1. INTRODUCTION

Internet of Things (IOT) is an upcoming technology that allows controlling hardware devices through internet. It is the internetworking of physical devices, buildings etc. Embedded with electronics, software, sensors and network connectivity such as cloud that enable these objects to collect and exchange data. It can also be defined as “The infrastructure of the information society”.

IOT allows the objects to get sensed or controlled remotely across existing network infrastructure creating window for more direct integration of the physical world into computer based systems. When it is supplemented with sensors and actuators, the technology becomes an illustration of more general class of cyber physical systems, which also cause to take place technologies such as smart grids, smart homes,intelligent transportation and smart cities.

An embedded system can be defined as a computing device that does a specific focused job. Appliances such as the air-conditioner, VCD player, DVD player, printer, fax machine, mobile phone etc., are examples of embedded systems. Each of these appliances will have a processor and special hardware to meet the specific requirement of the application along with the embedded software that is executed by the processor for meeting that specific requirement. The embedded software is also called “firm ware”. The desktop/laptop computer is a general purpose computer. You can use it for a variety of applications such as playing games, word processing, accounting, software development and so on.

Embedded systems do a very specific task, they cannot be programmed to do different things. Embedded systems have very limited resources, particularly the memory. Generally, they do not have secondary storage devices such as the CDROM or the floppy disk. Embedded systems have to work against some deadlines. A specific job has to be completed within a specific time. In some embedded systems, called real-time systems, the deadlines are stringent. Missing a deadline may cause a catastrophe-loss of life or damage to property. Embedded systems are constrained for power. As many embedded systems operate through a battery, the power consumption has to be very low.

Some embedded systems have to operate in extreme environmental conditions such as very high temperatures and humidity.

1.1 ABOUT THE PROJECT

Imagine how helpful it will be to analyze the gas leakage, intimating the leakage before any threat or losses. This is what home automation is about and there is no end to its application. The process of controlling or operating various equipment, machinery, industrial processes, and other applications using various control systems and also with less or no human intervention is termed as automation. In view of this perspective, thispaper aims at proposing a method for automatic gas leakage detection, which aims to be much easier to be widely deployed and more powerful in ability than before.

In this paper we are presenting a system called IOT based gas leakage monitoring system or detection system. Using this, we can monitor the gas leakage in the cloud server. The system uses a gas sensor which can detect gases such aspropone, iso-butane and lpg. A buzzer goes off whenever any gas leakage is detected by the system. The gas sensor detects if any leakage occurs, and we get an alert message to our mobile handset. The message will also intimate the user about the leakage position i.e., the latitude and longitude, so that we can easily track the location of leakage point.This makes it easier for us to know the source of the leakage and control or stop it.

1.2 OBJECTIVE OF PROJECT

This project profile envisages the production of automatic buzzer if there isany gas leakage by setting up a gas sensor. It is used in industrial, biomedical and home etc., This mechanism consists of Arduino microcontroller which is connected gps, gsm modems, gas sensor, ThingSpeak server. The whole circuit is provided with power supply. When gas is sensed by sensor message and location is sent to registered mobile and the values are stored in ThingSpeak server.

1.3 LIMITATIONS OF THE PROJECT

First, due to the nature of the system, there is no automatic control of gas leakage, but we can locate the position of leakage. Another limitation is that the system should be placed at fixed point and a sim card with message or SMS balance is also a must in this system as the messages are sent to the phone using this sim card. Making sure that sensors and other equipment is secured within the distance limit is vital to obtaining proper signal strength and connectivity that will make the system effective.

2. LITERATURE SURVEY

Survey from (Falohun A.S., Oke A.O., and Abolaji B.M)

In this paper they proposed their dangerous gas detection using an integrated circuit and MQ-9. In this basically, they used an embedded design which includes typical input and output devices include switches, relays, solenoids, LEDs, small or custom LCD displays, radio frequency devices, and sensors for data such as temperature, humidity, light level etc.. Generally, heat detection will be used in all areas that are not considered high value. The principle of operation which is proposed in this paper is the gas detector alarm system is designed with the intention to ensure that the event of gas is intelligently detected, promptly notified and interactively managed. It is built around a timer to accept input from the gas sensor, MQ-9, and activate a buzzer and set of led that alerts in the event of gas. The sensor used is the MQ9 and from the datasheet, it specializes in gas detection equipment for carbon monoxide and CH4, LPG family and any other relevant industry or car assemblage.

Survey from (Anandhakrishnan S, Deepesh Nair, Rakesh K, Sampath K,Gayathri S Nair)

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 gasleakage 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. Sometimes if the burner is left on bymistake, the consumer could be alerted about the problem. If the burner is on and there is no vessel on top of it, an alarm goes off.

Survey from (Prof.M.Amsaveni, A.Anurupa, R.S.Anu Preetha, C.Malarvizhi, M.Gunasekaran)

Meenakshi Vidya proposed the leakage detection and real time gas monitoring system. In this system, the gas leakage is detected and controlled by means of exhaust fan. The level of LPG in cylinder is also continuously monitored. K.Padmapriya proposed the design of wireless LPG monitoring system. In this project, the user is alerted about the gas leakage through SMS and the power supply is turned off. Selvapriya proposed the system in which the leakage is detected by the gas sensor and produce the results in the audio and visual forms. It provides a design approach on software as well as hardware. L.K.Hema proposed the smart sensor technology. In this flexible reliable smart gas detection system is developed. In this, the leakage is detected and controlled by using exhaust fan. B. D. Jolhe proposed the system in which two sensors are used for detecting the gas leakage and for monitoring the level of gas in the cylinder respectively. Ashish Shrivastava proposed the system in which two types of gases namely LPG and CNG are detected for home safety as well for vehicles. R.Padmapriya proposed the system which ARM7 processor and simulates using keil software to alert the user by sending SMS. V.Ramya proposed the system that uses two different sensors for detecting the leakage and requires resetting manually after every situation. A.Mahalingam proposed the system to meet UK occupational health and safety standards and also it alerts the user by SMS. M.B.Frishproposed the system that uses trace sensing technology and also detects the leakage.

3. SYSTEM ANALYSIS

System analysis will be performed to determine if it feasible to design information based on policies and plans of the organization and on user requirements and to eliminate the weakness of the present system.

- The new system should be cost effective.
- To augment management, improve services.
- To enhance user/system interface.
- To improve information quality and usability.
- To upgrade system reliability, availability, flexibility and time saving.
- To provide information security.

3.1 EXISTING SYSTEM

In the present day, they are using gas detectors which detect the gas leakage and intimates the signal to the control room. After getting the signal, they will send the workers to the specific leak points and they will overcome the problem. The main disadvantage in this system is, it takes more time for this process that is, around 15 to 20 minutes.

3.2 PROPOSED SYSTEM

In this paper we are proposing a new system called IoT based gas leakage monitoring system or detection system. In this system, we are monitoring the gas leakage in the server. A buzzer goes off whenever any gas leakage is detected by the system. The gas sensor detects if any leakage occurs, and we get an alert message to our mobile handset. The message will also intimate the user about the leakage position i.e., the latitude and longitude, so that we can easily track the location of leakage point.

3.2.1 IMPACT ON ENVIRONMENT

Gas leakage causes air pollution. Exposure to a gas leak in your house or apartment may cause deadly symptoms including sickness, weakness, nausea, suffocation, and headaches.

3.2.2 COST

The project would cost approximately Rs 2000. The user also needs to recharge the SIM card placed in the GSM module regularly in order to receive message alerts and real time data to be uploaded to the thingspeak server.

3.2.3 TYPE

This is an IOT based project. This project can be placed anywhere in order to monitor the gas level, temperature and humidity of the surroundings and get alerts and also location data if the gas level is above set threshold.

GLDSAS is a mobile application developed for the end user to easily access, share data and useful information. It also allows user to quickly contact the concerned authority during emergency.

3.2.4 STANDARDS

Agile Method

A software development method. Most agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames that typically last from one to four weeks. Every iteration involves a cross-functional team working in all steps: planning, requirements/analysis, design, coding and testing. At the end of the iteration, a working product is demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly. Iteration might not add enough functionality to be useable on its own, but the goal is to have an available release at the end of each iteration. Multiple iterations might be required to release a product or new features.

3.3 SCOPE OF THE PROJECT

Gas leakage can cause major damage to life and property, therefore itshould be used in safe handling manner and additional care has to be taken in order to prevent any leakage. The project is used to detect and monitor gas leakage in any industry or household. It helps to detect any gas leakage using a gas sensor. If there is aleakage, then the system alerts the surroundings using a buzzer. Along with that, it alsosends a message to the concerned person or authority along with the location of the gas leakage. In order to monitor the leakage, it also sends these values to the server. This helps to monitor the gas leakage in the industry and as it can also alert the concerned person and surroundings, it is efficient and effective.

3.4 SYSTEM CONFIGURATION

It specifies the hardware and software requirements that are required in order to run the application properly.

Hardware Requirements:

Power Supply : +5V, 750mA Regulated Power Supply

Micro Controller : Arduino ATMEGA 328

Gas Sensor

GPS

GSM

DHT 11 Sensor

Software Requirements:

Programming language : Embedded C

IDE : Arduino

Frontend :MIT App Inventor

Backend :Google Firebase

3.5 HARDWARE REQUIREMENTS

a) Power Supply: The input to the circuit is applied from the regulated power supply. The a.c. input i.e., 230V from the mains supply is step down by the transformer to 12V and is fed to a rectifier.

The output obtained from the rectifier is a pulsating d.c voltage. So in order to get a pure d.c voltage, the output voltage from the rectifier is fedto a filter to remove any a.c components present even after rectification. Now, this voltage is given to a voltage regulator to obtain a pure constant dc voltage.

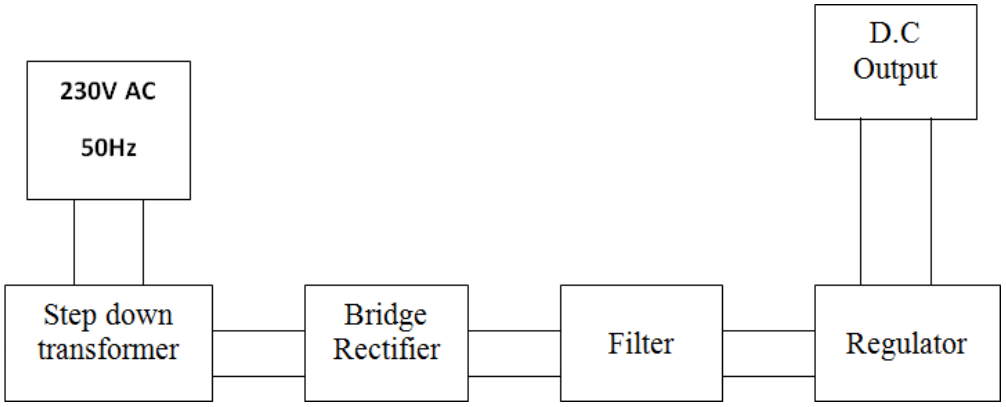

Fig 3.5.1: Block Diagram of Power Supply

b) Transformer: Usually, DC voltages are required to operate various electronic equipment and these voltages are 5V, 9V or 12V. But these voltages cannot be obtained directly. Thus the a.c input available at the mains supply i.e., 230V is to be brought down to the required voltage level. This is done by a transformer. Thus, a stepdown transformer is employed to decrease the voltage to a required level.

A transformer is an electrical device that transfers energy from one circuit to another by magnetic coupling with no moving parts. A transformer comprises two or more coupled windings, or a single tapped winding and, in most cases, a magnetic core to concentrate magnetic flux. A changing current in one winding creates a time- varying magnetic flux in the core, which induces a voltage in the other windings.

Fig 3.5.2: Step down Transformer

c) Rectifier: The output from the transformer is fed to the rectifier. It converts A.C. into pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification.

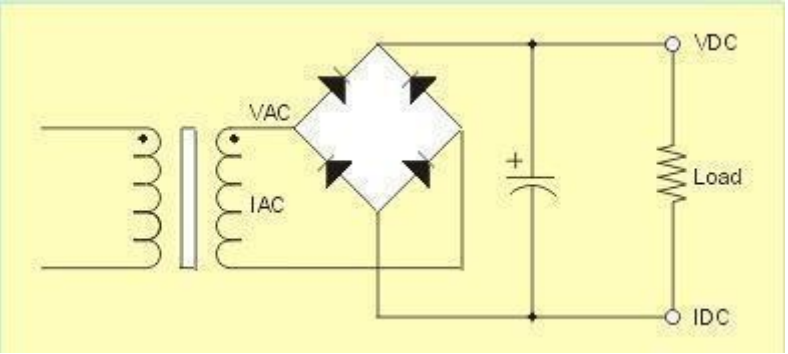

Fig 3.5.3: Rectifier

The Bridge rectifier is a circuit, which converts an ac voltage to dc voltage using both half cycles of the input ac voltage. The Bridge rectifier circuit is shown in the figure. The circuit has four diodes connected to form a bridge. The ac input voltage is applied to the diagonally opposite ends of the bridge. The load resistance is connected between the other two ends of the bridge.

For the positive half cycle of the input ac voltage, diodes D1 and D3 conduct, whereas diodes D2 and D4 remain in the OFF state. The conducting diodes will be in series with the load resistance R_L and hence the load current flows through R_L.

For the negative half cycle of the input ac voltage, diodes D2 and D4 conduct whereas, D1 and D3 remain OFF. The conducting diodes D2 and D4 will be in series with the load resistance R_L and hence the current flows through R_L in the same direction as in the previous half cycle. Thus a bi-directional wave is converted into a unidirectional wave.

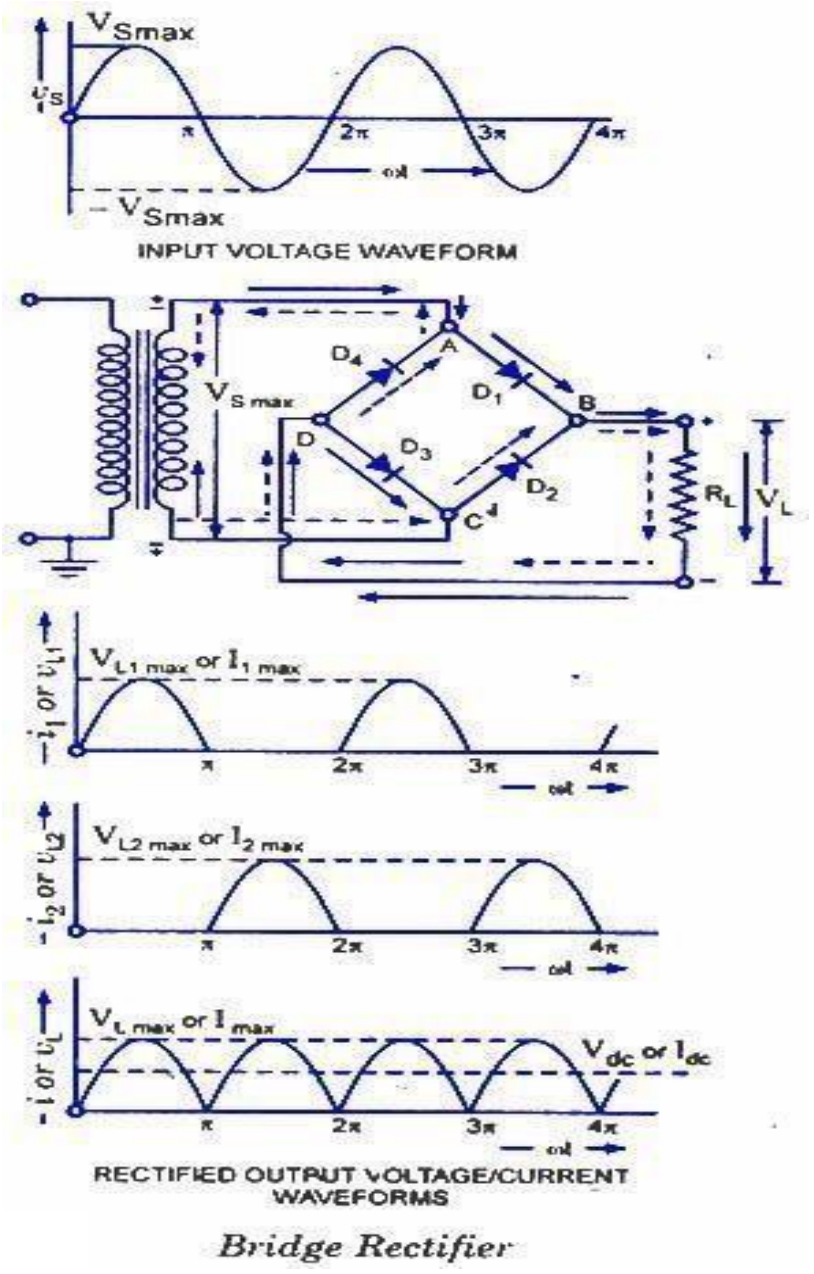

Fig 3.5.4: Bridge Rectifier

d) Filter: Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the D.C. Output received from this filter is constant until the mains voltage and load is maintained constant. However, if either of the two is varied, D.C. voltage received at this point changes. Therefore a regulator is applied at the output stage.

