POLITEKNIK SULTAN SALAHUDDIN ABDUL AZIZ SHAH

AUTOMATIC GAS LEAKAGE WITH PHONE CALL

NAME	REGISTRATION NO
NORAIMIE ELYANA BINIT NORDDIN	08DEU20F2026

JABATAN KEJURUTERAAN ELEKTRIK

SESI 1 2022/2023

POLITEKNIK

SULTAN SALAHUDDIN ABDUL AZIZ SHAH

AUTOMATIC GAS LEAKAGE WITH PHONE CALL

NAME	REGISTRATION NO
NORAIMIE ELYANA BINTI NORDDIN	08DEU20F2026

This report submitted to the Electrical Engineering Department in fulfillment of the requirement for a Diploma in Electrical Engineering

JABATAN KEJURUTERAAN ELEKTRIK

SESI 1 2022/2023

CONFIRMATION OF THE PROJECT

The project report titled "Automatic Gas Leakage With Phone Call" has been submitted, reviewed and verified as a fulfills the conditions and requirements of the Project Writing as stipulated

Checked by:

Supervisor's name : PN IRMA BAIZURI BINTI MOHD AKHIR

Supervisor's signature:

Date :

Verified by:

Project Coordinator name :

Signature of Coordinator :

Date :

"I acknowledge this work is my own work except the excerpts I have already explained to our source"

1. Signature :

Name : **NORAIMIE ELYANA BINTI NORDDIN**

Registration Number : **08DEU20F2026**

Date :

DECLARATION OF ORIGINALITY AND OWNERSHIP

TITLE : AUTOMATIC GAS LEAKAGE WITH PHONE CALL

SESSION: SESI 1 2022/2023

1. I, **1. NORAIMIE ELYANA BINTI NORDDIN (08DEU20F2026)**

   is a final year student of **Diploma in Electrical Engineering, Department of Electrical, Politeknik Sultan Salahuddin Abdul Aziz Shah**, which is located at **Persiaran Usahawan,40140 Shah Alam Selangor Darul Ehsan**. (Hereinafter referred to as 'the Polytechnic').

2. I acknowledge that 'The Project above' and the intellectual property therein is the result of our original creation /creations without taking or impersonating any intellectual property from the other parties.

3. I agree to release the 'Project' intellectual property to 'The Polytechnics' to meet the requirements for awarding the **Diploma in Electrical Engineering** to me.

Made and in truth that is recognized by;

a) **NORAIMIE ELYANA BINTI NORDDIN**) .
(Identification card No: - 020818010538)) .....
) **NORAIMIE ELYANA BINTI NORDDIN**

In front of me, **PN IRMA BAIZURI BINTI MOHD AKHIR** (Click here to enter text.) As a project supervisor, on the date:

) .....
) **PN IRMA BAIZURI BINTI MOHD AKHIR**

ACKNOWLEDGEMENTS

I have taken efforts in this Project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. I am highly indebted to Pn Irma Baizuri Binti Mohd Akhir for their guidance and constant supervision as well as for providing necessary information regarding the Project & also for their support in completing the Project.

I would like to express my gratitude towards my parents & member of Politeknik Sultan Salahuddin Abdul Aziz Shah for their kind co-operation and encouragement which help me in completion of this Project. I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the Project and people who have willingly helped me out with their abilities.

ABSTRACT

LPG is widely used for cooking. Hence, it can leak both as a liquid or as a gas if it is not handled cautiously. If the gas leakage is not detected in the early stages, then it can lead to a very big disaster. The main objective of the project to build a gas LPG leakage detector using an LPG gas sensor and micro-controller. It developed a security system by providing an early warning system to give a sign if there is a smell of gas around home. If this system has been the existence of leakage and smell of LPG gas, then the system will give an early warning of the system such as the system will call the number right after the MQ-6 detect the gas leak. It can work with the GSM SIM900C Module combine with ESP32 then can make a call . It becomes essential to protect gas leakage from damage and accident.

ABSTRAK

LPG digunakan secara meluas untuk memasak. Oleh itu, ia boleh bocor sebagai cecair atau gas jika ia tidak dikendalikan dengan berhati-hati. Sekiranya kebocoran gas tidak dikesan pada peringkat awal, maka ia boleh membawa kepada bencana yang sangat besar. Objektif utama projek membina pengesan kebocoran gas LPG menggunakan sensor gas LPG dan pengawal mikro. Ia membangunkan sistem keselamatan dengan menyediakan sistem amaran awal untuk memberi tanda jika terdapat bau gas di sekitar rumah. Sekiranya sistem ini telah wujud kebocoran dan bau gas LPG, maka sistem akan memberi amaran awal kepada sistem tersebut seperti sistem akan menghubungi nombor tersebut sejurus selepas MQ-6 mengesan kebocoran gas tersebut. Ia boleh berfungsi dengan Modul GSM SIM900C digabungkan dengan ESP32 kemudian boleh membuat panggilan. Ia menjadi penting untuk melindungi kebocoran gas daripada kerosakan dan kemalangan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURE

CHAPTER 1

1 INTRODUCTION

1.1 Introduction

Liquified Petroleum Gas (LPG) is nowadays commonly used in households; Hence, it can leak both as a liquid or as a gas if it is not handled cautiously. Accidents and disasters related to LPG gas leakage are not unheard of. These leakage accidents can cause huge fire and explosion. This project show implementation and design gas leakage detection system. The main objective of the project to build a gas LPG leakage detector using an LPG gas sensor and micro-controller. It developed a security system by providing an early warning system to give a sign if there is a smell of gas around home. If this system has been the existence of leakage and smell of LPG gas, then the system will give an early warning of the system such as the system will call the particular number right after the MQ-6 detect the gas leak. It can works with the GSM SIM900C Module combine with ESP32 then can make a call . When the ppm reach 60 ppm .It becomes essential to protect gas leakage from damage and accident.

1.2 Background Research

I purpose this project because there are over 300 cases cause of gas . Its affect the health and environment beside danger the lives. This project is using IOT Smart Gas Monitor application to show the value of gas . Besides that , user also can control the valve and fan . This apps also were programmed to make a call to user when the value of gas reach the maximum digit which is 60 ppm .In this project circuit, the circuit need to connect the power which is 12 V and connect to Wi-Fi named "IOTGasMonitor" . The input is MQ-6 gas sensor .The data collect by the processor WIFI Module (Node MCU ESP32) . The output of this project are buzzer , warning LED , GSM SIM900A module to make phone call , relay to active the 12V DC fan and direct to IOT server (User application which is IOT Smart Gas monitor)

1.3 Problem Statement

Gas leakage can cause fire that will bring big disaster. It also could contribute serious injury or death. Users not alert and aware that the gas is leaking. When nobody's at home, what is point the function of alarm. The current system is the users get the warning message when the gas level reaches the maximum value. Even though the users get the messages, they are still not making their phone as their priority when they're busy, but when the emergency call incoming, they will notice faster than messages. This project is applicant for kitchen in the house

1.4 Research Objectives

The main objective of this Project is this gas leakage detection system, which will help in detecting any gas leakage with the help of MQ6 gas sensor and send this data over the internet to the IoT module and that will in turn alert the user about this gas leakage. Hence, following this process, we can detect gas leakage in the early stages and prevent any future accidents. The finished device is connected to the IoT module over Wi-Fi.

More specifically the principal objective of this research are:

To design a system that can detect leaked gas by using MQ-6 sensor such as LPG gas.

To make a direct call to user alert that the gas leaked by using GSM 900C Module and programming language C in NODE MCU ESP32

1.5 Scope of Research

1. This Project is focusing on households.
2. The focus of this project is to blow out the gases that leak, turn off the valve gas and call the owner when there are not at home .
3. The main controller is using ESP32.
4. This project will be designed as a prototype and the estimated budget for the implementation of this project will not exceed RM500.

1.6 Project Significance

Based on the research that has been done , the existence of gas leakage detector just enough for safety at home from alarm , SMS alert and phone call alert . I invent this project to make multi-function in one time . About my project , besides make a phone call , they also can remove the gas at kitchen and turn off the valve gas . They will detect the gas and when reach 60 ppm , the system will automatically make phone call , turn on the fan and turn off the valve gas .

1.7 Chapter Summary

As conclusion of this chapter , we know that the danger of LPG gas even though it happens from small mistakes that we made . We also know the problem statement and solution for it . Last but not least , the existence project help me to improve my project to do more function with low budget .

CHAPTER 2

2 LITERATURE REVIEW

2.1 Introduction

- Despite the substantial losses of lives and livelihoods in fire mishaps, the fire safety awareness among Malaysians remains slow.
  As we can see there were so many injuries and lost properties worth billions of ringgits were damaged or destroyed in flames, according to the latest statistics from the Fire and Rescue Department of Malaysia (Bomba).

_202201120956420_208.-jbpm_statistik-kebakaran-mengikut-jenis-kebakaran-2020

TAHUN	JENIS KES KEBAKARAN	PLS	KED	PP	PRK	SEL	KL	NS	MEL	JOH	PHG	TRG	KEL	SBH	SWK	LAB	PUT
2020	Bangunan dan Isinya	26	385	309	337	1020	399	311	155	458	176	139	146	419	294	16	9
2020	Kenderaan	20	181	204	355	987	253	203	146	514	196	96	99	173	183	5	7
2020	Mesin	0	15	22	11	25	2	5	3	53	7	3	5	7	7	0	1
2020	Alat Perkakas	22	56	122	173	285	55	30	35	321	62	50	41	42	111	4	10
2020	Petrol	0	0	0	1	4	0	1	0	0	0	0	0	0	1	0	0
2020	Bahan Kimia	0	0	1	0	1	0	1	0	2	0	0	0	0	0	0	0
2020	Gas	10	28	64	43	103	18	10	34	41	34	25	40	29	21	0	4
2020	Kapal Terbang	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
2020	Helikopter	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2020	Kapal Laut	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
2020	Feri	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2020	Bot	0	1	0	6	2	0	0	2	0	5	3	0	2	2	1	0
2020	Kebun/Ladang	21	100	11	55	248	0	29	29	429	24	21	24	114	20	0	0
2020	Hutan	6	208	153	79	410	6	175	28	21	87	93	105	594	27	9	1
2020	Belukar/Lalang	523	1831	761	1179	1241	68	525	908	1780	731	821	526	995	503	204	4
2020	Sampah	15	146	356	309	1001	133	156	131	604	83	57	80	112	160	13	8
2020	Gerai	4	12	3	8	20	17	4	2	8	6	4	2	3	7	0	0
2020	Lain-lain	264	897	752	1040	2174	521	316	425	834	480	356	490	437	295	28	16

Table 1 The fire statistics by fire type from Jabatan Bomba Dan Penyelamat

2.2 LPG GAS

- Liquefied petroleum gas (LPG or LP gas) is a fuel gas which contains a flammable mixture of hydrocarbon gases, specifically propane, propylene, butylene, isobutane, and n-butane. LPG is used for cooking in many countries for economic reasons, for convenience or because it is the preferred fuel source.

