Универзитет у Београду

Електротехнички факултет

Катедра за сигнале и системе

Милош Милошевић 3186/2016

Мастер рад

**Препознавање разделних линија коловозних трака на основу снимка са камере на ветробранском стаклу**

Ментор: Доц. др Вељко Папић

# Захвалност

Желео бих да се захвалим и професору Жељку Ђуровићу на неисцрпном извору знања, идеја и мотивације које несебично преноси на студенте.

На крају, највећу захвалност дугујем својој породици.

# Сажетак

Циљ овог рада био је развој С++ корисничке апликације за управљање осветљењем у паметној згради. Развијена апликација се покреће на *Raspberry Pi* платформи и подржава графичко корисничко окружење. Рад даје и опис елемената паметних зграда, преглед модерних технологија које се користе у системима кућне аутоматизације и осврт на архитектуру *Oblo* система као једног од конкретних решења кућне аутоматизације.

# Кључне речи

*Lane detection, Driver assistance,* V*ideo processing, Computer vision, OpenCV*

# Садржај

[1. Увод 6](#_Toc518151802)

[2. Системи за напредну помоћ возачу 7](#_Toc518151803)

[3. Обрада видеа употребом *OpenCV* рачунарске библиотеке 8](#_Toc518151804)

[3.1. Представљање слика 8](#_Toc518151805)

[3.1.1. Вредносна квантизација и системи боја 8](#_Toc518151806)

[3.1.2. Просторна квантизација 12](#_Toc518151807)

[3.2. Канијев детектор ивица 13](#_Toc518151808)

[3.3. Хафова трансформација 16](#_Toc518151809)

[4. Предлог решења 17](#_Toc518151810)

[5. Имплементација решења 18](#_Toc518151811)

[6. Закључак 19](#_Toc518151812)

[Литература 20](#_Toc518151813)

[Прилог А: *Makefile* за превођење апликације 21](#_Toc518151814)

# Листа симбола

ADAS *Advanced driver-assistance system -* системи за напредну помоћ возачу

g++ *GNU C++ Compile -* стандардни *Linux* преводилац за С++

GUI *Graphical User Interface -* графичко корисничко окружење

gtkmm *GTK minus minus -* GTK омотач за С++

GTK *GNU Image Manipulation Toolkit -* скуп библиотека за рад са графиком

JSON *JavaScript Object Notation -* формат размене података

MQTT *MQ Telemetry Transport* - протокол размене података

OHM *Oblo Home Manager -* централна јединица *Oblo* система

POCO *Portable Components -* библиотека за развој наменских система

SSH *Secure Shell* - протокол за удаљени приступ рачунару

UML *Unified Modeling Language* - језик за представљање објектног модела

# Увод

Модерно доба своју примену налазе и у аутономном управљању аутомобилима. Систем за препознавање коловозних трака на основу снимка камере монтиране на ветробранско стакло представља само један од сегмената напредних система за помоћ возачу. Према неким очекивањима ови системи би, већ наредне деценије, напредовали до нивоа да потпуно самостално управљају возилима.

Овај рад предлаже једно занимљиво решење за регулацију и управљање расветом у паметној кући. Решење се ослања на *Oblo* систем, идејно решење   
научно-истраживачког института „RT-RK“ у области кућне аутоматизације. Такође, рад укључује и примену *Raspberry Pi* рачунара, рачунара опште намене и малих габарита чија област примене сеже до граница људске креативности. Срж рада односи се на методе аутоматског управљања примењене у оквиру система за регулацију осветљености.

Структура рада састоји се од седам целина. Након уводног поглавља представљени су елементи и архитектура паметних зграда. У другом поглављу описане су и технологије релевантне за развој решења описаног овим радом. Треће поглавље односи се на *Raspberry Pi* платформу за коју је решење развијено. Додатно, треће поглавље говори и о техници унакрсног превођења која омогућава развој софтвера на персоналном рачунару и његово превођење у облику који је погодан за извршавање на некој од других платформи. Четврто поглавље описује архитектуру предложеног решења. Док је конкретна имплементација предложене архитектурe дата у петом поглављу. Претпоследње поглавље представља осврт на успешност реализације и правце даљег развоја решења, a последње поглавље садржи списак литературе која је коришћена приликом реализације рада.

