GAS LEAKAGE DETECTION AND SMART ALERTING SYSTEM USING IOT

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
```

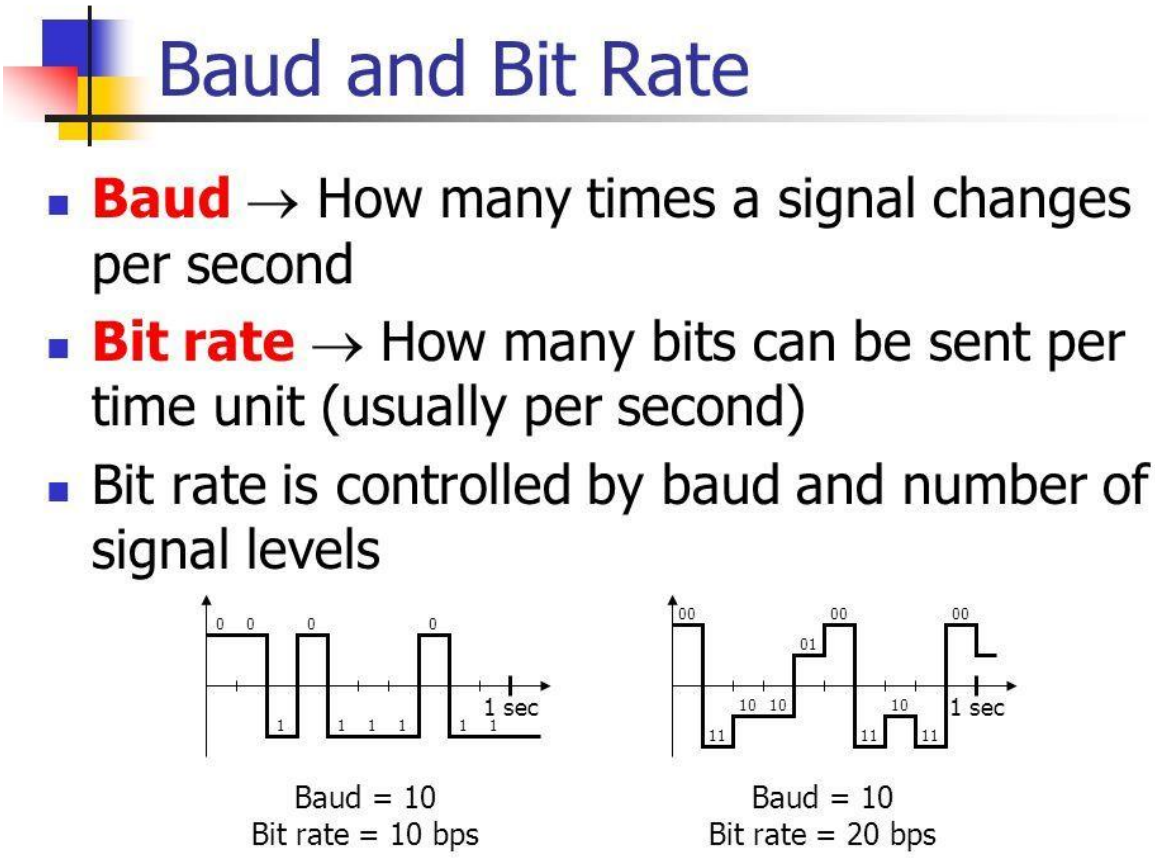
```
  delay(5000);

  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);
}
```

Baud Rate



Arduino

Arduino UNO is a microcontroller board based on the ATmega328P. 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. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

ATmega328P

The classic high-performance, low-power AVR® microcontroller.

Replaceable chip

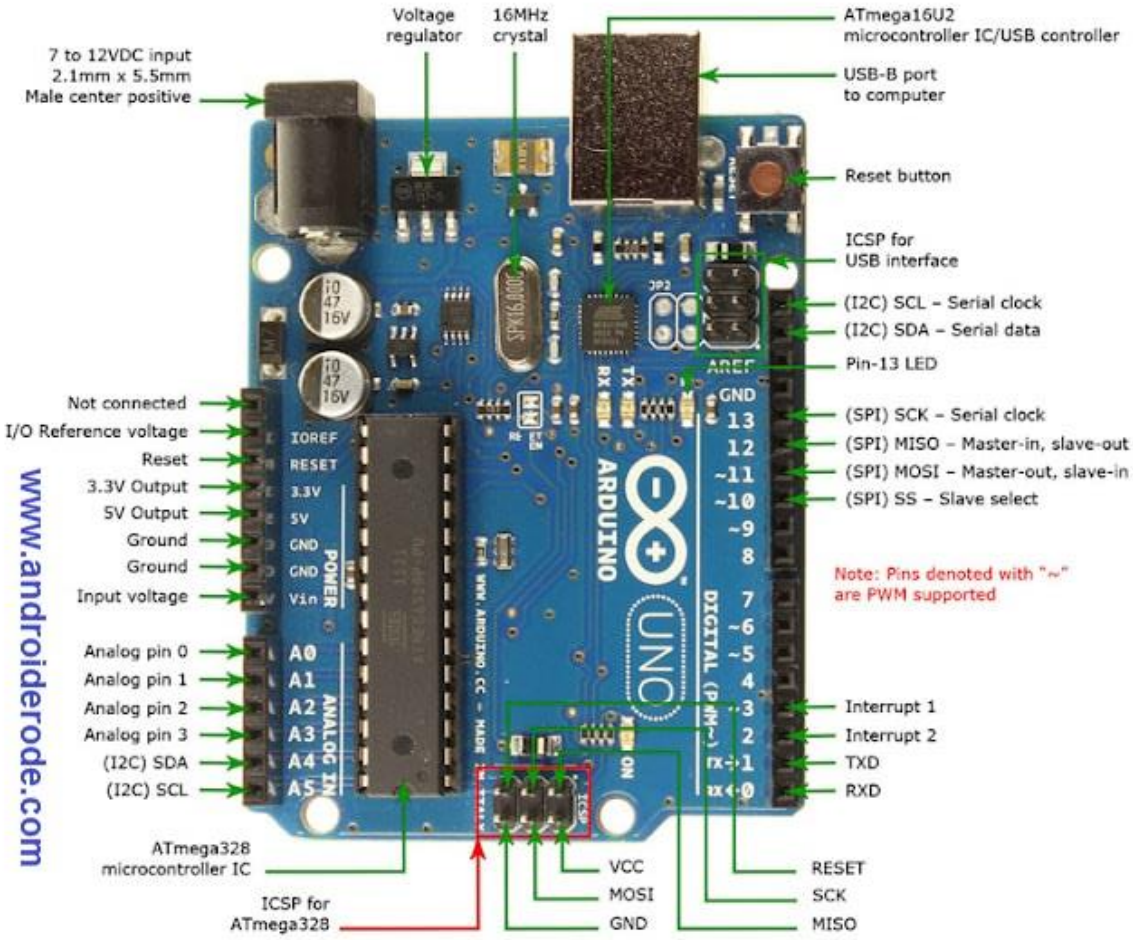
The ATmega328P can easily be replaced, as it is not soldered to the board.

