IOT BASED GAS LEAKAGE DETECTION SYSTEM

A PROJECT REPORT

Submitted by

R.NISHANTH 2017504024

A.PARTHIBAN 2017504027

K.MEENAA 2017504018

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 044

OCTOBER 2020

ANNA UNIVERSITY: CHENNAI 600 044

BONAFIDE CERTIFICATE

Certified that this project report **"IOT BASED GAS LEAKAGE DETECTION SYSTEM"** is the bonafide work of

R.NISHANTH 2017504024

A.PARTHIBAN 2017504027

K.MEENAA 2017504018

who carried out the project work.

SIGNATURE

Dr. M. GANESH MADHAN
HEAD-IN-CHARGE
Associate Professor,
Department of Electronics Engg,
Madras Institute of Technology,
Anna University,
Chennai – 600 044.

SIGNATURE

Mrs. M.S. VINOTHENI
PROJECT CO-ORDINATOR
Teaching Fellow,
Department of Electronics Engg,
Madras Institute of Technology,
Anna University,
Chennai - 600 044.

ACKNOWLEDGEMENT

We are extremely grateful to our faculty in-charge **Ms.M.S.Vinotheni**, Teaching Fellow, Department of Electronics Engineering, MIT campus, for providing us valuable information, letting us use laboratory and for her relentless efforts in making sure that we have gained useful knowledge from this experience. We also thank her for providing continuous support throughout the semester for the successful completion of this project.

We take exorbitant pleasure in thanking **Prof.T.Thiyagarajan**, Dean, Madras Institute Of Technology, Anna University, for providing us with all the facilities required for our project work.

We are thankful to our head of the department **Dr.M.Ganesh Madhan**, Associate Professor, department of electronics engineering, MIT campus for giving us the opportunity to undertake this project and for providing us with the state of the art facilities.

We sincerely thank panel members for the valuable suggestions .we also thank our Faculty in-charge **Ms.M.S.Vinotheni** and all the teaching and non-teaching staff members of the department of Electronics engineering for their support in all aspects.

R.NISHANTH 2017504024

A.PARTHIBAN 2017504027

K.MEENAA 2017504018

CONTENTS

CONTENTS

ABSTRACT

Recent trend is the development of Smart homes all around the world. Home automation has become very affordable and many people, industries has started to automate daily routines like light, fans, setting the temperature, etc,. Gas leakages are one of the major reasons behind fires and blast accidents. It is important to monitor the gas level . The main objective of the project is to build a Gas leakage detector using LPG gas sensor and also connect it with IoT using ESP8266 module for safety and security and to monitor the gas level anywhere from the world and to control the home appliance. ESP8266 is used as the main controller. The system automatically detects gas leakages. If the leakage is detected, system automatically starts exhaust fan. Exhaust fan is used to suck the gas out of the room. At the same time buzzer is turned on to alert about gas leakage. Also the system informs the user through mail.The user can also monitor the sensor data using internet from any location and control the appliance in case of any emergency. This can prevent more accidents due to gas leakages. The outcome of the project will be small device which will be capable of detecting a gas leakage from cylinders and also turn on exhaust fan also notify the user by connecting via IoT software.

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 INTERNET OF THINGS

IoT is an expanding network of physical devices that are linked with different types of sensors and with the help of connectivity to the internet, they are able to exchange data. Through IoT, internet has now extended its roots to almost every possible thing present around us and is no more limited to our personal computers and mobile phones. Safety, the elementary concern of any project, has not been left untouched by IoT. Gas Leakages in open or closed areas can prove to be dangerous and lethal. The traditional Gas Leakage Detector Systems though have great precision, fail to acknowledge a few factors in the field of alerting the people about the leakage. Therefore we have used the IoT technology to make a Gas Leakage Detector having Smart Alerting techniques involving calling, sending text message and an e-mail to the concerned authority and an ability to predict hazardous situation so that people could be made aware in advance by performing data analytics on sensor readings.

Safety plays a major role in today's world and it is necessary that good safety systems are to be implemented in places of education and work. This work modifies the existing safety model installed in industries and this system also be used in homes and offices. The main objective of the work is designing microcontroller based toxic gas detecting and alerting system. The hazardous gases like LPG and other Natural gases were sensed and displayed and notify each and every second.

The advantage of this automated detection and alerting system over the manual method is that it offers quick response time and accurate detection of an emergency and in turn leading faster diffusion of the critical situation.

1.2 OVERVIEW

In Chapter 2, MQTT protocol basics is explained and how does it differs from HTTP protocol is also explained. Adafruit server and setup of Adafruit account is also discussed.

In Chapter 3, block diagram and working principle of the project are discussed.

In Chapter 4, the hardware component and its specification and pin configuration and working principle are discussed.

In Chapter 5, software used for the project and overview of programming are discussed.

In Chapter 6, the implementation of MQTT protocol using NodeMCU and results are discussed.

CHAPTER 2
MQTT PROTOCOL

2.1 MQTT PROTOCOL

- MQTT (MQ Telemetry Transport) is a lightweight messaging protocol that provides resource-constrained network clients with a simple way to distribute telemetry information.
- The protocol, which uses a publish/subscribe communication pattern, is used for machine-to-machine (M2M) communication and plays an important role in the internet of things (IoT).
- The MQTT protocol is a good choice for wireless networks that experience varying levels of latency due to occasional bandwidth constraints or unreliable connections.

