УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Младен Матић

Проширење *WISE* протокола подршком за концентраторске уређаје и уређаје за конверзију протокола

МАСТЕР РАД

Нови Сад, (2021)

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Дипломски – мастер рад
Аутор, **АУ**:		**Младен Матић**
Ментор, **МН**:		**проф. др. Иштван Пап**
Наслов рада, **НР**:		**Проширење Бајз протокола подршком за концентраторске уређаје и уређаје за конверзију протокола**
Језик публикације, **ЈП**:		Српски / ћирилица
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		**2021**
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/68/0/6/66/3/0
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кључне речи, **ПО**:		*WISE*, *IP*, *IoT*, комуникациони протоколи
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		**У овом раду је представљена надоградња *WISE* протокола са подршком за уређаје који служе за концентрацију уређаја у мрежама интернета ствари. Додавањем подршке се омогућава *WISE* контролеру да комуницира са концентраторским уређајем, као и са свим уређајима којима концентратор рукује. *WISE* омогућава да се уређаји са концентратора виде на истоветан начин као и други паметни уређаји у систему. Концентратори су уређаји који омогућавају проширење мреже интернета ствари користећи интернет протокол када физичка технологија саме мреже то не дозвољава (нпр. *ZigBee* и *Z-Wave*) као и лако спрезање са уређајима који раде на технологији коју централни контролер паметне куће не познаје.**
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	доц. др Марија Антић
	Члан:	проф. др Иван Мезеи — Потпис ментора
	Члан, ментор:	проф. др Иштван Пап

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		Monographic publication
Type of record, **TR**:		Textual printed material
Contents code, **CC**:		Master Thesis
Author, **AU**:		**Mladen Matic**
Mentor, **MN**:		**PhD Istvan Papp**
Title, **TI**:		**Improvements of the WISE protocol with concentrator devices and devices for protocol conversion**
Language of text, **LT**:		Serbian
Language of abstract, **LA**:		Serbian
Country of publication, **CP**:		Republic of Serbia
Locality of publication, **LP**:		Vojvodina
Publication year, **PY**:		**2020**
Publisher, **PB**:		Author's reprint
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/68/0/6/66/3/0
Scientific field, **SF**:		Electrical Engineering
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, **S/KW**:		WISE, IP, IoT, communication protocols
UC		
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, **N**:		
Abstract, **AB**:		**This paper describes an extension of the WISE protocol with support for devices that concentrate other devices in the home network that may be running on a different protocol. With the addition of concentrator devices, the WISE controller gains ability to communicate with the concentrator devices and all other devices with whom the concentrator communicates. The WISE protocol allows the adaptation of the concentrator devices and uniform presentation in the same way as the rest of the smart home devices. Concentrators are the devices that enable IoT network expansion using the IP protocol when a physical technology of the network does not allow that (for example, ZigBee and Z-Wave). Also, easy connection with the devices which operate on technology which smart home central controller does not understand.**
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		
Defended Board, **DB**:	President:	Marija Antic, PhD
	Member:	Ivan Mezei, PhD
	Member, Mentor:	Istvan Papp, PhD

Menthor's sign

Захвалност

Захваљујем се родитељима Весни и Драгиши на безусловној подршци током школовања, као и ујни Лели, ујаку Милошу и стрицу Драгану.

Захваљујем се и целом гејтвеј тиму који је био увек ту да помогне када ми је била потребна помоћ, а посебно колеги Игору који ми је пренео сво знање о *WISE* протоколу потребно да на њему самостално радим. Такође се захваљујем и вођи тима Роману и Професору Иштвану на прилици да радим у оваквом тиму и на овако занимљивом пројекту.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

IoT - *Internet of Things*, Интернет ствари

OHM - *Oblo Home Manager*, Програмска подршка за апстракцију уређаја

OMB - *Oblo Message Broker*, Програмска подршка за комуникацију унутар система паметне куће

OSM - *Oblo System Manager*, Програмска подршка за управљање системом

MQTT - *Message Queuing Telemetry Transport* [1], Протокол за размену података на принципу претплате и објаве

TCP - *Transmission Control Protocol*, Протокол транспортног слоја

IP - *Internet Protocol*, Протокол мрежног слоја

OTA - *Over the Air Upgrade*, Надоградња управљачке програмске подршке преко интернета

HTTP - *Hypertext Transfer Protocol* [2], Комуникациони протокол апликативног слоја, у овом раду се користи за пренос датотека путем мреже

cURL - Библиотека која се користи за трансфер података преко мреже [3]

API - *Application Programming Interface*, Апликациона програмска спрега

JSON - *JavaScript Object Notation* [4], Синтакса којом се описују објекти приликом размене порука

1. Увод

У данашње време, скоро сваки уређај може да се повеже на интернет или да оствари неки вид мрежне комуникације. Почев од паметног телевизора, преко паметног сата, па све до паметног фрижидера и паметног аутомобила. Интернет ствари (eng. "*Internet of Things*", "*IoT*") је постала појава која је све присутнија у људским животима.

Интернет ствари утопија би била када би сви постојећи паметни уређаји могли комуницирати једни са другима на униформан начин и омогућити корисницима да са једног места могу контролисати све уређаје који припадају различитим системима, као и постављати узрочно последичне везе при обављању задатака између свих тих паметних уређаја.

Многи произвођачи система паметних кућа покушавају да обједине што је више могуће различитих уређаја да би корисницима био што више олакшан процес куповине и подешавања својих паметних уређаја. Неким корисницима су ти уређаји играчке, па их скупљају жељни да осете напредак технологије, док неким корисницима представљају потребу зарад сигурности и олакшања свакодневних задатака.

Међутим, колико год се произвођачи трудили да омогуће повезивање што више различитих уређаја, то на крају не буде изводљиво како због огромног броја уређаја који комуницирају помоћу великог броја различитих комуникационих протокола, тако и због високе цене производње контролера и развоја софтвера који би свим тим протоколима комуницирао.

Тада се поставља питање: зашто сви уређаји не би комуницирали преко интернет протокола? Он је ипак стандард за велики број паметних уређаја, а има и велики пропусни опсег па би количина података коју би уређај слао могла бити много већа. Проблем настаје јер је већина уређаја батеријски напајана и нема стални извор електричне енергије. Циљ произвођача је да корисник има потребу да батерију мења што је могуће ређе. На

једноставним уређајима, који функционишу преко неких комуникационих протокола специјално намењених за примену у паметним кућама, батерија може трајати и више година.

Мотивација овог рада је да пронађе решење за све ове проблеме. Циљ је осмислити и реализовати програмску подршку која би се уграђивала у уређај намењен за конверзију протокола, који би комуницирао путем интернет протокола са централним контролером паметне куће, и тако омогућио произвођачу контролера да смањи трошкове производње и развоја, а истовремено повећа палету подржаних уређаја. Произвођачи уређаја, који не користе интернет протокол за комуникацију, би направили један уређај за конверзију протокола, који би усмеравао сав саобраћај са уређаја ка централном контролеру и њега представљао као уређај заснован на интернет протоколу.

Тада би корисници могли купити један централни контролер који би могао управљати сваким уређајем уколико постоји уређај који служи за конверзију протокола за комуникацију којим се тај уређај служи.

У раду је описан *WISE* [5] (eng. "*WiFi Sensors*", "*WISE*") комуникациони протокол, који је заснован на интернет протоколу, и он је проширен подршком за уређај за конверзију протокола. Како корисници не би морали мењати уређај сваки пут када изађе нова верзија управљачког софтвера, додата је подршка у *WISE* протоколу за надоградњу истог. И на крају, додата је и подршка за чување резервне копије података уређаја да би се, у случају изненадне грешке у раду уређаја, вратила конфигурација последње контролне тачке која је функционисала без грешке.

Рад се састоји од седам поглавља: у првом поглављу дат је увид у тему рада, у другом поглављу објашњене су технологије коришћене приликом израде самог рада, у трећем поглављу је описан дизајн решења, у четвртом имплементација тог дизајна, у петом је описан систем за тестирање и резултати тестирања, у шестом поглављу је описан закључак који је извучен након завршетка рада, а у седмом поглављу је наведена литература коришћена при изради овог рада.

2. Теоријске основе

У овом поглављу биће описан систем паметне куће (Слика 1), протоколи који омогућавају комуникацију унутар система, апликације које се налазе на централном контролеру, клијентска апликација, сервис у облаку и све што је потребно да постави основу за даље проучавање овог рада.

Слика 1 Систем паметне куће

2.1 *MQTT* протокол за комуникацију

MQTT је протокол за мрежну комуникацију између уређаја, део је *ISO* (eng. *International Organization for Standardization*) стандарда и део је апликативног слоја *TCP/IP* скупа комуникационих протокола. Заснован је на моделу пријаве и објаве где уређаји могу

објављивати поруке или слушати на надолазеће поруке. Како *MQTT* протокол, чак и при слању кратких порука, због малог заглавља поруке не захтева велики пропусни опсег за слање порука, то га чини веома погодним за коришћење у *IoT* системима.

У систему паметне куће, који је описан у овом раду, ће се користити верзија *MQTT* протокола 3.1.1.

У *MQTT* комуникацији учествују брокер и клијенти. Брокер је својеврстан сервер који прослеђује поруке једног клијента ка другима, без да они знају једни за друге, док клијенти могу бити уређаји који објављују своје поруке и слушају на поруке других клијената. Такав вид комуникације је могућ на основу модела пријаве и објаве по ком ће један клијент објавити своју поруку на одређену тему (eng. "*Topic*"), а на основу тога брокер ту поруку проследи свим клијентима који слушају на ту исту тему. Тиме само брокер мора познавати адресе клијената док сваки клијент треба да познаје само адресу брокера и структуру тема од интереса за систем.

Клијент приликом своје прве конекције на брокер може поставити подразумевану поруку, коју би брокер објавио клијентима пријављеним да на њу слушају, у случају изненадног прекида везе.

При слању *MQTT* поруке може се навести и квалитет услуге (eng. "*Quality of Service*", "*QoS*") који може бити следећи:

1. Највише једном – порука је послата само једном без потребе за потврдом њеног пријема,
2. Макар једном – порука се изнова шаље све док се не добије потврда да је примљена,
3. Тачно једном – пошиљалац и прималац морају обезбедити да се пошаље и прими само једна копија поруке.

На слици испод приказан је изглед пакета за потврду пријаве на одређену тему ().

Пакет за потврду пријаве на тему	
packetId	12345
returnCode1	0
returnCode2	73

Слика 2 Пример пакета за потврду пријаве на тему

На слици испод наведен је изглед пакета за пријаву на одређену тему (Слика 3).

Пакет за пријаву на тему	
packetId	12345
qos1	0
topic1	protocolx/123/somereqname
qos2	1
topic2	protocolx/123/anotherreqname

Слика 3 Пример пакета за пријаву на тему

На слици испод приказан је изглед пакета за објаву поруке на одређену тему (Слика 4).

Пакет за објаву на тему	
packetId	12345
topicName	protocolx/123/somereqname
qos	1
retainedFlag	false
payload	open: true, temperature: 23.5
dupFlag	false

Слика 4 Пример пакета за објаву на тему

На слици испод приказан је изглед пакета за одјаву са одређене теме (Слика 5).

Пакет за одјаву са теме	
packetId	12345
topic1	protocolx/123/somereqname
topic2	protocolx/123/anotherreqname

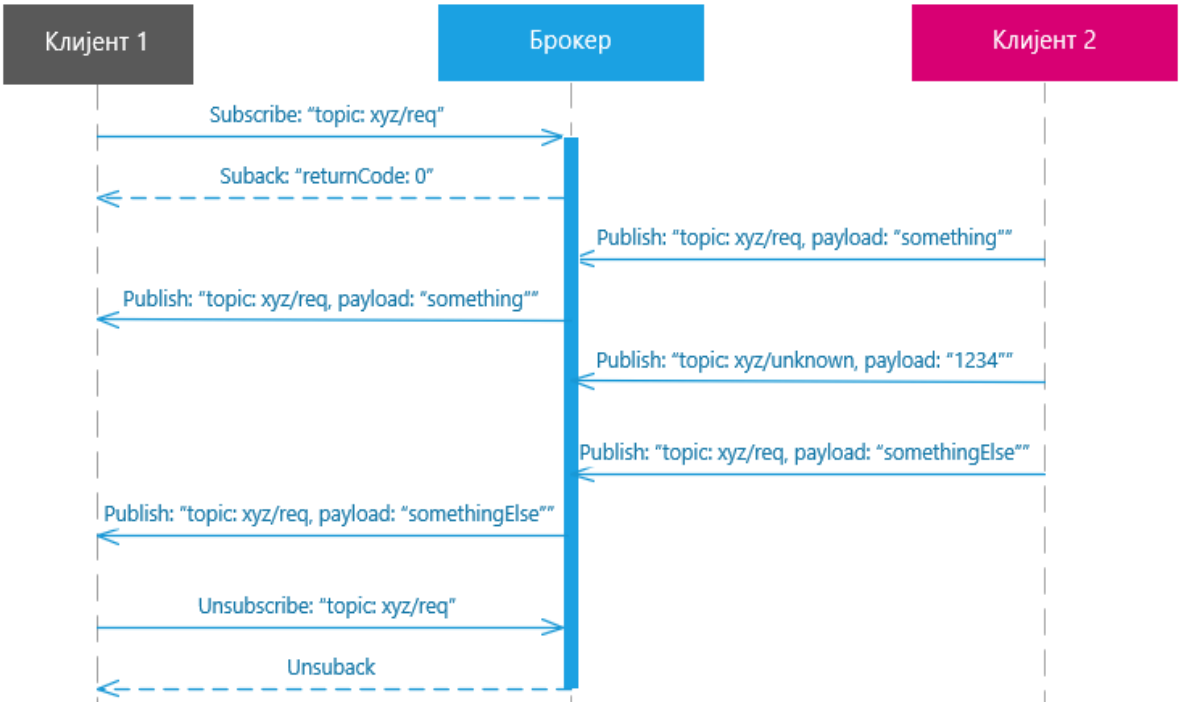
Слика 5 Пример пакета за одјаву са теме

На слици испод приказан је изглед пакета за потврду одјаве са одређене теме (Слика 6).

Пакет за потврду одјаве са теме	
packetId	12345

Слика 6 Пример пакета за потврду одјаве са теме

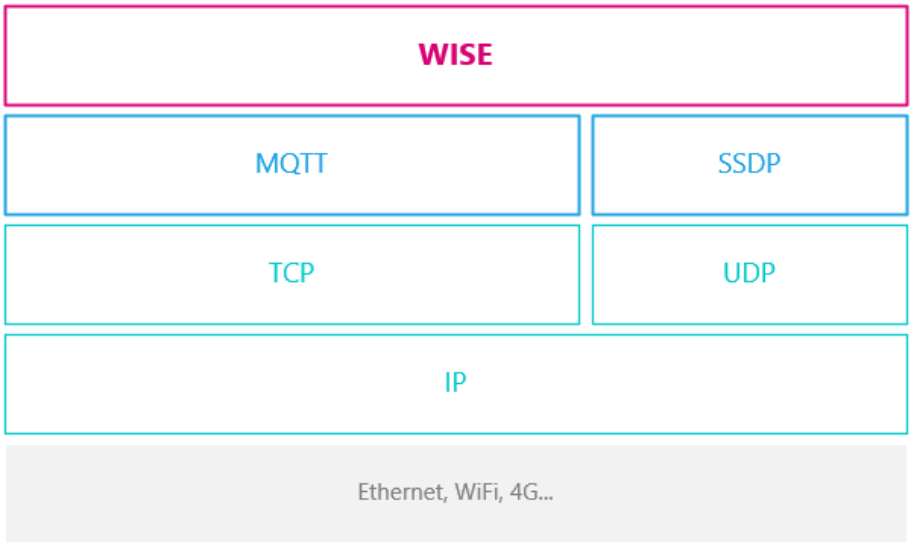
Пример комуникације између два клијента преко једног брокера приказан је на слици испод (Слика 7). Комуникација је приказана у виду дијаграма тока порука где су приказани случајеви успешног пријављивања на тему, успешне и неуспешне објаве на тему и одјаве са теме.



Слика 7 Пример комуникације између два клијента посредством брокера

2.2 *WISE* комуникациони протокол

WISE је протокол за комуникацију осмишљен да би се омогућила униформна комуникација између паметних уређаја који користе интернет протокол за комуникацију. *WISE* је заснован на *IP*-ју, а комуникација између уређаја се остварује преко *MQTT*-а (Слика 8).



