**SIM808**

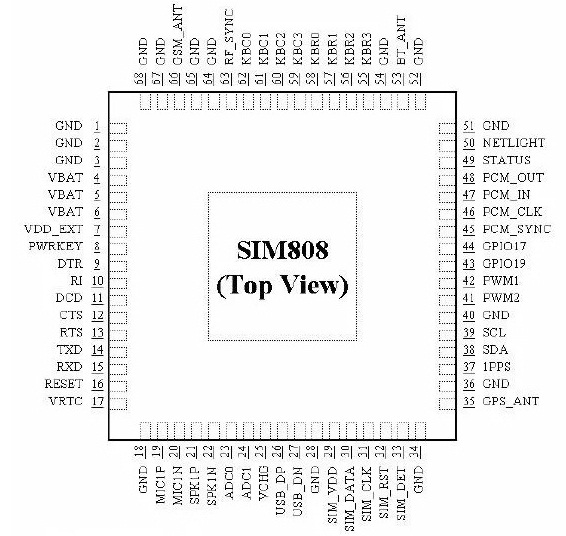


SIM808 GSM/GPRS/GPS UART mini Modem,helps you with wireless connectivity for your project using TTL UART interface. It is a complete Quad-Band GSM/GPRS module that works on frequencies 850/ 900/ 1800/ 1900 MHz which combines GPS technology for satellite navigation. The compact design which integrated GPRS and GPS in a SMT package will significantly save both time and costs for customers to develop GPS enabled applications.

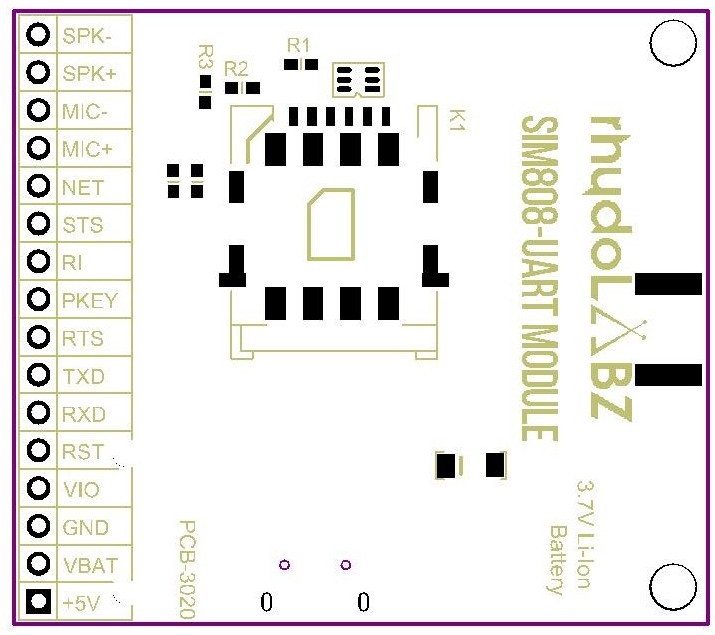
The Modem is manufactured with Automatic Pick and place machine with high quality standard. The Modem is coming with selectable interfacing voltage,which allows you to connect 2V8 to 5V , including 3V3 microcontroller directly without any extra level conversion chips. This module supports HTTP and TCP/IP protocol. It is suitable for SMS,voice,GPS,bluetooth,as well as data transfer application in M2M interface.It provide ultra low power consumption in sleep mode and integrated with charging circuit for Li-Ion battery,that make it get a super long standby time and convenient for projects.

SIM808 module is a GSM and GPS two-in-one function module. It is based on the latest GSM/GPS module SIM808 from SIMCOM, supports GSM/GPRS Quad-Band network and combines GPS technology for satellite navigation. It features ultra-low power consumption in sleep mode and integrated with charging circuit for Li-Ion batteries, that make it get a super long standby time and convenient for projects that use rechargeable Li-Ion battery. It has high GPS receive sensitivity with 22 tracking and 66 acquisition receiver channels. Besides, it also supports A-GPS that available for indoor localization. The module is controlled by AT command via UART and supports 3.3V and 5V logical level.

