GPRS Shield V1.0

From Wiki

GPRS Shield : SLD33149P (http://www.seeedstudio.com/depot/gprs-shield-p-779.html)

Contents

Introduction

The GPRS Shield provides you a way to use the GSM cell phone network to receive data from a remote location. The shield allows you to achieve this via any of the three methods:

- Short Message Service
- Audio
- GPRS Service

The GPRS Shield is compatible with all boards which have the same form factor (and pinout) as a standard Arduino Board. The GPRS Shield is configured and controlled via its UART using simple AT commands. Based on the SIM900 module from SIMCOM, the GPRS Shield is like a cell phone. Besides the communications features, the GPRS Shield has 12 GPIOs, 2 PWMs and an ADC.

Model: SLD33149P (http://www.seeedstudio.com/depot/gprs-shield-p-779.html)

↑TOP

Features

- Based on SIMCom (http://wm.sim.com) 's SIM900 Module
- Quad-Band 850 / 900/ 1800 / 1900 MHz - would work on GSM networks in all countries across the world.
- 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.
- Speaker and Headphone jacks - so that you can send DTMF signals or play recording like an answering machine.
- SIM Card holder and GSM Antenna - present onboard.
- 12 GPIOs, 2 PWMs and an ADC (all 2.8 volt logic) - to augment your Arduino.

- Low power consumption - 1.5mA(sleep mode)
- Industrial Temperature Range - -40°C to +85 °C

Source(s): [GPRS Shield V1.0 Features support]

Application Ideas

- M2M (Machine 2 Machine) Applications - To transfer control data using SMS or GPRS between two machines located at two different factories.
- Remote control of appliances - Send SMS while you are at your office to turn on or off your washing machine at home.
- Remote Weather station or a Wireless Sensor Network - Mate it with Seeeduino Stalker and create a sensor node capable of transferring sensor data (like from a weather station - temperature, humidity etc.) to a web server (like pachube.com (http://www.pachube.com)).
- Interactive Voice Response System - Couple the GPRS Shield with an MP3 Decoder and DTMF Decoder (besides an Arduino) to create an Interactive Vocice Response System (IVRS) (http://en.wikipedia.org/wiki/Interactive_voice_response) .
- Vehicle Tracking System - Couple the GPRS Shield with an Arduino and GPS module and install it in your car and publish your location live on the internet. Can be used as a automotive burglar alarm.

Cautions

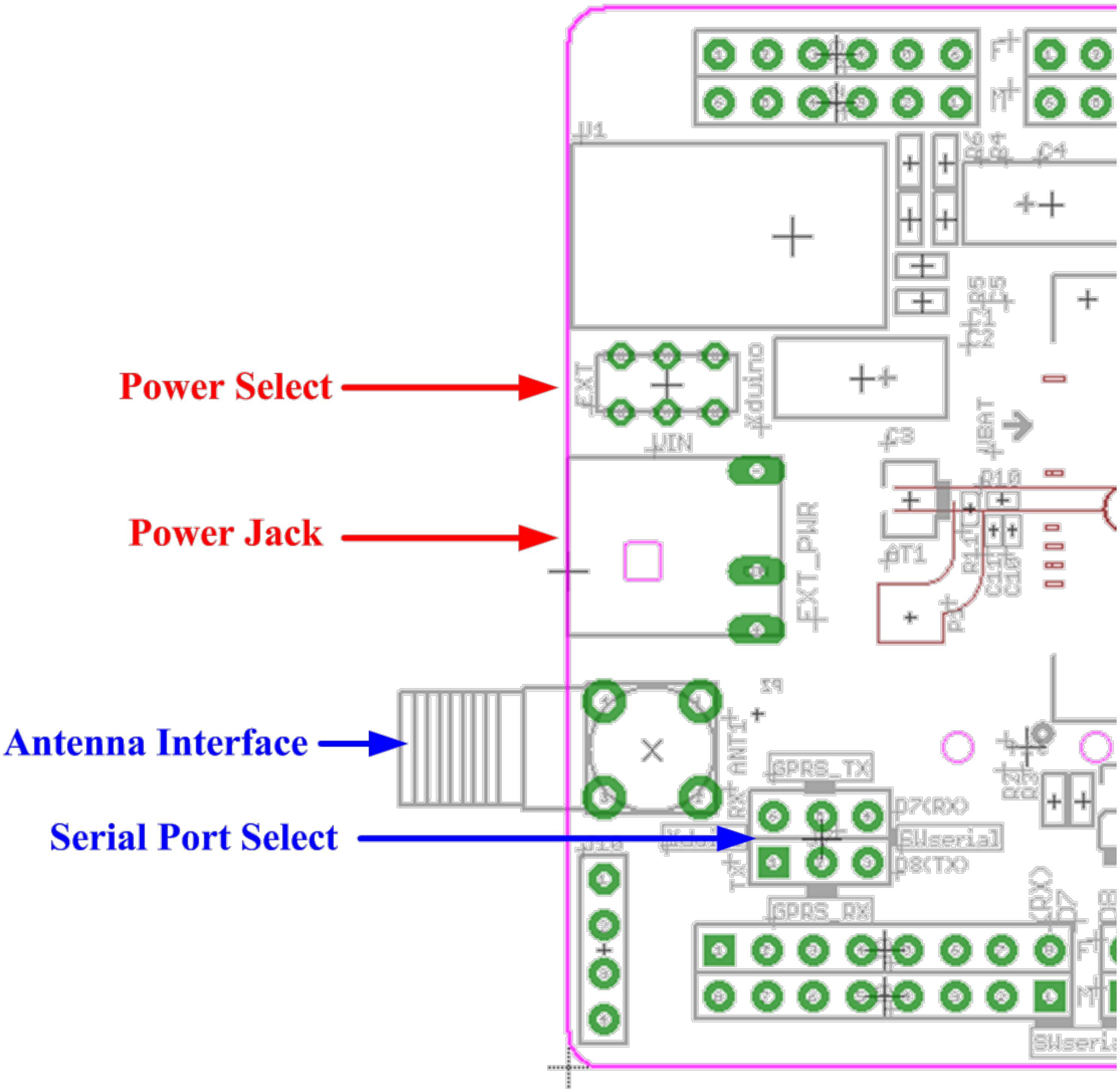
- Make sure your SIM card is unlocked.
- The product is provided as is without an insulating enclosure. Please observe ESD precautions specially in dry (low humidity) weather.
- The factory default setting for the GPRS Shield UART is 19200 bps 8-N-1. (Can be changed using AT commands).
- When using GPRS Shield with Seeeduino Stalker v2.0 please remember to dismount the **OK_READ** Jumper (i.e. open it). This will disconnect the Battery Charger IC's OK pin from the microcontrollers Digital Pin 7 and hence allow unhindered communication with GPRS Shield using NewSoftSerial Library.

