



**Cuprins**

Capitolul 1: Prefata………………………………………..5

Prezentare……………………………………………6

Motivatie…………………………………………….7

Pe scurt despre raspberry PI…………………………8

Capitolul 2: Notiuni Fundamentale……………………….9

Android…………………………………………….10

Java…………………………………………………13

Bluetooh……………………………………………16

Capitolul 3: Instalare si utilizare………………………...20

De ce este nevoie ? ………………………………...21

Configurarea telefonului …………………………..22

Configurarea dispozitivului………………………..23

Asociere……………………………………………25

Prezentarea Aplicatiei……………………………..26

Legenda……………………………………………27

Componente……………………………………….28

Exemple……………………………………………33

Capitolul 4: Partea tehnica a aplicatiei………………….36

Android Studio…………………………………….37

Notiuni Tehnice……………………………………38

Alcatuirea Aplicatiei……………………………….45

Structura Clientului………………………………....46

Folderul manifests……………………………..47

Permisiuni………………………………..49

Folderul java………………………………………..51

Folderul java(generated)……………………………57

Folderul res…………………………………………58

Folderul Gradle Scripts……………………………..60

Bibliografie…………………………………………62

**Capitolul I**

**Prefata**

**Prezentare**

Overmind este o aplicatie dezvoltata pentru sistemul de operare **Android** ce permite controlarea de la distanta, prin **Bluetooth**, a altor dispozitive compatibile.

Comunicarea intre device-uri este asemanatoare **SSH**-ului (Secure Shell), si anume, se poate controla dispozitivul prin intermediului “Liniei de comanda” .

Principala caracteristica a acestei aplicatii este ca nu are nevoie de internet pentru a functiona, tot traficul facandu-se local, prin Bluetooth.

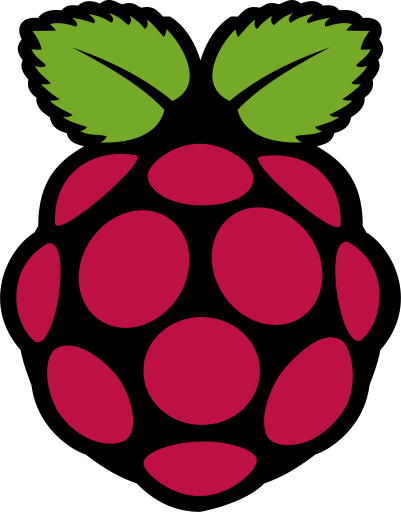
**Motivatie**

Motivatia acestui proiect a fost dorinta mea de a putea comunica cu alte dispozitive (initial, o placa de dezvoltare **Raspberry Pi**) chiar si in absenta internetului.

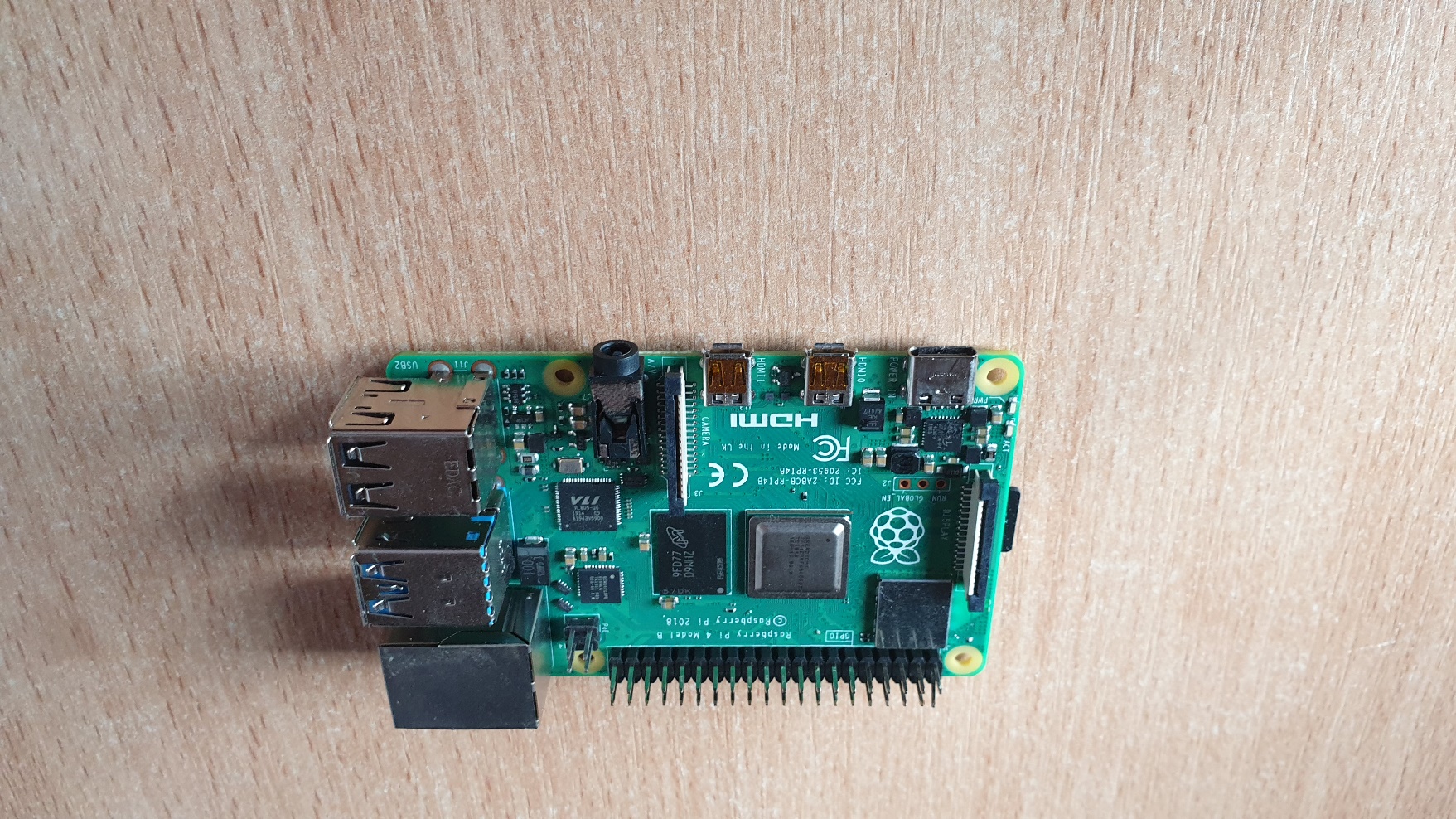
De obicei placile Raspberry se controleaza prin intermediul SSH-ului, acest lucru necesitand o conexiune la Wi-fi.

Daca nu exista o astfel de conexiune sau daca dispozitivul nu era autentificat pe o retea, era necesara conectarea unui monitor, al unei tastaturi si al unui mouse pentru a putea fi folosita interfata grafica a placii de dezvoltare, ca apoi sa introducem manual datele de login.

Pentru a evita aceasta situatie am creeat aplicatia “**Overmind**” care imi permite sa controlez un dispozitiv prin Bluetooth, chiar si daca nu ma aflu intr-o zona acoperita de Wi-fi.

**Pe scurt despre Raspberry Pi**

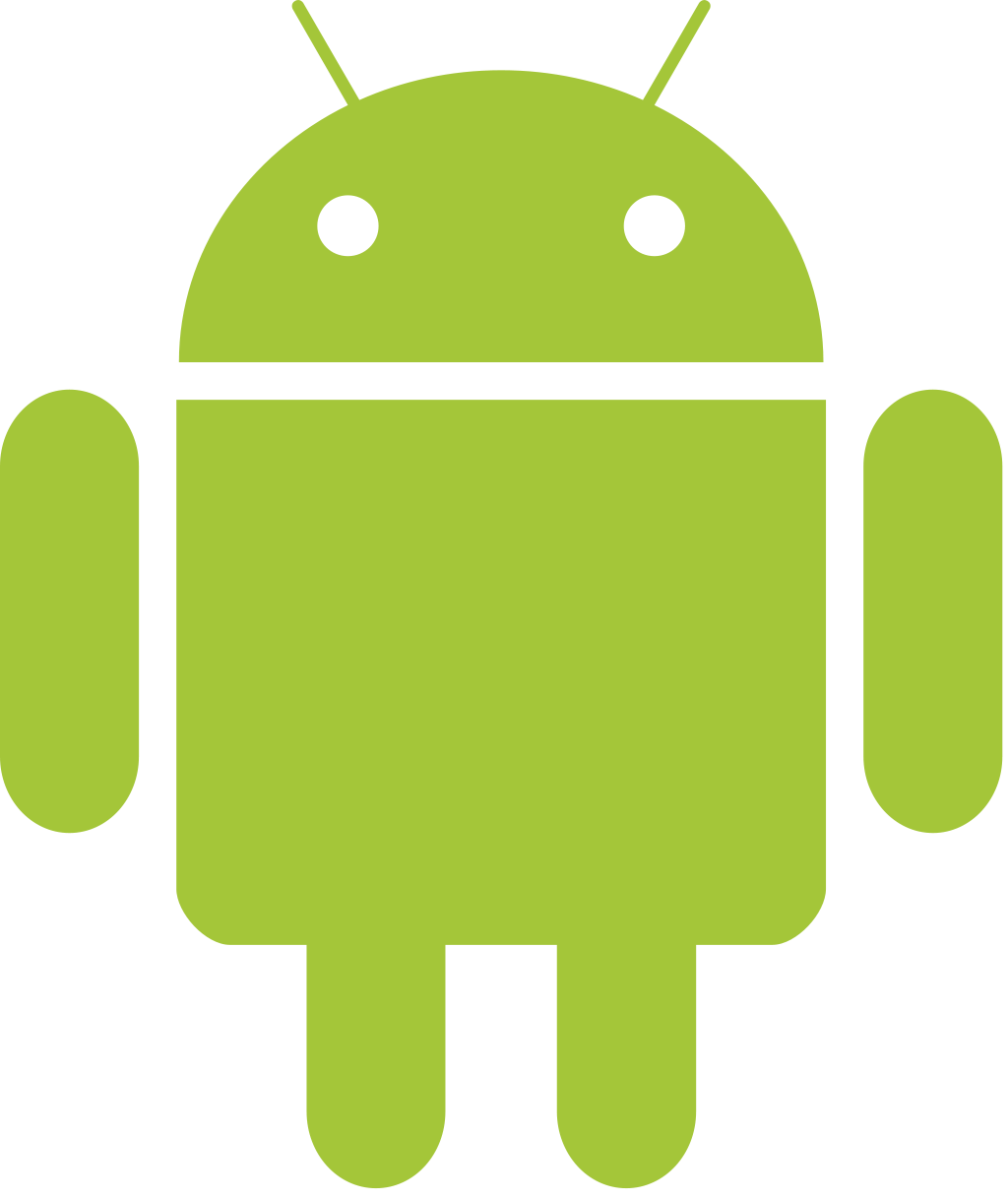
Placile de dezvoltare pot fi numite “mini-computere”, si au numeroase aplicatii in **IOT** (Internet of things) si informatica aplicata.

