Final Project Assignment

CS-650: Internet of Things

Project Name: Smoke Detection System

Group No: 5

Group Members: Govind Yatnalkar, Prashant Kundeshwar & Sharon Joy

Date: 9th December 2018

Introduction

Smoke Detection System is a system designed for detection of smoke, liquefied petroleum gas (LPG) and carbon dioxide (CO2) using MQ2 gas sensor with Raspberry Pi and sending notifications to the users on events of detection. It is an IOT based system which also utilizes cloud resources for performance enhancements and flexibility.

Following is a list of the hardware used in our system:

Hardware and Software Used:

1. Raspberry Pi 3b+ Model
2. Smoke Sensor- MQ2 (Provides values for **CO, LPG & SMOKE**)
3. IC MCP3800 - 8 Channel Analog to Digital Converter for fetching digital values from Pi.
4. Logical Level Converter - Supplying appropriate voltages and avoiding damage to GPIO Pins of Raspberry Pi
5. LED Light which can be replaced by an auto-controlled cooling system.
6. Bread Board & Jumper Wires.

The Project possesses a PHP Web App and MySQL database forming the website for user interface. The website can be accessed through the following link:

https://cloudui-223803.appspot.com/

The main motive of the system is to notify users if traces of LPG/SMOKE are found in the air of user's home environment. If these traces are identified, an email notification is sent to user citing traces of smoke, lpg or co found.

The Web App is built using the following languages: PHP, HTML & JavaScript. For Backend or Database Purposes, MySQL is utilized. We are using Google Cloud Platform for webapp and database hosting. Despite access from any device via a shell, Google Cloud also provides features like scalability, availability, uniform access and consistent performance.

Background: Why do we need Smoke Detection System even though we have smoke detectors installed at our houses?

Smoke Detectors fire their alarms when smoke is detected. An action can be further taken after the alarm goes off like notifying others about the smoke detector alarm. But, it does not tackle the situations in which we are in our workplaces and smoke or other by-products of fire are detected in our house. These situations can be dangerous as no actions are taken leading to serious accidents. The only missing facet is therefore, notification, when we are away from the house.

Hence, our system brings this feature of sending immediate notifications to the users. Also, besides smoke, it also detects LPG and CO.

If these events are larger in number, then in such cases, it can be stated there is a gas leak or a source can be identified as the values are plotted in graphs (visual representations) or shown in the history with timestamps.

For better cause or for making human life easier and avoiding human accidents, Smoke Detection System is an easy operable and replaceable system to current smoke detectors.

Our system can be included in SMART HOME SYSTEMS which are built using IOT & CLOUD technologies.

Requirements

Requirements define the basis of implementation or the needs of users to be fulfilled in the project. They are categorized as follows: Functional & Non-Functional Requirements.

Functional Requirements

Functional requirements are the conditions proposed by the users.

1. Besides Detection, Send Notification

In this functional requirement, email is used to fulfil this requirement. We have used SMTP to send an email to the concerned person whose house is being infiltrated by smoke. It is an important part of the system as it alerts the person.

2. A User Interface for Users to Actually View Real Time Sensor Data

To satiate this requirement, we are providing a website which registers users and gives them access to sensor related data, the amounts of LPG, CO2 and Smoke. It also visually shows how these different types of gases affect the home environment and what levels are dangerous to the human body.

Non-functional Requirements:

Non-functional requirements define the requirements which may not be proposed by users but are important from a system's or project's perspective.