На самом крају, у прилогу A дат је *Makefile* фајл којим је вршено превођење апликације. Коришћење овог фајла омогућава превођење апликације и за извршавање на рачунару и на *Raspberry Pi* платформи.

# Системи за напредну помоћ возачу

Системи за напредну помоћ возачу

# Обрада видеа употребом *OpenCV* рачунарске библиотеке

*OpenCV* представља рачунарску библиотеку намењену дигиталној обради слике, машинском учењу и рачунарској визији. Оптимизована је за извршавање на системима у реалном времену и лако се укључује у апликације писане у *С++,* *python* и *Java* програмским језицима. Њен изворних код је јаван (енгл. *оpen source*) и објављена је под BSD лиценцом (енгл *Berkeley Software Distribution* *license*) што значи да се слободно може користити и у академске и у комерцијалне сврхе. *OpenCV* рачунарска библиотека подржана је на *Windows, Linux, Android* и *Mac OS* оперативним системима. [1].

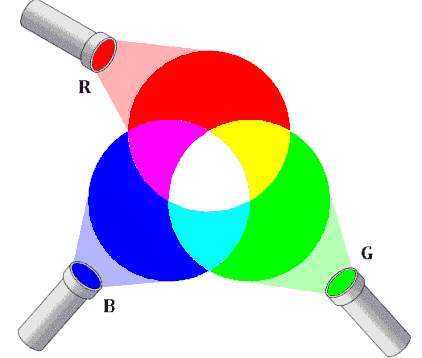
*OpenCV* рачунарска библиотека је већ пронашла своју улогу у занимљивим програмским решењима водећих компанија попут спајања фотографија усликаних возилима *Google Street View*-а. У имплементацији овог рада, *OpenCV* ће бити главно оруђе за обраду слике.

## Представљање слика

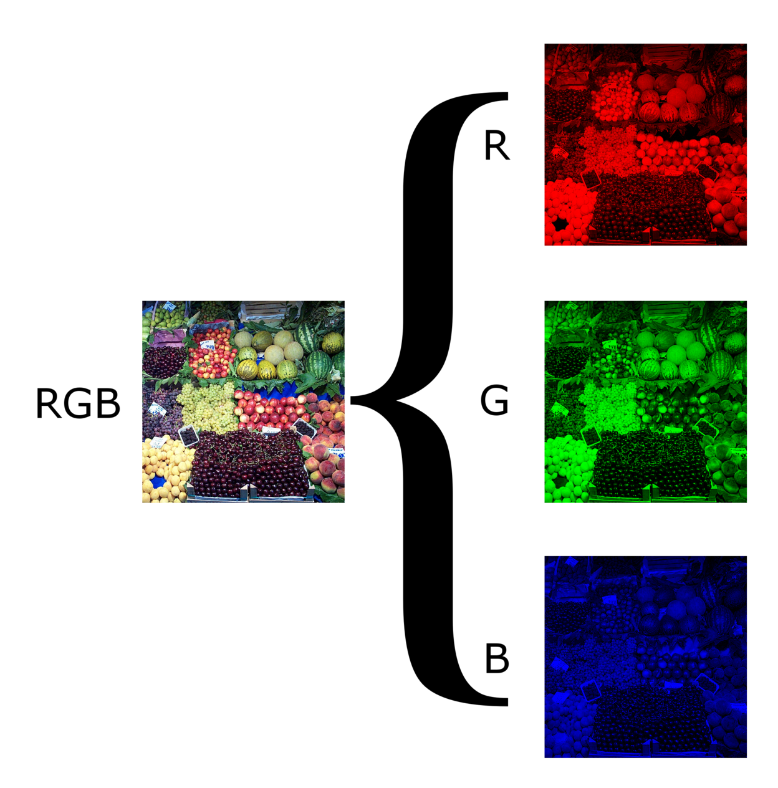
Људско око је један од најосетљивијих и најсложенији органа у људском организму. Око нам омогућава да разноврстан садржај електромагнетног зрачења из спољашње средине доживимо као јасну слику. Електромагнетни таласи различитих таласних дужина у људском оку побуђују хемијске реакције. Овакве реакције резултују стварањем акционог потенцијала који путем осећајног нерва путује од ока до коре великог мозга где се ствара визуелна интерпретација, слика. Како рачунари баратају једино цифрама, оваква интерпретација слика им није позната. Да бисмо омогућили складиштење, обраду и приказивање слика на рачунару, нужна је вредносна и просторна квантизација.