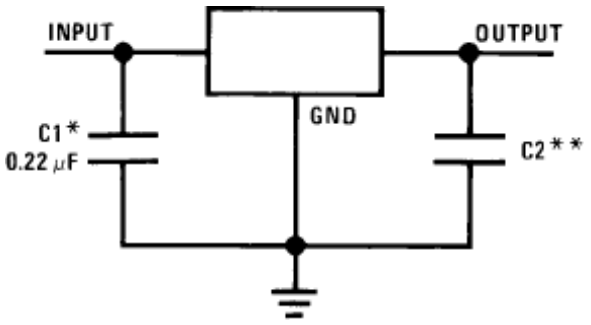

Fig 3.5.5: Filter

e) Voltage regulator: As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. In this project, power supply of 5V and 12V are required. In order to obtain these voltage levels, 7805 and 7812 voltage regulators are to be used. The first number 78 represents positive supply and the numbers 05, 12 represent the required output voltage levels. The L78xx series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3, D2PAK and DPAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1 A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external \components to obtain adjustable voltage and currents.

Fig 3.5.6: Voltage Regulator

f) Arduino Microcontroller: The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

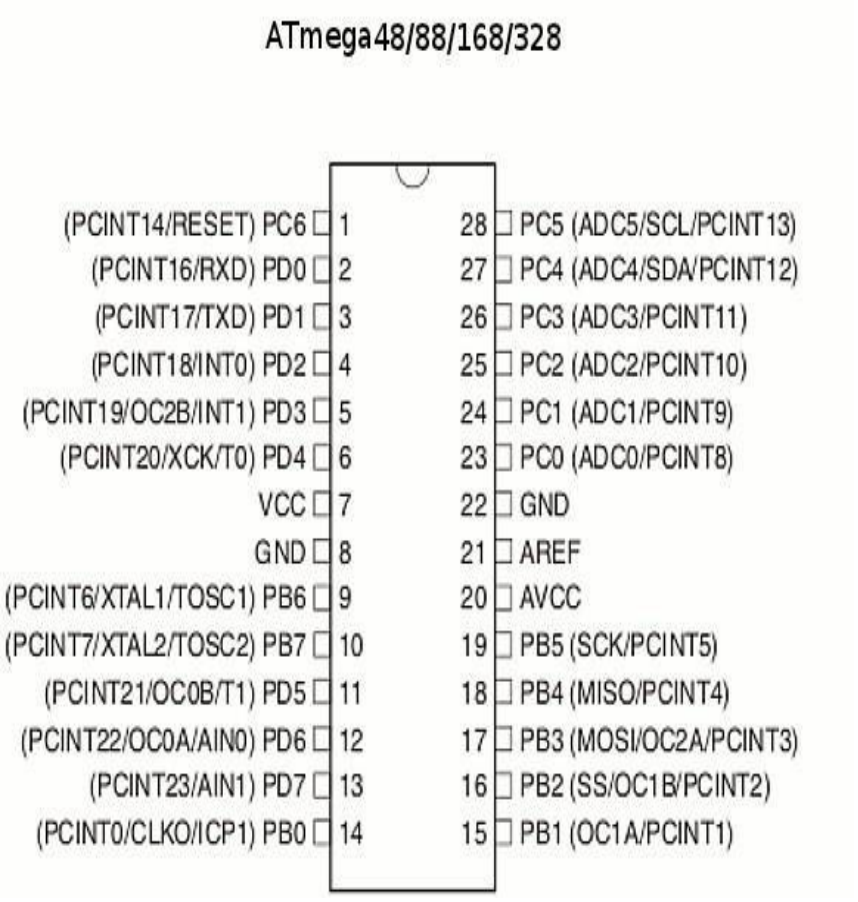

Fig 3.5.7: Arduino Microcontroller pin diagram

f) Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, 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. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor whichallows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serialchip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins. ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 Nano farad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the boot loader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the boot loader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET- EN". You may also be able to disable the auto-reset by connecting a 110-ohm resistor from 5V to the reset line; see this forum thread for details.

USB Over Current Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and over current. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

g) **MQ 2:** The MQ series of gas sensors use a small heater inside with an electro- chemical sensor. They are sensitive for a range of gasses and are used indoors at room temperature.It can be used in gas leakage detecting equipment in consumer and industry applications.MQ2 Gas sensor works on 5V DC and draws around 800mW.

It can detect LPG, Smoke, Alcohol, Propane, Hydrogen, Methane and Carbon Monoxide concentrations anywhere from 200 to 10000ppm.

Fig 3.5.8 MQ 2 Gas sensor

Features

- High sensitivity to LPG, Iso-butane, propane.
- Small sensitivity to alcohol, smoke.
- Fast response
- Stable and long life
- Simple drive circuit

Specifications

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	ACOR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω ±5%	Room Tem
P_H	Heating consumption	less than 800mw	

B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-20℃-50℃	
Tas	Storage Tem	-20℃-70℃	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

Symbol	Parameter name	Technical parameter	Remarks
Rs	Sensing Resistance	3KΩ-30KΩ (1000ppm iso-butane)	Detecting concentration scope: 200ppm-5000ppm LPG and propane 300ppm-5000ppm butane 5000ppm-20000ppm methane 300ppm-5000ppm H_2 100ppm-2000ppm Alcohol
α (3000/1000) isobutane	Concentration Slope rate	≤0.6	
Standard Detecting Condition	Temp: 20℃±2℃ Humidity: 65%±5%	Vc:5V±0.1 Vh: 5V±0.1	
Preheat time	Over 24 hour		

Table 1: MQ 2 Specifications Table

h) Buzzer

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or electronic. Typical uses of buzzers and beepers include alarms, timers and confirmation of user input such as a mouse click or keystroke.

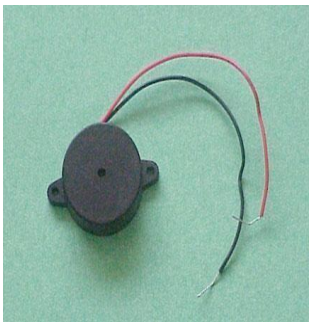

Fig 3.5.9: Buzzer

Features

- The PB series are high-performance buzzers with a unimorph piezoelectric ceramic element and an integral self-excitation oscillator circuit.
- They exhibit extremely low power consumption in comparison to electromagnetic units.
- They are constructed without switching contacts to ensure long life and no electrical noise.
- Compact, yet produces high acoustic output with minimal voltage.

3.6 SOFTWARE TOOLS

Arduino and Arduino Software and Drivers Installation

This describes the installation of the Arduino IDE Development software and drivers for the Windows Operating System. The images and description is based on installation under Windows 7, but the process should be similar for Vista and Windows 8 or Above. First we need to get the latest version of the Arduino software this can be downloaded from the Arduino website.

STEP 1: Next, plug in your Arduino board to your computer with a USB cable and wait while Windows detects the new device. Windows will fail to detect the device as it doesn't know where the drivers are stored. You will get an error similar to the one right. Select the Install from a list or specific location (Advanced) option and click Next.

Fig 3.6.1: Installation of software

STEP 2: Now choose location that the Arduino drivers are stored in. This will be in a subfolder called **drivers** in your Arduino directory.

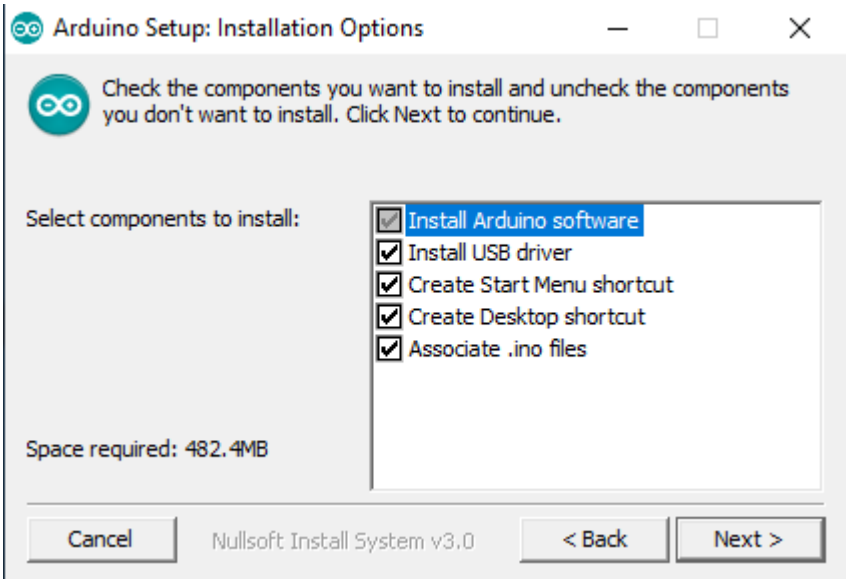

Fig 3.6.2: Selection of Drivers

STEP3: After selecting Next you may get a message like the one shown right. Select Continue Anyway.

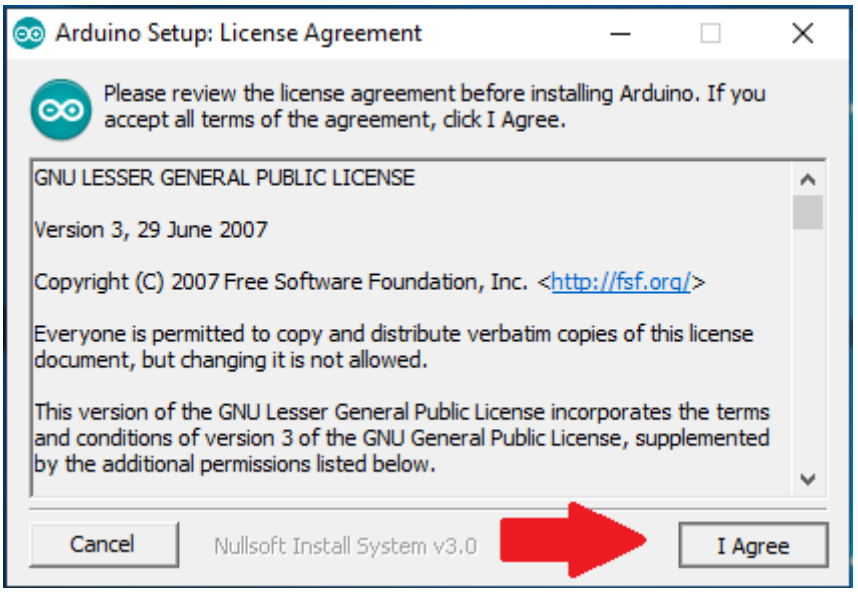

Fig 3.6.3: Installation Verification

STEP 4: Windows should now have found the Arduino drivers. Click **Close** tocomplete the installation

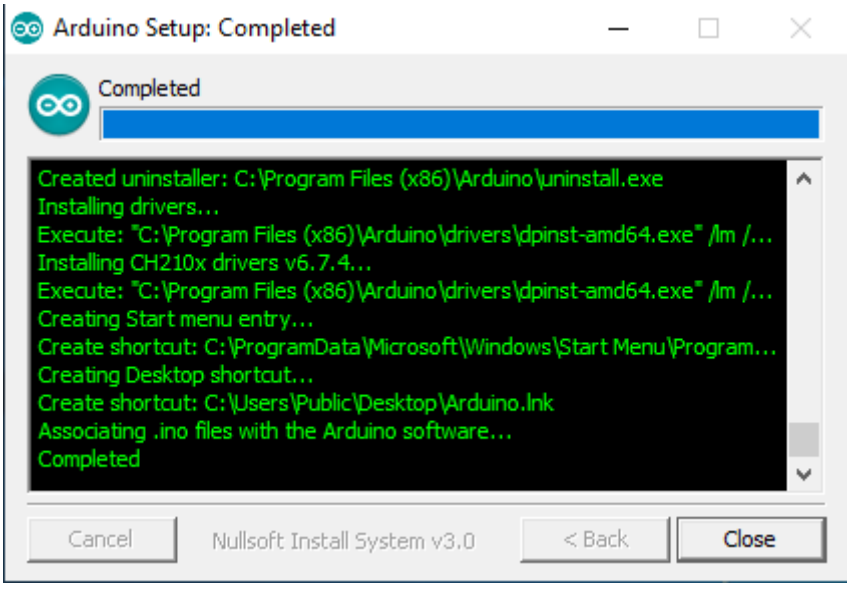

Fig 3.6.4: Software for Arduino

STEP 5: The computer communicates with the Arduino board via a special serial port chip built into the Arduino board. The Arduino IDE software needs to know the serial port number that Windows has just allocated to it. Open the Windows Control Panel and select the System app. Click on the Hardware tab and then on the Device Manager button. Click on the Ports (COM and LPT) option and note what COM port has been allocated to the Arduino UNO.

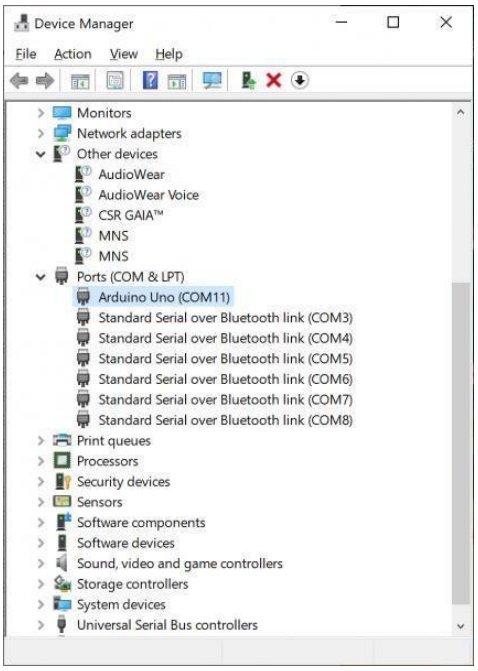

Fig 3.6.5: Arduino Board

STEP 6: Next, run the Arduino IDE application, which will be in c:\program files\arduino-0021 or similar. Click on Tools | Serial Port and select the port number from above

STEP 7: Next click on Tools | Board and select the type of board that you have.

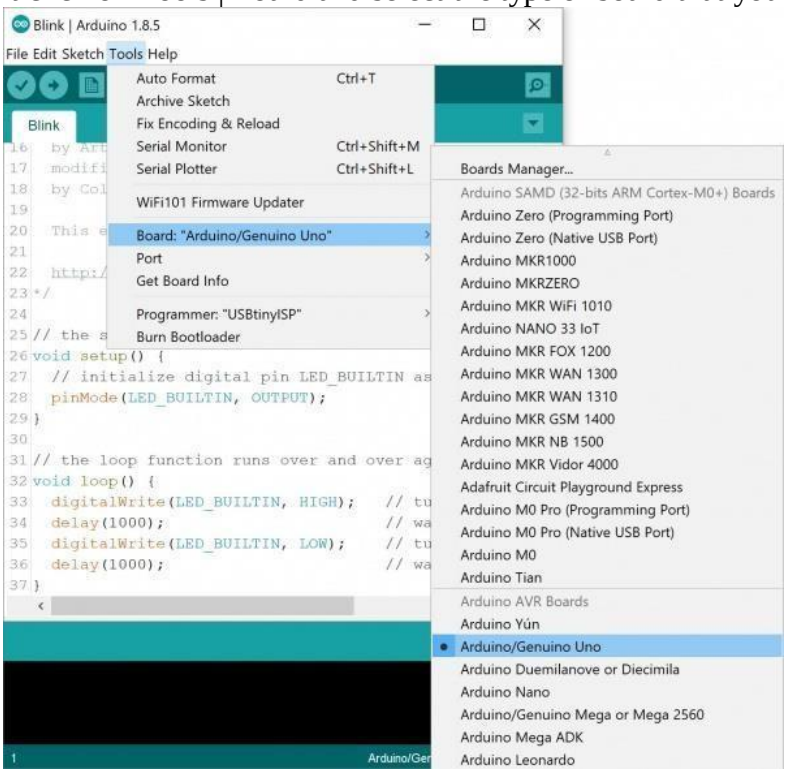

Fig 3.6.6: Selection of Board in Arduino

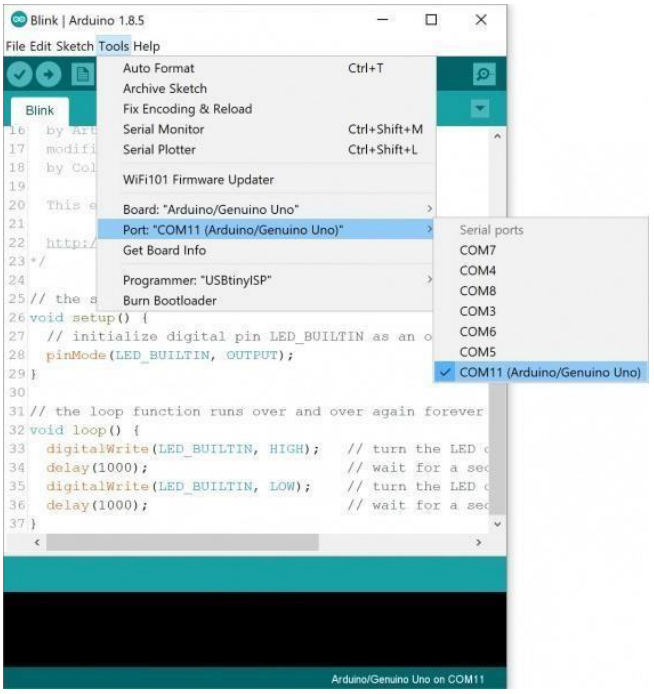

Fig 3.6.7: Selection of Serial Port in Arduino

STEP 8: Now try opening the Simple program from the example directory within the Arduino IDE, Verify/Compile it and upload it to your board. You should see the TX and RX led's on the board flash showing you that it is working. Finally, the built in LED connected to Pin 13 will flash. That's your first program running.

