

Uniwersytet Mikołaja Kopernika
Wydział Matematyki i Informatyki

Łukasz Ogan
nr albumu: 260196

Praca licencjacka
na kierunku informatyka

Dystrybucje Linuxa na systemie Android

Opiekun pracy dyplomowej
dr Andrzej Kurpiel

Toruń 2016

Pracę przyjmuję i akceptuję

Potwierdzam złożenie pracy dyplomowej

.....
data i podpis opiekuna pracy

.....
data i podpis pracownika dziekanatu

Spis treści

Wstęp	4
0.1 Załącznik	5
1 Dystrybucje Linuxa na system Android	6
1.1 Procesory - Architektura ARM	6
1.2 Emulacja	7
1.3 Emulacja Debiana przeznaczonego na architekturę ARM	10
1.4 Narzędzia linuksowe do tworzenia odseparowanego systemu plików	12
1.5 Uruchomienie dystrybucji Linuxa na telefonie z systemem Android	13
1.6 Komunikacja z wykorzystaniem protokołu VNC	14
1.7 Debian - historia	15
1.7.1 Pakiety	16
1.7.2 Instalacja pakietów	17
1.7.3 APT jako narzędzie do pobierania i instalacji pakietów	19
1.7.4 Debootstrap	20
1.7.5 Debootstrap - instalacja dystrybucji	21
2 Protokół SIP	26
2.1 Elementy sieci SIP	27
2.2 Działanie protokołu SIP	28
2.3 Implementacje protokołu SIP w systemie Linux - serwery	31
2.3.1 miniSipServer	31
2.3.2 Asterisk	32
2.3.3 Cipango	33
2.3.4 Elastix	33
2.3.5 FreeSWITCH	33
2.3.6 Kamailio	33
2.3.7 Inne serwery SIP	33
2.4 Implementacje protokołu SIP w systemie Android - serwery	34
2.4.1 uSipServer	34

2.5	Implementacje protokołu SIP w systemie Linux - klienci . . .	34
2.5.1	Blink	34
2.5.2	KPhone	34
2.5.3	Linphone	34
2.5.4	Zoiper	35
2.5.5	Inne klienci SIP	35
2.6	Implementacje protokołu SIP w systemie Android - klienci . .	35
2.6.1	Konfiguracja natywnego klienta SIP	35
2.6.2	Sipdroid	36
2.6.3	CSipSimple	36
2.6.4	Bria VoIP Softphone	37
2.7	Rynek telefonii SIP	38
3	Projekt informatyczny AndroidLinuxSIPService	40
3.1	Narzędzia programistyczne użyte do realizacji	41
3.1.1	Przygotowanie dystrybucji	41
3.1.2	Przygotowanie skryptu	41
3.1.3	Wybór serwera SIP	42
3.1.4	Przygotowanie aplikacji na system Android	42
3.2	Kod źródłowy aplikacji i jej działanie	44
3.2.1	Struktura klas aplikacji	44
3.2.2	Interfejs aplikacji	48
3.2.3	Interfejs webowy serwera miniSipSerwer	53
	Zakończenie	56
	Spis rysunków	58

Wstęp

Celem pracy dyplomowej jest omówienie zagadnień związanych z dystrybucjami systemu Linux, które można uruchomić na urządzeniach mobilnych opartych o architekturę ARM. Praktyczna część pracy skupia się na wykorzystaniu dystrybucji Linuxa uruchomionej obok systemu Android, która ma posłużyć jako serwer telefonii SIP. Część praktyczna składa się z następujących czynności:

- Przygotowanie dystrybucji Debiana na architekturę ARM z wykorzystaniem narzędzia debootstrap.
- Utworzenie pliku `.img` z przygotowaną dystrybucją.
- Zainstalowanie w dystrybucji pakietu `miniSipServer`.
- Przygotowanie skryptów startowych.
- Przygotowanie aplikacji na urządzenie mobilne służącej do zarządzania systemem oraz pakietem `miniSipServer`.

Rozdział pierwszy pracy "Dystrybucje Linuxa na system Android", zawiera omówienie architektury ARM. Opisane zostało zagadnienie emulacji systemu. Zśród popularnych emulatorów wybrany został emulator QEMU, na którym to została pokazana przykładowa emulacja systemu Debian przeznaczonego na architekturę ARM. W dalszych częściach rozdziału opisywane są metody tworzenia odseparowanego systemu plików. Opisany został program na urządzenie mobilne który realizuje uruchomienie wybranej dystrybucji Linuxa. Komunikacja z uruchomionym systemem w tle może odbywać się poprzez protokół VNC, o którym mowa jest w podrozdziale 1.6. Druga część rozdziału skupia się na dystrybucji Debian. Omówiony został zarys historyczny oraz podstawowe narzędzia administracyjne. Rozdział kończy się opisem narzędzia `debootstrap`, które służy do przygotowania wybranej dystrybucji Debiana w katalogu uruchomionego już systemu.

Rozdział drugi "Protokół SIP", skupia się na omówieniu zasady działania protokołu SIP. Wprowadzone zostają podstawowe pojęcia opisujące protokół oraz wyróżnione zostają jego elementy. W kolejnych rozdziałach opisano dokładne działanie protokołu z uwzględnieniem zrzutu danych protokołu podczas działania. Podrozdział 2.3 przedstawia listę implementacji protokołu SIP w systemie Linux oraz Android z podziałem na oprogramowanie serwerowe i kliencie. W nawiązaniu do pakietu miniSipServer została pokazana jego instalacja. Koniec rozdziału opisuje rynek telefonii SIP

Rozdział trzeci "Projekt informatyczny AndroidLinuxSIPService", opisuje realizację praktyczną pracy, której celem jest uruchomienie serwera SIP miniSipServer na systemie Debian Wheezy 7.0 obok systemu Android oraz przygotowanie aplikacji na system Android do zarządzania dystrybucją i serwerem SIP. Opisane zostały etapy tworzenia projektu oraz jego możliwości.

0.1 Załącznik

Do pracy dołączono DVD-ROM, na którym znajduje się

- Źródła pracy w języku LaTeX,
- Plik PDF pracy,
- Aplikacja w formacie .apk,
- Źródła aplikacji,
- Obraz dystrybucji,
- Skrypt startowy.

Rozdział 1

Dystrybucje Linuxa na system Android

1.1 Procesory - Architektura ARM

Procesor jest to cyfrowe urządzenie sekwencyjne, które występuje w każdym złożonym układzie elektronicznym. Istnieje podział na ze względu na architekturę, czyli wspólne określenie najważniejszych z punktu widzenia budowy i funkcjonalności cech procesora (zbiór reguł). Powszechnie w komputerach stosuje się procesory produkowane przez firmę Intel lub AMD odpowiednio 32 lub 64 bitowe. Jeżeli chodzi o urządzenia mobilne lub komputery zminiaturyzowane to stosuje się procesory oparte na architekturze ARM (Advanced RISC Machine) również odpowiednio 32 lub 64 bitowe. Procesory o architekturze ARM należą do rodziny RISC, są szeroko bardzo często stosowane w systemach wbudowanych ze względu na swoją energooszczędną architekturę. Oprócz telefonów komórkowych procesorów ARM używa się między innymi w dyskach twardych, routerach, kalkulatorach czy odtwarzaczach iPod. Aby móc zainstalować system operacyjny na urządzeniu, które jest wyposażone w procesor ARM, musimy posiadać system który jest przeznaczony na tą architekturę. Przykładem takiego systemu jest Android, który obecnie dominuje na rynku urządzeń mobilnych. System Linux możemy zainstalować na procesorach ARM które posiadają jednostkę MMU (Memory Management Unit)¹. Na procesory, które nie mają jednostki MMU istnieje projekt uClinux (czyt. "you see linux").

¹Marcin Bis - Linux w systemach Embedded, układy pełniące realizację dostępu do pamięci fizycznej żądanej przez CPU

1.2 Emulacja

Mamy do dyspozycji jako system operacyjny Windows lub Linux oparty na architekturze Intel lub AMD i uruchomimy na nim poprzez emulację inny system operacyjny, który docelowo jest przeznaczony na architekturę ARM.

Scenariusz:

- host: Windows 8 64-bit ²
- gość: Debian 32-bit ARM

Do takiej operacji służy nam wcześniej wspomniana emulacja czyli programowe symulowanie działania określonego oprogramowania lub platformy sprzętowej przez inny system lub na sprzęcie innego typu. Dostępnych jest wiele projektów mających na celu zapewnienie wsparcia dla emulacji konkretnych urządzeń.

Lista popularnych emulatorów:

- VirtualBox
- VMWare
- KVM (Kernel-based Virtual Machine)
- QEMU ³
- Softgun
- SkyEye
- ARMWare

Jednym z popularnych emulatorów jest QEMU <http://wiki.qemu.org/>. Jest to szybki emulator, który na bieżąco tłumaczy wykonywany kod maszynowy. Potrafi emulować procesory x86, ppc, arm, sparc, mips.

²Słowa, adresy, jednym słowem mówiąc dane mieszczą się w najwyżej 64 bitach pamięci

³QEMU - emulator napisany przez Fabrice Bellarda. Jest Bardzo szybki przez co wyprzedza konkurencję. Niestety jest trudniejszy w obsłudze niż inneulatory.

QEMU pracuje w dwóch trybach:

1. Emulacja pełnego systemu - procesora i peryferii

```
$ qemu-system-arm -M versatilepb -kernel zImage.bin
```

2. Emulacja interfejsu jądra dla przestrzeni użytkownika - umożliwia uruchamianie pojedynczych programów w systemie o innej architekturze. Przykład: uruchomienie w systemie o architekturze x86 z zainstalowanym QEMU Busyboxa skompilowanego dla architektury ARM:

```
$ qemu-arm -L /cross-tools \
```

Do uruchomienia konkretnej aplikacji potrzebna jest biblioteka standardowa w wersji dla architektury docelowej oraz wszystkie dodatkowe biblioteki, których może wymagać aplikacja. Ścieżkę do bibliotek podajemy po parametrze -L.

Instalacja

```
$ apt-get install qemu-system
```

Lista obsługiwanych platform (urządzeń)

```
$ qemu-system-arm -M ?
```

Analogicznie, do wyświetlenia listy procesorów służy polecenie:

```
$ qemu-system-arm -cpu ?
```

Dla systemów bazujących na architekturze ARM9 (armv5) dobrym wyborem będzie emulowanie przez QEMU urządzenie ARM Versatile.

W celu uruchomienia przygotowanego w emulatorze systemu na urządzeniu należy:

1. Skompilować wersję jądra z opcjami i sterownikami dla konkretnego urządzenia.
2. Przygotować bootloader dla konkretnego systemu.
3. Przenieść bootloader, jądro oraz system plików na urządzenie, na docelowy typ pamięci.

