SEWAGE IOT DEVICE FOR SEWAGE GAS MONITORING AND ALERT SYSTEM

A MINOR PROJECT REPORT

Submitted by

Ridhima Bahl - RA1511004010695

Nitin Asthana - RA1511004010717

Under the guidance of

Mr. M. Manigandan

(Assistant Professor (O.G), Department of Electronics & Communication Engineering)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS & COMMUNICATION ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Kancheepuram District

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

MAY 2018

BONAFIDE CERTIFICATE

Certified that this minor project report titled **"SEWAGE IOT DEVICE FOR SEWAGE GAS MONITORING AND ALERT SYSTEM"** is the bonafide work of "Ridhima Bahl (RA1511004010695) and Nitin Asthana (RA1511004010717)", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE OF PROJECT GUIDE

Mr. M. Manigandan
Assistant Professor (O. G)
Department of Electronics & Communication Engineering

SIGNATURE OF PROJECT COORDINATOR

Dr. M. Neelaveni Ammal
Assistant Professor (Sr. G),
Department of Electronics and Communication Engineering

ABSTRACT

This project aims at providing smart solutions to monitor the poisonous sewage gases and works on live sewage detection and monitoring. Whenever a certain threshold level will be crossed, an alert will be sent to the observer who is examining the conditions from a remote location and will forward the various gas values indicating whether it is safe for the worker to clean in that environment or not. The IOT remote monitoring equipment will be integrated with an IOT platform. This shall send a prior alert to the sewage worker to ensure their safety. Various type of sensors are utilized to monitor parameters present in sewage like gas, temperature etc. When the threshold value is lesser than the sensed values, this system alerts the sewage worker/cleaner regarding concentrations of different poisonous gases help sewer them to protect themselves from risk and harmful disease.

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our guide, **Mr. M. Manigandan** for his valuable guidance, consistent encouragement, personal caring, timely help and providing us with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support for completing this research work.

We would also like to thank our project coordinator **Dr. M. Neelaveni Ammal** for providing valuable insights in the project.

Table of Contents

Table of Figures

CHAPTER 1

I.INTRODUCTION

1.1. Background-

Sewage environment IOT device and sewage IOT platform to monitor poisonous gas has been proposed as a solution to help the sewer workers who put their lives at jeopardy, and ensure minimal health risk. Because of these poisonous gases, the death rate of sewer workers has been increased in the recent years .The lack of treatment of sewage after crossing dangerous levels work leads to the deaths of thousands of sewage cleaners throughout the year from accidents and various diseases such as hepatitis and typhoid due to sudden or sustained exposure to hazardous gases. Septic tanks are quite common in residential and industrial areas to cater to sewage wastes.

Natural decomposition and mixture of sewage leads to production of sewage gases. These gases can be toxic if inhaled in high concentrations or for a prolonged period of time. Septic tank gases contain methane, hydrogen sulphide (H2S), carbon dioxide, sulphur dioxide, ammonia, nitrogen dioxide and traces of carbon monoxide.

1.2. Project Objectives-

In order to evaluate the gases which are present in sewage environment, sensors has been used to analyze the amount of hazardous gas and send an alert. The hazardous gases like hydrogen sulphide, methane and carbon monoxide turns out from sewage are sensed by gas sensors every moment and updated when it surpass the normal grade.

II.Gases to be monitored

CarbonMonoxide(CO):

The Carbon Monoxide which is a colorless and unfragrant gas is given out when charcoal is burnt in open areas. Similarly it is generated when gasoline/diesel generators or other fuel-driven tools are used in adequately ventilated workplaces. Unmasking to carbon monoxide on sewage environment at concentrations over 350ppm can cause blurring, fainting on exertion and collapse. An aerial concentration of carbon monoxide more than 1200ppm is immediately dangerous to health and life.

Hence, this is one of the gases that needs to be monitored.

Methane (CH4):

Methane is commonly produced when nuclear matter is crumbled by a variety of bacterial processes. It is a colorless and ignitable gas that can cause fire and explosion. The accumulation of methane in a poorly ventilated area will lead to

displacement of normal air and results in an oxygen deficient environment for the sewage workers. This is the other gas being monitored by this system.

1.3. Scopes-

The project aims at designing a prototype for monitoring a sewage plant or septic tank in realtime for keeping a check on concentration levels of gases.

- The designed system can be installed in various sewage facilities, both rural and urban.
- The system can be used in domestic or industrial plants.
- For remote access of concentration or ppm levels, thingspeak platform can be accessed from anywhere in the world via internet.

1.4. Project Management-

This project constituted development of an IOT platform as its major part as well as the hardware to monitor the setup. Work was effectively divided and coordinated.

1.5.Experimentation

In this step, the necessary equipment and

Materials like Arduino, GSM and sensors were tested and calibrated. We studied similar projects, gathering the information required to execute the system.

1.6. Design

In this phase, we designed the layout of the sewage monitoring system. The necessary features like real time graph plotting and alert sending facilities were included.

1.7. Development and Testing-

Here, the development of thingspeak channels was performed. The bugs in the Arduino code and setting up of GPRS was done using AT commands. Errors were identified and removed.

