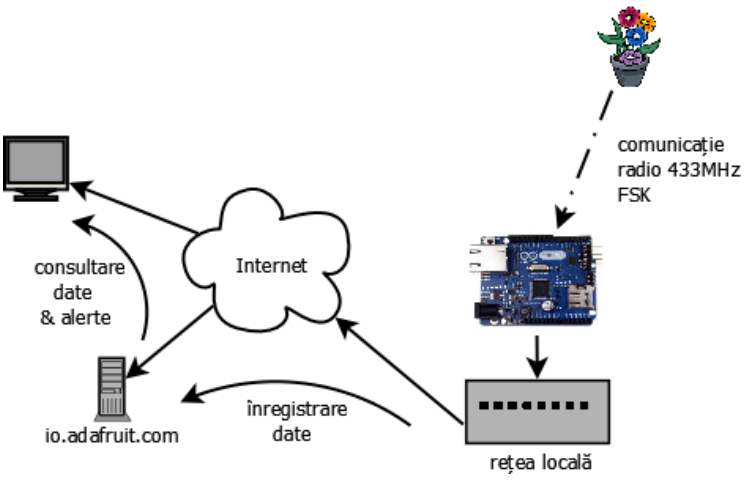
Cum faci ca ghivecele de flori să-ți trimită email când le e prea cald sau nu au fost udate?
sau
Cum conectezi ghivecele de flori în Cloud?

Componente necesare

Modul bază:
Arduino Ethernet
Brick 5V RFM12B
LCD alfanumeric 2x16 I2C
Alimentator sau baterie 9V

Modul ghiveci / achiziție:
Reduino Radio
Senzor de temperatură TMP36
Brick led
Regulator step-up U1V10F3 3.3V
Baterie 1.5V

Descrierea proiectului



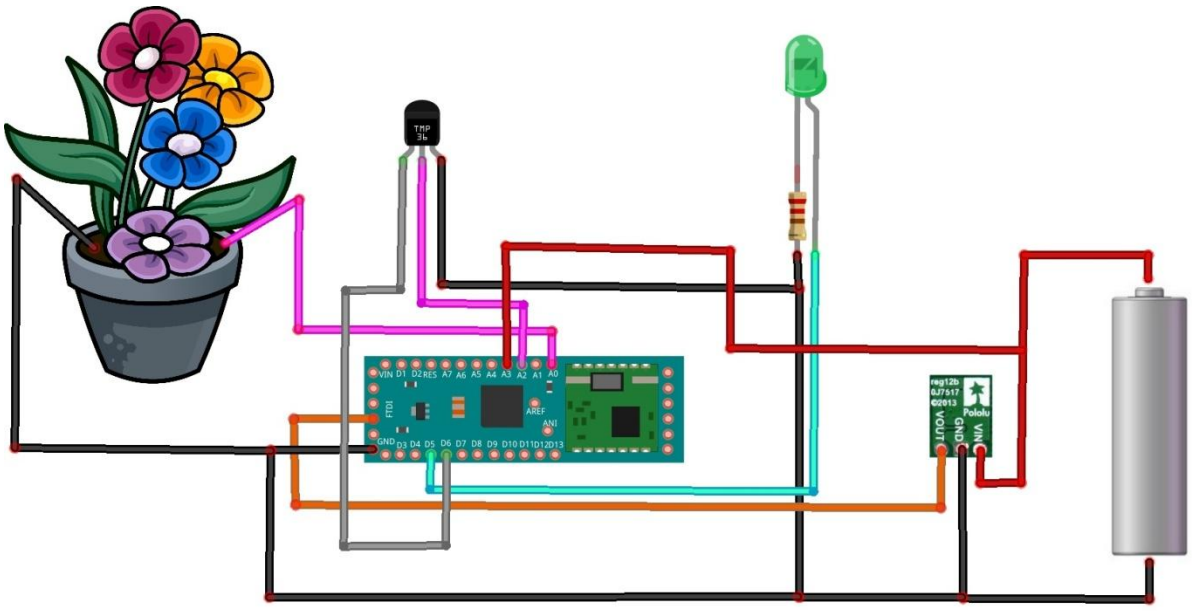
Proiectul își propune să realizeze un sistem de urmărire a umidității solului și a temperaturii ambientale pentru ghivecele cu flori. Sistemul de urmărire va permite raportarea către utilizator în timp real a valorilor parametrilor măsurați prin intermediul Internetului, realizarea de grafice de variație și definirea de alerte prin email pentru anumite valori limită. Arhitectura sistemului de urmărire se bazează pe un modul bază ce preia informațiile prin radio de la modulele de achiziție și le transmite prin Internet către serverul io.adafruit [1] de unde pot fi accesate de oriunde de pe Internet. Comunicația radio între modulele de achiziție și modulul bază se va realiza în frecvență de 433MHz (bandă ISM [2]) modulație FSK [3] și va fi asigurată de module radio RFM12B de la HopeRF [4]. Datele primite prin radio vor fi înregistrate prin intermediul protocolului

MQTT [5] pe serverul io.adafruit ce va stoca și prelucra informațiile oferind posibilitatea de a le consulta sub formă de grafice și de a genera alerte prin email.

