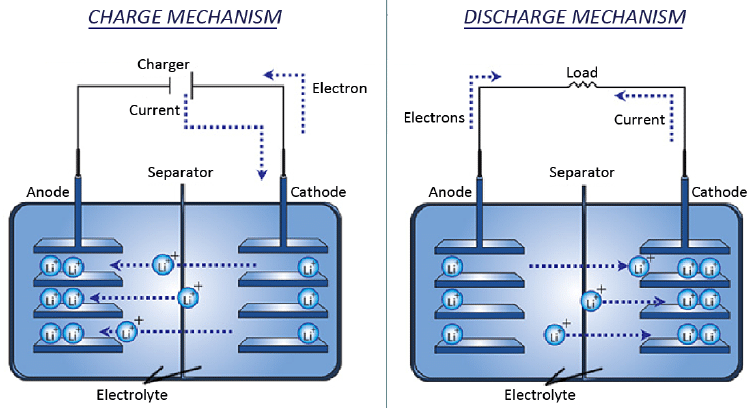
Tracking Device :-

* Rechargeable Li-ion Battery -



A **lithium-ion battery** or **Li-ion battery** is a type of rechargeable battery. Lithium-ion batteries are commonly used for portable electronics and electric vehicles and are growing in popularity for military and aerospace applications.

In the batteries, [lithium](https://en.wikipedia.org/wiki/Lithium) [ions](https://en.wikipedia.org/wiki/Ion) move from the negative [electrode](https://en.wikipedia.org/wiki/Electrode) through an [electrolyte](https://en.wikipedia.org/wiki/Electrolyte) to the positive electrode during discharge, and back when charging.  
 

Li-ion batteries use an [intercalated](https://en.wikipedia.org/wiki/Intercalation_(chemistry)) lithium [compound](https://en.wikipedia.org/wiki/Chemical_compound) as the material at the positive electrode and typically [graphite](https://en.wikipedia.org/wiki/Graphite) at the negative electrode. The batteries have a high [energy density](https://en.wikipedia.org/wiki/Energy_density), no [memory effect](https://en.wikipedia.org/wiki/Memory_effect) and low [self-discharge](https://en.wikipedia.org/wiki/Self-discharge).

* Arduino Nano :

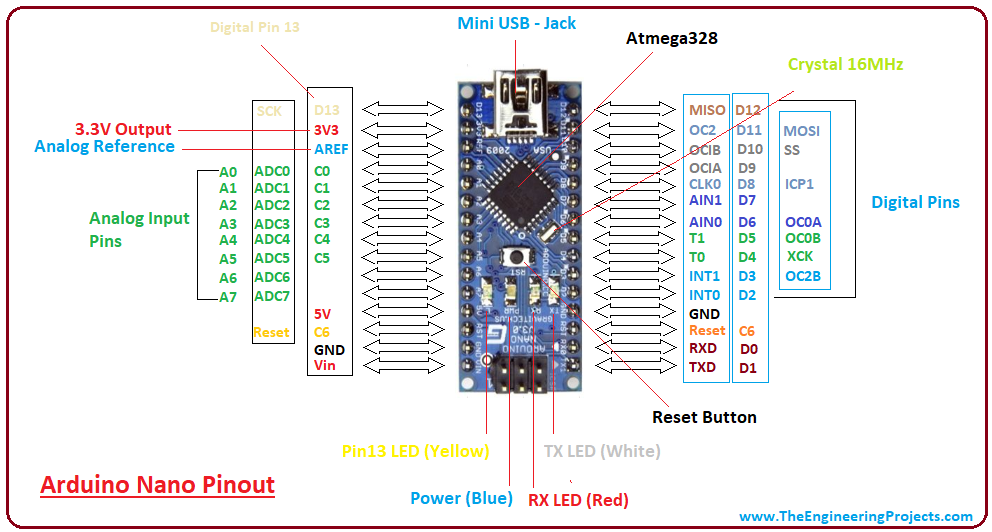
The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P; offers the same connectivity and specs of the UNO board in a smaller form factor. It can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source. Each of the 14 digital pins on the Nano can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms

**Physical Characteristics**

It has 22 input/output pins in total. 14 of these pins are digital pins. Arduino Nano has 8 analogue pins. It has 6 PWM pins among the digital pins. It has a crystal oscillator of 16MHz. It's operating voltage varies from 5V to 12V.It also supports different ways of communication, which are:

* Serial Protocol.
* I2C Protocol.
* SPI Protocol.

