

GSM-BASED SWITCH

PROJECT CODE: GP 03

BY THE ELECTRONICS DR

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PROTOTYPING PROJECT | | | | | |
|  | FIELD  Electronics Engineering |  | SPECIALIZATION  Prototyping |  | ASPECT  Programming |

STATEMENT OF THE PROBLEM

Table of Contents

[ABSTRACT 4](#_Toc18609707)

[OBJECTIVES 4](#_Toc18609708)

[ABSTRACT 4](#_Toc18609709)

[DESIGN SECTION 6](#_Toc18609710)

[APPLICATIONS OF THE CIRCUIT 6](#_Toc18609711)

[CONCLUSION 7](#_Toc18609712)

# A laptop computer, notebook, mug, pen and plant arranged on a blue surfacePROJECT BACKGROUND

## OBJECTIVES

The following were the goals of this project

* Analyzing the client specifications and developing the needed Bill of Materials.
* To develop the logic flow and Arduino sketch of the program as per the client’s specifications.
* To simulate the program using Proteus to assess its viability prior to implementing it physically.

## ABSTRACT

As per the information from you, it was clear that this project would be implemented on an arcade machine such that when a customer cuts a string it would open the door. With the consideration, that pressure from the switch would be released. After looking at the system, I developed an Arduino code to aid in sending the SMS each time the customer cuts the string.

