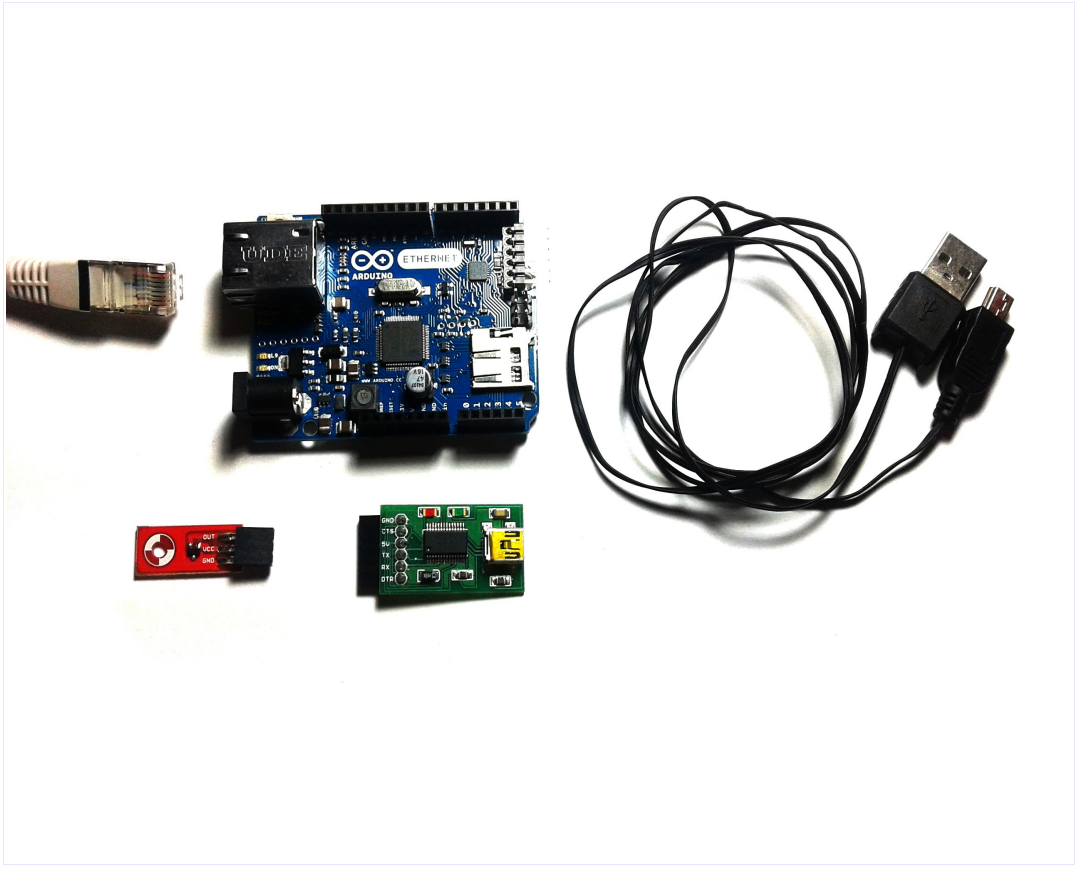
Arduino Ethernet – senzor de temperatura – Xively.com

In acest tutorial vei descoperi cum se conecteaza un senzor de temperatura brick la o placa Arduino Ethernet si cum se poate programa placa pentru a incarca valoarea temperaturii, pe un serviciu online de stocare al datelor denumit Xively.

Pentru acest tutorial, iti vor fi necesare urmatoarele componente:

- O placa Arduino Ethernet
  - http://www.robofun.ro/arduino/arduino_ethernet_w_o_poe
- Un modul FTDI
  - http://www.robofun.ro/conector-ftdi-5v
- Fire de conexiune tata-tata
  - http://www.robofun.ro/fire_conexiune_tata_tata-140mm
- Un senzor de temperatura brick
  http://www.robofun.ro/senzor-temperatura-brick

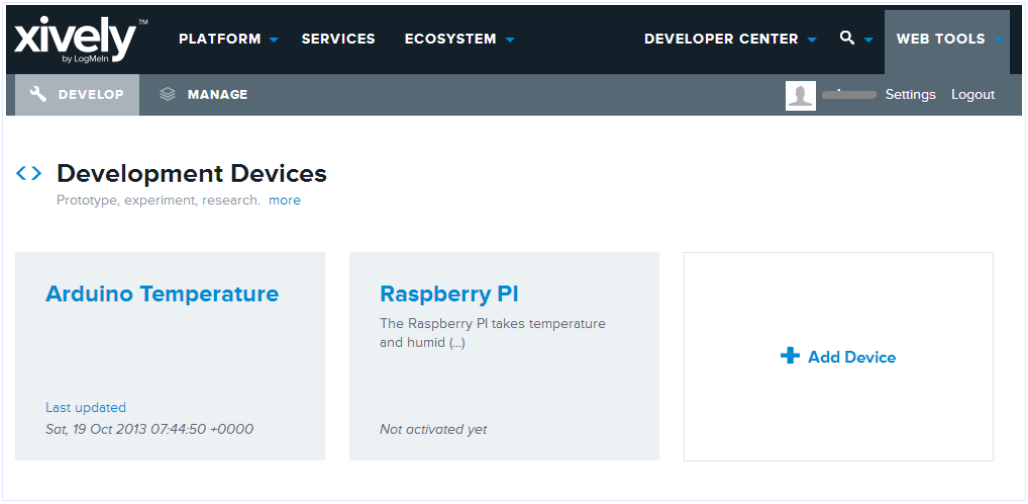
- Un cablu mini USB
- Acces prin cablu la Ethernet



