
**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**ELEKTROS IR ELEKTRONIKOS FAKULTETAS**

**Vaidas Kriščeliūnas**

# PROGRAMŲ SKIRTŲ IŠMANIOJO TELEFONO APLIKACIJŲ KŪRIMUI ANALIZĖ

Baigiamasis bakalauro projektas

**Vadovas**
Doc. dr. Vitas Grimaila

**KAUNAS, 2015**

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**ELEKTROS IR ELEKTRONIKOS FAKULTETAS**

**TELEKOMUNIKACIJŲ.KATEDRA**

# PROGRAMŲ SKIRTŲ IŠMANIOJO TELEFONO APLIKACIJŲ KŪRIMUI ANALIZĖ

Baigiamasis bakalauro projektas

**Telekomunikacijos (612H64001)**

**Vadovas**

Doc. dr. Vitas Grimaila

**Recenzentas**

Lekt. Aurelijus Budnikas

**Projektą atliko**

Vaidas Kriščeliūnas

**KAUNAS, 2015**

**TVIRTINU**:
KTU Elektros ir elektronikos fakulteto
Telekomunikacijų katedros vedėjas
Doc. Saulius Japertas
201….. ..... .....

# BAKALAURO BAIGIAMOJO PROJEKTO UŽDUOTIS

**Išduota studentui:** *Vaidas Kriščeliūnas* Grupė *RT-1*

**1. Darbo tema:**

Lietuvių kalba: Programų skirtų išmaniojo telefono aplikacijų kūrimui analizė

Anglų kalba: Analysis of Smartphone Application Development Software

Patvirtinta 201___ m. _____ mėn. __ ____ d. dekano potvarkiu Nr. ____

**2. Darbo tikslas:** *Atlikti programų skirtų išmaniųjų telefonų aplikacijų kūrimui analizę ir sukurti aplikaciją.*

**3. Reikalavimai ir sąlygos:** *Bakalauro baigiamasis projektas turi atitikti "Bendrąjį baigiamųjų projektų rengimo, gynimo ir saugojimo aprašą", patvirtintą KTU rektoriaus 2014 m. birželio 9 d. įsakymu Nr. A-313 ir "Baigiamųjų projektų rengimo ir gynimo metodinius reikalavimus", patvirtintus KTU Elektros ir elektronikos fakulteto dekano 2015 m. kovo 16 d. potvarkiu Nr. ST17-F-03-2.*

**4. Projekto struktūra.** *Turinys konkretizuojamas kartu su vadovu, atsižvelgiant į BBP pobūdį, pateiktą Metodinių reikalavimų 14 ir 15 punktuose.*

*Užduoties analizės dalyje išanalizuoti darbo objektą, atlikti literatūros analizę, išsikelti darbo tikslą ir uždavinius.*

*Analitinėje dalyje atlikti Android architektūros analizę, išanalizuoti programas skirtas išmaniųjų telefonų aplikacijų kūrimui, atlikti programų pasirinkimo kriterijų analizę.*

*Projektinėje dalyje sukurti mobiliąją aplikaciją naudojantis viena išanalizuotų programų.*

**5. Ekonominė dalis.** *Jei reikia ekonominio pagrindimo; turinys ir apimtis konkretizuojama darbo eigoje kartu su vadovu.*

**6. Grafinė dalis.** *Jei reikia, pateikiama schemos, algoritmai ir surinkimo brėžiniai; turinys ir apimtis konkretizuojama darbo eigoje kartu su vadovu.*

**5. Ši užduotis yra neatskiriama bakalauro baigiamojo projekto dalis**

**6. Projekto pateikimo gynimui kvalifikacinėje komisijoje terminas** *iki 2015-05-30*
*(data)*

Užduotį gavau: *Vaidas Kriščeliūnas* *2015-02-23*
*(studento vardas, pavardė, parašas)* *(data)*

Vadovas: *Doc. dr. Vitas Grimaila* *2015-02-23*
*(pareigos, vardas, pavardė, parašas)* *(data)*

KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Vaidas Kriščeliūnas

Telekomunikacijos (612H64001)

Baigiamojo projekto „Programų skirtų išmaniojo telefono aplikacijų kūrimui analizė“

**AKADEMINIO SĄŽININGUMO DEKLARACIJA**

20 ___ m. _____ ____ d.
Kaunas

Patvirtinu, kad mano **Vaido Kriščeliūno** baigiamasis projektas tema „Programų skirtų išmaniojo telefono aplikacijų kūrimui analizė“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

*(vardą ir pavardę įrašyti ranka)* *(parašas)*

Kriščeliūnas, V. Programų skirtų išmaniojo telefono aplikacijų kūrimui analizė. Telekomunikacijų inžinerijos baigiamasis projektas / vadovas doc. dr. Vitas Grimaila; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, Telekomunikacijų katedra.

Kaunas, 2015. 45 psl.

## SANTRAUKA

*Darbo tikslas - išanalizuoti programas skirtas išmaniųjų telefonų aplikacijų kūrimui ir sukurti aplikaciją.*

*Užduoties analizės dalyje buvo aptartas temos aktualumas, atlikta literatūros analizė, buvo iškelti darbo tikslai ir uždaviniai.*

*Analitinėje dalyje buvo analizuojama Android OS ir jos struktūra. Atlikta programų skirtų išmaniųjų telefonų aplikacijų kūrimų analizė, kurios metu aptartas jų naudojimo sudėtingumas, išmaniųjų įrenginių funkcijų palaikymas ir aplikacijos sugeneravimas.*

*Projektinėje dalyje sukurta mobili aplikacija, naudojantis viena iš analizuotų programų. Aplikacijoje įgyvendintos vietinio tinklo įrenginių paieškos ir informacinių celės pranešimų priėmimo funkcijos.*

Kriščeliūnas, Vaidas. Analysis of Smartphone Application Development Software. Final project of Telecommunication Engineering / supervisor doc. dr. Vitas Grimaila; Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, department of Telecommunications.

Kaunas, 2015. 45 psl.

## SUMMARY

*The aim of the thesis is to analize smartphone application development software and to develope smartphone application.*

*Object analysis consists of thesis topic relevance considerations, review of literature and thesis aim and goals were formed.*

*Analysis part consists of android platform and its structure analysis, analysis of smartphone development software. Considerations of aplication development dificulty, smartphone interface support and aplication deployment in software.*

*Practical part of aplication developement with one of the analyzed software programs. Created aplication implements local network device search function and recieving of cell broadcast messages.*

# TURINYS

# SANTRUMPŲ IR ŽENKLŲ AIŠKINIMO ŽODYNAS

| | |
|---|---|
| OS | Operacinė sitema. |
| API | aplikacijų programavimo sąsaja (angl. *Application Programming Interface*). |
| Ping | tai programa, testuojanti, ar užduotas tinklo įrenginys pasiekiamas.(angl. *Packet InterNet Groper*) |
| XML | duomenų struktūrų ir jų turinio aprašomoji kalba (angl. *Extensible Markup Language*) |
| QML | JavaScript programavimo kalba paremta vartotojo sąsajos sąsajos programavimo kalba (angl. *Qt Modeling Language*) |
| Android SDK | Programinės įrangos paketas Android aplikacijoms kurti (angl. *Android Software Development Kit*) |
| Android NDK | programinės įrangos paketas Android leidžiantis naudoti C ir C++ programavimo kalbas kuriant Android aplikacijas (angl. *Android Native Development Kit*) |
| Wi-Fi | belaidžio ryšio technologija |
| ARP | protokolas susiejantis IP ir MAC adresus (angl. *Address Resolution Protocol*) |
| Ping | Internet paketų griebėjas (angl. *Packet InterNet Groper*) |
| OSI | ryšio protokolų, naudojamų ryšio ir kompiuteriniuose tinkluose, aprašymas (angl. *Open Systems Interconnection Reference Model*) |
| DNS | Srities vardų struktūra (angl. *Domain Name System*) |
| ICMP | Internet valdymo žinučių protokolas (angl. *Internet Control Message Protocol*) |
| Shell | komandinės eilutės sąsaja |

# ĮVADAS

Šiuo metu technologijos plečiasi ypač sparčiai, tai galima lengvai pastebėti mobiliųjų įrenginių rinkoje. Vis plečiantis mobiliųjų įrenginių techninei įrangai kartu su jomis atsinaujino ir mobiliųjų įrenginių operacinės sistemos. Pirmosios operacinės sistemos buvo uždaros, ir jos nebuvo pritaikytos naujų aplikacijų įdiegimui. Laikui bėgant gamintojai pradėjo įdiegti vis daugiau funkcijų į savo įrenginius, tai buvo vykdoma įdiegiant naujas aplikacijas. Pirmosios aplikacijos buvo elementarūs arkadiniai žaidimai, telefonų melodijų kūrėjas, kalkuliatorius, kalendorius.

Lūžis mobiliųjų aplikacijų rinkoje įvyko atsiradus naujoms mobiliųjų įrenginių operacinėms sistemos, tokioms kaip Apple iOS, Android, Windows Mobile, Symbian, RIM. Šių operacinių sistemų pagrindinis privalumas yra jų atvirumas, jas nesudėtinga papildyti naujomis aplikacijomis. Tai paskatino aplikacijų kūrėjus kurti mobiliąsias aplikacijas išmaniesiems telefonams.

Šiame darbe bus aptarti pagrindiniai literatūros šaltiniai susiję su aplikacijų kūrimu, bus tiriamos programos, skirtos išmaniųjų telefonų aplikacijų kūrimui, bus sukurta aplikacija naudojant vieną iš tiriamų programų.

Darbo tikslas – atlikti programų skirtų išmaniųjų telefonų aplikacijų kūrimui analizę ir sukurti aplikaciją.

Darbo uždaviniai:

- Išanalizuoti Android OS struktūrą
- Išanalizuoti programas, skirtas išmaniųjų telefonų aplikacijų kūrimui
- Programos išsirinkimas aplikacijos kūrimui
- Aplikacijos kūrimas, naudojantis pasirinkta programa
- Sukurtos aplikacijos testavimas

# 1 UŽDUOTIES ANALIZĖ

Atliekama užduoties analizė, kurioje aptariamas pasirinktos temos aktualumas, atliekama objekto ir literatūros analizė

## 1.1 Temos aktualumas, objekto analizė

Šiuo metu mobiliosios aplikacijos yra ypač populiarios ir užima didžiąją dalį programinės įrangos rinkos, tai lėmė didelis mobiliųjų įrenginių populiarumas ir lengvas aplikacijų prieinamumas. Šiame darbe bus nagrinėjamos programos, kuriomis galima kurti aplikacijas išmaniesiems telefonams. Kadangi šiuo metu Android OS yra pati populiariausia ir lengviausiai prieinama OS mobiliųjų įrenginių rinkoje, bus nagrinėjamos programos, kurių sukurtos aplikacijos veikia Android OS įrenginiuose.

Darbo metu bus analizuojamos MIT App Inventor 2 , Qt Creator ir Android Studio programos, bus aptartas programų naudojimo sudėtingumas, aplikacijų testavimo galimybės, Android aplikacijų paketo generavimas.

Išanalizavus programas bus pasirinkta programa, kuri bus naudojama aplikacijos kūrimui, bus atsižvelgiamą į aplikacijos kūrimo sudėtingumą ir Android API sąsajų palaikymą programose.

Naudojantis pasirinkta programa bus sukurta Android OS aplikacija, kurioje bus įgyvendintos *ping* ir ARP lentelės nuskaitymo funkcijos, bei gyventojų perspėjimo pranešimų priėmimo funkcijas.

### 1.1.1 Aplikacijos funkcijų aprašymas

#### *Ping*

Ping (angl. xx) tai programa, skirta tikrinti įrenginių prieinamumui IP tinkle. Programa siunčia ICMP pranešimo užklausą nurodomam IP adresui, jei nurodytas įrenginys yra pasiekiamas, gavęs ICMP pranešimo užklausą įrenginys siunčia atsakymą. Jei užklausą išsiuntęs įrenginys gauną atsakymą, esame tikri kad nurodytas IP adresas yra pasiekiamas.

Ping funkcija išmatuoja laiko tarpą tarp išsiustos užklausos ir gauto atsakymo, ir gali būti naudojama skirtingo dydžio paketų siuntimui, taip išsamiau atlikdama tinklo diagnostiką. [11]

#### *ARP*

ARP protokolas yra skirtas antrojo ir trečiojo OSI lygmenų apjungimui. Tai padaroma

sudarant adresų lentelę, kurioje susiejami trečiojo OSI lygmens įrenginių adresai (pvz. IP adresas) su antrojo OSI lygmens įrenginių adresais (pvz. MAC adresas).

Android įrenginyje ARP lentelė yra sudaroma dinamiškai, ją sudaro pats įrenginys. Lentelės informacija yra saugoma įrenginio atmintyje ir yra atnaujinama. Tikslus ARP lentelės atnaujinimo laikas priklauso nuo įrenginio, tačiau dažniausiai lentelės duomenys yra atnaujinami greičiau nei per 5 minutes. [12]

#### *Informaciniai celės pranešimai*

Informaciniai celės pranešimai (angl. *cell broadcast*) technologija yra mobiliesiems įrenginiams skirta technologija leidžianti žinutes transliuoti visiems mobiliesiems įrenginiams esantiems celės ryšio zonoje. Ši technologija yra palaikoma daugelio mobiliųjų tinklų operatorių. Technologija yra sukurta vienalaikiam žinučių perdavimui daugeliui vartotojų nurodytoje teritorijoje. Lyginant su SMS žinutėmis, kurios yra skirtos vieno vartotojo žinutės siuntimui vienam arba keliems vartotojams, *cell broadcast* žinutės yra siunčiamos iš vieno vartotojo daugumai. Šie pranešimai yra siunčiami mobiliųjų tinklų operatorių iš nurodytų celių. [10,13]

## 1.2 Literatūros analizė. Darbo tikslai ir uždaviniai

Android OS yra sparčiai besivystanti ir nuolat atnaujinama, todėl labiausiai buvo remiamasi internetiniais literatūros šaltiniais, kadangi jie yra lengviau atnaujinami. Toliau pateikiami pagrindiniai literatūros šaltiniai kuriuose galima rasti naudingos informacijos apie Android OS ir aplikacijų kūrimą:

### 1.2.1 „Android Developers“ puslapis

Tai oficialus Android OS aplikacijų kūrėjams sukurtas puslapis, jis yra prižiūrimas Google kompanijos. Puslapis yra padalintas į kelias dalis: dizaino, kūrimo ir platinimo. Kiekvienoje dalyje yra suteikiama informaciją apie skirtingas aplikacijos kūrimo pakopas. Šiame darbe mes nenaudosime dizaino ir programos platinimo elementų, todėl dėmesį suteikėme į aplikacijos kūrimo dalį.

