

INVESTIGATION OF IOT DEVICES FOR
FIRE ALARM SYSTEMS USING RASPBERRY
PI

NIK NURJIHAN NISRINA BINTI NIK ZAID

BACHELOR OF ELECTRICAL
ENGINEERING (ELECTRONICS) WITH
HONOURS

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : Nik Nurjihan Nisrina Binti Nik Zaid

Date of Birth : 02 October 1997

Title : Investigation Of IOT Devices For Fire Alarm Systems Using
Raspberry Pi

Academic Session : 2021/2022

I declare that this thesis is classified as:

- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*
- RESTRICTED (Contains restricted information as specified by the organization where research was done)*
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:



(Student's Signature)

971002-03-6130

New IC/Passport Number
Date: 11 February 2022



(Supervisor's Signature)

Dr. Norazlianie Binti Sazali

Name of Supervisor
Date: 11 February 2022

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

| | |
|---------------|------------------------------------------------------------------------|
| Author's Name | Nik Nurjihan Nisrina Binti Nik Zaid |
| Thesis Title | Investigation of IOT Devices For Fire Alarm Systems Using Raspberry Pi |

| | |
|---------|-------|
| Reasons | (i) |
| | (ii) |
| | (iii) |

Thank you.

Yours faithfully,



(Supervisor's Signature)

Date: 11 February 2022

Stamp:

DR. NORZLIANE BINTI SAZALI
PENYARAH KANAN UNIVERSITI
FAKULTI KEJURUTERAAN MEKANIKA
UNIVERSITI MALAYSIA PAHANG
26600 PEKAN, PAHANG.
TEL: 09-424 5846

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.



SUPERVISOR's DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the Bachelor of Electrical Engineering (Electronics) with Honours.

A handwritten signature in black ink, appearing to be 'N. Sazali', is positioned above a horizontal line.

(Supervisor's Signature)

Full Name : Dr. Norazlianie Binti Sazali

Position : Senior Lecturer

Date : 11 February 2022

(Co-supervisor's Signature)

Full Name :

Position :

Date :



STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

A handwritten signature in black ink, appearing to be 'Nik Nurjihan Nisrina Binti Nik Zaid', is written above a horizontal line.

(Student's Signature)

Full Name : Nik Nurjihan Nisrina Binti Nik Zaid

ID Number : EA18143

Date : 11 February 2022

INVESTIGATION OF IOT DEVICES FOR FIRE ALARM DETECTION SYSTEMS
USING RASPBERRY PI

NIK NURJIHAN NISRINA BINTI NIK ZAID

Thesis submitted in fulfillment of the requirements
for the award of
Bachelor of Electrical Engineering (Electronics) with Honours

College of Engineering
UNIVERSITI MALAYSIA PAHANG

FEBRUARY 2022

ACKNOWLEDGEMENTS

Alhamdulillah, praise to Allah S.W.T, our creator. I am so blessed that I have managed to put an end to my Final Year Project session successfully with Allah's blessings I would like to thank Him for giving me good health and ability to go through my Final Year Project peacefully and well.

I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. A special gratitude I give to my supervisor, Ts. Dr. Norazlianie Binti Sazali, whose contribution in stimulating suggestions and encouragement, helped me to through my project and for their careful and precious guidance which were extremely valuable for my study both theoretically and practically.

I also want to express my deepest thanks to my senior, Ridzuan Hakimi M. Jafri, for helping me during my project session with abundance of information and guidance to ease my journey during the project. I choose this moment to acknowledge his contribution gratefully.

The project opportunity was a great chance for learning and professional development. Therefore, I consider myself as an incredibly lucky student as I was provided with an opportunity to be a part of it. I perceive as this opportunity as a big milestone in my study development.

Lastly, I also want to thank to my family for their love, sacrifice, and continuous support throughout my whole university life.

ABSTRACT

Every home and industrial building in today's world, from the simplest to the most complex is fitted with an automation system. Fire has the potential to kill people and destroy whole nations. Fire is an accidental event. This study investigate the design of a fire alarm detection system based on the Internet of Things (IoT) and the potency of an alarm system with the IoT in reducing time taken for system to alert the user. The design utilizes a system prototype with wireless access in an open source physical computing platform based on Raspberry Pi technology using a personal computer that provides the communication and user interface for both the control system and the fire alarm systems. This includes wireless connectivity between sensors and apps for monitoring and control. The sensors using to measure the related parameters that is temperature, presence of gas and humidity of the area. The Raspberry Pi is programmed to turn on the buzzer when the temperature and smoke reach a threshold value. This system equipped with WiFi module built in the Raspberry Pi will send the notification to the user via Blynk apps. The trial findings demonstrated the Raspberry Pi's affordability, effectiveness, and responsiveness in reducing the time it takes for the system to alert the user.

ABSTRAK

Setiap bangunan rumah dan perindustrian di dunia hari ini, dari yang paling sederhana hingga yang paling kompleks dilengkapi dengan sistem automasi. Api mempunyai potensi untuk membunuh orang dan memusnahkan seluruh manusia. Kebakaran adalah kejadian yang tidak sengaja. Kajian ini mengkaji reka bentuk sistem pengesanan penggera kebakaran berdasarkan 'Internet of Things (IoT)' dan potensi sistem penggera dengan IoT dalam mengurangkan masa yang diperlukan agar sistem dapat memberi amaran kepada pengguna. Reka bentuk menggunakan prototaip sistem dengan akses tanpa wayar dalam platform pengkomputeran fizikal sumber terbuka berdasarkan teknologi Raspberry Pi menggunakan komputer peribadi yang menyediakan komunikasi di antara muka pengguna untuk kedua-dua sistem kawalan dan sistem penggera kebakaran. Ini termasuk penyambungan tanpa wayar antara sensor dan aplikasi untuk pemantauan dan kawalan. Sensor yang digunakan untuk mengukur parameter yang berkaitan iaitu suhu, kehadiran gas dan kelembapan kawasan. Raspberry Pi berfungsi untuk menghidupkan bunyi ketika suhu dan asap mencapai nilai yang telah ditetapkan. Sistem ini dilengkapi dengan modul WiFi yang dibina di Raspberry Pi akan menghantar pemberitahuan kepada pengguna melalui aplikasi Blynk. Hasil daripada eksperimen ini menunjukkan kemampuan, keberkesanan, dan responsif Raspberry Pi dalam mengurangkan masa yang diperlukan oleh sistem untuk memberi amaran kepada pengguna.

TABLE OF CONTENT

| | |
|-----------------------------------------|------------|
| DECLARATION | |
| ACKNOWLEDGEMENTS | ii |
| ABSTRACT | iii |
| ABSTRAK | iv |
| TABLE OF CONTENT | v |
| LIST OF TABLES | x |
| LIST OF FIGURES | xi |
| LIST OF ABBREVIATIONS | xiv |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Project Background | 1 |
| 1.2 Problem Statement | 2 |
| 1.3 Objectives | 2 |
| 1.4 Significance | 3 |
| 1.5 Scopes and Limitations of the Study | 3 |
| 1.6 Chapter Summary | 4 |
| CHAPTER 2 LITERATURE REVIEW | 5 |
| 2.1 Introduction | 5 |
| 2.2 Internet of Things (IoT) | 5 |
| 2.2.1 IoT Vision | 6 |
| 2.2.2 IoT History | 6 |

| | | |
|------------------------------|-------------------------------------------------------------------------|-----------|
| 2.2.3 | IoT Architecture | 7 |
| 2.3 | Fire Detection Devices | 8 |
| 2.3.1 | Principle of Fire Detector | 10 |
| 2.4 | Raspberry Pi | 11 |
| 2.5 | Comparison Raspberry Pi with Arduino | 12 |
| 2.5.1 | Arduino | 12 |
| 2.5.2 | Comparison between Raspberry Pi and Arduino | 13 |
| 2.5.3 | Advantages of Raspberry Pi | 14 |
| 2.5.4 | Disadvantages of Raspberry Pi | 14 |
| 2.6 | Automatic smoke detection system with favoriot Platform using Iot | 15 |
| 2.7 | Intelligent fire detection and alert system using LabVIEW | 16 |
| 2.8 | Fire Detection System using Raspberry Pi | 17 |
| 2.9 | Arduino based Fire Detection and Alarm System Using Smoke Sensor | 18 |
| 2.10 | Redundant for Fire Monitoring System Using Raspberry Pi and Arduino Uno | 19 |
| 2.11 | Comparison between different approached study | 20 |
| 2.12 | Chapter Summary | 22 |
| CHAPTER 3 METHODOLOGY | | 23 |
| 3.1 | Introduction | 23 |
| 3.2 | Project Management | 23 |
| 3.2.1 | Gantt Chart of PSM 1 | 24 |
| 3.2.2 | Gantt Chart of PSM 2 | 25 |

| | | |
|-------|-----------------------------------------------------|----|
| 3.3 | Block Diagram | 26 |
| 3.4 | Flowchart | 27 |
| 3.5 | Circuit Diagram | 28 |
| 3.6 | Product Diagram/Sketch | 30 |
| 3.7 | Simulation Software | 31 |
| 3.7.1 | Python | 31 |
| 3.7.2 | Proteus | 32 |
| 3.7.3 | Blynk Application | 33 |
| 3.8 | List of Component | 34 |
| 3.8.1 | Raspberry Pi 3 Model B | 35 |
| 3.8.2 | DHT11 Temperature and Humidity Sensor | 37 |
| 3.8.3 | MQ2 Smoke Sensor | 39 |
| 3.8.4 | Buzzer & LED | 40 |
| 3.8.5 | MPC3008 Analog to Digital converter set up | 41 |
| 3.9 | Software Development | 42 |
| 3.9.1 | Configuration of Raspberry Pi | 42 |
| 3.9.2 | Installation of programming library in Raspberry Pi | 44 |
| 3.9.3 | HDMI Configuration | 49 |
| 3.9.4 | VNC (Virtual Network Computing) settings | 51 |
| 3.9.5 | Blynk App set up and configuration | 53 |
| 3.9.6 | Python software coding | 59 |

| | | |
|--------|-------------------------------------------------------------------------------|-----------|
| 3.10 | Hardware Development | 63 |
| 3.10.1 | DHT11 Module Installation | 63 |
| 3.10.2 | MQ-2 Module Installation | 64 |
| 3.10.3 | Buzzer and LED Installation Process | 64 |
| 3.10.4 | Analog to Digital Converter MCP3008 Installation | 65 |
| 3.11 | Cost Estimation | 67 |
| 3.12 | Chapter Summary | 67 |
| | CHAPTER 4 RESULT AND DISCUSSION | 68 |
| 4.1 | Introduction | 68 |
| 4.2 | Prototype Test | 69 |
| 4.3 | Effectiveness an alarm system with IoT to reduce time taken to warn the users | 71 |
| 4.4 | Fire breakout detection test | 74 |
| 4.5 | Actual Prototype | 77 |
| 4.6 | Chapter Summary | 79 |
| | CHAPTER 5 CONCLUSION AND RECOMMENDATION | 80 |
| 5.1 | Introduction | 80 |
| 5.2 | Conclusion | 80 |
| 5.3 | Recommendations | 81 |
| 5.4 | Chapter Summary | 81 |
| | REFERENCES | 82 |
| | APPENDIX A PYTHON CODING | 87 |

| | |
|------------------------------------------|-----------|
| APPENDIX B MQ-2 SENSOR DATASHEET | 91 |
| APPENDIX C DHT11 SENSOR DATASHEET | 92 |
| APPENDIX D MCP3008 ADC DATASHEET | 93 |

LIST OF TABLES

| | | |
|-----------|--------------------------------------------------------------|----|
| Table 2.1 | Comparison between Raspberry Pi and Arduino | 13 |
| Table 2.2 | Comparison for the journal | 20 |
| Table 3.1 | List of components used for this project | 34 |
| Table 3.2 | Code for setting HDMI port and configuration | 49 |
| Table 3.3 | Code for setting up aspect ratio for HDMI | 50 |
| Table 3.4 | Table of Cost Estimation | 67 |
| Table 4.1 | Result of time taken to warn the users | 71 |
| Table 4.2 | Result of time taken to warning user based on fire detection | 75 |

LIST OF FIGURES

| | | |
|-------------|------------------------------------------------------------|----|
| Figure 2.1 | Theory of IoT | 6 |
| Figure 2.2 | Architecture of IoT | 7 |
| Figure 2.3 | Fire Detection Device | 9 |
| Figure 2.4 | Raspberry Pi component | 11 |
| Figure 2.5 | Arduino Uno | 12 |
| Figure 2.6 | The block diagram of automatic fire alarm system | 15 |
| Figure 2.7 | Block diagram of overall system | 16 |
| Figure 2.8 | Flow chart of fire detection system | 17 |
| Figure 2.9 | Block diagram of Arduino based fire alarm | 18 |
| Figure 2.10 | Block diagram for Fire detection alarm system | 19 |
| Figure 3.1 | Gantt Chart of PSM 1 | 24 |
| Figure 3.2 | Gantt Chart of PSM 2 | 25 |
| Figure 3.3 | Block Diagram | 26 |
| Figure 3.4 | Flow chart | 27 |
| Figure 3.5 | Project schematic circuit diagram | 28 |
| Figure 3.6 | Product sketch main view | 30 |
| Figure 3.7 | Product sketch bottom view | 30 |
| Figure 3.8 | Python Software | 31 |
| Figure 3.9 | Proteus 8 Design suite software | 32 |
| Figure 3.10 | Blynk application | 33 |
| Figure 3.11 | Blynk interface for Ios and Android | 34 |
| Figure 3.12 | Raspberry Pi ports and header layout | 35 |
| Figure 3.13 | Raspberry Pi model comparison table | 36 |
| Figure 3.14 | Temperature and Humidity Sensor | 37 |
| Figure 3.15 | Schematic diagram for temperature sensor module connection | 38 |

| | | |
|-------------|----------------------------------------------------|----|
| Figure 3.16 | Smoke sensor | 39 |
| Figure 3.17 | Buzzer & LED | 40 |
| Figure 3.18 | Schematic Raspberry Pi | 40 |
| Figure 3.19 | MCP3008 ADC IC Pinout | 41 |
| Figure 3.20 | MQ-2 to ADC wiring diagram | 42 |
| Figure 3.21 | Raspberry Pi first time setup | 43 |
| Figure 3.22 | Raspberry Pi write image | 43 |
| Figure 3.23 | Raspberry Pi dialog box | 44 |
| Figure 3.24 | RPi.GPIO installation | 45 |
| Figure 3.25 | Coding of Blynk library installation | 45 |
| Figure 3.26 | Blynk Library installation | 46 |
| Figure 3.27 | Coding of MQ-2 sensor | 46 |
| Figure 3.28 | MQ-X library installation | 47 |
| Figure 3.29 | Coding of Adafruit Python DHT library installation | 47 |
| Figure 3.30 | DHT-11 library installation | 48 |
| Figure 3.31 | Coding of MCP3008 installation | 48 |
| Figure 3.32 | MCP3008 library installation | 49 |
| Figure 3.33 | VNC Viewer setup connection | 51 |
| Figure 3.34 | VNC Viewer on Raspberry Pi | 52 |
| Figure 3.35 | Log in sign up interface on Blynk application | 53 |
| Figure 3.36 | Log in Blynk application | 54 |
| Figure 3.37 | Blynk application create new project interface | 54 |
| Figure 3.38 | User get Auth Token in Blynk application | 55 |
| Figure 3.39 | Eventor setting interface | 56 |
| Figure 3.40 | Real time main display interface | 57 |
| Figure 3.41 | Value display gas settings | 58 |
| Figure 3.42 | Value display temperature settings | 59 |

| | | |
|-------------|-------------------------------------------------------------------|----|
| Figure 3.43 | The coding in Python | 60 |
| Figure 3.44 | The coding of setup GPIO in Python | 61 |
| Figure 3.45 | The coding of MCP3008 in Python | 61 |
| Figure 3.46 | The primary coding to execute in Python | 62 |
| Figure 3.47 | DHT11 installation process | 63 |
| Figure 3.48 | MQ-2 installation process | 64 |
| Figure 3.49 | Piezo buzzer & LED installation | 65 |
| Figure 3.50 | ADC MCP3008 IC installation | 66 |
| Figure 4.1 | Prototype testing in progress | 69 |
| Figure 4.2 | Successful test connection between | 70 |
| Figure 4.3 | Successful connection between Blynk server and generic board | 70 |
| Figure 4.4 | Imitating gas leakage scenario using pocket lighter (butane gas) | 72 |
| Figure 4.5 | Python real time gas reading and warning | 73 |
| Figure 4.6 | Detection on gas alert on the user smartphone | 74 |
| Figure 4.7 | Python temperature real time reading | 76 |
| Figure 4.8 | Blynk notification when sensor detected heat over threshold limit | 77 |
| Figure 4.9 | Top view of the final year project design | 78 |
| Figure 4.10 | Inside view of the project | 78 |
| Figure 4.11 | Front view of the project | 79 |

LIST OF ABBREVIATIONS

| | |
|------|----------------------------------|
| 2D | Two Dimensional |
| ADC | Analog-to-Digital |
| CO | Carbon Monoxide |
| CSI | Camera Serial Interface |
| DHT | Digital Temperature and Humidity |
| FRD | FIRE Rescue Department |
| GND | Ground |
| GPIO | General Purpose Input Output |
| GUI | Graphical User Interface |
| HSV | Hue Saturation Value |
| IoT | Internet of Things |
| LPG | Liquefied Petroleum Gas |
| NTC | Negative Temperature Coefficient |
| OTP | One Time Programmable |
| PPM | Particle Per Million |
| UI | User Interface |
| USB | Universal Serial Bus |
| WiFi | Wireless Fidelity |

CHAPTER 1

INTRODUCTION

1.1 Project Background

According to the Malaysia Fire and Rescue Department (BOMBA), 96.7 percent of reported fires in 2019 were caused by human error (*Majority of Structural Fires Can Be Avoided* / *The Edge Markets*, n.d.). Fire is an unwelcome tragedy that can result in death and widespread devastation of people and nations. Many products such as smoke detector, fire alarm and automatic sprinkler are designed to prevent the spread of fire. Sensor based technology has become more popular as technology has improved and construction costs have decreased. The Raspberry Pi, a low cost single board credit card sized computer, has made it possible to build a range of automated and monitoring devices with low power consumption and higher processing capabilities at a lower cost (Nayyar & Puri, 2015). The proposed fire alarm system in this report combines low-cost equipment, networking, and wireless communication.

The early identification of a developing fire incident, as well as the alerting of the house's occupants and fire rescue organizations, is a vital component of fire safety. This is where fire detection and warning devices come in. To begin, the occupant must be warned by a fire alarm fitted with the system as soon as possible in order to detect the presence of fire, smoke or heat in the house. The Blynk app, which was installed to reach users through the Internet, will also alert occupants via their smartphone. Another standard function is the transmitting of an alarm warning signal to the fire department or another emergency response agency. Electrical, air building or special process processes can all be turned off and automatic suppression systems can be activated as a result (Reddy et al., 2020).

The fire detection system comes in a variety of shapes and sizes to help prevent fires from forming. A safe person can be a powerful fire alarm similar to those used in the past that detects flame, smoke and the presence of fire. Humans on the other hand could be less powerful

because they are not all present when a fire begins. Automatic detectors are created to mimic one or more of the human senses of touch, smell, and vision. Thermal alarms are similar to our ability to detect high temperatures, and smoke detectors are similar to our ability to detect low temperatures are similar to our sense of smell and flame detectors are electronic eyes. A properly chosen and automatic detectors were placed can serve as a highly accurate fire sensor.

1.2 Problem Statement

There are some questions concerning the development building fire alarm system. This argument is backed up by statistic released by the Malaysian Fire and Rescue Department (FRD) which show that 2400 houses are burned each year and 120-150 people are killed in wildfires (*GLOBAL FOREST FIRE ASSESSMENT 1990-2000 - FRA WP 55*, n.d.). Such figures are alarming and campaigning alone was insufficient. The problems mentioned above are common when a fire breaks out in a residential building.

- I. The presence of a fire in the building was not immediately announced by the occupant, decreasing the amount of time available to respond.
- II. Putting the victim's life in jeopardy because it takes longer to contact the emergency team and vigilantes in the area to arrive on the scene.
- III. The existing fire alarm system on the market has a high selling price as well as a high maintenance cost due to its sophistication.
- IV. When the house is unattended, the fire alarm is insufficient to warn the building occupants. If there is a fire in the building, people will be alerted via their mobile phone.

1.3 Objectives

The research objectives of this study were as follows:

- I. To create a device that can detect the presence of a fire in a building and warn occupants and users using an IoT service.
- II. To create a framework for a building fire protection device using Raspberry Pi and IoT for infinite reachability.
- III. The use of Raspberry Pi microcontroller to process the data obtained from input sensors and implementation of alert system using Blynk application to notify user.
- IV. To see how effective an alarm system with IoT is at reducing the time it takes for the system to warn the occupant.

1.4 Significance

According to previous studies, there are many types of fire detection systems used in both residential and commercial buildings. In comparison to the writer project, these methods are essential. I did some residential and commercial buildings. In comparison to the writer project, these methods are essential. I did some research for the previous journal and research as a blogger.

The study on Suparman, M, & Siat, L. (2019) (Suparman & Jong, 2019). The Automatic Favoriot platform smoke alert system using IoT demonstrates that the project follows the same idea as the previous one in that it alerts users via IoT but on different platforms. The previous project used an Arduino Uno as the microcontroller (Suparman & Jong, 2019), while the author used a Raspberry Pi to speed up the processing. Futhermore, the writer employs Blynk's alarm request, which is both simple and effective in alerting users of the building compound to the existence of a fire.

1.5 Scopes and Limitations of the Study

The aim of this project is to show how important it is to take immediate action to alert building residents and emergency personnel to the existence of a fire in the building using a fire detection system that includes an alarm and IoT software. The project can be expanded to include new residential developments and complexes. In contrast, with a system equipped with both alarm and mobile usage alerts, the chances of customers and residents getting a fire alert are strong even if the buildings are unoccupied. The Raspberry Pi 3 model 3 microprocessor is used in this project to track the unwanted fire presence from afar. Users and fire services should be notified as soon as possible to prevent the fire from spreading. Furthermore, researchers can easily upgrade the system with Artificial Intelligence (AI) and the Internet of Things (IoT) for better future development of the Raspberry Pi. The project also required users to download and install the Blynk software on their smartphones in order to communicate with the Raspberry Pi and receive notifications and alert about the presence of fire in the building.

For each project design, the researcher or developer must face constraints. To begin with, this device lacks an automated water sprinkler to extinguish and prevent a fire from spreading. Before the consumer is informed of a fire in the house, the sensor can detect the fire and transmit the signal to the microprocessor. To monitor the fire's performance, users who are alerted to the presence of fire and smoke must manually douse the blaze. Then, using

the Blynk application warning, this device equipped with a Wifi module built into the Raspberry Pi will transmit the notification to the user and occupant. The real stumbling block is that each building and home must have its own WiFi network in order to link the module to the server. Users who live in a remote area may experience problems with area communication signal and warnings may be delayed.

1.6 Chapter Summary

This chapter covers the project's background, problem statement, goals, key project, scope, and project limitations. The subject of the project is a fire alarm detection device. The device will be installed inside residential buildings to detect fires and gas leaks in order to prevent house fires and saved lives and property.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The aim of the journal review is to provide an in-depth analysis of a similar subject to this project that was completed by a previous researcher. Journals, books and articles are used as a source of information. This development also involved the operation of circuits, software and hardware. This chapter also gives insight into the project's device or hardware components. Based on the tools that have been applied, this project can be composed from the previous project to increase the efficiency and operability of this project.

2.2 Internet of Things (IoT)

The Internet of Things (IoT) is a new IT paradigm shift. The two terms are combined to form the expression "Internet of Things" also known as IoT. "Internet" is the first title and "Things" is the second (Madakam et al., 2015). The Internet is a worldwide network of interconnected computer networks that serves billions of users around the world using the standard Internet protocol suite (TCP/IP). It is a networks made up tens of millions of people in the private, public, and academic sectors, corporate and government networks ranging in size from local to global and electronic, wireless, and optical networking technologies are used to link them. (Madakam et al., 2015).



Figure 2.1 Theory of IoT (H. et al., 2015)

2.2.1 IoT Vision

The Internet of Things (IoT) is a concept and a philosophy that takes into account a variety of things pervasive presence in the environment can communicate cooperating with each other and other stuff develop new applications/services and achieve a shared target using wireless and wired connections and complex addressing schemes. A world in which physical, digital, and virtual worlds intersect to create smart communities that boost energy, transportation, cities, and a number of other fields. The Internet of Things' goal is to enable everything and anyone to connect with each other at anytime, anywhere, and over any path/network and service. (Burange & Misalkar, 2015).

