**IoT BASED GAS LEAKAGE DETECTION & SMART ALERTING SYSTEM**

**A PROJECT REPORT**

***Submitted by***

**SHEIK FAARUK.M**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

**SONA COLLEGE OF TECHNOLOGY, SALEM**

**(AUTONOMOUS)**

**ANNA UNIVERSITY : CHENNAI 600 025**

**APRIL 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report “**IoT BASED GAS LEAKAGE DETECTION & SMART ALERTING SYSMTE**” is the bonafide work of “SHEIK FAARUK M(Reg.No:1517106078)”who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.J.Akilandeswari Dr.J.SenthilKumar**

HEAD OF THE DEPARTMENT SUPERVISOR

Professor Assistant Professor

Department of Information Technology Department of Information Technology

Sona College of Technology, Sona College of Technology,

Salem-636 005. Salem-636 005.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# ACKNOWLEDGEMENT

First and foremost, we thank to **power of almighty** for showing us inner peace and for all blessings. Special gratitude to our parents, for showing their support and love always.

We express our sincere thanks to Chairman **Sri.C.Valliappa** and Principal

**Dr.S.R.R.SENTHIL KUMAR** for providing adequate facilities to complete the

project.

We are immensely grateful to Head of Information Technology, **Dr.J.Akilandeswari** for the continues encouragement to complete the project.

We express our heartfelt thanks to our project supervisor **Dr.J.SenthilKumar** and project coordinators **Dr.P.Iyyanar, Dr.P.Ilanchezhian and Dr.P. Shanmugaraja** for their valuable guidance and fruitful discussions throughout the course of the project work.

We feel proud in sharing this success with all our staff members and friends

who helped directly or indirectly in completing this project successfully

**ABSTRACT**

The fundamental goal of the venture is to identify the smoke and gas spillage utilizing ESP8266.Using MQ2 sensor to distinguish the gas spillage and to recognize the smoke . Furthermore, it interface with the Thinkspeak (cloud) gather the information and put away in database.Then It access by Programming interface keys to show the data in vacillate application.The ripple application is created by the dart language. In this venture to discover smoke or gas spillage and illuminate to the client by vacillate application and furthermore dynamic the Fumes Fan to emulat the spillage gas to outside. Utilizing MQ2 sensor to identify the gas spillage likewise to distinguish the smoke additionally MQ2 sensor can recognize or gauge gasses like LPG, Liquor, Propane, Hydrogen, CO and even methane .In this project, we will learn about Gas Level and smoke level On Internet Using ESP8266 & Gas Sensor Module, i.e MQ2. We will measure the quantity of gas in percentage and send it over the internet using the thingspeak server. With this system, the data can be monitored remotely staying at any part of the world. We just need gas/smoke/LPG sensor like MQ2/MQ3/MQ5/MQ7/MQ135 that is directly connected to Nodemcu ESP8266-12E Module. ThingSpeak**is an open-source** Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network.

Table of Contents

**TITLES PAGE NO**

[ABSTRACT 4](#_Toc70066620)

[ACKNOWLEDGEMENT 3](#_Toc70066621)

[CHAPTER – 1 7](#_Toc70066622)

[1.1 INTRODUCTION 7](#_Toc70066623)

[1.2 WHAT IOT CAN DO? 8](#_Toc70066624)

[1.3 PROBLEM DESCRIPTION 9](#_Toc70066625)

[Literature Survey 10](#_Toc70066626)

[1.4 MODULES : 10](#_Toc70066627)

[1.4.1 Gas Leakage Detection 11](#_Toc70066628)

[1.4.2.Send the data to the Thingspeak: 13](#_Toc70066629)

[1.4.3.Making Alert Alarm in Flutter Application 15](#_Toc70066632)

[CHAPTER –2 16](#_Toc70066633)

[2.1 Requirements 16](#_Toc70066634)

[2.1.1 ESP8266-Nodemcu: 16](#_Toc70066635)

[NodeMCU ESP8266 Specifications & Features 20](#_Toc70066636)

[2.1.2 MQ2: 23](#_Toc70066651)

[2D model of MQ-2 Gas sensor*:* 27](#_Toc70066652)

[2.1.3 Arduino 28](#_Toc70066653)

[2.1.4 Thinkspeak cloud 32](#_Toc70066655)

[2.1.5 Flutter 36](#_Toc70066656)

[2.2 Source Code of Arduino : 40](#_Toc70066657)

[2.2.1 Gas\_ESP8266.ino 41](#_Toc70066658)

[2.3 Source Code of Flutter Application: 45](#_Toc70066659)

[2.3.1 generated\_plugin\_registrant.dart: 45](#_Toc70066660)

[2.3.2 Main.dart: 46](#_Toc70066661)

[2.3.3 Gas\_predicition.dart: 47](#_Toc70066662)

[2.4 Source Code of Flutter web-Application 54](#_Toc70066663)

[2.4.1 Index.html 54](#_Toc70066664)

[2.4.2 Main.json 58](#_Toc70066665)

[CHAPTER-3 59](#_Toc70066666)

[CONCLUSION 59](#_Toc70066667)

[REFERENCES 60](#_Toc70066668)

# 

## 

# CHAPTER – 1

## 1.1 INTRODUCTION

Nowadays, there is an increase in the number of accidents that happen in the world. As the population is increasing, there is the number of cars increasing on the road that contributes to severe accidents that happen daily. Around 40 per cent of accidents contribute to the loss of many lives. Mostly, the growing countries are being targeted by the day to day gas leakage accidents. Gas leak accidents either cause damage through an explosion or through breathing in the gas for a period of time. There are three different levels of gas leaks. Grade 1 is the most serious type of gas leak. The industry standard is to repair these immediately. In a grade 1 leak, the gas is already ignited, or is likely to ignite soon. If you can hear a leak, smell the super smell, or see evidence of the leak, it is a grade 1 leak.

In this undertaking, we will find out about Gas Level Screen On Web Utilizing ESP8266 and Gas Sensor Module, i.e MQ2. We will gauge the amount of gas in rate and send it over the web utilizing the thingspeak worker. With this framework, the information can be observed distantly remaining at any piece of the world.We simply need gas/smoke/LPG sensor like MQ2/MQ3/MQ5/MQ7/MQ135 that is straightforwardly associated with Nodemcu ESP8266-12E Module. ThingSpeak is an open-source Web of Thing7s (IoT) application and Programming interface to store and recover information from things utilizing the HTTP convention over the Web or by means of a Neighborhood

## 1.2 WHAT IOT CAN DO?

The principle objective of the task is to recognize the smoke and gas spillage utilizing ESP8266 .Utilizing MQ2 sensor to distinguish the gas spillage and to identify the smoke . Furthermore, it interface with the Thinkspeak (cloud) gather the information and put away in data set When gas is recognized Nodemcu send message to client and mood killer the gas and furthermore consequently ON the fumes fan to imitate the gas to outside

Grade 2 and grade 3 leaks are less serious. However, they should be repaired as quickly as possible, within at least six months.

Some form of negligence causes almost all gas leaks. The most causes of gas leaks are:

Defective equipment or pipes

Poorly maintained equipment

Poorly maintained pipes and connections

Failure to properly inspect underground gas lines

Damage to service lines or underground pipelines

Lax safety environment

Often natural gas leak accidents occur at chemical plants and natural gas refineries. If a leak is severe enough it can impact entire neighborhoods. Lines can easily become damaged during construction if the proper precautions are not taken.

## 1.3 PROBLEM DESCRIPTION

Many accidents occur in day to day life like explosion because of LPG leakage. Major harm is caused, if gas leakage is not detected early. But now we can detect the gas leakage using the MQ2 gas sensor. In this IOT gas leakage detector, device will get connected to WIFI, the minimum and maximum parameter can be set accordingly. Such IOT as well as Arduino based gas leakage detector systems can be installed in homes, hotels LPG gas storage areas. In this LPG gas detector system senses the LPG gas using MQ2 gas sensor. This device will continuously monitor the level of LPG gas present in the air. While monitoring, if the value of LPG gas in air is within the set limit then the send the information to the thinkspeak and then make alarm alert in flutter application .And whenever the gas exceeds above the predefined limit than the RGB LED will glow red and simultaneously solenoid value will turn off and update it over IOT. This Arduino and IOT project will help in detecting gas leakage in the surrounding.

