Aplicaţie experimentală de tip platformă *IoT* software

Ştefan Alexandru PREDA, Mihnea Horia VREJOIU
Institutul Naţional de Cercetare-Dezvoltare în Informatică - ICI Bucureşti,
B-dul Mareşal Averescu Nr. 8-10, Bucureşti, 011455, România
stefan.preda5@gmail.com
mihnea@ici.ro

Rezumat: Sintagma „Internetul obiectelor/lucrurilor” (*Internet of Things* – IoT) a devenit (aproape) una comună astăzi. Numărul şi diversitatea obiectelor fizice (dispozitive) conectate la Internet şi având încorporată tehnologia necesară pentru a sesiza şi comunica starea lor internă şi pentru a interacţiona cu aceasta şi cu mediul extern au crescut vertiginos. În ultimii ani, necesităţile de colectare şi stocare a volumelor uriaşe de date furnizate de acestea în infrastructuri dedicate, centralizate/localizate sau distribuite (de tip cloud), precum şi de valorificare a datelor respective prin tehnici de analiză avansată (*analytics*) utilizând servicii specializate, au condus la dezvoltarea a numeroase soluţii de interconectare a acestor obiecte inteligente în sisteme cu scale, obiective şi caracteristici diverse. Lucrarea de faţă prezintă o încercare de dezvoltare şi implementare practică experimentală a unei astfel de platforme IoT *software* proprii, iniţial cu un set relativ redus de funcţionalităţi specifice, ca bază pentru potenţiale dezvoltări, implementări şi extinderi viitoare. Sunt descrise arhitectura, componentele, funcţionalităţile şi legăturile funcţionale implicate, precum şi câteva specificaţii de implementare şi utilizare pentru cele mai importante dintre acestea.

Cuvinte cheie: Platformă IoT, Cloud, Obiecte/dispozitive inteligente, Senzori, Microcontroler, Protocol de comunicaţie, API de tip REST.

An Experimental Application of *IoT* Software Platform Type

Abstract: The term *Internet of Things* (IoT) has become a(n almost) common one today. The number and diversity of physical objects (devices) connected to the Internet and incorporating the technology for sensing and communicating their internal state and to interact with it and with the external environment, have increased steeply. In recent years, the need to collect and store huge amounts of data they provide in dedicated, centralized/localized or distributed (cloud) infrastructures, as well as to get value from these data through analytics techniques by using specialized services, have led to the development of many solutions for interconnecting these smart objects in systems with different scales, objectives and features. This paper presents an own experimental development and implementation of such an IoT software platform, initially with a relatively small set of specific functionalities, as basis for potential future developments, deployments, and expansions. The architecture, components, functionalities and functional links involved are described, as well as some implementation and usage specifications for the most important of them.

Keywords: IoT platform, Cloud, Smart objects/devices, Sensors, Microcontroller, Communication protocol, REST API.

1. Introducere

Sintagma „Internetul obiectelor/lucrurilor” (*Internet of Things – IoT*) denumeşte reţeaua de obiecte fizice (dispozitive) „inteligente” conectate la Internet şi capabile, prin tehnologia încorporată, de a sesiza evoluţia stării lor interne în interacţiunea

cu mediul extern şi de a comunica date despre aceasta. *IoT* tinde să devină una din paradigmele definitorii ale lumii în care trăim şi lucrăm azi, determinând evoluţii atât de ordin cantitativ, cât şi calitativ, în cunoaşterea umană şi interacţiunea noastră cu mediul [4].

Numărul diverselor obiecte inteligente interconectate este într-o continuă creştere exponenţială; la fel şi varietatea acestora: de la telefoane, tablete, televizoare şi alte obiecte electrodomestice inteligente, la dispozitive de monitorizare şi transmitere a parametrilor de sănătate şi mobilitate a oamenilor sau animalelor, de calitate a apei sau aerului, a parametrilor unor echipamente industriale complexe (şi de control al acestora), a produselor transportate în containere pentru livrare ş.a.m.d. [10].

Sistemele *IoT* dezvoltate în ultimii ani au ţinut cont de această expansiune a numărului şi diversităţii acestor obiecte, urmărind, pe de o parte, standardizarea soluţiilor de comunicaţie şi interacţiune cu ele şi, pe de altă parte, valorificarea datelor furnizate de acestea, prin achiziţia, transmiterea şi stocarea acestui volum uriaş de date în flux continuu (*Big Data*) în infrastructuri dedicate, centralizate/localizate sau distribuite (de tip *cloud*) şi aplicarea unor metode de analiză avansată (*analytics*) a acestora, utilizând servicii specializate, pentru a identifica, extrage, sintetiza şi utiliza informaţia relevantă conţinută în ele.

Principalele funcţionalităţi oferite de sistemele *IoT* pot fi grupate în 5 categorii [1]: culegerea şi pregătirea datelor; conectivitate, protocoale de comunicaţii; servicii de monitorizare, control şi descoperire dispozitive; autentificare, autorizare, controlul integrităţii şi securitatea datelor; analiza şi procesarea datelor, asigurarea interfeţei utilizator pentru acces la funcţiunile sistemului.

Din punct de vedere al tehnologiilor implicate în dezvoltarea soluţiilor *IoT*, pot fi enumerate: reţele de senzori *wireless*, *cloud computing*, *Big Data analytics*, protocoale de comunicaţie, dispozitive cu microprocesor încorporat.

Platformele *IoT* reprezintă soluţii de implementare şi dezvoltare a funcţionalităţilor şi tehnologiilor menţionate mai sus, unitar şi standardizat la nivel de platformă, ca suport pentru valorificarea performantă a acestora, permiţând utilizatorilor să se poată concentra pe conectarea echipamentelor, pe selectarea resurselor şi opţiunilor de care au nevoie (accesul la infrastructura de stocare şi regăsire date, selectarea serviciilor de procesare, vizualizare şi analiză a datelor) şi pe utilizarea efectivă a rezultatelor furnizate de platformă.

Lucrarea de faţă prezintă o încercare de dezvoltare şi implementare experimentală a unei astfel de platforme *IoT software* proprii, iniţial cu un set relativ redus de funcţionalităţi de bază specifice, ca bază pentru potenţiale dezvoltări, implementări şi extinderi ulterioare. Sunt prezentate arhitectura, componentele, funcţionalităţile

şi legăturile funcţionale implicate, precum şi câteva specificaţii de implementare şi modul de utilizare pentru cele mai importante dintre acestea.