Stocarea temperaturii online.

Xively.com este un serviciu care permite stocarea informatiei preluata de la senzori. Pagina de web iti pune la dispozitie nu doar inregistrarea temperaturii dar si evolutia in timp, prin grafice. Pentru asta, 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).

Asa arata 2 dispozitive adaugate.



Dupa ce ai creat dispozitivul, asigura-te ca ai urmatoarele informatii, pentru ca iti vor fi necesare in sketch:

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

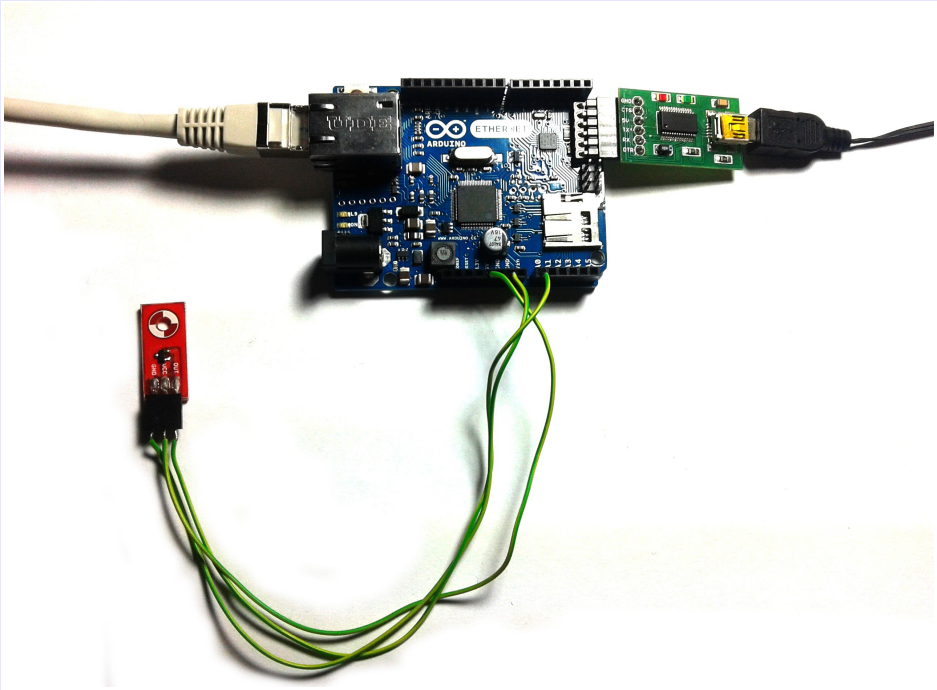
Cum functioneaza ?

