Arduino GSM/GPRS Shield

Shield-ul Arduino GSM/GPRS permite placii Arduino sa se conecteze la reteaua de Internet, sa trimita sau sa primeasca mesaje SMS si sa realizeze apeluri de voce. Shield-ul este compatibil cu placa Arduino UNO, fara nici un fel de modificare. Pentru placile Arduino Mega si Arduino Leonardo sunt necesare cateva modificari minore. Momentan, placa Due nu este compatibila cu shield-ul.

In acest tutorial vei descoperi cum se realizeaza o conexiune la reteaua de Internet si cum se poate programa o aplicatie care citeste temperatura si o incarca la fiecare 5 secunde, intr-un serviciu de stocare al datelor online. Shield-ul GSM/GPRS arata ca in imaginea de mai jos (fara modificarea necesara)

Pentru acest tutorial, vei avea nevoie de urmatoarele componente:

- O placa Arduino UNO - http://www.robofun.ro/arduino/arduino_uno_v3

  SAU
- O placa Arduino Mega - http://www.robofun.ro/arduino_mega2560
- Un shield Arduino GSM/GPRS - http://www.robofun.ro/arduino_gsm_gprs_shield
- Cablu USB.
- Fire de conexiune tata-tata

  http://www.robofun.ro/fire_conexiune_tata_tata-140mm

- Un alimentator extern pentru Arduino (9V @ 1A)
  http://www.robofun.ro/alimentator-extern-arduino
- Un senzor de temperatura brick.
  http://www.robofun.ro/senzor-temperatura-brick?keyword=temperatura&category_id=0

Modificarea necesara pentru placa Arduino Mega

Daca folosesti placa Arduino UNO, sari peste aceasta sectiune. Daca folosesti Arduino Mega sau Arduino Leonardo, fa pasii care urmeaza :

- pinul 2 al shield-ului se indoaie deoarece nu trebuie sa fie conectat cu placa Arduino.
- pinul 2 al shield-ului se conecteaza printr-un fir cu capete tata-tata la pinul 10.

Pentru celelalte placi, respectiv pentru placa Arduino UNO, nu este necesara nici o modificare. Nu trebuie decat sa conectezi shield-ul si atat. Pentru placa Leonardo, urmeaza aceeasi pasi ca mai sus cu diferenta ca pinul 2 se conecteaza la pinul 8.

Conexiunea la Internet

Conexiunea la Internet se realizeaza printr-o cartela SIM. Cartela necesita configurarea in prealabil cu optiunea de Internet pe mobil. Acest pas il poti realiza citind instructiunile providerului de la care detii cartela (vei gasi instructiunile online sau pe pagina lor de internet). Iti sunt necesare si urmatoarele date de conectare (pe care le vei gasi online sau dupa ce ai configurat cartela cu serviciul de Internet):