În continuare, lucrarea este structurată după cum urmează: în Secţiunea 2 este prezentată sintetic situaţia actuală, cu câteva dintre cele mai reprezentative platforme *IoT* şi principalele caracteristici specifice acestora; Secţiunea 3 este dedicată descrierii arhitecturii, componentelor, funcţionalităţilor şi legăturilor funcţionale implicate în platforma *software IoT* experimentală propusă; Secţiunea 4 prezintă câteva specificaţii de implementare pentru cele mai importante dintre acestea, precum şi câteva instrucţiuni de utilizare; în Secţiunea 5, finală a lucrării, sunt formulate câteva concluzii.

2. Situaţie actuală platforme *IoT*

Există astăzi o abundenţă de soluţii de platforme *IoT* (compuse din infrastructură şi *middleware*), care oferă conectivitate la Internet pentru senzori şi elemente de acţionare (*actuators*), permiţând interacţiunea cu obiectele inteligente dotate cu astfel de senzori şi/sau elemente de acţionare, colectarea, stocarea şi analiza avansată a datelor de la acestea.

În lucrarea [9] am prezentat o trecere în revistă a ofertei de platforme *IoT* existente, cu analiza exhaustivă a principalelor caracteristici, funcţionalităţi şi criterii de evaluare, precum şi a perspectivelor şi tendinţelor de evoluţie a acestora în perioada următoare. Reluăm în cele de mai jos, sumar, câteva idei.

Dintre cele mai reprezentative platforme *IoT* [5][12], care se disting prin facilităţile oferite pentru dezvoltarea de soluţii orientate *IoT*, au fost amintite următoarele: Amazon Web Services (AWS) *IoT*, Microsoft Azure *IoT* suite, ThingWorx, IBM Watson *IoT*, Cisco *IoT* Cloud Connect, Salesforce *IoT* Cloud, Carr*IoT*s, Oracle Integrated Cloud *IoT*, General Electric's Predix şi Kaa, ultima fiind *open source*.

Totodată, în contextul complementarităţii şi integrării tehnologiilor cloud computing şi *IoT*, a fost propusă următoarea selecţie de platforme *IoT* [2]: *IoT*Cloud, Open*IoT*, *IoT* Toolkit, NimBits, openPicus, Xively, Open.Sen.se, ThingSpeak, CloudPlugs, Carr*IoT*s, NetLab, Intel *IoT* Analytics, Synapse *IoT* Cloud, ClouT. Cu două excepţii (openPicus şi Synapse *IoT* Cloud) toate aceste platforme suportă dispozitive de tip „open" pe lângă cele proprietare. Platformele *IoT*Cloud, Open*IoT*, *IoT* Toolkit, NimBits, NetLab, Intel *IoT* Analytics sunt *open source*, iar marea majoritate sunt distribuite gratuit (*free license*). Toate aceste platforme suportă cloud privat, iar Open*IoT*, Synapse *IoT* Cloud şi ClouT suportă şi cloud public.

Principalele caracteristici definitorii ale platformelor *IoT* actuale sunt [6]:

- ***Tipurile de dispozitive suportate***: există platforme care necesită o poartă de acces (*gateway*) proprietară pentru conectarea dispozitivelor *IoT*.

- ***Tipul platformei IoT***: în cele mai multe cazuri platformele sunt furnizate în cloud, ca *Platform as a Service* (PaaS), sau ca *Software as a Service* (SaaS).

- ***Tipul arhitecturii***: arhitecturile de tip centralizat/localizat sunt specifice soluţiilor independente, în timp ce acelea de tip descentralizat/distribuit (cloud) includ mai multe subreţele de senzori şi dispozitive de acţionare, fiecare controlată independent.

- ***Gradul de deschidere***: platformele open source au, în general, potenţial de perspectivă mai mare decât alternativele proprietare.

- ***Disponibilitatea unui API de tip REST***: majoritatea platformelor oferă interfeţe de programare aplicaţii – API (*Application Programming Interface*) – de tip REST (*REpresentational State Transfer*).

- ***Controlul accesului la date***: este relevant pentru platformele care nu arhivează datele local şi implementează diverse niveluri de control al accesului la distanţă, de la accesul de tip privat/public, până la un control mai nuanţat al accesului, atunci când datele pot fi private, protejate, publice sau anonime.

- ***Mecanismele de descoperire a serviciilor***: sunt încă relativ puţin răspândite la nivelul platformelor *IoT* actuale şi se referă în principal la protocoale de descoperire pentru comunicaţii M2M cu restricţii.

- ***Securitatea şi confidenţialitatea***: platformele *IoT* bazate pe cloud sunt predispuse la atacuri de securitate *Web* şi de reţea tradiţionale. Pentru asigurarea securităţii şi confidenţialităţii, atât în scenarii centralizate, cât şi distribuite, sunt necesare protocoale de nivel scăzut (*low level*).

Platformele *IoT software* pun accentul mai cu seamă pe componentele de administrare, analiză avansată şi vizualizare a datelor. Astfel, din cauza complexităţii acestor componente, oferta de platforme *IoT software* este mult mai restrânsă decât cea de platforme *IoT* hardware (care pun accentul mai ales pe infrastructura tehnică de colectare şi transmisie a datelor). Principalele caracteristici ale acestei clase de platforme *IoT* sunt [3]:

- ***Administrarea dispozitivelor***: platforma *IoT* trebuie să administreze lista dispozitivelor conectate la aceasta şi să urmărească starea lor operaţională, să asigure configurarea şi actualizarea *software*-ului pe dispozitive şi să ofere acestora modalităţi de raportare şi rezolvare a erorilor. Astfel, utilizatorii pot obţine, prin intermediul platformei, statistici individuale pentru fiecare dispozitiv.

- ***Suportul pentru integrare***: interfaţa de programare (API) trebuie să furnizeze acces la operaţiile şi datele importante ce necesită să fie descărcate din platforma *IoT*. De obicei se utilizează API-uri de tip REST.

- ***Securitatea informaţiei***: din cauza numărului mare de dispozitive conectate la o platformă *IoT*, numărul de vulnerabilităţi creşte proporţional cu numărul acestora. Conexiunile ar trebui să fie criptate prin mecanisme puternice, pentru a se evita potenţialele furturi de date. Totuşi, majoritatea dispozitivelor implicate în platformele moderne *IoT* nu pot susţine astfel de cerinţe avansate de control pentru accesul la date. Astfel, platforma *IoT* trebuie să implementeze soluţii alternative pentru îmbunătăţirea nivelul de securitate.