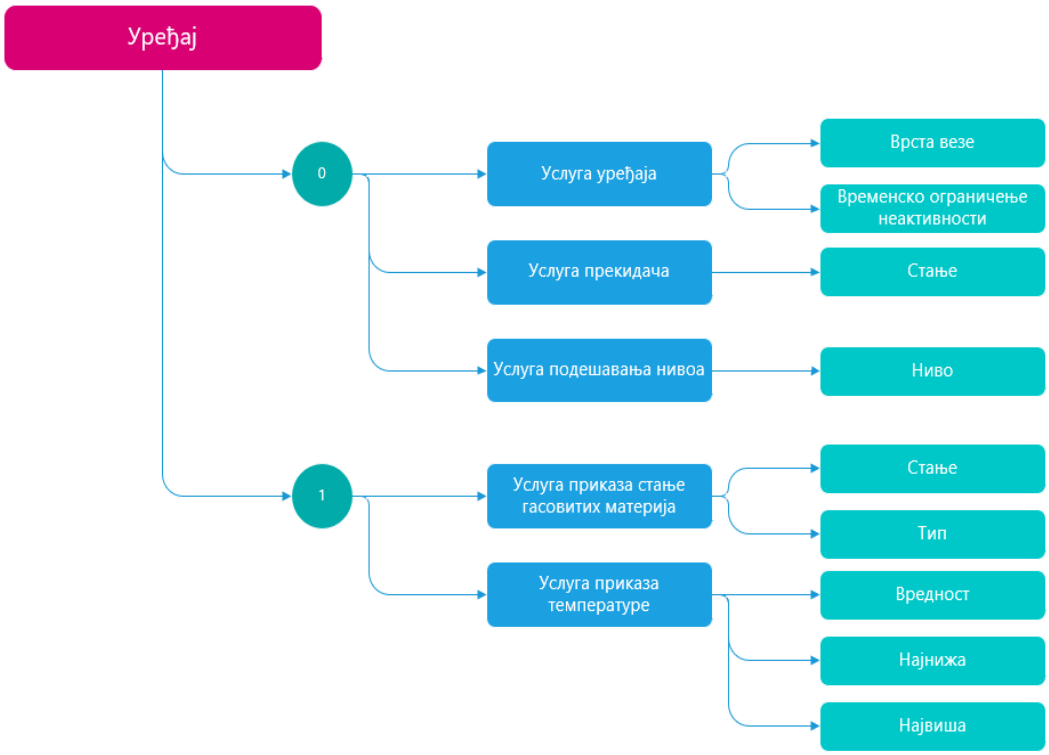
Слика 8 Група протокола на коју се ослања *WISE*

WISE подржава сензорске и актуаторске уређаје, као и комплексне уређаје који могу имати више различитих функционалности. Уређаје описује моделом заснованим на сервисима, где сваки уређај може подржавати једну или више услужних група (eng. "*Service Groups*"). Свака група може подржавати једну или више различитих услуга (eng. "*Service*"). Свака услуга се може састојати од једне или више особина (eng. "*Property*") које описују појединачне функционалности подржане одређеним уређајем. Осим особина, услуга може подржавати једну или више команди или захтева које централни контролер шаље ка уређају. На захтев контролера уређај мора одговорити поруком која садржи протоколом одређене параметре. Такође, услуга може имати дефинисане догађаје који служе за асинхроно обавештавање контролера о променама на уређају.

WISE објекат је логичка структура која описује један ентитет у *WISE* комуникацији дефинисана са једном или више услужних група. *WISE* протокол тренутно подржава два типа објекта:

1. Контролер (ознака за тему је "*hub*") – управља *WISE* мрежом и уређајима који се у њој налазе,
2. Уређај (ознака за тему је "*ipd*") – представља сензорски, актуаторски или комплексни уређај.

На слици испод (Слика 9) је приказан пример структуре *WISE* објекта. Розе бојом је пријазан *WISE* објекат, тамно тиркизно услужна група, плаво услуга и светло тиркизно особина.



Слика 9 Пример *WISE* структуре крајњег уређаја

На основу стандардних *MQTT* порука за пријаву и објаву, *WISE* дефинише директну размену порука између два објекта у систему преко захтева (eng. "*Request*") и одговора (eng. "*Response*"). Осим тога, *WISE* уводи догађаје (eng. "*Event*") као начин обавештавања претплатника о променама у систему. *WISE* прецизно дефинише *MQTT* теме ради препознавања објеката и њихових функционалности, као и да би се контролисала размена порука унутар система.

MQTT теме, у *WISE* комуникацији (Слика 10), су дефинисане на начин да се *WISE* објекти могу разазнати једни од других. Дефинисана тема такође прави разлику између функционалности једног објекта. Тема садржи:

1. *WISE* идентификатор – обавезни део теме који представља идентификатор *WISE* објекта и садржи:
    i. Идентификатор домена – домен размене порука (нпр. "*wise*", "rtrk", "ftn"),
    ii. Идентификатор мреже – означава мрежу у којој се налазе објекти који комуницирају,

iii. Тип објекта – означава тип објекта који комуницира (нпр. контролер или уређај),

iv. Идентификатор објекта.

2. Идентификатор услуге – опциони део теме који означава функционалност уређаја од интереса. Садржи:
   i. Услужну групу,
   ii. Тип услуге.
3. Тип поруке – обавезни део теме који означава тип поруке.

Слика 10 Тема у *WISE* комуникацији

Да би се омогућила контролисана размена порука, *WISE* протокол дефинише следеће типове порука:

1. Захтев (eng. "*Request*") – омогућава *WISE* објекту да захтева одређену акцију или извештај стања од другог објекта,
2. Одговор (eng. "*Response*") – одговор *WISE* објекта на примљени захтев,
3. Догађај (eng. "*Event*") – обавештавање претплатника о променама у систему,
4. Статус (eng. "*Status*") – обавештавање претплатника о променама у доступности *WISE* објекта унутар мреже.

На слици испод приказана је структура за тип поруке Захтев (Слика 11).

Слика 11 Структура поруке типа Захтев

На слици испод приказана је структура за тип поруке Одговор (Слика 12).

Слика 12 Структура поруке типа Одговор

На слици испод приказана је структура за тип поруке Догађај (Слика 13).

Слика 13 Структура поруке типа Догађај

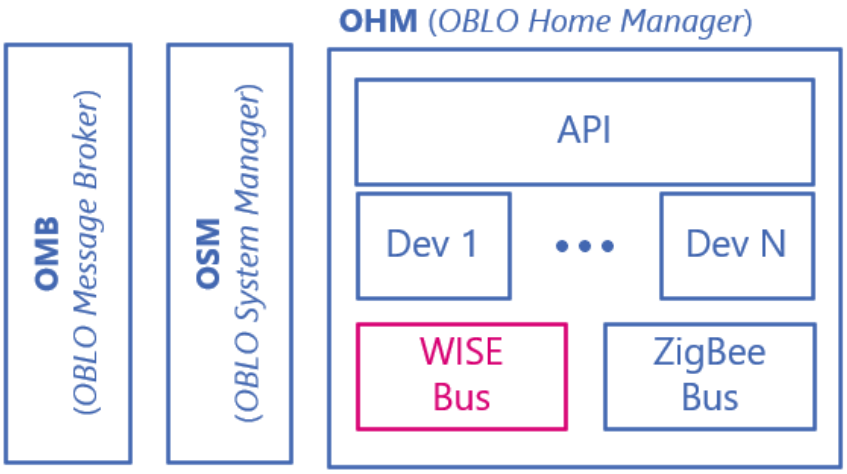
На слици испод приказана је структура за тип поруке Статус (Слика 14).

Слика 14 Структура поруке типа Статус

2.3 Централни контролер паметне куће

Централни контролер паметне куће служи за управљање уређајима у мрежи паметне куће, комуникацијом са сервисом у облаку и управљање системом којем контролер припада. Контролер, у својој основи, представља скуп апликација за управљање паметном кућом, развијене у језику Це плус плус (eng. "*C++*"), које су покренуте на одређеној циљаној хардверској архитектури. Апликације контролера могу бити покренуте на посебном уређају намењеном само управљању паметном кућом, али исто тако могу бити покренуте на локалном телевизијском претварачу (eng. "*Set Top Box*", "*STB*"), мрежном усмеривачу, персоналном рачунару, као и низу других платформи које су подржавају умрежавање.

Централни контролер подржава три основне апликације за управљање паметном кућом: *OHM*, *OSM* и *OMB*. Осим те три, постоји још једна апликација, *OSI* (eng. "Oblo Script Interpreter"). Она омогућава централном контролеру да покреће *JavaScript* скрипте које могу контролисати цео систем. *OSI* је и даље у развоју и не користи се активно на сваком контролеру. Блок дијаграм централног контролера налази се на слици испод (Слика 15).



Слика 15 Архитектура програмске подршке централног контролера

2.3.1 *OHM*

OHM је апликација централног контролера која садржи логичку представу крајњих уређаја, контролише крајње уређаје, очитава и чува њихова стања и задужен је за управљање надоградњом управљачке програмске подршке крајњих уређаја.

Осим тога, *OHM* садржи логичку апстракцију различитих врста мрежних протокола садржаних унутар паметне куће у виду логичких магистрала (eng. "*Bus*"), које омогућавају комуникацију са уређајима, као што су *ZigBee* [6], *Z-Wave* [7], *UPnP* [8] и многе друге. Логичка магистрала централног контролера битна за овај рад је *WISE* магистрала. По потреби, централни контролер може садржати само једну или више логичких магистрала, у зависности од потребе корисника или доступних ресурса на циљаној хардверској архитектури.

Сцене (eng. "*Scene*") унутар *OHM* апликације представљају могућност аутоматског управљања уређајима од стране централног контролера. Кориснику је омогућено да прецизно дефинише понашања крајњих уређаја, или самог контролера, као и да дефинише реакције на одређене промене у систему.

2.3.2 *OMB*

OMB представља апликацију централног контролера која омогућава комуникацију између апликација централног контролера, као и комуникацију самог контролера са сервисом у облаку и клијентском апликацијом на мобилном телефону. За комуникацију унутар система паметне куће, *OMB* користи *MQTT* комуникациони протокол и представља *MQTT* брокер у тој комуникацији. *OMB* је проширен подршком за *WISE* комуникацију, што омогућује оптимизацију ресурса на централном контролеру.

2.3.3 *OSM*

OSM представља апликацију централног контролера која управља хардверском архитектуром на којој се налази путем системских позива, контролише рад осталих апликација система, обавештава остале апликације уколико дође до промена у систему хардверске архитектуре, и управља улазима и излазима, а такође омогућава надоградњу апликација централног контролера.

2.4 Сервис у облаку

Сервис у облаку омогућава управљање корисниковим профилом, проверу идентитета, садржи информације о активним централним контролерима и повезује их са одређеним корисничким налозима, омогућава преузимање датотеке за евиденцију понашања

контролера и чува више верзија програмске подршке централног контролера које се касније могу користити током надоградње.

Сервис у облаку још може обавештавати и кориснике о стању система путем електронске поште или клијентских апликација. Када се клијентске апликације налазе на удаљеној локацији, ван локалне мреже у којој је и централни контролер, сервис у облаку служи апликацијама као *MQTT* брокер и опслужује их информацијама о централном контролеру и стању уређаја и омогућује њихову контролу.

2.5 Клијентске апликације

Клијентске апликације омогућавају корисницима управљање системом паметне куће путем мобилног телефона или интернет претраживача. Осим управљања системом, из клијентске апликације корисник може управљати својим налогом, правити распоред уређаја по зонама унутар објекта у ком се систем налази, давати дозволе за надоградњу апликација централног контролера и покретати надоградњу управљачке програмске подршке самих крајњих уређаја.

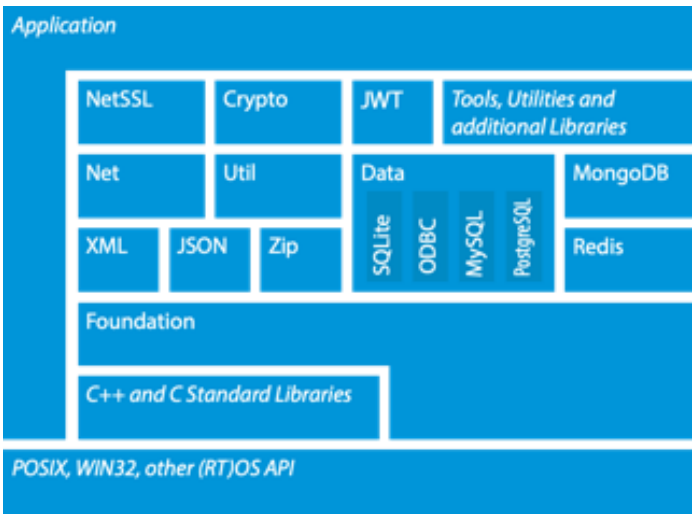
Клијентска апликација у мобилном телефону подржава Андроид (eng. "*Android*") и оперативни систем компаније Епл (eng. "*Apple*") *iOS*. Клијентска апликација на мобилном телефону остварује комуникацију са централним контролером паметне куће директно када је у локалној мрежи, користећи *OMB* као *MQTT* брокер, док када је ван локалне мреже користи брокер сервиса у облаку као посредника у комуникацији.

Клијентска апликација у интернет претраживачу је директно повезана са сервисом у облаку па тако информације о систему добија искључиво посредством брокера сервиса у облаку.

2.6 *POCO* Колекција библиотека

POCO [9] (eng. "*Portable Components*") колекција библиотека служи за развој Це плус плус апликација. Подржава развој апликација за различите хардверске архитектуре, а првенствено је намењена раду са рачунарским мрежама. Омогућава једноставан рад са мрежним утичницама, мрежним комуникационим протоколима, али и базама података, паметним показивачима, нитима... Ослања се на стандардну библиотеку шаблона (eng. "*Standard Template Library*") Це плус плус програмског језика.

Приказ библиотека које се могу наћи унутар *POCO* колекције библиотека може се видети на слици испод (Слика 16).



Слика 16 *POCO* колекција библиотека

2.7 База података

База података представља уређени скуп међусобно повезаних података по неком, од стране корисника, креираном дизајну. У систему паметне куће, централни контролер базу података користи за чување информација о уређајима, информација о конфигурацији аутоматског управљања системом, као и информација битних за сам контролер.

Централни контролер користи, као систем за управљање базом података (eng. "*Database Management System*"), релациони модел базе података, низ табела са редовима и колонама. Као језик за програмирање базе података, централни контролер користи *SQL* (eng. "*Structured Query Language*") и то за дефиницију базе, модификацију и повлачење података из исте.

С обзиром да је, у системима паметне куће, уштеда меморијског простора и процесорских ресурса од кључне важности, за рад са базом података и *SQL* упитима користи се *SQLite* [10] библиотека Це језика (eng. "*C*") апстракована кроз *API POCO* колекције библиотека.

3. Концепт решења

Пре имплементације решења на самим апликацијама централног контролера, потребно је осмислити дизајн самих измена. Измене описане у овом раду тичу се директно комуникационог протокола, па је самим тим пре имплементације потребно осмишљен дизајн уклопити у досадашњи дизајн протокола и сваку нову измену јасно и прецизно дефинисати.

Овај део рада описује проширивање самог *WISE* протокола подршком за уређаје за конверзију протокола, надоградњу управљачке програмске подршке уређаја и чување резервне копије конфигурације и података крајњих уређаја.

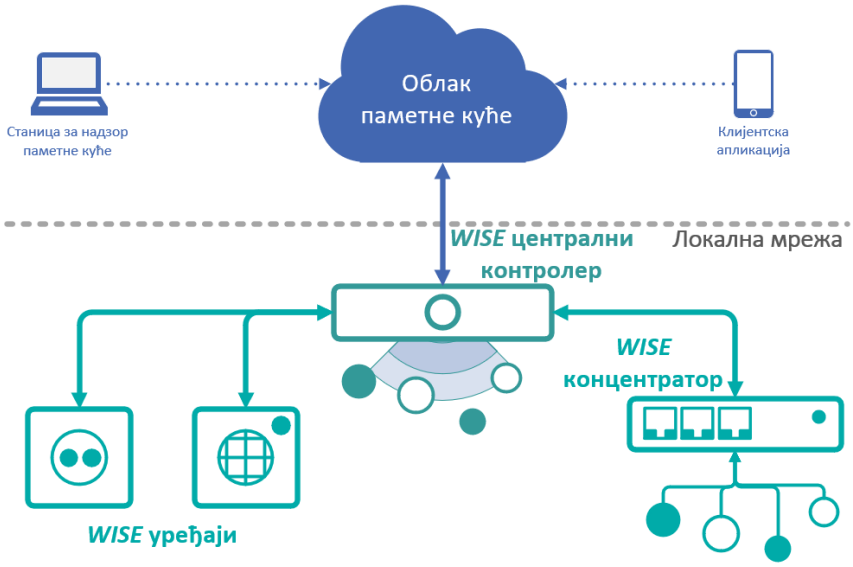
3.1 Уређај за конверзију протокола

Уређај за конверзију протокола, као што и сам назив каже, служи као посредник за комуникацију *WISE* централног контролера са уређајима који комуницирају путем неког другог комуникационог протокола, као на пример *ZigBee, Z-Wave* или *Bluetooth* [11].

