CONTENTS

1. INTRODUCTION

The main objective of our project is to detect the leakage of the LPG gas in Industries asa well as in domestic houses.

Liquefied petroleum gas is being used for the past decades as industrial fuel and for domestic purpose. It has a characteristic of smokeless burning in the air. The main constituents of LPG are propane and butane and depending on the applications their proportions vary. Gas leakage detection in residential houses has become one of the fundamental issues in the recent times. Accidents mainly occur due to the negligence and technical fault. Electronic and press media have reported many accidents which were caused mainly because of gas leakage in residential houses and industries. A better system needs to be developed to reduce the accidents because of gas leakage. The gas is generally stored in metallic cylinders as its boiling point is lower than ambient temperature. Gas is molecularly heavy than other gases present in the air. Sowhenever thegas is leaked it settles closest to the ground level. And unless you provide a powerful exhaust system it cannot be forcefully disposed into open atmosphere.Now-a-days LPG leakage detection in homes, restaurants has been a common issue and the detection systems find applications in the market. Presently they are using load cell to measure the weight of the cylinder. When they find it become empty, consumer will order for a new cylinder. There may be a delay in providing the cylinder for few reasons like we may inform the service provider at the last moment when the gas is empty or there may be a delay in informing the gas provider. So in this system we will use a pressure sensor to measure the amount of gas present in the cylinder and also book the gas automatically when it reaches to a certain level.LPG is generally odourless and cannot be detected by human sense of smell because of its odourless nature. A pungent chemical is added to it purposely so that humans can detect the gas. There are few disadvantages anyhow. Firstly, it requires human presence in the vicinity. Secondly, by the time gas leakage is detected, its concentration in the vicinity may exceed the threshold level and may lead to explosion with the spark like light switch. Soin order to monitor its presence,sensing systems are deployed in the premises to detect the leakage and avoid accidents. Many sensors are available in the market which can warn the gas leakage. They make use of transmitters, controllers and other accessories but the cost of these kind of sensing systems is high and has technical complexity and also inaccurate with delays. Therefore, there is a need for the development of lower complexity, low cost and fast response systems.

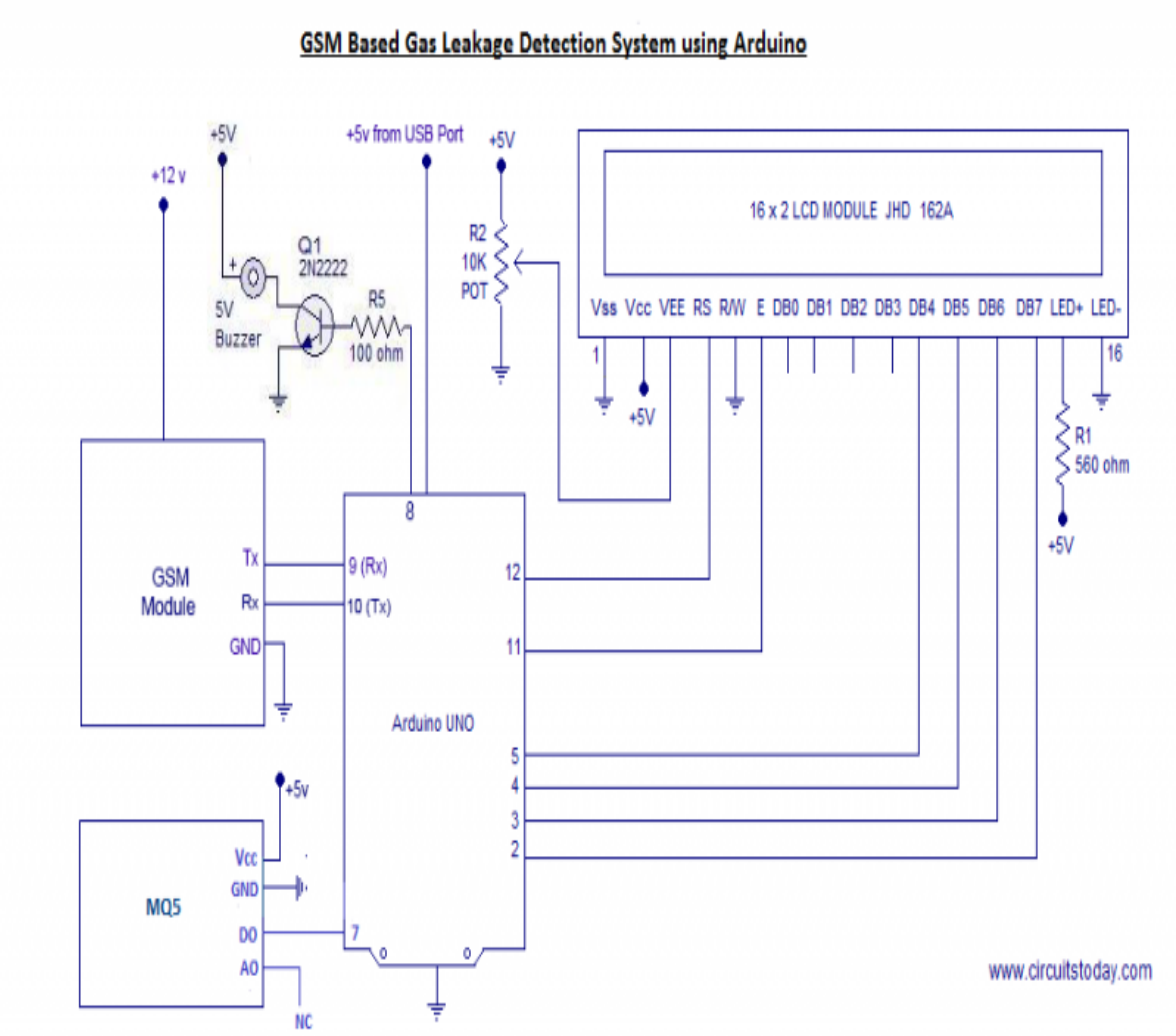
Abstract :

While LPG is an essential need of every household, its leakage could lead to a disaster. To alert on LPG leakage and prevent any mishappening there are various products to detect the leakage. Here we have developed an Arduino based LPG gas detector alarm. If gas leakage occurs, this system detects it and makes an alert by buzing the buzzer attached with the circuit.We have used a LPG gas sensor module to detect LPG Gas. When LPG gas leakage occurs, it gives a HIGH pulse on its DO pin and arduino continuously reads its DO pin. When Arduino gets a HIGH pulse from LPG Gas module it shows “LPG Gas Leakage Alert” message on 16x2 LCD and activates buzzer which beeps again and again until the gas detector module doesn't sense the gas in environment. When LPG gas detector module gives LOW pulse to arduino, then LCD shows “No LPG Gas Leakage” message.

The main aim of this project is to monitor for liquefied petroleum gas (LPG) leakage to avoid fire accidents providing house safety feature where security has been an important issue. The system detects the leakage of LPG using gas sensor and alerts the consumer about the gas leakage by sending SMS. Also it closes the regulator using electromagnetic valve and also switch ON the exhaust fan. The wireless communication is used between the exhaust fan and LPG gas module. The proposed uses the GSM to alert the person about the gas leakage via SMS. When the system detects the LPG concentration in the air exceeds the certain level then it immediately takes action by closing the regulator and switch ON the exhaust fan and alert the consumer by sending the SMS to the registered mobile phone. The gas sensor is used to monitor the leakage of gas and RF is used to switch ON the exhaust fan. This project is also used for automatic booking of the cylinder by using the pressure sensor. When the pressure is low, it indicates the cylinder is going to empty. At that time, the gas cylinder is automatically booked through SMS.

Keywords: GSM module, Electro Magnetic valve, gas sensor, pressure sensor, Micro controller, black marketing.

2. CIRCUIT DIAGRAM



3.WORKING

As shown in the schematic diagram above, it contains Arduino board, LPG GAS Sensor Module, buzzer and 16x2 LCD module. Arduino controls the whole process of this system like reading LPG Gas sensor module output, sending message to LCD and activating buzzer. We can set sensitivity of this sensor module by inbuilt potentiometer placed on it.

