IoT Simulator

Iacob Denisa-Alexandra

Grupa 2B4

Tema 2

1 Introducere

1.1 Motivatie alegere proiect

Am ales proiectul IoT Simulator deoarece mi-am dorit sa aplic cunostintele dobandite nu doar pe partea de client, parte ce se axeaza mai mult de grafica, ci si pe partea de server, parte ce reprezinta „creierul" proiectului, iar acesta contine ambele parti, dar si pentru ca este un proiect organizat in mai multe servere concurente ce pot fi urmarite mult mai usor.

1.2 Rezumat functionalitati

Acest proiect implementeaza un client, cu interfata grafica, ce va juca rol de telecomanda pentru trei dispozitive: lumini, TV si aspirator. Aceste trei dispozitive sunt de fapt trei servere concurente TCP. Clientul poate alege la ce dispozitiv sa se conecteze si in functie de acesta vom avea telecomanda de unde ii putem modifica starea.Telecomanda pentru lumini va avea functiile de pornire,oprire, schimbarea culorii luminii dar si a intensitatii acesteia. Telecomanda pentru TV va avea obtiunile de pornire, oprire, schimbarea canalului si modificarea volumului. Telecomanda pentru aspirator va avea obtiunile de pornire,oprire si aria in care va aspira data prin lungimea si latimea camerei in centimetrii. Schimbarile de stare vor fi vizibile in client cat si in starea serverului.

2 Tehnologii utilizate

2.1 Protocolul

Pentru implementarea acestui proiect am ales sa utilizez modelul TCP concurent pentru implementarea conexiunii client-server deoarece este un protocol orientat spre conexiune, adica dispozitivele stabilesc o conexiune inainte de transmiterea datelor. Avem nevoie de acest lucru deoarece intre client si serverul la care alegem sa ne conectam trebuie sa existe o conexiune pentru ca schimbarile de stare dorite sa fi vizibile atat in client cat si in server, iar UDP nu ne poate oferi o garantie a livrarii

datelor in server si nu avem garantia ca schimbarile de stare vor avea loc sau ca datele vor fi transmise in ordine. Avem nevoie de TCP si pentru a sti la care dintre serverele concurente alegem sa ne conectam. Fiecare server TCP va avea o adresa IP si un port si in functie de acestea clientul va putea alege cu ce dispozitiv sa se conecteze.

2.2 Biblioteca interfata grafica

Biblioteca pentru interfata grafica pe care o voi utiliza este GTK deoarece este o biblioteca usor de folosit, cu foarte multe obtiuni dar in acelasi timp nu foarte complexa din punct de vedere al functiilor predefinite. Am ales aceasta biblioteca deoarece interfata mea grafica va trebui sa joace rol de telecomanda, iar aceasta biblioteca este una des utilizata pentru crearea de butoane si interactiunea cu utilizatorul spre deosebire de alte biblioteci care nu sunt concepute pentru interactiunea cu utilizatorul, ci mai mult pentru desenarea unei interfate de program sau prezentari. Voi utiliza GTK in loc de biblioteca Qt, biblioteca care este foarte asemanatoare cu GTK si la fel de des utilizata si in cea mai mare parte a software-ului GUI din lumea open source deoarece cred ca aceasta se pliaza mult mai bine pe crearea unei interfate tip telecomanda, pe langa faptul ca poate fi utilizata si in C, nu doar in C++.