EEPROM

The ATmega328P also features 1kb of EEPROM, a memory which is not erased when powered off.

Battery Connector

The Arduino UNO features a barrel plug connector, that works great with a standard 9V battery.



Analog pin can be used only for reading, while Digital pin can be used for both reading and writing.

An analog signal is one that can take on any number of values, unlike a digital signal which has only two values: HIGH and LOW. To measure the value of analog signals, the Arduino has a built-in analog-to-digital converter (ADC). The ADC turns the analog voltage into a digital value. The function that you use to obtain the value of an analog signal is analogRead(pin). This function converts the value of the voltage on an analog input pin and returns a digital value from 0 to 1023, relative to the reference value. The default reference voltage is 5 V (for 5 V Arduino boards) or 3.3 V (for 3.3 V Arduino boards). It has one parameter which is the pin number.

The Arduino does not have a built-in digital-to-analog converter (DAC), but it can pulse-width modulate (PWM) a digital signal to achieve some of the functions of an analog output. The function used to output a PWM signal is analogWrite(pin, value). pin is the pin number used for the PWM output. value is a number proportional to the duty cycle of the signal. When value = 0, the signal is always off. When value = 255, the signal is always on. On most Arduino boards, the PWM function is available on pins 3, 5, 6, 9, 10, and 11

GPS

NEO-6M GPS Module

What is GPS

The Global Positioning System (GPS) is a satellite-based navigation system made up of at least 24 satellites. GPS works in any weather conditions, anywhere in the world, 24 hours a day, with no subscription fees or setup charges.

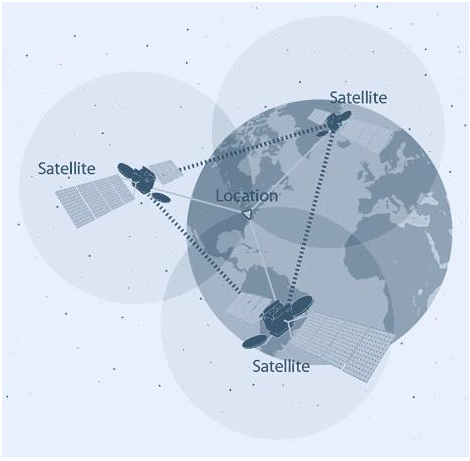
How GPS works

GPS satellites circle the Earth twice a day in a precise orbit. Each satellite transmits a unique signal and orbital parameters that allow GPS devices to decode and compute the precise location of the satellite. GPS receivers use this information and trilateration to calculate a user's exact location. Essentially, the GPS receiver measures the distance to each satellite by the amount of time it takes to receive a transmitted signal. With distance measurements from a few more satellites, the receiver can determine a user's position and display it.

To calculate your 2-D position (latitude and longitude) and track movement, a GPS receiver must be locked on to the signal of at least 3 satellites. With 4 or more satellites in view, the receiver can determine your 3-D position (latitude, longitude and altitude). Generally, a GPS receiver will track 8 or more satellites, but that depends on the time of day and where you are on the earth.

GPS receivers actually work by figuring out how far they are from a number of satellites. They are pre-programmed to know where the GPS satellites are at any given time.

The satellites transmit information about their position and the current time in the form of radio signals towards the Earth. These signals identify the satellites and tell the receiver where they are located.



The receiver then calculates how far away each satellite is by figuring out how long it took for the signals to arrive. Once it has information on how far away at least three satellites are and where they are in space, it can pinpoint your location on Earth.

This process is known as Trilateration.

What's the signal?

GPS satellites transmit at least 2 low-power radio signals. The signals travel by line of sight, meaning they will pass through clouds, glass and plastic but will not go through most solid objects, such as buildings and mountains. However, modern receivers are more sensitive and can usually track through houses.

A GPS signal contains 3 different types of information:

Pseudorandom code is an I.D. code that identifies which satellite is transmitting information. You can see which satellites you are getting signals from on your device's satellite page.

Ephemeris data is needed to determine a satellite's position and gives important information about the health of a satellite, current date and time.

Almanac data tells the GPS receiver where each GPS satellite should be at any time throughout the day and shows the orbital information for that satellite and every other satellite in the system.

2. Download and install required libraries for GPS to work in Arduino IDE

(i) SoftwareSerial library

(ii) TinyGPS library

3. NEO-6M GPS module

The NEO-6M GPS module is shown in the figure below. It comes with an external antenna and does not come with header pins. So you will need to solder it.

The heart of the module is a NEO-6M GPS chip from u-blox. It can track up to 22 satellites on 50 channels and achieves the industry's highest level of sensitivity i.e. -161 dB tracking, while consuming only 45mA supply current. The u-blox 6 positioning engine also boasts a Time-To-First-Fix (TTFF) of under 1 second. One of the best features the chip provides is Power Save Mode(PSM). It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically reduces power consumption of the module to just 11mA making it suitable for

power sensitive applications like GPS wristwatch. The necessary data pins of NEO-6M GPS chip are broken out to a "0.1" pitch headers. This includes pins required for communication with a microcontroller over UART.