2.1.1 MQTT PROTOCOL WORKING

The MQTT protocol surrounds two subjects: a client and a broker. An MQTT broker is a server, while the clients are the connected devices. When a device -- or client -- wants to send data to a server -- or broker-- it is called a publish. When the operation is reversed, it is called a subscribe. If the connection from a subscribing client to a broker is broken, then the broker will buffer messages and push them out to the subscriber when it is back online. If the connection from the publishing client to the broker is disconnected without notice, then the broker can close the connection and send subscribers a cached message with instructions from the publisher.

2.1.2 MQTT STAGES

An MQTT session is divided into four stages: connection, authentication, communication and termination. A client starts by creating a Transmission Control Protocol/Internet Protocol (TCP/IP) connection to the broker by using either a standard port or a custom port defined by the broker's operators. During the communication phase, a client can perform publish, subscribe, unsubscribe and operations.

2.2 MQTT VS HTTP PROTOCOL

HTTP is used as transport mechanism between the devices and the IoT Agent. HTTP uses a request/response paradigm where each device connects directly to the IoT Agent. MQTT is different in that publish-subscribe is event-driven and pushes messages to clients. It requires an additional central communication point (known as the MQTT broker) which it is in charge of dispatching all messages between the senders and the rightful receivers.

Each client that publishes a message to the broker, includes a topic into the message. The topic is the routing information for the broker. Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client. Therefore the clients don't have to know each other, they only communicate over the topic.

HTTP PROTOCOL	**MQTT PROTOCOL**
IoT Agent communicates with IoT devices **directly**	IoT Agent communicates with IoT devices **indirectly** via an MQTT Broker
Request-Response Paradigm	**Publish-Subscribe** Paradigm
IoT Devices must always be ready to receive communication	IoT Devices choose when to receive communication
Higher Power Requirement	Low Power Requirement

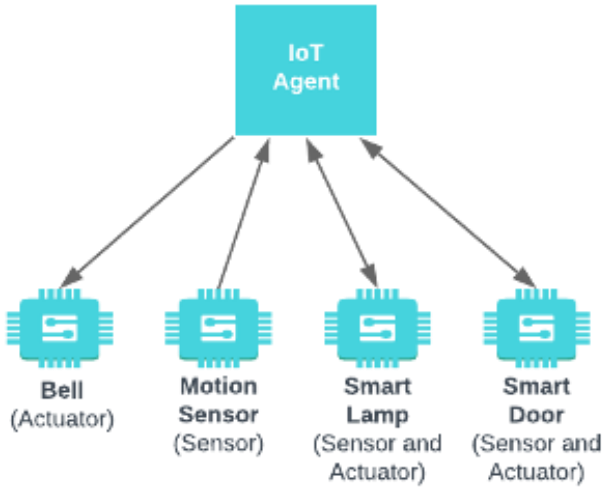

Fig. 2.2 (a) HTTP Protocol (Request-Response)

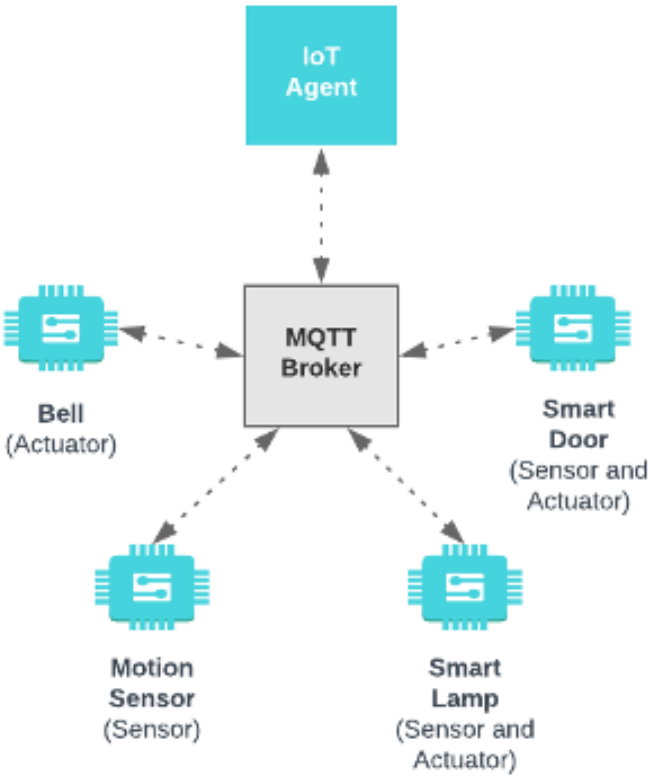

Fig. 2.2 (b) MQTT Protocol (Publish-Subscribe)

2.3 ADAFRUIT MQTT SERVER

2.3.1 ADAFRUIT

Adafruit IO is the easiest way to get the projects onto the Internet of Things. Adafruit IO uses a special type of MQTT Topic named a Feed to store data along with metadata (information about the data). In Adafruit topics are regarded as feed. Data from the various sensors are sent to Adafruit as feeds. These data can be monitored in Adafruit in laptop or smartphone.