1. Cloud for Performance, Availability, Scalability & Uniform Access

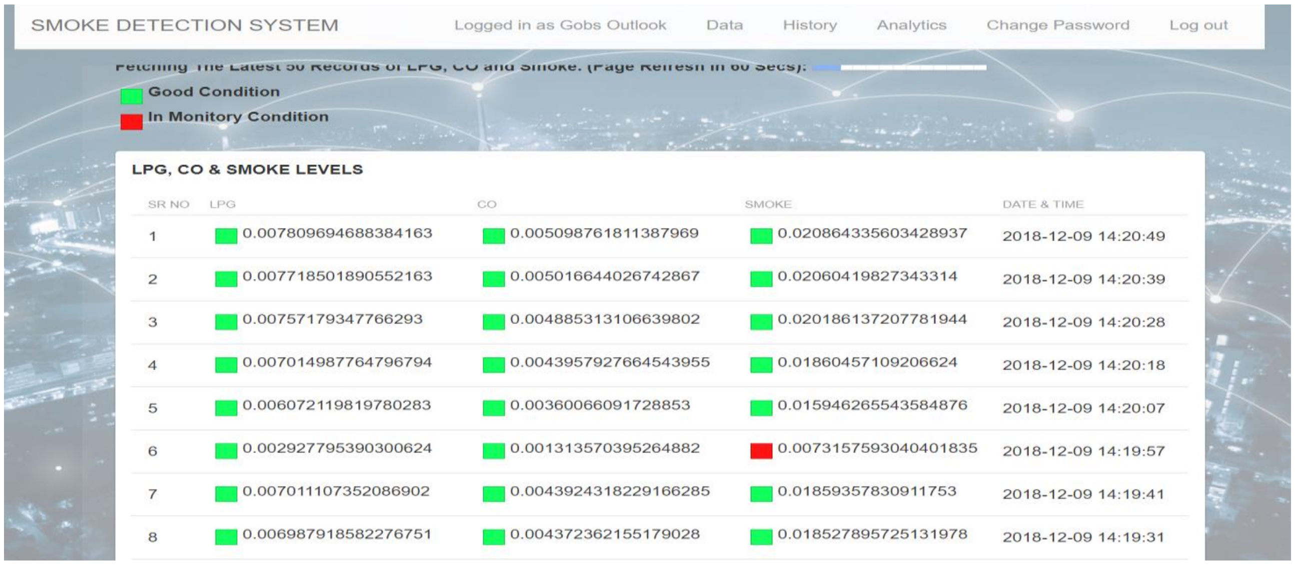
Using Google Cloud, we have a backup, or the data is replicated at various places. This is done for both, the PHP web-application and the database instances. The element of fault tolerance is included along with auto-scaling features and load optimization. Therefore, instances remain safe in case of failures, allowing multiple access from different platforms. As it is a web-application it has a uniform access across all the platforms.

2. LED now, COOLING SYSTEM later.

Glowing of LED indicates smoke, co or lpg is detected. In future, this can be replaced by a cooling system or sprinklers activators for any kind of detections. This allows users to eliminate the overhead of taking any actions as the system will take care of such issues.

3. Showing Graphs & Historical Data

A user can view the latest fetched data from the database. But, to provide a better idea if detection events are high in number or to understand if there is a gas leak, users are provided with a history page which shows historical sensor data. Also, users can view overall working of sensors through a graphical representation of these sensor values.



Fetching The Latest 50 Records of LPG, CO and Smoke. (Page Refresh in 60 Secs):

🟩 Good Condition

🟥 In Monitory Condition

LPG, CO & SMOKE LEVELS

SR NO	LPG	CO	SMOKE	DATE & TIME
1	🟩 0.007809694688384163	🟩 0.005098761811387969	🟩 0.020864335603428937	2018-12-09 14:20:49
2	🟩 0.007718501890552163	🟩 0.005016644026742867	🟩 0.02060419827343314	2018-12-09 14:20:39
3	🟩 0.00757179347766293	🟩 0.004885313106639802	🟩 0.020186137207781944	2018-12-09 14:20:28
4	🟩 0.007014987764796794	🟩 0.0043957927664543955	🟩 0.01860457109206624	2018-12-09 14:20:18
5	🟩 0.006072119819780283	🟩 0.00360066091728853	🟩 0.015946265543584876	2018-12-09 14:20:07
6	🟩 0.002927795390300624	🟩 0.001313570395264882	🟥 0.0073157593040401835	2018-12-09 14:19:57
7	🟩 0.007011107352086902	🟩 0.0043924318229166285	🟩 0.01859357830911753	2018-12-09 14:19:41
8	🟩 0.006987918582276751	🟩 0.004372362155179028	🟩 0.018527895725131978	2018-12-09 14:19:31

