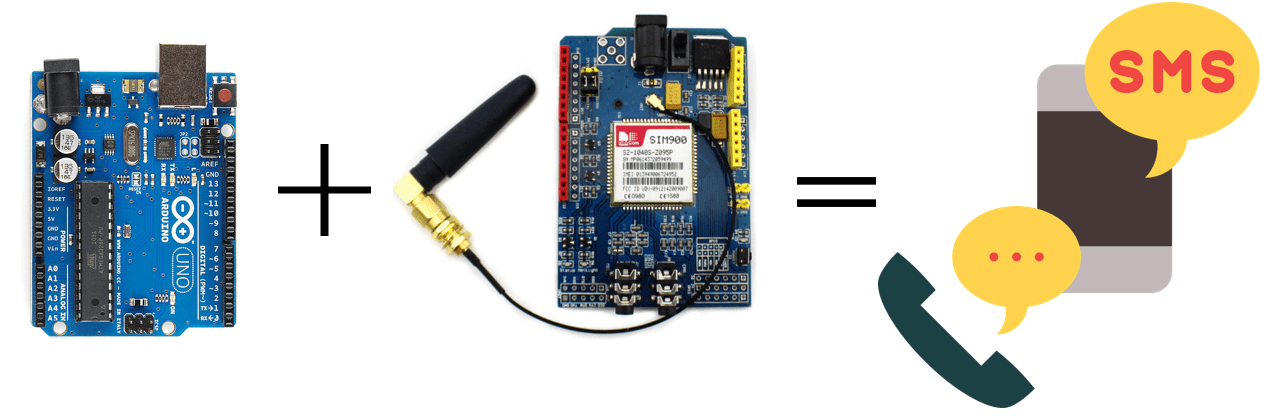
**Guide to SIM900 GSM GPRS Shield with Arduino**

93 Shares

This post is a complete guide to the SIM900 GSM GPRS Shield with the Arduino. Learn how to send and receive SMS, make and receive phone calls with Arduino.

In this guide we’ll cover:

* Introducing the SIM900 GSM GPRS Shield
* SIM900 GSM GPRS Shield Hardware
* SIM900 AT Commands
* Testing the shield
* Sending and receiving SMS – example
* Making and answering phone calls – example

**Introducing the SIM900 GSM GPRS Shield**

The SIM900 GSM GPRS Shield is shown in figure below.



**GSM** stands for **G**lobal **S**ystem for **M**obile Communications and is the global standard for mobile communications.

**GPRS** stands for **G**eneral **P**acket **R**adio **S**ervice. GPRS is a mobile service on the 2G and 3G cellular communication.

**Applications:**

**ENROLL IN:**[**Arduino Step-by-step Projects Course - Build 25 Projects**](https://randomnerdtutorials.com/arduino-step-by-step-projects/)

The GSM GPRS shield is particularly useful as it allows to:

* Connect to the Internet over GPRS network
* Send and receive SMS
* Place and receive phones calls

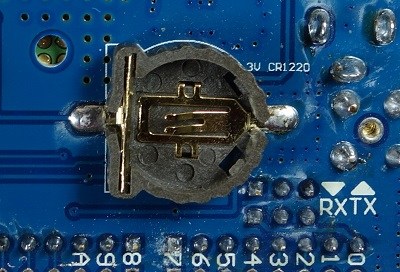
Its capabilities make it perfect for projects with Arduino like:

* Remote control of electronic appliances – sending an SMS to turn something on;
* Receive notifications – send SMS to your cell phone if movement is detected in your house;
* Receive sensor data – send periodic SMS to your cell phone with daily weather data.

**Features**

Here’s some of the most important features of the shield:

* Compatible with Arduino and clones
* Based on SIM900 module from SIMCOM
* Allows you to send SMS, MMS, GPRS and Audio via UART using AT commands.
* It has 12 GPIOs, 2 PWMs and buit-in ADC of the SIM900 module
* Quad Band: 850; 900; 1800 and 1900 MHZ, so it should work in all countries with GSM (2G) networks
* Control via AT commands
* Supports RTC (real time clock) – it has a holder for a 3V CR1220 battery at the back



* Has microphone and headphone jacks for phone calls



**Where to buy?**

[](https://makeradvisor.com/tools/sim900-gsm-gprs-shield/)

You can check the [SIM900 GSM GPRS shield on Maker Advisor](https://makeradvisor.com/tools/sim900-gsm-gprs-shield/) and find the best price.

You can use the preceding links or go directly to [MakerAdvisor.com/tools](https://makeradvisor.com/tools/?utm_source=rnt&utm_medium=post&utm_campaign=post) to find all the parts for your projects at the best price!