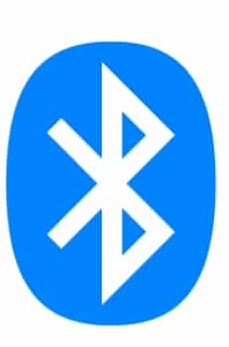


Raspberry Pi Model 4 B

De cele mai multe ori, acestea ruleaza pe un sistem de operare numit **Raspbian**, o distributie de **Linux** special creeata pentru aceste dispozitive.

**Capitolul II**

**Notiuni Fundamentale**

****

**Android**

Este un sistem de operare dezvoltat de **Google**, pentru dispozitivele cu touchscreen (telefoane, tablete, etc…) . Cu toate acestea, distributii ale sistemului android se pot regasi si pe o varietate larga de electronice cum ar fi: **televizoare smart, console, masini, camere de filmat** si chiar **desktopuri.**

Android este de asemenea cel mai folosit sistem de operare din lume, avand peste 3 miliarde de utilizatori lunari, principalul lui competitor pe piata fiind **IOS** (dezvoltat de Apple).



**Putin despre partea tehnica a Android**

Din punct de vedere software, android este un proiect **open-source** (oricine poate contribui la dezvoltarea lui) deosebit de complex deoarece acesta a fost creeat pentru a rula pe o gama foarte larga de dispozitive.

Aplicatiile pentru android sunt scrise in limbajele de programare **Java** sau **Kotlin** (bazat pe Java) si iau forma unor fisiere **APK** (Android Package).

Un detaliu remarcabil este faptul ca android, este construit pe **kernelul** Linux, deoarece si acesta este open-source, iar inginerii Google au putut modifica usor codul sursa pentru a satisface cerintele sistemului de operare.

\*Kernel = componenta unui sistem de operare ce face legatura intre partea software si partea hardware a dispozitivului (procesor, memorie, etc..)

**Arhitectura sistemului Android**





**Java**

Este un limbaj de programare, **low-level** , **orientat pe obiecte** lansat in 1995, intretinut si dezvoltat de **Oracle**, fiind al treilea cel mai rapid din punct de vedere al timpului de executie (dupa C si C++).

Principala caracteristica a limbajului este ca, odata compilata, o aplicatie poate rula pe orice platforma si orice arhitectura de procesor, fara a mai fi necesara o **recompilare**. Acesta este unul din principalele motive pentru care **Android** ruleaza aplicatii scrise in **Java** sau **Kotlin**.

****Potrivit Stack Overflow, **Java** este al treilea cel mai folosit limbaj de programare din lume (dupa JavaScript si Python)

**Cum functioneaza Java**

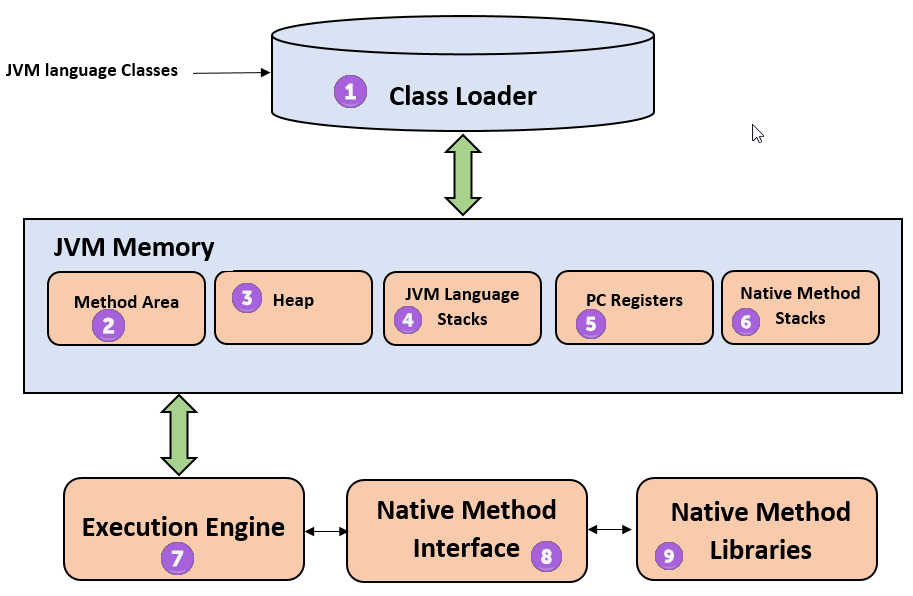
Motivul pentru care programele scrise in **Java** ruleaza pe orice dispozitiv este datorat compilarii, care nu transforma codul in instructiuni care pot fi intelese de procesor, ci in instructiuni ce sunt executate de **JVM** (Java Virtual Machine).

**JVM**-ul, asa cum ii precizeaza si numele, este o masina virtuala ce poate rula numai instructiuni Java. O **masina virtuala** este un sistem de calcul abstract (virtual), adica este o simulare software a unui computer fizic (un computer in interiorul altui computer 😊).

****Programele Java sunt dezvoltate cu ajutorul unei unelte ce se numeste **JDK** (Java Development Kit) ce contine unelte utile in scrierea programelor software.

**Arhitectura JVM**

**(Java Virtual Machine)**



****

**Bluetooth**

Este este o **tehnologie wireless** de raza scurta folosita pentru schimbul de date intre dispozitive mobile folosind **unde radio**.

Bluetooth este folosit in general, ca o **alternativa** a conexiunilor prin fir intre dispozitive, fiind tehnologia ce face posibila functionarea tuturor dispozitivelor **wireless** (casti, boxe portabile, tastaturi wireless, etc…)

Frecventele radio pe care opereaza **Bluetooth** se numesc **ISM** deoarece acestea sunt dedicate folosirii in scop **I***ndustrial*, **S***tiintific* si **M***edical.*

****Ultima versiune de Bluetooth existenta la momentul redactarii, ce este folosita si in proiect este **Bluetooth 5**.

**Cum functioneaza Bluetooth**

Comunicarea Bluetooth se face prin unde radio cu frecvente cuprinse intre **2.400 GHz** si **2.483** **GHz**, acestea reprezentand frecventele **ISM**.

In cel mai folosit mod al sau, puterea de transmisie a Bluetooth este limitata la **2.5 miliwati**, acest lucru imbunatatind consumul bateriei si dandu-i o raza foarte scurta de actiune de numai **10 metri**.

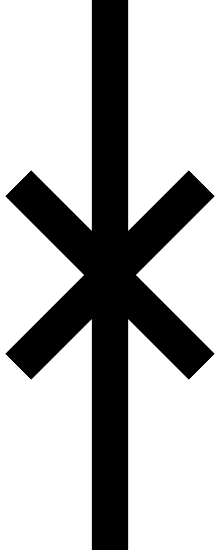
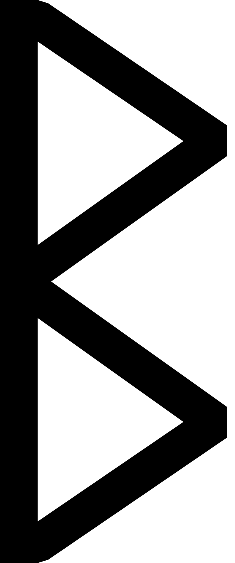
Bluetooth este de 2 tipuri:

* Bluetooth classic (Bluetooth BR/EDR) ce este cel mai folosit, putand transfera date mai rapid. De asemenea acest mod consuma mai multa energie.
* ****Bluetooth Low Energy (Bluetooth LE) care este mai incet dar mai eficient din punct de vedere al energiei

**De ce Bluetooth ?**

Bluetooth a fost numit dupa un fost rege din Danemarca numit Harald Gormsson. Acesta era cunoscut deoarece avea un dinte albastru, de unde si porecla de Blue-Tooth.

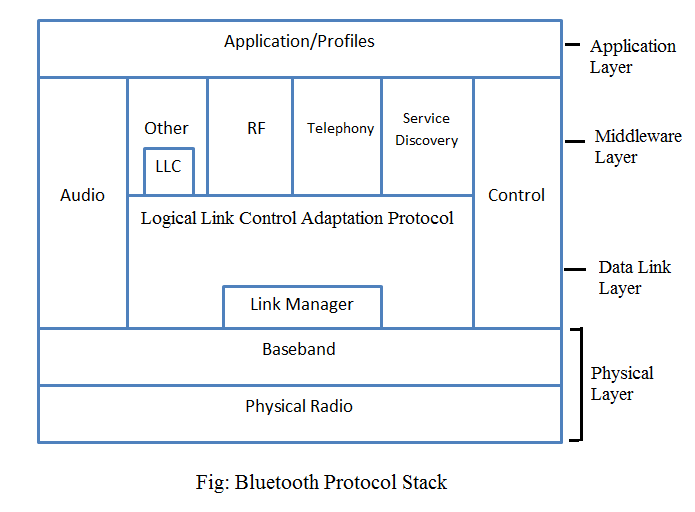
Nestiind cum sa isi numeasca proiectul, creatorii i-au dat numele de Bluetooth dupa ce unul din ei a avut o conversatie interesanta despre istoria Scandinaviei.