1.8. Real-World Testing- Finally, our system was ready to be tested in the real sewage plants or septic tanks.

1.9 FLOWCHART

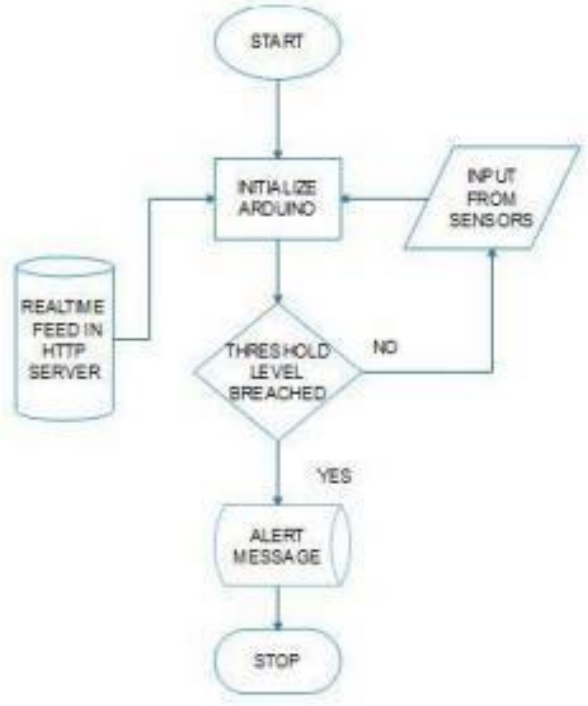


Figure 1.1: Flowchart of system

1.10 DIAGRAM

Figure 1.2: Block Diagram

CHAPTER 2

LITERATURE REVIEW

In the year of 2008, Chen Peijiang and Jiang Xuehhua, “Design and implementation of Remote Monitoring System Based on GSM”, this paper focuses on the wireless monitoring system, because the wireless remote monitoring system has more applications a remote monitoring system based on SMS through GSM. Literature survey was conducted upon gas detection system and modelling using IOT techniques. We referred the paper which was released in 2017 titled as IOT based Gas Detection and Monitoring System Using MQ-2 Sensor. It emphasized on using a microcontroller for detecting and alerting based on the levels of the toxic gases. Water quality is the important factor need to be considered in today's world. Water quality is affected by both point and non-point sources. Poor quality of water causes lots of diseases. We referred the paper Real time Water Monitoring System Using IOT for water quality study purposes. We also referred several other IOT based gas monitoring systems which used different microcontrollers depending upon their need. In the year of 2008, LIU zhen-ya, WANG Zhen-dong and CHEN Rong released the paper “Intelligent Residential Security Alarm and Remote Control System Based On Single Chip Computer”, and it emphasis on Intelligent residential burglar alarm, emergency alarm, fire alarm, toxic gas leakage remote automatic sound alarm and remote control system, these are based on

89c51 single chip computer, it is also known as 8051 microcontroller. This device was designed in such a way that it sends SMS to emergency number given or to the police hotline number in case of emergencies, it can be a voice call and it can even provide the location of the site. It was further expanded that this device can also be utilized for electricity control through phone.

CHAPTER 3

WORKING

The IoT remote monitoring is integrated to IoT platform which can send in prior alert for the sewage worker so that they can ensure their safety before they start working in such sewage cleaning environment. It uses various type of sensors to monitor parameters present in sewage like gas, temperature etc. When the threshold value is lesser than the sensed values, it will give us alert message which will help sewer workers to protect their lives from risk and harmful disease. This project uses MQ4 and MQ7 sensors for methane and Carbon Monoxide respectively. The values obtained from the sensors are in units of voltage. Thus, we use an appropriate code for converting voltage values to ppm (parts per million) or percentage value to plot a legible graph on the IOT platform – 'ThingSpeak'. The calibrated values in the form of ppm or percentage are sent to the ThingSpeak server via GSM module, making use of GPRS connectivity. The corresponding graph is displayed on the IOT platform indicating the corresponding accent and descent in the values over a range of time.

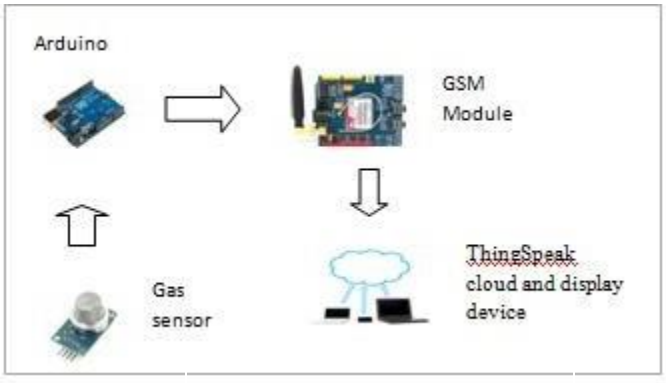


Figure 3.1: Working model

3.1 Calibrating the sensors

MQ series sensors use a small heater inside with an electro-chemical sensor in order to measure different kind of gases combinations. They can be calibrated, but, in order to do that a know concentration of the measured gas or gasses is needed for that. For industrial purpose calibrations are done in special metrology laboratories with accurate probes and tests. In our case, we will test it as it comes from the producer without any additional calibration or settings.

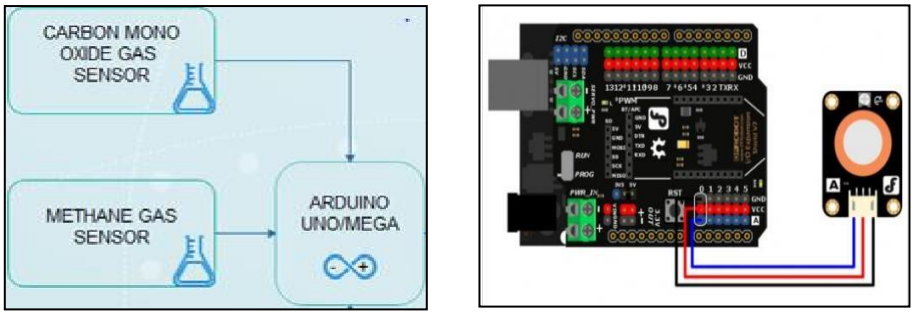


Figure 3.2 Connecting sensors to arduino

For detecting methane gas and CO gas the MQ-4 and MQ-7 sensors are appropriate. They are cheap and work well. The sensor itself returns a analog voltage that can be converted using an ADC. The converted value can be used in calculations to get the ppm value of the detected gas. The supply voltage for the sensor is 5 V. If a lower voltage is used, the heating element inside the sensor will not work correctly and the received values will be incorrect. First, the value of R0, which is the resistance of the sensor at a known concentration without the presence

of other gases, or in fresh air is calculated. Then the value of R0 displayed is used for calibrating voltage values to ppm or percentage values appropriately.

Resistance value of MQ-4 is difference to various kinds and various concentration gases. So, when using these components, sensitivity adjustment is very necessary. It is recommended to calibrate the detector for 5000ppm of CH4 concentration in air and use value of Load resistance (RL) about 20KΩ(10KΩ to 47KΩ). When accurately measuring, the proper alarm point for the gas detector should be determined after considering the temperature and humidity influence.