**DOWNLOAD FREE GUIDE:**[Ultimate Guide for Arduino Modules and Sensors](https://randomnerdtutorials.com/sim900-gsm-gprs-shield-arduino/javascript%3Avoid(0))

[https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2017/10/header-200.png?ssl=1](https://makeradvisor.com/tools/?utm_source=rnt&utm_medium=post&utm_campaign=post)

**Preliminary steps**

Before getting started with your SIM900 GSM GPRS module, you need to consider some aspects about the SIM card and the shield power supply.

**GSM coverage**

Ensure you have coverage on a GSM 850 MHz, GSM 900 MHz, DCS 1800 MHz or PCS 1900 MHz network. By GSM we mean 2G.

**Prepaid SIM Card**

We recommend that you use a prepaid plan or a plan with unlimited SMS for testing purposes. Otherwise, if something goes wrong, you may need to pay a huge bill for hundreds of SMS text messages sent by mistake. In this tutorial we’re using a prepaid plan with unlimited SMS.

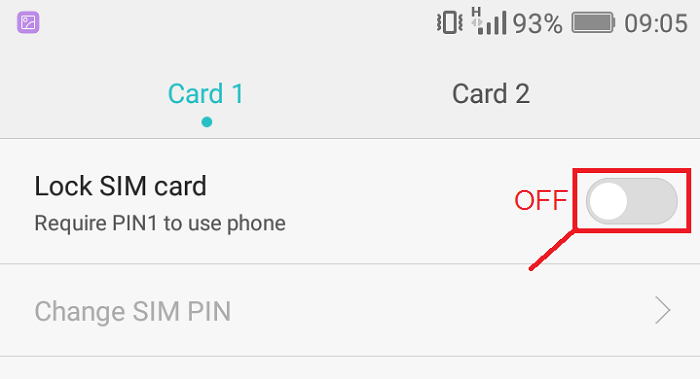


The shield uses the original SIM card size, not micro or nano. If you have micro or nano you may consider getting a [SIM card size adapter](http://ebay.to/2gbdOcC).

**Turn off the PIN lock**

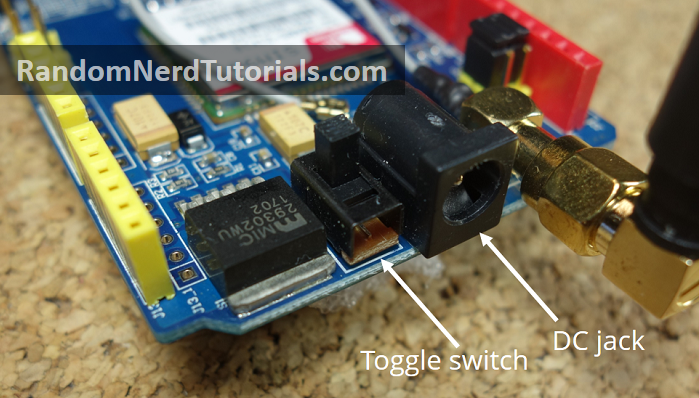
To use the SIM card with the shield, you need to turn off the pin lock. The easiest way to do this, is to insert the SIM card in your smartphone and turn off the pin lock in the phone security settings.

In my case, I needed to go through: **Settings** > **Advanced** **Settings** > **Security** > **SIM lock** and turn off the lock sim card with pin.



**Getting the right power supply**

The shield has a DC socket for power as shown in figure below.



Next to the power jack there is a toggle switch to select the power source. Next to the toggle switch on the board, there is an arrow indicating the toggle position to use an external power supply – move the toggle switch to use the external power supply as shown above.

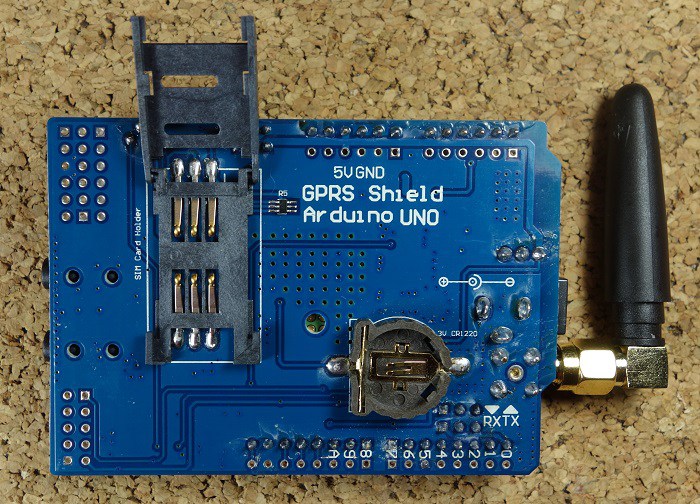
To power up the shield, it is advisable to use a 5V power supply that can provide 2A as the one shown below. It can also be powered with 9V 1A, or 12V 1A.