Note:- The module supports baud rate from 4800bps to 230400bps with default baud of 9600.

Position Fix LED Indicator

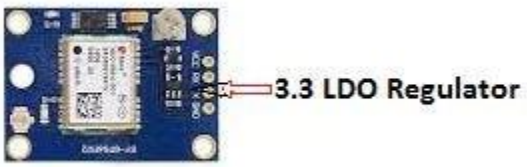
There is an LED on the NEO-6M GPS Module which indicates the status of Position Fix. It'll blink at various rates depending on what state it's in

No Blinking ==> means It is searching for satellites

Blink every 1s – means Position Fix is found

3.3V LDO Regulator

The operating voltage of the NEO-6M chip is from 2.7 to 3.6V. But, the module comes with MIC5205 ultra-low dropout 3V3 regulator from MICREL. The logic pins are also 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using any logic level converter.



Battery & EEPROM

The module is equipped with an HK24C32 two wire serial EEPROM. It is 4KB in size and connected to the NEO-6M chip via I2C.The module also contains a rechargeable button battery which acts as a super-capacitor.

An EEPROM together with battery helps retain the battery backed RAM (BBR). The BBR contains clock data, latest position data (GNSS or bit data) and module configuration. But it is not meant for permanent data storage.

As the battery retains clock and last position, time to first fix (TTFF) significantly reduces to 1s. This allows much faster position locks.

Without the battery the GPS always cold-start so the initial GPS lock takes more time. The battery is automatically charged when power is applied and maintains data for up to two weeks without power.

Pinout

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

TxD (Transmitter) pin is used for serial communication.

RxD (Receiver) pin is used for serial communication.

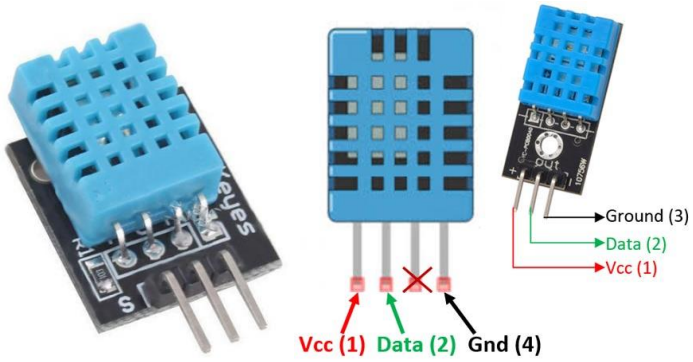
VCC supplies power for the module. You can directly connect it to the 5V pin on the Arduino.

S No.	Arduino	Raspberry Pi
1.	Control unit of Arduino is from Atmega family.	While control unit of Raspberry Pi is from ARM family.
2.	Arduino is based on a microcontroller.	While Raspberry Pi is based on a microprocessor.
3.	It is designed to control the electrical components connected to the circuit board in a system.	While Raspberry Pi computes data and produces valuable outputs, and controls components in a system based on the outcome of its computation.
4.	Arduino boards have a simple hardware and software structure.	While Raspberry Pi boards have a complex architecture of hardware and software.
5.	CPU architecture: 8 bit.	CPU architecture: 64 bit.
6.	It uses very less RAM, 2 kB.	While Raspberry Pi requires more RAM, 1 GB.
7.	It clocks a processing speed of 16 MHz.	While Raspberry Pi clocks a processing speed of 1.4 GHz.
8.	It is cheaper in cost.	While Raspberry Pi is expensive.
9.	It has a higher I/O current drive strength.	While Raspberry Pi has a lower I/O current drive strength.
10.	It consumes about 200 MW of power.	While it consumes about 700 MW of power.

DHT11

The DHT11 is a commonly used Temperature and humidity sensor that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

NTC stands for "Negative Temperature Coefficient". NTC thermistors are resistors with a negative temperature coefficient, which means that the resistance decreases with increasing temperature.



For DHT11 Sensor module		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

DHT11 Specifications

Operating Voltage: 3.5V to 5.5V

Operating current: 0.3mA (measuring) 60uA (standby)

Output: Serial data

Temperature Range: 0°C to 50°C

Humidity Range: 20% to 90%

Resolution: Temperature and Humidity both are 16-bit

Accuracy: ±1°C and ±1%

MQ2 Gas Sensor

MQ2 is one of the commonly used gas sensors in MQ sensor series. It is a Metal Oxide Semiconductor (MOS) type Gas Sensor also known as Chemiresistors as the detection is based upon change of resistance of the sensing material when the Gas comes in contact with the material. Using a simple voltage divider network, concentrations of gas can be detected.

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.

Here are the complete specifications

Operating voltage	5V
Load resistance	20 KΩ
Heater resistance	33Ω ± 5%
Heating consumption	<800mw
Sensing Resistance	10 KΩ – 60 KΩ
Concentration Scope	200 – 10000ppm
Preheat Time	Over 24 hour

What is 1 ppm equal to?

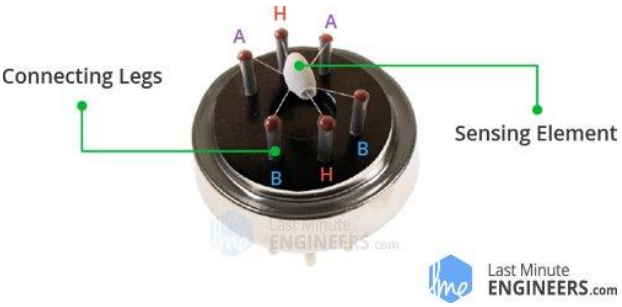
When measuring gases like carbon dioxide, oxygen, or methane, the term concentration is used to describe the amount of gas by volume in the air. The 2 most common units of measurement are parts-per-million, and percent concentration.

Parts-per-million (abbreviated ppm) is the ratio of one gas to another. For example, 1,000ppm of CO means that if you could count a million gas molecules, 1,000 of them would be of carbon monoxide and 999,000 molecules would be some other gases.