Resistance value of MQ-7 is difference to various kinds and various concentration gases. So, when using these components, sensitivity adjustment is very necessary. It is recommended to calibrate the detector for 200ppm CO in air and use value of Load resistance that (RL) about 10 KΩ (5KΩ to 47 KΩ). When accurately measuring, the proper alarm point for the gas detector should be determined after considering the temperature and humidity influence.

3.2 Connecting GSM Module to Arduino

There are two ways of connecting GSM module to arduino. In any case, the communication between Arduino and GSM module is serial. So we are supposed to use serial pins of Arduino (Rx and Tx). So if we are going with this method, we may connect the Tx pin of GSM module to Rx pin of Arduino and Rx pin of GSM

module to Tx pin of Arduino. We read it right ? **GSM Tx –> Arduino Rx** and **GSM Rx –> Arduino Tx**. Now connect the ground pin of arduino to ground pin of gsm module! So that's all! We made 3 connections and the wiring is over! Now we can load different programs to communicate with gsm module and make it work.

The problem with this connection is that, while programming Arduino uses serial ports to load program from the Arduino IDE. If these pins are used in wiring, the program will not be loaded successfully to Arduino. So we have to disconnect wiring in Rx and Tx each time we burn the program to arduino. Once the program is loaded successfully, we can reconnect these pins and have the system working!

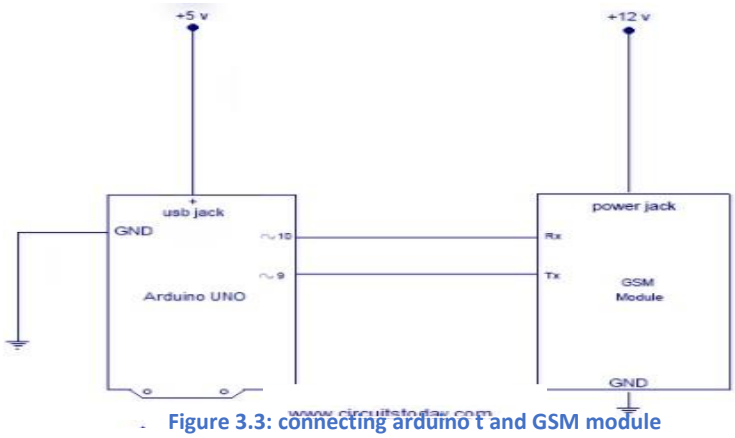


Figure 3.3: connecting arduino t and GSM module

3.3 Connecting GSM Module to ThingSpeak

ThingSpeak is an IoT platform that uses channels to store data sent from apps or devices. With the settings in Channel configurtion, we create a channel, and then send and retrieve data to and from the channel. We can make our channels public to share data. Using the REST API calls such as GET, POST, PUT, and DELETE, we can create a channel and update its feed, update an existing channel, clear a channel feed, and delete a channel. We can also use the MQTT Publish method to update a channel feed and MQTT Subscribe to receive messages when a channel updates.

ThingSpeak **Channel Access**

To read and write to a ThingSpeak™ channel, our application sends requests to the ThingSpeak server, either by issuing HTTP requests or using MATLAB® functions. Each ThingSpeak channel can have up to eight fields of data, in either numeric or alphanumeric format. A channel also has location information and a status update field.

Each channel data entry is stored with a date and timestamp and is assigned a unique entry ID (entry_id). We can retrieve stored data by time or by entry ID. Use the ThingSpeak API to process numeric data, which includes timescaling, averaging, median, summing, and rounding. We can create and update a

ThingSpeak channel by posting a feed with our API key and data by using HTTP POST. The channel feeds supports JSON, XML, and CSV formats for integration into applications.

Channel Property Settings

Change your channel properties in the **Channel Settings** tab:

- **Channel ID**: Auto-generated ID of our unique channel. Our application uses this ID to read data from the channel. We cannot change its value.
- **Name**: Enter a unique name for the ThingSpeak channel.
- **Description**: Enter a description of the ThingSpeak channel.
- **Field#**: Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to eight fields.
- **Metadata**: Enter information about channel data, including JSON, XML, or CSV data.
- **Tags**: Enter keywords that identify the channel. Separate tags with commas.
- **URL**: If we have a website that contains information about our ThingSpeak channel, specify the URL.
- **Elevation**: Specify the position of the sensor or thing that collects data in meters. For example, the elevation of the city of London is 35.052.

- **Show Location**: Check this box to enable entering device location data on this page. The result is a Google® map with a location pinpoint displayed in the channel view. Even if this box is cleared, we can still read and write latitude and longitude information using the API.
  - **Latitude**: Specify the position of the sensor or thing that collects data in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude**: Specify the position of the sensor or thing that collects data in decimal degrees. For example, the longitude of the city of London is -0.1275.
- **Show Video**: Check the box to include a video display on our channel view, using the following settings:
  - **YouTube/Vimeo**: Select our video service.
  - **Video URL**: Specify the full URL for your video.
- **Show Status**: Check this box to add a window to your channel view for status updates. For example, this window can show executed commands from a TalkBack App that is configured to log to this channel, or it can show information that we send with an API command using the status parameter as described in Write Data.

Related Topics

For alternative ways of setting channel properties, see Write Settings.

Search by Tag

We can use tags to sort and filter our public and private channels. Set tags in our **Channel Settings** by adding individual words or phrases separated by a comma. In **My Channels**, we can enter a search tag in the search box, and the channel view is filtered to show only channels with that tag.

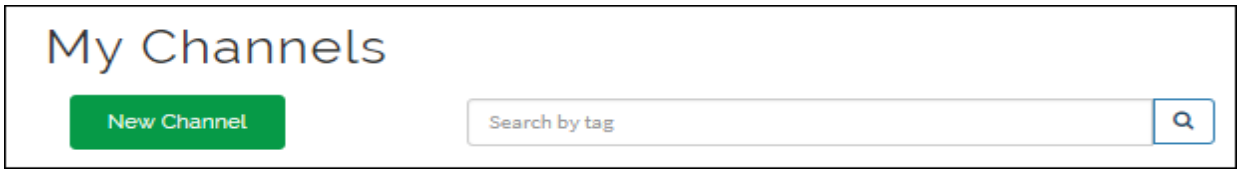


Figure 3.4 Channel Header

For example, consider a collection of ten channels for five different sensor types. Add a tag for each channel based on the sensor type, and we can rapidly filter to find the channels of interest.