##### **Pin out of SIM808**

[](https://wiki.rhydolabz.com/wp-content/uploads/pinout23.jpg)

##### **Pin Details**

[](https://wiki.rhydolabz.com/wp-content/uploads/PCB-3020-SIM808-UART-MODULE-BTM-LAYOUT.jpg)

**Sample Applications:**

* Send a text message in English.
* Specified calls, automatically hang up after 1 minute.
* Received specified text messages, control LED open circuit.
* Detection module registered to the network, send the message until reading a SIM card.
* Connect to the specified server and send the specified data by GPRS.
* Provide information of dormant power consumption at about 10 MA

#### **Features:**

1. Quad-band 850/900/1800/1900MHz
2. GPRS multi-slot class12 connectivity: max. 85.6kbps(down-load/up-load)
3. GPRS mobile station class B
4. Controlled by AT Command (3GPP TS 27.007, 27.005 and SIMCOM enhanced AT Commands)
5. Supports charging control for Li-Ion battery
6. Supports Real-Time Clock
7. Integrated GPS/CNSS and supports A-GPS
8. Low power consumption, 10mA in sleep mode
9. Supports GPS NMEA protocol
10. AT command operation, can support SMS, phone, GPRS data, GPS data is automatically output, embedded TCP protocol to support transparent mode and command mode, supports HTTP protocol to support DTMF decoding, support MMS, support recording function, etc.
11. Standard SIM Card

#### **Package Includes:**

1 x GSM GPRS SIM808 Module SMS Chip Development Board

**Connections should be like below :**

* TX(modem)-RX(converter TTL)
* RX(modem)-TX(converter TTL)
* VIO(Modem)-PWR(converter)
* Make ground (GND-GND)common.gistered with the network.

## **Arduino Code – Testing AT Commands**

To send AT commands and communicate with the SIM800L module, we will be using the Serial Monitor. The sketch below will enable the Arduino to communicate with the SIM800L module over the serial monitor. Before we proceed with the detailed analysis of the code, connect your Arduino to PC, compile the below code and upload it to Arduino.

#include <SoftwareSerial.h>

//Create software serial object to communicate with SIM800L

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2

void setup()

{

//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)

Serial.begin(9600);

//Begin serial communication with Arduino and SIM800L

mySerial.begin(9600);

Serial.println("Initializing...");

delay(1000);

mySerial.println("AT"); //Once the handshake test is successful, it will back to OK

updateSerial();

mySerial.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31 is the best

updateSerial();

mySerial.println("AT+CCID"); //Read SIM information to confirm whether the SIM is plugged

updateSerial();

mySerial.println("AT+CREG?"); //Check whether it has registered in the network

updateSerial();

}

void loop()

{

updateSerial();

}

void updateSerial()

{

delay(500);

while (Serial.available())

{

mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port

}

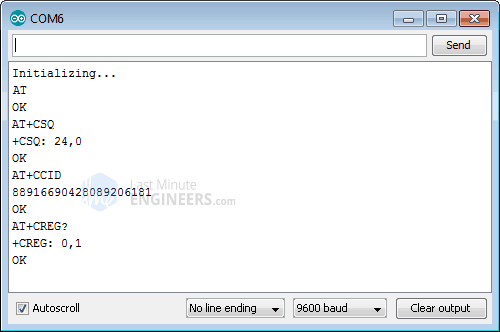
while(mySerial.available())

{

Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port

}

}

Once you have uploaded the sketch, open the serial monitor at baud rate 9600. You should see the output below on the serial monitor. 

### Code Explanation:

The sketch begins by including the SoftwareSerial.h library and initializing the SoftwareSerial object with the Arduino pins to which the Tx and Rx of the SIM800L module are connected.

#include <SoftwareSerial.h>

//Create software serial object to communicate with SIM800L

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2

In the setup function, we initiate a serial communication link between the Arduino, the Arduino IDE and the SIM800L module at a baud rate of 9600.

//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)

Serial.begin(9600);

//Begin serial communication with Arduino and SIM800L

mySerial.begin(9600);

Now that we have established a basic connection, we will try to communicate with the SIM800L module by sending AT commands.

AT – This is the most basic AT command. It also initializes the Auto-bauder. If all is well, it sends the OK message, telling you that it is understanding you correctly. You can then send some commands to query the module and get information about it.

AT+CSQ – It checks ‘Signal Strength’. The first number in the output response is the signal strength in dB. It should be more than about 5. Higher is better. Note that this signal strength depends on your antenna and location!