LPG gas sensor module's DO pin is directly connected to pin 18 (A4) of Arduino and Vcc and GND are connected to Vcc and GND of arduino. LPG gas sensor module consist a MQ3 sensor which detects LPG gas. This MQ3 sensor has a heater inside which needs some heater supply to heat up and it may takes up to 15 minute to get ready for detecting LPG gas. And a comparator circuit is used for converting Analog output of MQ3 in digital. A 16x2 LCD is connected with arduino in 4-bit mode. Control pin RS, RW and En are directly connected to arduino pin 2, GND and 3. And data pin D0-D7 are connected to 4, 5, 6, 7 of arduino. A buzzer is connected with arduino pin number 13 through a NPN BC547 transistor having a 1 k resistor at its base.

The MQ6 gas sensor detects the concentration of gas in ppm and outputs analog value which can be converted to digital measure using inbuilt Analog to Digital Convertor of Arduino. The value of the digital measure will be 10-bit long and varies from 0 to 1023. The project allows the user to set the dangerous level for leakage based on the same digital measure. When the value set by the user matches with that of the value detected by the sensor, it invokes the alarm. The MQ6 sensor can be calibrated by interfacing a load resistance of fixed value with the sensor.

The project is built on Arduino uno and is a portable device that can be installed anywhere. The Arduino sketch on the board manages to read data from the MQ-6 sensor, compare data and invoke an alarm.

4.THE CODE

```
include <SoftwareSerial.h>

#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

SoftwareSerial mySerial(9, 10);


int sensor=7;

int speaker=8;

int gas_value,Gas_alert_val, Gas_shut_val;

int Gas_Leak_Status;

int sms_count=0;




void setup()

{


pinMode(sensor,INPUT);

pinMode(speaker,OUTPUT);

Serial.begin(9600);

Serial.begin(9600);

lcd.begin(16,2);

delay(500);


}
```

```
void loop()
{
CheckGas();
//CheckShutDown();
}

void CheckGas()
{

lcd.setCursor(0,0);
lcd.print("Gas Scan - ON");
Gas_alert_val=ScanGasLevel();
if(Gas_alert_val==LOW)
{
 SetAlert(); // Function to send SMS Alerts
}}

int ScanGasLevel()
{
gas_value=digitalRead(sensor); // reads the sensor output (Vout of LM35)

return gas_value; // returns temperature value in degree celsius
}
```

```
void SetAlert()
{
digitalWrite(speaker,HIGH);
while(sms_count<3) //Number of SMS Alerts to be sent
{
SendTextMessage(); // Function to send AT Commands to GSM module
}
Gas_Leak_Status=1;
lcd.setCursor(0,1);
lcd.print("Gas Alert! SMS Sent!");
}

void CheckShutDown()
{
if(Gas_Leak_Status==1)
{

Gas_shut_val=ScanGasLevel();
if(Gas_shut_val==HIGH)
{

lcd.setCursor(0,1);
lcd.print("No Gas Leaking");
digitalWrite(speaker, LOW);
sms_count=0;
```

```
Gas_Leak_Status=0;
}}}

void SendTextMessage()
{
  Serial.println("AT+CMGF=1");    //To send SMS in Text Mode
  delay(1000);
  Serial.println("AT+CMGS=\"+917981297508\"\r"); // change to the phone number you using
  delay(1000);
  Serial.println("Gas Leaking!");//the content of the message
  delay(200);
  Serial.println((char)26);//the stopping character
  delay(1000);
  Serial.println("AT+CMGS=\"**********\"\r"); // change to the phone number you using
  delay(1000);
  Serial.println("Gas Leaking!");//the content of the message
  delay(200);
  Serial.println((char)26);//the message stopping character
  delay(1000);
  sms_count++;
}
```

5. ARDUINO UNO

Arduino is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL).permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as do-it-yourself (DIY) kits.

Arduino board designs use a variety of microprocessors and controllers. The boards are 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 boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages ((C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the ((Processing language project.

The Arduino project started in 2003 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

The name *Arduino* comes from a bar in Ivrea, Italy, where some of the founders of the project used to meet. The bar was named after Arduin of Ivrea, who was the margrave of the March of Ivrea and King of Italy from 1002 to 1014.

History

The Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy.[2] At that time, the students used a BASIC Stamp microcontroller at a cost of $100, a considerable expense for many students. In 2003 Hernando Barragán created the development platform ((*Wiring* as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas, who are known for work on the ((Processing language. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller. In 2003, Massimo Banzi, with David Mellis, another IDII student, and David Cuartielles, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they ((forked the project and renamed it *Arduino.*

The initial Arduino core team consisted of Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis, but Barragán was not invited to participate.

Following the completion of the Wiring platform, lighter and less expensive versions were distributed in the open-source community.

Adafruit Industries, a New York City supplier of Arduino boards, parts, and assemblies, estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands.

In October 2016, Federico Musto, Arduino's former CEO, secured a 50% ownership of the company. In April 2017, ((Wired reported that Musto had "fabricated his academic record.... On his company's website, personal LinkedIn accounts, and even on Italian business documents, Musto was until recently listed as holding a PhD from the Massachusetts Institute of Technology. In some cases, his bios also claimed an MBA from New York University." Wired reported that neither University had any record of Musto's attendance, and Musto later admitted in an interview with Wired that he had never earned those degrees.

Around that same time, Massimo Banzi announced that the (Arduino Foundation would be "a new beginning for Arduino." But a year later, the Foundation still hasn't been established, and the state of the project remains unclear.

The controversy surrounding Musto continued when, in July 2017, he reportedly pulled many Open source licenses, schematics, and code from the Arduino website, prompting scrutiny and outcry.

In October 2017, Arduino announced its partnership with ARM Holdings (ARM). The announcement said, in part, "ARM recognized independence as a core value of Arduino ... without any lock-in with the ARM architecture.” Arduino intends to continue to work with all technology vendors and architectures.

Trademark dispute

In early 2008, the five cofounders of the Arduino project created a company, Arduino LLC, to hold the trademarks associated with Arduino. The manufacture and sale of the boards was to be done by external companies, and Arduino LLC would get a royalty from them. The founding bylaws of Arduino LLC specified that each of the five founders transfer ownership of the Arduino brand to the newly formed company.

At the end of 2008, Gianluca Martino's company, Smart Projects, registered the Arduino trademark in Italy and kept this a secret from the other cofounders for about two years. This was revealed when the Arduino company tried to register the trademark in other areas of the world (they originally registered only in the US), and discovered that it was already registered in Italy. Negotiations with Gianluca and his firm to bring the trademark under control of the original Arduino company failed. In 2014, Smart Projects began refusing to pay royalties. They then appointed a new CEO, Federico Musto, who renamed the company *Arduino SRL* and created the website *arduino.org*, copying the graphics and layout of the original *arduino.cc*. This resulted in a rift in the Arduino development team.

In January 2015, Arduino LLC filed a lawsuit against Arduino SRL.

In May 2015, Arduino LLC created the worldwide trademark **Genuino**, used as brand name outside the United States.

At the World Maker Faire in New York on October 1, 2016, Arduino LLC co-founder and CEO Massimo Banzi and Arduino SRL CEO Federico Musto announced the merger of the two companies.

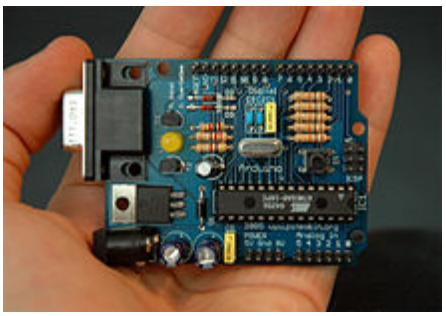
In July 2017 BCMI, founded by Massimo Banzi, David Cuartielles, David Mellis and Tom Igoe, acquired Arduino AG and all the Arduino trademarks. Fabio Violante is the new CEO replacing Federico Musto, no more working for Arduino AG.

Hardware

Arduino-compatible R3 UNO board made in China with no Arduino logo, but with identical markings, including "*Made in Italy*" text

Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available. The source code for the IDE is released under the GNU General Public License, version 2. Nevertheless, an official Bill of Materials of Arduino boards has never been released by Arduino staff.

Although the hardware and software designs are freely available under copyleft licenses, the developers have requested the name *Arduino* to be exclusive to the official product and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product. Several Arduino-compatible products commercially released have avoided the project name by using various names ending in *-duino*.



An early Arduino board with an RS-232 serial interface (upper left) and an Atmel ATmega8 microcontroller chip (black, lower right); the 14 digital I/O pins are at the top, the 6 analog input pins at the lower right, and the power connector at the lower left.