- ***Protocoalele de colectare de date***: datorită numărului mare de dispozitive *IoT* şi, implicit, volumului mare al datelor transmise către platformă, trebuie folosite protocoale simple (*light*) de comunicaţie de date, pentru economisirea de energie şi utilizarea mai eficientă a lărgimii de bandă a reţelei.

- ***Analiza avansată a datelor***: datele colectate de la senzorii conectaţi la platforma *IoT* trebuie să fie analizate într-o manieră inteligentă pentru a obţine rezultate semnificative. Există patru tipuri de analiză ce pot fi aplicate datelor de tip *IoT*: în timp real, pe loturi, predictivă şi interactivă. Analiza în timp real se realizează *online*, pe fluxul de date transmis de către senzori. Analiza pe loturi se face pe totalul de date acumulate, la anumite intervale de timp, şi poate dura câteva ore sau zile. Analiza predictivă se concentrează pe generarea de predicţii bazându-se pe tehnici statistice şi de învăţare automată. Analiza interactivă se aplică atât pe fluxul de date transmis, cât şi pe loturi.

3. Platforma *IoT* propusă

Platforma experimentală *Internet of Things* proprie propusă, dezvoltată şi implementată, a fost proiectată iniţial doar cu scopul de a colecta datele furnizate de diferite dispozitive inteligente într-o bază de date comună, pentru ca acestea să poată fi accesate şi prelucrate de la distanţă, ca prim pas pentru potenţiale dezvoltări, implementări şi extinderi ulterioare, în diferite aplicaţii [7][8].

Platforma se adresează oricărui utilizator ce deţine dispozitive inteligente şi doreşte să stocheze datele de la acestea în mod automat. De asemenea, se adresează utilizatorilor care doresc să monitorizeze, consulte şi/sau să analizeze respectivele date într-un mod simplu. Un utilizator poate fi o persoană sau un grup de utilizatori ce partajează datele în cadrul unui proiect de grup. Totodată, platforma poate permite utilizatorilor şi controlul de la distanţă al dispozitivelor înregistrate şi conectate la platforma *Internet of Things*.

Pentru a folosi platforma, un utilizator trebuie să îşi creeze un cont în cadrul interfeţei *Web* oferite de aceasta. Ulterior, accesând acest cont, va avea posibilitatea să adauge şi să gestioneze un număr nelimitat de dispozitive inteligente proprii de la care să fie colectate şi stocate date, să monitorizeze, consulte sau analizeze datele provenite de

la acestea. Prin dispozitive inteligente înţelegem orice dispozitiv ce se poate conecta la Internet şi trimite sau primi date prin intermediul conexiunii Internet.

Conexiunea între platforma *Internet of Things* şi dispozitivele *IoT* se realizează pe bază de IP sau URL. Astfel, în programul script de la nivelul obiectelor inteligente va fi setată adresa IP a serverului pe care este instalată platforma, pentru ca datele culese de acestea să poată fi transmise către baza de date a platformei. În baza de date a platformei, datele se stochează automat în mod privat pentru fiecare utilizator şi diferenţiat, în funcţie de dispozitivul de la care au fost primite datele respective. Odată ce sunt primite şi stocate date de la dispozitivele unui utilizator, acesta va avea posibilitatea de a le monitoriza, consulta şi/sau de a le analiza, utilizând scripturile incluse în platforma *IoT*. Scopul esenţial al acestor scripturi este acela de a permite utilizatorilor monitorizarea şi descoperirea apariţiei unor valori neobişnuite sau a unor valori similare în datele primite şi stocate în cursul timpului.

Un exemplu practic pentru utilitatea descoperirii de valori neobişnuite ar fi măsurarea temperaturii şi controlul unui dispozitiv inteligent de răcire/încălzire într-o cameră. Astfel, pentru a fi informat în mod automat despre creşterea sau scăderea temperaturii în mod neaşteptat, sau în afara unui interval prestabilit, utilizatorul va putea comanda din pagina sa de comandă a platformei *IoT* implementarea unei alarme care să-l anunţe, prin intermediul unui email sau a unui mesaj Tweeter sau Facebook, asupra valorii respective citite. După primirea alarmei, utilizatorul va putea alege să controleze dispozitivul inteligent de răcire/încălzire pentru a readuce temperatura la valorile dorite. De asemenea aceste acţiuni se vor putea realiza şi automat, indicând platformei valorile la care aceasta va trebui să pornească sau să oprească dispozitivul inteligent de răcire/încălzire cu scopul de a menţine o temperatură constantă.

Arhitectura platformei *IoT* experimentale propuse este redată în Figura 1.

Platforma a fost dezvoltată utilizând scripturi PhP, HTML şi Javascript şi foloseşte o bază de date MySQL ce conţine atât datele de autentificare a utilizatorilor, cât şi dispozitivele inteligente pe care aceştia le-au înscris în platformă pentru a stoca datele primite de la ele.

Legătura între platforma *IoT* şi dispozitivele inteligente se face printr-un script PhP ce este apelat de către programele locale, rulând pe dispozitivele inteligente, atunci când acestea transmit date către platformă.

Baza de date MySQL

Pentru această platformă am folosit o bază de date relaţională MySQL, a cărei arhitectură este schiţată în Figura 2.