2.2.2 IoT History

Laurence G Roberts and Thomas Merrill bind two of the first supercomputers in Massachusetts and California in 1965, becoming the first wide-area network in the world (WAN). ARPANET sends its first message in 1969, taking the world's first working packet switching network using TCP/IP online and laying the groundwork for how the Internet works today. Mario Cardullo, an Italian inventor, received the first patent for a passive RFID tag in 1973. The first Internet-connected machine was a Coke vending machine that was rigged by Carnegie Mellon University Computer Science graduate students to tell them whether or not the machine was stocked with cold soda in 1982. John Romkey invents the first Internet-controlled appliance, a toaster that can be switched on and off through the Internet, in 1990.

Mark Weiser's article "The Computer in the Twenty-First Century" appears in Scientific American in 1991. In 1999, British tech visionary Kevin Ashton (now at Belkin, behind WeMo) coined the word "Internet of Things" as the title of a presentation given to P&G (Procter & Gamble) about supply chain innovation. For the first time in 2007, the number of connected computers outnumbers the world's human population. The first M2M and Internet of Things Global Summit was held in Washington, DC in 2013. In 2014, NEST laboratories, a company that "designs and manufactures sensor-driven, Wi-Fi-enabled, self-learning, programmable thermostats and smoke detectors," was purchased by Google for \$3.2 billion, signalling the Internet of Things' transition into the mainstream market spotlight, mainly through smart home devices (H. et al., 2015).

2.2.3 IoT Architecture

IoT architecture is made up of a variety of technologies that work together to sustain it. Its aim is to show how different technologies interact with one another as well as to explain the scalability, modularity, and configuration of IoT deployments in future options.

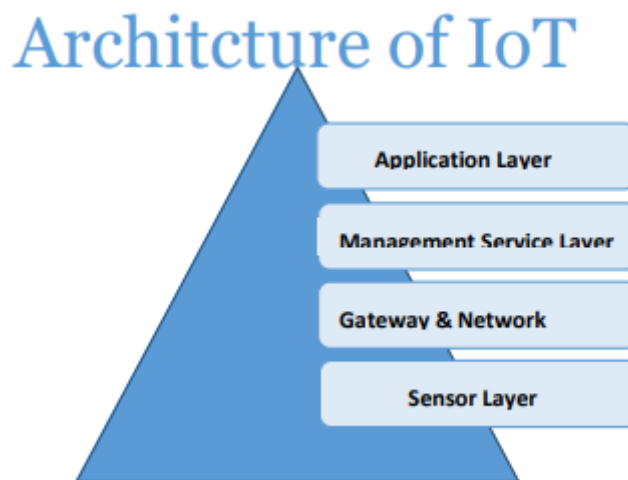


Figure 2.2 Architecture of IoT (Madakam et al., 2015)

The lowest layer, the Sensor Layer, is made up of smart objects that have sensors built in. This layer's primary goal is to collect and share different types of static and dynamic data from the real world using various types of sensors and Internet connectivity. Effective sensors can now be rendered in much smaller packages and built into physical objects due to hardware miniaturization. Sensors come in a variety of shapes and sizes, and they are used for a variety of purposes. Temperature, air quality, distance, movement, and electricity are just some of the

measurements that the sensors can make. They may still have some memory, enabling them to take a limited number of measurements under specific circumstances. A sensor can recognize a physical property and translate it into a signal that a device can identify.

Networks and Gateways. These sensors can generate a large amount of data, which necessitates the use of a reliable and high-performance wired or wireless network infrastructure as a transport medium. The network aids in the identification of each entity in the physical world that is linked. Current networks, which are often related to very different protocols, have embraced machine-to-machine (M2M) networks and their implementations. It is getting more difficult to keep up with the demand for a wider range of IoT services and applications, such as high-speed transactional services, context-aware applications, and so on. To work together, multiple networks with different technologies and access protocols are needed in a diverse configuration. These networks are built to meet communication requirements such as latency, bandwidth, security and needs may be private, public, or hybrid in nature.

Data processing is enabled by the Management Service Layer, which includes analytics, security controls, process modeling, and system management. The product and system rule engines were among the most important parts of the management service layer. IoT connects and interacts with objects and systems to provide information in the form of events or contextual data, such as temperature of products, fuel consumption, current location, and traffic information. Some of these tasks, such as collecting periodic sensory data, require filtering or routing to post-processing systems, while others, such as responding to medical emergencies require immediate responses. The ability to control data knowledge flow is known as data processing. Data processing in the management service layer can be used to view, implement, and manage information.

The Application Layer, which is at the top of the stack, is in charge of delivering various applications to various users in the IoT. It is made up of protocols that concentrate on communication between processes over an IP network, as well as a reliable end-user services and networking interface (Gubbi et al., 2013; Sethi & Sarangi, 2017; Weyrich & Ebert, 2016; Wu et al., 2010).

2.3 Fire Detection Devices

Manually actuated and automatically actuated systems are the two most common forms of fire detection devices.

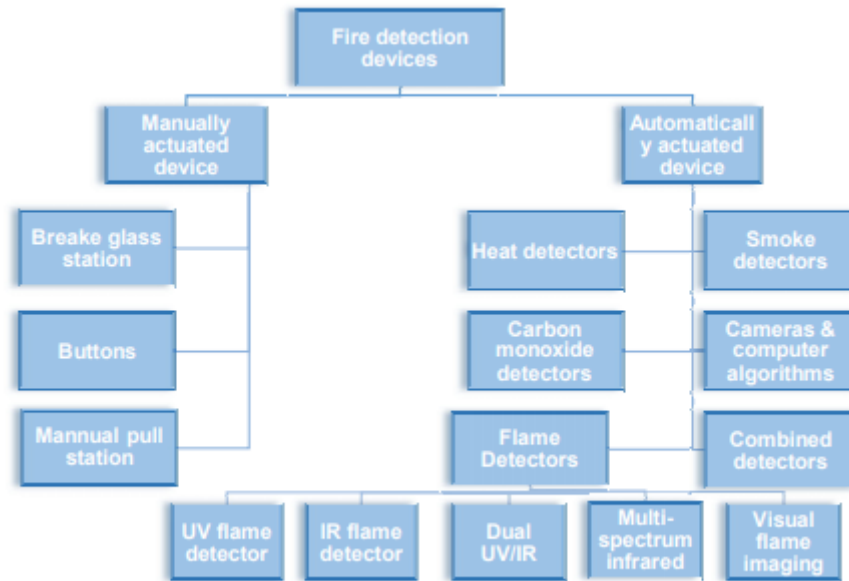


Figure 2.3 Fire Detection Device (Reddy et al., 2020)

Manually operated devices are often found near exits and resemble the red button on the wall. In the event of a burn, someone should press the trigger. If someone pushes the button after seeing the blaze, however, it will be too late (Petrov, 2012).

Modern fire alarms with automatic activation are used to detect fire as soon as possible. They were designed to work nonstop for 24 hours a day, seven days a week. Flame sensors, heat sensors, smoke sensors carbon monoxide sensors, and computer-controlled cameras are all forms of fire detection sensors. Typically, fire detection sensors can activate when a critical value is reached. There are three types of fire detectors: point, linear, and sampling. Point fire detectors collect data in the area where they are placed. The linear one is made up of components that resemble cameras. They are based on laser technology and are mounted in on the opposite walls of the space in front of each other. This form of detector can detect objects in areas up to 100 meters long. Sampling fire detectors are the kind of smoke detectors that can be installed in the ventilation system or elsewhere and take air samples at predetermined intervals (Petrov, 2012).

Heat detectors are temperature-sensitive devices. They can be either passive or aggressive in nature. Passive sensors do not consume energy, and when the temperature reaches a critical level, the sensor generates a particular signal (due to the thermoelectric effect) or breaks the alarm circuit's electrical circuit. Energy sources are used by active detectors. They

can not only warn you when the temperature reaches a critical level, but they can also sound a warning if the temperature rises too quickly.

Flame detectors use an inserted photo detector to detect infrared or ultraviolet radiation from a flame. Both smouldering and burning fires can be detected by them. They're used in places where a fire can start quickly without the presence of smoke. (Petrov, 2012).

Carbon monoxide (CO) detectors are an essential component of building fire safety systems. The biggest disadvantage of CO is that it has no odour. Portable generators, stoves, coal and wood burning all emit carbon monoxide. Since people cannot sense the presence of CO, it will kill you even though you are not sleeping. That is why carbon monoxide detectors are so critical. Metal-oxide-semiconductor (MOS), biometric, and electrochemical are the three basic groups. Since the first technology (MOS) is still plugged into the mains, there is no need to check batteries. The gel-coated disk on the biometric detector turns dark when CO is present. The alarm is activated by changing the color of the disk. Electrochemical detectors operate since a chemical reaction between CO and electricity produces an electrical current. CO detectors, for the most part, sound an alarm as CO levels rise rapidly. There are detectors that detect long-term shifts in CO concentration, but they are very costly. Many countries require that each household have at least one CO detector (Carlsen, n.d.).

Computer-controlled cameras are often used in lofty, voluminous areas or other locations where fire safety has historically been the most difficult. In tunnels, large stadiums, manufacturing parks, and other places, they are indispensable. Smoke and flames can be detected by cameras at great distances in seconds. They broadcast live video to the fire station and activate the alarm system. Computer algorithms aid in the detection of smoke and flame patterns, as well as the obliteration of protected events (Alamgir, 2020).

2.3.1 Principle of Fire Detector

Process engineers in the oil and gas industry, for example, are still searching for new ways to reduce the risk of toxic and harmful gases in their everyday operations. The danger of a fire accident is one of the most serious threats the industry faces. Proper flame detection should be installed to avoid catastrophic fires, and before choosing such devices, it's a good idea to study the concepts of flame detectors and the different types of detectors available on the market.

Optical methods such as ultraviolet (UV), infrared (IR) spectroscopy, and visual flame imaging are used by the majority of flame detectors. For example, flames in a refinery. For example, are typically fueled by hydrocarbons, which generate heat, carbon dioxide, and other combustion products when combined with oxygen and an ignition source. Infrared and ultraviolet light are emitted during combustion, and flame detectors work by detecting UV and IR light at particular wavelengths (*How UV, IR and Imaging Detectors Work*, 2020).

2.4 Raspberry Pi

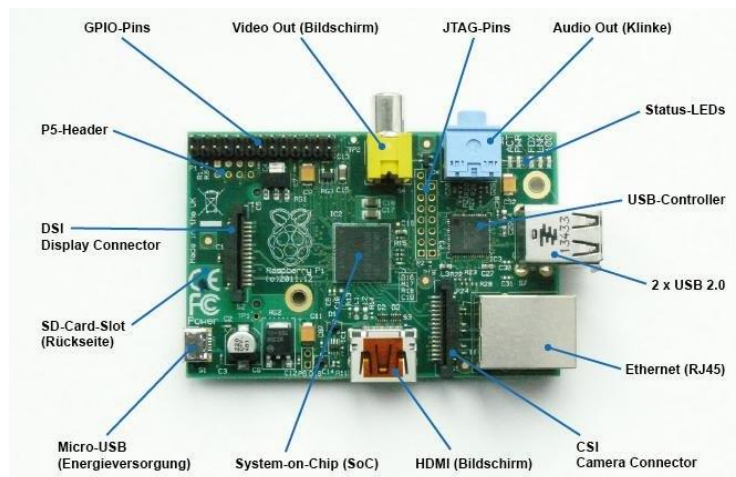


Figure 2.4 Raspberry Pi component (Nayyar & Puri, 2015)

The Raspberry Pi is a single-board device the size of a credit card developed by the Raspberry Pi Foundation in the United Kingdom. The board is a miniature marvel with extreme computing capability and the ability to create incredible projects. The machine ranges in price from \$5 to \$35 and is ideal for performing a wide range of computing tasks as well as interfacing various devices via GPIO (General Pin Input Output). The A Broadcom ARM (Advanced RISC Machines) processor, a graphics chip, RAM, GPIO, and other external device connectors are all included on the Raspberry Pi board. The Raspberry Pi's operating system is similar to that of a PC, with the exception that it requires additional hardware such as a keyboard, mouse, display unit, power supply, and SD Card with OS mounted (acting as a hard disk) to function. USB ports and Ethernet for Internet/Network-Peer-to-Peer connectivity are also available on the Raspberry Pi. As in every other device, the operating system serves as the foundation for its service. Raspberry Pi is a computer that runs Linux-based open source operating systems. More than 30 operating systems based on various flavors of Linux have been released to date. The Raspberry Pi Foundation has also released a number of accessories,

such as the Camera, Gertboard, and Compute Model Kit, that can be used to deploy add-on hardware modules (Nayyar & Puri, 2015).

2.5 Comparison Raspberry Pi with Arduino

Main difference between them is Raspberry Pi basically is a single-board computer while Arduino is based on the ATmega family. Both of them have a CPU which executes the instruction, timers, memory and input output pins. The comparison between Raspberry Pi and Arduino will be explain more about the difference and similarities between Raspberry Pi and Arduino.

2.5.1 Arduino



Figure 2.5 Arduino Uno (Noor et al., 2018)

Arduino is a simple-to-use prototyping platform based on open source hardware and software. It's a basic microcontroller board. It consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software or an IDE (Integrated Development Environment) that runs on the computer, used to write and upload computer code to the physical board (*What Is Arduino?* / *Arduino*, n.d.). Arduino can read inputs from different sensors, such as light on a sensor or a finger on a button, and convert them to outputs, such as triggering a motor or turning on a light. By sending a series of commands/instructions to the board's microcontroller, it can tell it what to do. Arduino can support up to two working nodes, each of which can be connected to a computer via USB or run independently (Patil, 2016).

2.5.2 Comparison between Raspberry Pi and Arduino

Table 2.1 Comparison between Raspberry Pi and Arduino (Noor et al., 2018)

| Specification | Raspberry Pi | Arduino |
|-------------------------------|----------------------------------------------------------------------|-----------------------------------------------|
| Processor | ARM BCM2835 | ATMEGA8, ATMEGA168 ATMEGA328ATMEGA1 280 |
| RAM | 256-512MB | 16-32KB |
| Power | 5V/USB | 7-12/USB |
| Board operating system | Raspbian, Ubuntu, Android, ArchLinux, FreeBSD, Fedora, RISC OS | / |
| Programming language | C, C++, Java Phyton | Arduino |
| LAN (Mbit) | 10/100 | - |
| WiFi Module | - | - |

Low power consumption is the key target in order to fulfill multilayer application requirements. The Raspberry Pi hardware's CPU is primarily responsible for processing information from computer programs. Raspberry Pi uses an Arm-based BCM2835 processor, which is inexpensive and efficient while consuming relatively little power. As a result, Raspberry Pi can run on 5v 1A power from the on board micro-usb port.

As a command, different platforms use various operating systems and programming languages. As seen in the table, the Raspberry Pi is compatible with a variety of operating systems. Raspberry Pi is best suited to the Linux version. Raspbian is a Linux distribution (*FrontPage - Raspbian*, n.d.). The linux version is most commonly used because it is an open source platform that is very inexpensive. It performs admirably on the Raspberry Pi's ARM processor.

The key gateway for communicating with other devices is the Ethernet port. Through the use of the Ethernet port, it can be directly connected to the router or another system. The Raspberry Pi A does not have a standard Rj45 ethernet socket, but the Raspberry Pi B does. Despite the fact that the Model A lacks with a USB Ethernet Adapter, it can be linked to a

wired network via a standard RJ45 ethernet port. There are two speed modes on the USB Ethernet Adapter: 10mb/s and 100mb/s (Upton, 2012).

2.5.3 Advantages of Raspberry Pi

Raspberry Pi advantages can be summarized as Raspberry Pi is a lightweight, versatile, and efficient cum compact form factor computer that is also very inexpensive to purchase. Raspberry Pi can be used by a variety of small and medium-sized businesses to perform tasks such as web server, database server, and media server. As a result, a significant amount of money can be saved on the procurement of different servers (Nayyar & Puri, 2015). Raspberry Pi can be used as a single board for a wide range of programming tasks. Pi supports a variety of programming languages, and users should install the appropriate compiler to ensure proper code execution. Python, the main programming language used by Pi, is a simple and easy-to-learn language. It allows for more efficient code creation, fewer lines of code, and automated memory management. Since the software is open source, it supports open source operating systems and applications. As a result, Raspberry Pi has access to a large number of operating systems in different version of Linux, as well as millions of applications for that operating system. Raspberry Pi also supports add-on hardware such as the Camera, Component Moduler Kit, Gertboard, and HAT board, allowing users to attach thousands of third-party devices such as buttons and LEDs to perform different tasks on the Pi (Maksimović et al., 2014). The product is energy efficient and offers small businesses a more environmentally friendly and ethical choice. This credit card-sized item is allowing to reuse and it helps to save a lot of money on cooling systems.

2.5.4 Disadvantages of Raspberry Pi

Based on the above, it can be inferred that the Raspberry Pi is a great device and offers a lot of nice features for a small price, but it is also missing a few useful features. The main disadvantages of Raspberry Pi are it lacks a backup battery-powered real-time clock (RTC), but it can easily get around this by using a network time server, as most operating systems do automatically (Nayyar & Puri, 2015). It lacks a Basic Input Output System (BIOS). As a result, it still boots from an SD card and does not come with Bluetooth or WiFi. However, USB dongles can be used to add these features. Most Linux distributions are also picky about their hardware, so it is best to see if the flavor of Linux they are using supports the system in question. Raspberry Pi doesn't have an Analog to Digital converter built in. For AD conversion,

an external part must be used. Although it has a limited number of digital I/O, it can be supplemented with external logic modules (Maksimović et al., 2014).

2.6 Automatic smoke detection system with favoriot Platform using Iot

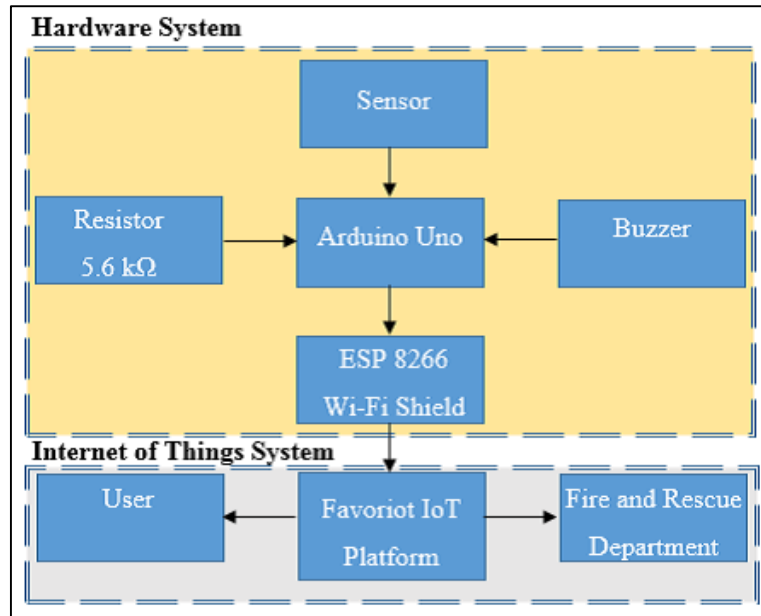


Figure 2.6 The block diagram of automatic fire alarm system (Suparman & Jong, 2019)

Previous work on the smoke detection device using the favoriot platform as the IoT by Suparman, M.A & Ling Jong, S. Through the Favoriot network, the research project is capable of not only monitoring the smoke condition of the room but also of alerting the client and the Fire and Rescue Department when a particular measure of smoke is detected by a gas sensor. The Arduino Uno is used to track both gadgets and the ESP8266 WiFi Shield serves as a mode for connecting gadgets to the network so that information about the smoke sensor can be shared on the Favoriot site. In this project experiment, the condition of the room is checked under various combustion materials and the amount of smoke is recorded. Smoke with a concentration of more than 100 parts per million has been related to swollen eyes, coughing and persistent coughing, both of which can lead to death. The hardware component for gas detection is the MQ-2 sensor. As a result, the best automatic smoke detection system threshold level is 80 ppm. The user can take early action to save lives and prevent a fire outbreak by using this project. (Suparman & Jong, 2019)

2.7 Intelligent fire detection and alert system using LabVIEW

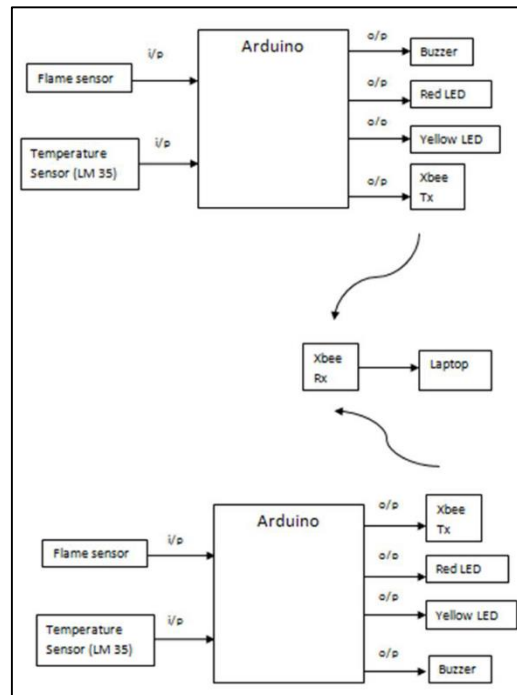


Figure 2.7 Block diagram of overall system (Idris et al., 2019)

To show the presence of fire, the project uses a flame and temperature sensor. The receiver and transmitter, both using wireless ZigBee technology are required for this project to function. The Arduino Uno is used as a microcontroller on the transmitter part to monitor warnings and sensor nodes when flame and temperature are detected. The data collected by the sensors is sent to the transmitter which acts as a router node through an XBee module. The data is collected for further processing on the receiver side by the XBee coordinator module which is connected to the device through USB for serial communication. It is still working on a Graphical User Interface (GUI) that is both immersive and user-friendly. The LabVIEW program is used to create a Graphical User Interface (GUI) that displays and evaluates the risk of a fire. The device will show the location of the fire and provide early warning in order to enable people to safety evacuate the building. (Idris et al., 2019)

2.8 Fire Detection System using Raspberry Pi

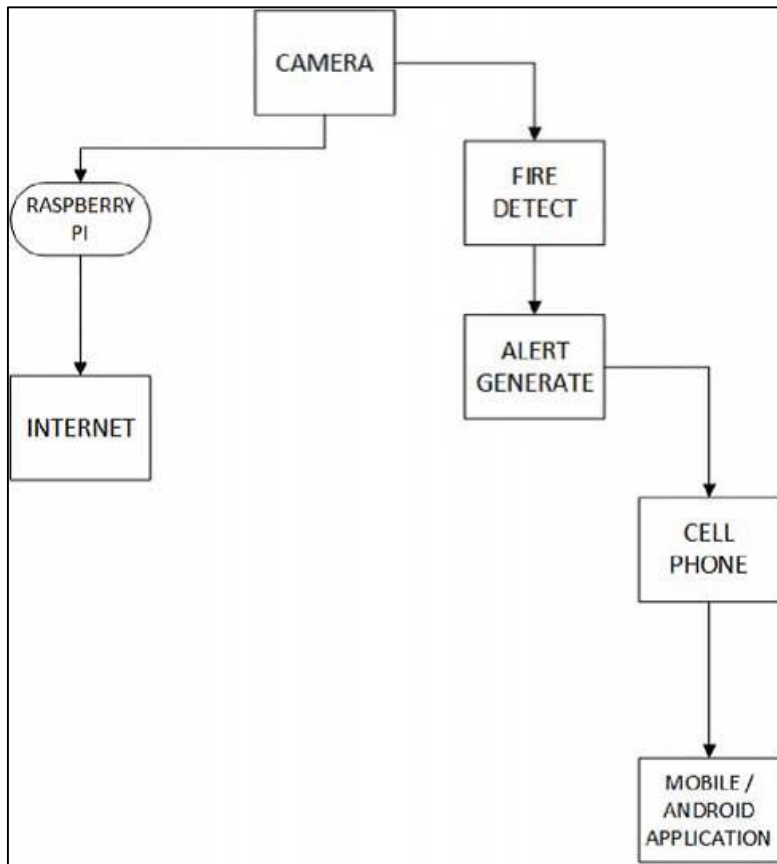


Figure 2.8 Flow chart of fire detection system (Khan et al., 2019)

The current model's new vision-based device is designed to work with the Raspberry Pi and detect a visible portion of the fire in remote areas. Heat signatures are used by the Raspberry Pi controller to process camera input and detect flames. The report is generated automatically by the image processing unit and sent to the person immediately after the fire is detected via Wi-Fi. The computer's emergency mode will be activated as a result of this action. The Android app generates an immediate alert. Simple spectrum factors of red, green and blue are mixed in the RGB model to respond to the fire detection system. The colour model is based on the Cartesian coordinate system. The camera captures the image and the surrounding video is converted from RGB to XYZ colour space. Since it is converted to HSV (Hue, Saturation Value) on a very high scale in the RGB model, colour conversion is very important in the fire detection system because it allows fire detection on a very low to high scale. The HSV colour combination is used for the proposed model and variations in flame colour and textured are examined. (Khan et al., 2019)

2.9 Arduino based Fire Detection and Alarm System Using Smoke Sensor

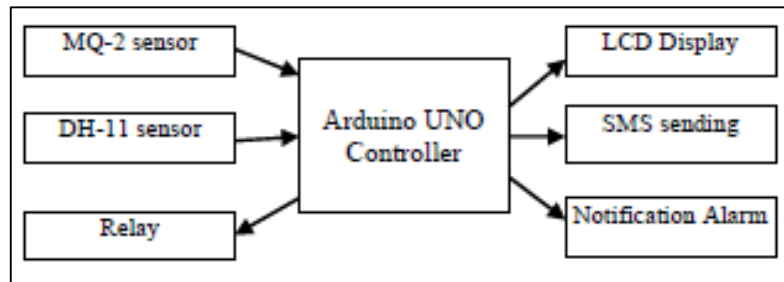


Figure 2.9 Block diagram of Arduino based fire alarm (Tun & Myint, 2020)

The fire alarm system is made up of smoke detectors, heat detectors and infrared detectors as well as a control unit and an alarm system. The unit for fire detection is based on temperature and smoke measurements taken at the same time. The alarm fire monitoring system reported fires that were not detected by smoke sensors and were detected for shorter periods of time than when smoke sensors were used alone. Sensor data from smoke, combustion and temperature were used in previous fire detection algorithms. The smoke sensor warns when the analogue output signal reaches or exceeds the threshold value. Analogue smoke, carbon monoxide (CO) and temperature sensors are included in the node. The fire alarm system should notify building occupants of the presence of fire signs such as smoke or high temperatures in a reliable and timely manner (Tun & Myint, 2020).