Most Arduino boards consist of an Atmel 8-bit AVR microcontroller (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560) with varying amounts of flash memory, pins, and features. The 32-bit Arduino Due, based on the Atmel SAM3X8E was introduced in 2012. The boards use single or double-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed *shields*. Multiple, and possibly stacked shields may be individually addressable via an I²C serial bus. Most boards include a 5 V linear regulator and a 16

MHz crystal oscillator or ceramic resonator. Some designs, such as the LilyPad, run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions.

Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory. The default bootloader of the Arduino UNO is the optiboot bootloader. Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. When used with traditional microcontroller tools, instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

An official Arduino Uno R2 with descriptions of the I/O locations

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The *Diecimila*, *Duemilanove* and current *Uno* provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.

Official boards

Further information: List of Arduino boards and compatible systems

The original Arduino hardware was produced by the Italian company Smart Projects. Some Arduino-branded boards have been designed by the American companies

SparkFun Electronics and Adafruit Industries. As of 2016, 17 versions of the Arduino hardware have been commercially produced.

1.Arduino RS232
(thru-hole parts)

2.Arduino Diecimila

3.Arduino Duemilanove
(rev 2009b)

4.Arduino Uno R2

5.Arduino Uno SMD R3

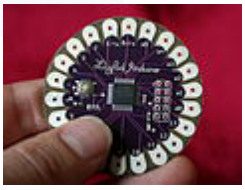
6.Arduino Leonard

7.Arduino Pro
(No USB)

8.Arduino Mega

9.Arduino Nano
(DIP-30 footprint)

10.Arduino LilyPad 00
(rev 2007) (No USB)



11.Arduino Robot

12.Arduino Esplora

13.Arduino Ethernet
(AVR + W5100)

14.Arduino Yun
(AVR + AR9331)

15.Arduino Due
(ARM Cortex-M3 core)

Shields

Arduino and Arduino-compatible boards use printed circuit expansion boards called *shields*, which plug into the normally supplied Arduino pin headers. Shields can provide motor controls for 3D printing and other applications, Global Positioning System (GPS), Ethernet, liquid crystal display (LCD), or breadboarding (prototyping). Several shields can also be made do it yourself (DIY).

1. Multiple shields can be stacked. In this example the top shield contains a solder less breadboard.

2.Dragino Lora Shield allows the user to send data and reach extremely long ranges at low data-rates.

3. Screw-terminal breakout shield in a wing-type format

4.Adafruit Motor Shield with screw terminals for connection to motors

5.Adafruit Datalogging Shield with a Secure Digital (SD) card slot and real-time clock (RTC) chip

Software and programming tools

A program for Arduino may be written in any programming language with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language ((Java. It originated from the IDE for the languages ((*Processing* and ((*Wiring*. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

A program written with the IDE for Arduino is called a *sketch*. Sketches are saved on the development computer as text files with the file extension *.ino*. Arduino Software (IDE) pre-1.0 saved sketches with the extension *.pde*.

The Arduino IDE supports the languages ((C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the ((Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

The open-source nature of the Arduino project has facilitated the publication of many free software libraries that other developers use to augment their projects.

Program structure

Power LED (red) and User LED (green) attached to Pin 13 on an Arduino compatible board

A minimal Arduino C/C++ program consist of only two functions:

setup(): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.

loop(): After *setup()* has been called, function *loop()* is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

Most Arduino boards contain a light-emitting diode (LED) and a load resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions. A typical program for a beginning Arduino programmer blinks a LED repeatedly.

Example:

```
#define LED_PIN 13                // Pin number attached to LED.

void setup() {
   pinMode(LED_PIN, OUTPUT);      // Configure pin 13 to be a digital output.
}

void loop() {
   digitalWrite(LED_PIN, HIGH);   // Turn on the LED.
   delay(1000);                   // Wait 1 second (1000 milliseconds).
   digitalWrite(LED_PIN, LOW);    // Turn off the LED.
   delay(1000);                   // Wait 1 second.
}
```

This program uses the functions *pinMode()*, *digitalWrite()*, and *delay()*, which are provided by the internal libraries included in the IDE environment. The program is usually loaded in the Arduino by the manufacturer.

Applications

- ★ Arduboy, a handheld game console based on Arduino.
- ★ Arduino Motion Control Rig.
- ★ Arduinome, a MIDI controller device that mimics the Monome.
- ★ ArduinoPhone, a do-it-yourself cellphone.
- ★ Ardupilot, drone software and hardware.
- ★ ArduSat, a cubesat based on Arduino.

6. MQ-5 GAS SENSOR

Grove - Gas Sensor(MQ5)

Introduction

The Grove - Gas Sensor(MQ5) module is useful for gas leakage detection (in home and industry). It is suitable for detecting H2, LPG, CH4, CO, Alcohol. Due to its high sensitivity and fast response time, measurements can be taken as soon as possible. The sensitivity of the sensor can be adjusted by using the potentiometer.

Sensor	Gas Type	Get One Now	
MQ2	Combustible Gas, Smoke		
MQ3	Alcohol Vapor		
MQ5	LPG, Natural Gas, Town Gas		
MQ9	Carbon Monoxide, Coal Gas, Liquefied Gas		

Features

- ★ Wide detecting scope
- ★ Stable and long life
- ★ Fast response and High sensitivity

Specification

Item	Parameter	Min	Typical	Max	Unit
VCC	Working Voltage	4.9	5	5.1	V
PH	Heating consumption	0.5	-	800	mW
RL	Load resistance		adjustable		
RH	Heater resistance	-	31±10%	-	Ω
Rs	Sensing Resistance	10	-	60	kΩ
Scope	Detecting Concentration	200	-	10000	ppm

Platforms Supported

★ Arduino uno
★ Wio board
★ Beaglebone module
★ Rasbarry pi
★ Linklt one

Application Ideas

- ★ Gas leakage detection.
- ★ Toys.
- ★ Gas reporter
- ★ Measurement application

Hardware Overview

This is an Analog output sensor. This needs to be connected to any one Analog socket in Grove Base Shield. The examples used in this tutorial makes uses of A0 analog pin. Connect this module to the A0 port of Base Shield.

It is possible to connect the Grove module to Arduino directly by using jumper wires by using the connection as shown in the table below:

Arduino	Gas Sensor
5V	VCC
GND	GND
NC	NC
Analog A0	SIG

The output voltage from the Gas sensor increases when the concentration of gas increases. Sensitivity can be adjusted by varying the potentiometer. Please note that the best preheat time for the sensor is above 24 hours. For detailed information about the MQ-5 sensor, please refer to the data-sheet provided in **Resources** section.

Getting Started

Connect the Grove - Gas Sensor(MQ5) to A0 port as shown in the picture above.

Gas Detection : Basic Example

In this example, the sensor is connected to A0 pin. The voltage read from the sensor is displayed. This value can be used as a threshold to detect any increase/decrease in gas concentration.

```
void setup() {

    Serial.begin(9600);
}

void loop() {
    float sensor_volt;
    float sensorValue;

    sensorValue = analogRead(A0);
    sensor_volt = sensorValue/1024*5.0;

    Serial.print("sensor_volt = ");
    Serial.print(sensor_volt);
    Serial.println("V");
    delay(1000);
}
```

Measurement : Approximation

This examples demonstrates a way to know the approximate concentration of Gas. As per the data-sheet of the MQ5 sensors, these equations are tested for standard conditions and are not calibrated. It may vary based on change in temperature or humidity.