Изглед система паметне куће са уређајем за конверзију протокола може се наћи на слици испод (Слика 17).

Додатни циљ овог рада је да се на лак начин интегришу други, интернет протоколом засновани, уређаји, тако што ће уградити подршку за *WISE* у свој производ. На произвођачима уређаја остаје да, уколико не користе *WISE* протокол, имплементирају уређај за конверзију протокола који би посредовао у комуникацији са њиховим уређајима.

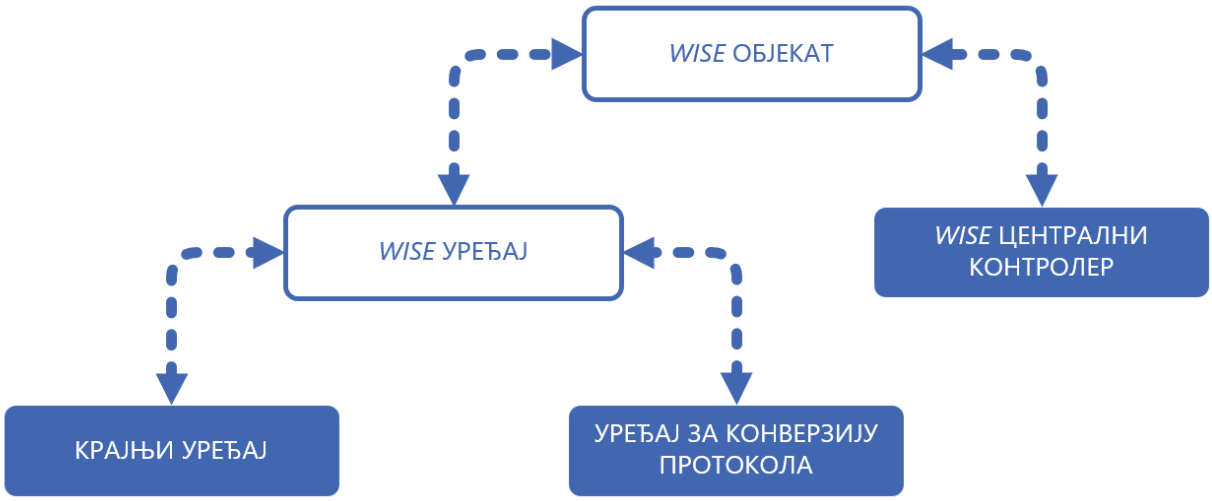
Уређај за конверзију протокола може да буде засебан уређај који би се преко локалне мреже повезао са централним контролером, а може се и апликација уређаја налазити на истом уређају као и апликације централног контролера па се повезати на брокер путем адресе локалне петље.



Слика 17 Уређај за конверзију протокола у мрежи паметне куће

3.1.1 Проширивање протокола

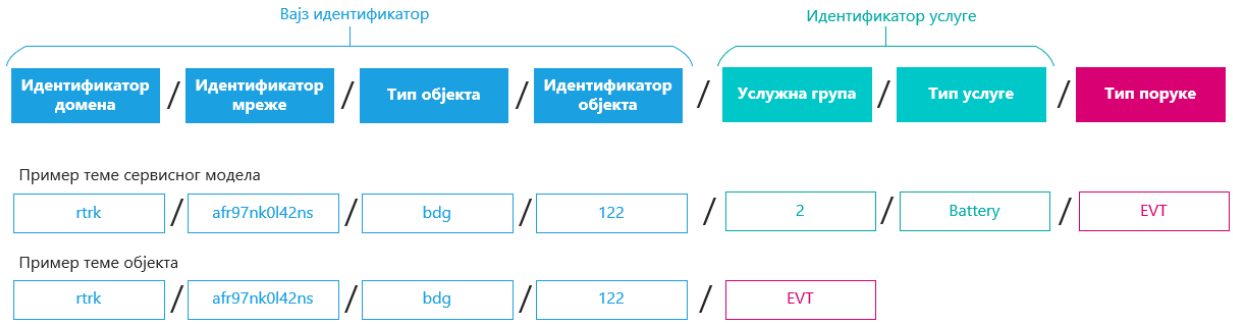
Први корак у дефиницији уређаја за конверзију протокола је његово прецизно дефинисање у склопу *WISE* објекта. *WISE* објекат је подељен на контролер и уређај. Уређај за конверзију протокола може се сматрати и уређајем, али и контролером, јер контролише уређаје који комуницирају другим комуникационим протоколима. Међутим, како се он укључује у мрежу, и централни контролер њега види као уређај, сматраће се да он има више својстава уређаја него контролера са становишта самог *WISE* протокола. С том идејом, објекат *WISE* уређај ће се поделити на крајњи уређај и уређај за конверзију протокола (Слика 18).



Слика 18 Уређај за конверзију протокола као део *WISE* објекта

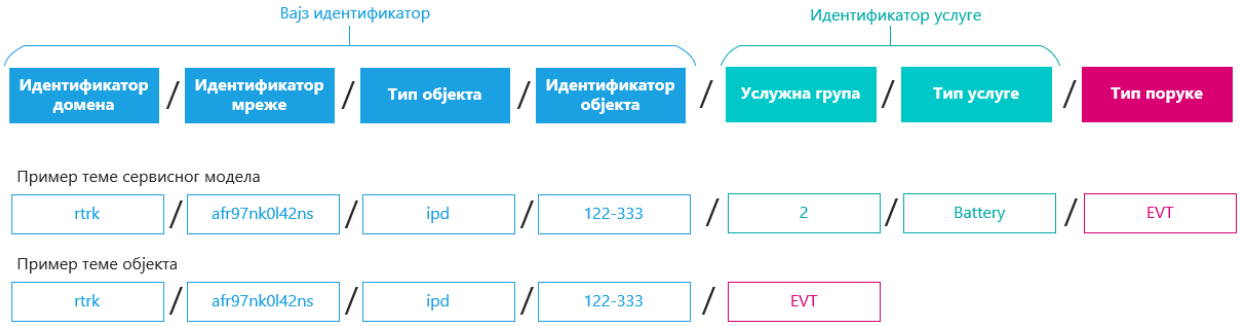
Да би уређај за контролу протокола могао комуницирати са централним контролером путем *WISE* протокола, потребно му је дефинисати ознаку за тип објекта, која би се користила као део *WISE* идентификатора при креирању теме. *WISE* протокол дефинише да ознака типа објекта буде "*bdg*". Пример *MQTT* теме коју уређај за конверзију протокола користи за комуникацију са брокером централног контролера приказана је на слици испод (Слика 19).

Након тога, дефинисан је начин на који ће се крајњи уређаји, чији саобраћај усмерава уређај за конверзију протокола, пријављивати централном контролеру, тј. под којим идентификатором објекта.



Слика 19 Пример теме уређаја за конверзију протокола

WISE протокол дефинише да идентификатор објекта за крајње уређаје буде комбинација идентификатора објекта уређаја за конверзију протокола и идентификатора објекта крајњег уређаја раздвојена цртицом ("-"). Пример *MQTT* теме, коју користи уређај за конверзију протокола да би послао захтев за један од уређаја чију комуникацију усмерава, приказан је на слици испод (Слика 20).



Слика 20 Пример теме уређаја који са контролером комуницира преко уређаја за конверзију протокола

Да би уређај за конверзију протокола могао објављивати и претплаћивати се нагоре наведене *MQTT* теме, потребно је да брокер централног контролера дозволи размену порука. То подразумева додавање одређених тема у контролну листу приступа брокера централног контролера. У табелама испод су наведене теме које је потребно додати у

контролну листу приступа да би уређај за конверзију протокола (Табела 1) могао размењивати поруке са контролером (Табела 2).

Бр.	Тема	Контролни пакет
1.	*<domain ID>/<home ID>/bdg/<bdg ID>/evt*	Објава
2.	*<domain ID>/<home ID>/bdg/<bdg ID>/ANY/ANY/evt*	Објава
3.	*<domain ID>/<home ID>/bdg/<bdg ID>/sts*	Објава
4.	*<domain ID>/<home ID>/ipd/ANY/req*	Објава
5.	*<domain ID>/<home ID>/ipd/ANY/rsp*	Објава
6.	*<domain ID>/<home ID>/ipd/ANY/ANY/ANY/req*	Објава
7.	*<domain ID>/<home ID>/hub/ANY/req*	Објава
8.	*<domain ID>/<home ID>/hub/ANY/rsp*	Објава
9.	*<domain ID>/<home ID>/hub/ANY/ANY/ANY/req*	Објава
10.	*<domain ID>/<home ID>/bdg/<bdg ID>/req*	Претплата
11.	*<domain ID>/<home ID>/bdg/<bdg ID>/ANY/ANY/req*	Претплата
12.	*<domain ID>/<home ID>/bdg/<bdg ID>/rsp*	Претплата
13.	*<domain ID>/<home ID>/ipd/ANY/evt*	Претплата
14.	*<domain ID>/<home ID>/ipd/ANY/ANY/ANY/evt*	Претплата
15.	*<domain ID>/<home ID>/ipd/ANY/sts*	Претплата
16.	*<domain ID>/<home ID>/hub/ANY/evt*	Претплата
17.	*<domain ID>/<home ID>/hub/ANY/ANY/ANY/evt*	Претплата
18.	*<domain ID>/<home ID>/hub/ANY/sts*	Претплата

Табела 1 Теме које користи уређај за конверзију протокола

Бр.	Тема	Контролни пакет
1.	*<domain ID>/<home ID>/hub/<hub ID>/evt*	Објава
2.	*<domain ID>/<home ID>/hub/<hub ID>/ANY/ANY/evt*	Објава
3.	*<domain ID>/<home ID>/hub/<hub ID>/sts*	Објава
4.	*<domain ID>/<home ID>/ipd/ANY/req*	Објава
5.	*<domain ID>/<home ID>/ipd/ANY/rsp*	Објава
6.	*<domain ID>/<home ID>/ipd/ANY/ANY/ANY/req*	Објава
7.	*<domain ID>/<home ID>/bdg/ANY/req*	Објава
8.	*<domain ID>/<home ID>/bdg/ANY/rsp*	Објава
9.	*<domain ID>/<home ID>/bdg/ANY/ANY/ANY/req*	Објава
10.	*<domain ID>/<home ID>/hub/<hub ID>/req*	Претплата
11.	*<domain ID>/<home ID>/hub/<hub ID>/ANY/ANY/req*	Претплата

Бр.	Тема	Контролни пакет
12.	*<domain ID>/<home ID>/hub/<hub ID>/rsp*	Претплата
13.	*<domain ID>/<home ID>/ipd/ANY/evt*	Претплата
14.	*<domain ID>/<home ID>/ipd/ANY/ANY/ANY/evt*	Претплата
15.	*<domain ID>/<home ID>/ipd/ANY/sts*	Претплата
16.	*<domain ID>/<home ID>/bdg/ANY/evt*	Претплата
17.	*<domain ID>/<home ID>/bdg/ANY/ANY/ANY/evt*	Претплата
18.	*<domain ID>/<home ID>/bdg/ANY/sts*	Претплата

Табела 2 Теме које користи централни контролер

Да би централни контролер пријављени уређај сматрао уређајем за конверзију протокола, неопходно је да уређај у нултој услужној групи има две услуге: услугу контролера и услугу уређаја. Услуга уређаја мора имати особине приказане у табели испод (Табела 3).

Бр.	Име	Приступ	Тип
1.	Временско ограничење неактивности (*sleepingTimeout*)	Читање	Број
2.	Протокол (*protocol*)	Читање	Број
3.	Верзија (*version*)	Читање	Знаковни низ

Табела 3 Особине услуге уређаја

Временско ограничење неактивности представља време које је дозвољено уређају да буде неактиван а да се и даље сматра као уређај који је присутан у мрежи. Протокол представља бројчану вредност верзије *WISE* протокола. До сада је подржавана искључиво вредност "*1*", а након увођења подршке за уређаје за конверзију протокола уводи се и вредност "*2*". Уређаји који не буду подржавали последњу верзију протокола имаће функционалности ограничене на верзију коју подржавају, али и даље могу бити део мреже. Верзија је текстуална представа верзије управљачке програмске подршке инсталиране на уређај.

Услуга контролера не поседује особине, али зато омогућава слање команди, захтева и објављивање догађаја (Табела 4).

Бр.	Име	Тип
1.	Додавање уређаја (*addDevice*)	Команда
2.	Уклањање уређаја (*removeDevice*)	Команда
3.	Преузимање листе уређаја (*getDeviceList*)	Захтев
4.	Преузимање информација о уређају (*getDevice*)	Захтев

Бр.	Име	Тип
5.	Преузимање листе идентификатора свих уређаја (*getDeviceIds*)	Захтев
6.	Уређај додат (*deviceAdded*)	Догађај
7.	Уређај уклоњен (*deviceRemoved*)	Догађај
8.	Промена стања процеса укључивања у мрежу (*inclusionStateChanged*)	Догађај

Табела 4 Команде, захтеви и догађаји услуге контролера

Команда за додавање уређаја (“*addDevice*”) покреће процес додавања уређаја у мрежу централног контролера. Том командом уређај за конверзију протокола покреће процес додавања крајњег уређаја чији мрежни саобраћај усмерава. Уз команду за додавање уређаја долази и параметар временског ограничења мреже (“*networkTimeout*”) који означава колико времена је мрежа спремна да прихвати нови уређај. Команда за уклањање уређаја (“*removeDevice*”) покреће процес уклањања крајњег уређаја из *WISE* мреже централног контролера. Да би било познато који уређај треба уклонити, уз команду се шаље параметар идентификатора уређаја (“*id*”).

Захтев за преузимање листе уређаја (“*getDeviceList*”) омогућава централном контролеру да преузме листу свих уређаја, чији саобраћај уређај за конверзију протокола усмерава. Повратна вредност овог захтева мора бити параметар листа уређаја (“*deviceList*”) који садржи листу свих уређаја у *JSON* формату. Захтев за преузимање информација о уређају (“*getDevice*”) омогућава централном контролеру да преузме детаљан опис стања крајњег уређаја. Информацију о томе који уређај је у питању пружа параметар идентификатор објекта (“*id*”), а повратна вредност овог захтева мора садржати параметар уређај (“*device*”) који у *JSON* формату садржи информације о уређају. Захтев за преузимање листе идентификатора свих уређаја (“*getDeviceIds*”), као што и сам назив каже, омогућава контролеру да преузме листу идентификатора свих уређаја чији саобраћај се усмерава. Повратна вредност мора бити параметар идентификатори уређаја (“*deviceIds*”) који представља листу свих уређаја.

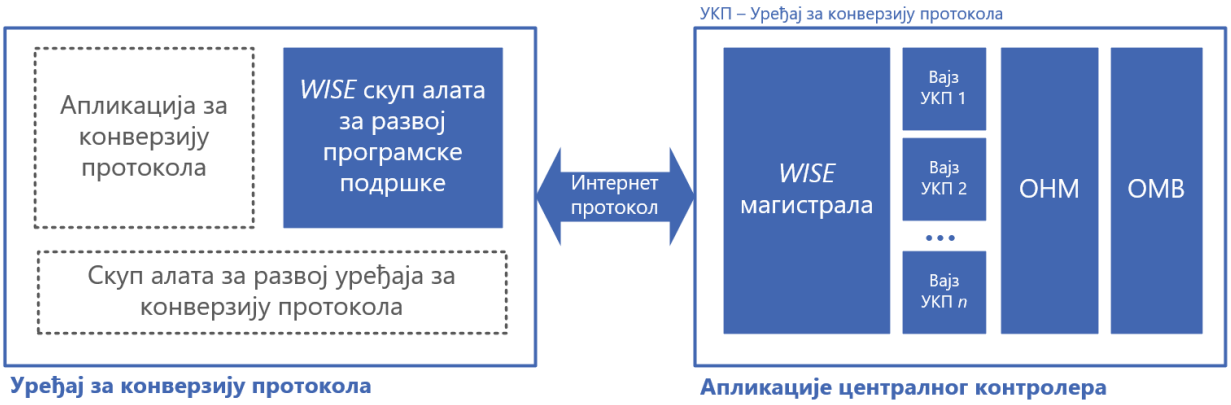
Догађај услуге контролера уређај додат (“*deviceAdded*”) мора се објавити када се успешно изврши додавање уређаја у *WISE* мрежу централног контролера са параметром идентификатор (“*id*”) који означава идентификатор додатог уређаја. Када се из *WISE* мреже успешно уклони уређај објављује се догађај уређај уклоњен (“*deviceRemoved*”) са параметром идентификатор (“*id*”) који означава идентификатор објекта уређаја који је уклоњен. Када се промени стање процеса укључивања у *WISE* мрежу, на централном контролеру објављује се догађај промена стања процеса укључивања у мрежу

(“*inclusionSateChanged*”) са параметром стање (“*state*”). Стање може бити: мировање (“*idle*”), оглашавање (“*advertising*”) и прикључен мрежи (“*joined*”).