Figure 1 LPG for cooking net weight 12kg

2.3 Maximum value for LPG gas leaking

- OSHA: The legal airborne permissible exposure limit (PEL) is 20 ppm not to be exceeded at any time, and 50 ppm as a maximum peak, not to be exceeded during any 10-minute work period.

2.4 Combustion and flammability

- Firstly, a gas leak from the cylinder or regulator gets mixed with air, forming a combustible mixture. To complete the fire triangle, we need a spark or a source of ignition. This spark ignites the combustible LPG-air mixture, and this leads to an explosion.
- The gas leaked by an LPG cylinder if inhaled can lead to suffocation, as well as cause difficulty in walking or speaking. Your nervous system can get

affected, while you can experience heart attack and rise in your blood pressure. Hence, it is important to be careful if you detect a LPG cylinder leak.

- What's the ignition temperature of LPG? The propane ignition temperature in air (ignition temperature of propane gas) is when it reaches a temperature between 470°C – 550°C (878°F – 1020°F). At this temperature, the propane will ignite without the need for a flame, spark, or other ignition sources.

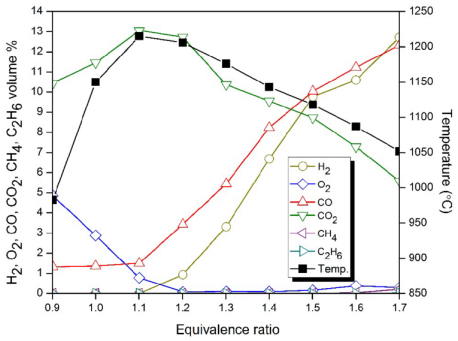


Figure 2 Combustion characterization of propane fuel in air at fuel/air equivalence ratios ranging from 0.9 to 1.7.

- LPG forms flammable mixtures with air in concentrations of between approximately 1.8% and 9.5%.

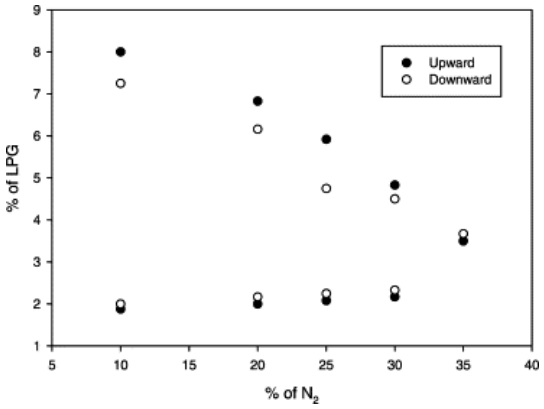


Figure 3 flammability limits of LPG/air mixtures

•The substances which have very low ignition temperature and can easily catch fire with a flame are called inflammable substances. Examples of inflammable substances are petrol, alcohol, Liquified Petroleum Gas (LPG)

2.5 MQ-6 Gas sensor

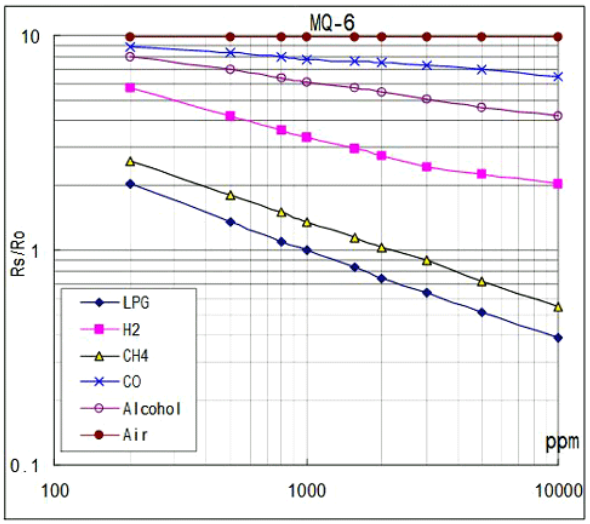


Fig.3 is shows the typical sensitivity characteristics of the MQ-6 for several gases.
in their: Temp: 20℃、
Humidity: 65%、
O_2 concentration 21%
RL=20kΩ
Ro: sensor resistance at 1000ppm of LPG in the clean air.
Rs:sensor resistance at various concentrations of gases.

Figure 4 typical sensitivity chateristics of MQ-6 for several gas

2.6 Microcontroller

ESP32 is a series of low-cost, low-power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules It is a successor to the ESP8266 microcontroller.

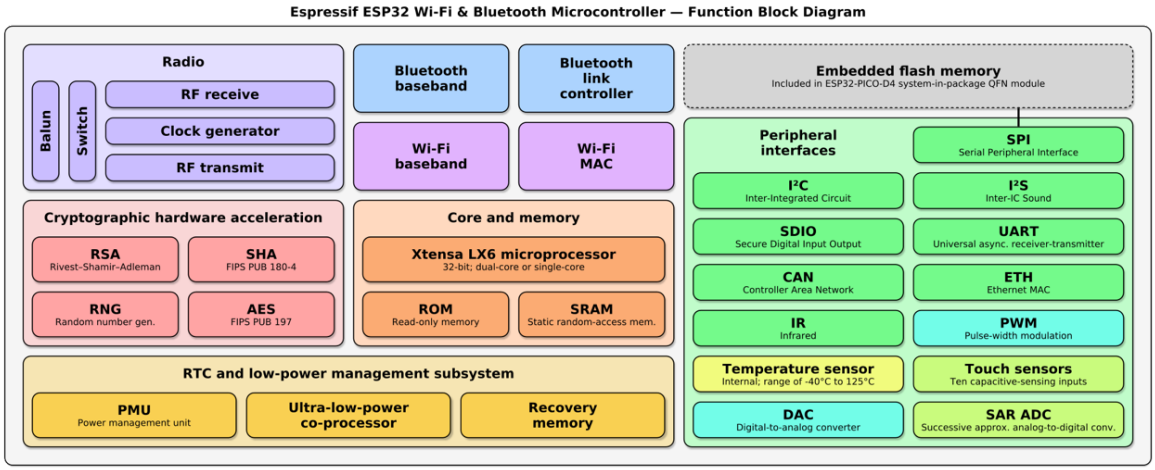


Figure 5 ESP 32 block diagram

2.6 Chapter Summary

This chapter focusing about literature review. In this chapter we have to listing a several journal about our project. Then, we have to find and investigate the method, objective , literature review and their project into the journal

CHAPTER 3

3 RESEARCH METHODOLOGY

3.1 Introduction

To realize this Project as a product that ready to use with safety characteristic, a very comprehensive plan is undertaking. A step by step procedure is done so that the Project can be completed in time. This includes ,design the mechanical part, circuit design testing and verification.

3.2 Project Design and Overview.

This project is using IOT Smart Gas Monitor application to show the value of gas . Besides that , user also can control the valve and fan . This apps also were programmed to make a call to user when the value of gas reaches the maximum digit which is 60 ppm .In this project circuit, the circuit need to connect the power which is 12 V and connect to WIFI named "IOTGasMonitor" . The input is MQ-6 gas sensor .The data collect by the processor WIFI Module (Node MCU ESP32) . The output of this project is buzzer , warning LED , GSM SIM900A module to make phone call , relay to active the 12V DC fan and direct to IOT server (User application which is IOT Smart Gas monitor)