### 3.1.1. Вредносна квантизација и системи боја

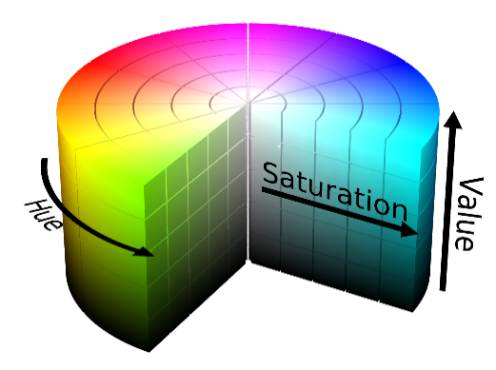
Вредносна квантизација подразумева додељивање јединственог бинарног кода свакој од боја. Од дужине кода зависи колико различитих боја је могуће представити тим типом кодовања. Број битова којим се кодира свака од боја назива се дубина боје (енгл. *color depth*). Скуп правила по којима се свакој боји додељује бинарни код назива се систем боја (енгл. *color space*). По правилу, у истом систему боја и истих димензија, слике веће дубине доживљавамо као квалитетније. Најчешћи системи боја су:

* RGB (скраћено од енгл. Red Green Blue) систем боја је најчешћи систем боја. Свака боја, у RGB систему, је представљена кроз три компоненте: црвену, зелену и плаву. Ове компоненте су независне једна од друге и узимају реалне вредности из опсега од 0 до 1. Како смо поменули да се боје кодирају бинарним вредностима, овај опсег се линеарно пресликава на скуп вредности од 0 до 2n где је n дубина боје слике. Захваљујући чињеници да је RGB систем боја адитиван, у стварности сваку од боја је могуће приказати користећи три светлеће диоде које емитују црвену, зелену и плаву светлост. Адитивност боја у RGB систему илустрована је на слици ФФ.

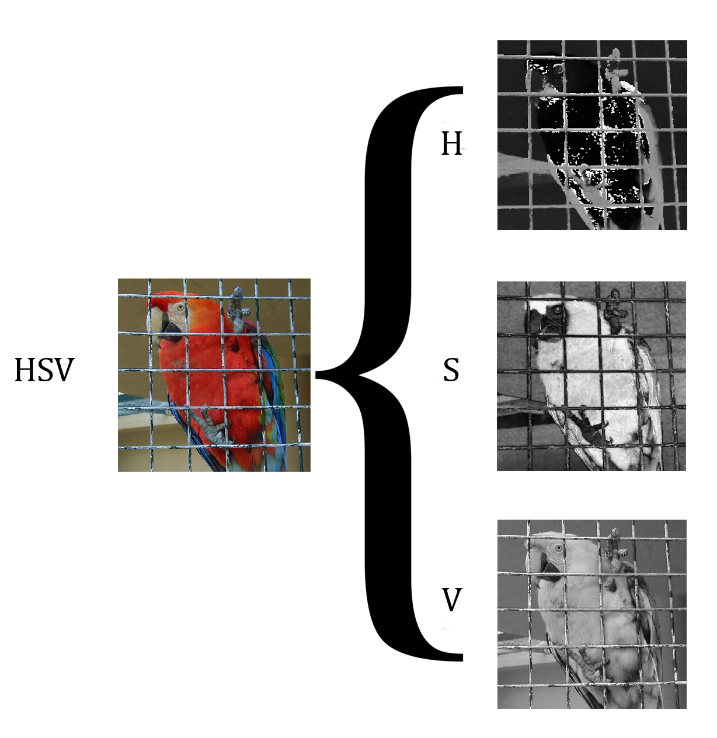
Слика ФФ: Илустрација адитивности RGB система боја