Create a shortcut to the Arduino IDE and place it on your desktop.

Fig 3.6.8: Running a Sample Program

3.7 MODULES DESCRIPTION

3.7.1 GSM MODULE

GSM/GPRS module is used to establish communication between a computer and a GSM-GPRS system. Global System for Mobile communication (GSM) is an architecture used for mobile communication in most of the countries. Global Packet Radio Service (GPRS) is an extension of GSM that enables higher data transmission rate. GSM/GPRS module consists of a GSM/GPRS modem assembled together with power supply circuit and communication interfaces (like RS-232, USB, etc.) for computer. The MODEM is the soul of such modules.

Fig 3.7.1.a: GSM Module

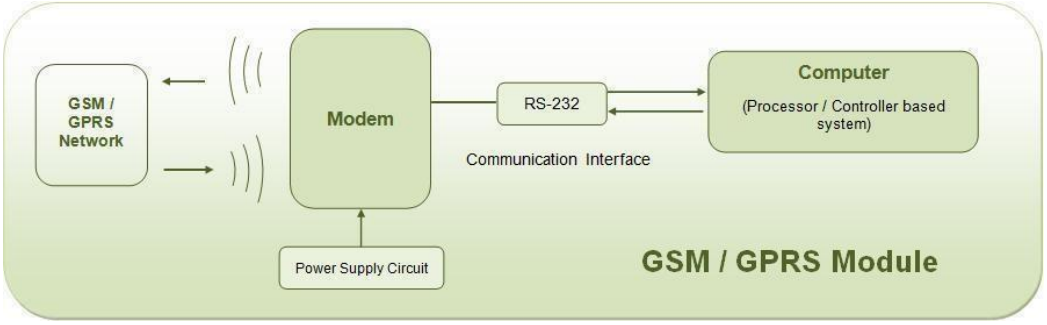

Fig 3.7.1.b: A GSM/GPRS Module Architecture

Wireless Modem

Wireless Modems are the modem devices that generate, transmit or decode data from a cellular network, for establishing communication between the cellular network and the computer. These are manufactured for specific cellular network (GSM/UMTS/CDMA) or specific cellular data standard (GSM/UMTS/GPRS/EDGE/HSDPA) or technology (GPS/SIM). Wireless MODEMs

like other MODEM devices use serial communication to interface with and need Hayes compatible AT commands for communication with the computer (any microprocessor or microcontroller system).

GSM/GPRS Modem

GSM/GPRS Modem is a class of wireless Modem devices that are designed for communication of a computer with the GSM and GPRS network. It requires aSIM (Subscriber Identity Module) card just like mobile phones to activate communication with the network. Also they have IMEI (International Mobile Equipment Identity) number similar to mobile phones for their identification. A GSM/GPRS MODEM can perform the following operations:

1. Receive, send or delete SMS messages in a SIM.
2. Read, add, search phonebook entries of the SIM.
3. Make, Receive, or reject a voice call.

The Modem needs AT commands, for interacting with processor or controller, which are communicated through serial communication. These commands are sent by the controller/processor. The MODEM sends back a result after it receives a command. Different AT commands supported by the Modem can be sent by the processor/controller/computer to interact with the GSM and GPRS cellular network.

GSM/GPRS Module

A GSM/GPRS module assembles a GSM/GPRS modem with standard communication interfaces like RS-232 (Serial Port), USB etc., so that it can be easily interfaced with a computer or a microprocessor / microcontroller based system. The power supply circuit is also built in the module that can be activated by using a suitable adaptor.

Mobile Station (Cell phones and SIM)

A mobile phone and Subscriber Identity Module (SIM) together form a mobile station. It is the user equipment that communicates with the mobile network. A

mobile phone comprises of Mobile Termination, Terminal Equipment and Terminal Adapter.

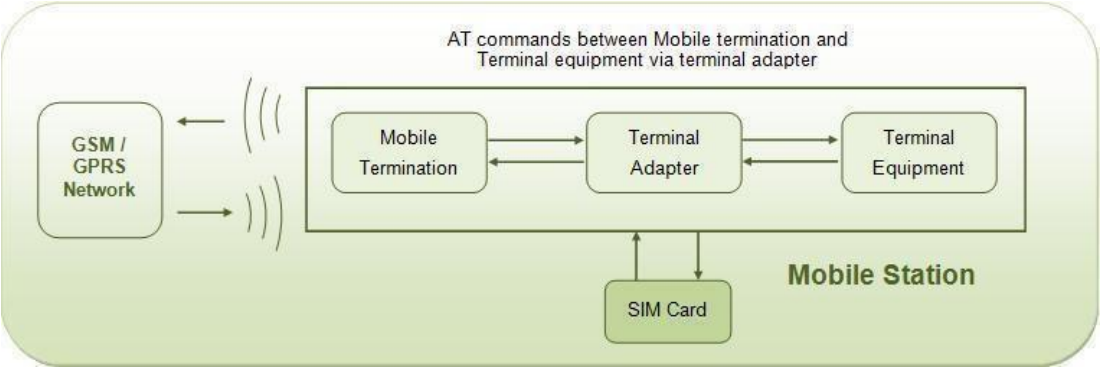

Fig 3.7.1.c*:* Block Diagram Showing Different Functions of SIM In Mobile Station

Mobile Termination is interfaced with the GSM mobile network and is controlled by a baseband processor. It handles access to SIM, speech encoding and decoding, signaling and other network related tasks. The Terminal Equipment is an application processor that deals with handling operations related to keypad, screen, phone memory and other hardware and software services embedded into the handset. The Terminal Adapter establishes communication between the Terminal Equipment and the Mobile Termination using AT commands. The communication with the network in a GSM/GPRS mobile is carried out by the baseband processor.

Applications of GSM/GPRS module

The GSM/GPRS module demonstrates the use of AT commands. They can feature all the functionalities of a mobile phone through computer like making and receiving calls, SMS, MMS etc. These are mainly employed for computer based SMS and MMS services.

AT Commands

AT commands are used to control MODEMs. AT is the abbreviation for Attention. These commands come from Hayes commands that were used by theHayes smart modems. The Hayes commands started with AT to indicate the attention from the MODEM. The dial up and wireless MODEMs (devices that involve machine to machine communication) need AT commands to interact with a computer.

These include the Hayes command set as a subset, along with other extended AT commands.

AT commands with a GSM/GPRS MODEM or mobile phone can be used toaccess following information and services:

1. Information and configuration pertaining to mobile device or MODEM and SIMcard.
2. SMS services.
3. MMS services.
4. Fax services.
5. Data and Voice link over mobile network.

The Hayes subset commands are called the basic commands and the commands specific to a GSM network are called extended AT commands.

Command, Information response and Result Codes

The AT commands are sent by the computer to the MODEM/ mobile phone. The MODEM sends back an Information Response i.e. the information requested by orpertaining to the action initiated by the AT command. This is followed by a Result Code. The result code tells about the successful execution of that command.

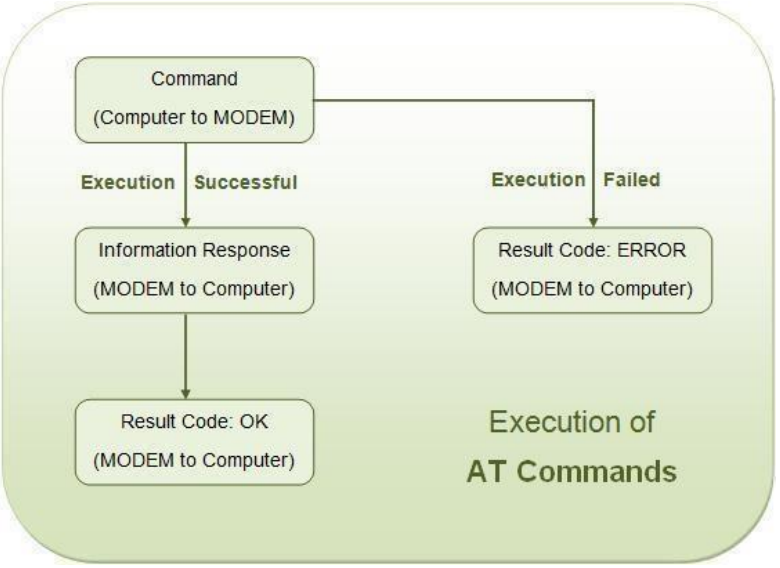

Fig 3.7.1.d: Data Flow Chart Showing Execution Of AT Commands

There are also unsolicited Result Codes that are returned automatically by the MODEM to notify the occurrence of an event. For example, the reception of a SMS will force MODEM to return an unsolicited result code.

AT command's syntax

1) Case Sensitivity

The AT commands are generally used in uppercase letters. However, some MODEMs and mobile phones allow both uppercase and small case letters.

2) Single Command

The AT commands include a **prefix AT** which indicates the beginning of the command to MODEM; and a **carriage return** which indicates the end of the command.

However, string 'AT' itself is not the part of the command. For example, in ATD, D is the command name not ATD.

The extended AT commands have a '+' in the command name.

For example: **AT+CGMI<Carriage return>**

Command Line

Multiple AT commands can be sent to MODEM in a single command line. The commands in a line are separated by a semi-colon (;).

String in Command Line

Strings in a command line are enclosed in double quotes.

For example: **AT+CGML=" ALL" <Carriage return>**

Information Response and Result Code

The Information Response and Result Codes, returned by the MODEM, have a carriage return and line feed in the beginning as well as at the end.

Sequence of Execution

In the command line, the command appearing first is executed first. The execution then follows for second appeared command and so on. The execution of commands in a command line takes place in sequential manner. If an error occurs in the execution of a command, an error result code is returned by the MODEM and the execution of the command line is terminated irrespective of presence of other commands next in the command line.

Interfacing of the GSM/GPRS module with the serial port of the computer involves following steps:

- Connect RS-232 port of GSM module with the serial port of the computer. Insert a SIM card in the module.
- Open HyperTerminal from Start -> All Programs -> Accessories -> Communications -> HyperTerminal.
- Enter the name for the connection and press OK.
- Now select the communication port (COM) at which GSM module is connected.
- Create a new connection set on HyperTerminal. Set parameters, like baud rate as 9600, handshaking mode as none, parity bit as none, stop bit as 1 and data bit as 8.

3.7.2 GPS MODULE

NEO-6M GY-GPS6MV2 GPS module features the u-blox NEO-6M GPS module with antenna and built-in EEPROM. This is compatible with various flight controller boards designed to work with a GPS module.

The core of the module [1], the "GPS unit", is from u-blox, a company specialized in wireless communications and industrial positioning. On board voltage regulator (KB33 MIC 5205; [4]). The module has a small battery [3], which is, apparently, used for the secure storage of configuration data on a EEPROM [5].There is a small green LED [7] which blinks when the location is fixed. The GPS antenna

[8] should be kept with the ceramic part upwards, looking to the skies! There are four main pins [2] : VCC, TX, RX, GND.

Fig 3.7.2: GPS Module

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 6 positioning engine. These flexible and cost effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints. The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppresses jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challenging environments.

Specifications

- GPS Module NEO-6M
- Model: GY-GPS6MV2
- Input Supply Voltage Range: 3.3V-6V, on board voltage regulator maintains 3.3V
- I/O Maximum Logic Level: 3.6V
- <1 second to first fix (TTFF) for hot starts
- 27 seconds to first fix (TTFF) for cold starts

- On board LED will blink after module acquires a position fix and will continue blinking as long as the module has a fix
- 50 Channel NEMA GPS receiver

Power Modes

Two continuous operating modes:

- **Maximum Performance Mode -** continuously uses the acquisition engine, resulting in the best possible time to first fix (TTFF) Eco Mode - optimizes the use of the acquisition engine to minimize current consumption.
- **One intermittent operating Mode -** Power Save Mode - draws only 11mA - Utilizes cyclic tracking, with configurable update periods, which reduces the average power consumption significantly. This GPS module does not have onboard compass. You need to get one I2C (Compass) separately if your flight controller does not have one onboard.

Handling GPS Data

Once the GPS receiver has transmitted the NMEA sentences to the Arduino, and they have been properly read and stored in nice and "simple format", then what follows is.

There are several options to convert the "simple format" data into more general format (GPX, KMZ, etc.). For instance:

1. GPS Visualizer is a free online utility "that creates maps and profiles from geographic data". You can input a CSV or tabbed file, a spreadsheet, or drag and drop the data. The appearance of the page is a bit odd but the content is good.
2. GPS Prune is intended to view, edit and convert GPS data. It allows to load text files as well as NMEA files, among quite a number of other options.
3. GPS Babel seems to be the most known GPS data converter. It reads text files with **RX** (or RXD) - receive pin. Connected to Arduino board TX pin.

TX (or TXD) - transmit pin. Connected to Arduino board RX pin.

VCC - power supply. Can be connected to +5VDC or +3.3VDC pin of Arduino board.

GND - ground. Connected to Arduino board GND pin.

PPS - Pulse per second. This is an output pin on some GPS modules. Generally, when this pin toggles, once a second, you can synchronize your system clock to the GPS clock.

4. SYSTEM DESIGN

4.1 INTRODUCTION

Design is a meaningful engineering representation of something that is to be built. Software design is a process through which the requirements are translated into a representation of the software. Design is the place where the quality is fostered in software engineering. Design is the perfect way to accurately translate a customer's requirement into finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. This chapter discusses about the design part of the project. Here in this document various UML diagrams that are used for the implementation of the project are discussed.

Data Flow Diagram

Data flow diagram will act as a graphical representation of the system in terms of interaction between the system, external entities and process and how data stored in certain location. DFD shows how the information moves through the system and it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons:

- Logical information flow of the system
- Determination of physical system construction requirements
- Simplicity of notation
- Establishment of manual and automated systems requirements.

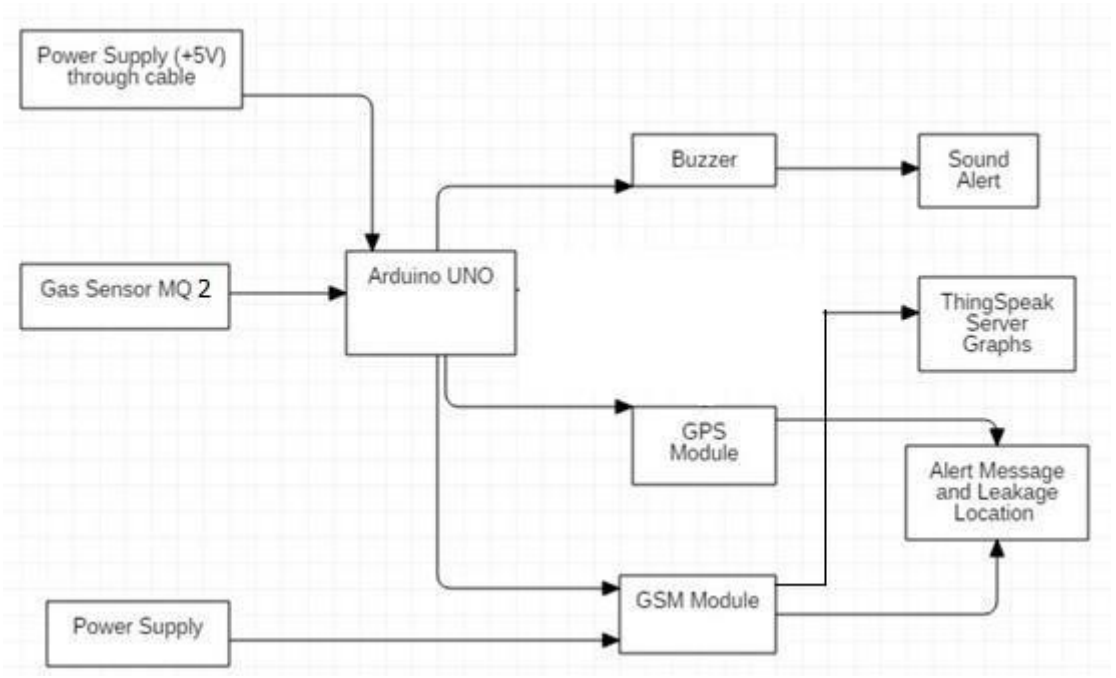

Fig 4.1.1: Data Flow Diagram

In the data flow diagram given above, the power supply is given through Adapter to the GSM module. The Arduino UNO is connected with the Laptop through A to B cable. The MQ 6 gas sensor and the Power supply are the inputs of this system. The power supply to the other modules like Wi-Fi module and GPS module is given from the Arduino micro-controller.