Modulul de achiziție

Modulul de achiziție ce va deservi un ghiveci va măsura umiditatea solului și temperatura ambientală și va transmite aceste informații prin radio către modulul bază. Temperatura ambientală va fi măsurată cu ajutorul senzorului analogic TMP36 [6] iar umiditatea solului cu ajutorul a două lamele de metal înfipte în pământul din ghiveci la o distanță de circa 1cm. Platforma de bază pentru modulul de achiziție este placa Reduino Radio [7] bazată pe un microcontroler Atmega328P ce funcționează la frecvența de 8MHz și este alimentat la o tensiune de 3.3V (întregul montaj va funcționa la această tensiune). Placa de dezvoltare Reduino Radio este similară cu placa Arduino Pro Mini 328 – 3.3V/8MHz [8] dar integrează și un modul radio RFM12B variantă SMD [4] permițând astfel dezvoltarea simplă de proiecte de comunicație radio (nu mai este necesară interconectarea plăcii de dezvoltare cu modulul radio). Pentru programarea plăcii Reduino Radio este necesar să utilizăm un conector FTDI 3.3V [9] (la fel ca și în cazul Arduino Pro Mini 328 – 3.3V). Modulul de achiziție mai dispune și de un led care va permite verificarea funcționării corecte a montajului – deoarece nu are afișaj și nici nu este gândit să funcționeze conectat la calculator, ledul de ”stare” este singura componentă de interfațare cu utilizatorul a montajului. Se poate utiliza un led brick [10] sau un led împreună cu o rezistență de 220 ohm. Alimentarea modului se va face de la o baterie de 1.5V utilizând un regulator step-up U1V10F3 [11] ce va ridica tensiunea bateriei de la 1.5V la 3.3V – tensiunea de funcționare a montajului.

Schema de interconectare a componentelor este următoarea:



Un aspect foarte important în funcționarea modulului de achiziție este consumul extrem de redus în funcționarea curentă – după cum se va vedea în detalierea următoare a programului, microcontrolerul și modulul radio vor fi ținute în starea de sleep [12] (consum redus) și se vor ”trezi” doar o dată la o oră pentru a citi senzorii și a transmite radio către modulul bază valorile citite. Singurul lucru ce va funcționa în acest interval este ledul de stare ce se va aprinde o secundă o dată la un minut – pentru a permite verificarea funcționării corecte a modulului. Tot din motiv de conservare a energiei bateriei, se poate observa și din schema de interconectare, senzorul de temperatură și lamelele de metal ce permit măsurarea umidității solului vor fi alimentate doar în momentul măsurării parametrilor. Pinul Vcc al senzorului de temperatură este conectat la pinul digital D6 ce va asigura curentul necesar funcționării senzorului doar pe perioada de măsurare, senzorul necesită extrem de puțin curent (mai puțin de 1mA) deci nu este o problemă să fie alimentat în acest mod (pinii digitali ai microcontrolerului pot debita până la 40mA). Pinul A0 ce va măsura umiditatea solului va funcționa și ca alimentare pentru divizorul de tensiune ce se formează și permite evaluarea cantității de apă din sol - se poate vedea principiul de funcționare în [13], singura diferență este utilizarea rezistenței de pull-up interne a pinului A0 în cadrul montajului modulului de achiziție – rezistența de pull-up va fi activă doar pe perioada de măsurare. Această modalitate, alimentarea limitată, mai are avantajul de a proteja lamelele de metal de o oxidare accelerată. Cu alte cuvinte, cu excepția ledului de stare, întregul montaj va ”dormi” o oră între fiecare măsurătoare prelungind semnificativ durata de viață a bateriei – montajul poate funcționa pe baza unei baterii AA între 3 și 6 luni. O altă funcție a montajului este supravegherea tensiunii bateriei (a gradului de descărcare), se poate observa conectarea liniei de ”+” a bateriei la pinul A3, raportarea radio va include și acest parametru măsurat.