Uruchomienie emulatora:

```
$ qemu-system-arm -M <platform> -m <ilosc pamieci  
operacyjnej> <opcje uruchamiania>
```

Domyślna ilość pamięci operacyjnej to 128MB. Opcje uruchamiania mają za zadanie:

- Wskazać jądro systemu operacyjnego.
- Ustawić parametry konsoli oraz przekazać parametry do uruchamianego jądra.

Wszystkie opcje, wraz z przykładami:

```
$ qemu-system-arm -help
```

1.3 Emulacja Debiana przeznaczonego na architekturę ARM

Potrzebne pliki znajdują się pod adresem: <https://people.debian.org/~aurel32/qemu/armel/>

Uruchomienie:

```
$ qemu-system-arm -M versatilepb -kernel vmlinuz
  -2.6.32-5-versatile -initrd \
initrd.img-2.6.32-5-versatile -hda
  debian_squeeze_armel_standard.qcow2 \
-append "root=/dev/sda1"
$ qemu-system-arm -M versatilepb -kernel vmlinuz
  -2.6.32-5-versatile -initrd \
initrd.img-2.6.32-5-versatile -hda
  debian_squeeze_armel_desktop.qcow2 \
-append "root=/dev/sda1"
```

Sterownik loop w systemie Linux udostępnia urządzenia blokowe `/dev/loopX`⁴. Sterownik loop umożliwia podłączanie do struktury katalogów wartości obrazów płyt oraz dysków twardych. Możemy również przygotować obraz systemu.

```
mount -o loop plik.iso /mnt
```

Przełącznik `-t` pozwala nam na wskazanie systemu plików.

```
mount -t <system plikow> plik.iso
```

Plik ISO zostaje wtedy podłączony do pierwszego wolnego urządzenia `/dev/loopX`, które jest montowane na katalogu `/mnt`.

W przypadku gdy dysponujemy obrazem całego dysku lub karty SD z partycjami musimy skorzystać z polecenia `losetup`⁵ ponieważ polecenie `mount` spodziewa się pliku w którym struktury systemu plików zaczynają się od początku.

```
losetup -o 21232 /dev/loop obraz.img
mount /dev/loop0 /mnt
```

⁴Istnieje w systemie plików jako `/dev/loop` (`/dev/loop0`, `/dev/loop1`, ...). Sterownik, którego zadaniem jest udawanie urządzeń blokowych (np. dysk twardy), wykorzystuje do tego zwykły plik

⁵Polecenie `losetup` służy do kojarzenia urządzeń `loop` z plikami lub urządzeniami blokowymi

1.3. EMULACJA DEBIANA PRZEZNACZONEGO NA ARCHITEKTURĘ ARM11

Parametr -o w tym przypadku oznacza adres w bajtach, od którego zaczynają się struktury systemu plików. Adresy kolejnych partycji można odczytać poleceniem fdisk.

1.4 Narzędzia linuksowe do tworzenia odseparowanego systemu plików

System Linux oferuje kilka narzędzi w postaci komend, za pomocą których jesteśmy w stanie stworzyć odseparowany system plików.

`chroot` (ang. change root) - polecenie, które służy do zmiany głównego katalogu (/). Po zmianie głównego katalogu, w nowym środowisku będą działać procesy potomne. Komenda może być użyta jako narzędzie do ochrony przed atakami, atakujący będzie miał dostęp tylko do wydzielonego fragmentu systemu. Sprawdza się również podczas testowania programów, ponieważ ewentualne szkody nie będą dotyczyły głównego systemu. Polecenia może używać tylko superużytkownik `root`. Nie jest to jednak metoda, która mogłaby się sprawdzić na serwerach dużych korporacji - w takim przypadku korzysta się z technik wirtualizacyjnych.

Składnia polecenia⁶:

```
chroot [opcje] katalog [argumenty]

--userspec=USER:GROUP  - użytkownik i grupa do
    użycia

--groups=G_LIST        - lista grup

--help - pomoc

--version - wersja polecenia
```

Drugim parametrem jest ścieżka do shella, nie trzeba go używać gdyż domyślnie jest to `/bin/sh`.

Przykład:

```
chroot /usr/local/fedora/rootfs
```

Inną komendą, która działa podobnie jest `pivot_root`. Jest jedna jedna różnica, która polega na tym że po wykonaniu polecenia `pivot_root` mamy dostęp do starego systemu plików, który możemy umieścić w wybranym przez nas katalogu.

Składnia polecenia:

```
pivot_root new_root put_old
```

Przykład:

⁶ Manual - <http://linux.die.net/man/1/chroot>

```
cd /new-root

mkdir old-root

pivot_root . old-root
```

1.5 Uruchomienie dystrybucji Linuxa na telefonie z systemem Android

Sklep Google Play oferuje aplikacje, za pomocą których możemy uruchomić wybraną dystrybucję systemu Linux

- **Linux Deploy**⁷ - Debian, Ubuntu, Arch Linux, Fedora, OpenSUSE, KaliLinux, Gentoo, RootFS (instalacja jako plik lub partycja).
 - Wspierane systemy plików: ext2, ext3, ext4
 - Obsługiwane architektury: ARM, x86. Obsługa: SSH, VNC, X, framebuffer. Środowiska graficzne: XTerm, LXDE, Xfce, GNOME, KDE.
 - Wspierane języki: Angielski, Rosyjski.

Dostęp do kodu źródłowego: <https://github.com/meefik/linuxdeploy>. Aplikacja napisana jest w języku Java, wykorzystuje skrypty napisane w Bashu (w tym również polecenie chroot).

Zadaniem programu jest stworzenie obrazu IMG na karcie pamięci, zamontowanie go oraz zainstalowanie na nim wybranej dystrybucji. Do poprawnego działania aplikacja potrzebuje dostęp do root oraz internetu.

Szacowane czasy instalacji systemu Debian wheezy/armhf na telefonie Samsung Galaxy S II:

- Without GUI 0:12 / 260 MB
- XTerm 0:14 / 290 MB
- LXDE 0:19 / 450 MB
- XFCE 0:20 / 495 MB

⁷<http://github.com/meefik/linuxdeploy>

- GNOME 0:55 / 1.3 GB
- KDE 1:20 / 1.3 GB

Prędkości zapisu i odczytu dla określonych systemów plików:

- vfat: prędkość odczytu 14.1 MB/s; prędkość zapisu 12.0 MB/s
 - ext2: prędkość odczytu 14.9 MB/s; prędkość zapisu 3.9 MB/s
 - ext4: prędkość odczytu 14.9 MB/s; prędkość zapisu 16.6 MB/s
 - ext2 (loop): prędkość odczytu 17.0 MB/s; prędkość zapisu 7.4 MB/s
 - ext4 (loop): prędkość odczytu 17.2 MB/s; prędkość zapisu 8.8 MB/s
- **Complete Linux Installer**⁸ - Ubuntu 13.04, 13.10, Debian 5 and 8, Kali Linux, Fedora 19, Arch Linux, boot widget.
 - **DebianDroid**⁹ - wykorzystuje pakiet debootstrap do przygotowania dystrybucji Debiana na procesor ARM.

1.6 Komunikacja z wykorzystaniem protokołu VNC

Po zainstalowaniu aplikacji Linux Deploy, pobraniu wybranej dystrybucji i jej zainstalowaniu jesteśmy w stanie połączyć się z zainstalowaną dystrybucją za pomocą protokołu VNC.

VNC (ang. Virtual Network Computing) jest systemem przekazywania obrazu z wirtualnego lub fizycznego środowiska graficznego. Protokół VNC działa w architekturze klient-serwer. Pakiet jest dostępny na najpopularniejsze systemy operacyjne: Android, Windows, Linux, BSD, Mac OS, Solaris, AmigaOS, SCO, Haiku i inne. Konkurentem dla VNC jest NX, który działa z większą wydajnością.

Domyślnie VNC korzysta z portów TCP 5900 - 5906, gdzie każdy z portów oznacza odrębną sesję (:0 do :6). Istnieje możliwość konfiguracji programu klienta i serwera do pracy na innych portach.

Popularnym programem, który implementuje protokół VNC jest wieloplatformowy RealVNC opracowany przez giganta w świecie telekomunikacji AT&T Laboratories, który jest dostępny na stronie <http://realvnc.com/>

⁸<http://linuxonandroid.org/complete-linux-installer/>

⁹<https://github.com/uberspot/DebianDroid>

Oprogramowanie składa się z części klienta i serwera. Po zainstalowaniu pakietu jesteśmy w stanie połączyć się z serwerem VNC, który jest uruchomiony na innej maszynie. Mamy ku temu dwie możliwości:

- Poprzez GUI programu VNC Viewer,
- Za pomocą przeglądarki internetowej - podając odpowiedni adres i numer portu, uruchamiany jest wtedy aplet napisany w języku Java, który umożliwia nam sterowanie.

Inne implementacje VNC:

- Tight VNC - darmowa implementacja,
- Apple Remote Desktop - oprogramowanie firmy Apple przeznaczone dla systemów Mac OS X,
- Ultra VNC - darmowa implementacja, która umożliwia szyfrowanie połączenia,
- x11vnc - serwer VNC pozwalający na kontrolowanie zwykłej sesji X11,
- TridiaVNC - zmodyfikowana wersja RealVNC.

1.7 Debian - historia

Autorem systemu Debian jest Ian Murdock¹⁰, niemiecki programista. Powstanie systemu datuje się na 16 sierpnia 1993r. kiedy to na grupie com.os.linux.development Ian Murdock opublikował Manifest Debiana, który definiował stworzenie otwartej dystrybucji opartej na systemie Linux i licencji GNU. Dystrybucja została zbudowana na postawie SLS. Softlanding Linux System (SLS) to jedna z pierwszych dystrybucji Linixa. Została stworzona w 1992 roku przez Petera McDonalda. Była to pierwsza dystrybucja zawierająca nie tylko jądro, ale i również dodatkowe programowanie. Początkowo Debian rozwijał się wolno, wersja 1.x została wydana w 1996. W późniejszych latach system zaczyna rozwijać się bardziej dynamicznie. W 2002 została wydana wersja 3.0 "Woody", która działa na 11 różnych architekturach sprzętowych. 4 maja 2013 została wydana póki co najnowsza wersja stabilna Debiana 7.0 - Wheezy. Wersja 7.0 umożliwia obsługę wielu architektur jednocześnie (multiarch), pozwala to na instalację pakietów z wielu architektur na tej samej jednostce oraz uruchamianie 32 i 64 bitowych programów.