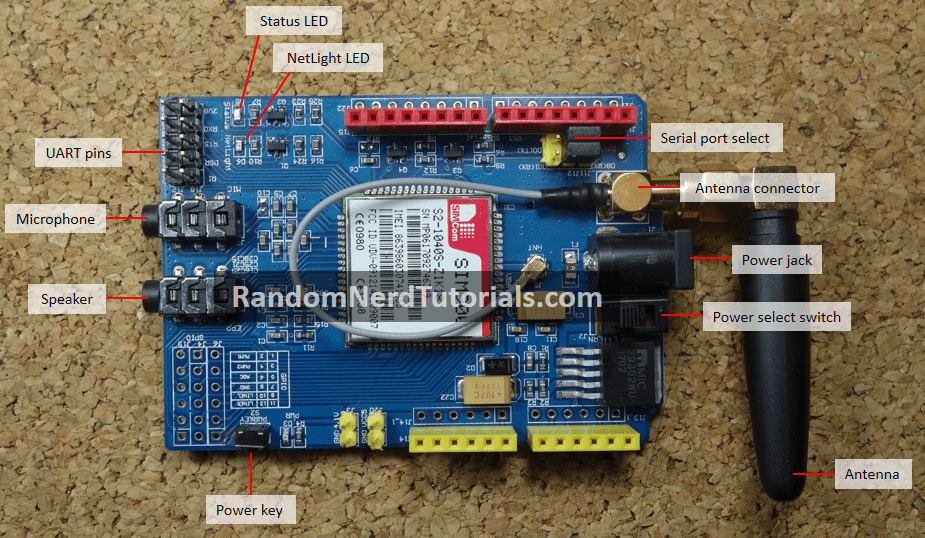
You can find the right power adapter for this shield [here](https://makeradvisor.com/tools/5v-2a-power-adaptor-dc-jack/). Make sure you select the model with 5V and 2A.

**SIM900 GSM GPRS Shield Hardware**

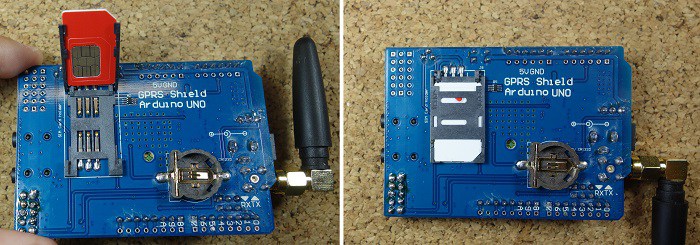
The figure below shows the back of the shield. It has a holder for the SIM card and for a 3V CR1220 battery for the RTC (real time clock).



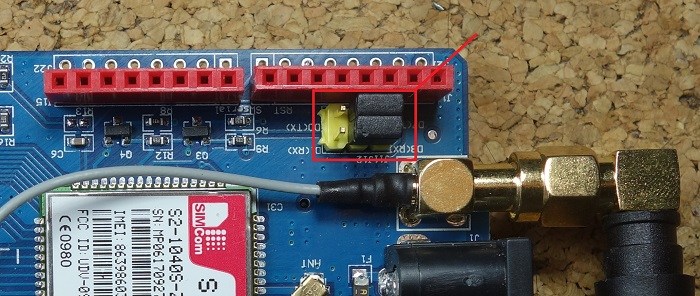
The figure below shows the shield most important components on the board that you need to pay attention to.

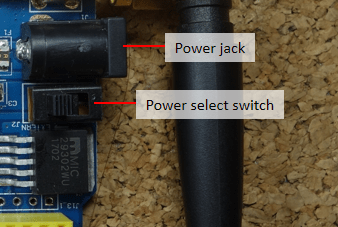
[](https://i1.wp.com/randomnerdtutorials.com/wp-content/uploads/2017/08/GSM-shield-labels.png?ssl=1)

**Getting started**

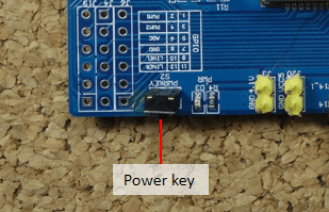
**1)** Insert the SIM card into the SIM card holder – make sure you’ve read the preliminary steps in the previous section.**2)**Make sure the antenna is well connected.

**3)**On the serial port select, make sure the jumper cap is connected as shown in figure below to use software serial.

**4)**Power the shield using an external 5V power supply. Make sure you select the external power source with the toggle switch next to the DC jack.



**5)**To power up/down the shield press the power key for about 2 seconds.

**6)**Then, the Status LED will light up and the NetLight LED will blink every 800 ms until it finds the network. When it finds the network, the NetLight LED will start blinking every three seconds.