Specifications

For SIM900's Specifications, please refer this PDF file: SIM900_SPEC.pdf (http://garden.seeedstudio.com/images/0/0b/SIM900_SPEC.pdf)

Item	Min	Typical	Max	Unit
Voltage	4.8	5.0	5.2	VDC
Current	/	50	450	mA
Dimension(with antenna)	110x58x19			mm
Net Weight	76±2			g

Interface Function


Power Select
Power Jack
Antenna Interface
Serial Port Select
EXT
Xduino
VIN
EXT_PWR
ANT1
GPRS_TX
GPRS_RX
D7(RX)
D8(TX)
SWserial
VBAT
BT1

Power select - select the power supply for GPRS shield(external power or 5v of arduino)
Power jack - connected to external 4.8~5VDC power supply
Antenna interface - connected to external antenna
Serial port select - select either software serial port or hareware serial port to be connected to GPRS Shield
Hardware Serial - D0/D1 of Arduino/Seeeduino
Software serial - D7/D8 of Arduino/Seeeduino only
Status LED - tell whether the power of SIM900 is on
Net light - tell the status about SIM900 linking to the net
UART of SIM900 - UART pins breakout of SIM900
Microphone - to answer the phone call
Speaker - to answer the phone call
GPIO,PWM and ADC of SIM900 - GPIO,PWM and ADC pins breakout of SIM900
Power key - power up and down for SIM900

Pins usage on Arduino

D0 - Unused if you select software serial port to communicate with GPRS Shield
D1 - Unused if you select software serial port to communicate with GPRS Shield
D7 - Used if you select software serial port to communicate with GPRS Shield
D8 - Used if you select software serial port to communicate with GPRS Shield
D9 - Used for software control the power up or down of the SIM900

Note: A4 and A5 are connected to the I2C pins on the SIM900. The SIM900 however cannot be accessed via the I2C .

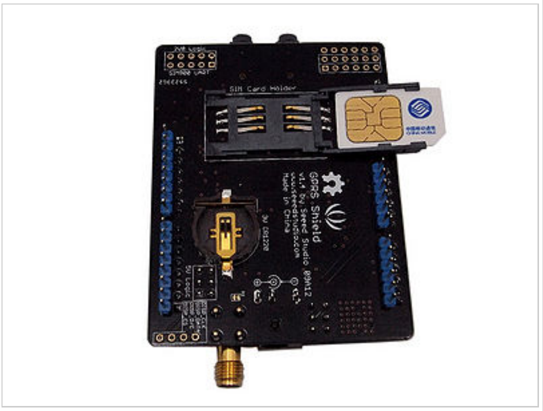
Usage

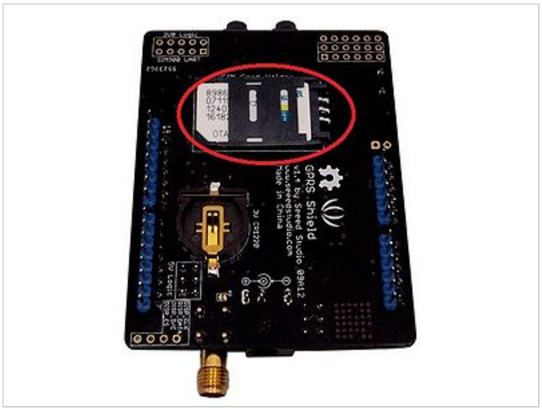
Light Status

LED	State	Function
Status	Off	Power Off
	On	Power On
Netlight	Off	SIM900 is not working
	64ms On/800ms Off	SIM900 does not find the network
	64ms On/3000ms Off	SIM900 find the network
	64ms On/300ms Off	GPRS communication