Programul va necesita includerea bibliotecii de comunicație radio JeeLib [14] precum și definirea parametrilor de comunicație (adresă nod, subrețea, frecvență de comunicație):

```
#include <JeeLib.h>

#define myNodeID 11
#define network 210
#define freq RF12_433MHZ
```

precum și definirea pinilor utilizați în schema de interconectare (led – D5, senzor umiditate sol – A0, senzor temperatură – A2 și D6 pentru alimentare, baterie – A3) :

```
#define led_pin 5
#define soil_pin A0
#define temp_pin A2
#define temp_vcc 6
#define bat_pin A3
```

Comunicația radio se va baza pe o structură proprie (*ParametriiTX*) instanțiată în variabila *parametrii*:

```
typedef struct { int soil_humidity; float temperature; float bat_voltage;}
```

```
                                                    ParametriiTX;
ParametriiTX parametrii;
```

În cadrul secțiunii *setup* se vor inițializa pinii utilizați în program și se va inițializa comunicația radio:

```
void setup() {
   pinMode(led_pin,OUTPUT);
  pinMode(soil_pin,INPUT);
  pinMode(temp_pin,INPUT);
  pinMode(temp_vcc,OUTPUT);
  pinMode(bat_pin,INPUT);
  digitalWrite(led_pin,LOW);
  digitalWrite(temp_vcc,LOW);
  Sleepy::loseSomeTime(1000);
  rf12_initialize(myNodeID,freq,network); }
```

O funcție importantă ce necesită definirea în program este ISR-ul (subrutina de întrerupere) pentru mecanismul de watchdog [15] ce ”trezește” microcontrolerul din starea de sleep:

```
ISR(WDT_vect) { Sleepy::watchdogEvent(); }
```

În cadrul secțiunii *loop* se vor citi cele trei valori (umiditate sol, temperatură ambientală și tensiune baterie), se vor trimite prin conexiunea radio și va trage un ”pui de somn” de o oră:

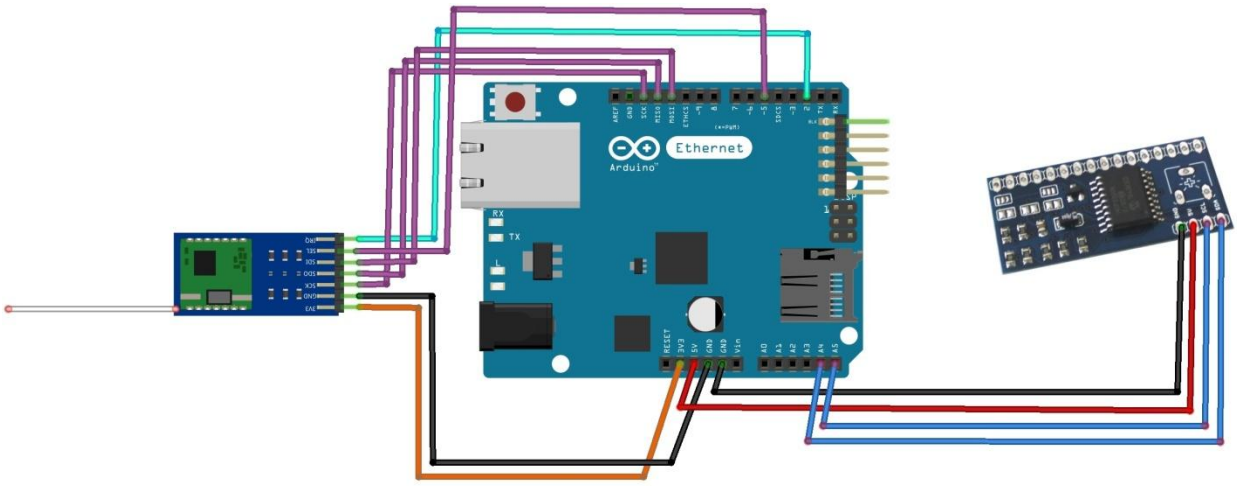
```
void loop() {
  pinMode(soil_pin,INPUT_PULLUP);
  Sleepy::loseSomeTime(1000);
  parametrii.soil_humidity = analogRead(soil_pin);
  pinMode(soil_pin,INPUT);
  digitalWrite(temp_vcc,HIGH);
  Sleepy::loseSomeTime(1000);
  parametrii.temperature = (analogRead(temp_pin)*3300.0)/1024.0;
  parametrii.temperature = (parametrii.temperature - 500.0) / 10.0;
  digitalWrite(temp_vcc,LOW);
  parametrii.bat_voltage = (analogRead(bat_pin)*3.3)/1024.0;
  rf12_sendNow(0, &parametrii, sizeof parametrii);
  rf12_sendWait(0);
  for (int i=0; i<60; i++) {
    rf12_sleep(RF12_SLEEP);
    // 60.000 ms = 1 minut
    Sleepy::loseSomeTime(60000);
    rf12_sleep(RF12_WAKEUP);
    digitalWrite(led_pin,HIGH);
    Sleepy::loseSomeTime(1000);
    digitalWrite(led_pin,LOW);  } }
```

În program trebuie observată succesiunea de alimentare și citire a celor doi senzori (umiditate sol și temperatură). Somnul de o oră constă în 60 de perioade a câte un minut permițând astfel funcționarea (aprinderea) ledului de stare.