3.3 Block Diagram of the Project

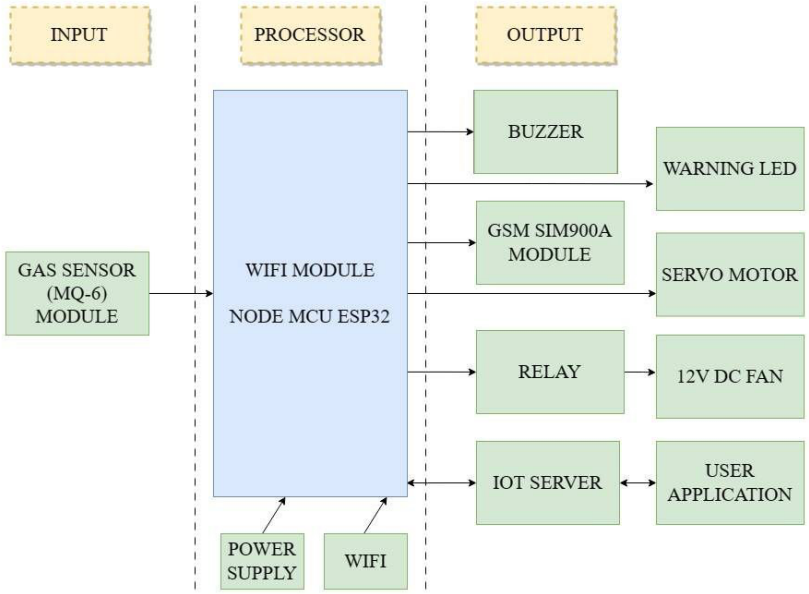


Figure 6 Project Block Diagram

3.4 Flowchart of the Project

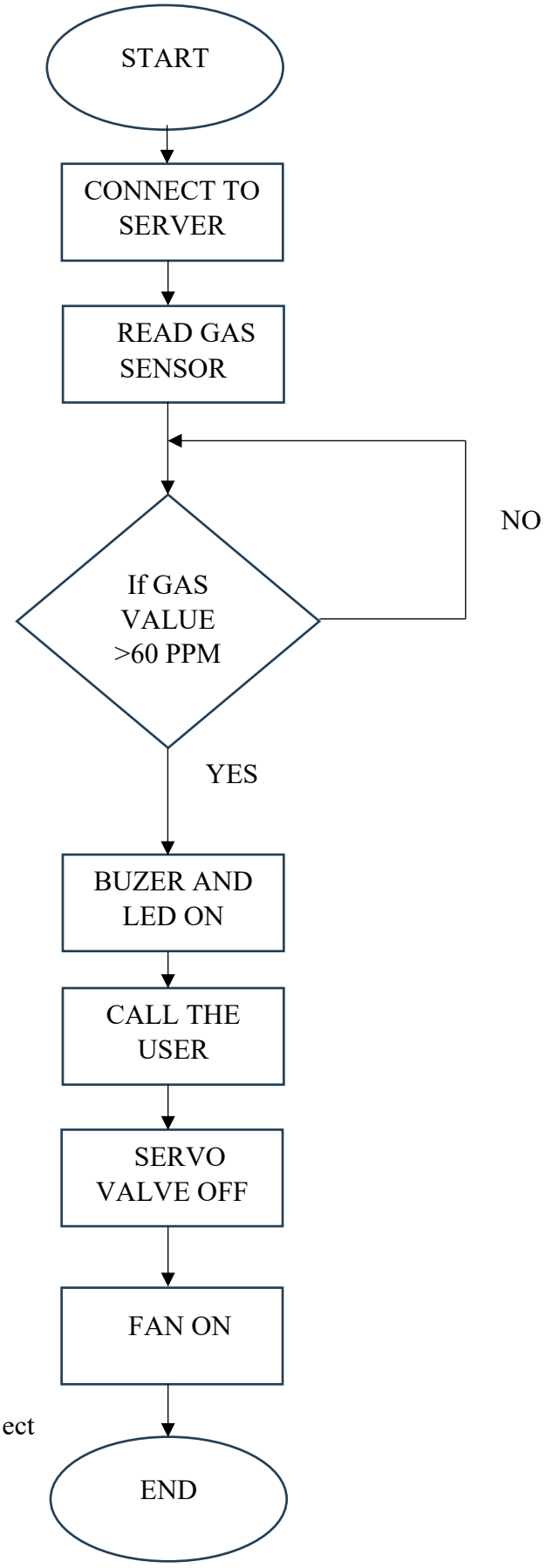


Figure 7 FlowChart project

3.5 Project Description

This project is using IOT Smart Gas Monitor application to show the value of gas . Besides that , user also can control the valve and fan . This apps also were programmed to make a call to user when the value of gas reach the maximum digit which is 60 ppm .In this project circuit, the circuit need to connect the power which is 12 V and connect to WIFI named "IOTGasMonitor" . The input is MQ-6 gas sensor .The data collect by the processor WIFI Module (Node MCU ESP32) . The output of this project are buzzer , warning LED , GSM SIM900A module to make phone call , relay to active the 12V DC fan and direct to IOT server (User application which is IOT Smart Gas monitor)

3.6 Project Hardware

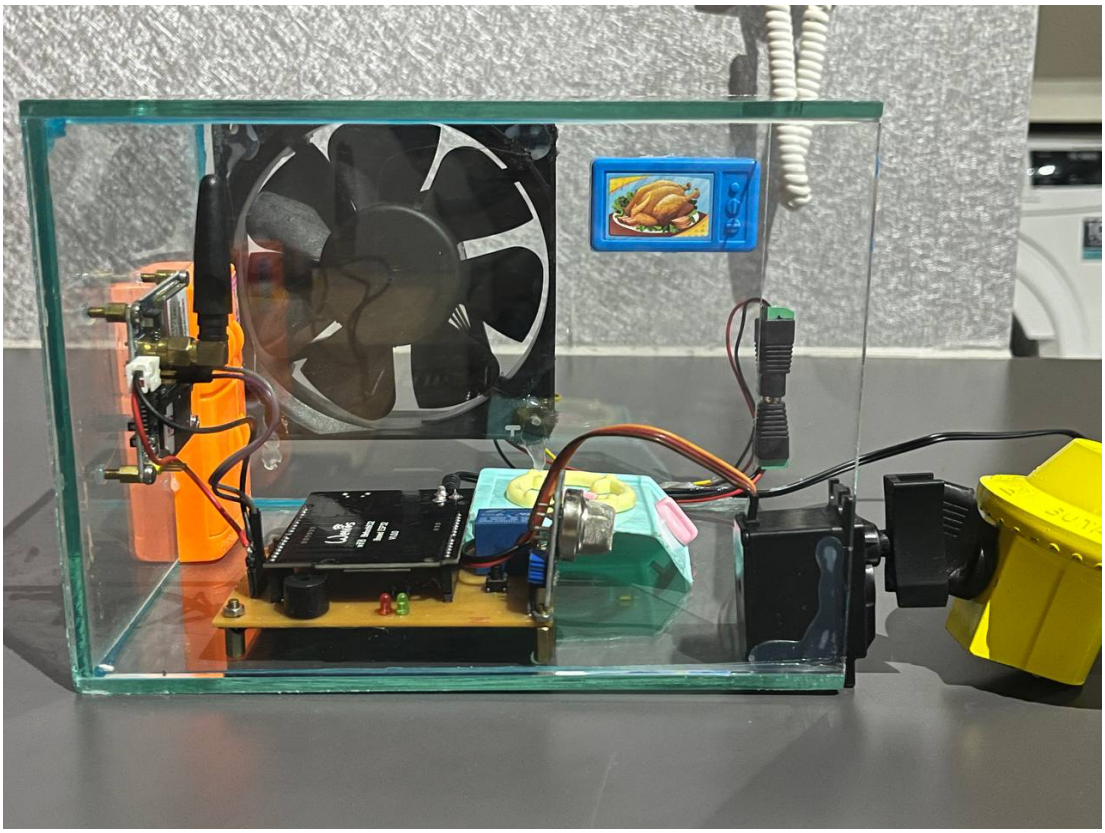


Figure 8 Project hardware

3.7 Schematic Circuit

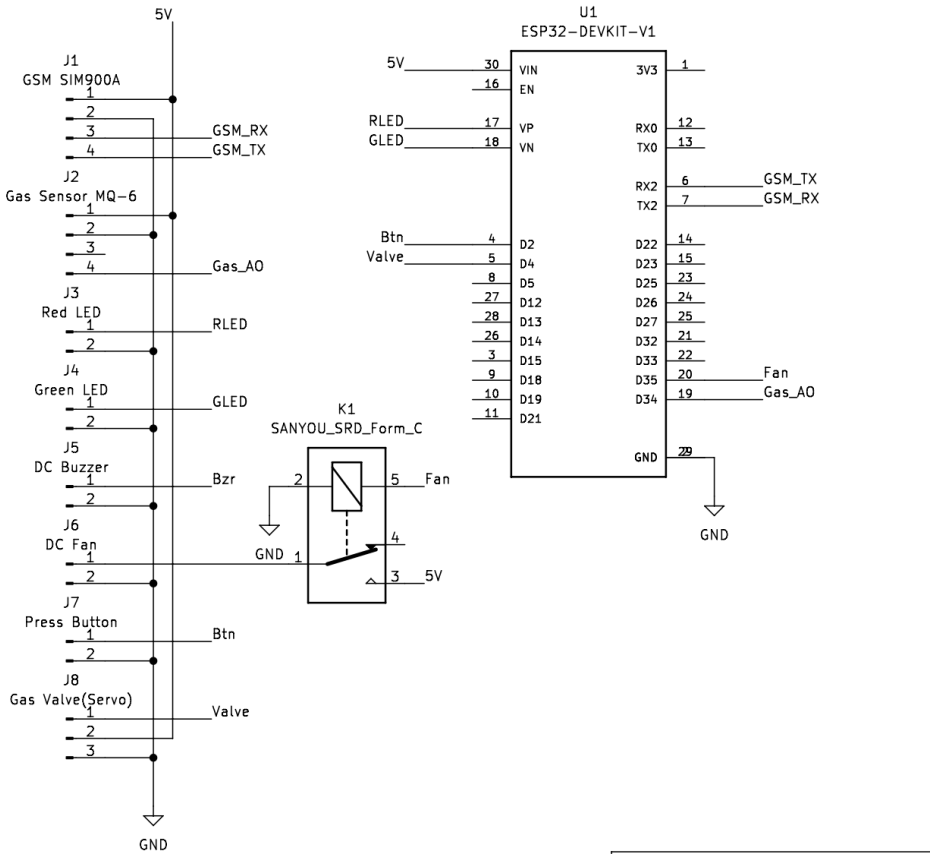


Figure 9 Project schematic circuit

3.8 Description of Main Component

3.8.1 ESP 32

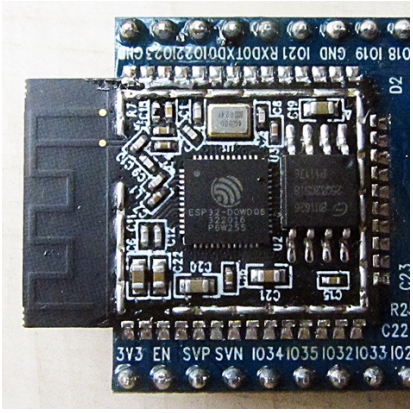


Figure 10 ESP 32

ESP32 is a series of low-cost, low-power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process.[2] It is a successor to the ESP8266 microcontroller.

3.8.2 MQ-6 Gas sensor.

Figure 11 MQ-6 Sensor

MQ6 Gas sensor is a Metal Oxide Semiconductor (MOS) type Gas Sensor mainly used to detect the LPG and Butane gas concentration in the air either at home or in industry. This sensor contains a sensing element, mainly aluminum-oxide based ceramic, coated with Tin dioxide, enclosed in a stainless-steel mesh.