**Note:**you can automatically turn on the shield via software. See how to do that in the **Automatically turn on the shield** section, after the code examples.

**7)**You can test if the shield is working properly by sending AT commands from the Arduino IDE using an FTDI programmer – as we’ll shown later in this guide.

**SIM900 AT commands**

* set the SIM900 to text mode: **AT+CMGF=1\r**
* send SMS to a number: **AT+CMGS=PHONE\_NUMBER**(in international format)
* read the first SMS from the inbox: **AT+CMGR=1\r**
* read the second SMS from the inbox: **AT+CMGR=2\r**
* read all SMS from the inbox: **AT+CMGR=ALL\r**
* call to a number: **ATDP+ PHONE\_NUMBER**(in international format)
* hang up a call: **ATH**
* receive an incoming call: **ATA**

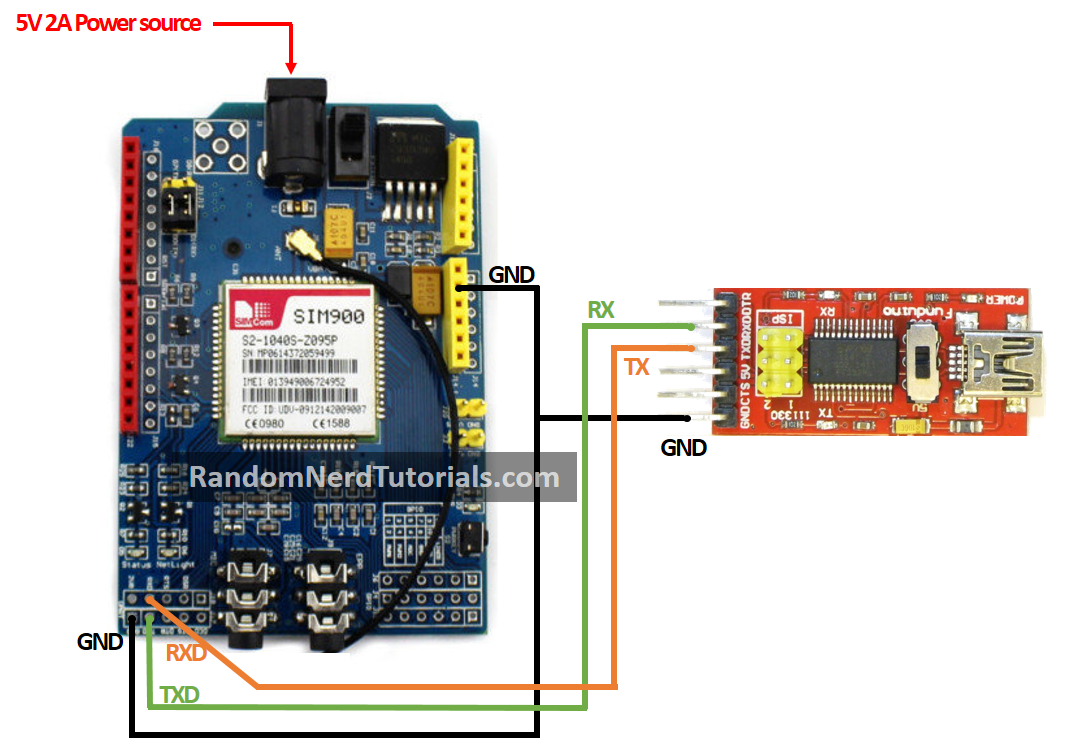
For more information, you can check the SIM900 AT commands manual [here](http://wiki.seeedstudio.com/images/a/a8/SIM900_AT_Command_Manual_V1.03.pdf).

**Testing the Shield with FTDI programmer**

To test if everything is working properly, you can test the shield by sending AT commands from the Arduino IDE serial monitor. For that, you need an FTDI programmer as the one shown in figure below. You can get an FTDI programmer like this [here](https://makeradvisor.com/tools/ftdi-programmer-board/).

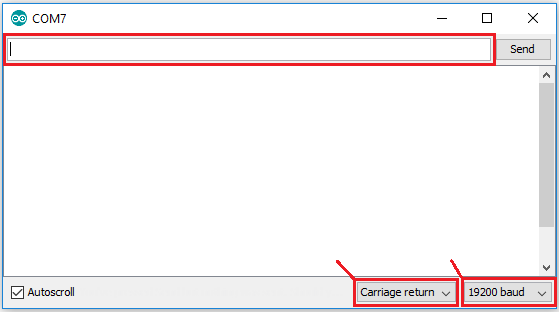


**1)**Connect the FTDI programmer to the GSM shield as shown in figure below.

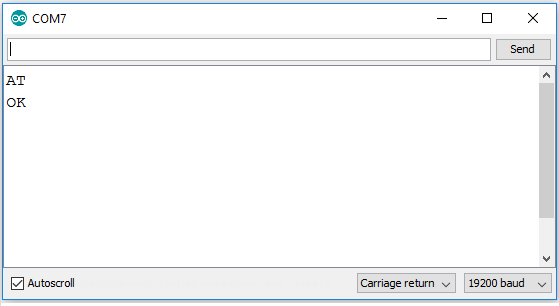
[](https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2017/08/GSM-FTDI-circuit-1.png?ssl=1)

