**SIM800L GSM Module & Arduino**

SIM800L GSM/GPRS module is a miniature GSM modem, which can be integrated into a great number of IoT projects. You can use this module 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 module supports quad-band GSM/GPRS network, meaning it works pretty much anywhere in the world.

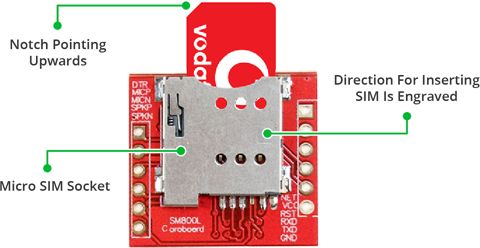
## Hardware Overview of SIM800L GSM/GPRS module

At the heart of the module is a SIM800L GSM cellular chip from SimCom. The operating voltage of the chip is from **3.4V to 4.4V**, which makes it an ideal candidate for direct LiPo battery supply. This makes it a good choice for embedding into projects without a lot of space.



All the necessary data pins of SIM800L GSM chip are broken out to a 0.1″ pitch headers. This includes pins required for communication with a microcontroller over **UART**. The module supports baud rate from **1200bps** to **115200bps** with Auto-Baud detection.

The module needs an external antenna to connect to a network. The module usually comes with a **Helical Antenna** and solders directly to NET pin on PCB. The board also has a U.FL connector facility in case you want to keep the antenna away from the board.



There’s a SIM socket on the back! Any activated, **2G micro SIM card** would work perfectly. Correct direction for inserting SIM card is normally engraved on the surface of the SIM socket.

This module measures only 1 inch² but packs a surprising amount of features into its little frame. Some of them are listed below:

* Supports Quad-band: GSM850, EGSM900, DCS1800 and PCS1900
* Connect onto any global GSM network with any 2G SIM
* Make and receive voice calls using an external 8Ω speaker & electret microphone
* Send and receive SMS messages
* Send and receive GPRS data (TCP/IP, HTTP, etc.)
* Scan and receive FM radio broadcasts
* Transmit Power:
  + Class 4 (2W) for GSM850
  + Class 1 (1W) for DCS1800
* Serial-based AT Command Set
* FL connectors for cell antennae
* Accepts Micro SIM Card

LED Status Indicators

There is an LED on the top right side of the SIM800L Cellular Module which indicates the status of your cellular network. It’ll blink at various rates to show what state it’s in:



Blink every 1s

The module is running but hasn’t made connection to the cellular network yet.



Blink every 2s

The GPRS data connection you requested is active.



Blink every 3s

The module has made contact with the cellular network & can send/receive voice and SMS.

## Selecting Antenna

An antenna is required to use the module for any kind of voice or data communications as well as some SIM commands. So, selecting an antenna could be a crucial thing. There are two ways you can add an antenna to your SIM800L module.

The first one is a Helical GSM antenna which usually comes with the module and solders directly to NET pin on PCB. This antenna is very useful for projects that need to save space but struggles in getting connectivity especially if your project is indoors.



The second one is any 3dBi GSM antenna along with a U.FL to SMA adapter which can be obtained online for less than $3. You can snap-fit this antenna to small u.fl connector located on the top-left corner of the module. This type of antenna has a better performance and allows putting your module inside a metal case – as long the antenna is outside.



## Supplying Power for SIM800L module

One of the most important parts of getting the SIM800L module working is supplying it with enough power.

Depending on which state it’s in, the SIM800L can be a relatively power-hungry device. The maximum current draw of the module is around 2A during transmission burst. It usually won’t pull that much, but may require around 216mA during phone calls or 80mA during network transmissions. This chart from the datasheet summarizes what you may expect:

|  |  |  |
| --- | --- | --- |
| Modes | Frequency | Current Consumption |
| Power down |  | 60 uA |
| Sleep mode |  | 1 mA |
| Stand by |  | 18 mA |
| Call | GSM850 | 199 mA |
| EGSM900 | 216 mA |
| DCS1800 | 146 mA |
| PCS1900 | 131 mA |
| GPRS |  | 453 mA |
| Transmission burst |  | 2 A |

Since SIM800L module doesn’t come with onboard voltage regulator, an external power supply adjusted to voltage between 3.4V to 4.4V (Ideal 4.1V) is required. The power supply should also be able to source 2A of surge current, otherwise the module will keep shutting down. Here are some options you can consider to correctly power your GSM module.

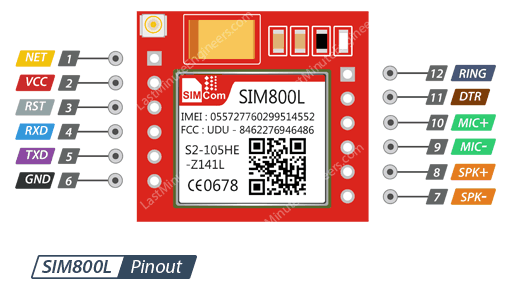
### DC-DC Buck Converter

Any 2A-rated DC-DC buck converter like **LM2596** would work. These are much more efficient than a liner voltage regulator like LM317 or LM338.