Întregul montaj poate fi instalat într-o cutie de plastic de dimensiuni 10cm x 6cm x 2.5cm rezultând astfel un sistem compact ce poate fi atașat ușor unui ghiveci cu flori. Trebui ținut seama că în cutie trebuie făcute trei orificii pentru antenă, ledul de stare și firele lamelelor ce măsoară umiditatea solului. Atenție!!! Nu este o idee bună lăsarea dispozitivului în lumina directă a soarelui, mai ales vara…

Modulul bază

Modulul bază este compus dintr-o placă Arduino Ethernet [16], un modul brick radio RFM12B 5V [17] și un afișaj LCD alfanumeric 2x16 I2C [18]. Programarea plăcii Arduino Ethernet necesită un conector FTDI 5V [19]. Schema de interconectare a componentelor este următoarea:



Placa Arduino Ethernet poate fi înlocuită cu un ansamblu compus din Arduino Uno [20] și Shield Ethernet [21]. Conectarea componentelor și programul nu necesită nici o modificare și nu mai este nevoie de utilizarea conectorului FTDI.

Alimentarea sistemului se poate face prin intermediul mufei de alimentare prezentă pe placa Arduino Ethernet de la o baterie de 9V sau de la un alimentator de 9V (recomandat). Placa se va conecta printr-un cablu de rețea la un switch sau un router ce va oferi setările de rețea dinamic (prin DHCP).

Utilizarea LCD-ului I2C reduce numărul de fire fiind utilizate doar cele de alimentare (5V, GND) și SCL (conectat la A5) / SDA (conectat la A4) – liniile de magistrală I2C. Din punct de vedere software, utilizarea LCD-ului I2C necesită instalarea bibliotecii LiquidTWI [22]. Modulul brick RFM12B 5V utilizează magistrala SPI a plăcii de dezvoltare SCK – pinul D13 / SCK, SDO – pinul D12 / MISO, SDE – pinul D11 / MOSI, SEL – pinul D5 precum și IRQ – pinul D2. Alimentarea brick-ului se va face la 3.3V deoarece modulul radio RFM12B funcționează la 3.3V – una dintre funcțiile asigurate de brick este translatarea nivelurilor logice de la 5V (tensiunea de funcționare a pinilor plăcii de dezvoltare) la 3.3V. Deoarece magistrala SPI este folosită și de controlerul ethernet prezent pe placa de dezvoltare Arduino Ethernet nu vom putea folosi pinul D10 pentru selecția modulului RFM12B – pin implicit în biblioteca JeeLib [14]. Din acest motiv este necesară modificarea bibliotecilor JeeLib și Ethernet pentru ca cele două componente să poată coexista – modificarea este descrisă în [23].

Programul necesar funcționării modului bază este destul de mare din acest motiv vom puncta în acest material doar secțiunile cele mai importante. Bibliotecile necesare funcționării programului sunt:

```
#include <SPI.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

#include <Ethernet.h>
#include <EthernetClient.h>
#include <Dns.h>
#include <Dhcp.h>

#include <TimeLib.h>
#include <EthernetUdp.h>

#include <JeeLib.h>

#include <Wire.h>
#include <LiquidTWI.h>
```

Biblioteca Adafruit MQTT [24] este necesară pentru implementarea protocolului MQTT ce va deservi comunicația cu serverul io.adafruit. Modalitatea de postare a datelor este preluat din exemplul *mqtt_ethernet* al bibliotecii. Tot din acest exemplu este preluată și procedura *MQTT_connect()* care va fi folosită în programul nostru.

Vom utiliza biblioteca Time [25] pentru a obține ora exactă din rețea prin intermediul protocolului NTP [26]. Procedurile *getNtpTime()*,*sendNTPpacket()*, *digitalClockDisplay()* și *printDigits()*, utilizate în cod, sunt preluate din exemplul *TimeNTP* al acestei biblioteci.