Following are the major reasons of accidents for gas leakage: Small gas leaks may not have a smell or other physical signs. However, if there is a gas leak in the home, a person may notice:

1. the smell of sulfur or rotten eggs
2. a hissing or whistling sound near a gas line
3. a white cloud or dust cloud near a gas line
4. bubbles in water
5. a damaged gas pipe
6. dead houseplants Also, gas bills may be higher than normal, as gas will be escaping from gas lines or appliances into the house.

# Literature Survey

In the proposed system we have designed “IOT based Smart Gas Monitoring System”. This proposed system aims to detect the economic fuels like petroleum, liquid petroleum gas, alcohol etc and allows a provision for controlling the gas leakage by closing the valve automatically. The next feature of the topic is to ensure the booking of gas cylinder from gas agency. For both the functioning the sensors detect the leaked gas from the sensor and send it to the internet .by programming on the internet, the sensed signal is directed to the android app by using the android app we give the signal for switching off gas valve from distant place. So it redirects again to the internet and close the gas cylinder valve through IOT. The problem of gas wastage could also be avoided using this system.

## 1.4 MODULES :

**There are three modules in this detection system and they are:**

1. Gas Leakage detection
2. Send the data to the Thingspeak
3. Making Alert Alarm in Flutter Application

### 1.4.1 Gas Leakage Detection

In this project, we will learn about **Gas Monitor On Internet Using ESP8266 & Gas Sensor Module**, i.e MQ2. We will measure the quantity of gas in percentage and send it over the internet using the **thingspeak server**. With this system, the data can be monitored remotely staying at any part of the world. We just need **gas/smoke/LPG sensor** like **MQ2/MQ3/MQ5/MQ7/MQ135** that is directly connected to **Nodemcu ESP8266-12E Module**.

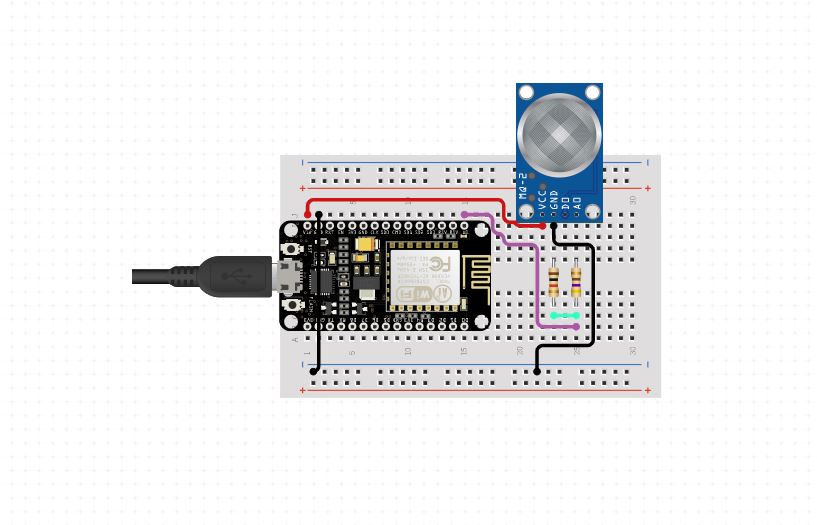
**Smoke and gas leakage detectors** are very useful in detecting smoke or fire in buildings, and so are the important safety parameters in order to prevent disasters. Bursting cylinders and accidental fires have caused lots of harm to the economies in the past.

This circuit triggers the alert system when smoke or gas leakage is detected. The circuit mainly uses the **MQ2 Smoke/Gas sensor and esp8266** to detect and smoke and gas leak. This **MQ2 gas sensor** is sensible to LPG, Alcohol, and Methane etc.It detects the presence of a dangerous LPG leak in your car or in a service station, storage tank environment.

The sensor has excellent sensitivity combined with the quick response time. The sensor can also sense iso-butane, propane, LNG, and cigarette smoke. If the LPG sensor senses any gas leakage from storage the output of this sensor goes low. This low signal is monitored by the **microcontroller** and sends the alert to **thinkspeak and alert**  the person.

#### ****Circuit Diagram & Connections****

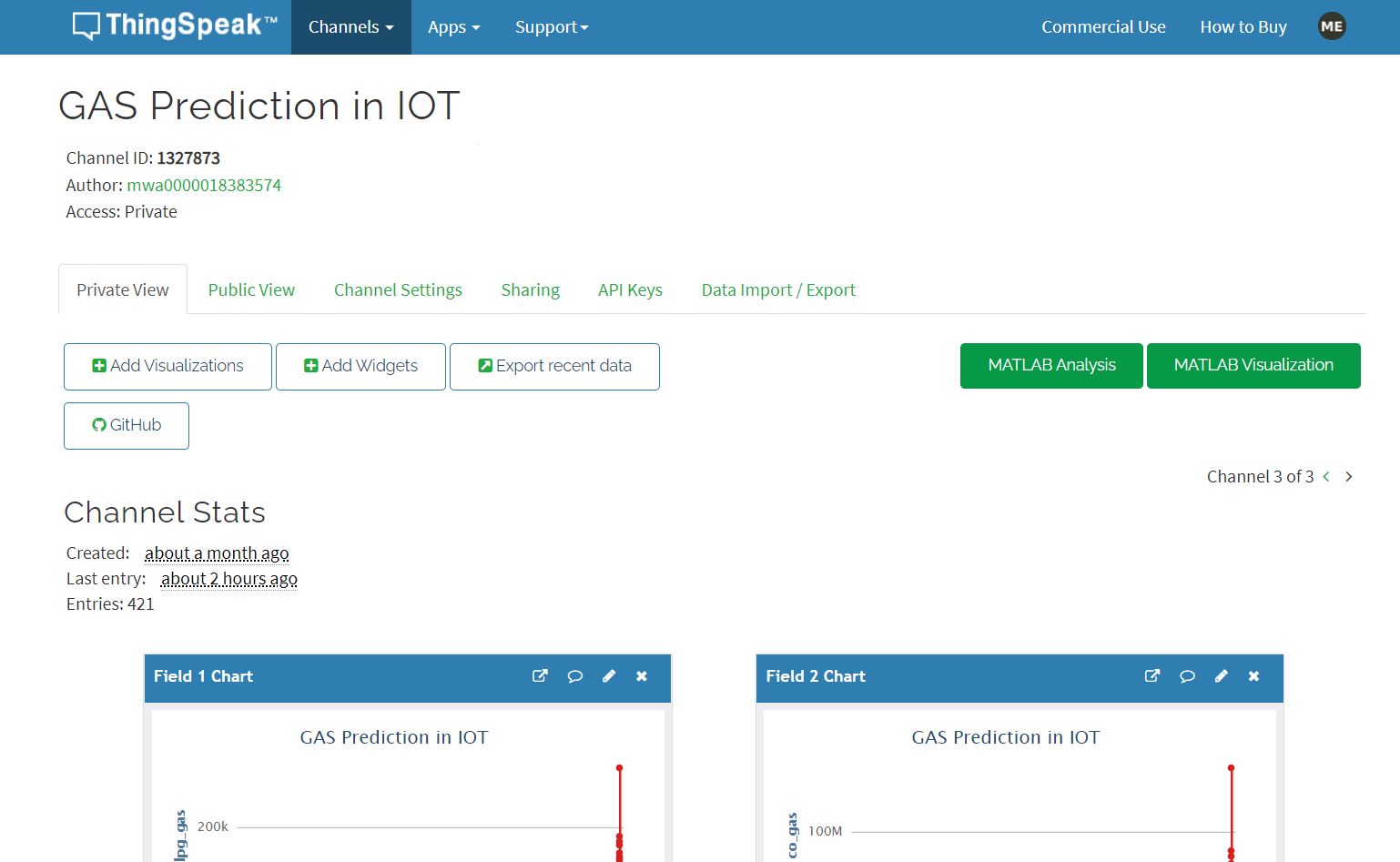
Make a connection as per the **circuit diagram** below. Connect the VCC pin of MQ2 to Vin of NodeMCU and GND to GND. Connect analog pin A0 of MQ2 to Analog pin A0 of NodeMCU



## 

### 1.4.2.Send the data to the Thingspeak:

1. Go to [**https://thingspeak.com/**](https://thingspeak.com/) and create an account if you do not have one. Login to your account.
2. Create a new channel by clicking on the button. Enter the basic details of the channel. Than Scroll down and save the channel. You can follow the video guide below.
3. Then go to API keys copy and paste this key to a separate notepad file. You will need it later while programming.