2.3.2 SETUP OF ADAFRUIT ACCOUNT

- First in the Adafruit IO make an account. Adafruit is a free MQTT broker. The feeds are the sensors and switch.
- In Adafruit Click on the actions and create new dashboard. New project is created .
- In Adafruit plus icon on the right corner is used to add the feeds. Toggle is used for on/off the appliances with the feed name. Thus many number of feeds can be created in Adafruit. MQ-2 sensor can be monitored by using gauge.
- The username and active key of Adafruit is used in the code to access to the Adafruit server and this must be maintained secret to avoid security hacks.

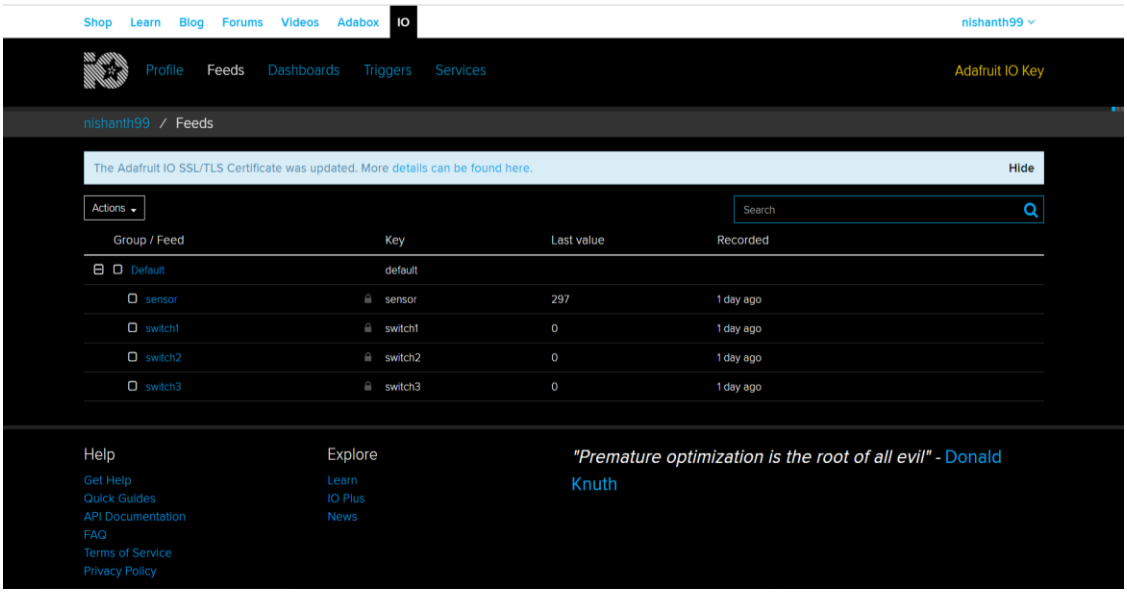

Fig. 2.3.2 (a) Feeds created in the Adafruit

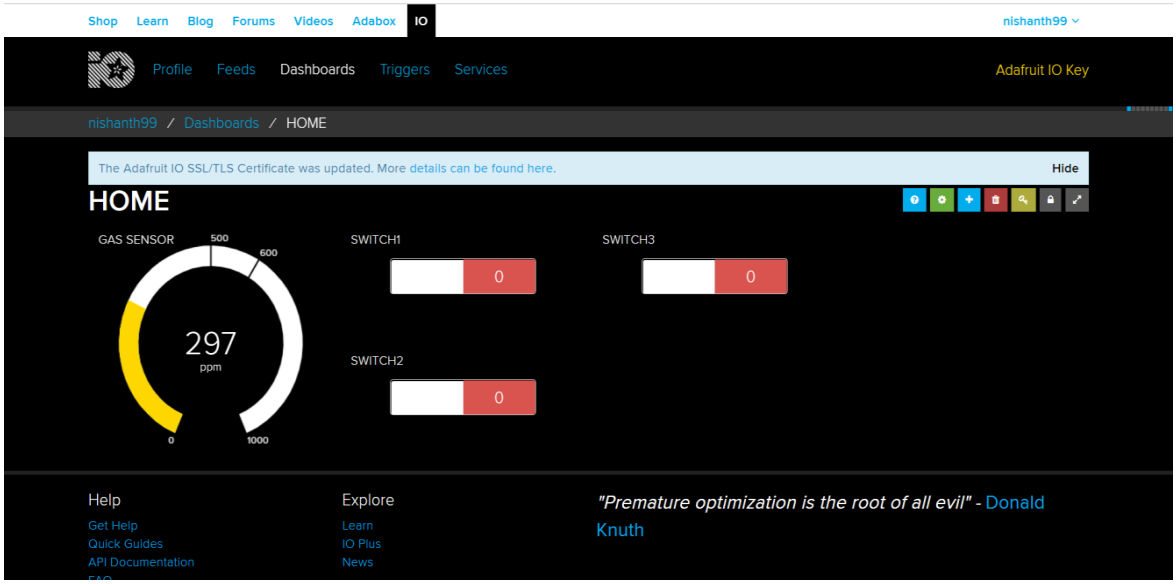

Fig. 2.3.2 (b) Dashboard containing the feeds

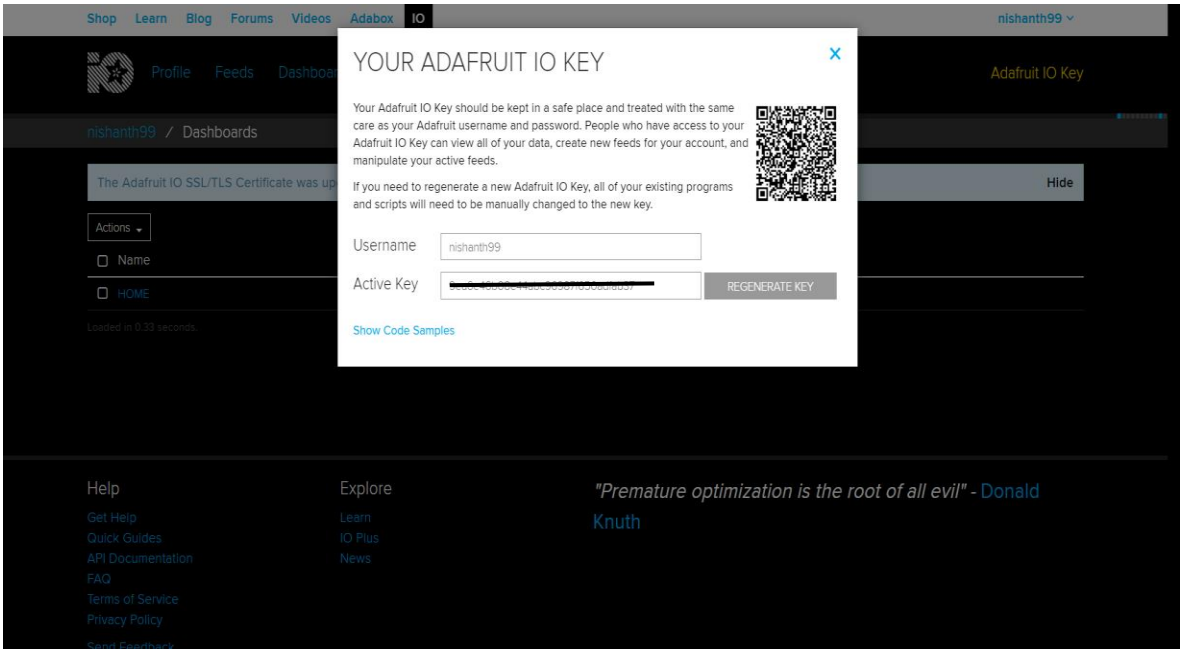

Fig. 2.3.2 (c) Username and Active key

CHAPTER 3
FLOW CHART OF WORKING MODEL

3.1 BLOCK DIAGRAM

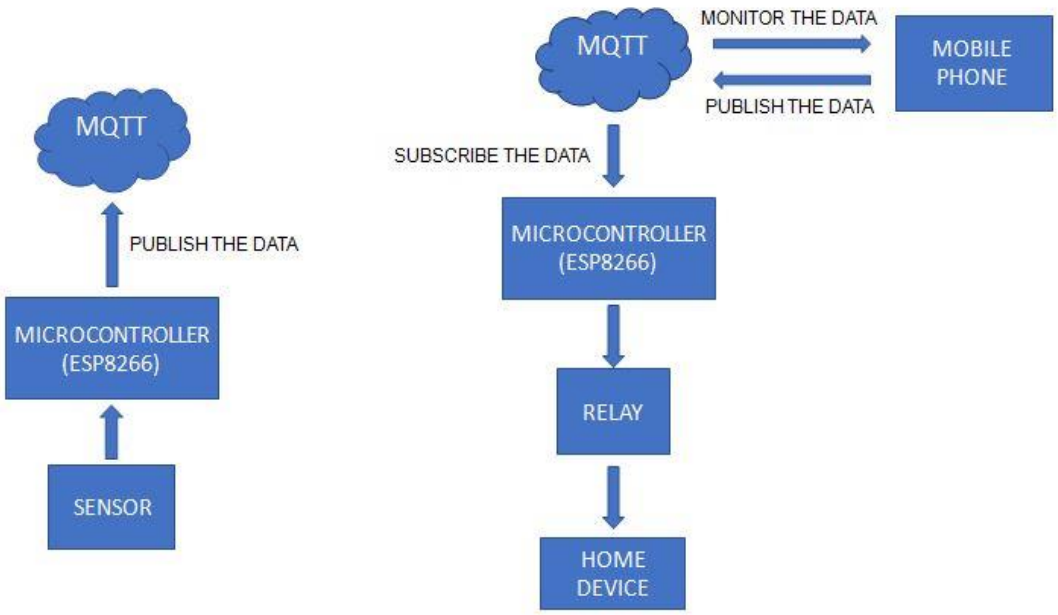

Fig. 3.1 Block Diagram

3.2 WORKING PRINCIPLE

The sensor is connected to the microcontroller (ESP8266) and in turn microcontroller is connected to MQTT server which is the Adafruit server via internet. Initially sensor senses the gas level and outputs the analog value. The analog value is processed by microcontroller and publishes the data to the Adafruit server and it also monitors the data of the gas sensor. When the gas sensor value crosses the threshold value, it turns on the home device which is nothing but the exhaust fan which is used to suck the gas out of the room. We can monitor the data of gas sensor through mobile or computer and control the home devices. We can publish the data to Adafruit server. Already microcontroller is subscribed to the feeds in the Adafruit through programming, microcontroller receives the data that has been published through phone or laptop and turns the state of the home device on or off based on the data received with the help of relay. The microcontroller continuously checks for the sensor value and awaits for any subscription data from the server. This process is keeps on running all the time.