Internal structure of MQ2 Gas Sensor

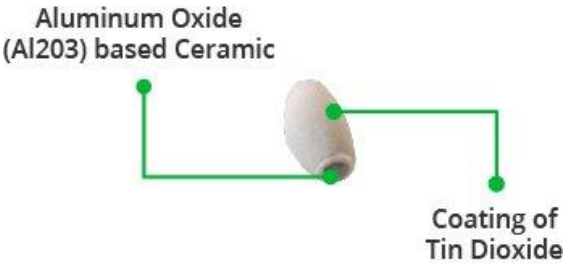
The sensor is actually enclosed in two layers of fine stainless steel mesh called Anti-explosion network. It ensures that heater element inside the sensor will not cause an explosion, as we are sensing flammable gases.

It also provides protection for the sensor and filters out suspended particles so that only gaseous elements are able to pass inside the chamber. The mesh is bound to rest of the body via a copper plated clamping ring.

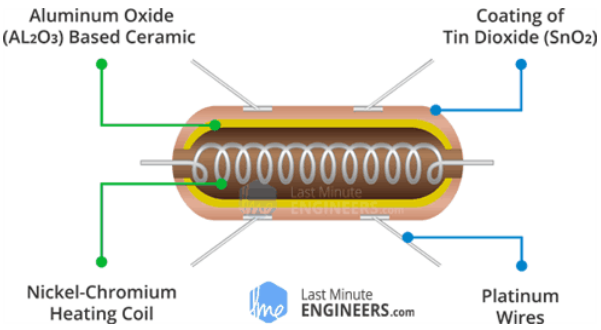


This is how the sensor looks like when outer mesh is removed. The star-shaped structure is formed by the sensing element and six connecting legs that extend beyond the Bakelite base. Out of six, two leads (H) are responsible for heating the sensing element and are connected through Nickel-Chromium coil, well known conductive alloy.

The remaining four leads (A & B) responsible for output signals are connected using Platinum Wires. These wires are connected to the body of the sensing element and convey small changes in the current that passes through the sensing element.



The tubular sensing element is made up of Aluminum Oxide (AL2O3) based ceramic and has a coating of Tin Dioxide (SnO2). The Tin Dioxide is the most important material being sensitive towards combustible gases. However, the ceramic substrate merely increases heating efficiency and ensures the sensor area is heated to a working temperature constantly.



So, the Nickel-Chromium coil and Aluminum Oxide based ceramic forms a Heating System; while Platinum wires and coating of Tin Dioxide forms a Sensing System.

How does a gas sensor work?

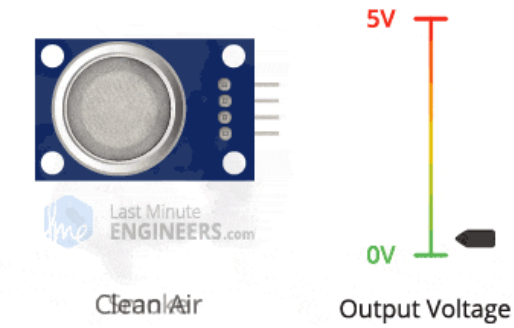
When tin dioxide (semiconductor particles) is heated in air at high temperature, oxygen is adsorbed on the surface. In clean air, donor electrons in tin dioxide are attracted toward oxygen which is adsorbed on the surface of the sensing material. This prevents electric current flow.

In the presence of reducing gases, the surface density of adsorbed oxygen decreases as it reacts with the reducing gases. Electrons are then released into the tin dioxide, allowing current to flow freely through the sensor.

Hardware Overview – MQ2 Gas Sensor Module

Since MQ2 Gas Sensor is not breadboard compatible, we do recommend this handy little breakout board. It's very easy to use and comes with two different outputs. It not only provides a binary indication of the presence of combustible gases but also an analog representation of their concentration in air.

The analog output voltage provided by the sensor changes in proportional to the concentration of smoke/gas. The greater the gas concentration, the higher is the output voltage; while lesser gas concentration results in low output voltage. The following animation illustrates the relationship between gas concentration and output voltage.

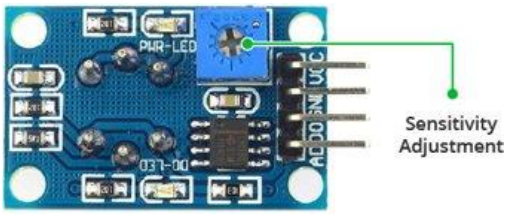


The analog signal from MQ2 Gas sensor is further fed to LM393 High Precision Comparator (soldered on the bottom of the module), of course to digitize the signal. Along with the comparator is a little potentiometer you can turn to adjust the sensitivity of the sensor. You can use it to adjust the concentration of gas at which the sensor detects it.

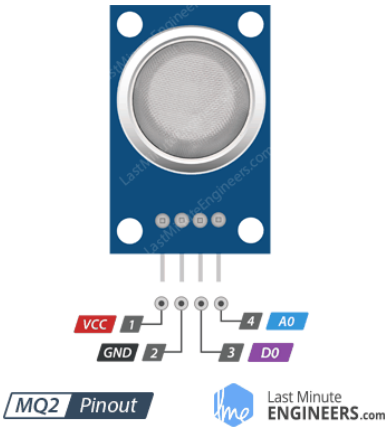
The sensor is sensitive to multiple gasses – but cannot tell which it is! That's normal; most gas sensors are like that. So, it is best for measuring changes in a known gas density, not detecting which is changing.

Calibrate MQ2 Gas Sensor Module

To calibrate the gas sensor you can hold the gas sensor near smoke/gas you want to detect and keep turning the potentiometer until the Red LED on the module starts glowing. Turn the screw clockwise to increase sensitivity or anticlockwise to decrease sensitivity.



The comparator on the module continuously checks if the analog pin (A0) has hit the threshold value set by potentiometer. When it crosses the threshold, the digital pin (D0) will go HIGH and signal LED turns on. This setup is very useful when you need to trigger an action when certain threshold is reached. For example, when the smoke crosses a threshold, you can turn on or off a relay or instruct your robot to blow air/sprinkle water. You got the idea!