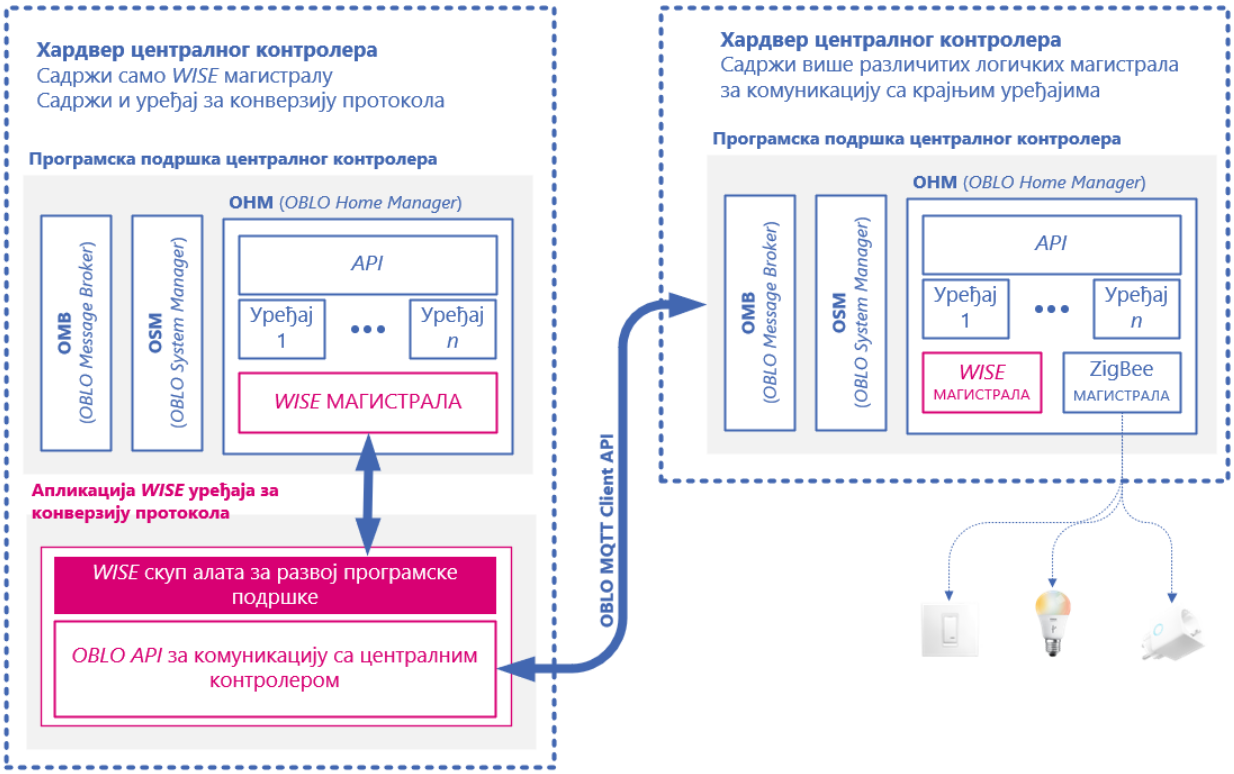
3.1.2 Дизајн уређаја

Дизајн самог уређаја за конверзију протокола је одговорност произвођача уређаја. Ипак, због тестирања измена *WISE* протокола, реализован један пример уређаја који би показао могућности које ово проширење протокола пружа.

Пример програмске архитектуре уређаја за конверзију протокола и његове комуникације са контролером приказан је на слици испод (Слика 21).



Слика 21 Пример програмске архитектуре уређаја за конверзију протокола



Слика 22 Архитектура апликација и комуникација између истих

Уређај за конверзију протокола, који у овом раду служи као пример једне од могућности коришћења, повезује два централна контролера. Један контролер има на себи омогућену само *WISE* логичку магистралу док други има омогућену *ZigBee* магистралу. Циљ је да уређај са *WISE* логичком магистралом прикаже и омогући контролу уређаја који се налазе на контролеру са *ZigBee* магистралом. Уређај за конверзију протокола се налази на истом физичком уређају као и контролер са *WISE* магистралом. Дијаграм апликација и комуникације између истих се може наћи на слици изнад (Слика 22).

3.2 Надоградња управљачке програмске подршке преко мреже

Надоградња управљачке програмске подршке преко мреже (eng. "*Over-The-Air Upgrade*", "*OTA*") представља преузимање најновије верзије управљачког софтвера крајњег уређаја и његова инсталација. *WISE* протокол до сада није подржавао надоградњу управљачког софтвера, па је циљ био проширити протокол подршком за исту.

Првенствено је потребно проширити *WISE* спецификацију услугом надоградње управљачког софтвера. Та услуга није обавезна, већ је произвођач може користити у свом уређају уколико подржава функционалност надоградње управљачке програмске подршке. Услуга надоградње управљачког софтвера нема подржане специфичне особине, а подржане команде, захтеви и догађаји су наведени у табели испод (Табела 5).

Команда за отпочињање надоградње управљачког софтвера ("*startUpgrade*") отпочиње процес надоградње. Да би процес био отпочет, команда мора садржати верзију софтвера за надоградњу ("*version*"), место са ког се датотека са програмском подршком може преузети ("*url*") и шифровану лозинку датотеке ("*md5*") као параметре команде. Команда за заустављање надоградње ("*stopUpgrade*") нагло прекида надоградњу без обзира на ступањ у ком се она налази.

Бр.	Име	Тип
1.	Отпочињање надоградње (*startUpgrade*)	Команда
2.	Заустављање надоградње (*stopUpgrade*)	Команда
3.	Понуда надоградње на нову верзију (*newFirmwareAvailable*)	Захтев
6.	Статус надоградње (*upgradeStatus*)	Догађај
7.	Надоградња неуспешна (*upgradeFailed*)	Догађај
8.	Индикатор тока надоградње (*upgradeProgress*)	Догађај

Табела 5 Команде, захтеви и догађаји услуге надоградње управљачког софтвера

Понуда надоградње на нову верзију (“*newFirmwareAvailable*”) је захтев који контролер упућује уређају са параметром који служи као идентификатор верзије софтвера (“*version*”). На тај захтев уређај мора одговорити параметром који информише контролер о потврди пристанка (“*accepted*”).

Статус надоградње (“*upgradeStatus*”) је догађај којим уређај обавештава контролер у ком стадијуму надоградње се уређај налази. То је обезбеђено параметром статус (“*status*”) који означава једно од пет могућих стања процеса надоградње:

i. отпочето (“*started*”),
ii. заустављено (“*stopped*”),
iii. датотека преузета (“*downloaded*”),
iv. верификовано (“*verified*”) и
v. завршено (“*done*”).

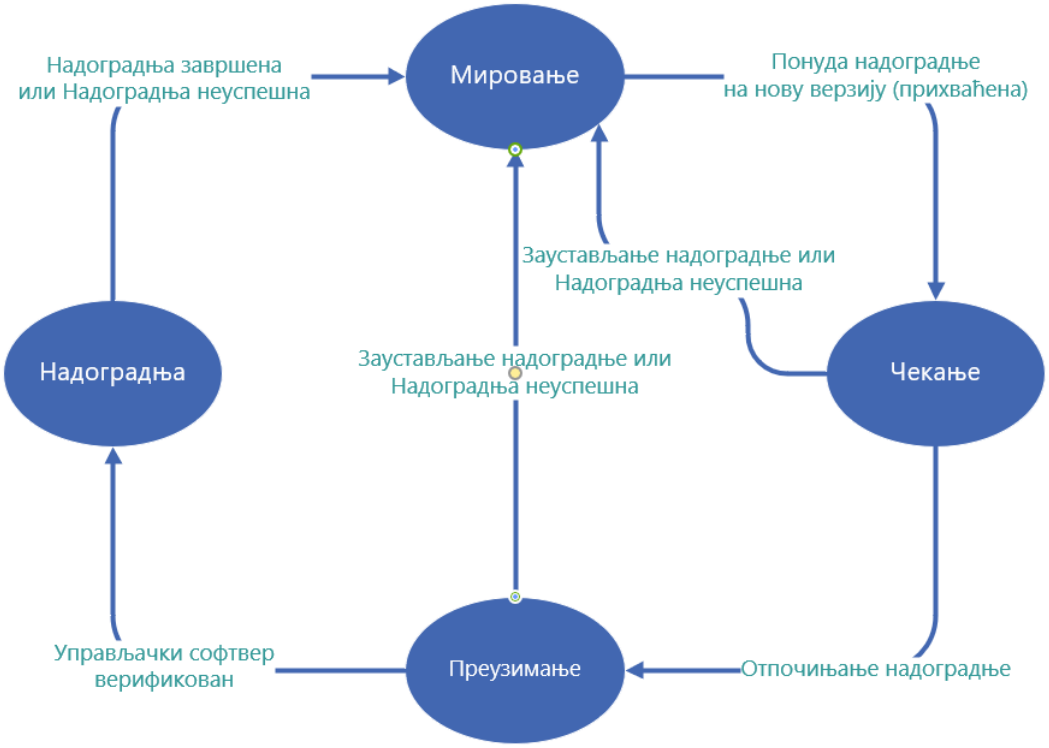
Надоградња неуспешна (“*upgradeFailed*”) представља догађај којим се контролер обавештава да надоградња управљачког софтвера није успела. Уз тај догађај се прослеђује и параметар о коду грешке (“*code*”) која се десила током надоградње. Индикатор тока надоградње (“*upgradeProgress*”) представља догађај који уређај може слати током процеса надоградње и њиме обавештава контролер о напретку процеса надоградње. Параметар који носи ту информацију зове се напредак (“*progress*”) и исказује се бројчаном вредношћу од нула до сто у процентима.

Током процеса надоградње управљачке програмске подршке, централни контролер се може наћи у четири стања:

i. мировање (“*idle*”),
ii. припрема (“*preparing*”),
iii. отпремање (“*uploading*”) и
iv. надоградња (“*upgrading*”).

Дијаграм стања у којима се може наћи централни контролер се могу наћи на слици испод (Слика 23).

Слика 23 Дијаграм стања контролера током процеса надоградње управљачког софтвера



Слика 24 Дијаграм стања уређаја током процеса надоградње управљачког софтвера

Крајњи уређај се, током процеса надоградње управљачког софтвера, може наћи у четири различита стања:

i. мировање (“*idle*”),

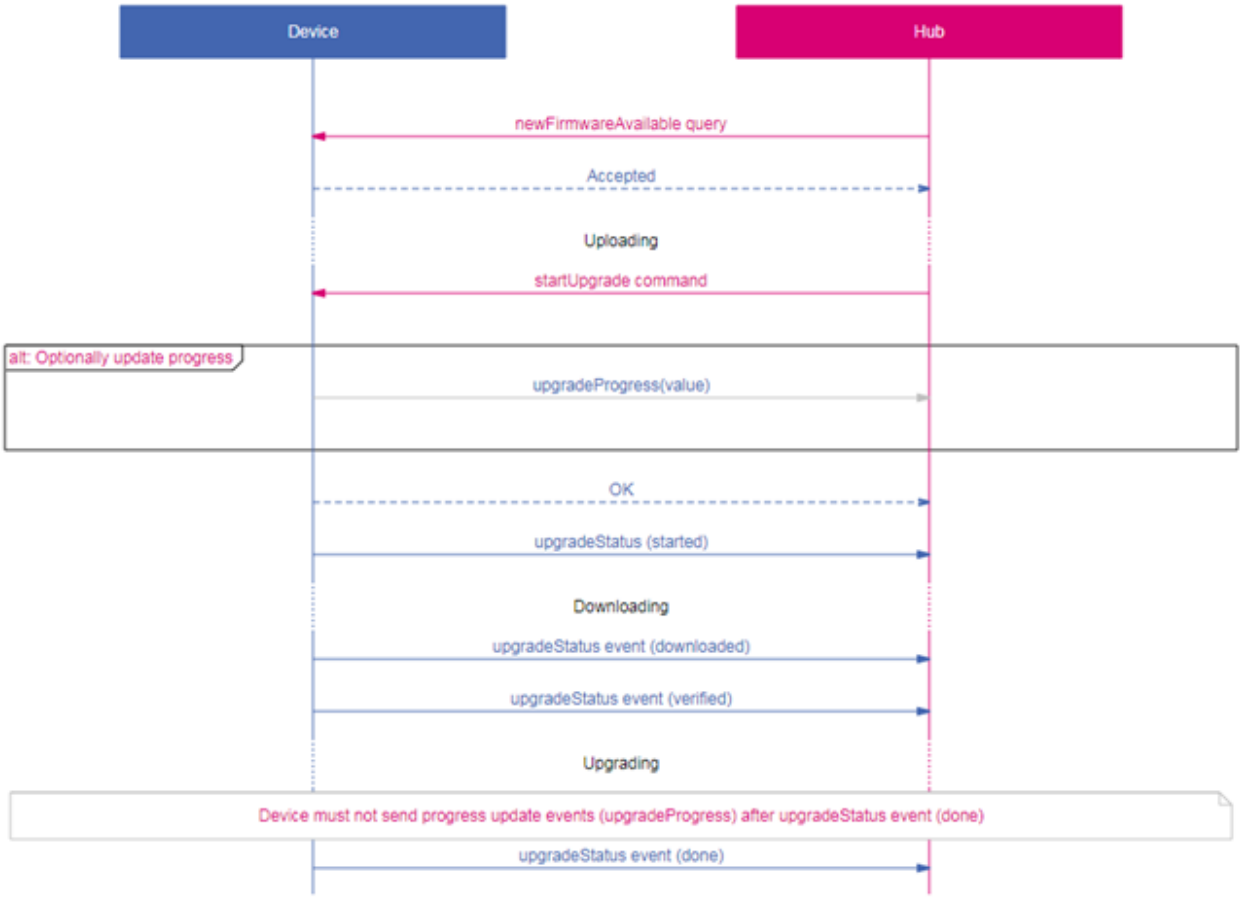
ii. чекање (“*waiting*”),

iii. преузимање (“*downloading*”) и

iv. надоградња (“*upgrading*”).

Дијаграм стања уређаја током овог процеса налази се на слици изнад (Слика 24).

На слици испод приказан је пример дијаграма тока порука који се одвија током процеса надоградње управљачке програмске подршке (Слика 25).



Слика 25 Пример дијаграма тока порука током процеса надоградње управљачког софтвера

3.3 Чување резервне копије података

Чување резервне копије података циља да омогући корисницима да у тренутку када уређај ради свој посао без грешака, сачувају последњу конфигурацију уређаја заједно са његовим подацима. Тиме се у случају грешке сачувана конфигурација и подаци могу реконструисати, и тиме омогућити поновно исправно функционисање уређаја.

Пре реализације овог рада, *WISE* протокол није подржавао чување резервне копије података за сваки појединачни уређај.

Пре проширења програмске подршке контролера, потребно је проширити *WISE* протокол подршком за чување резервне копије података. Потребно је додати нову услугу, у *WISE* спецификацију, под називом услуга чувања резервне копије. Сваки уређај који подржава ову могућност мора имати дефинисану услугу чувања резервне копије у нултој

услужној групи. Ова услуга се користи и за реконструкцију неке од претходно сачуваних резервних копија, па тиме обухвата и чување и реконструкцију резервне копије података, али их третира као два одвојена процеса. Оба процеса могу се обављати на два начина:

1. Локално – Уређај је задужен за чување података, а контролеру се прослеђује само идентификатор процеса помоћу кога би се касније пронашла жељена датотека у меморији самог уређаја,
2. Са удаљене локације – Централни контролер је задужен за чување података. Током овог процеса од уређаја се захтева да отпреми датотеку са подацима централном контролеру. Када се врши враћање података, уређај мора да преузме жељене податке са контролера.

Услуга чувања резервне копије података има две особине: начин рада (“*mode*”) и време последњег чувања података (“*lastBackup*”). Уређај особину начин рада мора поставити одмах по уласку у *WISE* мрежу на једну од три подржане вредности: локално (“*local*”), са удаљене локације (“*remote*”), или локално и са удаљене локације (“*local and remote*”). Осим тога, ова услуга подржава и шест команди, један захтев и осам догађаја који се могу видети у табели испод (Табела 6).

