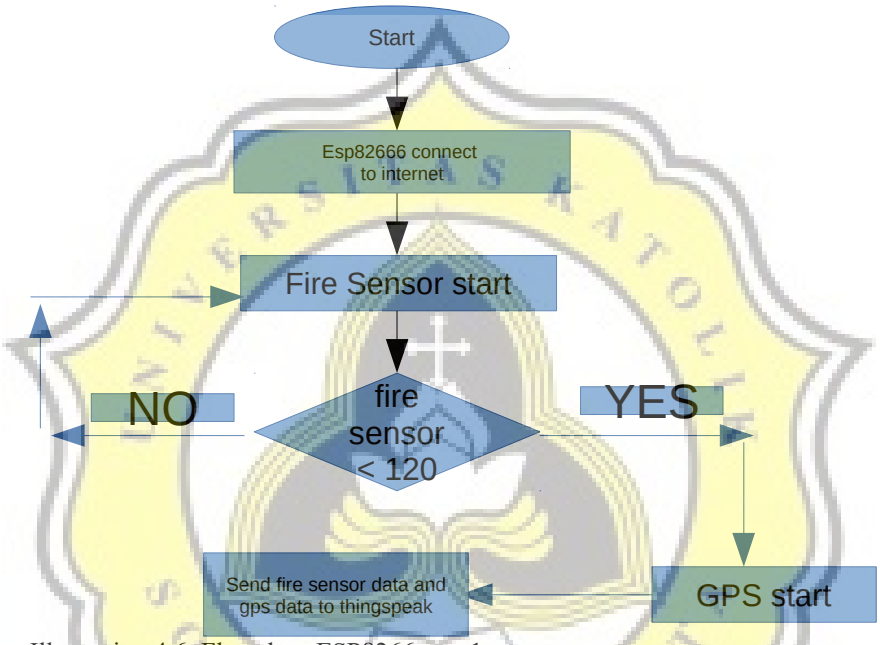

Illustration 4.6: Flowchart ESP8266 part 1

In the ESP8266 connect to internet process there is a flowchart as follows:

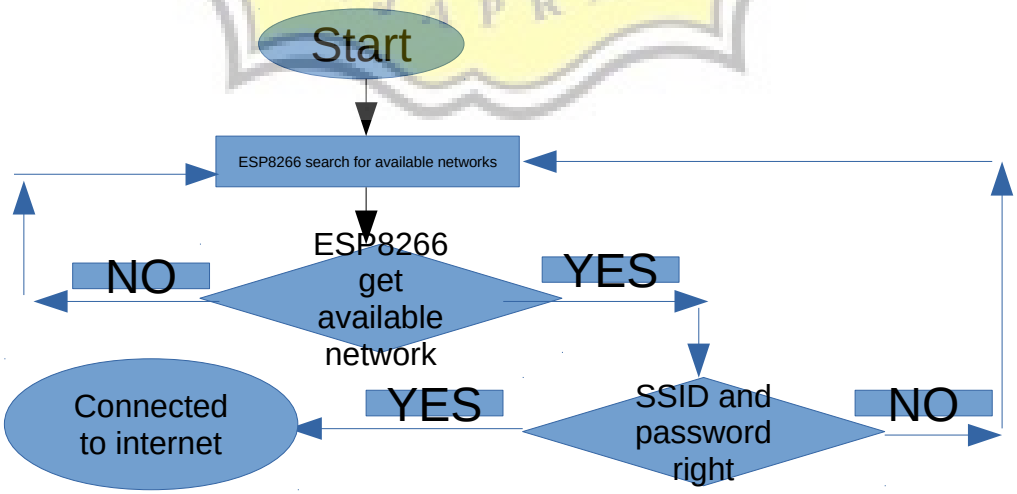

Illustration 4.7: Flowchart ESP8266 part 2

When Arduino Uno turns on, the first run is ESP8266 in order to connect to the internet to send data to thingspeak.com. First ESP8266 will search for available networks, otherwise ESP8266 will keep searching until it can, then if it can then input SSID in the form of username and password. Only then can connect to the internet. Then the fire sensor works to detect the presence of fire, here the sensitivity level of the fire sensor is set to 70 If the fire sensor does not detect anything or value above 70 then the fire sensor will continue to work. But if the fire sensor value below 70 then GPS Ublox Neo-7m will work to display data latitude and longitude then the data will be sent to thingspeak.com.