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

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

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

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

Link

https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/

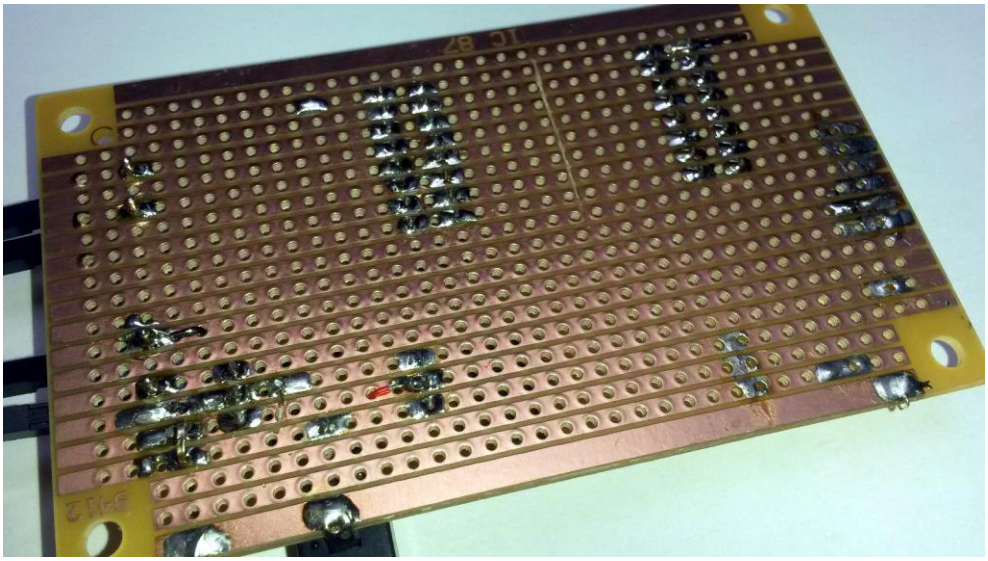
Jumper wires

Jumper wires are used for making connections between items on your breadboard and your Arduino's header pins.

Material	Aluminum

StripBoard

Stripboard is the generic name for a widely used type of electronics prototyping board characterized by a 0.1 inches (2.54 mm) regular (rectangular) grid of holes, with wide parallel strips of copper cladding running in one direction all the way across one side of the board.



AT Commands

AT in the command is the short form of an Attention. The command line used in every modem starts with 'AT' otherwise 'at' so these commands are named as AT commands.

These are supported by modems like mobile phones, GSM or GPRS. There are some commands which support GSM. These commands which are used for GSM mainly include SMS based commands like AT+CMGS, AT+CMSS, AT+CMGL & AT+CMGR. Here the prefix AT in these commands informs the modem regarding the begin of a command line.

These commands are used in GSM, GPRS, or mobile phone MODEMs can be used to access the information as well as services which include the following.

The information & configuration related to phone otherwise SIM card & MODEM.

The services like SMS, MMS and Fax services.

Voice and data link on a mobile network.

Types of AT Commands

These commands are classified into four types namely Test, Read, Set and Execution.

Test Command

The test AT command is mainly used for checking the command's compatibility using a modem. The SYNTAX for this command is AT < name of the command>. The best example of this command is AD =?

Read Command

The Read command is mainly used for changing the settings of mobile phone otherwise modem required for operations. The SYNTAX for this command is AT < name of the command>. The best example of this command is AT+CBC =?

Set Command

The Set command is mainly used for making modifications in the settings of mobile phone otherwise modem required for operations. The SYNTAX for this command is AT < name of the command> = Value 1, Value 2….Value N. The best example of this is AT+CBC ="+923140", 110

Execution Command

The Execution command is mainly used for executing the said operation. The SYNTAX for this command is AT < name of the command> = parameter-1, parameter-2… parameter-N. The best example of this is AT+CBC = 2,"+ 4867512120", 210.

Commands

AT command returns OK which implies that the communication between the device and the application has been verified.

AT+CPIN AT command sets the password of the mobile device.

AT+CREG AT command gives information about the registration status and access technology of the serving cell.

Values from 0 to 7

AT+CGATT AT command is used to attach or detach the device to packet domain service

AT+CIPSHUT will close the GPRS PDP(General Packet Radio Service packet data protocol) context.

AT+CIPMUX AT command configures the device for a single or multi IP connection.

AT+CIPMUX=? - Test Command

AT+CIPMUX? - Get the current setting

AT+CIPMUX= - Value can be '0' (single IP) or '1' (multi IP)

AT+CIPSTATUS AT comamnd returns the current connection status. This command returns the applicable server status, client status, conenction number (for multi-ip) and GPRS bearer info.

AT+CSTT AT command sets up the APN, user name and password for the PDP context.

An Access Point Name(APN) is the name of a gateway between a GSM, GPRS, 3G and 4G mobile network and another computer network, frequently the public Internet.

AT+CIICR command brings up the GPRS or CSD call depending on the configuration previously set by the AT+CSTT command.

AT+CIFSR command returns the local IP address. It is imperative the the PDP context must have been activated before to get the IP address.

AT+CIPSPRT is used to set whether echo prompt ">" after issuing "AT+CIPSEND" command.

AT+CIPSTART comamnds starts a TCP or UDP connection.

Parameters
- 0..7 - Connection number
- "TCP" or "UDP"
- Remote server IP address
- Remote server port
- remote domain name

AT+CIPSEND AT command is used to send the data over the TCP or UDP connection.