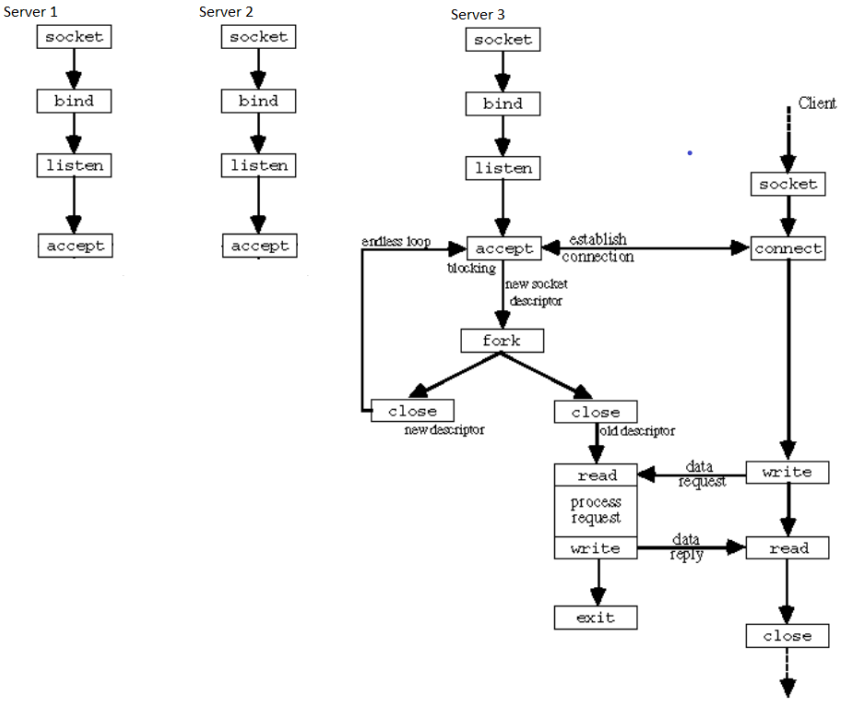
3 Arhitectura aplicatiei

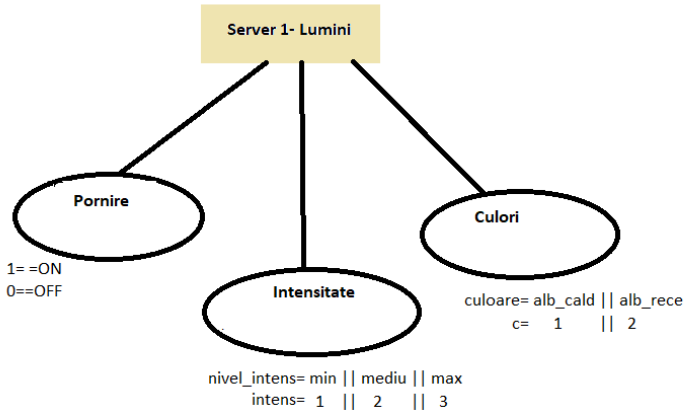
3.1 Concepte implicate

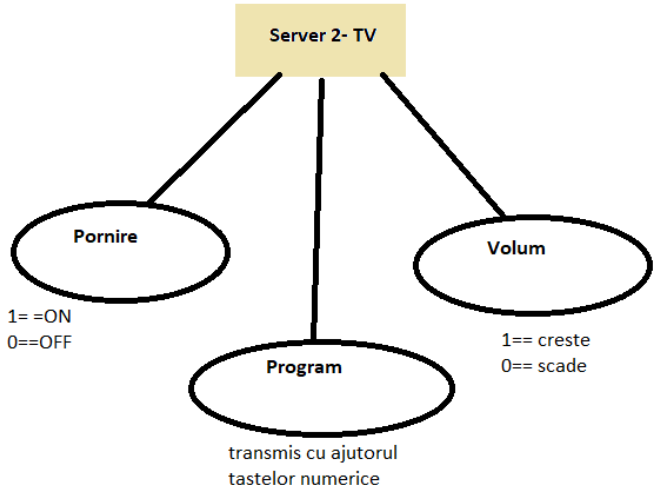
Dupa cum am mentionat si anterior, pentru implementare voi folosi modelul client/server concurent TCP. Acesta este necesar deoarece vom avea trei servere concurente, cate unul pentru fiecare dispozitiv, fiecare avand cate un port si o adresa IP. Clientul reprezentat printr-o interfata grafica cu rol de telecomanda isi va crea un socket si va alege la ce dispozitiv sa se conecteze cu ajutorul portului si adresei IP. Dupa ce se va realiza conexiunea clientul poate face anumite cereri in functie de serverul ales. Avem nevoie ca serverele sa fie concurente pentru a putea executa mai multe cereri inainte ca prima sa se fi incheiat de executat. Dupa executarea cererilor serverul va trimite raspuns clientului si astfel modificarile de stare vor fi vizibile atat in server cat si in client.

