ABLEDUCATION™
ACTIVITY BASED LEARNING

GSM SIM(900) Calling

This is a very low cost and simple Arduino GSM and GPRS module. We use the module SIM Com SIM900A . It's the cheaper module now available In the market. This post will allow you to make arduino controlled calls and also send text messages.

ABLEDUCATION™
ACTIVITY BASED LEARNING

Step 1: Some Important Notes and Powering Up the GSM Module

1. We use SIM900 GSM Module – This means the module supports communication in 900MHz band. We are from India and most of the mobile network providers in this country operate in the 900Mhz band.
2. If you are from another country, you have to check the mobile network band in your area. A majority of United States mobile networks operate in 850Mhz band (the band is either 850Mhz or 1900Mhz). Canada operates primarily on 1900 Mhz band

ABLEDUCATION™
ACTIVITY BASED LEARNING

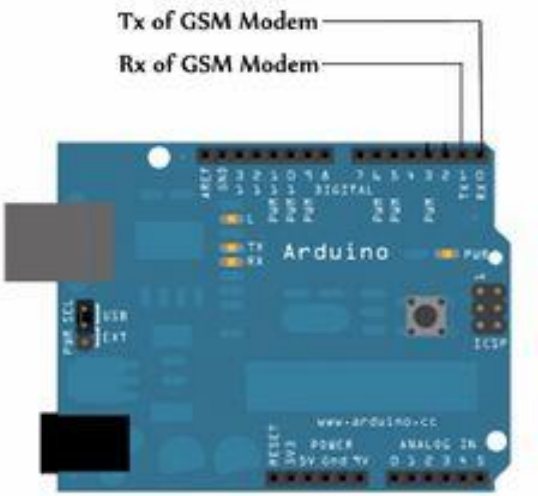
2. Check the power requirements of GSM module – GSM modules are manufactured by different companies. They all have different input power supply specs. You need to double check your GSM modules power requirements. In this tutorial, our GSM module requires a 12 volts input.

So we feed it using a 12V,1A DC power supply. I have seen GSM modules which require 15 volts and some other which needs only 5 volts. They differ with manufacturers. If you are having a 5V module, you can power it directly from Arduino 5V out.

ABLEDUCATION™
ACTIVITY BASED LEARNING

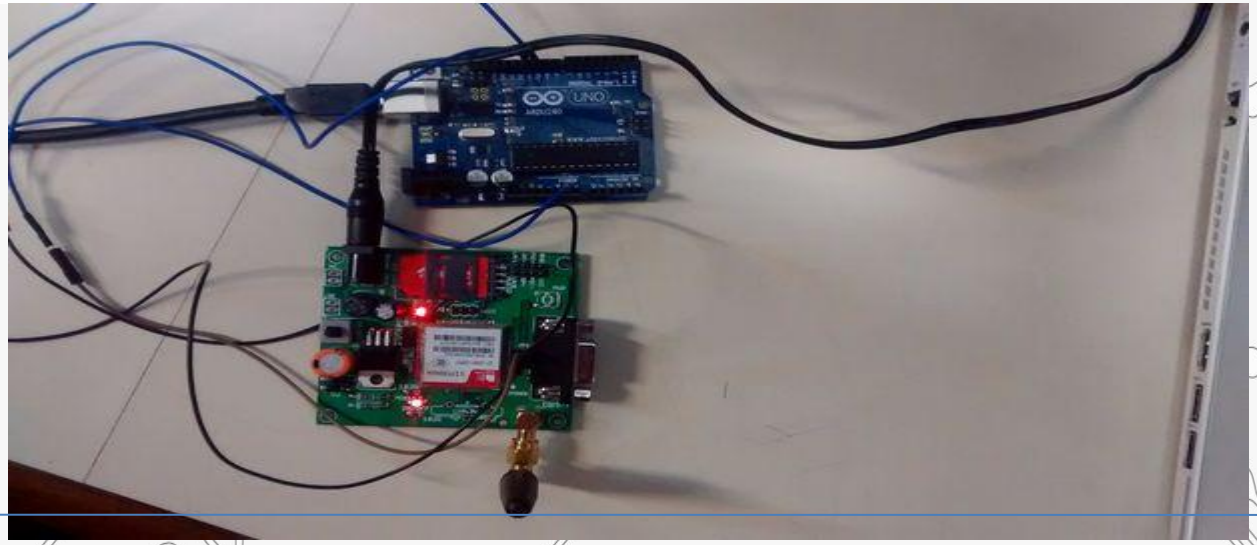
Booting Up The GSM:

1. Insert the SIM card to module and lock it.
2. Connect the adapter to module and turn it ON!
3. Now wait for some time (say 1 minute) and see the blinking rate of ‘NWK LED’ (network led)
4.GSM module will take some time to establish connection with mobile network.
5. Once the connection is established successfully, the NWK LED will blink continuously every 3 seconds.
6. Now you can check the module by calling to the SIM inserted and you will get a ring on that number. Then your SIM900 module is ready. Otherwise check connections.



There are two ways of connecting GSM module to arduino.

In any case, the communication between Arduino and GSM module is serial.



ABLEDUCATION™
ACTIVITY BASED LEARNING

So we are supposed to use serial pins of Arduino (Rx and Tx). So 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.

Now connect the ground pin of arduino to ground pin of gsm module! So that's all! You made 3 connections and the wiring is over! Now you can load different programs to communicate with GSM module and make it work.

ABLEDUCATION™
ACTIVITY BASED LEARNING

Note:- The problem with this connection is while programming. Arduino uses serial ports to load program from the Arduino IDE. If these pins are used in wiring, the program will not be loaded successfully to Arduino.

So you have to disconnect wiring in Rx and Tx each time you burn the program. Once the program is loaded successfully, you can reconnect these pins and have the

ABLEDUCATION™
ACTIVITY BASED LEARNING

- To avoid this difficulty, I am using an alternate method in which two digital pins of arduino are used for serial communication. We need to select two PWM enabled pins of arduino for this method. So I choose pins 9 and 10 (which are PWM enabled pins).

- This method is made possible with the Software Serial Library of Arduino. Software Serial is a library of Arduino which enables serial data communication through other digital pins of Arduino. The library replicates hardware functions and handles the task of serial communication.

GSM SIM(900) WIRING TO ARDUINO

1.12V adaptor to SIM900
2.Arduino 5V pin to SIM900 VCC
3.Arduino GND to SIM900 GND
4.SIM900 TX to Arduino pin 10
5.SIM900 RX to Arduino pin 11

ABL EDUCATION™
ACTIVITY BASED LEARNING

Code

SIM900A_GSM_Modem_with_Arduino | Arduino 1.8.19

File Edit Sketch Tools Help

SIM900A_GSM_Modem_with_Arduino

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup()
{
// Open serial communications and wait for port to open:
Serial.begin(9600);

Serial.println("Calling through GSM Modem");

// set the data rate for the SoftwareSerial port
mySerial.begin(9600);
delay(2000);
mySerial.println("ATD9870812020;"); // ATD81290255XX; -- watch out here for semicolon at the end!!

Serial.println("Called ATD7217397270");
}

void loop() // run over and over
{
// print response over serial port
if (mySerial.available())
Serial.write(mySerial.read());
}
```