

**GSM Communication System for remote controlled devices
using LinkSprite SM5100B**

Jeff Pawloski

March 30th, 2012

Abstract:

The following application notes will describe the implementation of the LinkSprite SM5100B Global System for Communications module which can be used and modified to signal remote applications anywhere in the world. This method requires the use of a microcontroller that has a serial input with a variable baud rate. In this application note I will be using an Arduino Uno (Revision 3) that can be found at any major electronics retailer. In addition to the SM5100B and a microcontroller, an antenna will be required that can operate on the GSM frequencies at your location, and an activated SIM card that can send and receive text messages through a provided carrier in your area.

Introduction:

Today's society is booming with applications that use wireless control systems and the high level technological devices. The Arduino platform has been designed as an easy to use and implement physical computing device for embedded applications. This is because of its java-based programming environment which caters to electronics hobbyists much like the HAM radio became popular in the early 20th century. The major benefit of the Arduino environment is the open-source nature of hobbyist communities which can provide helpful support and the ability to rapidly prototype designs. To learn more information about the Arduino environment and the extensive applications it can provide please visit their homepage^[1].

Along with the our serial friendly microcontroller we will be using the SM5100B GSM module which has the full serial capabilities as well as quad-band technology which allows it to operate on any of the four worldwide GSM frequencies depending on your location of operation. By connecting the Tx/Rx serial pins of the GSM device to the Rx/Tx serial pins of the Arduino we are able to seamlessly communicate messages in between platforms with as little as 4 pins(Tx/Rx/5V/GND). This will create a GSM text message server that can receive up to 140 bytes of control data that allows the microcontroller to decide what to do with each text message and send analog or digital outputs to its wide array of output pins.

Keywords:

Arduino, LinkSprite, SM5100B, GSM Communication, SMS server

Setup:

Hardware:

Arduino Uno^[2]

LinkSprite SM5100B^[3]

GSM Antenna^[4]

Sparkfun SM5100B breakout board^[3]

5V Power Supply (up to 2A of current)

Software:

Arduino IDE^[1]



Figure 1

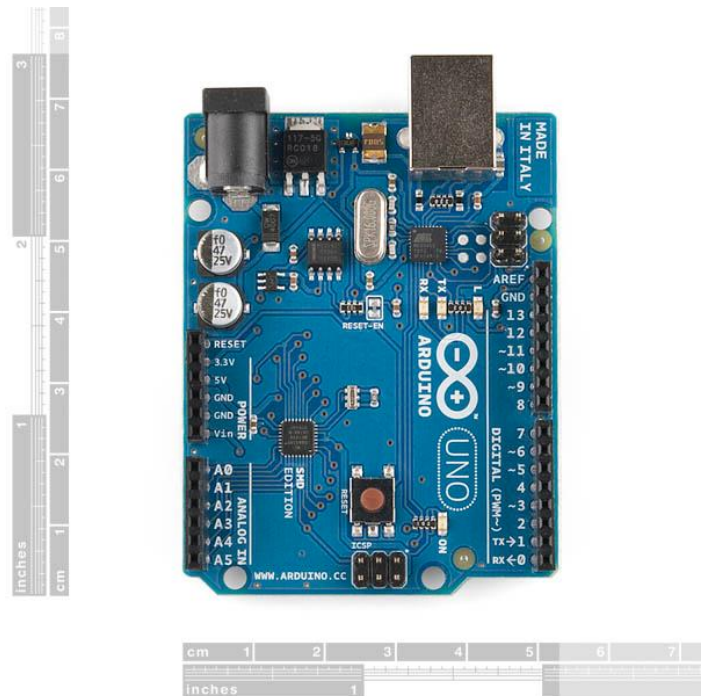


Figure 2

1. Download and install the Arduino IDE onto your computer and follow the steps to install the drivers for your board after connecting it to a USB serial port on your computer.
2. Install and solder pin headers to your SM5100B breakout board so that the board will fit directly into the Arduino stackable headers shown in Figure 3. Or you can directly wire pins(2->2),(3->3),(5V->(3.3V-4.2V), and (GND-> GND).
3. Attach your antenna to the SM5100B device and insert your SIM card into the SIM holder attached to the Sparkfun breakout board.
4. Power on your device using a 5V power supply rated at 2Amps of current. A red LED should be shown on the top of the board, and a green LED should be noticeable when looking at the Arduino board.



Figure 3

Configure your GSM module:

Now that your chip is properly connected to the Arduino platform, we can now control and monitor signals coming from the GSM module and additionally configure some settings essential to making sure your device will function properly.

1. Load your Arduino IDE
2. Upload the following serial monitoring sketch available on page 5.
3. Open the Arduino IDE serial monitor box and make sure that it is configured at 9600 baud. Please reference Figure 4 for what should be presented. If the letters are illegible please change the baud rate and reset the device using the reset push-button until you can see the following in Figure 5. To change the baud rate that the GSM module sends bytes type the command `At+IPR =9600` and press enter on the serial monitoring program. This will set the baudrate to 9600 which is the most common baudrate for serial communications using the Arduino platform.