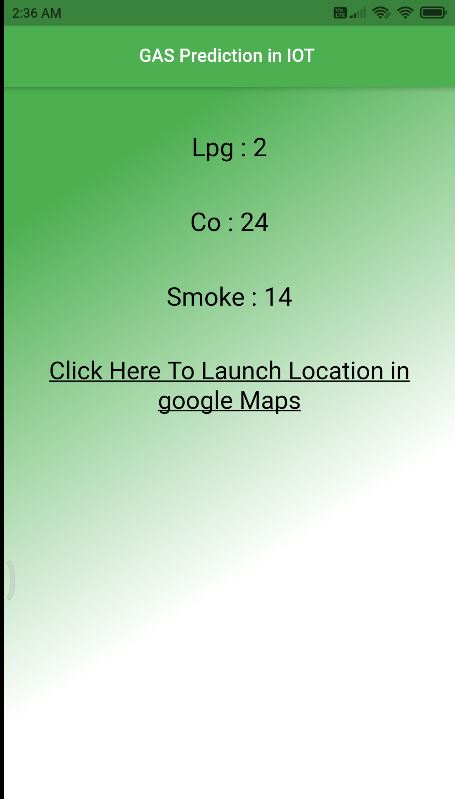
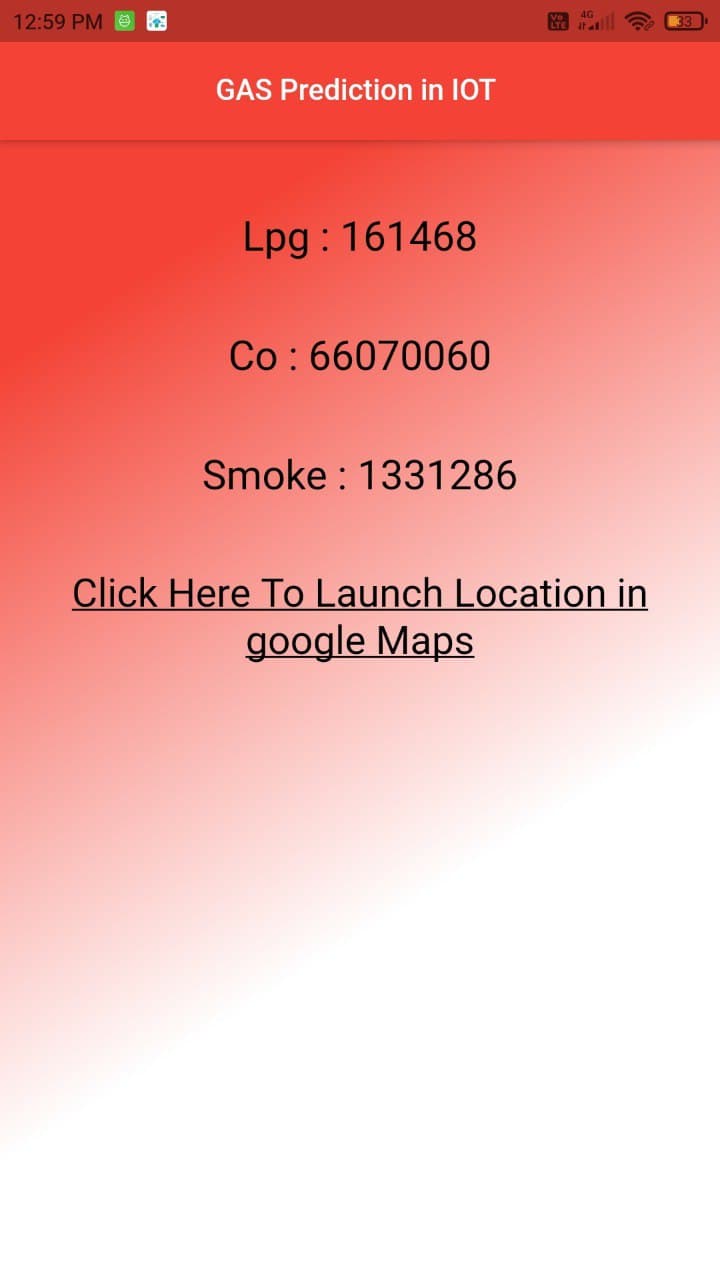


## API Requests Results

### 

### 1.4.3.Making Alert Alarm in Flutter Application

Flutter is a UI toolkit for creating fast, beautiful, natively compiled applications for mobile, web, and desktop with one programing language and single codebase. It is free and open-source. It was initially developed from **Google** and now manages by an **ECMA** standard. Flutter apps use Dart programming language for creating an app. The **dart programming** shares several same features as other programming languages, such as Kotlin and Swift, and can be trans-compiled into JavaScript code. Flutter is mainly optimized for 2D mobile apps that can run on both Android and iOS platforms. We can also use it to build full-featured apps, including camera, storage, geolocation, network, third-party SDKs, and more.

** **

# CHAPTER –2

## 2.1 Requirements

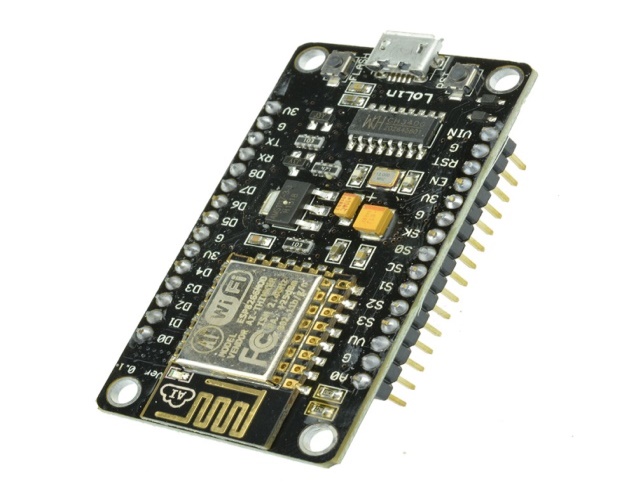
1. ESP8266-Nodemcu
2. MQ2
3. Arduino IDE
4. Thinkspeak cloud
5. Flutter

### 2.1.1 ESP8266-Nodemcu:

NodeMCU is a minimal effort open source IoT stage. It at first included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Frameworks, and equipment which depended on the ESP-12 module.Later, support for the ESP32 32-digit MCU was added.

NodeMCU is an open source firmware for which open source prototyping board plans are accessible. The name "NodeMCU" joins "hub" and "MCU" (miniature regulator unit).The term "NodeMCU" carefully talking alludes to the firmware instead of the related advancement packs. Both the firmware and prototyping board plans are open source.The firmware utilizes the Lua scripting language. The firmware depends on the eLua project, and based on the Espressif Non-operating system SDK for ESP8266. It utilizes many open source projects, for example, lua-cjson and SPIFFS.Due to asset limitations, clients need to choose the modules applicable for their task and construct a firmware custom fitted to their necessities. Backing for the 32-bit ESP32 has likewise been executed. The prototyping equipment commonly utilized is a circuit board working as a double in-line bundle (Plunge) which incorporates a USB regulator with a more modest surface-mounted board containing the MCU and radio wire. The decision of the Plunge design takes into consideration simple prototyping on breadboards. The plan was at first dependent on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC incorporated with a Tensilica Xtensa LX106 center, broadly utilized in IoT applications

Below the image is ESP8266:



### ****NodeMCU ESP8266 Specifications & Features****

* Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
* Operating Voltage: 3.3V
* Input Voltage: 7-12V
* Digital I/O Pins (DIO): 16
* Analog Input Pins (ADC): 1
* UARTs: 1
* SPIs: 1
* I2Cs: 1
* Flash Memory: 4 MB
* SRAM: 64 KB
* Clock Speed: 80 MHz
* USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
* PCB Antenna
* Small Sized module to fit smartly inside your IoT projects

## The NodeMCU ESP8266 development board comes with the ESP-12E module containing ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

## NodeMCU can be powered using Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

## NodeMCU ESP8266 Layout

## NodeMCU ESP8266 Layout

## Programming NodeMCU ESP8266 with Arduino IDE

## The NodeMCU Development Board can be easily programmed with Arduino IDE since it is easy to use.

## Programming NodeMCU with the Arduino IDE will hardly take 5-10 minutes. All you need is the Arduino IDE, a USB cable and the NodeMCU board itself. You can check this Getting Started Tutorial for NodeMCU to prepare your Arduino IDE for NodeMCU.