2.10 Redundant for Fire Monitoring System Using Raspberry Pi and Arduino Uno

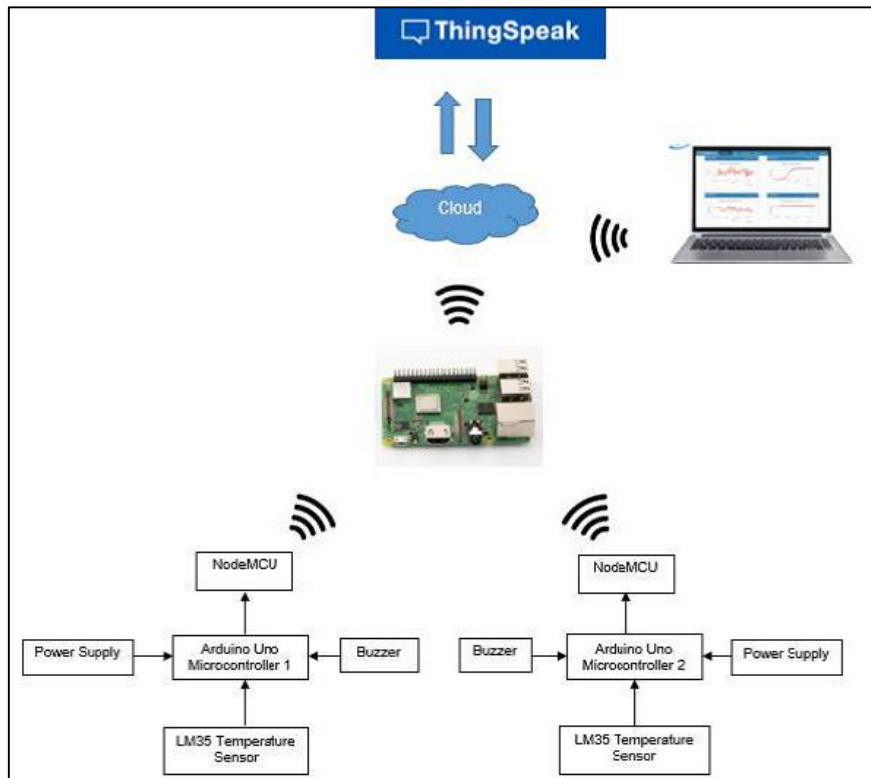


Figure 2.10 Block diagram for Fire detection alarm system (Sazali, 2019)

A research project involving two LM35 temperature sensors has been used to develop a redundancy principle to enhance the feedback functionality of both temperature sensors to be tested inside the space. At regular intervals, the obtained values are automatically uploaded to the Cloud by Raspberry Pi through the NodeMCU Wi-Fi module. The Thingspeak framework is used for tracking and the user can see the uploaded values in a graphical format. To install all of the programs that provide instructions for the proper operation of this computer, this project used Arduino Uno, NodeMCU and Raspberry Pi. Finally, the buzzer sensor's job in this alarm when the temperature rises above the set point. When the temperature sensor detects a reading inside the room, the data is uploaded to the cloud and can be displayed graphically from both sensors on the Thingspeak network. As the temperature rises above 30 ° C, the buzzer sounds a warning to alert the user to search for signs of a fire and to take precautions to avoid a fire (Sazali, 2019).

2.11 Comparison between different approached study

Table 2.2 Comparison for the journal

| | | | | | |
|------------------|--------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| The study | Mohd Alif b. Suparman, Siat Ling Jong (2019) | Fakruradzi Idris, Norlezhah Hashim, Ahmad Fauzan Kadmin, Lee Boon Yee (2018) | Muhammad Niman Aqeel Khan, Talha Tanveer, Kiran Khurshid (2019) | May Zaw Tun, Htay Myint. (2020) | N. Lyana M. Sazali (2019) |
| Title | “Automatic smoke detection system with favoriot platform using IoT” (Suparman & Jong, 2019) | “Intelligent fire detection and alert system using LabView” (Idris et al., 2019) | “Fire Detection System using Raspberry Pi” (Khan et al., 2019) | “Arduino based Fire Detection and Alarm System Using Smoke Sensor” (Tun & Myint, 2020) | “Redundant for Fire Monitoring System Using Raspberry Pi and Arduino Uno” (Sazali, 2019) |
| Method | The data from the sensor was sent to the Arduino Uno which was then transferred to Favoriot IoT (Suparman & Jong, 2019). | The data from the temperature and fire sensors is sent to the ZigBee network coordinator (Idris et al., 2019). | Raspberry Pi with camera system (Khan et al., 2019). | The Arduino Uno's sensors sensed fires which were detected by the alarm algorithm (Tun & Myint, 2020). | The Arduino Uno, NodeMCU, and Raspberry Pi were used in this project, which was then uploaded to |

| | | | | | |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | | | | Thingspeak (Sazali, 2019). |
| Results | The customer and fire fighter will be informed and alerted if the ppm smoke value surpassed the Arduino's threshold value. (Suparman & Jong, 2019) | If the location of a fire is identified, the emergency responders will act rapidly (Idris et al., 2019). | The camera will take a picture of a fire and issue a warning to the consumer (Khan et al., 2019) | The alarm will be activated when the analog output signal reaches or equals the threshold value (Tun & Myint, 2020). | Temperature data that has been transferred to the server is being read. The buzzer will sound if the temperature rises above the set point (Sazali, 2019). |

2.12 Chapter Summary

The project's past, previous work and current work have all been collected in this chapter. The journals in this chapter were used as a guide for the project and they all included similar fire alarm systems and sensors to measure relevant parameters such as temperature, different gas types and humidity of the environment. Even though the data processing method is different such as Arduino, Zigbee and LabView, the implemented concept operation can be used as a reference to improve the creation of IOT applications in the fire alarm system to solve fire outbreaks in residential buildings.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The study approach for the current day is detailed in this chapter. It explains in deep about the objectives, project scope and how to accomplish the desired results. The block Diagram, the flow map, the model, the simulation program, the layout of the design, the list of parts, the measurement of the hardware and infrastructure costs involve and finally the Gantt chart will be covered in this chapter. The rationale for each of the chosen components and hardware will be briefly explained as well as how the data from the previous experiment will be collected. The running software as if Python and Blynk Application were studied in every aspect. Configuration of Raspberry Pi and installation of programming library in Raspberry Pi also has been explained in this chapter for detail.

3.2 Project Management

Project management that has been used for this project is Gantt chart. Gantt chart below show the timeline for PSM 1 and PSM 2 which help to plan work around deadlines and properly allocate resources to achieve the objective for this project. The researcher set the timeline for the each task which is read research paper, prepare a thesis, software and hardware implementation and presentation preparation.

3.2.1 Gantt Chart of PSM 1

| NO. | DESCRIPTION | 2021 | | | | | | | | | | | | | | | |
|------------------------------|--------------------------------------|----------------------|---------------------------------------|--------|--------|--------|--------|------------------------|----------------|--------|--------|--------|--------|---------------------------|--------|-------------------------------|--------|
| | | MARCH | | | | APRIL | | | | MAY | | | | JUNE | | | |
| | | WEEK 1 | WEEK 2 | WEEK 3 | WEEK 4 | WEEK 1 | WEEK 2 | WEEK 3 | WEEK 4 | WEEK 1 | WEEK 2 | WEEK 3 | WEEK 4 | WEEK 1 | WEEK 2 | WEEK 3 | WEEK 4 |
| | SCHEDULE FYP | FYP Briefing Session | Selection of FYP Title and Supervisor | | | | | 1st General Evaluation | Mid Term Break | | | | | FYP Progress Presentation | | Report and Logbook submission | |
| 1.0 LITERATURE REVIEW | | | | | | | | | | | | | | | | | |
| 1.1 | Research for Paper/Journal | | | | | | | | | | | | | | | | |
| 1.2 | LogBook Process | | | | | | | | | | | | | | | | |
| 2.0 ACTIVITY WITH SUPERVISOR | | | | | | | | | | | | | | | | | |
| 2.1 | Choosing Topic | | | | | | | | | | | | | | | | |
| 2.2 | First Draft of Project Chosen | | | | | | | | | | | | | | | | |
| 2.3 | Presenting the Process to Supervisor | | | | | | | | | | | | | | | | |
| 2.4 | Preparation for Presentation | | | | | | | | | | | | | | | | |
| 3.0 SKETCHING AND DESIGNING | | | | | | | | | | | | | | | | | |
| 3.1 | Listing of Components | | | | | | | | | | | | | | | | |
| 3.2 | Sketching Block Diagram | | | | | | | | | | | | | | | | |
| 3.3 | Flowchart | | | | | | | | | | | | | | | | |

Figure 3.1 Gantt Chart of PSM 1

3.2.2 Gantt Chart of PSM 2

| NO | DESCRIPTION | 2021 | | | | | | | | | | | | | |
|-------------------------------|-------------------------------|---------|--------|--------|----------|--------|--------|--------|----------|--------|---------|---------|---------|---------|---------|
| | | OCTOBER | | | NOVEMBER | | | | DECEMBER | | | | JANUARY | | |
| | | WEEK 1 | WEEK 2 | WEEK 3 | WEEK 4 | WEEK 5 | WEEK 6 | WEEK 7 | WEEK 8 | WEEK 9 | WEEK 10 | WEEK 11 | WEEK 12 | WEEK 13 | WEEK 14 |
| SCHEDULE FYP | | | | | | | | | | | | | | | |
| 1.0 PLANNING & REVIEW | | | | | | | | | | | | | | | |
| 1.1 | Project Scheduling | | | | | | | | | | | | | | |
| 1.2 | Review of the journal | | | | | | | | | | | | | | |
| 2.0 COMPONENT | | | | | | | | | | | | | | | |
| 2.1 | Survey Component | | | | | | | | | | | | | | |
| 2.2 | Estimation of budget | | | | | | | | | | | | | | |
| 2.3 | Purchase component | | | | | | | | | | | | | | |
| 3.0 SOFTWARE IMPLEMENTATION | | | | | | | | | | | | | | | |
| 3.1 | Circuit design | | | | | | | | | | | | | | |
| 3.1 | Write coding | | | | | | | | | | | | | | |
| 3.3 | Configuration Raspberry Pi | | | | | | | | | | | | | | |
| 4.0 HARDWARE IMPLEMENTATION | | | | | | | | | | | | | | | |
| 4.1 | Hardware Design | | | | | | | | | | | | | | |
| 4.2 | Hardware Implementation | | | | | | | | | | | | | | |
| 4.3 | Testing and troubleshooting | | | | | | | | | | | | | | |
| 5.0 THESIS & DOCUMENTATION | | | | | | | | | | | | | | | |
| 5.1 | Thesis Draft Chapter 1 - 5 | | | | | | | | | | | | | | |
| 5.2 | Slide Preparation | | | | | | | | | | | | | | |
| 5.3 | Research Paper | | | | | | | | | | | | | | |
| 6.0 PRESENTATION & SUBMISSION | | | | | | | | | | | | | | | |
| 6.1 | Project Presentation | | | | | | | | | | | | | | |
| 6.2 | Submission Thesis and Logbook | | | | | | | | | | | | | | |

Figure 3.2 Gantt Chart of PSM 2

3.3 Block Diagram

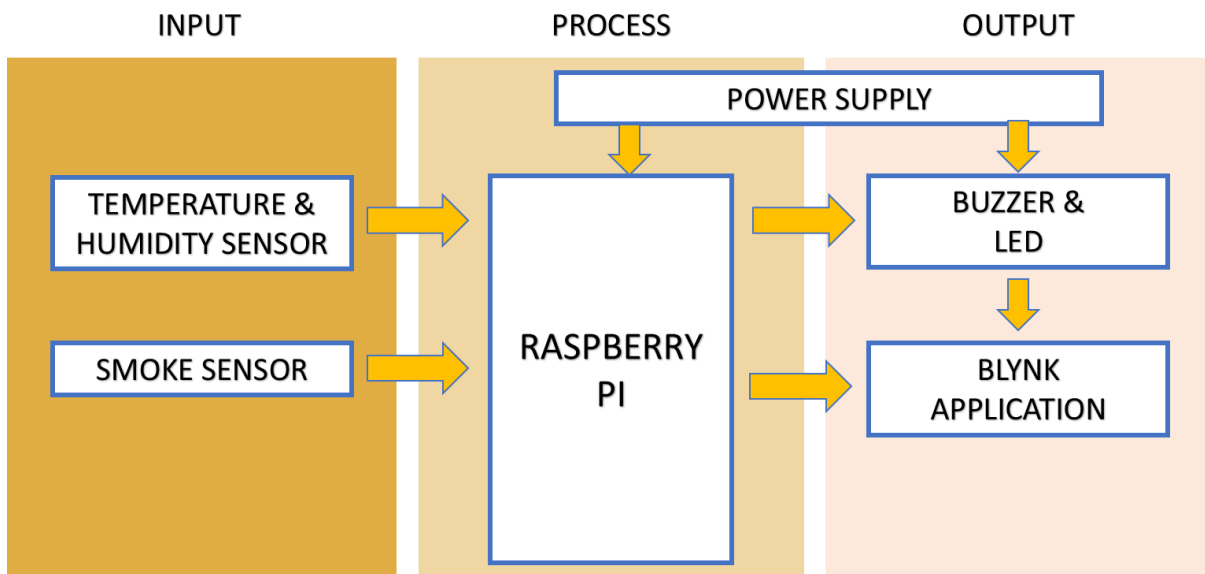


Figure 3.3 Block Diagram

A block diagram is a visual illustration of how a project will operate. Readers will get a general idea of how the workflow works from input to procedure to output. The temperature sensor and the smoke sensor are the only types of sensors used in this project's input block for hardware implementation. This sensor will act as the safety device's first layer of protection. The output from the sensor will be processed through the Raspberry Pi microprocessor to determine if both sensors exceed the particle per million (PPM) smoke concentration threshold. The output of the microprocessor is transmitted to the Blynk application for the output block on the user's smartphone. The program will work with the Raspberry Pi board to optimize the Wi-Fi module. In addition to the app warning, occupants will be notified of the existence of fire and smoke with a fitted alarm / buzzer device.

3.4 Flowchart

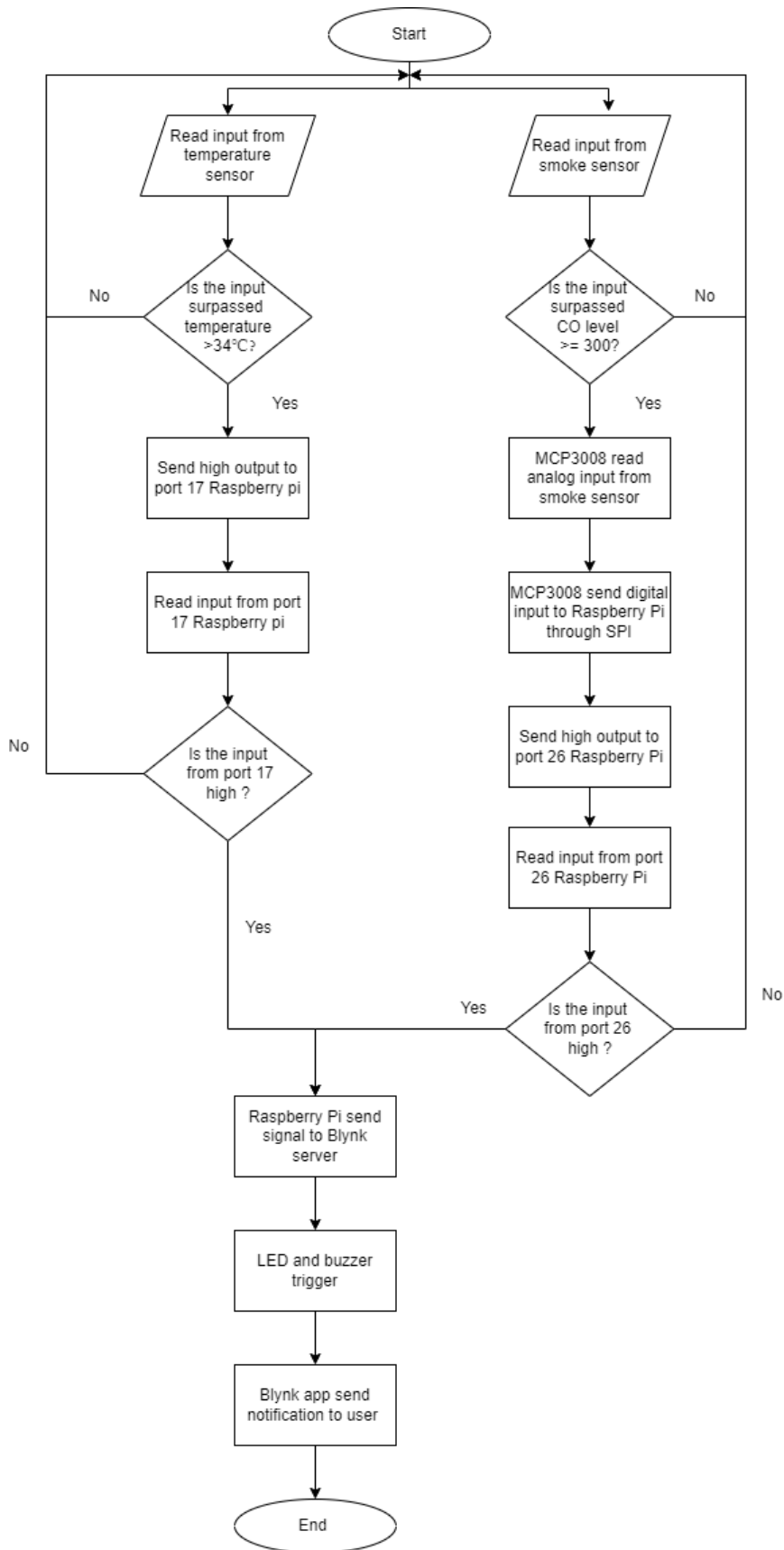


Figure 3.4 Flow chart

A flowchart is a visual representation of the steps and decisions that must be made in order to complete a procedure. Figure 2 depicts the entire project's series. The flow begins and the input flow is read by both sensors which is smoke and temperature. The sensor's input will be processed by a microprocessor in the decision stage which will classify the gas concentration and heat from the sensor. The flow will return to input flow if no smoke or fire is observed. If YES, or if the gas concentration and temperature reach the threshold, the process will proceed to the next flow which will activate the alarm/buzzer. When a fire or smoke is detected, a buzzer will sound to warn the building's occupants. In the next step of the procedure, the microprocessor will send a signal to an app on the user's smartphone, alerting the user or occupant to a fire or gas leak in the house. After the whole phase has been completed, the flow will come to an end.

3.5 Circuit Diagram

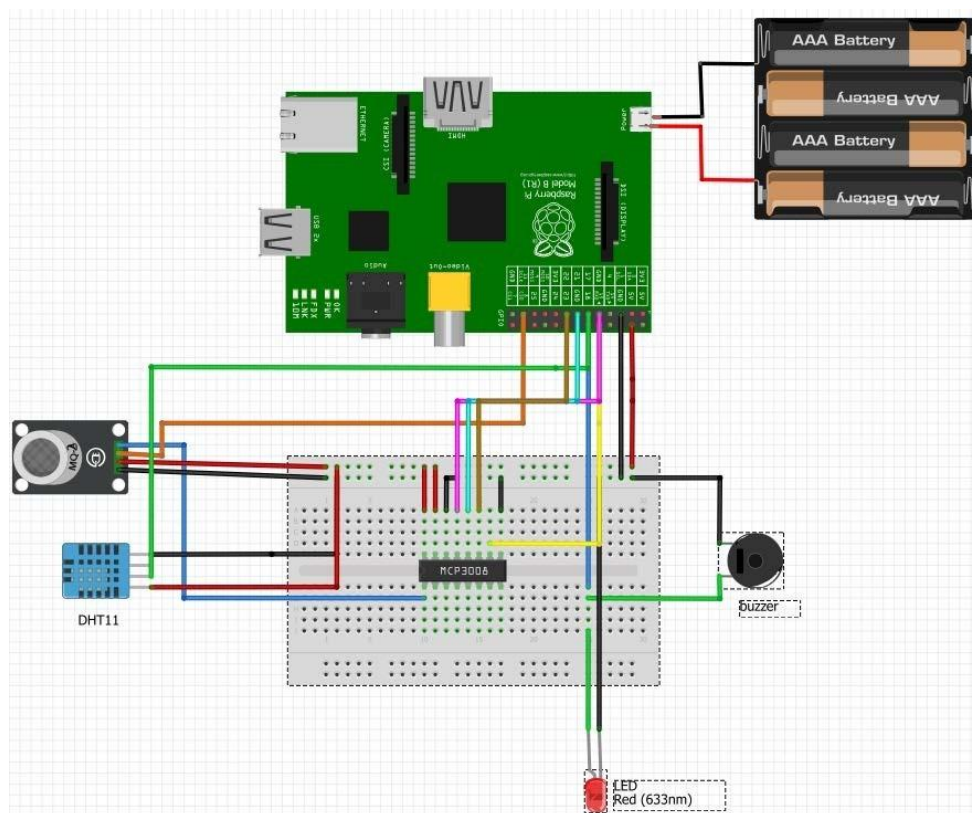


Figure 3.5 Project schematic circuit diagram

A schematic circuit diagram is a two-dimensional representation of the project's link configuration (2-D). A schematic is a diagram that shows something using symbols in a straightforward manner. A diagram is a representation of a system, apparatus or other object

made up of abstract, often uniform letters and rows. While some diagram details may be unnecessary or presented to help the model understand, schematic diagrams only show the significant parts of a scheme. Plan diagrams do not provide details that are not needed for the analysis of data intended for transmission.

The circuit layout for this project was created using the Fritzing program. A schematic is needed to identify the link between the Raspberry Pi module and the sensor as well as each module's port to another module. Before moving on to the next connection, the developer will double-check each of the connections using the schematic diagram.