## SIM800L GSM Module Pinout

The SIM800L module has total 12 pins that interface it to the outside world. The connections are as follows:



NET is a pin where you can solder Helical Antenna provided along with the module.

VCC supplies power for the module. This can be anywhere from 3.4V to 4.4 volts. Remember connecting it to 5V pin will likely destroy your module! It doesn’t even run on 3.3 V! An external power source like Li-Po battery or DC-DC buck converters rated 3.7V 2A would work.

RST (Reset) is a hard reset pin. If you absolutely got the module in a bad space, pull this pin low for 100ms to perform a hard reset.

RxD (Receiver) pin is used for serial communication.

TxD (Transmitter) pin is used for serial communication.

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

RING pin acts as a Ring Indicator. It is basically the ‘interrupt’ out pin from the module. It is by default high and will pulse low for 120ms when a call is received. It can also be configured to pulse when an SMS is received.

DTR pin activates/deactivates sleep mode. Pulling it HIGH will put module in sleep mode, disabling serial communication. Pulling it LOW will wake the module up.

MIC± is a differential microphone input. The two microphone pins can be connected directly to these pins.

SPK± is a differential speaker interface. The two pins of a speaker can be tied directly to these two pins.

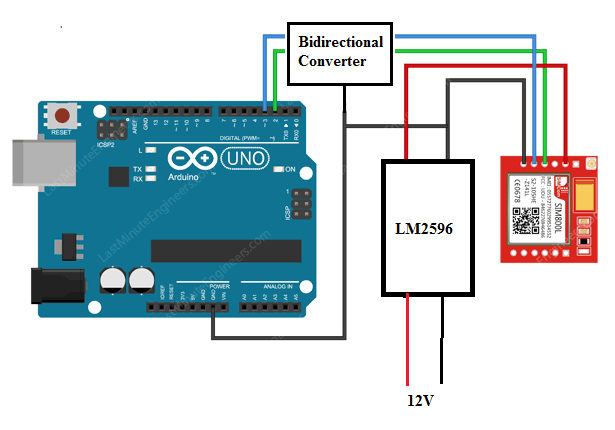
## Wiring – Connecting SIM800L GSM module to Arduino UNO

Now that we know everything about the module, we can begin hooking it up to our Arduino!

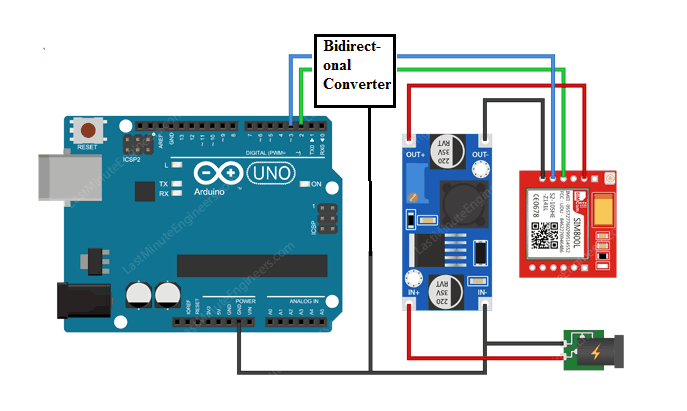
Start by soldering/connecting the antenna, insert fully activated Micro SIM card in the socket. Now, connect Tx pin on module to digital pin#3 on Arduino as we’ll be using [software serial](https://www.arduino.cc/en/Reference/softwareSerial) to talk to the module.

We cannot directly connect Rx pin on module to Arduino’s digital pin as Arduino Uno uses 5V GPIO whereas the SIM800L module uses 3.3V level logic and **is NOT 5V tolerant**. This means the Tx signal coming from the Arduino Uno must be stepped down to 3.3V so as not to damage the SIM800L module. There are several ways to do this but the easiest way is to use a simple resistor divider. A 10K resistor between SIM800L Rx and Arduino D2, and 20K between SIM800L Rx and GND would work fine.

Now we are remaining with the pins that are used to supply power for the module. As you have multiple choices for powering up the module, we have provided two example schematics. The one uses 1200mAh Li-Po battery and other one uses LM2596 DC-DC buck converter.



Wiring SIM800L GSM GPRS Module with Arduino UNO



Wiring SIM800L GSM GPRS Module with Arduino UNO

### TIP

In case you are using LM2596 buck converter to power up the module, remember to common all the ground in the circuit.

Once you have everything hooked up you are ready to go!