**2)**Open the Arduino IDE and select the right COM port.

**3)**Open the Serial monitor 

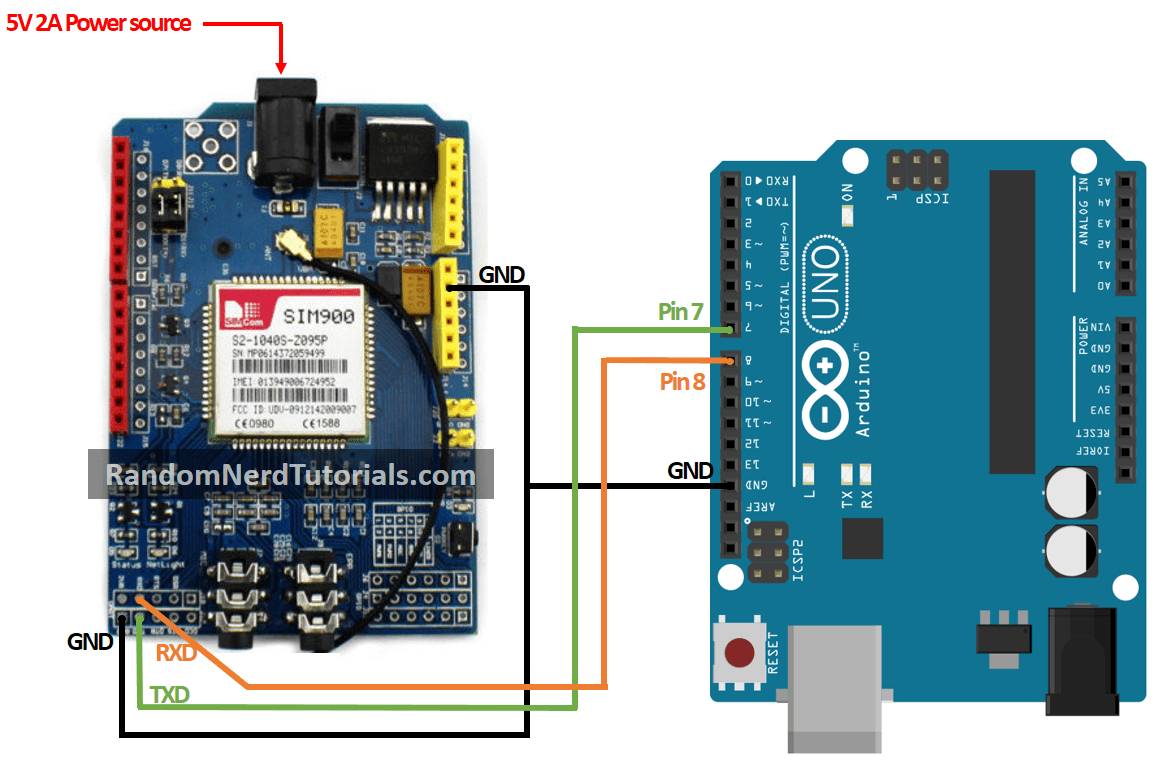
**4)**Select **19200** baud rate – the shield default setting is 19200 – and **Carriage return.** Write **AT** at the box highlighted in red and then press enter. See figure below. 

**5)**The shield will respond with **OK**, if everything is working properly.



**Connecting the Shield to Arduino**

Connect the shield to the Arduino as shown in the schematics below.

[](https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2017/08/GSM-Arduino-circuit.png?ssl=1)

**Sending an SMS**

To send an SMS, upload the code below to your Arduino board.

/\*\*\*\*\*\*\*\*\*  
  Complete project details at http://randomnerdtutorials.com    
\*\*\*\*\*\*\*\*\*/  
  
#include <SoftwareSerial.h>  
  
// Configure software serial port  
SoftwareSerial SIM900(7, 8);   
  
void setup() {  
  // Arduino communicates with SIM900 GSM shield at a baud rate of 19200  
  // Make sure that corresponds to the baud rate of your module  
  SIM900.begin(19200);  
  // Give time to your GSM shield log on to network  
  delay(20000);     
    
  // Send the SMS  
  sendSMS();  
}  
  
void loop() {   
    
}  
  
void sendSMS() {  
  // AT command to set SIM900 to SMS mode  
  SIM900.print("AT+CMGF=1\r");   
  delay(100);  
  