Матрице које садрже вредности кодова сваке од компоненти називамо равнима боја. Свака од равни боја оваквог система не носи довољно информација о целокупној слици и често није могуће препознати садржај слике само на основу једне равни. Слика ФФ приказује пример слику богату разноврсним бојама и садржај свих њених RGB равни. *OpenCV* рачунарска библиотека складишти RGB слике у нешто измењеном редоследу. *OpenCV* рачунарска библиотека користи BGR систем боја.

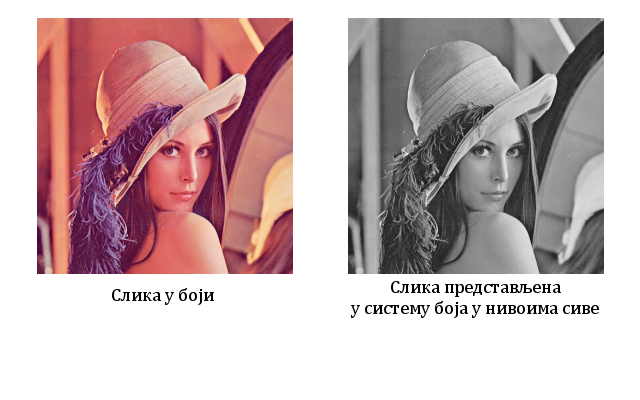
Слика ФФ: Слика представљена у RGB систему боја и њене равни

* HSV (скраћено од енгл. *Hue Saturation Value*) систем боја у којем је код сваке боје подељен у три сегмента: нијанса (енгл. *hue*), засићење (енгл. *saturation*) и осветљење (енгл. *value*). Овакав метод груписања компоненти боја близак је начину на који човек доживљава боје. Његова главна предност јесте чињеница да је информација о нијанси одвојена од информације о осветљености боје. Опсег вредности за нијансу је од 0 до 360 и ова вредност дефинише боју онако како је људски мозак доживљава. Засићење се представља реалном вредношћу у опсегу од 0 до 1 и оно носи информацију о уделу сиве боје. Осветљење се представља интензитет боје и узима вредност из скупа бројева од 0 до 100 где 0 представља потпуно црну боја независно од вредности осталих компоненти. [2] Организацију HSV система боја је најлакше визуелизовати када се боје распореде у цилиндричном координатном систему као на слици ФФ.

Слика ФФ: Распоред боја у HSV систему боја

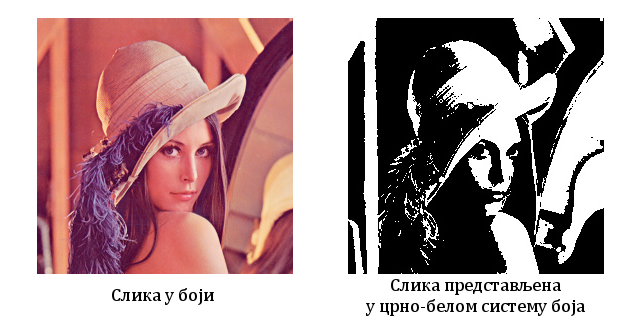
Овако постављен систем боја погодан је за рачунарску сегментацију слике када је од интереса издвајање области одређене боје. На слици ФФ се види како области исте боје имају сличне вредности нијансе. У имплементацији овог рада, HSV систем боја ће бити коришћен за потребе издвајања белих и жутих области коловозних трака.

Слика ФФ: Слика представљена у HSV систему боја и њене равни

* Систем боја у нивоима сиве (енгл. *grayscale*) је систем боја у којем се бинарни код сваке боје одређује на основу осветљености те боје. Овакав систем боја садржи само једну компоненту коју називамо интензитет. Интензитет се представља реалном вредношћу у опсегу од 0 до 1 где 0 представља црну боју, а 1 представља белу боју. Вредност интензитета се линеарно пресликава у бинарне вредности од 0 до 2n где је n дубина боје слике. Слика ФФ приказује изглед слике у боји и изглед слике након трансформације у систем боја у нивоима сиве.