Logoul este constituit din initialele regelui in rune nordice:

Bjarkan

Hagall

****

**Structura Protocolului Bluetooth**

****

**\*Mai multe informatii despre bluetooth vor fi prezentate in capitolele urmatoare.**

**Capitolul III**

**Instalare si Utilizare**

**De ce este nevoie ?**

De un telefon personal si dispozitivul pe care dorim sa il utilizam.

Telefonul:

* **Android** 8 sau versiuni superioare

(Aplicatia ar putea functiona si pe versiuni mai vechi, dar nu a fost testata si pot aparea bug-uri)

* Capabilitate Bluetooth

Dispozitiv:

* Sistem de operare **Windows**, **Linux** sau **MacOS**
* **Python** 3.8 sau o versiune mai mare (pentru Windows nu este neaparat necesar).
* Capabilitate Bluetooth

**Configurarea Telefonului**

Dupa ce a fost descarcata, aplicatia poate fi insalata cu un simplu tap pe numele ei.

Inainte de pornirea aplicatiei, ne asiguram daca Bluetooth este pornit:

Setari > Conexiuni > Bluetooth

In cazul in care Bluetooth nu este pornit cand pornim aplicatia, o casuta text va aparea si ne va intreba daca vrem sa activam Bluetooth.

Daca raspunsul este nu, aplicatia se va inchide si nu o vom putea folosi pana cand Bluetooth nu este activat.

**Configurarea Dispozitivului**

**Linux / MacOs:**

Mai intai activam Bluetooth.

Pentru instalarea server-ului, este necesar un terminal unde vom scrie comenzi:

**1.**

Schimbam folderul de lucru in cel in care vrem sa intalam serverul, in cazul meu **“/home/pi/btSSH**”



**2.**

Descarcam codul serverului folosind **wget**.



**3.**

Rulam serverul in background si cu drept de **root**.

****

**Windows:**

Metoda 1:

1. Accesam cu un browser linkul: [www.overmindapp.cf/Undermind.exe](http://www.overmindapp.cf/Undermind.exe)

Apoi descaracm fisierul.

1. Rulam cu drepturi de administrator.

(serverul nu va creea nici o fereastra ci va rula in background.)

Metoda 2:

1. Accesam cu un browser linkul:  [www.overmindapp.cf /UndermindW.py](http://www.nezrak.000webhostapp.com/UndermindW.py)
2. Copiem codul sursa al server-ului
3. Rulam cu **Python**.

**Asociere**

In primul rand, trebuie sa asociem telefonul cu dispozitivul prin procedeul standard (asa cum am face cu niste casti wireless, de exemplu).

Acest lucru il putem face din:

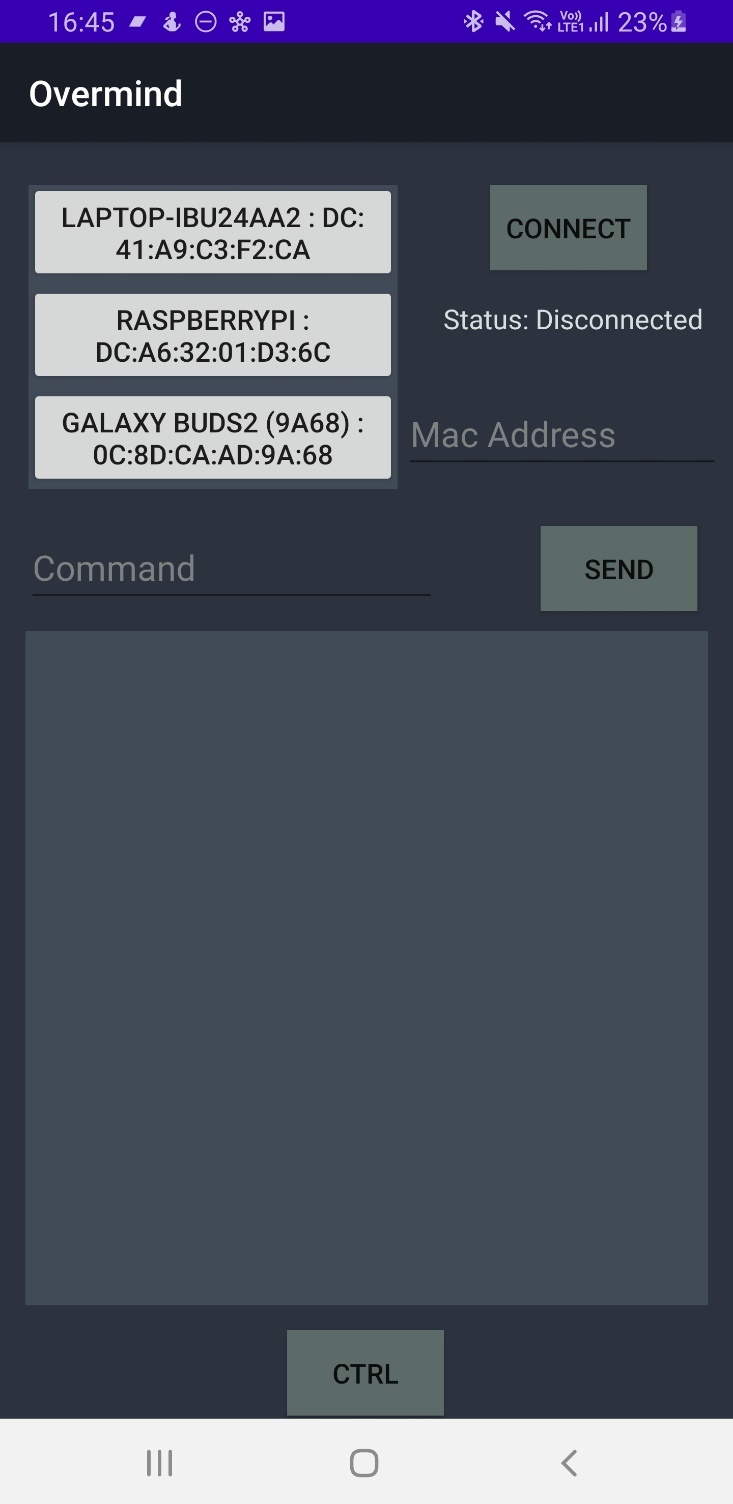
Setari > Conexuni > Bluetooth

Telefonul ne va arata o **lista** cu dispozitivele bluetooth ce sunt deja asociate cu telefonul nostru, iar sub aceasta, o **lista** cu dispozitivele din jurul nostru.

Dupa ce localizam dispozitivul dorit, dam un simplu tap pe numele lui **si Android** va face restul asocierii.

Acest pas trebuie facut doar prima data, deoarece dispozitivul si telefonul vor ramane asociate de acum inainte.

**Prezentarea Aplicatiei**

****

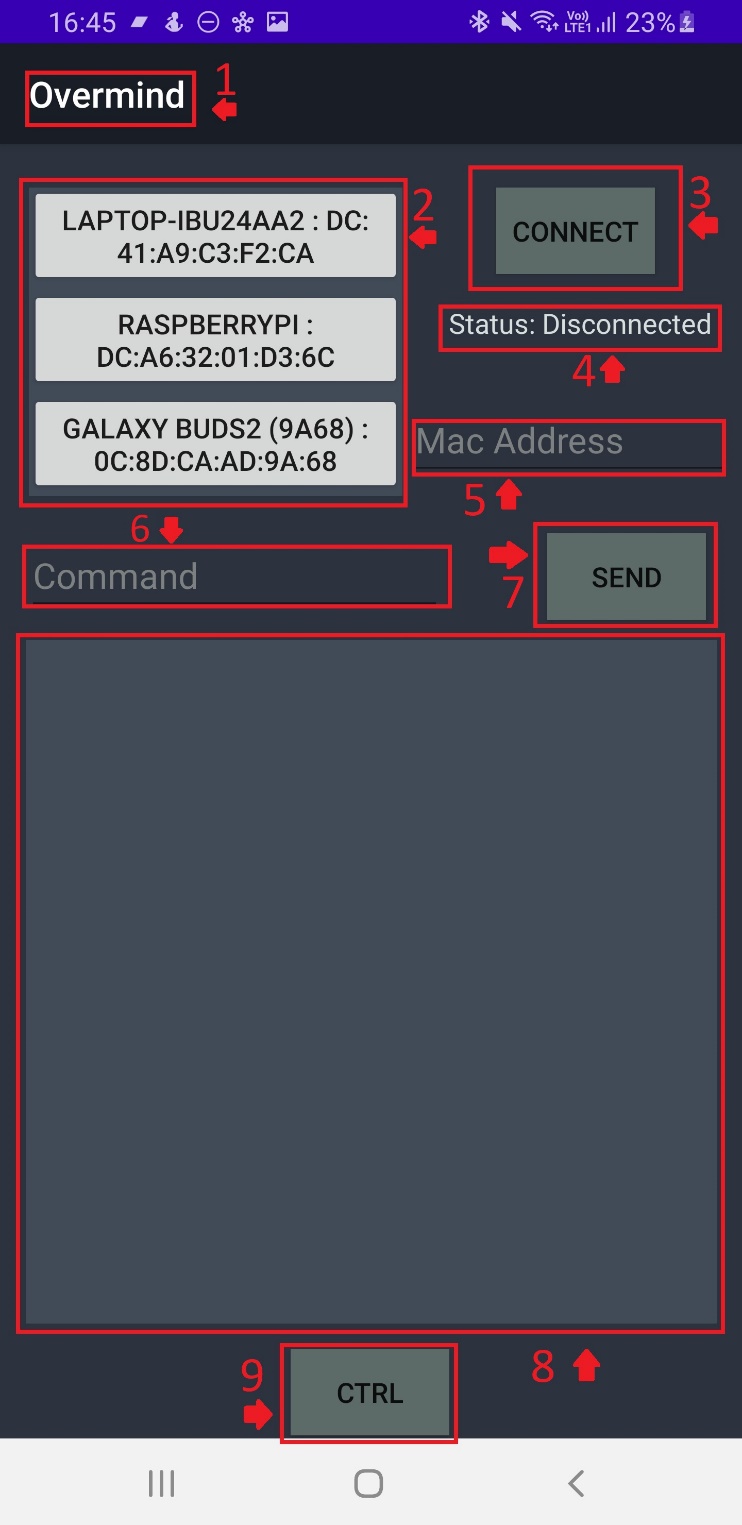
* Aplicatia se deschide direct in meniul principal unde vom face toate interactiunile.
* Culorile aplicatiei sunt alese in asa fel sa fie in tema cu “Bluetooth” si sa nu solicite ochiul privitorului, mai ales noaptea.