0 SIM card removed	6 Released call whose ID=<idx>
1 SIM card inserted	7 The network service is available for an emergency call
2 Ring melody	8 The network is lost
3 AT module is partially ready	9 Audio ON
4 AT module is totally ready	10 Show the status of each phonebook after init phrase
5 ID of released calls	11 Registered to network

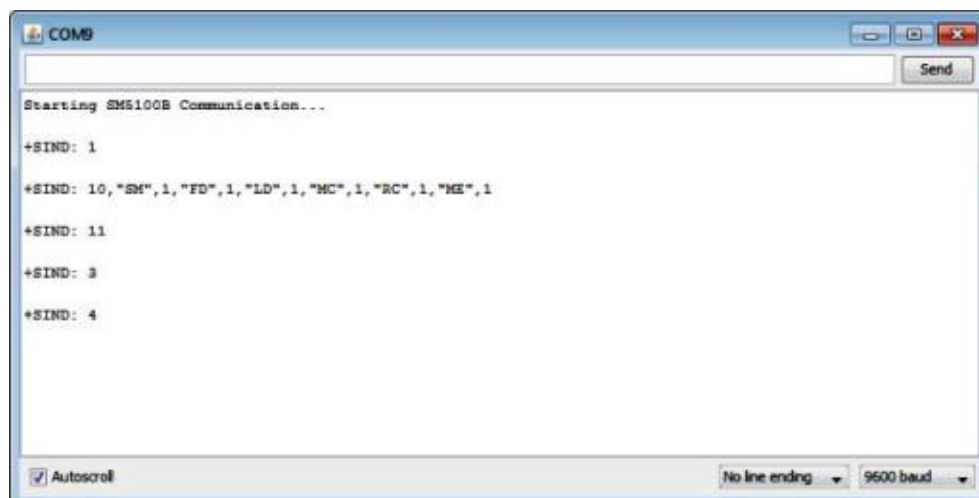
Figure 4

Figure 4 is a breakdown of the commands received from the device during the registration phase.

+SIND: 8 means that the connection between your antenna and the network is below threshold or the GSM Module is operating on the wrong frequency. To check these, use the commands AT+CSQ=? (Cell Service Quality), and AT+SBAND=? (Operating Frequency). To view the full command list please visit [5].

```
//Serial Monitoring Sketch for Arudino and SM5100B
/* SparkFun Cellular Shield - Pass-Through Sample Sketch
SparkFun Electronics Written by Ryan Owens CC by v3.0 3/8/10
*/
#include <NewSoftSerial.h>
//Include the NewSoftSerial library to send serial commands to the cellular module.
#include <string.h>
//Used for string manipulations
char incoming_char=0;
//Will hold the incoming character from the Serial Port.
NewSoftSerial cell(2,3);
//Create a 'fake' serial port. Pin 2 is the Rx pin, pin 3 is the Tx pin.
void setup()
{
    //Initialize serial ports for communication.
    Serial.begin(9600);
    cell.begin(9600);
    Serial.println("Starting SM5100B Communication...");
}
void loop()
{
    //If a character comes in from the cellular module...
    if(cell.available() >0)
    {
        incoming_char=cell.read(); //Get the character from the cellular serial port.
        Serial.print(incoming_char); //Print the incoming character to the terminal.
    }
    //If a character is coming from the terminal to the Arduino...
    if(Serial.available() >0)
    {
        incoming_char=Serial.read(); //Get the character coming from the terminal
        cell.print(incoming_char); //Send the character to the cellular module.
    }
}
```

Figure 5



Server Setup:

Finally, we want to setup our device to act a server that can be powered full-time by either battery power or grid-power. To do this we need to create a loop that will set our device to read ASCII data bytes from the serial connection and constantly check for SMS messages. To do this we will use the following sketch.

1. Create a new blank Arduino sketch.
2. Insert and upload the following sketch from Page 8 which will act as a server to control a single LED on a digital output port on the microcontroller.
3. Plug in and turn on, and send a text message to it using #a1 and #a0

- 1 represents LED-ON and 0 represents LED off.

Notes:

- The GSM Network in the United States operates on SBAND 7. To set this use AT+SBAND=7
- The GSM Module requires a regulated voltage of 3.3V to 4.2V. Do not plug the 5V line from the Arduino directly to the GSM module battery pin-out. For a schematic of the module please visit [6].
- In idle mode the GSM Module can pull less than 7mA and can last on a single battery for quite some time. During a low service cell phone call the module can pull up to 2Amps of power and drain a battery cell very quickly.

```

#include <NewSoftSerial.h>
//Include the NewSoftSerial library to send serial commands to the cellular module.
char inchar;
//Will hold the incoming character from the Serial Port.
NewSoftSerial cell(2,3);
//Create a 'fake' serial port. Pin 2 is the Rx pin, pin 3 is the Tx pin.
int led1 = 9;
//control LED to signal that we recieved a command to turn on a digital I/O port.
void setup()
{ // prepare the digital output pins
pinMode(led1, OUTPUT);
digitalWrite(led1, LOW);
//Initialize GSM module serial port for communication.
cell.begin(9600);
delay(30000);
// give time for GSM module to register on network etc.
cell.println("AT+CMGF=1");
// set SMS mode to text
delay(200);
cell.println("AT+CNMI=3,3,0,0");
// set module to send SMS data to serial out upon receipt
delay(200);
}
void loop()
{
//If a character comes in from the cellular module...
if(cell.available() >0)
{
    inchar=cell.read();
    if (inchar=='#') // # sign signals the reception of a command and acts as a 'key'
    {
        delay(10);
        inchar=cell.read();
        if (inchar=='a')
        {
            delay(10);
            inchar=cell.read();
            if (inchar=='0')
            {
                digitalWrite(led1, LOW);
            }
            else if (inchar=='1')
            {
                digitalWrite(led1, HIGH);
            }
            delay(10);
            cell.println("AT+CMGD=1,4"); // delete all SMS
        }
    }
}
}

```

Sources:

[1] <http://www.arduino.cc>

[2] <http://www.sparkfun.com/products/10356>

[3] <http://www.sparkfun.com/products/9607>

[4] <http://www.sparkfun.com/products/8347>

[5] <http://www.sparkfun.com/datasheets/CellularShield/SM5100B%20AT%20Command%20Set.pdf>

[6] <http://www.sparkfun.com/datasheets/Cellular%20Modules/CEL-09533-User%27s%20Manual.pdf>