Keep the Gas Sensor in clean air environment. Upload the program below.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  float sensor_volt;
  float RS_air; //  Get the value of RS via in a clear air
  float R0;  // Get the value of R0 via in H2
  float sensorValue;

    /*--- Get a average data by testing 100 times ---*/
  for(int x = 0 ; x < 100 ; x++)
  {
    sensorValue = sensorValue + analogRead(A0);
  }
  sensorValue = sensorValue/100.0;
    /*-----------------------------------------*/

  sensor_volt = sensorValue/1024*5.0;
  RS_air = (5.0-sensor_volt)/sensor_volt; // omit *RL
  R0 = RS_air/6.5; // The ratio of RS/R0 is 6.5 in a clear air from Graph (Found using WebPlotDigitizer)

  Serial.print("sensor_volt = ");
  Serial.print(sensor_volt);
  Serial.println("V");

  Serial.print("R0 = ");
  Serial.println(R0);
  delay(1000);
}
```

Then, open the serial monitor of Arduino IDE. Write down the value of R0 and this needs to be used in the next program. Please node down the R0 after the reading stabilizes.

Replace the R0 below with value of R0 tested above . Expose the sensor to any one of the gas listed above.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
```

```
  float sensor_volt;
  float RS_gas; // Get value of RS in a GAS
  float ratio; // Get ratio RS_GAS/RS_air
  int sensorValue = analogRead(A0);
  sensor_volt=(float)sensorValue/1024*5.0;
  RS_gas = (5.0-sensor_volt)/sensor_volt; // omit *RL

      /*-Replace the name "R0" with the value of R0 in the demo of First Test -*/
  ratio = RS_gas/R0;  // ratio = RS/R0
      /*-----------------------------------------------------------------------*/

  Serial.print("sensor_volt = ");
  Serial.println(sensor_volt);
  Serial.print("RS_ratio = ");
  Serial.println(RS_gas);
  Serial.print("Rs/R0 = ");
  Serial.println(ratio);

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

  delay(1000);
```

Now, we can get the concentration of gas from the figure below.According to the figure, we can see that the minimum concentration we can test is 200ppm and the maximum is 10000ppm, in a other word, we can get a concentration of gas between 0.02% and 1%. However, we can't provide a formula because the relation between ratio and concentration is nonlinear.

Table 1 – Technical Specifications of the MQ-5 Gas leakage sensor.

Model No.			MQ-5
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			LPG, Methane, coal gas
Concentration			300-10000ppm(Methane, Propane, Butane, H2)
Circuit	Loop Voltage	V_C	≤24V DC
	Heater Voltage	V_H	5.0V±0.2V AC or DC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3ΩRoom Tem.
	Heater consumption	P_H	≤900mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 2000ppm C_3H_8)
	Sensitivity	S	Rs(in air)/Rs(1000ppm C_3H_8)≥5
	Slope	α	≤0.6($R_{1000ppm}/R_{500ppm}$ H_2)
Condition	Tem. Humidity		20±265%±5%RH
	Standard test circuit		Vc:5.0V±0.1V V_H: 5.0V±0.1V
	Preheat time		Over 48 hours

7. GSM MODULE

The GSM logo is used to identify compatible devices and equipment. The dots symbolize three clients in the home network and one roaming client.

GSM (**Global System for Mobile Communications**, originally ***Groupe Spécial Mobile***) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation digital cellular networks used by mobile devices such as tablets, first deployed in Finland in December 1991.[2] As of 2014, it has become the global standard for mobile communications – with over 90% market share, operating in over 193 countries and territories.

2G networks developed as a replacement for first generation (1G) analog cellular networks, and the GSM standard originally described as a digital, circuit-switched network optimized for ((full duplex voice telephony. This expanded over time to include data communications, first by circuit-switched transport, then by packet data transport via GPRS (General Packet Radio Services) and EDGE (Enhanced Data rates for GSM Evolution, or EGPRS).

Subsequently, the 3GPP developed third-generation (3G) UMTS standards, followed by fourth-generation (4G) LTE Advanced standards, which do not form part of the ETSI GSM standard.

"GSM" is a trademark owned by the GSM Association. It may also refer to the (initially) most common voice codec used, Full Rate.

History

In 1983, work began to develop a European standard for digital cellular voice telecommunications when the European Conference of Postal and Telecommunications Administrations (CEPT) set up the Groupe Spécial Mobile committee and later provided a permanent technical-support group based in Paris. Five years later, in 1987, 15 representatives from 13 European countries signed a memorandum of understanding in Copenhagen to develop and deploy a common cellular telephone system across Europe, and EU rules were passed to make GSM a mandatory standard. The decision to develop a continental standard eventually resulted in a unified, open, standard-based network which was larger than that in the United States.

In February 1987, Europe produced the very first agreed GSM Technical Specification. Ministers from the four big EU countries cemented their political support for GSM with the Bonn Declaration on Global Information Networks in May and the GSM MoU was tabled for signature in September. The MoU drew in mobile operators from across Europe to pledge to invest in new GSM networks to an ambitious common date.

In this short 38-week period, the whole of Europe (countries and industries) had been brought behind GSM in a rare unity and speed guided by four public officials: Armin Silberhorn (Germany), Stephen Temple (UK), Philippe Dupuis (France), and Renzo Failli (Italy). In 1989, the Groupe Spécial Mobile committee was transferred from CEPT to the European Telecommunications Standards Institute (ETSI).

In parallel, France and Germany signed a joint development agreement in 1984 and were joined by Italy and the UK in 1986. In 1986, the European Commission proposed reserving the 900 MHz spectrum band for GSM. The former Finnish prime minister Harri Holkeri made the world's first GSM call on July 1, 1991, calling Kaarina Suonio (mayor of the city of Tampere) using a network built by Telenokia and Siemens and operated by Radiolinja. In the following year, 1992, saw the sending of the first short messaging service (SMS or "text message") message, and Vodafone UK and Telecom Finland signed the first international roaming agreement.

Work began in 1991 to expand the GSM standard to the 1800 MHz frequency band and the first 1800 MHz network became operational in the UK by 1993, called and DCS 1800. Also that year, Telecom Australia became the first network operator to deploy a GSM network outside Europe and the first practical hand-held GSM mobile phone became available.

In 1995, fax, data and SMS messaging services were launched commercially, the first 1900 MHz GSM network became operational in the United States and GSM subscribers worldwide exceeded 10 million. In the same year, the GSM Association formed. Pre-paid GSM SIM cards were launched in 1996 and worldwide GSM subscribers passed 100

million in 1998.

In 2000, the first commercial GPRS services were launched and the first GPRS-compatible handsets became available for sale. In 2001, the first UMTS (W-CDMA) network was launched, a 3G technology that is not part of GSM. Worldwide GSM subscribers exceeded 500 million. In 2002, the first Multimedia Messaging Service (MMS) were introduced and the first GSM network in the 800 MHz frequency band became operational. EDGE services first became operational in a network in 2003, and the number of worldwide GSM subscribers exceeded 1 billion in 2004.

By 2005, GSM networks accounted for more than 75% of the worldwide cellular network market, serving 1.5 billion subscribers. In 2005, the first HSDPA-capable network also became operational. The first HSUPA network launched in 2007. (High-Speed Packet Access (HSPA) and its uplink and downlink versions are 3G technologies, not part of GSM.) Worldwide GSM subscribers exceeded three billion in 2008.

The GSM Association estimated in 2010 that technologies defined in the GSM standard served 80% of the mobile market, encompassing more than 5 billion people across more than 212 countries and territories, making GSM the most ubiquitous of the many standards for cellular networks.

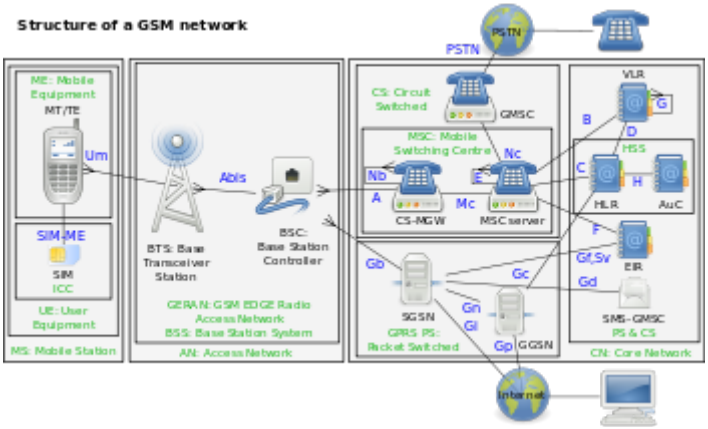
GSM is a second-generation (2G) standard employing time-division multiple-Access (TDMA) spectrum-sharing, issued by the European Telecommunications Standards Institute (ETSI). The GSM standard does not include the 3G Universal Mobile Telecommunications System (UMTS) code division multiple access (CDMA) technology nor the 4G LTE orthogonal frequency-division multiple access (OFDMA) technology standards issued by the 3GPP.[12]

GSM, for the first time, set a common standard for Europe for wireless networks. It was also adopted by many countries outside Europe. This allowed subscribers to use other GSM networks that have roaming agreements with each other. The common standard reduced research and development costs, since hardware and software could be sold with only minor adaptations for the local market.

