Lucrarea 11.
IoT – Internet of Things - Accesul prin Internet la obiecte sau dispozitive simple

1. Obiectivul lucrarii

Lucrarea isi propune sa studieze posibilitatile de conectare a unor dispozitive simple la reteaua Internet. Vor fi monitorizate si controlate de la distanta anumite dispozitive simple (ex. LED, senzor de temperatura, motor electric, etc.) prin intermediul unui browser de web.

2. Consideratii teoretice

IoT sau “Internetul obiectelor” este o tendinta noua in domeniul comunicatiei pe Internet care isi propune sa asigure mecanismele necesare pentru conectarea unui numar foarte mare de dispozitive simple pe Internet si controlul acestora de la distanta. In viziunea IoT toate dispozitivele care ne inconjoara ar putea fi conectate pe Internet cu scopul de a fi monitorizate sau controlate din orice punct geografic care asigura o conectivitate la Internet. Dispozitive industriale, echipamente medicale, dispozitive de monitorizare a mediului sau a sanatatii pacientilor pot fi conectate la Internet in masura in care dispun de o interfata de retea si au capabilitatea de a implementa stiva de protocoale tipica pentru internet, adica TCP/IP.

Fig. 1. IoT- Internet of Things (wikipedia)

Realizarea unui astfel de deziderat implica solutionarea anumitor probleme specifice, cum ar fi: adresarea unui numar foarte mare de dispozitive (de ordinul zecilor de miliarde), optimizarea protocoalelor pentru transmisia de date de lungime redusa (biti, octeti), asigurarea securitatii si fiabilitatii datelor transmise, inregistrarea automata a datelor culese de la senzori, detectarea si tratarea evenimentelor, transmisia in timp-real a

datelor, etc. Multe din aceste cerinte se regasesc si in cazul retelelor industriale de comunicatii.

Deocamdata s-au facut putini pasi in aceasta directie, ceea ce ne obliga sa utilizam protocoalele existente pe Internet: in primul rand TCP/IP, HTTP sau XMPP (folosit pentru “almost real-time messeging).

Principalele aplicatii ale unei astfel de conectivitati ar fi: controlul de la distanta al proceselor industriale, monitorizarea mediului, automatizarea cladirilor, monitorizarea pacientilor, controlul traficului rutier si feroviar, urmarirea si asigurarea trasabilitatii produselor, etc.. O astfel de abordare ar permite utilizarea multipla a unor senzori sau dispozitive. De exemplu un senzor de nivel care masoara debitul unui râu ar fi utilizat de agentia de monitorizare a raurilor, de o companie hidroenergetica sau de Agentia de monitorizare a catastrofelor. Un alt exemplu ar fi o camera web care monitorizeaza traficul intr-o anumita zona si ar fi accesibil atat institutiilor responsabile cu controlul traficului cat si publicului larg.

Din punct de vedere al dispozitivelor conectabile pe Internet principala provocare este de a implementa setul de protocoale specifice Internetului pe un microsistem de calcul cu resurse limitate. Principalele neajunsuri ar fi: dimensiunea foarte mica a memoriei de date, lipsa unui sistem de operare (care sa inglobeze driverele de protocol), lipsa unui mod de lucru multi-threaded (necesar pentru tratarea in paralel a mesajelor transmise si receptionate.

Exista o serie de exemple de implementare a unei interfete Ethernet si chiar Internet pe sisteme bazate pe microcontroloare [1]. In general aceste solutii exploateaza in mod ingenios memoria de date existena si simuleaza o executie concurenta de tip multithreading. In cazul implementarii pe un microcontrolor de tip PIC16/18Fxx memoria de date disponibila este comparabila sau chiar mai mica decat dimensiunea unui mesaj minim Ethernet, ceea ce inseamna ca formarea mesajelor transmise si interpretarea celor primite se face “on the fly” adica in timpul tansmisiei propriu-zise.

Placa Arduino Intel Galileo este un caz fericit in care exista elementele constitutive necesare pentru implementarea ideii de IoT si anume: pe placa exista o interfata Ethernet (controlor Ethernet), pe placa ruleaza un sistem de operare (Linux minimal) si procesorul (un SoC – System on chip) dispune de memorie suficienta pentru gestionarea mesajelor si are performante de calcul suficiente (ex. frecventa ceasului de 400MHz). In plus placa dispune de interfete pentru achizitia si generarea de semnale analogice si digitale, prin care pot fi conectate diverse dispozitive de masura si de executie (senzori, relee, motoare electrice, etc.).

Printre exemplele oferite de mediul de programare Arduino exista cateva care demonstreaza modul in care placa poate fi conectata la Internet. Accesul la aceste exemple se face din meniul aplicatiei: File->examples->Ethernet. Primele exemple recomandate sunt “Web server” si Web client” in care aplicatiile implementeaza rolurile de server si client web. Aceste exemple arata secventa de comenzi necesare pentru realizarea unei conexiuni HTTP si transmiterea respectiv receptia unor pagini web. Prin aceste pagini web pot fi vizualizate anumite semnale achizitionate de placa. Se observa ca functiile de comunicatie in retea sunt solutionate prin biblioteca “Ethernet” accesibila prin fisierul header Ethernet.h. Detalii privind aceste functii pot fi accesate din mediul de programare Help->Reference->Librarie->Ethernet (ex: file:///C:/arduino-1.5.3/reference/Ethernet.html).