\*Un screenshot al aplicatiei

\*Schema de culori folosita:

https://colorhunt.co/palette/0820322c394b334756ff4c29

**Legenda**

****

1. Title
2. Device Listbox
3. Connect Button
4. Label status
5. Mac input
6. Command input
7. Send Button
8. Output
9. CTRL Button

**Componente**

1. **Titlu**

Titlul este pus automat de sitemul Android in partea superioara a aplicatiei, majoritatea programatorilor aleg sa il scoata manual, dar eu am decis sa il las acolo😊 .

1. **Device Listbox**

Este o lista unde vor aparea toate dispozitivele cu care telefonul a fost asociat in trecut. Din motive de spacing, marimea acestei liste este limitata la 3 elemente dar, daca exista mai multe dispozitive, putem da scroll in lista. Daca dam tap pe un element, **MAC**-ul asociat lui se va insera automat in “Mac input” (4).

\*MAC = Media Acces Control, este un tip de adrese care ne ajuta sa diferentiem dispozitivele intre ele, fiecare avand o adresa unica in lume.

1. **Connect Button**

Dupa ce il apasam, telefonul va incerca sa se conecteze la dispozitivul pe care l-am ales si va da un mesaj in cazul unei reusite sau al unui esec.

1. **Label Status**

Este o secventa de text ce ne arata daca aplicatia este sau nu conectata la un dispozitiv, pentru a face folosirea aplicatiei mai usoara.

1. **Mac Input**

Conexiunea Bluetooth se face pe baza adresei MAC. Aici introducem adresa dispozitivului la care urmeaza sa ne conectam. Este recomandat ca adresa sa nu se introduca manual ci prin intermediul “Device Listbox” (2).

1. **Command Input**

Acest camp nu poate fi folosit pana cand nu a fost efectuata o conexiune la un dispozitiv. Odata conectat, un simplu tap pe camp va face vizibila tastatura telefonului pe care o vom utiliza pentru a scrie comenzile pe care urmeaza sa le trimitem catre dispozitiv.

1. **Send Button**

Nici acest buton nu poate folosit pana cand telefonul nu se conecteaza la alt dispozitiv. Dupa ce scriem o comanda in “Command Input”, o trimitem catre dispozitiv prin intermediul acestui buton.

1. **Output**

In acest spatiu vor aparea rezultatele comenzilor trimise catre dispozitiv. Acesta se actualizeaza in timp real odata cu trimiterea outputul de la dispozitiv la telefon. Aici se salveaza tot ce se trimite si tot ce se primeste in comunicarea dintre device-uri. Cand informatiile din output umplu toata capacitatea campului, acestea se vor adauga in continuarea celor existente, avand capabilitate de **scroll**.

1. **CTRL Button**

Si acest buton este blocat cat timp telefonul nu e conectat. Cand il apasam, in “Command Input” (6) va fi introdus caracterul special “%CTRL%”. Acest caracter ne permite sa trimitem comenzi precum (CTRL+C), asa cum am putea face cu o tastatura de desktop. Aceste tipuri de instructiuni sunt esentiale pentru interactiunile cu linia de comanda. In cazul nostru, o comanda ca “%CTRL%Z” va fi echivalenta cu (CTRL+Z).

**Utilizare**

Dupa ce conectam dispozitivul si telefonul asa cum a fost precizat in sectiunea “Componente”, putem incepe sa trimitem comenzi catre dispozitiv.

Comenzile acceptate, sunt cele ale liniei de comanda a sistemului de operare de pe dispozitiv. In cazul unei comenzi gresite, se va afisa mesajul de eroare al sistemului de operare.

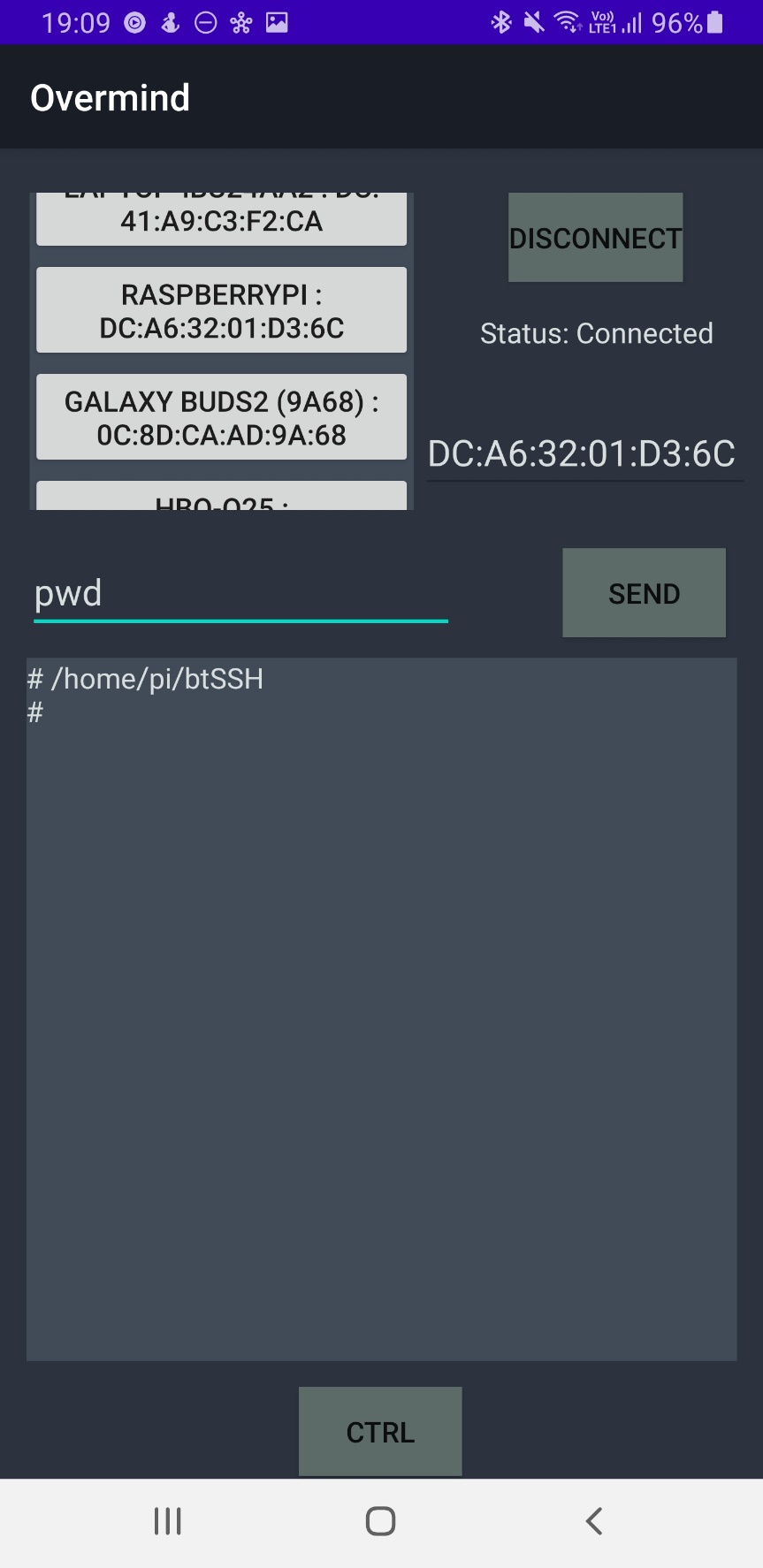
Din moment ce totul se petrece in timp real, se pot anula comenzile care se afla in curs de executie, in modul pus la dispozitie de sistemul de operare al server-ului (de cele mai multe ori, prin introducerea comenzii CTRL+C).

**Exemple:**

In exemplele urmatoare, conexiunea a fost realizata intre telefonul meu si o placa de dezvoltare Raspberry Pi ce ruleaza sistemul de operare Raspbian (Linux).