Hardware installation

- **Insert an unlocked SIM card to SIM Card Holder** - 6 Pin Holder for SIM Cards. Both 1.8 volts and 3.0 volts SIM Cards are supported by SIM900 - the SIM card voltage type is automatically detected.


Inserting the SIM Card into the holder


Locking the SIM Card Holder

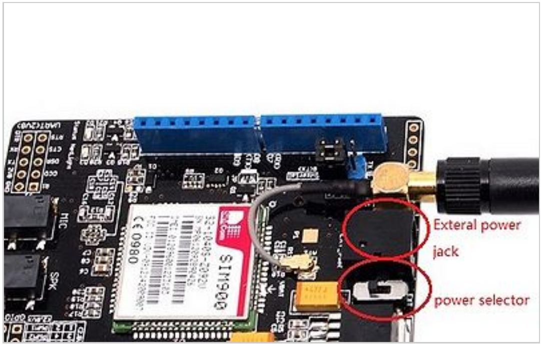
- **Make sure the antenna pad buckled properly** - A miniature coaxial RF connector is present on the GPRS Shield board to connect with a GSM Antenna. The connector present on the GPRS Shield is called a U.FL connecto (http://en.wikipedia.org/wiki/Hirose_U.FL)]. The GSM Antenna supplied with the GPRS Shield has an SMA connector (http://en.wikipedia.org/wiki/SMA_connector) (and not an RP-SMA connector) on it. A patch cord is also supplied with the GPRS Shield to interface the antenna to the board. The connection topology is shown in the diagram below:

Antenna pad properly buckled

- **Assemble the GSM antenna**

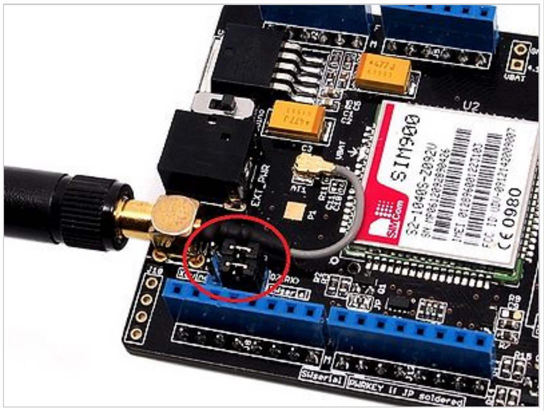
Assemble the GSM antenna

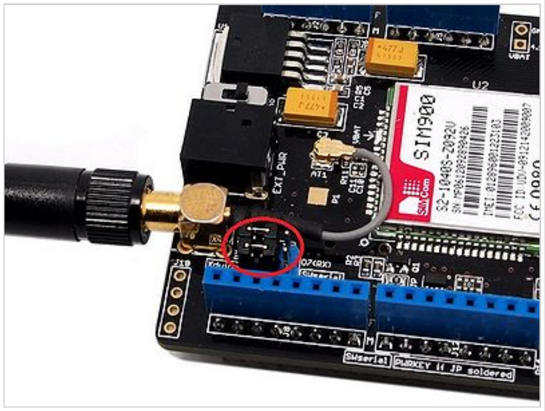
- **Power supply for GPRS shield** - Select power source with the switch on board, you can select the 5V power supply from arduino or exteral power.Select the 5V source from Arduino as the following picture:



select 5v of arduino

- **Communication port configure.**- The GPRS shield can be controlled via Hardware serial port of Arduino or Software serial port, Select the Software serial port with the jumpers:.


GPRS communicate with arduino by software serial


GPRS communicate with arduino by hardware serial

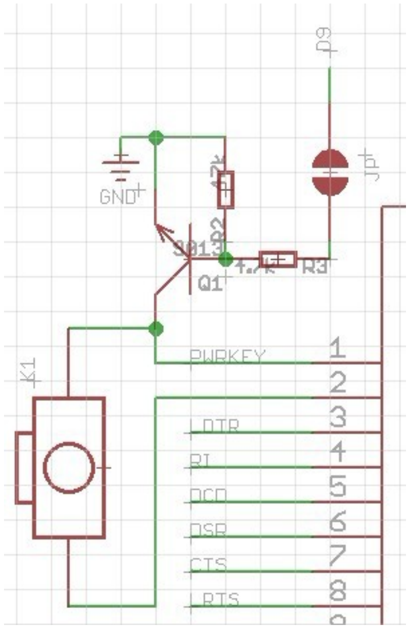
- **Plug to Arduino UNO R3** - The GPRS Shield, like any other well designed shield, is stackable as shown in the photo below.