Šiame skyriuje pateikiami mokomieji gidai, API sąsajų gidai, paketų aprašymai, įrankiai reikalingi aplikacijų kūrimui, Google paslaugų aprašymai ir pavyzdžiai. Mokomuosiuose giduose pateikiamos pamokos kaip susidoroti su pagrindiniais uždaviniais kuriant aplikacijas. Kiekviena tema suskirstyta į kelias dalis, suteikiamos naudingos nuorodos apie aptariamus aplikacijos elementus. API sąsajų giduose pateikiami detalūs aprašymai, kaip susidoroti su įvairiais uždaviniais kuriant aplikacijas. Paketų aprašymuose pateikiamas sąrašas Android OS

naudojamų paketų, kuriuos gali naudoti aplikacijų kūrėjas savo aplikacijose. Yra pateikiami atskiri sąrašai kiekvienai Android OS versijai, juose rasime paketui priklausančių objektų aprašymus, jų naudojamas funkcijas ir kintamuosius. Įrankiuose yra pateikiami programiniai paketai reikalingi Android OS aplikacijų kūrimui ir jų aprašymui. Google paslaugose rasime paslaugų aprašymus, kuriuos pateikia Google. Aprašymuose nurodoma, kaip galima šias paslaugas panaudoti savo aplikacijose. Taip pat pateikiami aplikacijų pavyzdžiai, kuriuos galima studijuoti ir pamatyti, kaip Android OS objektai yra naudojami aplikacijose. [1]

### 1.2.2 „Operacinių sistemų architektūros“

Knygoje aprašomi įvairūs OS tipai, mobiliosios OS, virtualizacija, debesų kompiuterija, tinklo įrenginių OS, paskirstytos sistemos. Toliau atkreipsime dėmesį į mobiliąsias operacines sistemas, nes kituose skyriuose pateikta informacija nėra susijusi su mūsų darbo tema.

Šiame skyriuje pateikiamas mobiliosios OS aprašymas, nurodomos populiariausios mobiliosios OS, aprašoma Android OS. Aprašyme pateikta OS architektūros sandara, pagrindiniai jos elementai, saugumas ir įkrovos procesas. [2]

### 1.2.3 „Qt Creator“ programos dokumentacijos puslapis

Internetinėje svetainėje yra sukurta Qt Creator programos kūrėjų. Čia rasime programos aprašymus, skirtingas programos versijas, gidus pradedantiesiems, kuriuose yra programos instaliavimo gidai ir pavyzdinės aplikacijos. Aprašomi ir Qt programos įrankiai, pateikiamos dokumentacijos Qt programos C++ klasėms, QML tipams, Qt programos moduliams, kur detaliai aprašomas šių programos elementų veikimas.[3]

Programos dokumentacijoje pateikiama ir atskira skiltis mobiliųjų aplikacijų kūrėjams. Nurodomas sąrašas Qt programos elementų, kurie gali būti naudojami Android OS ir nurodomi pagrindiniai uždaviniai kurie yra įgyvendinami naudojant Qt Creator programą. Suteikiami papildomi aprašymai, kuriuose rasime informacijos apie[3]:

- Reikalinga programinę įrangą norint kurti aplikacijas Android OS
- Sukurtos aplikacijos perkėlimą į įrenginį.
- Papildomos „OpenSSL“ bibliotekos (biblioteka naudojama užtikrinti saugiems įrenginio sujungimams) įdiegimą į aplikaciją.
- Papildomų Java programavimo kalbos bibliotekų įdiegimą į aplikaciją.
- Aplikacijos paketo sugeneravimą.
- Aplikacijų publikavimą „Google Play“ internetinėje aplikacijų parduotuvėje.

- Aplikacijos pavyzdžių
- Pastabos kuriant aplikacijas naudojant Qt Creator programą.

### 1.2.4 „MIT App Inventor 2“ programos puslapis

Internetinė svetainė yra pateikta MIT App Inventor 2 programos kūrėjų. Svetainėje rasime aprašymų skiltį, kur rasime programos aprašymą, naujienas ir išteklių skiltį, kurioje pateikiama [4]:

- Gidas pradedantiesiems, kuriame pateikiamos instrukcijos kaip paruošti mobilųjį įrenginį arba emuliatorių, aprašomi pagrindiniai programos blokai (programavimo elementai naudojami programoje), elementarių mobiliųjų aplikacijų kūrimo gidas ir aplikacijos platinimo galimybės.
- Programos išteklius, kur pateikta visa mokomoji medžiaga, medžiaga yra surūšiuota pagal skirtingus kriterijus. Sudėtingumo lygį, medžiagos tipą (knyga, straipsnis, gidas ar kita), tematika (matematika, fizika, informatika ar kita).
- Skyrius norintiems mokinti kitus, kaip kurti aplikacijas naudojant App Inventor 2 programą, pateikiamas klasės paruošimo gidas, bei literatūra, kuria rekomenduojama remtis paskaitose ar pamokose.
- Dokumentacija, kurioje pateikiami blokų aprašymai, kurie naudojami aplikacijų programavimui, aplikacijų šablonai, komponentų dokumentacija.

### 1.2.5 „Vogella“

Šiame internetiniame puslapyje pateikiama: nemokami gidai, paslaugos, produktai, knygos.

Giduose pateikiamas platus spektras gidų, ne tik Android programuotojams, bet ir Eclipse programinės įrangos įrankiams, internetinių puslapių kūrimą ir publikavimą, standartinį Java programavimą, Google paslaugų panaudojimą, algoritmus ir kt.

Android OS giduose yra pateikiami gidai aprašantys Android aplikacijų kūrimą pradedantiesiems, aplikacijų kūrimo įrankius, testavimą, pagrindinių Android OS elementų panaudojimą (pasaugas (angl. *services*), transliacinių pranešimų priėmėją (angl. *Broadcast Receiver*), vartotojo pranešimų valdiklį (angl. *Notification Manager*), fragmentus ir kt,

Šiame internetiniame puslapyje yra didelė gausa gidų Android programuotojams, pateikti gidai apima labai platų temų spektrą. Pateikti gidai yra plačiai aprašyti, pateikiami programinio kodo pavyzdžiai ir nuorodos į papildomus informacijos šaltinius.[14]

### 1.2.6 „StackOverFlow“

Šis internetinis puslapis yra „Q&A“ stiliaus, vartotojai gali patiekti savo klausimus į kuriuos gali atsakyti kiti puslapio lankytojai. Vartotojai gali vertinti klausimus ir atsakymus pagal informatyvumą, bei klausimą uždavęs vartotojo pasirinktas teisingas atsakymas yra pažymimas, taip kiti svetainės lankytojai gali žinoti, kurie atsakymai buvo vertingiausi ir kuris atsakymas padėjo išspręsti minėtą problemą.

Puslapyje yra aptariamos tokios temos kaip Android, Java, C ir C++, Eclipse, Qt, ir kt. Puslapyje skirtas tiek pradedantiesiems tiek pažengusiems programuotojams, todėl yra sprendžiamos įvairios problemos, taip padeda ir pradedantiesiems programuotojams, kadangi į jų klausimus gali atsakyti ir profesionalūs programuotojai.[15]

### 1.2.7 „XDA Developers“ ir „XDA University“

„XDA Developers“ puslapyje yra gausu informacijos susijusios su Android įrenginių administratoriaus teisių atrakinimas (angl. *root*), bei papildomos programinės įrangos įdiegimu, bei jos naudojimų Android telefonuose. Puslapyje yra gausu straipsniu, bei forumas, kuriame galima rasti dažniausiai aptariamas problemas, bei idėjas aplikacijų kūrėjams.[16]

„XDA University“ taip pat gausu informacijos susijusios su Android įrenginių administratoriaus teisių atrakinimu (angl. *root*), bei naujos programinės įrangos įdiegimu. Tačiau čia rasite ir papildomu gidu, naudojantiems Android Studio ar Eclipse programas, bei „Android“ biblioteka.[17]

Abiejuose puslapiuose yra didelis kiekis informacijos tiek pradedantiesiems tiek pažengusiems vartotojams.

### 1.2.8 Darbo tikslas ir uždaviniai

Darbo tikslas – atlikti programų skirtų išmaniųjų telefonų aplikacijų kūrimui analizę ir sukurti aplikaciją.

Darbo uždaviniai:

- Išanalizuoti Android OS struktūrą
- Išanalizuoti programas, skirtas išmaniųjų telefonų aplikacijų kūrimui
- Programos išsirinkimas aplikacijos kūrimui
- Aplikacijos kūrimas, naudojantis pasirinkta programa
- Sukurtos aplikacijos testavimas

# 2 ANDROID OS IR APLIKACIJŲ KŪRIMO PROGRAMŲ ANALIZĖ

Prieš analizuojant programas, skirtas išmaniųjų telefonų aplikacijų kūrimui bus aptarta Android OS ir jos struktūra struktūrą.

## 2.1 Android OS

Android OS – tai Google kompanijos kuriama OS, paremta Linux OS branduoliu. Šiuo metu Android OS yra populiariausia OS mobiliuosiuose įrenginiuose, tai lėmė OS atvirumas, prieinamumas ir lankstumas.

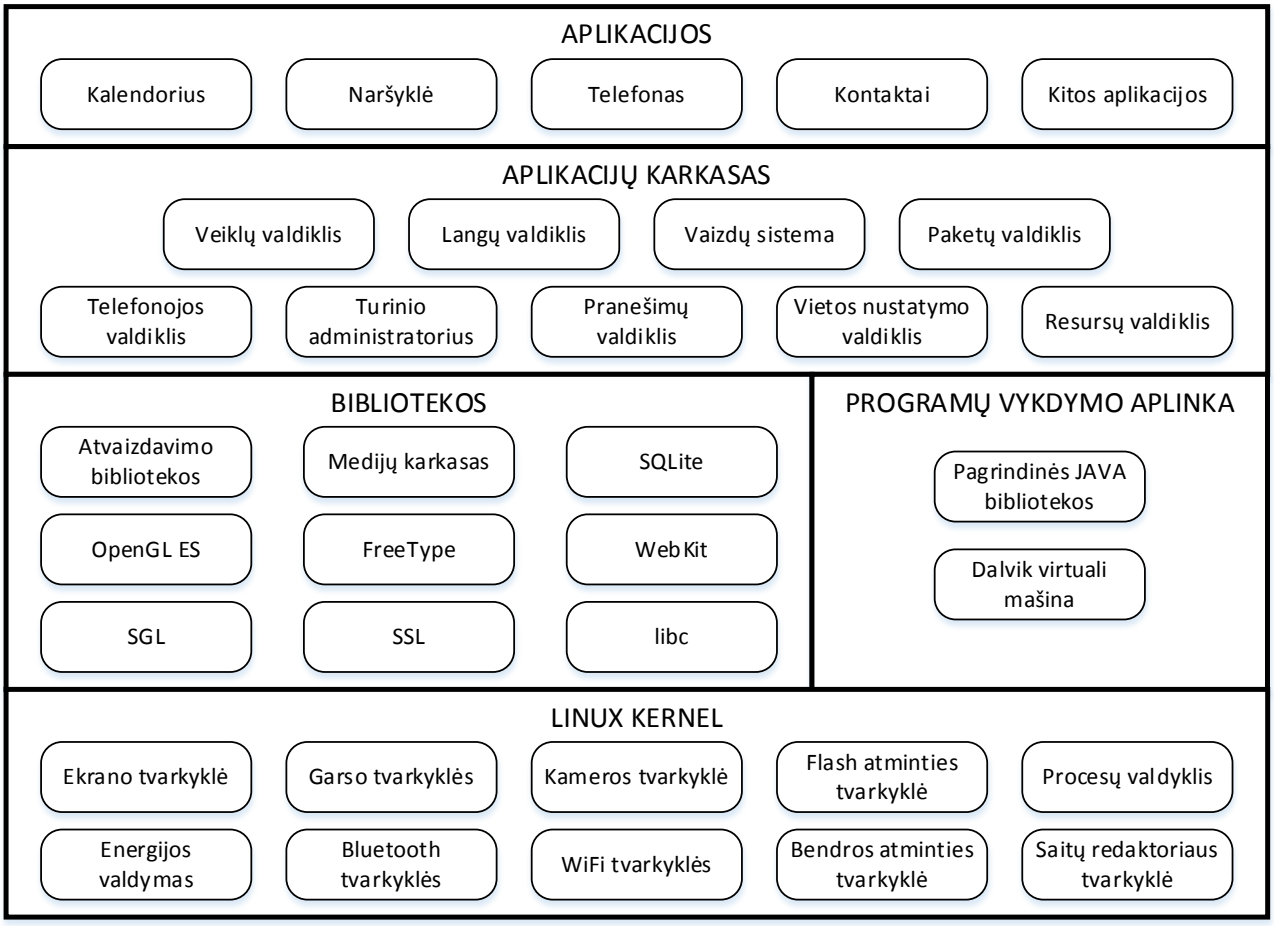
Android OS yra atvira kodo, tai reiškia kad originalus programos kodas yra laisvai prieinamas, jį galima peržiūrėti, modifikuoti ir sukurti savo OS versijas. Tai yra pagrindinė priežastis lėmusi Android OS privalumą, kadangi mobiliųjų įrenginių gamintojai gali modifikuoti OS ir pritaikyti ją savo įrenginiams.

Android atvirumas taip pat paspartino ir programinės įrangos vystymąsi, kadangi programų kūrėjai galėjo modifikuoti OS kodą, daug daugiau programuotojų prisijungė prie OS vystymo. Atsirado modifikuotų OS versijų (tokių kaip „CyanogenMod“, „Android Open Kang Project“, „MIUI“, „Slim ICS“, „Liquid Smooth ROMS“). Šiose OS versijose buvo įdiegta daug papildomų funkcijų, kai kurios iš jų buvo net įdiegtos naujesnėse oficialiose OS versijose. [18]

Android OS lanksti, ji gali būti pritaikoma skirtingo dydžio ir net skirtingo tipo įrenginiams, tokiems kaip išmaniesiems telefonams, planšetiniams kompiuteriams, išmaniesiems laikrodžiams ir kt.

## 2.2 Android architektūros analizė

Android OS architektūra yra sudaryta iš skirtingų programinės įrangos komponentų, kurie gali būti skirstomi į penkias kategorijas, kaip pavaizduota 2.1 pav. Toliau bus aptariamas kiekvienas iš struktūros komponentų.




2.1 pav. Android OS architektūros stekas [2]

### 2.2.1 Linux OS branduolys

Operacinės sistemos pagrindą sudaro Linux Kernel. Svarbu paminėti, kad tai nėra Linux OS, o tik jos pagrindas. Android OS naudojamas Linux Kernel branduolys yra optimizuotas veikti mobiliose sistemose, jis yra padarytas kompaktiškesnis siekiant užtikrinti spartų veikimą įrenginiuose turinčiuose ribotus duomenų apdorojimo ir atminties resursus. Kiekvienoje Android OS versijoje yra naudojama skirtingos Linux Kernel versijos, 2.1 lentelėje pateikiamos Linux Kernel versijos Android OS versijose, tačiau naudojama branduolio versija gali kisti skirtingų gamintojų įrenginiuose.

2.1 lentelė. Linux Kernel versijos Android OS versijose [5]

| Linux Kernel versijos | Android versijos |
|---|---|
| 2.6.* | 1.5-3.2.6 |
| 3.0.* | 4.0-4.1.2 |
| 3.4.* | 4.2-5.1.1 |

Viena iš pagrindinių priežasčių dėl kurių yra naudojama Linux Kernel yra galimybė aptarnauti kelis vartotojus vienu metu. Šiuo atveju vartotojais yra laikomos paleidžiamos aplikacijos, kurioms priskiriami vartotojo identifikavimo numeriai. Numeriai yra žinomi tik pačios OS, todėl aplikacijos negali jų koreguoti. Priskirti vartotojo numeriai yra naudojami

leidžiamų duomenų nustatymui, aplikacijos gali naudoti tik duomenis, kurie yra būtini jos veikimui. Kiekvienas procesas yra paleidžiamas atskiroje virtualioje mašinoje, taip aplikacijos veikia nepriklausomai nuo kitų aplikacijų.[6]

Linux Kernel branduolyje yra įdiegtos tvarkyklės, kurios susieją įrenginio techninės įrangos elementus su programine vartotojo įranga. Tvarkyklės yra modifikuojamos įrenginių gamintojų, dėl skirtingų elementų naudojimo. Todėl Android OS gali kisti, priklausomai nuo įrenginio kuriame yra naudojama.[2]

### 2.2.2 Programų vykdymo aplinka

Programų vykdymo aplinką atsakinga už aplikacijų paleidimą, ją sudaro pagrindinės Java programavimo kalbos bibliotekos ir Dalvik virtualia mašina (programinė įranga veikianti įrenginyje, galinti paleisti aplikacijas). Programų vykdymo aplinka veikia virš OS branduolio ir naudojo jo pagrindines funkcijas. Atlieka sąsajos su technine įranga ir aplikacijų atminties valdymo funkcijas.