It also has a mini USB Pin which is used to upload code. It also has a Reset button on it.



The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

**Features:-**

1.Microcontroller-ATmega328

2.Operating Voltage- 5V

3.Input Voltage (recommended) - 7-12V

4.Input Voltage (limits)- 6-20V

5.Digital I/O Pins- 22 (of which 6 provide PWM output)

6.Analog Input Pins- 8

7.DC Current per I/O Pin - 40 mA

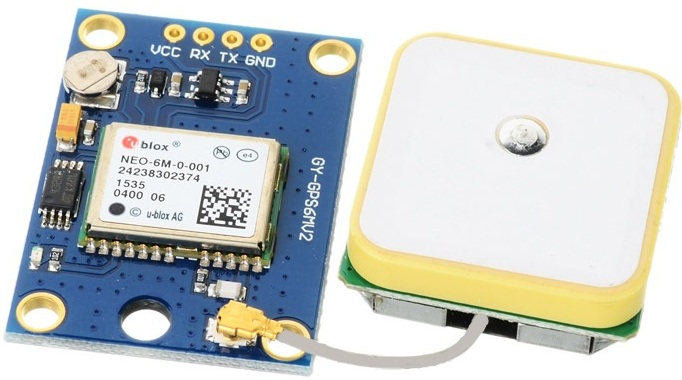
8.Flash Memory -32 KB (ATmega328) of which 2 KB used by bootloader

9.SRAM -2 KB (ATmega328)

10.EEPROM -1 KB (ATmega328)

11.Clock Speed -16 MHz

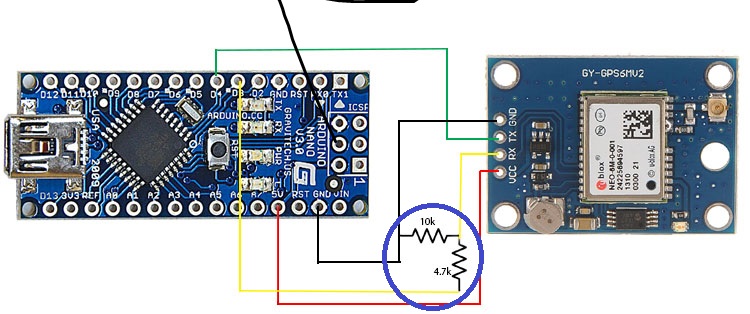
* GPS :



Global Positioning System. The GPS concept is based on time and the known position of GPS specialized satellites. The satellites carry very stable atomic clocks that are synchronized with one another and with the ground clocks. Any drift from time maintained on the ground is corrected daily. In the same manner, the satellite locations are known with great precision. GPS receivers have clocks as well, but they are less stable and less precise.

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.

**Circuit Diagram:**



* GSM 800 :