Бр.	Име	Тип
1.	Отпочињање локалног чувања (*startLocalBackup*)	Команда
2.	Отпочињање чувања на удаљену локацију (*startRemoteBackup*)	Команда
3.	Прекид чувања (*abortBackup*)	Команда
4.	Отпочињање локалне реконструкције (*startLocalRestore*)	Команда
5.	Отпочињање реконструкције са удаљене локације (*startRemoteRestore*)	Команда
6.	Прекид реконструкције (*abortRestore*)	Команда
7.	Отпочињање чувања (*startBackup*)	Захтев
8.	Чување почело (*backupStarted*)	Догађај
9.	Индикатор тока чувања (*backupProgress*)	Догађај
10.	Чување завршено (*backupFinished*)	Догађај
11.	Прекид чувања (*abortBackup*)	Догађај
12.	Реконструкција почела (*restoreStarted*)	Догађај
13.	Индикатор тока реконструкције (*restoreProgress*)	Догађај
14.	Реконструкција завршена (*restoreFinished*)	Догађај
15.	Прекид реконструкције (*abortRestore*)	Догађај

Табела 6 Команде, захтеви и догађаји услуге чувања резервне копије

Када уређај прими команду за отпочињање локалног чувања (“*startLocalBackup*”) од контролера, мора да почне процес локалног чувања резервне копије података. Како се подаци чувају на самом уређају, нема потребе за слањем било каквих параметара уз ову команду. Команда отпочињање чувања на удаљену локацију (“*startRemoteBackup*”) мора покренути на уређају процес чувања резервне копије података на удаљеној локацији. Том приликом се шаље параметар локација (“*location*”) који означава место на које ће се отпремити датотека са подацима. Команда за прекид чувања (“*abortBackup*”) обавештава уређај да одмах прекине чување резервне копије без обзира на то у ком се стању процеса уређај налазио.

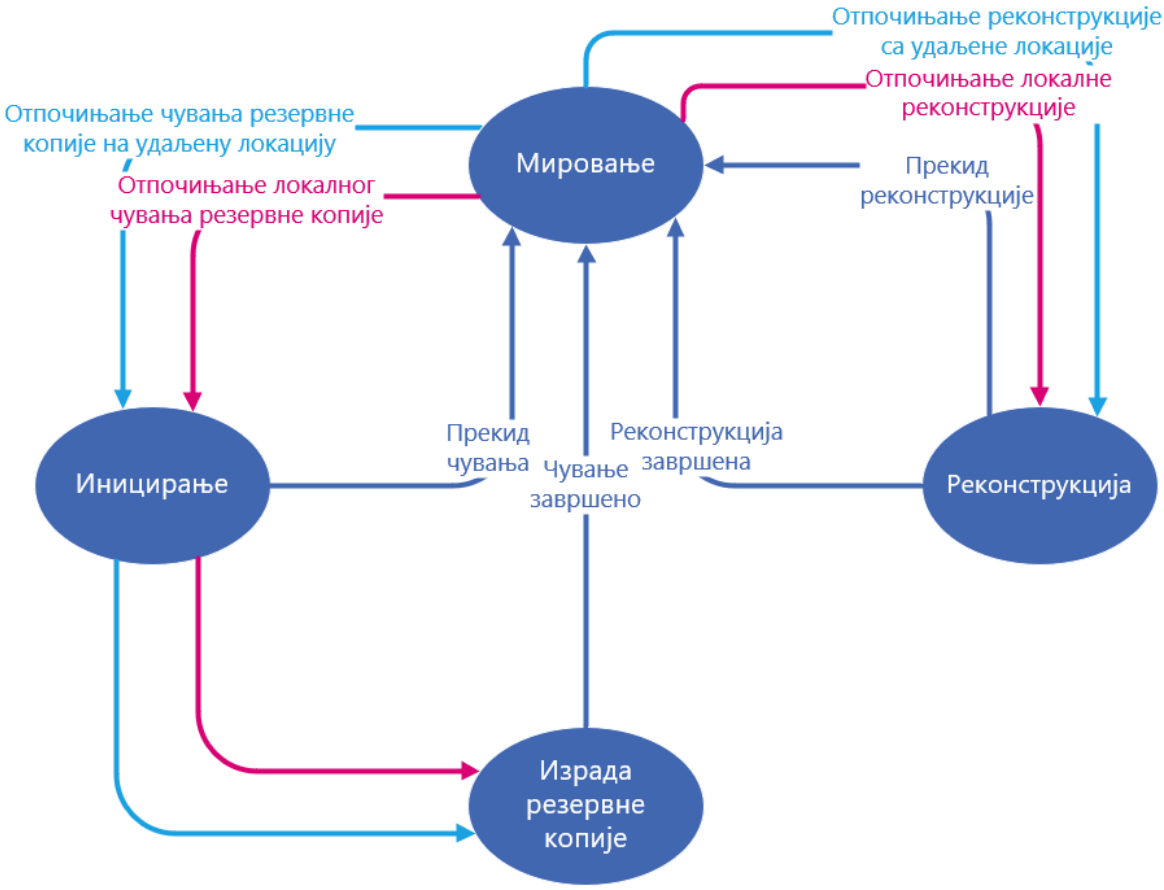
Команда за почетак локалне реконструкције (“*startLocalRestore*”) мора отпочети процес локалне реконструкције података. Том приликом се прослеђује параметар идентификатор процеса (“*bid*”) како би уређај знао којом датотеком да изврши реконструкцију. Команда за почетак реконструкције са удаљене локације (“*startRemoteRestore*”) почиње процес реконструкције података на уређају са удаљене локације. Уз ту команду прослеђују се и два параметра, идентификатор процеса (“*bid*”) и локација (“*location*”), како би уређај знао одакле да преузме датотеку са одређеним идентификатором процеса. Када уређај добије команду за прекид реконструкције (“*abortRestore*”), мора одмах да прекине процес независно од стања у ком се уређај налази.

Захтев за отпочињање чувања (“*startBackup*”) представља упит који централни контролер шаље уређају да би проверио да ли је уређај спреман за процес чувања резервне копије података. На тај захтев уређај мора одговорити параметром тип (“*type*”) који означава да ли се процес прихвата, и који тип процеса, и може имати једну од четири вредности: локално (“*local*”), са удаљене локације (“*remote*”), није потребно (“*not needed*”) и није подржано (“*not supported*”).

Догађај који означава да је процес чувања резервне копије почео (“*backupStarted*”) мора се послати одмах након обраде поруке за почетак једног од два типа процеса чувања (“*startLocalBackup*” или “*startRemoteBackup*”), да би централни контролер знао да је процес успешно отпочет. Догађај који служи као индикатор тока процеса чувања (“*backupProgress*”) може се слати током тог процеса контролеру. Тим догађајем контролер добија информацију о томе у ком стадијуму процеса се уређај налази, а параметар који то означава је напредак (“*progress*”) и он представља бројчану вредност од нула до сто у процентима. Чување завршено (“*backupFinished*”) је догађај који уређај шаље након успешног завршетка процеса чувања резервне копије података. Том приликом се као параметар прослеђује индикатор процеса (“*bid*”) који означава датотеку која ће се после реконструисати. Чување нагло прекинуто (“*backupFailed*”) је догађај који се шаље у случају

изненадног прекида процеса. Уз њега се шаље параметар код грешке (“*code*”) који може имати вредности: време чекања је истекло (“*waiting timeout expired*”), креирање датотеке није успело (“*creating package failed*”), отпремање није успело (“*upload failed*”) и непознат разлог (“*unknown reason*”).

Догађај који означава да је реконструкција почела (“*restoreStarted*”) објављује се одмах након обраде поруке за почетак једног од процеса за реконструкцију података (“*startLocalRestore*” или “*startRemoteRestore*”) како би контролер знао да је процес успешно отпочет. Индикатор тока процеса реконструкције (“*restoreProgress*”) је догађај који може бити послат током процеса реконструкције. Он садржи параметар напредак (“*progress*”) који означава који је напредак тока реконструкције на уређају у бројчаној вредности од нула до сто, изражене у процентима. Реконструкција завршена (“*restoreFinished*”) је догађај који мора бити послат одмах након успешног завршетка процеса реконструкције података на уређају. Реконструкција неуспешна (“*restoreFailed*”) је догађај који се шаље уколико се процес реконструкције изненада заврши. Уз догађај се прослеђује параметар код грешке (“*code*”) који може имати вредности: време чекања је истекло (“*waiting timeout expired*”), преузимање неуспешно (“*download failed*”), датотека садржи грешку (“*corrupted file*”), погрешна датотека (“*invalid file*”) и непознат разлог (“*unknown reason*”).



Слика 26 Дијаграм стања контролера током процеса чувања резервне копије података

Током процеса чувања резервне копије података, контролер се може наћи у четири различита стања:

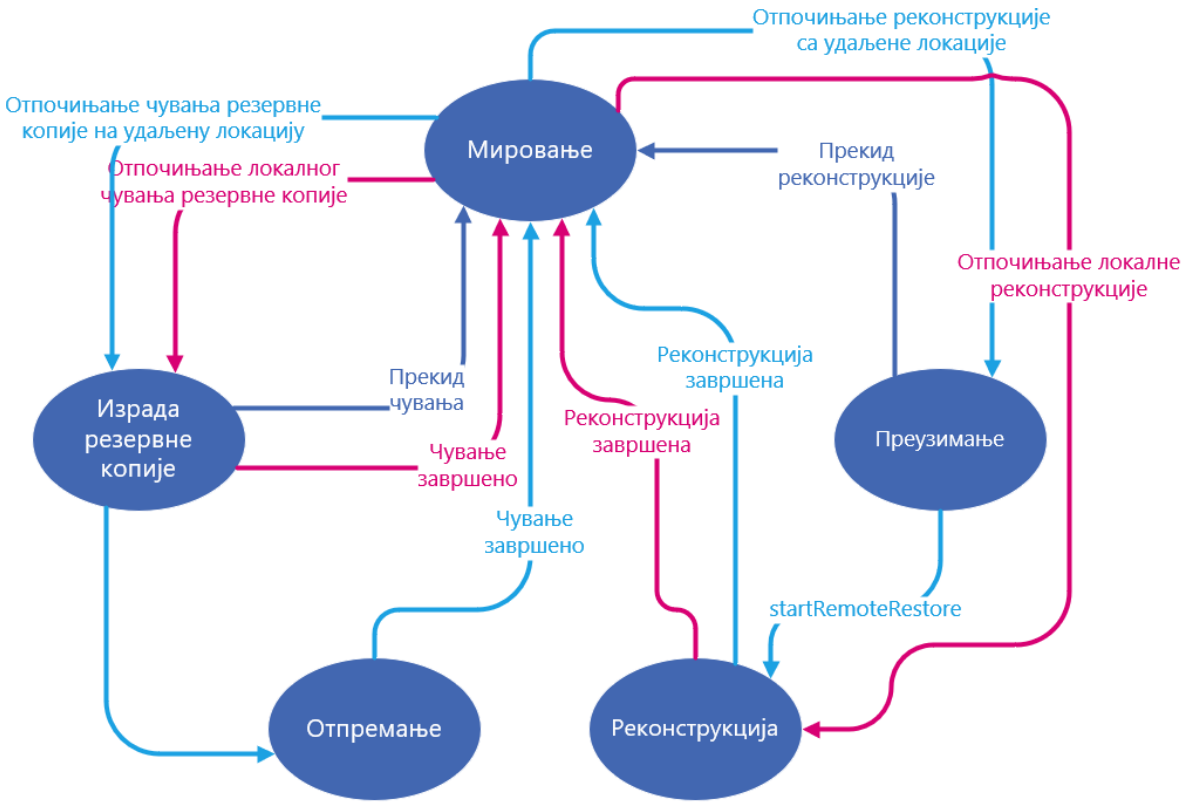
i. мировање (“*idle*”),
ii. иницирање (“*initiating*”),
iii. израђивање резервне копије (“*backup*”) и
iv. реконструкција (“*restoring*”).

Дијаграм стања контролера током овог процеса може се наћи на слици изнад (Слика 26).

Током процеса чувања резервне копије података, уређај се може наћи у пет различитих стања:

i. мировање (“*idle*”),
ii. израђивање резервне копије (“*backup*”),
iii. отпремање (“*uploading*”), преузимање (“*downloading*”) и
iv. реконструкција (“*restoring*”).

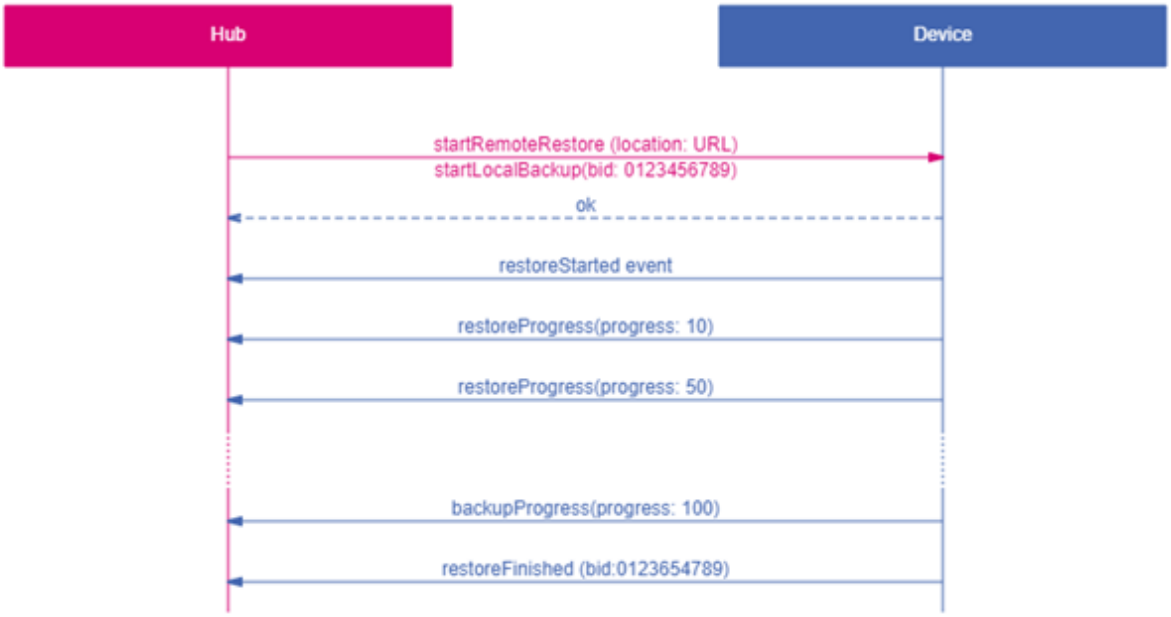
Дијаграм стања контролера током овог процеса може се наћи на слици испод (Слика 27).



Слика 27 Дијаграм стања уређаја током процеса чувања резервне копије података

Примери дијаграма тока порука за чување резервне копије (Слика 28) и враћање претходно сачуване резервне копије (Слика 29) могу се наћи на сликама испод.

Слика 28 Пример дијаграма тока порука за чување резервне копије



Слика 29 Пример дијаграма тока порука за враћање претходно сачуване резервне копије

4. Програмско решење

У овом поглављу ће бити описан начин на који је имплементирана подршка за уређај за конверзију протокола на апликацијама централног контролера, како је имплементиран сам пример уређаја за конверзију протокола, како је имплементирана програмска подршка за надоградњу управљачког софтвера и чување резервне копије података, као и проширење примера крајњих уређаја подршком за надоградњу управљачког софтвера и чување резервне копије података.

4.1 Подршка за уређај за конверзију протокола на апликацијама централног контролера

Да би уређај за конверзију протокола могао комуницирати са централним контролером, потребно је било додати дозволе (Табела 1 и Табела 2) у листу приступа *OMB* брокера и потребно је да се локални *WISE MQTT* клијент, који представља спону *OMB* апликације са *WISE* магистралом, претплати на слушање пристиглих порука на *OMB* брокер (Слика 30).

```
_Subscriptions.push_back(CClient::Subscription(Poco::format("%s/%s/bdg/+/sts"
                         , m_szRemoteDomainId, _szAccount, _szHubId)
                         , Mqtt::CMessage::QOS1));
_Subscriptions.push_back(CClient::Subscription(Poco::format("%s/%s/bdg/+/evt"
                         , m_szRemoteDomainId, _szAccount, _szHubId)
                         , Mqtt::CMessage::QOS1));
_Subscriptions.push_back(CClient::Subscription(Poco::format("%s/%s/bdg/+/+/+/evt"
                         , m_szRemoteDomainId, _szAccount, _szHubId)
                         , Mqtt::CMessage::QOS1));
```

Слика 30 Претплата на слушање порука пристиглих од стране уређаја за конверзију протокола

Да би уређај за конверзију протокола могао да управља *OHM* апликацијом централног контролера путем услуге контролер, *WISE* магистралу је потребно проширити на начин да, када уређај за конверзију протокола пошаље захтев за додавање или уклањање крајњег уређаја, позову се функције регистроване на магистрали за обраду тих захтева. Оне користе већ постојећи модел *WISE* уређаја који користи и централни контролер, па тако у базу уређаја пријављених на централни контролер додају и крајњи уређај (Слика 31), у протоколом дефинисаном формату, или га из исте уклањају (Слика 32).

```
CDeviceObject::Ptr temp = *itr;
std::string deviceId = bridgeId + "-" + temp->GetId();
OBJID objid = CreateDevice(temp->GetDeviceType(), deviceId, true);
Device::CWiseDevice* dev = GetWiseDevice(deviceId);
```

Слика 31 Креирање крајњег уређаја пријављеног од уређаја за конверзију протокола

```
Async::CRemoveBridgeDevice& cmd = static_cast<Async::CRemoveBridgeDevice&>
                                  (*_pCommand);
Device::CWiseDevice* bridge = GetWiseDeviceById(_DevId);
if (bridge) {
    std::string deviceId = bridge->GetAddress() + "-" + cmd.GetDeviceId();
    Device::CWiseDevice* wbdev = GetWiseDevice(deviceId);
    if (wbdev) {
        SendExclusionRequest(wbdev);
        m_pDeviceManager->DeleteDevice(wbdev->GetObjId());
    }
}
```

Слика 32 Брисање крајњег уређаја из *WISE* мреже

Класа *WISE* Уређај је проширена тако да може садржати листу идентификатора крајњих уређаја, чији саобраћај уређај за конверзију протокола усмерава.

