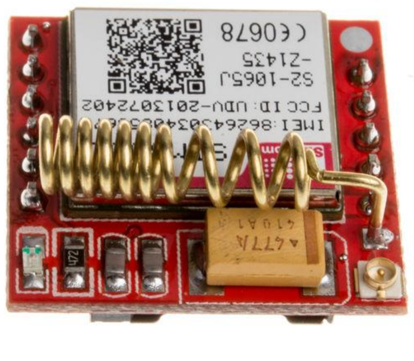
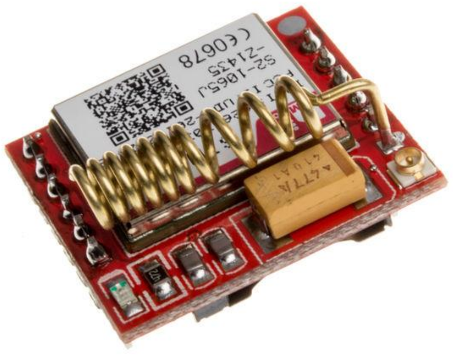
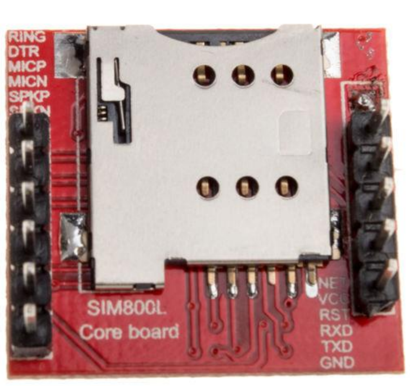
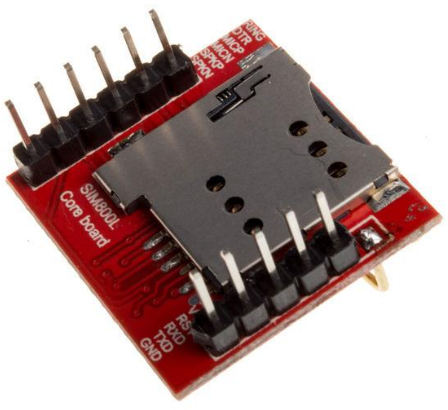
SIM800L GSM GPRS Module Breakout Board

Introductiom

Mini GSM / GPRS breakout board is based on SIM800L module, supports quad-band GSM/GPRS network, available for GPRS and SMS message data remote transmission.The board features compact size and low current consumption. With power saving technique, the current consumption is as low as 1mA in sleep mode.It communicates with microcontroller via UART port, supports command including 3GPP TS 27.007, 27.005 and SIMCOM enhanced AT Commands. It's operating voltage: 3.7 ~ 4.2V , peak current: 1A .



Features

- Quad-band 850/900/1800/1900MHz
- Connect onto any global GSM network with any 2G SIM (in the USA, T- Mobile is suggested).
- Make and receive voice calls using a headset or an external 8 speaker and electret microphone.
- Send and receive SMS messages.
- Send and receive GPRS data (TCP/IP, HTTP, etc.) .
- Scan and receive FM radio broadcasts.
- Lead out buzzer and vibrational motor control port.
- AT command interface with "auto baud" detection.

QUICKSTART SIM800L WITH ARDUINO

SIM800 is one of the most commonly used GSM module among hobbyists and Arduino community. Even though AT command reference is available with a quick Google search, it is not very easy for a beginner to properly understand and use Arduino with SIM800. Therefore, this post summarizes how a beginner could interact with SIM800 using Arduino and in few future posts we’ll be going ahead with several other real life use cases discussing how SIM800 can be used with Arduino effectively.

Wiring

Caution! :SIM800 module itself uses input voltage of 3.7V – 4.2V. Therefore, if you directly connect the Vcc pin to Arduino 5V pin, it might damage the module as well. I tried 3.3V Arduino pin with no luck. Therefore if you have the SIM800 module itself, you will have to get the input voltage in range, maybe with a LM317 (and few calculations with a LM317 calculator).

- SIM800 5v/4v ↔ Arduino 5v
- SIM800 GND (either one) ↔ Arduino GND
- SIM800 SIM_TXD ↔ Arduino D8 (read through for the reason)
- SIM800 SIM_RXD ↔ Arduino D7 (read through for the reason)

Connect module with Arduino as mentioned above or with any changes depending on the module being used. Insert a valid SIM card and connect the Arduino with Arduino IDE.

Simple Serial Communication

Below is the simplest program usable to interact with SIM800 :

```
#include <SoftwareSerial.h>

//SIM800 TX is connected to Arduino D8
#define SIM800_TX_PIN 8

//SIM800 RX is connected to Arduino D7
#define SIM800_RX_PIN 7

//Create software serial object to communicate with SIM800
SoftwareSerial serialSIM800(SIM800_TX_PIN,SIM800_RX_PIN);

void setup() {
  //Begin serial comunication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);
  while(!Serial);

  //Being serial communication witj Arduino and SIM800
  serialSIM800.begin(9600);
  delay(1000);

  Serial.println("Setup Complete!");
}

void loop() {
  //Read SIM800 output (if available) and print it in Arduino IDE Serial Monitor
  if(serialSIM800.available()){
    Serial.write(serialSIM800.read());
  }
  //Read Arduino IDE Serial Monitor inputs (if available) and send them to SIM800
  if(Serial.available()){
    serialSIM800.write(Serial.read());
  }
}
```