Fig 2.0 History Page in WebApp

System Architecture

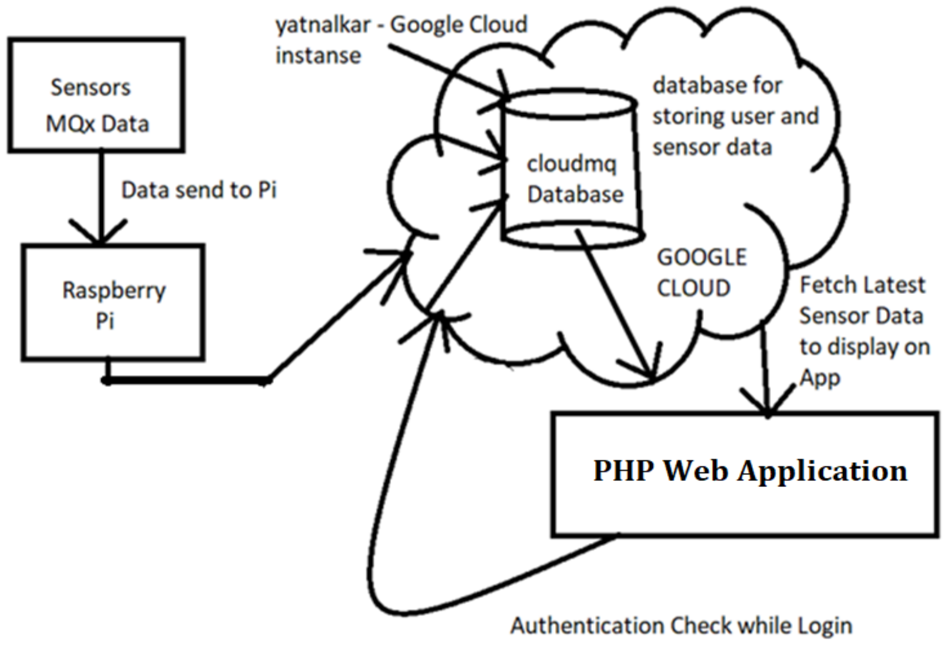

Sensors
MQx Data
Data send to Pi
Raspberry Pi
yatnalkar - Google Cloud instanse
cloudmq Database
database for storing user and sensor data
GOOGLE CLOUD
Fetch Latest Sensor Data to display on App
PHP Web Application
Authentication Check while Login

Fig 1.0 System Architecture

The architecture reflects Raspberry Pi getting values from MQ2 sensor and providing outputs on the GPIO pins. This data is then sent to the Google cloud database which is then fetched and shown in the PHP web application.

Hardware Architecture:

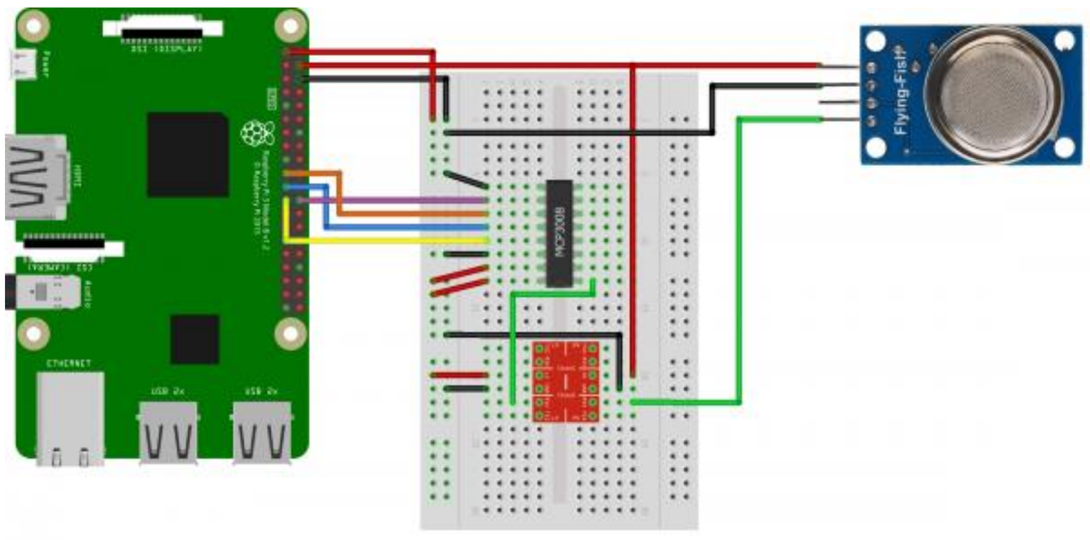


Fig 1.1 Hardware Architecture

The hardware architecture shows the GPIO pins connected to the MCP3800 IC which is the Analog to Digital Convertor. Pi is also connected to the Logical Level Convertor which bolsters in avoiding any damages to the GPIO pins by functioning as a voltage controller. From MQ2 sensor, the main data pin goes to the logical level convertor and which further goes to the Analog to Digital Convertor. Outputs are then established on the GPIO pins of Raspberry Pi and are fetched using a Python Script.