GPRS Shield + Arduino UNO R3

Power up and power down

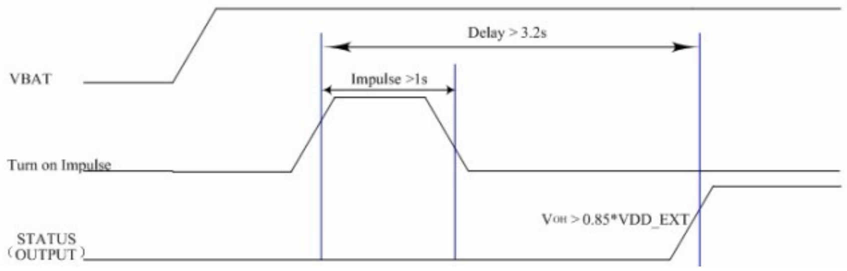
- **Hardware trigger**

Press the power key for about 2 seconds to power up or power down.



- **Software trigger**

You can power up/down the shield with Software, Control D9 of Arduino to add software triggering in your firmware.



As the timing of turn on/off , a >1s pulse was need to trigger the turning, and a >3.2s delay was need to get the timing stable. Add the following code in your firmware to turn on / off the shield without pressing the button:

```
void powerUpOrDown()
{
  pinMode(9, OUTPUT);
  digitalWrite(9,LOW);
  delay(1000);
  digitalWrite(9,HIGH);
  delay(2000);
  digitalWrite(9,LOW);
  delay(3000);
}
```

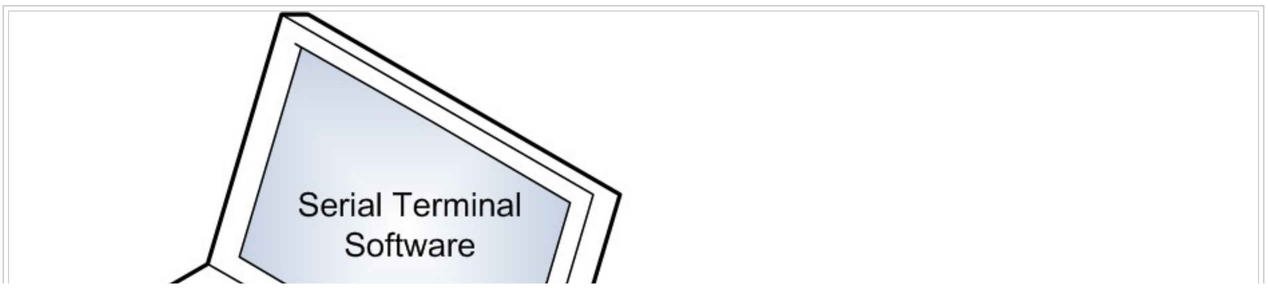
Getting Started - Fun with AT Commands

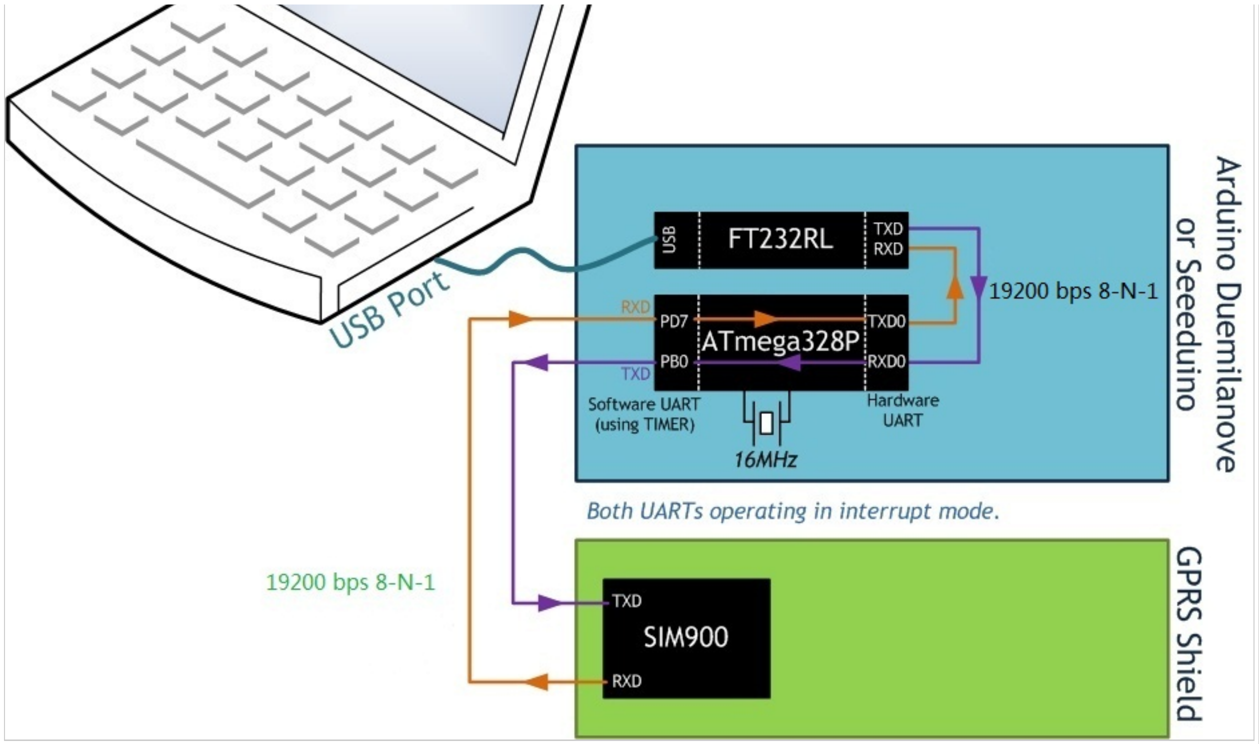
The GPRS Shield comes with all the accessories that you need to get started with sending data over the GSM network except an Arduino board and a GSM SIM Card. If you want to make voice calls, you would also require a headset with microphone.