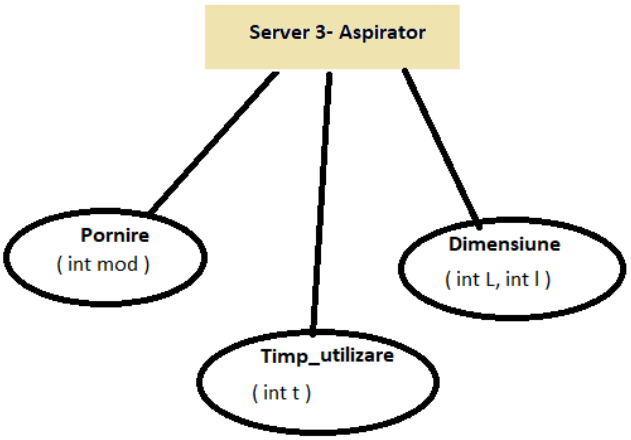
Interfata grafica pentru procesul client este utilizata deoarece o telecomanda poate avea foarte multe functii si este greu de retinut numele tuturor sau comanda pentru fiecare cererea. Interfata grafica va avea cate un buton pentru fiecare functie, aceasta fiind cat se poate de vizibila, prin urmare cu ajutorul butonului vom obtine mai rapid rezultatul dorit. Pe langa butoane, telecomanda noastra va avea si un meniu cu functiile, in dreptul careia va aparea si starea acesteia. Spre exemplu pentru functia pornire va aparea da imediat dupa ea, iar pentru culoare lumini va aparea culoarea.Acest lucru ne ajuta sa putem vedea schimbarile chiar daca nu ne aflam in apropierea dispozitivului.

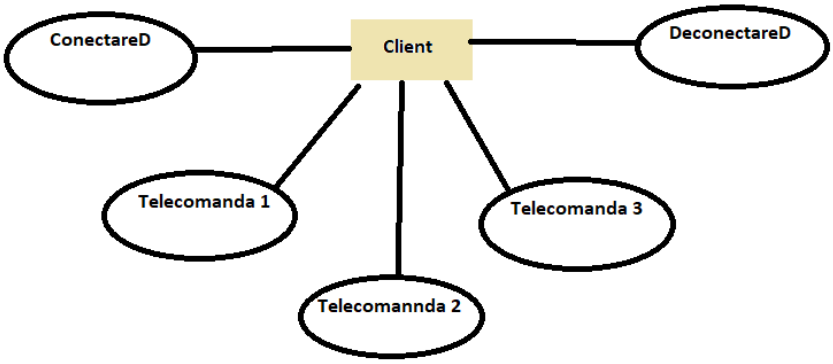
3.2 Diagrama aplicatiei detaliata






Server 2- TV
Pornire
1= =ON
0==OFF
Program
transmis cu ajutorul
tastelor numerice
Volum
1== creste
0== scade


Server 3- Aspirator
Pornire
(int mod)
Timp_utilizare
(int t)
Dimensiune
(int L, int l)