Private, Shared, and Public Channels

We can select private, public, or shared channel. On the **My Channels** page, go to the **Sharing** tab. By default, our channel is private and requires a Read API key to access its feed. A public channel gives other users the ability to use our feed without a Read API key. We can also make a channel shared with specified users. Users who have access to shared channels can see only the private view of a channel; they cannot add channels, see alerts, or see license information.

In both the private and public views, we can use plugins to enable visualization and analysis. We can also choose to display different plugins in each view.

API Keys

When we read or write data to our channel using the ThingSpeak API or MATLAB code, we need the appropriate read and write permissions. The 16-digit API key allows us to read from a private channel and write to a channel. We do not need an API key to read from a public ThingSpeak channel. Account level API keys are described in User Accounts and Channels.

Write API Key

We use the Write API key to update a channel. If our Write API key is compromised, we can generate a new key. If we use MATLAB Analysis or MATLAB Visualization, the API info is displayed in the Help pane on the right.

To find our Write API key:

- Click **Channels** > **My Channels**.
- Select the channel to update.
- Select **API Keys** tab.

Read API Key

The Read API key enables our application to read data from the API. We can generate multiple Read API keys for different applications.

To get a Read API key:

- Click **Channels** > **My Channels**.
- Select the channel to update.
- Select **API Keys** tab.
- Click **Generate New Read API Key**.

API Endpoints

HTTP API Address

For non secure communication to ThingSpeak using HTTP, use the address:

http://api.thingspeak.com

For secure communication to ThingSpeak using HTTPS, use the address:

https://api.thingspeak.com

HTTP API Static IP Address

To communicate with the ThingSpeak HTTP server IP, use the address:

http://184.106.153.149

MQTT API Address

To communicate with the ThingSpeak MQTT broker at the port 1883, use the address:

mqtt.thingspeak.com

Cross-Domain XML

To post using cross-domain XML, use the address:

http://api.thingspeak.com/crossdomain.xml

Channel Access in MATLAB

To read data from a private channel in MATLAB, use the thingSpeakReadfunction:

thingSpeakRead(channelID,'ReadKey','Your.Read.API.Key.String');

To write data from MATLAB, use the thingSpeakWrite function:

thingSpeakWrite(channelId,data,'WriteKey','Your.Write.API.Key.String');

API Rate Limits

A free user can update a ThingSpeak channel every 15 seconds, and a paid user can update every 1 second. Updating more frequently results in an error. The time between read requests is not limited by ThingSpeak for any users.

API Caching

We can cache data when we transmit it via XML or JSON. Feeds returning more than 100 entries are cached for 5 minutes. This limit improves application performance. The last call or feeds specifying "results=100" or fewer are not cached, enabling production of real-time applications.

Channel Data Import

We can import data from a CSV file directly into a ThingSpeak channel.

To import data into a ThingSpeak channel:

- Click **Channels** > **My Channels**.
- Select the channel.
- Select **Data Import / Export** tab.
- Choose a file.
- Click **Upload**.

We must make sure that CSV file is properly formatted with all the relevant field headers defined as:

datetime,field1,field2,field3,field4,field5,field6,field7,field8,latitude,longitude,elevation,status

CSV imports are considered updates and consume messages in the same way as other updates. The CSV file is limited to 259,200 rows. We must send at least the datetime stamp and one field. The datetime stamp can be in these formats:

Format	Sample	Result Datetime
Epoch	1494878628 (seconds elapsed since 1-1-1970)	May 15, 2017 20:03:48 GMT
ISO 8601	2017-05-15T20:03:48-05:00	May 15, 2017 20:03:48 EST
MySQL	2017-05-15 20:03:48 UTC	May 15, 2017 20:03:48 EST

Figure 3.5: Formats

If the datetime stamp includes a GMT/UTC offset, ThingSpeak uses this information to import data. If the datetime stamp does not have a GMT/UTC offset, specify a time zone.

CHAPTER 4

SYSTEM DESIGN

4.1 HARDWARE SETUP :

In this proposed system we used the various gas sensors for example MQ4 and MQ7 for analyzing the presence of hazardous gases in the sewage. The setup at different nodes is controlled by single receiver end. The sensor produces range of values which is being emitted from the sewage to the controlling kit. Depending on the predefined condition, the output will be sent through the GSM module to the cloud server. Here we have used Things Speak server for our project.

Our prototype are deployed at strategic locations by analyzing the sewer mapand inspection demands. They are typically located at an entry point to the sewer. The dispensing schedule can be configured based on the application scenario. For instance, if the operators want to understand how in-sewer gas level changes over time, they can analyze the graph plotted by our system on the things speak server.

4.1.1 Methane Sensor:

This sensor is highly sensitive to CH4 and natural gas. It gives quick response with long life with stability. It is highly used in gas leakage detection system. Resistance value of this sensor is different to various concentration gases. So, sensitivity adjustment is very necessary.

- MQ-4 sensors are used to sense the Methane gas and it has an digital input signals Low and High.
- If the input signal is Low there is no gas affected then if the input signal is High gas is affected .
- It has an three states such as low, medium and high .
- If beyond the threshold limit (750ppm) in high state then the SMS alert will be sent to the workers.