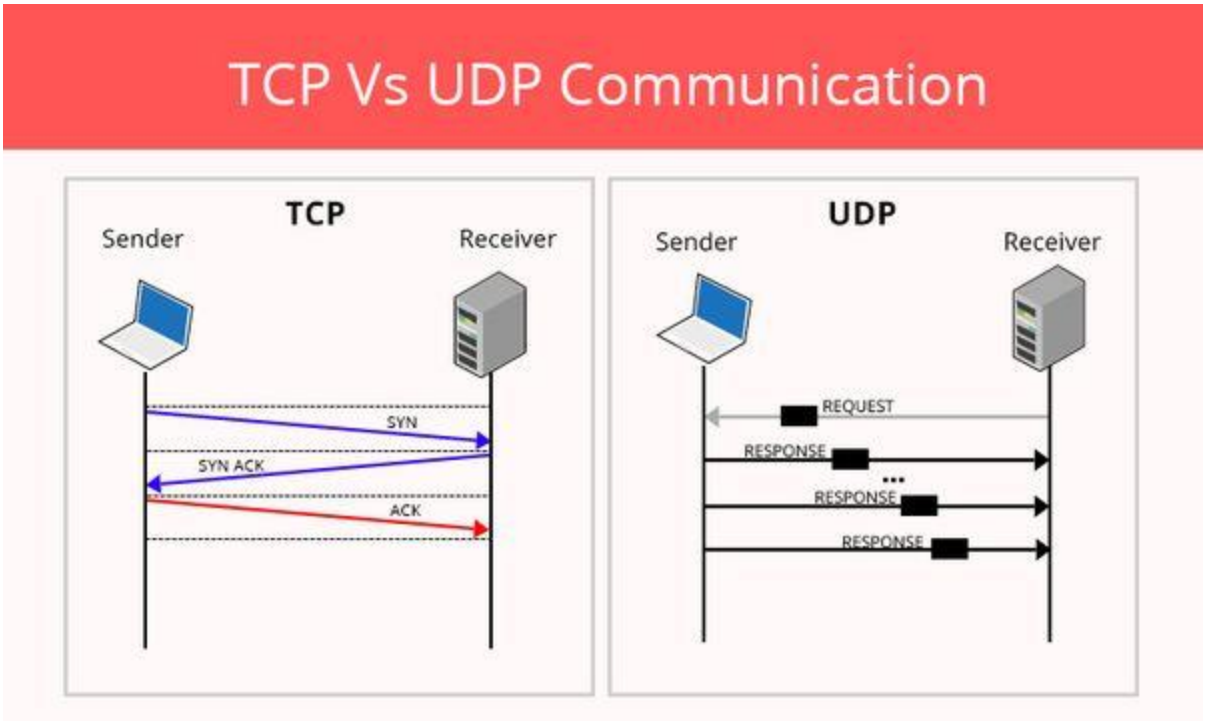
AT+CIPSHUT will close the GPRS PDP context.

What is TCP?

TCP stands for Transmission Control Protocol a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks.

TCP is one of the basic standards that define the rules of the internet and is included within the standards defined by the Internet Engineering Task Force (IETF). It is one of the most commonly used protocols within digital network communications and ensures end-to-end data delivery.

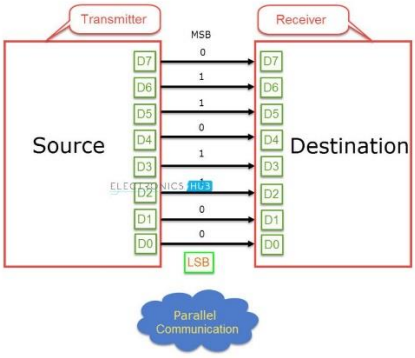
TCP organizes data so that it can be transmitted between a server and a client. It guarantees the integrity of the data being communicated over a network. Before it transmits data, TCP establishes a connection between a source and its destination, which it ensures remains live until communication begins. It then breaks large amounts of data into smaller packets, while ensuring data integrity is in place throughout the process.



UART

Transfer of Digital Data from one device to another can be achieved in two ways: Parallel Data Transfer and Serial Data Transfer. In parallel data transfer, all the bits are transferred from the source to destination at once.

This is possible because parallel data transfer uses multiple lanes or wires between the transmitter and receiver in order to transfer the data.

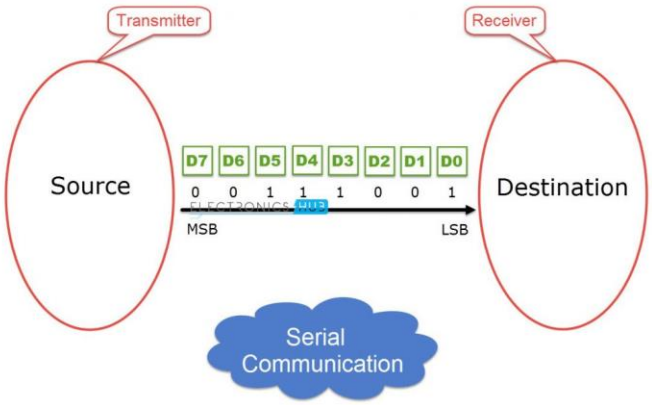


Parallel Data Transfer methods are faster and expensive as they needs more hardware and a lot of wires. Olden day's printers are the best example for external parallel communication. Other examples are RAM, PCI, etc.

With the progress in integrated circuit technology, the digital IC's are becoming smaller and faster and as a result the transfer rates in Parallel Communication with multiple lanes have reached a bottle neck.

Serial Communication on the hand, transfers data bit by bit using a single line or wire. For two way communication between the transmitter and receiver, we need just two wires for successful serial data transfer.

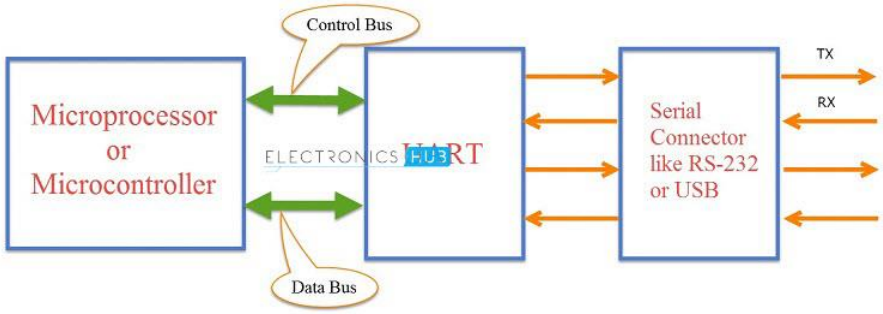
Since serial communication needs less circuitry and wires, the cost of implementing is less. As a result, using serial communication in complex circuitry might be more practical than parallel communication.