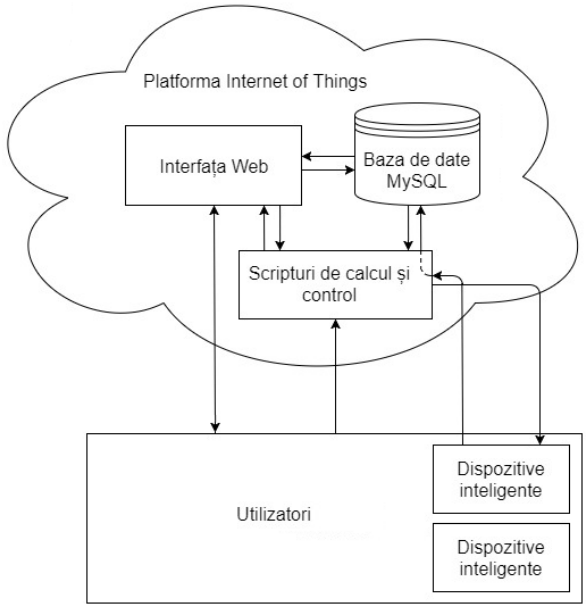

Figura 1. Arhitectura platformei IoT experimentale dezvoltate

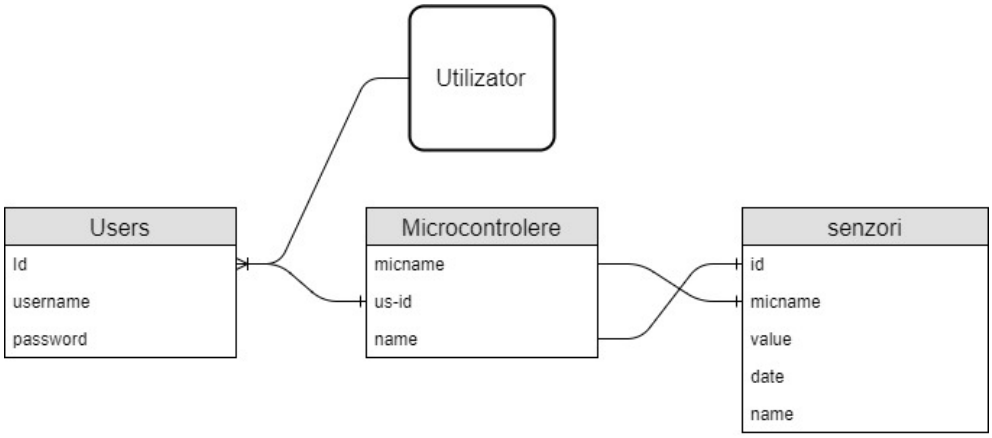

Figura 2. Arhitectura bazei de date a platformei experimentale IoT dezvoltate şi relaţiile între tabele

Această bază de date conţine trei tabele:

- *Users*,
- *Microcontrolere* şi
- *senzori*.

Tabelul *Users* conţine:

- un câmp de tip cheie unică, *Id*, folosit pentru identificarea utilizatorului logat pe site şi câmpurile:

- *username* şi

- *password*,

reprezentând numele şi parola utilizatorului, necesare pentru autentificarea şi identificarea acestuia pe platformă.

Parola utilizatorului se criptează automat în mod MD5 la crearea unui cont nou, pentru a nu apărea în clar în baza de date.

Tabelul *Microcontrolere* conţine câmpurile *micname*, *us-id* şi *name*:

- Câmpul *micname* conţine numele fiecărui microcontroler sau dispozitiv inteligent adăugat/înregistrat de către utilizator.

- Câmpul *us-id* va fi completat automat de sistem cu *Id*-ul utilizatorului care a adăugat dispozitivul inteligent la platformă.

- Câmpul *name* va fi completat de către utilizator cu numele senzorului conectat la microcontroler şi de la care platforma va prelua date.

Tabelul *senzori* va fi completat în mod automat de către dispozitivele inteligente conectate la platforma *IoT*. Acesta conţine următoarele câmpuri:

- *id*, ce va conţine Id-ul utilizatorului căruia îi aparţine dispozitivul inteligent respectiv,

- *micname*, ce va conţine numele dispozitivului inteligent,

- *name*, care va conţine numele senzorului conectat la microcontrolerul de la care se primesc date,

- *value*, care va conţine valorile transmise de către microcontroler şi

- *date*, ce va conţine data şi ora asociate fiecarei valori transmise de către microcontroler.

Conectorul dispozitivelor inteligente cu platforma *IoT*

Acest conector constă într-un script PhP conţinând datele specifice de autentificare pentru conectarea la baza de date MySQL şi care va fi apelat de către fiecare dispozitiv inteligent atunci când acesta doreşte să transmită date către platformă.

Datele pe care va trebui să le transmită dispozitivul inteligent sunt:

- *id* = *Id*-ul utilizatorului, asociat la autentificarea acestuia pe platformă;
- *micname* = numele microcontrolerului aşa cum a fost înregistrat de către utilizator în baza de date a platformei;
- *name* = numele sau tipul de senzor de la care microcontrolerul preia date, aşa cum a fost denumit de către utilizator în platformă;
- *value* = valori de date provenite de la microcontroler.

Toate aceste date se transmit prin codul implementat pe microcontroler.

Acest conector este amplasat în directorul *htdocs* al serverului.

Interfaţa *Web* a platformei *IoT*

Interfaţa *Web* a acestei platforme este compusă din fişiere de tip .php ce sunt localizate în interiorul *folder*-ului *htdocs* al serverului.

Pagina de start se numeşte *Test.php*, şi conţine *link*-uri către pagina de autentificare, pagina de înregistrare şi către pagina de descriere a proiectului.

Pagina de înregistrare (Figura 3) permite unui utilizator nou crearea/înregistrarea unui cont în cadrul platformei. Pentru aceasta, el va trebui mai întâi să introducă un nume şi o parolă în câmpurile *username* şi *password*.

Pagina de autentificare conţine două câmpuri care trebuie completate de către utilizator cu numele şi parola cu care s-a înregistrat anterior pe platformă, pentru a intra în contul său. După ce utilizatorul s-a autentificat astfel cu succes la platforma *IoT*, va