When gas is sensed by the gas sensor, the buzzer starts ringing. The location is taken from the GPS module and through the SIM kept in the GSM module , an alert message along with the location (the latitude and longitude values) is sent to the user. The graphs in the ThingSpeak Server are also updated. The system keeps checking for gas leakage continuously and intimates the user whenever any leakage takes place.

4.2 UML DIAGRAMS

Unified Modelling Language

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspectives. Each view is defined by a set of diagrams which is as follows:

User Model View

- This view represents the system from the user's perspective.
- The analysis representation describes a usage scenario from the end-user's perspective.

Structural Model View

- In this model the data and functionality are arrived from inside the system.
- This model view models the static structure.

Behavioural Model View

It represents the dynamic of behavioural as part of the system, depicting the interactions of collection between various structural elements described the user model and structural model view.

Implementation Model View

In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are

- UML Analysis modelling, this focuses on the user model and structural model views of the system.
- UML Design modelling, this focuses on the behavioural modelling implementation modelling and environmental model views.

4.2.1 Use Case Diagram

Use case diagrams as used to gather the requirements of the system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

A use case represents a particular functionality of a system. So use case diagram is used to describe the relationships among the functionalities and their internal/externalcontrollers. These controllers are also known as actors.

Use Case

Use Case describes the behavior of the system. It is used to structure things ina model. It contains multiple scenarios, each of which describes a sequence of actions that is clear enough for outsiders to understand.

Actor

An actor represents a coherent set of roles that users of system play interesting with the use cases of the system. An actor can represent the role of the human, a device or any other system

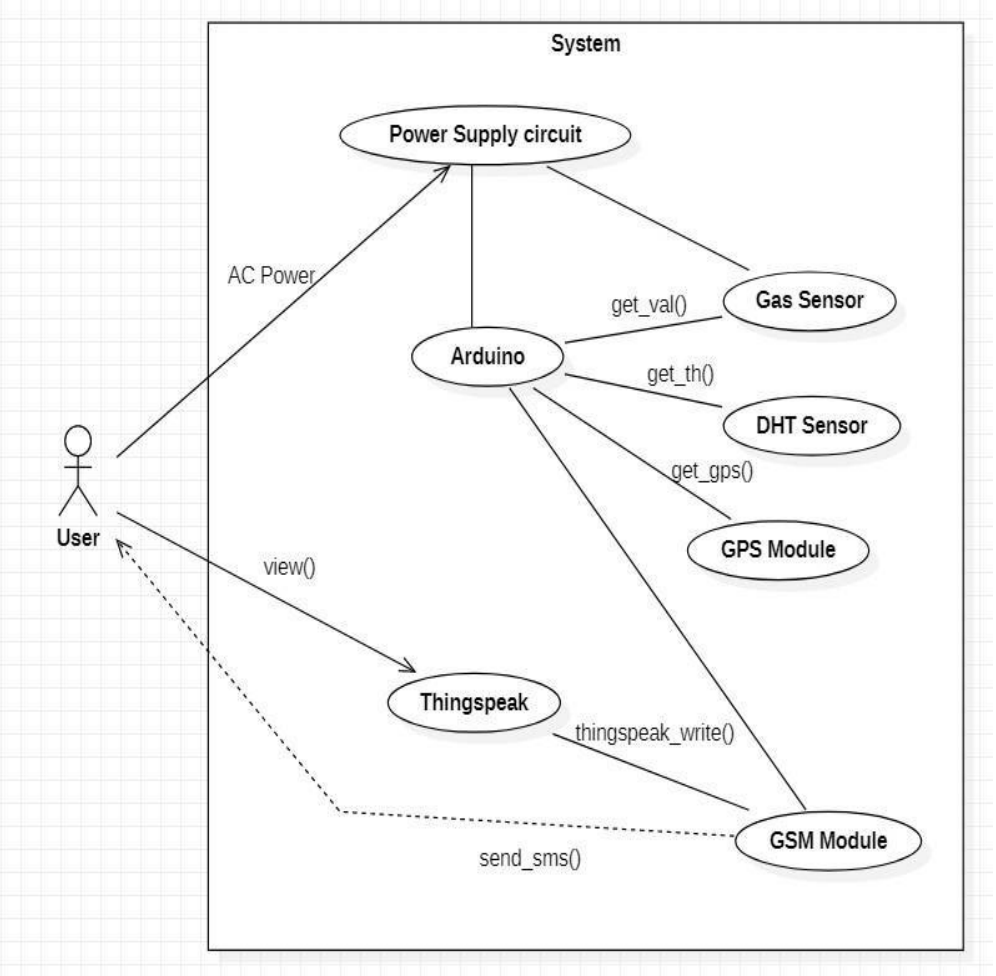

Fig 4.2.1: Use Case Diagram

4.2.2 Class Diagram

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code ofthe software application.

Class diagrams basically represent basically the object oriented view of a system which is static in nature. Active class is used in the class diagram to represent the concurrency of the system. S o it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

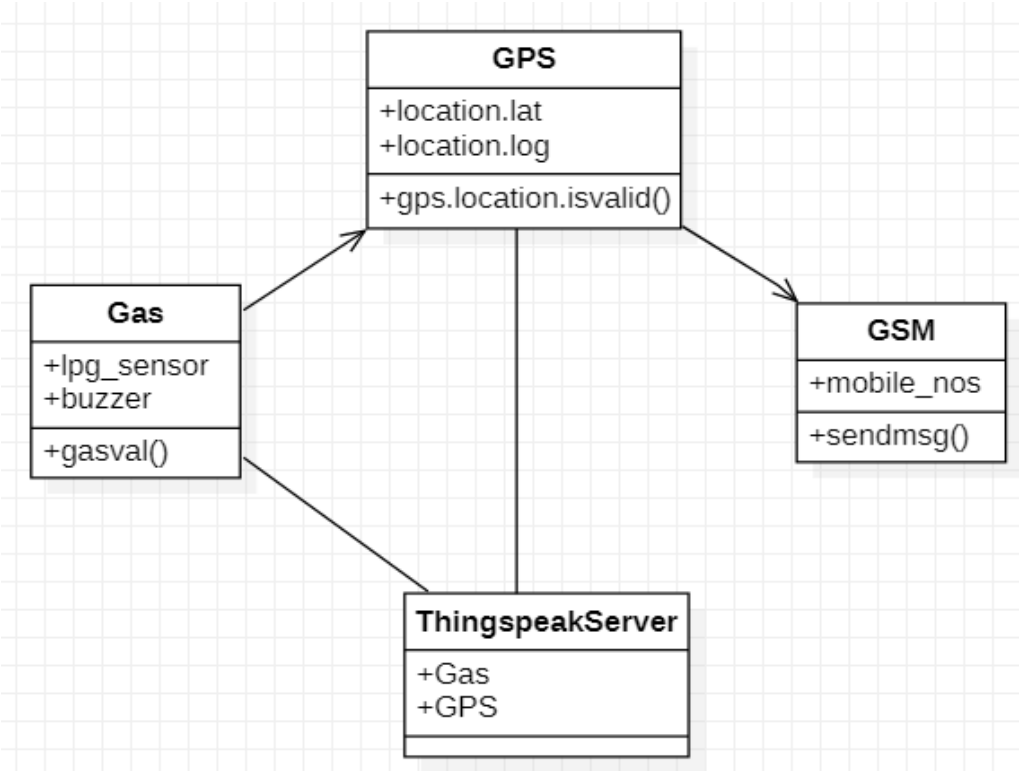

Fig 4.2.2: Class Diagram

4.2.3 Sequence Diagram

A sequence diagram is an interaction diagram. From the name it is clear that the diagram deals with some sequences, which are the sequences of messages flowingfrom one object to another.

Interaction among the components of a system is very important from implementation and execution perspective. So sequence diagram is used to visualize the sequence of calls in the system to perform a specific functionality.

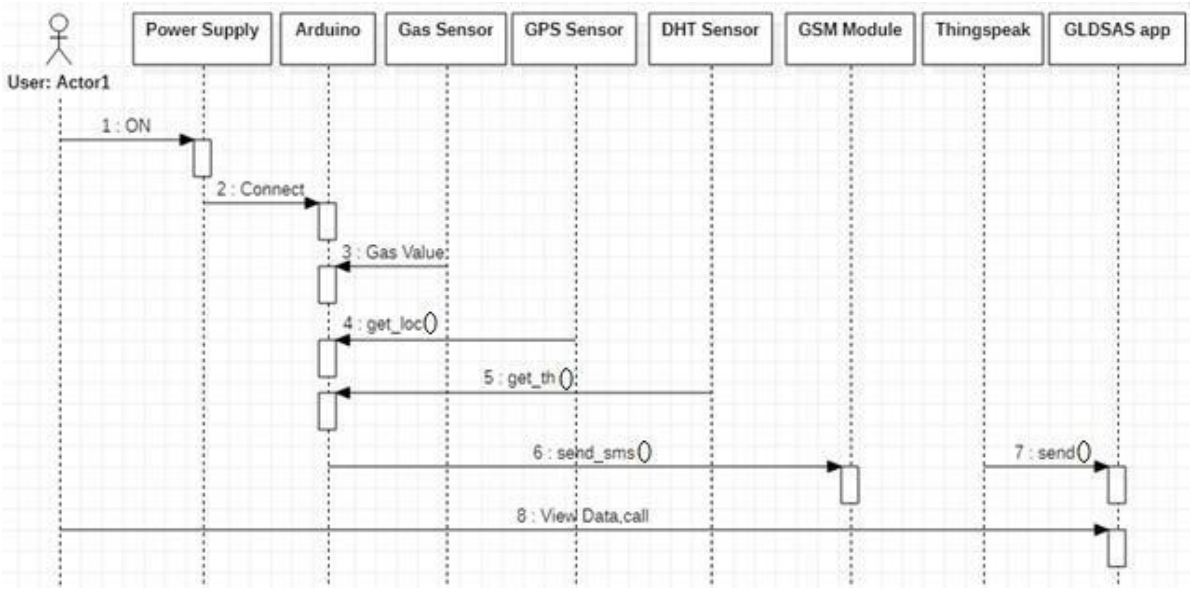

Fig 4.2.3: Sequence Diagram

4.2.4 Activity Diagram

Activity diagrams are graphical representations of workflow of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modelling Language, activity diagram can be used to described the business and operational step-by-step workflows of components in the system. An activity diagram shows the overall flow of control.

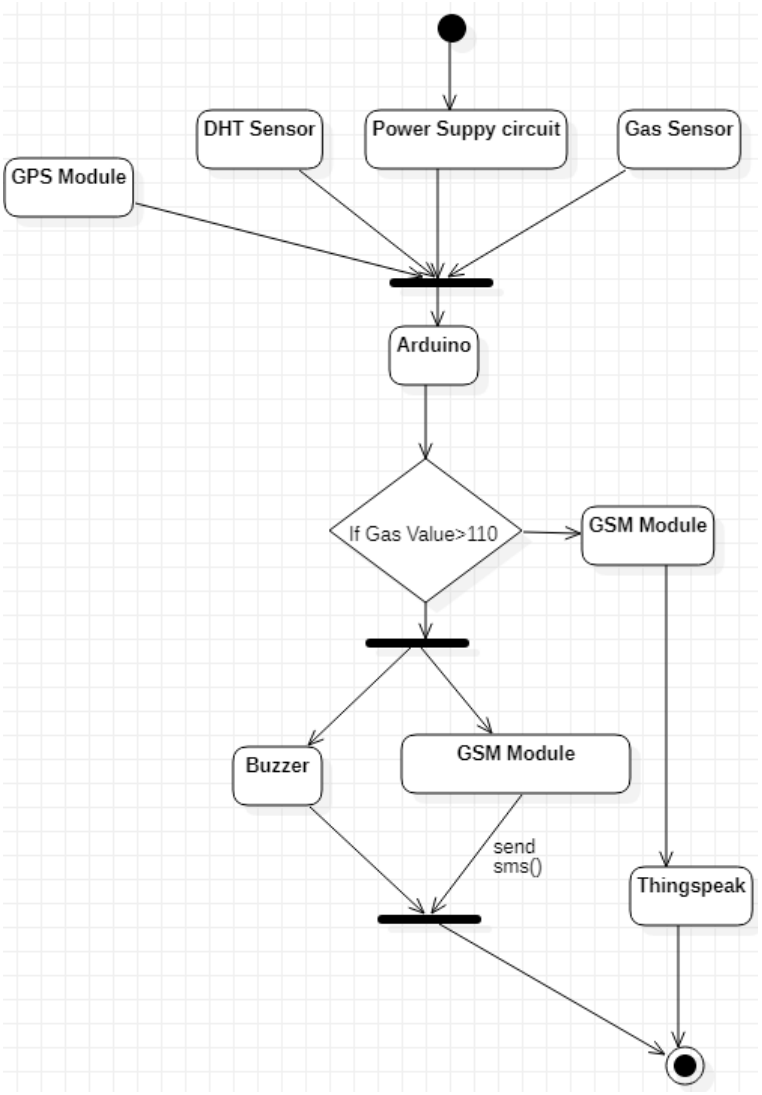

Fig 4.2.4: Activity Diagram

4.2.5 Deployment diagram

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related. Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

An efficient deployment diagram is very important because it controls the following parameters