CHAPTER 4
HARDWARE COMPONENTS

4.1 NODE MCU ESP8266

NodeMCU is an open-source Lua based firmware and development board specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. 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.

4.1.1 PIN CONFIGURATION

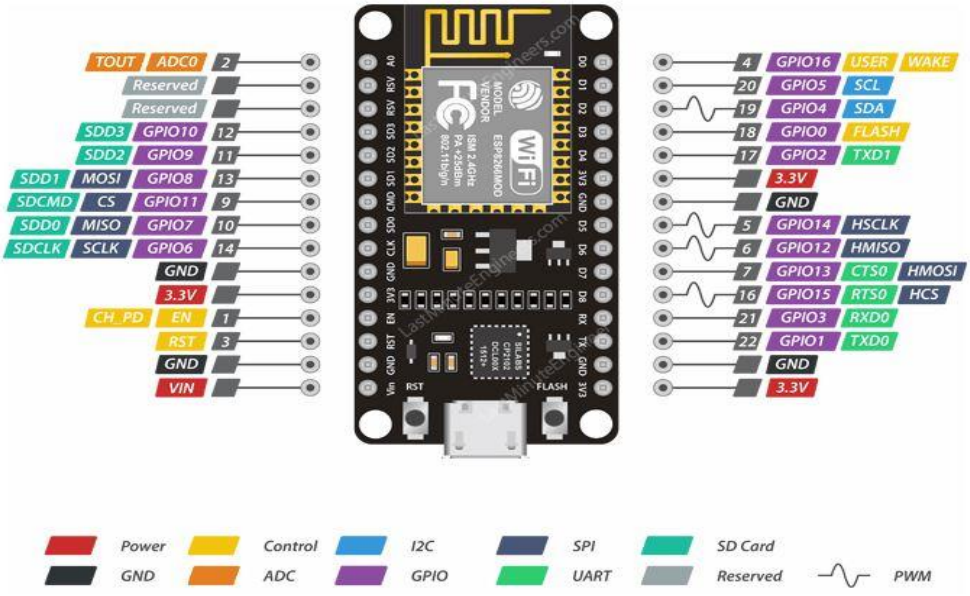

Fig. 4.1.1 Pin configuration of NodeMCU ESP8266

1. POWER PINS: There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.

2. GND: It is a ground pin of ESP8266 NodeMCU development board.

3. I2C PINS: are used to hook up all sorts of I2C sensors and peripherals in your project. 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.

4. GPIO PINS: ESP8266 NodeMCU has 17 GPIO pins which can be assigned to various 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.

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

6. UART PINS: ESP8266 NodeMCU has 2 UART interfaces, i.e. 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.

7.SPI PINS: 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

8.SDIO PINS: ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards.

9.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, i.e., between 100 Hz and 1 kHz.

10.CONTROL PINS: These are used to control ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

•EN pin – The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.

•ST pin – RST pin is used to reset the ESP8266 chip.

•WAKE pin – Wake pin is used to wake the chip from deep-sleep

4.1.2 TECHNICAL SPECIFICATION

- 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

4.2 MQ-2 GAS SENSOR

Gas Sensor(MQ2) module is useful for gas leakage detection (home and industry). It is suitable for detecting H2, LPG, CH4, CO, Alcohol, Smoke or Propane. Due to its high sensitivity and fast response time, measurement can be taken as soon as possible. The sensitivity of the sensor can be adjusted by potentiometer.

Fig. 4.2 MQ-2 Sensor

4.2.1 PIN CONFIGURATION

Fig. 4.2.1 Pin configuration of MQ-2 sensor

1.VCC supplies power for the module. You can connect it to 5V output from your Arduino.

2.GND is the Ground Pin and needs to be connected to GND pin on the Arduino.

3.D0 provides a digital representation of the presence of combustible gases.

4.A0 provides analog output voltage in proportional to the concentration of smoke/gas

4.2.2 TECHNICAL SPECIFICATIONS

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

4.2.3 WORKING PRINCIPLE

The gas sensitive material used in the MQ-2 gas sensor is SnO2, a low electrically conductive material in clean air. When there is combustible gas in the surrounding air, the electrical conductivity of the sensor will increase with the higher intensity of the combustible gas. Here, we can convert the changing electrical conductivity into output signal by building a simple circuit.

When tin dioxide (semiconductor particles) is heated in air at high temperature, oxygen is adsorbed on the surface. In clean air, donor electrons in tin dioxide are attracted toward oxygen which is adsorbed on the surface of the sensing material. This prevents electric current flow. In the presence of reducing gases, the surface density of adsorbed oxygen decreases as it reacts with the reducing gases. Electrons are then released into the tin dioxide, allowing current to flow freely through the sensor.

4.2.4 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. 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. 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.

4.3 BUZZER

A buzzer or beeper is an audio signalling device,which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

4.4 RELAY

A relay is an electrically operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals. The switch may have any number of contacts in multiple contact forms, such as make contacts, break contacts, or combinations thereof.

