

The GPRS Shield is based on SIM900 module from SIMCOM and compatible with Arduino and its clones. The GPRS Shield provides you a way to communicate using the GSM cell phone network. The shield allows you to achieve SMS, MMS, GPRS and Audio via UART by sending AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands). The shield also has the 12 GPIOs, 2 PWMs and an ADC of the SIM900 module(They are all 2V8 logic) present onboard.

Features

- Quad-Band 850 / 900/ 1800 / 1900 MHz - would work on GSM networks in all countries across the world.
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
- Class 4 (2 W @ 850 / 900 MHz)
- Class 1 (1 W @ 1800 / 1900MHz)
- Control via AT commands - Standard Commands: GSM 07.07 & 07.05 | Enhanced Commands: SIMCOM AT Commands.
- Short Message Service - so that you can send small amounts of data over the network (ASCII or raw hexadecimal).
- Embedded TCP/UDP stack - allows you to upload data to a web server.
- RTC supported.
- Selectable serial port.
- Speaker and Headphone jacks
- Low power consumption - 1.5mA(sleep mode)
- Industrial Temperature Range - -40°C to +85 °C

Specifications

Application Ideas

- M2M (Machine 2 Machine) Applications.
- Remote control of appliances.
- Remote Weather station or a Wireless Sensor Network.
- Vehicle Tracking System with a GPS module.

Cautions

- Make sure your SIM card is unlocked.
- The product is provided as is without an insulating enclosure. Please observe ESD precautions specially in dry (low humidity) weather.
- The factory default setting for the GPRS Shield UART is 19200 bps 8-N-1. (Can be changed using AT commands).

Also, in the serial monitor you should see messages from the shield such as RDY

+CFUN: 1

+CPIN: READY

Call Ready

If you can not see the messages in the serial monitor, you should click the "send new" option that will add carriage return at the end of AT command and then send AT command "AT+IPR=19200" to set the baud rate of the SIM900.



▪ Step 2: Sending a text message (SMS)

Now that our test setup is ready, let's play around with some AT Commands manually before moving on to programming the Arduino to do this. Let's try sending an SMS to start.

1. .Create the setup as described in Step 1 above.
2. .Through your serial terminal software, send AT+CMGF=1 and press the Enter key. The GPRS Shield can send SMSes in two modes: Text mode and PDU (or binary) mode. Since we want to send out a human readable message, we will select the text mode. The GPRS Shield will respond with an OK.
3. .Click "send new" option and send AT+CMGS="+918446043032". This will instruct the GPRS Shield to start accepting text for a new message meant for the phone number specified (replace the number with the phone number of the target phone). The GPRS Shield will send a > to remind you typing the message.



1. Start typing your message and when you are done, and click "send hex" option and then send a hex: 1A. The modem will accept the message and respond with an OK. A few moments later, the message should be received on the handset whose number you had specified. You can refer to the picture below.



NOTE: If, in spite of following the steps as specified above, you aren't able to receive the message on the target handset, then it might be that you need to set the SMS Message Center number. Send the command `AT+CSCA="+919032055002"` and press the Enter Key. Send this command in between the `AT+CMGF` and `AT+CMGS` commands. Replace the phone number specified in the command above with the SMS Center number of your GSM Service Provider. The message center number is specific to each service provider (for example +919032055002 is the message center number for Tata DoCoMo, Pune,

India). You can get the message center number by calling up the customer care center of the GSM Service Provider and asking them for it.

SoftwareSerial library Notes

With Arduino 1.0 you should be able to use the SoftwareSerial library included with the distribution (instead of NewSoftSerial). However, you must be aware that the buffer reserved for incoming messages are hardcoded to 64 bytes in the library header, "SoftwareSerial.h": `1.define _SS_MAX_RX_BUFF 64 //` RX buffer size

This means that if the GPRS module responds with more data than that, you are likely to lose it with a buffer overflow! For instance, reading out an SMS from the module with "AT+CMGR=xx" (xx is the message index), you might not even see the message part because the preceding header information (like telephone number and time) takes up a lot of space. The fix seems to be to manually change `_SS_MAX_RX_BUFF` to a higher value (but reasonable so you don't use all your precious memory!)

The Softwareserial library has the following limitations (taken from arduino page) If using multiple software serial ports, only one can receive data at a time. <http://arduino.cc/hu/Reference/SoftwareSerial> This means that if you try to add another serial device ie grove serial LCD you may get communication errors unless you craft your code carefully.

A Simple Source Code Examples

The demo code below is for the Xduino to send SMS message/dial a voice call/submit a http request to a website and upload datas to the pachube. It has been tested on Arduino Duemilanove but will work on any compatible variant, please note that this sketch uses the software UART of ATmega328P. please follow the following steps for running this sketch.