3.6 Product Diagram/Sketch

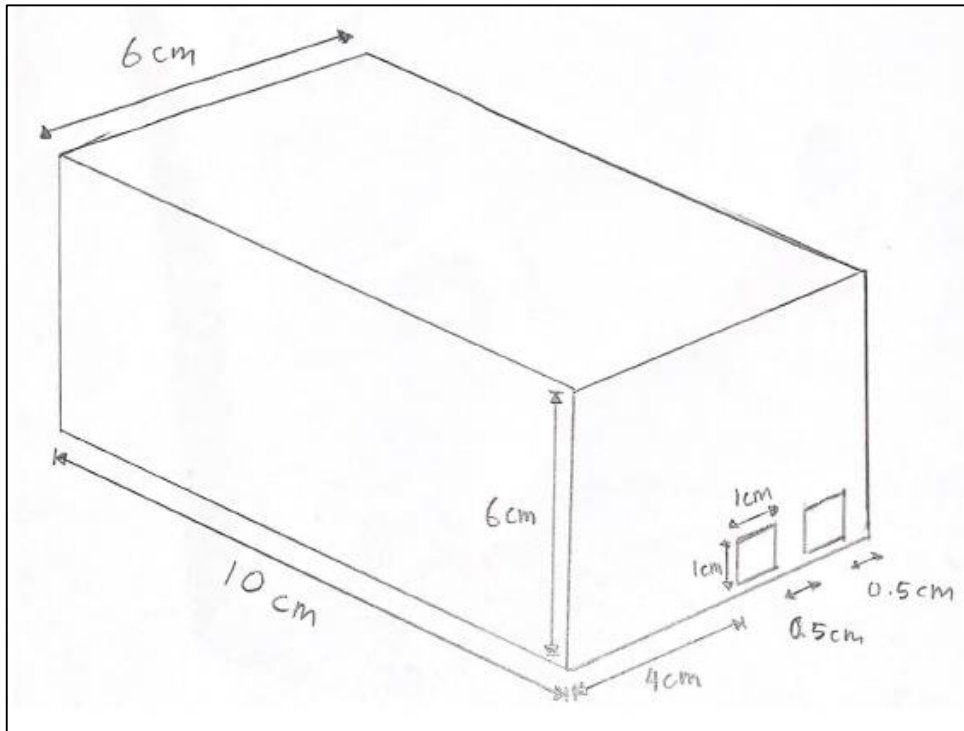


Figure 3.6 Product sketch main view

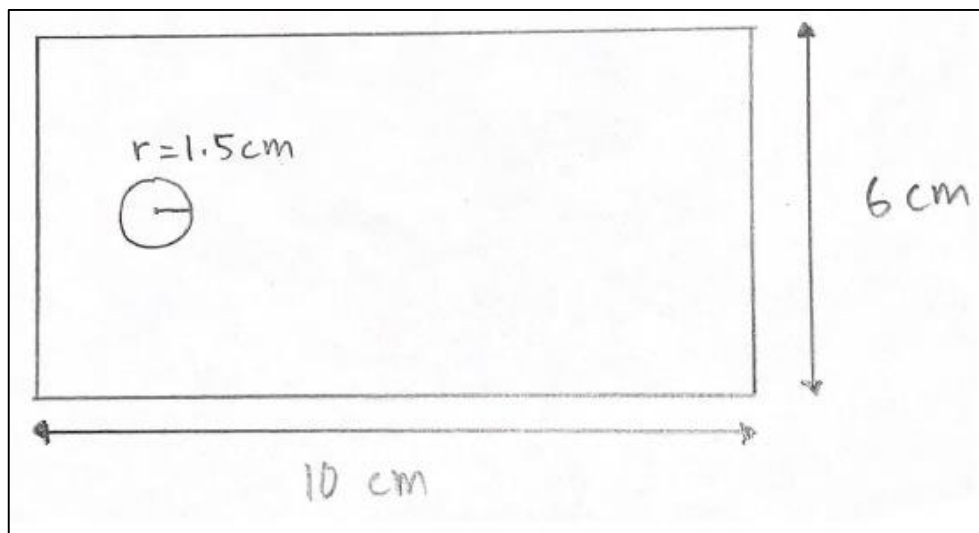


Figure 3.7 Product sketch bottom view

The product's definition is derived from a modeling diagram, which is used to determine the product's distance, height and scale. The sketch also shows where the sensor and system

framework should be placed in a housing that the customer can easily install. The hole in the sketch has a 1.7 cm radius which can be used to position the MQ-2 smoke sensor, which can accurately detect smoke. Furthermore, the hole on the side is for the Raspberry Pi power supply cable as well as the loudly audible Buzzer which will warn the building's owner.

3.7 Simulation Software

3.7.1 Python



Figure 3.8 Python Software (Bogdanchikov et al., 2013)

Python is the chosen programming language. The reason for choosing this program is due to the large number of applications available including prototyping (Bogdanchikov et al., 2013). Furthermore, Python language use is highly sought after in industry 4.0, making it one of the most in-demand markets due to its flexible features and minimal programming code (Sharma et al., 2020). In comparison to C, this language has a simpler syntax and comprehensive support libraries to boost efficiency.

3.7.2 Proteus



Figure 3.9 Proteus 8 Design suite software (*Introduction to Proteus - The Engineering Projects*, n.d.)

The Proteus design suite is a piece of software that aids in the creation of schematics and electronic prints (*Introduction to Proteus - The Engineering Projects*, n.d.). As a result, it's often used to run simulations virtually before trying them out on real hardware. The researcher will predict the outcome and fine-tune the predicted outcome with the aid of simulation. The package also includes a large library of component libraries including Raspberry Pi and sensor libraries. As a result, the provided result is nearly identical to the actual real-world hardware presentation.

3.7.3 Blynk Application

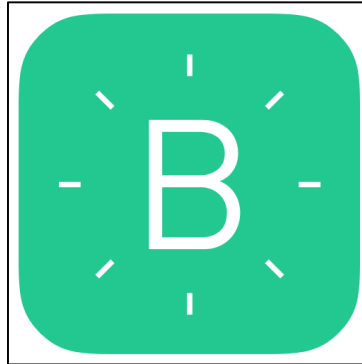


Figure 3.10 Blynk application ((*No Title*), n.d.)

Blynk is an example IoT platform that can be used to remotely control hardware and display data including graphics, tables, and other information about any project user input ((*No Title*), n.d.). The user interface (UI) can be customized to your preferences. Aside from that, it can store data and information obtained from any IoT platform-connected devices. For a first-timer, the setup is simple and straightforward. Before adding a project, the user must first build an account. To ensure that the connection and synchronization are perfect, the authentication token from the IoT platform must match the project coding (Media's et al., 2019). If a project has been developed, the user may begin configuring it, and adding widgets to the project is as simple as ABC.



Figure 3.11 Blynk interface for Ios and Android (*Blynk Program with Multiple WiFi Networks / FactoryForward, n.d.*)

3.8 List of Component

Table 3.1 List of components used for this project

| No | Materials | Quantity |
|----|---------------------------------------|-----------|
| 1 | Raspberry Pi 3 model B | 1 |
| 2 | Gas Sensor MQ2 | 1 |
| 3 | Humidity and Temperature sensor DHT11 | 1 |
| 4 | 40 ways male-to-male jumper wires | 1 |
| 5 | Cable sleeve | 1 |
| 6 | Casing | 1 |
| 7 | HDMI cable | 1 |
| 8 | Monitor | 1 |
| 9 | Peripherals (Mouse & Keyboard) | 2 |
| 10 | 2.1 Amp micro USB charger | 1 |
| 11 | Buzzer | 1 |
| 12 | LED | 1 |
| 13 | MCP3008 IC | 1 |
| | Total | 12 |

3.8.1 Raspberry Pi 3 Model B

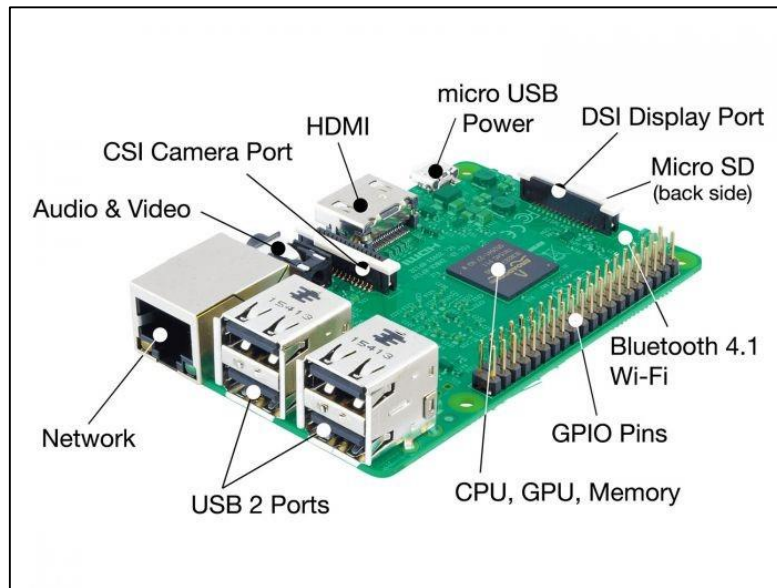


Figure 3.12 Raspberry Pi ports and header layout (Govardhini & Sumalatha, n.d.)

A single board machine that is both flexible and efficient enough to run its own operating system and function as a computer. Because of its processor's ability to process at a higher clock speed, Raspberry Pi was chosen for this project growth. The Raspberry Pi 3 model B has an ARM cortex processor that can clock up to 1.2GHz. Furthermore, this board has a variety of port extensions including HDMI for high-resolution image processing and Micro USB as a 3.3V supply port. This board has 4 USB 2.0 ports, a Micro SD memory expansion slot and a camera serial interface (CSI) that allows it to be connected to a camera for real-time monitoring. This board also includes 40 General Purpose Input Output (GPIO) ports that can be used to link and extend other boards and modules. Apart from the features mentioned above, this board supports dual-band WiFi 802.11ac for fast 2.4GHz and 5GHz wireless connections (Maksimović et al., 2014). Stable wireless communication is critical for IoT-based applications to achieve the desired results and efficiency. Finally, with such a capable and strong board, this project can be extended and improved in the future with features such as real-time monitoring, infrared image sensor and automation for increased protection.

|  RASPBERRY PI MODEL COMPARISON TABLE  | | | | | | | | | | |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------|
| RANGE | MODEL B | | | | | MODEL A | COMPUTE | | | |
| MODEL | Raspberry Pi 3 Model B+ | Raspberry Pi 3 Model B | Raspberry Pi 2 Model B V1.2 | Raspberry Pi 2 Model B | Raspberry Pi 1 Model B+ | Raspberry Pi 1 Model A+ | Raspberry Pi CM3 | Raspberry Pi CM3 Lite | Raspberry Pi CM1 | |
| IMAGE |  |  |  |  |  |  |  |  |  | |
| RS PART NO. | 137-3331 | 896-8660 | 125-9525 | 832-6274 | 811-1284 | 833-2699 | 123-2011 | 1232012 | 826-8825 | |
| BULK OPTION | 137-3331 | 896-8664 | 125-9526 | 847-2816 | 880-0608 | - | - | - | - | |
| BROADCOM SOC | BCM2837B0 | BCM2837 | | BCM2836 | BCM2835 | | BCM2837 | | BCM2835 | |
| Processor | ARM Cortex-A53 | | | ARM Cortex-A7 | ARM11 | | ARM Cortex-A53 | | ARM11 | |
| No. of Cores | Quad | | | | Single | | Quad | | Single | |
| CPU Speed | 1.4 GHz | 1.2 GHz | 900MHz | | 700MHz | | 1.2 GHz | | 700MHz | |
| Memory | 1 GB | | | | 512MB | | 1 GB | | 512MB | |
| Graphics (GPU) | Dual Core VideoCore IV® 1080p60 | | | Dual Core VideoCore IV® 1080p30 | | | Dual Core VideoCore IV® 1080p60 | | Dual Core VideoCore IV® 1080p30 | |
| USB 2.0 Ports | 4 | | | | | 1 | - | | | |
| GPIO Header |  |  |  |  |  |  | - | - | - | |
| HDMI Socket |  |  |  |  |  |  | - | - | - | |
| A/V Jack |  |  |  |  |  |  | - | - | - | |
| Micro SD Card Slot |  |  |  |  |  |  | - | - | - | |
| MIPI CSI Display Port |  |  |  |  |  |  | - | - | - | |
| CSI Camera Port |  |  |  |  |  |  | - | - | - | |
| Ethernet |  |  |  |  |  |  | - | - | - | |
| WLAN |  |  | - | - | - | - | - | - | - | |
| Bluetooth |  |  | - | - | - | - | - | - | - | |
| Bluetooth Low Energy |  |  | - | - | - | - | - | - | - | |
| PoE Enabled |  | - | - | - | - | - | - | - | - | |
| eMMC Flash | - | - | - | - | - | - |  | Enabled* | - | |
| HAT Compatible |  |  |  |  |  |  | - | - | - | |
| Dimensions (mm) | 85 x 56 x 17 | | | | | 66 x 56 x 14 | | 67 x 31 x 5 | | 67 x 30 x 5 |
| Weight (g) | 45 | | | | | 23 | | 7 | | |
| Power Input | MicroUSB 5.1V / 2.5A | | MicroUSB 5V / 2A | | | | MicroUSB 5.1V / 2.5A | | | |

Figure 3.13 Raspberry Pi model comparison table (*Raspberry Pi Model Comparison Table*, n.d.)

The table above shows the output class, processor type (clock speed), memory speed, ports, and header for each Raspberry Pi available on the market. These circuit boards are upgraded with each generation.

3.8.2 DHT11 Temperature and Humidity Sensor

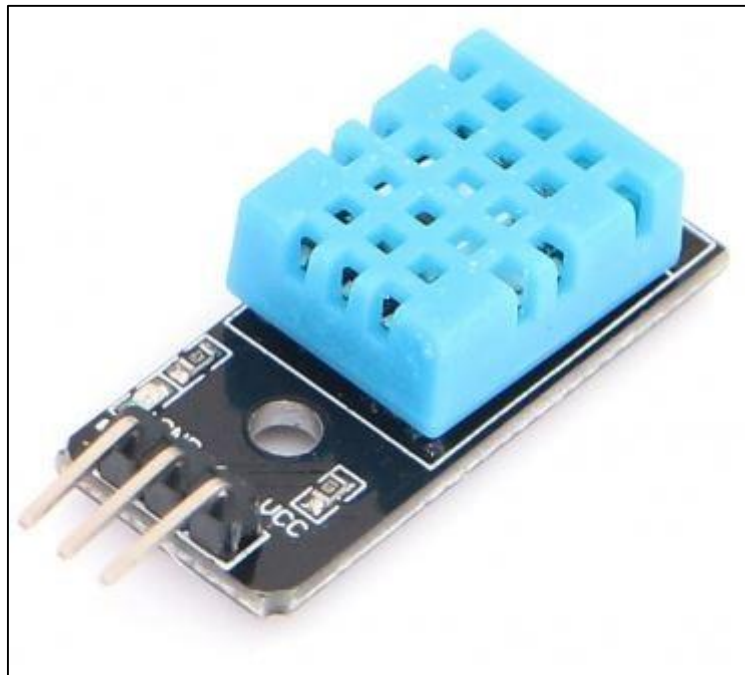


Figure 3.14 Temperature and Humidity Sensor (Wijaya et al., 2019)

The temperature and humidity detector's calibrated digital signal output is integrated into the DHT11 sensor. It is paired with an 8-bit microcontroller. Its program has a high level of dependability and consistency over time. A resistive part and an NTC wet temperature measurement detector are used in this device. It has exceptional power, speed, performance, and anti-interference capabilities (Srivastava et al., 2018).

A humidity calibration chamber is included with each DHT11 sensor. Internal sensors detect signals using the calibrating coefficients stored in the OTP device memory in this process. The serial interface single-wire unit is simple to set up and use. Small sizes, low power, and a signal transmission distance of up to 20 meters allow a wide range of applications, even the most difficult. The service is a four-pin single-pin package. Special packages may be created based on the needs of the users for a more convenient connection (Srivastava et al., 2018).

The sensor is integrated and adjusted in this application for this device to detect both temperature and humidity at the same time. The reason for selecting this type of sensor is that

it is simpler to use with the Raspberry Pi (Wijaya et al., 2019). When using this type of module, there are a few drawbacks to the device, such as the fact that it displays the humidity by default and is not changeable. Over and beyond the disadvantages, the DHT11 sensor's advantages include its high reliability and accuracy. The sensor's reading accuracy is above 96 percent, making it one of the best in the game. It can detect temperatures ranging from 0°C to 50°C, which is sufficient to detect a fire in the house (Srivastava et al., 2018).

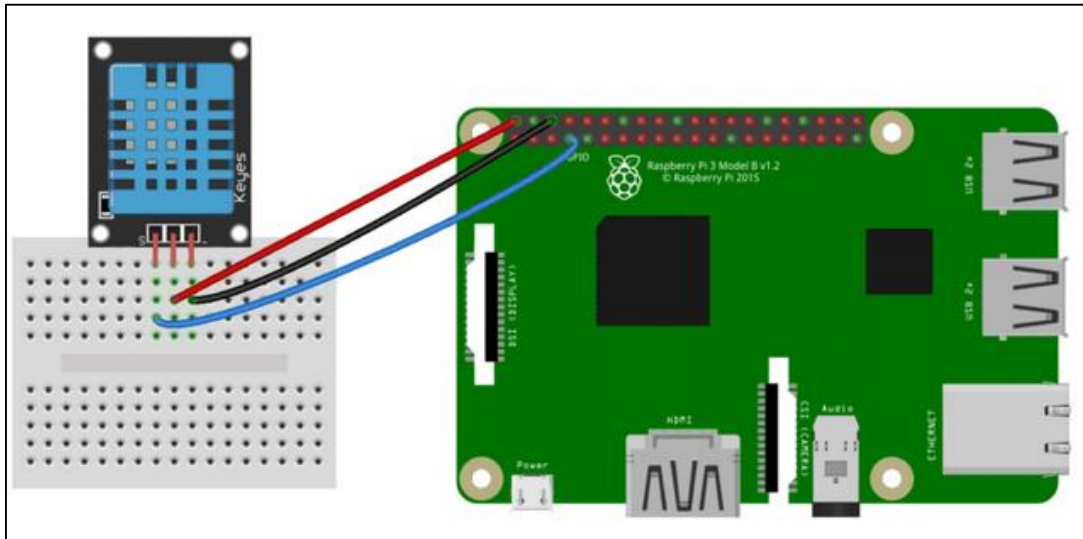


Figure 3.15 Schematic diagram for temperature sensor module connection (*How to Set Up the DHT11 Humidity Sensor on the Raspberry Pi - Circuit Basics*, n.d.)

The relation of the temperature sensor module to the Raspberry Pi GPIO header is shown in the diagram above. The DHT11 module and DHT11 sensor have different connections. The module only has a three-pin link to the RP GPIO header, while the DHT11 sensor has an additional pin (Vatsal & Bhavin, 2017).

3.8.3 MQ2 Smoke Sensor



Figure 3.16 Smoke sensor (1pcs MQ-2 MQ2 Gas Sensor Module Smoke Methane Butane Detection for Arduino | EBay, n.d.)

This MQ 2 gas detector is sensitive to a wide range of gases and is designed to be used indoors at room temperature. It can only output analog data. Raspberry Pi can only process and interpret digital signals; it is not capable of processing analog signals (Thakre, 2019). As a result, an analog-to-digital (ADC) converter is needed to convert analog signals to digital signals. As a result, the MCP3002 ADC will be used to convert any analog signal produced by the MQ 2 sensor to a digital signal. Not only can the MQ-2 Gas Sensor module detect smoke, but it can also detect a variety of combustible gases such as methane, liquefied petroleum gas (LPG), hydrogen, and carbon dioxide. The sensor can detect smoke concentrations ranging from 300 to 10,000 particles per million (PPM) (Al-zaheiree et al., 2020). It can also provide a stable and fast output signal to the microprocessor.

3.8.4 Buzzer & LED

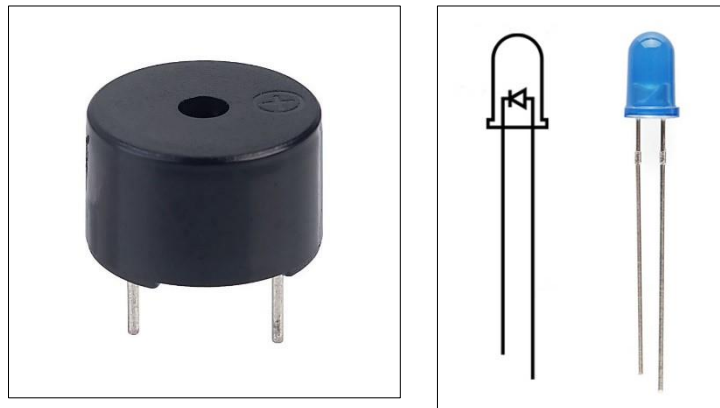


Figure 3.17 Buzzer & LED (*Buzzer 5V*, n.d.)

A buzzer is an electronic component with its own beeping or ringing function. The sound level of the buzzer can also be adjusted by turning up the volume with a possible meter (*Can a Buzzer Function as a Switch in a Circuit? - Electronic Guidebook*, n.d.). The light-emitting diode (LED) that can emit light when have current flow passed it. When the amount of smoke or fire in the building exceeds a certain threshold, the microprocessor will send a signal to the buzzer and LED, alerting the building occupants to extinguish the fire or leave the building.

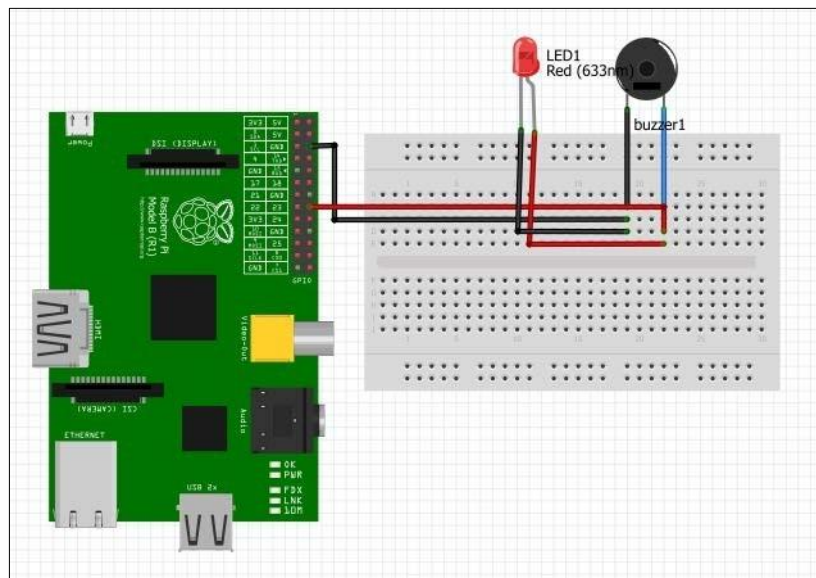


Figure 3.18 Schematic Raspberry Pi

The diagram above shows how to attach the buzzer to the Raspberry Pi PIN link board. The connection is simple, the buzzer has only two pins that are connected to pin 23 to initialize and another pin to GND on the Raspberry Pi board's GPIO header.

3.8.5 MPC3008 Analog to Digital converter set up

The MCP3008 is a low-cost 10-bit analogue to digital converter with eight channels (Tony Dicola, n.d.). This chip is a good choice for reading simple analogue signals, such as those from the MQ-2 gas sensor, for this project. Because the Raspberry Pi can only interpret digital signals, the analogue data from the MQ-2 sensor must be processed using an analogue to digital converter (ADC). It may detect smoke in the air based on the voltage value. The MCP3008 is a well-liked ADC chip that comes with a high recommendation.

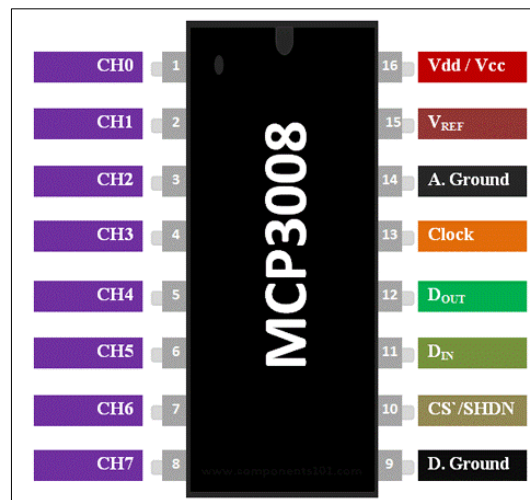


Figure 3.19 MCP3008 ADC IC Pinout (*MCP3008 ADC IC Pinout, Features, Equivalent & Datasheet, n.d.*)

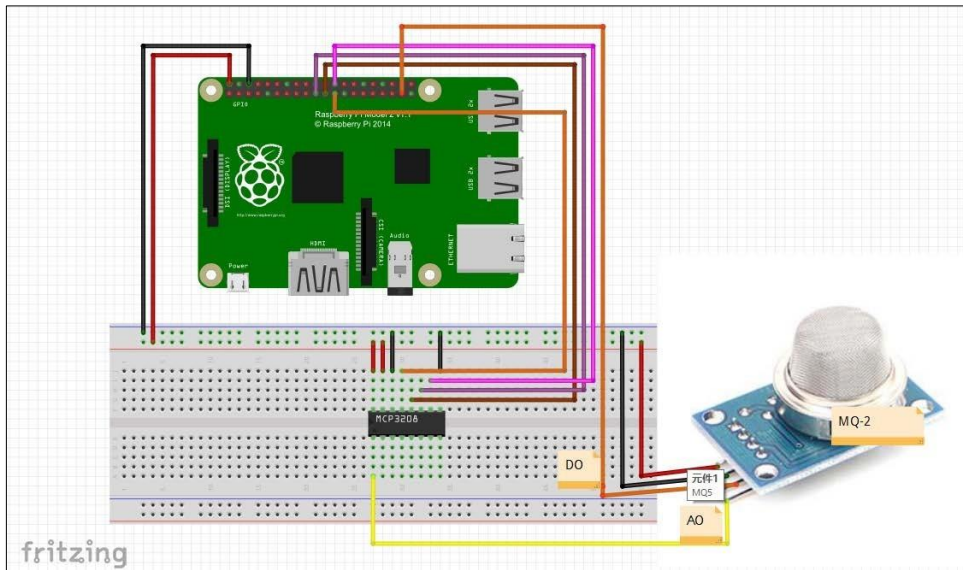


Figure 3.20 MQ-2 to ADC wiring diagram

3.9 Software Development

It is regarded as one of the project's most important components because it signals the system's non-physical development. This subtopic will cover Raspberry Pi configuration and Blynk framework setup.