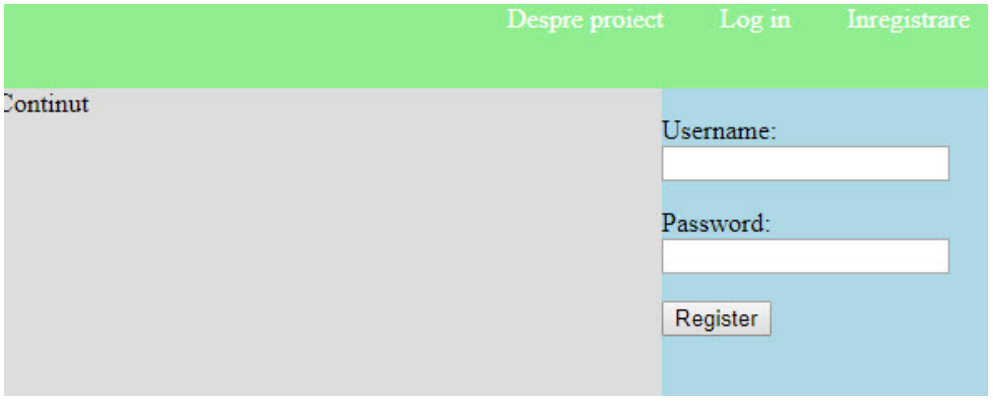

Figura 3. Pagina de înregistrare a utilizatorilor platformei IoT

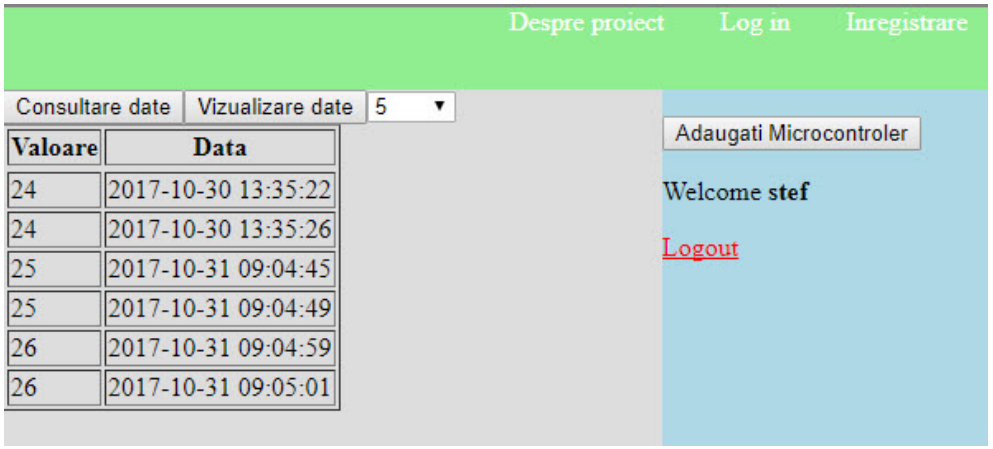

Figura 4. Pagina de vizualizare date a unui utilizator al platformei IoT

fi redirecţionat la pagina sa personală, de unde va putea adăuga un microcontroler şi/sau un senzor, sau va putea consulta datele stocate pe platformă, provenind de la dispozitivele inteligente deja înregistrate de el anterior şi care transmit şi/sau au transmis date către aceasta (Figura 4).

4. Specificaţii de dezvoltare

Dispozitive inteligente. Microcontrolere

Microcontrolerele sunt dispozitive inteligente ce conţin unul sau mai multe procesoare, memorie proprie, tehnologie de conectare la Internet (Ethernet sau WiFi) şi diferiţi pini sau port-uri pentru conectarea senzorilor.

Au fost utilizate/testate microcontrolerele Arduino Yún [13], Raspberry Pi şi Raspberry Pi v2 [14].

Microcontrolerul Arduino Yún

Arduino Yún este o placă microcontroler cu un *chip* ATmega32u4 (microcontrolerul propriu-zis) şi un microprocesor Atheros AR9331 pe care rulează o versiune mini integrată de Linux bazată pe OpenWRT, denumită Linino OS, care asigură comunicaţia prin interfeţele de reţea WiFi şi Ethernet integrate pe placă. Placa mai oferă un port USB-A (pentru conectare cu un PC gazdă), un slot pentru card micro-SD, 20 de intrări/ieşiri digitale (7 dintre acestea pot fi utilizate ca ieşiri PWM şi 12 ca intrări analogice), oscilator cu cuarţ de 16 MHz, o conexiune micro USB (utilizată mai ales pentru alimentare), un conector pentru programare *onboard* ICSP (*In-Circuit Serial Programming*) şi 3 butoane de resetare (pentru 32u4, WiFi şi Linino).

În plus faţă de comenzile Linux cum ar fi cURL, pot fi scrise scripturi Shell şi Python proprii, existând Python preinstalat în Linino.

Programare Arduino

Yún poate fi programat cu ajutorul *software*-ului Arduino, fie prin editor *Web*, fie prin intermediul IDE (*Integrated Development Environment*) dedicat ce poate fi descărcat de la adresa: *https://arduino.cc/software* selectând „Arduino Yún" din meniul Tools > Board (în funcţie de microcontrolerul de pe placa respectivă).

Chipul ATmega32U4 de pe Arduino Yún vine cu un *bootloader* preinstalat care permite încărcarea unui nou cod în el fără a se utiliza un programator hardware extern. Acesta comunică utilizând protocolul AVR109. De asemenea, se poate evita *bootloader*-ul şi programa microcontrolerul prin intermediul conectorului ICSP, folosind un dispozitiv de programare Arduino ISP (*In-System Programming*) sau altceva similar.

Pentru folosirea unui microcontroler Arduino de sub sistemul de operare Windows este necesară instalarea în prealabil a driverelor corespunzătoare. Pentru sistemele de operare MAC OS sau Linux, nu este nevoie de instalarea unor drivere.

Pentru dezvoltarea programelor pentru microcontrolerul Arduino au fost testate mediile de programare Arduino IDE şi, respectiv, Visual Studio împreună cu pachetul *IoT* pentru acesta. Limbajul de programare utilizat este similar limbajului C. Indiferent de mediul folosit, programarea pentru microcontrolerul Arduino se face la fel.

Astfel, orice program Arduino (numit *sketch*) are două secţiuni: secţiunea *setup*, care va fi rulată doar o singură dată atunci când microcontrolerul se conectează la alimentare, având rolul de a realiza configurarea acestuia şi secţiunea *loop*, care va fi rulată continuu în buclă atâta timp cât este alimentat microcontrolerul. În secţiunea *setup* sunt de obicei declaraţi pinii de intrare/ieşire şi sunt declarate de asemenea alte variabile folosite apoi în secţiunea *loop*. Secţiunea *loop* este folosită pentru citirea la anumite intervale de timp a datelor transmise de către senzorii conectaţi la pinii declaraţi, pentru transformarea acestor date primite în valori ale unor mărimi specifice utile şi, apoi, pentru transmiterea acestora din urmă către platforma *Internet of Things*.

Majoritatea senzorilor sau actuatorilor (dispozitivelor de acţionare) se conectează fizic direct la pinii/porturile digitale sau analogice ale microcontrolerului, putând astfel primi comenzi sau transmite date către acestea. Un senzor se poate conecta la un pin analogic folosind comanda "`analogRead(nr. pin)`". Pentru conectarea la un pin digital se foloseşte comanda "`digitalRead(nr. pin)`". Anumite componente, cum ar fi un LCD, se pot conecta simultan la mai mulţi pini de acelaşi fel.

