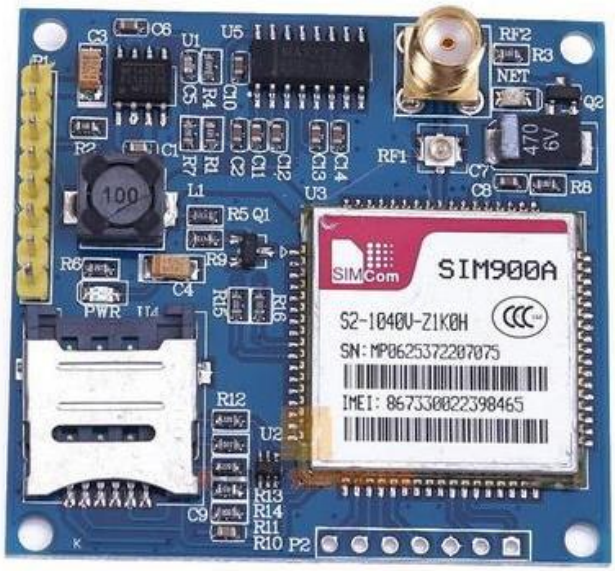
Interfacing GSM Module to Arduino Uno

Interfacing GSM SIM900A with Arduino:

SIM900A is an ultra-compact and reliable wireless module. The SIM900A is a complete Dual-band GSM/GPRS solution in a SMT module which can be embedded in the customer applications. Featuring an industry-standard interface, the SIM900A delivers GSM/GPRS 900/1800MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. With a tiny configuration of 24mmx24mmx3mm, SIM900A can fit in almost all the space requirements in user applications, especially for slim and compact demand of design.

Features:

- Dual-Band 900/ 1800 MHz
- GPRS multi-slot class 10/8GPRS mobile station class B
- Compliant to GSM phase 2/2+Class 4 (2 W @850/ 900 MHz)
- Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- Low power consumption: 1.5mA (sleep mode)'
- Operation temperature: -40°C to +85 °C
- Status indicator (D5): It will flash continuously whenever the call arrives otherwise it is left ON.
- Network LED (D6): This led will blink every second which indicates that the GSM module is not connected to the mobile network. Once the connection is established successfully, the LED will blink continuously every 3 seconds.



Booting the GSM Module:

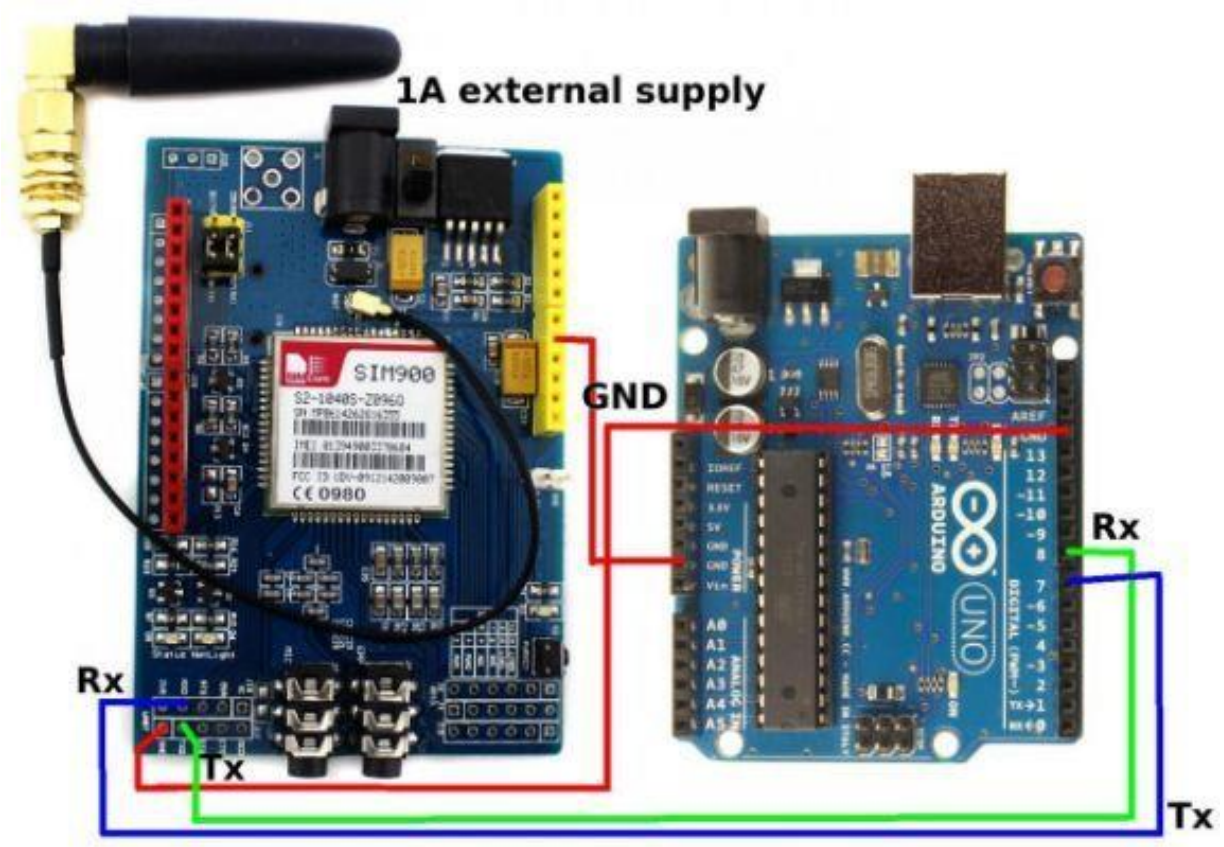
- Insert the SIM card to GSM module and lock it.
- Connect the adapter to GSM module and turn it ON.