3.9.1 Configuration of Raspberry Pi

Before proceeding with hardware installation and configuration, the first student must initialize the Raspberry Pi module. This microprocessor is a small machine with a Linux operating system that must be built and configured before it can be used and programmed by the user. The method is tedious and time-consuming for new customers. The setup guide is widely available on the internet. The RP module's boot mechanism should be mounted on a USB or Micro-USB drive (*Configuration - Raspberry Pi Documentation*, n.d.)

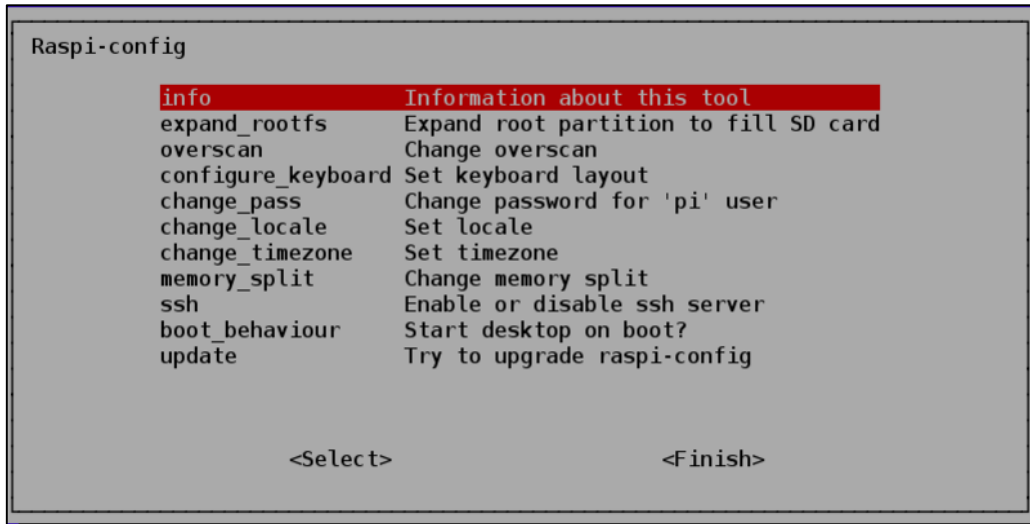


Figure 3.21 Raspberry Pi first time setup

The Raspberry Pi has no internal storage and must be booted using a microSD memory card that must be provided by the researcher. To speed up the procedure, the memory card should be larger than 8GB and have a class 10 speed. The first step is to get the official Raspberry Pi Imager from their website and install it on the PC that be using. Then, in the computer, insert the microSD card and run the Imager. Researchers select the Raspian OS for writing to the microSD card and then press the write button.



Figure 3.22 Raspberry Pi write image

The screen monitor should show the "Welcome to the Raspberry Pi" dialogue box after the Raspberry Pi boots up from the MicroSD. This process will get to minor adjustments based on user needs.

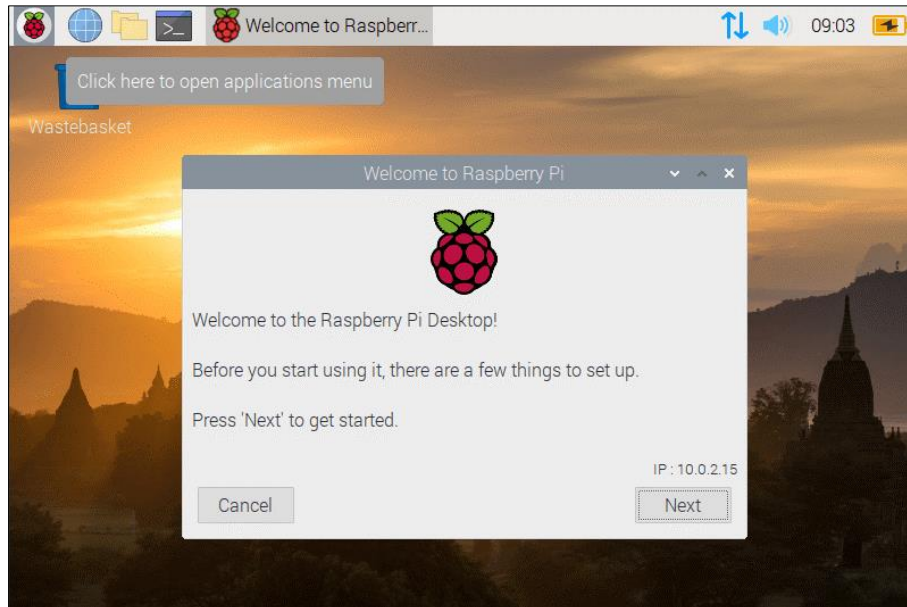


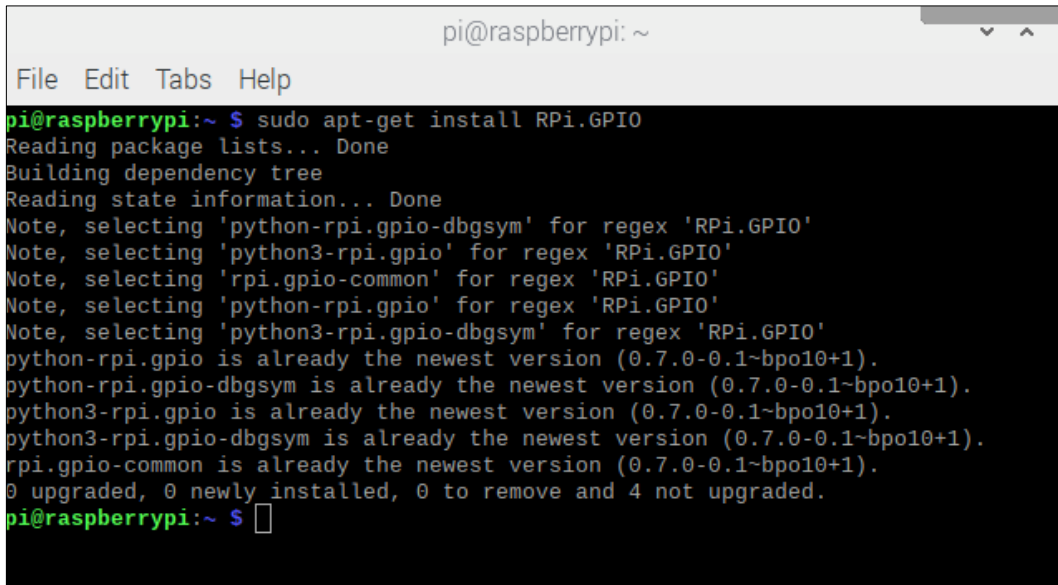
Figure 3.23 Raspberry Pi dialog box

3.9.2 Installation of programming library in Raspberry Pi

A programming library is a collection of resources that offer functionality, methods, and processes to assist programmers in completing various tasks. These libraries are instantiable, which implies that researchers can use them immediately away by launching them and calling the functions, methods, or procedures they need. Students must use the terminal to run commands to install all of the required libraries.

Working with libraries, particularly third-party libraries, is handled differently in each programming language. RPi, for example, is a library that fulfils a certain function or aim. GPIO libraries allow researchers to rapidly set and read/write the input/output pins on the Pi's GPIO header from within a Python script.

RPi.GPIO is a Python module for controlling the Raspberry Pi's GPIO interface. Ben Croston was the one who came up with the idea. GPIO pins can be referred to using either the actual pin numbers on the GPIO connection or the BCM channel names from the Broadcom SOC to which the pins are linked in RPi.GPIO.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo apt-get install RPi.GPIO  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Note, selecting 'python-rpi.gpio-dbgSYM' for regex 'RPi.GPIO'  
Note, selecting 'python3-rpi.gpio' for regex 'RPi.GPIO'  
Note, selecting 'rpi.gpio-common' for regex 'RPi.GPIO'  
Note, selecting 'python-rpi.gpio' for regex 'RPi.GPIO'  
Note, selecting 'python3-rpi.gpio-dbgSYM' for regex 'RPi.GPIO'  
python-rpi.gpio is already the newest version (0.7.0-0.1~bpo10+1).  
python-rpi.gpio-dbgSYM is already the newest version (0.7.0-0.1~bpo10+1).  
python3-rpi.gpio is already the newest version (0.7.0-0.1~bpo10+1).  
python3-rpi.gpio-dbgSYM is already the newest version (0.7.0-0.1~bpo10+1).  
rpi.gpio-common is already the newest version (0.7.0-0.1~bpo10+1).  
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.  
pi@raspberrypi:~ $
```

Figure 3.24 RPi.GPIO installation

The Blynk Library is the next library to be installed. This library is required for the Raspberry Pi to communicate with the Blynk application on the user's smartphone. Hardware can use hardware connections on the board (such as ESP32) or other shields to connect to the Blynk Cloud (open-source server) over the Internet (Ethernet, WiFi, GSM, LTE, etc). Blynk Cloud is a free service that all Blynk users can access. There's also a direct Bluetooth connection.

Before to install the library, the student needs to check first for the version of the Node.js in the Raspberry. The version should be above v0.10.38.

```
sudo apt-get update && sudo apt-get upgrade  
sudo apt-get install build-essential  
sudo npm install -g npm  
sudo npm install -g onoff  
sudo npm install -g blynk-library
```

Figure 3.25 Coding of Blynk library installation (*GitHub - Blynkkk/Lib-Python: Blynk IoT Library for Python and Micropython, n.d.*)

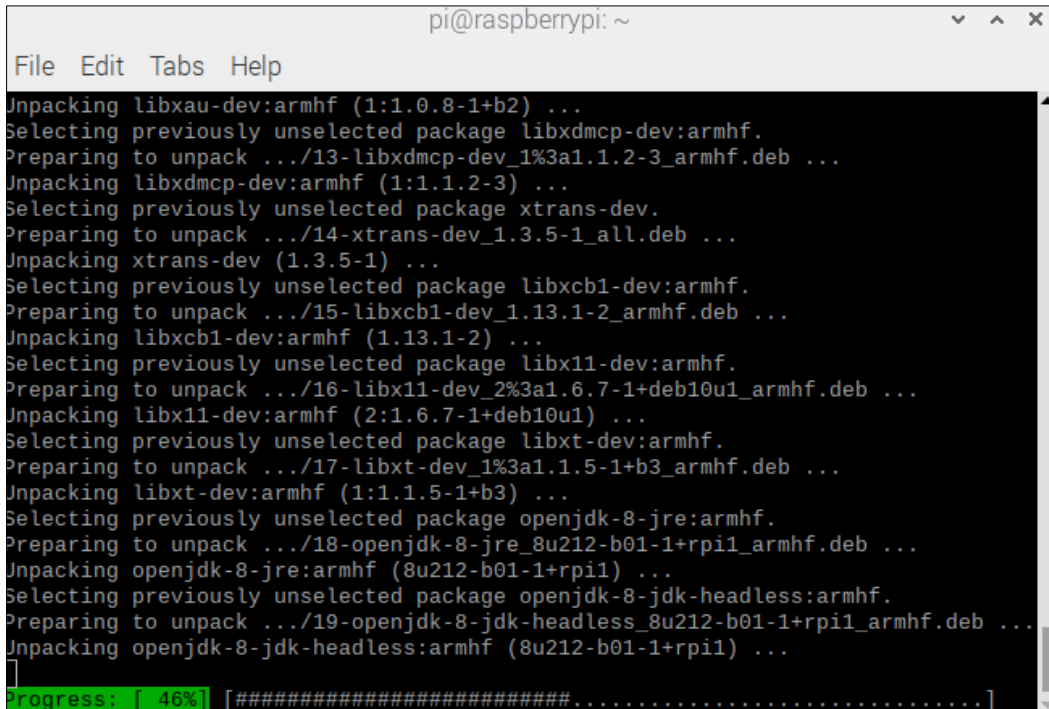
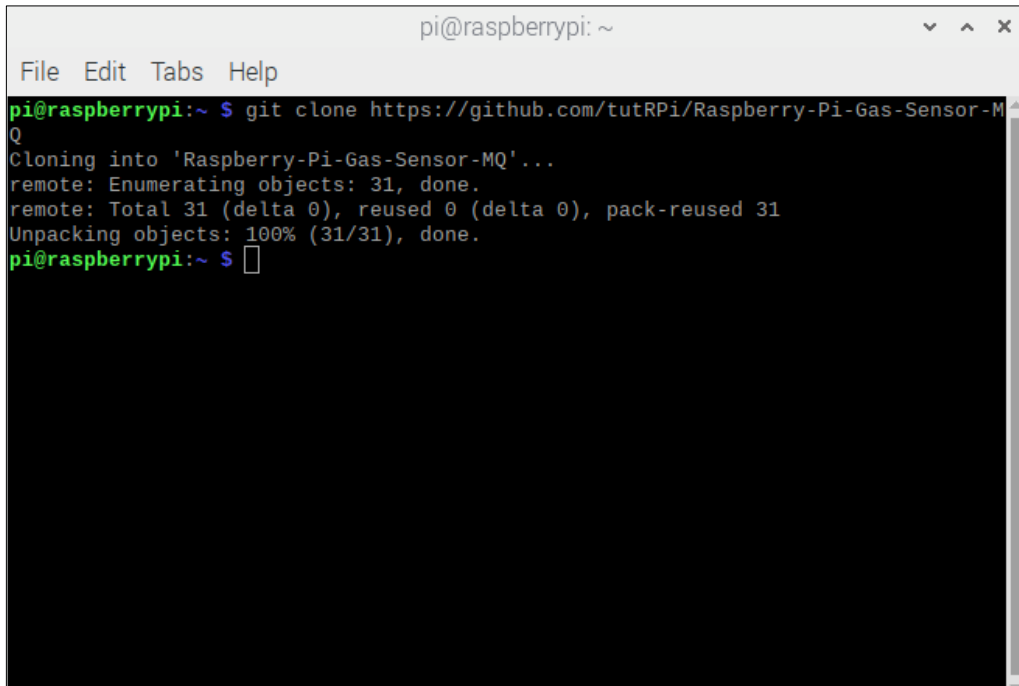


Figure 3.26 Blynk Library installation

The MQ-X library is the next library to be installed. It is necessary to detect the components of the air, which needs the employment of sensors. Smoke detectors, for example, contain them. Because this project only uses MQ-2 sensors, the library is ready to take other MQ sensors if more are needed.

```
git clone https://github.com/tutRPi/Raspberry-Pi-Gas-Sensor-MQ
```

Figure 3.27 Coding of MQ-2 sensor (*GitHub - Blynkkk/Lib-Python: Blynk IoT Library for Python and Micropython, n.d.*)

A terminal window titled 'pi@raspberrypi: ~' with a menu bar containing 'File Edit Tabs Help'. The terminal output shows the command 'git clone https://github.com/tutRPi/Raspberry-Pi-Gas-Sensor-MQ' being executed. The output includes: 'Cloning into 'Raspberry-Pi-Gas-Sensor-MQ'...', 'remote: Enumerating objects: 31, done.', 'remote: Total 31 (delta 0), reused 0 (delta 0), pack-reused 31', and 'Unpacking objects: 100% (31/31), done.'. The prompt returns to 'pi@raspberrypi:~ \$' with a cursor.

```
pi@raspberrypi:~ $ git clone https://github.com/tutRPi/Raspberry-Pi-Gas-Sensor-MQ
Cloning into 'Raspberry-Pi-Gas-Sensor-MQ'...
remote: Enumerating objects: 31, done.
remote: Total 31 (delta 0), reused 0 (delta 0), pack-reused 31
Unpacking objects: 100% (31/31), done.
pi@raspberrypi:~ $
```

Figure 3.28 MQ-X library installation

The DHT-11 input library must also be installed by the researcher. This library will allow the Raspberry Pi to read temperature and humidity data from the DHT22. Researchers can use pip to install a library from PyPi.

```
sudo pip3 install Adafruit_DHT
pip3 install adafruit-circuitpython-dht
```

Figure 3.29 Coding of Adafruit Python DHT library installation (*GitHub - Blynkkk/Lib-Python: Blynk IoT Library for Python and Micropython, n.d.*)

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ pip3 install adafruit-circuitpython-dht  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Requirement already satisfied: adafruit-circuitpython-dht in ./local/lib/python3.7/site-packages (3.5.3)  
Requirement already satisfied: Adafruit-Blinka in ./local/lib/python3.7/site-packages (from adafruit-circuitpython-dht) (5.9.1)  
Requirement already satisfied: rpi-ws281x>=4.0.0 in ./local/lib/python3.7/site-packages (from Adafruit-Blinka->adafruit-circuitpython-dht) (4.2.5)  
Requirement already satisfied: pyftdi>=0.40.0 in ./local/lib/python3.7/site-packages (from Adafruit-Blinka->adafruit-circuitpython-dht) (0.52.0)  
Requirement already satisfied: RPi.GPIO in /usr/lib/python3/dist-packages (from Adafruit-Blinka->adafruit-circuitpython-dht) (0.7.0)  
Requirement already satisfied: Adafruit-PureIO>=1.1.7 in ./local/lib/python3.7/site-packages (from Adafruit-Blinka->adafruit-circuitpython-dht) (1.1.8)  
Requirement already satisfied: sysv-ipc in ./local/lib/python3.7/site-packages (from Adafruit-Blinka->adafruit-circuitpython-dht) (1.0.1)  
Requirement already satisfied: Adafruit-PlatformDetect>=2.18.1 in ./local/lib/python3.7/site-packages (from Adafruit-Blinka->adafruit-circuitpython-dht) (2.23.0)  
Requirement already satisfied: pyusb>=1.0.0 in ./local/lib/python3.7/site-packages (from pyftdi>=0.40.0->Adafruit-Blinka->adafruit-circuitpython-dht) (1.1.0)  
Requirement already satisfied: pyserial>=3.0 in /usr/lib/python3/dist-packages (from pyftdi>=0.40.0->Adafruit-Blinka->adafruit-circuitpython-dht) (3.4)  
pi@raspberrypi:~ $
```

Figure 3.30 DHT-11 library installation

Lastly, the library that need to be install is MCP3008 analog to digital converter.

```
sudo pip install adafruit-mcp3008
```

Figure 3.31 Coding of MCP3008 installation (*GitHub - Blynkkk/Lib-Python: Blynk IoT Library for Python and Micropython, n.d.*)