Pateikiamos Java programavimo kalbos bibliotekos skiriasi nuo standartinių Java bibliotekų, tačiau jos suteikia daugumą funkcijų aprašytų standartinėse Java programavimo kalbos bibliotekose.[8]

Dalvik virtuali mašina yra modifikuota Java virtualios mašinos versija, ji yra optimizuota veikti ribotų resursų sistemose, tokiose kaip išmanieji telefonai. Vienas iš pagrindinių šios virtualios mašinos skirtumų nuo standartinių Java virtualiųjų mašinų yra skirtingo tipo failų nuskaitymas. Standartinės Java virtualiosios mašinos naudoja *.class tipo failus, tuo tarpu Dalvik naudoja *.dex. Dalvik naudojami failai yra kompaktiškesni, todėl sunaudoja mažiau atminties resursų.[9]

Android OS „KitKat“ versijoje buvo pristatyta nauja virtuali mašina vadinama ART (angl. *Android Runtime*), paliekant Dalvik virtualiąją mašiną kaip numatytąją virtualiąją mašiną programų vykdymui. Android 5.0 „Lolipop“ versijoje Dalvik virtuali mašina buvo visiškai pakeista ART virtualia mašina.[1]

Pagrindiniai ART virtualios mašinos privalumai:

- Aplikacijų kodo konvertavimas į mašininį kodą instaliuojant aplikacijas, tai leidžia padidinti aplikacijų efektyvumą ir sumažinti galios išteklius.
- Patobulinta šiukšlių surinkimo sistema, kuri sumažina padidina sistemos veikimo efektyvumą.

Kaip jau minėta kiekviena aplikacija yra paleidžiama atskirtoje virtualioje mašinoje, tai

turi kelis privalumus. Dėl to kiekviena aplikacija veikia atskiroje aplinkoje, taip jos netrukdo kitoms aplikacijoms, operacinei sistemai ir yra atskirtos nuo techninės įrangos. Tai suteikia aplikacijoms nepriklausomumą nuo techninės įrangos, todėl ta pati aplikacija gali veikti skirtingą techninę įrangą turinčiuose Android įrenginiuose.[2]

### 2.2.3 Bibliotekos

Pateikiamos bibliotekos atskirtų įrenginio elementų valdymui, 2.2 lentelėje pateikti kelių bibliotekų pavyzdžiai. Šios bibliotekos yra aprašytos C ir C++ programavimo kalba ir veikia tame pačiame lygmenyje kaip ir programų vykdymo aplinka, tačiau jos gali būti papildomos, todėl įrenginių gamintojai gali įdėti papildomų bibliotekų savo įrenginio valdymui.

2.2 lentelė. Android OS architektūros steko bibliotekų aprašymai [2, 7]

| Biblioteka | Aprašymas |
|---|---|
| SQLite | Duomenų bazės biblioteka, naudojama aplikacijų informacijai saugoti |
| WebKit | Interneto naršyklės biblioteka, naudojama internetinių svetainių atvaizdavimui aplikacijose |
| SGL | Biblioteka naudojama 2D grafikos elementų atvaizdavimui |
| OpenGL ES | Biblioteka naudojama 3D grafikos elementų atvaizdavimui |

### 2.2.4 Aplikacijų karkasas

Aplikacijų karkasas (angl. *Application Framework*) pateikia sistemines bibliotekas, funkcijas ir sąsajas naudojamas aplikacijų komponentų valdymui. Šiuos komponentus naudoja ir standartinės aplikacijos, todėl aplikacijų kūrėjai gali kurti aplikacijas ne tik papildančias jau esamas aplikacijas, bet ir pakeičiančias jau esamas aplikacijas. 2.3 lentelėje pateikiami aplikacijų karkaso komponentų pavyzdžiai.[2]

2.3 lentelė. Aplikacijos karkaso komponentai [9]

| Komponentas | Aprašymas |
|---|---|
| Veiklų valdiklis (angl. *Activity Manager*) | valdo aplikacijų gyvavimo ciklą |
| Vaizdų sistema (angl. *View System*) | leidžia kurti programų vartotojo sąsają |
| Paketų valdiklis (angl. *Package Manager*) | sistema leidžianti aplikacijoms sužinoti apie jau įrašytas aplikacijas |
| Resursų valdiklis (angl. *Resource Manager*) | suteikia prieigą prie aplikacijos išteklių (audio ar video failų, papildomų duomenų failų) |
| Vietos nustatymo sistema (angl. *Location Manager*) | atlieka vietos nustatymo funkcijas naudojant įrenginio GPS įrenginiu ar informacija iš bazinės stoties |
| Pranešimų valdiklis (angl. *Notification Manager*) | leidžia programoms leidimą sudaryti informacinius pranešimus vartotojui (aplikacijos atnaujinimo ir informaciniai pranešimai) |

2.3 lentelės tęsinys [9]

| Komponentas | Aprašymas |
|---|---|
| Telefonijos valdiklis (angl. *Phone Manager*) | leidžia naudotis telefonijos funkcijomis (skambučiai, bei SMS žinutės) |
| Turinio administratorius (angl. *Content Provider*) | leidžia aplikacijoms prieiti prie kitų aplikacijų duomenų arba dalintis savo duomenimis. |

### 2.2.5 Aplikacijos

Šiame lygmenyje yra visos vartotojo įdiegtos aplikacijos, Android OS standartinės aplikacijos ir įrenginių gamintojų aplikacijos. 2.4 lentelėje pateikti aplikacijų komponentai, kurie yra naudojami aplikacijoje.

2.4 lentelė. Aplikacijų komponentai [1]

| Komponentas | Aprašymas |
|---|---|
| Veiklos (angl. *Activities*) | komponentas naudojamas ekrano vaizdo užpildimui su vartotojo sąsaja. Vienoje aplikacijoje gali būti ne viena veikla ir jos gali atidaryti ir kitų aplikacijų veiklas, jei aplikacijai buvo suteiktas toks leidimas. |
| Paslaugos (angl. *Services*) | komponentas naudojamas atlikti ilgai trumkančioms operacijoms, kurios bus atliekamos antrajame plane. Šie komponentai nesuteikia vartotojo sąsajos. |
| Turinio administratoriai (angl. *Content providers*) | naudojamas dalintis aplikacijos duomenimis arba leisti aplikacijai prieiti prie kitų duomenų. Duomenis galima saugoti failų sistemoje, SQLite duomenų bazėje, internete ar kitoje duomenų saugykloje, kurią aplikacija gali pasiekti. Naudojantis šiuo komponentu aplikacijos gali ne tik perskaityti per ir koreguoti duomenis, jei aplikacijai buvo suteikti šie leidimai) |
| Transliacijos gavėjai (angl. *Broadcast receivers*) | šie komponentai reaguoja į transliuojamus pranešimus. Šie pranešimai gali būti ir pačios sistemos sugeneruoti pranešimai (kraunamo telefonono ar senkančios įrenginio baterijos pranešimas), šiuos pranešimus gali generuoti ir pačios aplikacijos, taip jos gali pranešti kitoms aplikacijoms apie tam tikrus įvykius. Šie komponentai nenaudoja vartotojo sąsajos, tačiau gali sukurti pranešimus (Notifications), kurie informuotu vartotoją apie gautus pranešimus (elektroninio pašto aplikacijos). |

## 2.3 Programų skirtų išmaniojo telefono programų kūrimui analizė

Šiame skyriuje bus tiriamos programos, skirtos išmaniųjų telefonų aplikacijų kūrimui, bus aprašomos programos aplikacijų kūrimo ir testavimo įrankiai, aplikacijos paketo (*.apk) generavimas ir papildomos funkcijos.

### 2.3.1 „MIT App Inventor 2“

Programa buvo sukurta MIT universitete, ir dabar yra toliau tobulinama MIT universiteto, kartu bendradarbiaujant su „Google“. Ši programa išsiskiria tuo, kad ji pati veikia

kaip puslapio aplikacija, ja galima naudotis per interneto naršykle, todėl ji yra prieinama įvairioms OS platformoms, aplikacijas galima kurti netgi naudojantis Android įrenginiu.[4]

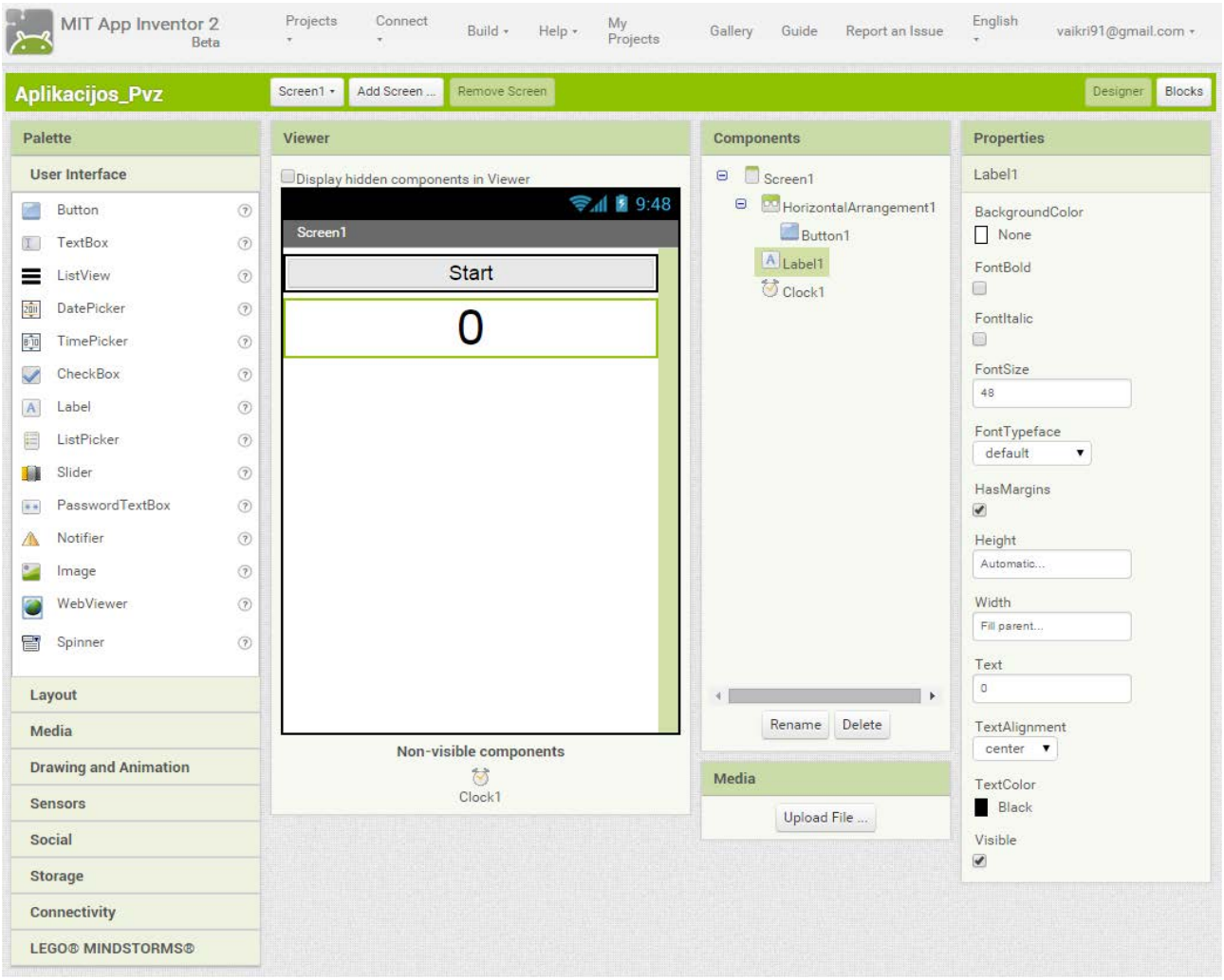
Aplikacija yra sukurta siekiant supaprastinti Android OS aplikacijų kūrimą, tam yra naudojamas aplikacijų programavimas naudojantis blokus, todėl norint kurti aplikacijas nereikia žinoti Java ar kitų programavimo kalbų, užtenka žinoti tik pagrindines programavimo kalbų funkcijas, tokias kaip „if“, „for“,“and“, „or“ ir kt.

Dėl savo paprastumo ši programa naudojama mokyklose ir universitetuose, kadangi ji puikiai tinka norint išmokti Android OS aplikacijų struktūrą, bei pagrindinius elementus.

### *Aplikacijos kūrimas*

Aplikacijų kūrimas naudojant MIT App Inventor yra labai supaprastintas, kuriant naują aplikaciją programa nesiūlo jokių aplikacijos šablonų, vartotojas visą programą sudaro pats. Yra keli aplikacijų kūrimo rėžimai, tai aplikacijos dizaino kūrimo rėžimas (2.2 pav.) ir blokų programavimo rėžimas (2.3 pav.).

Dizaino kūrimo režime sudaroma vartotojo sąsaja ir pasirenkami visi elementai kurie bus naudojami aplikacijoje. Programos siūlomi elementai yra jau suprogramuoti, todėl reikia tik pasirinkti jų norima vietą ekrane (jei naudojami matomi elementai). Visi elementai yra pateikti elementų paletėje (angl. *Palette*), kurioje elementai yra suskirstyti į kelias klases. Šios elementų klasės yra patektos 2.5 lentelėje.