Telstra in Australia shut down its 2G GSM network on December 1, 2016, the first mobile network operator to decommission a GSM network. The second mobile provider to shut down its GSM network (on January 1, 2017) was AT&T Mobility from the United States. Optus in Australia completed the shut down its 2G GSM network on August 1, 2017, part of the Optus GSM network covering Western Australia and the Northern Territory had earlier in the year been shut down in April 2017. Singapore shut down 2G services entirely in April 2017.

Technical details

The structure of a GSM network



Main article: GSM services

Network structure

The network is structured into a number of discrete sections:

Base station subsystem – the base stations and their controllers explained

Network and Switching Subsystem – the part of the network most similar to a fixed network, sometimes just called the "core network"

GPRS Core Network – the optional part which allows packet-based Internet connections

Operations support system (OSS) – network maintenance

Base station subsystem

GSM cell site antennas in the Deutsches Museum, Munich, Germany

GSM is a cellular network, which means that cell phones connect to it by searching for cells in the immediate vicinity. There are five different cell sizes in a GSM network—macro, micro, pico, femto, and (umbrella cells. The coverage area of each cell varies according to the implementation environment. Macro cells can be regarded as cells where the base station((antenna is installed on a mast or a building above average rooftop level. Micro cells are cells whose antenna height is under average rooftop level; they are typically used in urban areas. Picocells are small cells whose coverage diameter is a few dozen meters; they are mainly used indoors. Femtocells are cells designed for use in residential or small business environments and connect to the service provider's network via a broadband internet connection. Umbrella cells are used to cover shadowed regions of smaller cells and fill in gaps in coverage between those cells.

Cell horizontal radius varies depending on antenna height, antenna gain, and propagation conditions from a couple of hundred meters to several tens of kilometres. The longest distance the GSM specification supports in practical use is 35 kilometres (22 mi). There are also several implementations of the concept of an extended cell, where the cell radius could be double or even more, depending on the antenna system, the type of terrain, and the timing advance.

Indoor coverage is also supported by GSM and may be achieved by using an indoor picocell base station, or an indoor repeater with distributed indoor antennas fed through power splitters, to deliver the radio signals from an antenna outdoors to the separate indoor distributed antenna system. These are typically deployed when significant call capacity is needed indoors, like in shopping centers or airports. However, this is not a prerequisite, since indoor coverage is also provided by in-building penetration of the radio signals from any nearby cell.

GSM carrier frequencies

GSM networks operate in a number of different carrier frequency ranges (separated into GSM frequency ranges for 2G and UMTS frequency bands for 3G), with most 2G GSM networks operating in the 900 MHz or 1800 MHz bands. Where these bands were already allocated, the 850 MHz and 1900 MHz bands were used instead (for example in Canada and the United States). In rare cases the 400 and 450 MHz frequency bands are assigned in some countries because they were previously used for first-generation systems.

For comparison, most 3G networks in Europe operate in the 2100 MHz frequency band. For more information on worldwide GSM frequency usage, see GSM frequency bands.

Regardless of the frequency selected by an operator, it is divided into timeslots for individual phones. This allows eight full-rate or sixteen half-rate speech channels per radio frequency. These eight radio timeslots (or burst periods) are grouped into a TDMA frame. Half-rate channels use alternate frames in the same timeslot. The channel data rate for all 8 channels is 270.833 kbit/s, and the frame duration is 4.615 ms.

The transmission power in the handset is limited to a maximum of 2 watts in GSM 850/900 and 1 watt in GSM 1800/1900.

Voice codecs

GSM has used a variety of voice codecs to squeeze 3.1 kHz audio into between 6.5 and 13 kbit/s. Originally, two codecs, named after the types of data channel they were allocated, were used, called Half Rate (6.5 kbit/s) and Full Rate (13 kbit/s). These used a system based on linear predictive coding (LPC). In addition to being efficient with bitrates, these codecs also made it easier to identify more important parts of the audio, allowing the air interface layer to prioritize and better protect these parts of the signal. GSM was further enhanced in 1997 with the enhanced full rate (EFR) codec, a 12.2 kbit/s codec that uses a full-rate channel. Finally, with the development of UMTS, EFR was refactored into a variable-rate codec called AMR-Narrowband, which is high quality and robust against interference when used on full-rate channels, or less robust but still relatively high quality when used in good radio conditions on half-rate channel.

Subscriber Identity Module (SIM)

One of the key features of GSM is the Subscriber Identity Module, commonly known as

a **SIM card**. The SIM is a detachable smart card containing the user's subscription information and phone book. This allows the user to retain his or her information after switching handsets. Alternatively, the user can also change operators while retaining the handset simply by changing the SIM. Some operators will block this by allowing the phone to use only a single SIM, or only a SIM issued by them; this practice is known as SIM locking.

Phone locking

Sometimes mobile network operators restrict handsets that they sell for use with their own network. This is called *locking* and is implemented by a software feature of the phone. A subscriber may usually contact the provider to remove the lock for a fee, utilize private services to remove the lock, or use software and websites to unlock the handset themselves. It is possible to hack past a phone locked by a network operator.

In some countries (e.g., Bangladesh, Belgium, Brazil, Canada, Chile, Germany, Hong Kong, India, Iran, Lebanon, Malaysia, Nepal, Pakistan, Poland, Singapore, South Africa, Thailand) all phones are sold unlocked.

GSM security

GSM was intended to be a secure wireless system. It has considered the user authentication using a pre-shared key and challenge-response, and over-the-air encryption. However, GSM is vulnerable to different types of attack, each of them aimed at a different part of the network.

The development of UMTS introduced an optional Universal Subscriber Identity Module (USIM), that uses a longer authentication key to give greater security, as well as mutually authenticating the network and the user, whereas GSM only authenticates the user to the network (and not vice versa). The security model therefore offers confidentiality and authentication, but limited authorization capabilities, and no non-repudiation.

GSM uses several cryptographic algorithms for security. The A5/1, A5/2, and A5/3 stream ciphers are used for ensuring over-the-air voice privacy. A5/1 was developed first and is a stronger algorithm used within Europe and the United States; A5/2 is weaker and used in other countries. Serious weaknesses have been found in both algorithms: it is possible to break A5/2 in real-time with a ciphertext-only attack, and in January 2007, (The Hacker's Choice started the A5/1 cracking project with plans to use FPGAs that allow A5/1 to be broken with a rainbow table attack. The system supports multiple algorithms so operators may replace that cipher with a stronger one.

Since 2000, different efforts have been made in order to crack the A5 encryption algorithms. Both A5/1 and A5/2 algorithms have been broken, and their cryptanalysis has been revealed in the literature. As an example, (Karsten Nohl (de) developed a number of rainbow tables (static values which reduce the time needed to carry out an attack) and have found new sources for known plaintext attacks. He said that it is possible to build "a full GSM interceptor...from open-source components" but that they had not done so because of legal concerns.Nohl claimed that he was able to intercept voice and text conversations by impersonating another user to listen to voicemail, make

calls, or send text messages using a seven-year-old Motorola cellphone and decryption software available for free online.

GSM uses General Packet Radio Service (GPRS) for data transmissions like browsing the web. The most commonly deployed GPRS ciphers were publicly broken in 2011.

The researchers revealed flaws in the commonly used GEA/1 and GEA/2 ciphers and published the open-source "gprsdecode" software for sniffing GPRS networks. They also noted that some carriers do not encrypt the data (i.e., using GEA/0) in order to detect the use of traffic or protocols they do not like (e.g., Skype), leaving customers unprotected. GEA/3 seems to remain relatively hard to break and is said to be in use on some more modern networks. If used with USIM to prevent connections to (fake base stations and downgrade attacks, users will be protected in the medium term, though migration to 128-bit GEA/4 is still recommended.

Standards information

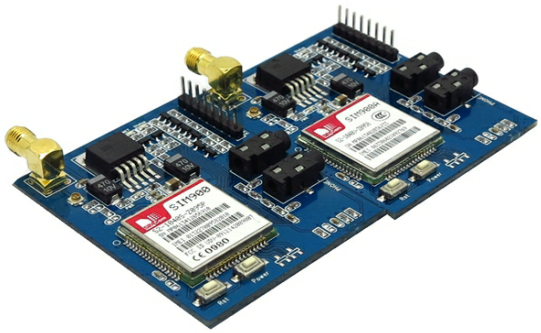
The GSM systems and services are described in a set of standards governed by ETSI, where a full list is maintained.