1. **Afisarea folderului de lucru**

Comanda: ***pwd***

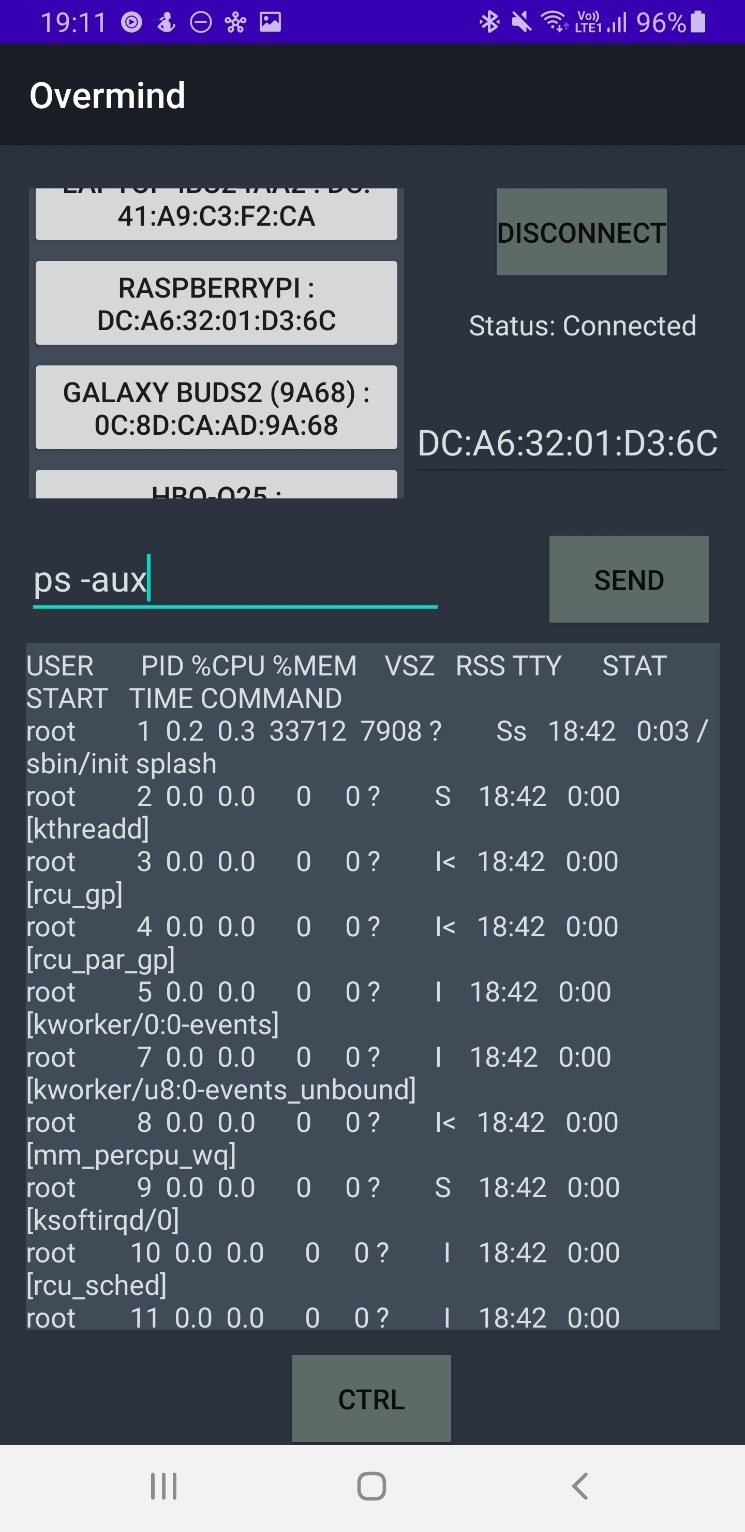
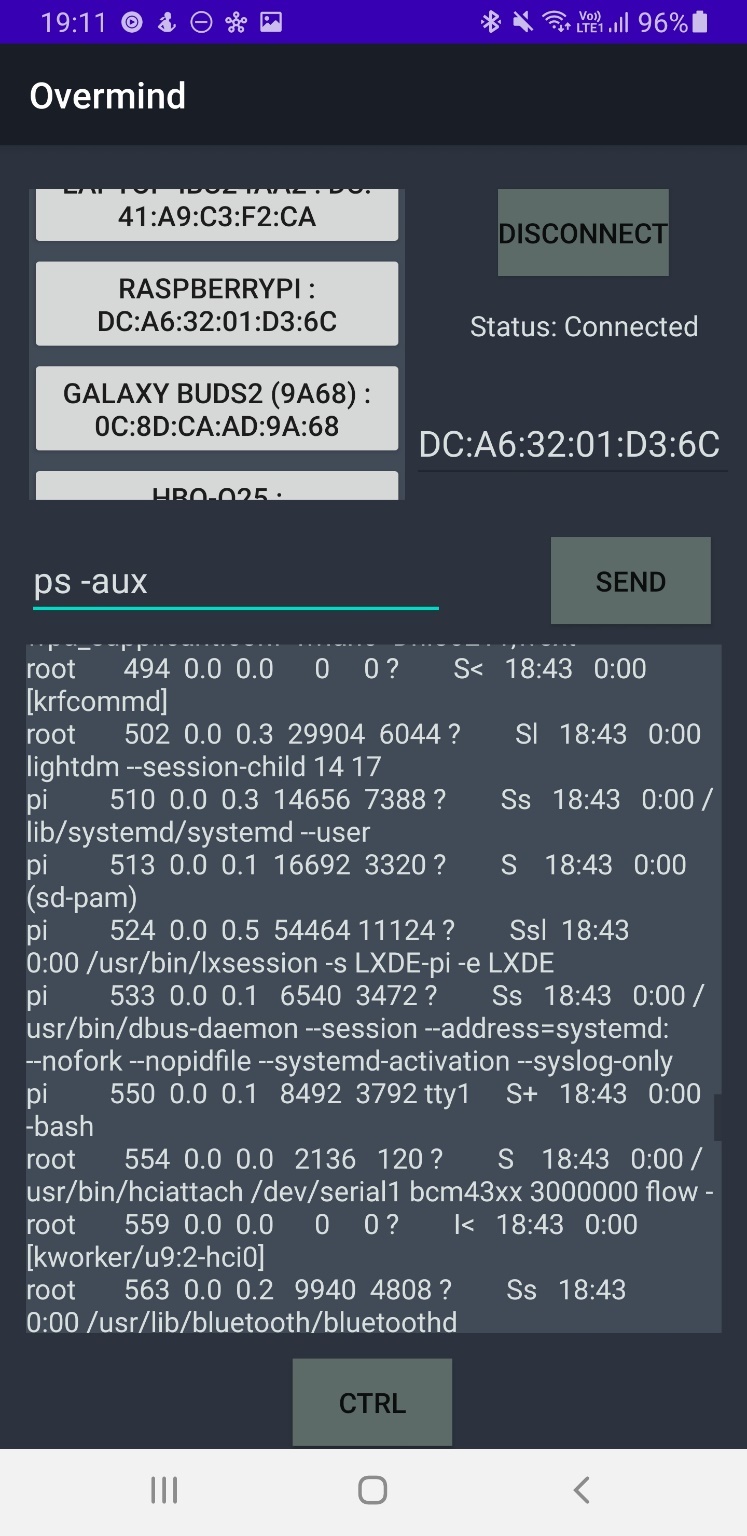


* Initial, folderul de lucru va fi chiar folderul in care se afla serverul.
* In cazul meu acesta este “/home/pi/btSSH”

1. **Revendicarea listei de procese**

Comanda : ***ps -aux***

Am ales aceasta comanda deoarece rezultatul ei nu incape integral in campul de “Output” fiind o oportunitate excelenta pentru demonstrarea capabilitatii de scroll.



Acelas rezultat in care am facut scroll, se poate observa o mica bara de scroll in dreapta campului

Ce vedem initial

1. **Folosirea caracterului %CTRL%**

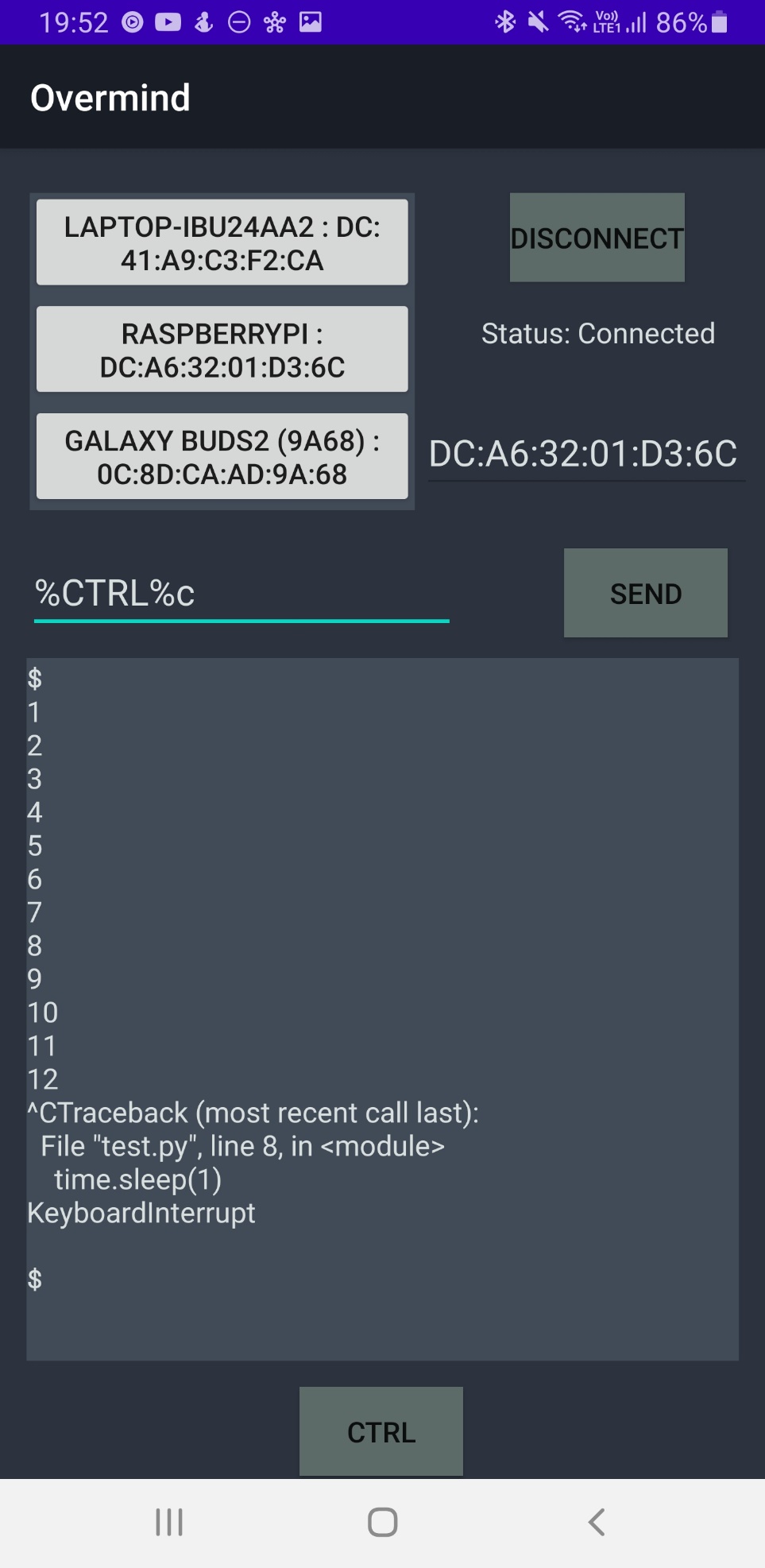
Pentru acest exemplu am creeat un script in python care numara la infinit secundele de cand a fost rulat.