But the only concern with serial data transfers is speed. Since the data transfer occurs over a single line, the speed of transfer in serial communication is less than that of parallel communication. Now – a – days, the speed of serial data transfer isn't a concern as advancements in technology have led to faster transfer speeds.

UART or Universal Asynchronous Receiver Transmitter is a serial communication device that performs parallel – to – serial data conversion at the transmitter side and serial – to – parallel data conversion at the receiver side. It is universal because the parameters like transfer speed, data speed, etc. are configurable.

As mentioned in the introduction section, UART is a piece of hardware that acts as a bridge between the processor and the serial communication protocol or port. The following image shows this interface briefly. The serial communication can be anything like USB, RS – 232, etc.



The letter 'A' in UART stands for Asynchronous i.e. there is no clock signal to synchronize or validate the data transmitted from transmitter and received by the receiver (Asynchronous Serial Communication).

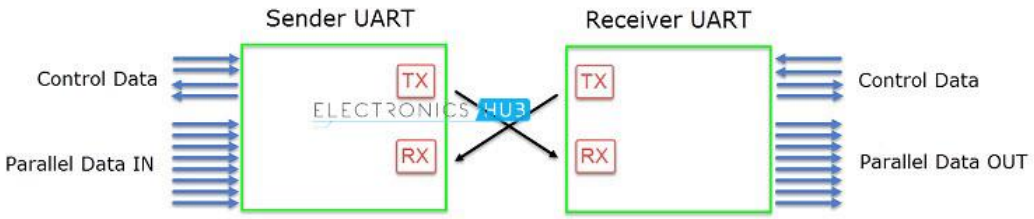
This is in contrast to Synchronous Serial Communication, which uses a clock signal that is shared between the transmitter and receiver in order to "Synchronize" the data between them.

If there is no clock (or any other timing signal) between the transmitter and receiver, then how does the receiver know when to read the data?

In UART, the transmitter and receiver must agree on timing parameters beforehand. Also, UART uses special bits at the beginning and ending of each data word to synchronize the transmitter and receiver. More about these special bits in the later sections.

In UART based Serial Communication, the transmitter and receiver communicate in the following manner. The UART on the sender device i.e. the transmitting UART receives parallel data from the CPU (microprocessor or microcontroller) and converts it in to serial data.

This serial data is transmitted to the UART on the receiver device i.e. receiving UART. The receiving UART, upon receiving the serial data, converts it back to parallel data and gives it to the CPU.



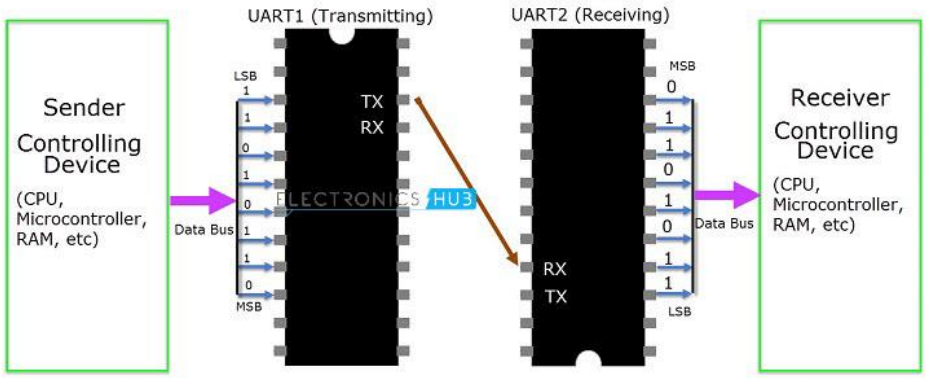
The pin on the transmitting UART, which transmits the serial data is called TX and the pin on the receiving UART, which receives the serial data is called RX.

Since the UART involves parallel – to – serial and serial – to – parallel data conversion, shift registers are an essential part of the UART hardware (two shift registers to be specific: Transmitter Shift Register and Receiver Shift Register).

How UART Works?

In UART Serial Communication, the data is transmitted asynchronously i.e. there is no clock or other timing signal involved between the sender and receiver. Instead of clock signal, UART uses some special bits called Start and Stop bits.

These bits are added to the actual data packet at the beginning and end respectively. These additional bits allows the receiving UART to identify the actual data.



The image above shows a typical UART connection. The transmitting UART receives data from the controlling device through the data bus. The controlling device can be anything like a CPU of a microprocessor or a microcontroller, memory unit like a RAM or ROM, etc. The data received by the transmitting UART from the data bus is parallel data.

To this data, the UART adds Start, Parity and Stop bits in order to convert it into a data packet. The data packet is then converted from parallel to serial with the help of shift register and is transmitted bit – by – bit from the TX pin.

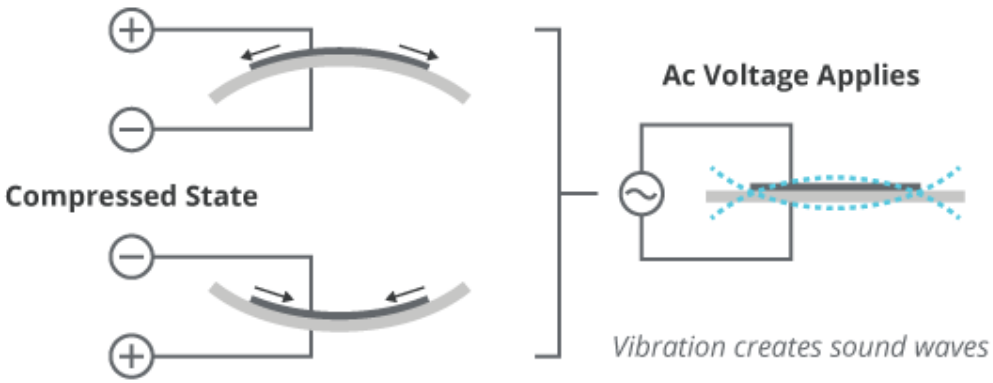
The receiving UART receives this serial data at the RX pin and detects the actual data by identifying the start and stop bits. Parity bit is used to check the integrity of the data.