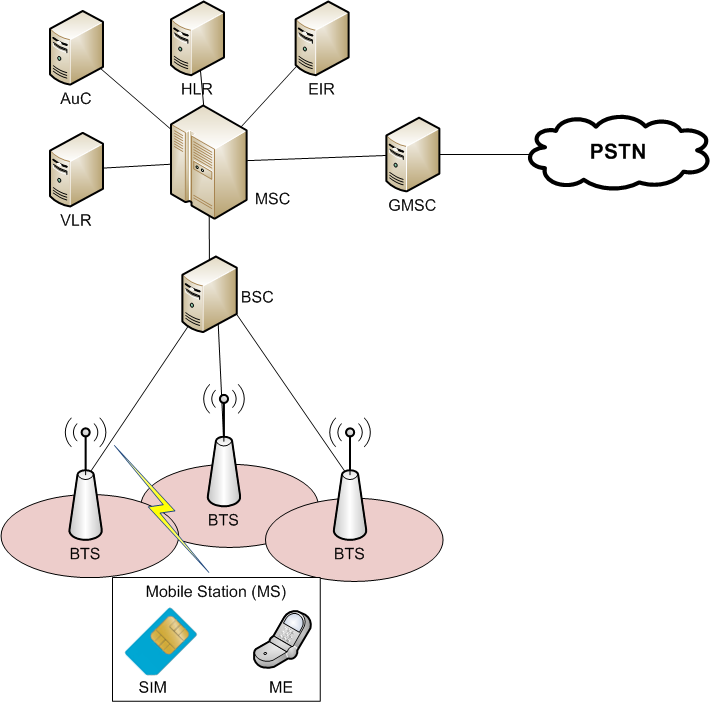
The GSM module is used to send the distress message to the numbers registered in the module. A GSM module is a chip or circuit that will be used to establish communication between a mobile device or a computing machine and a GSM system. The modem (modulator-demodulator) is a critical part here. The GSM standard was developed for setting protocols for second generation (2G) digital cellular networks. In GSM, geographical area is divided into hexagonal cells whose side depends upon power of transmitter and load on transmitter (number of end user). At the center of cell, there is a base station consisting of a transceiver (combination of transmitter and receiver) and an antenna.

These modules consist of a GSM module powered by a power supply circuit and communication interfaces (like RS-232, USB 2.0, and others) for computer. A GSM modem can be a dedicated modem device with a serial, USB or Bluetooth connection, or it can be a mobile phone that provides GSM modem capabilities.

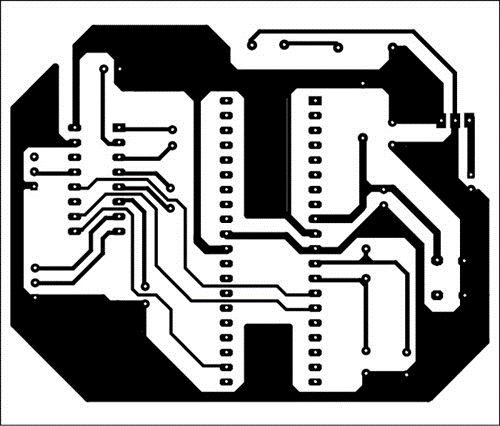
**GSM Architecture:**

The GSM architecture is divided into Radio Subsystem, Network and Switching Subsystem and the Operation Subsystem. The radio sub system consists of the Mobile Station and Base Station Subsystem.

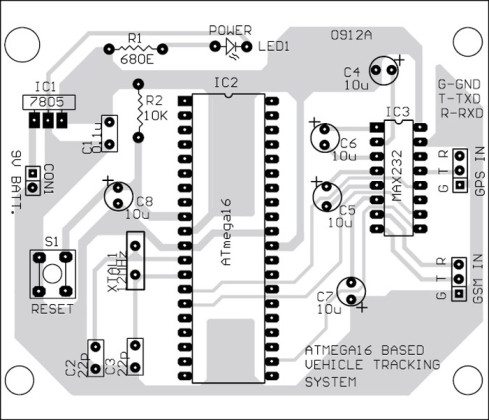
The mobile station is generally the mobile phone which consists of a transceiver, display and a processor. Each handheld or portable mobile station consists of a unique identity stored in a module known as SIM (Subscriber Identity Chip). It is a small microchip which is inserted in the mobile phone and contains the database regarding the mobile station.



* PCB Etching of GPS and GSM module for tracking and alerting system:



*An actual-size, single-side PCB for the GSM and GPS based vehicle tracking circuit*



*Component layout for the PCB*

**For Construction and Testing:**

An actual-size, single-side PCB layout of the GPS and GPS based vehicle tracking circuit and its component layout are shown in above figures.

Assemble the components on the PCB with IC bases for ATmega16 and MAX232. Burn the code into the MCU and mount it on the PCB. Insert the SIM card with sufficient balance in the GSM module.

Now, for testing the device:

Connect the circuit to GPS and GSM modem.