¹⁰Zmarł 28 grudnia 2015

Gałęzie dystrybucji

Debian jest rozwijany w 3 równoległych gałęziach:

- **stablina** (stable)
- **testowa** (testing)
- **niestablina** (eksperymentalna)

1.7.1 Pakiety

Pakiet to skompilowane źródła danego programu gotowe do instalacji. Pakiety dla Debiana i dystrybucji bazujących na Debianie, mają rozszerzenie **.deb**¹¹. Typ MIME pliku .deb to application/vnd.debian.binary-package. Pakiet .deb występuje wraz z instalatorem dpkg, który zapewnia zaawansowaną kontrolę powiązań i zależności pomiędzy poszczególnymi składnikami systemu (programy, biblioteki). Nazwy plików pakietów binarnych systemu Debian są podporządkowane następującej konwencji: *foo_VersionNumber-DebianRevisionNumber.deb*

Pakiet **.deb** to archiwum **ar** z trzema plikami:

- plik debian-binary, w którym zawarta jest informacja o wersji pakietu,
- plik kontrolny control.tar.gz - zawiera zależności pakietu, sumy kontrolne oraz skrypty instalacyjne,
- plik danych data.tar.gz - pliki konfiguracyjne, binarne, biblioteki, dokumentacja.

W pakietach .deb stosuje się również silne kompresujące formaty jak bzip2, lzma, xz.

Wypakowanie pakietu .deb

```
root@debian:~# ar xv sudo_1.8.5p2-1+nm2_1386.deb
x - debian-binary
x - control.tar.gz
x - data.tar.gz
root@debian:~#
```

¹¹Od wersji 0.93 systemu pliki deb są implementowane jako archiwum ar

Podgląd pliku control.tar.gz

```
root@debian:~# tar -tzvf control.tar.gz
drwxr-xr-x root/root          0 2015-02-16 12:08 ./
-rw-r--r-- root/root       4169 2015-02-16 12:08 ./
    md5sums
-rw-r--r-- root/root        716 2015-02-16 12:08 ./
    control
-rw-r--r-- root/root         68 2015-02-16 12:08 ./
    conffiles
-rwxr-xr-x root/root        453 2015-02-16 12:08 ./
    postrm
-rwxr-xr-x root/root       1380 2015-02-16 12:08 ./
    prerm
-rwxr-xr-x root/root        441 2015-02-16 12:08 ./
    preinst
-rwxr-xr-x root/root       1971 2015-02-16 12:08 ./
    postinst
root@debian:~#
```

Wypakowanie tylko pliku data.tar.gz

```
root@debian:~# ar p sudo_1.8.5p2-1+nmu2_i386.deb
    data.tar.gz | tar zx
```

1.7.2 Instalacja pakietów

Do instalacji pakietów **.deb** służy wcześniej wspomniane narzędzie **dpkg**.

Informacje na temat pakietu

```
root@debian:~# dpkg --info sudo_1.8.5p2-1+nmu2_i386.
deb
new debian package, version 2.0.
size 836432 bytes: control archive=4352 bytes.
68 bytes,      4 lines      conffiles
716 bytes,    17 lines      control
4169 bytes,   60 lines      md5sums
1971 bytes,   75 lines *    postinst          #
    !/bin/sh
453 bytes,    28 lines *    postrm           #!/
    bin/sh
```

```

441 bytes,      20 lines    *   preinst          #!/
    bin/sh
1380 bytes,     50 lines    *   prerm           #
    !/bin/sh
Package: sudo
Version: 1.8.5p2-1+nmu2
Architecture: i386
Maintainer: Bdale Garbee <bdale@gag.com>
Installed-Size: 1586
Depends: libc6 (>= 2.11), libpam0g (>= 0.99.7.1),
        libselinux1 (>= 1.32), libpam-modules
Conflicts: sudo-ldap
Replaces: sudo-ldap
Section: admin
Priority: optional
Description: Provide limited super user privileges
            to specific users
Sudo is a program designed to allow a sysadmin to
            give limited root
            privileges to users and log root activity. The
            basic philosophy is to give
            as few privileges as possible but still allow people
            to get their work done.
.
This version is built with minimal shared library
            dependencies, use the
            sudo-ldap package instead if you need LDAP support
            for sudoers.
root@debian:~#

```

Instalacja pakietu

```

root@debian:~# root@debian:~# dpkg -i sudo_1.8.5p2-
    -1+nmu2_i386.deb

```

Usunięcie pakietu

```

root@debian:~# root@debian:~# dpkg -r nazwa_pakietu

```

Lista zainstalowanych pakietów

```

root@debian:~# root@debian:~# dpkg -l

```

Rekonfiguracja pakietu

```
root@debian:~# dpkg --configure -a sudo_1.8.5p2-1+  
nmu2_i386.deb
```

Narzędzia deweloperskie

- dpkg-source pakuje i rozpakowuje pliki źródłowe pakietu,
- dpkg-deb pakuje i rozpakowuje pakiety binarne,
- dpkg-gencontrol generuje na podstawie informacji zawartych w plikach źródłowych pakietu, plik control dla pakietu binarnego,
- dpkg-shlibdeps obliczają zależności od bibliotek,
- dpkg-genchanges czyta drzewo katalogów źródłowych po zbudowaniu pakietu i generuje na tej podstawie plik kontrolny (.changes),
- dpkg-buildpackage to skrypt pozwalający na automatyczne zbudowanie pakietu,
- dpkg-distaddfile dodaje plik do debian/files,
- dpkg-parsechangelog czyta plik z zapisem zmian (changelog) rozpakowanego pakietu źródłowego i tworzy opis zmian.

1.7.3 APT jako narzędzie do pobierania i instalacji pakietów

Narzędzie apt-get (część APT) służy to pobierania i instalacji pakietów z uwzględnieniem zależności w bardzo prosty sposób. To przeglądania pakietów i operacji na nich służy polecenie aptitude, jest bardziej zaawansowanym narzędziem od apt-get.

Instalacja pakietu

```
root@debian:~# apt-get install nazwa_pakietu
```

Szukanie pakietu

```
root@debian:~# apt-cache search nazwa_pakietu
```

Update pakietów

```
root@debian:~# apt-get update
```

Miejsca skąd można pobierać pakiety to repozytoria, które dzielimy na oficjalne i nieoficjalne. Lista repozytoriów znajduje się w pliku `/etc/apt/sources.list`

Przykładowy wpis z pliku `/etc/apt/sources.list`

```
deb-src http://ftp.pl.debian.org/debian/ etch main
non-free contrib
```

`deb-src` informują nas o typie archiwum, w tym przypadku są to źródła pakietów, które można ściągnąć i skompilować. Dalszą część wpisu stanowi adres repozytorium oraz nazwa dystrybucji. Po nazwie dystrybucji jest sekcja.

W Debianie wyróżniamy trzy sekcje:

- `main` - w niej znajdują się wszystkie wolne pakiety,
- `non-free` - tutaj znajdują się wszystkie niewolne pakiety, czyli takie, które mają zamknięty kod źródłowy,
- `contrib` - tutaj znajdują się wszystkie pakiety wolne, które zależą od pakietów niewolnych.

1.7.4 Debootstrap

Pakiet **debootstrap**¹² jest narzędziem, które umożliwia przygotowanie wybranej dystrybucji Debiana w podkatalogu uruchomionego już systemu. Proces nie wymaga użycia płyty CD, a jedynie dostęp do domyślnego repozytorium Debiana.

Syntaktyka polecenia `debootstrap`

```
debootstrap [OPTION]... <suite> <target> [<mirror>
[<script>]]
```

Przykładowy scenariusz z wykorzystaniem pakietu `debootstrap`:

Nowy katalog dla Debiana:

```
root@debian:~# mkdir /debian
```

Instalacja pakietu `debootstrap` z repozytorium:

```
root@debian:~# apt-get install debootstrap
```

Instalacja pakietu `debootstrap` z pliku `.deb`¹³

¹²<http://linux.die.net/man/8/debootstrap>, <https://www.debian.org/releases/stable/amd64/apds03.html.en>

¹³<http://ftp.debian.org/debian/pool/main/d/debootstrap/>

```
# ar -x debootstrap_0.X.X_all.deb
# cd /
# zcat data.tar.gz | tar xv
```

1.7.5 Debootstrap - instalacja dystrybucji

Instalacje należy rozpocząć od utworzenia katalogu /wheezy-chroot

```
export MY_CHROOT=/wheezy-chroot
cd /
mkdir $MY_CHROOT
debootstrap --arch i386 wheezy $MY_CHROOT http://ftp
.pl.debian.org/debian
```

Mamy do wyboru następujące architektury: alpha, amd64, arm, armel, hppa, i386, ia64, m68k, mips, mipsel, powerpc, s390, sparc. Po poprawnym wykonaniu polecenia debootstrap dodajemy repozytoria do pliku /wheezy-chroot/etc/apt/sources.list:

```
deb http://ftp.pl.debian.org/debian/ wheezy main
contrib non-free
deb-src http://ftp.pl.debian.org/debian/ wheezy main
contrib non-free

deb http://security.debian.org/ wheezy/updates main
contrib non-free
deb-src http://security.debian.org/ wheezy/updates
main main contrib non-free
```

Oraz do pliku /wheezy-chroot/etc/network/interfaces adresy IP serwerów DNS¹⁴:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Opcjonalnie można z katalogu w którym znajduje się nowy system zrobić obraz.

Tworzenie pliku obrazu o rozmiarze 2GB¹⁵:

```
dd if=/dev/zero of=plik.img bs=1024 count=2000000
```

¹⁴Adresy DNS Google

¹⁵Polecenie dd służy do niskopoziomowego kopiowania surowych danych

Za pomocą polecenia `mkfs.ext4` zakładamy system plików na obrazie `img`.

```
mkfs.ext4 plik.img
```

Zamontowanie obrazu:

```
mount -t ext4 -o loop plik.img /mnt
```

Przekopiowanie zawartości katalogu `/wheezy-chroot`:

```
cp -r /wheezy-chroot /mnt
```

Odmontowanie:

```
umount /mnt
```

W następnym kroku przystępujemy do zamontowania katalogów: `/proc`, `/dev`, `/sys`. Ważne jest aby zamontować wyżej wspomniane katalogi, ponieważ:

`/proc` - zawiera informacje o procesach, za pośrednictwem znajdujących się tam plików jądro udostępnia nam informacje na temat parametrów pracy systemu. `/proc` w przeciwieństwie do innych systemów plików przechowywany jest w pamięci, a nie na dysku.

```
ls /proc
[...]
```

3647	403	4348	5	588	682	749	86	92	988
404	44	5020	594	696	75	864	93	99	

```
[...]
```

Każdy katalog, którego nazwa jest liczbą odpowiada uruchomionemu w systemie procesowi. Liczby te informują o numerze PID procesu.