2.2 pav. MIT App Inventor vartotojo sąsajos kūrimo langas

2.5 lentelė. MIT App Inventor vartotojo sąsajos kūrimo elementai [4]

| Elementų klasė | Aprašymas |
|---|---|
| Vartotojo sąsajos (angl. *User Interface*) | elementai vartotojo sąsajai sudaryti (Mygtukai, teksto laukeliai, pavadinimai, sąrašų elementas, datos ir laiko pasirinkimo elementai ir kt.). |
| Išdėstymo elementai (angl. *Layout*) | elementai padedantis išdėstyti kitus elementus norima tvarka, šie elementai yra nematomi, jie veikia kaip atskiri konteineriai, kuriuose sudėjus elementus jie ekrane atvaizduojami atitinkama tvarka (sustatomi į stulpeli, eilutes ar lenteles). |
| Medijos elementai (angl. *Media*) | medijos failų elementai galintys groti muzikinius failus, valdyti telefono vibraciją, atvaizduoti video failus, teksto skaitymo elementai, galintys perskaityti tekstą (veikia tik anglų kalba), kameros naudojimo elementai, video ir audio įrašymo elementai. |
| Piešimo ir animacijos elementai (angl. *Drawing and Animation*) | elementai naudojami atvaizduoti pagrindini grafikos ekraną, kuriame galima patalpinti įvairius grafinius elementus (dažniausiai naudojami žaidimams) |
| Jutikliai (angl. *Sensors*) | sąsajos su išmaniojo telefono jutikliais (akselerometru, orientacijos jutikliu, NFC sąsaja, Bar kodų skaitytuvas, vietos nustatymo jutikliai, laikrodis ir kiti) |
| Socialiniai komponentai (angl. *Social*) | sąsajos su telefono komunikacijų funkcijomis, tokiomis kaip žinučių siuntimas, skambinimas, informacijos dalinimasis įvairiuose socialiniuose tinkluose. |

2.5 lentelės tęsinys

| Elementų klasė | Aprašymas |
|---|---|
| Saugyklos (angl. *Storage*) | elementai naudojami informacijos įrašymui ir nuskaitymui į failus, duomenų bazę ar internetinę duomenų bazę. |
| Sujungimas (angl. *Connectivity*) | sąsajos su telefono bluetooth įrenginiu, HTTP funkcijų elementai |
| „LEGO MINDSTORMS“ | atskiri elementai sukurti „LEGO MINSTORMS“ robotukų valdymui. |

Aplikacijos grafinė sąsaja yra matoma peržiūros laukelyje (angl. *Viewer*), čia rodomi visi matymo elementai, vartotojas gali keisti jų vietą juos perstumdamas.

Komponentų (angl. *Components*) laukelyje parodyti visi elementai kurie jau yra įdėti į aplikaciją, bei jų hierarchija.

Savybių (angl. *Properties*) laukelyje nurodomos elemento savybės, kurias vartotojas gali pakeisti. Rodomos savybės priklauso nuo įdėto elemento.


2.3 pav. MIT App Inventor blokų programavimo langas

Blokų programavimo režime sudaroma aplikacijos logika, aprašomas elementų veikimas. Programa pati pateikia galimas funkcijas kurias gali atlikti atitinkami elementai, vartotojas gali sukurti atskirus blokus savo kintamiesiems. Taip pat programa pateikia papildomus operacinius blokus kurie yra pateikiami blokų (angl. *Blocks*) paletėje. Blokų elementai yra pateikti 2.6 lentelėje.

2.6 lentelė. MIT App Inventor programavimo blokų elementai

| Elementų klasė | Aprašymas |
|---|---|
| Kontrolės (angl. *Control*) | pateikiami „if“, „for“ ir „while“ ir panašių funkcijų blokai. |
| Logikos (angl. *Logic*) | pateikiamos „and“ ir „or“ funkcijos, taip pat „true“ ir „false“ logikos kintamieji |
| Matematikos (angl. *Math*) | pateikiami matematikos funkcijų blokai. |
| Teksto apdorojimo (angl. *Text*) | pateikiami teksto apdorojimo funkcijų blokai. |
| Sąrašų apdorojimo (angl. *List*) | pateikiami blokai sąrašų sudarymui. |
| Spalvų (angl. *Color*) | pateikiami spalvų sudarymo blokai |
| Kintamųjų (angl. *Variable*) | pateikiami naujų kintamųjų sudarymo, bei kintamųjų reikšmės nuskaitymo ir nustatymo blokai. |
| Funkcijų (angl. *Procedures*) | pateikiami blokai naujų funkcijų sudarymui. |

Peržiūros (angl. *Viewer*) laukelyje galime peržiūrėti sudėtą aplikacijos blokų struktūrą, bei ją redaguoti.

### *Aplikacijos testavimas*

Testuoti aplikacijoms „App Inventor2“ programa suteikia kelis variantus, tai aplikacijų testavimas Android emuliatoriuje arba realiame Android įrenginyje. Norint testuoti aplikacijas reikalingas papildomas programos paketas, kuris sudaro sąsaja su Android emuliatoriumi ar realiu įrenginiu.

Didelis programos privalumas yra gyvo testavimo galimybės, jei testuojant aplikacija pastebime įsivėlusią klaidą ją galime iš karto ištaisyti ir aplikacijos nereikia iš naujo įkelti į įrenginį. Tai pasiekiama papildomos įdiegus papildoma programinę įranga kompiuteryje ir Android įrenginyje. Taip pat yra siūlomi keli variantai įrenginio sujungimui su asmeniniu kompiuteriu, tai galima atlikti naudojant USB arba Wi-Fi sąsajas.

### *Aplikacijos paketo generavimas*

Programa pateikia kelis variantus aplikacijos paketo sugeneravimui. Paketą galima parsisiųsti į įrenginį, kuriuo prisijungta prie programos svetainės, arba programa gali patalpinti programą savo internetiniame serveryje dviem valandoms ir pateikią QR kodą, kurioje užkuoduotą nuorodą į programos parsiuntimo svetainę, aplikacijos parsisiuntimų skaičius nėra

ribojamas.

### *Papildomos funkcijos*

„App inventor2“ programa taip pat pateikia aplikacijų galerija, kurioje talpinamos įvairios vartotojų aplikacijos. Aplikacijas galima pridėti prie savo aplikacijų ir peržiūrėti jų blokų sandarą. Juos taip pat lengva naudoti savo aplikacijose, kadangi galime tuos pačius programinius blokus panaudoti savo aplikacijose tik pakoreguodami funkcijų kintamuosius.

### *Apibendrinimas*

App Inventor 2 programa labai supaprastina aplikacijų kūrimą ir yra rekomenduojamas aplikacijų kūrimo įrankis pradedantiesiems. Programa leidžia greitai sukurti aplikacijas panaudojant jau programoje aprašytais įrankiais, kurie leidžia lengvai panaudoti sudėtingesnius Android OS elementus. Taip pat programavimas blokais stipriai sumažina padarytų klaidų skaičių programuojant (išvengiama sintaksės klaidų ir klaidingai aprašytų funkcijų).

Programa leidžia lengvai platinti sukurta aplikacija tiek ją parsisiunčiant, tiek panaudojant pačios programos internetinius serverius. Pateikiamos „gyvo“ testavimo funkcijos sutrumpina aplikacijos testavimo laiką, kadangi kiekvieną karta koregavus aplikaciją, jos nereikia iš naujo įkelti į įrenginį. Didelis pavyzdinių programų kiekis, kuriuose galima surasti aplikacijų kuriuos sėkmingai panaudojo mūsų aplikacijai reikalingus elementus.

Tačiau blokų programavimas turi ir savų trūkumų, kadangi daugelis veikimo elementų jau yra aprašyti, vartotojas jų negali koreguoti. Kuriant aplikacijas nėra galimybės pasirinkti tikslios Android OS versijos, todėl sukurta aplikacija gali veikti nevienodai skirtingose Android OS versijose. Programa pateikia tik vieną Android OS emuliatoriaus tipą, todėl nėra galimybės patikrinti aplikacijos veikimo įvairiose Android OS versijose.

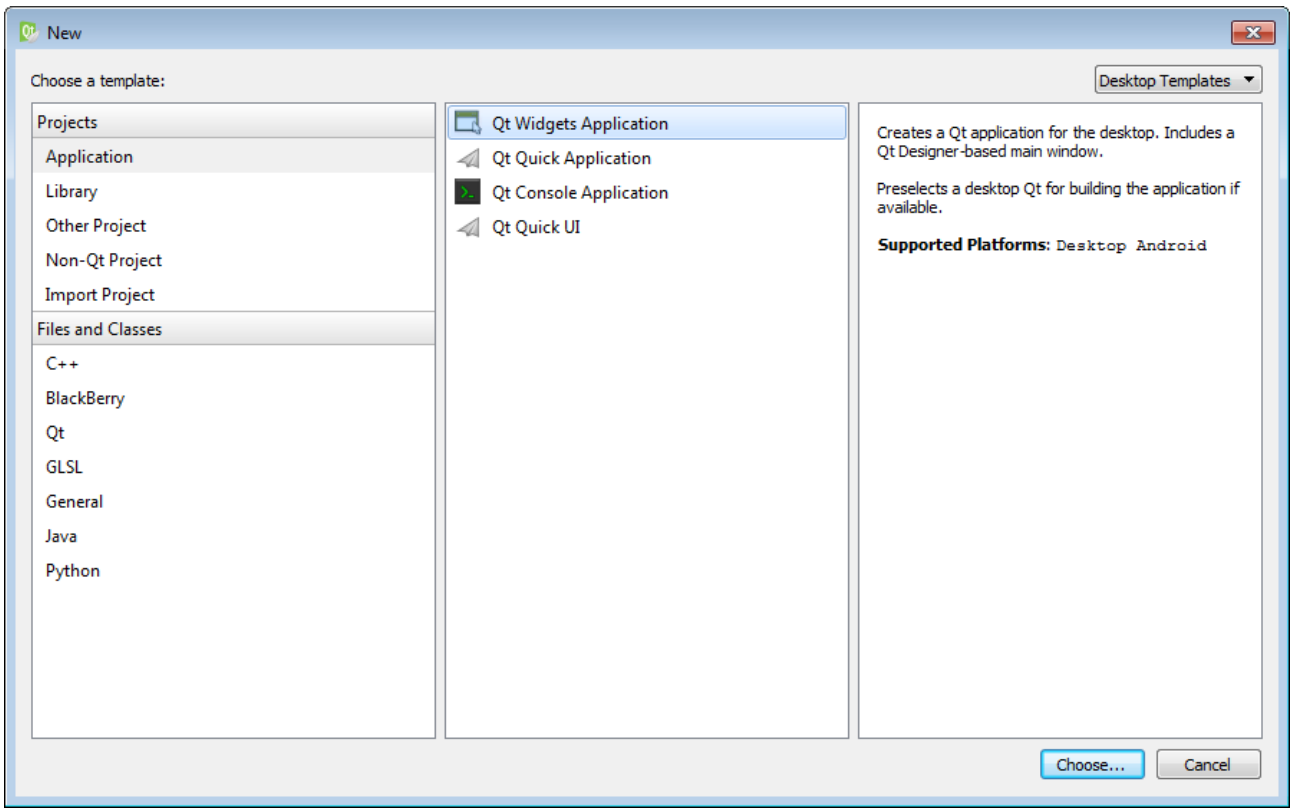
## 2.3.2 „Qt Creator“

Qt Creator programa yra skirta įvairių OS aplikacijų kūrimui, naudojant šia programa galima kurti aplikacijas tiek įvairioms išmaniųjų telefonų platformoms, tokioms kaip Apple iOS, Android, BlackBerry ar Symbian, bei namų kompiuterių platformoms Microsoft Windows, Mac OS ir Linux. Pagrindinis programos pranašumas yra galimybė panaudoti tą pati aplikacijos kodą ir programinę sąsają skirtingoms OS platformoms. [3]

Pagrindinė programavimo kalba naudojama programoje yra C++, bei QML, tai programavimo kalba naudojama programų vartotojo sąsajai kurti, joje aprašomi vartoto sąsajos atributai ir gali būti aprašomos aplikacijos funkcijos naudojant javascript programavimo kalbos funkcijas.

Norint pateikti aplikacijas Android OS sistemai reikalingi papildomi programiniai paketai, kadangi ši programa nėra specializuota Android aplikacijų kūrimui. Programa naudoja Android SDK (leidžia naudoti Android API sąsajas) ir Android NDK (leidžia naudoti C++ programavimo kalbą Android aplikacijose) programinius paketus.

### *Aplikacijos kūrimas*




2.4 pav. Qt Creator aplikacijos tipo pasirinkimo langas

Norint kurti aplikacijas išmaniesiems telefonams programa siūlo kelis šablonus, tai yra „Qt Widgets Application“ ir „Qt Quick Application“.

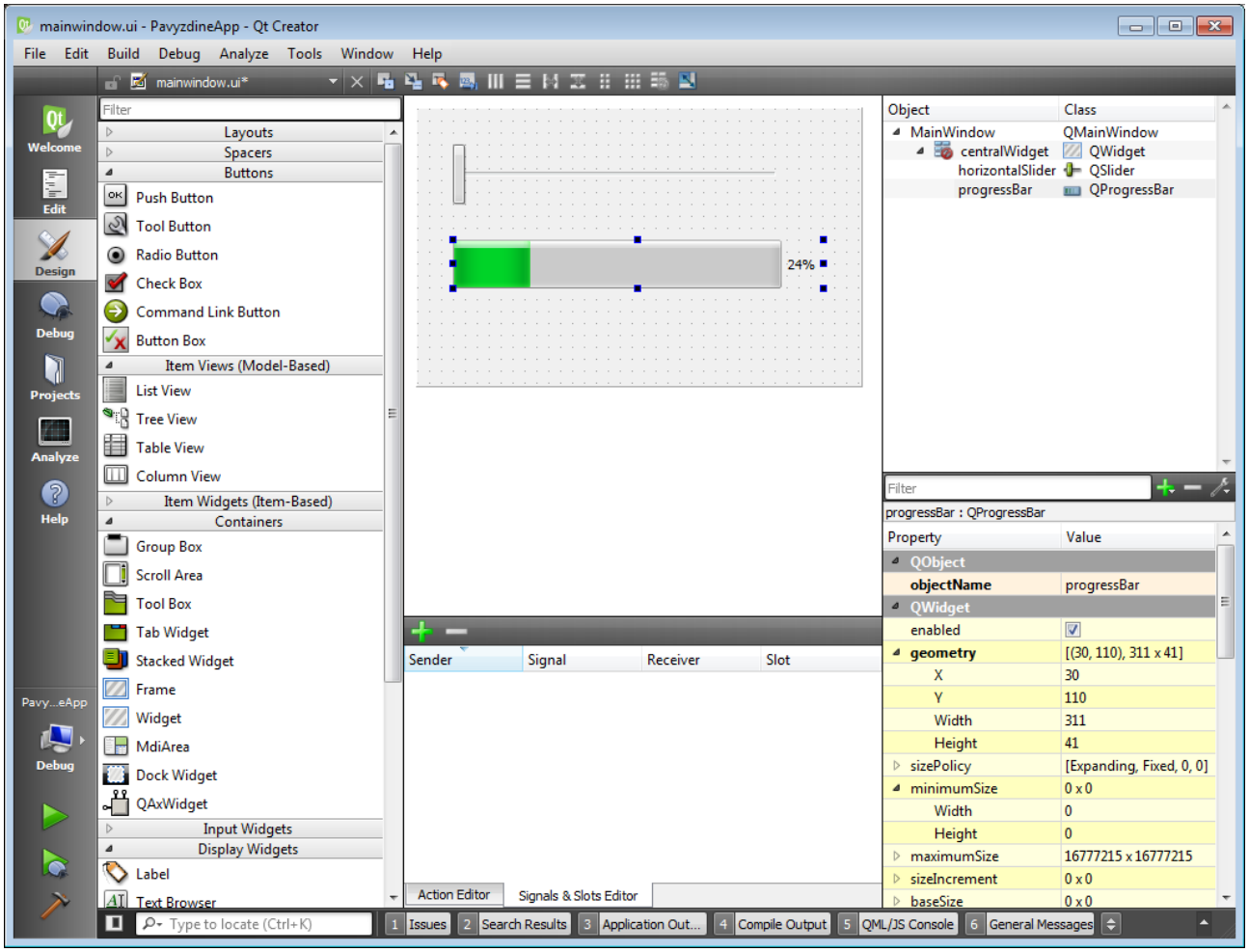
Pagrindiniai skirtumas tarp pateikiamų aplikacijos kūrimo šablonų yra vartoto sąsajos kūrimas, naudojant „Qt Widgets Application“ šabloną pateikiamas „Drag and Drop“ stiliaus vartotojo sąsajos kūrimo langas (2.5 pav.), kuriame pateikiami įvairūs programos aprašyti valdikliai, su savo aprašytomis funkcijomis. Valdikliai pateikiami 2.7 lentelėje.