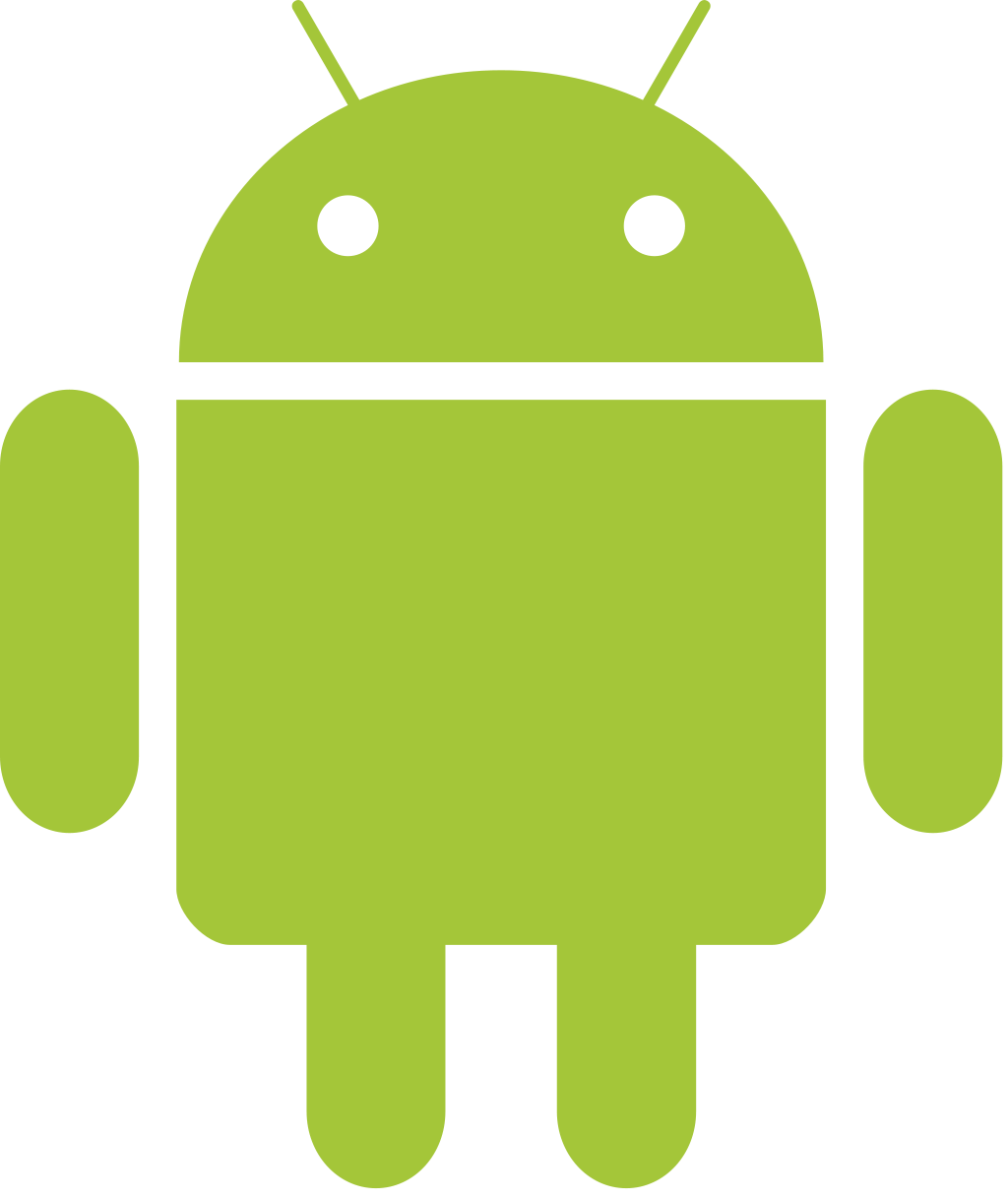
Cod (test.py): Rezultat:

import time

a = 1

while True:  
 print(a)  
 a += 1  
 time.sleep(1)





Mai intai am rulat scriptul prin comanda “python test.py” apoi am introdus “%CTRL%c” ce este echivalent cu (CTRL+C). In linia de comanda linux aceasta instructiune intrerupe procesul ce se afla in derulare.

**Capitolul IV**

**Partea tehnica a aplicatiei**

**Android Studio**

Majoritatea aplicatiilor pentru Android sunt dezvoltate cu ajutorul **Android Studio**, un IDE (integrated development environment) creeat de Google in colaborare cu JetBrains. Acesta contine toate uneltele necesare dezvoltarii de aplicatii pentru orice dispozitiv ce ruleaza Android.

Am ales acest **IDE** deoarece are cea mai buna documentatie dintre toate, fiind cel mai folosit, astfel putand gasi resurse cu usurinta pe internet.

Ca limbaj de programare, am folosit **Java** deoarece este mai “low-level” decat Kotlin. De asemenea, deja eram familiarizat cu acest limbaj din proiectele anterioare.

**Notiuni tehnice**

Inainte de a continua, trebuie sa ne familiarizam cu cateva dintre notiunile tehnice mai avansate folosite in acest proiect:

1. Buffer
2. Socket
3. Toast
4. Thread
5. Stdin, Stdout, Stderr

Acestea sunt fundamentale pentru intelegerea proiectului, fiind notiuni care nu sunt atat de cunoscute de toata lumea. Ele se refera la o multime de procedee si concepte abstracte folosite in dezvoltarea aplicatiilor.

1. **Buffer**

Un buffer este o zona din memorie care este folosita pentru a stoca date temporar, cat timp sunt mutate dintr-o parte in alta.

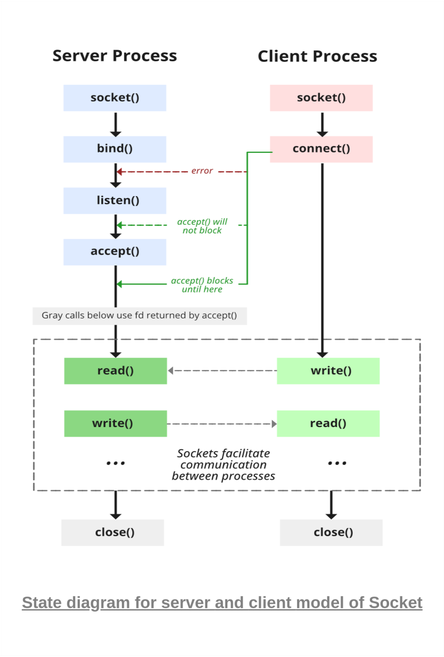
Acestea pot fi folosite pentru comunicarea dintre procese, dintre dispozitive (prin retea) sau pur si simplu pentru management-ul memoriei.

Cel mai cunoscut exemplu de folosirea a bufffer-ului este in streamingul online (Ex: Youtube, Spotify, Netflix, etc…), unde platforma nu incarca continul pe masura ce este redat de utilizator, ci incarca din avans toate datele ce urmeaza sa fie redate intr-un buffer.

1. **Socket**

Un **socket** este un punct terimnal din comunicarea dintre 2 programe ce se afla in aceeasi retea (nu neaparat in acelas computer). Pe scurt, este un mod prin care doua procese pot comunica intre ele. Fiecare **socket** are o adresa si un port, adresa fiind reprezentata de **IP-ul** computerului (in cazul bluetooth, de **MAC**) si portul find alocat automat de sistemul de operare sau manual de programator.

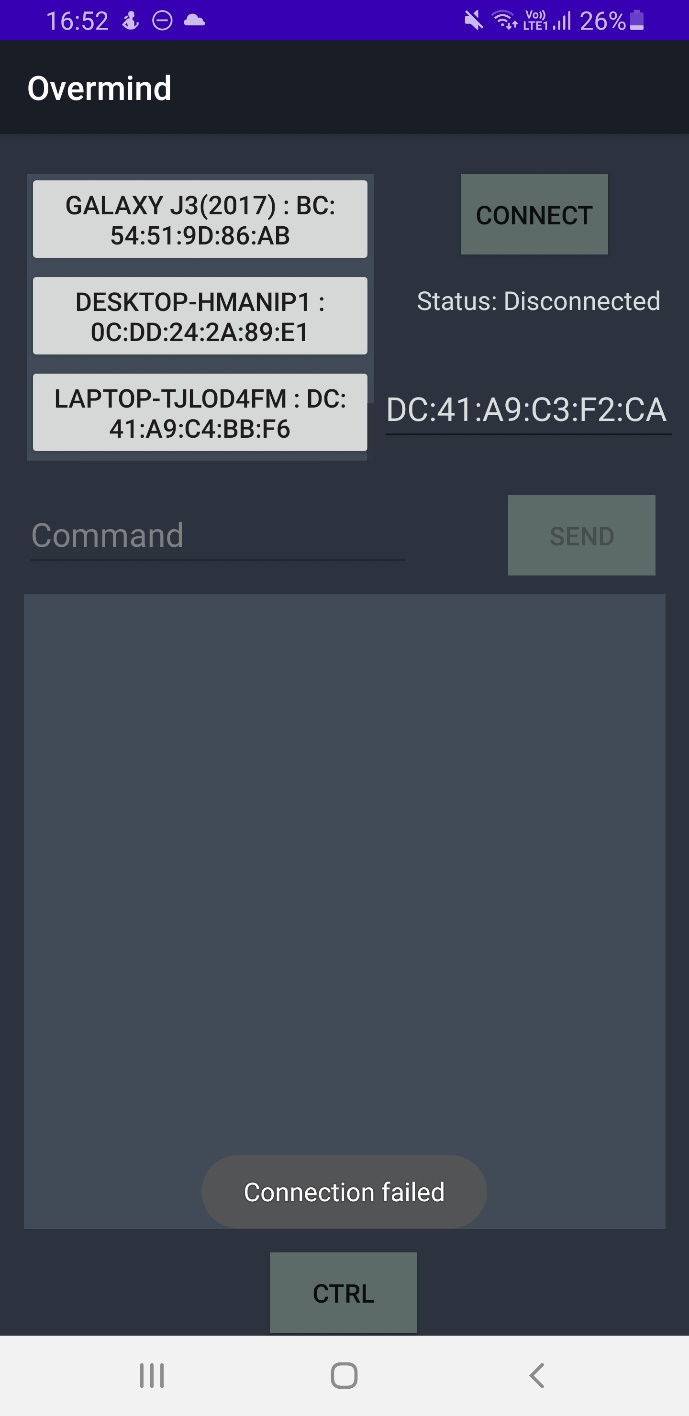
Sa spunem ca vrem sa programam un **server** pe computerul nostru. Mai intai trebuie sa creem un **socket** care va astepta o conexiune de la un alt computer (clientul). **Clientul** foloseste de asemenea tot un socket pentru a se conecta la server. O data conectate, doua **socket-uri** creeaza o legatura si pot comunica intre ele.