- **Step 1: Creating a test setup for the GPRS Shield**

As you received your GPRS Shield, what would be the first thing you want to do with it? Send out a text (SMS)? or call up someone (headset required)? You can do all of this by talking to the GPRS Shield using AT Commands - which is a special language that it understands. AT Commands are simple textual commands sent to the GPRS modem over its serial interface (UART), so you can use any serial terminal software to communicate with it.
Note: Almost all the AT commands should be sent followed by **carriage return** and you need to select the "+CR"option in the serial port terminal.
To experiment with AT commands, you would require a way to power up and communicate with your GPRS Shield. The best way to do this using an Arduino Duemilanove board described below. The same steps are applicable for Seeeduino or Seeeduino Stalker.

- Follow the previously described hardware installation steps to set up the hardware system;
- Make sure the GPRS_TX & GPRS_RX jumpers on the GPRS Shield are mounted in SWSerial position - that is we want GPRS_TX to be connected to D7(RX) and GPRS_RX to D8(TX).
- Connect the Arduino Duemilanove (or Seeeduino) to your computer using a USB cable.
- The ATmega328P microcontroller on Duemilanove board has only one UART which is used for communicating with the PC. What we need is an Arduino Sketch running inside the ATmega328P that would emulate a second serial port (UART) using software on the digital pins D8 and D7 and patch through all the communication between this second software serial port and the actual hardware serial port. By doing this, all the data coming from the computer (connected to the actual hardware UART) would be routed to the GPRS Shield (connected to software UART) then, we would be able to issue AT commands to control the GPRS Shield. The block diagram outlining this scheme is shown below.