The RS is the sense resistance during the presence of a particular gas whereas the RO is the sense resistance in clean air without any gas. The below logarithmic graph taken from the datasheet provides an overview of the gas concentration with the sense resistance of the MQ6 sensor. The MQ6 sensor is used to detect LPG gas concentration. Therefore, the MQ6 sensor will provide a particular resistance during the clean air condition where the LPG gas is unavailable. Also, the resistance will change whenever the LPG gas is detected by the MQ6 sensor.

3.8.3 GSM 900a Module

Figure 12 GSM 900a module

SIM900A is an ultra-compact and reliable wireless module. This is a complete GSM/GPRS module in a SMT type and designed with a very powerful single-chip processor integrating AMR926EJ-S core, allowing you to benefit from small dimensions and cost-effective solutions. Specification. Dual-Band 900/ 1800 MHz.

The GSM-900 GSM/GPRS module is a readily available GSM/GPRS module, which can provide the network connectivity to your project. It can do all the

work your mobile phone would do like making a call, receive a call, send a message, connect to the internet using GPRS.

3.8.4 Servo valve gas

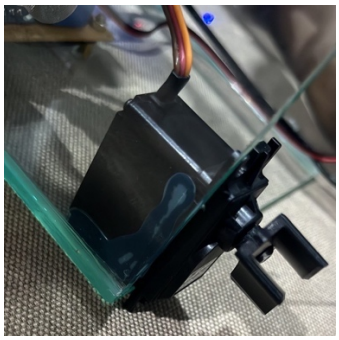


Figure 13 Servo valve gas

Servo valves and Servo-Proportional Valves are electrohydraulic, continuously acting valves that transform a changing analog or digital input signal into a stepless hydraulic output (flow or pressure).

A servo motor is an electromechanical device that produces torque and velocity based on the supplied current and voltage. A servo motor works as part of a closed loop system providing torque and velocity as commanded from a servo controller utilizing a feedback device to close the loop.

3.8.5 12V fan

Figure 14 Ventilator fan

12V DC Cooling Fan 2-inch 50mm, The direct current fans, or DC fans, are powered with a potential of fixed value such as the voltage of a battery. It features maintenance-free double ball bearings, long service life, sufficient heat dissipation air volume, and air pressure.

3.8.6 **Relay 3V**

Figure 15 Relay 3v

Relays are electrically operated switches that open and close the circuits by receiving electrical signals from outside sources.

3.9 Project Software

In this project , I am using Arduino programming in ESP 32 that can make connection with the apps that I use and the project .Firstly , build the coding for ESP 32 to make command for the fan and servo . Secondly ,build the coding for GSM 900a Module to make a connection and can make a phone call . For the application , I am using Android studio and the server from Firebase to make IoT Smart Gas monitor , besides make a direst call and automatically on the fan and off the servo , it also can control the fan and the servo . We can control whether we want to on or off the fan and the servo.

3.10 Chapter Summary

In this chapter we are doing research methodology. In this chapter we have to find the way or the method that we use to make a project become real. In this chapter, we make a listing and description about the components that we use to make a project. We also have a flowchart and circuit project.

CHAPTER 4

4 RESULTS AND DISCUSSION

4.1 Introduction

This Automatic Gas Leakage with phone call is designed to detect any gas leaks for households especially in the kitchen . The primary propose of the system is to prevent fire and disaster happened at early stage .

Here are some key aspects of a typical gas leakage system :

1.MQ-6 sensors : This very main component that uses in this system to detect LPG leakage . This sensor was connected at the board and was programmed when reading reach maximum value.

2.ESP 32 Module : This microcontroller as a brain of this system , that have WiFi and Bluetooth to give instruction in this system . This microcontroller also was programmed and have connection between board and server on smartphone to control the system.

3.GSM 900a Module : This module that can make the system make a call when the sensor triggered and reached the maximum value . This module with the antenna also were programmed to make a call to a particular number.

4.Servo valve gas : This component was connected at the board and receive the command when sensor triggered and this servo valve gas automatically off .

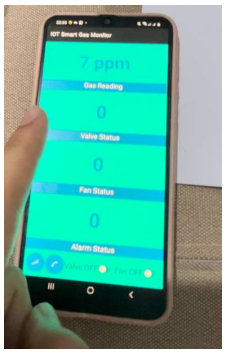
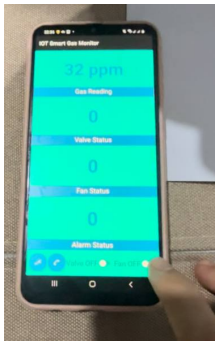
5.Ventilator fan 12V : This fan automatically on when the gas leak above 60 ppm to remove all the gas in that area

4.2 Results and Analysis

Automatic Gas Leakage with phone call are primally designed for fire protection at kitchen for early stages . While the system give alert to the users , this system also reduce and remove the gas leaks at kitchen .

As we can see below , this reading at application shown that the gas reading was red color which is the gas that were leaking is in danger reading , above 60 ppm . The valve status is 1 shown that the valve is automatically off meanwhile the fan status is 1 show that the ventilator fan is automatically on until the gas disappear in that area . Alarm status is 1 shown that the buzzer were beep to give alert .

Table 2 shows the value of PPM gas taken after the gas has been released at a certain time, with the result slightly different between short time and long time .

Time for gas release / sec	Value of ppm	Results	Remarks
1.0 sec	7 ppm		Valve : off Fan : off Alarm : off
2.0 sec	32 ppm		Valve : off Fan : off Alarm : off

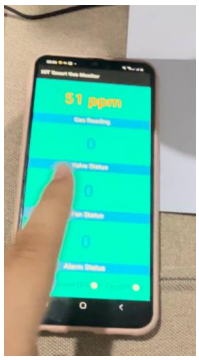
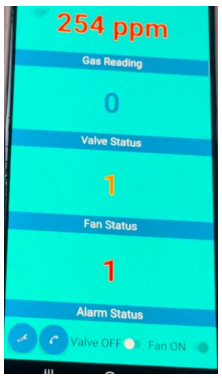
5 sec	51 ppm		Valve : off Fan : off Alarm : off
10 sec	254 ppm		Valve : off Fan : on Alarm: on

Table 2 Result and analysis

Figure 16 shows the result after the MQ-6 sensor detects gas leaks. After the gas leak is detected the ESP 32 starts triggering the GSM to make calls to the registered number in the coding.

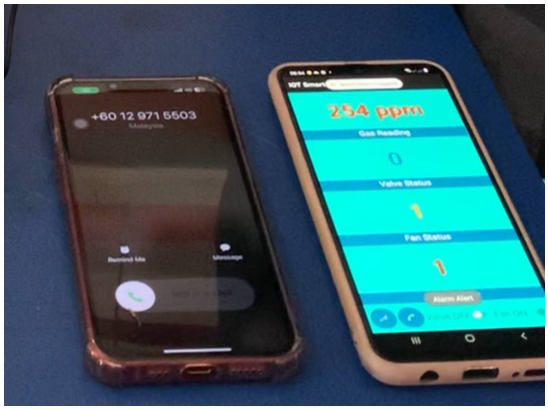


Figure 16 GSM make calling

4.3 Discussion

This Automatic Gas Leakage with phone call can detect the LPG at kitchen. Once the MQ-6 sensors detect any gas leak above 60 ppm , they will automatically call the particular number , the ventilator fan automatically on and the servo valve off . The ventilator fan will on until the gas reading below 60 ppm . We also can control the on off button of the fan and servo from the apps.When the sensors detect the gas , the red LED will blink.

CHAPTER 5

5 CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

This Automatic Gas Leakage with phone call finally at the end of the report after succeed and the result has been taken . This chapter were going to conclude all about the project and suggestion for further work to make this project for sufficient .

5.2 Conclusion

To conclude the project progress of the Automatic Gas Leakage with phone call for kitchen applicant is successful. The system functions can detect the presence of flammable gas and give alertness to the user through an alarm system and also through GSM. The essential of this system is to prevent explosion caused by gas leaks which can take lives by detecting the gas before it will explode, this system also will protect the property from losses caused by the explosion. More crucial, this system does not need human high skill to use, because it just needs to be ON and placed at the nearest gas stove. When the gas leak ,this project will detect the gas leak and give an alert to humans around them by triggering the alarm, automatically on the ventilator fan , off the servo valve gas . it also will call the users to take some action.

5.3 Suggestion for Future Work

The hardware development gas leakage detector using GSM for factory safety is complete. But some improvements in terms of design and additional functions can be made, to make this project more efficient and more priceless.

i. The sensor used in this project is only one gas sensor, we know that factory has huge coverage to cover to detect the gas leak. Adding more gas sensors will give more coverage to detect a gas leak.

ii. The protection of the sensor, in this system sensor, does not have waterproof features. This will make the life cycle of the sensor short, especially when doing maintenance, it will cause a short circuit, to overcome the problem, use the gas sensor that has protection.

iii. Adding a cutoff switch the function of this switch is to close the main switches that can react with the flammable gas this feature is also very essential in preventing the explosion.

CHAPTER 6

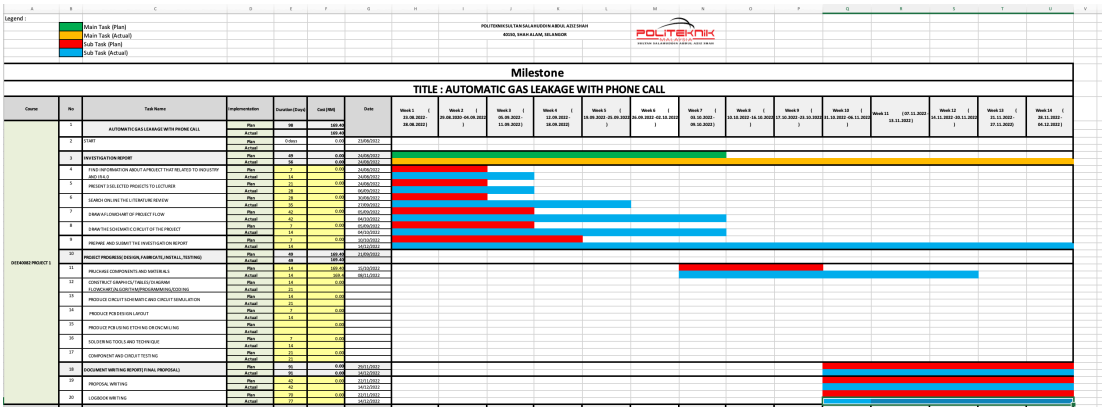
6 PROJECT MANAGEMENT AND COSTING

6.1 Introduction

A Gantt chart is a type of bar chart that illustrates a project schedule. It provides a visual representation of project tasks, their durations, and the dependencies between them. A Gantt chart is an effective tool for project managers to plan, schedule, and track progress. Gantt charts are widely used in various industries and are particularly helpful for visualizing complex projects with multiple tasks and dependencies. They offer a comprehensive overview of project timelines, aiding in planning, scheduling, and monitoring progress to ensure successful project completion.