AT+CCID – It checks whether the SIM card is valid or not and sends the SIM card number.

AT+CREG? – It checks whether you are registered to the network or not. The second number in the output response should be 1 or 5. 1 indicates that you are registered to a home network and 5 indicates a roaming network. Any number other than these two indicates that you are not registered to any network.

mySerial.println("AT"); //Once the handshake test is successful, it will back to OK

updateSerial();

mySerial.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31 is the best

updateSerial();

mySerial.println("AT+CCID"); //Read SIM information to confirm whether the SIM is plugged

updateSerial();

mySerial.println("AT+CREG?"); //Check whether it has registered in the network

updateSerial();

In the loop, we call a custom function called updateSerial() which continuously waits for any input from the serial monitor and sends it to the SIM800L module via the D2 pin (the module’s RX). It also continuously reads the D3 pin (the module’s TX) to see if there is a response from the SIM800L module.

void updateSerial()

{

delay(500);

while (Serial.available())

{

mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port

}

while(mySerial.available())

{

Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port

}

}

### Try Sending Different AT Commands

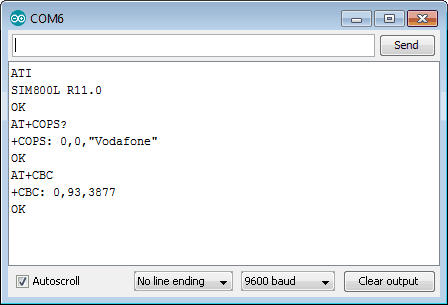
You can now send commands like the ones below through the Serial Monitor to get more information about your network connection and battery status:

ATI – Returns the module name and revision.

AT+COPS? – Checks which network you are connected to.

AT+COPS=? – Returns the list of operators present in the network.

AT+CBC – Returns Li-Po battery status. The second number is the battery level (in our case it is 93%) and the third number is the actual voltage in mV (in our case 3.877 V)



## **Arduino Code – Sending an SMS**

Let’s program our Arduino to send an SMS.

Before trying out the sketch, you’ll need to enter the phone number to which you want to send an SMS. Look for the highlighted string ZZxxxxxxxxxx and replace ZZ with the county code and xxxxxxxxxx with the 10 digit phone number.

#include <SoftwareSerial.h>

//Create software serial object to communicate with SIM800L

SoftwareSerial mySerial(8, 12); //Define SIM800L Tx & Rx pins which are connected to Arduino #8 & #12

void setup()

{

//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)

Serial.begin(9600);

//Begin serial communication with Arduino and SIM800L

mySerial.begin(9600);

Serial.println("Initializing...");

delay(1000);

SerialCMD("AT"); //First handshake test is successful, it will back to OK

SerialCMD("AT+CMGF=1"); // Configuring TEXT mode

SerialCMD("AT+CMGS=\"+ZZxxxxxxxxxx\"");// ZZ with country code

delay(1000);

mySerial.print("Hello this is SMS Testing From SIM808");

delay(1000);

// Read and print the MSG

while (mySerial.available())

{

Serial.write(mySerial.read()); // Serial received to Serial Port

}

mySerial.write(26); //ESC Character Indicating Terminate the MSG

}

void loop()

{

}

void SerialCMD(String command) {

mySerial.println(command); // Send the AT command

delay(1000); // Wait for the response

// Read and print the response

while (mySerial.available()) {

Serial.write(mySerial.read()); // Forward what Software Serial received to Serial Port

}

}

The screenshot below shows the SMS sent from the SIM800L GSM module.

### Code Explanation:

The sketch is almost the same as the first sketch except for the code snippet below. Once the connection is established, we send the following AT commands:

AT+CMGF=1 – Selects the SMS message format as text. The default format is[Protocol Data Unit](https://en.wikipedia.org/wiki/Protocol_data_unit) (PDU).

AT+CMGS=+ZZxxxxxxxxxx – Sends SMS to the specified phone number. After this AT command any text message followed by ‘Ctrl+z’ character is treated as SMS. ‘Ctrl+z’ is actually a 26th non-printing character that is described as ‘substitute’ in the[ASCII table](https://www.asciitable.com/). Therefore, we need to send 26 (0x1A) at the end of the command.