## Uploading your first program

## Once Arduino IDE is installed on the computer, connect the board with the computer using the USB cable. Now open the Arduino IDE and choose the correct board by selecting Tools>Boards>NodeMCU1.0 (ESP-12E Module), and choose the correct Port by selecting Tools>Port. To get it started with the NodeMCU board and blink the built-in LED, load the example code by selecting Files>Examples>Basics>Blink. Once the example code is loaded into your IDE, click on the ‘upload’ button given on the top bar. Once the upload is finished, you should see the built-in LED of the board blinking.

## Applications of NodeMCU

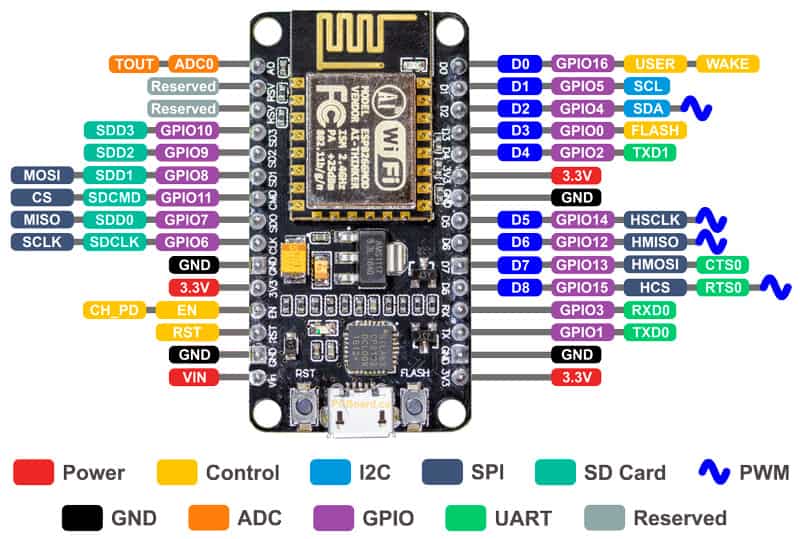
## Prototyping of IoT devices

## Low power battery operated applications

## Network projects

## Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

**NodeMCU Pinout and Functions Explained**



NodeMCU ESP8266 Pinout Listing To Matching Functions

**Power Pins** There are four power pins. VIN pin and three 3.3V pins.

* **VIN** can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the VIN pin
* **3.3V** pins are the output of the onboard voltage regulator and can be used to supply power to external components.

**GND** are the ground pins of NodeMCU/ESP8266

**I2C Pins** are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

**GPIO Pins** NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

**ADC Channel** The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

**UART Pins** NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

**SPI Pins** NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

* 4 timing modes of the SPI format transfer
* Up to 80 MHz and the divided clocks of 80 MHz
* Up to 64-Byte FIFO

**SDIO Pins** NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

**PWM Pins** The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).

**Control Pins** are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

* EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
* RST: RST pin is used to reset the ESP8266 chip.
* WAKE: Wake pin is used to wake the chip from deep-sleep.

**Tiny Sine WaveControl Pins** are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

### 2.1.2 MQ2:

The MQ-2 sensor is used to detect the levels of gas around the area, so you can utilize this sensor as a gas leak monitoring system for homes, businesses, or factories, and is suitable for monitoring devices such as gas, butane, propane, methane, alcohol, hydrogen, and smoke. MQ2 gas sensor is an electronic sensor used for sensing the concentration of gases in the air such as LPG, propane, methane, hydrogen, alcohol, smoke and carbon monoxide. MQ2 gas sensor is also known as chemiresistor. It contains a sensing material whose resistance changes when it comes in contact with the gas. This change in the value of resistance is used for the detection of gas.

#### Use MQ-2 Sensors to detect gas

Using an MQ sensor it detects a gas is very easy. You can either use the digital pin or the analog pin to accomplish this. Simply power the module with 5V and you should notice the power LED on the module to glow and when no gas it detected the output LED will remain turned off meaning the digital output pin will be 0V. Remember that these sensors have to be kept on for pre-heating time (mentioned in features above) before you can actually work with it. Now, introduce the sensor to the gas you want to detect and you should see the output LED to go high along with the digital pin, if not use the potentiometer until the output gets high. Now every time your sensor gets introduced to this gas at this particular concentration the digital pin will go high (5V) else will remain low (0V).

You can also use the analog pin to achieve the same thing. Read the analog values (0-5V) using a microcontroller, this value will be directly proportional to the concentration of the gas to which the sensor detects. You can experiment with this values and check how the sensor reacts to different concentration of gas and develop your program accordingly.

Below the image is MQ2 sensor:



MQ2 is a metal oxide semiconductor type gas sensor. Concentrations of gas in the gas is measured using a voltage divider network present in the sensor. This sensor bworks on 5V DC voltage. It can detect gases in the concentration of range 200 to 10000ppm.

#### Working Principle

This sensor contains a sensing element, mainly aluminium-oxide based ceramic, coated with Tin dioxide, enclosed in a stainless steel mesh. Sensing element has six connecting legs attached to it. Two leads are responsible for heating the sensing element, the other four are used for output signals.Oxygen gets adsorbed on the surface of sensing material when it is heated in air at high temperature. Then donor electrons present in tin oxide are attracted towards this oxygen, thus preventing the current flow. When reducing gases are present, these oxygen atoms react with the reducing gases thereby decreasing the surface density of the adsorbed oxygen. Now current can flow through the sensor, which generated analog voltage values.These voltage values are measured to know the concentration of gas. Voltage values are higher when the concentration of gas is high.

#### Applications

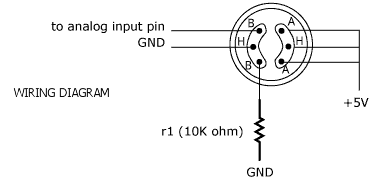
These sensors are used to detect the presence of gases in the air such as methane, butane, LPG and smoke but they are unable to distinguish between gases. Thus, they cannot tell which gas it is. Module version of this sensor can be used without interfacing to any microcontroller and is useful when detecting only one particular gas. This can only detect the gas. But if ppm has to be calculated then the sensor should be used without module.This sensor is also used for Air quality monitoring, Gas leak alarm and for maintaining environmental standards in hospitals. In industries, these are used to detect the leakage of harmful gases.

#### Features

1. Operating Voltage is +5V
2. Can be used to Measure or detect LPG, Alcohol, Propane, Hydrogen, CO and even methane
3. Analog output voltage: 0V to 5V
4. Digital Output Voltage: 0V or 5V (TTL Logic)
5. Preheat duration 20 seconds
6. Can be used as a Digital or analog sensor
7. The Sensitivity of Digital pin can be varied using the potentiometer

If you are looking for some accuracy with your readings then measuring the PPM would be the best way to go with it. It can also help you to distinguish one gas from another. So to measure PPM you can directly use a module.

**A basic wiring for the sensor from datasheet is shown below.**



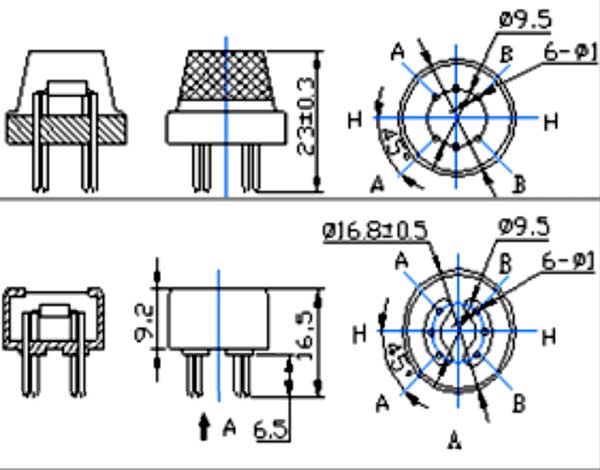
It comes to measuring or detecting a particular Gas the **MQ series Gas sensors** are the most commonly used ones. These sensors can either be purchased as a module or as just the sensor alone. If you are trying to only detect (not measuring ppm) the presence of a gas then you can buy it as a module since it comes with an op-amp comparator and a digital out pin. But if you planning to measure the ppm of a gas it is recommended to buy the sensor alone (without module).