4.1.4 SMS server Gammu

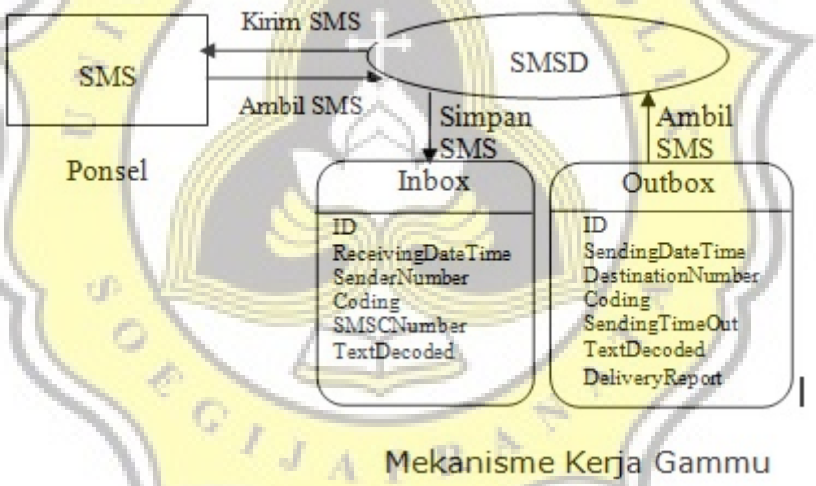

Illustration 4.8: Mekanisme kerja Gammu

(Source: merta12.files.wordpress.com)

SMS Server of gammu is used to send sms to mobile phone in order to send a message in the form of distance to the location of the house that fires. From here, php insert data in the form of mobile phone number of the recipient (destination number), the contents of sms (text decoded), and creatorId go to outbox which then will be channeled to SMSD. If successful, it will be moved to the table sentitem. If not due to operator constraints then still inside the outbox.

Here is the flowchart of the sms gateway gammu:

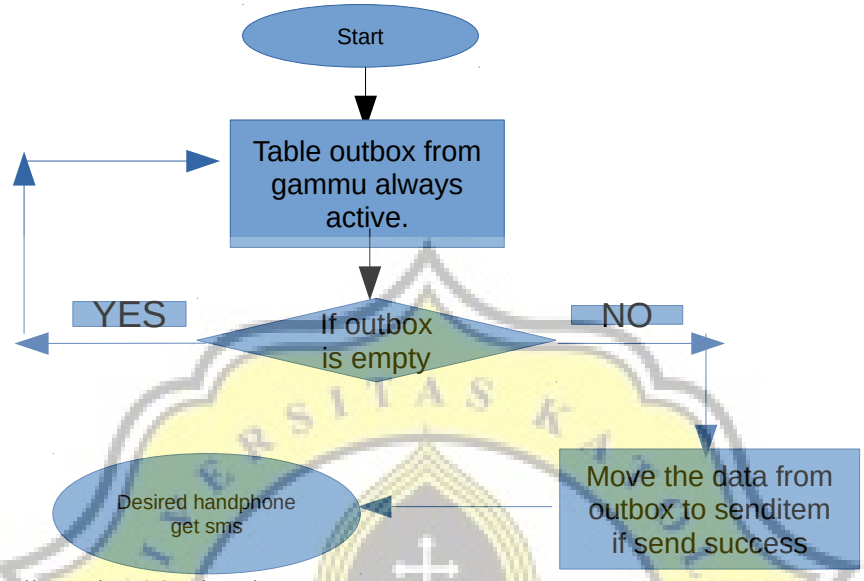

Illustration 4.9: Flowchart sms gateway Gammu

Here the outbox on the gammu database is always active, the outbox only requires TextDecoded, DestinationNumber, and CreatorID to send a SMS. If the outbox is filled with data, then the data on the outbox will be moved to the sentitem mark that the sms has been successfully sent. If not successfully sent, the data will remain in the outbox. Constraints here are usually due to pulses run out, config gammu-smsdrc wrong, and constraints on the sim operator.

4.1.5 The nearest fire station search section.

Using the method of the formula because Haversine can calculate the distance between one coordinate with other coordinates in a straight line, and ignore hills or valleys that exist on the surface. The input of this method is the latitude and longitude (coordinate) of the location to be calculated and the output of the distance value between the two locations. Using PHP to do the calculation then the result will be stored on a variable.

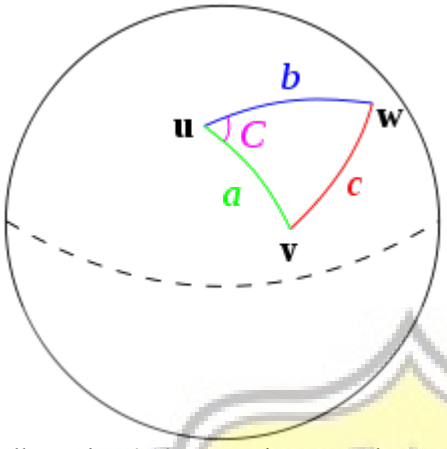

Illustration 4.10: Haversine Formula

Example Calculations of Haversine Formula

Setting up the point of location of latitude and longitude at two locations to calculate the distance

Location 1: lon1 = 119.800801, lat1 = -0.790175

Location 2: lon2 = 119.8428, lat2 = -0.8989

Then change the point of latitude and longitude into radian units

radian lat1 = -0.790175 * 0.0174532925 radian = -0.013791155

radians lon1 = 119.800801 * 0.0174532925 radian = 2.090918422

radian lat2 = -0.8989 * 0.0174532925 radian = -0.01569

radians lon2 = 119.8428 * 0.0174532925 radian = 2.091651

Then calculate the delta latitude and delta longitude

delta latitude = lat1 - lat2 = -0.0018976

delta longitude = lon1 - lon2 = 0.00073302

Then calculate the angle of delta latitude, delta longitude

angle = 2 * salted (sqrt (pow (sin (delta latitude / 2), 2) +

cos (radian lat1) * cos (radian lat2) * pow (sin ($ delta longitude / 2), 2)))