2.7 lentelė. MIT App Inventor vartotojo sąsajos kūrimo elementai

| Valdikliai | Aprašymas |
|---|---|
| Išdėstimo (angl. *Layout*) | Naudojami elementų grafiniam išlygiavimui |
| Atskyrimo elementai (angl. *Spacers*) | naudojami sudaryti fiksuotam atstumui tarp elementų. |
| Mygtukai | pateikiama keletas mygtuku variantų įvairioms funkcijoms atlikti |
| Kuriamu elementu atvaizdavimo elementai (angl. *Item Views*) | leidžia atvaizduoti aplikacijoje sukurtus elementus tam tikra tvarka. |

2.7 lentelės tęsinys.

| Valdikliai | Aprašymas |
|---|---|
| Elementų valdikliai (angl. *Item Widgets*) | leidžia paskirstyti vartotojo sukurtus valdiklius norima tvarka. |
| Konteineriai | naudojami sugrupuoti elementus |
| Duomenų įvedimo elementai | naudojami įvairių tipų duomenų įvedimui. |
| Duomenų išvedimo elementai | Naudojami įvairių tipų duomenų išvedimui |

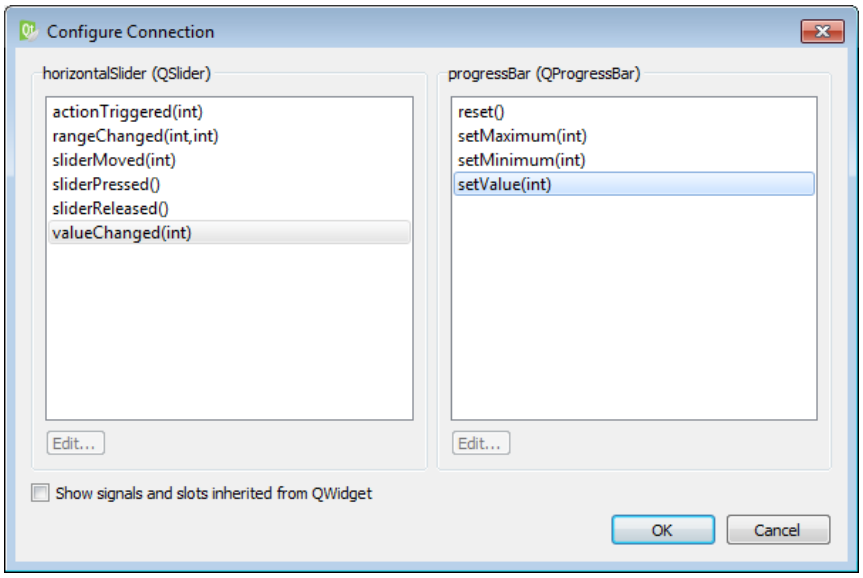



2.5 pav. Qt Creator vartotojo sąsajos kūrimo langas

Programoje pateikiamas ir vartotojo sąsajos struktūros sandara, kuriame galime peržiūrėti visus elementus naudojamus kuriamoje vartotojo sąsajoje. Taip pat pateikiamas ir elementų atributų aprašymo langas, kuriame galime peržiūrėti ir koreguoti įvairius elementų atributus.

Qt Creator programa, pateikia naudingą elementų sujungimo funkciją, kuri leidžia nesudėtingai sujungti elementų veikimą (2.6 pav.). Programa leidžia tai atlikti dizaino režime ir programavimo režime, tai naudinga jei norime sujungti elementus kurie nėra matomi ar naudojame savo sukurtas funkcijas (pavyzdys pateiktas žemiau).

```
connect(ui->horizontalSlider,SIGNAL(valueChanged(int)),ui-
>progressBar,SLOT(setValue(int)));
```




2.6 pav. Qt Creator elementų sujungimo langas

Šis programavimo būdas leidžia sujungti elementu funkcijas nenaudojant plataus jų aprašymo.

Naudojant „Qt Quick Application“ šabloną, vartotojo sąsaja yra kuriama naudojant Qt Quick modulį (2.7 pav.), elementai gali būti sukurti naudojant dizaino arba programavimo režimus. Programavimo režime naudojama QML kalba, kuria aprašomi elementai ir jų atributai, taip pat gali būti naudojamos JavaScript programavimo kalbos funkcijos. Šiame šablone taip pat yra pateikiami jau aprašyti programų valdikliai, kuriuos vartotojas gali panaudoti iškart, tai yra:

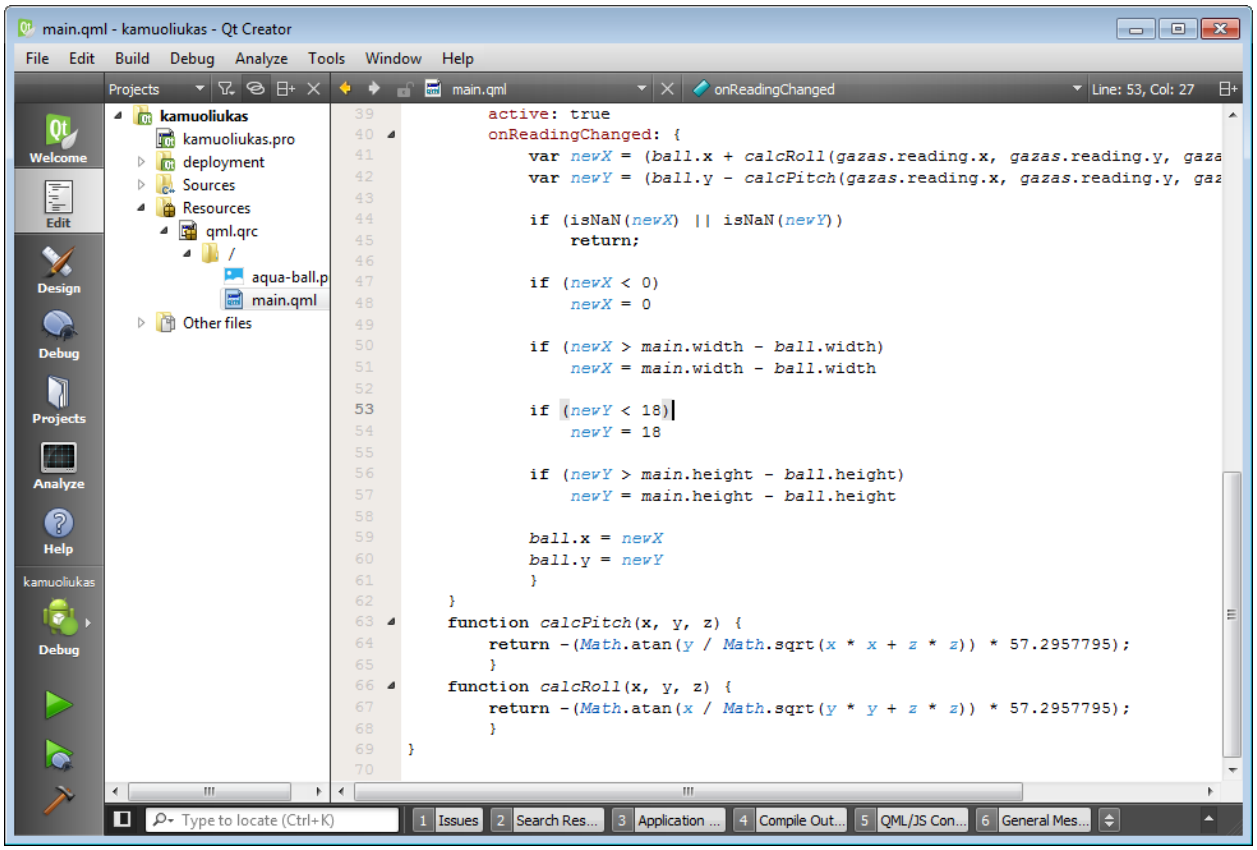
- Pagrindiniai elementai, struktūriniai elementai pagrindinėms vartotojo funkcijoms atlikti
- Valdymo elementai, mygtukai, pažymėjimo laukeliai ir kiti
- Išdėstymo elementai
- Poziciniai elementai
- Peržiūros elementai

Šioje sąsajoje pateikiami sukurtos vartotojo sąsajos peržiūra, kurioje matome elementus naudojamus vartotojo sąsajoje, taip pat pateikiama vartotojo sąsajos elementų struktūra, bei elemento savybių langas.

Šiame režime taip pat galima daryti anksčiau minėtus Qt programos elementų sujungimus, tačiau jie yra tik aprašomi programavimo sąsajoje.

Šio rėžimo pagrindiniai pranašumas yra lankstumas, vartotojas pats gali sukurti savo elementus ir priskirti jiems reikiama funkcionalumą. Elementų pozicijos aprašymui galime naudoti matematines funkcijas, tai leidžia elementus atvaizduoti sąlyginai kitiems objektams

pagal nustatytus santykius, ir net keisti jų atributus atsižvelgiant į kitų objektų padėtį. Modulis taip pat leidžia keisti elemento atributus mūsu norima tvarka, leidžia nesudėtingai aprašyti elementų animacijos elementus, kintant jo atributams (spalvai, dydžiui, pozicijai, orientacijai). Galima naudoti JavaScript funkcijas, kurios suteikia patildomo funkcionalumo, ir net leidžia išvengti aplikacijos programavimo naudojant C++ programavimo kalba.




2.7 pav. Qt Creator Qt Quick aplikacijos kūrimo langas

Programavimo lange galime aprašyti aplikacijos logika, tam naudojama C++ arba QML (jeigu nodojame Qt Quick modulį) programavimo kalba. Qt programa pateikia daug savo aprašytų C++ klasių, dėl to aplikacijų kūrimas yra šiek tiek paprastesnis, tačiau šie programos aprašyti moduliai ir klasės naudojami tik šioje programoje ir nors jie yra sukurti kaip analogai standartinėms C++ programavimo kalbos klasėms ir sąsajoms, jie turi ir savo specifinių savybių. Tačiau tai apsunkina aplikacijų projektų pasidalinimą, kadangi norint peržiūrėti aplikacijos projektą reikalinga Qt Creator programa arba Qt programinis paketas naudojamoje programavimo aplinkoje, tačiau jis nėra prieinamas visose programavimo aplinkose.

### *Programos testavimas*

Aplikacijų testavimas gali būti atliekamas arba virtualiame Android emuliatoriuje arba savo Android įrenginyje, jį prijungus prie kompiuterio naudojant USB sąsają. Aplikacijas veikimą galima testuoti ir nustačius aplikacijos dislokaciją namų kompiuteriams, tačiau taip

nebus galima išnaudoti telefono API sąsajų, bei naudojamų elementai gali atrodyti skirtingai.

Virtualus Android emuliatorius yra sukuriamas naudojant Android SDK paketą. Galima testuoti aplikacijos veikimą įvairiose jau paruoštuose Android emuliatorių šablonuose, kurie atitinką įvairius įtaisus ir galima sukurti naują emuliatorių. Tačiau šioje programoje reikia atsižvelgti į emuliatoriaus naudojamą procesorių. Kadangi dislokuojant aplikacijas naudojami skirtingi aplikacijų generavimo būdai skirtingiems įrenginių procesoriams, todėl reikalingas suderinamumas. Tai taip pat reikalinga ir testuojant aplikacijas realiuose įrenginiuose.

#### *Aplikacijos paketo generavimas*

Norint sugeneruoti aplikacijos paketa reikalingas papildomo kodo įdiegimas, ir Android OS skirtų failų sugeneravimas. Tai reikalinga norint kad aplikacija naudotų tam tikrus įrenginio resursus, šie resursai yra neprieinami aplikacijoms, jei jie nėra nurodomi AndroidManifest.xml faile. Instaliuojant aplikacija įrenginyje vartotojas yra informuojamas, kokius įrenginio resursus naudos ši aplikacija.

Aplikacijos sukurtos naudojant Qt Creator programą naudoja savo papildomas Qt bibliotekas, kurios nėra standartinės ir nėra naudojamos Android OS, todėl naudojama papildoma Qt programa Android OS, vadinama Ministro, kuri nustato kurių Qt bibliotekų reikės mūsų įrenginyje, pagal įrašytas aplikacijas kurios buvo sukurtos naudojant Qt Creator programą.

#### *Apibendrinimas*

Qt Creator programa yra universali, kadangi ji suteikia galimybę kurti aplikacijas ne tik Android OS, bet ir kitoms OS platformoms, šis įrenginys labai naudingas norint sukurti kelias aplikacijos versijas skirtingoms OS. Tačiau kadangi šis įrenginys nėra specializuotas išmaniųjų telefonų aplikacijų kūrimui, aplikacijų kūrimas yra sudėtingesnis, nei kitose programose. Programai reikalingi papildomi programiniai paketai (Android SDK ir Android NDK), sudėtingas Android aplikacijos paketo generavimas.

Aplikacijos elementų sujungimas ir papildomi Qt programos moduliai ir C++ programavimo kalbos klasės supaprastina programavimą, tačiau šios funkcijos reikalauja savų bibliotekų. Todėl norint dislokuoti aplikacijas mobiliuose įrenginiuose į juos reikia įdiegti papildomas bibliotekas, kad sukurtos aplikacijos veiktų įrenginyje. Tai reikalinga tiek Android OS sistemose, tiek namų įrenginių sistemose. Nors ši problema sprendžiama įdiegiant papildomą programą Android OS, tačiau tai vis tiek apsunkina jos įdiegimą.

### 2.3.3 „Android Studio“

Android Studio yra oficiali programų kūrimo aplinka, skirta Android OS aplikacijų

kūrimui. Ši programa yra sukurta Google kompanijos ir yra nemokama. Programa yra paremta Intellij IDEA Java programų kūrimo aplinkos varikliu. Programos veikimui reikalingi papildomi programinės įrangos paketai, tokie kaip Android SDK ir Java Developement Kit. Ši programa veikia ant Windows OS, Linux ir Mac OS X platformų.