Structurile de date necesare comunicației MQTT sunt următoarele (trebuie personalizate *AIO_USERNAME* și *AIO_KEY* conform contului creat pe io.adafruit):

```
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "…"
#define AIO_KEY         "…"

const char MQTT_SERVER[] PROGMEM    = AIO_SERVER;
const char MQTT_CLIENTID[] PROGMEM  = __TIME__ AIO_USERNAME;
const char MQTT_USERNAME[] PROGMEM  = AIO_USERNAME;
const char MQTT_PASSWORD[] PROGMEM  = AIO_KEY;

Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, AIO_SERVERPORT,
                            MQTT_CLIENTID,MQTT_USERNAME, MQTT_PASSWORD);

const char TEMPERATURA_FEED[] PROGMEM = AIO_USERNAME "/feeds/temperatura";
const char HUMIDITY_FEED[] PROGMEM = AIO_USERNAME "/feeds/humidity";
const char VOLTAGE_FEED[] PROGMEM = AIO_USERNAME "/feeds/voltage";

Adafruit_MQTT_Publish temperatura = Adafruit_MQTT_Publish(&mqtt,
                                                    TEMPERATURA_FEED);
Adafruit_MQTT_Publish humidity = Adafruit_MQTT_Publish(&mqtt, HUMIDITY_FEED);
Adafruit_MQTT_Publish voltage = Adafruit_MQTT_Publish(&mqtt, VOLTAGE_FEED);
```

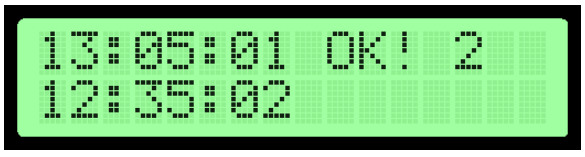
Se pot observa cele trei canale (feeds) de postare a datelor: *temperatura*, *humidity* și *voltage* pentru cele trei tipuri de informații salvate pe server.

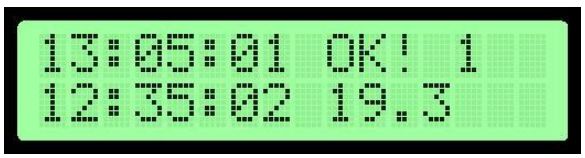
În cadrul secțiunii *setup()* vom inițializa conexiunea radio, conexiunea de rețea și vom efectua sincronizarea de ceas:

```
void setup() {
  rf12_initialize(myNodeID,freq,network);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print(F("Init the Client..."));
  lcd.setCursor(0,1);
  if (Ethernet.begin(mac) == 0) { lcd.print(F("Failed DHCP")); while(1); }
  else
    { lcd.print("IP");
      for (byte thisByte = 0; thisByte < 4; thisByte++) {
        lcd.print(Ethernet.localIP()[thisByte], DEC);
        lcd.print(".");
      }
    }
  Udp.begin(localPort);
  setSyncProvider(getNtpTime);
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,1);
  digitalClockDisplay(); }
```

În cadrul secțiunii *loop()* se va inițializa (*MQTT_connect()*) și se va verifica periodic (*mqtt.ping()*) conexiunea MQTT, se va verifica primirea unui mesaj radio (*rf12_recvDone()*) și în cazul primirii unui mesaj valid de la modulul de achiziție se vor trimite datele (*---.publish()*) către serverul io.adafruit. Chiar dacă modulul de achiziție este programat să trimită informații la un interval de o oră, programul modului bază face o verificare suplimentară și nu permite postarea de informații la un interval mai mic de 10 secunde (*long interval = 10000;*) pentru a nu încălca regulile de utilizare a serviciului io.adafruit [27] în cazul funcționării anormale a modulului de achiziție (de exemplu repetări repetate ale modulului de achiziție).

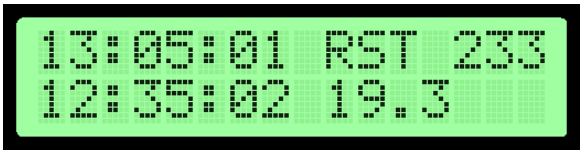
```
void loop() {
 MQTT_connect();
 if (rf12_recvDone()){
  if (rf12_crc == 0 && (rf12_hdr & RF12_HDR_CTL) == 0) {
    int node_id = (rf12_hdr & 0x1F);
    if (node_id == extNodeID1)  {
       mesaj1=*(ParametriiRX1*) rf12_data;
       lcd.setCursor(0,1);
       digitalClockDisplay();
       boolean ok = 1;
       unsigned long currentMillis1 = millis();
       if(currentMillis1 - previousMillis1 > interval) {
        if (! temperatura.publish(mesaj1.temperature)) ok = 0;
        if (! humidity.publish((uint32_t)mesaj1.soil_humidity)) ok = 0;
        if (! voltage.publish(mesaj1.bat_voltage)) ok = 0;
        previousMillis1 = currentMillis1;
       }
       lcd.setCursor(9,1);
       if (ok) { lcd.print(mesaj1.temperature); lcd.print("  "); }
       else lcd.print(F("Failed"));
  }  }   }
 if(! mqtt.ping()) {
    mqtt.disconnect();
  }

 if (timeStatus() != timeNotSet) {
    if (now() != prevDisplay) {
      prevDisplay = now();
      lcd.setCursor(0,0);
      digitalClockDisplay();  }
  }
}
```