Relays are used where it is necessary to control a circuit by an independent low-power signal, or where several circuits must be controlled by one signal. Relays were first used in long-distance telegraph circuits as signal repeaters: they refresh the signal coming in from one circuit by transmitting it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

CHAPTER 5
SOFTWARE COMPONENTS

5.1 ARDUINO 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.

Fig. 5.1 Screenshot of the Arduino IDE showing a blink program

CHAPTER 6
HARDWARE IMPLEMENTATION

6.1 CIRCUIT DIAGRAM

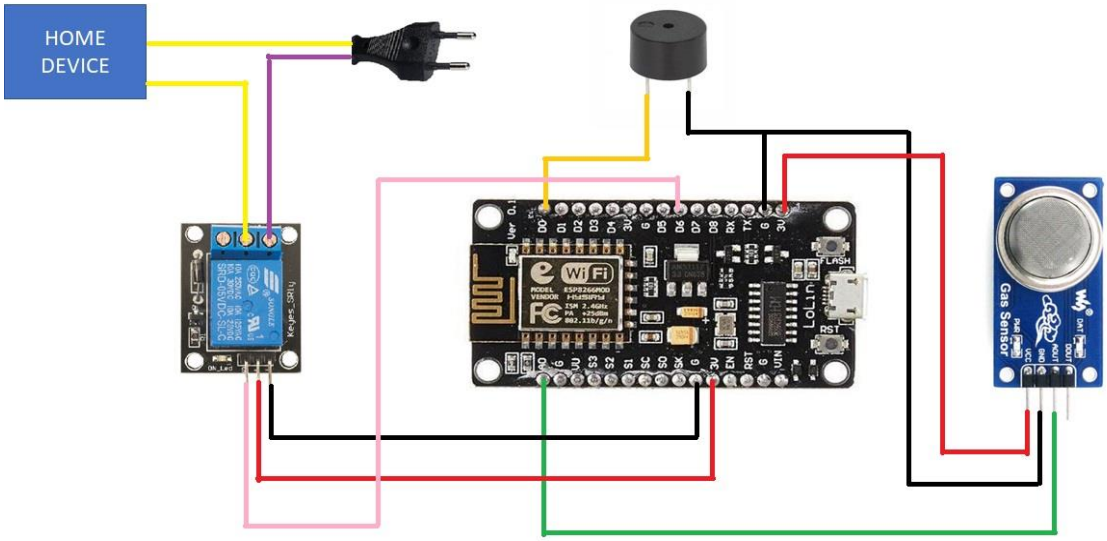

Fig. 6.1(a) Circuit Diagram

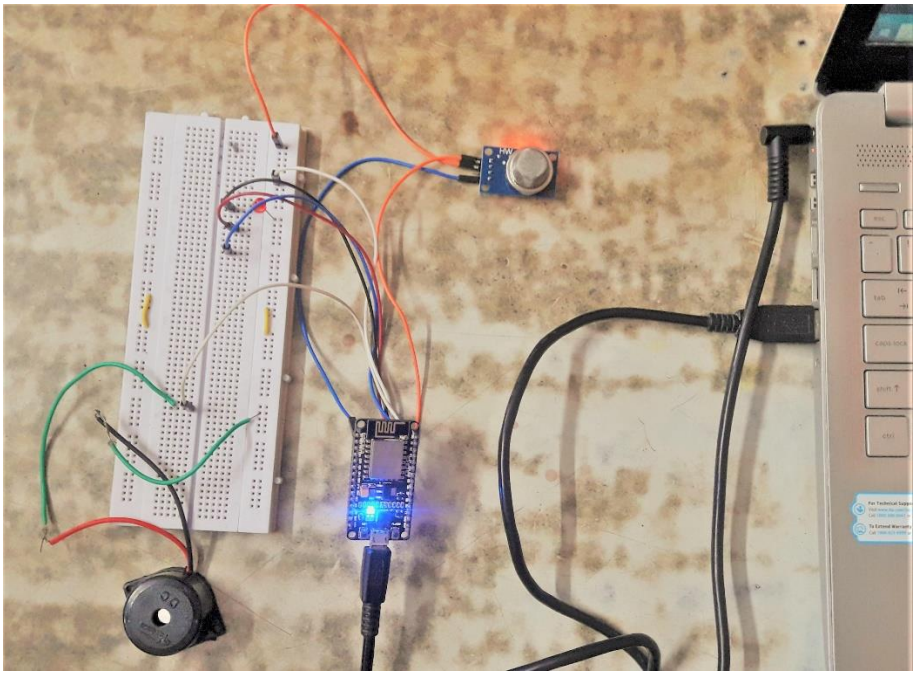

Fig.6.1 (b) Nodemcu connected with sensor and buzzer

6.2 RESULTS AND DISCUSSION

As the circuit is excited with the power supply, it starts publishing the sensor data to Adafruit server. In Adafruit server, the gas sensor value is indicated by a gauge meter as shown in Fig 6.2 (a).