= 0.0020342

Then calculate the distance using earth radius = 6371000

distance = angle * earthRadius

= 12.960

Here is a flowchart of Haversine calculations on PHP:

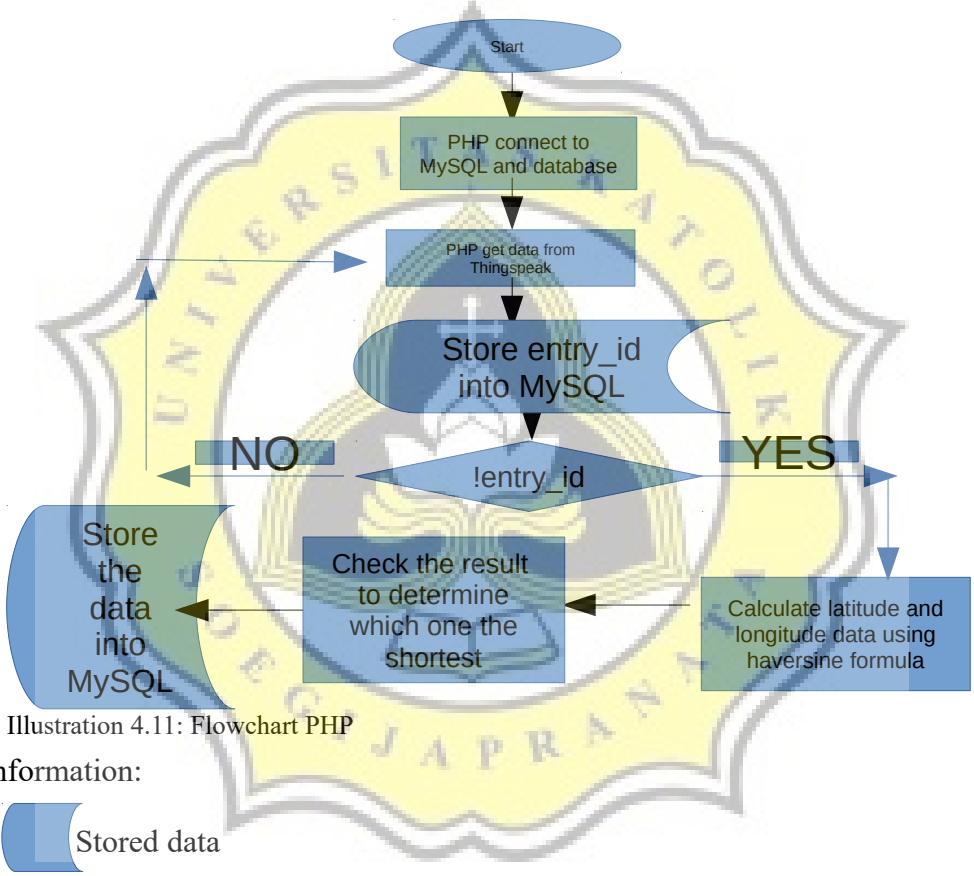

Illustration 4.11: Flowchart PHP

Information:

Stored data

First to connect MySQL database in PHP so that the occurrence of data storage on the database that is connected. Then PHP do get data from thingspeak to get entry_id, latitude, and longitude. Here storing the entry_id on the MySQL database, and if the entry_id is the same as the previous entry_id it will perform the data retrieval once again. If the entry_id is not the same as the previous entry_id it will return NULL then it will do the calculation by doing the formulas of Haversine, checking the closest distance, and stored in the MySQL database.

4.2 Testing

1. Performance Test

   To test performance, there are five to test which are:

   A) Fire Sensor KY-026, test the distance fire sensor can detect fire and how much ideal threshold for fire sensor to detect fire. Will be tested five times at different length: 15 cm, 30 cm, 45 cm, 60 cm, and 75 cm using lighter and candle.

   B) GPS Ublox Neo-7M, test the accurate gps can detect the current location. Will be tested five times at different location.

   C) ESP8266, test how fast it send data to thingspeak. Will be tested five times using wifi Indihome.

   D) Wavecom SMS gateway gammu, takes how many seconds it can send the warning sms to handphone. Will be tested five times using different sim provider.

   E) PHP calculate using Haversine Formula, how accurate the calculation from two different location. Will be tested five times from home location to different location.

   F) Fire Sensor KY-026 and PHP, how much time needed from detect fire to send sms.

2. Functional Test

   To test four module, which need to be considered is as follows:

   A) Red led on fire sensor KY-026, whether stays lit when close to the fire.

   B) Blue led on ESP8266, whether blinking when started up and connect to the internet.

   C) Yellow led on GPS Ublox Neo-7M, whether blinking when the satelite get the signal.

D) Red led on Wavecom, whether blinking when connected to the computer.