Papildomos programos funkcijos [1]:

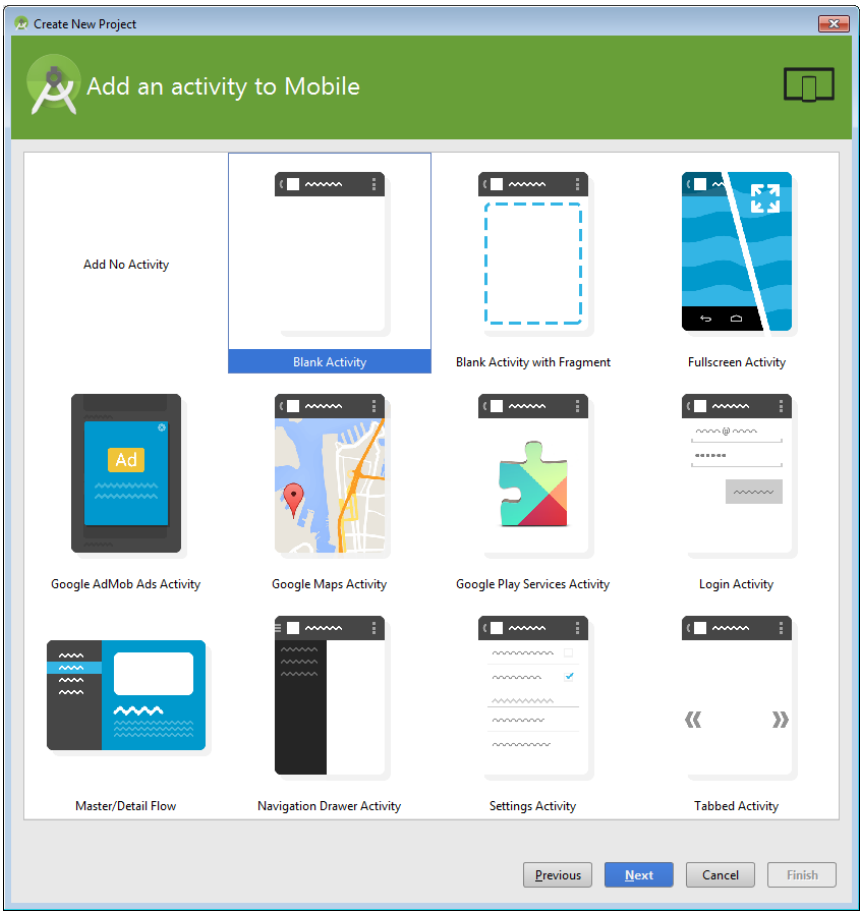
- Aplikacijų generavimo sistema Gradle, kuri leidžia sukurti skirtingus tos pačios aplikacijos variantus, neperašant programos kodo.
- Aplikacijų kodo pavyzdžiai padedantys sukurti aplikacijos pagrindus
- Vartotojo sąsajos kūrimo variklis suteikiantis „Drag and Drop“ kūrimo funkcijas
- Idiegtas „Google Cloud“ platformos palaikymas

### *Aplikacijos kūrimas*

Kuriant aplikacijas Android OS turime daug pasirinkimų, galime kurti aplikacijas išmaniesiems telefonams ir planšetiniams kompiuteriams, išmaniesiems televizoriams, nešiojamiems Android įrenginiams (išmanieji laikrodžiai), bei įdiegus papildomą programinės įrangos paketą galime kurti aplikacijas Google Glass akiniams. Taip pat galime pasirinkti ir minimalia Android OS versija ant kurios veiks aplikacija, pasirinkus žemesnes Android OS versijas programa mus perspės jei naudosime elementus, kurie yra dar nėra įdiegti į mūsų pasirinktą OS versiją.

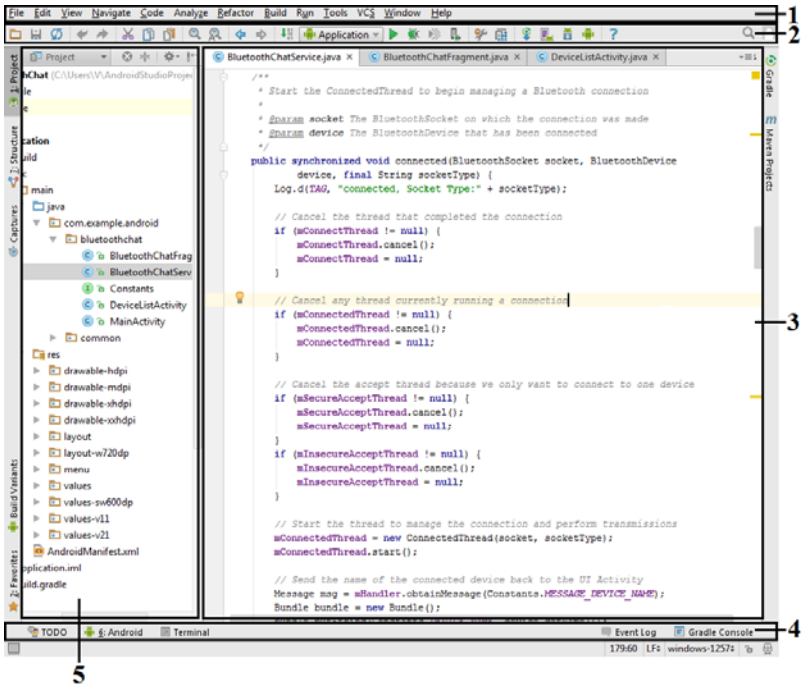
Programa taip pat suteikia galimybe pasirinkti ir galimus pradinio kodo šablonus įvairioms aplikacijoms (2.4 pav). Šablonai gali būti naudojami kaip programos pagrindas žinant kokią aplikaciją mes norime sukurti, taip pat jie gali pasitarnauti kaip geras pavyzdys, parodantis kaip galima panaudoti Android OS elementus.

Programa suteikia galimybe pasirinkti ir galimus pradinio kodo šablonus įvairiems aplikacijų tipams, šablonų pasirinkimo langas parodytas 2.8 pav. Šablonai gali būti naudojami kaip programos pagrindas žinant kokią aplikaciją mes norime sukurti, taip pat jie gali pasitarnauti kaip geras pavyzdys, parodantis kaip galima panaudoti Android OS elementus.




2.8 pav. Android Studio aplikacijos šablonų pasirinkimo langas

Pasirinkus norima aplikacijos veiklos šabloną galime įdėti į aplikacija savo norimus elementus, bei programuoti aplikacijos veikimą. 2.9 pav. pavaizduota programavimo aplinka, jos elementai aprašyti 2.8 lentelėje, tai yra pagrindinis programos langas kuriame aprašoma aplikacijos veikimas.




2.8 pav. Android Studio aplikacijos veikimo aprašymo langas

2.8 lentelė. Android Studio aplikacijos veikimo aprašymo lango elementai

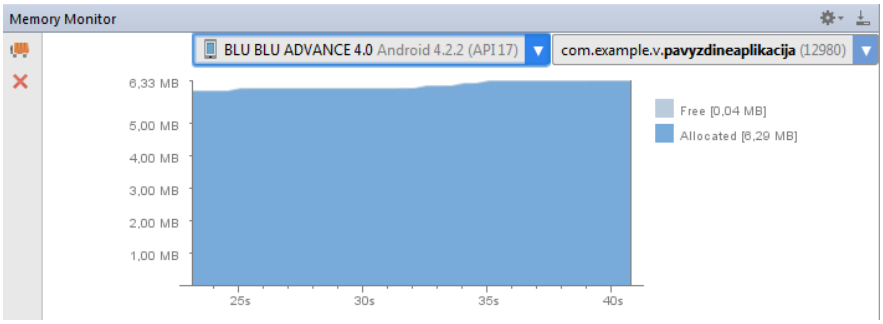
| Nr. | Elementas | Aprašymas |
|---|---|---|
| 1 | Menu | Programinės aplinkos nustatymai |
| 2 | Įrankių juosta | Programinės aplinkos įrankiai, greito priėjimo mygtukai prie SDK ir AVD įrankių |
| 3 | Teksto redaktorius | Leidžia vartotojui parašyti ir keisti programos tekstą |
| 4 | Android įrankiai | Rodomi prijungti įrenginiai, terminalo komandos, priėjimas prie „Grade“ konsolės |
| 5 | Projekto pelelė | Parodoma projekto struktūra, projekto esama vieta kompiuterio kietajame diske. |

Aplikacijos veiklos vartotojo sąsajai kurti naudojamas Android Studio kurimo variklis, kuriame aplikacijos dizainas gali būti kuriamas dviem būdais: dizaino kūrimas tekstiniame režime (2.7 pav) ir dizaino kūrimas „Drag and Drop“ režime. Kuriant tekstiniame režime naudojama XML programavimo kalba, kur aprašomi elementų tipai ir jų padėtis ekrane ir pagrindinės jų savybės. „Drag and Drop“ dizaino režime matome pagrindinius elementus kuriuos galime įtraukti į mūsų aplikacijos veiklą. Kurdami aplikacijos vartotojo sąsają padarytus pokyčius matome peržiūros lange (angl. *Preview*). Lange galima pasirinkti skirtingus įrenginių šablonus, taip galime pamatyti aplikacija bus atvaizduojama įrenginiuose su skirtinga rezoliucija. Dizaino kūrimo režime aprašoma tik vartotojo sąsajos elementai ir nėra aprašomas aplikacijos veikimas.

### *Aplikacijos testavimas*

Android Studio programa pateikia kelis būdus kaip galime testuoti savo sukurtas aplikacijas. Tai galima atlikti naudojant realų Android OS įrenginį arba Android emuliatorių.

Programas testuojant ant savo įrenginio naudojama USB sąsaja prijungti Android įrenginį prie kompiuterio. Programų leidimas savo įrenginyje yra greitesnis nei naudojant Android emuliatorių, kadangi jis sunkiai apkrauna kompiuterį, tačiau naudojant Android emuliatorius galime pasirinkti skirtingas Android OS versijas, kurios leidžia stebėti aplikacijų veikimą skirtingose Android versijose. Android Studio programa taip pat leidžia stebėti kiek atminties resursu sunaudoja mūsų aplikacija (2.9 pav).




2.9 pav. Android Studio aplikacijos sunaudojamos atminties langas

Android OS emuliatorius yra kuriamas naudojant Android SDK programinėje įrangoje pateiktus įrankius, kaip ir Qt Creator programoje. Tačiau Android Studio yra atlaidesnė, kadangi nereikalauja įrenginių procesorių suderinamumo, su šiomis problemomis programa susidoroja vartotojui nematant.

#### *Aplikacijos paketo generavimas*

Norint sugeneruoti savo aplikacijos paketą Android Studio programoje, programa reikalauja aplikacijų parašo. Šis parašas gali būti sugeneruojamas programoje. Šis parašas yra naudojamas aplikacijos autoriaus identifikavimui.

Turint aplikacijų parašą, aplikacijos gali būti sugeneruojamos. Aplikacijos paketas yra sugeneruojamas aplikacijos projekto aplanke.

#### *Papildomos funkcijos*

Programoje taip pat yra pateikiami įvairių aplikacijų pavyzdžiai, kuriuose galime peržiūrėti įvairių Android OS elementų įdiegimą į aplikaciją. Aplikacijų pavyzdžiai yra suskirstyti į kategorijas ir pateikiami trumpi jų aprašymai.

#### *Apibendrinimas*

Android Studio programa suteikia programuotojui daug naudingų įrankių kuriant aplikacijas, kadangi ši programa yra sukurta Android OS aplikacijų kūrimui, programa nereikalauja papildomų programinės įrangos paketų norint dirbti su programa, jie yra automatiškai įdiegiami kartu su programa. Ši programa leidžia patikrinti aplikacijų funkcionalumą skirtingose Android OS versijose, bei įrenginiuose. Programa leidžia sugeneruoti skirtingas aplikacijos paketų versijas, neperrašant programos kodo, tai yra naudinga aplikacijų kūrėjams norintiems išleisti kelias savo aplikacijos versijas „Google Play“ aplikacijų parduotuvėje (pvz. mokamas ir nemokamas versijas). Taip pat programa suteikia aplikacijų paketų pasirašymo funkcijas. Norint sukurti vartotojo sąsają nebūtina išmanyti XML programavimo kalbos, programos dizainas gali būti sukurtas naudojantis „Drag and Drop“ dizaino kūrimo režimu, tačiau norint aprašyti aplikacijos veikimą reikės Java programavimo kalbos žinių.

## 2.4 Programų pasirinkimo kriterijų analizė

Lyginant programas svarbiausi kriterijai į kuriuos atsižvelgsime yra Android OS API sąsajų palaikymas, bei aplikacijos kūrimo sudėtingumas. Programų naudojamų programavimo kalbų bei API sąsajų programinės įrangos paketas yra pateikiamas 2.9 lentelėje.

2.9 lentelė. Programų naudojama programavimo kalba ir naudojamas programinės įrangos paketas Android OS API sąsajoms.

| Programa | Programavimo kalba | Programinės įrangos paketas |
|---|---|---|
| App Inventor 2 | Programavimo blokai | - |
| Qt Creator | C++, JavaScript, QML | Android SDK, Android NDK |
| Android Studio | Java | Android SDK |

### 2.4.1 Aplikacijų kūrimo sudėtingumas

Iš analizuotų programų „App Inventor 2“ lengviausia perprasti ir naudoti, tačiau tai yra todėl, kad programoje mes turime mažiau kontrolės programuodami aplikacijos veikimą.

Qt Creator programoje pateikiami keli varianta aplikacijos veikimo aprašymui, galime programuoti aplikacijos veikimą naudodami QML ir JavaScript arba C++ programavimo kalbas. Nors turime daug pasirinkimo variantų, tačiau ne visi jie yra optimalūs, kadangi naudodami QML ir JavaScript programavimo kalbas aplikacijos veikimui mes esame apriboti, kadangi galime įdiegti tik robotą kiekį funkcijų. Programa leidžia įterpti papildomus modulius, kurie leidžia valdyti kai kuriuos išmaniojo įrenginio elementus (pvz. QtSensors), tačiau reikalingos papildomos Qt modulių žinios.

Aplikacijoms aprašyti galime naudoti ir C++ programavimo kalbą, tam yra naudojamas Android NDK programinės įrangos paketas, kuris leidžia naudoti C ir C++ programavimo kalbas aplikacijų kūrimui. Tačiau ne visi Android OS elementai gali būti aprašomi naudojant C ir C++ programavimo kalbas, todėl juos reikia aprašyti įterpiant Java programavimo kalbos modulius.

Nors programoje nėra būtina aprašyti aplikacijas pasirenkant tik vieną iš galimų aplikacijų kūrimo variantų, galima panaudoti juos visus aplikacijos kūrime, tačiau tai apsunkina aplikacijos kūrimo procesą ir padidina kuriamos aplikacijos sudėtingumą.

Android Studio programoje aplikacijos veikimui naudojama Java programavimo kalba, kuri yra standartinė programavimo kalba Android OS elementų aprašymui, todėl nereikalingas kelių programavimo kalbų aprašymas. Dėl šios priežasties, ši programa turi didžiausią vartotojų ratą, todėl informaciją apie elementų aprašymą yra lengviau prieinama ir yra daugiau šių elementų panaudojimo pavyzdžių aplikacijose.

### 2.4.2 Android API sąsajų palaikymas

App Inventor programa turi pagrindinių Android OS sąsajų palaikymą ir įdiegtą Google paslaugų palaikymą. Tačiau Android API sąsajų kiekis kurios yra prieinamos programoje yra ribotas ir programoje negalime įterpti papildomų sąsajų aprašymų.

Qt Creator ir Android Studio programos naudoja tą pati Android SDK programinį paketą, kuris suteikia priėjimą prie įvairių Android API sąsajų, tačiau nė visos jos yra prieinamos Qt

Creator programoje. Ši programa naudoja savo modulius API sąsajų aprašymui ir juos susieja su Android OS API sąsajomis, tačiau programoje nėra pateikiami moduliai visoms galimoms Android API sąsajoms.

### 2.4.3 Apibendrinimas

Dėl ankstesniuose skyriuose paminėtų priežasčių, buvo pasirinkta Android Studio. Programa naudoja standartinius Android API sąsajų aprašymus, todėl juos lengviau panaudoti savo aplikacijose, nes nereikia ieškoti papildomų modulių, kurie naudoją šias API sąsajas. Projektinėje darbo dalyje ši programa bus naudojama aplikacijos kūrimui.