6.2 Gant Chart and Activities of the Project

Project 1



Project 2

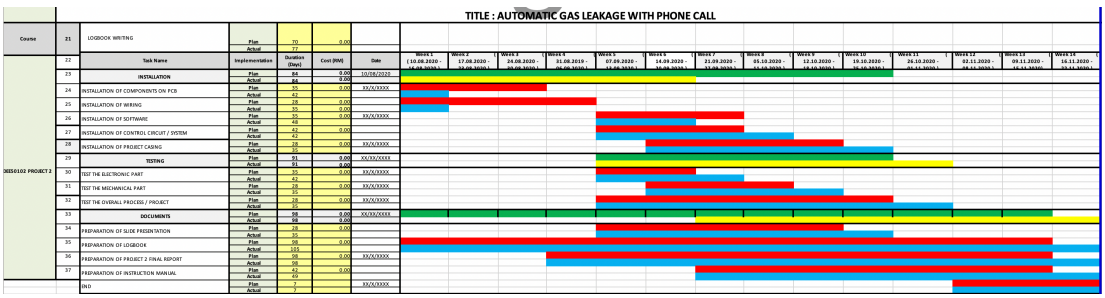


Table 3 Gantt chart of project

6.3 Cost and Budgeting

This project involves the cost of purchasing components and materials throughout its implementation. components involving cost are hardware ESP32, GSM SIM 900aModule, MQ-6 gas sensor,servo valve gas , ventilator fan, connecting cable,12V adapter, clear case. All of these components are purchased through online purchase methods to make it easier as well as save on costs.
The overall gross budget estimate in the implementation of this project is RM 116.28 and other expenses is at RM 50 as shown in Table 1 According to this budget cost, this project is can be considered as a less costly project. The cost of the project is also in line with one of the key features of a good project developer that is low cost but have a high quality project.

Table 4 Costing of project

No.	Component and materials	The unit price	Quantity	Total
1	ESP 32 Board	RM29	1	RM29
2	GSM SIM 900a Module	RM23.99	1	RM23.99
3	MQ-6 gas sensor	RM6.49	1	RM6.49
4	Relay	RM 1.49	1	RM1.49
5	12 V fan	RM9.52	1	RM9.52
6	Acrylic Case	RM35	1	RM35
7	Sim card	RM10	1	RM10
8	Buzzer	RM0.89	1	RM0.89
	List of other costing			
1	Postage	RM3.90	-	RM3.90
2	Accessories	RM5		RM5
			Total :	**116.28**
			Overall total	**RM200**

REFERENCES

1. Khan, Mohammad Monirujjaman. "Sensor-based gas leakage detector system." Engineering Proceedings 2.1 (2020): 28.

2. Leavline, E. Jebamalar, et al. "LPG gas leakage detection and alert system." International Journal of Electronics Engineering Research 9.7 (2017): 1095-1097.

3. olhe, B. D., P. A. Potdukhe, and N. S. Gawai. "Automatic lpg booking, leakage detection and real time gas measurement monitoring system." International Journal of Engineering Research & Technology (IJERT) 2.4 (2013): 1192-1195.

4. Shrivastava, Ashish, et al. "GSM based gas leakage detection system." International Journal of Technical Research and Applications 1.2 (2013): 42-45.

5. Raj, Arun, Athira Viswanathan, and T. Athul. "LPG gas monitoring system." IJITR 3.2 (2015): 1957-1960.

6. Raslavičius, Laurencas, et al. "Liquefied petroleum gas (LPG) as a medium-term option in the transition to sustainable fuels and transport." Renewable and Sustainable Energy Reviews 32 (2014): 513-525.

7. Demirbas, Ayhan. "Fuel properties of hydrogen, liquefied petroleum gas (LPG), and compressed natural gas (CNG) for transportation." Energy Sources 24.7 (2002): 601-610.

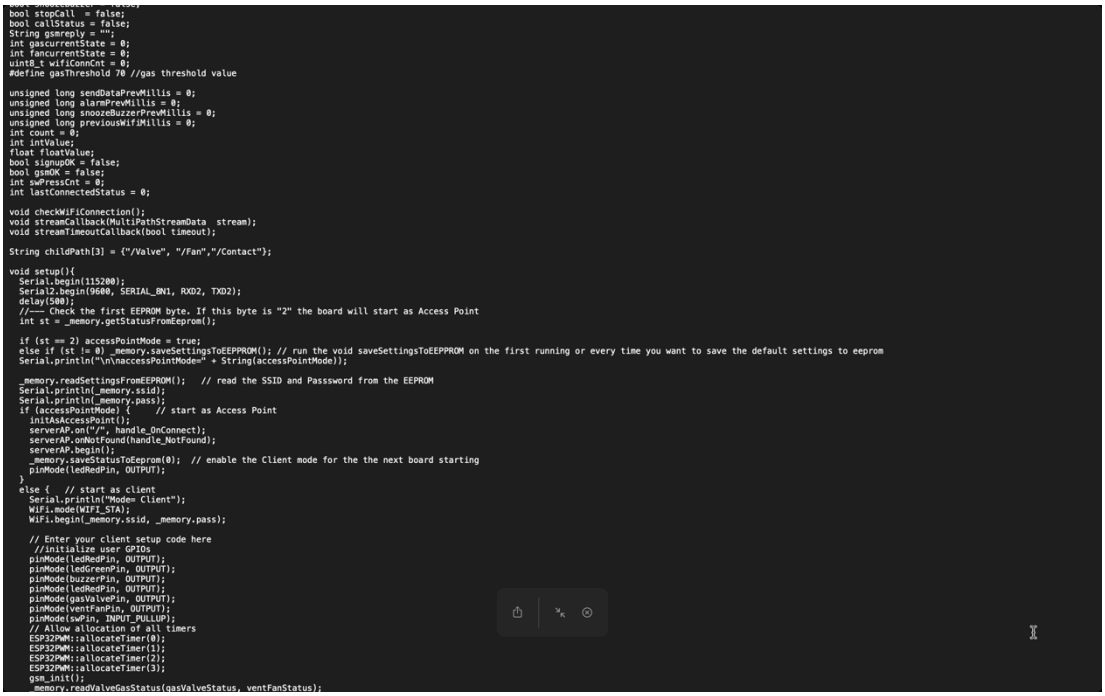
8. Rahim, M. S. N. A. "The current trends and challenging situations of fire incident statistics." Malaysian Journal of Forensic Sciences 6.1 (2015): 63-78.

9. Kebakaran, Tempat. "Forensic gas chromatography analysis of time elapsed gasoline in fire scene investigation." Malaysian Journal of Analytical Sciences 22.1 (2018): 72-79.

10. Ajiboye, A. T., et al. "Analytical determination of load resistance value for MQ-series gas sensors: MQ-6 as case study." TELKOMNIKA (Telecommunication Computing Electronics and Control) 19.2 (2021): 575-582.