Sprawdzenie wersji jądra

```
cat /proc/version
Linux version 3.19.5-100.fc20.x86_64 (
  mockbuild@bkernel02.phx2.fedoraproject.org) (gcc
  version 4.8.3 20140911 (Red Hat 4.8.3-7) (GCC) )
#1 SMP Mon Apr 20 19:51:16 UTC 2015
```

Ilość pamięci RAM

```
ls -l /proc/kcore
-r----- . 1 root root 140737477881856 08-07 15:25
/proc/kcore
```

Lista partycji

```
cat /proc/partitions
```

Inne informacje zawarte na katalogu /proc:

- /proc/cpuinfo - informacje o procesorze
- /proc/meminfo - informacje o fizycznej pamięci RAM, obszarze wymiany
- /proc/mounts - lista zamontowanych systemów plików
- /proc/devices - lista dostępnych urządzeń
- /proc/filesystems - wspierane systemy plików
- /proc/modules - lista załadowanych modułów
- /proc/version - wersja jądra
- /proc/cmdline - parametry przekazane do jądra podczas uruchomienia
- /dev - katalog zawierający pliki specjalne, które odpowiadają urządzeniom. Pliki urządzeń tworzone są podczas instalacji. (komunikacja niskopoziomowa)
- /dev/sdb - dysk twardy
- /dev/console - plik odpowiedzialny za klawiaturę
- /dev/cdrom - plik optycznego nośnika danych, CDROM
- /sys - katalog zawierający pliki systemowe, w których można zmieniać parametry jądra

Modyfikacja pliku fstab tak aby katalogi montowały się automatycznie:

```
echo "proc $MY_CHROOT/proc proc defaults 0 0" >> /
etc/fstab
mount proc $MY_CHROOT/proc -t proc
echo "sysfs $MY_CHROOT/sys sysfs defaults 0 0" >> /
etc/fstab
mount sysfs $MY_CHROOT/sys -t sysfs
```

Montowanie wszystkich systemów plików z fstab:

```
mount -a
```

Opcjonalnie kopiujemy pliku `hosts` i `mounts`.

```
cp /etc/hosts $MY_CHROOT/etc/hosts
cp /proc/mounts $MY_CHROOT/etc/mtab
```

Zmiana głównego katalogu plików.

```
chroot $MY_CHROOT
```

Jeżeli uruchamiamy system który jest w innej architekturze niż komputer hosta konieczne jest dokończenie instalacji.

```
/debootstrap/debootstrap --secondstage
```

Istnieje również pakiet o nazwie `dchroot`¹⁶, który umożliwia uruchomienie programu, który jest zainstalowany w systemie do którego mamy dostęp za pomocą `chroot`.

Montowanie partycji

```
root@debian:~# apt-get install dchroot
```

Konfiguracja - plik `/etc/dchroot.conf` dodajemy 1 linie

```
x64 /wheezy -chroot
```

Uruchomienie programu w `chroot` za pomocą `dchroot`

```
root@debian:~# dchroot -c i386 -d nazwa_programu
```

¹⁶<http://linux.die.net/man/1/dchroot>

Rozdział 2

Protokół SIP

SIP (Session Initiation Protocol) to protokół drugiej warstwy modelu OSI, który działa w architekturze klient-serwer opisany w dokumencie RFC3261¹. Dokument można znaleźć na stronach organizacji IETF (Internet Engineering Task Force). Jego zadanie to inicjowanie sesji pomiędzy jednym lub wieloma klientami, pozwala również na modyfikacje i zakończenie sesji. Uściślając, główne zadanie protokołu to dostarczenie zestawu funkcji obsługi połączenia. Obecnie jest to dominujący protokół sygnalizujący dla telefonii Voice over IP. Nie jest protokołem transportowym, dlatego konieczne jest jednocześnie zastosowanie protokołu RTP. SIP jest alternatywą dla H.323. Pomysłodawcami protokołu są: M. Handley, H. Schulzrinne, E. Schooler i J. Rosenberg

Protokół składa się z 2 komponentów:

- user agents - użytkownicy
- network agents - serwery sieciowe

User agent (UA) jest punktem końcowym, może odbierać i ustanawiać połączenia. User agent client (UAC) inicjuje żądania SIP. Serwer user agent server (UAS) odbiera żądania od UAC i zwraca odpowiedź do usera.

SIP obsługuje pięć funkcji połączenia i zakończenia komunikacji

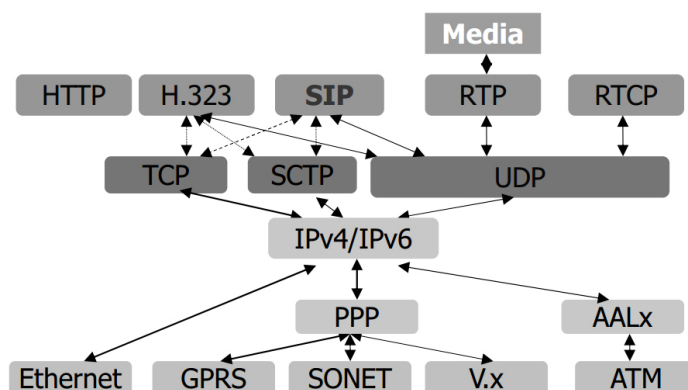
- lokalizacja użytkownika - określenie systemu końcowego, który będzie wykorzystywany do komunikacji
- dostępność użytkownika - sprawdzenie gotowości użytkownika, który ma być wywołany do komunikacji
- możliwości użytkownika - lista parametrów użytkownika

¹<https://www.ietf.org/rfc/rfc3261.txt>

- połączenie sesji - dzwonienie, zestawienie sesji po obu stronach połączenia
- zarządzanie sesją - modyfikowanie parametrów sesji

SIP można zdefiniować jako komponent, który można użyć z innymi protokołami IETF aby zbudować kompletną architekturę multimedialną. Należy tutaj wymienić protokoły takie jak Real-time Transfer Protocol (RTP) dla transportu danych w czasie rzeczywistym i dostarczania informacji o QoS (jakości usług). Kolejnym protokołem to Real-Time Streaming protocol (RTSP), który wykorzystywany jest do sterowania strumieniem medialnym, Media Gateway Control Protocol (MEGACO) wykorzystywany do sterowania bramami w sieci publicznej PSTN oraz Session Description Protocol (SDP) dla opisanie sesji.

Protokół dostarcza również szereg usług bezpieczeństwa, które między innymi chronią przed atakami DOS (denial-of-service), dbają o uwierzytelnienie, chronią integralność, oferują możliwość szyfrowania. SIP jest kompatybilny z protokołem IPv4 oraz IPv6.



Rysunek 2.1: Stos protokołów

2.1 Elementy sieci SIP

Terminale końcowe (nodes) to aplikacje uruchamiane na komputerze lub na aparatach telefonicznych, które używają protokołu SIP i RTP.

Rejestrator SIP (registrar) baza danych, która komunikuje się z węzłami SIP w celu zbierania i archiwizacji informacji na temat użytkowników SIP. Zebrane dane są wykorzystywane w trakcie nawiązywania połączenia do odnajdowania w sieci węzłów docelowych.

Połączenia SIP, które nie są lokalne muszą być przekazywane przez serwery pośredniczące SIP proxy.

2.2 Działanie protokołu SIP

Dzwoniący do siebie użytkownicy stosują identyfikację SIP URI (Uniform Resource Identifier). Adres wygląda następująco:

`uzytkownik@domena:port`

Gdzie korzysta się z domyślnego numeru portu 5060. SIP jest podobny do protokołu HTTP, dzieli z nim wiele zasad konstrukcyjnych między innymi używa zwykłego tekstu i wykorzystuje mechanizm żądanie-odpowiedź. SIP jest w stanie zagwarantować bezpieczne, szyfrowane połączenie TLS (Transport Layer Security) SIPS URI. Protokół zbudowany jest warstwowo co oznacza, że jego zachowanie jest opisane przez niezależne stany przetwarzania z luźnym powiązaniem między poszczególnymi stanami (stadiami). Najniższą warstwą SIP jest składnia i kodowanie, które wykonywane jest przy użyciu gramatyki BNF (Backus-Naur Form). Warstwa transportowa plasuje się na drugim miejscu w hierarchii. Jej zadanie to określenie w jaki sposób klient wysyła żądania i otrzymuje odpowiedzi oraz jak serwer wysyła/odbiera żądania/odpowiedzi poprzez sieć. Trzecia warstwa to warstwa transakcyjna. Transakcja jest żądaniem wysyłanym przez transakcję klienta do transakcji serwera, z wszystkimi odpowiedziami na żądanie, wysłanymi z transakcji serwera do klienta. Warstwa obsługuje retransmisję warstwy aplikacyjnej, dopasowanie odpowiedzi do żądań. Ostatnia warstwa to warstwa użytkownika transakcji. Każdy z użytkowników jest użytkownikiem transakcji.

Protokół SIP korzysta z następujących metod:

- INVITE - zapoczątkowanie wywołania przez zaproszenie użytkownika do sesji
- ACK - potwierdza odebranie odpowiedzi na żądanie INVITE
- BYE - zakończenie wywołania
- CANCEL - anulowanie żądania trwającego
- REGISTER - rejestracja agenta użytkownika
- OPTIONS - odpytanie o możliwości serwera
- INFO - przenoszenie informacji dodatkowej, np. cyfr DTMF

Kody odpowiedzi SIP (podobieństwo do kodów odpowiedzi protokołu HTTP):

- 1xx - Wiadomości informacyjne
- 2xx - Odpowiedzi pozytywne
- 3xx - Odpowiedzi przekierowania
- 4xx - Odpowiedzi błędnych żądań
- 5xx - Odpowiedzi błędu serwera
- 6xx - Odpowiedzi błędów globalnych

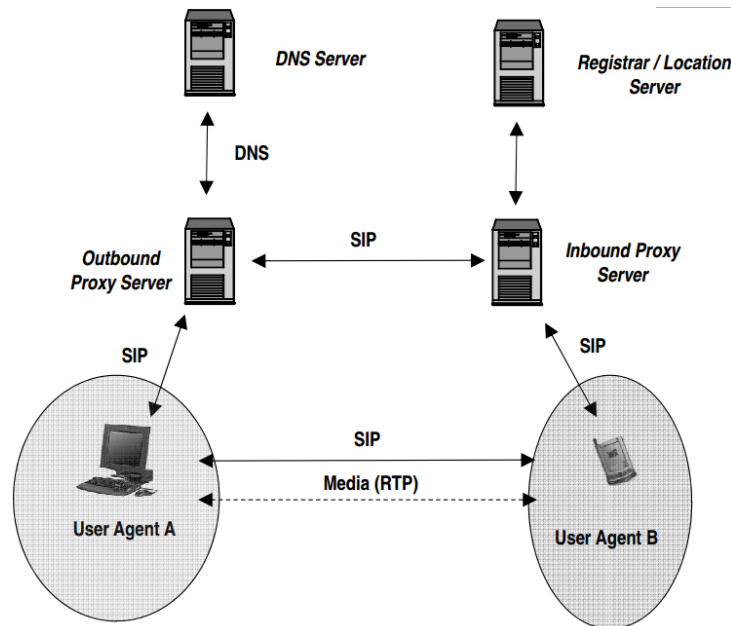
Wyżej wspomniane metody oraz odpowiedzi używane są do ustanowienia sesji połączenia.