Слика ФФ: Изглед слике у боји и након трансформације   
у систем боја у нивоима сиве

Слике представљене у систему боја у нивоима сиве погодне су за даљу обраду па често представљају резултат фазе предпроцесирања и припреме за примену алгоритама попут препознавања ивица.

* Црно-бели систем боја је најчешћи бинарни систем боја. У овом систему боја дефинисане су само две боје, црна и бела. Како су дефинисане само две боје, дужина кода којим обе боје могу бити једнозначно одређене је један. Слика ФФ приказује изглед слике у боји и у црно-белом систему боја.

Слика ФФ: Изглед слике у боји и у црно-белом систему боја

Црно-беле слике се брзо обрађују па су често резултат сегментације слика представљених у другим системима боја. Сегментацијом се их почетне слике издвајају области који су задовољавају постављени услов.

Једну слику могуће је представити и запамтити у сваком од система боја. Познате су и функције које дефинишу пресликавање кодова из једног у други систем боја. За потребе пребацивања слике из једног у други систем боја, *OpenCV* рачунарска библиотека нуди функцију *cvtColor*. Ова функција прима три аргумента, изворну слику, слику у коју се смешта резултат и трећи аргумент који означава из којег у који систем боја се врши конверзија.

### 3.1.2. Просторна квантизација

Просторна квантизација остварена је дељењем слике на велики број једнобојних региона. Такви региони називају се пиксели (енгл. *pixels*) и сваком од пискела додељена је једна квантована вредност боје. Уређени пар који чине број пиксела који стају у ширину и број пиксела који стају у висину слике назива се резолуција. По правилу, у истом систему боја са истом дубином, слике веће резолуције доживљавамо као квалитетније. Слика ФФ приказује утицај резолуције на квалитет слике.

Слика ФФ: Утицај резолуције на квалитет слике

Када сумирамо вредносну и просторну квантизацију, јасно је да је слика, у рачунару, представљена у облику матрице. Резолуција слике диктира димензије матрице, а сваки елемент матрице садржи бинарни код боје једног пиксела.

За потребе складиштења слика, у *OpenCV* рачунарској библиотеци, дефинисана је *Mat* класа. Ова класа доступна је од верзије 2.0 поменуте библиотеке. *Mat* класа складишти два типа података, метаподатке о слици и показивач на матрицу у којој су смештене вредности пиксела. Матрица вредности пиксела често је дељена између више *Mat* објеката и копирање објеката ове класе имплицитно подразумева дељење матрице вредности између оригинала и копије. На овај начин избегава се заузимање новог меморијског простора при прослеђивању слике као аргумента функције. Како обрада слике најчешће подразумева позивање низа смислених функција над једном сликом, на овај начин се штеди драгоцено време. [3]

## Канијев детектор ивица

Канијев детектор ивица (енгл. *Canny edge detector*) представља добро познати рачунарски алгоритам за издвајање ивица на слици. Развио га је аустралијански научник Џон Кани (енгл. *John Canny*). Алгоритам је развијен 1986. године за време Канијевог рада на MIT-у (скраћено од енгл. *Massachusetts Institute of Technology*). [5] Издвајање ивица овим алгоритмом се показало успешнијим од издвајања ивица на основу градијента.

Канијев детектор ивица осмишљен је са циљем да се:

* Максимално умањи грешка издвајања ивица
* Тачке које чине једну ивицу уско групишу дуж средишта те ивице
* За јединичну ивицу добије само један одзив

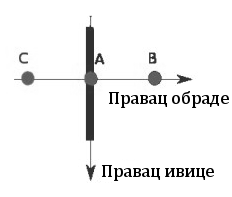
Након истраживања, Кани је дошао да закључка да су горе наведени услови испуњени када се обрада дате слике проведе кроз низ узастопних корака. Пре описа засебних корака важно је напоменути да Канијев алгоритам очекује почетну слику која је представљена у систему боја у нивоима сиве.