7 APPENDICES

APPENDIX A- PROGRAMMING



```
bool stopCall  = false;
bool callStatus = false;
String gsmreply = "";
int gascurrentState = 0;
int fancurrentState = 0;
uint8_t wifiConnCnt = 0;
#define gasThreshold 70 //gas threshold value

unsigned long sendDataPrevMillis = 0;
unsigned long alarmPrevMillis = 0;
unsigned long snoozeBuzzerPrevMillis = 0;
unsigned long previousWifiMillis = 0;
int count = 0;
int intValue;
float floatValue;
bool signupOK = false;
bool gsmOK = false;
int swPressCnt = 0;
int lastConnectedStatus = 0;

void checkWiFiConnection();
void streamCallback(MultiPathStreamData  stream);
void streamTimeoutCallback(bool timeout);

String childPath[3] = {"/Valve", "/Fan","/Contact"};

void setup(){
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
  delay(500);
  //--- Check the first EEPROM byte. If this byte is "2" the board will start as Access Point
  int st = _memory.getStatusFromEeprom();

  if (st == 2) accessPointMode = true;
  else if (st != 0) _memory.saveSettingsToEEPPROM(); // run the void saveSettingsToEEPPROM on the first running or every time you want to save the default settings to eeprom
  Serial.println("\n\naccessPointMode=" + String(accessPointMode));

  _memory.readSettingsFromEEPROM();   // read the SSID and Passsword from the EEPROM
  Serial.println(_memory.ssid);
  Serial.println(_memory.pass);
  if (accessPointMode) {      // start as Access Point
    initAsAccessPoint();
    serverAP.on("/", handle_OnConnect);
    serverAP.onNotFound(handle_NotFound);
    serverAP.begin();
    _memory.saveStatusToEeprom(0);  // enable the Client mode for the the next board starting
    pinMode(ledRedPin, OUTPUT);
  }
  else {   // start as client
    Serial.println("Mode= Client");
    WiFi.mode(WIFI_STA);
    WiFi.begin(_memory.ssid, _memory.pass);

    // Enter your client setup code here
     //initialize user GPIOs
    pinMode(ledRedPin, OUTPUT);
    pinMode(ledGreenPin, OUTPUT);
    pinMode(buzzerPin, OUTPUT);
    pinMode(ledRedPin, OUTPUT);
    pinMode(gasValvePin, OUTPUT);
    pinMode(ventFanPin, OUTPUT);
    pinMode(swPin, INPUT_PULLUP);
    // Allow allocation of all timers
    ESP32PWM::allocateTimer(0);
    ESP32PWM::allocateTimer(1);
    ESP32PWM::allocateTimer(2);
    ESP32PWM::allocateTimer(3);
    gsm_init();
     _memory.readValveGasStatus(gasValveStatus, ventFanStatus);
```

```
#define RXD2 16
#define TXD2 17


#include <WiFi.h>
#include "WiFiGeneric.h"
#include <WebServer.h>
#include <FirebaseESP32.h>
#include <ESP32Servo.h>
//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"
#include "pitches.h"
#include "memory.h"
#include <cmath>

boolean debug = true;
memory _memory;
WebServer serverAP(80);   // the Access Point Server
boolean accessPointMode = false;    // is true every time the board is started as Access Point

/* ------ USER PARAMETERS--------*/
//Define Firebase Data object
FirebaseData fbdo, stream;
FirebaseAuth auth;
FirebaseConfig config;

Servo servoMotor;

TaskHandle_t Task0;
/* Enable these lines in case you want to change the default Access Point ip: 192.168.4.1. You have to enable the line:
  WiFi.softAPConfig(local_ip, gateway, subnet);
  on the void initAsAccessPoint too */
//IPAddress local_ip(192,168,1,1);
//IPAddress gateway(192,168,1,1);
//IPAddress subnet(255,255,255,0);

//define users parameter
#define writeInterval 3000 //3 seconds
#define normalAlarmInterval 1000
#define resetSnoozeBuzzerInterval 300000 //5minutes

//user GPIO
#define gasSensorPin 39
#define gasValvePin 27 //pwm
#define ventFanPin 25
#define buzzerPin  26//pwm
#define ledRedPin 2
#define ledGreenPin 4
#define swPin 34


struct sensors{
  int gasReading;
}_sensors;

int gasValveStatus = 0;
int prevgasValveStatus = 3;
int ventFanStatus = 0;
int prevventFanStatus = 3;
int alarmStatus = 0;
int prevAlarmStatus = 3;
int ledStatus = 0;
int ledInterval = normalAlarmInterval;
bool snoozeBuzzer = false;
bool stopCall  = false;
```

```
    _memory.readValveGasStatus(gasValveStatus, ventFanStatus);
//    ledcSetup(0, 2000, 8);
//    ledcAttachPin(buzzerPin, 0);

    //servoMotor.setPeriodHertz(50);    // standard 50 hz servo

  }
}

void loop(){
    if(accessPointMode){
      serverAP.handleClient();
      playAccessPointLed();       // blink the LED every time the board works as Access Point
    }
    else{
      checkWiFiConnection();
      readAndUploadSensors_toDatabase();
      Valve_FanController();
      alarmChecker();
      alarmController();
      swChecker();
      serialEvent();
      callHandler();
    }

}


void checkWiFiConnection() {
  //check if every connection is OK
  if(WiFi.isConnected() && Firebase.ready() && gsmOK)  {
    digitalWrite(ledGreenPin, HIGH);
    if(alarmStatus == 0)
      digitalWrite(ledRedPin, LOW);
  }
  else{
    digitalWrite(ledGreenPin, LOW);
    if(alarmStatus == 0)
      digitalWrite(ledRedPin, HIGH);
  }
  if(WiFi.status() != WL_CONNECTED && millis()-previousWifiMillis >= 3000) {
//    WiFi.disconnect();
//    WiFi.reconnect();
    if (debug) Serial.print("wifi disconnect");
    if (debug) Serial.println(wifiConnCnt);
    digitalWrite(ledRedPin, !digitalRead(ledRedPin));
    previousWifiMillis = millis();
    if(wifiConnCnt <= 5)
      wifiConnCnt++;
    else
      ESP.restart();

  }
  else if(WiFi.isConnected()&& millis()-previousWifiMillis >= 3000) {
    previousWifiMillis = millis();
    if (debug) Serial.println("wifi connected");
    wifiConnCnt = 0;
    if(lastConnectedStatus == 0){
      //  typedef enum {
//    WIFI_POWER_19_5dBm = 78,// 19.5dBm
//    WIFI_POWER_19dBm = 76,// 19dBm
//    WIFI_POWER_18_5dBm = 74,// 18.5dBm
//    WIFI_POWER_17dBm = 68,// 17dBm
//    WIFI_POWER_15dBm = 60,// 15dBm
//    WIFI_POWER_13dBm = 52,// 13dBm
//    WIFI_POWER_11dBm = 44,// 11dBm
//    WIFI_POWER_8_5dBm = 34,// 8.5dBm
//    WIFI_POWER_7dBm = 28,// 7dBm
```

```
//    WIFI_POWER_7dBm = 28,// 7dBm
//    WIFI_POWER_5dBm = 20,// 5dBm
//    WIFI_POWER_2dBm = 8,// 2dBm
//    WIFI_POWER_MINUS_1dBm = -4// -1dBm
//} wifi_power_t;
    WiFi.setTxPower(WIFI_POWER_11dBm);
    //WIFI_POWER_18_5dBm = 74
    int a = WiFi.getTxPower();
    Serial.print("TX power:");
   Serial.println(a);
        /* Assign the api key (required) */
    // config.api_key = API_KEY;

     /* Assign the RTDB URL (required) */
     config.database_url = DATABASE_URL;
     config.signer.tokens.legacy_token = DATABASE_SECRET;
//      config.signer.test_mode = true;
     /* Sign up */
//     if (Firebase.signUp(&config, &auth, "", "")){
//       Serial.println("ok");
//       signupOK = true;
//     }
//     else{
//       Serial.printf("%s\n", config.signer.signupError.message.c_str());
//     }
     /* Assign the callback function for the long running token generation task */
     config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

     Firebase.begin(&config, &auth);
     Firebase.reconnectWiFi(true);

     Firebase.RTDB.setMaxRetry(&fbdo, 3);
     Firebase.RTDB.setMaxErrorQueue(&fbdo, 30);
     //Firebase Stream for fast read every data change
     if(!Firebase.RTDB.beginMultiPathStream(&stream, "IOTApp"))
       Serial.printf("stream  begin error, %s\n\n", stream.errorReason().c_str());
     Firebase.RTDB.setMultiPathStreamCallback(&stream, streamCallback, streamTimeoutCallback);


     lastConnectedStatus = 1;
   }
  }

}

void streamCallback(MultiPathStreamData  stream)
{
  size_t numChild = sizeof(childPath) / sizeof(childPath[0]);

  for (size_t i = 0; i < numChild; i++)
  {
    if (stream.get(childPath[i]))
    {
      if(i == 0){

        gasValveStatus = stream.value.toInt();
        if (debug) Serial.print("Successful READ from APP VALVE " + gasValveStatus);
        if (debug) Serial.println(gasValveStatus);
      }

      else if(i == 1){

        ventFanStatus = stream.value.toInt();
        if (debug) Serial.print("Successful READ from APP FAN ");
        if (debug) Serial.println(ventFanStatus);
      }

      else if(i == 2){
```