Architektura składa się z następujących elementów:

- User Agent Client (UAC)
 - systemy końcowe
 - UAC odpowiedzialny jest za wysyłanie żądań SIP
- User Agent Server - UAS
 - odbiera żądania połączeń
 - wysyła odpowiedzi
- User Agent (terminal)
 - UAC oraz UAS
- Redirect Server
 - zajmuje się przekierowaniem użytkownika na inny serwer
- Proxy Server
 - reprezentuje żądanie do innego serwera
- Registrar
 - odpowiedzialny za odebranie zgłoszenia rejestracyjnego

- Location Server
 - baza danych użytkowników

Graficzna reprezentacja została przedstawiona na rysunku 2.2.



Rysunek 2.2: Architektura SIP - User Agent

2.3. IMPLEMENTACJE PROTOKOŁU SIP W SYSTEMIE LINUX - SERWERY³¹

Nagłówek SIP ma postać tekstową dzięki czemu jest łatwy do odczytania.

```
INVITE sip:UserB@there.com SIP/2.0
Via: SIP/2.0/UDP 4.3.2.1:5060
To: User B <sip:UserB@there.com>
From: User A <sip:UserA@here.com>
Call-ID: 4598998103413@4.3.2.1
CSeq: 1 INVITE
Contact: <sip:UserA@4.3.2.1>
Content
-Type: application/sdp
Content
-Type: application/sdp
Content-Length: 201
v=0
o=UserB 289375749 289375749 IN IP5 100.101.102.103
s=-
c=IN IP4 100.101.102.103
t=0 0
m=audio 5004 RTP/AVP 0
m=audio 53000 RTP/AVP 96
c=IN IP4 200.201.202.203
a=rtpmap:96 telephone-event
```

2.3 Implementacje protokołu SIP w systemie Linux - serwery

2.3.1 miniSipServer

MiniSIPServer² to profesjonalny, wieloplatformowy serwer VoIP. Serwer, do którego można pobrać klienta tej samej firmy o nazwie miniSIPPhone jest oparty o standard SIP. Autorzy oprogramowania zadbali również o to aby system był kompatybilny z urządzeniami, które oparte są na procesorze ARM.

Instalacja

Przed zainstalowaniem serwera SIP, należy zainstalować niezbędne biblioteki.

²<https://www.myvoipapp.com/>

```
sudo apt-get install gcc g++ libqt4-dev libqtcore4  
libqtdiag4 libqt4-network libqt4-xml libssl-dev  
libmysqlclient18 libmysqlclient-dev python-dev  
libsrtplib0-dev
```

Plik: <http://www.myvoipapp.com/download/index.html>.

```
sudo dpkg -i mss_pi_u20.deb
```

Jeżeli wszystko się powiedzie to pliki serwera znajdują się w katalogu `/opt/sip-server/`

Uruchomienie

Serwer uruchamiamy za pomocą polecenia:

```
/opt/sipserver/msscli&
```

Możliwość zarządzania serwerem jest możliwa z poziomu przeglądarki internetowej. W tym celu należy w oknie przeglądarki wpisać własny numer IP i port 8080.

2.3.2 Asterisk

Najpopularniejszym serwerem wykorzystującym protokół SIP jest Asterisk³. Jest to kompletne oprogramowanie centrali telefonicznej PBX. Główne cechy:

- dostępny na systemy Windows, Linux, BSD, Mac OS X
- obsługa protokołów SIP, IAX, H.323, ADSI, MGCP, SCCP
- telekonferencje
- poczta głosowa
- nagrywanie rozmów
- współpraca z bazami danych MySQL, PostgreSQL

³<http://www.asterisk.org/>

2.3. IMPLEMENTACJE PROTOKOŁU SIP W SYSTEMIE LINUX - SERWERY33

2.3.3 Cipango

Rozszerzenie do popularnego serwera Jetty napisane w języku Java. Główne cechy:

- łatwość w użyciu
- wspiera SIP Servlets oraz HTTP Servlets

2.3.4 Elastix

Oprogramowanie przeznaczone na system Linux, opiera swoją funkcjonalność na wspominanym wcześniej Asteriskiem, FreePBX, HylaFAX, Openfire oraz Postfix.

2.3.5 FreeSWITCH

Wieloplatformowe oprogramowanie służące do komunikacji za pomocą dźwięku, wideo oraz tekstu. Główne cechy:

- wsparcie dla Skype, SIP, H.323, WebRTC
- moduł konferencji

2.3.6 Kamailio

Serwer umożliwiający zapewnienie ułatwienie z konfiguracji Asteriska. Za pomocą niego można rozbić konfiguracje na serwer SIP obsługujący rejestracje oraz serwer Asterisk realizujący połączenia.

2.3.7 Inne serwery SIP

- 2600Hz blue.box
- FreePBX
- GNU SIP Witch
- Mobicents
- Mysipswitch
- OpenSIPS
- SailFin

- SIP Express Router (SER)
- Enterprise Communications System sipXecs
- Yate
- YXA

2.4 Implementacje protokołu SIP w systemie Android - serwery

2.4.1 uSipServer

Prosty w użyciu serwer SIP na system Android, do pobrania ze sklepu Google Play.

2.5 Implementacje protokołu SIP w systemie Linux - klienty

2.5.1 Blink

Klient SIP działający w systemach Linux oraz Windows. Napisany w oparciu o bibliotekę Qt. Umożliwia między innymi obsługę wideo, wymianę plików między użytkownikami, udostępnianie pulpitu.

2.5.2 KPhone

Klient przeznaczony tylko na systemy Linux. Ma zaimplementowaną funkcjonalność telefonu VoIP, ale nie ogranicza się tylko do niego. Wspiera NAT oraz STUN. Obsługuje systemy dźwięku ALSA i OSS. Wykorzystuje protokół SRTP to szyfrowania rozmów głosowych.

2.5.3 Linphone

Program klienta napisany w języku C przez francuską firmę Belledonne Communications. Wspiera dodatkowo system Windows, Max OS X oraz systemu mobilne Windows Phone, iOS, Android. Zapewnia szyfrowanie rozmów z użyciem protokołu ZRTP.

2.5.4 Zoiper

Wieloplatformowe oprogramowanie o szerokiej funkcjonalności. Oferuje wysoką jakość rozmów oraz szyfrowanie ich z wykorzystaniem protokołów SRTP oraz TLS. Występuje również w postaci programu klienta.

2.5.5 Inne klienty SIP

- Ekiga
- Empathy
- Jitsi
- MicroSIP
- QuteCom
- SFLphone
- Telephone
- Twinkle
- Yate client

2.6 Implementacje protokołu SIP w systemie Android - klienty

System Android oferuje natywnego klienta SIP⁴. Jest on dostępny od wersji 2.3 systemu.

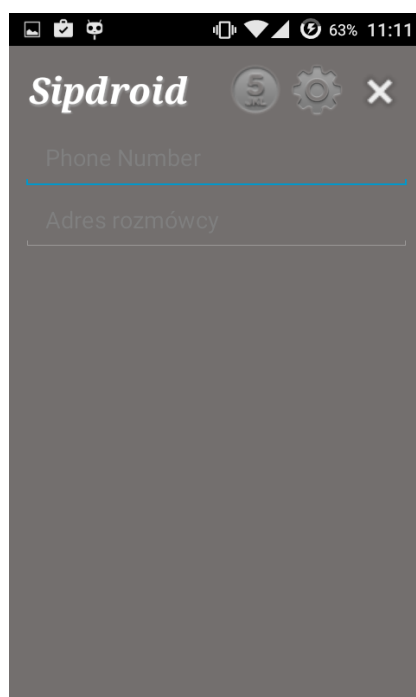
2.6.1 Konfiguracja natywnego klienta SIP

Należy przejść do Ustawień, wybrać ustawienia rozmów i następnie poszukać zakładki Konta. Następnie pojawi się lista kont SIP, aby dodać nowe konto należy wybrać przycisk "Dodaj konto". Na dodanie nowego konta składa się: podanie nazwy użytkownika, hasła, serwera. Po pomyślnym dodaniu konta system jest gotowy do dzwonienia przez internet wykorzystując WiFi. Dzwonienie z wykorzystaniem sieci 3G jest niemożliwe.

⁴<http://www.voipvoip.com/android/sip.html>

2.6.2 Sipdroid

Projekt oparty o dostęp do kodu źródłowego. W najnowszej wersji 3.0 umożliwia szyfrowanie rozmów. Aplikacja oferuje wysyłanie VideoSMSów oraz stworzenie darmowego konta PBX ale użytkowników posiadających konto w usłudze Google Voice. Aplikacja pozwala wybrać w jakich sieciach ma korzystać z VoIP (Wi-Fi, 3G, EDGE).

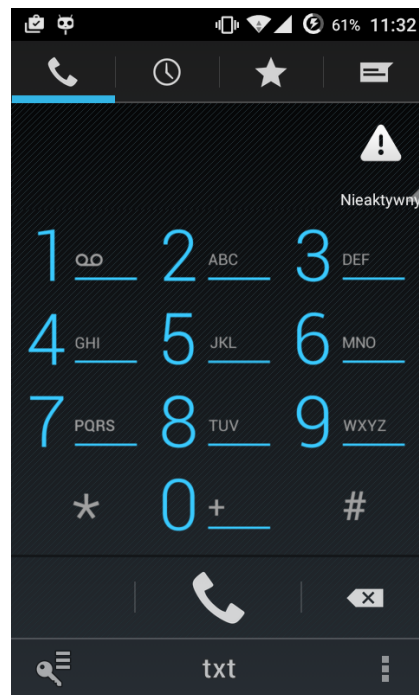


Rysunek 2.3: Sipdroid

2.6.3 CSipSimple

Prosty klient SIP na system Android. Cechuje się wysoką wydajnością i łatwą konfiguracją. Oferuje nagrywanie rozmów oraz obsługuje kodek HD.

2.6. IMPLEMENTACJE PROTOKOŁU SIP W SYSTEMIE ANDROID - KLIENTY37



Rysunek 2.4: CSipSimple

2.6.4 Bria VoIP Softphone

Kolejny klient SIP dostępny w sklepie Google (komercyjny). Zarządzany przez firmę CounterPath Corporation, lidera na rynku produktów i rozwiązań oprogramowania VoIP. Produkt cechuje się wysoko bezpieczną komunikacją między użytkownikami oraz wysoką jakością dźwięku. Konto Premium umożliwia połączenia wideo. Umożliwia połączenia z wykorzystaniem sieci Wi-Fi oraz 3G.