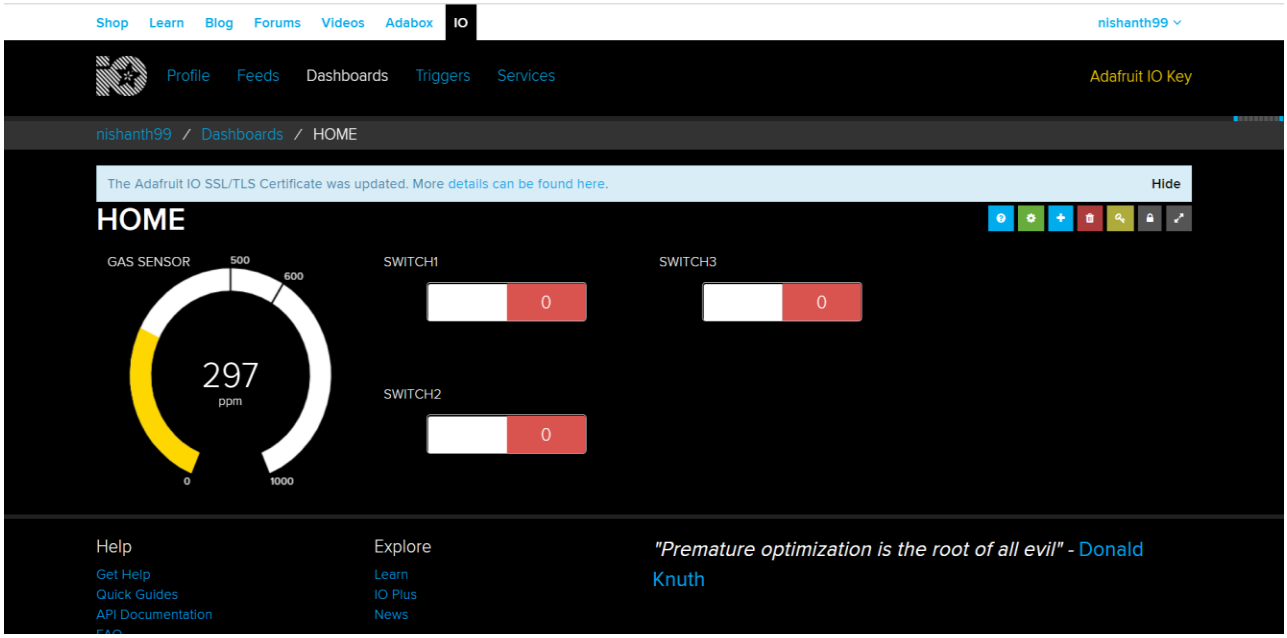

Fig. 6.2 (a) Showing Adafruit displaying the sensor data

When the sensor value crosses the threshold value, the microcontroller detects the change over the threshold value and turns on the exhaust fan and it is also printed in the serial monitor as shown in Fig 6.2 (b).

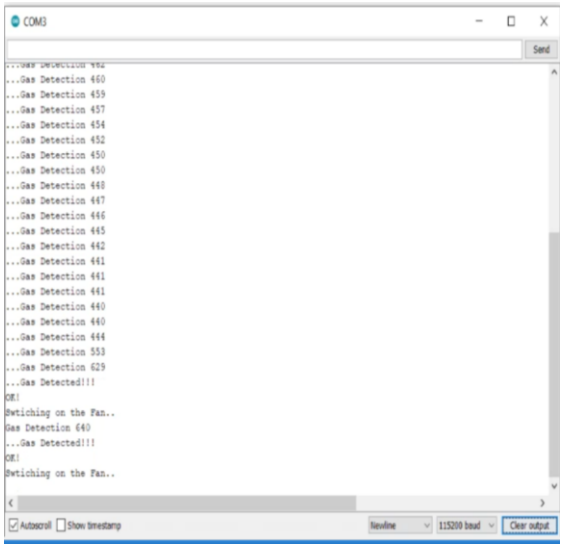

Fig. 6.2 (b) Arduino serial monitor

When the microcontroller detects the threshold value and turns on the exhaust fan, parallelly on the Adafruit server switch 1 which corresponds to the exhaust fan device is turned on by indicating value 1 on the switch 1 as shown in Fig 6.2 (c)

Fig. 6.2 (c) Switch 1 is turned on the Adafruit server

As soon as the sensor value crosses the threshold value, exhaust fan is immediately turned on and it is parallelly reported to Adafruit server as well. It is also possible to control home devices by turning on the switch available on Adafruit Home Dashboard by sliding the button.

CONCLUSION

Health Safety is a major issue in current era and good safety systems are needed to be implemented in places which are related to work, education and living. This project modifies the existing safety model installed in the industrial areas and this system can also be used in houses and offices.

The purpose of this project is to avoid accidents regarding gas leakages and also provide home automation at the cost of a movie ticket. This project detects gas leakage within a fraction of seconds and turns on the exhaust fan. It is also proved to efficient and low power consumption. It is a cost - efficient model, based on NodeMCU and gas sensors.

It provides real time information available on internet for faster accessing with a gas sensor that can detect various other hazardous gases. The advantage of this auto detection and alerting system over the traditionally used manual method is that it offers quickest response time possible and accurate detection of an emergency situation and in turn helps in faster diffusion of the critical situation.

FUTURE WORKS

- This project can be used as a Intelligent Gas Detection System. Here system can detect gas leakges as well as measure the gas level of the gas cylinder. There are many methods available for booking a Gas Refill, methods include online booking, telephonic booking etc. It will be difficult situation for the one who uses LPG gas for cooking regularly. So this new system aims to present a new system which automatically books a cylinder when the gas is about to empty is by sending a notification to the gas agency using wifi using Internet of Things approach.

- This project can be used for security purpose. A various number of sensos can be interfaced like PIR sensor, camera to add security to the home or industry. So, home automation and security at the hands of the user adds great value to the current society.