# DESIGN SECTION

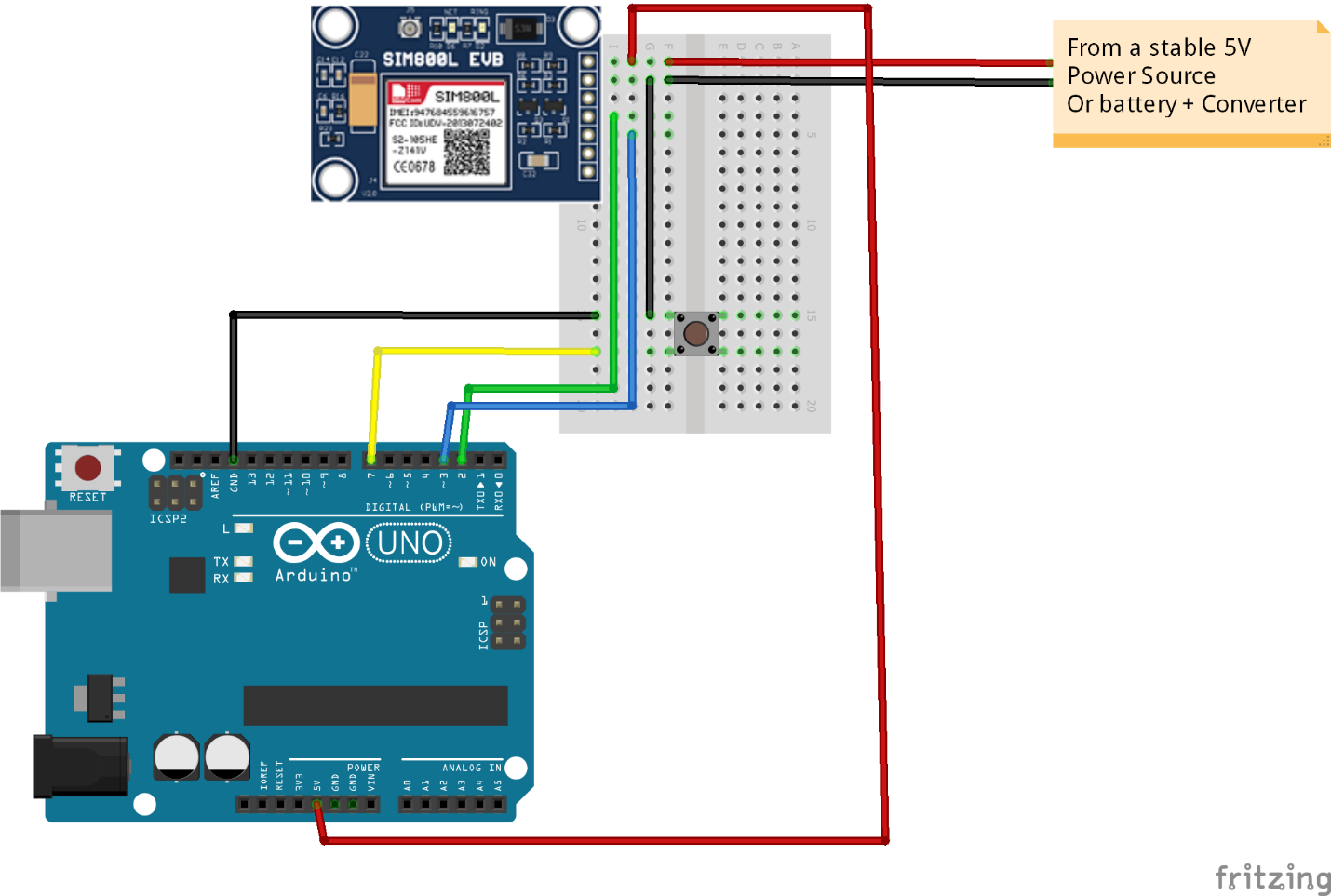
## PRECAUTIONS

1. Disable the sim card lock before connecting the SIM card to the GSM module- please make sure to remove the sim card lock by connecting the sim card to your phone and go to the Settings>>Set up sim card lock then disable it.
2. Load some credit to the sim card- the carrier SMS charges will apply when sending the messages so be sure to load some credit on to your desired sim card.
3. Solder the male to male header pins in place- if you have recently purchased the GSM module please make sure you’ve soldered the header pins in appropriately make sure the heat is not above 2500c for safety purposes

## CONNECTIONS

Just to get the module to properly function, please make the connections as follows;

|  |  |
| --- | --- |
| ARDUINO BOARD | GSM MODULE- SIM 800C |
| Digital pin 2 | Rx |
| Digital pin 3 | TX |
| GND | GND |
| +5V | +5V |

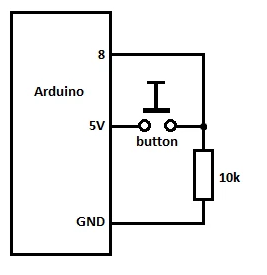


*Figure illustrating the connections of the GSM module with Arduino*

Kindly note: For your case where you may not require the external power supply because the Arduino will power your GSM module with 5V power supply. Therefore, you can ignore the external battery in the above technical illustration

CONNECTING THE BUTTON

|  |  |
| --- | --- |
| ARDUINO BOARD | BUTTON |
| Digital pin 7 | One pin of the button |
| GND | The other pin of the button |



*Figure illustrating the Arduino connection with the button through a pull up resistor.*

Kindly note the pin 8 is just for illustration, I changed the code for the button to connect through the digital pin 7. Therefore while making the hardware connection please connect to the digital pin 7 and not 8.

## SOFTWARE IMPLEMENTATION

This step was primarily developing the Arduino code as per the client specifications, needs and requirements for the project

Below is the final code in plain text in addition the screenshots of the project

*/\* This code works with Sim800C and a push button*

*\* Press the button to send a simple SMS/Text to a specified phone number*

*\* Refer to the 'Electronics Dr' for more details*

*\*/*

*#include <SoftwareSerial.h>*

*SoftwareSerial sim800c(2, 3); // RX,TX for Arduino and for the module it's TXD RXD, they should be inverted*

*#define button 7 //Button pin, on the other pin it's wired with GND*

*bool button\_State; //Button state*

*void setup()*

*{*

*pinMode(button, INPUT); //The button is always on HIGH level, when pressed it goes LOW*

*sim800c.begin(9600); //Module baude rate, this is on max, it depends on the version*

*Serial.begin(9600);*

*delay(1000);*

*}*

*void loop()*

*{*

*button\_State = digitalRead(button); //We are constantly reading the button State*

*if (button\_State == HIGH) { //And if it's pressed*

*Serial.println("Button pressed"); //Shows this message on the serial monitor*

*delay(200); //Small delay to avoid detecting the button press many times*

*SendSMS(); //And this function is called*

*}*

*if (sim800c.available()){ //Displays on the serial monitor if there's a communication from the module*

*Serial.write(sim800c.read());*

*}*

*}*

*void SendSMS()*

*{*

*Serial.println("Sending SMS..."); //Show this message on serial monitor*

*sim800l.print("AT+CMGF=1\r"); //Set the module to SMS mode*

*delay(100);*

*sim800l.print("AT+CMGS=\"+\*\*\*\*\*\*\*\*\*\"\r"); //Your phone number don't forget to include your country code, example +212123456789"*

*delay(500);*

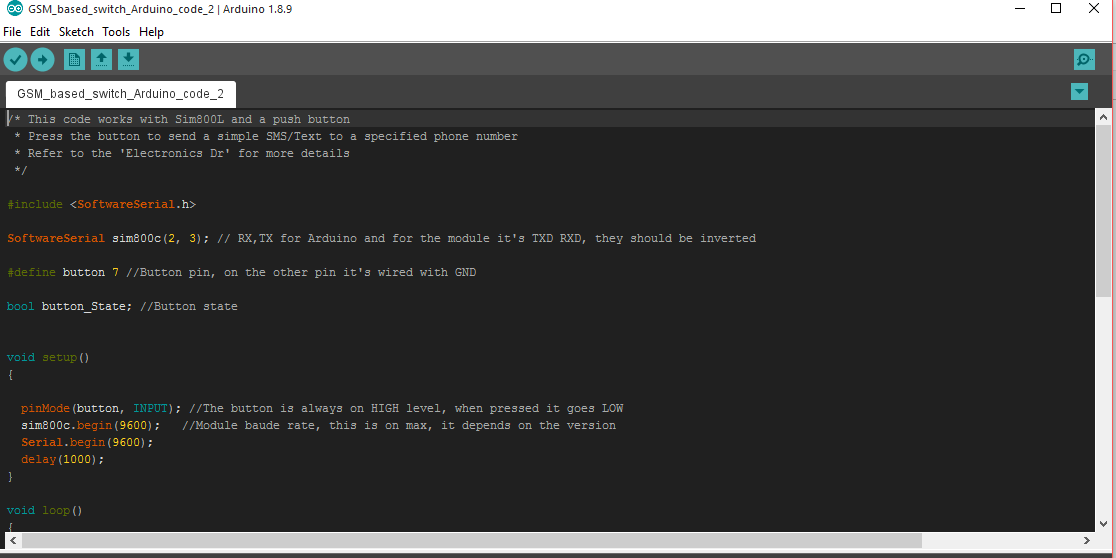
*sim800l.print("Congratulations, A prize has been WON"); //This is the text to send to the phone number, don't make it too long or you have to modify the SoftwareSerial buffer*