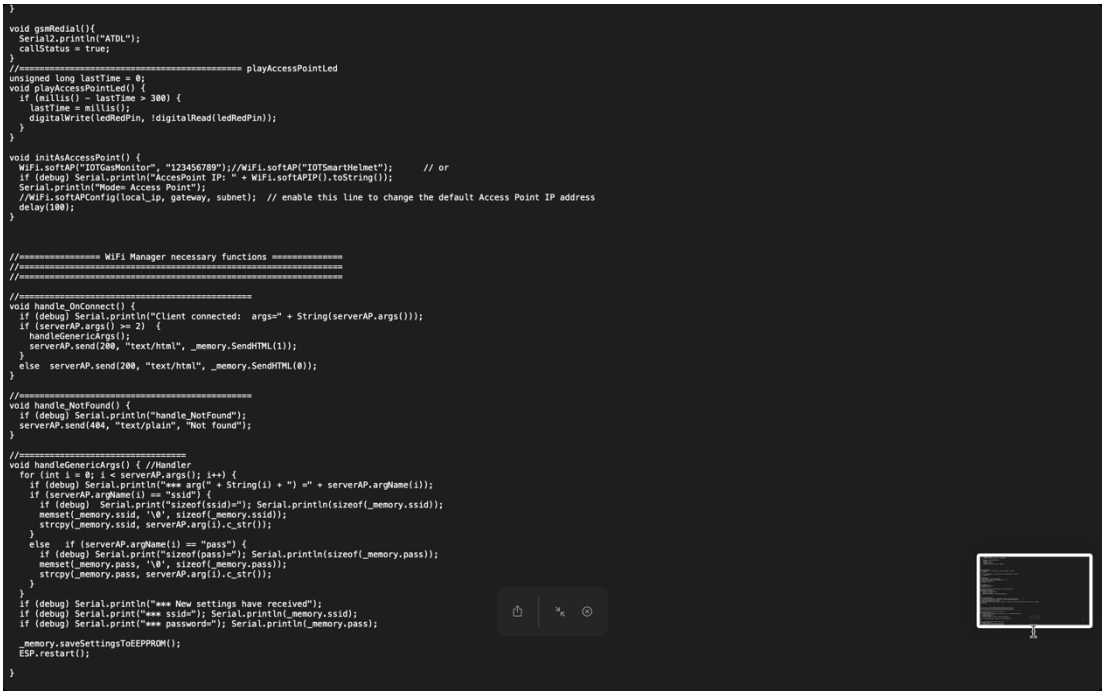
```
      else if(i == 1){

        ventFanStatus = stream.value.toInt();
        if (debug) Serial.print("Successful READ from APP FAN ");
        if (debug) Serial.println(ventFanStatus);
      }

      else if(i == 2){

        String _string = stream.value;
        if (debug) Serial.println("Successful READ from APP CONTACT "+ _string);
        strcpy(_memory.ctc, _string.c_str());
        _memory.saveSettingsToEEPROM();
      }

    }
  }
}

void streamTimeoutCallback(bool timeout)
{
  if (timeout)
    Serial.println("stream timed out, resuming...\n");

  if (!stream.httpConnected())
    Serial.printf("error code: %d, reason: %s\n\n", stream.httpCode(), stream.errorReason().c_str());
}
/* Funtion to read from Sensors and upload to database every set Interval */
void readAndUploadSensors_toDatabase(){
  //read users GPIO

if((millis() - sendDataPrevMillis > writeInterval || sendDataPrevMillis == 0)){
  _sensors.gasReading = map(analogRead(gasSensorPin), 0, 4096, 0, 255);
  if (Firebase.ready())
  {

    sendDataPrevMillis = millis();

    if (debug) Serial.print("Gas Sensor: ");
    if (debug) Serial.println(_sensors.gasReading);
    // Write gas sensor to database
    if (!Firebase.RTDB.setInt(&fbdo, "IOTGasMon/Gas", _sensors.gasReading)){
      if (debug) Serial.println("PASSED");
      if (debug) Serial.println("PATH: " + fbdo.dataPath());
      if (debug) Serial.println("TYPE: " + fbdo.dataType());
    }
    else {
      if (debug) Serial.println("FAILED");
      if (debug) Serial.println("REASON: " + fbdo.errorReason());
    }
  }
}
//upload to database every writeInterval/3s

}

/* Function to check for the sensors reading exceeded safe threshold/limit */
void alarmChecker(){
  if(_sensors.gasReading > gasThreshold){
    alarmStatus = 1; //set alarm ON
    gasValveStatus = 0; // turn OFF valve
    ventFanStatus = 1; // turn ON fan
  }
  else{
    alarmStatus = 0;
    //gasValveStatus = prevgasValveStatus;
    //ventFanStatus = prevventFanStatus;
    //digitalWrite(ledRedPin, LOW);
    noTone(buzzerPin);
    stopCall = false;
```

```
    stopCall = false;
    callStatus = false;
  }

}
/* Function to control buzzer/alarm sound */
void alarmController(){
  if(alarmStatus == 1){
    if(millis() - alarmPrevMillis > ledInterval || alarmPrevMillis == 0){
      alarmPrevMillis = millis();
      digitalWrite(ledRedPin, ledStatus);
      ledStatus = !ledStatus;

      if(!snoozeBuzzer) playGasWarning(buzzerPin, ledStatus);
      else noTone(buzzerPin);


    }
  }
}
/* Function to control the gas valve */
void Valve_FanController(){

  if(alarmStatus != prevAlarmStatus){
    if(Firebase.ready()){
      if (Firebase.RTDB.setInt(&fbdo, "IOTGasMon/Alarm", alarmStatus )){
      if (debug) Serial.println("PASSED");
      if (debug) Serial.println("PATH: " + fbdo.dataPath());
      if (debug) Serial.println("TYPE: " + fbdo.dataType());
      if(prevAlarmStatus == 1 && alarmStatus == 0){
        Serial.print("changes in alarm ");
        Firebase.RTDB.getInt(&fbdo, "/IOTApp/Valve/");
        gasValveStatus = fbdo.intData();
        Firebase.RTDB.getInt(&fbdo, "/IOTApp/Fan/");
        ventFanStatus = fbdo.intData();
        Serial.print(gasValveStatus);
        Serial.println(ventFanStatus);
        prevgasValveStatus = 3;
        prevventFanStatus = 3;
      }
      }
      prevAlarmStatus = alarmStatus;

    }
  }
    if(ventFanStatus != prevventFanStatus ){
    turnOnOffFan(ventFanStatus);
    if(Firebase.ready()){
      if (Firebase.RTDB.setInt(&fbdo, "IOTGasMon/Fan", ventFanStatus )){
      if (debug) Serial.println("PASSED");
      if (debug) Serial.println("PATH: " + fbdo.dataPath());
      if (debug) Serial.println("TYPE: " + fbdo.dataType());
      prevventFanStatus = ventFanStatus;
      }
    }

  }
  if(gasValveStatus != prevgasValveStatus){
    turnOnOffValve(gasValveStatus);
    if(Firebase.ready()){
      if (Firebase.RTDB.setInt(&fbdo, "IOTGasMon/Valve", gasValveStatus )){
      if (debug) Serial.println("PASSED");
      if (debug) Serial.println("PATH: " + fbdo.dataPath());
      if (debug) Serial.println("TYPE: " + fbdo.dataType());
      prevgasValveStatus = gasValveStatus;
      }
    }
```

```
  }


}

void turnOnOffValve(int gasValveStatus){
  servoMotor.attach(gasValvePin,1000,2500);
  if(debug)Serial.print("gas valve ");
  if(debug)Serial.println(gasValveStatus);
  if(gasValveStatus == 1){ //turn ON gas valve
    //for (int pos = 0; pos <= 180; pos += 1) {
    // in steps of 1 degree
    servoMotor.write(180);
    delay(1000);
    servoMotor.detach();
    //delay(15); // waits 15ms to reach the position
    //}
  }
  else if(gasValveStatus == 0){ // turn OFF gas valve
    //for (int pos = 180; pos >= 0; pos -= 1) {
    servoMotor.attach(gasValvePin,1000,2500);
    servoMotor.write(0);
    delay(1000);
    servoMotor.detach();
    //delay(15); // waits 15ms to reach the position
    //}
  }
}

void turnOnOffFan(int ventFanStatus){
  if(debug)Serial.print("fan ");
  if(debug)Serial.println(ventFanStatus);
  digitalWrite(ventFanPin, ventFanStatus);
}

/* Function to interpret the switch press */
void swChecker(){
  //switch is connected to GND, when pressed read as zero
  while(!digitalRead(swPin)){
      swPressCnt++;
      if (debug) Serial.println(swPressCnt);
      delay(1000);
  }
  if(swPressCnt == 0) return;
  //short press to snooze and unsnooze buzzer
  if(swPressCnt > 1 && swPressCnt < 5){
    snoozeBuzzer = !snoozeBuzzer;
    if (debug) Serial.println("Snooze buzzer");
    snoozeBuzzerPrevMillis = millis();
  }
  // extra long press to set up wifi ssid and password
  else if(swPressCnt>=10){
    if (!accessPointMode) _memory.saveStatusToEeprom(2);   // write the number 2 to the eeprom
      ESP.restart();
  }

  swPressCnt = 0;

  //reset snooze buzzer after 5 minutes
  if((snoozeBuzzer == 1) && (millis() - snoozeBuzzerPrevMillis > resetSnoozeBuzzerInterval)){
    snoozeBuzzer = !snoozeBuzzer;
  }

}

void gsm_init(){
```

```
void gsm_init(){
  //GSM handshake
  uint8_t gsmAT = 0;
  uint8_t gsmOffEcho = 0;
  while(gsmAT<10){
    Serial2.println("AT");
    delay(500);
    if(Serial2.available()){
    if(Serial2.readString().indexOf("OK")>=0){
      if(debug)Serial.println("GSM at OK");
      gsmOK = true;
      break;
    }
    else
      gsmOK = false;

    }
    gsmAT++;
  }
  //GSM off echo
  while(gsmOffEcho<10){
    Serial2.println("ATE0");
    delay(500);
    if(Serial2.available()){
    if(Serial2.readString().indexOf("OK")>=0){
      if(debug)Serial.println("GSM off echo OK");
      gsmOK = true;
      break;
    }
    else
      gsmOK = false;

    }
    gsmOffEcho++;
  }
}

void serialEvent()
{
  if(Serial2.available())

  {
    gsmreply = Serial2.readString();

    if(gsmreply.indexOf("OK")>=0){
      gsmreply = "";
      callStatus = false;
      stopCall = true; // only stops call when user picked up
      if(debug) Serial.println("Call : OK");
    }

    if(gsmreply.indexOf("BUSY")>=0){
      gsmreply = "";
      callStatus = false;
      stopCall = false;
      if(debug) Serial.println("Call : BUSY");
    }

    if(gsmreply.indexOf("NO CARRIER")>=0){
       gsmreply = "";
       callStatus = false;
       stopCall = false;
       if(debug) Serial.println("Call : No carrier ERROR");
    }

    if(gsmreply.indexOf("NO ANSWER")>=0){
      gsmreply = "";
      callStatus = false;
      stopCall = false;
```

```
      stopCall = false;
      if(debug) Serial.println("Call : NO ANSWER");
    }
    if(gsmreply.indexOf("ERROR")>=0){
      gsmreply = "";
      callStatus = false;
      stopCall = false;
      if(debug) Serial.println("Call : ERROR");
    }

  }

}
void callHandler(){
  if(alarmStatus == 1 && callStatus == false && stopCall == false){
    gsmCall();
  }
  // else if(alarmStatus == 1 && callStatus == false && stopCall == true){
  //   gsmRedial();
  // }
}
void gsmCall(){
  if(debug)Serial.println("GSM CALLING");
  String callCMD = "ATD" + (String)_memory.ctc + ";";
  Serial2.println(callCMD);
  callStatus = true;
}

void gsmRedial(){
  Serial2.println("ATDL");
  callStatus = true;
}
//======================================== playAccessPointLed
unsigned long lastTime = 0;
void playAccessPointLed() {
  if (millis() - lastTime > 300) {
    lastTime = millis();
    digitalWrite(ledRedPin, !digitalRead(ledRedPin));
  }
}

void initAsAccessPoint() {
  WiFi.softAP("IOTGasMonitor", "123456789");//WiFi.softAP("IOTSmartHelmet");      // or
  if (debug) Serial.println("AccessPoint IP: " + WiFi.softAPIP().toString());
  Serial.println("Mode= Access Point");
  //WiFi.softAPConfig(local_ip, gateway, subnet);  // enable this line to change the default Access Point IP address
  delay(100);
}


//=============== WiFi Manager necessary functions =============
//==============================================================
//==============================================================

//===========================================
void handle_OnConnect() {
  if (debug) Serial.println("Client connected:  args=" + String(serverAP.args()));
  if (serverAP.args() >= 2)  {
    handleGenericArgs();
    serverAP.send(200, "text/html", _memory.SendHTML(1));
  }
  else  serverAP.send(200, "text/html", _memory.SendHTML(0));
}

//===========================================
void handle_NotFound() {
  if (debug) Serial.println("handle_NotFound");
  serverAP.send(404, "text/plain", "Not found");
```



```
}

void gsmRedial(){
  Serial2.println("ATDL");
  callStatus = true;
}
//========================================== playAccessPointLed
unsigned long lastTime = 0;
void playAccessPointLed() {
  if (millis() - lastTime > 300) {
    lastTime = millis();
    digitalWrite(ledRedPin, !digitalRead(ledRedPin));
  }
}

void initAsAccessPoint() {
  WiFi.softAP("IOTGasMonitor", "123456789");//WiFi.softAP("IOTSmartHelmet");      // or
  if (debug) Serial.println("AccesPoint IP: " + WiFi.softAPIP().toString());
  Serial.println("Mode= Access Point");
  //WiFi.softAPConfig(local_ip, gateway, subnet);  // enable this line to change the default Access Point IP address
  delay(100);
}


//=============== WiFi Manager necessary functions =============
//==============================================================
//==============================================================

//==========================================
void handle_OnConnect() {
  if (debug) Serial.println("Client connected:  args=" + String(serverAP.args()));
  if (serverAP.args() >= 2)  {
    handleGenericArgs();
    serverAP.send(200, "text/html", _memory.SendHTML(1));
  }
  else  serverAP.send(200, "text/html", _memory.SendHTML(0));
}

//==========================================
void handle_NotFound() {
  if (debug) Serial.println("handle_NotFound");
  serverAP.send(404, "text/plain", "Not found");
}

//==============================
void handleGenericArgs() { //Handler
  for (int i = 0; i < serverAP.args(); i++) {
    if (debug) Serial.println("*** arg(" + String(i) + ") =" + serverAP.argName(i));
    if (serverAP.argName(i) == "ssid") {
      if (debug)  Serial.print("sizeof(ssid)="); Serial.println(sizeof(_memory.ssid));
      memset(_memory.ssid, '\0', sizeof(_memory.ssid));
      strcpy(_memory.ssid, serverAP.arg(i).c_str());
    }
    else   if (serverAP.argName(i) == "pass") {
      if (debug) Serial.print("sizeof(pass)="); Serial.println(sizeof(_memory.pass));
      memset(_memory.pass, '\0', sizeof(_memory.pass));
      strcpy(_memory.pass, serverAP.arg(i).c_str());
    }
  }
  if (debug) Serial.println("*** New settings have received");
  if (debug) Serial.print("*** ssid="); Serial.println(_memory.ssid);
  if (debug) Serial.print("*** password="); Serial.println(_memory.pass);

  _memory.saveSettingsToEEPROM();
  ESP.restart();

}
```