- This project can be used in food processing industries. Consider in a industry, some kind of goods is stored in a room. The important parameter for the goods to remain good inside the room is the level of the gas (say gas X) must be greater than a certain value. In such cases, we don’t need to monitor the sensor value on hourly basis. Instead, these kind of systems can be introduced to monitor the sensor value on any device using internet of things and take necessary steps in case of emergency.

REFERENCES

[1] P.Meenakshi Vidya, S.Abinaya, G.Geetha Rajeswari, N.Guna ,“Automatic LPG detection and hazard controlling “ published in April 2014.

[2] K.Padmapriya, Surekha, Preethi, “Smart Gas Cylinder Using Embedded System”, published in 2014.

[3] C.Selvapriya, S.Sathyaprabha, M.Abdul rahim,” LPG leakage monitoring and multilevel alerting system”, published in 2013.

[4] L.K.Hema, Dr.D.Murugan, M.Chitra,” WSN Based Smart System for LPG Detection & Combustible Gases”, published in 2013.

APPENDIX A

PROGRAM CODE

```
//Including libraries
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

// Pin definiton
//Relays for switching appliances
#define Relay1          D6
#define Relay2          D2
#define Relay3          D1

//buzzer to know the status of MQTT connections and for sensor value
#define buzzer          D0

//Analog pin to read the incoming analog value from different sensors.
#define analogpin       A0

//Wifi Access Point
#define WLAN_SSID       "Airtel"
#define WLAN_PASS       "airtelwifi"

// Adafruit.io Setup
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "nishanth99"
#define AIO_KEY         "9ea6e46b08e44abc96987f650adfab37"
```

```
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

//Feeds
Adafruit_MQTT_Publish sensor = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/sensor");
Adafruit_MQTT_Publish light11 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/switch1");

// Setup a feed called 'onoff' for subscribing to changes.
Adafruit_MQTT_Subscribe light1 = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/switch1");
Adafruit_MQTT_Subscribe light2 = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/switch2");
Adafruit_MQTT_Subscribe light3 = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/switch3");

//sketch code
void MQTT_connect();

void setup() {
  Serial.begin(115200);

  delay(10);

  pinMode(buzzer, OUTPUT);
  pinMode(Relay1, OUTPUT);
  pinMode(Relay2, OUTPUT);
  pinMode(Relay3, OUTPUT);
```

```
  pinMode(A0, INPUT);

  Serial.println(F("Adafruit MQTT demo"));

  // Connect to WiFi access point.
  Serial.println(); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("WiFi connected");
  Serial.println("IP address: "); Serial.println(WiFi.localIP());

  // Setup MQTT subscription for onoff feed.
  mqtt.subscribe(&light1);
  mqtt.subscribe(&light2);
  mqtt.subscribe(&light3);

}

uint32_t x = 0;




void loop() {
  MQTT_connect();
```

```
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(2000))) {
  if (subscription == &light1) {
    int Light1_State = atoi((char *)light1.lastread);
    if(Light1_State==1)
    {
      Serial.println(F("Swtiching on the Fan.. "));
    }
    else
    {
      Serial.println(F("Swtiching off the Fan.. "));
    }
    digitalWrite(Relay1, Light1_State);

  }
  if (subscription == &light2) {

    int Light2_State = atoi((char *)light2.lastread);
    if(Light2_State==1)
    {
      Serial.println(F("Swtiching on the Device 2.. "));
    }
    else
    {
      Serial.println(F("Swtiching off the Device 2.. "));
    }
    digitalWrite(Relay2, Light2_State);
  }
  if (subscription == &light3) {

    int Fan1_State = atoi((char *)light3.lastread);
    if(Fan1_State==1)
```

```
    {
      Serial.println(F("Swtiching on the Device 3.. "));
    }
    else
    {
      Serial.println(F("Swtiching off the Device 3.. "));
    }
    digitalWrite(Relay3, Fan1_State);
  }

}

// Now we can publish stuff
Serial.print("Gas Detection ");
Serial.println(analogRead(analogpin));
Serial.print("...");
int Value = 644;
if(! sensor.publish(Value))
{
   Serial.println(F("Failed"));
} else {
  Serial.println(F("OK!"));

}
if(Value>580)
{
  Serial.println("Gas Detected!!!");
  if (! light11.publish(1)) {
  Serial.println(F("Failed"));
}
else {
  Serial.println(F("OK!"));
}
```

```
    }
}

// Function to connect and reconnect as necessary to the MQTT server.
// Should be called in the loop function and it will take care if connecting.
void MQTT_connect() {
  int8_t ret;

  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 3;
  digitalWrite(buzzer, HIGH);
  delay(200);
  digitalWrite(buzzer, LOW);
  delay(200);
  digitalWrite(buzzer, HIGH);
  delay(200);
  digitalWrite(buzzer, LOW);
  delay(200);
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
    Serial.println(mqtt.connectErrorString(ret));
    Serial.println("Retrying MQTT connection in 5 seconds...");
    mqtt.disconnect();
    delay(5000);  // wait 5 seconds
    retries--;
```

```
  if (retries == 0) {
     // basically die and wait for WDT to reset me
     while (1);
    }
  }
  Serial.println("MQTT Connected!");
  digitalWrite(buzzer, HIGH);
  delay(2000);
  digitalWrite(buzzer, LOW);
}
```