Први корак у примени Канијевог алгоритма јесте отклањање шума. Познато је да су сви алгоритми за детекцију ивица веома осетљиви на шум. Нарочите проблеме може направити импулсни шум. Канијев алгоритам користи Гаусов филтер за отклањање шума. Ширина Гаусовог филтера зависи од резолуције слике. Повећање ширине Гаусовог филтра умањује осетљивост алгоритма на шум по цену губитка тањих и краћих ивица на слици.

Други корак Канијевог алгоритма за издвајање ивица подразумева примену Собеловог оператора. Практично ово је корак у којем се од почетне слике добија слика која издваја ивице. У овом кораку се, на слику која је добијена након примене првог корака, независно примењују Собелов оператор за издвајање водоравних ивица и Собелов оператор за извдвајање усправних ивица. Две слике које се добију као резултати филтрирања се затим стапају у једну коришћењем формуле где је слика која садржи водоравне ивице, а слика која садржи усправне ивице. Овако добијена слика се користи као улазна слика за наредни корак Канијевог алгоритма. Слика ФФ илуструје почетну слику, међурезултате и слику која представља поменути збир. Додатно, резултат овог корака је још једна матрица која ће бити од користи у наредном кораку обраде. Ова матирца носи информацију о правцу пружања издвојених ивица и добија се по формули .

Слика ФФ: Примена Собеловог оператора као један од корака  
Канијевог алгоритма

Наредни кораци имају за циљ да још јасније издвоје ивице које задовољавају набројане критеријуме. Трећи корак у целокупној обради подразумева потискивање локалних немаксимума са слике која је добијена као резултат претходног корака. Потискивање локалних немаксимума узрокује истањавање ивица.

За обраду у овом кораку користимо матрицу *theta* која је била споредни резултат претходног корака. Вредности у овој матрици носе информацију о орјентацији ивице. Пре употребе матрице врши се редукција информација тако што се све вредности квантизују у четири правца, водоравни, усправни и два правца по дијагонали. Пошто нам је позната орјентација ивице можемо да потражимо локалне макимуме, а вредности које нису локални максимуми заменимо нулама. Пролазимо кроз целу слику, пиксел по пиксел и испитујемо два од осам првих суседа сваког пиксела. Која два пиксела ћемо посматрати зависи од вредности у матици *theta* односно од правца којим се простире ивица. Посматрамо пикселе који су нормални на правац простирања ивице. На пример, уколико је ивица усправна, посматрамо први леви и први десни пиксел пискела који обрађујемо. Уколико је вредност пиксела који се обрађује већа од вредности оба суседна пиксела, пиксел задржава своју вредност иначе добија вредност нула. Оваквим алгоритмом задржавамо само локалане максимуме у правцу пружања ивице. Слика ФФ илуструје описани поскупак. Пиксел А задржава своју вредност само уколико је њехова вредност већа и од вредности пиксела C и од вредности пиксела B.

Слика ФФ: Потискивање локалног немаксимума

Последњи корак има за задатак крајње издвајање. Излаз овог корака је бинарна слика која је уједно и резултат примене Канијевог детектора ивица. За примену овог корака потребне су нам две граничне вредности које одређују горњу и доњу граничну вредност сегемнтације. Сви пиксели чија је вредност изнад горње границе препознајемо као пикселе који сачињавају ивице и одмах им додељујемо белу боју. Слично, све пискеле чија је вредност испод доње границе одбацујемо и додељујемо им црну боју. Питање је шта са пикселима чија је вредност између доње и горње границе. Ове пискеле препознајемо као делове ивица уколико у свом суседству имају барем један пиксел који је класификован као део ивице. Овакав поступак одлучивања се ланчано наставља. Слика ФФ осликава описано закључивање. Ивицу А препознајемо као ивицу док ивицу В одбацијемо јер нема повезаних пиксела чија је вредност већа од горње границе. Овакав поступак се још назива и сегментација са хистерезисом.