1. Switch on the circuit and you will see LED1 glow.
2. Switch on the GPS module and wait for 10-15 minutes for initialisation.
3. Switch on the GSM modem.
4. Dial the mobile number in the GSM modem. After two rings, the ringing stops automatically. Wait for a few seconds. You will get an SMS alert in your mobile.
5. Check your SMS inbox. You will see the latitude and longitude data in the form of SMS text.
6. Open a standard map and locate the point on the map. You can also enter latitude and longitude values in a software such as on http://www.latlong.net/Show-Latitude-Longitude.html or any other suitable software.

**Program for working of microcontroller:**

#include <SoftwareSerial.h>

SoftwareSerial gps(10,11); // RX, TX

//String str="";

char str[70];

String gpsString="";

#define swt 2

#define led 13

char \*test="$GPGGA";

String latitude="No Range ";

String longitude="No Range ";

int temp=0,i;

boolean gps\_status=0;

void setup()

{

pinMode(swt,OUTPUT);

pinMode(led,OUTPUT);

Serial.begin(9600);

gps.begin(9600);

gsm\_init();

Serial.println("AT+CMGF=1");

delay(2000);

Serial.println("AT+CMGD=4");

delay(2000);

//Serial.println("AT+CMGR=1");

delay(2000);

Serial.println("AT+CNMI=2,2,0,0,0");

delay(2000);

Serial.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode

delay(1000);

Serial.println("AT+CMGS=\"+919179997205\"\r"); // Replace x with mobile number

delay(1000);

Serial.println("women security system");

Serial.println((char)26);// ASCII code of CTRL+Z

get\_gps();

delay(2000);

temp=0;

}

void loop()

{

digitalWrite(led,HIGH);

delay(150);

digitalWrite(led,LOW);

delay(150);

if(digitalRead(swt)==0)

{

delay(5000);

if(digitalRead(swt)==0)

{

while(1)

{

get\_gps();

tracking();

delay(10000);

}

}

}

}

void gpsEvent()

{

gpsString="";

while(1)

{

while (gps.available()>0) //checking serial data from GPS

{

char inChar = (char)gps.read();

gpsString+= inChar; //store data from GPS into gpsString

i++;

if (i < 7)

{

if(gpsString[i-1] != test[i-1]) //checking for $GPGGA sentence

{

i=0;

gpsString="";

}

}

if(inChar=='\r')

{

if(i>65)

{

gps\_status=1;

break;

}

else

{

i=0;

}

}

}

if(gps\_status)

break;

}

}

void gsm\_init()

{

boolean at\_flag=1;

while(at\_flag)

{

Serial.println("AT");

while(Serial.available()>0)

{

if(Serial.find("OK"))

at\_flag=0;

}

delay(1000);

}

boolean echo\_flag=1;

while(echo\_flag)

{

Serial.println("ATE0");

while(Serial.available()>0)

{

if(Serial.find("OK"))

echo\_flag=0;

}

delay(1000);

}

boolean net\_flag=1;

while(net\_flag)

{

Serial.println("AT+CPIN?");

while(Serial.available()>0)

{

if(Serial.find("+CPIN: READY"))

net\_flag=0;

}

delay(1000);

}

}

void get\_gps()

{

gps\_status=0;

int x=0;

while(gps\_status==0)

{

gpsEvent();

int str\_lenth=i;

latitude="";

longitude="";

int comma=0;

while(x<str\_lenth)

{

if(gpsString[x]==',')

comma++;

if(comma==2) //extract latitude from string

latitude+=gpsString[x+1];

else if(comma==4) //extract longitude from string

longitude+=gpsString[x+1];

x++;

}

int l1=latitude.length();

latitude[l1-1]=' ';

l1=longitude.length();

longitude[l1-1]=' ';

i=0;x=0;

str\_lenth=0;

delay(500);

}

}

void tracking()

{

Serial.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode

delay(1000);

Serial.println("AT+CMGS=\"+919179997205\"\r"); // Replace x with mobile number

delay(1000);

Serial.println("Emergency Alert:");

Serial.print("User Current Location is:");

Serial.print("Latitude:");

Serial.print(latitude);

Serial.print("Longitude:");

Serial.println(longitude);

Serial.print("Please take some action");

Serial.println((char)26);// ASCII code of CTRL+Z

delay(2000);

}