Pin Configuration***:***

|  |  |  |
| --- | --- | --- |
| Pin No: | Pin Name: | Pin Name: |
| 1 | VCC | This pin powers the module, typically the operating voltage is +5V |
| 2 | GROUND | Used to connect the module to system ground |
| 3 | DIGITAL  OUT | It can also use this sensor to get digital output from this pin, by setting a threshold value using the potentiometer |
| 4 | ANALOG  OUT | This pin outputs 0-5V analog voltage based on the intensity of the |

### **2D model of MQ-2 Gas sensor***:*****

Following dimensions to create your own PCB for your application



### 2.1.3 Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. [Arduino boards](https://www.arduino.cc/en/Main/Products) are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of [accessible knowledge](http://forum.arduino.cc/) that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The [software](https://www.arduino.cc/en/Main/Software), too, is open-source, and it is growing through the contributions of users worldwide.

It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* **Inexpensive**

Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

* **Cross-platform**

The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

* **Simple, clear programming environment**

The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

* **Open source and extensible software**

The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

* **Open source and extensible hardware**

The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the [breadboard version of the module](https://www.arduino.cc/en/Main/Standalone) in order to understand how it works and save money.

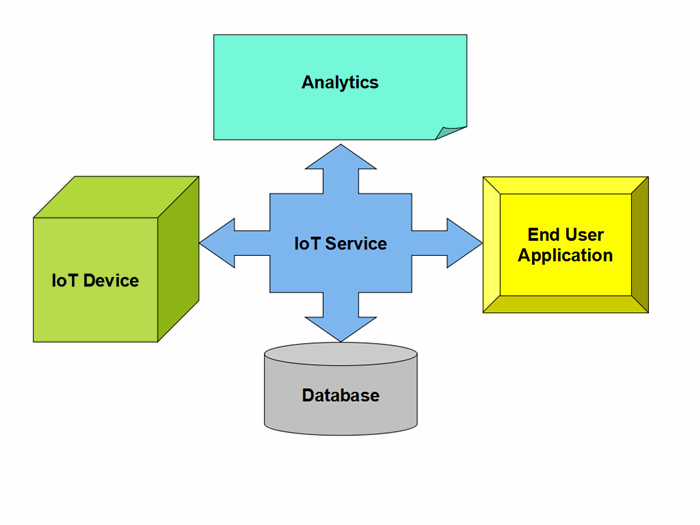
#### IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.The source code for the IDE is released under the GNU General Public License, version 2.The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards

## The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

### 2.1.4 Thinkspeak cloud

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts



The Internet of Things(IoT) is a system of ‘connected things’. The things generally comprise of an embedded operating system and an ability to communicate with the internet or with the neighboring things. One of the key elements of a generic IoT system that bridges the various ‘things’ is an IoT service. An interesting implication from the ‘things’ comprising the IoT systems is that the things by themselves cannot do anything. At a bare minimum, they should have an ability to connect to other ‘things’. But the real power of IoT is harnessed when the things connect to a ‘service’ either directly or via other ‘things’. In such systems, the service plays the role of an invisible manager by providing capabilities ranging from simple data collection and monitoring to complex data analytics. The below diagram illustrates where an IoT service fits in an IoT ecosystem

ThingSpeak is a platform providing various services exclusively targeted for building IoT applications. It offers the capabilities of real-time data collection, visualizing the collected data in the form of charts, ability to create plugins and apps for collaborating with web services, social network and other APIs. We will consider each of these features in detail below.

The core element of ThingSpeak is a ‘ThingSpeak Channel’. A channel stores the data that we send to ThingSpeak and comprises of the below elements:

* 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device.
* 3 location fields - Can be used to store the latitude, longitude and the elevation. These are very useful for tracking a moving device.
* 1 status field - A short message to describe the data stored in the channel.

To use ThingSpeak, we need to signup and create a channel. Once we have a channel, we can send the data, allow ThingSpeak to process it and also retrieve the same. Let us start exploring ThingSpeak by signing up and setting up a channel.

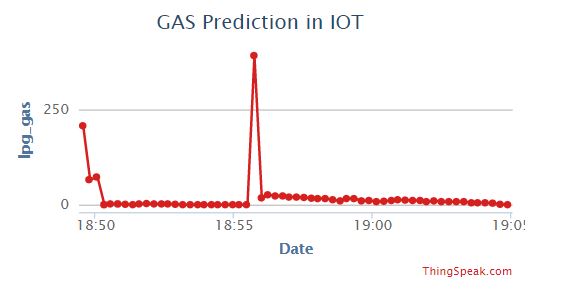
## 

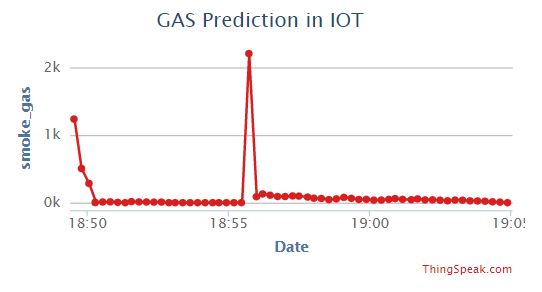
#### ThingSpeak Apps

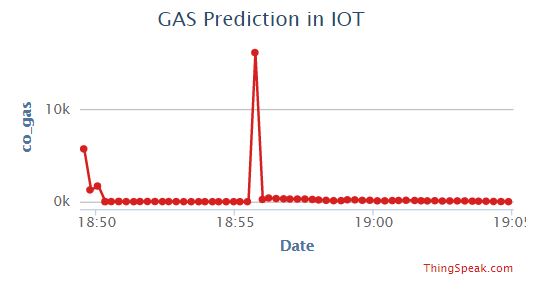
ThingSpeak provides apps that allow us for an easier integration with the web services, social networks and other APIs. Below are some of the apps provided by ThingSpeak:

* **ThingTweet** - This allows you to post messages to twitter via ThingSpeak. In essence, this is a TwitterProxy which re-directs your posts to twitter.
* **ThingHTTP** - This allows you to connect to web services and supports GET, PUT, POST and DELETE methods of HTTP.
* **TweetControl** - Using this, you can monitor your Twitter feeds for a specific key word and then process the request. Once the specific keyword is found in the twitter feed, you can then use ThingHTTP to connect to a different web service or execute a specific action.
* **React** - Send a tweet or trigger a ThingHTTP request when the Channel meets a certain condition.
* **TalkBack** - Use this app to queue up commands and then allow a device to act upon these queued commands.
* **Timecontrol** - Using this app, we can do a ThingTweet, ThingHTTP or a TalkBack at a specified time in the future. We can also use this to allow these actions to happen at a specified time throughout the week.

#### Out Of Thinkspeak



****

****

### 2.1.5 Flutter

In general, creating a mobile application is a very complex and challenging task. There are many frameworks available, which provide excellent features to develop mobile applications. For developing mobile apps, Android provides a native framework based on Java and Kotlin language, while iOS provides a framework based on Objective-C/Swift language. Thus, we need two different languages and frameworks to develop applications for both OS. Today, to overcome form this complexity, there are several frameworks have introduced that support both OS along with desktop apps. These types of the framework are known as **cross-platform** development tools.

The cross-platform development framework has the ability to write one code and can deploy on the various platform (Android, iOS, and Desktop). It saves a lot of time and development efforts of developers. There are several tools available for cross-platform development, including web-based tools, such as Ionic from Drifty Co. in 2013, Phonegap from Adobe, Xamarin from Microsoft, and React Native form Facebook. Each of these frameworks has varying degrees of success in the mobile industry. In recent, a new framework has introduced in the cross-platform development family named **Flutter** developed from Google.

#### Features of Flutter

**Open-Source:** Flutter is a free and open-source framework for developing mobile applications.

**Cross-platform:** This feature allows Flutter to write the code once, maintain, and can run on different platforms. It saves the time, effort, and money of the developers.

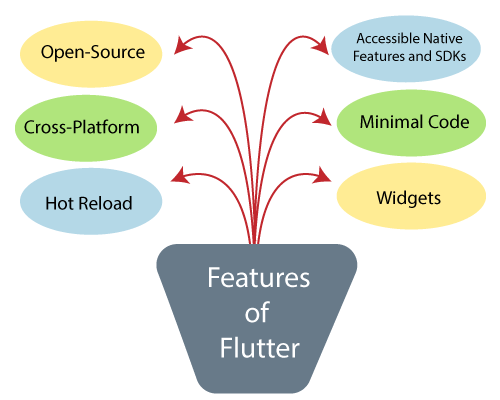
**Hot Reload:** Whenever the developer makes changes in the code, then these changes can be seen instantaneously with Hot Reload. It means the changes immediately visible in the app itself. It is a very handy feature, which allows the developer to fix the bugs instantly.