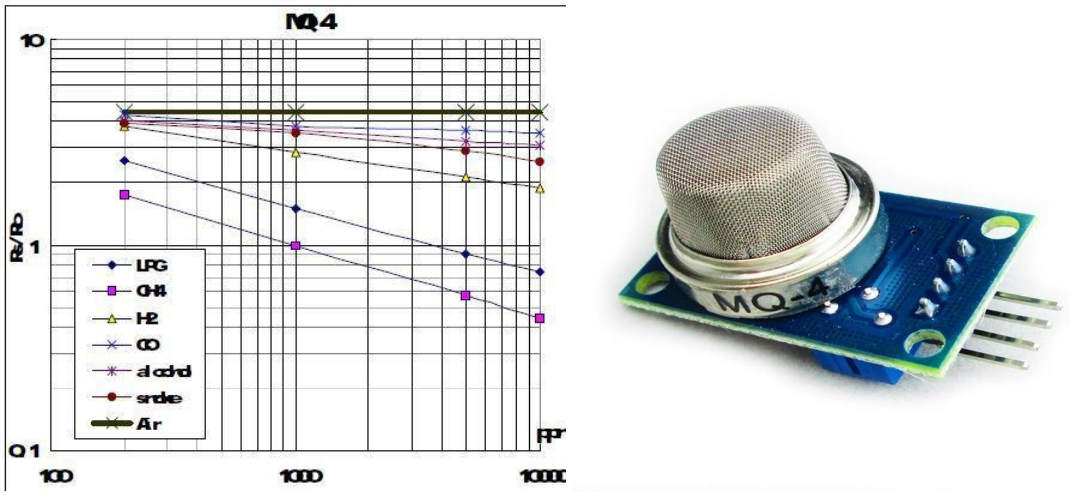


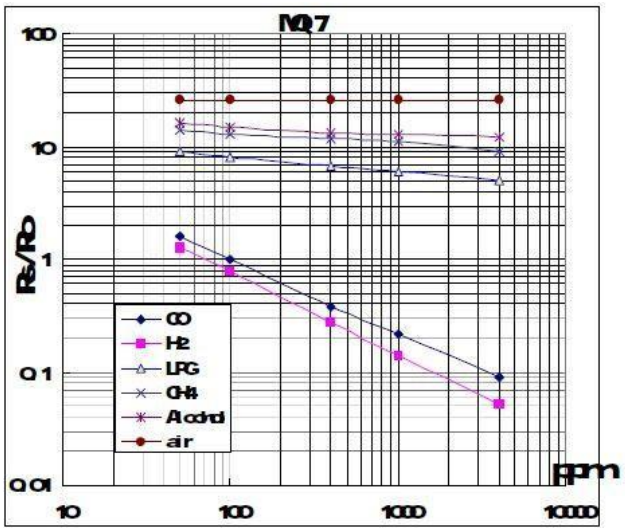
Fig. 4.1 Sensitivity characteristics graph of MQ4 and MQ4 sensor

4.1.2 Carbon Monoxide (CO):

The Carbon Monoxide which is a colorless and unfragrant gas is given out when charcoal is burnt in open areas. Similarly it is generated when gasoline/diesel generators or other fuel-driven tools are used in adequately ventilated workplaces.Unmasking to carbon monoxide on sewage environment at concentrations over 350ppm can cause blurring, fainting on exertion and collapse.

An aerial concentration of carbon monoxide more than 1200ppm is immediately dangerous to health and life.

- CO sensors are used to sense the carbon monoxide gas and it has an digital input signals Low and High.
- If the input signal is Low, there is no gas affected then if the input signal is High, the gas is affected.
- It has an three level such as low, medium, high beyond the threshold limit (35ppm) in high state then the SMS alert will be sent to the workers.



4.2 Sensitivity Characteristic of MQ7 and MQ7 sensor

4.1.3 GSM Module (SIM900A) :

GSM/GPRS Modem-RS232 is built with Dual Band GSM/GPRS engine-SIM900A, works on frequencies 900/ 1800 MHz . The Modem has RS232 interface, which allows you connect PC as well as microcontroller with RS232

Chip(MAX232). The baud rate is configurable from 9600-115200 through AT command. The GSM/GPRS Modem is having internal TCP/IP stack to enable you to connect with internet via GPRS. It is suitable for SMS, Voice as well as DATA transfer application in M2M interface. The onboard Regulated Power supply allows you to connect wide range unregulated power supply . Using this modem, you can make audio calls, SMS, Read SMS, attend the incoming calls and internet through simple AT commands.

- Sim900 A is used to upload our live readings to the sever hosting our website and can be used for monitoring the live feeds, by directly login in to our website.
- The Sim900 A generates a Alert message that sends an alert message to the recepient or worker responsible. When the level of sensor readings are breached.
- The GSM module uses a RS232 communication mode to communicate.
- To communicate with the Arduino uno we need to have TTL mode of communication, thus we used a MAX232 ic to convert the RS232 to TTL

Fig 4.3 GSM module (SIM900A)

4.1.3.1 Basic AT commands

AT command	Meaning
+CMGS	Send message
+CMSS	Send message from storage
+CMGW	Write message to memory
+CMGD	Delete message
+CMGC	Send command
+CMMS	More messages to send

4.1.3.2 AT commands for http, post and get

Demonstration	Syntax	Expert Results
Configure bearer profile 1	AT+SAPBR=3,1,"Contype","GPRS" AT+SAPBR=3,1,"APN","CMNET"	OK OK
To open a GPRS context.	AT+SAPBR=1,1	OK
To query the GPRS context	AT+SAPBR=2,1	+SAPBR:1,1,"10.89.193.1" OK
To close the GPRS context.	AT+SAPBR=0,1	OK
GPRS context is released by network		+SAPBR 1: DEACT

4.1.3.3 GET method

3.2 HTTPS Get Method

Demonstration	Syntax	Expert Results
Enable HTTPS	AT+HTTPSSL=1	OK
Init https service	AT+HTTPPARA="CID",1 AT+HTTPPARA="URL","https://www.example.com"	OK OK
GET session start	AT+HTTPACTION=0	OK
GET successfully		+HTTPACTION:0,200,1000
Read the data of HTTPS server	AT+HTTPREAD	+HTTPREAD:1000 … *//output the data to uart*

4.1.3.4 Post Method

Demonstration	Syntax	Expert Results
Enable HTTPS	AT+HTTPSSL=1	OK
Set parameters for HTTPS session	AT+HTTPPARA="CID",1 AT+HTTPPARA="URL","https://www.example.com"	OK OK
POST the data whose size is 100 bytes and the maximum latency time for inputting is 10000 ms. It is recommended to set the latency time long enough to download all the data in the latency time	AT+HTTPDATA=100,10000	DOWNLOAD …… *//It is ready to receive data from uart, and DCD has been set to low.* OK *//All data has been Received over, and DCD is set to high.*
POST session start	AT+HTTPACTION=1	OK
POST successfully		+HTTPACTION:1,200,0
Terminate https service	AT+HTTPTERM	OK

ARDUINO UNO:

The Arduino UNO is a widely used open-source microcontroller board based on the ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board features 14 Digital pins and 6 Analog pins. It is programmable with the Arduino IDE(Integrated Development Environment) via a type B USB cable.[4] It can be powered by a USB cable or by an external 9 volt battery, though it accepts voltages between 7 and 20 volts.

The Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer.

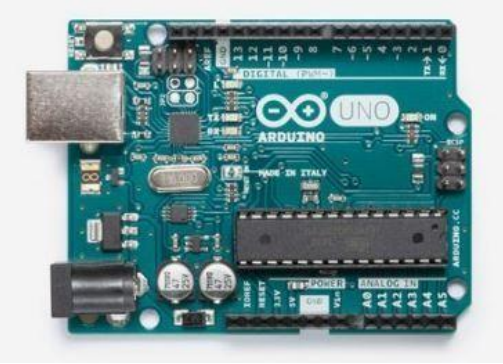


fig. 4.4 Arduino Uno

4.1.4 ThingSpeak (Cloud Server) :

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. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates.

In this project, we transmit the data taken by the sensors to the IOT platform by proving the API in the respective code. The data is plotted in the field section of the ThingSpeak platform. The GSM module sends the data over the server and as the reading changes the graph tha has been plotted changes.

The server can be accessed from anywhere with proper credentials that is needed to access the server.

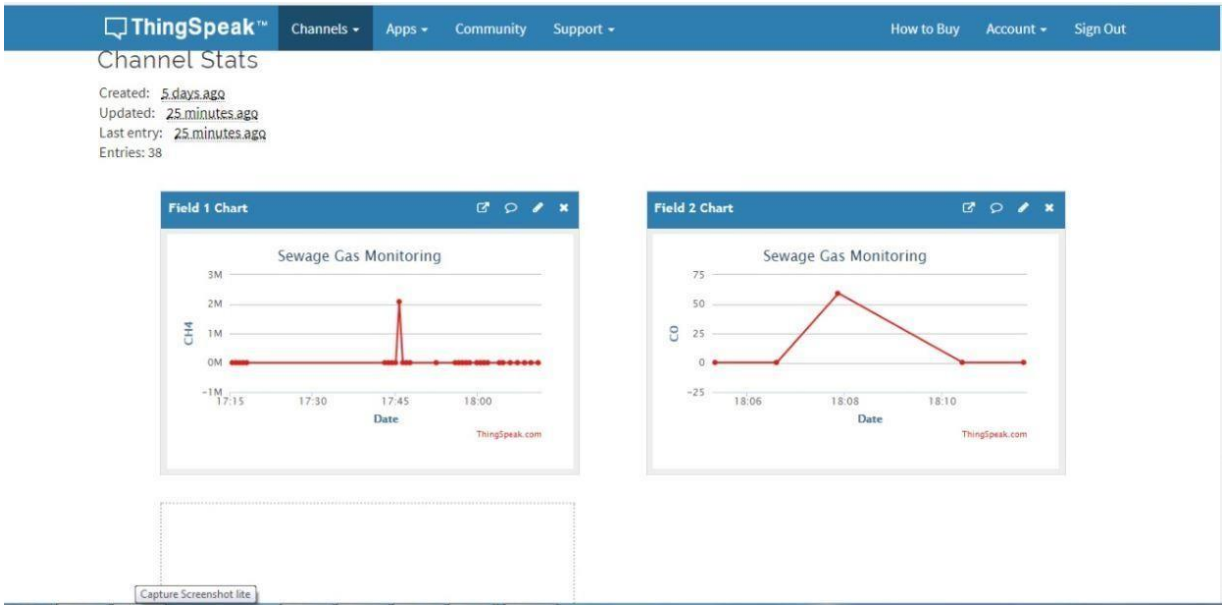


Fig 4.5 ThingSpeak output screen

CHAPTER 5

CODING AND TESTING

```
#include <String.h>

int gas_sensor1 = A0; //Sensor pin MQ4

int gas_sensor2 = A1; //Sensor pin MQ7

float m = -0.318; //Slope

float b = 1.133; //Y-Intercept

float R01 = 11.820; //Sensor Resistance in fresh air from previous code

float R02 = 5.4; //Sensor Resistance in fresh air from previous code

float ppm1;

 float ppm2;

void setup() {

  Serial.begin(9600); //Baud rate

  pinMode(gas_sensor1, INPUT);

  pinMode(gas_sensor2, INPUT);//Set gas sensor as input

}

void loop() {

  float sensor_volt1; //Define variable for sensor voltage

  float sensor_volt2;

  float RS_gas1; //Define variable for sensor resistance
```

```
  float RS_gas2;
  float ratio1; //Define variable for ratio
  float ratio2;
  float sensorValue1 = analogRead(gas_sensor1); //Read analog values of sensor
  float sensorValue2 = analogRead(gas_sensor2);
  sensor_volt1 = sensorValue1 * (5.0 / 1023.0); //Convert analog values to voltage
  sensor_volt2 = sensorValue2 * (5.0 / 1023.0);
  RS_gas1 = ((5.0 * 10.0) / sensor_volt1) - 10.0; //Get value of RS in a gas
  RS_gas2 = ((5.0 * 10.0) / sensor_volt2) - 10.0;
  ratio1 = RS_gas1 / R01;
  ratio2 = RS_gas2 / R02;// Get ratio RS_gas/RS_air
  float ppm_log1 = (log10(ratio1) - b) / m;
   float ppm_log2 = (log10(ratio2) - b) / m;//Get ppm value in linear scale
according to the the ratio value
   float ppm1 = pow(10, ppm_log1);
   float ppm2 = pow(10, ppm_log2);//Convert ppm value to log scale
   float percentage1 = ppm1 / 10000;
   float percentage2 = ppm2 / 10000;//Convert to percentage
  Serial.print("ppm1 = "); //Display "R0"
  Serial.println(percentage1); //Display value of R0
```

```
  delay(1000);
  Serial.print("ppm2 = "); //Display "R0"
  Serial.println(percentage2); //Display value of R0
  delay(1000);
   Send2Pachube();
 String str1="GET http://api.thingspeak.com/update?api_key=I92PNT1AHLZPC4T3&field1=" + String(percentage1);