  // REPLACE THE X's WITH THE RECIPIENT'S MOBILE NUMBER  
  // USE INTERNATIONAL FORMAT CODE FOR MOBILE NUMBERS  
  SIM900.println("AT + CMGS = \"+XXXXXXXXXXXX\"");   
  delay(100);  
    
  // REPLACE WITH YOUR OWN SMS MESSAGE CONTENT  
  SIM900.println("Message example from Arduino Uno.");   
  delay(100);  
  
  // End AT command with a ^Z, ASCII code 26  
  SIM900.println((char)26);   
  delay(100);  
  SIM900.println();  
  // Give module time to send SMS  
  delay(5000);   
}

[view raw](https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/raw/master/Projects/GSM/sendSMS.ino) [Projects/GSM/sendSMS.ino](https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/GSM/sendSMS.ino)

In this code,  you start by  including the **SoftwareSerial.h** library and create a software serial port on pins 7 and 8. (Pin 7 is being set as RX and 8 as TX)

**#include <SoftwareSerial.h>**

**SoftwareSerial SIM900(7, 8);**

The **sendSMS()** function created is what actually sends the SMS. This function uses the AT commands: **AT+CMGF=1\**r and **AT + CMGS**.

You need to change the recipient’s mobile number at: (replace the **X**‘s with the recipient’s phone number)

**SIM900.println("AT + CMGS = \"++++++++++++++"");**

The recipient’s mobile number should be in international format.

Then, at the following line you can edit the text you want to send.

// REPLACE WITH YOUR OWN SMS MESSAGE CONTENT

**SIM900.println("Message example from Arduino Uno.")**

**Reading received SMS**

To read incoming SMS, upload the code below to your Arduino. After uploading, wait 20 seconds for the shield to establish communication. Then, test the script by sending an SMS to the shield SIM card number. The SMS is shown on the Arduino serial monitor – baud rate: 19200.

/\*\*\*\*\*\*\*\*\*  
  Complete project details at http://randomnerdtutorials.com    
\*\*\*\*\*\*\*\*\*/  
  
#include <SoftwareSerial.h>  
  
// Configure software serial port  
SoftwareSerial SIM900(7, 8);  
//Variable to save incoming SMS characters  
char incoming\_char=0;  
  
void setup() {  
  // Arduino communicates with SIM900 GSM shield at a baud rate of 19200  
  // Make sure that corresponds to the baud rate of your module  
  SIM900.begin(19200);  
  // For serial monitor  
  Serial.begin(19200);   
  // Give time to your GSM shield log on to network  
  delay(20000);  
  
  // AT command to set SIM900 to SMS mode  
  SIM900.print("AT+CMGF=1\r");   
  delay(100);  
  // Set module to send SMS data to serial out upon receipt   
  SIM900.print("AT+CNMI=2,2,0,0,0\r");  
  delay(100);  
}  
  
void loop() {  
  // Display any text that the GSM shield sends out on the serial monitor  
  if(SIM900.available() >0) {  
    //Get the character from the cellular serial port  
    incoming\_char=SIM900.read();   
    //Print the incoming character to the terminal  
    Serial.print(incoming\_char);   
  }  
}

[view raw](https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/raw/master/Projects/GSM/receiveSMS.ino) [Projects/GSM/receiveSMS.ino](https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/GSM/receiveSMS.ino)

In this code, you set the module to send the SMS data to the serial output:

**SIM900.print("AT+CNMI=2,2,0,0,0\r");**

You store the incoming characters from the SMS message on the **incoming\_char** variable. You read the chars using the **SIM900.read()** function.

**Making a phone call**

To make a phone call, upload the following code to your Arduino.

**Don’t forget to edit the code with the phone number you want to call.**

/\*\*\*\*\*\*\*\*\*  
  Complete project details at http://randomnerdtutorials.com    
\*\*\*\*\*\*\*\*\*/  
  
#include <SoftwareSerial.h>  
  
// Configure software serial port  
SoftwareSerial SIM900(7, 8);   
  
void setup() {  
  // Arduino communicates with SIM900 GSM shield at a baud rate of 19200  
  // Make sure that corresponds to the baud rate of your module  
  SIM900.begin(19200);  
  // Give time to your GSM shield log on to network  
  delay(20000);  
  
  // Make the phone call  
  callSomeone();  
}  
  
void loop() {  
    
}  
  
void callSomeone() {  
  // REPLACE THE X's WITH THE NUMER YOU WANT TO DIAL  
  // USE INTERNATIONAL FORMAT CODE FOR MOBILE NUMBERS  
  SIM900.println("ATD + +XXXXXXXXX;");  
  delay(100);  
  SIM900.println();  
    
 // In this example, the call only last 30 seconds  
 // You can edit the phone call duration in the delay time  
  delay(30000);  
  // AT command to hang up  
  SIM900.println("ATH"); // hang up  
}

[view raw](https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/raw/master/Projects/GSM/makePhoneCall.ino) [Projects/GSM/makePhoneCall.ino](https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/GSM/makePhoneCall.ino)

To make the call, you use the **callSomeone()** function that uses the **ATD** command.