SIM900/SIM900A GSM/GPRS Minimum System Module

GPRS module is a breakout board and minimum system of SIM900 Quad-band/SIM900A Dual-band GSM/GPRS module. It can communicate with controllers via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands). This module supports software power on and reset.

Features

- ★ Quad-Band 850/ 900/ 1800/ 1900 MHz
- ★ Dual-Band 900/ 1900 MHz
- ★ GPRS multi-slot class 10/8GPRS mobile station class B
- ★ Compliant to GSM phase 2/2+Class 4 (2 W @850/ 900 MHz)
- ★ Class 1 (1 W @ 1800/1900MHz)
- ★ Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- ★ Low power consumption: 1.5mA(sleep mode)
- ★ Operation temperature: -40°C to +85 °C



Specifications

PCB size	71.4mm X 66.0mm X1.6mm
Indicators	PWR, status LED, net LED
Power supply	5V
Communication Protocol	UART
RoHS	Yes

Electrical Characteristics

Parameter	**Min.**	**Typical**	**Max.**	**Unit**
Power voltage (Vsupply)	4.5		5.5	VDC
Input voltage VH	0.7VCC		5.5	V
Input voltage VL	-0.3	0	0.3VCC	V
Current Consumption (pulse)	-		2000	mA
Current Consumption (continuous)			500	mA
Baud rate		115200		bps

Hardware

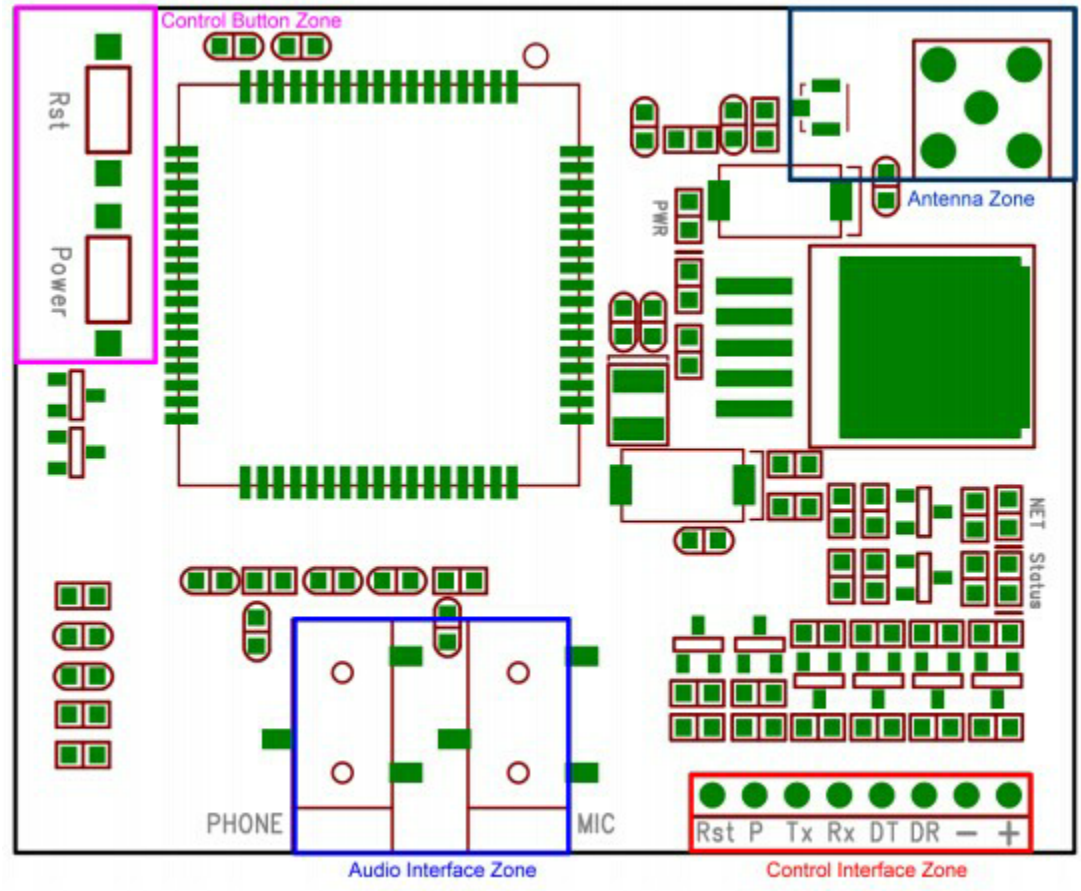


Figure 1 Top Map

Interface	Pin	Description
Rst	1	Reset the SIM900 module
P	2	Power switch pin of SIM900 module
Tx	3	UART data output
Rx	4	UART data in

DT	5	Debug UART data output
DR	6	Debug UART data input
		ND
+	8	VCC

Installation

Power on GPRS module

User can power on the GPRS module by pulling down the PWR button or the P pin of control interface for at least 1 second and release. This pin is already pulled up to 3V in the module internal, so external pull up is not necessary. When power on procedure is completed, GPRS module will send following URC to indicate that the module is ready to operate at fixed baud rate.

Indicator LED and Buttons:

NETSTATUS: The status of the NETSTATUS LED is listed in following table:

Status	Description
Off	SIM900 is not running 64ms On/800ms
Off	SIM900 not registered the network
64ms On/3000ms Off	SIM900 registered to the network
64ms On/300ms Off	GPRS communication is established

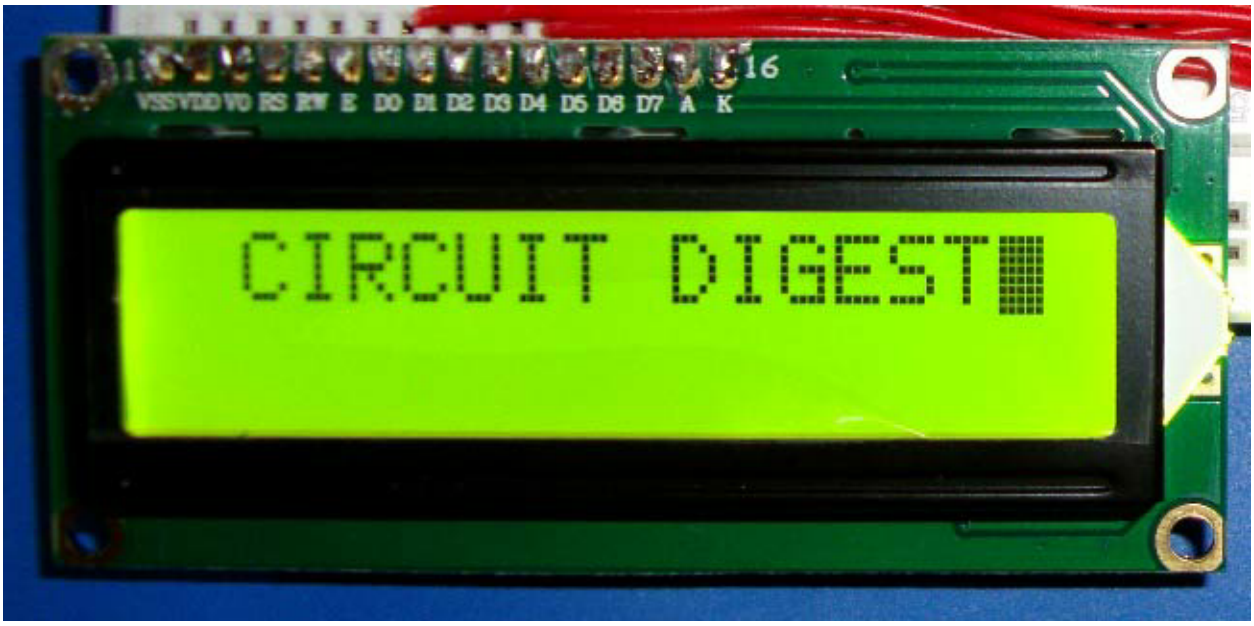
STATUS: Power status of SIM900.

PWR: Power status of GPRS module.

PWR: After the GPRS module power on, you need to press the POWER button for a moment to power on the SIM900 module.