**Accessible Native Features and SDKs:** This feature allows the app development process easy and delightful through Flutter's native code, third-party integration, and platform APIs. Thus, we can easily access the SDKs on both platforms.

**Minimal code:** Flutter app is developed by Dart programming language, which uses JIT and AOT compilation to improve the overall start-up time, functioning and accelerates the performance. JIT enhances the development system and refreshes the UI without putting extra effort into building a new one.

**Widgets:** The Flutter framework offers widgets, which are capable of developing customizable specific designs. Most importantly, Flutter has two sets of widgets: Material Design and Cupertino widgets that help to provide a glitch-free experience on all platforms

Before learning Flutter in-depth, you must have a sound understanding of Dart programming, Android Studio, and web scripting languages such as HTML, JavaScript, and CSS.

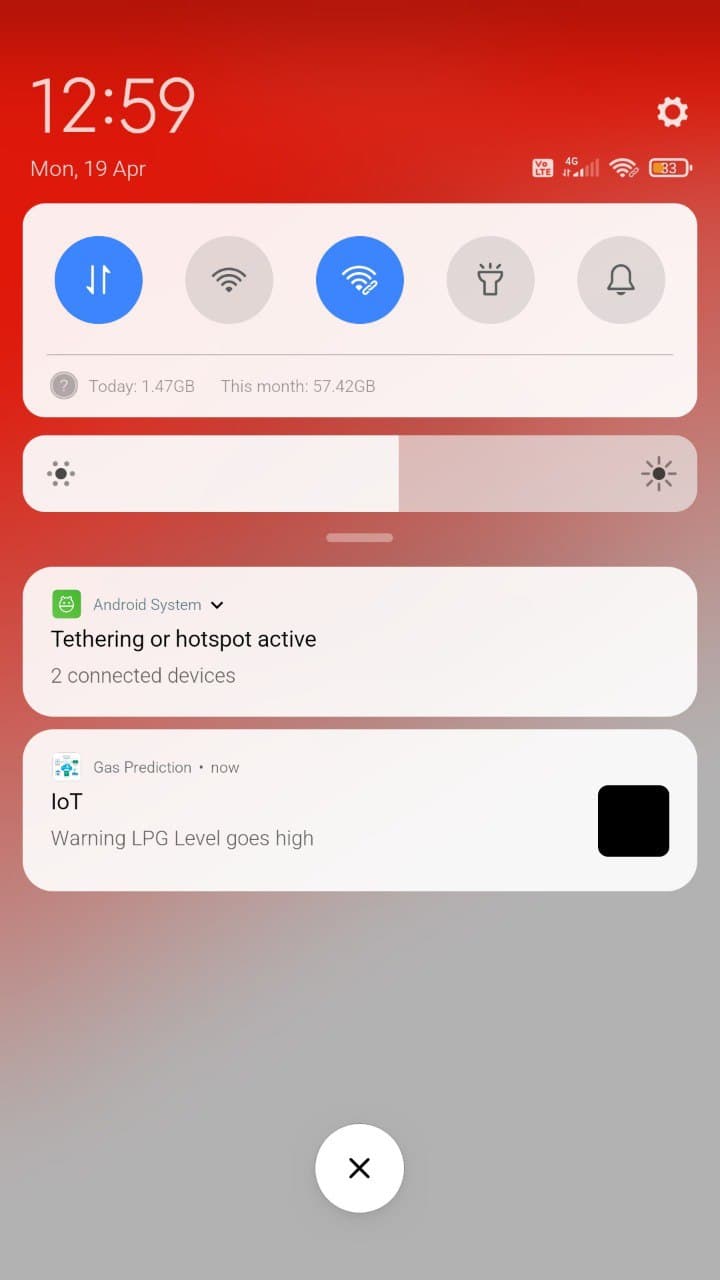
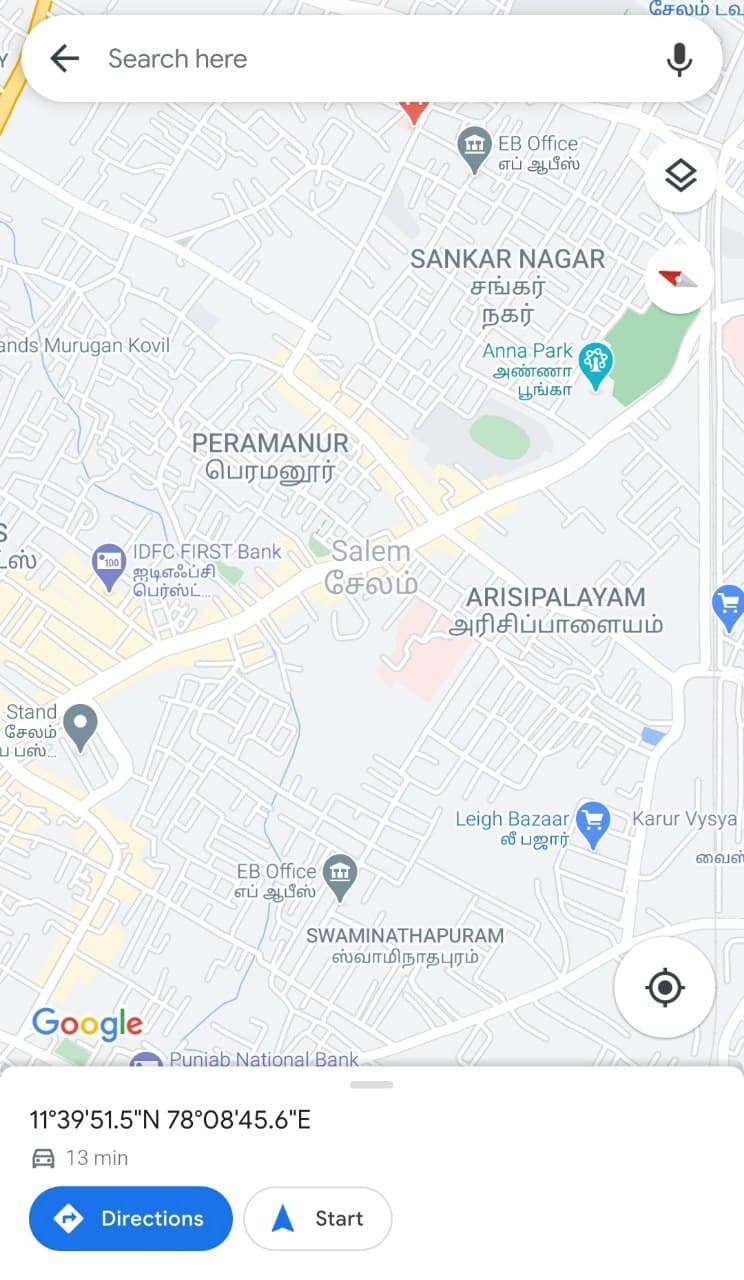


#### History of Flutter

Flutter is a free and open-source UI software development kit introduced by Google. It is used to build applications for Android, iOS, Windows, and the web. The first version of Flutter was announced in the year **2015** at the **Dart Developer Summit**. It was initially known as codename **"Sky"** and can run on the Android OS. After the announcement of Flutter, the first Flutter Alpha version (v-0.06) was released in May **2017**. Later, during the keynote of Google Developer days in Shanghai, Google launched the second preview of Flutter in **September 2018** that was the last big release before Flutter 1.0 version. On **December 4, 2018**, the first stable version of the Flutter framework was released at the Flutter Live event, denoting Flutter 1.0. The current stable release of the framework is Flutter v1.9.1+hotfix.6 on October 24, 2019.

Flutter is different from other frameworks because it neither uses **WebView** nor the **OEM** widgets that shipped with the device. Instead, it uses its own high-performance rendering engine to draw widgets. It also implements most of its systems such as animation, gesture, and widgets in Dart programing language that allows developers to read, change, replace, or remove things easily. It gives excellent control to the developers over the system.