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo pip install adafruit-mcp3008
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting adafruit-mcp3008
  Downloading https://files.pythonhosted.org/packages/1f/47/a76b3897ef7fa48c4350
05149ab625c4200f29ab9413829c2cea7fecbec7/Adafruit_MCP3008-1.0.2-py2-none-any.whl
Collecting Adafruit-GPIO>=0.6.5 (from adafruit-mcp3008)
  Downloading https://files.pythonhosted.org/packages/db/1c/2dc8a674514219f287fa
344e44cadfd77b3e2878d6ff602a8c2149b50dd8/Adafruit_GPIO-1.0.3.tar.gz
Collecting adafruit-pureio (from Adafruit-GPIO>=0.6.5->adafruit-mcp3008)
  Downloading https://files.pythonhosted.org/packages/93/e4/e6e25699445b4d8aafa9
7ed705ed43b39bcd6db17127ea7073aeaa76aad8/Adafruit_PureIO-1.0.1.tar.gz
Requirement already satisfied: spidev in /usr/lib/python2.7/dist-packages (from
Adafruit-GPIO>=0.6.5->adafruit-mcp3008) (3.4)
Building wheels for collected packages: Adafruit-GPIO, adafruit-pureio
  Running setup.py bdist_wheel for Adafruit-GPIO ... done
  Stored in directory: /root/.cache/pip/wheels/4c/f3/5e/762e517d0fa6217290169f4c
b2fb9a0e44d2c2ac8093bd68a5
  Running setup.py bdist_wheel for adafruit-pureio ... done
  Stored in directory: /root/.cache/pip/wheels/74/1a/42/562e829207b81e5e9be8bb26
f8c83c9d6780c3054c836cac2f
Successfully built Adafruit-GPIO adafruit-pureio
Installing collected packages: adafruit-pureio, Adafruit-GPIO, adafruit-mcp3008
Successfully installed Adafruit-GPIO-1.0.3 adafruit-mcp3008-1.0.2 adafruit-purei
o-1.0.1

```

Figure 3.32 MCP3008 library installation

3.9.3 HDMI Configuration

Before using the HDMI port, the user must first set it up. The CEA, which is often used by TVs, and the DMT, or Display Monitor Timing, which is typically used by monitors, are the two most prevalent HDMI groups. HDMI is divided into two classes. In each band, the mode sets the size, frame rate, clock speeds, and aspect ratio for the output.

Table 3.2 Code for setting HDMI port and configuration

| Timing | Purpose |
|------------------------------|-----------------------------------------------|
| <code>h_active_pixels</code> | The horizontal resolution |
| <code>h_sync_polarity</code> | 0 or 1 to define the horizontal sync polarity |
| <code>h_front_porch</code> | Number of horizontal front porch pixels |
| <code>h_sync_pulse</code> | Width of horizontal sync pulse |
| <code>h_back_porch</code> | Number of horizontal back porch pixels |
| <code>v_active_lines</code> | The vertical resolution |
| <code>v_sync_polarity</code> | 0 or 1 to define the vertical sync polarity |
| <code>v_front_porch</code> | Number of vertical front porch pixels |

| | |
|------------------------------|----------------------------------------|
| <code>v_sync_pulse</code> | Width of vertical sync pulse |
| <code>v_back_porch</code> | Number of vertical back porch pixels |
| <code>v_sync_offset_a</code> | Leave at 0 |
| <code>v_sync_offset_k</code> | Leave at 0 |
| <code>pixel_rep</code> | Leave at 0 |
| <code>frame_rate</code> | Frame rate of mode |
| <code>interlaced</code> | 0 for non-interlaced, 1 for interlaced |
| <code>pixel_freq</code> | The mode pixel frequency |
| <code>aspect_ratio</code> | The aspect ratio required |

Table 3.3 Code for setting up aspect ratio for HDMI

| Ratio | aspect_ratio ID |
|--------------|------------------------|
| 4:3 | 1 |
| 14:9 | 2 |
| 16:9 | 3 |
| 5:4 | 4 |
| 16:10 | 5 |
| 15:9 | 6 |
| 21:9 | 7 |
| 64:27 | 8 |

The student used a 24" full HD monitor with a 16:9 aspect ratio for this project. Researchers used the code '3' to get the 16:9 aspect ratio, which was the ideal resolution for the monitor. Before the user can use HDMI, users must first configure it.

3.9.4 VNC (Virtual Network Computing) settings

VNC (Virtual Network Computing) is a graphical desktop sharing technology that allows researcher to control the desktop interface of one computer (running VNC Server) from another computer or mobile device (running VNC Viewer) (*What Is Virtual Network Computing (VNC)? - Definition from Techopedia, n.d.*). VNC Viewer receives the connection and allows the user to make changes using the keyboard, mouse, and remote control. Researchers can use VNC to make updates and modifications remotely without having to attach an HDMI wire to the Raspberry Pi module. The Raspberry Pi's desktop will appear on the computer or mobile device that opens it. The user must first enable VNC Server before using it. VNC Server allows users to make modifications to the Raspberry Pi from any location as long as they have internet access.

Direct connections are quick and easy if the user is connected to the same private local network as the Raspberry Pi. For example, this could be a wired or wireless network at home, school, or the office. To find the user's private IP address on the Raspberry Pi, use this step or execute 'ifconfig' (command used for display current network configuration network) via a terminal window or via SSH. Then, on the device that the user will be using to take control of, download the VNC Viewer. Type the user Raspberry Pi's private IP address into VNC Viewer.



Figure 3.33 VNC Viewer setup connection

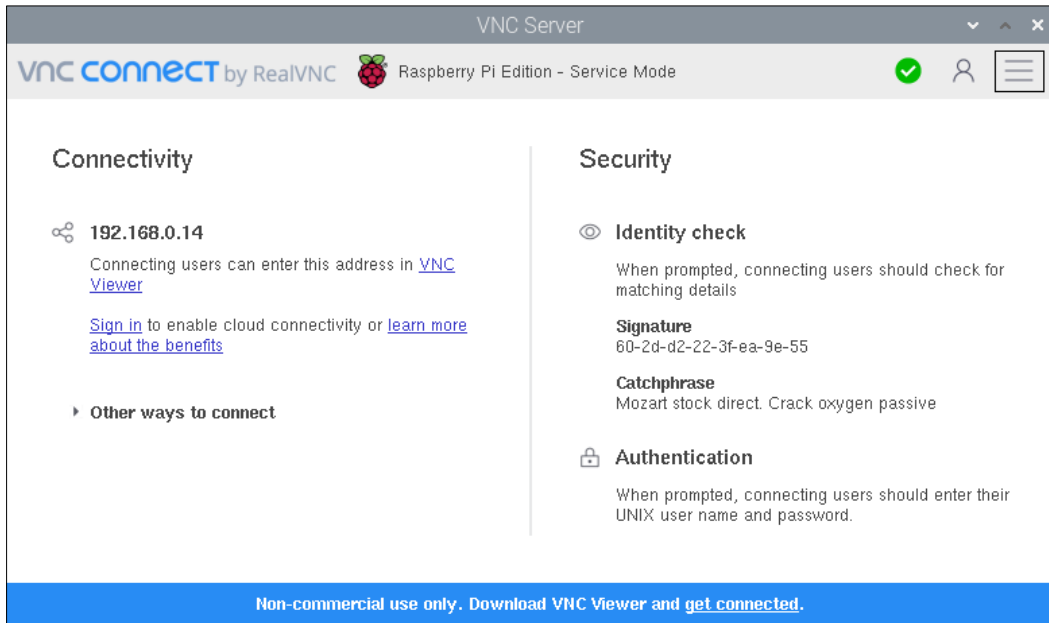


Figure 3.34 VNC Viewer on Raspberry Pi

3.9.5 Blynk App set up and configuration

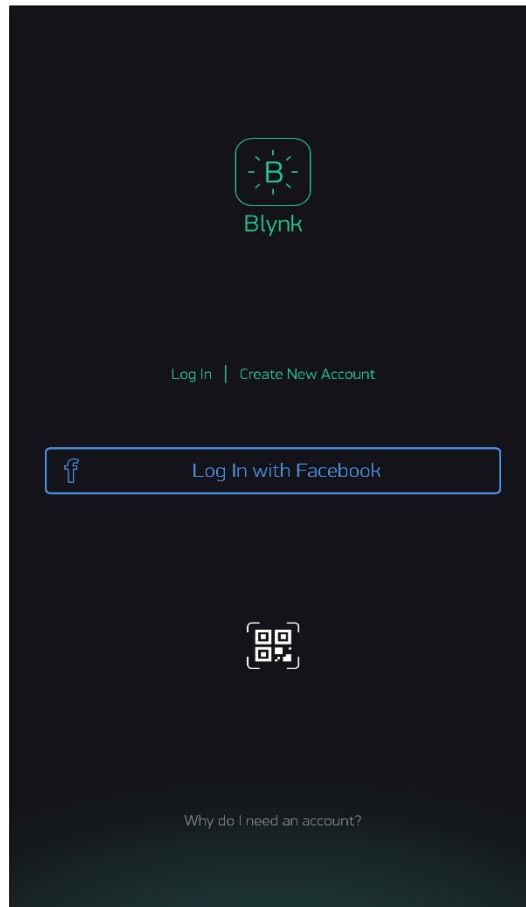


Figure 3.35 Log in sign up interface on Blynk application

The Blynk application was listed as an IoT platform in this system theoretically in chapter 3 by the researcher. The researcher will demonstrate how to set up the Blynk application in this subtopic. To begin, the researcher set up an account on the Blynk app. Once all is set up, the researcher will log in as shown in Figure 3.19. The second step is to build a new project and choose the necessary hardware, which in this case is a Raspberry Pi. Following that, the user will receive an Auth Token via email from the Blynk app ((No Title), n.d.).

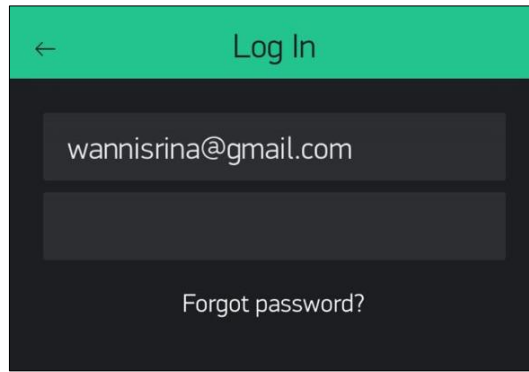


Figure 3.36 Log in Blynk application

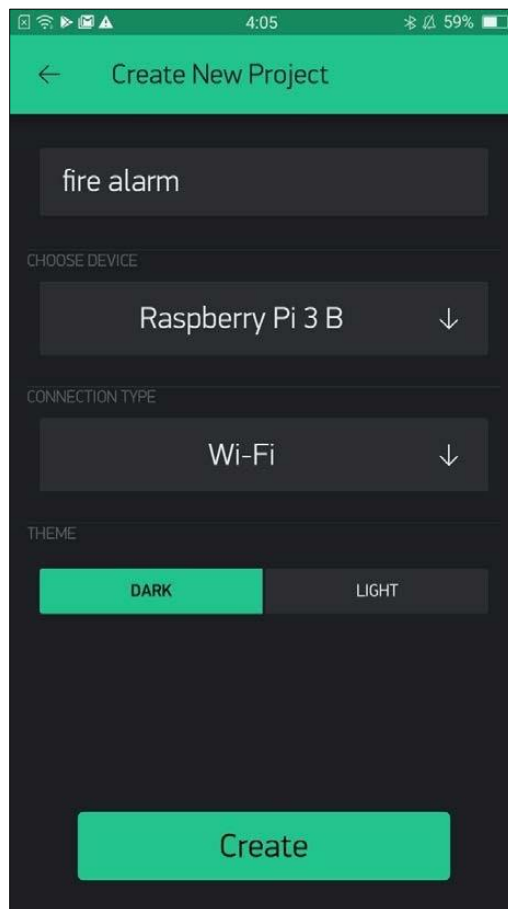


Figure 3.37 Blynk application create new project interface

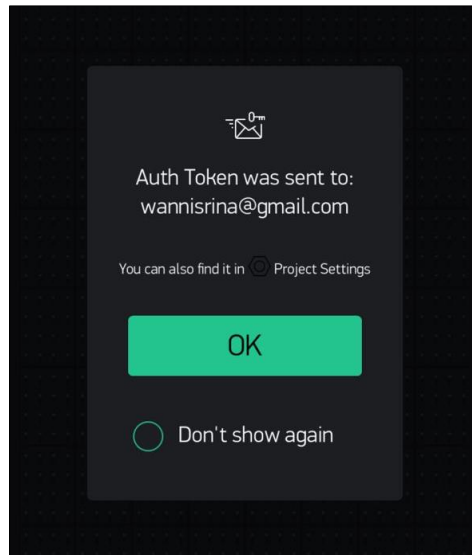


Figure 3.38 User get Auth Token in Blynk application

In the figure 3.37, the user creates a new project and names it whatever they want; in this case, the researcher named it "fire alarm," and then chose "Raspberry Pi 3" as the hardware model and WiFi as the preferred connection between the IoT platform and the generic board, which is the Raspberry Pi model 3 B. On the IoT platform, once the researcher has created the project, it will generate an authentication token. The authentication token can be emailed to any email address. To ensure that the connection is successful, the authentication token on the IoT must match the authentication token in Python code. In this case, the user chose to send the email through Gmail.

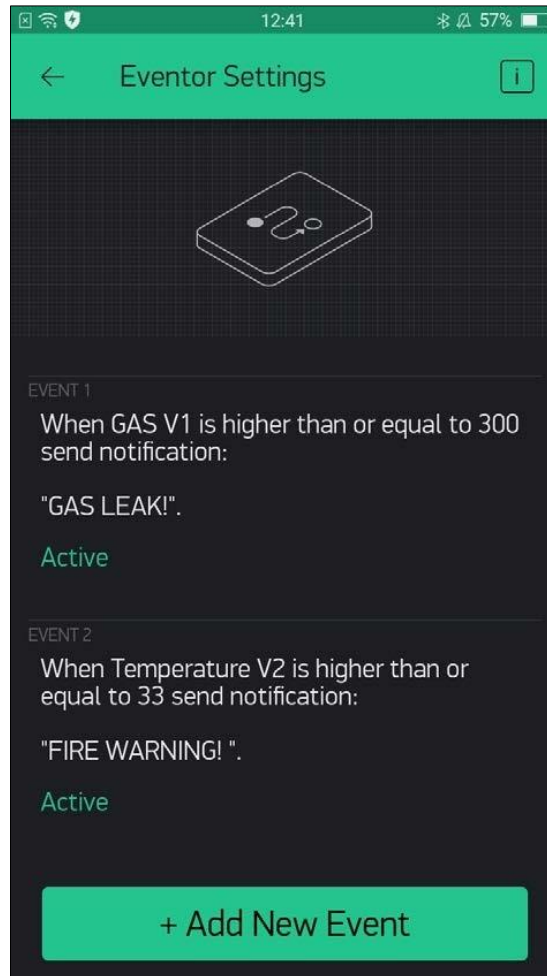


Figure 3.39 Eventor setting interface

In the 'Eventor setup' interface, the researcher configures and sets the project's parameters. Because two sensors are linked, the researcher opted to add two events to the eventor configuration, one for the gas leakage sensor and the other for the temperature sensor. The gas leakage parameter was set using analogue to digital converted input from the Raspberry Pi. When the gas is set to threshold state '300,' the Raspberry Pi will send data to the IoT platform, and the Blynk will process the data. The gas leakage threshold has been set at 300ppm (particle per million). However, because the Raspberry Pi only processes analogue input, the researchers installed the MCP3008 analogue to digital converter so that the IoT could display the exact reading for the gas detection sensor. The user will then receive a notification with the words "GAS LEAK!" displayed. The researcher configures the second eventor option to process and transmit a notification when a flame is spotted. The temperature threshold for the second eventor has been set to above room temperature. Blynk will analyse and send the

notification and message "FIRE WARNING!" displayed when the temperature is more than or equal to 33°C.



Figure 3.40 Real time main display interface

On the main display of this interface, all information from the generic board is displayed. The value presented on the interface is a real-time value retrieved via Blynk server from the generic board. After the application was completed and tested, it was discovered that the connection to the Blynk server had failed during the first trial. This was due to an unsuccessful connection to the server. To fix the issue, the researcher needs to refresh and generate a new authentication token, as well as restarting the WiFi connection. The connection was successful the second time around, and the interface displayed the real-time value as it tested.

The value display settings are illustrated in the figure below. The widget is used by the user to display the real-time value that the device's sensor is reading. The input must match the

code exactly. The value cannot be displayed in the Blynk app if the input pin is different. For the settings, the researcher additionally employs a virtual pin.

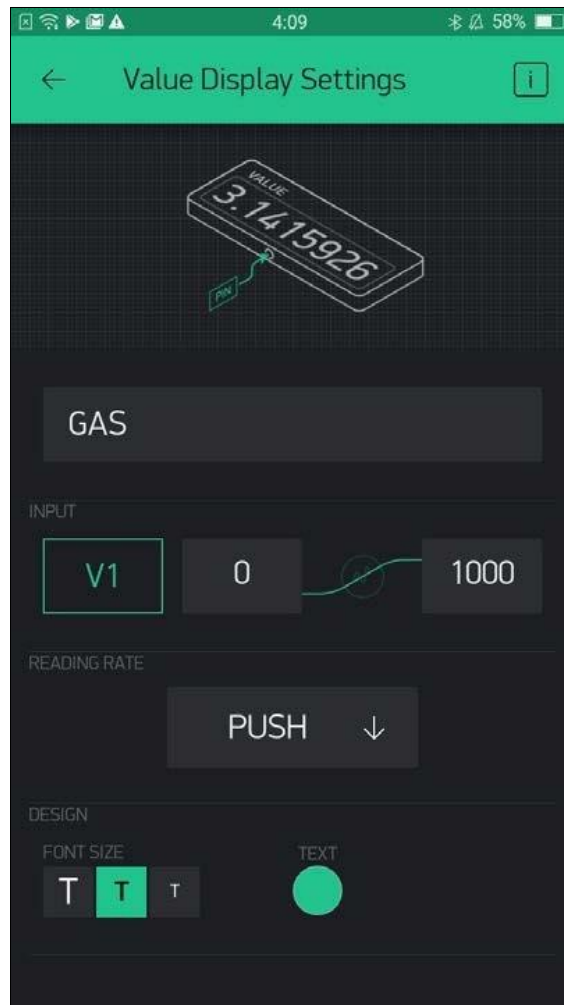


Figure 3.41 Value display gas settings

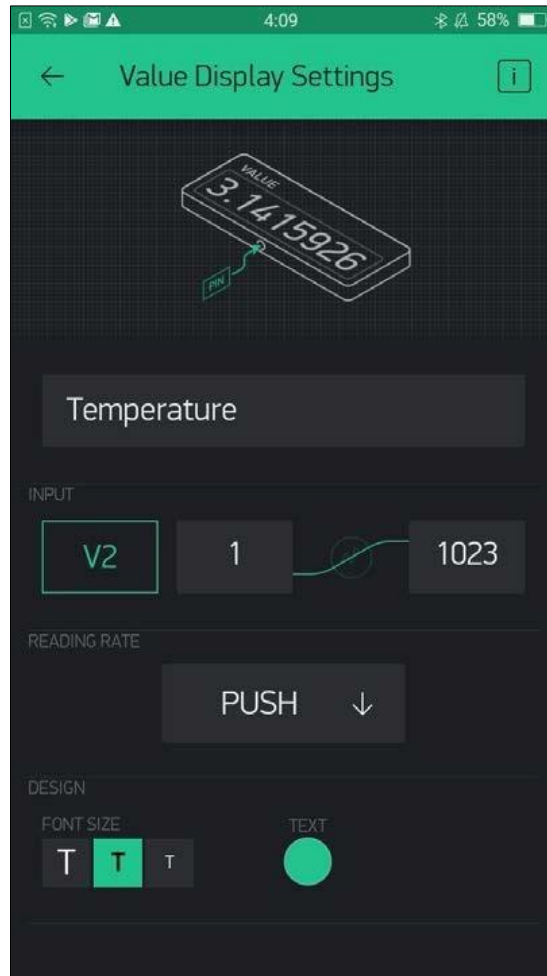


Figure 3.42 Value display temperature settings

3.9.6 Python software coding

The researcher used the Raspberry Pi as a computer and connected all peripherals, including a keyboard, mouse, and LED screen, to insert the Python codex on the Raspbian interface. Python coding is basic and straightforward to produce. The application was written in the Python programming language. The code was done in Python 3.7 by the researcher. In the coding box below, it shows the coding for the core programming structure. Each coding line utilised in the system will be explained by the researcher.

```

#Libraries
import BlynkLib
import Adafruit_DHT
import RPi.GPIO as GPIO
import time
from time import sleep
GPIO.setwarnings(False)

#SPI port on the ADC to the Cobbler
SPICLK = 11
SPIMISO = 9
SPIMOSI = 10
SPICS = 8
mq2_dpin = 26
mq2_apin = 0
CE1 = 7
BLYNK_AUTH = 'g0HRVGNzbGk4k3i9rRgIlx-bQ2iTY5uO'
sensor_type = Adafruit_DHT.DHT11
sensor_pin = 17

```

Figure 3.43 The coding in Python

The researcher imports the blynk library into the codex on the first line to ensure that the blynk server is reachable between the generic board and the IoT platform. The DHT11 temperature sensor library is thereafter imported by the researcher. By default, this sensor comes with a humidity sensor in the same module. The researcher then imported the GPIO port library on the following line. This is important to know because GPIO is one of the most important components on the generic board for connecting the sensors and buzzer. The importance of sleep is the next item on the list. The sleep () function suspends (waits) the current thread's execution for a specified number of seconds.

Then there's SPI, which stands for serial peripheral interface. The SPI is a data transfer protocol used by microcomputers like the Raspberry Pi to communicate with peripheral devices. Sensors or actuators are examples of peripheral devices. The sensor used in this project is the MQ-2, which is an analogue to digital sensor that takes an analogue voltage and changes it to a digital value that the Raspberry Pi can understand. To interact with the target device, SPI requires four distinct connections. Serial clock (CLK), Master Input Slave Output (MISO), Master Output Slave Input (MOSI), and Chip Select are the connections (CS).

```

# GPIO.setmode(GPIO.BOARD)

# Initialize Blynk
blynk = BlynkLib.Blynk(BLYNK_AUTH)

#port init
def init():
    GPIO.setwarnings(False)
    GPIO.cleanup()           #clean up at the end of the
script
    GPIO.setmode(GPIO.BCM)   #to specify which pin
numbering system

# set up the SPI interface pins
GPIO.setup(SPIMOSI, GPIO.OUT)
GPIO.setup(SPIMISO, GPIO.IN)
GPIO.setup(SPICLK, GPIO.OUT)
GPIO.setup(SPICS, GPIO.OUT)
GPIO.setup(mq2_dpin,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(CE1, GPIO.OUT, initial=GPIO.LOW) #LED &
Buzzer

```

Figure 3.44 The coding of setup GPIO in Python

The initialise Blynk line establishes a connection between the Raspberry Pi board and the Blynk server, using the Blynk Auth that the user receives in his or her email. This method is called when an object is formed from a class, and it allows the class to initialise the attributes of the class. GPIO. Using numbering schemes, BCM will designate which pin is to be used.

```

#Read SPI data from MCP3008(or MCP3204) chip,8 possible adc's (0
thru 7)
def readadc(adcnum, clockpin, mosipin, misopin, cspin):
    if ((adcnum > 7) or (adcnum < 0)):
        return -1
    GPIO.output(cspin, True)

    GPIO.output(clockpin, False) # start clock low
    GPIO.output(cspin, False)    # bring CS low

    commandout = adcnum
    commandout |= 0x18 # start bit + single-ended bit
    commandout <= 3    # we only need to send 5 bits here
    for i in range(5):

```

Figure 3.45 The coding of MCP3008 in Python

Starting with this line, the MCP3008 analogue to digital converter must function properly. This line employs the MCP3008 Python library's read adc () method. The function's

only parameter is the number of channels to read (a value of 0 to 7). As a result, the function will return the current ADC value for the channel.

```

def main():
    init()
    print("please wait...")
    time.sleep(5)
    while True:
        COlevel=readadc(mq2_apin, SPICLK, SPIMOSI,
SPIMISO, SPICS)
        gas=GPIO.input(SPICS)
        humidity, temperature =
Adafruit_DHT.read_retry(sensor_type, sensor_pin)
        print(('Humidity = {:.1fI}%\tTemperature =
{:.1f}°C'format(humidity, temperature))
        print("Gas = " +str("%.1f"%(COlevel))+ " ppm")

        if (COlevel)>=300:
            print("GAS LEAKAGE WARNING!")
            time.sleep(0.5)
            GPIO.output(CE1, GPIO.HIGH) # Turn on
LED & Buzzer

            print()

            elifI temperature>33:
                GPIO.output(CE1, GPIO.HIGH) # Turn on
LED & Buzzer

                print("FIRE WARNING!")
                print()

            else:
                print("Safe Area")
                print()
                GPIO.output(CE1, GPIO.LOW) # Turn off
LED & Buzzer

                blynk.virtual_write(1,
' {:.1f}'.format((COlevel)))
                blynk.virtual_writIe(2,
' {:.1f}'.format(temperature))