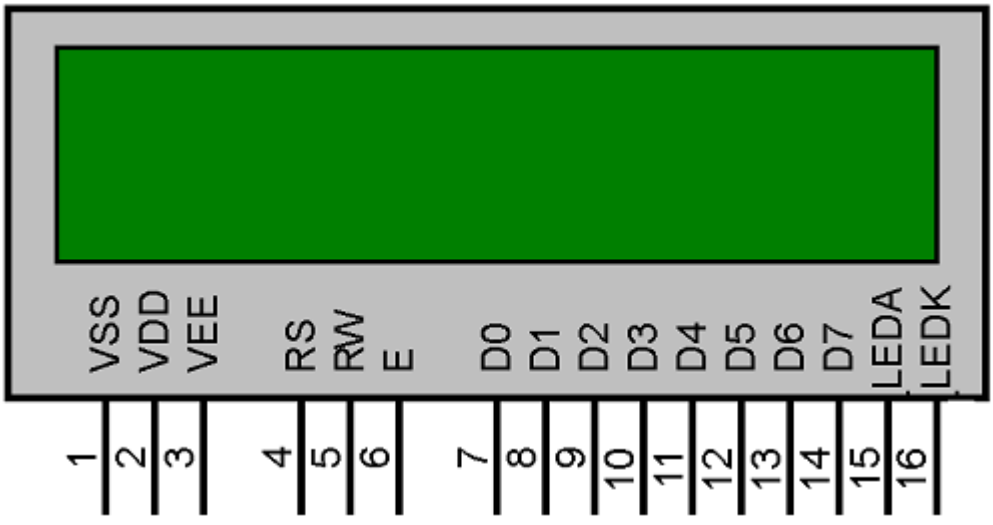
RESET: Reset the SIM900 module.

8.LCD DISPLAY (16×2)



16×2 LCD is named so because; it has 16 Columns and 2 Rows. There are a lot of combinations available like, 8×1, 8×2, 10×2, 16×1, etc. But the most used one is the 16*2 LCD, hence we are using it here.

All the above mentioned LCD display will have 16 Pins and the programming approach is also the same and hence the choice is left to you. Below is the **Pinout and Pin Description of 16x2 LCD Module**:



Sr. No	Pin No.	Pin Name	Pin Type	Pin Description	Pin Connection
1	Pin 1	Ground	Source Pin	This is a ground pin of LCD	Connected to the ground of the MCU/ Power source
2	Pin 2	VCC	Source Pin	This is the supply voltage pin of LCD	Connected to the supply pin of Power source
3	Pin 3	V0/VEE	Control Pin	Adjusts the contrast of the LCD.	Connected to a variable POT that can source 0-5V
4	Pin 4	Register Select	Control Pin	Toggles between Command/Data Register	Connected to a MCU pin and gets either 0 or 1. 0 -> Command Mode 1-> Data Mode
5	Pin 5	Read/Write	Control Pin	Toggles the LCD between Read/Write Operation	Connected to a MCU pin and gets either 0 or 1. 0 -> Write Operation 1-> Read Operation
6	Pin 6	Enable	Control Pin	Must be held high to perform Read/Write Operation	Connected to MCU and always held high.
7	Pin 7 -14	Data Bits (0-7)	Data/Command Pin	Pins used to send Command or data to the LCD.	In 4-Wire Mode Only 4 pins (0-3) is connected to MCU In 8-Wire Mode

					All 8 pins(0-7) are connected to MCU
8	Pin 15	LED Positive	LED Pin	Normal LED like operation to illuminate the LCD	Connected to +5V
9	Pin 16	LED Negative	LED Pin	Normal LED like operation to illuminate the LCD connected with GND.	Connected to ground

It is okay if you do not understand the function of all the pins, I will be explaining in detail below. Now, let us turn back our LCD:

Okay, what is this two black circle like things on the back of our LCD?

These black circles consist of an interface IC and its associated components to help us use this LCD with the MCU. Because our LCD is a 16*2 Dot matrix LCD and so it will have (16*2=32) 32 characters in total and each character will be made of 5*8 Pixel Dots. A Single character with all its Pixels enabled is shown in the below picture.

So Now, we know that each character has (5*8=40) 40 Pixels and for 32 Characters we will have (32*40) 1280 Pixels. Further, the LCD should also be instructed about the Position of the Pixels.

It will be a hectic task to handle everything with the help of MCU, hence an **Interface IC like HD44780** is used, which is mounted on LCD Module itself. The function of this IC is to get the **Commands and Data** from the MCU and process them to display meaningful information onto our LCD Screen.

Let's discuss the different type of mode and options available in our LCD that has to be controlled by our Control Pins.

4-bit and 8-bit Mode of LCD:

The LCD can work in two different modes, namely the 4-bit mode and the 8-bit mode. In **4 bit mode** we send the data nibble by nibble, first upper nibble and then lower nibble. For those of you who don't know what a nibble is: a nibble is a group of four bits, so the lower four bits (D0-D3) of a byte form the lower nibble while the upper four bits (D4-D7) of a byte form the higher nibble. This enables us to send 8 bit data.

Whereas **in 8 bit mode** we can send the 8-bit data directly in one stroke since we use all

the 8 data lines.

Now you must have guessed it, Yes 8-bit mode is faster and flawless than 4-bit mode. But the major drawback is that it needs 8 data lines connected to the microcontroller. This will make us run out of I/O pins on our MCU, so 4-bit mode is widely used. No control pins are used to set these modes. It's just the way of programming that change.

Read and Write Mode of LCD:

As said, the LCD itself consists of an Interface IC. The MCU can either read or write to this interface IC. Most of the times we will be just writing to the IC, since reading will make it more complex and such scenarios are very rare. Information like position of cursor, status completion interrupts etc. can be read if required, but it is out of the scope of this tutorial.

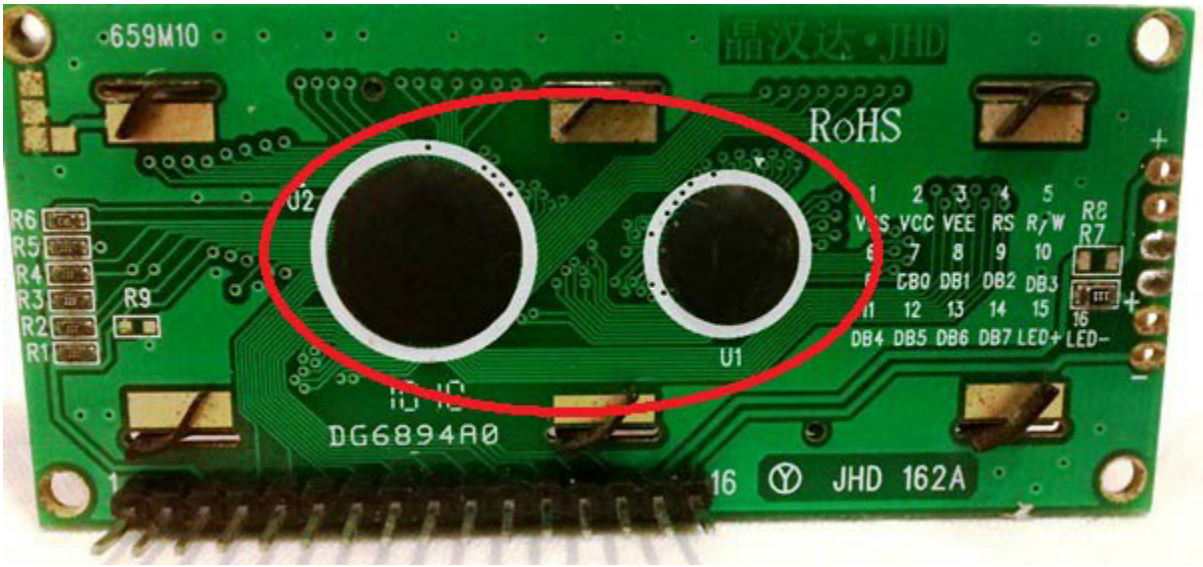
The Interface IC present in most of the LCD is **HD44780U,** in order to program our LCD we should learn the complete datasheet of the IC. The datasheet is given here.