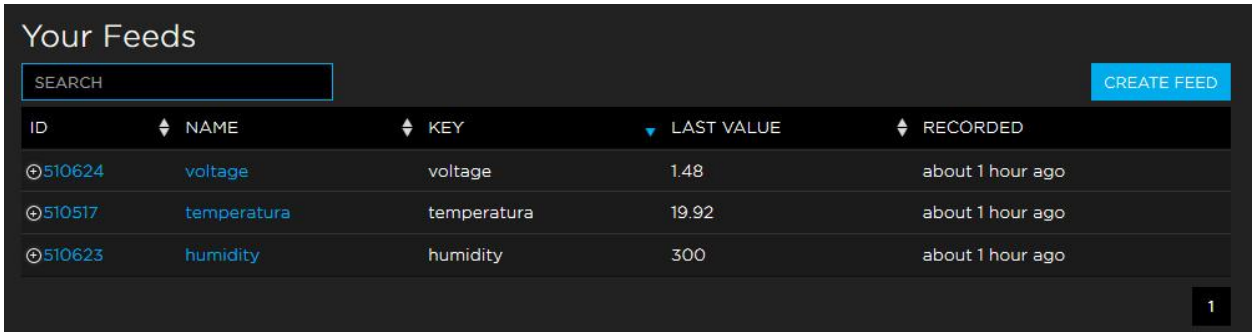
Afișajul LCD al modulului va permite afișarea de informații de stare pentru verificarea funcționării sistemului fără a conecta modulul bază la calculator. Prima linie a afișajului va afișa ora curentă și mesajele OK! (conexiune MQTT funcțională) sau RST (conexiune MQTT pierdută, încercare de reconectare). În

cazul RST afișarea orei curente se va opri la momentul pierderii conexiunii. Numărul de după mesajul OK! reprezintă numărul de încercări până la obținerea conexiunii MQTT iar numărul de după mesajul RST reprezintă numărul de tentative de conectare în curs. Dacă numărul de tentative de conectare atinge valoarea 360 (încercările de conectare se fac o dată la 5 secunde, 360 de încercări durează aproximativ 30 de minute) sistemul se va autoreseta (*resetFunc()*) deoarece se presupune că este vorba de o blocare a controlerului ethernet sau a comunicației cu routerul local. Linia a doua a ecranului LCD afișează ora pornirii sistemului (dacă nu este urmată de nici un alt număr) sau ora primirii ultimului mesaj de la modulul de achiziție și în acest caz este urmată de afișarea parametrului temperatură din respectivul mesaj. În acest fel se poate urmării întreaga funcționare a sistemului: ora la care a repornit sistemul, ora ultimului mesaj radio și starea și stabilitatea funcționării conexiunii MQTT.