Prezentăm în continuare, ca exemplu, un program Arduino (*sketch*), comentat, pentru citirea continuă, la intervale constante de timp, a datelor de la un senzor de temperatură

şi transmiterea şi stocarea acestora într-o bază de date MySQL. Pentru această ultimă parte, este apelat un script *Write_data.php* bazat pe cel furnizat în pachetul XAMPP (distribuţie Apache conţinând MySQL, PhP şi Pearl) şi localizat pe serverul cu baza de date (şi platforma *IoT*), al cărui conţinut de principiu este redat, de asemenea, imediat mai jos.

Final_client.ino

```
#include <Bridge.h>

#include <YunClient.h>


// IP-ul la care se conecteaza

// (unde e si BD MySQL in cazul acesta)

char server[] = „192.168.1.19”;

int server_port = 8080;

char data_name[] = „temp”;

int delta_t = 100;

BridgeClient client;


void setup() {

  pinMode(13, OUTPUT);

  digitalWrite(13, LOW);

  Bridge.begin();

  digitalWrite(13, HIGH);

  Serial.begin(9600);

}


void loop() {

  // citire voltaj pin + transformare in grade C

  // Valoarea 3.3 pt. pinul de 3.3 V (exista si de 5V)

  int reading = analogRead(0);

  float voltage = (reading * 3.3) / 1024.0;

  float temperatureC = (voltage - 0.5) * 100;
```

```
  // Conectare la serverul si portul specificate
  if(client.connect(server, server_port)) {
    Serial.println(„--> connection succ\n”);
    // foloseste scriptul Write_data.php din pachetul XAMPP
    // (da userul, parola si BD in care se scrie)
    client.print(„GET /Write_data.php?”);
    client.print(„name=”);
    client.print(data_name);
    // scrie in BD MySQL &value=<valoarea temperaturii citite>
    client.print(„&value=”);
    client.print(temperatureC);
    client.println(„ HTTP/1.1”);  // protocol utilizat – parte din GET
    client.print(„Host: „);
    client.println(server);
    client.println(„Connection: close”);  // parte din GET
    client.println();
    client.println();
    client.stop();  // terminare conectare
  }
  else Serial.println(„--> connection failed\n”);
  // interval de timp intre citiri (ms)
  delay(delta_t);
}
```

Write_data.php

```
// Pregatire variabile pentru conectare la BD
$db_name = „Test”;  // nume BD
$db_username = <username>;  // nume utilizator BD
$db_password = <password>;  // parola utilizator BD
$server = „localhost”;  // URL/IP server BD
// Conectare la BD
```

```
$con = mysqli_connect(„p:” . $server, $db_username, $db_password, $db_name);
if($con) {
  // Compunere instructiune SQL de scriere in BD
  $sql = „INSERT INTO senzori (value) VALUES (,”.$_GET[„value”].”’)”;
  // Executie instructiune SQL de scriere in BD
  mysqli_query($con, $sql);
  // Inchidere conexiune
  mysqli_close($con); }
```

Specificaţii software platformă *IoT*

În continuare vom descrie succint elementele principale care apar în codurile sursă ale componentelor platformei *IoT* dezvoltate şi implementate.

Styles1.css

Acest script este folosit pentru a defini structura, modelul şi cosmetica tuturor elementelor paginilor din interfaţa *Web* a platformei *Internet of Things* dezvoltate.

Server.php

Acesta este un script ce conţine funcţii pentru funcţionalitatea paginilor interfeţei *Web*.

În primul rând, acest script se conectează la baza de date folosită de către platformă folosind comanda:

```
$db = mysqli_connect(,localhost’, ,root’, ,password’, ,login’);
```

unde `localhost` reprezintă serverul pe care este instalată baza de date, `root` este numele administratorului bazei de date, `password` este parola administratorului bazei de date, iar `login` este numele bazei de date folosite de către platforma *IoT*.

Restul funcţiilor declarate în acest script vor fi descrise la secţiunea paginilor .php care utilizează acele funcţii.

Test.php

Aceasta este pagina de start a platformei. Este un document HTML ce foloseşte scriptul *Server.php* şi *Styles1.css*.

În bara de navigaţie sunt definite *link*-uri către pagina de conţinut, de înregistrare şi de autentificare. Cosmetica acestora este definită în scriptul *Styles1.css*.

Login.php

Aceasta este pagina unde utilizatorii ce au deja un cont la platforma *IoT* se pot conecta. În cadrul din dreapta paginii se pot observa două câmpuri denumite *username* şi *password* unde utilizatorul va putea introduce datele sale de identificare şi un buton *Login* prin care se va apela funcţia de autentificare din scriptul *Server.php*.

Funcţia de autentificare va prelua datele completate de către utilizator, apoi aceste date vor fi salvate în două variabile ce vor fi transformate cu ajutorul comenzii:

```
mysqli_real_escape_string();
```

pentru a nu se putea face injectarea bazei de date MySQL.

Funcţia verifică apoi dacă nu cumva unul, sau ambele câmpuri a/au fost cumva necompletate. Într-o astfel de situaţie, funcţia va introduce într-o matrice numită *errors* o expresie text prin care se va cere introducerea datelor în câmpul care nu a fost completat.

Odată validate condiţiile de mai sus, funcţia va căuta în baza de date a platformei, în tabelul *users*, o pereche de câmpuri *username* şi *password* cu valori identice cu cele introduse de către utilizator. Dacă nu se va găsi o astfel de pereche, utilizatorul va primi un mesaj de eroare care îi va transmite că parola sau numele de utilizator nu sunt corecte:

```
$query = „SELECT * FROM users WHERE username = ,$username’ AND password =
,$password’”;

$result = mysqli_query($db, $query);

$record = mysqli_fetch_assoc($result);

$id = $record[,id’];

$count = mysqli_num_rows($result);

if($count != 1) {

  array_push($errors, „Parola/user incorecte”);

  header(,location: login.php’); }
```

Dacă funcţia găseşte o pereche *username* şi *password* identice cu datele completate de către utilizator, atunci acesta va fi redirecţionat către pagina personală a platformei *IoT*.