- Performance
- Scalability
- Maintainability

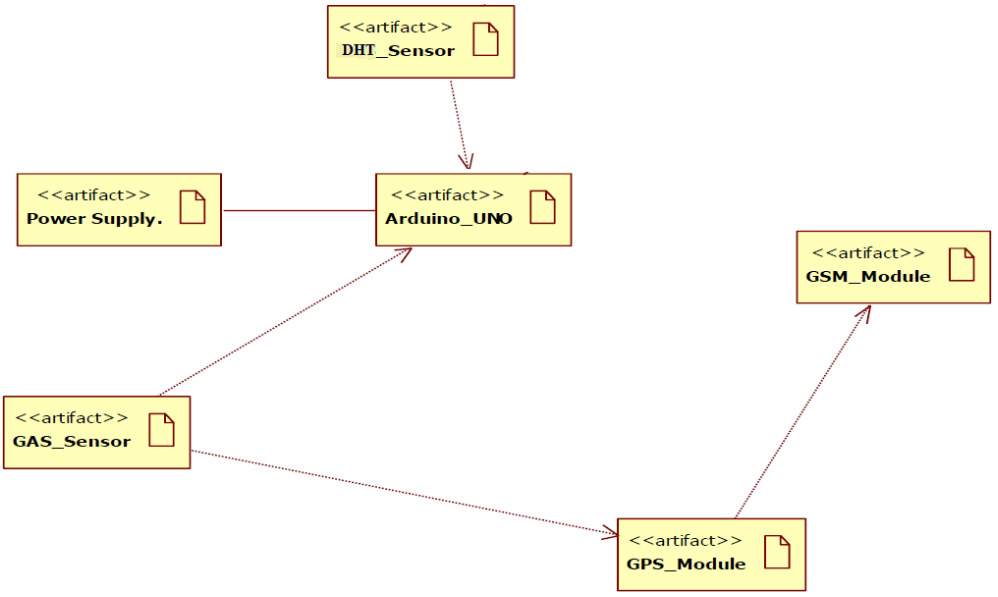

Fig 4.2.5: Deployment diagram

4.2.6 Component Diagram

A Component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as −

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

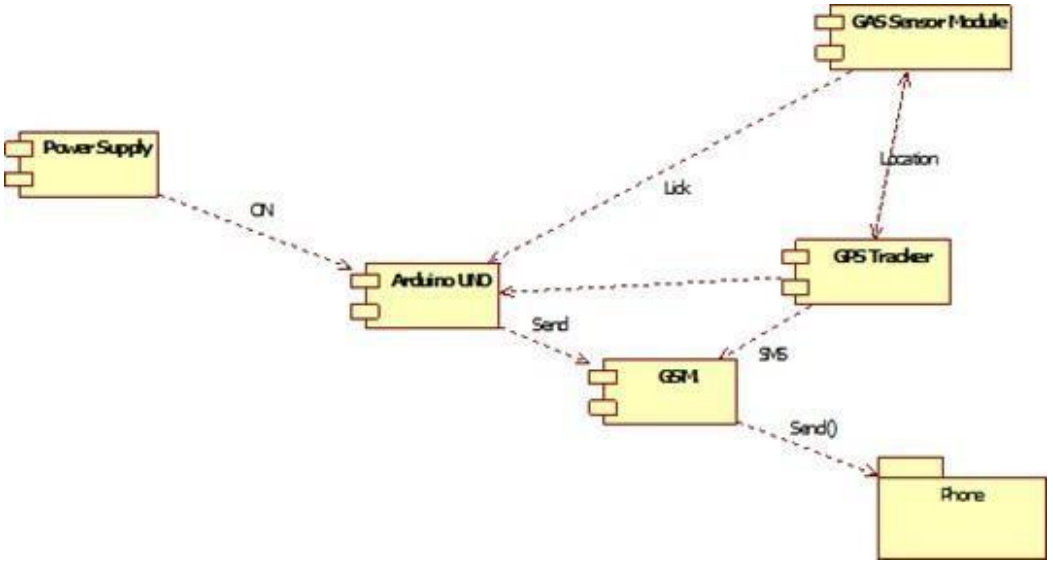

Fig 4.2.6: Component diagram

5. SYSTEM IMPLEMENTATION

System Implementation uses the structure created during architectural design and the results of system analysis to construct system elements that meet the stakeholder requirements and system requirements developed in the early life cycle phases. Implementation is the process that actually yields the lowest-level system elements in the system hierarchy. System elements are made, bought, or reused. Production involves the hardware fabrication processes of forming, removing, joining, and finishing, the software realization processes of coding and testing, or the operational procedures development processes for operators' roles. During the implementation process, engineers apply the design properties and/or requirements allocated to a system element to design and produce a detailed description. They then fabricate, code, or build each individual element using specified materials, processes, physical or logical arrangements, standards, technologies, and/or information flows outlined in detailed descriptions. A system element will be verified against the detailed description of properties and validated against its requirements.

5.1 MODULES DESIGN AND ORGANIZATION

In this IOT gas leakage monitoring system, to run all the modules, we need +5v DC power. To get the required voltage we need power supply circuit, so we converting 230v AC mains to +5V DC. Arduino controller is connected to all modules; each module is executed with respective commands given by Arduino. Insert the SIM in the GSM module to send alert SMS.

We connect the Thingspeak module to the mobile data from GSM Module, so that we can send the values to the server. Here we are using ThingSpeak server to monitor the gas leakage values sent by the controller. As soon as we give the power supply, all the modules get activated by the controller.

GPS will track the Latitude and Longitude values of that particular position. Gas sensor will check the any leakage in the pipe if no leakage, sensor value is given to the controller. MQ 2 is the sensor to detect the LPG, Propane and butane gases.

If any gas leakage is present immediately controller will activate the buzzer, send an alert SMS to the user to notify the leakage with latitude and longitude, so that we can track the location.

```
void sendMessage(int i)
{
  String msg;//="Alert! \n ";
  msg+="https://www.google.co.in/maps/place/";
  msg+=latitude+","+longitude;
  switch(i)
  {
    case 1:msg+="\nTemperature HIGH";break;
    case 2:msg+="\nGas level HIGH";break;
  }
  Serial.println("AT+CMGF=1");//set the GSM Module in Text mode
  delay(1000);
  Serial.println("AT+CMGS=\"+917XXXXXXXXX\
  "\r");delay(1000);
  Serial.println(msg);//the SMS text you want to send
  delay(100);
  Serial.println((char)26);
  delay(100);
}
```

Meanwhile the values1 , for the gas leakage field and the temperature and humidity values for the other two fields are sent to the think speak server. The graphs are updated in the server and thus the gas leakage is monitored.

```
void Send2thing()
{
  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);
Serial.println("AT+CIPMUX=0");
delay(2000);
Serial.println("AT+CSTT=\"ideagprs.com\"");//start task and setting the APN
delay(1000);
 Serial.println("AT+CIICR");//bring up wireless connection
delay(3000);
 Serial.println("AT+CIFSR");//get local IP adress
delay(2000);
 Serial.println("AT+CIPSPRT=0");
delay(3000);
Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");
//start up the connection
delay(5000);
 Serial.println("AT+CIPSEND");//begin send data to remote server
 delay(500);
String
str="GET http://api.thingspeak.com/update?api_key=PWRRSCYYTJGGY5T0&field1=" +
String(temp);
str+="&field2=";
str+=String(humi);
str+="&field3=";
str+=String(gas_value);
Serial.println(str);//begin send data to remote server
```

```
  delay(500);
  Serial.println((char)26);//sending
  delay(2000);//waiting for reply, important! the time is base on the condition of internet
  Serial.println();

  Serial.println("AT+CIPSHUT");//close the connection
  delay(100);
}
```

This is how the gas leakage monitoring system is implemented. The gas leakageis observed continuously and even if there is any leakage or not, everything is updated and monitored.

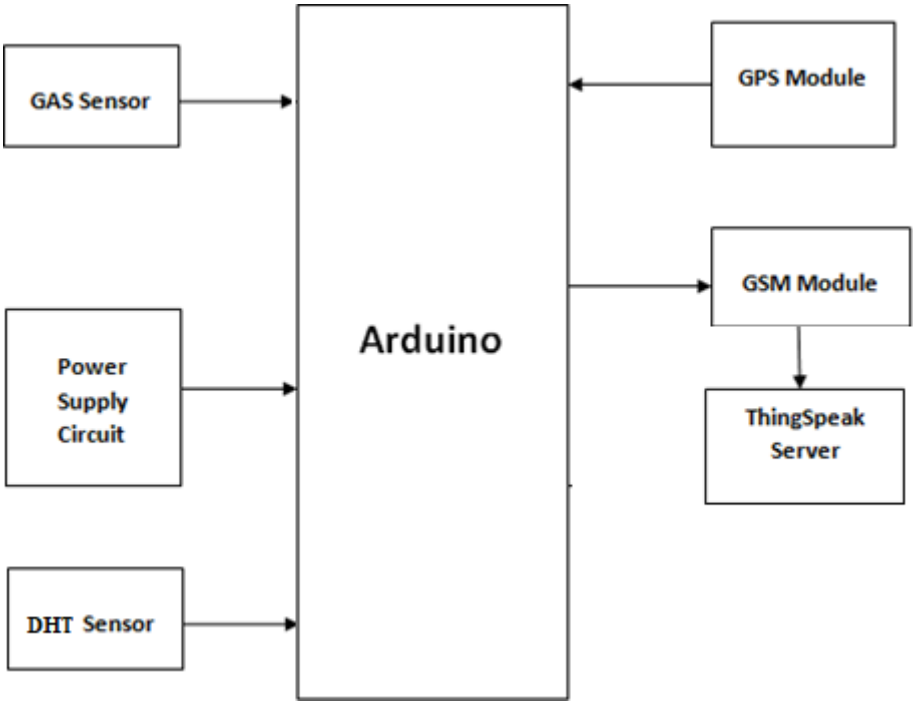

Fig 5.1.1: Block diagram of the system

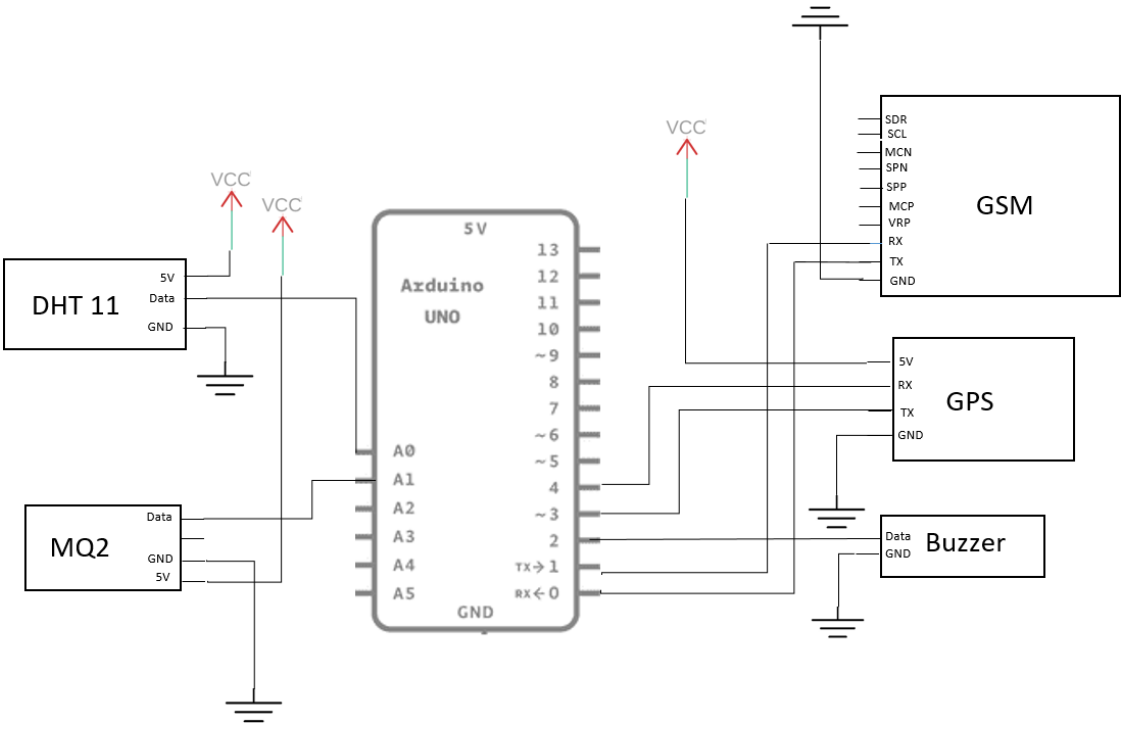

Fig 5.1.2: Schematic Diagram of the circuit

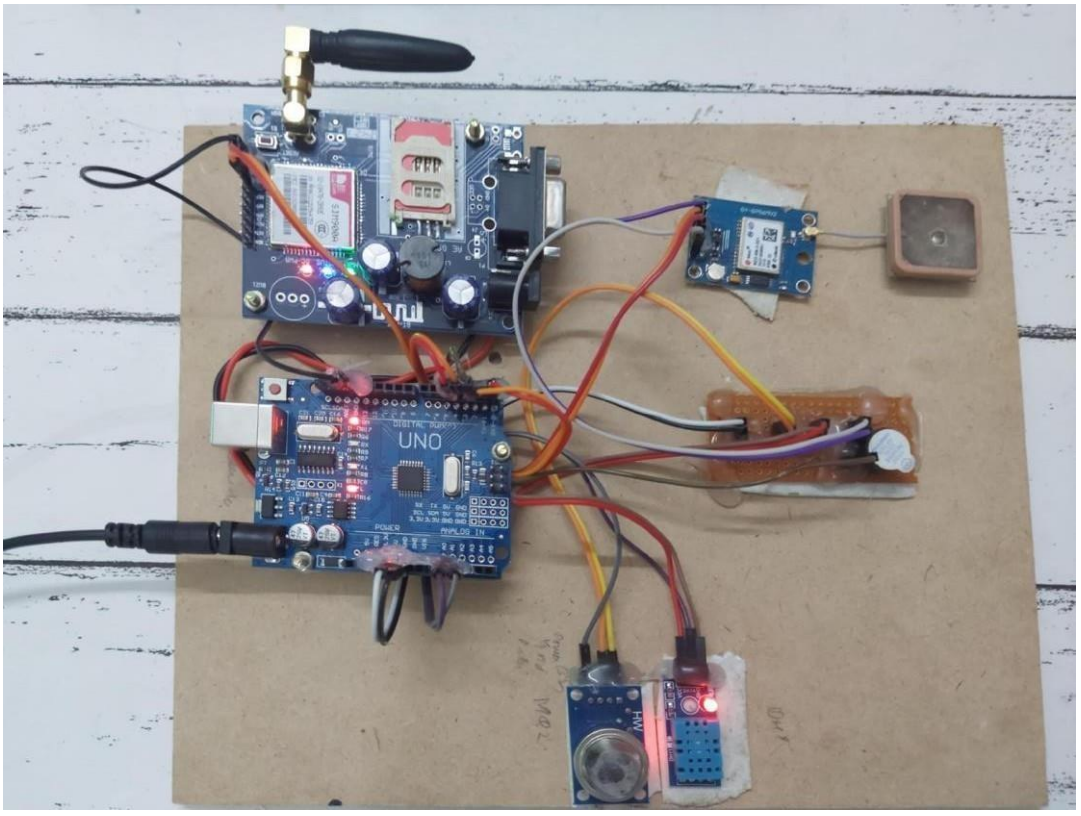

Fig 5.1.3: Gas Leakage monitoring system

6. SAMPLE CODE

Arduino code:

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
SoftwareSerial ss(3,4);
//SoftwareSerial mySerial(12,13);
#include<dht.h>
#define dht11 A0
#define gas A1
#define buzzer 2
TinyGPSPlus gps;
float lat = 18.5206,lon = 78.6310; // create variable for latitude and longitude object
dht DHT;
String latitude="17.438249",longitude="78.445030";
int temp,humi,gas_value;
void setup() {
  Serial.begin(9600);
  ss.begin(9600);
 // mySerial.begin(9600);
   pinMode(buzzer,OUTPUT);
}
void loop()
{
          //sendMessage();
          while (ss.available() > 0)
          if (gps.encode(ss.read()))
          {
          displayInfo();
            }
}
```

```
void displayInfo()
{
  Serial.print(F("Location: "));
  if (gps.location.isValid())
  {
    latitude=String(gps.location.lat(), 6);
    longitude=String(gps.location.lng(), 6);
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
    data();
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.print(F(" Date/Time: "));
  if (gps.date.isValid())
  {
    Serial.print(gps.date.month());
    Serial.print(F("/"));
    Serial.print(gps.date.day());
    Serial.print(F("/"));
    Serial.println(gps.date.year());
  }
  else
  {
    Serial.println(F("INVALID"));
  }
}
```

```
void data()
{
   DHT.read11(dht11);
  temp=DHT.temperature;
  humi=DHT.humidity;
  gas_value=analogRead(gas);
  Serial.print("Temperature : ");
  Serial.print(temp);
  Serial.println("C");
  Serial.print("Humidity :");
  Serial.println(humi);
  Serial.print("Gas Level : ");
  Serial.println(gas_value);
  if(temp>35)
  {
    Serial.println(F("Temperature HIGH"));
    digitalWrite(buzzer,HIGH);
    delay(1000);
    digitalWrite(buzzer,LOW);
    sendMessage(1);
  }
  if(gas_value>110)
  {
    Serial.println(F("Gas level HIGH"));
    digitalWrite(buzzer,HIGH);
    delay(1000);
    digitalWrite(buzzer,LOW);
    sendMessage(2);
  }
  delay(2000);
  Send2thing();
}
```