4 Detalii de implementare

4.1 Cod relevant proiectului

Conform diagramelor anterioare vom avea 3 servere concurente.Fiecare dintre ele isi va crea un socket pentru comunicare apoi vor face un apel de bind unde atasam socketul pentru a cunoaste adresa si portul, dupa care vor intra in starea de listen in care serverele vor asculta daca vin clienti. In momentul in care va aparea clientul serverele vor intra in starea de accept, stare in care se pregatesc sa primeasca conexiunea. Clientul isi va crea si el un socket pentru comunicare dupa care va face o cerere de conectare in functie de dispozitivul ales, catre serverul ce reprezinta acel dispozitiv. Cu ajutorul apelului de bind facut de serverul respectiv, vom avea adresa si portul serverului, iar clientul se va putea conecta la socketul serverului. Dupa ce serverul accepta conexiunea cu clientul, acesta va face un fork pentru a putea sa se intoarca la a prelua cereri cat timp procesul copil se ocupa de cererea deja facuta. Procesul copil inchide socketul pentru acceptarea de conexiuni si printr-o serie de apeluri read si write va procesa cererea. Procesarea cererii se va face cu ajutorul functiilor implementate pentru fiecare server in parte (pornire, culori, volum, etc.). Dupa ce serverul prin apelul de read va primi de la client modificarile facute, va apela functia necesara cererii dupa care va trimite prin apelul de write un raspuns clientului ce consta intr-un mesaj care va fi citit si afisat de catre client. Dupa solutionarea cererilor facute de client, acesta va inchide conexiunea, iar procesul copil din server va da exit, adica va disparea.

In cele 3 servere codul este acelasi, deoarece vor face acelasi lucru, difera doar portul acestora pentru ca totusi sunt dispozitive diferite.

```
/* portul folosit pentru serverul TV */
#define PORT 2125

/* portul folosit pentru serverul Luminilor */
#define PORT 2126

/* portul folosit pentru serverul Aspiratorului*/
#define PORT 2127
```

Codul pentru implementarea serverelor concurente este cel din laboratorul 7 https://profs.info.uaic.ro/~gcalancea/lab7/servTcpConc.c la care am modificat transmiterea raspunsului catre client

```
printf ("[server]Mesajul a fost receptionat...%s\n", msg);
      /*pregatim mesajul de raspuns */
      bzero(msgrasp,100);
       if(strstr(msg,"pornit")==0)
         strcat(msgrasp,"Nu ati pornit dispozitivul");  /*Aceasta este o atentionare trimisa in cazul in care s-a cerut salvarea modificarilor dar fara a porni dispozitivul */
        else
         strcat(msgrasp,"Modificari salvate"); // serverul salveaza modificarile
      printf("[server]Trimitem mesajul inapoi...%s\n",msgrasp);
```

In client avem mai multe functii pentru implementarea widget-urilor. Pentru implementarea acestora am folosit portiuni de cod de pe urmatoarele pagini https://zetcode.com/gui/gtk2/ dar si http://www.manpagez.com/html/gtk3/gtk3-3.16.5/index.php