Да би централни контролер могао распознати да ли је догађај, команда или захтев објављен на *MQTT* тему крајњег уређаја, уређаја за конверзију протокола или крајњег уређаја који је додат од стране уређаја за конверзију протокола, потребно је било додати проверу идентификатора објекта (Слика 33). Уколико објекат, који објављује на тему од интереса, има идентификатор крајњег уређаја садржан у склопу дела теме који означава идентификатор објекта, потребно је саставити идентификатор уређаја по ком би се, из базе уређаја садржане на контролеру, пронашла инстанца објекта класе *WISE* Уређај и над њом позвала тражена функција (Слика 34).

```
CSenderTopic::Ptr sender = CSenderTopic::Parse(_szSender);
Event::CPropertyChanged::Ptr evt = Event::CPropertyChanged::Parse(_Event);
evt->SetSender(sender);
std::string id = "";
if (sender->GetDeviceId().empty()) {
    id = sender->GetObjectId();
} else {
    id = sender->GetObjectId() + "-" + sender->GetDeviceId();
}
```

Слика 33 Провера идентификатора објекта

```
Device::CWiseDevice* dev = GetWiseDevice(id);
if (dev) {
    UpdateLastActivity(dev, _Event.GetTimestamp());
    string propKey = GeneratePropertyKey(evt->GetGroupId(),
                                         evt->GetServiceType(),
                                         evt->GetPropertyName());
    dev->OnPropertyChanged(propKey, evt->GetValue());
}
```

Слика 34 Позивање функције за промену особине над инстанцом објекта класе *WISE* Уређај

Такође, било је потребно претплатити се *MQTT* брокеру на слушање свих захтева, команди и догађаја на које уређај за конверзију протокола објављује (Слика 35).

```
Poco::format(bridges, "%s/%s/bdg/+", configSSDP->GetDomainID(),
             configSSDP->GetHomeID()); // ANY Bridge
Messenger::CUser::Ptr bdgs = m_pWiseMessenger->GetUser(bridges);
bdgs->GetFunctionality("+/+")
    ->RegisterForEvents(NULL); // groupId ANY + serviceType ANY
bdgs->RegisterForLwm(NULL);
```

Слика 35 Претплата *MQTT* брокеру на слушање саобраћаја који стиже са уређаја за конверзију протокола

4.2 Пример уређаја за конверзију протокола

При креирању примера уређаја за конверзију протокола коришћена је основа већ развијене *WISE* паметне утичнице. Како се, у великој мери, уређај за конверзију протокола понаша као *WISE* уређај у хијерархији *WISE* објеката (Слика 18), било је могуће надоградити се на ту основу. Овај уређај ће се у даљем тексту звати усмеривач, јер је његова сврха да усмерава све промене у особинама на неком уређају једног циљаног контролера, команде, захтеве и догађаје ка другом контролеру, на ком је усмеривач регистрован као уређај.

Прво је било потребно уклонити услуге које се вежу за паметну утичницу и оставити само услугу уређаја. Након тога било је потребно додати услугу контролера. Пошто паметна утичница није подржавала рад са другим уређајима, пре регистровања функција које би обрађивале команде и захтеве који су подржани услугом контролера, било је потребно проширити усмеривач тако да подржи додавање нових уређаја.

Прво је додата класа Опис Уређаја која садржи информације о сваком додатом уређају, парсира *JSON* објекат који описује уређај и служи за комуникацију са централним контролером и распоређује парсиране објекте на поља класе која чувају те информације. Да би уређај комуницирао са оба контролера, додат је Усмеривач *API* који омогућава класи Уређај да буде коришћена приликом посредовања усмеривача у комуникацији, везаној за одређени крајњи уређај, између два контролера. Он у себи садржи поље домаћин, који представља апстрактну класу која обрађује захтеве за уређај и услуге уређаја. Усмеривач *API* региструје, код домаћина, функције које се позивају када пристигну поруке од централног контролера, у чију мрежу се усмеривач уређај укључује, за комуникацију са крајњим уређајем.

Затим је у класи Комуникатор, која служи као *MQTT* клијент, додата функционалност слушања на захтеве који стижу на *MQTT* тему крајњег уређаја (Слика 36) и услуга (Слика 37) на циљаном централном контролеру. После тога је проширена функција за обраду пристигле поруке (Слика 38) тако да прибави идентификатор уређаја из теме и проследи поруку функцији за процесирање захтева. Проширује се и класа Крајњи Уређај тако да имплементира функције Усмеривач *API* интерфејса. Функција затим, захтеве намењене крајњем уређају проследи уређају који позива функције регистроване од стране Усмеривач *API*-ја.

```
void CMessenger::ActivateDevice(const std::string& _szDeviceIdentifier) {
    std::stringstream ss;
    ss << m_UserTag + DEVICE_ID_SEPARATOR << _szDeviceIdentifier
                                          + REQUEST_TOPIC_ELEMENT;
    std::string topic = ss.str();

    try {
        m_Client.Subscribe(topic, Message::QOS1);
    } catch (const BaseException& exc) {
        LOG(Error, "Error while activating service");
        exc.Rethrow();
    }
}
```

Слика 36 Пријављивање на захтеве који стижу на *MQTT* тему уређаја

```
void CMessenger::ActivateService(unsigned int _groupId,
                                 const std::string& _serviceType,
                                 const std::string& _szDeviceIdenti-
fier) {
    std::stringstream ss;
    ss << m_UserTag;
    if (!_szDeviceIdentifier.empty()) {
        ss << DEVICE_ID_SEPARATOR << _szDeviceIdentifier;
    }

    ss << TOPIC_SEPARATOR << _groupId << TOPIC_SEPARATOR + _serviceType
                                      + REQUEST_TOPIC_ELEMENT;

    std::string topic = ss.str();
    try {
        m_Client.Subscribe(topic, Message::QOS1);
    } catch (const BaseException& exc) {
        LOG(Error, "Error while activating service");
        exc.Rethrow();
    }
}
```

Слика 37 Пријављивање на захтеве који стижу на *MQTT* тему услуга уређаја

```
if (StartsWith(topic, m_UserTag)) {
    std::string deviceIdWithFunc = Replace(topic, m_UserTag, EMPTY_STRING);
    std::string deviceId;
    if (StartsWith(deviceIdWithFunc, DEVICE_ID_SEPARATOR)) {
        deviceIdWithFunc = Replace(deviceIdWithFunc, DEVICE_ID_SEPARATOR,
                                   EMPTY_STRING);
        deviceId = deviceIdWithFunc.substr(0, deviceIdWithFunc
                                           .find(TOPIC_SEPARATOR));
        deviceIdWithFunc = Replace(deviceIdWithFunc, deviceId, EMPTY_STRING);
    }
    unsigned int groupId = 0;
    std::string serviceType;
    if (!deviceIdWithFunc.empty()) {
        deviceIdWithFunc.replace(0, 1, EMPTY_STRING);
        groupId = StoI(deviceIdWithFunc.substr(0, deviceIdWithFunc
                                               .find(TOPIC_SEPARATOR)));
        serviceType = deviceIdWithFunc.substr(deviceIdWithFunc
                                              .find(TOPIC_SEPARATOR) + 1);
    }
    ProcessRequest(payload, groupId, serviceType, deviceId);
}
```

Слика 38 Проширење функције за обраду пристигле поруке

Након тога додата је класа Усмеривач, која у себи садржи мапу уређеног пара индикатор објекта и крајњи уређај, у којој је индикатор објекта кључ. Сваком крајњем

уређају који се дода на усмеривач, Усмеривач класа постаје домаћин и на њу се региструју функције које ће се позивати приликом обраде порука. Када пристигне порука за обраду, на регистровану функцију, са идентификатором уређаја ком је намењена, из мапе се по кључу вади објекат крајњег уређаја и њиме управља по инструкцијама из поруке.

Свака промена особине уређаја, захтев или команда, која пристигне од контролера који управља усмеривачем, прво буду обрађене од стране уређаја, да би се сачувале било какве промене у опису уређаја, а затим прослеђују циљаном централном контролеру објављивањем пристигле теме на његов *MQTT* брокер.

4.3 *HTTP* сервер

Да би крајњи уређај могао преузети нову верзију управљачког софтвера и резервну копију података са контролера или отпремити резервну копију података на контролер, потребно је имплементирати *HTTP* сервер на централном контролеру. За имплементацију се користи језик Це плус плус и *POCO* колекција библиотека. Сервер се имплементира као део *OHM* апликације.

```
for (std::vector<std::string>::iterator it = ifaces.begin();
    it != ifaces.end(); ++it) {
    try {
        Poco::Net::NetworkInterface nis;
        nis = Poco::Net::NetworkInterface::forName(*it,
              Poco::Net::IPAddress::IPv4);
        if (nis.isUp()) {
            Poco::ScopedLock<Poco::Mutex> ifacelock(m_IfaceMutex);
            m_RunningInterfaces.push_back(*it);
            Poco::Net::SocketAddress addr(nis.address().toString(), serverPort);
            m_pHTTPServerSockets.push_back(new Poco::Net::ServerSocket(addr));
            debug("Creating HTTP Server on network interface <" + *it + ">");
        } else {
            throw Poco::Exception("Interface not UP.");
        }
    } catch (Poco::Exception& e) {
        warning("Unable to create HTTP Server on network interface <" + *it +
                "> reason: " + e.displayText());
    }
}
```

Слика 39 Део кода за креирање мрежних утичница на активним мрежним интерфејсима

HTTP сервер се користио и до сада на контролеру за опслуживање уређаја *XML* датотекама које садрже информације о контролеру и паметне звучнике мултимедијалном садржају, али за сваку нову функционалност контролера било је потребно додати нови сервер. Са циљем да се не отвара велики број мрежних утичница за исту функционалност и

да се омогући лакши развој и одржавање нових функционалности *HTTP* сервера, одлучено је да се промени тај приступ. Идеја је имплементација једног *HTTP* сервер контролера који би усмеравао сав саобраћај који пристигне на сервер и контролисао сервер у случају потребе за отпремањем података. Број *HTTP* сервера се свео на број активних мрежних интерфејса. За сваки мрежни интерфејс креира се једна мрежна утичница (Слика 39), а за сваку утичницу један сервер (Слика 40).

```
for (std::vector<Poco::Net::ServerSocket*>::iterator
    it = m_pHTTPServerSockets.begin(); it != m_pHTTPServerSockets.end(); ++it) {
    m_pHTTPServers.push_back(new Poco::Net::HTTPServer(
                            new HTTPHandlerFactory(*this), m_Threads,
                            **it, params));
    m_pHTTPServers.back()->start();
}
```

Слика 40 Део кода за креирање *HTTP* сервера на свакој од мрежних утичница

Да би *HTTP* контролер знао коме је порука намењена, уведен је појам слушаоца (“*responder*”). Свака логичка магистрала или било који други део *OHM* апликације се могу пријавити да буду слушаоци на *HTTP* захтеве који у префиксу путање (“*url*”) имају њихову ознаку (нпр. “*wise-ssdp*”, “*wise-ota*” или “*wise-backup*”). Када на сервер стигне *HTTP* захтев, ако је пристигла метода за отпремање (“*PUT*”), прелази се на обраду поруке за пријем података, а уколико је метода за преузимање (“*GET*”), контролер поруку даље прослеђује слушаоцу на обраду (Слика 41).

```
if (method == Poco::Net::HTTPServerRequest::HTTP_PUT) {
    if (!AcceptUploadRequest(_Request, _Response, param)) {
        _Response.setStatus(Poco::Net::HTTPServerResponse::HTTP_BAD_REQUEST);
        warning("HTTP PUT not supported for responder: " + requestResponder);
    }
    return;
} else if (method == Poco::Net::HTTPServerRequest::HTTP_GET) {
    param[PARAM_SERVER_ADDR] = _Request.serverAddress().host().toString();
    CGenericResult res;
    m_pControllerEnv->QueryObject(m_Responders[requestResponder],
                                  HTTP_SERVER_RESPONDER_SERVICE,
                                  QRY_PROCESS_REQUEST, &param, &res);
    ProcessResponderObjRsp(_Response, res);
}
```

Слика 41 Обрада поруке у зависности од методе *HTTP* захтева

Ако је пристигли захтев означен методом за преузимање, слушалац, након обраде поруке, контролеру враћа одговор на ту поруку, који у себи може садржати инструкције за управљање сервером. Те инструкције могу обавестити сервер да је потребно послати одређеном уређају неке податке, да је дошло до грешке или да игнорише захтев (Слика 42).

```
if (rspType == SEND_BUFFER) {
    debug("Sending buffer.");
    _Response.setContentType(response[PARAM_CONTENT_TYPE].convert<std::string>())
;
    std::string buf = response[PARAM_BUFFER].convert<std::string>();
    _Response.sendBuffer(buf.c_str(), buf.length());
} else if (rspType == SEND_FILE) {
    debug("Sending file");
    _Response.sendFile(response[PARAM_FILE].convert<std::string>(),
                       response[PARAM_CONTENT_TYPE].convert<std::string>());
} else if (rspType == EXIT) {
    warning("Exit response.");
    _Response.setStatus(Poco::Net::HTTPResponse::HTTP_NOT_FOUND);
} else {
    error("Something went wrong.");
}
```

Слика 42 Управљање сервером након што слушалац обради захтев

```
bool CController::AcceptUploadRequest(Poco::Net::HTTPServerRequest& _Request,
    Poco::Net::HTTPServerResponse& _Response, const Parameters &_Parameters) {
    CGenericResult res;
    m_pControllerEnv->QueryObject(m_Responders[_Parameters[PARAM_RESPONDER]
                                  .convert<std::string>()],
                                  HTTP_SERVER_RESPONDER_SERVICE,
                                  QRY_ACCEPT_DOWNLOAD_REQUEST,
                                  &_Parameters, &res);
    std::string paramUri(_Parameters[PARAM_URI].convert<std::string>());
    if (static_cast<EResponseType>(res.getValue()[PARAM_RESPONSE_TYPE]
        .convert<int>()) == EXIT) {
        debug("PUT request has not been accepted for URI <" + paramUri + ">");
        return false;
    }
    debug("Downloading file...");
    std::istream& stream = _Request.stream();
    long len = _Request.getContentLength();
    char* buffer = new char[len];
    stream.read(buffer, len);
    std::filebuf newFile;
    newFile.open(paramUri.c_str(), std::ios::out);
    trace("File opened <" + paramUri + ">");
    newFile.sputn(buffer, len);
    delete [] buffer;
    _Response.send();
    return true;
}
```

Слика 43 Обрада захтева за отпремање датотеке

Уколико је пристигли захтев ипак означен методом за отпремање, прво је потребно проверити регистрованог слушаоца за тај захтев да ли је потребно дозволити уређају да

отпреми датотеку на контролер. Ако је одговор потврдан, прелази се на преузимање датотеке, уколико је одричан, захтев се одбацује (Слика 43).

4.4 Подршка за чување резервне копије података

Да би се подржало чување резервне копије података на централном контролеру, потребно је било да, када *OSM* апликација прими захтев за чување резервне копије података или захтев за реконструкцију, проследи тај захтев *OHM* апликацији. У *OHM* апликацији је додато да прође кроз сваку логичку магистралу централног контролера и провери да ли подржава чување резервне копије података. Уколико подржава, шаље се команда да се изврши жељени захтев (Слика 44).

```
vector<OBJID> buses;
m_pControllerEnv->GetObjectListByType(OBJ_TYPE_BUS, buses);
FOR_EACH(vector<OBJID>, bus, buses) {
    if(m_pControllerEnv->HasService(*bus,CBusService::TYPE)
       && (m_pControllerEnv->IsBackupSupported(*bus))) {

        debug("Doing restore on bus[" +
              Poco::NumberFormatter::format(*bus) + "]");
        m_pControllerEnv->SendCommand(*bus, CBusService::TYPE,
                                      CBusService::CMD_RESTORE,
                                      _pCommandParams);
    }
}
```

Слика 44 Пример обраде команде за реконструкцију података

Као део овог рада, циљ је подржати чување резервне копије само за *WISE* магистралу па ће се команда послати само на *WISE* магистралу. Када команда стигне до *WISE* магистрале, магистрала прелази из стања мировања у стање иницирање. У том стању контролер поставља временско ограничење од сто двадесет секунди, у ком се процес мора завршити и шаље захтев свим уређајима да би проверио за колико уређаја је потребно покренути чување резервне копије. Ако ни једном уређају није потребно чување резервне копије података, магистрала се враћа у стање мировања. Ако има уређаја којима треба чување резервне копије података, отпочиње процес чувања резервне копије података преласком магистрале у стање израђивање резервне копије, а на *HTTP* контролер се пријављује слушалац "*wise-backup*". Када уређај пошаље догађај да је отпочет процес чувања резервне копије података, он се ставља у листу уређаја који су у процесу израде. Тај догађај мора стићи до контролера најкасније пет секунди након што је команда за чување резервне копије послата. Током тог процеса, уређај датотеку шаље помоћу *HTTP* захтева на адресу централног контролера.