1. .With the GPRS Shield removed, download this sketch into your Arduino.
2. .Disconnect the Xduino from USB port to remove power source.
3. .Set the Serial Port jumpers on the GPRS Shield in SWserial position, to use the Soft Serial port of Arduino.
4. .Connect the antenna to the GPRS Shield and insert the SIM Card.
5. .Mount the GPRS Shield on Arduino.
6. .Connect the Arduino to the computer by USB, and fire up your favorite serial terminal software on computer, choose the COM port for Arduino, set it to operate at 19200 8-N-1.
7. .Type command in the terminal to execute different function, there are 4 functions in the demo:
 1. .If you input 't', the demo will send a SMS message to another cellphone which you set(you need set the number in the code);
 2. .If you input 'd', the program will dial a call to the other cellphone that you set(it is also need you set in the code);
 3. .If you input 'h', it will submit a http request to a web that you want to access(it need you set the web adress in the code), it will return a string from the website if it goes correctly;
 4. .If you input 's', it will upload the datas to the pachube(for detail you can refer to the explanation in the code). I strongly recommend you input 'h' before input 's', because

uploading datas to the pachube need do some setting, after execute the function of submit a http request, the setting will be set.

8. If the program returns error in the terminal after you typed the command, don't worry, just try input the command again.

```
/*Note: this code is a demo for how to using gprs shield to send sms message,
dial a voice call and
send a http request to the website, upload data to pachube.com by TCP connection,

The microcontrollers Digital Pin 7 and hence allow unhindered
communication with GPRS Shield using SoftSerial Library.
IDE: Arduino 1.0 or later
Replace the following items in the code:
1.Phone number, don't forget add the country code
2.Replace the Access Point Name
3. Replace the Pachube API Key with your personal ones assigned
to your account at cosm.com
*/
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial mySerial(7, 8);
void setup()
{
    mySerial.begin(19200);          // the GPRS baud rate
    Serial.begin(19200);           // the GPRS baud rate
    delay(500);
}
void loop()
{
    //after start up the program, you can using terminal to connect the serial of
    gprs shield,
    //if you input 't' in the terminal, the program will execute SendTextMessage(),
    it will show how to send a sms message,
    //if input 'd' in the terminal, it will execute DialVoiceCall(), etc.
    if (Serial.available())
        switch(Serial.read())
        {
            case 't':
                SendTextMessage();
                break;
            case 'r':
```

```

    RecieveTextMessage();//This program code by directive'r'to receive, by
receiving the information after the return to call the function
        //to verify receiving function. But it can not display the
received content in SIM.
    DialVoiceCall();
    break;
case 'd':
    DialVoiceCall();
    break;
case 'h':
    SubmitHttpRequest();
    break;
case 's':
    Send2Pachube();
    break;
}
if (mySerial.available())
    Serial.write(mySerial.read());
}
///SendTextMessage()
///this function is to send a sms message
void SendTextMessage()
{
    mySerial.print("AT+CMGF=1\r");    //Because we want to send the SMS in text mode
    delay(100);
    mySerial.println("AT + CMGS = \""+86138xxxxx615\"); //send sms message, be
careful need to add a country code before the cellphone number
    delay(100);
    mySerial.println("A test message!"); //the content of the message
    delay(100);
    mySerial.println((char)26); //the ASCII code of the ctrl+z is 26
    delay(100);
    mySerial.println();
}
void RecieveTextMessage()
{
    //mySerial.print("AT+CMGF=1\r"); //Because we want to recieve the SMS in text
mode
    //delay(100);
    mySerial.print("AT+CMGR=1\r"); //Because we want to recieve the SMS in text mode
    delay(100);
    mySerial.println("AT + CSCA = \""+86135*****\"); //recieve sms message, be
careful need to add a country code before the cellphone number

```

```

    delay(100);
    mySerial.println("A test message!"); //the content of the message
    delay(100);
    mySerial.println((char)26); //the ASCII code of the ctrl+z is 26
    delay(100);
    mySerial.println();
    //return r;
}
///DialVoiceCall
///this function is to dial a voice call
void DialVoiceCall()
{
    mySerial.println("ATD + +86138xxxxx615;"); //dial the number
    delay(100);
    mySerial.println();
}
///SubmitHttpRequest()
///this function is submit a http request
///attention:the time of delay is very important, it must be set enough
void SubmitHttpRequest()
{
    mySerial.println("AT+CSQ");
    delay(100);
    ShowSerialData(); // this code is to show the data from gprs shield, in order to
    easily see the process of how the gprs shield submit a http request, and the
    following is for this purpose too.
    mySerial.println("AT+CGATT?");
    delay(100);
    ShowSerialData();
    mySerial.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"); //setting the SAPBR,
    the connection type is using gprs
    delay(1000);
    ShowSerialData();
    mySerial.println("AT+SAPBR=3,1,\"APN\",\"CMNET\"); //setting the APN, the
    second need you fill in your local apn server
    delay(4000);
    ShowSerialData();
    mySerial.println("AT+SAPBR=1,1"); //setting the SAPBR, for detail you can refer
    to the AT command mamual
    delay(2000);
    ShowSerialData();
    mySerial.println("AT+HTTPINIT"); //init the HTTP request
    delay(2000);