```
void sendMessage(int i)
{
  String msg;//="Alert! \n ";
  msg+="https://www.google.co.in/maps/place/";
  msg+=latitude+","+longitude;


switch(i)
  {
    case 1:msg+="\nTemperature HIGH";break;
    case 2:msg+="\nGas level HIGH";break;
  }
  Serial.println("AT+CMGF=1");//set the GSM Module in Text mode
  delay(1000);
  Serial.println("AT+CMGS=\"+917XXXXXXXXX\
  "\r");delay(1000);
  Serial.println(msg);//the SMS text you want to send
  delay(100);
  Serial.println((char)26);
  delay(100);
}
void Send2thing()
{
  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);
 Serial.println("AT+CIPMUX=0");
 delay(2000);
 Serial.println("AT+CSTT=\"ideagprs.com\"");//start task and setting the APN
 delay(1000);
  Serial.println("AT+CIICR");//bring up wireless connection
delay(3000);
Serial.println("AT+CIFSR");//get local IP adress
delay(2000);
Serial.println("AT+CIPSPRT=0");
delay(3000);
Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");
 //start up the connection
   delay(5000);
    Serial.println("AT+CIPSEND");//begin send data to remote server
    delay(500);
   String
 str="GET http://api.thingspeak.com/update?api_key=PWRRSCYYTJGGY5T0&field1=" +
 String(temp);
  str+="&field2=";
  str+=String(humi);
  str+="&field3=";
  str+=String(gas_value);
   Serial.println(str);//begin send data to remote server
   delay(500);
   Serial.println((char)26);//sending
   delay(2000);//waiting for reply, important! the time is base on the condition of internet
   Serial.println();
    Serial.println("AT+CIPSHUT");//close the connection
   delay(100);
 }
```

7. TESTING AND VALIDATION

7.1 INTRODUCTION

The system once finished has to go through a series of testing in order to ensure that it works the way it ought to. The various types of testing measures to be takes are test to see if the requirements are taken care of. Test to see if all the inputs are handled effectively. Test the system by traversing all the paths and discover my surprises check if errors and the exceptions have been handled properly. See ifvalidation of output data is taken care of testing objectives.

Testing

Testing is the process of evaluating a system or its components with the intent to find that whether it satisfied the specified requirements or not. Testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.

7.2 TESTING METHODOLIGIES

The following are the testing methodologies :

1.Unit Testing

2.Integration Testing

3.User Acceptance Testing

4.Validation Testing

5.Output Testing

1. Unit Testing

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether there are fit for use. In this the individual modules are tested to determine if there are

any issues by the developer himself. It is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Unit Testing Techniques:

- **Black Box Testing -** Using which the user interface, input and output are tested.
- **White Box Testing -** used to test each one of those functions behaviour is tested.
- **Gray Box Testing -** Used to execute tests, risks and assessment methods.

2. Integration Testing

Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drives and test stubs are used to assist in integration testing. It is also used to verify the functional, performance, and reliability between the modules that are integrated.

White Box Testing

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application based on the specifications. It is also known as Specifications based testing. Independent Testing Team usually performs this type of testing during the software testing life cycle.

This method of test can be applied to each and every level of software testing such as unit, integration, system and acceptance testing.

3. User Acceptance Testing

User acceptance testing is a testing methodology where the client/end users involved in testing the product to validate the product against their requirements. It is performed at client location at developer's site.

4. Validation Testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. It ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

5. Output Testing

After performing the validation testing, the next step is output testing of the proposed system. Since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in two ways- one is on screen and another is printed format.