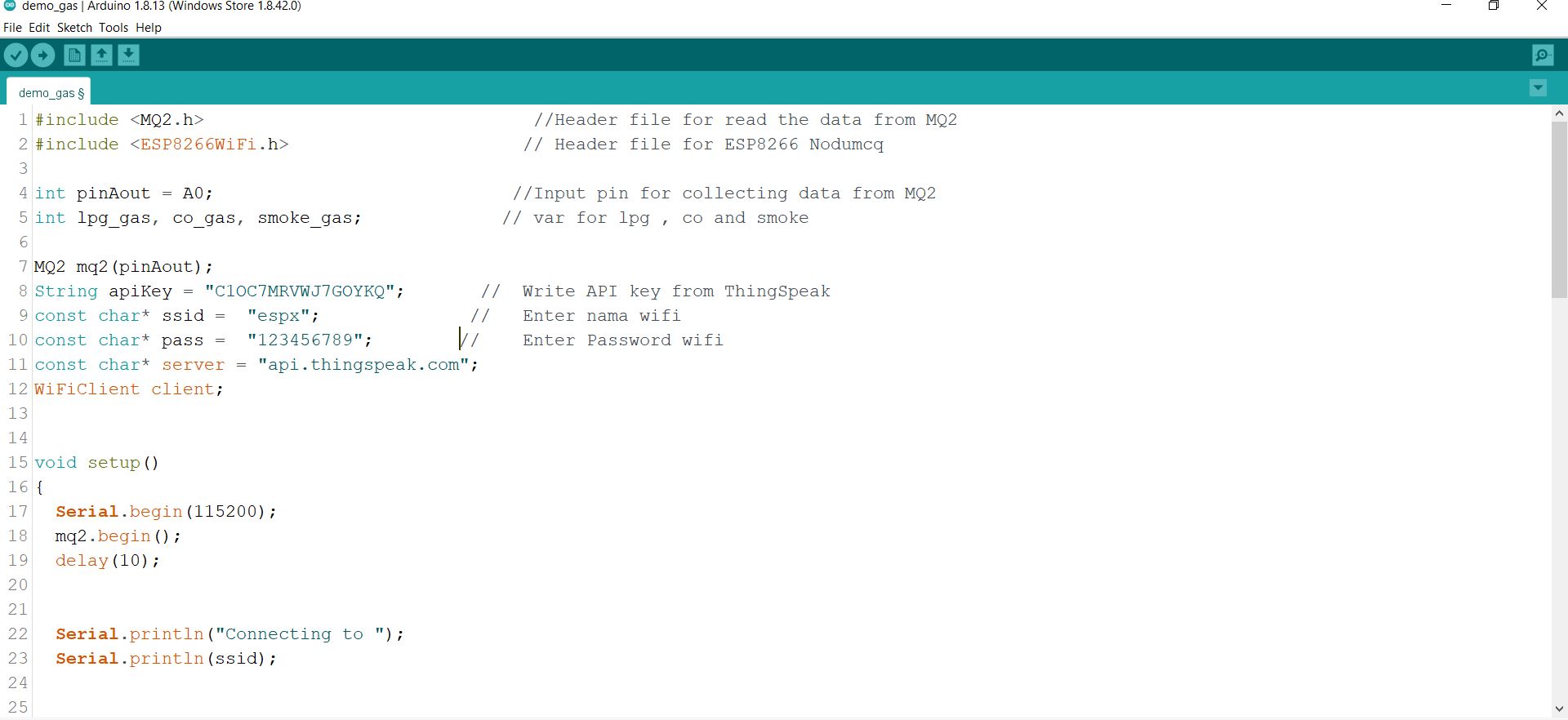
Blow the image is lanching Gmap : Blow the image is notification Alert :

## 2.2 Source Code of Arduino :

Here is a program for IoT BASED GAS LEAKAGE DETECTION & SMART ALERTING SYSTEM. Copy this code and upload it to Nodemcu.

* Change the wifi SSID,
* Change the password, and
* Change the thingspeak API key.
* String apiKey = "C1OC7MRVWJ7GOYKQ"; // Write API key from ThingSpeak
* const char\* ssid = "espx"; // Enter nama wifi
* const char\* pass = "123456789"; // Enter Password wifi



## 

### 2.2.1 Gas\_ESP8266.ino

#include <MQ2.h> //Header file for read the data from MQ2

#include <ESP8266WiFi.h> // Header file for ESP8266 Nodumcq

int pinAout = A0; //Input pin for collecting data from MQ2

int lpg\_gas, co\_gas, smoke\_gas; // var for lpg , co and smoke

MQ2 mq2(pinAout);

String apiKey = "C1OC7MRVWJ7GOYKQ"; // Write API key from ThingSpeak

const char\* ssid = "espx"; // Enter nama wifi

const char\* pass = "123456789"; // Enter Password wifi

const char\* server = "api.thingspeak.com";

WiFiClient client;

void setup()

{

Serial.begin(115200);

mq2.begin();

delay(10);

Serial.println("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, pass);

while (WiFi.status() != WL\_CONNECTED)

{

delay(100);

Serial.print("\*");

}

Serial.println("");

Serial.println("\*\*\*WiFi Connected\*\*\*");

}

void loop()

{

if (client.connect(server, 80)) // "184.106.153.149" or api.thingspeak.com

{

String sendData = apiKey + "&field1=" + String(lpg\_gas) + "&field2=" + String(co\_gas) + "&field3=" + String(smoke\_gas) + "\r\n\r\n";

//Serial.println(sendData);

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(sendData.length());

client.print("\n\n");

client.print(sendData);

float\* values = mq2.read(false);

lpg\_gas = mq2.readLPG();

co\_gas = mq2.readCO();

smoke\_gas = mq2.readSmoke();

Serial.print("LPG:");

Serial.print(lpg\_gas);

Serial.print("\n");

Serial.print(" CO:");

Serial.print(co\_gas);

Serial.print("\n");

Serial.print("SMOKE:");

Serial.println(smoke\_gas);

Serial.print("\n");

Serial.println("%. Data send to Thingspeak.");

}

client.stop();

Serial.println("send data....");

delay(2000);

}

## 

## 2.3 Source Code of Flutter Application:

### 2.3.1 generated\_plugin\_registrant.dart:

import 'package:url\_launcher\_web/url\_launcher\_web.dart';

import 'package:vibration\_web/vibration\_web.dart';

import 'package:flutter\_web\_plugins/flutter\_web\_plugins.dart';

// ignore: public\_member\_api\_docs

void registerPlugins(Registrar registrar) {

UrlLauncherPlugin.registerWith(registrar);

VibrationWebPlugin.registerWith(registrar);

registrar.registerMessageHandler();

}

#### 

### 2.3.2 Main.dart:

import 'package:flutter/material.dart';

import 'package:flutter\_local\_notifications/flutter\_local\_notifications.dart';

import 'package:gas\_prediction/weather.dart';

final FlutterLocalNotificationsPlugin flip = FlutterLocalNotificationsPlugin();

void main() async {

WidgetsFlutterBinding.ensureInitialized();

var android = AndroidInitializationSettings('iot');

var settings = InitializationSettings(android: android);

await flip.initialize(settings, onSelectNotification: (String payload) async

{

if (payload != null) {

print("notification payload: $payload");

}

});

runApp(

MaterialApp(

title: 'WeatherPrediction',

debugShowCheckedModeBanner: false,

home: WeatherPrediction(),

),

);

}

### 2.3.3 Gas\_predicition.dart:

import 'dart:async';

import 'package:flutter/material.dart';

import 'package:flutter\_local\_notifications/flutter\_local\_notifications.dart';

import 'dart:convert';

import 'package:http/http.dart' as http;

import 'package:url\_launcher/url\_launcher.dart';

import 'package:vibration/vibration.dart';

import 'main.dart';

class WeatherPrediction extends StatefulWidget {

@override

\_WeatherPredictionState createState() => \_WeatherPredictionState();

}

class \_WeatherPredictionState extends State<WeatherPrediction> {

static final url =

'https://api.thingspeak.com/channels/1327873/feeds.json?api\_key=N6F0DZIXW82I47WW&results=2';

var channel;

var feeds;

var lat = "";

var lon = "";

var title = "";

int lpg = 0;

int smoke = 0;

int co = 0;

Timer timer;

bool isNotify = false;

void showNotifications(String message) async {

var android = AndroidNotificationDetails(

'IoT\_notif',

'IoT\_notif',

'Channel for IoT notification',

icon: 'iot',

largeIcon: DrawableResourceAndroidBitmap('iot'),

);

var platformSpecifics = NotificationDetails(android: android);

flip.show(0, "IoT", message, platformSpecifics);