Слика ФФ: Издвајање ивица са хистерезисом на основу горње и доње граничне врендости

У оквиру рачунарске библиотеке *OpenCV* доступна је функција која спроводи описану обраду и издвајање ицива. Поменута функција се назива *Canny* и прима 5 аргумената. Прва два аргумента намењена су улазној и излазној слици дакле слика која је резултат обраде враћа се кроз прослеђени аргумент. Трећи параметар одређује ширину Гаусовог филтра који ће бити примењен за отклањање шума и замућивање улазне слике. Последња два аргумента служе за прослеђивање горње и доње границе за потребе сегемнтације са хистерезисом.

Утицај параметара

[4]

## Хафова трансформација

# Предлог решења

# Имплементација решења

# Закључак

# Литература

[1] *Oblo living*

Нови Сад: RT-RK Institute for Computer Based Systems

<http://www.obloliving.com>

[2] *MQTT Essentials*

Landshut: HiveMQ

<http://www.hivemq.com/blog/mqtt-essentials>

[3] *УВОД - Raspberry Pi рачунар*

Нови Сад: Факултет техничких наука

<http://www.rt-rk.uns.ac.rs/sites/default/files/materijali/lab/uvod_0.pdf>

[4] *Poco C++ libraries Documentation*

Maria Elend: Applied Informatics Software Engineering GmbH

<http://pocoproject.org/documentation>

[5] *Programming with gtkmm 3*

GNOME

<https://developer.gnome.org/gtkmm-tutorial/stable/index.html.en>

[6] *How* *X10 works*

SmartHomeUSA

<http://www.smarthomeusa.com/how-x10-works>

[7] *A Simple Makefile Tutorial*

Waterville: Colby College

<http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor>

[8] *Home Automation buying guide*

CNET

<http://www.cnet.com/news/smart-home-buying-guide-home-automation>

[9] *About Z-Wave Technology*

Z-Wave Alliance

<http://z-wavealliance.org/about_z-wave_technology>

[10] *What’s The Difference Between ZigBee And Z-Wave?*

Electronic Design

<http://goo.gl/oKpY7c>

# Прилог А: *Makefile* за превођење апликације

Помоћу овог *Makefile* фајла могуће је превођење апликације и за извршавање на персоналном рачунару и за извршавање на *Raspberry Pi* рачунару. У случају превођења за персонални рачунар користи се *g++* преводилац, а у случају унакрсног превођења *arm-linux-gnueabi-g++* унакрсни преводилац.

Превођење за персонални рачунар реализује позивањем наредбе make LightRegulationPC из терминала рачунара. А у случају превођења за *Raspberry Pi* потребно је позвати make LightRegulationPi. У оба случаја неопходно је да радни фолдер терминала буде фолдер у коме се налази изворни код апликације. Извршив фајл смешта се у *bin* фолдер.

INCDIR = -I./include -I./include/messenger -I./include/paho -I./include/Poco

LIBDIR = -L./lib

TARGET = ./bin/LightRegulationPC

CC = g++

INCDIR\_PI = -I./includePi -I./includePi/messenger -I./includePi/paho -I./includePi/Poco

LIBDIR\_PI = -L./libPi

TARGET\_PI = ./bin/LightRegulationPi

CROSS\_CC = arm-linux-gnueabi-g++

LIBS = -loblomessenger -lPocoCrypto -lPocoJSON -lPocoFoundation

SRC = ./src/main.cpp ./src/UserException.cpp ./src/MqttWrapper.cpp ./src/MqttAbstractMessage.cpp ./src/GTWStorage.cpp ./src/Display.cpp ./src/GUIHandlers.cpp

GTKMM\_TAG = `pkg-config gtkmm-3.0 --cflags --libs`

LightRegulationPC: **$(**SRC**)**

**$(**CC**)** **$(**INCDIR**)** **$(**LIBDIR**)** -o **$(**TARGET**)** **$(**SRC**)** **$(**LIBS**)** **$(**GTKMM\_TAG**)**

LightRegulationPi: **$(**SRC**)**

**$(**CROSS\_CC**)** **$(**INCDIR\_PI**)** **$(**LIBDIR\_PI**)** -o **$(**TARGET\_PI**)** **$(**SRC**)** **$(**LIBS**)** **$(**GTKMM\_TAG**)**