Register.php

Aceasta este pagina de înregistrare a utilizatorilor noi pe platforma experimentală.

La fel ca şi pagina *Login.php*, aceasta are două câmpuri ce pot fi completate, denumite *username* şi *password*, şi un buton *Register*, prin care se apelează funcţia de înregistrare din fişierul *Server.php*.

Funcţia este asemănătoare cu cea de autentificare, diferenţa fiind descrisă mai jos.

Dacă cele două câmpuri au fost completate corespunzător, funcţia va verifica în baza de date dacă mai există utilizatori cu acelaşi nume. Dacă există, utilizatorul va primi un mesaj prin care i se comunică că este nevoie să aleagă un alt nume, deoarece numele este folosit de către un alt utilizator.

În fine, dacă cele două câmpuri au fost completate şi validate, atunci funcţia va crea o nouă intrare în tabelul *Users* din baza de date, în care se va atribui automat un *id* nou, pentru care vor fi asociate datele preluate din câmpurile *username* şi *password* completate de către utilizator. Înainte de preluarea parolei, aceasta va fi codată MD5, pentru a nu apărea în clar în baza de date, prin comanda:

```
$password = md5($password);.
```

Persoana.php

Aceasta este pagina utilizatorului platformei *IoT*. Utilizatorul este direcţionat automat către această pagină după ce a trecut cu succes de pagina de autentificare a platformei.

În cadrul de conţinut al paginii sunt prezente două butoane: *Consultare date* şi *Vizualizare date*, împreună cu o listă a microcontrolerelor înregistrate de către utilizatorul respectiv.

În cadrul din dreapta al paginii este afişat un buton *Adauga Microcontroler* prin care utilizatorul va putea adăuga în baza de date, asociat lui, un nou dispozitiv inteligent. De asemenea, tot în acest cadru este afişat *numele utilizatorului* şi *id*-ul acestuia, care va fi folosit la transmiterea datelor de la microcontroler către platformă. Este important de ştiut pentru utilizatori că *id*-ul, *numele microcontrolerelor* adăugate şi *numele senzorilor* adăugaţi trebuie precizate (şi) în codul sursă al fiecărui microcontroler, pentru ca datele să fie stocate corespunzător utilizatorului deţinător respectiv. Odată apăsat butonul *Adauga Microcontroler*, pagina va afişa câmpuri de editare în care utilizatorul va putea introduce *numele microcontrolerului* şi *numele senzorului* de la care microcontrolerul va citi datele. După introducerea acestor nume, toate aceste noi date vor fi adăugate în baza de date, iar microcontrolerul respectiv va putea fi pornit pentru a transmite date către platforma *IoT*.

```
if(isset($_POST[,inserarem’])) {

  $numem = mysqli_real_escape_string($db, $_POST[,col1’]);

  $numes = mysqli_real_escape_string($db, $_POST[,col2’]);

  $id = $_SESSION[,id’];
```

```
  $sql = „INSERT INTO microcontrolere (micname, ,us-id', name) VALUES
(,$numem', ,$id', ,$numes')";

  $try = mysqli_query($db, $sql);

  if($try == false){

    echo ,error - ,;

    echo mysqli_error($db);

    } }
```

Tot în cadrul din stânga, utilizatorii vor putea să îşi consulte datele microcontrolerelor personale deţinute, prin selectarea acestora dintr-o listă *drop-down* şi apăsând pe butonul *Consultare date*. Această acţiune va afişa în pagina personală a utilizatorului tabelele cu valorile datelor transmise de către dispozitivele inteligente înregistrate de către utilizator şi stocate în baza de date. În codul sursă, această acţiune se face prin memorarea poziţiei pe care utilizatorul o selectează din meniul *drop-down*, această poziţie reprezentând numele câmpurilor ce urmează a fi interogate din baza de date.

```
while($record = mysqli_fetch_assoc($data)) {

  echo „<tr>";

  echo „<td>" . $record[,value'] . „</td>";

  echo „<td>" . $record[,date'] . „</td>";

  echo „</tr>"; }
```

Butonul *Vizualizare date* oferă posibilitatea utilizatorului de a comanda afişarea grafică a valorilor transmise de către microcontroler în funcţie de dată, prin redirecţionare către pagina *Chart.php*.

Chart.php

Aceasta este pagina ce oferă utilizatorilor posibilitatea vizualizării intuitive a datelor sub forma unui grafic. Implementarea acestui grafic s-a realizat prin folosirea modulului Javascript Google charts. Pentru folosirea acestui modul trebuie mai întâi format un tabel în format JSON. Pentru implementarea acestuia, se citesc datele senzorului de la care se doreşte vizualizarea de date şi se inserează într-o matrice PhP pentru care se vor defini capetele de tabel:

```
$table = array();

$table[,cols'] = array(

array(,label' => ,value', ,type' => ,number'),

array(,label' => ,date', ,type' => ,number') )
```

Tabelul în care au fost inserate datele din baza de date a platformei va fi transformat apoi în format JSON:

```
while($record = mysqli_fetch_array($test)) {

  $temp = array();

  $temp[] = array(‚v’ => (string) $record[‚value’]);

  $temp[] = array(‚v’ => (int) $record[‚date’]);

  $rows[] = array(‚c’ => $temp); }

$table[‚rows’] = $rows;

$jsonTable = json_encode($table);

echo $jsonTable;
```

Odată ce tabelul a fost transformat în format JSON, se poate apela funcţia `drawChart` a scriptului Javascript Google Charts, unde se va declara o variabilă ce va primi ca intrare tabelul în format JSON:

```
var data = new google.visualization.DataTable(<?php echo $jsonTable; ?>);
```

Apelarea funcţiei se face prin comanda:

```
chart.draw(data, options);
```

unde variabila `options` reprezintă configurări cosmetice ale graficului generat.

5. Concluzii

În lucrarea de faţă este prezentată o încercare de dezvoltare şi implementare funcţională proprie a unei platforme *IoT software*, iniţial cu un set relativ redus de funcţionalităţi de bază specifice, ca bază pentru potenţiale dezvoltări, implementări şi extinderi ulterioare.

Astfel, au fost prezentate arhitectura, componentele, funcţionalităţile şi legăturile funcţionale implicate în platforma *IoT software* experimentală proprie propusă, dezvoltată şi testată localizat, precum şi specificaţiile de dezvoltare şi implementare pentru cele mai importante dintre acestea şi câteva instrucţiuni de utilizare.