Challenges:

The challenges we faced while working on the project were:

- Finding out which was the best sensor to use in the MQX series for Smoke Detection.
- Mapping of hardware due to limited knowledge.
- Serial Port Communication disabled.
- Getting analogue output from GPIO pins.
- Data types of sensor values incompatible because of the length.
- There was a performance drag due to high traffic of data which was fetched by the pi and sent to the database.
- IP whitelisting (while connecting database to the cloud)

Solution to the Challenges

Following are the solutions implemented to the challenges stated above:

- Following a tutorial by Felix (https://tutorials-raspberrypi.com/configure-and-read-out-the-raspberry-pi-gas-sensor-mq-x/) lead us to select MQ2 sensor. The main motive was detection of smoke - a common gas detection observed in other sensors as well. In some countries, LPG gases are still used and as LPG is detected by MQ2, therefore, it was a good choice for selecting MQ2 sensor. Mapping, serial port communication and analogue output challenges were also solved following this tutorial. A Python script calibrated the sensor data and provided digital outputs at GPIO pins.
- Using Text as the data type in database, solved the issue of data compatibility.
- When the app was deployed on Cloud, a consistent performance was observed, and no delays were observed for data fetching and data rendering on web pages.
- Due to security reasons, IP whitelisting is done is Google Web App. Using quad route or adding an IP like 0.0.0.0/0 solved the issue as the device IP was accepted irrespective of network changes.