Upload above code to the Arduino (Code itself is self explanatory. Hence, will not repeat same). Once upload is complete start the Arduino Serial Monitor from Tools menu. Set Baud rate to 9600 and **in drop down left to boud rate selection set “Both NL and CR”.**

Once done, you can freely send AT commands to SIM800 and see the output in real time. Few examples :

AT – is to check if interface is working fine.
AT+CFUN – is used to set phone functionality
AT+CFUN? – returns currently set value for AT+CFUN
AT+CFUN=? – returns all possible values that can be set for AT+CFUN (similar to help)
AT+CFUN=1 – is to sent AT+CFUN to 1 (full functionality)
AT+CREG? – to get network registration information. stat=1 means you are registered with home network
AT+COPS? – returns currently registered operator details
AT+COPS=? – returns all the operators available

Sending SMS with SoftwareSerial

In below code delay of 1 second is used after each command to give necessary time for SIM800 to respond to each command. With this approach it is not possible to clearly identify any ERROR conditions because program will not ready the responses sent. Proper method of doing this is by checking each response against an expected value. This is already handled in most of the Arduino libraries including “Seeeduino” library we’ll be using below. Hence, for this stage, 1 second delay is used for the sake of simplicity.

Note : Replace 07194XXXXX with mobile number SMS should be sent to.

```
#include <SoftwareSerial.h>

//SIM800 TX is connected to Arduino D8
#define SIM800_TX_PIN 8

//SIM800 RX is connected to Arduino D7
#define SIM800_RX_PIN 7

//Create software serial object to communicate with SIM800
SoftwareSerial serialSIM800(SIM800_TX_PIN,SIM800_RX_PIN);

void setup() {
  //Begin serial comunication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);
  while(!Serial);

  //Being serial communication witj Arduino and SIM800
  serialSIM800.begin(9600);
  delay(1000);

  Serial.println("Setup Complete!");
  Serial.println("Sending SMS...");
  //Set SMS format to ASCII
  serialSIM800.write("AT+CMGF=1\r\n");
  delay(1000);

  //Send new SMS command and message number
  serialSIM800.write("AT+CMGS=\"07194XXXXX\"\r\n");
  delay(1000);

  //Send SMS content
  serialSIM800.write("TEST");
  delay(1000);

  //Send Ctrl+Z / ESC to denote SMS message is complete
  serialSIM800.write((char)26);
  delay(1000);

  Serial.println("SMS Sent!");
}

void loop() {
}
```

SIM800 Libraries

With a quick Google search you will be able to find several SIM800 Arduino libraries. After going through source codes of several libraries my selection was “Seeeduino_GPRS” library which provides basic SIM800 features as well as additional set of GPRS related features.

Sending SMS with Seeeduino Arduino library

Note : Seeeduino library assumes that TX connected to D8 and RX is connected to D7 on Arduino. This is the reason we used relevant pins at first place. If you need to connect SIM800 with any other Arduino pin, you will have to modify the library source (gprs.h) and add a new constructor. Library uses MIT license.

- Clear steps relevant to installing Arduino library is available at : https://www.arduino.cc/en/Guide/Libraries
- Seeeduino_GPRS library is available for download at : https://github.com/Seeed-Studio/Seeeduino_GPRS

Once library is installed in Arduino IDE File menu, Examples section you will find “Seeeduino_GPRS” library and withing examples you will find “GPRS_SendSMS” example which reads as follows :

```
/*
Sketch: GPRS Connect TCP

Function: This sketch is used to test seeeduino GPRS's send SMS func.to make it
work,
you should insert SIM card to Seeeduino GPRS and replace the phoneNumber,enjoy it!
*********************************************************************************
*
note: the following pins has been used and should not be used for other purposes.
  pin 8   // tx pin
  pin 7   // rx pin
  pin 9   // power key pin
  pin 12  // power status pin
*********************************************************************************
*
created on 2013/12/5, version: 0.1
by lawliet.zou(lawliet.zou@gmail.com)
*/
#include <gprs.h>;
#include <SoftwareSerial.h>;

GPRS gprs;

void setup() {
  Serial.begin(9600);
  while(!Serial);
  Serial.println("GPRS - Send SMS Test ...");
  gprs.preInit();
  delay(1000);
  while(0 != gprs.init()) {
      delay(1000);
      Serial.print("init error\r\n");
  }
  Serial.println("Init success, start to send SMS message...");
  gprs.sendSMS("07194XXXXX","hello,world"); //define phone number and text
}
 void loop() {
  //nothing to do
}
```

If you go through Seeeduino library you will notice that it is possible to send commands directly for any advanced use cases. For examples there are library methods such as :

- sendCmdAndWaitForResp(const char* cmd, const char *expectedResp, unsigned timeout)
- sendCmd(const char* cmd)
- waitForResp(const char *resp, unsigned int timeout)

Hence, you could simply correctly rewrite the SMS sending application as below (reinvent the wheel) :

```
if(0 != gprs.sendCmdAndWaitForResp("AT+CMGF=1\r\n", "OK", DEFAULT_TIMEOUT)) { //
Set message mode to ASCII
    ERROR("ERROR:CMGF");
    return;
}
delay(500);
if(0 !=
gprs.sendCmdAndWaitForResp("AT+CMGS=\"07194XXXXX\"\r\n",">",DEFAULT_TIMEOUT)) {
    ERROR("ERROR:CMGS");
    return;
}
delay(1000);
gprs.serialSIM800.write(data);
delay(500);
gprs.serialSIM800.write((char)26);
return;
```