GSM Web Server

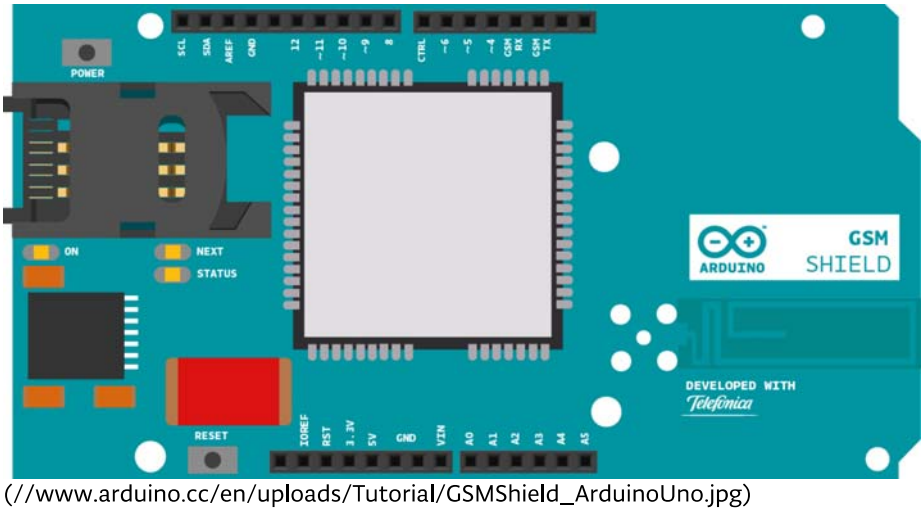
This sketch turns the Arduino or Genuino board with the GSM shield and a data enabled SIM card into a web server. When the board receives a request from a connected client, it sends back the value of analog inputs 0-5.

Not all network operators allow incoming data requests from outside their network. This means you can create a web server with the GSM shield, but you may not be able to connect to it from the public internet; only from another data enabled device from the same provider on the same network. You should check with your provider to see what specific policies they have in place regarding incoming data connections.

Hardware Required

- Arduino or Genuino Board
- Arduino + Telefonica GSM/GPRS Shield (//www.arduino.cc/en/Main/ArduinoGSMShield)
- SIM card enabled for Data
- (optional) 6 potentiometers or other analog inputs attached to A0-A5

Circuit



(//www.arduino.cc/en/uploads/Tutorial/GSMShield_ArduinoUno.jpg)
image of the Arduino GSM Shield on top of an Arduino or Genuino board

Optional analog sensors like photoresistors, potentiometers and such may be connected, as explained elsewhere in our tutorials, to pins A0 - A5

Code

First, import the GSM library

```
#include <GSM.h>
```

SIM cards may have a PIN number that enables their functionality. Define the PIN for your SIM. If your SIM has no PIN, you can leave it blank :

```
#define PINNUMBER ""
```

Define a number of constants that contain information about the GPRS network you're going to connect to. You'll need the Access Point Name (APN), login, and password. To obtain this information, contact your network provider for the most up to date information. This page (http://forums.pinstack.com/f24/tcp_apn_wap_gateway_port_carrier_settings-360/) has some information about various carrier settings, but it may not be current.

```
#define GPRS_APN       "GPRS_APN"
#define GPRS_LOGIN     "login"
#define GPRS_PASSWORD  "password"
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=2)

Initialize instances of the classes you're going to use. You're going to need the GSM, GPRS, and GSMServer classes. When you instantiate the GSMServer class, you'll need to tell it which port to listen for incoming connections. Port 80 is the default port for HTTP requests.

```
GPRS gprs;
GSM gsmAccess;
GSMServer server(80);
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=3)

In `setup`, open a serial connection to the computer. After opening the connection, send a message indicating the sketch has started.

```
void setup(){
  Serial.begin(9600);
  Serial.println("Starting Arduino web client.");
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=4)

Create a local variable to track the connection status. You'll use this to keep the sketch from starting until the SIM is connected to the network :

```
boolean notConnected = true;
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=5)

Connect to the network by calling `gsmAccess.begin()`. It takes the SIM card's PIN as an argument. You'll also connect to the GPRS network using `gprs.attachGPRS()`. This requires the APN, login, and password you declared earlier. By placing this inside a `while()` loop, you can continually check the status of the connection and wait for them to both become `true` before proceeding.

When the modem does connect and has attached itself to the GPRS network, `gsmAccess()` will return `GSM_READY`. Use this as a flag to set the `notConnected` variable to `true` or `false`. Once connected, the remainder of `setup` will run.

```
while(notConnected)
  {
    if(gsmAccess.begin(PINNUMBER)==GSM_READY)
        (gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD)==GPRS_READY))
      notConnected = false;
    else
    {
      Serial.println("Not connected");
      delay(1000);
    }
  }
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=6)

Start the server using `server.begin()`. You can request the server's IP address with `grps.getIPAddress()` and end the setup.

```
server.begin();

  IPAddress LocalIP = gprs.getIPAddress();
  Serial.println("Server IP address=");
```

```
  Serial.println(LocalIP);
}
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=7)

In `loop`, create an instance of `GSMClient` and check if there are any active connections

```
void loop() {
  GSMClient client = server.available();

  if (client)
  {
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=8)

While the client is connected, and there is data waiting to be read, begin to read the request. Read through the available bytes until a newline character has been received.

In this instance, you won't actually do anything with the request, it's assumed that it is a HTTP request, and you'll serve up a web page.