**SIM900.println("ATD + +XXXXXXXXX;");**

You need to replace the **X**‘s (highlighted in red) with the phone number you want to call.

Don’t forget to connect a microphone and earphones to make the call.

In this code example, the call is hang up after 30 seconds, using the **ATH** command:

**SIM900.println("ATH");**

Hanging up after 30 seconds is not very useful, but it works well for an example. The idea is that you use the **ATH** command when an event is triggered. For example, connect a push button to the Arduino, that when pressed sends the **ATH** command to hang up the phone.

**Answering incoming phone calls**

The following code answers incoming calls.

/\*\*\*\*\*\*\*\*\*  
  Complete project details at http://randomnerdtutorials.com    
\*\*\*\*\*\*\*\*\*/  
  
#include <SoftwareSerial.h>  
  
// Configure software serial port  
SoftwareSerial SIM900(7, 8);   
char incoming\_char=0;  
  
void setup() {  
  // Arduino communicates with SIM900 GSM shield at a baud rate of 19200  
  // Make sure that corresponds to the baud rate of your module  
  SIM900.begin(19200); // for GSM shield  
  // For serial monitor  
  Serial.begin(19200);   
  // Give time to log on to network.  
  delay(20000);   
    
  SIM900.print("AT+CLIP=1\r"); // turn on caller ID notification  
  delay(100);  
}  
  
void loop() {  
  // Display any text that the GSM shield sends out on the serial monitor  
  if(SIM900.available() >0) {  
    // Get the character from the cellular serial por  
    // With an incomming call, a "RING" message is sent out  
    incoming\_char=SIM900.read();  
    //  Check if the shield is sending a "RING" message  
    if (incoming\_char=='R') {  
      delay(10);  
      Serial.print(incoming\_char);  
      incoming\_char=SIM900.read();  
      if (incoming\_char =='I') {  
        delay(10);  
        Serial.print(incoming\_char);  
        incoming\_char=SIM900.read();  
        if (incoming\_char=='N') {  
          delay(10);  
          Serial.print(incoming\_char);  
          incoming\_char=SIM900.read();  
          if (incoming\_char=='G') {  
            delay(10);  
            Serial.print(incoming\_char);  
            // If the message received from the shield is RING  
            // Send ATA commands to answer the phone  
            SIM900.print("ATA\r");  
          }  
        }  
      }  
    }  
  }  
}

[view raw](https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/raw/master/Projects/GSM/receiveCall.ino) [Projects/GSM/receiveCall.ino](https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/GSM/receiveCall.ino)

When someone calls the SIM900 number, it sends a message saying “RING”. To know if someone is calling you, you can wait for incoming characters from the SIM900 and then, compare if it was a RING message.  That’s what is done in this code. When it receives a RING message, you send the **ATA** command to answer the phone.