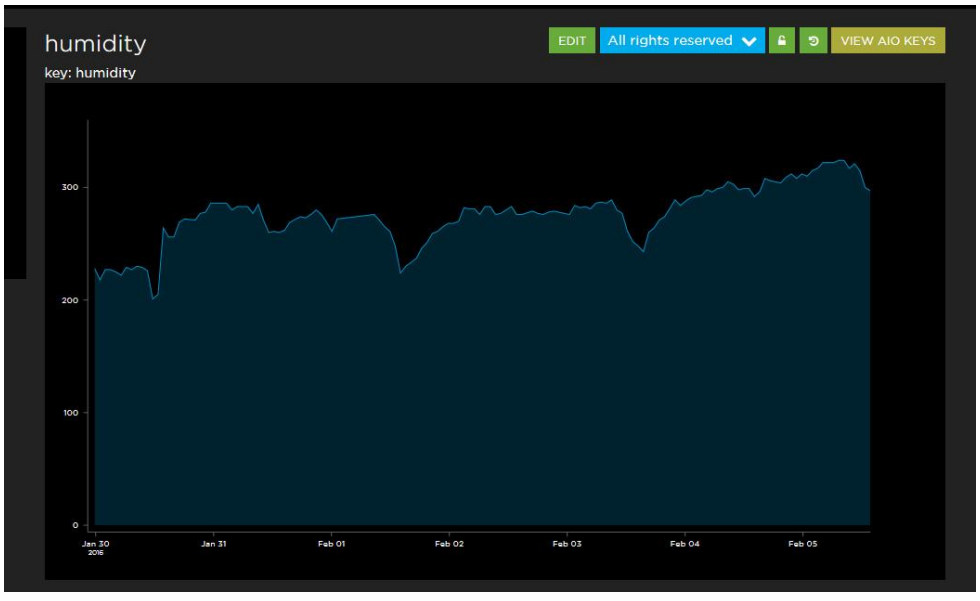


Adafruit IO

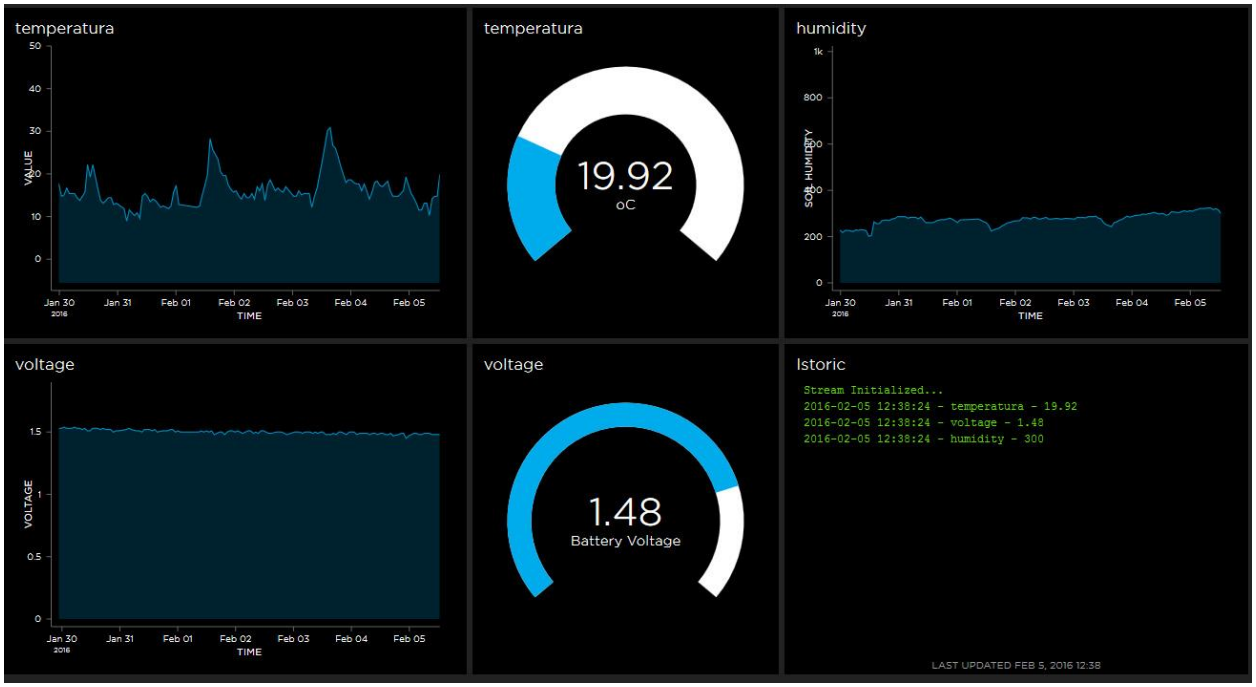
Adafruit IO [28] este un serviciu de tip IoT (Internet of Things), gratuit dar care se află în stare de dezvoltare (beta). Serviciul permite stocarea de date provenite de la dispozitivele specializate de achiziție dar și comanda acestora prin intermediul rețelei Internet (această facilitate nu este exemplificată de proiectul de față). Datele sunt organizate sub formă de canale de înregistrare (feeds).



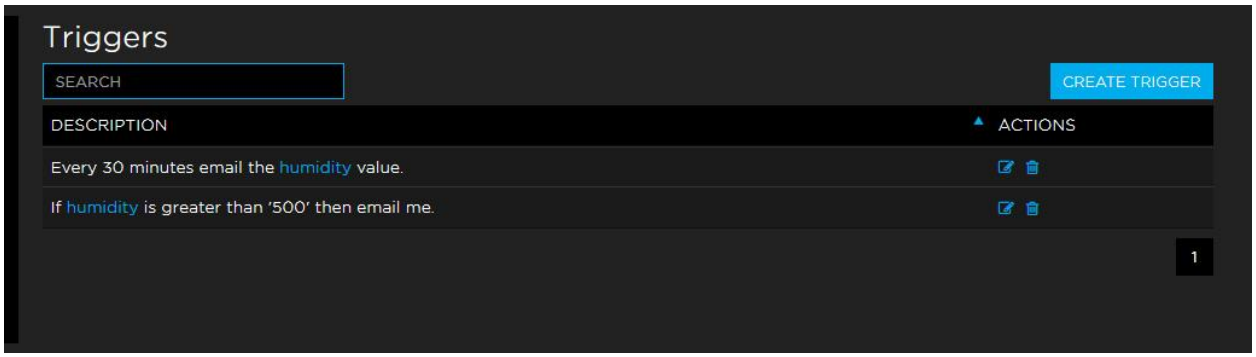
Aceste canale de înregistrare pot fi urmărite (numeric și grafic) individual:



sau pot fi grupate sub forma unui panou de urmărire (dashboards):