```
while (client.connected())
    {
      if (client.available())
      {
        Serial.println("Receiving request!");
        bool sendResponse = false;
        while(char c=client.read()) {
          if (c == '\n') sendResponse = true;
        }
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=9)

Once the request has been read, start to send a standard HTTP response header with `client.print()` and `client.println()`.

```
if (sendResponse)
       {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          client.println("<html>");
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=10)

Read through the analog inputs and send the values to the client.

```
for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(analogRead(analogChannel));
            client.println("<br />");
          }
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=11)

Send a closing tag for the webpage, and stop the client connection before closing the `loop`.

```
client.println("</html>");
          //necessary delay
          delay(1000);
          client.stop();
        }
      }
    }
  }
}
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=12)

Once your code is uploaded, open the serial monitor. Once the IP address is printed to the serial monitor, enter it into a web browser. You should see a webpage that reports the analog input values on each the Arduino's six inputs.

if you cannot connect to the IP address, make sure your network operator enables incoming traffic.

The complete sketch is below.

```
/*
 GSM Web Server

 A simple web server that shows the value of the analog input pins.
 using a GSM shield.

 Circuit:
 * GSM shield attached
 * Analog inputs attached to pins A0 through A5 (optional)

 created 8 Mar 2012
 by Tom Igoe
 */

// libraries
#include <GSM.h>

// PIN Number
#define PINNUMBER ""

// APN data
#define GPRS_APN       "GPRS_APN" // replace your GPRS APN
#define GPRS_LOGIN     "login"    // replace with your GPRS login
#define GPRS_PASSWORD  "password" // replace with your GPRS password


// initialize the library instance
GPRS gprs;
GSM gsmAccess;     // include a 'true' parameter for debug enabled
GSMServer server(80); // port 80 (http default)

// timeout
const unsigned long __TIMEOUT__ = 10 * 1000;

void setup() {
  // initialize serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // connection state
  boolean notConnected = true;

  // Start GSM shield
  // If your SIM has PIN, pass it as a parameter of begin() in quotes
  while (notConnected) {
    if ((gsmAccess.begin(PINNUMBER) == GSM_READY) &
        (gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD) == GPRS_READY)) {
      notConnected = false;
    } else {
      Serial.println("Not connected");
      delay(1000);
    }
  }

  Serial.println("Connected to GPRS network");

  // start server
  server.begin();

  //Get IP.
  IPAddress LocalIP = gprs.getIPAddress();
  Serial.println("Server IP address=");
  Serial.println(LocalIP);
}

void loop() {


  // listen for incoming clients
  GSMClient client = server.available();
```

```
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        Serial.println("Receiving request!");
        bool sendResponse = false;
        while (char c = client.read()) {
          if (c == '\n') {
            sendResponse = true;
          }
        }

        // if you've gotten to the end of the line (received a newline
        // character)
        if (sendResponse) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          client.println("<html>");
          // output the value of each analog input pin
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(analogRead(analogChannel));
            client.println("<br />");
          }
          client.println("</html>");
          //necessary delay
          delay(1000);
          client.stop();
        }
      }
    }
  }
}
```

[Get Code] (//www.arduino.cc/en/Tutorial/GSMExamplesWebServer?action=sourceblock&num=1)

See Also

- GPRS (//www.arduino.cc/en/Reference/GPRSConstructor) Constructor
- attachGPRS (//www.arduino.cc/en/Reference/AttachGPRS)()
- GSMServer (//www.arduino.cc/en/Reference/GSMServerConstructor) Constructor
- ready (//www.arduino.cc/en/Reference/GSMServerReady)()
- beginWrite (//www.arduino.cc/en/Reference/GSMServerBeginWrite)()
- write (//www.arduino.cc/en/Reference/GSMServerWrite)()
- endWrite (//www.arduino.cc/en/Reference/GSMServerEndWrite)()
- read (//www.arduino.cc/en/Reference/GSMServerRead)()
- available (//www.arduino.cc/en/Reference/GSMServerAvailable)()
- stop (//www.arduino.cc/en/Reference/GSMServerStop)()

- Arduino GSM Shield (//www.arduino.cc/en/Main/ArduinoGSMShield) – Complete product description.
- Getting started with the GSM Shield (//www.arduino.cc/en/Guide/ArduinoGSMShield) – Get everything set up in minutes.
- GSM library (//www.arduino.cc/en/Reference/GSM) – Your reference for the GSM Library.

- GSMExamplesMakeVoiceCall (//www.arduino.cc/en/Tutorial/GSMExamplesMakeVoiceCall) - How to make a voice call with mic and speaker.
- GSMExamplesReceiveVoiceCall (//www.arduino.cc/en/Tutorial/GSMExamplesReceiveVoiceCall) - The call is received and connected, the number that is calling is shown on serial monitor and then the call is hung up.

- GSMExamplesReceiveSMS (//www.arduino.cc/en/Tutorial/GSMExamplesReceiveSMS) - How to receive an SMS message.
- GSMExamplesSendSMS (//www.arduino.cc/en/Tutorial/GSMExamplesSendSMS) - How to send an SMS entering number and text through the serial monitor.
- GSMExamplesWebClient (//www.arduino.cc/en/Tutorial/GSMExamplesWebClient) - Connect to the Arduino.cc home and print the contents on the serial monitor window.
- GSMToolsTestGPRS (//www.arduino.cc/en/Tutorial/GSMToolsTestGPRS) – Tries to access the internet over GPRS with supplied APN and credentials.

Last revision 2015/08/13 by SM

Share