## **Arduino Code – Reading an SMS**

Now let’s program our Arduino to read incoming messages. This sketch can be very useful when you need to perform an action like turn ON/OFF lights, enable fan, enable sprinkler system when you receive a specific SMS.

#include <SoftwareSerial.h>

//Create software serial object to communicate with SIM800L

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2

void setup()

{

//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)

Serial.begin(9600);

//Begin serial communication with Arduino and SIM800L

mySerial.begin(9600);

Serial.println("Initializing...");

delay(1000);

mySerial.println("AT"); //Once the handshake test is successful, it will back to OK

updateSerial();

mySerial.println("AT+CMGF=1"); // Configuring TEXT mode

updateSerial();

mySerial.println("AT+CNMI=1,2,0,0,0"); // Decides how newly arrived SMS messages should be handled

updateSerial();

}

void loop()

{

updateSerial();

}

void updateSerial()

{

delay(500);

while (Serial.available())

{

mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port

}

while(mySerial.available())

{

Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port

}

}

Once you send SMS to the SIM800L GSM module, you will see the following output on the serial monitor.

As you can see the output response starts with +CMT: All the fields in the response are comma separated where the first field is the phone number and the second field is the name of the person sending the SMS. The third field is a timestamp while the fourth field is the actual message.

### Code Explanation:

The sketch is almost the same as the first sketch except for the code snippet below. Once the connection is established, we send the following AT commands:

AT+CMGF=1 – Selects the SMS message format as text. The default format is Protocol Data Unit (PDU)

AT+CNMI=1,2,0,0,0 – Specifies how incoming SMS messages should be handled. This way you can tell the SIM800L module to either forward incoming SMS messages directly to the PC, or save them to the message storage and then inform the PC about their locations in the message storage.

## **Arduino Code – Making a Call**

Now let’s program our Arduino to make a call. This sketch can be very useful when you want your Arduino to make a SOS/distress call in case of an emergency like the temperature is exceeded or someone breaks into your house.

Before trying out the sketch, you’ll need to enter the phone number you want to call. Look for the highlighted string ZZxxxxxxxxxx and replace ZZ with the county code and xxxxxxxxxx with the 10 digit phone number.

#include <SoftwareSerial.h>

//Create software serial object to communicate with SIM800L

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2

void setup()

{

//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)

Serial.begin(9600);

//Begin serial communication with Arduino and SIM800L

mySerial.begin(9600);

Serial.println("Initializing...");

delay(1000);

mySerial.println("AT"); //Once the handshake test is successful, i t will back to OK

updateSerial();

mySerial.println("ATD+ +ZZxxxxxxxxxx;"); // change ZZ with country code and xxxxxxxxxxx with phone number to dial

updateSerial();

delay(20000); // wait for 20 seconds...

mySerial.println("ATH"); //hang up

updateSerial();

}

void loop()

{

}

void updateSerial()

{

delay(500);

while (Serial.available())

{

mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port

}

while(mySerial.available())

{

Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port

}

}

The following screenshot shows a call made from the SIM800L GSM module.

### Code Explanation:

The following AT commands are used to make a call:

ATD+ +ZZxxxxxxxxxx; – Dials a specified number. Don’t forget the semicolon ; at the end of the command.

ATH – Hangs up the call.

## **Arduino Code – Receiving a Call**

No special code is required to receive calls; You just have to keep listening to the SIM800L module. You may find this sketch very useful when you need to perform an action when a call is received from a specific phone number.

#include <SoftwareSerial.h>

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx

void setup()

{

Serial.begin(9600);

mySerial.begin(9600);

Serial.println("Initializing...");

}

void loop()

{

updateSerial();

}

void updateSerial()

{

delay(500);

while (Serial.available())

{

mySerial.write(Serial.read());

}

while(mySerial.available())

{

Serial.write(mySerial.read());

}

}

The following output on the serial monitor shows the calls received by the SIM800L GSM module.

As you can see incoming calls are usually represented by ‘RING’ on the serial monitor, followed by the phone number and caller ID. The following AT commands can be used to accept or hang up the call:

ATA – Accepts an incoming call.

ATH – Hangs up the call. On hanging up the incoming call, we get NO CARRIER on the serial monitor indicating that the call could not be connected.