Pe lângă funcționalitatea de urmărire a parametrilor înregistrați, Adafruit IO mai oferă și posibilitatea de a defini alarme automate (triggers) prin email dacă parametrii înregistrați depășesc anumite valori critice (în cazul nostru umiditatea solului, temperatura ambientală sau tensiunea bateriei).



Serviciul Adafruit IO este foarte ușor de folosit și permite realizarea de aplicații IoT într-o manieră elegantă și extrem de puternică din punct de vedere tehnologic. Existența bibliotecii Arduino MQTT reprezintă o accelerare importantă în dezvoltarea de sisteme IoT bazate pe platformele de dezvoltare Arduino.

Referințe online

[1] Adafruit IO - The Internet of Things for Everyone
https://learn.adafruit.com/adafruit-io/overview

[2] ISM band
https://en.wikipedia.org/wiki/ISM_band

[3] Frequency-shift keying
https://en.wikipedia.org/wiki/Frequency-shift_keying

[4] RFM12B wireless FSK transceiver module
http://www.hoperf.com/rf_transceiver/modules/RFM12B.html

[5] MQTT https://en.wikipedia.org/wiki/MQTT

[6] Senzor de temperatura - TMP36
http://www.robofun.ro/senzor-de-temperatura-tmp36

[7] Reduino Radio
http://www.robofun.ro/wireless/wireless-433?product_id=1745

[8] Arduino Pro Mini 328 - 3.3V/8MHz
http://www.robofun.ro/platforme/arduino_dev/arduino_pro_mini_328_8mhz

[9] Conector FTDI 3.3 V
http://www.robofun.ro/platforme/arduino_dev/conector-ftdi-3V3

[10] Led Brick Verde
http://www.robofun.ro/bricks/led-brick-verde

[11] Regulator step-up U1V10F3 3.3V
http://www.robofun.ro/surse_de_alimentare/regulator_step_up/regulator_step_up_U1V10F3

[12] Sleep mode
https://en.wikipedia.org/wiki/Sleep_mode

[13] Analogie Electricitate - Curgerea Fluidelor
http://www.robofun.ro/forum/viewtopic.php?f=16&t=169

[14] JeeLib for Arduino IDE: Ports, RF12, and RF69 drivers from JeeLabs

https://github.com/jcw/jeelib

[15] Watchdog timer
https://en.wikipedia.org/wiki/Watchdog_timer

[16] Arduino Ethernet
http://www.arduino.org/products/boards/4-arduino-boards/arduino-ethernet

[17] Transmitator Radio Brick RFM12B 5V
http://www.robofun.ro/wireless/wireless-433/radio-rfm12b-5v

[18] LCD 16X2 cu conectare pe 2 fire (I2C)
http://www.robofun.ro/lcd/lcd-16x2-i2c-negru-verde

[19] Conector FTDI 5 V
http://www.robofun.ro/platforme/arduino_dev/conector-ftdi-5v

[20] Arduino Uno v3
http://www.robofun.ro/platforme/arduino_dev/arduino_uno_v3

[21] Arduino Ethernet Shield
https://www.arduino.cc/en/Main/ArduinoEthernetShield

[22] LiquidTWI: High-performance LCD library for I2C Backpack Module
http://forums.adafruit.com/viewtopic.php?f=19&t=21586&p=113177&sid=abdfecd9a4230e56b9d6fd25b6c680e7

[23] RFM12B and Arduino Ethernet with WizNet5100 chip
https://harizanov.com/2012/04/rfm12b-and-arduino-ethernet-with-wiznet5100-chip/

[24] Adafruit_MQTT_Library
https://github.com/adafruit/Adafruit_MQTT_Library

[25] Arduino Time library
http://playground.arduino.cc/code/time

[26] Network Time Protocol
https://en.wikipedia.org/wiki/Network_Time_Protocol

[27] Adafruit IO - Data Policies
https://learn.adafruit.com/adafruit-io/data-policies

[28] Adafruit IO The Internet Of Things For Everyone
https://io.adafruit.com/