# 3 PASIRINKTOS PROGRAMOS PANAUDOJIMAS IŠMANIOJO TELEFONO APLIKACIJOS KŪRIMUI

Kaip jau minėta praeitame skyriuje, aplikacijos kūrimui bus naudojama Android Studio programa. Naudojantis programa kuriamoje aplikacijoje bus realizuotos *ping* pranešimų siuntimo, ARP lentelės nuskaitymo ir perspėjimo pranešimų priėmimo funkcijos.

## 3.1 Aplikacijos sukūrimas

Kuriama aplikacija sudaryta iš dviejų dalių:

- Vietinio tinklo skanavimo – aplikacijoje bus panaudotos *ping* ir ARP lentelės nuskaitymo funkcijos. *Ping* pranešimai naudojami patikrinti įrenginių pasiekiamumą, sudarydami sujungimus Android OS dinamiškai užpildo ARP lentelę.
- Informacinių celės pranešimų priėmimas – aplikacijoje bus panaudotas *BroadcastReceiver* elementas, kuris bus naudojamas pranešimams priimti, gauti pranešimai bus išsaugoti Android *SQLite* duomenų bazėje

Kadangi naudosime ryšio funkcijas aplikacijai reikia suteikti papildomas prieinamų procesų privilegijas, tai atliekama *AndroidManifest.xml* faile. 3.1 lentelėje pateiktas komandų sąrašas papildomų privilegijų pridėjimui.

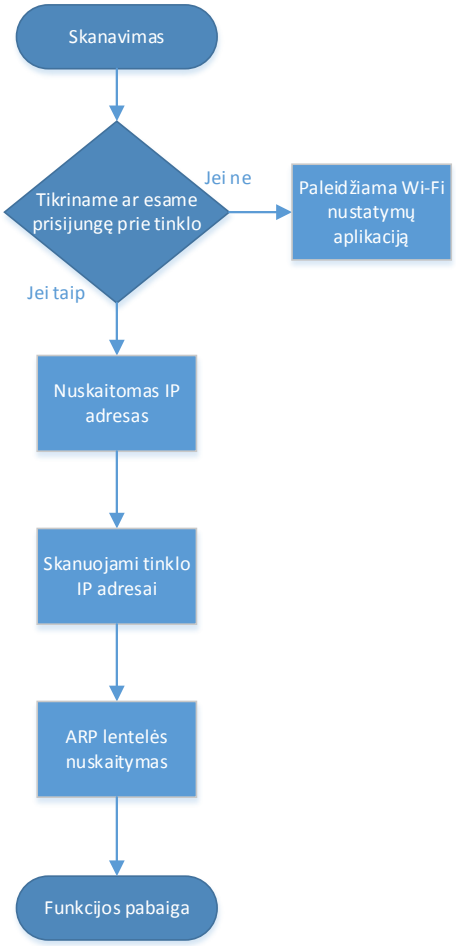
3.1 lentelė. *AndroidManifext.xml* failo privilegijų suteikimo komandos

| Suteikiamos privilegijos | Privilegijos suteikimo programinis kodas |
|---|---|
| Wi-Fi sąsajos būsenos priėjimas | *android.permission.ACCESS_WIFI_STATE* |
| Wi-Fi sąsajos būsenos keitimas | *android.permission.CHANGE_WIFI_STATE* |
| Tinklo būsenos priėjimas | *android.permission.ACCESS_NETWORK_STATE* |
| Interneto būsenos nuskaitymas | *android.permission.INTERNET* |
| Žinučių priėmimas | *android.permission.RECEIVE_SMS* |

Toliau bus aprašomos kiekvienos dalies naudojamos funkcijos

### 3.1.1 Vietinio tinklo įrenginių paieška

Siekiant atlikti tinklo įrenginių paiešką, aplikacijoje panaudosime *ping* funkciją ir ARP lentelę, kuri yra sudaroma pačio Android įrenginio. 3.1 pav. pateikiamas aplikacijos algoritmas.




3.1 pav. Vietinio tinklo skanavimo algoritmas

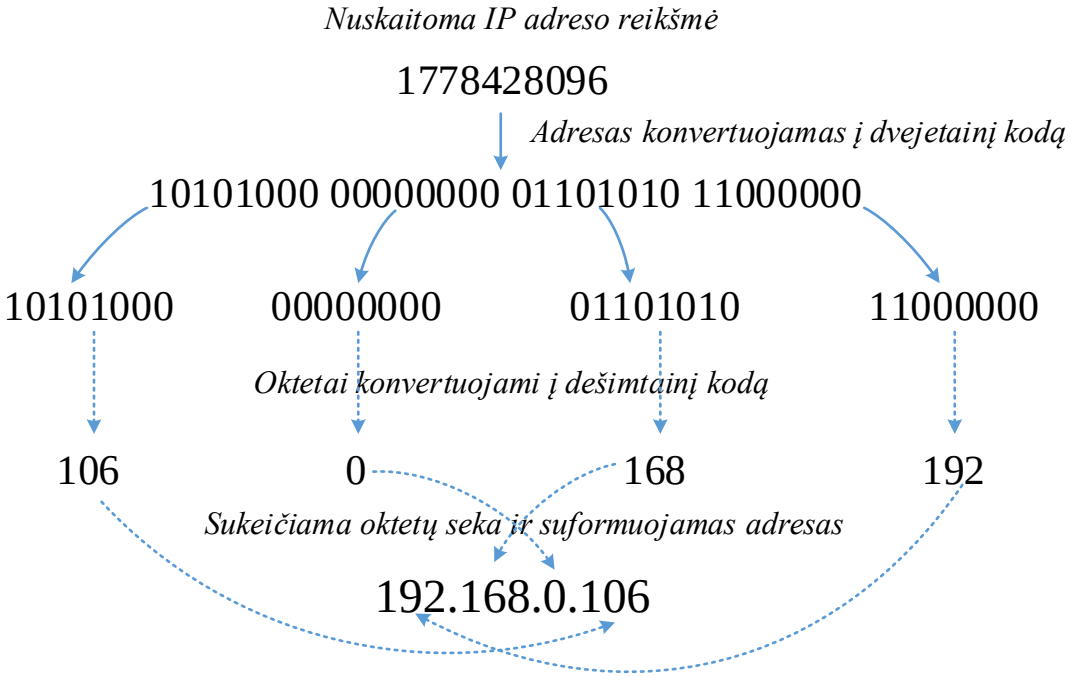
Prieš atlikdami skanavimo funkcijas pirma patikriname ar esame prisijungę prie Wi-Fi tinklo, jeigu nesame prisijungę prie Wi-Fi tinklo, paleidžiama Android įrenginio Wi-Fi nustatymų aplikacija, kurioje galime prisijungti prie Wi-Fi tinklo. Žemiau pateikiamas programinis kodas tikrinti ar įrenginys yra prisijungęs prie Wi-Fi tinklo.

```
private boolean isWifiConnected(){
    ConnectivityManager connectivityManager = (ConnectivityManager)
          getSystemService(CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo =
          connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);

    return networkInfo.isConnected();}
```

Jeigu esame prisijungę prie Wi-Fi tinklo nuskaitomas mūsų įrenginio IP adresas. Šis adresas yra reikalingas nustatyti IP adresų intervalui kurių pasiekiamumą tikrinsime.

Vietinio tinklo adresai gali būti nuskaitomi naudojant mūsų IP adresą ir tinklo kaukęs adresą. Android OS naudojami IP adresai yra užrašomi dvejetainiu pavidalu ir yra apdorojami *int* tipo kintamajame. Kadangi adresas pradedamas nuskaityti nuo pirmojo okteto, jo reikšmė yra mažiausią, ir sekantys oktetai yra perstumiami per 8 bitus, 3.2 pav. pateiktas pavyzdys, kaip IP adresą konvertuoti į dešimtainį kodą iš *int* tipo kintamojo.




3.2 pav. IP adreso konvertavimas

Žinant, kad IP adresas yra saugomas dvejetainių formatu, galima atlikti tinklo skaičiavimo operacijas ir suskaičiuoti tinklo adresą, transliacijos adresą, maksimalų ir minimalų vartotojo adresą potinklyje. Žemiau pateikiama IP adreso konvertavimo funkcija.

```
private String intToIP(int i){
    return (i & 0xFF) + "." +
            ((i >>> 8 ) & 0xFF) + "." +
            ((i >>> 16 ) & 0xFF) + "." +
            ((i >>> 24 ) & 0xFF);
}
```

Sujungimams sudaryti naudosime *ping* funkcija, kurioje nurodome IP adresą, kurio pasiekiamumą norime patikrinti. Naudosime „*-w 1*“ funkcijos priedelį, juo nustatome kad funkcija bus nutraukiama įvykdžius sėkmingą sujungimą arba jei funkcija negaus atsakymo iš nurodyto įrenginio po 1 sekundės. Pasirinktas vienos sekundės laiko tarpas, kadangi esant vietiniame tinkle tai yra pakankamas laiko tarpas gauti atsakymui iš tame pačiame tinkle esančio įrenginio. Žemiau pateikiamas programinis kodas, kuriuo tikrinamas įrenginių prieinamumas.

```
private boolean ping(String ip) {
    Runtime runtime = Runtime.getRuntime();

    try {
        Process  ipAddrProcess = runtime.exec("/system/bin/ping -w 1 " + ip);
        int exitValue = ipAddrProcess.waitFor();
        Log.d(TAG, "checking " + ip + " " + exitValue);
        if(exitValue == 0){
            return true;
        } else {
            return false;
        }
```

```
    } catch (InterruptedException ignore){
        ignore.printStackTrace();
        Log.d(TAG, "Exception " + ignore);
    }
    return false;
}
```

Tačiau net ir naudojant viena sekundę *ping* funkcijos atsakymo laukimui, visų vietinių adresų prieinamumo tikrinimas standartiniame vietiniame tinkle gali užtrukti daugiau nei keturias minutes, kadangi yra galimi 254 skirtingi IP adresai. Šis laiko tarpas yra ilgas ir tinklo skanavimo metu ARP lentelės įrašai gali būti pašalinami.

Siekiant greičiau atlikti tinklo įrenginių skanavimą, buvo panaudoti atskiri sukurti procesai (angl. *Thread*). Buvo panaudoti dvidešimt šeši skirtingi procesai, kuriems buvo priskirtas dešimties IP adresų ruožas. Naudojant atskirus procesus adresų prieinamumo tikrinamas užtrunka apie 10 sekundžių, priklausomai nuo įrenginio spartos.

ARP lentelė yra saugoma atmintyje ir yra atnaujinama, todėl lentelės įrašai gali būti panaikinami, jei nebuvo sudarytą jokių sujungimų su nurodytais įrenginiais. Todėl norint gauti tiksliausius rezultatus ARP lentelės duomenis geriausia nuskaityti iškart po to, kai atliekame sujungimų sudarymo funkciją. Žemiau pateikiamas ARP lentelės duomenų nuskaitymo funkcija.

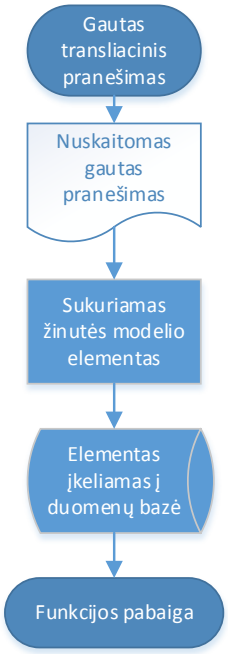
```
    BufferedReader localBufferdReader = new BufferedReader(new FileReader(new
File("/proc/net/arp")));
    String line;

    while ((line = localBufferdReader.readLine()) != null){
        // nuskaitoma arp lentelės eilutė
    }
```

Svarbu paminėti, kad tai nėra vienintelis būdas atlikti įrenginių paieškai vietiniame tinkle ir turi savo trūkumų, kadangi šiuo būdų nėra nuskaitomi įrenginių pavadinimai (angl. *Host Name*) ir nuskaitomi tik standartiniai C klasės IP tinklai. Alternatyvos gali būti *inetAdress* klasės funkcijų panaudojimas, tačiau šiuo būdų turime mažiau lankstumo, kadangi negalime nustatyti laiko tarpo, kuri skirsime sujungimo sudarymui. Kita alternatyva gali būti įrenginio vietinio DNS serverio duomenų nuskaitymas, tačiau tam reikalingas papildomų bibliotekų įdiegimas.

### 3.1.2 Informacinių pranešimų priėjimas

Informacinių celės pranešimų priėjimui naudosime Android OS transliacijų priėmėją (angl. *Broadcast Receiver*) ir *SQLite* duomenų bazę, kurioje bus saugomi pranešimai. 3.3 pav. pateiktas aplikacijos algoritmas.




3.3 pav. Informacinių celės pranešimų priėmimo algoritmas

Ši funkcija yra iškviečiama kai transliacijos priėmėjas gauna informacinį pranešimą. Norint, kad transliacijų priėmėjas reaguotų į transliacinius pranešimus jį reikia registruoti, tai galima padaryti dviem būdais:

- Registruojant transliacijos priėmėją aplikacijos pagrindiniame kode
- Registruojant transliacijos priėmėją aplikacijos AndroidManifest.xml faile.

Registruojant transliacijos priėmėja aplikacijos pagrindiniame kode, jis yra iškviečiamas pačios aplikacijos, todėl aplikacija turi būti įjungta, kad priimtu pranešimus. Registruojant transliacijos priėmėja aplikacijos AndroidManifest.xml faile, jis yra iškviečiamas pačios Android OS sistemos, todėl aplikacija neturi būti įjungta tam kad priimti registruotus transliacinius pranešimus. Šioje aplikacijoje transliacijų priėmėjas yra registruojamas AndroidManifest.xml faile, žemiau programinis kodas naudojamas registruoti transliacijų priėmėją.

```
    <receiver android:name=".CbReceiver" android:exported="true" >
        <intent-filter>
            <action android:name="android.provider.Telephony.CB_RECEIVED" />
        </intent-filter>
        <intent-filter>
            <action
android:name="android.provider.Telephony.CB_SETTINGS_AVAILABLE" />
        </intent-filter>
    </receiver>
```

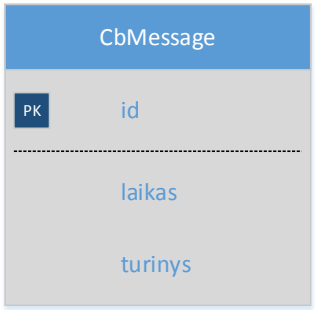
Svarbu paminėti, kad transliacijų priėmėjai iššaukiami Android OS turi ribota veikimo laiką, apie 10 sekundžių, todėl norint atlikti sudėtingas operacijas rekomenduojama transliacijos priėmėja naudoti paslaugos (angl. *service*) paleidimui, kuri veikia užsklandoje.

Informacinio pranešimo nuskaitymas aprašomas transliacijos priėmėjo klasėje. Jos

programinis kodas pateikiamas žemiau

```
public void onReceive(Context context, Intent intent) {
        // atliekamos operacijos duomenu nuskaitymui iš pranešimo
        Bundle bundle = intent.getExtras();
        SmsCbMessage[] msgs = null;
        String str = "";
        if (bundle != null) {
            //-jei gautas prenešimas nėra tusčias
            Object[] pdus = (Object[]) bundle.get(SMS_BUNDLE);
            msgs = new SmsCbMessage[pdus.length];
            for (int i = 0; i < msgs.length; i++) {
                // suformuojama žinutė
                msgs[i] = SmsCbMessage.createFromPdu((byte[]) pdus[i]);
        }
    }
}
```

Išsaugoti informacinio pranešimo informacijai sukuriama atskira klasė, kurioje išsaugojamas informacinio pranešimo laikas ir informacinio pranešimo turinys. 3.4 pav. pateikiamas informacinio pranešimo modelis.