7.3 DESIGN OF TEST CASES AND SCENARIOS

7.3.1 Test Case

A Test case is a document, which has a set of test data, preconditions, expected results and post-conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post-condition.

7.3.2 Test Scenario

It is a one line statement that notifies what area in the application will be tested. Test scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.

The terms 'test scenario' and 'test cases' are used interchangeably, however a test scenario has several steps, whereas a test case has a single step. Viewed from this perspective, test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.

7.3.3 Test cases and Results

Test-case ID	Test case Name	Expected Output	Actual Output	Status
1	Power on the system.	The system should be switched on and all the modules should be in working condition.	The system is switched on successfully.	Pass
2	Sensing Gas Leakage	The green led on the MQ- 2 Gas sensor lights up and the buzzer goes off.	Gas Leakage is sensed successfully.	Pass
3	SMS to the user.	The SMS Alert with the location is sent to number given in the code.	SMS received successfully.	Pass
4	ThingSpeak Server Connection	The gas leakage, temperature and humidity graphs are updated.	The graphs in the ThingSpeak server are updated.	Pass

Table 2: Test cases and results

7.4 VALIDATION

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. It ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfils its intended use when deployed on appropriate environment.

Validation helps in unfolding the exact functionality of the features and helps the testers to understand the product in much better way. It helps in making the product more user friendly. The system has been tested and implemented successfully and thus ensured that all the requirements are listed in the software requirements specification are completely fulfilled.

8. OUTPUT SCREENS

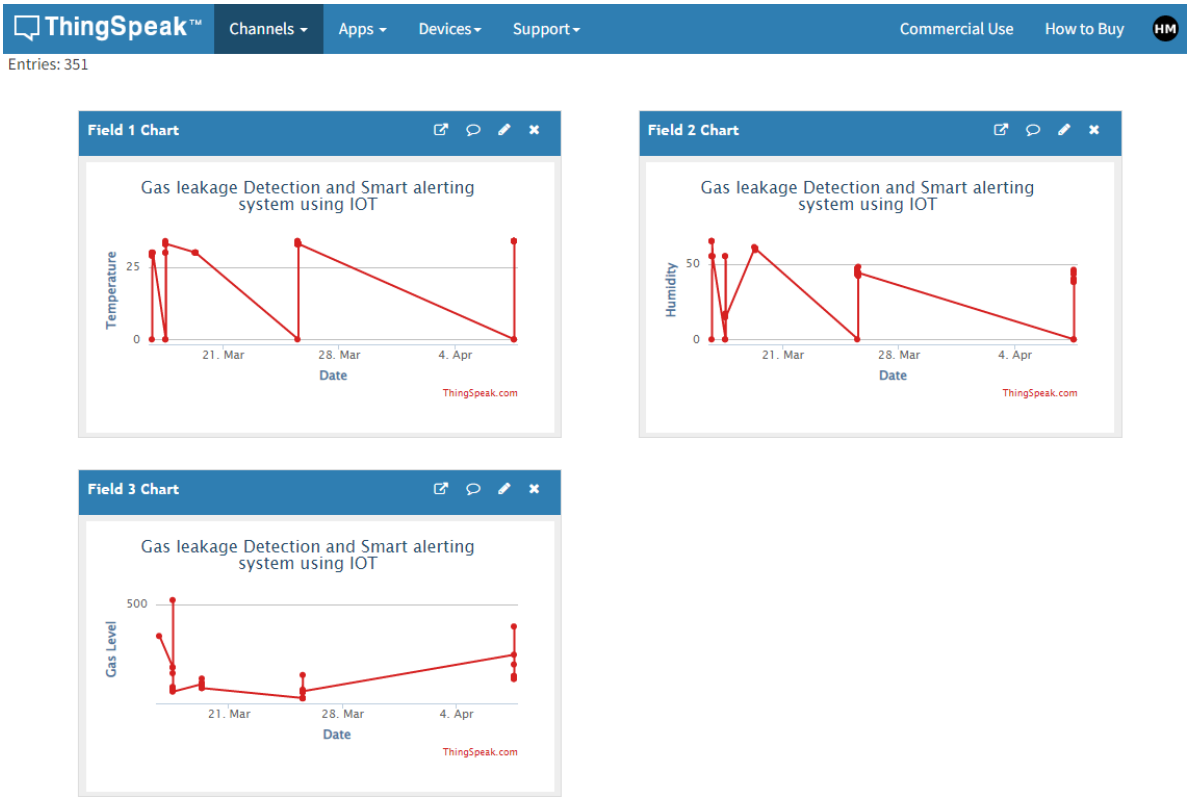

Fig 8.1: Gas Leakage Monitoring In ThingSpeak Server

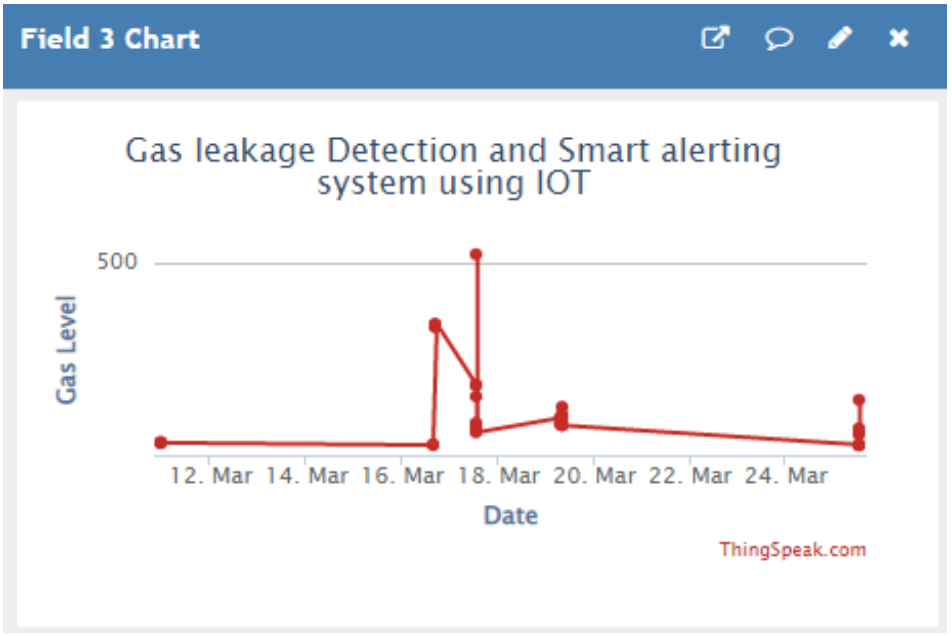

Fig 8.2: Gas Level Graph

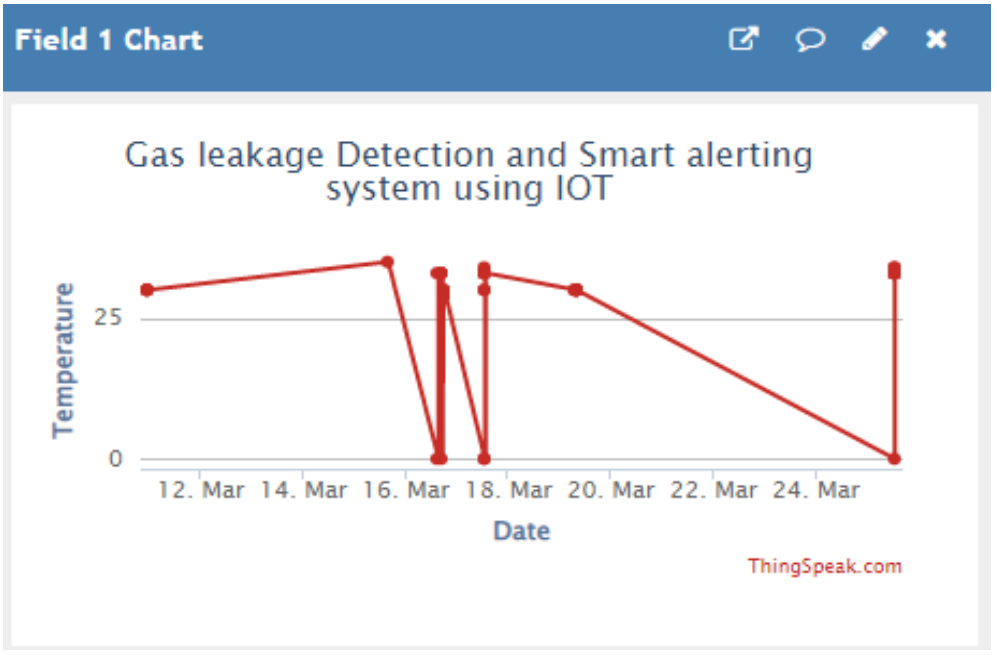

Fig 8.3:Temperature Latitude Graph

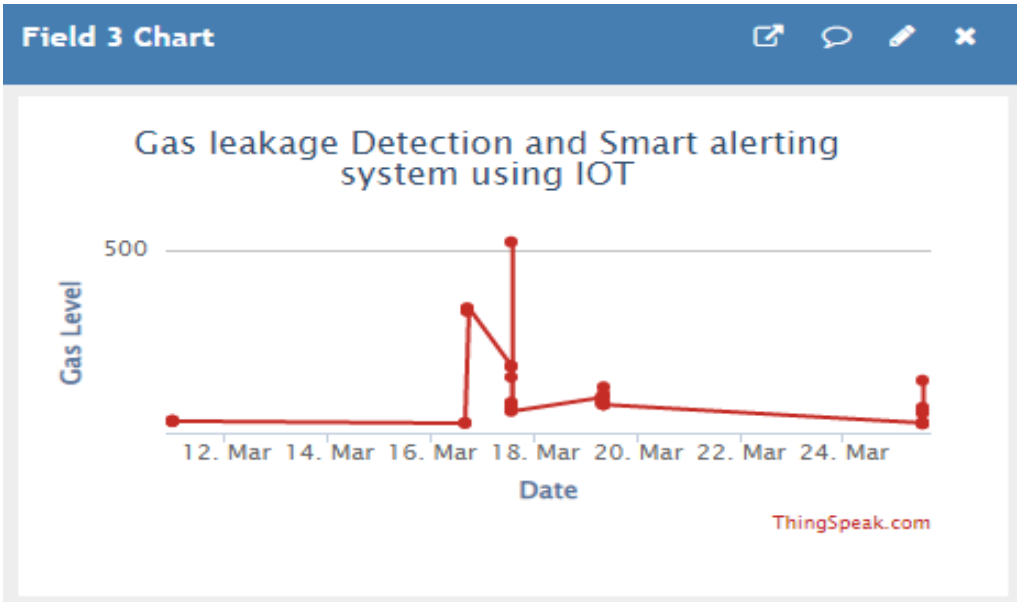

Fig 8.4: Humidity Graph

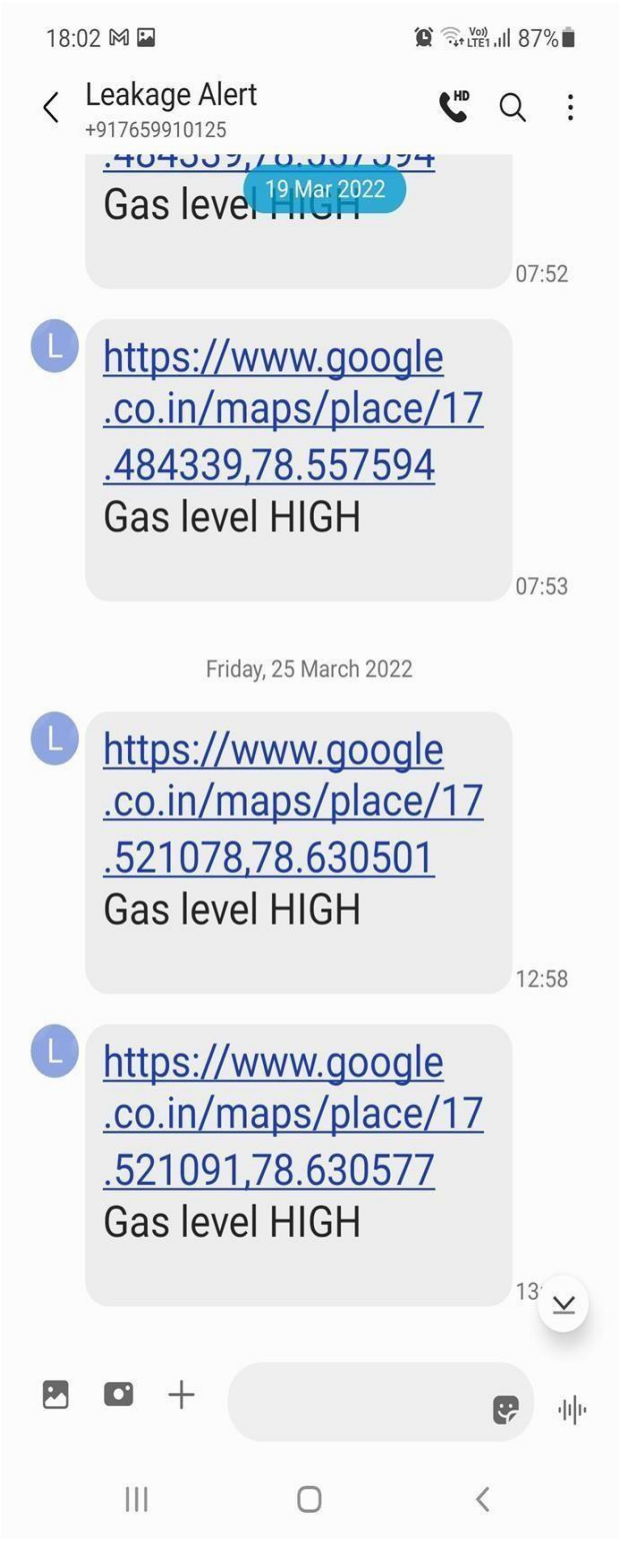

Fig 8.5: Message Alert For Gas Leakage

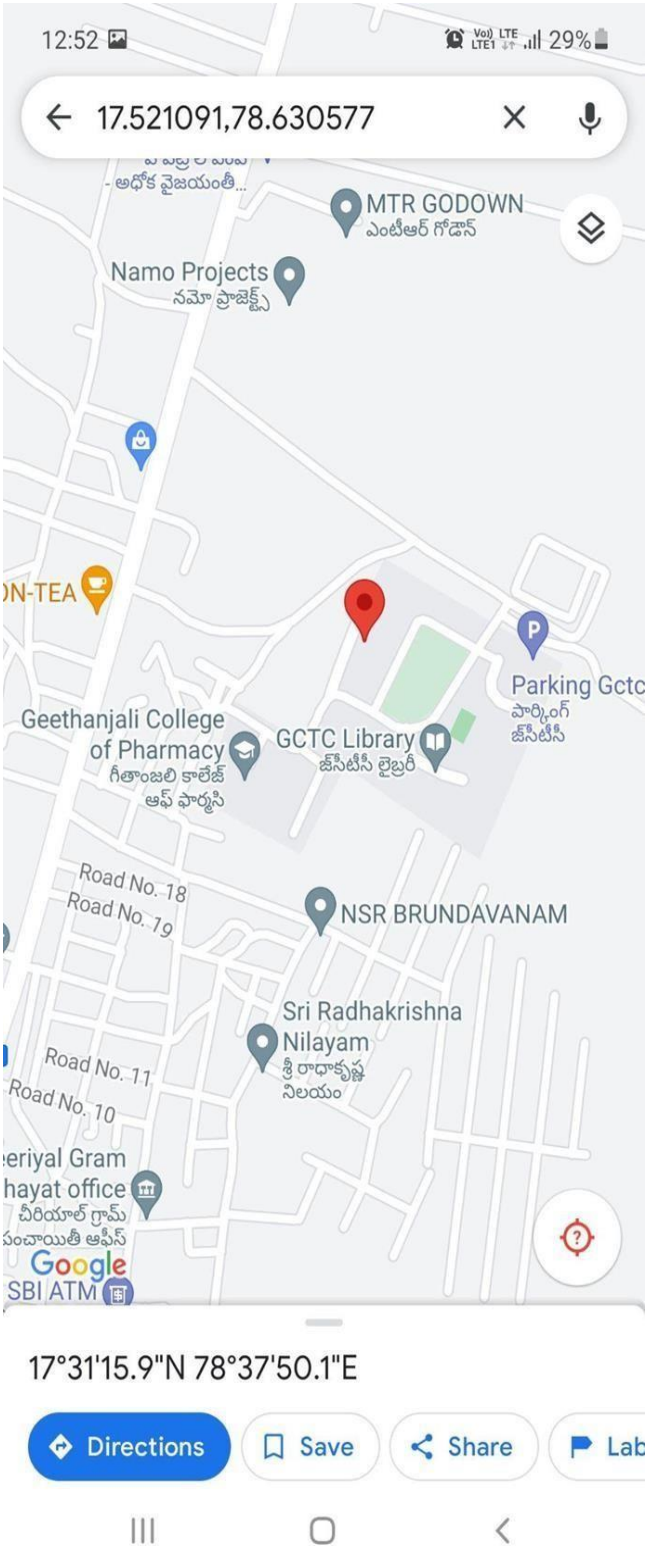

Fig 8.6: Google Maps Location of the gas leakage

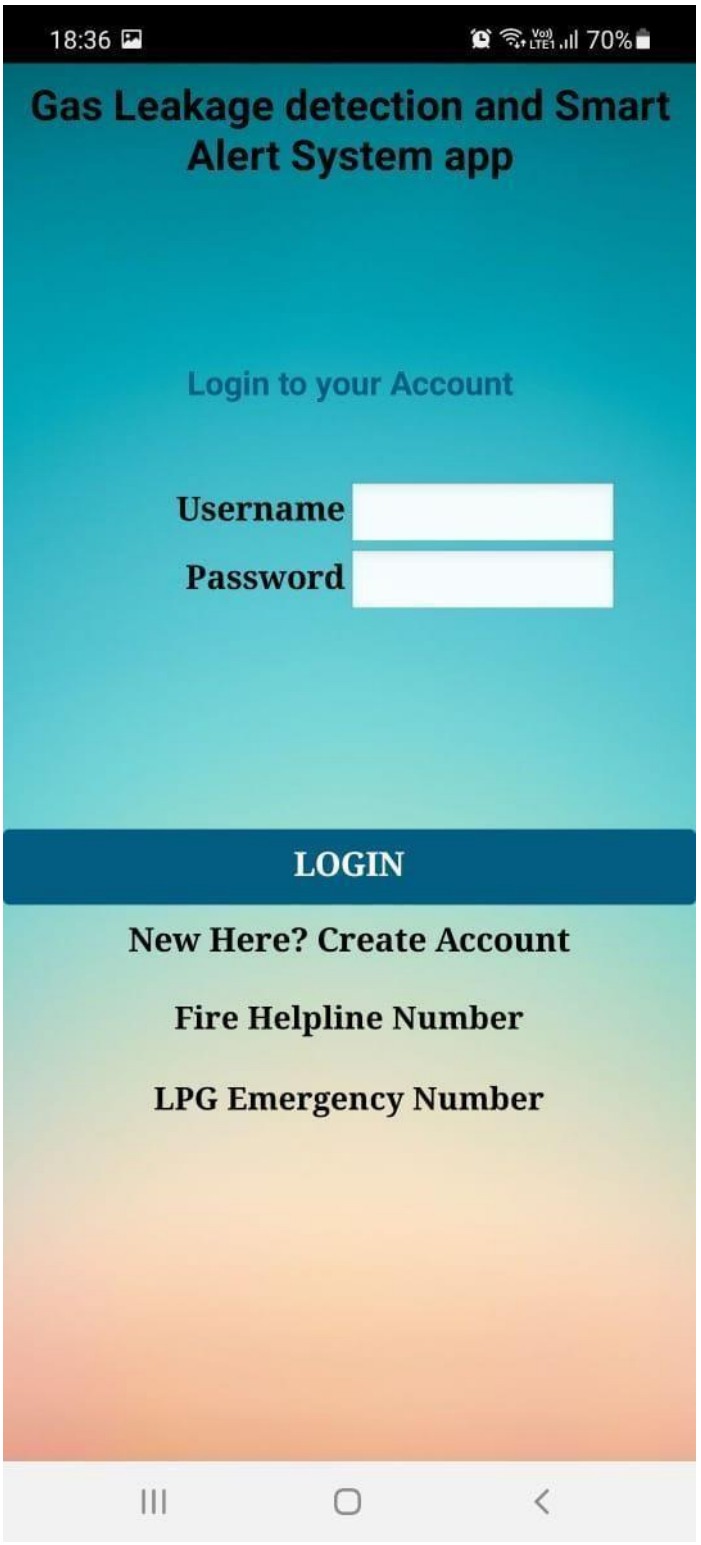

Fig 8.7: Login Screen of GLDSAS App

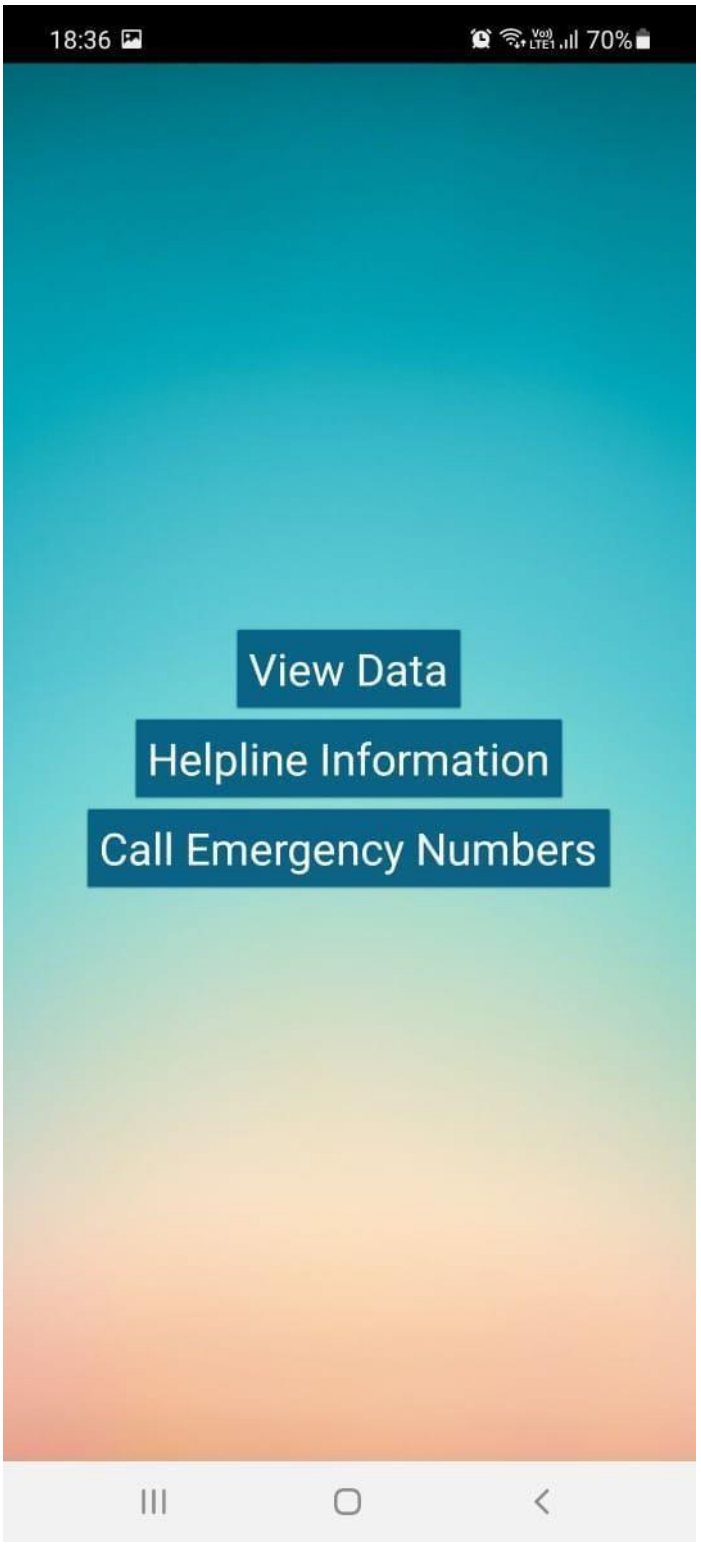

Fig 8.8: Home Screen of GLDSAS App

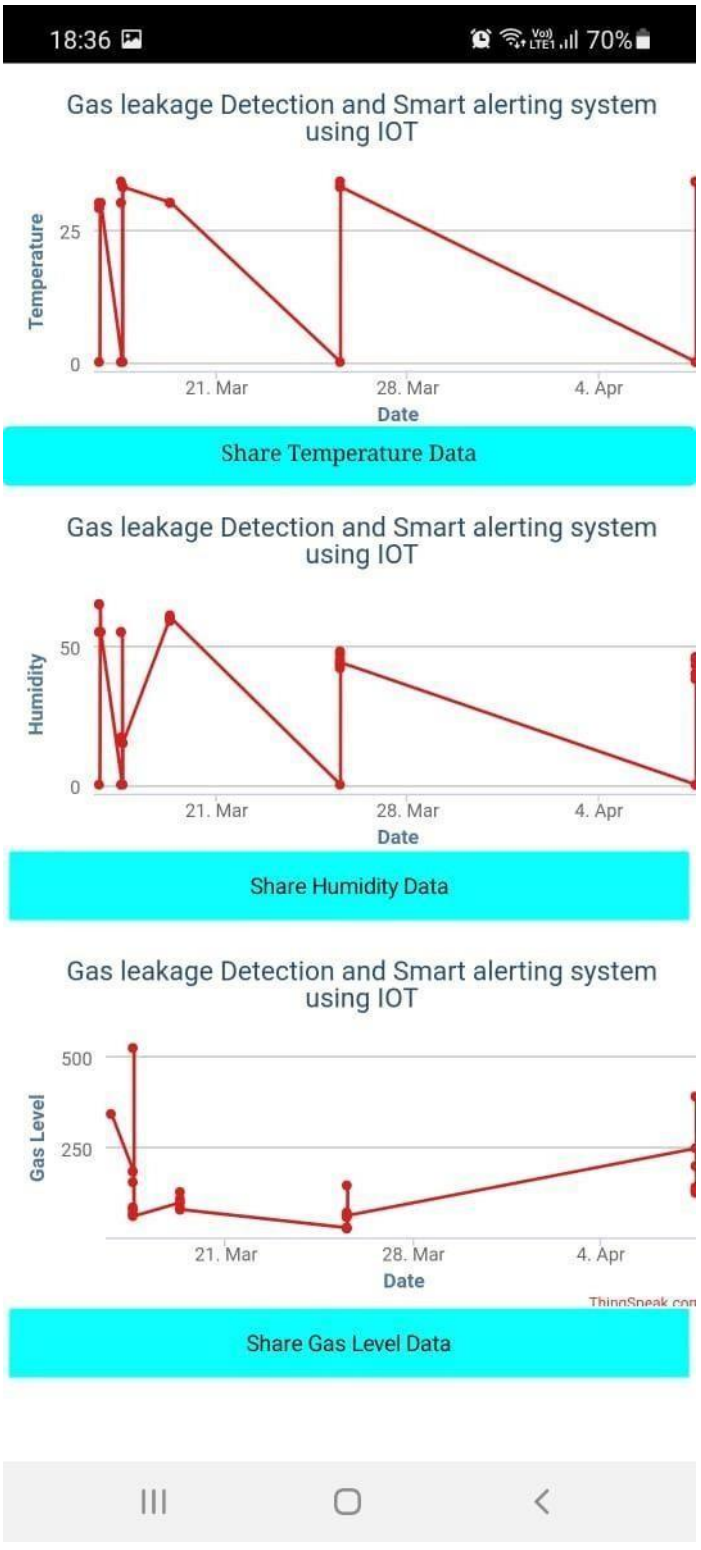

Fig 8.9: Show Data Screen of GLDSAS App

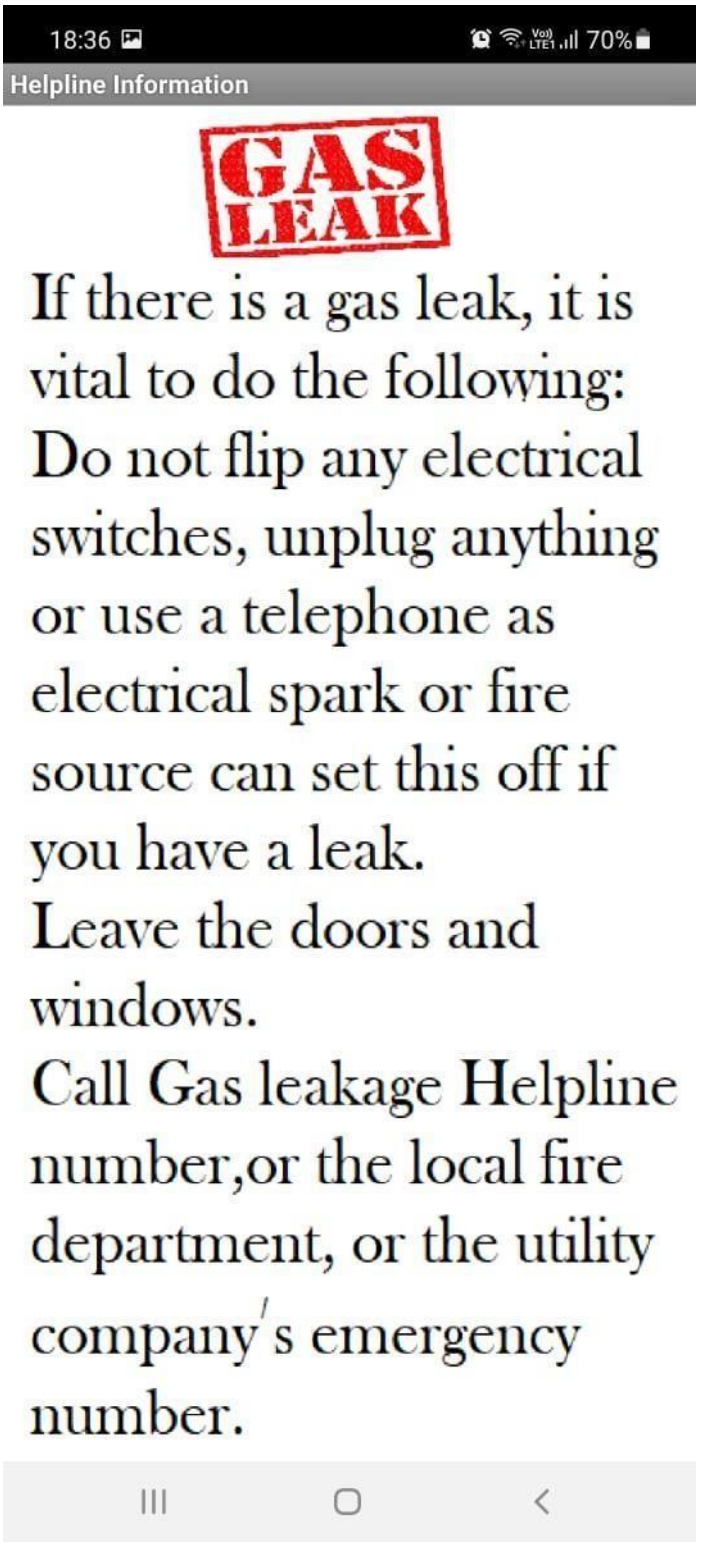

Fig 8.10: Precautions Information Screen of GLDSAS App

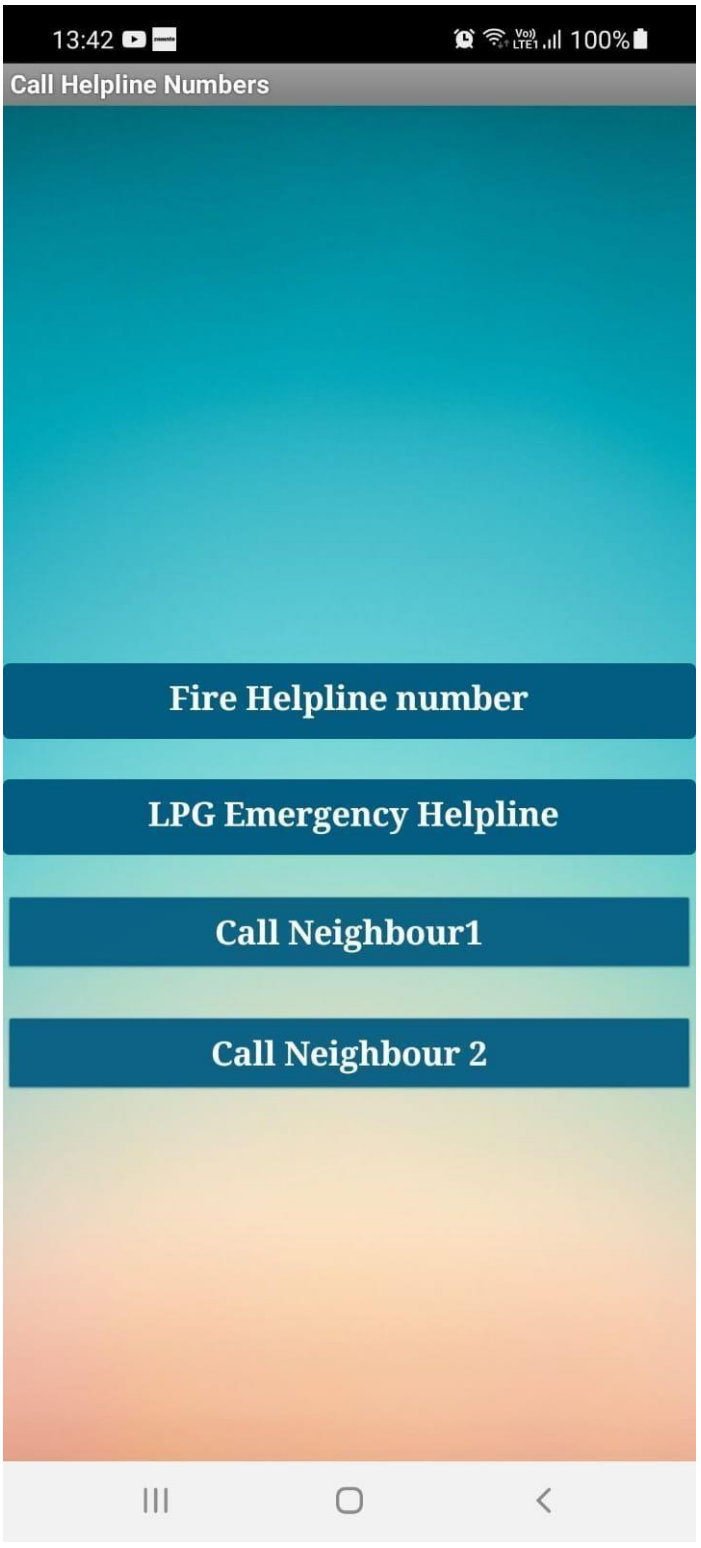

Fig 8.11: Call Helpline Numbers Screen of GLDSAS App

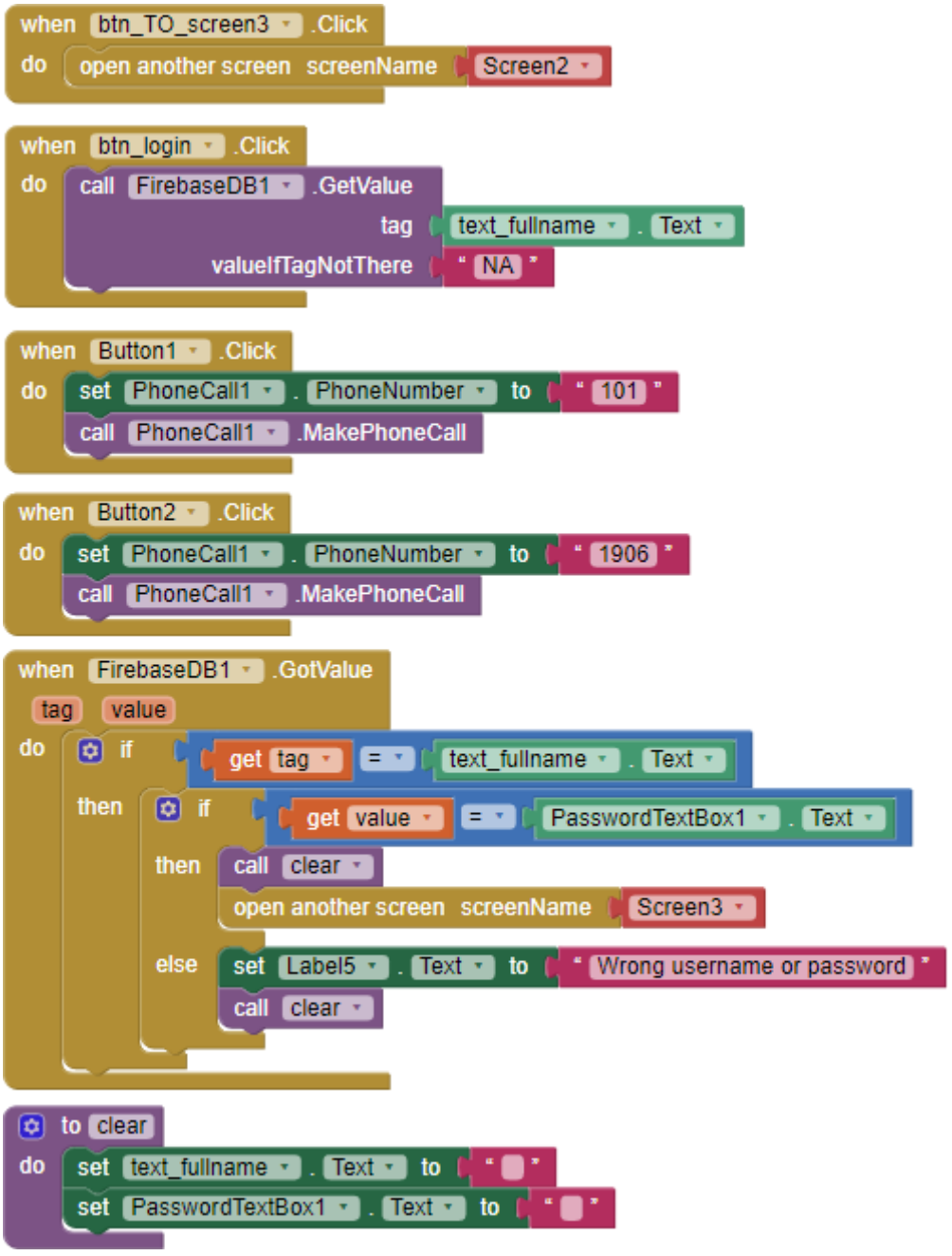

Fig 8.12: MIT App Inventor Blocks for Login Screen

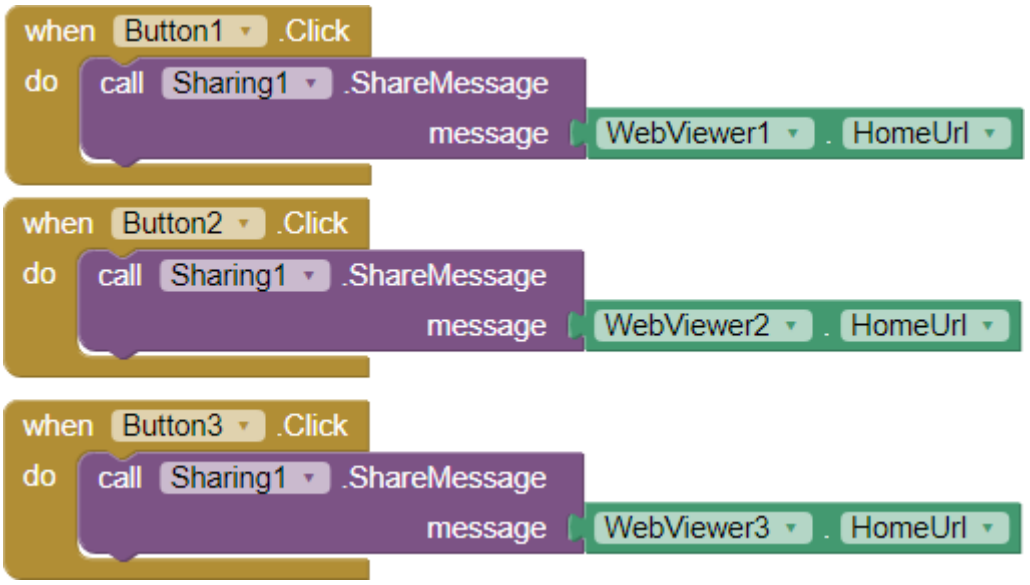

Fig 8.13: MIT App Inventor Blocks for Show Data Screen Blocks

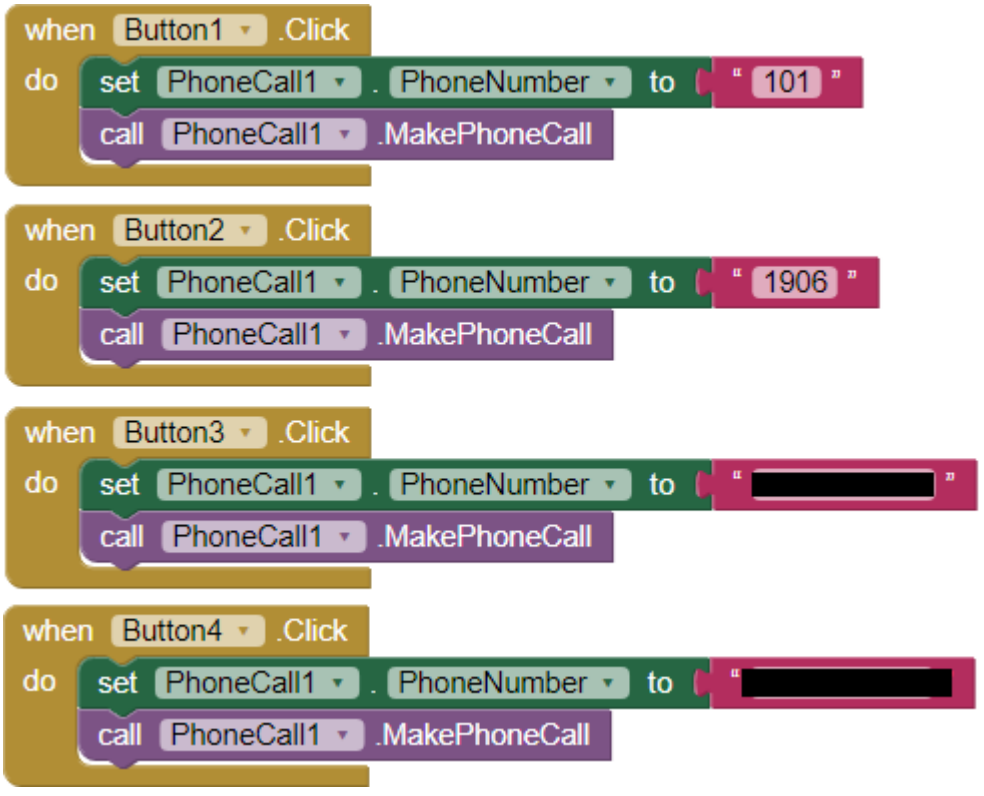

Fig 8.14: MIT App Inventor Blocks for Call Helpline Numbers Screen

9. CONCLUSION

In our modern scenario the usage of LPG has increased in a greater manner.As a result of this, the damages caused by the leakage of the gas is increasing day by day. Gas leakage leads to severe accidents resulting in material losses and human injuries. Gas leakage occurs mainly due to poor maintenance of equipments andinadequate awareness of the people. So as to eradicate this problems we are introducing highly advanced system known as Internet of Things (IoT). Gas leakage monitoring is essential to prevent accidents and to save human lives. This paper presented gas leakagemonitoring and alert system. This system triggers buzzer to alert people when gas leakage is detected and also sends a message along with the location. Using the Arduino microcontroller makes the system cheaper. Easy access and control makes the system simple yet reliable.

9.1 FUTURE ENHANCEMENTS