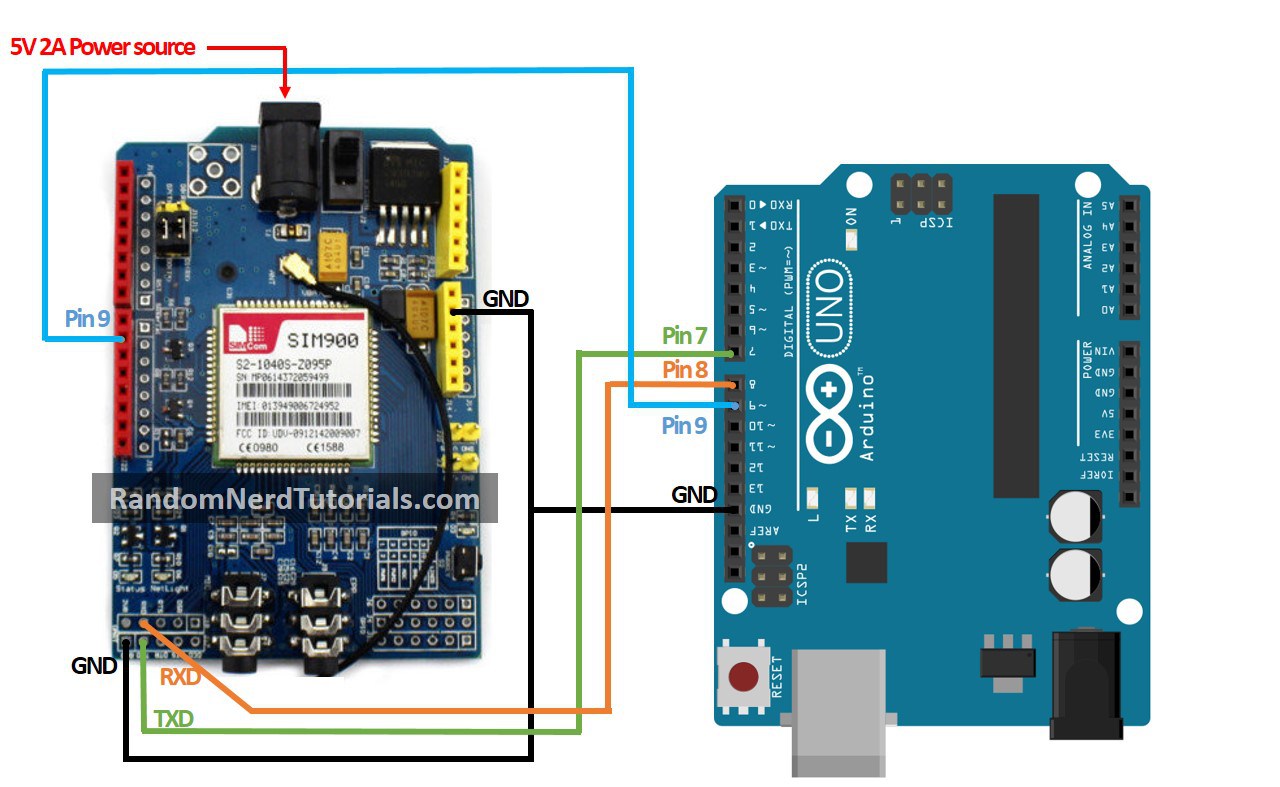
**Automatically turn on the shield**

Instead of manually pressing the “power” key to turn on the shield, you can automatically turn on the shield via software.

**1)** First, you need to solder R13 connections on the shield as shown in the figure below – highlighted in red.

[](https://i1.wp.com/randomnerdtutorials.com/wp-content/uploads/2017/10/R13.jpg?ssl=1)

**2)** Connect D9 on the shield to the D9 Arduino pin as shown in the schematic below.

[](https://i1.wp.com/randomnerdtutorials.com/wp-content/uploads/2017/10/GSM-Arduino-circuit.jpg?ssl=1)

**3)**Add the following code snippet in the setup() function. This is the equivalent of pressing the shield “power” button.

digitalWrite(9, HIGH);

delay(1000);

digitalWrite(9, LOW);

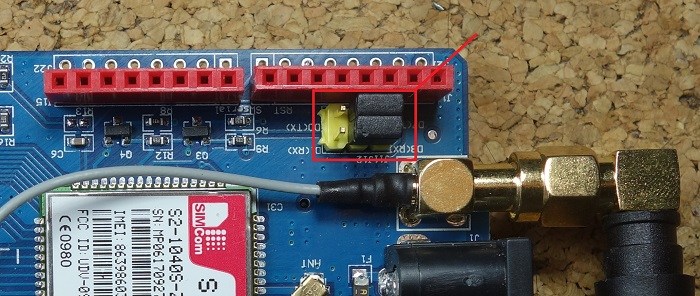
delay(5000);

**Troubleshooting**

**Shield doesn’t respond with OK**

Check your TX and RX connections to the Arduino. Try repeating the process by changing the TX with the RX pins.

Also check if you have selected the software serial by placing the jumper cap on the appropriate place on the serial selector.



**Cannot see messages in the serial monitor**

To see the messages in the serial monitor, the shield and the Arduino’s serial port baud rate should be the same. The SIM900 GSM GPRS shield default baud rate is 19200. So, select the Arduino’s baud rate to 19200.

However, if you need to change the shield baud rate, you can send the following AT command to change it to 19200 or other appropriate baud rate.

**AT+IPR=19200**

**Wrapping up**

This tutorial shows you how to send and receive SMS and making and receiving phone calls with the Arduino. You can apply the concepts learned in this tutorial to build your own projects to communicate over a cell network. We have other projects that use GSM, check them below:

* [Control a 12V lamp via SMS](https://randomnerdtutorials.com/control-a-12v-lamp-via-sms-with-arduino/)
* [Request Sensor Data via SMS using Arduino](https://randomnerdtutorials.com/request-sensor-data-sms-arduino-sim900-gsm-shield/)

If you like Arduino projects, make sure you check our latest Arduino: [**Arduino Step-by-step Projects – Build 25 Projects**](https://randomnerdtutorials.com/arduino-step-by-step-projects/)

We hope you’ve found this guide useful.

Thanks for reading.