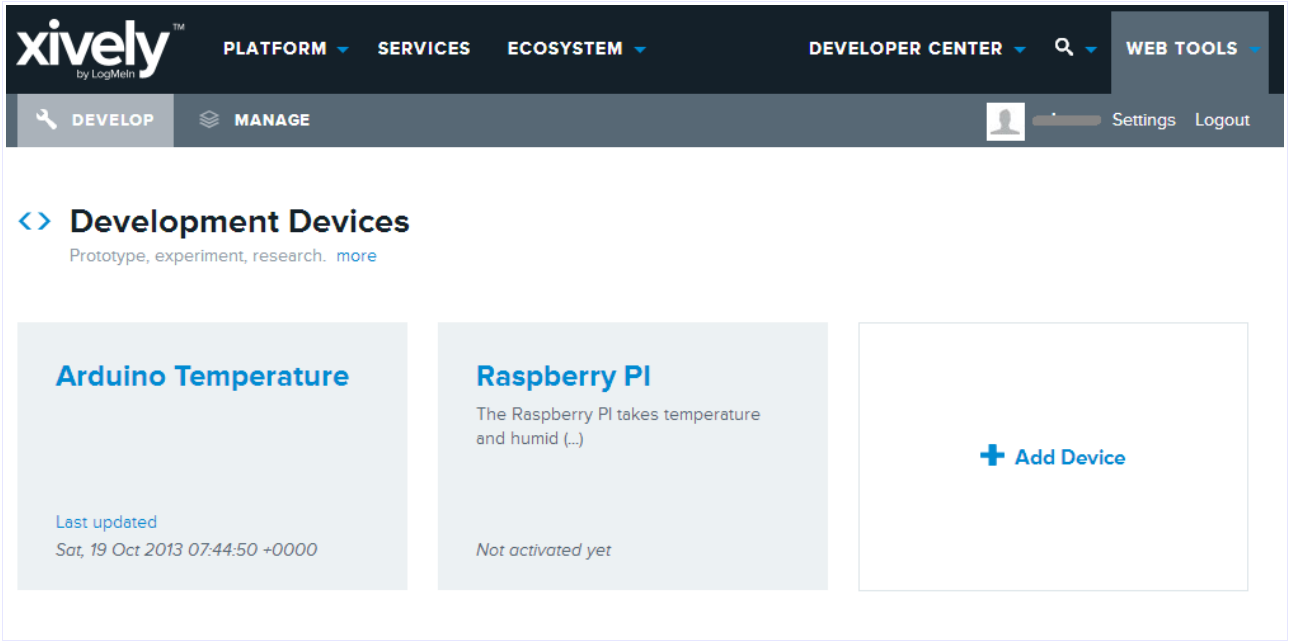
- GPRS APN
- GPRS Login
- GPRS Password

Datele de conectare le vei folosi mai tarziu in sketch, dar pentru moment asigura-te ca ai realizat pasul precedent.

Stocarea temperaturii online

In continuare vei folosi o conexiune Internet peste reteaua de telefonie 3G pentru a stoca temperatura citita de Arduino de la un senzor pe serviciul online Xively.com. Pagina de web iti pune la dispozitie nu doar inregistrarea temperaturii dar si evolutia in timp, prin grafice.

Pentru aceasta, ai nevoie de un cont pe care il inregistrezi la adresa: https://xively.com/ iar aici iti vei adauga primul dispozitiv pe care il vei numi dupa preferinta ta (numele il vei utiliza mai tarziu in sketch).



Dupa ce ai creat primul dispozitiv, asigura-te ca ai urmatoarele informatii, deoarece iti vor fi necesare in sketch-ul de mai jos:

- API KEY
- FEED ID
- Numele proiectului (ales anterior)

Conectarea shield-ului si a senzorului de temperatura.

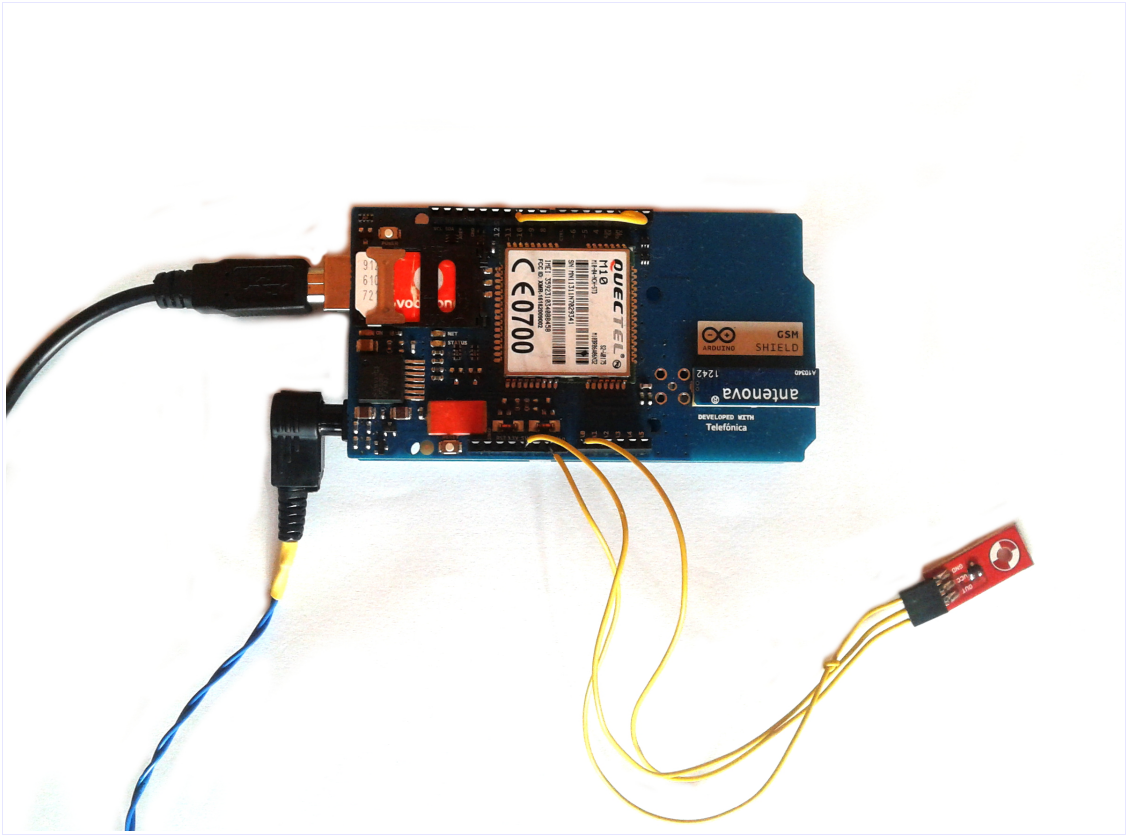
Conecteaza componentele in urmatoarea ordine:

- Shield-ul GSM/GPRS (cu modificarea pentru Arduino Mega, daca ai utilizat Arduino Mega). Acesta se infige, foarte usor, in placa Arduino.
- Cartela SIM se introduce in slotul shield-ului.
- Senzorul de temperatura brick se conecteaza dupa urmatorul tabel:

Arduino PIN GND	Senzor brick PIN GND
Arduino PIN VCC	Senzor brick PIN VCC
Arduino PIN A0	Senzor brick PIN OUT

- Conecteaza sursa de alimentare de 9V si cablul USB. Atentie ! Nu se recomanda alimentarea shield-ului direct din USB, fara alimentator.

Modemul de pe placa shield-ului consuma mult mai mult decat poate sa genereze portul USB. Acest lucru poate duce la o functionare incorecta sau, in cel mai rau caz, la o defectiune.



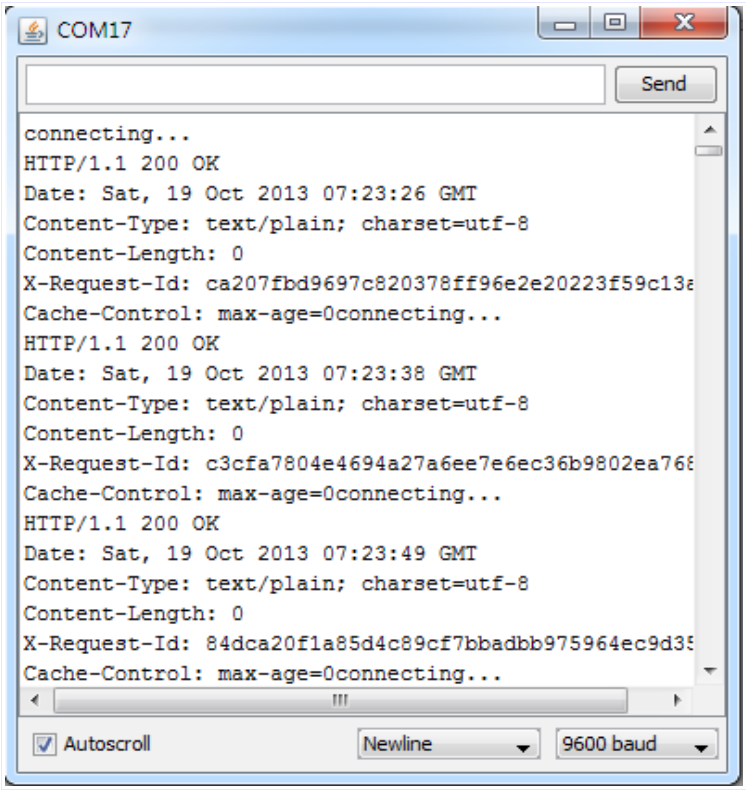
Codul sursa.

Copiaza cu copy/paste codul sursa de mai jos. Cauta in cod liniile urmatoare si modifica-le cu datele personale:

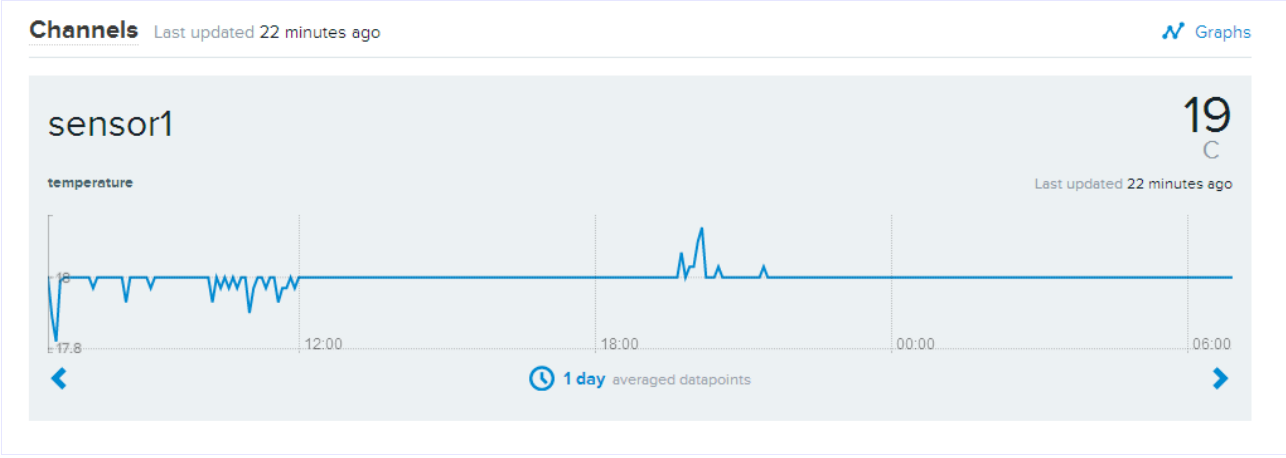
```
#define APIKEY          "API KEY"
#define FEEDID          0000000000
#define USERAGENT       "Project Name"

#define GPRS_APN        "live.vodafone.com"
#define GPRS_LOGIN      "live"
#define GPRS_PASSWORD   "vodafone"
```