\n"; ptr += " \n"; ptr += "\n"; ptr += "\n"; ptr += "\n"; ptr += "

ESP WiFi Manager Using EEPROM

\n"; if (st == 1)ptr += "

WiFi settings has saved successfully!

\n"; else if (st == 2)ptr += "

WIFI Credentials has saved successfully!

\n"; else ptr += "

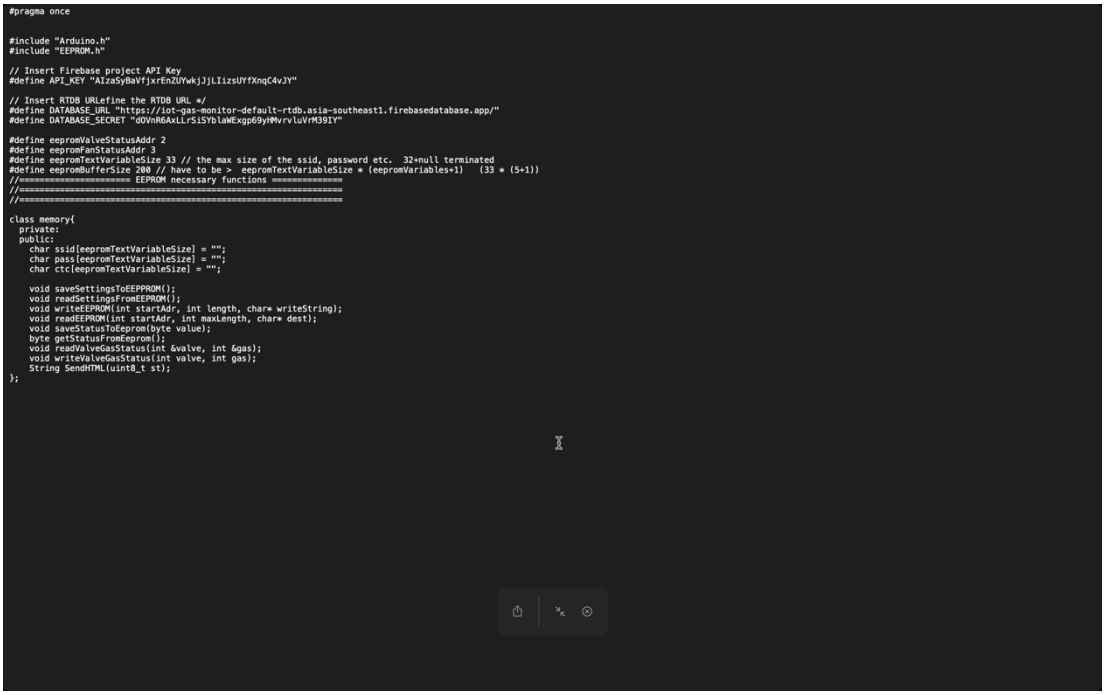
Enter the WiFi settings

\n"; ptr += "
"; ptr += "
WiFi SSID \"";
\n"; ptr += "
WiFi Password \"";
\n"; ptr += "
\"Submit\"accesskey=\"s\"

"; ptr += "

\n"; ptr += "\n"; ptr += "\n"; return ptr; /* Font Definitions */ @font-face {font-family:"Cambria Math"; panose-1:2 4 5 3 5 4 6 3 2 4; mso-font-charset:0; mso-generic-font-family:roman; mso-font-pitch:variable; mso-font-signature:3 0 0 0 1 0;} /* Style Definitions */ p.MsoNormal, li.MsoNormal, div.MsoNormal {mso-style-unhide:no; mso-style-qformat:yes; mso-style-parent:""; margin-top:0cm; margin-right:0cm; margin-bottom:8.0pt; margin-left:0cm; line-height:107%; mso-pagination:widow-orphan; font-size:11.0pt; font-family:"Calibri",sans-serif; mso-ascii-font-family:Calibri; mso-ascii-theme-font:minor-latin; mso-fareast-font-family:"Times New Roman"; mso-fareast-theme-font:minor-fareast; mso-hansi-font-family:Calibri; mso-hansi-theme-font:minor-latin; mso-bidi-font-family:Arial; mso-bidi-theme-font:minor-bidi; mso-font-kerning:1.0pt; mso-ligatures:standardcontextual;} .MsoChpDefault {mso-style-type:export-only; mso-default-props:yes; font-family:"Calibri",sans-serif; mso-ascii-font-family:Calibri; mso-ascii-theme-font:minor-latin; mso-fareast-font-family:"Times New Roman"; mso-fareast-theme-font:minor-fareast; mso-hansi-font-family:Calibri; mso-hansi-theme-font:minor-latin; mso-bidi-font-family:Arial; mso-bidi-theme-font:minor-bidi;} .MsoPapDefault {mso-style-type:export-only; margin-bottom:8.0pt; line-height:107%;} @page WordSection1 {size:595.3pt 841.9pt; margin:72.0pt 72.0pt 72.0pt 72.0pt; mso-header-margin:35.4pt; mso-footer-margin:35.4pt; mso-paper-source:0;} div.WordSection1 {page:WordSection1;} -->

\n"; ptr += "\n"; ptr += "



```
#pragma once


#include "Arduino.h"
#include "EEPROM.h"

// Insert Firebase project API Key
#define API_KEY "AIzaSyBaVfjxrEnZUYwkjJjLIizsUYfXnqC4vJY"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://iot-gas-monitor-default-rtdb.asia-southeast1.firebasedatabase.app/"
#define DATABASE_SECRET "dOVnR6AxLLrSiSYblaWExgp69yHMvrvluVrM39IY"

#define eepromValveStatusAddr 2
#define eepromFanStatusAddr 3
#define eepromTextVariableSize 33 // the max size of the ssid, password etc.  32+null terminated
#define eepromBufferSize 200 // have to be >  eepromTextVariableSize * (eepromVariables+1)   (33 * (5+1))
//==================== EEPROM necessary functions ============
//============================================================
//============================================================

class memory{
  private:
  public:
    char ssid[eepromTextVariableSize] = "";
    char pass[eepromTextVariableSize] = "";
    char ctc[eepromTextVariableSize] = "";

    void saveSettingsToEEPROM();
    void readSettingsFromEEPROM();
    void writeEEPROM(int startAdr, int length, char* writeString);
    void readEEPROM(int startAdr, int maxLength, char* dest);
    void saveStatusToEeprom(byte value);
    byte getStatusFromEeprom();
    void readValveGasStatus(int &valve, int &gas);
    void writeValveGasStatus(int valve, int gas);
    String SendHTML(uint8_t st);
};
```

```
#include "esp32-hal-ledc.h"
#include "Arduino.h"
/*************************************************
 * Public Constants
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
```

```
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978

int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};
int siren[2] = {NOTE_FS6, NOTE_CS6};
int lowLightSound[] = {NOTE_B6, NOTE_C6, NOTE_C6};
int hotTempSound[2] = {800, 970};
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};
void playMelody(int buzzerPin){
  for (int thisNote = 0; thisNote < 8; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(buzzerPin, melody[thisNote], noteDuration);

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(buzzerPin);
  }
}

void playEmergencySiren(int buzzerPin, int index){
         tone(buzzerPin, siren[index]);
}
void playGasWarning(int buzzerPin, int onOff){
         if(onOff == 1)
           tone(buzzerPin, 970);
         else
           noTone(buzzerPin);
}
void playLowLightWarning(int buzzerPin){
  for (int thisNote = 0; thisNote < 3; thisNote++) {
    int noteDuration = 1000 / 4;
    tone(buzzerPin, lowLightSound[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(buzzerPin);
  }
}
void playHighTempWarning(int buzzerPin){
  for (int thisNote = hotTempSound[0]; thisNote <= hotTempSound[1]; thisNote++) {
    tone(buzzerPin, thisNote);
    delay(15);
  }
  noTone(buzzerPin);
}
```

APPENDIX B- PROJECT PROTOTYPE AND OVERVIEW