The project currently is mainly intended to alert the user and monitor the gas leakage. It can be enhanced further by making it possible for the system to not only monitor the gas leakage but also to control the leakage by switching off the source of the gas leakage. It is also possible to improve it further by allowing the system to sendmessages to the multiple people which is more effective.

10.BIBLIOGRAPHY

1. Embedded System By Vinod Kumar
2. The Internet Of Things by Samuel Greengard
3. Getting Started With Arduino by Massimo Banzi
4. www.Electronic projects.com
5. http://iosrjournals.org/iosr-jece/papers/Vol.%2011%20Issue%204/Version-1/B1104010612.pdf
6. http://ieeexplore.ieee.org/document/7972304/?anchor=authors
7. https://www.ijedr.org/papers/IJEDR1702333.pdf
8. http://www.iosrjournals.org/iosr-jeee/Papers/Conf.17017/Volume-3/13.%2082-87.pdf
9. https://thingspeak.com/
10. https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/
11. Gershenfeld, N., Krikorian, R. and Cohen, D. (2004) The Internet of Things.Scientific American
12. https://www.researchgate.net/publication/299597959_internet_of_things_iot_based_real_time_gas_leakage_monitoring_and_controlling
13. https://temboo.com/iot-applications/gas-leak-monitor
14. https://www.engineersgarage.com/contribution/microcontroller-based-lpg-gas-detector-using-gsm-module
15. https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/
16. J. Tsado, O. Imoru, S.O. Olayemi , ―"Design and construction of a GSM based gas leak Alert system"‖, IEEE Transaction,. IRJEEE Vol. 1(1), pp. 002-006, September, 2014.
17. Arduino: A Technical Reference by J.M.Huges
18. http://theijes.com/papers/NCIECE/Q01120116.pdf

11.APPENDICES

A. SOFTWARE USED

Arduino

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

MIT App Inventor

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology(MIT). It allows newcomers to computer programming to create application software (apps) for two operating systems (OS): Android, and iOS,

Google Firebase

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

B. METHODOLOGIES USED

Agile Methodology

Agile methodology is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model. The Agile software development methodology is one of the simplest and effective processes to turn a vision for a business need into software solutions. Agile is a term used to describe software development approaches that employ continual planning,

learning, improvement, team collaboration, evolutionary

The Roles in Agile Methodology

An agile software development process always starts by defining the users and documenting a vision statement on a scope of problems, opportunities, and values to be addressed. The product owner captures this vision and works with a multidisciplinary team (or teams) to deliver on this vision. Here are the roles in that process.

User

Agile processes always begin with the user or customer in mind. Today, we often define them with user personas to illustrate different roles in a workflow the software is supporting or different types of customer needs and behaviors.

Product Owner

The agile development process itself begins with someone who is required to be the voice of the customer, including any internal stakeholders. That person distills all the insights, ideas, and feedback to create a product vision. These product visions are often short and straightforward, but they nonetheless paint a picture of who the customer is, what values are being addressed, and a strategy on how to address them.

His or her responsibility is to define this vision and then work with a development team to make it real.

To work with the development team, the product owner breaks down the product vision into a series of user stories that spell out in more detail who the target user is, what problem is being solved for them, why the solution is important for them, and what constraints and acceptance criteria define the solution. These user stories are prioritized by the product owner, reviewed by the team to ensure they have a shared understanding on what is being asked of them.

Software Development Team

A development team is a group of people that work together to create software. This is complex, creative work that requires adaptability as technical challenges arise and business requirements evolve. In agile, the development team and its members' responsibilities differ from those in traditional software development.