Када уређај пријави да је успешно завршио израду резервне копије, он се избацује из листе уређаја који су у процесу израде и у бази података се чува идентификатор процеса чувања резервне копије за тај уређај под кључем у виду броја, који представља идентификатор тог уређаја. Ако уређај пријави да израда није успешно извршена или да је нагло прекинута, уређај се само избацује из листе уређаја који су у процесу израде. Када се у листи уређаја који су у процесу израде не налази више ни један уређај или истекне временско ограничење у ком се процес мора завршити, процес се завршава, а *OHM* апликација јавља *OSM* апликацији да је процес завршен, одјављује слушаоца "*wise-backup*" и прелази у стање мировања.

```
void HTTPDownloader::DownloadFile() {
    std::cout << "\nFile name:\t" << m_FileName << "\nUrl:\t"
              << GetUrl() << "\n";
    FILE* file = fopen(m_FileName.c_str(), "wb");
    if (file == NULL) {
#ifdef WISE_DEBUG
        fprintf(stderr, "File could not be opened.");
#endif
        return;
    }
    SetOption(CURLOPT_TIMEOUT, REQ_TIMEOUT);
    SetOption(CURLOPT_URL, GetUrl().c_str());
    SetOption(CURLOPT_WRITEFUNCTION, DownloadDataCallback);
    SetOption(CURLOPT_WRITEDATA, file);
    SetOption(CURLOPT_NOPROGRESS, 1L);
#ifdef WISE_DEBUG
    SetOption(CURLOPT_DEBUGFUNCTION, Trace);
    SetOption(CURLOPT_VERBOSE, 1L);
#endif
    SendRequest();
    fclose(file);
}

size_t HTTPDownloader::DownloadDataCallback(void *_ptr, size_t _size,
                                            size_t _nmemb, FILE *_pInstance) {
    return fwrite(_ptr, _size, _nmemb, _pInstance);
}
```

Слика 45 Преузимање датотеке на примеру *WISE* паметне утичнице

Када до *WISE* магистрале стигне команда за реконструкцију података, магистрала прелази из стања мировања у стање реконструкције. Стање реконструкције има временско ограничење, у трајању од сто двадесет секунди, у оквиру ког процес мора бити завршен. Након тога, свим уређајима се шаље захтев за реконструкцију, а *HTTP* серверу се пријављује слушалац "*wise-backup*". Током процеса, уређаји шаљу захтеве *HTTP* серверу за преузимање датотеке са сачуваним подацима. После тога, уређај пријави да је отпочео

реконструкцију, смешта се у листу уређаја који су у процесу реконструкције. Да би уређај био смештен у ту листу потребно је и да догађај стигне највише пет секунди након што је команда за реконструкцију послата. Након што уређај пријави да је реконструкција била успешна, неуспешна или нагло прекинута, он се брише из листе уређаја који су у процесу реконструкције. Када је та листа празна, процес се прекида, а *OHM* апликација јавља *OSM* апликацији да је процес реконструкције завршен, *WISE* магистрала одјављује слушаоца "*wise-backup*" и прелази у стање мировања.

```
void HTTPUploader::UploadFile() {
    Utility::ScopedLock lock(m_Mutex);
    m_BytesRead = 0;
    struct stat fileInfo;
    stat(m_FileName.c_str(), &fileInfo);
    m_File = fopen(m_FileName.c_str(), "rb");
    if (m_File == NULL) {
#ifdef WISE_DEBUG
        fprintf(stderr, "File could not be opened.");
#endif
        return;
    }
    char url[256];
    std::string urlToUpload(GetUrl() + m_RemotePath + m_FileName);
    curl_msnprintf(url, 128, urlToUpload.c_str());
    SetOption(CURLOPT_READFUNCTION, ReadDataCallback);
    SetOption(CURLOPT_UPLOAD, 1L);
    SetOption(CURLOPT_URL, url);
    SetOption(CURLOPT_READDATA, this);
    SetOption(CURLOPT_INFILESIZE_LARGE, static_cast<curl_off_t>
                                        (fileInfo.st_size));
#ifdef WISE_DEBUG
    SetOption(CURLOPT_DEBUGFUNCTION, Trace);
    SetOption(CURLOPT_VERBOSE, 1L);
#endif
    SendRequest();
    fclose(m_File);
}


size_t HTTPUploader::ReadDataCallback(void *_ptr, size_t _size, size_t _nmemb,
                                      void *_pInstance) {
    HTTPUploader* obj = static_cast<HTTPUploader*>(_pInstance);
    Utility::ScopedLock lock(obj->m_Mutex);
    size_t retcode = fread(_ptr, _size, _nmemb, obj->m_File);
    obj->m_BytesRead += retcode;
    return retcode;
}
```

Слика 46 Отпремање датотеке на примеру *WISE* паметне утичнице

Када се процес чувања резервне копије или реконструкције података заврше, уређај поставља особину време последњег чувања података на вредност једнаку временској ознаци тренутка у ком је процес завршен.

Као пример уређај коришћен је већ постојећи *WISE* уређај, паметна утичница, којој је додата услуга за чување резервне копије података у нулту услужну групу. Додате су функције које одговарају када уређај прими услугом дефинисане команде од контролера. Имплементиран је *HTTP* клијент помоћу *cURL* библиотеке који би преузимао (Слика 45) и слао (Слика 46) датотеку у процесима чувања и реконструкције података.

4.5 Подршка за надоградњу управљачког софтвера

Како је, за разлику од подршке за чување резервне копије података, подршка за надоградњу управљачког софтвера већ постојала на централном контролеру за одређен број магистрала, додавање подршке на *WISE* магистрални није захтевало пуно измена на самом контролеру.

Када уређај има додату услугу за надоградњу управљачког софтвера и на сервису у облаку се у том тренутку налази нова верзија управљачког софтвера спремна за преузимање, кориснику се на клијентској апликацији појави иконица која, када се притисне, иницира процес надоградње.

Када корисник иницира процес надоградње на *WISE* уређају, *WISE* магистрала региструје слушаоца "*wise-ota*" на *HTTP* серверу и уређају шаље команду за отпочињање надоградње. У току процеса, уређај може преузети датотеку слањем *HTTP* захтева за преузимање података ка централном контролеру. Клијентске апликације приказују напредак процеса надоградње, у виду траке напредовања, која се помера када год се прими нови догађај који означава индикатор тока надоградње. Процес се завршава када уређај пошаље догађај статус надоградње, који означава да је процес завршен, или догађај да надоградња није успела.

Да би се демонстрирала ова функционалност, коришћен је постојећи пример *WISE* утичнице који је проширен услугом за надоградњу управљачке програмске подршке у нултој сервисној групи. Додате су функције које реагују на захтеве и команде послате са централног контролера где, када уређај пошаље захтев који нуди надоградњу на нову верзију управљачког софтвера, уређај одговори прихватањем, а када стигне команда за покретање надоградње, уређај ће искористити *HTTP* клијента описаног у претходном поглављу (поглавље 4.4) којим ће преузети датотеку. Након успешног преузимања, уређај шаље статус контролеру да је процес завршен.

5. Резултати

У овом поглављу биће описан систем на ком су тестиране имплементиране функционалности, начин тестирања имплементираних функционалности и резултати тог тестирања.

Резултати тестирања су преузети из датотеке за евиденцију догађаја контролера и уређаја. Функционално тестирање је вршено на основу увида у испис порука у датотеци за евиденцију, док су резултати тестирања перформанси извучени на основу увида у временске ознаке од почетка па до краја извршавања неке од функционалности.

5.1 Систем паметне куће над којим су вршени тестови

У систему паметне куће над којим је рађено тестирање састоји се од два централна контролера:

- један који комуницира *WISE* протоколом, који ће се на даље називати *WISE* контролер, и
- други који комуницира *ZigBee* протоколом, даље ће се називати *ZigBee* контролер.

Осим тога, за тестирање имплементираних функционалности коришћен је један *WISE* усмеривач (Поглавље 4.2) и једна *WISE* паметна утичница која има подршку за чување резервне копије података и надоградњу управљачког софтвера. Осим тога, на *ZigBee* контролер је додата једна паметна *ZigBee* утичница помоћу које је проверавана функционалност уређаја за конверзију протокола.

5.2 Тестирање *WISE* усмеривача

WISE усмеривач је тестиран тако што је прво, на *ZigBee* контролер, додата *ZigBee* паметна утичница (Слика 47). У конфигурационој датотеци усмеривача је подешен *ZigBee* контролер као циљани централни контролер са ког треба преузети информације о

уређајима, те уређаје укључити у мрежу *WISE* контролера и усмеравати саобраћај између њих. Након покретања, *WISE* контролер се успешно претплатио *MQTT* брокеру на слушање саобраћаја који се тиче уређаја за конверзију протокола (Слика 48). Након тога, на *WISE* контролер, додат је усмеривач уређај који има подршку за услугу контролера (Слика 49). Затим је усмеривач успешно додао *ZigBee* паметну утичницу на *WISE* контролер. Поруке са централних контролера и усмеривача за додавање уређаја (Слика 50, и Слика 51) приказане су на сликама испод.

Слика 47 *ZigBee* паметна утичница на *ZigBee* контролеру

```
19.01.21 02:16:38.140 T [WISE] 0: Registering on events from: wise/000000000018/bdg/+/+/+.
19.01.21 02:16:38.140 D [WISE] 0: Subscription stored: wise/000000000018/bdg/+/+/+/evt
19.01.21 02:16:38.140 D [WISE] 0: Registered on events from: wise/000000000018/bdg/+/+/+.
19.01.21 02:16:38.140 I [WISE] 0: Added events listener for: wise/000000000018/bdg/+/+/+.
19.01.21 02:16:38.140 T [WISE] 0: Registering on LWM from: wise/000000000018/bdg/+.
19.01.21 02:16:38.140 D [WISE] 0: Subscription stored: wise/000000000018/bdg/+/sts
19.01.21 02:16:38.140 D [WISE] 0: Registered on LWM from: wise/000000000018/bdg/+.
19.01.21 02:16:38.140 I [WISE] 0: Added LWM listener for: wise/000000000018/bdg/+.
19.01.21 02:16:39.160 D [WISE] 0: Subscription stored: wise/000000000018/bdg/+/+/+/evt
19.01.21 02:16:39.160 D [WISE] 0: Subscription stored: wise/000000000018/bdg/+/sts
```

Слика 48 Претплата *WISE* контролера на теме уређаја за конверзију протокола

Слика 49 *WISE* услуга контролера

```
19.01.21 14:09:51.098 I [WISE] 7: Network is opened
19.01.21 14:09:51.248 I [ZigBee] 61: Association table has 0 router devices.
19.01.21 14:09:51.312 I [ZigBee] 61: Association table has 0 end devices.
19.01.21 14:10:15.039 I [ZigBee] 61: Device with mac[00124b000722e921] nwkAddress[e850] type[2] paired
19.01.21 14:10:15.040 I [SystemController] 53: Network state: closing
19.01.21 14:10:15.040 I [SystemController] 53: CloseNetwork: InternalBus
19.01.21 14:10:15.040 I [Internal] 6: Closing the network
19.01.21 14:10:15.071 I [ZigBee] 61: Association table has 1 router devices.
19.01.21 14:10:15.135 I [ZigBee] 61: Association table has 0 end devices.
19.01.21 14:10:17.985 I [DeviceManager] 63: Creating device with class[ObloSmartPlugv2] id[0] name[]
                        address[00124b000722e921] bus[1018]
19.01.21 14:10:17.986 I [DeviceManager] 63: id changed to 1019
19.01.21 14:10:17.986 I [ZigBee] 63: Add device[1019]
```

Слика 50 Додавање *ZigBee* паметне утичнице на *ZigBee* контролер

```
[2021-01-19 14:10:48.431] D [4154] [12] [ObloBridgeDevice] Messenger: Request received: {"params":{
                          "input":{},"name":"getDeviceList"},"name":"executeQuery"}
[2021-01-19 14:10:48.432] D [4154] [12] [ObloBridgeDevice] Messenger: Sending response: {"params":{
                          "output":{"deviceList":[{"manufacturer":"OBLO Living",
                          "model":"SmartPlugv2-WISE","version":"5.2.2-4918","type":"ObloSmartPlugv2",
                                                              ...
                          "type":"switch","proprietary":false,"propertyList":[{"name":"state",
                          "value":false,"readOnly":false,"type":2,"options":[]}]}],"id":0}]}]}},
                          "code":0}
[2021-01-19 14:10:48.440] D [4154] [6] [ObloCommunicator] Event processed (s=gtw/000000000018/ohm,
                          n='EVENT_ADDING_MULTIPLE_DEVICE_IN_PROGRESS')
[2021-01-19 14:10:48.440] D [4154] [6] [ObloCommunicator] Event processed (s=gtw/000000000018/ohm,
                          n='EVENT_MULTIPLE_DEVICE_ADDED')
[2021-01-19 14:10:48.448] D [4154] [12] [ObloBridgeDevice] Messenger: Request received: {"params":{
                          "since":1611061848447},"name":"getObjectState"}
[2021-01-19 14:10:48.448] T [4154] [12] [ObloBridgeDevice] CommunicationManager: Responded on
                          request <r=getObjectState>
[2021-01-19 14:10:48.448] D [4154] [12] [ObloBridgeDevice] Messenger: Sending response: {"params":{
                          "state":{"manufacturer":"OBLO Living","model":"SmartPlugv2-WISE",
                          "version":"5.2.2-4918","type":"ObloSmartPlugv2","id":"1019","role":"ipd"}},
                          "code":0}
[2021-01-19 14:10:48.449] D [4154] [12] [ObloBridgeDevice] Messenger: Request received: {"params":{
                          "since":0},"name":"getObjectState","sender":"wise\/000000000018\/hub\/0"}
[2021-01-19 14:10:48.449] D [4154] [12] [ObloBridgeDevice] Messenger: Sending response: {"params":{
                          "state":{"manufacturer":"OBLO Living","model":"SmartPlugv2-WISE",
                          "version":"5.2.2-4918","type":"ObloSmartPlugv2","id":"1019","role":"ipd",
                                                              ...
                          "value":0},{"name":"voltage","value":0}]},{"type":"switch",
                          "proprietary":false,"propertyList":[{"name":"state","value":false}]}],
                          "id":0}]}},"code":0}
```

Слика 51 Додавање *ZigBee* утичнице на *WISE* контролер са стране усмеривача

```
19.01.21 14:47:00.159 I [WISE] 25: Query for bridge devices
19.01.21 14:47:00.165 I [WISE] 25: Adding device (i=012345678910.bdg-1019)
19.01.21 14:47:00.165 I [DeviceManager] 25: Creating device with class[ObloSmartPlugv2] id[0] name[]
                      address[012345678910.bdg-1019] bus[1011]
19.01.21 14:47:00.165 I [DeviceManager] 25: id changed to 1026
19.01.21 14:47:00.165 I [WISE] 25: Add device[1026]
19.01.21 14:47:00.165 I [WISE] 25: Device added (d=1026)
19.01.21 14:47:00.286 I [WISE] 25: (1026) Firmware version has changed (nv='5.2.2-4918', ov='')
19.01.21 14:47:00.287 I [WISE] 25: Adding generic device, asking for device info...
19.01.21 14:47:00.287 I [WISE] 25: Device info received, configuring device...
19.01.21 14:47:00.290 I [WISE] 25: Device info assigned.
19.01.21 14:47:00.320 D [WISE] 26: Subscription stored: wise/000000000018/bdg/012345678910.bdg-1019/evt
19.01.21 14:47:00.320 D [WISE] 26: Registered on events from: wise/000000000018/bdg/012345678910.bdg-1019.
19.01.21 14:47:00.320 I [WISE] 26: Added events listener for: wise/000000000018/bdg/012345678910.bdg-1019.
19.01.21 14:47:00.359 I [WISE] 26: Device state set (d=1026, s=RUN)
19.01.21 14:47:00.359 I [WISE] 26: Session has been established (d=1026, h=192.168.1.1)
```