## Arduino Code – Testing AT Commands

For sending AT commands and communicating with the SIM800L module, we will use the serial monitor. The sketch below will enable the Arduino to communicate with the SIM800L module on serial monitor. Before we proceed with detailed breakdown of code, connect your Arduino to PC, compile below code and upload it to the Arduino.

Once you open a serial monitor,**make sure that ‘Both NL & CR’ option is selected!**

#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

}

}

The sketch starts by including a **SoftwareSerial.h** library and initializing it with the Arduino pins to which Tx and Rx of SIM800L module is 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 setup function: we initialize a serial communication link between Arduino, Arduino IDE and 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** – It is the most basic AT command. It also initializes Auto-baud’er. If it works you should see the AT characters echo and then OK, telling you it’s OK and it’s understanding you correctly! You can then send some commands to query the module and get information about it such as

**AT+CSQ** – Check the ‘signal strength’ – the first # is dB strength, it should be higher than around 5. Higher is better. Of course it depends on your antenna and location!

**AT+CCID** – get the SIM card number – this tests that the SIM card is found OK and you can verify the number is written on the card.

**AT+CREG?** Check that you’re registered on the network. The second # should be 1 or 5. 1 indicates you are registered to home network and 5 indicates roaming network. Other than these two numbers indicate 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 looping part of the code, we call custom function called **updateSerial()**which continuously waits for any inputs from the serial monitor and send it to the SIM800L module through the D2 pin (Rx of module). It also continuously reads the D3 pin (Tx of module) if the SIM800L module has any responses.

void updateSerial()

{

delay(500);

while (Serial.available())

{

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

}

while(mySerial.available())

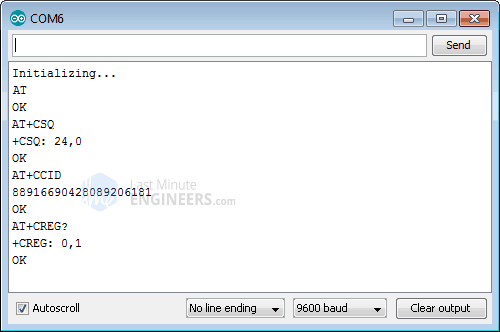
{

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

}

}

You should see below output on serial monitor.

Basic AT Commands on SIM800L GSM Module

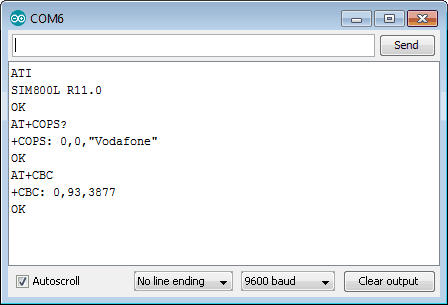
You are now free to send any commands through serial monitor like below which gives more information about network connection & battery status:

**ATI** – Get the module name and revision

**AT+COPS?** – Check that you’re connected to the network, in this case BSNL

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

**AT+CBC** – will return the lipo battery state. The second number is the % full (in this case its 93%) and the third number is the actual voltage in mV (in this case, 3.877 V)

Network Connection AT Commands on SIM800L GSM Module

## Arduino Code – Sending SMS

Let’s move on to the interesting stuff. Let’s program our Arduino to send an SMS to any phone number you wish. Before trying the sketch out, you need to enter the phone number. Search for the string ZZxxxxxxxxxx and replace ZZ with 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, it will back to OK

updateSerial();

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

updateSerial();

mySerial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");//change ZZ with country code and xxxxxxxxxxx with phone number to sms

updateSerial();

mySerial.print("Last Minute Engineers | lastminuteengineers.com"); //text content

updateSerial();

mySerial.write(26);

}

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

**AT+CMGF=1** – Selects SMS message format as text. Default format is [**P**rotocol **D**ata **U**nit](https://en.wikipedia.org/wiki/Protocol_data_unit) (PDU)

**AT+CMGS=+ZZxxxxxxxxxx** – Sends SMS to the phone number specified. The text message entered followed by a ‘Ctrl+z’ character is treated as SMS. ‘Ctrl+z’ is actually a 26th non-printing character described as ‘substitute’ in [ASCII table](https://www.asciitable.com/). So, we need to send 26DEC (1AHEX) once we send a message.

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

updateSerial();

mySerial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");//change ZZ with country code and xxxxxxxxxxx with phone number to sms

updateSerial();

mySerial.print("Last Minute Engineers | lastminuteengineers.com"); //text content

updateSerial();

mySerial.write(26);

The loop is kept empty as we want to send SMS only once. If you wish to send SMS one more time, just hit the RESET key on your Arduino. Below screenshot shows SMS sent from SIM800L GSM module.

## Arduino Code – Reading SMS