 Serial.println(str1);//begin send data to remote server
  delay(4000);
  ShowSerialData();
  Serial.println((char)26);//sending
  delay(5000);//waitting for reply, important! the time is base on the condition of internet
  Serial.println();
  ShowSerialData();
  Send2Pachube();
  String str2="GET http://api.thingspeak.com/update?api_key=I92PNT1AHLZPC4T3&field2=" + String(percentage2);
```

```
  Serial.println(str2);//begin send data to remote server
  delay(4000);
  if (float(percentage2)>0.2) //&& (int(percentage1)<2300))
      {
        Serial.println("GSM Modem Message :");
         SendMessage();
         delay(5000);
      }
  ShowSerialData();
  Serial.println((char)26);//sending
  delay(5000);//waitting for reply, important! the time is base on the condition of
internet
  Serial.println();
  ShowSerialData();
 /* Serial.println("AT+CIPSHUT");//close the connection
  delay(100);
  ShowSerialData();*/
  if (Serial.available())
    Serial.write(Serial.read());
}
```

```
 void Send2Pachube()
{
  Serial.println("AT");
  delay(1000);
  Serial.println("AT+CPIN?");
  delay(1000);
  Serial.println("AT+CREG?");
  delay(1000);
  Serial.println("AT+CGATT?");
  delay(1000);
  Serial.println("AT+CIPSHUT");
  delay(1000);
  Serial.println("AT+CIPSTATUS");
  delay(2000);
  sSerial.println("AT+CIPMUX=0");
  delay(2000);
  ShowSerialData();
  Serial.println("AT+CSTT=\"internet\"");//start task and setting the APN,
  delay(1000);
  ShowSerialData();
```

```
  Serial.println("AT+CIICR");//bring up wireless connection
  delay(3000);
  ShowSerialData();
  Serial.println("AT+CIFSR");//get local IP adress
  delay(2000);
  ShowSerialData();
  Serial.println("AT+CIPSPRT=0");
  delay(3000);
  ShowSerialData();
Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");//start
up the connection
  delay(6000);
  ShowSerialData();
  Serial.println("AT+CIPSEND");//begin send data to remote server
  delay(4000);
  ShowSerialData();
}
void ShowSerialData()
{
  while(Serial.available()!=0)
```

```
    Serial.write(Serial.read());
}
void SendMessage()
{
  Serial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
delay(5000); // Delay of 1000 milli seconds or 1 second
Serial.println("AT+CMGS=\"+918939708340\"\r"); // Replace x with mobile
number
  delay(5000);
  Serial.println("Emergency Please Evacuate");// The SMS text you want to send
  delay(5000);
  Serial.println((char)26);// ASCII code of CTRL+Z
  delay(3000);
}
```

5.1 EXPLANATION:

5.1.1ALGORITHM:

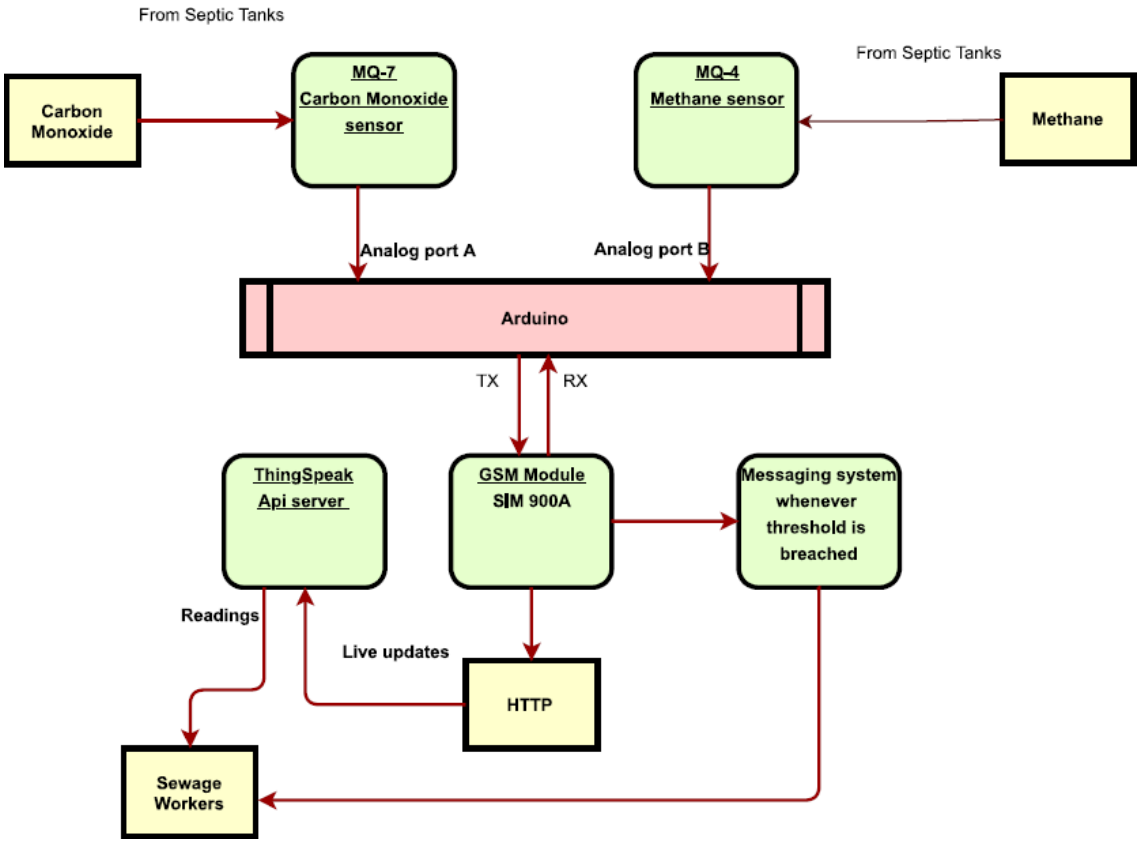


Fig. 5.1 Algorithm flow chart

5.1.2 PROCESS:

1.The above code is written in arduino.