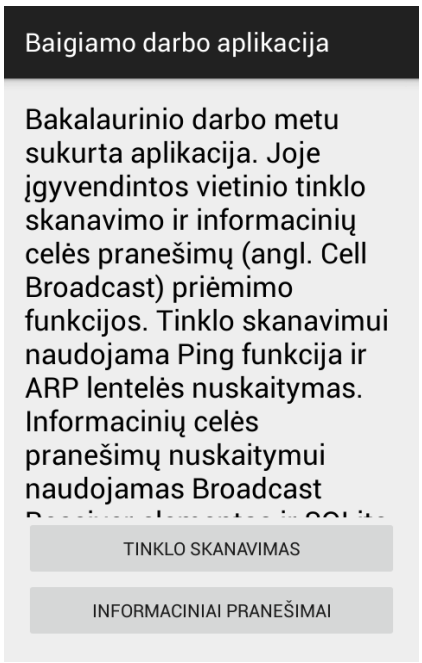



3.4 pav. Informacinio pranešimo modelis

Suformuotas informacinio pranešimo modelis yra išsaugojamas Android *SQLite* duomenų bazėje.

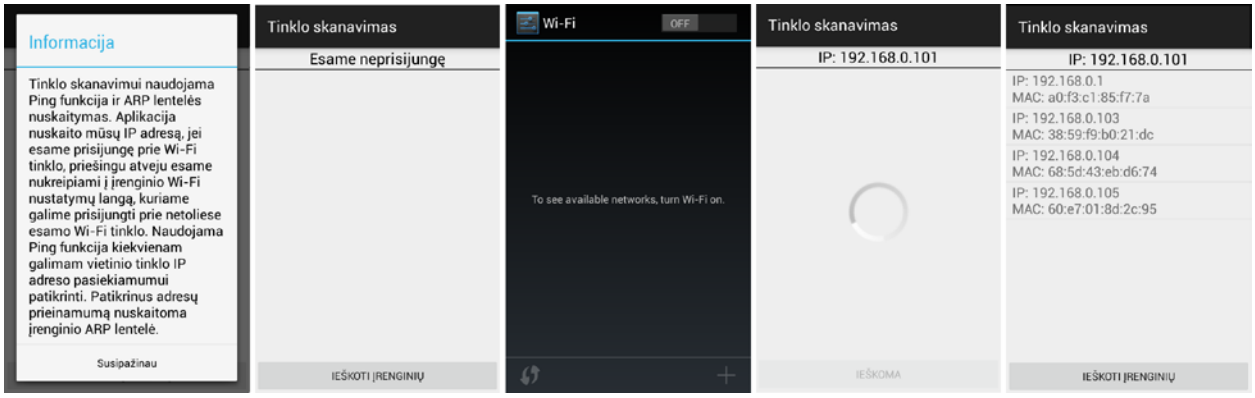
### 3.1.3 Aplikacijos testavimas

Aplikacijos testavimas buvo atliekamas Android OS išmaniajame telefone ir pačioje Android Studio programoje stebint registruotus (angl. *Log*) pranešimus ir naudojama įrenginio atmintį. 3.5 pav. pateikiamas aplikacijos pagrindinis meniu



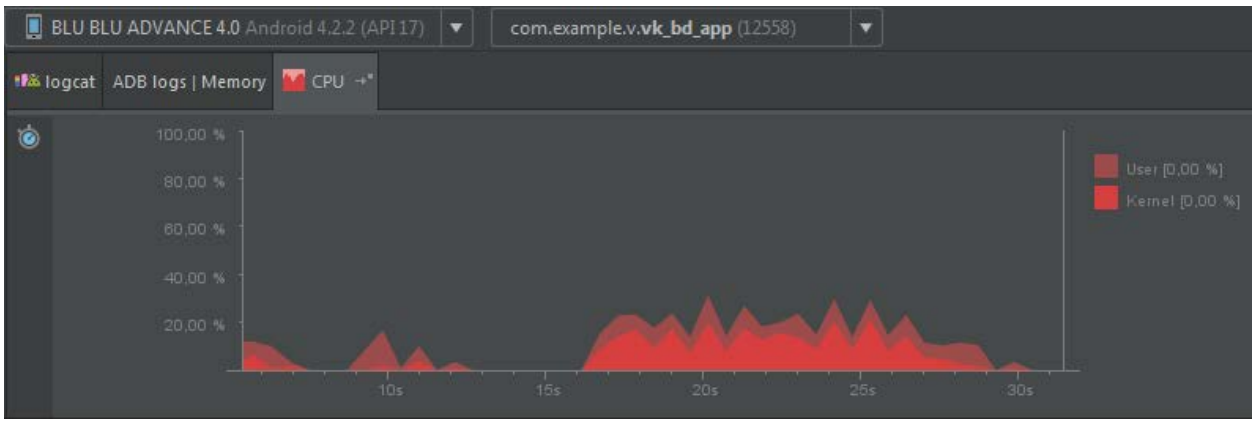

3.5 pav. Pagrindinis aplikacijos langas

Pagrindiniame aplikacijos lange pateikimas aplikacijos aprašymas ir nuorodos į tinklo skanavimo ir informacinių pranešimų aplikacijas. 3.6 pav. pateikiamas tinklo skanavimo aplikacijos veikimas.

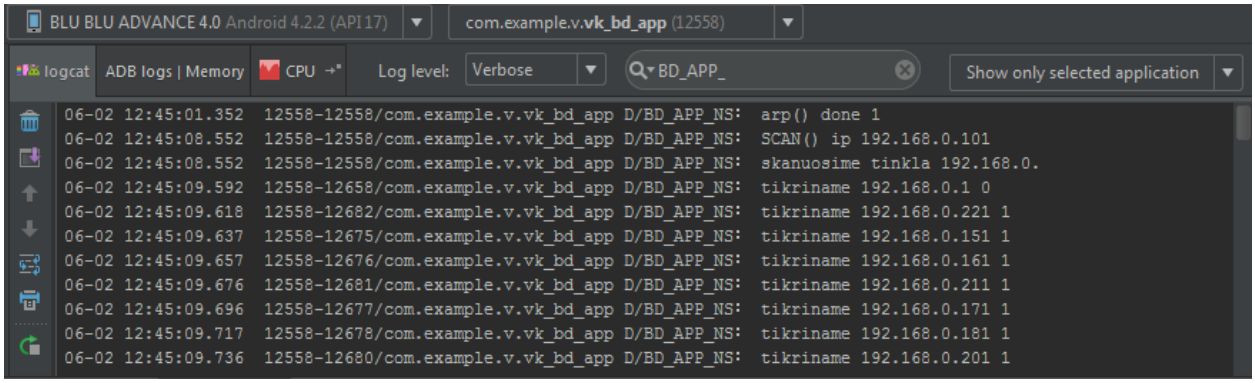



3.6 pav. Tinklo skanavimo aplikacijos veikimas

Aplikacijoje pateikiama aplikacijos informacija atskirame lange, pateikiama prisijungimo informacija, jei nesame prisijungę prie tinklo, esame nukreipiami į įrenginio Wi-Fi nustatymų aplikaciją. Vykdant tinklo skanavimą vartotojas yra informuojamas naudojant laukimo animacija. Baigus tinklo įrenginių skanavimą ARP lentelėje esantys IP ir MAC adresai yra parodomi vartotojui. 3.7 ir 3.8 pav. pateikiami aplikacijoje naudojami registro pranešimai (angl. *Log*) ir sunaudojami proceso resursai.

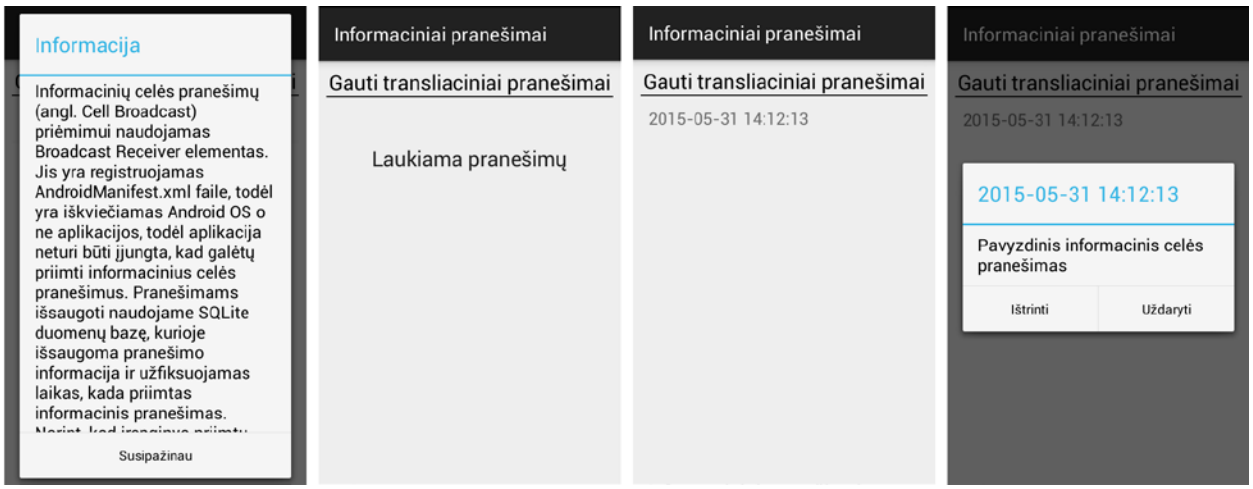



3.7 pav. Tinklo skanavimo aplikacijos sunaudojami resursai įrenginyje




3.8 pav. Tinklo skanavimo aplikacijos registro pranešimai

Sunaudojamų resursų lange galime matyti keik įrenginio resursų sunaudoja sukurta aplikacija, bei kuriais laiko momentais sunaudojama jų daugiausiai. Registro pranešimuose galime sužinoti aplikacijos veikimo eigą, kadangi galime nustatyti kurios aplikacijų funkcijos pateiks registro pranešimus. Tai labai naudinga ieškant aplikacijos klaidų, kadangi galime matyti, kurioje aplikacijos funkcijoje sustoja aplikacijos veikimas. 3.9 pav. pateikiamas informacinių celės pranešimų priėmimo aplikacijos veikimas.




3.9 pav. Informacinių celės pranešimų priėmimo aplikacijos veikimas

Aplikacijoje pateikiama aplikacijos informacija atskirame lange, pateikiama prisijungimo informacija, jei nėra gautų informacinių pranešimų vartotojas informuojamas pagrindiniame

lange pateikiant tekstinę informaciją aplikacijos lange. Jei turime gautų informacinių pranešimų, jie yra pateikiami vartotojui, pagal pranešimų gavimo laiką. Paspaudus ant pranešimo išmetamas papildomas informacinis langas, kuriame pateikiama pasirinktos žinutės informacija. Vartotojas pasirinktą pranešimą gali ištrinti.

# IŠVADOS IR PASIŪLYMAI

1. Android OS kiekvienai aplikacijai naudoja atskirą Dalvik arba Android Runtime virtualią mašiną, taip užtikrinant OS saugumą, kadangi aplikacijos yra atskirtos viena nuo kitos.
2. Android OS naudojamos bibliotekos ir techninės įrangos tvarkykles yra papildomos įrenginių gamintojų ir yra specifinės įrenginio techninei įrangai. Todėl skirtingų gamintojų įrenginiuose naudojamos modifikuotos Android OS versijos.
3. Programos naudojančios grafinio programavimo elementus supaprastina aplikacijų kūrimą, bet jų sukurtų aplikacijų funkcionalumas yra ribotas, dėl fiksuotos elementų struktūros ir riboto funkcijų kiekio.
4. Programos skirtos kurti aplikacijas įvairioms OS platformoms yra patogus įrankis, norint tą patį aplikacijos kodą panaudoti įvairių platformų aplikacijoms. Tačiau šių programų naudojimas yra sudėtingesnis, joms reikia papildomų programinės įrangos paketų.
5. Naudojant standartines Android SDK paketo API sąsajų funkcijas galima atlikti standartines Linux Shell funkcijas ir nuskaityti sisteminius Android OS failus.
6. Android aplikacijos gali apdoroti transliacinius pranešimus ir tada kai aplikacijos yra išjungtos, jeigu aplikacijos transliacijų gavėjas (angl. *BroadcastReceiver*) yra registruojamas *AndroidManifest.xml* faile.

# INFORMACIJOS ŠALTINIŲ SĄRAŠAS


1. Get Started with Android Studio [žiūrėta 2015-04-19]. Priega per internetą <http://developer.android.com/develop/index.html>
2. Sarafinienė N., Lagzdinytė-Budnikė I., Matulis D., Vilutis G., Zakarevičius R. Operacinių sistemų architektūros: mokomoji knyga 2012 - 48p. ISBN, 978-609-02-0494-8
3. Qt for Android [žiūrėta 2015-04-19]. Prieiga per internetą <http://doc.qt.io/qt-5/android-support.html>
4. The MIT App Inventor Library: Documentation & Support (AI2) [žiūrėta 2015-04-19]. Prieiga per internetą <http://appinventor.mit.edu/explore/library.html>
5. Android version history [žiūrėta 2015-05-05]. Prieiga per internetą <http://en.wikipedia.org/wiki/Android_version_history>
6. ARCHITECTURE [žiūrėta 2015-04-19]. Prieiga per internetą <https://greatdreams4alifetime.wordpress.com/2013/08/19/3-architecture/>
7. Android Architecture Guides for beginners [žiūrėta 2015-04-19]. Prieiga per internetą <http://www.edureka.co/blog/beginners-guide-android-architecture/>
8. Android Architecture- Detailed tutorial about the Architecture of Android [žiūrėta 2015-04-19]. Prieiga per internetą <http://www.eazytutz.com/android/android-architecture/>
9. An Overview of the Android Architecture[žiūrėta 2015-04-19]. Prieiga per internetą <http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture>
10. MassAlert – gyventojų perspėjimo ir informavimo sistema [žiūrėta 2015-05-05]. Prieiga per internetą <http://www.ntservice.lt/go.php/Sprendimai5983784960931423>
11. ping(8) - Linux man page [žiūrėta 2015-05-31]. Prieiga per internetą <http://linux.die.net/man/8/ping>
12. Address Resolution Protocol (arp) [žiūrėta 2015-05-31]. Prieiga per internetą <http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html>
13. Cell broadcast explained [žiūrėta 2015-05-31]. Prieiga per internetą <http://www.one2many.eu/en/cell-broadcast/how-it-works>
14. Eclipse, Android and Java training and support [žiūrėta 2015-05-31]. Prieiga per internetą <http://www.vogella.com/>
15. Stack Overflow [žiūrėta 2015-05-31]. Prieiga per internetą <http://stackoverflow.com/>
16. XDA-Developers Android Forums [žiūrėta 2015-05-31]. Prieiga per internetą <http://www.xda-developers.com/>
17. Introduction to XDA-University [žiūrėta 2015-05-31]. Prieiga per internetą <http://xda-university.com/introduction-to-xda-university>
18. Five Best Android ROMs [žiūrėta 2015-05-31]. Prieiga per internetą <http://lifehacker.com/5915093/five-best-android-roms>