                blynk.run()

```

Figure 3.46 The primary coding to execute in Python

The primary coding to execute the project is in the last section. All of the essential coding is contained in this paragraph. The majority of the coding, for example, is to call back the sleep function from the preceding paragraph of the code. To prepare the sensors, the researcher chose to add a 5-second delay before starting the main coding. The researcher then commands MQ-2 to read the sensor, then DHT 11 to read temperature, and finally the Python function to report out the reading. If the output is greater than or equivalent to 300PPM, the researcher set a high threshold for the gas, and if the output exceeds it, the LED and buzzer

will switch on. When the temperature rises above 33 degrees, the trigger is triggered. It will print "Safe Area" if the aforesaid circumstance does not occur.

3.10 Hardware Development

The hardware development shows how far the system has progressed physically. This subtopic was used to track and record the entire procedure. Because there are fewer components involved in this project, physical development is easier and more straightforward than software development. Sensors, buzzer, and LED installation, MCP3008, shell preparation, and prototype testing results will all be covered in this subtopic.

3.10.1 DHT11 Module Installation

On the module, this sensor is configured with three pins. Female to female jumper cable was used to connect the VCC, Data, and GND pins. The VCC pin is linked to the GPIO port's 5V pin, while the data wire is connected to pin 11 or GPIO17 as specified in the python coding above, and the ground pin is connected to the ground pin. The installation technique for the sensor is shown in the diagram below.

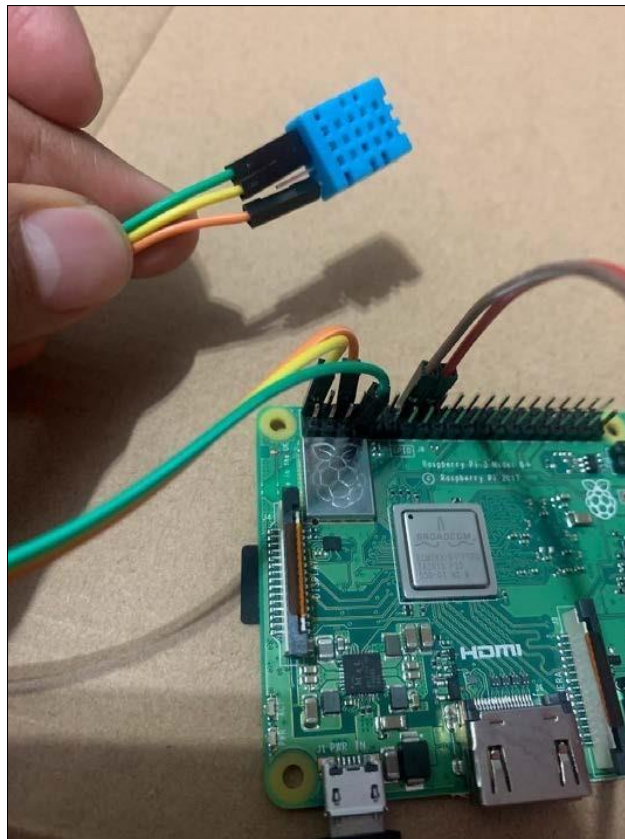


Figure 3.47 DHT11 installation process

3.10.2 MQ-2 Module Installation

This module contains four connection headers, but only the first two was utilised to make it work. The red jumper cord in the diagram below is the VCC connection to the GPIO's power connector on RP. This module receives 3.3V of power from the GPIO port. The green input cable from the module is then attached to pin 23 or GPIO11, as indicated in the preceding coding. This connection allows the sensor's input to be read before it is processed by the microprocessor. Finally, connect GND to the GPIO's ground port. All connections are double-checked to ensure that they are on the correct port.

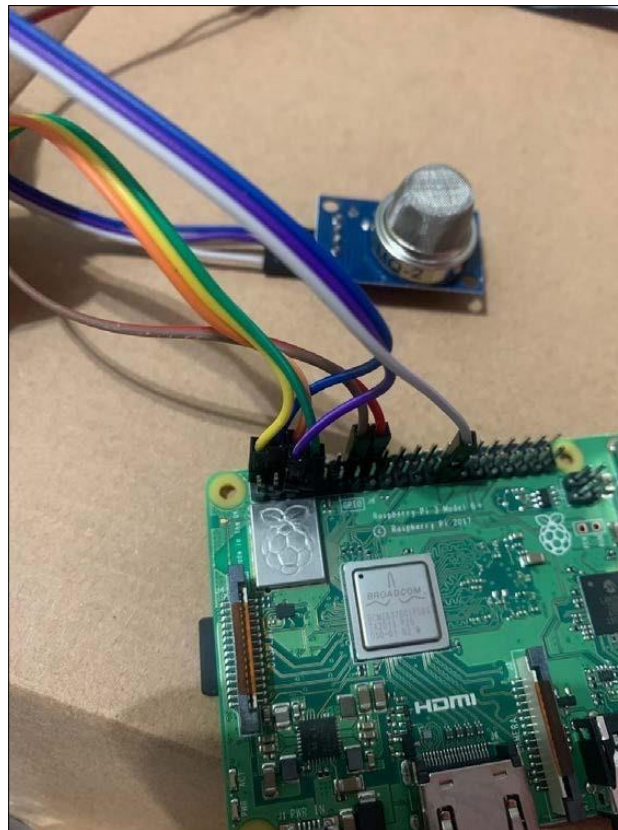


Figure 3.48 MQ-2 installation process

3.10.3 Buzzer and LED Installation Process

The connection between the buzzer and the LED to the RP is the simplest and most straightforward. Piezo buzzer has two pins, one of which is designated with a positive sign (cable violet) and is attached to GPIO8 according to the python coding. The second cable connects the GND designated pin to the GPIO's ground pin. However, the buzzer's operation was initially tested on the breadboard with a multimeter. When the buzzer begins to make noise, it is in good working order and ready to be used in the system. When it comes to the

LED, the researcher must distinguish between the cathode and anode, and then use the same output pin as the buzzer to turn it on.

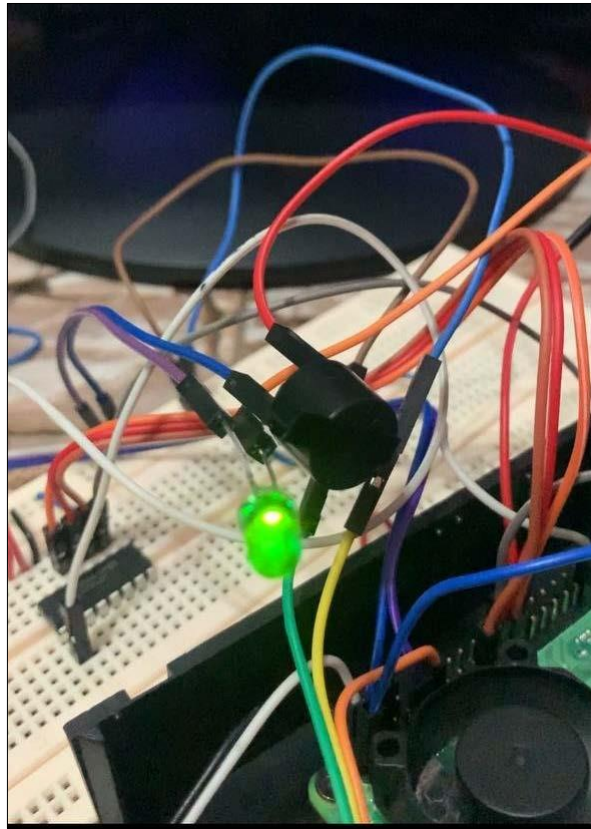


Figure 3.49 Piezo buzzer & LED installation

3.10.4 Analog to Digital Converter MCP3008 Installation

For the first time, connecting the ADC MCP3008 IC may be challenging. This is due to the Raspberry Pi's many pins. For proper wire connections, the researcher frequently consults the IC datasheet pinout. The Vcc pins on pins 16 and 15 are connected to the RPI's 5V power supply, and hence to the ground. Pin 14 and 9 are the two pins that must be grounded. Pins 12 and 11 from the ADC are linked to GPIO 9 (MISO) and GPIO 10 (MOSI), respectively, and pin 10 from the ADC is connected to GPIO 8. The analogue out pin is used to link channel 1's input because the researcher only used one sensor input from MQ-2. To avoid a short circuit, each connection must undergo repeated checks.

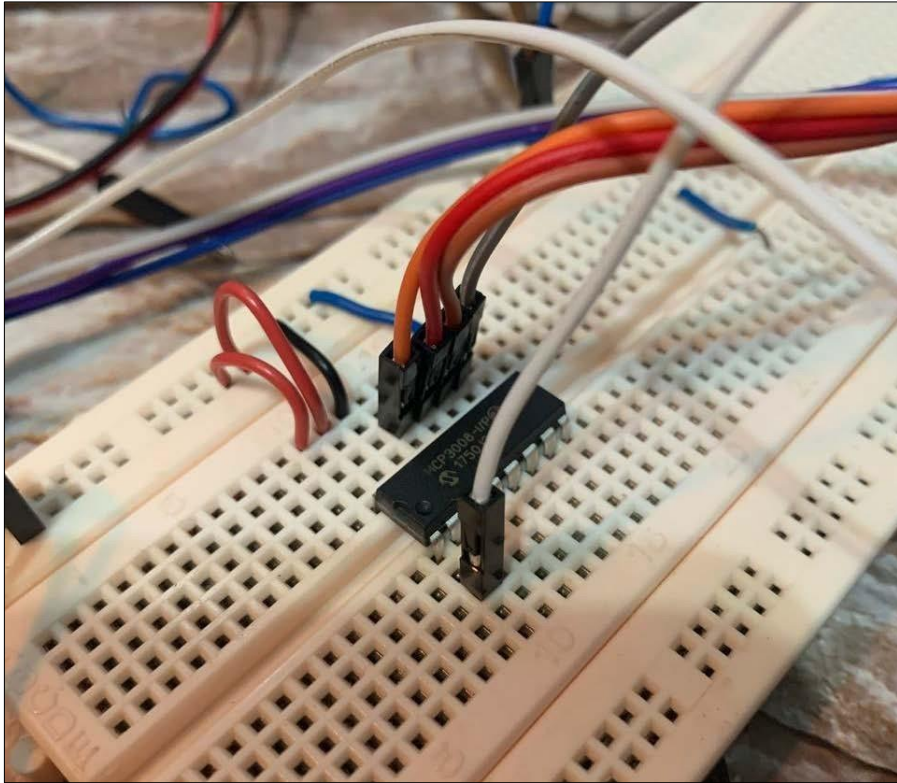


Figure 3.50 ADC MCP3008 IC installation

3.11 Cost Estimation

Table 3.4 Table of Cost Estimation

| No | Item | Unit | Price/Unit |
|----|------------------------------|------|------------|
| 1 | Raspberry Pi 3 Model B | 1 | RM186.00 |
| 2 | DHT11 Temperature Module | 1 | RM4.90 |
| 3 | MQ 2 Smoke Sensor | 1 | RM7.40 |
| 4 | 40 pcs Male to Male Jumper | 1 | RM3.70 |
| 5 | 40 pcs Female to Male Jumper | 1 | RM3.70 |
| 6 | Piezo Buzzer & LED | 1 | RM2.00 |
| 7 | Casing | 1 | RM11.00 |
| 8 | MCP3008 ADC Converter | 1 | RM15.00 |
| 9 | Breadboard | 1 | RM3.90 |
| 10 | Micro SD Card | 1 | RM30.00 |
| 11 | Strip Board | 1 | RM4.00 |
| | Total | | RM271.60 |

3.12 Chapter Summary

For this chapter, all of the subtopic was explained briefly and each of the component chosen rationale explain. Each of the component chosen for this project are well studied to ensure it is suitable and not exceeding the budget cost and meet the longevity and sustainable. Reader can relate the whole system from the block diagram and flow chart from this topic.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

The reader will be briefed on the process and equipment used to construct the system in this chapter. To ensure that the system runs smoothly and flawlessly, it must adhere to the flow and proper execution approach. The researcher will demonstrate how to retrieve the results, as well as a table and information on the system. In this chapter, the researcher will also assemble the test results based on the experiments and tests that were conducted during the project's development. Although some of the components and hardware may not function properly, the student was able to run and diagnose the issue to ensure that the system runs smoothly as butter.

4.2 Prototype Test

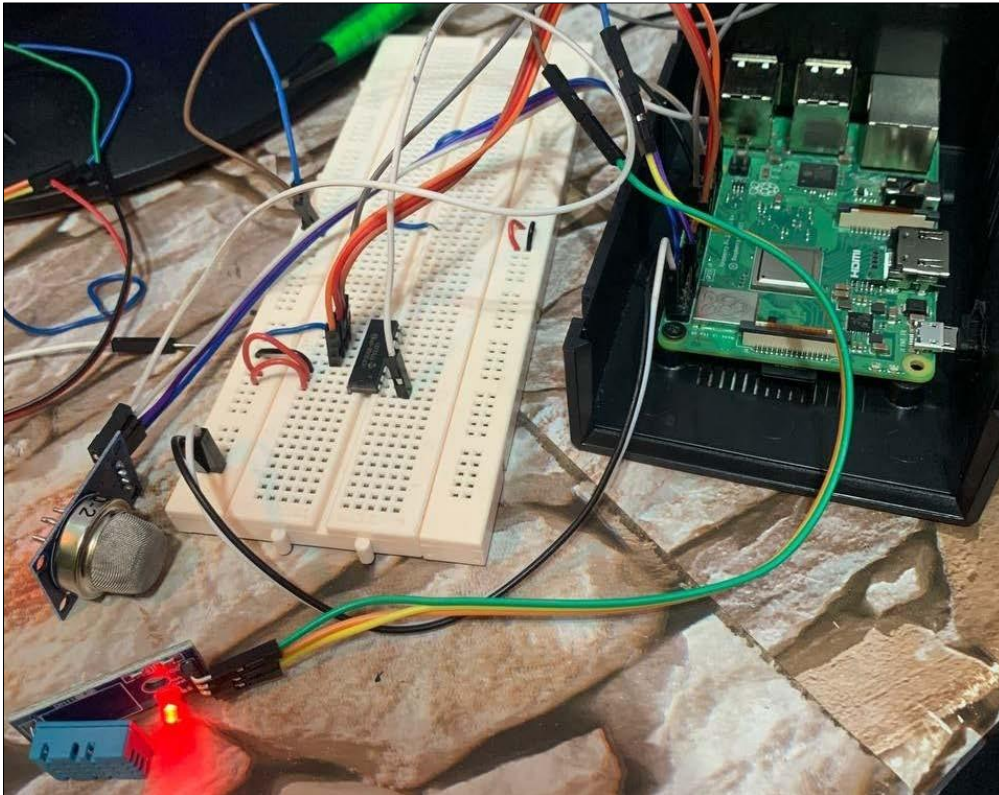


Figure 4.1 Prototype testing in progress

The researcher used an HDMI cable and peripherals like as mouse and keyboard to connect the LED monitor to the Raspberry Pi module in order to test the prototype. After that, the microprocessor is powered through a microUSB cable connected to a 2.1 Ampere current outlet. The buzzer will sound and the LED will turn on for a few seconds when the plug is turned on. On the monitor, the system will begin to boot, indicating that the boot system is operational. The next stage is for the researcher to load and run the coding file (.py) into Python software to run the module in the system and connect the system to the Blynk server. First and foremost, the system must be linked to the internet via WiFi. It will connect to the user's WiFi automatically. Then, on a smartphone with an internet connection, open the Blynk app and select the project to run by tapping the 'play' button. Before researchers run the application, they must first do a Blynk test connection to ensure that the Rpi is connected to the Blynk server, as illustrated in the diagram below.


```

/usr/local/bin/blynk-ctrl -> /usr/local/lib/node_modules/blynk-library/
+ blynk-library@0.5.4
added 1 package from 1 contributor in 8.376s
pi@raspberrypi:~ $ export PATH=$PATH:/opt/nodejs/bin/
pi@raspberrypi:~ $ unset NODE_PATH
pi@raspberrypi:~ $ blynk-client U_7uVADibbCCII6ufe3PmfIwZ-yyFCVZ

Give Blynk a Github star! => https://github.com/vshymanskyi/blynk-library-

Connecting to: blynk-cloud.com 443
SSL authorization...
Connected
Authorized
Blynk ready.

```

Figure 4.2 Successful test connection between

If the connection between the system and the IoT platform is successful, the system will display the Blynk logo on the monitor, followed by current and real-time readings. The humidity percentage will be displayed on the screen because the DHT11 temperature sensor module comes with a humidity sensor by default.

```

*Python 3.7.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/fire alert systems.py =====

please wait...
Humidity = 70.0%      Temperature = 31.0°C
Gas = 0.045 ppm
Safe Area

```

Figure 4.3 Successful connection between Blynk server and generic board

4.3 Effectiveness an alarm system with IoT to reduce time taken to warn the users

The test was conducted to ensure that the system was functional and to assess the effectiveness of the system. The test record is critical to achieving the project's goal, and the results must be delivered on time. Three experiments with varying ranges were set up to measure the time it took for the sensor to detect and send the notification to the user while ringing the buzzer and lighting up the LED. The table below contains all of the results.

Table 4.1 Result of time taken to warn the users

| Test | Test 1 | Test 2 | Test 3 |
|--------------------------------------|---------------|---------------|---------------|
| Range fire and sensor | 2cm | 3cm | 5cm |
| Time for sensor to detect | 1 seconds | 2.3 seconds | 4.5 seconds |
| Time to buzzer & LED | 5 seconds | 8 seconds | 15 seconds |
| Time to send the notification | 5.2 seconds | 8.4 seconds | 15.3 seconds |

The gas type used in the gas sensor test was butane gas from a pocket lighter. The test approach involves simulating a gas leaking scenario with open air, a sensor capable of picking up the gas in all three tests, and a variety of time responses. For the first test, the distance between the sensor and the gas source was increased from 1cm to 3cm to 5cm.

The MQ-2 LED turns on when the sensor detects the gas, indicating that the sensor has detected the gas. The shortest distance resulted in the quickest detection, taking only 1 second to detect, 5 seconds for the buzzer and LED to beep, and 5.2 seconds for the user to get the notification. With only 2.3 seconds for the sensor to detect, 8 seconds for the buzzer and LED to sound, and 8.4 seconds for the user to receive the notification, the medium distance gave a different result.

The detection time was the slowest at the farthest distance, taking only 4.5 seconds, 15 seconds for the buzzer and LED to beep, and 15.3 seconds for the user to receive the notification. Because of the open-air test situation, the results of all tests were varied. It tampered with the gas concentration in order for the sensor to notice it. The test procedure is shown in Figure 4.4 below.

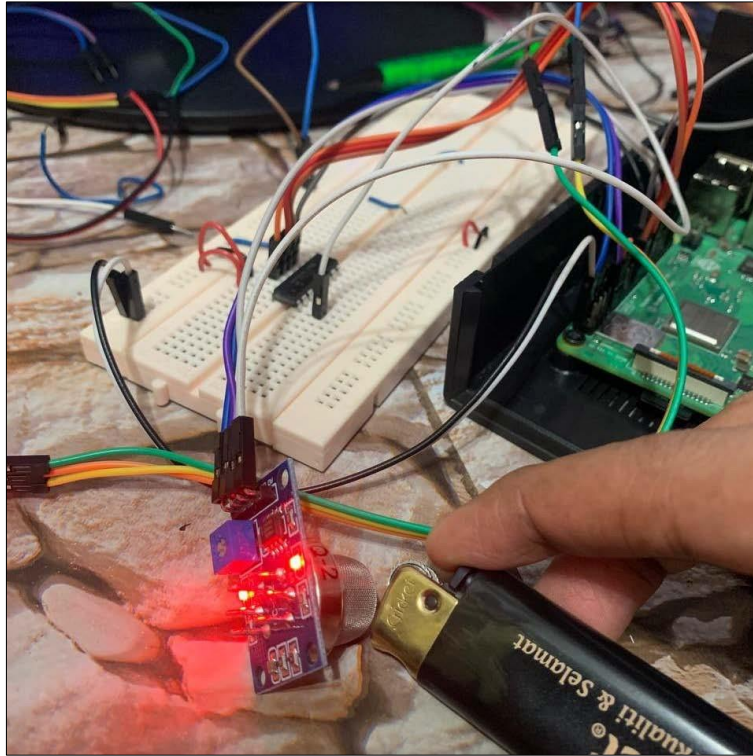


Figure 4.4 Imitating gas leakage scenario using pocket lighter (butane gas)

The user will receive an alert on their smartphone via the IoT platform Blynk server, as indicated in table 4.1. When the generic board connects to the Blynk server, the user will receive a notice on their smartphone with the message "GAS LEAK!"

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Humidity = 59.0%      Temperature = 30.0°C
Gas = 180.0 ppm
Safe Area
Humidity = 59.0%      Temperature = 30.0°C
Gas = 168.0 ppm
Safe Area
Humidity = 59.0%      Temperature = 30.0°C
Gas = 130.0 ppm
Safe Area
Humidity = 59.0%      Temperature = 30.0°C
Gas = 848.0 ppm
GAS LEAKAGE WARNING!
Humidity = 59.0%      Temperature = 30.0°C
Gas = 971.0 ppm
GAS LEAKAGE WARNING!
Humidity = 59.0%      Temperature = 30.0°C
Gas = 985.0 ppm
GAS LEAKAGE WARNING!
Humidity = 59.0%      Temperature = 30.0°C
Gas = 988.0 ppm
GAS LEAKAGE WARNING!
Humidity = 59.0%      Temperature = 30.0°C
Gas = 991.0 ppm
GAS LEAKAGE WARNING!
Humidity = 59.0%      Temperature = 30.0°C
Gas = 992.0 ppm
GAS LEAKAGE WARNING!
Humidity = 58.0%      Temperature = 30.0°C
Gas = 990.0 ppm
GAS LEAKAGE WARNING!
Ln: 4570 Col: 4
```

Figure 4.5 Python real time gas reading and warning

The Python shell that runs on the Raspberry Pi is shown in Figure 4.5. The threshold for the gas to trigger above 300 ppm was determined by the researchers, and the message "GAS LEAKAGE WARNING" was shown. It will print "Safe Area" in addition to the restriction. When the sensor's reading exceeds the threshold value selected, the buzzer and LED on the device will be triggered.

The screenshot below shows a smartphone with an alarm issued during the test. If the user has a strong and reliable internet connection on their smartphone, they will be able to get the notification without being limited by range.

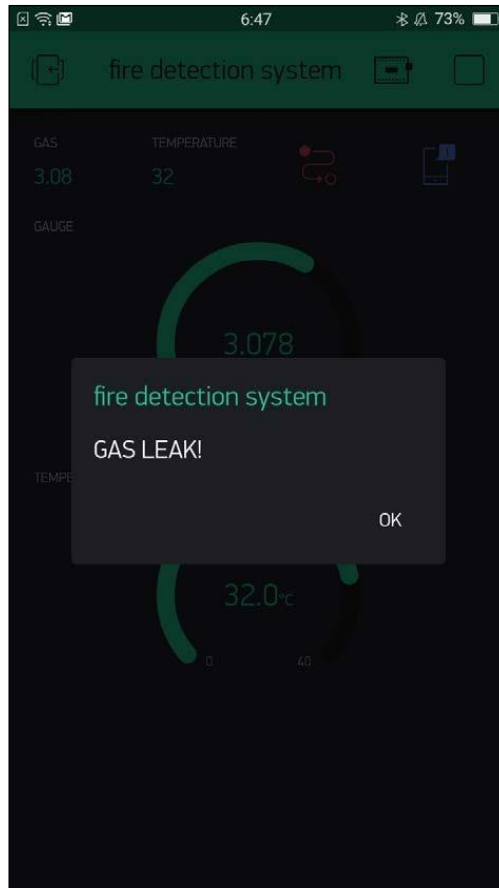


Figure 4.6 Detection on gas alert on the user smartphone

In this case, butane gas is one sort of gas that is readily available. The MQ-2 sensor can detect six different types of gas, including butane. The test, according to the researcher, is sufficient to demonstrate the sensor's and system's functionality. With the system's time response documented, this system has improved greatly in comparison to the existing detector on the market, which has a higher price tag.

4.4 Fire breakout detection test

The pocket light butane lighter from the previous experiment was utilised as the heat source for the temperature sensor experiment. The research method is to simulate an open-air fire breakout, which allows the sensor to detect temperature rises with distinct time responses in all three trials. The first test distance between detectors and heat sources is 2 cm, then 4 cm, and finally 5 cm. To avoid the sensor melting when it is too close, the beginning point is set at 2 cm.

Table 4.2 Result of time taken to warning user based on fire detection

| Test | Test 1 | Test 2 | Test 3 |
|--------------------------------------|---------------|---------------|---------------|
| Range fire and sensor | 2cm | 4cm | 5cm |
| Time to buzzer & LED | 10 seconds | 15 seconds | 17 seconds |
| Time to send the notification | 11 seconds | 16 seconds | 18 seconds |

The maximum range between sensor and firing, according to the above table of results, is 5 cm, while the minimum range is 2 cm. The longer DHT 11 sensor distance takes more time, according to the results. The sensor's range is 2 cm, while the test 1 range is 2 cm.

It took Blynk 11 seconds to convey the message to users. When the distance is raised to 5 cm, test 3 shows that the detecting sensor increases by 7 seconds from test 1. The increasing distance between the sensor module and the heat source was the condition that altered the discrepancies between all trial results. As a result, the time it takes for Blynk to transmit the message is increasing.

Despite the time difference, all three experiments had a much faster response time to notify the tenant. To fix the problem, researchers need to update the sensor to a laser so that it can identify fire outbreaks faster, reducing response time. The researcher set a threshold of 33° in the picture below, which causes the python shell to print "FIRE WARNING."