Results

1. Running IOT Project – When smoke is not detected.

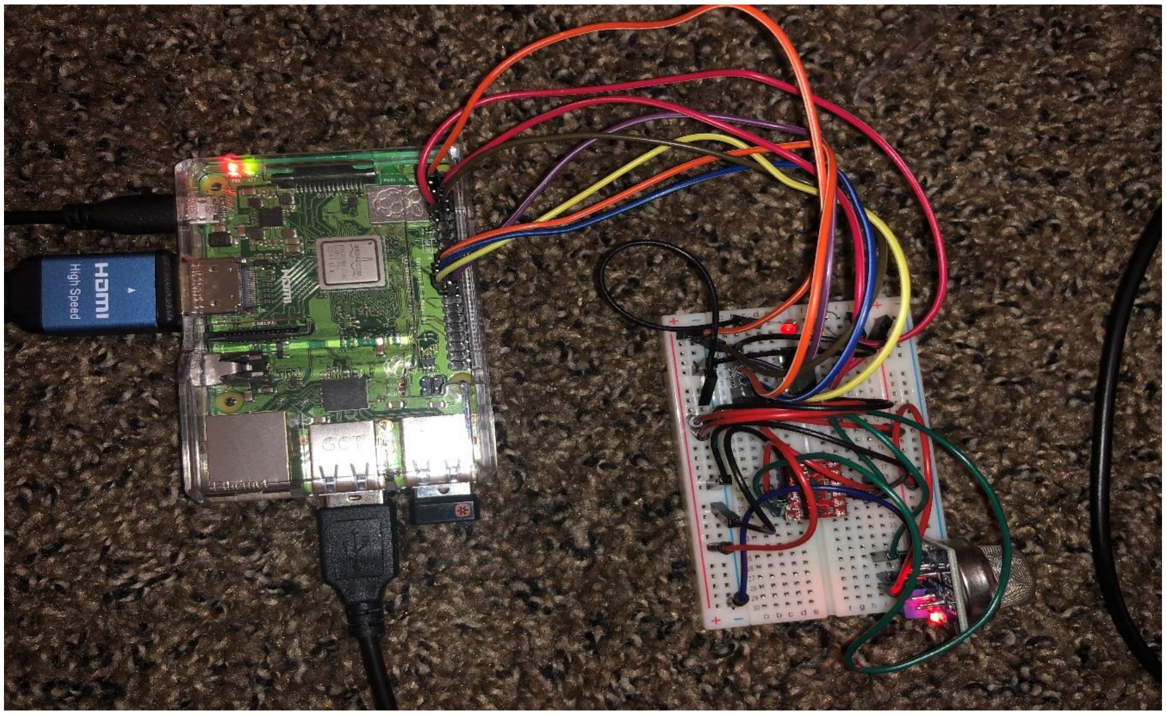


Fig 3.0 LPG/ CO/ Smoke Not Detected

2. Running IOT Project – When smoke is detected.

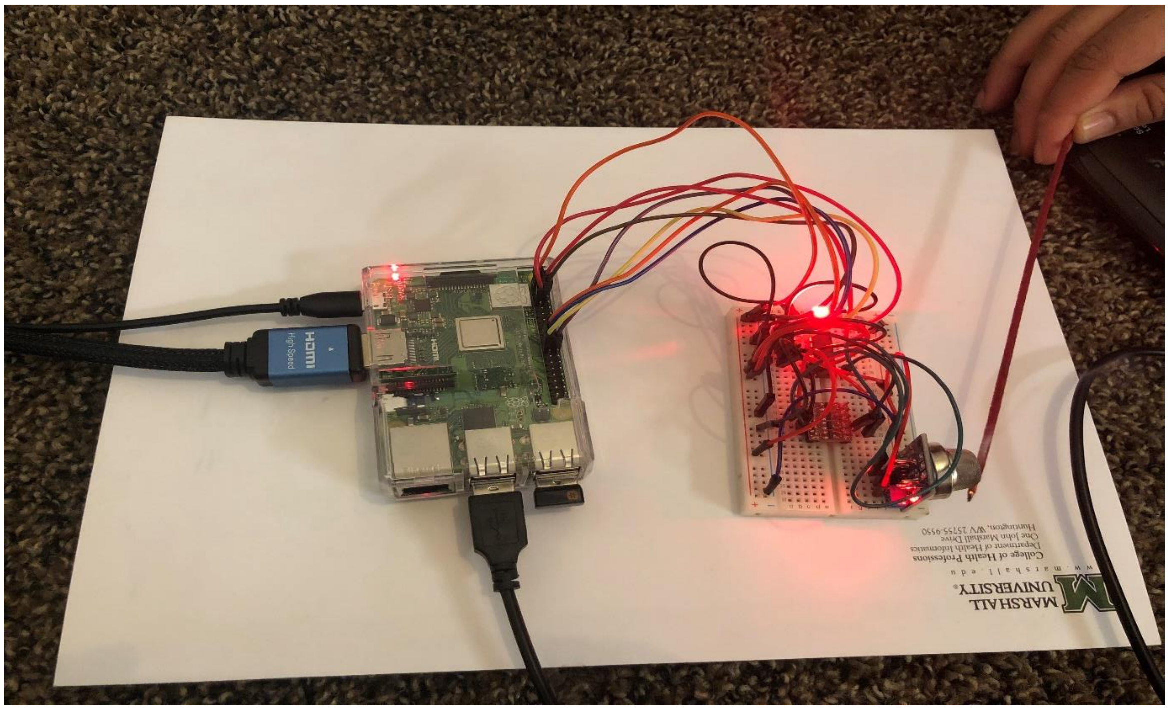
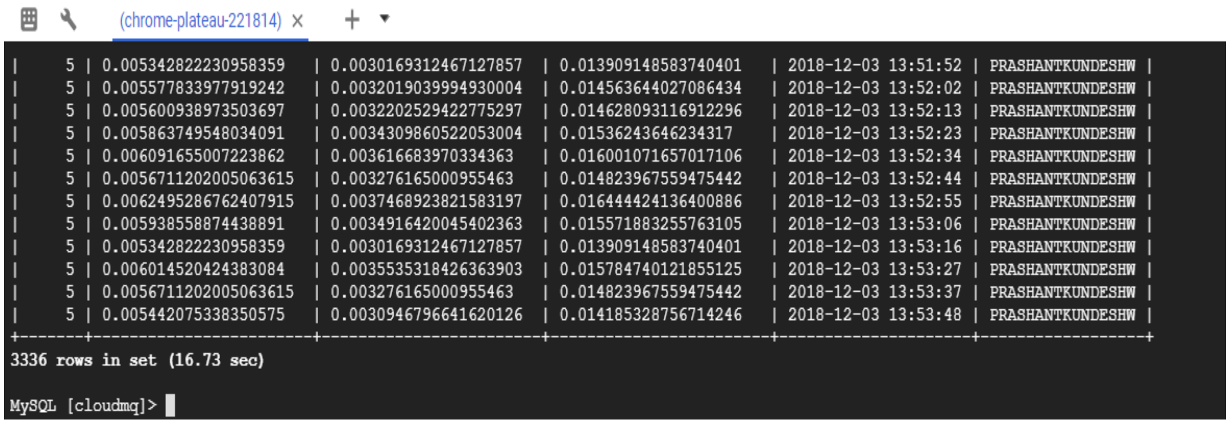