```

```

ShowSerialData();
mySerial.println("AT+HTTPPARA=\"URL\", \"www.google.com.hk\"); // setting the
httppara, the second parameter is the website you want to access
delay(1000);
ShowSerialData();
mySerial.println("AT+HTTPACTION=0");//submit the request
delay(10000);//the delay is very important, the delay time is base on the return
from the website, if the return datas are very large, the time required longer.
//while(!mySerial.available());
ShowSerialData();
mySerial.println("AT+HTTPREAD");// read the data from the website you access
delay(300);
ShowSerialData();
mySerial.println("");
delay(100);
}
///send2Pachube()///
///this function is to send the sensor data to the pachube, you can see the new
value in the pachube after execute this function///
void Send2Pachube()
{
mySerial.println("AT+CGATT?");
delay(100);
ShowSerialData();
mySerial.println("AT+CSTT=\"CMNET\");//start task and setting the APN,
delay(1000);
ShowSerialData();
mySerial.println("AT+CIICR");//bring up wireless connection
delay(300);
ShowSerialData();
mySerial.println("AT+CIFSR");//get local IP adress
delay(2000);
ShowSerialData();
mySerial.println("AT+CIPSPRT=0");
delay(3000);
ShowSerialData();
mySerial.println("AT+CIPSTART=\"tcp\", \"api.cosm.com\", \"8081\");//start up
the connection
delay(2000);
ShowSerialData();
mySerial.println("AT+CIPSEND");//begin send data to remote server
delay(4000);
ShowSerialData();

```

```

    String humidity = "1031";//these 4 line code are imitate the real sensor data,
because the demo didn't add other sensor, so using 4 string variable to replace.
    String moisture = "1242";//you can replace these four variable to the real sensor
data in your project
    String temperature = "30";//
    String barometer = "60.56";//
    mySerial.print("{\"method\": \"put\", \"resource\":
\"/feeds/42742/\", \"params\": \"\"}); //here is the feed you apply from pachube
    delay(500);
    ShowSerialData();
    mySerial.print(": {}, \"headers\": {\"X-PachubeApiKey\": \"\"}); //in here, you
should replace your pachubeapikey
    delay(500);
    ShowSerialData();
    mySerial.print("
\"_cXwr5LE8qW4a2960-cDwOUvfddFer5pGmaRigPsi00\"");//pachubeapikey
    delay(500);
    ShowSerialData();

mySerial.print("jEB90jK-W6vej56j9ItaSlIac-hgbQjxExuveD95yc8BttXc");//pachube
apikey
    delay(500);
    ShowSerialData();
    mySerial.print("Z7_seZqLVjeCOMNbEXUva45t6FL8AxOcuNSsQS\", \"body\": \"\");
    delay(500);
    ShowSerialData();
    mySerial.print(" {\"version\": \"1.0.0\", \"datastreams\": \"\");
    delay(500);
    ShowSerialData();
    mySerial.println("[{\"id\": \"01\", \"current_value\": \"\" + barometer +
\"\"}, \"\");
    delay(500);
    ShowSerialData();
    mySerial.println("{\"id\": \"02\", \"current_value\": \"\" + humidity + \"\"}, \"\");
    delay(500);
    ShowSerialData();
    mySerial.println("{\"id\": \"03\", \"current_value\": \"\" + moisture + \"\"}, \"\");
    delay(500);
    ShowSerialData();
    mySerial.println("{\"id\": \"04\", \"current_value\": \"\" + temperature +
\"\"}}\", \"token\": \"lee\"}");
    delay(500);
    ShowSerialData();

```

```
mySerial.println((char)26);//sending
delay(5000);//waitting for reply, important! the time is base on the condition
of internet
mySerial.println();
ShowSerialData();
mySerial.println("AT+CIPCLOSE");//close the connection
delay(100);
ShowSerialData();
}
void ShowSerialData()
{
while(mySerial.available() !=0)
Serial.write(mySerial.read());
}
```