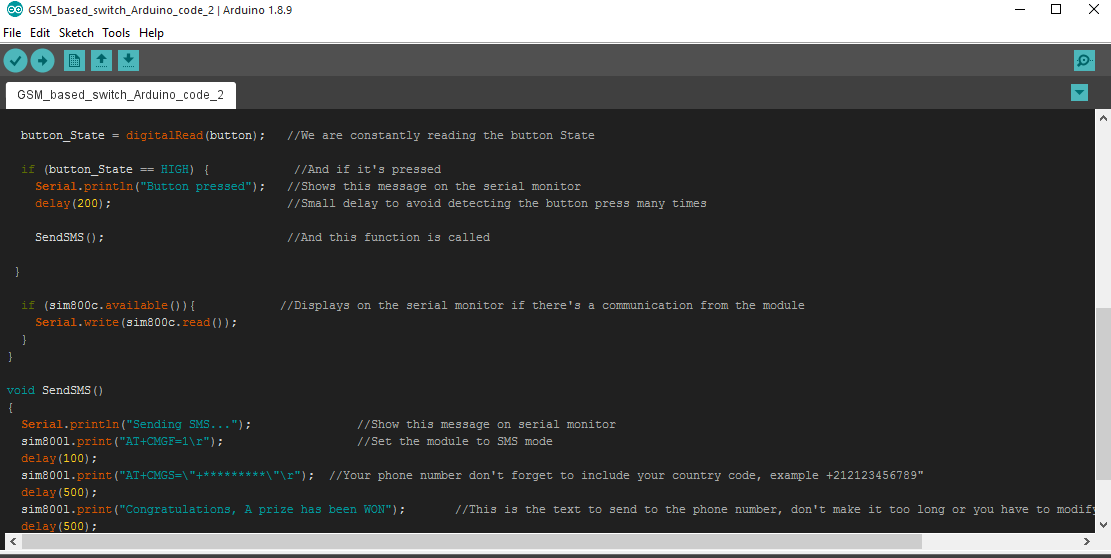
*delay(500);*

*sim800l.print((char)26);// (required according to the datasheet)*

*delay(500);*

*}*





### CODE EXPLANATION

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);**

Therefore, in our code, pin number 2 will act as Rx of Arduino and 3 will act as Tx of Arduino.  Let’s get to 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. Our 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 here in our tutorial.  Configuration part is over with setting baud rates and its good to give a small delay of 100 milli seconds.

**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). Returns the first byte of incoming serial data.

**Sim800c.available()** – checks for any data coming from GSM module through the SoftwareSerial pins 9 and 10. Returns the number of bytes available to read from software serial port. Returns a -1 if no data is available to read.

**Sim800c.read()** – Reads the incoming data through software serial port.

**Serial.write()** – Prints data to serial monitor of arduino. So the function Serial.write(sim800c.read()) – prints the data collected from software serial port to serial monitor of arduino.

Lets get the functions **SendMessage()**

These are the functions in which we actually send commands to GSM module from Arduino. These commands to communicate with GSM module are called AT Commands. There are different commands to perform different tasks using the GSM module. You can read complete AT Commands Library to understand all that is possible with GSM module.

**SendMessage()** – is the function we 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 sim800c.println() function. Sim800c.println writes data to software serial port (the TX pin of our Software Serial – that is pin 10) and this will be captured by GSM module (through its Rx pin). After setting the GSM module to Text mode, we then set 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 sim800c.println((char)26); Each and every AT command may be followed by 1 second delay. We must give some time for GSM module to respond properly. Once these commands are send to GSM module, you shall receive an SMS in the set mobile number.

# Overhead view of hands shaking over a business papersCONCLUSION

The project implementation can be deemed a success and it is always a pleasure to work with you. I have made sure to attach all the relevant documents in regards to this project. I believe in being open and transparent as such in case of anything just let me know. I look forward to implementing more projects with you in future.