Fig 3.1 LPG/CO/ Smoke Detected

3. Database saving sensor values with time stamps and device ids.

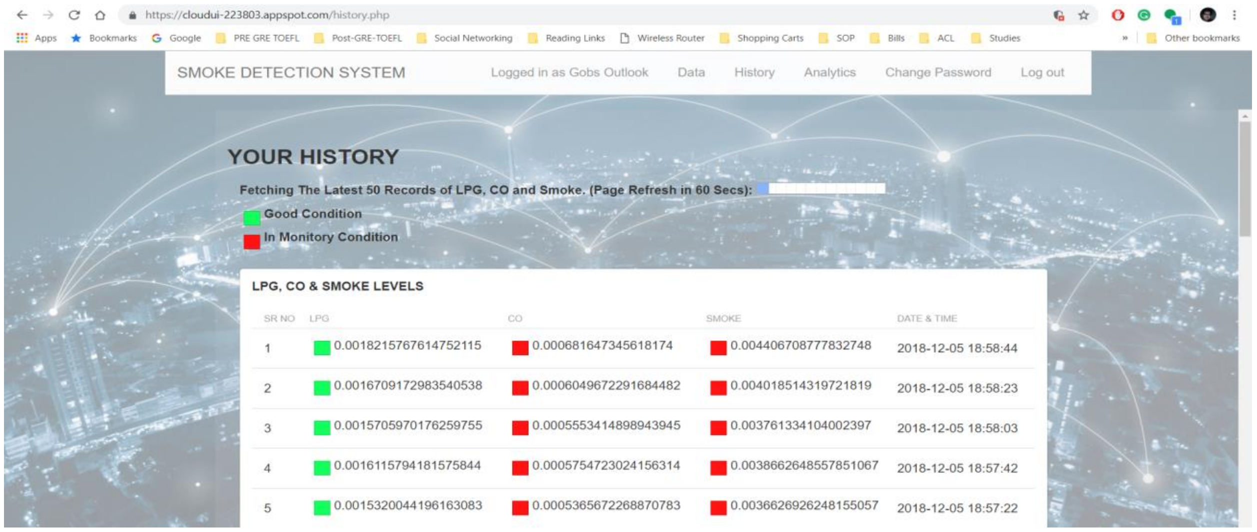


```
|      5 | 0.005342822230958359   | 0.003016931246712785 7 | 0.013909148583740401   | 2018-12-03 13:51:52 | PRASHANTKUNDESHW |
|      5 | 0.005577833977919242   | 0.0032019039994930004 | 0.014563644027086434   | 2018-12-03 13:52:02 | PRASHANTKUNDESHW |
|      5 | 0.005600938973503697   | 0.0032202529422775297 | 0.014628093116912296   | 2018-12-03 13:52:13 | PRASHANTKUNDESHW |
|      5 | 0.005863749548034091   | 0.0034309860522053004 | 0.01536243646234317    | 2018-12-03 13:52:23 | PRASHANTKUNDESHW |
|      5 | 0.006091655007223862   | 0.003616683970334363  | 0.016001071657017106   | 2018-12-03 13:52:34 | PRASHANTKUNDESHW |
|      5 | 0.0056711202005063615  | 0.003276165000955463  | 0.014823967559475442   | 2018-12-03 13:52:44 | PRASHANTKUNDESHW |
|      5 | 0.0062495286762407915  | 0.0037468923821583197 | 0.016444424136400886   | 2018-12-03 13:52:55 | PRASHANTKUNDESHW |
|      5 | 0.005938558874438891   | 0.0034916420045402363 | 0.015571883255763105   | 2018-12-03 13:53:06 | PRASHANTKUNDESHW |
|      5 | 0.005342822230958359   | 0.0030169312467127857 | 0.013909148583740401   | 2018-12-03 13:53:16 | PRASHANTKUNDESHW |
|      5 | 0.006014520424383084   | 0.0035535318426363903 | 0.015784740121855125   | 2018-12-03 13:53:27 | PRASHANTKUNDESHW |
|      5 | 0.0056711202005063615  | 0.003276165000955463  | 0.014823967559475442   | 2018-12-03 13:53:37 | PRASHANTKUNDESHW |
|      5 | 0.005442075338350575   | 0.0030946796641620126 | 0.014185328756714246   | 2018-12-03 13:53:48 | PRASHANTKUNDESHW |
+-------+------------------------+-----------------------+------------------------+---------------------+------------------+
3336 rows in set (16.73 sec)

MySQL [cloudmq]>
```