Platforma *IoT* experimentală a fost proiectată cu scopul colectării datelor furnizate de diferite dispozitive inteligente într-o bază de date comună, pentru ca acestea să poată fi accesate şi prelucrate de la distanţă. Totodată, platforma poate permite utilizatorilor şi controlul de la distanţă al dispozitivelor înregistrate şi conectate la platforma *Internet of Things*.

A fost prezentată arhitectura bazei de date şi tabelele definite şi conţinute de aceasta, precum şi legăturile între aceste tabele. În baza de date a platformei, datele se

stochează automat în mod privat pentru fiecare utilizator, diferenţiat în funcţie de dispozitivul deţinut de acesta de la care au fost primite datele respective. Utilizatorul are posibilitatea de a le monitoriza, consulta, vizualiza şi/sau de a le analiza, utilizând scripturile furnizate de platforma *IoT*, pentru a putea observa/descoperi apariţia unor valori neobişnuite sau a unor valori similare în datele primite şi stocate în cursul timpului.

A fost descrisă interfaţa *Web* a platformei *IoT* şi modul de lucru cu aceasta. Prin intermediul acesteia, un utilizator se poate înregistra/autentifica, poate adăuga (înregistra) microcontrolere şi/sau senzori sau poate consulta şi/sau vizualiza datele stocate pe platformă, provenind de la dispozitivele proprii, înregistrate de el anterior.

A fost prezentat, de asemenea, succint, microcontrolerul Arduino Yún utilizat în experimente şi modalităţile de comunicaţie între acesta şi senzori/actuatori şi, respectiv, între acesta şi platforma *IoT*, cu exemplificarea codului sursă al unui program (*sketch*) Arduino, scris în limbaj (similar cu limbajul) C, pentru citirea datelor de la un senzor de temperatură şi transmiterea acestora către serverul gazdă al bazei de date MySQL (şi al platformei *IoT*), prin apelul la un script PhP localizat pe aceasta.

Platforma a fost dezvoltată utilizând scripturi PhP, HTML şi Javascript şi foloseşte o bază de date MySQL ce conţine datele de autentificare ale utilizatorilor, cât şi dispozitivele inteligente pe care aceştia le-au înscris în platformă pentru a primi şi stoca datele de la ele. Legătura între platforma *IoT* şi dispozitivele inteligente se face printr-un script PhP ce este apelat de către dispozitivele inteligente atunci când acestea transmit date către platformă.

Au fost prezentate principalele scripturi PhP create şi utilizate, precum şi modul de lucru cu platforma.

Menţiuni

Prezenta lucrare are la bază parte din activităţile şi rezultatele fazei a II-a a proiectului PN 1609-0401 [11], derulat la ICI Bucureşti (2016-2017), în cadrul Programului naţional nucleu „COGNOTIC”, finanţat de Ministerul Cercetării şi Inovării.

Bibliografie

1. Bahga, A. & Madisetti, V. (2014). *Internet of Things: A Hands-On Approach*. Published by Bahga & Madisetti, ISBN: 978-099605515.

2. Botta. A., De Donato, W., Persico, V. & Pescapé, A. (2016). Integration of Cloud computing and Internet of Things: A survey, *Future Generation Computer Systems*, *56*, 684-700. Elsevier ScienceDirect.

3. Dayarathna, M. (2016). Comparing 11 *IoT* Development Platforms, *IoT Zone.* <https://dzone.com/articles/*IoT*-software-platform-comparison>.

4. Lehong, H. & Alfonso, V. (2014). *Hype Cycle for the Internet of Things*. Gartner Group.

5. Mercer, C. (2016). Internet of things platforms: Azure, AWS, IBM Watson and more - Which is the best *IoT* platform for your business?, *Computerworld*. <http://www.computerworlduk.com/galleries/data/-of-best-internet-of-things-platforms-3635185/>.

6. Mineraud, J., Mazhelis, O., Su, X. & Tarkoma, S. (2016). A gap analysis of Internet-of-Things platforms, *Computer Communications.* Elsevier ScienceDirect, DOI:10.1016.

7. Neagu, G., Florian, V., Preda, Ş. & Stanciu, A. (2016). Sensing as a service approach in health monitoring. In *IEEE Proc. of 15th RoEduNet Conference: Networking in Education and Research, Bucharest, Romania, Sept. 7-9* (pp. 225-229). eISSN: 2247-5443, DOI: 10.1109/RoEduNet.2016.7753240. <http://ieeexplore.ieee.org/abstract/document/7753240/>.

8. Neagu, G., Preda, Ş., Stanciu, A. & Florian, V. (2017). A Cloud-*IoT* based sensing service for health monitoring. In *IEEE Proc. of the 6th IEEE International Conference on E-Health and Bioengineering - EHB 2017, Sinaia, Romania, June 22-24* (pp. 53-56). ISBN: 978-1-5386-0358-1, DOI: 10.1109/EHB.2017.7995359. <http://ieeexplore.ieee.org/abstract/document/7995359/>.

9. Neagu, G., Vrejoiu, M. H., Preda, Ş. A. & Stanciu, A. (2017). Platforme *IoT* – Situaţia actuală şi tendinţe de evoluţie, *Revista Română de Informatică şi Automatică, 27*(3), 5-18. ISSN: 1220-1758, e-ISSN: 1841-4303. <https://rria.ici.ro/art-01-vol-27-nr-3-2017/>.

10. Perera, C., Zaslavsky, A., Christen, P. & Georgakopoulos, D. (2014). Sensing as a service model for smart cities supported by Internet of Things, *Trans. Emerg. Telecommun. Technol., 25*(1), 81-93.

11. Preda, Ş. A., Neagu, G., Vrejoiu, M. H. & Stanciu, A. (2017). Evaluare şi experimentare pentru platforme „Internet of Things", *Raport de cercetare etapa 2: Dezvoltare şi experimentare platformă de referinţă „Internet of Things", proiect PN 1609-0401*, ICI Bucureşti.

12. Singh, S. (2016). Top 10 *IoT* Platforms, *Internet of Things wiki*. <http://internetofthingswiki.com/top-10-*IoT*-platforms/634/>.

13. ***: *Site-ul Web Arduino Yún* <https://store.arduino.cc/arduino-yun>.

14. ***: *Site-ul Web Raspberry Pi* <https://www.raspberrypi.org/>.