1. Senzorul brick de temperatura se conecteaza la placa Arduino Ethernet, urmand tabelul de mai jos:

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

2. Conectorul FTDI se conecteaza prin cablu mini USB, la calculator si prin conector la placa Arduino Ethernet.

3. Placa Arduino Ethernet se conecteaza la router, prin cablul Ethernet.



Codul sursa

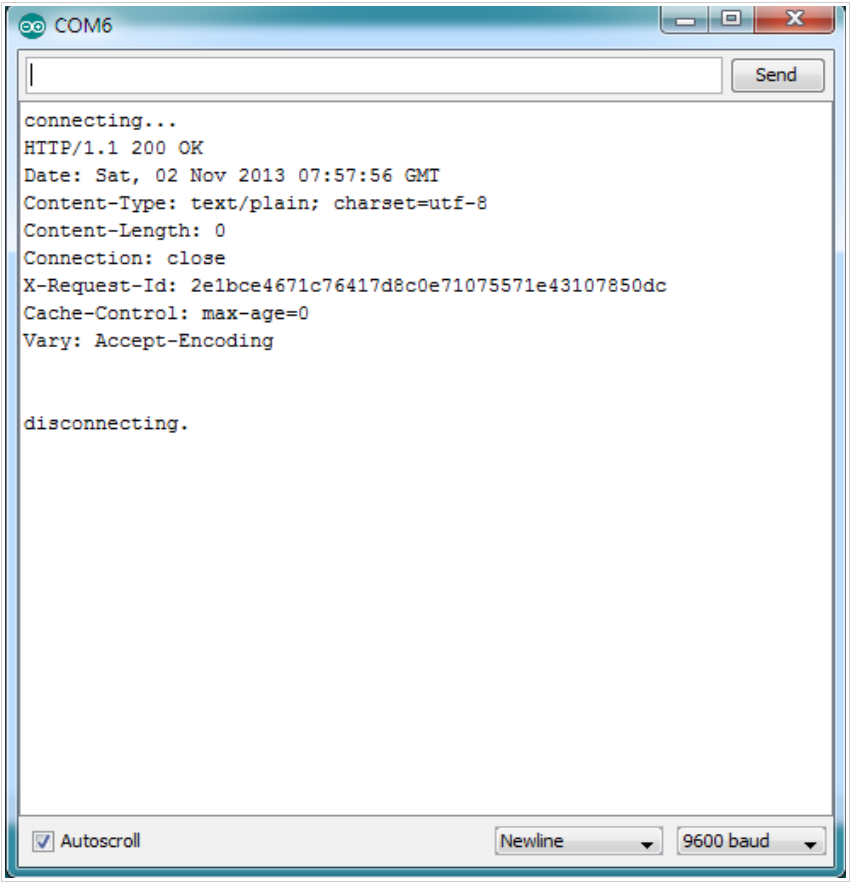
Copiaza cu copy/paste codul sursa de mai jos. Cauta in cod liniile urmatoare si modifica-le cu datele personale:

```
#define APIKEY          "api_key"
#define FEEDID          999999999999
#define USERAGENT       "Arduino Temperature"
```

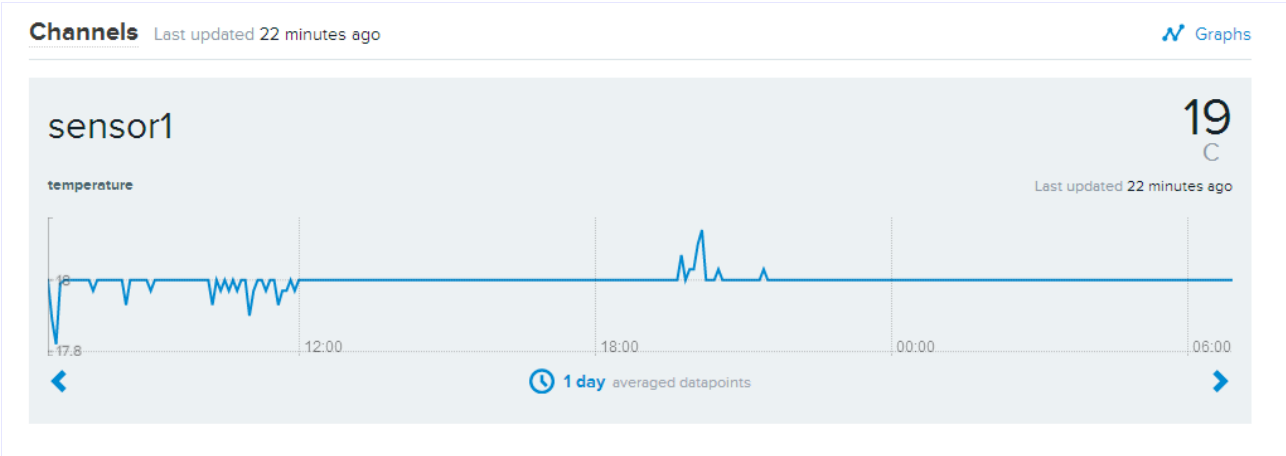
Cauta in cod urmatoarea linie si modifica adresa IP conform setarilor tale din router.

```
IPAddress ip(192,168,2,110);
```

Incarca sketch-ul in router si deschide Serial Monitor. Iti vor aparea urmatoarele date, ca in imaginea de mai jos.



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.



```
#include <SPI.h>
#include <Ethernet.h>

#define APIKEY         "api_key" // replace your pachube api key
here
#define FEEDID         9999999999 // replace your feed ID
#define USERAGENT      "Arduino Temperature" // user agent is the
project name

byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
```

```
IPAddress ip(192,168,2,110);
EthernetClient client;

IPAddress server(216,52,233,122);
char server[] = "api.pachube.com";

unsigned long lastConnectionTime = 0;
boolean lastConnected = false;
const unsigned long postingInterval = 10*1000;

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

  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    Ethernet.begin(mac, ip);
  }
}

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

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

  if (!client.connected() && lastConnected) {
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
  }

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

// this method makes a HTTP connection to the server:
void sendData(int thisData) {
  // if there's a successful connection:
  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-PachubeApiKey: ");
    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();
}

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;
}
```