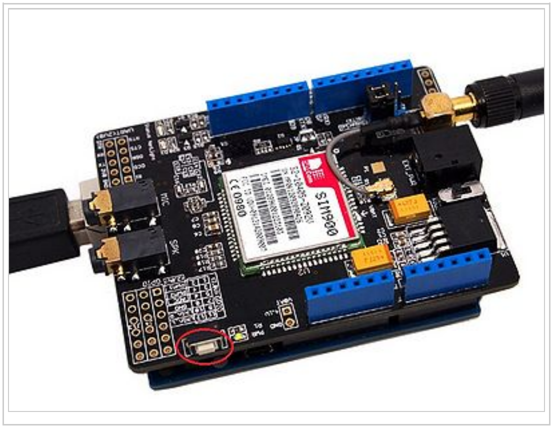


GPRS aurduino uart.jpg

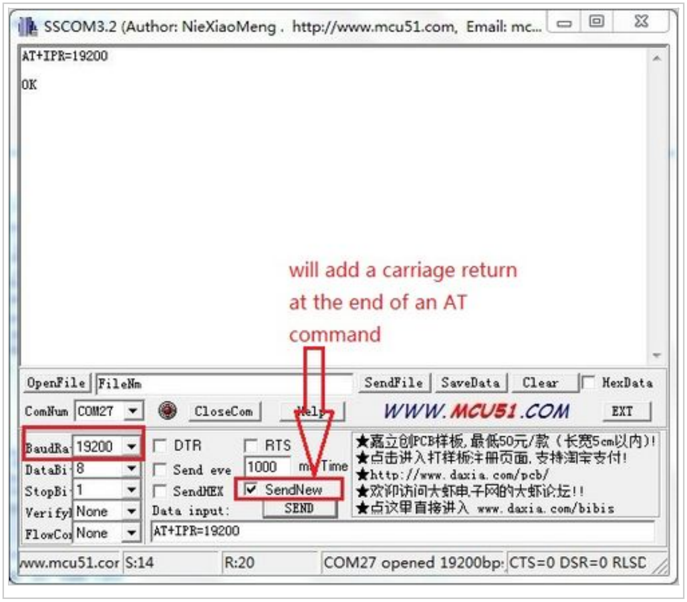
For developing such a program, we require to use the SoftwareSerial library included in the libraries of Arduino 1.0 already and the demo code below.

```
//Serial Relay - Arduino will patch a
//serial link between the computer and the GPRS Shield
//at 19200 bps 8-N-1
//Computer is connected to Hardware UART
//GPRS Shield is connected to the Software UART

#include <SoftwareSerial.h>

SoftwareSerial GPRS(7, 8);
unsigned char buffer[64]; // buffer array for data recieve over serial port
int count=0;     // counter for buffer array
void setup()
{
  GPRS.begin(19200);               // the GPRS baud rate
  Serial.begin(19200);             // the Serial port of Arduino baud rate.

}

void loop()
{
  if (GPRS.available())              // if date is comming from softwareserial port ==> data is comming from gprs shield
  {
    while(GPRS.available())          // reading data into char array
    {
      buffer[count++]=GPRS.read();     // writing data into array
      if(count == 64)break;
  }
    Serial.write(buffer,count);            // if no data transmission ends, write buffer to hardware serial port
    clearBufferArray();              // call clearBufferArray function to clear the storaged data from the array
    count = 0;                       // set counter of while loop to zero


  }
  if (Serial.available())            // if data is available on hardwareserial port ==> data is comming from PC or notebook
    GPRS.write(Serial.read());       // write it to the GPRS shield
}
void clearBufferArray()              // function to clear buffer array
{
  for (int i=0; i<count;i++)
    { buffer[i]=NULL;}                  // clear all index of array with command NULL
}
```

- Upload the sketch to the Arduino board.
- Download and fire up serial tool (http://www.seeedstudio.com/wiki/images/b/b2/Sscom32E.zip) if you don't have one. Choose the correct COM port for Arduino, and set it to operate at **19200** 8-N-1 and then click "Open Com".
- You can power on or off the the SIM900 by pressing the button about 2 second. After power on, the red LED will be on, and the green one beside it will blink and the shield has found the net if it blinks every 3 seconds.



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.



SSCOM3.2 (Author: NieXiaoMeng . http://www.mcu51.com, Email: mc...

```
AT+IPR=19200

OK
```

will add a carriage return at the end of an AT command

BaudRa 19200, DataBi 8, StopBi 1, Verifyl None, FlowCo None, ComNum COM27

☐ DTR ☐ RTS ☐ Send eve 1000 ms Time ☐ SendHEX ☑ SendNew

Data input: AT+IPR=19200

★嘉立创PCB样板,最低50元/款（长宽5cm以内)!
★点击进入打样板注册页面,支持淘宝支付!
★http://www.daxia.com/pcb/
★欢迎访问大虾电子网的大虾论坛!!
★点这里直接进入 www.daxia.com/bibis

www.mcu51.cor S:14 R:20 COM27 opened 19200bp: CTS=0 DSR=0 RLSD

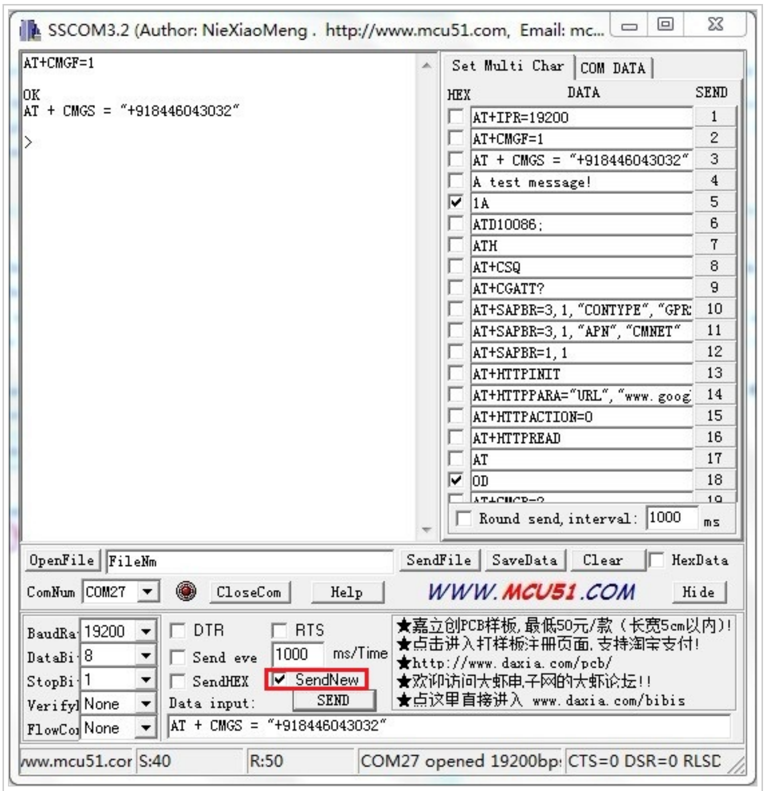
Note that the baud rate of GPRS shield and the Arduino's serial port which is connected to GPRS shield should be the same before you set the GPRS shield's baud rate.