Pentru vizualizarea datelor receptionate sau transmise de catre placa Arduino se foloseste un canal serial virtual implementat pe conexiunea USB dintre PC si placa Arduino. Dupa lansarea programului se va deschide din meniu un monitor de canal serial (hyperterminal): Tools->Serial Monitor. In cadrul programului clasa "Serial" implementeaza functiile de receptie si transmisie.

3. Mersul lucrarii

3.1 Se vor testa exemplele simple (Web server si Web client) date in mediul de programare. Pentru accesarea placii se va folosi o aplicatie browser de web. In setarile browserului se va scoate temporar accesul la Internet prin "proxy" (daca exista) sau tratarea adresei IP setate ca o exceptie de la accesul prin proxy (ex. exception 10.129.025/32).
Atentie!!! adresa IP se va seta corespunzator configuratiei de retea in care se conecteaza placa; se recomanda utilizarea unui cablu de retea de la un calculator existent si copierea adresei de IP a calculatorului inlocuit.

3.2 Se vor conecta diferite dispozitive (ex. potentiometru, comutator, LED) la intrarile si iesirile analogice si digitale si se va testa posibilitatea de a citi valoarea intrarilor si de a seta valoarea semnalelor de iesire

3.3 Se va implementa aplicatia din anexa 1 si se va verifica transferul bidirectional al datelor. Se vor face modificari in program pentru a schimba regimul de lucru al aplicatiei.

Bibliografie

1. Jerome Bentham, TCP/IP Lean: Web Servers for Embedded Systems, Second Edition, http://read.pudn.com/downloads30/sourcecode/embed/95033/TCPIP%20Lean.pdf

Anexa 1 Aplicatie de citire a intrarilor analogice si control al unei iesiri (LED).

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

byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(10,129,0,25);

// Initializarea bibliotecii Ethernet server
// cu adresa  IP si portul dorit
// (port 80 este “default” pentru HTTP):
EthernetServer server(80);
String readString;

void setup() {
  // Deschide canal serial:
  Serial.begin(9600);
   while (!Serial) {
    ; // asteapta sa se deschida; doar pentru placa Leonardo
  }

 pinMode(13, OUTPUT);
  Ethernet.begin(mac, ip);  // porneste conexiunea Ethernet si serverul:
  server.begin();
  Serial.print("serverul este la ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  EthernetClient client = server.available();  // asculta apelul clientilor
  if (client) {
    Serial.println("client nou");
    boolean currentLineIsBlank = true;    // o cerere http se sfarseste cu o linie goala

    while (client.connected())
    {
      if (client.available())
      {
        char c = client.read();
         if (readString.length() < 100) {
              readString += c; //stocheaza caracterele in string
          //Serial.print(c);
      }
        Serial.write(c);
        // daca ai ajuns la sfarsitul liniei (receptionezi caracterul newline)
```

```
// si linia este goala, cererea http s-a sfarsit,
// deci poti sa trimiti un raspuns
if (c == '\n' && currentLineIsBlank)
{
  // trimite un header standard HTTP
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
  client.println();
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  // adauga un meta tag de refresh pentru ca
  //browserul sa resolicite pagina la fiecare 5 secunde:
  //client.println("<meta http-equiv=\"refresh\" content=\"5\">");
  // trimite valoarea fiecarui canal analogic
  for (int analogChannel = 0; analogChannel < 6; analogChannel++)
  {
    int sensorReading = analogRead(analogChannel);
    client.print("intrarea analogica ");
    client.print(analogChannel);
    client.print(" este ");
    client.print(sensorReading);
    client.println("<br />");
  }
  client.println("<a href=\"/?lighton\"\">Aprinde LED</a>");
  client.println("<a href=\"/?lightoff\"\">Stinge LED</a><br />");
  client.println("</html>");
  break;
}
if (c == '\n')
{
  // incepi o noua linie
  currentLineIsBlank = true;
}
else if (c != '\r')
{
  // ai primit un caracter pe noua linie
  currentLineIsBlank = false;

 }
       // Serial.println(readString);
 ////////////////////// controlul LED-ului de pe arduino
  if(readString.indexOf("?lighton") >0)//verifica “Aprins
  {
    digitalWrite(13, HIGH);    // pune bitul 13 pe HIGH
    Serial.println("Led Aprins");
```

```
          //Serial.println(readString);
          readString="";
         }
         else{
         if(readString.indexOf("?lightoff") >0)//verifica pentru stins
         {
          digitalWrite(13, LOW);    // pune bitul 13 pe LOW
          Serial.println("Led Stins");
          //Serial.println(readString);
          readString="";
         }
        }

      }
    }
    // timp necesar pentru ca browserul sa receptioneze data
    delay(1);
    // inchide conexiunea:
    client.stop();
    Serial.println("client deconectat");
    readString="";
  }
}
```