2.7 Rynek telefonii SIP

Jednym z problemów związanym z protokołem SIP, który wchodzi w skład VoIP jest niewłaściwa konfiguracja zapory sieciowej co uniemożliwia zainicjowanie połączenia z zewnątrz. Rozproszona natura protokołu utrudnia spełnienie wszystkich prawnych wymagań nałożonych na operatorów telekomunikacyjnych. Na polskim rynku jest co raz więcej rozwiązań telefonicznych opartych o protokół SIP, oferują je między innymi firmy Orange, HaloNet, Tlenofon, Freeconet, HopIn.pl, INTERFON, IPFON, EuroTC, DigiTEL, Siec T2, Actio, Newfon, easyCALL, PLFON, iFON. Duże firmy telekomunikacyjne takie jak 3Com, Avaya, Cisco oraz Nortel umożliwiają współpracę z sieciami SIP. Na polskim rynku telefonii dla firm najczęściej sprzedaje się klasycznych systemów telefonicznych, ale rozwiązania telefonii IP cieszą się coraz większym zainteresowaniem. Głównymi odbiorcami systemów telefonii IP są firmy średniej wielkości, które zatrudniają 300 - 800 pracowników oraz duże korporacje. Infonetics Research prognozuje, że do 2018 rynek usług VoIP przeznaczonych dla odbiorów domowych i biznesowych wzrośnie do 88 miliardów dolarów na świecie. Najszybciej rośnie segment hostowanych usług VoIP oraz Unified Communications przeznaczonych dla przedsiębiorstw. Największe przychody w grupie usług biznesowych generują usługi IP PBX. Z usług operatorów VoIP korzystają mieszkańcy Francji, Dani i Niemiec. Polska ma swój udział w około 4 %.

Rozdział 3

Projekt informatyczny AndroidLinuxSIPService

Celem projektu jest uruchomienie serwera SIP miniSIPServer na systemie Debian Wheezy 7.0 obok systemu Android oraz przygotowanie aplikacji na system Android do zarządzania dystrybucją systemu linux oraz serwerem SIP. Zadanie obejmuje następujące czynności:

- przygotowanie dystrybucji systemu Linux na system Android
- zainstalowanie środowiska graficznego Xfce na przygotowanej dystrybucji
- zainstalowanie programu miniSipServer na przygotowanej dystrybucji
- zainstalowanie serwera SSH oraz serwera VNC Tightvnc
- przygotowanie skryptów startowych dla SSH i Tightvnc i umieszczenie ich w nowo przygotowanej dystrybucji
- uruchomienie przygotowanej dystrybucji razem z serwerem SIP na telefonie z systemem Android
- przygotowanie aplikacji na system Android do zarządzania systemem Linux.

3.1 Narzędzia programistyczne użyte do realizacji

3.1.1 Przygotowanie dystrybucji

Dystrybucja przygotowywana jest za pomocą narzędzia debootstrap.

```
mkdir /home/debian-chroot
debootstrap --arch armhf wheezy /home/debian-chroot
    http://ftp.pl.debian.org/debian
```

Następnie tworzony jest obraz z powstałego katalogu.

```
dd if=/dev/zero of=plik.img bs=1024 count=2000000

mkfs.ext4 debian.img

mkdir /mnt/debian-chroot
mount -t ext4 -o loop debian.img /mnt/debian-chroot

cp -r /home/debian-chroot /mnt/debian-choort
```

3.1.2 Przygotowanie skryptu

W następnym kroku przygotowany zostaje skrypt w języku Bash, który ma za zadanie:

- zamontowanie przygotowanego obrazu
- zamontowanie katalogów /proc, /dev, /sys¹, /dev/pts²
- opcjonalnie uruchomienie serwera SSH i serwera SIP w "nowym systemie" wykorzystując polecenie chroot i skryptów startowych
- zatrzymanie uruchomionych usług
- odmontowanie katalogów /proc, /dev, /sys, /dev/pts
- odmontowanie przygotowanego obrazu

¹/sys zawiera pliki systemowe w których można zmieniać parametry jądra

²/dev/pts to wirtualny system plików, który zawiera informacje o pseudo terminalach

3.1.3 Wybór serwera SIP

Jako serwer SIP wykorzystywany jest wcześniej wspomniany miniSipServer w wersji 22. miniSipServer jest wieloplatformowym serwerem. Firma udostępnia również dystrybucje na procesory ARM, dedykowaną na Raspberry Pi. Serwer ten został wybrany ze względu na stabilność działania na systemie w architekturze ARM.

3.1.4 Przygotowanie aplikacji na system Android

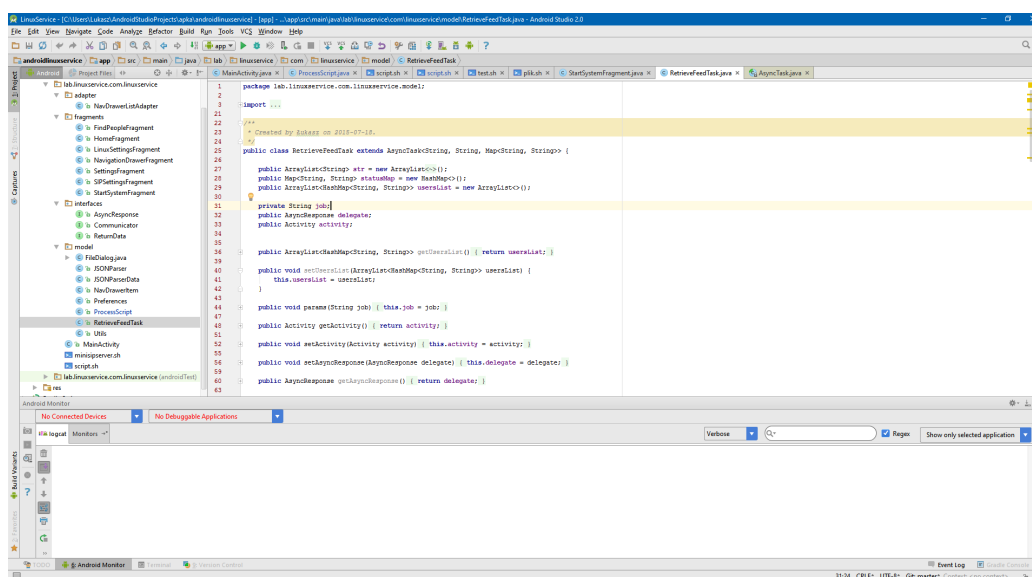
Aplikacja na system Android została przygotowana w środowisku Android Studio³, które jest oparte na IntelliJ⁴. Jest to kompletne środowisko programistyczne, wszystko jest w jednym miejscu. Bogata funkcjonalność zwalnia nas z ciągłego testowania kodu poprzez kompilacje. Aplikacja wymaga do pracy jedynie bibliotek JDK. SDK Andoirda pobierane jest automatycznie w razie potrzeby. Za pomocą specjalnego menedżera można zarządzać zainstalowanymi składnikami. Interfejs aplikacji jest bardzo intuicyjny. Układ elementów i okien jest konfigurowalny. Powyższe zalety zdecydowały na skorzystaniu z Android Studio. Do wyboru na rynku jest jeszcze środowisko Eclipse⁵ z wtyczką ADT. Interfejs aplikacji prezentuje rysunek 3.1.

³<http://developer.android.com/intl/es/tools/studio/index.html>

⁴<https://www.jetbrains.com/idea/> - IDE do programowania w języku Java firmy JetBrains, której środowiska uważane są za najlepsze na rynku

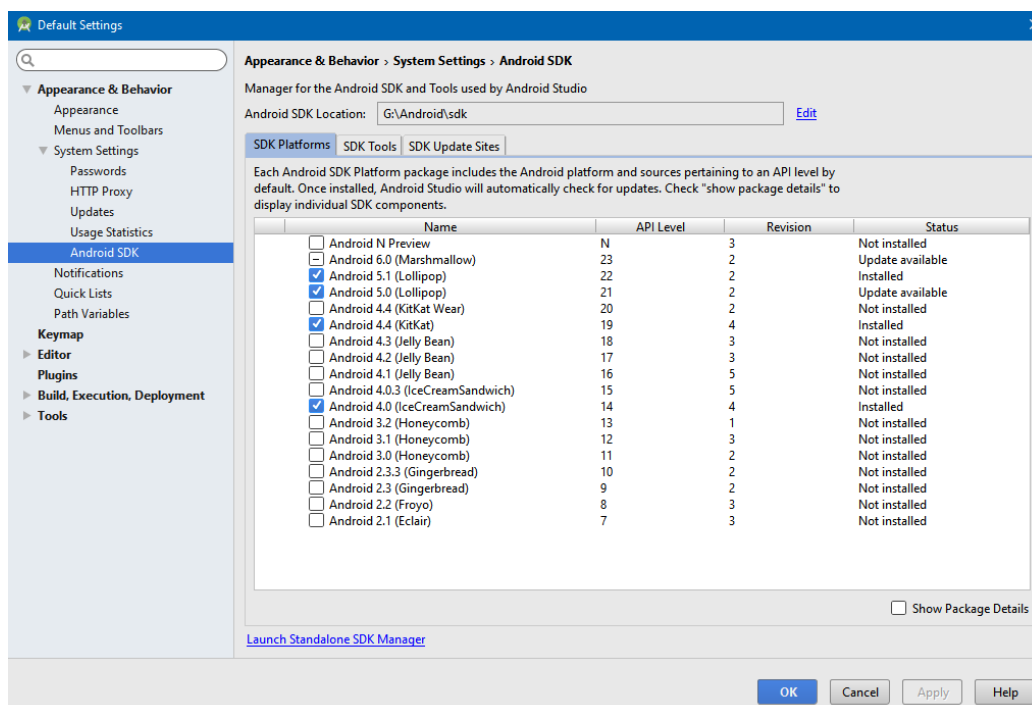
⁵<https://eclipse.org/>

3.1. NARZĘDZIA PROGRAMISTYCZNE UŻYTE DO REALIZACJI 43



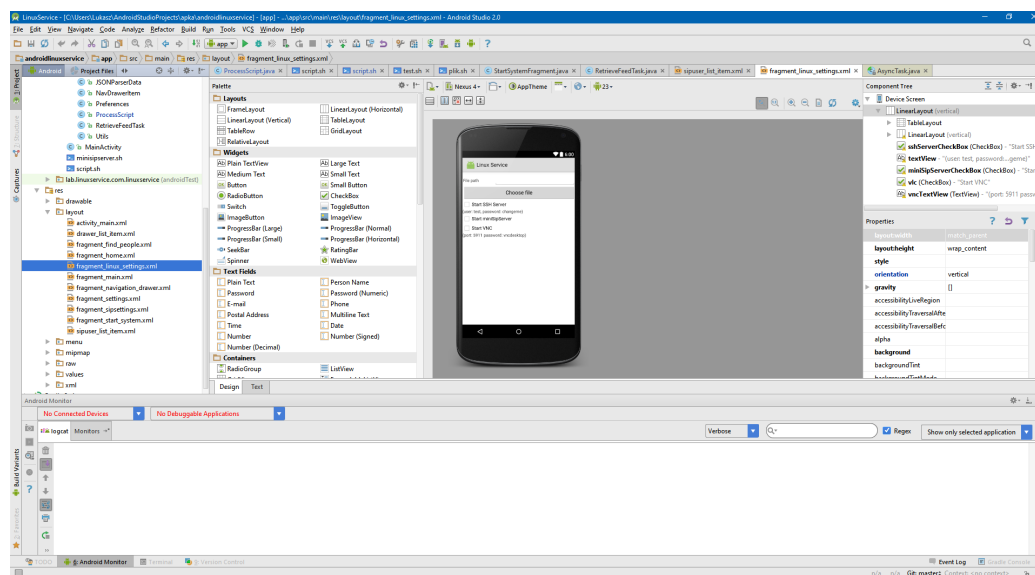
Rysunek 3.1: Interfejs Android Studio

Za pomocą modułu SDK Manager można pobrać wersje systemu Android.