# **Mini GSM/GPRS + GPS Breakout - SIM808**

## **Overview**

This board based on the latest SIMCOM SIM808 GSM/GPS module, it offers cellular GSM and GPRS data along with GPS technology for satellite navigation.

The board features ultra-low power consumption in sleep mode, giving the project incredibly long standby times. Furthermore there's an onboard battery charging circuit that can be used with LiPo batteries.

The GPS receiver is incredibly sensitive with 22 tracking and 66 acquisition channels, and also supports assisted-GPS (A-GPS) for indoor localisation. The board is controlled by AT command via UART and supports 3.3V and 5V logical level. It comes with a mini GPS and GSM antenna, however a battery is optional.

The board uses the 2G (not 3G or LTE) GSM networks.

## **Features**

* Quad-band 850/900/1800/1900MHz
* GPRS multi-slot class12 connectivity: max. 85.6kbps(down-load/up-load)
* GPRS mobile station class B
* Controlled by AT Command (3GPP TS 27.007, 27.005 and SIMCOM enhanced AT Commands)
* Supports charging control for Li-Ion battery
* Supports Real Time Clock
* Supply voltage range 3.4V ~ 4.4V
* Integrated GPS/CNSS and supports A-GPS
* Supports 3.0V to 5.0V logic level
* Low power consumption, 1mA in sleep mode
* Supports GPS NMEA protocol
* Compact size 27mm x 46mm x 10mm
* Standard SIM Card

## **GPS Specifications**

* Receiver channels: 22 tracking / 66 acquisition
* Coarse/Acquisition code: GPS L1
* Tracking sensitivity: -165dBm
* Cold start time: 30s (typ.)
* Hot start time: 1s (typ.)
* Warm start time: 28s (typ.)
* Horizontal position accuracy: < 2.5m CEP
* Power consumption - Acquisition: 42mA
* Power consumption - Continuous tracking: 24mA
* Update rate: 5Hz

## **Functional Block**

* GPS Antenna: this is an uFL GPS antenna connector. You can connect either passive or active GPS antenna to it. Active GPS antenna runs at 2.8V voltage.
* MicroUSB: the charging interface for Li-Ion battery, of input voltage range from 5V to 7V.
* Power Button: this is the hard power switch for the module. When the module is power up, you can turn on or turn off the module by pressing the button for 2s.
* Net Indicator: Red LED, it will tell the what status is about the module linking to network. It can be turned off by LEDs\_EN jumper pad.
* Status Indicator: Green LED, it will tell whether the module is on, light when the module is running. It can be turned off by LEDs\_EN jumper pad.
* Li-ion Battery: this is power supply for the module, input voltage is from 3.4V to 4.4V. It uses the JST-2.0mm connector, that make it convenient to connect to 3.7V Li-Po Battery.
* GSM Antenna: this is an uFL GSM antenna connector, just connect it to a GSM antenna for receiving GSM signal.
* SIM - Card Holder: SIM card holder for standard SIM card

### Pin Definitions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **I/O** | **Description** | **Note** |
| **BAT** |  | Power input / output | 3.4V - 4.4V DC |
| GND |  | Power ground / logic ground |  |
| VIO | I | Logic level reference | 3.0V - 5.0V DC |
| DTR | I | Wake up pin for module in sleep mode |  |
| PWR | O | Power switch | Low level pulse |
| RI | O | Event/ message pin |  |
| TXD | O | Transmit data | UART output from SIM808 |
| RXD | I | Receive data | UART Input to SIM808 |
| RST | I | Module rest |  |

### Indicator LEDs

|  |  |  |
| --- | --- | --- |
| **Indicator LEDs** | **Status** | **Behavior** |
| **Operating Status (Green)** | Off | SIM808 is not running |
|  | On | SIM808 is running |
| **Network Status (Red)** | Off | SIM808 is not running |
|  | 64ms on/ 800ms Off | SIM808 not registered to the network |
|  | 64ms On/ 3000ms Off | SIM808 registered to the network |
|  | 64ms On/ 300ms Off | PPP GPRS communication is established |

### Getting Started with AT Command

The module is controlled by AT command via serial port, here we use Arduino as USB to serial tool. Upload the following code to Arduino and open the serial monitor. If you use other USB to serial tools, you can use [AT Command Tester](https://seeeddoc.github.io/AT_Command_Tester_Application) or [SSCOM32](https://seeeddoc.github.io/Mini_GSM-GPRS_Plus_GPS_Breakout-SIM808/res/Sscom32E.zip) to test AT commands.