****Alaturi avem diagrama comunicarii intre server si clientul sau. Putem observa ca ambele parti au nevoie de un **socket** pentru a putea comunica.

Ambele socket-uri sunt initial la fel doar ca sunt folosite diferit de fiecare parte.

Se observa ca serverul trebuie mai intai sa creeze un **socket**, pe care il asociaza manual unui port (“**bind()**”) apoi incepe sa astepte conexuni de la potentiali clienti (“**listen()**”), iar in final, dupa ce clientul a cerut conectarea (“**connect()**”), serverul sa accepte conexiunea (“**accept()**”) pentru a putea incepe comunicarea dintre cele doua procese.

1. **Toast**

In programarea pe Android, **Toast** nu se refera la paine prajita ci la o unealta prin care o aplicatie poate transmite informatii utilizatorului prin intermediul unor ferestre de tip **pop-up** ce apar in partea de jos a ecranului si se sterg singure dupa un anumit timp.

In stanga este un exemplu de mesaj Toast din acest proiect.

Sageata rosie arata fereastra pop-up care apare atunci cand o conexiune nu poate fi efectuata.

Mesajele **Toast** sunt foarte utile deoarece pot transmite utilizatorului informatii fara a ocupa degeaba ecranul.

1. **Thread**

Procesorul unui computer executa instructiunile unui program intr-un mod **liniar**, de obicei nefiind in stare sa execute mai multe operatii deodata.

Un fir de executie (**thread**) este un set de instructiuni ce este rulat **secvential** de procesor. Orice program are un **Thread** principal (Ex: in C++ tot ce se executa in functia main se intampla pe threadul principal) unde se afla cele mai importante functii ale programului, fiind primul care se executa.

Tot din **thread**-ul principal, se pot creea si alte fire **secundare** de executie care ruleaza independent de cel **principal**, astfel programul putand face mai multe lucruri in acelas timp.

Folosirea a mai multor thread-uri intr-un singur proces (cum e si cazul Overmind) se numeste **Multithreading**.

1. **Stdin, Stdout, Stderr**

Standard input device (**stdin**) este un “dispozitiv” din care se citesc datele de intrare intr-un sistem. In cazul sistemelor de operare, acesta poate fi reprezentat de: tastatura, mouse, microfon, (etc…) .

Standard output device (**stdout**) si Standard error device (**stderr**) sunt ambele dispozitive in care sistemul de operare scrie date de iesire. Diferenta dintre ele este ca in **stderr** sunt scrise doar erorile care apar pe parcurs, iar in **stdout** se scriu restul datelor de iesire.

Cand ne referim la un **terminal** (consola), fiecare dintre stdin,stdout,stderr este reprezentat un **buffer propriu**.

**Alcatuirea proiectului**

Ca orice aplicatie de networking, Overmind este alcatuit din 2 parti: Server si Client.

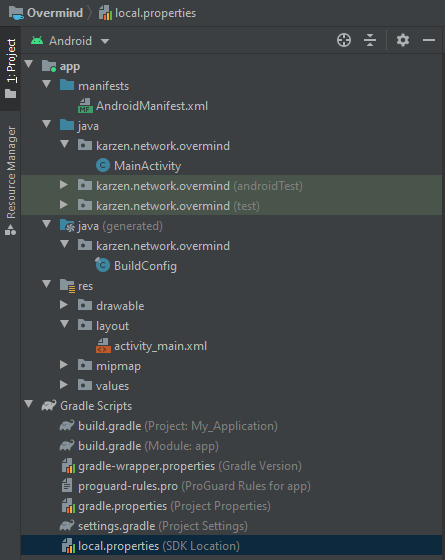
Clientul este reprezentat de aplicatia pentru Android, telefonul conectandu-se la dispozitiv si nu invers. Aceasta parte este cea mai complexa din tot proiectul si va fi discutata in amanunt in cele ce urmeaza.

Serverul se afla pe celalalt dispozitiv si asteapta conexiuni de la telefon. Acesta este un program extrem de simplu fiind scris in limbajul Python. Aceasta parte nu este asa de importanta iar toata atentia noastra se va axa pe client.

Tot ceea ce face serverul este ca dupa ce se conecteaza un client, acesta sa faca o legatura intre stdin-ul, stdout-ul si stderr-ul consolei cu cele ale socket-ului.

**Structura clientului**

Orice proiect in Android Studio, este alcatuit din 2 parti: folderul propriu-zis al aplicatiei numit “app” si folderul “Gradle Scripts”.



Folderul app este compus la randul lui din alte 4 foldere (manifest, java, java (generated) , res) care vor fi discutate in cele ce urmeaza

\*Nota: orice element care are o sageata in dreptul lui este un folder, iconita acestuia nu conteaza

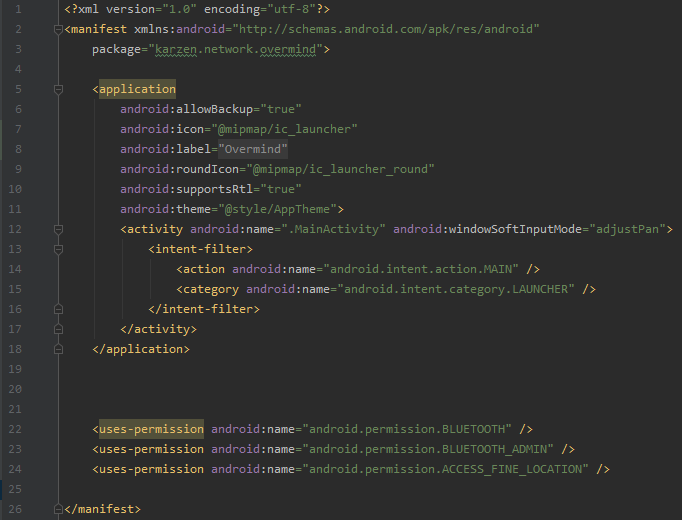
1. **Folderul Manifests**

Exista cu un singur scop: sa gazduiasca fisierul “AndroidManifest.xml”

**Ce este “AndroidManifest.xml” ?**

Numit pe scurt “manifest”, orice aplicatie de **Android** trebuie sa aiba neaparat un fisier ca acesta (numit exact in acest fel). Acest fisier contine informatii esentiale despre aplicatie ( numele, versiunea, uneltele folosite pentru dezvoltarea aplicatiei, permisiunile necesare aplicatiei, etc.. ) care sunt necesare pentru sistemul de operare si **Google Play**.

Asa cum poate reiese din nume, informatiile sunt stocate sub forma **XML** (extended markup language).



\*Un screenshot al AndroidManifest.xml

Fisierul AndroidManifest.xml este mereu generat automat la crearea unui nou proiect si este rareori modificat de programator. Principalul motiv pentru care este necesara modificarea acestui fisier, este adaugarea permisiilor necesare aplicatiei.

**Permisiuni**

Un detaliu interesant care se regaseste in manifest sunt cele 3 permisiuni necesare pentru rularea aplicatiei:

Fiind o aplicatie bazata pe Bluetooth, explicarea primelor doua permisiuni ar fi redundanta.

**Dar de ce are nevoie aplicatia de locatia exacta (“fina”) a dispozitivului ?**

Motivul este unul surprinzator, si este legat de politica de confidentialitate a Android. Si asta este deoarece dispozitivele bluetooth trebuie sa emita un “camp” ca sa poata comunica intre ele.

Problema este ca nimeni nu stie ce se poate afla in proximitatea acestor dispozitivie. De exemplu, pe strada (si nu numai) pot fi amplasate niste dispozitivie numite “Bluetooth Beacon” ce analizeaza “campurile bluetooth” din jurul lor si pot localiza cu exactitate un dispozitiv. Practic, aceste beacon-uri creeaza o harta exacta cu cine, unde si cand a trecut printr-o anumita zona. Deci aplicatia are nevoie de permisiune la locatia dispozitivului, nu pentru ca se foloseste de ea, ci pentru ca o poate dezvalui in mod neintentionat altora. Acesta este unul din riscurile mai putin stiute ale Bluetooth.

Totusi, acest fapt nu ar trebui sa ne sperie deoarece “Bluetooth Beacon-urile” functioneaza doar daca serviciul Bluetooth-ul este pornit pe telefon.

Aceste Beacon-uri au totusi si aplicatii mai “frumoase”. De exemplu, la gradina zoologica din Sibiu, sunt folosite “Beacon-uri Bluetooth” pentru ca serverele gradinii sa stie in ce zona te afli si sa iti trimita ulterior pe telefon (folosind o aplicatie descarcata in prealabil) informatii despre animalul la care te uiti.

1. **Folderul java**

Este alcatuit la randul lui din alte 3 foldere cu nume aproape identice. Cel care ne intereseaza este primul, unde se afla codul prorpiu-zis al aplicatiei.

Celelalte 2 sunt generate automat de Android Studio pentru testarea aplicatiei si a dispozitivului.

In interiorul primului folder gasim fisierul “MainActivity.java”, care in cazul Overmind, contine tot codul in java necesar pentru functionarea aplicatiei. In cazul unui proiect mai elaborat, codul se poate imparti in mai multe fisiere pentru a fi mai organizat.

Deoarece codul din acest fisier este mult prea mare pentru a fi introdus integral aici (346 linii de cod), vom discuta in continuare despre cele mai importante metode din codul sursa.

**“onCreate()”**

****

Este cea mai importanta metoda din aplicatie, fiind prima care se executa (Entrypoint-ul) si este asemanatoare ca si concept cu functia “main()” din C++. Ea este de asemenea constructorul clasei principale.

In aplicatia “Overmind”, metoda onCreate() face in aceasta ordine urmatoarele lucruri: initializarea interfetei, solicitarea permisiunilor si adaugarea dispozitivelor deja asociate in listbox.

De asemenea se observa ca nu exista un “loop principal” deoarece restul proceselor din aplicatie se executa pe thread-uri diferite de cel principal, in background sau la apasarea butoanelor. In acest mod aplicatia nu va “ingheta” in asteptarea raspunsului de pe celalalt dispozitiv.