Fig 3.2 Database Saving Sensor Values

4. Hosted PHP Web Application in Google Cloud



YOUR HISTORY

Fetching The Latest 50 Records of LPG, CO and Smoke. (Page Refresh in 60 Secs):

🟩 Good Condition

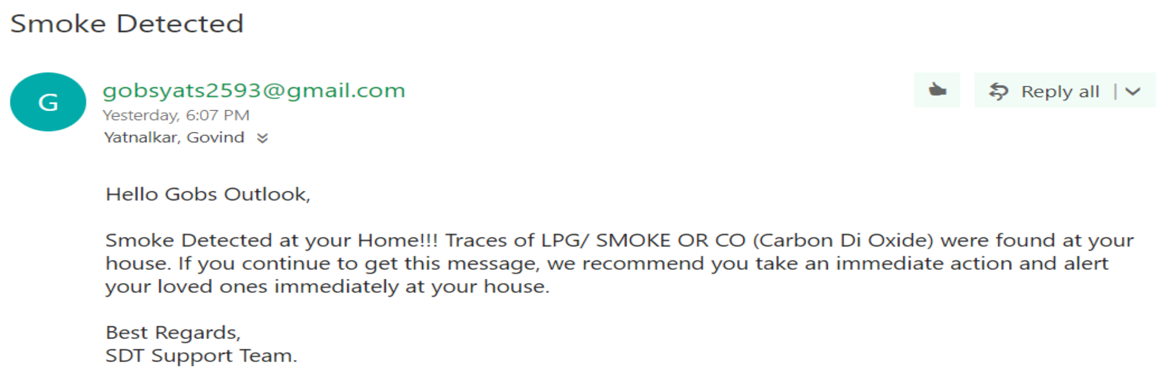
🟥 In Monitory Condition

LPG, CO & SMOKE LEVELS

SR NO	LPG	CO	SMOKE	DATE & TIME
1	🟩 0.0018215767614752115	🟥 0.000681647345618174	🟥 0.004406708777832748	2018-12-05 18:58:44
2	🟩 0.0016709172983540538	🟥 0.0006049672291684482	🟥 0.004018514319721819	2018-12-05 18:58:23
3	🟩 0.0015705970176259755	🟥 0.0005553414898943945	🟥 0.003761334104002397	2018-12-05 18:58:03
4	🟩 0.0016115794181575844	🟥 0.0005754723024156314	🟥 0.0038662648557851067	2018-12-05 18:57:42
5	🟩 0.0015320044196163083	🟥 0.0005365672268870783	🟥 0.0036626926248155057	2018-12-05 18:57:22