- Now, click "send new" option and then type and send "AT" (without the quotes) to the Arduino board. The GPRS Shield should respond by sending back an "OK". This would mean that you have been able to successfully setup your GPRS Shield can can now play around with various AT Commands. (If you are using the readily available Serial Monitor in the Arduino IDE, you should set the line ending to "Carriage return" along with a baud rate of 19200).

***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 a
```

- Create the setup as described in Step 1 above.
- 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**.
- 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. Please note that phone numbers specified as parameters in any AT Command must be in E.123 format (http://en.wikipedia.org/wiki/E.123) .



```
SSCOM3.2 (Author: NieXiaoMeng . http://www.mcu51.com, Email: mc...

AT+CMGF=1

OK
AT + CMGS = "+918446043032"

>
```

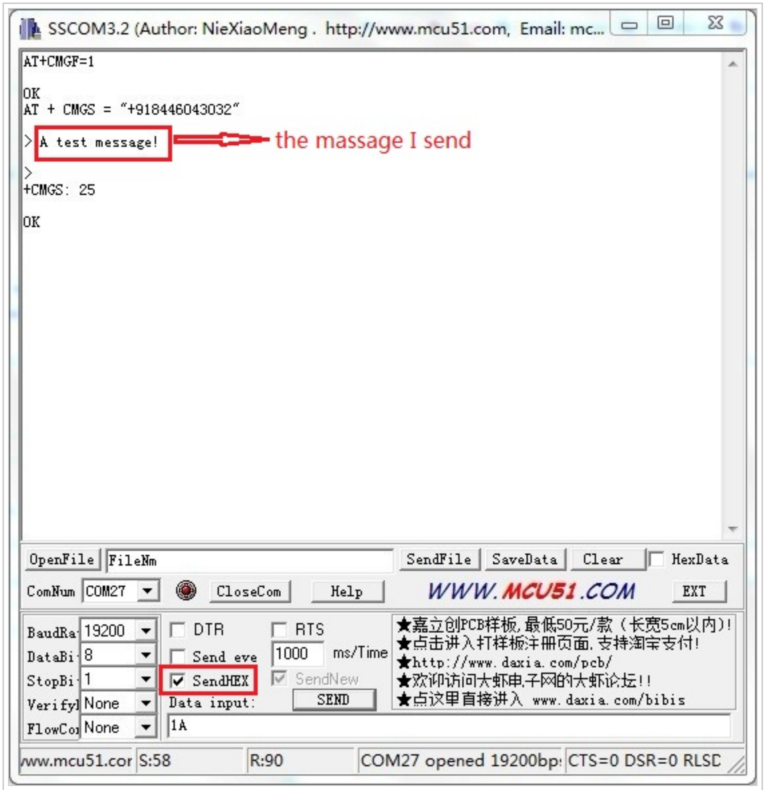
HEX	DATA	SEND
☐	AT+IPR=19200	1
☐	AT+CMGF=1	2
☐	AT + CMGS = "+918446043032"	3
☐	A test message!	4
✓	1A	5
☐	ATD10086;	6
☐	ATH	7
☐	AT+CSQ	8
☐	AT+CGATT?	9
☐	AT+SAPBR=3,1,"CONTYPE","GPR	10
☐	AT+SAPBR=3,1,"APN","CMNET"	11
☐	AT+SAPBR=1,1	12
☐	AT+HTTPINIT	13
☐	AT+HTTPPARA="URL","www.goog	14
☐	AT+HTTPACTION=0	15
☐	AT+HTTPREAD	16
☐	AT	17
✓	0D	18

```
Round send,interval: 1000 ms
ComNum COM27   BaudRate 19200   DataBits 8   StopBits 1   Verify None   FlowControl None
DTR ☐  RTS ☐  Send eve ☐ 1000 ms/Time  SendHEX ☐  SendNew ✓
Data input: AT + CMGS = "+918446043032"
www.mcu51.com S:40  R:50  COM27 opened 19200bps  CTS=0 DSR=0 RLSD
```

★嘉立创PCB样板,最低50元/款(长宽5cm以内)!
★点击进入打样板注册页面,支持淘宝支付!
★http://www.daxia.com/pcb/
★欢迎访问大虾电子网的大虾论坛!!
★点这里直接进入 www.daxia.com/bibis

GPRS send a message.jpg

- 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.



```
SSCOM3.2 (Author: NieXiaoMeng . http://www.mcu51.com, Email: mc...

AT+CMGF=1

OK
AT + CMGS = "+918446043032"

>A test message!