}

Future<void> getResponse() async {

var response = await http.get(url);

Map map = json.decode(response.body) as Map;

setState(() {

channel = map['channel'];

feeds = map['feeds'];

title = channel['name'];

lat = channel['latitude'];

lon = channel['longitude'];

lpg = int.parse(feeds[0]['field1']);

co = int.parse(feeds[0]['field2']);

smoke = int.parse(feeds[0]['field3']);

});

if (lpg >= 700) {

isNotify = true;

showNotifications("Warning LPG Level goes high");

vibrateDevice();

} else if (smoke >= 1000) {

showNotifications("Warning Smoke Level goes high");

vibrateDevice();

isNotify = true;

} else if (co >= 10000) {

showNotifications("Warning Co Level goes high");

vibrateDevice();

isNotify = true;

} else {

isNotify = false;

}

}

void vibrateDevice() async {

if (await Vibration.hasVibrator()) {

Vibration.vibrate();

}

}

void launchUrl(String url) async {

await canLaunch(url) ? await launch(url) : throw "Couldn't launch $url";

}

Widget textWidget({String text, double textSize}) {

return Center(

child: Text(

"$text",

style: TextStyle(fontSize: textSize, color: Colors.black),

textAlign: TextAlign.center,

),

);

}

@override

void initState() {

super.initState();

timer = Timer.periodic(Duration(seconds: 1), (Timer t) => getResponse());

getResponse();

}

@override

void dispose() {

timer.cancel();

super.dispose();

}

@override

Widget build(BuildContext context) {

var boxSide = MediaQuery.of(context).size.width \* 0.10;

return Scaffold(

appBar: AppBar(

backgroundColor: isNotify ? Colors.red : Colors.green,

title: Text(

title,

style: TextStyle(fontSize: boxSide \* 0.50),

),

centerTitle: true,

),

body: feeds == null && channel == null

? Center(

child: Text(

'Fetching Data Please Wait....',

style: TextStyle(fontSize: boxSide \* 0.70),

textAlign: TextAlign.center,

),

)

: Container(

decoration: BoxDecoration(

gradient: LinearGradient(colors: [

isNotify ? Colors.red : Colors.green,

Colors.white

], begin: Alignment.topCenter)),

child: ListView(

children: [

SizedBox(

height: boxSide,

width: boxSide,

),

textWidget(text: 'Lpg : $lpg', textSize: boxSide \* 0.70),

SizedBox(

height: boxSide,

width: boxSide,

),

textWidget(text: 'Co : $co', textSize: boxSide \* 0.70),

SizedBox(

height: boxSide,

width: boxSide,

),

textWidget(text: 'Smoke : $smoke', textSize: boxSide \* 0.70),

SizedBox(

height: boxSide,

width: boxSide,

),

Container(

height: boxSide \* 2.5,

width: boxSide \* 2.5,

child: MaterialButton(

child: Image.asset('images/map.png'),

onPressed: () {

launchUrl("https://maps.google.com/?q=$lat,$lon");

},

),

),

SizedBox(

height: boxSide,

width: boxSide,

),

],

),

),

);

}

}

## 2.4 Source Code of Flutter web-Application:

### 2.4.1 Index.html

<!DOCTYPE html>

<html>

<head>

<!--

If you are serving your web app in a path other than the root, change the

href value below to reflect the base path you are serving from.

The path provided below has to start and end with a slash "/" in order for

it to work correctly.

For more details:

\* https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base

-->

<base href="/">

<meta charset="UTF-8">

<meta content="IE=Edge" http-equiv="X-UA-Compatible">

<meta name="description" content="A new Flutter project.">

<!-- iOS meta tags & icons -->

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-title" content="gas\_prediction">

<link rel="apple-touch-icon" href="icons/Icon-192.png">

<title>gas\_prediction</title>

<link rel="manifest" href="manifest.json">

</head>

<body>

<!-- This script installs service\_worker.js to provide PWA functionality to

application. For more information, see:

https://developers.google.com/web/fundamentals/primers/service-workers -->

<script>

var serviceWorkerVersion = null;

var scriptLoaded = false;

function loadMainDartJs() {

if (scriptLoaded) {

return;

}

scriptLoaded = true;

var scriptTag = document.createElement('script');

scriptTag.src = 'main.dart.js';

scriptTag.type = 'application/javascript';

document.body.append(scriptTag);

}

if ('serviceWorker' in navigator) {

// Service workers are supported. Use them.

window.addEventListener('load', function () {

// Wait for registration to finish before dropping the <script> tag.

// Otherwise, the browser will load the script multiple times,

// potentially different versions.

var serviceWorkerUrl = 'flutter\_service\_worker.js?v=' + serviceWorkerVersion;

navigator.serviceWorker.register(serviceWorkerUrl)

.then((reg) => {

function waitForActivation(serviceWorker) {

serviceWorker.addEventListener('statechange', () => {

if (serviceWorker.state == 'activated') {

console.log('Installed new service worker.');

loadMainDartJs();

}

});

}

if (!reg.active && (reg.installing || reg.waiting)) {

// No active web worker and we have installed or are installing

// one for the first time. Simply wait for it to activate.

waitForActivation(reg.installing ?? reg.waiting);

} else if (!reg.active.scriptURL.endsWith(serviceWorkerVersion)) {

// When the app updates the serviceWorkerVersion changes, so we

// need to ask the service worker to update.

console.log('New service worker available.');

reg.update();

waitForActivation(reg.installing);

} else {

// Existing service worker is still good.

console.log('Loading app from service worker.');

loadMainDartJs();

}});

// If service worker doesn't succeed in a reasonable amount of time,

// fallback to plaint <script> tag.

setTimeout(() => {

if (!scriptLoaded) {

console.warn(

'Failed to load app from service worker. Falling back to plain <script> tag.' ) loadMainDartJs();

}

}, 4000);

});

} else {

// Service workers not supported. Just drop the <script> tag.

loadMainDartJs();

}

</script>

</body>

</html>

### 2.4.2 Main.json

{

"name": "gas\_prediction",

"short\_name": "gas\_prediction",

"start\_url": ".",

"display": "standalone",

"background\_color": "#0175C2",

"theme\_color": "#0175C2",

"description": "A new Flutter project.",

"orientation": "portrait-primary",

"prefer\_related\_applications": false,

"icons": [

{

"src": "icons/Icon-192.png",

"sizes": "192x192",

"type": "image/png"

},

{

"src": "icons/Icon-512.png",

"sizes": "512x512",

"type": "image/png"

}

]

}

# CHAPTER-3

## CONCLUSION

The system provides constant monitoring and detection of gas leakage along with storage of data in database for predictions and analysis. The IOT components used helps in making the system much more cost effective in comparison with traditional Gas detector systems. The system alerts and responds quickly in case of gas leakage with the help of alerting mechanism and by sending alert alarm in flutter app or concerned authority. The system also allows user to perform analysis and prediction.

The main objective of the project is to detect the smoke and gas leakage using ESP8266 . Using MQ5 sensor to detect the gas leakage and MQ135 sensor to detect the smoke . And it connect with the Thinkspeak (cloud) collect the data and stored in database .Then It access by API keys to display the information in flutter application .The flutter application is developed by the dart language. In this project to find smoke or gas leakage and inform to the user by flutter application and also active the alarm in home . MQ2 sensor can detect or measure gasses like LPG, Alcohol, Propane, Hydrogen, CO and even methane.

## REFERENCES

[1] Alam June 26, 2020 Gas Level Monitor On Internet Using ESP8266 & Gas Sensor https://how2electronics.com/gas-level-monitor-esp8266-gas-sensor/

[2] MQ2 Gas Sensor Working and Its Applications https://www.elprocus.com/an-introduction-to-mq2-gas-sensor/

[3] ESP8266 Opensource Community,NodeMCU From Wikipedia https://en.wikipedia.org/wiki/NodeMCU

[4] ThingSpeak From Wikipedia, the free encyclopedia <https://en.wikipedia.org/wiki/ThingSpeak>

[5] Flutter Tutorial from JavaTpoint https://www.javatpoint.com/flutter

[6] https://www.iosrjen.org/Papers/Conf.ICIATE-2018/Volume-13/3-11-16.pdf