// this sketch is used for testing LoNet with Arduino

// Connect VIO to +5V

// Connect GND to Ground

// Connect RX TX pins with 8 and 12

#include <SoftwareSerial.h>

SoftwareSerial mySerial(8, 12); // RX, TX

void setup()

{

// Open serial communications and wait for port to open:

Serial.begin(9600);

mySerial.begin(9600);

}

void loop() // run over and over

{

if (mySerial.available())

Serial.write(mySerial.read());

if (Serial.available())

{

while(Serial.available())

{

mySerial.write(Serial.read());

}

mySerial.println();

}

}

#### **Set Baud and Enable Charging Function**

It is recommended to execute this process when first time to use the module. In the Serial Monitor columns of following tables, input of AT commands are in back, module returns values are in orange.

|  |  |
| --- | --- |
| **Serial Monitor** | **Description** |
| AT OK | Send command “AT” to synchronize baud rate. Serial port of module is by default set at auto-baud mode, and in this mode, it will not output any indications when the module is on. |
| AT+IPR=9600 OK | Set baud rate at 9600bps, supports baud rate from 1200bps to 115200bps. |
| AT+ECHARGE=1 OK | Send command “AT+ECHARGE=1” to enable battery charging function. By default the charging function is closed. |
| AT&W OK | Save parameter setting. |
| AT+CPOWD=1 NORMAL POWER DOWN | Power down the module. |
| RDY +CFUN: 1 GPS Ready +CPIN: READY Call Ready SMS Ready | Turn on the module again by the power button, it will response status about GPS and GSM. |
| AT+CBC +CBC: 1,96,4175 OK | Inquire charging status and remaining battery capacity. |
| AT+CSQ +CSQ: 14,0 OK | Inquire GSM signal quality. |

#### **Get location with GPS**

|  |  |
| --- | --- |
| **Serial Monitor** | **Description** |
| AT+CGPSPWR=1 OK | Open GPS |
| AT+CGPSSTATUS? +CGPSSTATUS: Location Not Fix OK | Read GPS fix status, “Location Not Fix” means that positioning is not successful. For the first time to start, it will take at least 30s. ***GPS must be tested by the window or outdoor.*** |
| AT+CGPSSTATUS? +CGPSSTATUS: Location 3D Fix OK | GPS has fixed with 3D status. |
| AT+CGPSINF=0 +CGPSINF: 0,2234.931817,11357.122485, 92.461185,20141031041141.000, 88,12,0.000000,0.000000 | Get the current GPS location information. Parameters formate: <mode>, <altitude>, <longitude>, <UTC time>, <TTFF>, <num>, <speed>, <course> |
| AT+CGPSOUT=32 OK $GPRMC,043326.000,A, 2234.9414,N,11357.1187,E, 0.000,143.69,311014,,,A\*50 | Read NMEA $GPRMC data, of which, “2234.9414 N, 11357.1187 E” is the location coordinates. For more details about NMEA sentences, [check this site](http://www.gpsinformation.org/dale/nmea.htm). |
| AT+CGPSRST=0 OK | Reset GPS in Cold Start Mode. |
| AT+CGPSRST=1 OK | Reset GPS in Hot Start Mode. |
| AT+CGPSPWR=0 OK | Close GPS. |

I sent AT command and return ok.  
However AT+CGPSPWR=? returns error. All GPS command showed error.

I think GPS is not initialized at all as i didn't see GPS message when module powered on.

Why is the GPS not working?

You have a version2 module. Use AT+CGNSxxx instead of AT+CGPSxxx

Use AT+CGNSxxx instead of AT+CGPSxxx commands

ATGNSPWR=1 and ATCGNSINF to get location.

if you get following output

1,0,19800106001337.000,,,,0.00,0.0,0,,,,,,0,0,,,,,

It means your GPS is not set.

Try to put GPS antenna outside your room, on open space for 10-15 minutes. GPS bulb will start blinking to confirm it is now ready for use.

Then use AT+CGNSINF.

Subnet mask = 255 255 255 0 ?? why

Default Ip address : 192.168.1.1