>
+CMGS: 25

OK
```

```
ComNum COM27   BaudRate 19200   DataBits 8   StopBits 1   Verify None   FlowControl None
DTR ☐  RTS ☐  Send eve ☐ 1000 ms/Time  SendHEX ✓  SendNew ✓
Data input: 1A
www.mcu51.com S:58  R:90  COM27 opened 19200bps  CTS=0 DSR=0 RLSD
```

★嘉立创PCB样板,最低50元/款(长宽5cm以内)!
★点击进入打样板注册页面,支持淘宝支付!
★http://www.daxia.com/pcb/
★欢迎访问大虾电子网的大虾论坛!!
★点这里直接进入 www.daxia.com/bibis

Gprssendsms1.jpg

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.

***Step 3: Exploring further**
Now that you have gotten a taste of how the AT Commands work, you can try out some more of them before moving on to developing sketches for Arduino to use the GPRS Shield. This involves creating a sketch for sending out these same sequence of AT Commands (on your behalf) out the serial port to the GPRS Shield to perform the same task of sending and SMS, making a call or sending data over a GPRS connection. You can go through the AT Commands reference manual (http://garden.seeedstudio.com/images/a/a0/SIM900_ATC_V1_00.pdf) to figure out the sequence of commands required to do a particular task. If while developing an Arduino sketch, you find that the GPRS Shield isn't what you expected it to do, then you will need to check your AT Commands and their sequence. To do this, reload the serial relay sketch attached above in the getting started section into ATmega328P and type out the AT Commands manually and check the output. The responses sent by the GPRS Shield will help you debug the AT Command sequence. NOTE: A C program to perfom the same task as the serial relay sketch present above has also been developed and attached: File:Softuart relay atmega328p.zip. The program was developed on a Windows PC. AVRStudio4 (http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725) was used as the IDE and WinAVR (http://winavr.sourceforge.net/) was used as the compiler. The ZIP file contains an AVRStudio4 Project. The C compiler (WinAVR) will generate an Intel Hex (.hex). To upload this .hex file into an Arduino board outside of Arduino IDE would require a program which is able to communicate with the Arduino boards bootloader. XLoader (http://xloader.russemotto.com/) is such a program which runs on Windows can upload .hex files generated by various compiler into an Arduino Board.

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":

```
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 loose 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 you 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, plesse note that this sketch uses the sorfware 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, threr 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 '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();
}

///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 h

  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 a
  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, t
  //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(1000);

  ShowSerialData();

  mySerial.println("AT+CSTT=\"CMNET\"");//start task and setting the APN,
  delay(1000);

  ShowSerialData();

  mySerial.println("AT+CIICR");//bring up wireless connection
  delay(3000);

  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 did't add other sensor, so using 4 strin
  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(" \"_cXwr5LE8qW4a296O-cDwOUvfddFer5pGmaRigPsiO0");//pachubeapikey
  delay(500);
  ShowSerialData();
  mySerial.print("jEB9OjK-W6vej56j9ItaSlIac-hgbQjxExuveD95yc8BttXc");//pachubeapikey
  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());
}
```

FAQ

Here is the GPRS Shield FAQ that list the frequently problems ,please read it before use :

- Why does GPRS Shield can not work with Stalker v2.1/2.0 via software serial port(D7/D8) ?

Stalker v2.1/2.0 has used the Digital Pin(D7/D8) , ou need to connected GPRS_TX/GPRS_RX to other Digital Pin as software serial port .And this problem had solved at Stalker 2.2 version.

- Why does the I2C can't be used when I assembled the GPRS Shield to Stalker or other Arduino boards ?

Because GPRS Shield conflict with other module via I2C . GPRS Shield doesn't use I2C port, you can disconnect the connection from SIM900 I2C port to A4/A5 .

Version Tracker

Revision	Descriptions	Release
v0.9b	Initial public release (beta)	Mar 3, 2011
v1.2	Added software port to power on/off of SIM90	Dec 2, 2011
v1.4	Re-design the power source circuit, re-lay the PCB layout	Aug 30, 2012

Resources

GPRS Shield v1.4 Schematic (http://www.seeedstudio.com/wiki/images/0/0c/GPRSshield_Schematic.pdf)
GPRS Shield v1.4 eagle file (http://www.seeedstudio.com/wiki/File:GPRS_shield_v1.4.zip)
AT Commands v1.00 (http://garden.seeedstudio.com/images/a/a0/SIM900_ATC_V1_00.pdf) & AT Commands v1.03 (http://garden.seeedstudio.com/images/a/a8/SIM900_AT_Command_Manual_V1.03.pdf) & Hardware Design (http://garden.seeedstudio.com/images/e/e3/SIM900_HD_V1.05.pdf) - SIM900 Documentation

Si5902BDC (http://www.vishay.com/docs/70415/si5902bd.pdf) - Dual N-Channel 30 V (D-S) MOSFETs (used for 2.8V <> 5.0V translation for Serial Interface)

SIM900 firmware and tool(firmware:1137B08SIM900M64_ST) (http://garden.seeedstudio.com/images/8/87/SIM900_firmware_and_tool.zip) for firmware upgrade

firmware:1137B09SIM900M64_ST (http://garden.seeedstudio.com/images/f/f9/1137B09SIM900M64_ST.zip)

SIM900datasheeet (http://www.seeedstudio.com/document/SIM900datasheeet.zip)