Aceasta functii sunt folosita de client pentru conexiunea cu serverul tv dar si pentru crearea butoanelor ce vor aparea in telecomanda tv:

```
void TV(GtkWidget *widget, gpointer data)
{
  // declararea widget-urilor
  GtkWidget *button;
  GtkWidget *hscale;
  GtkWidget *radio;
  GtkWidget *radio2;
  GtkWidget *label;
  GtkWidget *label1;
  GtkWidget *entry1;
  GtkWidget *buttone;

//mesajul si descriptorul de socket sunt declarte global
  struct sockaddr_in server; // structura folosita pentru conectare
  //stabilim portul serverului TV
  port = atoi ("2125");
```

```
    /* cream socketul */
    if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
      {
       perror ("Eroare la socket().\n");
       return errno;
      }

    /* umplem structura folosita pentru realizarea conexiunii cu serverul */
    /* familia socket-ului */
    server.sin_family = AF_INET;
    /* adresa IP a serverului */
    server.sin_addr.s_addr = inet_addr(IP);  /*adresa IP este aceeasi pentru toate
serverele si a fost declarata global*/
    /* portul de conectare */
    server.sin_port = htons (port);

     /* ne conectam la server */
    if (connect (sd, (struct sockaddr *) &server,sizeof (struct sockaddr)) == -1)
      {
       perror ("[client]Eroare la connect().\n");
       return errno;
      }

       bzero (msg, 100);

    gtk_window_set_title(window, "TV");  //setam titlul ferestrei
    gtk_widget_destroy(vbox); /*distrugem cutiile folosite anterior pentru a goli
containarul si a putea fi refolosit*/

   //initializam cutii noi
    vbox = gtk_vbox_new(FALSE,50);
    hbox = gtk_hbox_new(0, 10);

   //adaugam cutia principala la container, celelalte vor fi adaugate in cea principala
    gtk_container_add (GTK_CONTAINER (window), vbox);

   //creem doua butoane radio pentru pornirea si oprirea dispozitivului
   //dupa care apelam functia pentru crearea mesajului care va fi trimis la sever
     radio = gtk_radio_button_new_with_label( NULL, "Oprit");
    g_signal_connect(radio,"toggled", G_CALLBACK(toggled_func), (gpointer)"1");

    radio2 = gtk_radio_button_new_with_label (gtk_radio_button_get_group(
GTK_RADIO_BUTTON(radio)),"Pornit");
    g_signal_connect(radio2,"toggled"G_CALLBACK(toggled_func), (gpointer)"2");
```

```
    //widget-ul pentru volum
      hscale = gtk_hscale_new_with_range(0, 100, 1);
      gtk_scale_set_draw_value(GTK_SCALE(hscale), TRUE);
      gtk_widget_set_size_request(hscale, 150, -1);
      label = gtk_label_new("Volum");

    //widget-ul pentru canal
      label1 = gtk_label_new("Canal");
      entry1 = gtk_entry_new();
      buttone = gtk_button_new_with_mnemonic("Modifica");
    //adaugarea acestora in boxul orizontal
      gtk_box_pack_start (GTK_BOX(hbox), label1, FALSE, FALSE, 0);
      gtk_box_pack_start (GTK_BOX(hbox), entry1, FALSE, FALSE, 0);
      gtk_box_pack_start (GTK_BOX(hbox), buttone, FALSE, FALSE, 0);

    //butonul pentru salvare care va trimite modificarile la server
       button = gtk_button_new_with_label ("Salveaza & Meniu");
      g_signal_connect (button, "clicked", G_CALLBACK (MeniuT), NULL);

    //adaugam tot la boxul principal vertical
      gtk_box_pack_start(GTK_BOX(vbox), radio, FALSE, FALSE, 0);
      gtk_box_pack_start(GTK_BOX(vbox), radio2, FALSE, FALSE, 0);
      gtk_box_pack_start(GTK_BOX(vbox), hscale, FALSE, FALSE, 0);
      gtk_box_pack_start(GTK_BOX(vbox), label, FALSE, FALSE, 0);
      gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 0);
      gtk_box_pack_start(GTK_BOX(vbox), button, TRUE, TRUE, 0);
      g_signal_connect(hscale,"value-changed",
G_CALLBACK(value_changed),NULL);
    //apelarea functiei pentru mesajul de schimbare a canalului
      g_signal_connect (buttone, "clicked", G_CALLBACK(button_entryc), entry1);

     gtk_widget_show_all (window);   //afiseaza fereastra

    }


    Urmatoarea functie este folosita pentru trimiterea modificarilor  la server

    void MeniuT()
    {

      GtkWidget *televizor;
      GtkWidget *lumini;
      GtkWidget *aspirator;
```

```
GtkWidget *quit;


  /* citirea mesajului */
fflush (stdout);
read (0, msg, 100);

/* trimiterea mesajului la server */
if (write (sd, msg, 100) <= 0)
  {
    perror ("[client]Eroare la write() spre server.\n");
    return errno;
  }

 /* citirea raspunsului dat de server
   (apel blocant pina cind serverul raspunde) */
if (read (sd, msg, 100) < 0)
  {
    perror ("[client]Eroare la read() de la server.\n");
    return errno;
  }
/* afisam mesajul primit */
printf ("[client]Mesajul primit este: %s\n", msg);
close (sd);

//Refacem fereastra de meniu principal
  gtk_widget_destroy(hbox);
  gtk_widget_destroy(vbox);
 vbox = gtk_vbox_new(TRUE, 50);
 gtk_container_add(GTK_CONTAINER(window), vbox);

 televizor = gtk_button_new_with_label("Televizor");
 lumini = gtk_button_new_with_label("Lumini");
 aspirator = gtk_button_new_with_label("Aspirator");
 quit = gtk_button_new_with_label("Quit");


 gtk_box_pack_start(GTK_BOX(vbox), televizor, TRUE, TRUE, 0);
 g_signal_connect (televizor, "clicked", G_CALLBACK (TV2), NULL);
 gtk_box_pack_start(GTK_BOX(vbox), lumini, TRUE, TRUE, 0);
 g_signal_connect (lumini, "clicked", G_CALLBACK (Lumini), NULL);
 gtk_box_pack_start(GTK_BOX(vbox), aspirator, TRUE, TRUE, 0);
 g_signal_connect (aspirator, "clicked", G_CALLBACK (Aspirator), NULL);
 gtk_box_pack_start(GTK_BOX(vbox), quit, TRUE, TRUE, 0);
```

```
    g_signal_connect_swapped(quit,"clicked", G_CALLBACK(gtk_widget_destroy),
window);

    gtk_widget_show_all(window);

  }
```