Up on separating the start, parity and stop bits from the data packet, the data is converted to parallel data with the help of shift register. This parallel data is sent to the controller at the receiving end through a data bus.

Buzzer

Piezo-type buzzer's core is the piezoelectric element. The piezoelectric element is made out of piezoelectric ceramic as well as the metal plate, they are held together in or piece by the adhesive.

Piezo buzzer



Due to the piezoelectric material, when an AC is passed through it, it'll shrink and expand. This would then cause a vibration which would produce sound waves as you can see from the diagram.

A piezo buzzer works by applying an alternating voltage to the piezoelectric ceramic material. The introduction of such an input signal causes the piezoceramic to vibrate rapidly, resulting in the generation of sound waves.

GSM

GSM SIM 900A

SIM900 GSM/GPRS shield is a GSM modem, which can be integrated into a great number of IoT projects. You can use this shield to accomplish almost anything a normal cell phone can; SMS text messages, Make or receive phone calls, connecting to internet through GPRS, TCP/IP, and more! To top it off, the shield supports quad-band GSM/GPRS network, meaning it works pretty much anywhere in the world.

DB9 RS232 interface for direct communication with computer or MCU kit

Compatible with ARDUINO, RASPBERRY PI, ARM, AVR, PIC, 8051, etc. - Can also be directly connected to computer via Serial Port (Use GSM Tester or write your own Software)

Best suited for GSM based Microcontroller Projects (better than SIM300 and other GSM Modems)

Option for connecting MIC and SPEAKER directly to GSM MODEM for calls (LINE IN also available)

Supports communication through RS232 with DB9 Connector, TTL Pins & I2C Pins

CALL SMS GPRS facility - MIC input, LINE input & SPEAKER output pins

SIM900A GSM Module is the smallest and cheapest module for GPRS/GSM communication. It is common with Arduino and microcontroller in most of embedded application. The module offers GPRS/GSM technology for communication with the uses of a mobile sim. It uses a 900 and 1800MHz frequency band and allows users to receive/send mobile calls and SMS. The keypad and display interface allows the developers to make the customize application with it. Furthermore, it also has modes, command mode and data mode. In every country the GPRS/GSM and different protocols/frequencies to operate. Command mode helps the developers to change the default setting according to their requirements.

SIM900A Modem is built with Dual Band GSM based SIM900A modem from SIMCOM. It works on frequencies 900MHz. SIM900A can search these two bands automatically. The frequency bands can also be set by AT Commands. The baud rate is configurable from 1200-115200 through AT command. SIM900A is an ultra compact and wireless module. The Modem is coming interface, which allows you connect PC as well as microcontroller with RS232 Chip(MAX232). It is suitable for SMS, Voice as well as DATA transfer application in M2M interface. The onboard Regulated Power supply allows you to connect wide range unregulated power supply. Using this modem, you can make audio calls, SMS, Read SMS, attend the incoming calls and ect. through simple AT commands. This is a complete GSM module in a SMT type and made with a very powerful single-chip, allowing you to benefit from small dimensions. SIM 900A GSM Modem with serial and TTL outputs.

The SIM900A is integrated with the TCP/IP protocol; extended TCP/IP AT commands are developed for customers to use the TCP/IP protocol easily, which is very useful for those data transfer applications.

Mobile phone converts voice, text, multi-media messages or data calls into Radio Frequencies (RF). Mobile phone base stations transmit and receive these RF signals and connect callers to other phones and other networks.

Understanding Modems

Wireless modems generate, transmit or decode data from a cellular network, in order to establish communication.

A GSM/GPRS modem is a class of wireless modem, designed for communication over the GSM and GPRS network. It requires a SIM (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.

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.

Its functions include:

Read, write and delete SMS messages.

Send SMS messages.

Monitor the signal strength.

Monitor the charging status and charge level of the battery.

Read, write and search phone book entries.

What is a mobile station?

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.

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, signalling 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 module or GPRS module

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.

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.

Libraries

TinyGPS++ is a new Arduino library for parsing NMEA data streams provided by GPS modules. Like its predecessor, TinyGPS, this library provides compact and easy-to-use methods for extracting position, date, time, altitude, speed, and course from consumer GPS devices.

SoftwareSerial library was developed to ensure that any pins of Arduino can exchange Serial data with other peripherals, like GNSS receivers, using software. Arduino Uno, for example, has only one HardwareSerial port (pins 0 and 1), which is connected to the USB via the USB to UART conversion chip. Thus, if you have any other peripheral that requires serial communication, in the absence of SoftwareSerial, you'd have to do away with USB Serial communication.

SoftwareSerial has some limitations −

If you are using multiple SoftwareSerial ports, only one can receive data at a time

Speeds can be up to a maximum of 115200 bps

Basic Usage of GPS functions

TinyGPS gps

Create the TinyGPS object, giving it a name of your choice. While you could create multiple TinyGPS objects, it's unlikely you would need to decode more than one NEMA format data stream.

gps.encode(c)

Each byte of NEMA data must be giving to TinyGPS by using encode(). True is returned when new data has been fully decoded and can be used.

gps.get_position(&latitude, &longitude, &age)

Get the position, where latitude and longitude are variables of long type, and the values are multiplied by 10,000. The age variable must be unsigned long type.

gps.f_get_position(&flatitude, &flongitude, &age)

Get the position, where latitude and longitude are variables of float type, and the actual values are returned. Float types are easier to use, but result in larger and slower code. The age variable must be unsigned long type.

National Electrical Manufacturers Association(NEMA) before GPS