Stage 1 :Conversion into PPM

2.We make use of analog pins A0 and A1 to read the respective sensor data.

3. At first the sensor readings are stored in the variable called sensorValue1 and sensorValue2

4. As the readings lied between 0-1023 we converted it to voltage and stored it in the varable sensor_volt1 and sensor_volt2, the formula used for this conversion is sensorValue1*(5.0/1023.0)

5. Next we find the resistance of the respective gas by using ((5.0 * 10.0) / sensor_volt1) - 10.0 and store them into the variable called RS_gas1 and RS_gas2.

6. Before taking the current rates and concentration of each gas we run a calibration code to find the R01 and R02 of the respective gas.

7. We find the ratio of each gas as ratio1 and ratio2 variables by using the formula RS_gas1/R01.

8. Now we convert the readings into ppm logarithm values using (log10(ratio1)-b)/m, and store them in ppm_log1 and ppm_log2.

9. We store the actual ppm1 and pp2 values in these variables respectively using pow(10,ppm_log1).

10. later we convert ppm into percentage easily by dividing it by 10000.

Stage 2: Sending the readings to Thingspeak server.

11. We designed a particular fuction named Send2Pachube() to send the data.

Here are the functions inside it:

These are used to initialize the Sim 900a: These are sent via the serial transmitter and receiver channel by directly printing in the serial monitor.

AT

AT+CPIN?

AT+CREG?

AT+CGATT?

AT+CIPSHUT

AT+CIPSTATUS

AT+CIPMUX=0

AT+CSTT=\"internet\”

AT+CIICR

AT+CIFSR

AT+CIPSPRT=0 \\ **This finally sets up the gsm module**

Connection starting:

AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"

AT+CIPSEND

12. While the connection is on by using the function Send2Pachube() we make a string variable str1 and by using the GET function for HTTP protocol, we print the url including the particular api key for the particular channel.

The api key for channel field 1 & 2 is : I92PNT1AHLZPC4T3

Forexample:str1="GET

http://api.thingspeak.com/update?api_key=I92PNT1AHLZPC4T3&field1=" +

String(percentage1);

CHAPTER 6

CONCLUSION

Septic tanks are quite common in residential and industrial areas to cater sewage wastes. Natural decomposition and mixture of sewage leads to production of sewage gases. These gases can be toxic if inhaled in high concentrations or for a prolonged period of time. Septic tank gases contain methane, hydrogen sulphide (H2S), carbon dioxide, sulphur dioxide, ammonia, nitrogen dioxide and traces of carbon monoxide. These toxic gases thus become dangerous especially for sewage workers and cleaners and sometimes lead to their death.

Hence to prevent this sewage poisonous gases, an IOT based monitoring system is designed which can monitor the levels of these poisonous gases present in the environment. In our project we have sensed carbon monoxide gas using sensor module MQ-7 and methane gas using sensor module MQ-4. These sensor modules detect the concentration levels. The readings obtained hence lie between analog read of 0 to 1023, hence by calibrating the sensors the concentration were respectively calculated in parts per million (ppm).Finally the ppm concentration values were converted to percentage and a graph was plotted accordingly using ThingSpeak api . 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, here particularly to plot the graph readings and then access it

from anywhere in the world. The code runs on Arduino and connects to the Thing Speak server via GSM modem to plot values in real time for viewing. When the threshold value is lesser than the sensed values, it will send the alert messages.

This system will help the sewage workers to protect their lives from risk and harmful disease like hepatitis and typhoid. This system with advanced technology based on IOT will significantly impact the lives of sewage workers. According to few recent updates in news many sewage workers lost their lives while doing their job by coming across the high concentration of such poisonous gases ,which once inhaled these poisonous gas leads to serious health issues. Hence this project will be able to lend a helping hand and aid the department of health and sanitation.

CHAPTER 7

FUTURE ENHANCEMENT

The lack of treatment of sewage after crossing dangerous levels, leads to the deaths of thousands of sewage cleaners throughout the year from accidents and various diseases such as hepatitis and typhoid due to sudden or sustained exposure to hazardous gases. Septic tanks are quite common in residential and industrial areas to cater to sewage wastes.

Through this project, we thus developed a prototype which can be further implemented in drainage or sewage system. At the initial stage only two gases were used for testing, but in future this technique will be used to detect other poisonous gases like hydrogen sulphide, esters, sulphur dioxide and nitrogen dioxide using sensor modules. The calibration of these sensors can also be done like as done for MQ-7 and MQ-4 while detecting carbon monoxide and methane. This can be then further integrated into a single device and then placed into sewage systems such as septic tanks etc. This is a real time monitoring system which will show the reading of concentration of poisonous gases with respect to time and these datas can be accessed from any place by the respective sewage worker. The values will be simultaneously uploaded to a web server from which all these data can be extracted after proper user authentication. The application of such system can certainly prove to be a leap towards the technical advancements and highly

beneficial for society. This will save sewage worker’s life. This would be a huge advantageous leap in the project as it would help to track all the poisonous gases and received data can be stored in the web server. This project this solves a social cause.

REFERENCES

1. Chen Peijiang and Jiang Xuehhua (2008) “Design and implementation of Remote Monitoring System Based on GSM*”. IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application.*
2. LIU zhen-ya, WANG Zhen-dong and CHEN Rong (2008) “Intelligent Residential Security Alarm and Remote Control System Based On Single Chip Computer”. *3rd IEEE Conference on Industrial Electronics and Applications.*
3. Rohan Chandra Pandey , Manish Verma, Lumesh Kumar Sahu (2017) “ Internet of Things (IOT) Based Gas Leakage Monitoring and Alerting System with MQ-2 Sensor”. *IJEDR | Volume 5, Issue 2 ISSN: 2321-9939.*
4. Vaishnavi V. Daigavane and Dr. M.A Gaikwad (2017) “Water Quality Monitoring System Based on IOT” *Advances in Wireless and Mobile Communications. ISSN 0973-6972 Volume 10, Number 5 (2017), pp. 1107-1116.*