Functiile din client pentru celelalta optiuni Lumini si Aspirator sunt asemanatoare cu cea pentru TV doar ca se folosesc widget-uri diferite

4.2 Scenarii de utilizare

Acest proiect cu ajutorul interfatei grafice din client va putea alege la ce dispozitiv sa se conecteze din cele trei date. Dupa ce conexiunea intre client si server va avea loc interfata grafica va arata ca o telecomanda de unde vom putea seta diferite modificari in starea dispozitivului, cum ar fi: pornire, oprire, etc., in functie de dispozitivul ales. Interfata grafica va contine si un meniu al optiunilor telecomenzii. Toate modificarile facute de client vor fi trimise la server iar acesta poate trimite ca raspuns erori intampinate sau simple atentionari cum ar fi “Nu ati pornit dispozitivul”, iar in acest caz modificarile nu pot fi salvate.Pentru a remedia eroarea utilizatorul va trebui sa se reconecteze la dispozitiv.

5 Concluzii

Solutia data cred ca ar putea fi imbunatatita cu ajutorul unei criptari pentru ca nu toata lumea sa poata avea acces la dispozitivele noastre deoarece aceasta telecomanda ar putea fi folosita si pentru sisteme de supraveghere si nu ne dorim ca oricine sa poata avea acces. In acest scop cred ca ar fi utile si niste functii de login si logout. O alta functie care ar fi folositoare ar putea fi un temporizator care spre exemolu ar putea stinge luminile dupa un anumit timp dat de utilizator in caz ca le uitam aprinse.O baza de date ar ajuta in care sa putem tine minte schimbarile de stare pentru ca acestea sa nu se piarda in momentul in care am ales sa ne conectam la alt dispozitiv.

6 Bibliografie

- https://profs.info.uaic.ro/~georgiana.calancea/laboratories.html#
- https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
- https://zetcode.com/gui/gtk2/introduction/
- https://www.gtk.org/
- http://www.manpagez.com/html/gtk3/gtk3-3.16.5/index.php

- https://www.alternative-computer-programming.com/cplusplus-with-qt-tutorial-index.html
- https://www.makeuseof.com/tag/difference-gtk-qt/
- https://www.geeksforgeeks.org/differences-between-tcp-and-udp/
- https://www.youtube.com/playlist?list=PL6A4216D6C8E00CFC