- Wait for some time (say 1 minute) and see the blinking rate of ‘status LED’ or ‘network LED’ (GSM module will take some time to establish connection with mobile network)
- Once the connection is established successfully, the status/network LED will blink continuously every 3 seconds. You may try making a call to the mobile number of the sim card inside GSM module. If you hear a ring back, the GSM module has successfully established network connection.

Connecting GSM module with Arduino:

- There are two ways of connecting GSM module to Arduino.
- The communication between Arduino and GSM module is serial.
- We are supposed to use serial pins of Arduino (Rx and Tx). If you are going with this method, you may connect the Tx pin of GSM module to Rx pin of Arduino and Rx pin of GSM module to Tx pin of Arduino.
- GSM Tx –> Arduino Rx and GSM Rx –> Arduino Tx.
- Connect the ground pin of Arduino to ground pin of GSM module.

Interfacing Diagram:



Code for GSM module interfacing with Arduino:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(8, 7);    //  Tx=7 and Rx=8
void setup()
{
  mySerial.begin(9600);   // Setting the baud rate of GSM Module
  Serial.begin(9600);    // Setting the baud rate of Serial Monitor (Arduino)
  delay(100);
}
void loop()
{
  if (Serial.available()>0)
   switch(Serial.read())
  {
    case 's':
      SendMessage();
      break;
    case 'd':
      DialCall();
      break;
  }
 if (mySerial.available()>0)
   Serial.write(mySerial.read());
}
 void SendMessage()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+xxxxxxxxxxx\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("I am SMS from GSM Module");// The SMS text you want to send
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
 /*void RecieveMessage()
{
  mySerial.println("AT+CNMI=2,2,0,0,0");    // AT Command to receive a live SMS
  delay(1000);
}
*/
  void DialCall()
 {
  mySerial.println("ATD+xxxxxxxxxxx;");      // ATDxxxxxxxxxx; -- watch out here for semicolon at
the end!!
  delay(100);
 }
```

Code explanation for GSM module interfacing with Arduino:

We begin by including SoftwareSerial library into the program. In the next line, we create a constructor of SoftwareSerial with name mySerial and we pass the digital pin numbers as parameters. The actual format is like SoftwareSerial mySerial (Rx, Tx);
The pin number 8 will act as Rx of Arduino and 7 will act as Tx of Arduino.

The configuration part of program inside setup ()

- The first task is to set baud rates of SoftwareSerial library to communicate with GSM module. We achieve this by invoking mySerial.begin function.
- The second task is to set the baud rate of Arduino IDE's Serial Monitor. We do this by invoking Serial.begin function. Both should be set at the same baud rate and we use 9600 bits/second.

The actual program inside loop ()
The program seeks user input via serial monitor of Arduino.
If the input is 's' the program will invoke function to send a sms from GSM module. If the user input is 'r', the program will invoke the function to receive a live SMS from GSM module and display it on serial monitor of Arduino.

Functions Used:
Serial.available() – checks for any data coming through serial port of arduino. The function returns the number of bytes available to read from serial buffer. If there is no data available, it returns a -1 (value less than zero).

Serial.read() – Reads all the data available on serial buffer (or incoming serial data if put otherwise). It returns the first byte of incoming serial data.

mySerial.available() – checks for any data coming from GSM module through the SoftwareSerial pins 8 and 7. It returns the number of bytes available to read from software serial port. It returns a -1 if no data is available to read.

mySerial.read() – Reads the incoming data through software serial port.

Serial.write() – Prints data to serial monitor of Arduino. So, the function Serial.write(mySerial.read()) – prints the data collected from software serial port to serial monitor of Arduino.

SendMessage() – The function is created in our Arduino sketch to send an SMS. To send an SMS, we should set our GSM module to Text mode first. This is achieved by sending an AT Command "AT+CMGF=1", We send this command by writing this to SoftwareSerial port. To achieve this, we use the mySerial.println() function. mySerial.println writes data to software serial port (the Tx pin of our Software Serial) and this will be captured by GSM module (through its Rx pin). After setting the GSM module to Text mode, we should the mobile number to which we shall send the SMS. This is achieved with AT command "AT+CMGS=\"+91xxxxxxxxxx\"\r" – where you may replace all x with the mobile number.

In next step, we should send the actual content of SMS. The end of SMS content is identified with CTRL+Z symbol. The ASCII value of this CTRL+Z is 26. So, we send a char (26) to GSM module using the line mySerial.println((char)26); Each AT command may be followed by 1 second delay. We must give some time for GSM module to respond properly. Once these commands are sent to GSM module, you shall receive an SMS in the set mobile number.

RecieveMessage() – is the function to receive an SMS (a live SMS). The AT command to receive a live SMS is "AT+CNMI=2, 2, 0, 0, 0" – we just need to send this command to GSM module and apply a 1 second delay. Once you send this command, try sending an SMS to the SIM card number put inside GSM module. You will see the SMS you had sent displayed on your Arduino serial monitor.