**“initConnection()”**

Aceasta este metoda care se apeleaza cand apasam pe butonul de conectare (7) si tot ceea ce face este sa faca niste verificari si apeleaza la randul ei metoda “connectToDevice”

**“connectToDevice()”**

Aceasta foloseste o metoda mai veche din interiorul sistemului de operare android si anume “createInsecureRfcommSocket”. Acesta nu este modul standard prin care se creeaza socket-uri bluetooth, dar am ales sa utilizez aceasta metoda scoasa din folosinta deoarece imi doream ca si telefoanele mai vechi sa fie in stare sa ruleze aplicatia.

\*Nota: “createInsecureRfcommSocket” nu este definita in aplicatie ci direct in sistemul de operare. Aplicatia se foloseste de aceasta metoda chiar daca nu ii apartine.

Dupa ce initializeaza socket-ul, medoda “connectToDevice” incearca sa se conecteze la celalalt dispozitiv si anunta reusita sau esecul prin intermediul Toast. In cazul unei reusite, aceasta functie va modifica si continutul campului “Label status”(4 – vezi legenda).In final se creeaza un nou thread pe care va rula functia “receiver”.

**“receiver()”**

****

Desi din antet pare simpla, metoda “receiver” este defapt cea mai complexa deoarece se ocupa de o mare parte din trafic, si anume de datele care vin de la dispozitivul secundar spre telefon. Aceasta se foloseste de o tehnica de programare numita buffering, adica nu citeste toata informatia de odata ci o citeste in parti mai mici pe care le adauga ulterior la setul de date final, astfel citirea se face cu un buffer mic. (128 de octeti in cazul “Overmind”)

Am folosit aceasta tehnica din 2 motive:

* Deoarece noi nu stim cat de multe date urmeaza sa trimita celalalt dispozitiv, astfel putand primi date date de orice marime
* Evitarea unei vulnerabilitati a aplicatiei de tip buffer-overflow. Daca bufferul in care primim informatiile ar fi si bufferul final, si un atacator ar trimite date de o marime mai mare decat cea a buffer-ului, atunci memoria si-ar depasi spatiul alocat si ar “da pe afara” (de aici si denumirea de overflow) direct peste instructiunile ce urmeaza a fi executate. Astfel, un hacker ar putea rula cod malitios direct pe procesor, lucru ce ar fi, evident, catastrofal.

**“endConnection()”**

Asa cum ii spune si numele, aceasta metoda inchide conexiunea, dupa ce transmite celuilalt dispozitiv ca vrea sa incheie. In final, anunta utilizatorul cu Toast daca operatia a avut succes sau nu. Evident, metoda se ruleaza cand apasam pe butonul de disconnect.

**“sendCommand()”**



Metoda prelucraza si trimite informatiile pe care le introducem in “Command input” (6). Este de precizat ca aici se proceseaza si caracterul special “%CTRL%”.

Chiar daca nu este o informatie stiuta de multa lume, comenzile precum (CTRL+C) au si ele o reprezentare ASCII. Astfel (CTRL+C) devine un caracter ce poate fi transmis cu usurinta print-o retea.

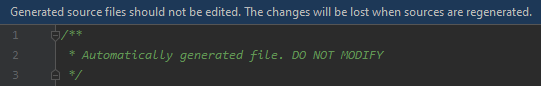
Codul ASCII asociat cu caracterul (CTRL+C) este 3, pentru ca “C” este a 3-a litera a alfabetului. Astfel (CTRL+A = 1, CTRL+B = 2, etc…)

Deci prelucrarea lui “%CTRL%” se face destul de usor deoarece trebuie doar sa aflam pe ce pozitie a alfabetului se afla cifra de dupa “%CTRL%” si sa tratam rezultatul ca pe un caracter.

1. **Folderul java(generated)**

Este generat automat de Android Studio pentru a fi folosit ulterior tot de el insusi. In acest folder se afla fisierul “BuildConfig.java” unde se afla informatii necesare pentru compilare.

Fisierul nu trebuie modificat niciodata de programator, acesta modificandu-se singur cand lucram la proiect.

Faptul ca acest fisier nu trebuie modificat ne este reamintit chiar de IDE care ne transmite un mesaj in legatura cu asta.

1. **Folderul res**

Contine toate felurile de date necesare aplicatiei. Acest folder nu contine cod, ci doar date ce se afla sub diferite forme pe care aplicatia le foloseste in timpul executarii.

Aceste date sunt impartite in 4 categorii:

* drawable
* layout
* mipmap
* values

1. In “drawable” se afla tot ceea ce poate fi randat pe ecran, in general imagini.
2. In “layout” se afla fisiere XML reprezinta interfata grafica. Aici exista un singur fisier numit “activity\_main.xml”.
3. In “mipmap” sunt toate iconitele de care se foloseste aplicatia
4. In values se afla date care specifica o anumita caracteristica a aplicatiei. De exemplu: culoarea unui buton, titlul, schemele de culori, etc…

**“activity\_main.xml”**

In acest fisier se regasesc toate instructiunile necesare sistemului Android, pentru a desena interfata grafica, cu care interactioneaza utilizatorul. Modul de functionare a acestor instructiuni este asemanator cu cel al HTML-ului.

Android Studio ne pune la dispozitie doua metode prin care putem edita acest fisier:

* Prin editarea efectiva a XML-ului
* Prin folosirea unui editor grafic ce converteste actiunile noastre in XML. Este cel mai folosit mod, fiind mai simplu ca editarea pura a fisierului, deoarece putem sa vedem in timp real elementele, sa le mutam si sa le schimbam marimea, avand totodata capabilitatea de “drag and drop”.

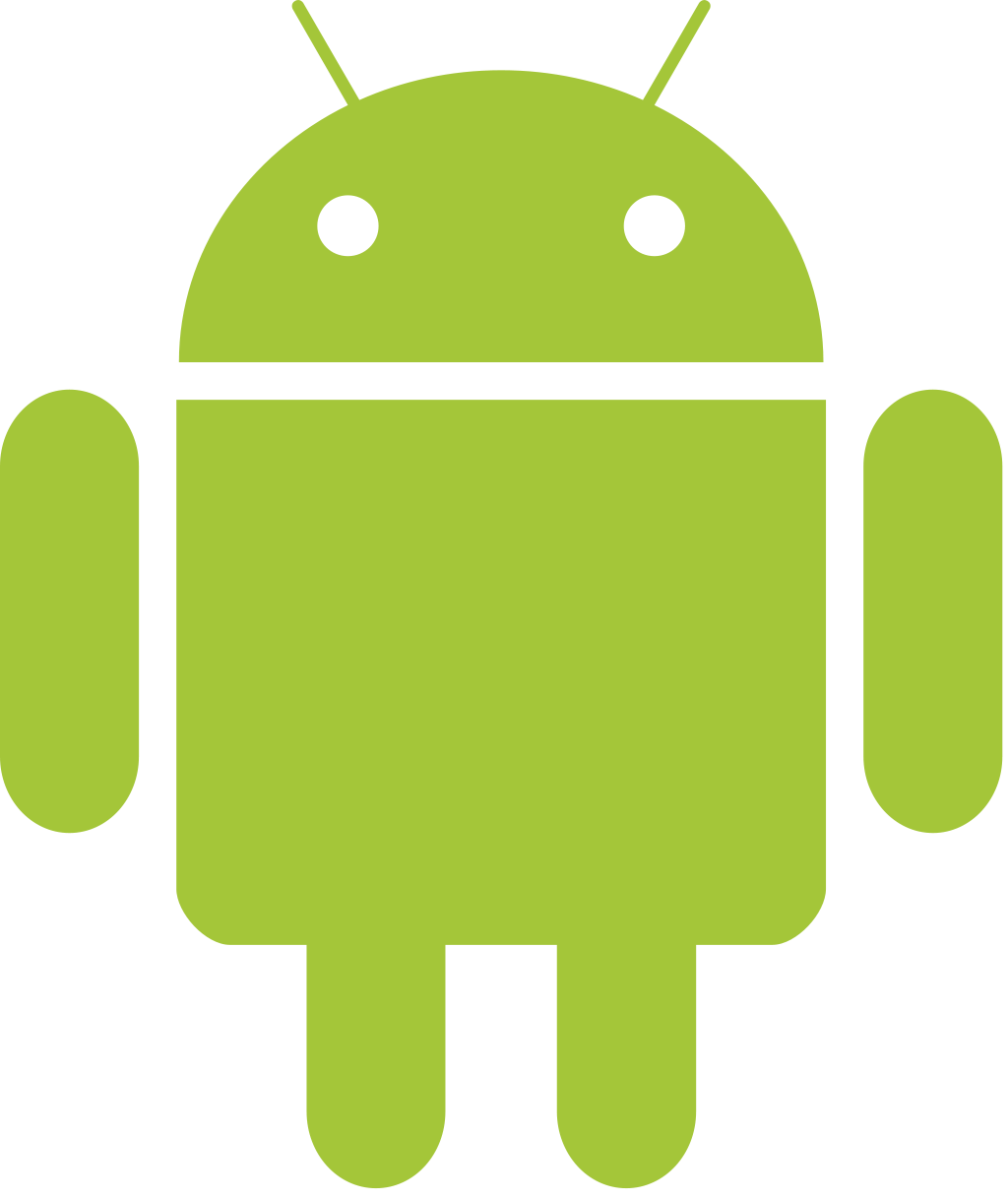
1. **Folderul “Gradle Scripts”**

Acest folder contine scripturi si informatii ce pot influenta procesul de compilare.

Android Studio foloseste o unealta numita Gradle pentru a se ocupa de compilarea unei aplicatii.

**Gradle** este un sistem automat ce se ocupa de testarea si compilarea unei aplicatii. Acesta se ocupa de toate folderele si fisierele proiectului, astfel ca programatorii sa poata avea un spatiu de lucru organizat.

**Gradle** este un sistem foarte customizabil, asa ca scripturile din “Gradle Scripts” pot fi modificate astfel incat inaintea compilarii **Gradle** sa faca tot felul de lucruri folositoare.

****

**Overmind**