Rysunek 3.2: Menedżer pakietów SDK

IDE oferuje graficzny edytor wyglądu aplikacji, który znacznie ułatwia pracę.



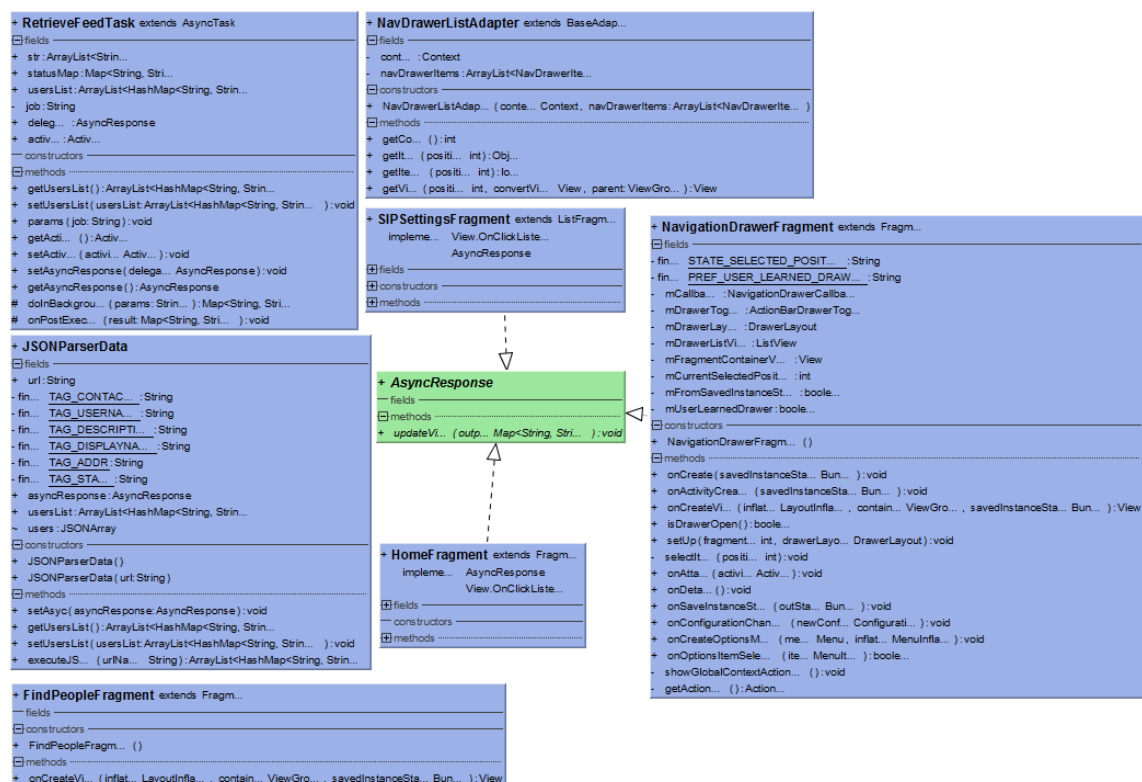
Rysunek 3.3: Widok projektowy wyglądu aplikacji

3.2 Kod źródłowy aplikacji i jej działanie

Aplikacja została przygotowana z wykorzystaniem API na poziomie 22, które obsługuje system Andoird 5.1 Lollipop.

3.2.1 Struktura klas aplikacji

Diagramy zostały wygenerowane na podstawie napisanego kodu za pomocą wtyczki simpleUML Diagram w programie Android Studio. Struktura klas aplikacji przedstawia się następująco:



Rysunek 3.4: Struktura klas aplikacji cz. 1



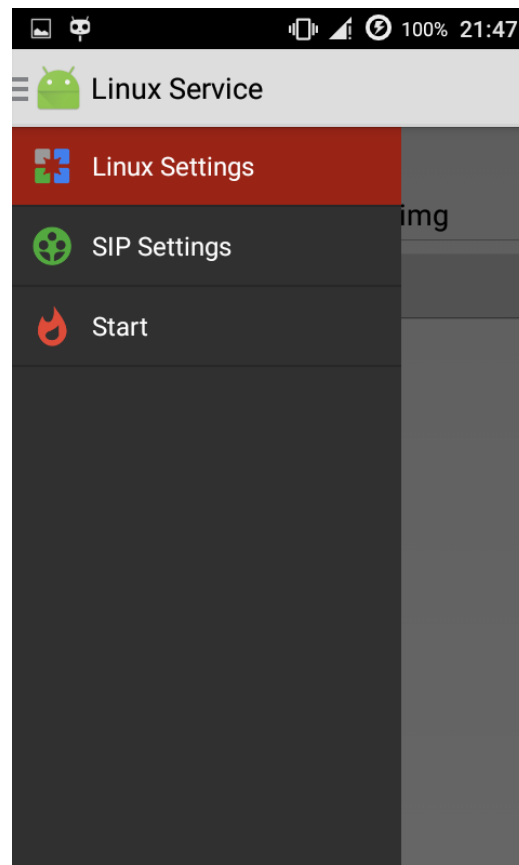
Rysunek 3.5: Struktura klas aplikacji cz. 2

<pre> + FileDialog - fields - fin... PARENT_DIR: String - fin... TAG: String - fin... Strin... - currentP... File - fileListener... ListenerList<FileSelectedListe... - dirListener... ListenerList<DirectorySelectedListe... - fin... activ...: Activ... - selectDirectoryOpt...: boolean... - fileEndsW...: String - constructors + FileDialog (activ...: Activ..., path: File) - methods + createFileDialog... (Dialog): void + addFileDialog... (listener: FileSelectedListe...): void + removeFileDialog... (listener: FileSelectedListe...): void + setSelectedDirectoryOpt... (selectedDirectoryOpt...: boolean...): void + addDirectoryListe... (listener: DirectorySelectedListe...): void + removeDirectoryListe... (listener: DirectorySelectedListe...): void + showDialog...(): void + fireFileSelectedE... (file: File): void + fireDirectorySelectedE... (directory: File): void + loadFile... (path: File): void + getChosenF... (fileChos...: String): File + setFileEndsW... (fileEndsW...: String): void </pre>	<pre> + LinuxSettingsFragment extends Fragment - implements View.OnClickListener - fields - fileChooser...: Button - scriptChooser: Button - logView...: Button - fileChooserT...: TextView - scriptChooserT...: TextView - logScroll...: ScrollView - sshServer: CheckBox - minSipSer...: CheckBox - vlc: CheckBox - preferences: Preference... - imagePathSta...: String - minSipServerSta...: String - sshServerStatus: String - vlcStatus: String - activ...: Activity - constructors - methods </pre>	<pre> + NavDrawerItem - fields - title: String - icon: int - count: String - isCounterVisib...: boolean... - constructors + NavDrawerItem() + NavDrawerItem(title: String, icon: int) + NavDrawerItem(title: String, icon: int, isCounterVisib...: boolean..., count: String) - methods + getTitle(): String + getIcon(): int + getCount(): String + getCounterVisibi...(): boolean... + setTitle(title: String): void + setIcon(icon: int): void + setCount(count: String): void + setCounterVisibi... (isCounterVisib...: boolean...): void </pre>	<pre> + Utils - fields - constructors - methods + bytesToHex(bytes: byte[]): String + getUTF8Byt... (str: String): byte[] + loadFileAsSt... (fileName: String): String + getMACAddress (interfaceName: String): String + getIPAddress (useIPv4: boolean): String </pre>
--	--	--	---

Rysunek 3.6: Struktura klas aplikacji cz. 3

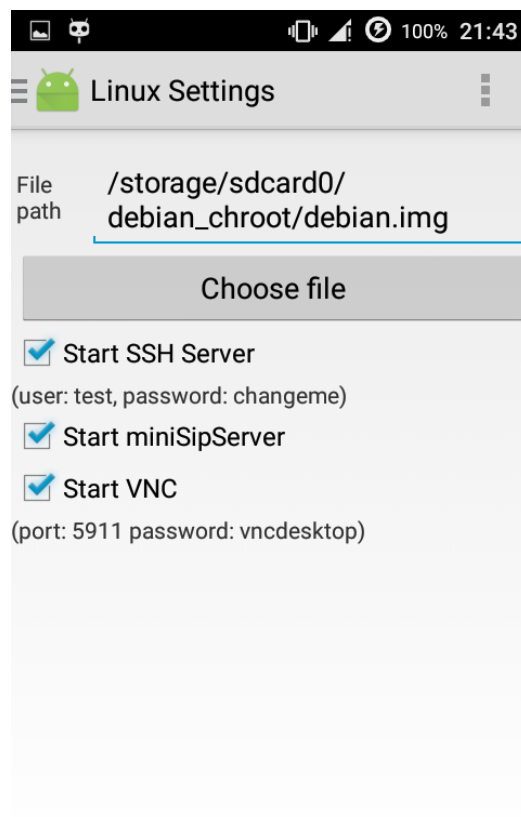
3.2.2 Interfejs aplikacji

Interfejs aplikacji składa się z trzech części:



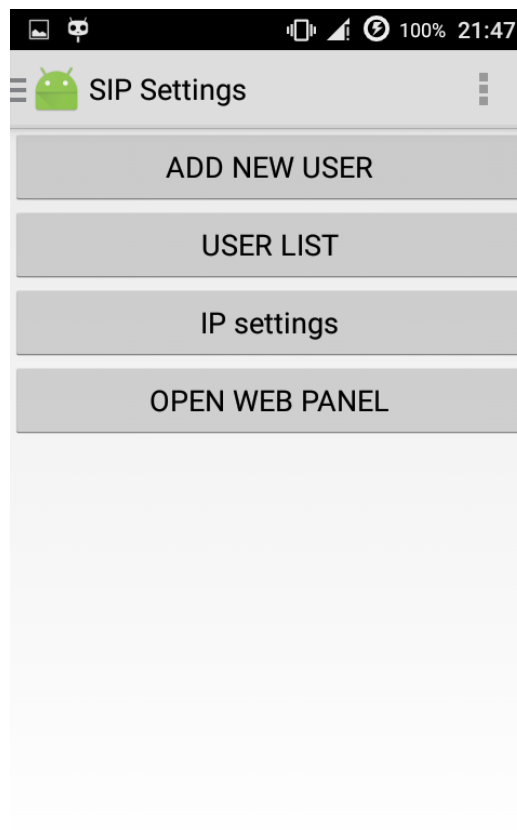
Rysunek 3.7: Menu aplikacji

- Linux Settings - zakładka umożliwia wybranie obrazu systemu, który ma być montowany. Plik wybieramy z okna dialogowego, w razie wybrania niepoprawnego pliku zostaniemy o tym poinformowani. Kolejno jest możliwość uruchomienia trzech usług: serwera SSH, serwera mini-SipServer oraz serwera VNC Tightvnc. Wybrane opcje są zapamiętywane w pamięci programu i są wczytywane przy kolejnym uruchomieniu programu.