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Safe Area
Humidity = 49.0%      Temperature = 33.0°C
Gas = 35.0 ppm
Safe Area
Humidity = 48.0%      Temperature = 33.0°C
Gas = 35.0 ppm
Safe Area
Humidity = 48.0%      Temperature = 33.0°C
Gas = 35.0 ppm
Safe Area
Humidity = 48.0%      Temperature = 33.0°C
Gas = 35.0 ppm
Safe Area
Humidity = 48.0%      Temperature = 33.0°C
Gas = 35.0 ppm
Safe Area
Humidity = 46.0%      Temperature = 34.0°C
Gas = 35.0 ppm
FIRE WARNING!
Humidity = 46.0%      Temperature = 34.0°C
Gas = 33.0 ppm
FIRE WARNING!
Humidity = 46.0%      Temperature = 34.0°C
Gas = 33.0 ppm
FIRE WARNING!
Humidity = 45.0%      Temperature = 34.0°C
Gas = 33.0 ppm
FIRE WARNING!
Humidity = 45.0%      Temperature = 34.0°C
Gas = 32.0 ppm
Ln: 4728 Col: 4
```

Figure 4.7 Python temperature real time reading



Figure 4.8 Blynk notification when sensor detected heat over threshold limit

Both the gas and fire breakout tests were accomplished, and the results were satisfactory, with both sensors and notifications arriving on time. The user can take further action to put out the fire and call the fire service if the residence is unattended after being warned by the buzzer, LED, and message.

4.5 Actual Prototype

The prototype has been completed after all of the calibration and assembly procedures have been performed. The prototype is completed by adding a hard plastic casing to keep all of the components safe and neat, based on the researcher's sketch. The wiring is soldered to the strip board, and everything works fine. The case's top is adhered with a hot glue gun. The two sensors on top of the casing detect incoming temperature and gas readings. The warning will be signalled via a buzzer and an LED.



Figure 4.9 Top view of the final year project design

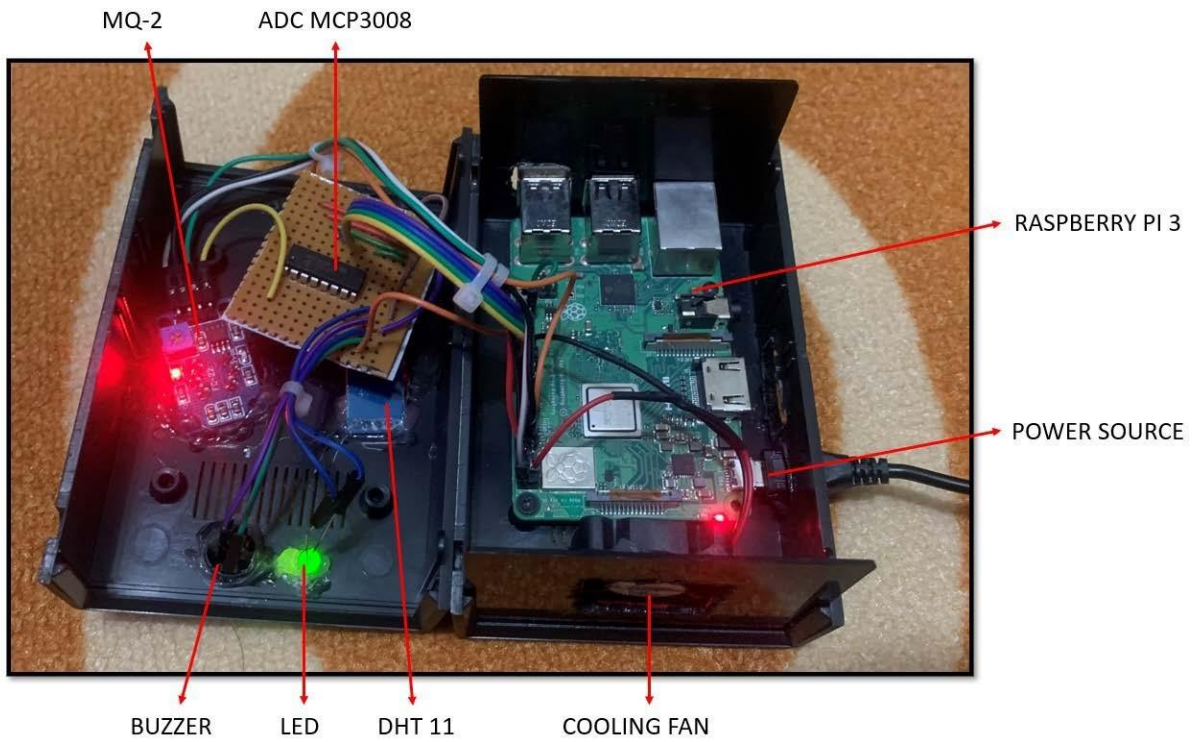


Figure 4.10 Inside view of the project

The Raspberry Pi microcontroller is screwed to the enclosure, as seen in figure 4.8. This will keep the microcontroller in place, even if it is upside down. A cooling fan is also installed on the casing by the user. This will lower the amount of heat created by the Raspberry Pi processor within the casing.



Figure 4.11 Front view of the project

4.6 Chapter Summary

The time response in both tests meets the project's expectations, as summarised in this chapter. Because of the shorter time it takes to warn the occupant, this method has the potential to change the statistics for fire-related deaths in the future.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Introduction

This chapter focused primarily on the conclusions that can be drawn from this research. Furthermore, it makes some suggestions for future research in the fire alarm sector.

5.2 Conclusion

The major purpose of this project is to solve problems that arise in Malaysia as a result of fires and to reduce the number of people who die as a result of fires. To conclude the project's success, the researcher must return to this project goal as an indicator of the system's success. This project is an IoT-based system that can detect a fire in a building and alert the users. Next objective is to create a framework for a building fire protection device using Raspberry Pi and IoT for infinite reachability. This microprocessor is a very capable and reliable small computer to compute a security system in the building whether industrial or home automation. It is also very versatile and can be attach with multiple device and sensor attachment. This objective is successfully obtained by the end of this project development where the Raspberry Pi can deliver an instant alert message no matter the condition, with no range restriction applied. The IoT platform enabled the notification to be deliver to the user, no matter the distance. Finally, this system has been proved can reduce response time taken for user to be notified. As the closing opinion, this project achieved and obtained all objectives set as benchmark. A quick examination of trials and experiments to record possible response time for the system to run from start (sensor detect) to end (user received notification) is shown in Chapter 4. As a result, all of the results were recorded, the outcome was excellent, and everything was completed on time. As a final conclusion, this project met and exceeded all four benchmark objectives. This project is progressing smoothly and according to schedule.

5.3 Recommendations

Future recommendations are critical for the next developer to recognise existing project constraints and problems so that a better system can be developed in the future. This project path has the potential to help society and Malaysia as a whole. There are numerous possibilities for improving future performance and adaptability. The first step is to incorporate an infrared heat detecting camera into the system. The Raspberry Pi includes a lot of expansion ports, which is one of the benefits of utilising it. With an infrared camera sensor installed in the system, users may obtain a better picture of fires and gas leaks in their building. To monitor and analyse the severity, the user can use the internet service to view the image from the IoT platform. The next step is to connect the notification system with the fire department. Even if the user was not notified, the fire department can still receive the notification alert using this approach. Furthermore, it has the potential to significantly lower overall response time. The third suggestion is to set up a system for evaluating false alarms. This system should assess the sensor reading and determine whether the sensor's alert is true or false. A true alert and a false alarm might have different outcomes, such as saving someone's life or causing death. Finally, a real-time monitoring camera should be installed in the system as a future upgrade. It can be used as a surveillance camera to monitor and provide a real-time view of the present condition. To boost security, most home and business automation systems now include video cameras.

5.4 Chapter Summary

To summarise the chapter, the project's goals were met and the project was completed effectively. The project's success is evidenced by the achievement of the result and production.

REFERENCES

- (No Title). (n.d.). Retrieved May 19, 2021, from <https://docs.blynk.cc/>
- Ipcs MQ-2 MQ2 Gas Sensor Module Smoke Methane Butane Detection for Arduino | eBay.* (n.d.). Retrieved June 7, 2021, from <https://www.ebay.com/itm/183649183685>
- Al-zaheiree, A. A., Al-zubaidi, Y. A. T., Gaikwad, S. S., & Kamat, R. K. (2020). Advanced Air Pollution Detection using IOT and Raspberry PI. *International Journal of Recent Technology and Engineering*, 8(5), 5390–5394. <https://doi.org/10.35940/ijrte.e5931.018520>
- Alamgir, N. (2020). *COMPUTER VISION BASED SMOKE AND FIRE DETECTION FOR OUTDOOR ENVIRONMENTS.*
- Blynk Program with Multiple WiFi Networks | FactoryForward.* (n.d.). Retrieved June 7, 2021, from <https://www.factoryforward.com/blynk-program-multiple-wifi-networks/>
- Bogdanchikov, A., Zhaparov, M., & Suliyev, R. (2013). Python to learn programming. *Journal of Physics: Conference Series*, 423(1). <https://doi.org/10.1088/1742-6596/423/1/012027>
- Burange, A. W., & Misalkar, H. D. (2015). Review of Internet of Things in development of smart cities with data management & privacy. *Conference Proceeding - 2015 International Conference on Advances in Computer Engineering and Applications, ICACEA 2015*, 189–195. <https://doi.org/10.1109/ICACEA.2015.7164693>
- Buzzer 5V.* (n.d.). Retrieved June 7, 2021, from <https://www.twinschip.com/Buzzer-5V>
- Can a buzzer function as a switch in a circuit? - Electronic Guidebook.* (n.d.). Retrieved May 19, 2021, from <https://electronicguidebook.com/can-a-buzzer-function-as-a-switch-in-a-circuit/>
- Carlsen, J. (n.d.). *What Does a Carbon Monoxide Detector Do and How Does it Work? | SafeWise.* Retrieved May 17, 2021, from <https://www.safewise.com/home-security-faq/carbon-monoxide-detector/>

- Configuration - Raspberry Pi Documentation*. (n.d.). Retrieved June 7, 2021, from <https://www.raspberrypi.org/documentation/configuration/>
- FrontPage - Raspbian*. (n.d.). Retrieved May 18, 2021, from <http://www.raspbian.org/>
- GitHub - blynkkk/lib-python: Blynk IoT library for Python and Micropython*. (n.d.). Retrieved February 2, 2022, from <https://github.com/blynkkk/lib-python>
- GLOBAL FOREST FIRE ASSESSMENT 1990-2000 - FRA WP 55*. (n.d.). Retrieved May 19, 2021, from <http://www.fao.org/3/ad653e/ad653e46.htm>
- Govardhini, S., & Sumalatha, G. (n.d.). *WORKING PRINCIPLES OF RASPBERRY PI AND ARDUINO UNO IN WINDOWS*. Retrieved May 18, 2021, from <https://www.raspberrypi.org/downloads/>
- Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660. <https://doi.org/10.1016/j.future.2013.01.010>
- H., Z., A., H., & M., M. (2015). Internet of Things (IoT): Definitions, Challenges and Recent Research Directions. *International Journal of Computer Applications*, 128(1), 37–47. <https://doi.org/10.5120/ijca2015906430>
- How to Set Up the DHT11 Humidity Sensor on the Raspberry Pi - Circuit Basics*. (n.d.). Retrieved June 7, 2021, from <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/>
- How UV, IR and Imaging Detectors Work*. (2020, October 12). Teledyne Gas and Flame Detection. <https://www.azosensors.com/article.aspx?ArticleID=815>
- Idris, F., Hashim, N., FauzanKadmin, A., & Yee, L. B. (2019). Intelligent fire detection and alert system using labVIEW. *International Journal of Electrical and Computer Engineering*, 9(3), 1842–1849. <https://doi.org/10.11591/ijece.v9i3.pp1842-1849>
- Introduction to Proteus - The Engineering Projects*. (n.d.). Retrieved May 19, 2021, from <https://www.theengineeringprojects.com/2020/01/introduction-to-proteus.html>
- Khan, M. N. A., Tanveer, T., Khurshid, K., Zaki, H., & Zaidi, S. S. I. (2019). Fire detection system using raspberry pi. *2019 International Conference on Information Science and Communication Technology, ICISCT 2019*.

<https://doi.org/10.1109/CISCT.2019.8777414>

Madakam, S., Ramaswamy, R., & Tripathi, S. (2015). Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*, 03(05), 164–173.

<https://doi.org/10.4236/jcc.2015.35021>

Majority of structural fires can be avoided | The Edge Markets. (n.d.). Retrieved May 19, 2021, from <https://www.theedgemarkets.com/article/majority-structural-fires-can-be-avoided>

Maksimović, M., Vujović, V., Davidović, N., Milošević, V., & Perišić, B. (2014). Raspberry Pi as Internet of Things hardware : Performances and Constraints. *Design Issues*, 3(JUNE), 8.

MCP3008 ADC IC Pinout, Features, Equivalent & Datasheet. (n.d.). Retrieved December 20, 2021, from <https://components101.com/ics/mcp3008-adc-pinout-equivalent-datasheet>

Media's, E., . S., & Rif'an, M. (2019). Internet of Things (IoT): BLYNK Framework for Smart Home. *KnE Social Sciences*, 3(12), 579. <https://doi.org/10.18502/kss.v3i12.4128>

Nayyar, A., & Puri, V. (2015). Raspberry Pi-A Small , Powerful , Cost Effective and Efficient Form Factor Computer : A Review *International Journal of Advanced Research in Raspberry Pi- A Small , Powerful , Cost Effective and Efficient Form Factor Computer : A Review. International Journal of Advanced Research in Computer Science and Software Engineering* 5(12), 5(12), 720–737.

Noor, N. Q. M., Jamaludin, A. A., Azizan, A., Abas, H., Kamardin, K., & Yakub, M. F. M. (2018). Arduino vs Raspberry Pi vs Micro Bit: Platforms for Fast IoT Systems Prototyping. *Open International Journal of Informatics (OIJI)*, 6(1), 96–107.

Patil, P. N. (2016). A comparative analysis of Raspberry pi Hardware with Adruino, Phidgets, Beaglebone black and Udoo Pankaj Naganath Patil. *International Research Journal of Engineering and Technology*, 1595–1600. www.irjet.net

Petrov, K. (2012). *Klim Petrov DETERMINING THE BEST LOCATION OF SMOKE SENSOR IN OFFICE ROOM Building Services Engineering Determining the best location of smoke sensor in office room.*

Raspberry Pi Model Comparison Table. (n.d.). Retrieved June 7, 2021, from <https://www.rs->

online.com/designspark/raspberry-pi-model-comparison-table

Raspberry Pi Tutorial: How to Use a Buzzer : 4 Steps - Instructables. (n.d.). Retrieved June 7, 2021, from <https://www.instructables.com/Raspberry-Pi-Tutorial-How-to-Use-a-Buzzer/>

Reddy, A. R., Sai, M. V. R., Vamsi, J., & Raju, C. S. K. (2020). *Automatic Fire Detection and Alert System. XII(V)*, 2109–2117.

Sazali, L. (2019). Redundant for Fire Monitoring System Using Raspberry Pi and Arduino Uno. *Journal of Computing Technologies and Creative Content (JTec)*, 4(1). <http://www.jtec.org.my/index.php/JTeC/article/view/188>

Sethi, P., & Sarangi, S. R. (2017). Internet of Things: Architectures, Protocols, and Applications. In *Journal of Electrical and Computer Engineering* (Vol. 2017). Hindawi Publishing Corporation. <https://doi.org/10.1155/2017/9324035>

Sharma, A., Khan, F., Sharma, D., Student, F. Y., & Area, S. I. (2020). Python : The Programming Language of Future. *International Journal of Innovative Research in Technology*, 6(12), 115–118.

Srivastava, D., Kesarwani, A., & Dubey, S. (2018). Measurement of Temperature and Humidity by using Arduino Tool and DHT11. *International Research Journal of Engineering and Technology*, 05(12), 876–878.

Suparman, M. A. Bin, & Jong, S. L. (2019). Automatic smoke detection system with favoriot platform using internet of things (IoT). *Indonesian Journal of Electrical Engineering and Computer Science*, 15(2), 1102–1108. <https://doi.org/10.11591/ijeecs.v15.i2.pp1102-1108>

Thakre, P. (2019). *Monitoring System of Different Air Quality Parameter using Raspberry PI and Web Of Thing.* 6(1), 627–631.

Tony Dicola. (n.d.). *MCP3008 / Raspberry Pi Analog to Digital Converters / Adafruit Learning System.* Retrieved December 13, 2021, from <https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008>

Tun, M. Z., & Myint, H. (2020). Arduino based Fire Detection and Alarm System Using Smoke Sensor. *International Journal of Advances in Scientific Research and Engineering*, 06(04), 89–94. <https://doi.org/10.31695/ijasre.2020.33792>

- Upton, E. (2012). *Meet the Raspberry Pi*. Wiley.
<https://catalogue.library.cern/literature/7tga4-2kt26>
- Vatsal, S., & Bhavin, M. (2017). Using Raspberry Pi To Sense Temperature and Relative Humidity. *International Research Journal of Engineering and Technology (IRJET)*, 4(2), 380–385. <https://irjet.net/archives/V4/i2/IRJET-V4I276.pdf>
- Weyrich, M., & Ebert, C. (2016). Reference architectures for the internet of things. *IEEE Software*, 33(1), 112–116. <https://doi.org/10.1109/MS.2016.20>
- What is Arduino? | Arduino*. (n.d.). Retrieved May 18, 2021, from <https://www.arduino.cc/en/Guide/Introduction>
- What is Virtual Network Computing (VNC)? - Definition from Techopedia*. (n.d.). Retrieved December 13, 2021, from <https://www.techopedia.com/definition/3308/virtual-network-computing-vnc>
- Wijaya, R., Christian, V., Yuwananda, Y. S., & Alexander, I. (2019). Temperature and humidity monitoring system in server room using raspberry Pi. *International Journal of Scientific and Technology Research*, 8(10), 3075–3078.
- Wu, M., Lu, T. J., Ling, F. Y., Sun, J., & Du, H. Y. (2010). Research on the architecture of Internet of Things. *ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings*, 5.
<https://doi.org/10.1109/ICACTE.2010.5579493>

APPENDIX A PYTHON CODING

```
#Libraries
import BlynkLib
import Adafruit_DHT
import RPi.GPIO as GPIO
import time
from time import sleep
GPIO.setwarnings(False)

# SPI port on the ADC to the Cobbler
SPICLK = 11
SPIMISO = 9
SPIMOSI = 10
SPICS = 8
mq2_dpin = 26
mq2_apin = 0
CE1 = 7
BLYNK_AUTH = 'g0HRVGNzbGk4k3i9rRgIlx-bQ2iTY5uO'
sensor_type = Adafruit_DHT.DHT11
sensor_pin = 17

# GPIO.setmode(GPIO.BOARD)

# Initialize Blynk
blynk = BlynkLib.Blynk(BLYNK_AUTH)

#port init
def init():
    GPIO.setwarnings(False)
    GPIO.cleanup()           #clean up at the end of
the script
    GPIO.setmode(GPIO.BCM)   #to specify which
pin numbering system
```

```

# set up the SPI interface pins
    GPIO.setup(SPIMOSI, GPIO.OUT)
    GPIO.setup(SPIMISO, GPIO.IN)
    GPIO.setup(SPICK, GPIO.OUT)
    GPIO.setup(SPICS, GPIO.OUT)

GPIO.setup(mq2_dpin,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
    GPIO.setup(CE1, GPIO.OUT, initial=GPIO.LOW) #LED
& Buzzer

#read SPI data from MCP3008(or MCP3204) chip,8 possible
adc's (0 thru 7)
def readadc(adcnum, clockpin, mosipin, misopin, cspin):
    if ((adcnum > 7) or (adcnum < 0)):
        return -1
    GPIO.output(cspin, True)

    GPIO.output(clockpin, False) # start clock low
    GPIO.output(cspin, False) # bring CS low

    commandout = adcnum
    commandout |= 0x18 # start bit + single-ended bit
    commandout <<= 3 # we only need to send 5 bits
here
    for i in range(5):
        if (commandout & 0x80):
            GPIO.output(mosipin, True)
        else:
            GPIO.output(mosipin, False)
        commandout <<= 1
        GPIO.output(clockpin, True)
        GPIO.output(clockpin, False)

    adcout = 0
    # read in one empty bit, one null bit and 10 ADC
bits

```

```

for i in range(12):
    GPIO.output(clockpin, True)
    GPIO.output(clockpin, False)
    adcout <<= 1
    if (GPIO.input(misopin)):
        adcout |= 0x1

    GPIO.output(cspin, True)

    adcout >>= 1      # first bit is 'null' so drop
it
    return adcout

#main loop
def main():
    init()
    print("please wait...")
    time.sleep(5)
    while True:
        COlevel=readadc(mq2_apin,          SPICLK,
SPIMOSI, SPIMISO, SPICS)
        gas=GPIO.input(SPICS)
        humidity,          temperature      =
Adafruit_DHT.read_retry(sensor_type, sensor_pin)
        print(('Humidity = {:.1f}%\tTemperature
= {:.1f}°C'.format(humidity, temperature)))
        print("Gas                               =          ")
+str("%.1f"%(COlevel))+i" ppm")

        if (COlevel)>=300:
            print("GAS LEAKAGE WARNING!")
            time.sleep(0.5)
            GPIO.output(CE1, GPIO.HIGH) #
Turn on LED & Buzzer
            print()

        elif temperature>33:

```

```

Turn on LED & Buzzer
GPIO.output(CE1, GPIO.HIGH) #

print("FIRE WARNING!")
print()

else:
    print("Safe Area")
    print()
GPIO.output(CE1, GPIO.LOW) #
Turn off LED & Buzzer

    blynk.virtual_write(1,
' {:.1f}'.format((COlevel)))
    blynk.virtual_write(2,
' {:.1f}'.format(temperature))

    blynk.run()

if __name__ == '__main__':
    try:
        main()
        pass
    except KeyboardInterrupt:
        pass

GPIO.cleanup()

```

APPENDIX B MQ-2 SENSOR DATASHEET

MQ-2 Semiconductor Sensor for Combustible Gas

Sensitive material of MQ-2 gas sensor is SnO₂ which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-2 gas sensor has high sensitivity to LPG, Propane and Hydrogen, also could be used to Methane and other combustible steam, It is with low cost and suitable for different application.

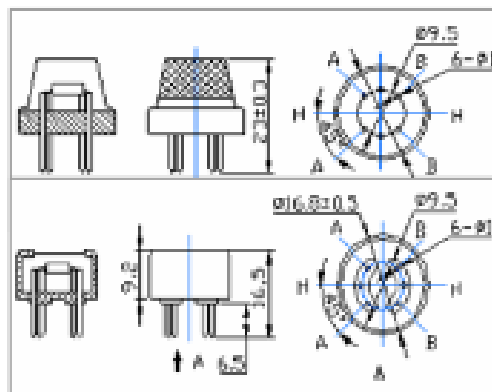
Character

- * Good sensitivity to Combustible gas in wide range
- * High sensitivity to LPG, Propane and Hydrogen
- * Long life and low cost
- * Simple drive circuit

Application

- * Domestic gas leakage detector
- * Industrial Combustible gas detector
- * Portable gas detector

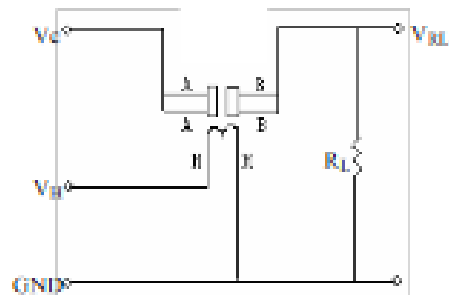
Configuration



Technical Data

| | | | |
|------------------------|-----------------------|-----------------------------------------------------------|---------------------------------------------------------------------|
| Model No. | | MQ-2 | |
| Sensor Type | | Semiconductor | |
| Standard Encapsulation | | Bakelite (Black Bakelite) | |
| Detection Gas | | Combustible gas and smoke | |
| Concentration | | 300-10000ppm (Combustible gas) | |
| Circuit | Loop Voltage | V _L | ±24V DC |
| | Heater Voltage | V _H | 5.0V±0.2V AC or DC |
| | Load Resistance | R _L | Adjustable |
| Character | Heater Resistance | R _H | 310±30 (Room Tem.) |
| | Heater consumption | P _H | ±500mW |
| | Sensing Resistance | R _s | 2KΩ-20KΩ (in 2000ppm C ₂ H ₆) |
| | Sensitivity | S | R _s (in air)/R _s (1000ppm Isobutane)≥5 |
| | Slope | α | ±0.5(R _{s2000ppm} /R _{s2000ppm} CH ₄) |
| Condition | Tem. Humidity | 20°C±2°C, 65%±5%RH | |
| | Standard test circuit | V _L : 5.0V±0.1V, V _H : 5.0V±0.1V | |
| | Preheat time | Over 48 hours | |

Basic test loop

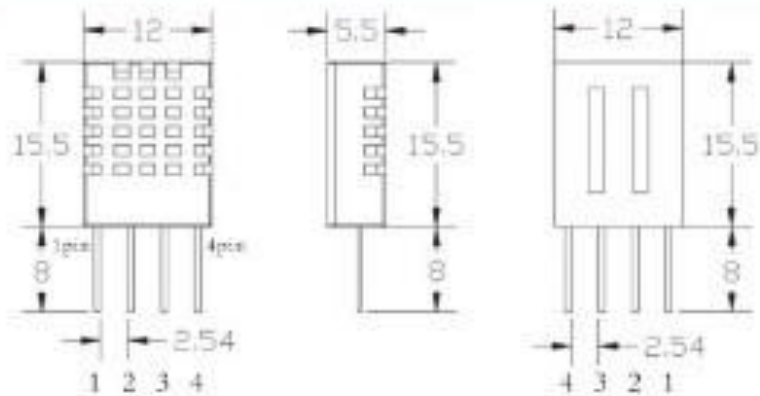


The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage (V_H) and test voltage (V_C). V_H used to supply certified working temperature to the sensor, while V_C used to detect voltage (V_{RL}) on load resistance (R_L) whom is in series with sensor. The sensor has light polarity, V_C need DC power. V_C and V_H could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable R_L value is needed:
Power of Sensitivity body (P_s):
 $P_s = V_C^2 \times R_s / (R_s + R_L)^2$

APPENDIX C DHT11 SENSOR DATASHEET



Temp., Humidity & Dew point measurement experts



5. Parameters

Relative Humidity

Resolution : 16Bit

Repeatability : $\pm 1\%RH$

Accuracy : 25°C $\pm 5\%RH$

Interchangeability : Fully interchangeable

Response time : 1/e (63%)25°C 6s

1m/s Air 6s

Hysteresis : $< \pm 0.3\%RH$

Long-term stability : $< \pm 0.5\%RH/yr$

Temperature

Resolution : 16Bit

Repeatability : $\pm 1^\circ C$

Accuracy : 25°C $\pm 2^\circ C$

Response time : 1/e (63%) 10S

Electrical Characteristics

Power supply : DC 3.3 ~ 5.5V

APPENDIX D
MCP3008 ADC DATASHEET



MCP3004/3008

2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface

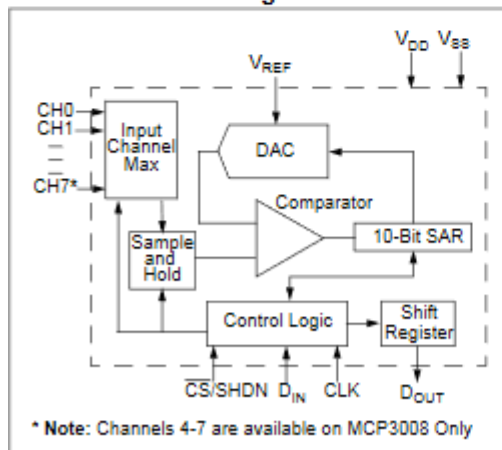
Features

- 10-bit resolution
- ± 1 LSB max DNL
- ± 1 LSB max INL
- 4 (MCP3004) or 8 (MCP3008) input channels
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V - 5.5V
- 200 ksp/s max. sampling rate at $V_{DD} = 5V$
- 75 ksp/s max. sampling rate at $V_{DD} = 2.7V$
- Low power CMOS technology
- 5 nA typical standby current, 2 μA max.
- 500 μA max. active current at 5V
- Industrial temp range: $-40^{\circ}C$ to $+85^{\circ}C$
- Available in PDIP, SOIC and TSSOP packages

Applications

- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

Functional Block Diagram



Description

The Microchip Technology Inc. MCP3004/3008 devices are successive approximation 10-bit Analog-to-Digital (A/D) converters with on-board sample and hold circuitry. The MCP3004 is programmable to provide two pseudo-differential input pairs or four single-ended inputs. The MCP3008 is programmable to provide four pseudo-differential input pairs or eight single-ended inputs. Differential Nonlinearity (DNL) and Integral Nonlinearity (INL) are specified at ± 1 LSB. Communication with the devices is accomplished using a simple serial interface compatible with the SPI protocol. The devices are capable of conversion rates of up to 200 ksp/s. The MCP3004/3008 devices operate over a broad voltage range (2.7V - 5.5V). Low-current design permits operation with typical standby currents of only 5 nA and typical active currents of 320 μA . The MCP3004 is offered in 14-pin PDIP, 150 mil SOIC and TSSOP packages, while the MCP3008 is offered in 16-pin PDIP and SOIC packages.

Package Types