Incarca codul in placa Arduino si deschide Serial Monitor. Asteapta cateva momente si vei obtine urmatoarea imagine.



Deschide pagina unde ti-ai creat contul si ar trebui sa obtii, dupa o perioada mai indelungata de timp, evolutia in timp a temperaturii sub forma de grafic.



http://www.robofun.ro/forum

```
#include <GSM.h>

// Pachube Client data
#define APIKEY         "API KEY"  // replace your pachube api key
here
#define FEEDID         0000000000                    // replace
your feed ID
#define USERAGENT      "Project Name"              // user agent is
the project name
// PIN Number
#define PINNUMBER ""
// APN data
#define GPRS_APN       "live.vodafone.com"  // replace your GPRS
APN
#define GPRS_LOGIN     "live"     // replace with your GPRS login
#define GPRS_PASSWORD  "vodafone"  // replace with your GPRS
password
// initialize the library instance:
GSMClient client;
GPRS gprs;
GSM gsmAccess;
char server[] = "api.pachube.com";      // name address for pachube
API


unsigned long lastConnectionTime = 0;         // last time you
connected to the server, in milliseconds
boolean lastConnected = false;                  // state of the
connection last time through the main loop
const unsigned long postingInterval = 10*1000;  //delay between
updates to Pachube.com

void setup()
{
  Serial.begin(9600);
  while (!Serial) {
    ;
  }
  boolean notConnected = true;

  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);
```

```
    }
  }
}

void loop()
{
  int sensorReading = readTempInCelsius(10,0);

  if (client.available())
  {
     char c = client.read();
     Serial.print(c);
  }

  if (!client.connected() && lastConnected)
  {
    client.stop();
  }

  if(!client.connected() && ((millis() - lastConnectionTime) >
postingInterval))
  {
  sendData(sensorReading);
  }

  lastConnected = client.connected();
}

/*
  Conexiunea HTTP cu server-ul.
*/
void sendData(int thisData)
{
  if (client.connect(server, 80))
  {
    Serial.println("connecting...");

    // send the HTTP PUT request:
    client.print("PUT /v2/feeds/");
    client.print(FEEDID);
    client.println(".csv HTTP/1.1");
    client.println("Host: api.pachube.com");
    client.print("X-ApiKey: ");
    client.println(APIKEY);
    client.print("User-Agent: ");
    client.println(USERAGENT);
    client.print("Content-Length: ");
```

```
    // calculate the length of the sensor reading in bytes:
    // 8 bytes for "sensor1," + number of digits of the data:
    int thisLength = 8 + getLength(thisData);
    client.println(thisLength);

    // last pieces of the HTTP PUT request:
    client.println("Content-Type: text/csv");
    client.println("Connection: close");
    client.println();

    // here's the actual content of the PUT request:
    client.print("sensor1,");
    client.println(thisData);
  }
  else
  {
    // if you couldn't make a connection:
    Serial.println("connection failed");
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
  }
  // note the time that the connection was made or attempted
  lastConnectionTime = millis();
}

/*
  This method calculates the number of digits in the
  sensor reading.  Since each digit of the ASCII decimal
  representation is a byte, the number of digits equals
  the number of bytes.
*/
int getLength(int someValue)
{
  // there's at least one byte:
  int digits = 1;

  // continually divide the value by ten,
  // adding one to the digit count for each
  // time you divide, until you're at 0:
  int dividend = someValue /10;
  while (dividend > 0)
  {
    dividend = dividend /10;
    digits++;
  }
```

```
  // return the number of digits:
  return digits;
}

float readTempInCelsius(int count, int pin) {
  float temperaturaMediata = 0;
  float sumaTemperatura;
  for (int i =0; i<count; i++) {
    int reading = analogRead(pin);
    float voltage = reading * 5.0;
    voltage /= 1024.0;
    float temperatureCelsius = (voltage - 0.5) * 100 ;
    sumaTemperatura = sumaTemperatura + temperatureCelsius;
  }
  return sumaTemperatura / (float)count;
}
```