LCD Commands:

There are some preset commands instructions in LCD, which we need to send to LCD through some microcontroller. Some important command instructions are given below:

Hex Code	Command to LCD Instruction Register
0F	LCD ON, cursor ON
01	Clear display screen
02	Return home
04	Decrement cursor (shift cursor to left)
06	Increment cursor (shift cursor to right)
05	Shift display right
07	Shift display left
0E	Display ON, cursor blinking
80	Force cursor to beginning of first line

C0	Force cursor to beginning of second line
38	2 lines and 5×7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, cursor OFF
C1	Jump to second line, position 1
OC	Display ON, cursor OFF
C1	Jump to second line, position 1
C2	Jump to second line, position 2

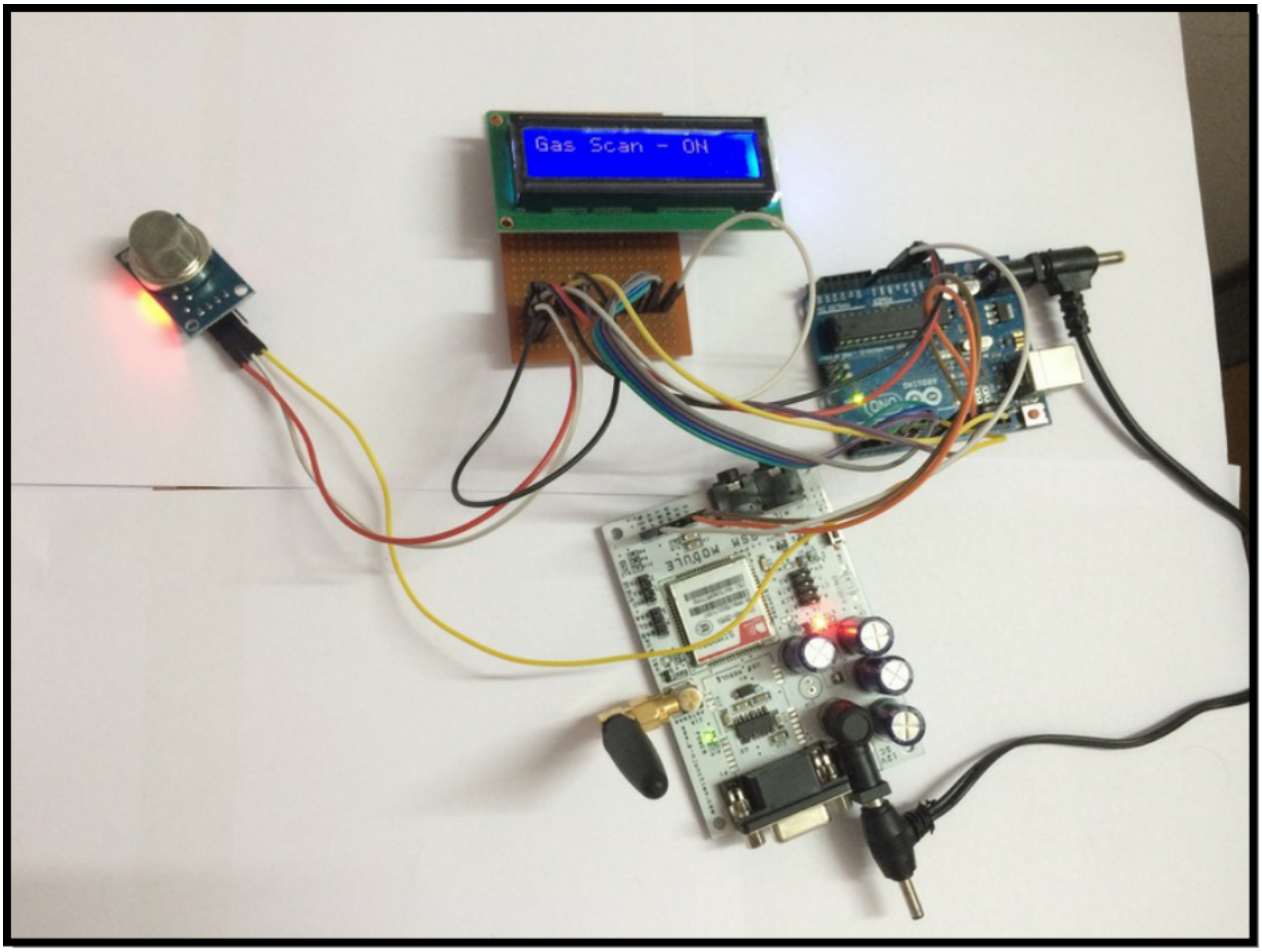


9. REFERENCES

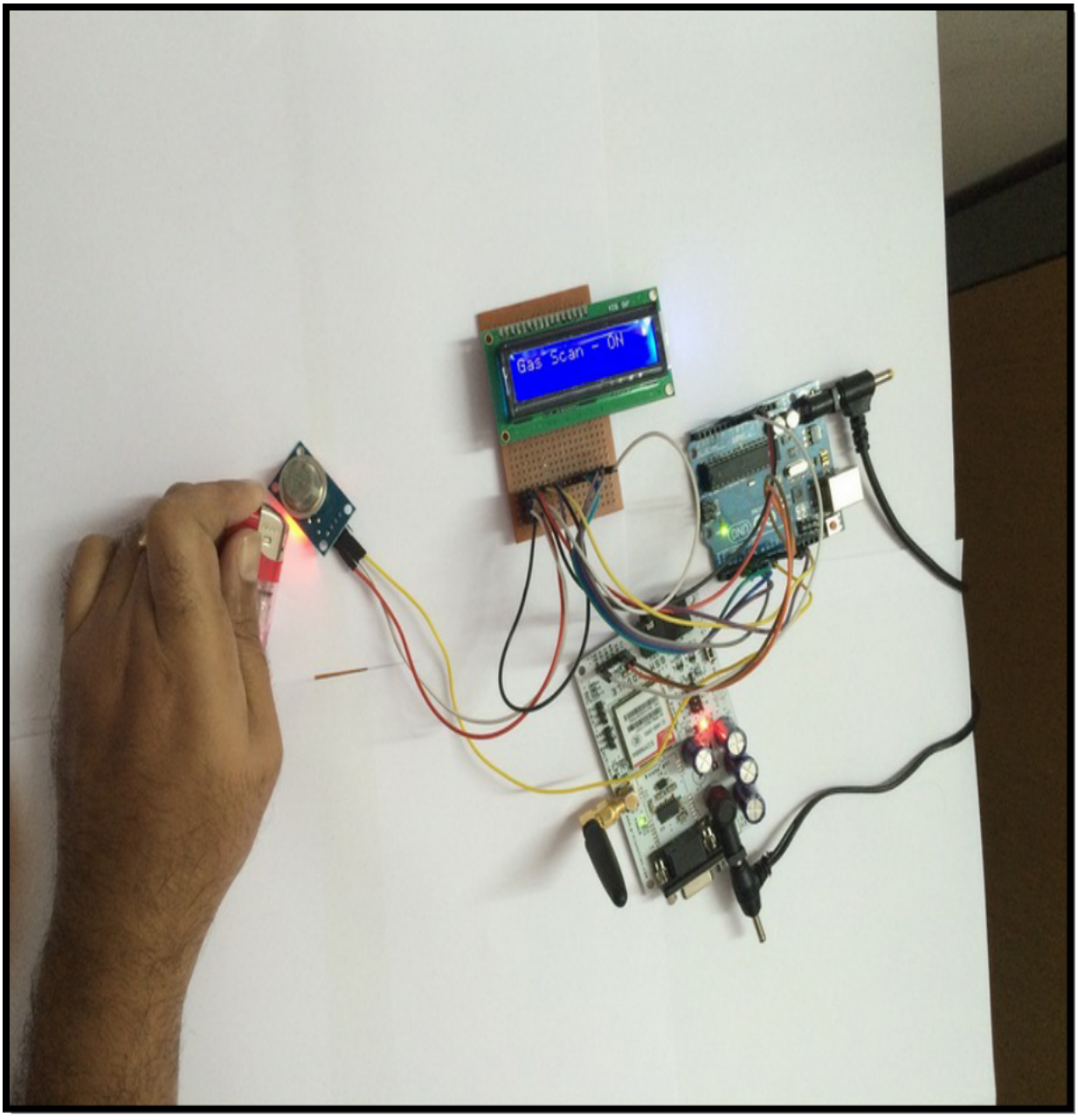
Circuit and connection diagram: .*http://www.circuitstoday.com/gas-leakage-detector-using-arduino-with-sms-alert*

10.RESULT

1.When the circuit is ready scan the Gas.



2. When the gas is leaking.



3. While sending SMS Alert to pre determined phone number.