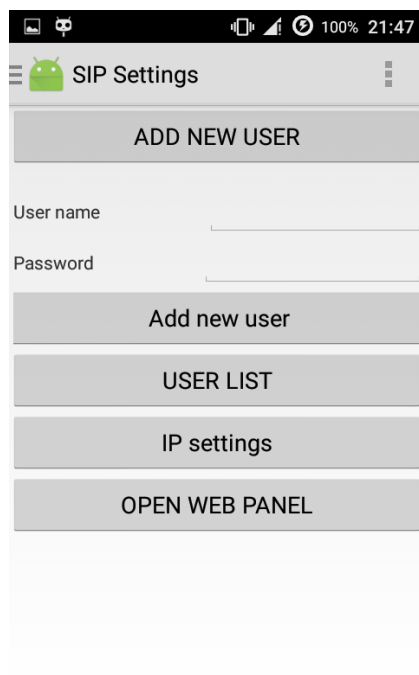


Rysunek 3.8: Karta Linux Settings

- SIP Settings - zakładka umożliwia zarządzania serwerem miniSipServer. Do zarządzania wykorzystywane jest API, które oferuje miniSipServer. W tej części mamy dostęp do:
 - dodania nowego użytkownika - Rysunek 3.10
 - wyświetlenie listy użytkowników - Rysunek 3.11
 - usunięcie użytkownika - Rysunek 3.11
 - jeżeli chcemy zarządzać serwerem miniSipServer, który nie jest uruchomiony na naszym urządzeniu możemy podać inny adres IP - Rysunek 3.12
 - dostęp do panelu administracyjnego poprzez przeglądarkę internetową.

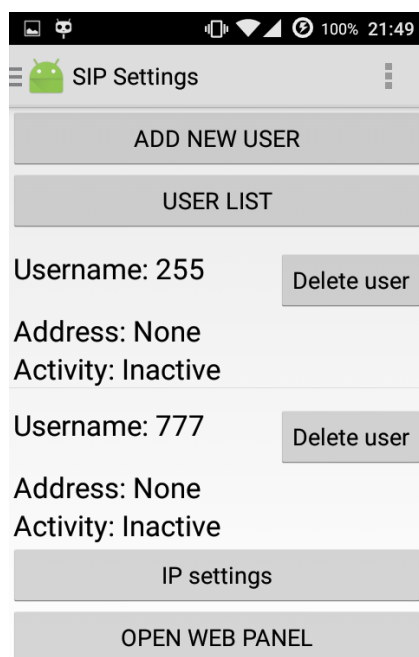


Rysunek 3.9: Karta SIP Settings



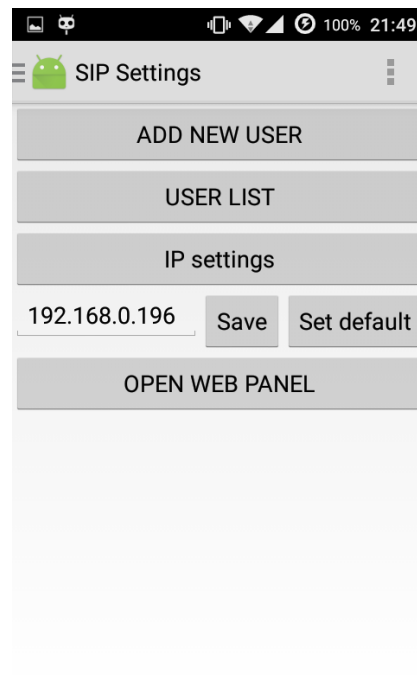
The screenshot shows the 'SIP Settings' application interface. At the top, there is a header bar with a hamburger menu icon, the app title 'SIP Settings', and a three-dot menu icon. Below the header, there is a large button labeled 'ADD NEW USER'. Underneath this button, there are two input fields: 'User name' and 'Password'. Below the input fields, there is a button labeled 'Add new user'. Further down, there are three more buttons: 'USER LIST', 'IP settings', and 'OPEN WEB PANEL'.

Rysunek 3.10: Formularz dodawania nowego użytkownika



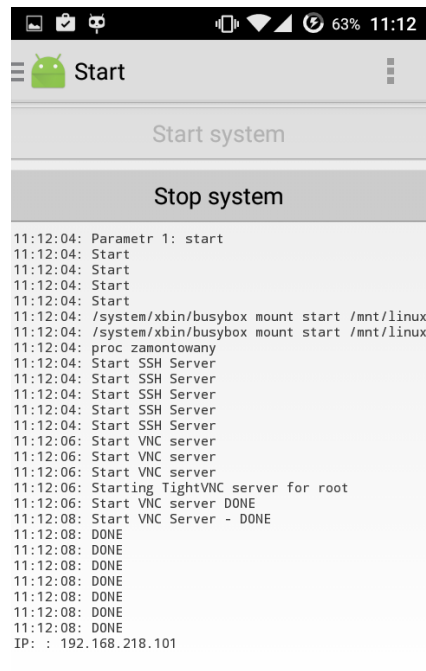
The screenshot shows the 'SIP Settings' application interface with the 'USER LIST' button selected. The screen displays a list of users. Each user entry consists of a text block and a 'Delete user' button. The first user entry shows 'Username: 255', 'Address: None', and 'Activity: Inactive'. The second user entry shows 'Username: 777', 'Address: None', and 'Activity: Inactive'. Below the list, there are two buttons: 'IP settings' and 'OPEN WEB PANEL'.

Rysunek 3.11: Lista użytkowników



Rysunek 3.12: Ustawienia adresu IP
















- Start - zakładka pozwala na uruchomienie systemu bądź jego zatrzymanie z dostępem do logu - Rysunek 3.13



Rysunek 3.13: Start / Stop systemu

3.2.3 Interfejs webowy serwera miniSipSerwer

miniSipServer oferuje zarządzanie serwerem poprzez stronę internetową. Interfejs webowy jest dostępny od wersji 3.1. Razem ze startem serwera SIP startuje serwer HTTP, którego zadaniem jest umożliwienie dostępu do strony internetowej. Domyślnie miniSipServer dla serwera HTTP otwiera port TCP 8080. Po wpisaniu `http://adresip:8080/` w przeglądarkę internetową wejdziemy na stronę umożliwiającą zarządzanie serwerem SIP.

SIP server					
Data ▾					
Dial plan ▾					
Services ▾					
Reports ▾					
Maintain ▾					
Help ▾					
<div><div>←</div><div>→</div><div>Add a local user</div></div>					
Local user					
User name	State	Description	Display name	Address	Operation
222		222			 
255					 
444		444		192.168.218.1	 
666		666			 
777		777		192.168.218.101	 

Rysunek 3.14: Lista użytkowników w systemie

Zakończenie

Dostępne obecnie urządzenia mobilne bazujące na systemie Android to nie tylko aparaty służące do wykonywania rozmów telefonicznych czy pisania wiadomości tekstowych. Urządzenie daje nam wiele możliwości jeżeli wiemy jak je wykorzystać. Z telefonu komórkowego możemy zrobić dowolne urządzenie, może to być opracowany w pracy serwer telefonii SIP czy chociażby sniffer sieci Wi-Fi. Możliwości jakie daje nam oprogramowanie bazujące na systemie Linux jest bardzo duże. Rynek telefonii SIP rozwija się z każdym rokiem. Praktycznie każda większa firma powinna mieć własną sieć telefoniczną, a instalacja takiej sieci wcale nie musi oznaczać dużej inwestycji.

Spis rysunków

2.1	Stos protokołów	27
2.2	Architektura SIP - User Agent	30
2.3	Sipdroid	36
2.4	CSipSimple	37
3.1	Interfejs Android Studio	43
3.2	Menedżer pakietów SDK	43
3.3	Widok projektowy wyglądu aplikacji	44
3.4	Struktura klas aplikacji cz. 1	45
3.5	Struktura klas aplikacji cz. 2	46
3.6	Struktura klas aplikacji cz. 3	47
3.7	Menu aplikacji	48
3.8	Karta Linux Settings	49
3.9	Karta SIP Settings	50
3.10	Folmularz dodawania nowego użytkownika	51
3.11	Lista użytkowników	51
3.12	Ustawienia adresu IP	52
3.13	Start / Stop systemu	53
3.14	Lista użytkowników w systemie	54

Bibliografia

- [1] Marcin Bis, *Linux w systemach Embedded, btc Warszawa 2011*
- [2] Karim Yaghmour, *Embedded Android, O'Reilly 2013*
- [3] https://dug.net.pl/tekst/193/instalacja_debiana_metoda_debootstrap/
- [4] <http://androlinux.com/android-ubuntu-development/how-to-build-chroot-arm-ubuntu-images-for-android/>
- [5] <http://androlinux.com/>
- [6] <http://ingvar.blog.redpill-linpro.com/2011/05/20/running-fedora-on-the-galaxy-s/>
- [7] <http://mitchtech.net/android-ubuntu-chroot/>
- [8] <http://www.cyberciti.biz/faq/unix-linux-chroot-command-examples-usage-syntax/>
- [9] <https://www.kernel.org/doc/Documentation/filesystems/ramfs-rootfs-initramfs.txt>
- [10] <http://linux.die.net/man/2/chroot>
- [11] http://linux.die.net/man/8/pivot_root
- [12] <http://students.mimuw.edu.pl/SO/Projekt04-05/temat4-g4/mikrokontrolery.htm>
- [13] <https://github.com/meefik/linuxdeploy>
- [14] <http://www.g-loaded.eu/2008/01/28/how-to-extract-rpm-or-deb-packages/>