Fig 3.2 PHP Webapp for User Interface

5. User Notification Screenshot



Smoke Detected

gobsyats2593@gmail.com
Yesterday, 6:07 PM
Yatnalkar, Govind

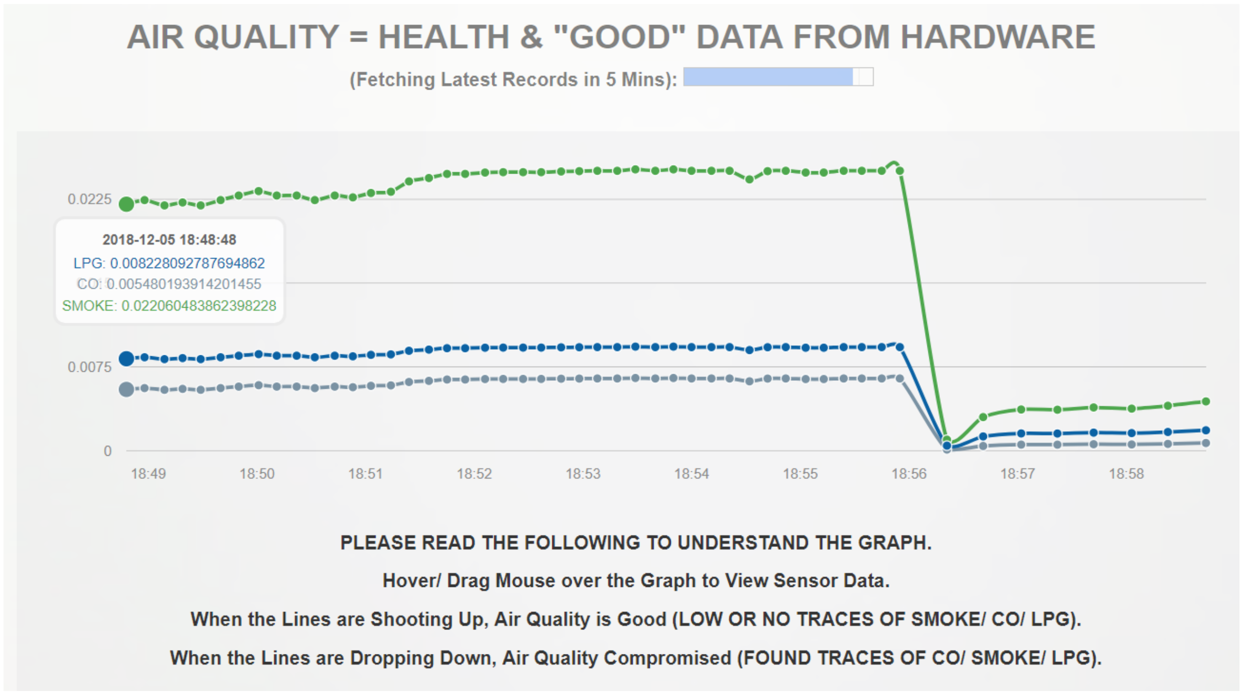
Hello Gobs Outlook,

Smoke Detected at your Home!!! Traces of LPG/ SMOKE OR CO (Carbon Di Oxide) were found at your house. If you continue to get this message, we recommend you take an immediate action and alert your loved ones immediately at your house.

Best Regards,
SDT Support Team.

Fig 3.3 User Notification

6. Sensor Data in form of Graphs


AIR QUALITY = HEALTH & "GOOD" DATA FROM HARDWARE

(Fetching Latest Records in 5 Mins):

2018-12-05 18:48:48
LPG: 0.008228092787694862
CO: 0.005480193914201455
SMOKE: 0.022060483862398228

PLEASE READ THE FOLLOWING TO UNDERSTAND THE GRAPH.

Hover/ Drag Mouse over the Graph to View Sensor Data.

When the Lines are Shooting Up, Air Quality is Good (LOW OR NO TRACES OF SMOKE/ CO/ LPG).

When the Lines are Dropping Down, Air Quality Compromised (FOUND TRACES OF CO/ SMOKE/ LPG).

Fig 3.4 Sensor Graphical Data

Conclusion:

In this project, we successfully built up a hardware for sensing entities like smoke, lpg and co. We then connected the Raspberry Pi device to Google Cloud. This led to saving sensor values along with their time stamps and device ids in a MySQL database.

Furthermore, hosting of PHP Web Application and database in Google Cloud was successfully completed. This resulted in uniform access and implemented Cloud features like scalability (Scale Up) and availability (Fault Tolerance). The hosted Web Application rendered the latest sensor data on HTML pages. User now have access to the latest updated records as well as to the historical data associated with their respective environment air quality.

At the end, users were notified with an email for smoke, co or lpg detection events which was the ultimate motive of the project. Hence users have an idea if there are smoke traces in the air and whether they should take any actions in advance before reaching to the point of accidents like a potential fire breakout.

References

- Ruchir Sharma. (2017, September 4). *How to Send Data to Google Cloud Database from RaspberryPi*. Retrieved from https://www.hackster.io/ruchir1674/how-to-send-data-to-google-cloud-database-from-raspberrypi-151ea2.
- Felix Stern, *Configure and read out the Raspberry Pi gas sensor (MQ-X)*. Retrieved from https://tutorials-raspberrypi.com/configure-and-read-out-the-raspberry-pi-gas-sensor-mq-x.
- Terry Ryan, Google Cloud Platform, (2017, February 24). *Deploying PHP Applications on App Engine*. Retrieved from https://www.youtube.com/watch?v=ioqVbyD-acA.
- Google Cloud Platform. (2018, November 8). *app.yaml Reference.* Retrieved from https://cloud.google.com/appengine/docs/standard/python/config/appref.
- Google Cloud Platform. (2018, May 9). *Google Cloud Platform 101 (Google I/O '18)*. Retrieved from: https://www.youtube.com/watch?v=trJaoEtBh6w&t=1694s
- Microcontrollers Lab. (2015). *ANALOG TO DIGITAL CONVERTER and how ADC works*. Retrieved from http://microcontrollerslab.com/analog-to-digital-adc-converter-working/