Слика 52 Додавање *ZigBee* паметне утичнице на *WISE* контролер

Након тога, тестирано је управљање уређајем са стране једног и другог контролера. Управљање је извршено успешно, а део размене порука између оба контролера и усмеривача приказан је на сликама испод (Слика 53 Слика 55, и Слика 54), када се стање утичнице промени са *WISE* контролера.

```
19.01.21 14:51:42.805 T [Messaging] 47: event name[device_property_changed] payload[{ "device_id" : 1019,
                      "event_source" : "client", "event_source_id" : 0, "property_name" : "State",
                      "property_value" : true, "service_name" : "OutletService",
                      "service_type" : "OutletService" }]
```

Слика 53 Промена стања утичнице на *ZigBee* контролеру

```
[2021-01-19 14:51:42.803] D [5925] [33] [ObloBridgeDevice] Messenger: Request received: {"params":{
                          "name":"state","value":true},"name":"setPropertyValue"}
[2021-01-19 14:51:42.803] D [5925] [177] [ObloBridgeDevice] OutletService
[2021-01-19 14:51:42.803] D [5925] [33] [ObloBridgeDevice] Messenger: Sending event: {"params":{
                          "name":"state","value":true},"name":"propertyChanged"}
[2021-01-19 14:51:42.803] T [5925] [33] [ObloBridgeDevice] CommunicationManager: Responded on
                          request <r=setPropertyValue>
[2021-01-19 14:51:42.803] T [5925] [177] [ObloCommunicator] Sending request: {
                          "name" : "set_service_property_value", "params" : { "id" : 1019,
                          "property_name" : "State", "property_value" : true,
                          "service_name" : "OutletService" }, "sender" : "gtw\/000000000018\/bdg_conn",
                          "type" : "command", "uid" : 110364491 } to: gtw/000000000018/ohm/req.
```

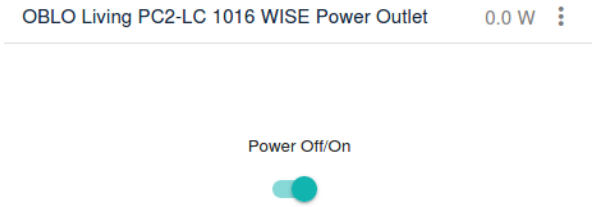
Слика 54 Промена стања утичнице са становишта усмеривача

```
19.01.21 14:51:42.802 T [WISE] 25: Sending request: { "name" : "setPropertyValue", "params" : {
                      "name" : "state", "value" : true }, "sender" : "wise\/000000000018\/hub\/0"}
                      to: wise/000000000018/bdg/012345678910.bdg-1019/0/switch/req.
19.01.21 14:51:42.802 D [WISE] 25: Request sent (d=wise/000000000018/bdg/012345678910.bdg-1019)
```

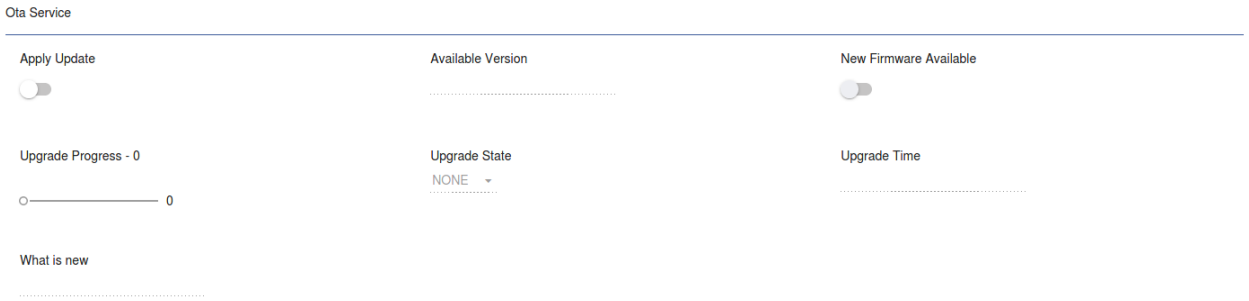
Слика 55 Промена стања утичнице са *WISE* контролера

5.3 Тестирање надоградње управљачког софтвера

У мрежу централног контролера је укључена *WISE* паметна утичница (Слика 56) са подршком за надоградњу управљачког софтвера (Слика 57).



Слика 56 *WISE* паметна утичница



Слика 57 Услуга за надоградњу управљачког софтвера

На сервис у облаку је додата нова верзија управљачког софтвера и обавештен је централни контролер да је та верзија спремна за преузимање. Након тога, са клијентске апликације је покренута надоградња управљачког софтвера за *WISE* паметну утичницу. Размена порука између централног контролера (Слика 58) и крајњег уређаја (Слика 59) приказана је на сликама испод.

```
19.01.21 02:40:21.794 T [Messaging] 62: Message arrived: {"name":"set_service_property_value","params":{
                        "id":1016,"service_name":"OTAService","property_name":"ApplyUpdate",
                        "property_value":true}}
19.01.21 02:40:21.795 I [NodeOtaController] 42: Obtaining new firmware file
19.01.21 02:40:21.795 T [Messaging] 40: Sending event: { "name" : "device_property_changed", "params" :{
                        "device_id" : 1016, "property_name" : "ApplyUpdate", "property_value" : true,
                        "service_name" : "OTAService", "service_type" : "OTAService" }}
19.01.21 02:40:21.795 T [Messaging] 9: Sending request: { "name" : "getNodeOtaFwUriReq", "params" : {
                        "manufacturerName" : "OBLO Living", "modelName" : "PC2-LC" }, "sender" :
                        "gtw\/000000000018\/ohm", "type" : "req",} to: svc/zota/gtw/000000000018/req.
19.01.21 02:40:21.796 T [Messaging] 40: Sending event: { "name" : "device_property_changed", "params" :{
                        "device_id" : 1016, "property_name" : "UpgradeState", "property_value" : 1,
                        "service_name" : "OTAService", "service_type" : "OTAService" }}
19.01.21 02:40:21.967 T [Messaging] 62: Message arrived: {"uid":1531980111,"code":0,"message":"Ok",
                        "params":{"fileName":"nMIhRU73aq.testiranje","secure":true,
                        "md5":"d41d8cd98f00b204e9800998ecf8427e",
                        "uri":"https://jiohome.oblo.rs:443/strat_zota/storage/nMIhRU73aq.testiranje"}}
19.01.21 02:40:21.967 I [NodeOtaController] 42: Downloading firmware from
                        https://jiohome.oblo.rs:443/strat_zota/storage/nMIhRU73aq.testiranje
                        to /tmp/nMIhRU73aq.testiranje
19.01.21 02:40:22.678 T [WISE] 24: Sending request: { "name" : "executeQuery", "params" : { "input" :{
                        "version" : "" }, "name" : "newFirmwareAvailable" }}
19.01.21 02:40:22.680 T [WISE] 70: Message arrived: {"uid":699925926,"ts":1611020422679,"params":{
                        "output":{"accepted":true}},"code":0}
19.01.21 02:40:22.681 I [WISE] 24: Local OTA download type starting...
19.01.21 02:40:22.681 I [HttpSrv] 7: Register Responder arrived for wise-ota with id: WiseIpBus
19.01.21 02:40:22.681 T [WISE] 25: Sending request: { "name" : "executeCommand", "params" : { "input":{
                        "md5" : "d41d8cd98f00b204e9800998ecf8427e",
                        "url" : "localhost:8181\/wise-ota\/tmp\/nMIhRU73aq.testiranje",
                        "version" : "2.7.8" }, "name" : "startUpgrade" }}
19.01.21 02:40:22.726 T [WISE] 70: Message arrived: {"params":{"progress":0},"name":"upgradeProgress"}
19.01.21 02:40:22.726 T [WISE] 83: Message arrived: {"params":{"progress":25},"name":"upgradeProgress"}
19.01.21 02:40:52.731 T [WISE] 83: Message arrived: {"params":{"progress":50},"name":"upgradeProgress"}
19.01.21 02:40:52.732 I [NodeOtaController] 42: Device OBLO Living PC2-LC 1016 WISE Power Outlet
                        start upgrading.
19.01.21 02:41:52.733 T [WISE] 83: Message arrived: {"params":{"status":3},"name":"upgradeStatus"}
19.01.21 02:41:52.733 D [WISE] 70: (1016) Property event processed (k='0.ota.upgradeProgress',
                        v={ "progress" : 100 })
19.01.21 02:41:52.734 D [WISE] 24: Ota done.
19.01.21 02:41:52.734 I [HttpSrv] 8: Unregister Responder arrived for wise-ota
19.01.21 02:41:52.734 I [NodeOtaController] 42: Removing file path from the OTA table...
19.01.21 02:41:52.734 I [NodeOtaController] 42: File /tmp/nMIhRU73aq.testiranje removed
```

Слика 58 Комуникација контролера са уређајем током процеса надоградње управљачког софтвера

```
[2021-01-19 02:40:22.679] D [31534] [32] [GenericDevice] Messenger: Request received: {"params":{"input":{
                          "version":""},"name":"newFirmwareAvailable"},"name":"executeQuery",
                          "sender":"wise\/000000000018\/hub\/0"}
[2021-01-19 02:40:22.680] D [31534] [32] [GenericDevice] Messenger: Sending response: {"params":{
                          "output":{"accepted":true}},"code":0}
[2021-01-19 02:40:22.682] D [31534] [32] [GenericDevice] Messenger: Request received: {"params":{"input":{
                          "md5":"d41d8cd98f00b204e9800998ecf8427e",
                          "url":"localhost:8181\/wise-ota\/tmp\/nMIhRU73aq.testiranje",
                          "version":""},"name":"startUpgrade"},"name":"executeCommand"}
[2021-01-19 02:40:22.683] W [31534] [32] [GenericDevice] localhost:8181/wise-ota/tmp/nMIhRU73aq.testiranje
[2021-01-19 02:40:22.683] D [31534] [32] [GenericDevice] Messenger: Sending response: {"code":0}
[2021-01-19 02:40:22.683] D [31534] [409] [GenericDevice] Messenger: Sending event: {"params":{
                          "progress":0},"name":"upgradeProgress"}
[2021-01-19 02:40:22.696] W [31534] [409] [GenericDevice] DOC400 potential issue during ota MD5 check.
[2021-01-19 02:40:22.696] W [31534] [409] [GenericDevice] nMIhRU73aq.testiranje
[2021-01-19 02:40:22.696] I [31534] [409] [GenericDevice] wget \
                          localhost:8181/wise-ota/tmp/nMIhRU73aq.testiranje
[2021-01-19 02:40:22.696] D [31534] [409] [GenericDevice] Messenger: Sending event: {"params":{
                          "progress":25},"name":"upgradeProgress"}
--2021-01-19 02:40:22--  http://localhost:8181/wise-ota/tmp/nMIhRU73aq.testiranje
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:8181... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [application/octet-stream]
Saving to: ‘nMIhRU73aq.testiranje’
2021-01-19 02:40:22 (0,00 B/s) - ‘nMIhRU73aq.testiranje’ saved [0/0]
[2021-01-19 02:40:52.730] I [31534] [409] [GenericDevice] New firmware downloaded successfully.
[2021-01-19 02:40:52.731] D [31534] [409] [GenericDevice] Messenger: Sending event: {"params":{
                          "status":2},"name":"upgradeStatus"}
[2021-01-19 02:40:52.731] D [31534] [409] [GenericDevice] Messenger: Sending event: {"params":{
                          "progress":50},"name":"upgradeProgress"}
[2021-01-19 02:40:52.731] W [31534] [409] [GenericDevice] nMIhRU73aq.testiranje
[2021-01-19 02:41:22.731] D [31534] [409] [GenericDevice] Messenger: Sending event: {"params":{
                          "progress":75},"name":"upgradeProgress"}
[2021-01-19 02:41:52.731] W [31534] [409] [GenericDevice] Content valid
```

Слика 59 Размена порука *WISE* уређаја са централним контролером током процеса надоградње управљачког софтвера

Размена порука је извршена успешно, као и преузимање саме датотеке са програмском подршком.

5.4 Тестирање чувања резервне копије података

За тестирање чувања и реконструкције резервне копије података коришћена је, већ додата, *WISE* паметна утичница (Слика 56) са подршком за чување резервне копије података (Слика 60).

Слика 60 Услуга чувања резервне копије података

Са клијентске апликације је покренуто чување резервне копије података. Након тога, уређаји су ступили у процес чувања резервне копије података. Размена порука између централног контролера (Слика 61) и *WISE* уређаја (Слика 62) приказана је на сликама испод.

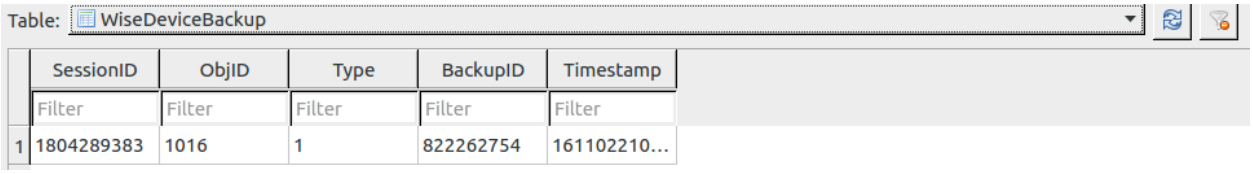
```
19.01.21 03:08:27.387 T [Messaging] 62: Message arrived: {
    "uid": 1570983441,
    "sender": "cli/test",
    "name": "backup",
    "params": {
        "service": "backupRestoreController",
        "params": {},
        "command": "backup"
    },
    "type": "command"
}
19.01.21 03:08:27.388 T [WISE] 8: Starting backup on WISE bus...
19.01.21 03:08:27.388 I [HttpSrv] 5: Register Responder arrived for wise-backup with id: WiseIpBus
19.01.21 03:08:27.388 T [WISE] 8: Sending request: { "name" : "executeQuery", "params" : { "input" : {  },
                      "name" : "startBackup" }} to: wise/000000000018/ipd/FEDCBA098765/0/backup/req.
19.01.21 03:08:27.389 T [WISE] 70: Message arrived: {"uid":163181510,"ts":1611022107388,"params":{
                      "output":{"type":1}},"code":0}
19.01.21 03:08:27.389 D [WISE] 8: Backup of WISE devices started successfully.
19.01.21 03:08:27.389 T [WISE] 24: Sending request: { "name" : "executeCommand", "params" : { "input" :{
                      "location" :
                      "localhost:8181\/wise-backup\/home\/rtrk\/projects\/oblo_ohm\/backup\/" },
                      "name" : "startRemoteBackup" }}
19.01.21 03:08:27.391 D [WISE] 70: (1016) Property event processed (k='0.backup.backupStarted', v={  })
19.01.21 03:08:27.436 D [WISE] 70: (1016) Property event processed (k='0.backup.backupFinished',
                      v={ "bid" : 822262754 })
19.01.21 03:08:27.436 T [WISE] 70: Sending process finished...
19.01.21 03:08:27.436 I [HttpSrv] 6: Unregister Responder arrived for wise-backup
19.01.21 03:08:27.436 I [WiseBackup] 70: Process finished.
19.01.21 03:08:27.436 T [Messaging] 40: Sending event: { "name" : "backupDone", "params" : {}}
19.01.21 03:08:27.436 D [Messaging] 40: Event (backupDone) sent.
```

Слика 61 Размена порука централног контролера са *WISE* уређајем током процеса чувања резервне копије података

```
[2021-01-19 03:08:27.388] D [31534] [32] [GenericDevice] Messenger: Request received: {"params":{
                          "input":{},"name":"startBackup"},"name":"executeQuery"}
[2021-01-19 03:08:27.388] T [31534] [32] [GenericDevice] Start backup received.
[2021-01-19 03:08:27.388] D [31534] [32] [GenericDevice] Messenger: Sending response: {"params":{
                          "output":{"type":1}},"code":0}
[2021-01-19 03:08:27.390] D [31534] [32] [GenericDevice] Messenger: Request received: {"params":{
                          "input":{"location":
                          "localhost:8181\/wise-backup\/home\/rtrk\/projects\/oblo_ohm\/backup\/"},
                          "name":"startRemoteBackup"},"name":"executeCommand"}
[2021-01-19 03:08:27.390] T [31534] [32] [GenericDevice] Start REMOTE backup received.
[2021-01-19 03:08:27.390] D [31534] [32] [GenericDevice] Messenger: Sending event: {
                          "name":"backupStarted"}
tar: Removing leading `../' from member names
../cfg/Config.txt
[2021-01-19 03:08:27.424] D [31534] [32] [GenericDevice] Sending backup file...
[2021-01-19 03:08:27.424] D [31534] [32] [GenericDevice] Starting HTTP client...
== Info:   Trying 127.0.0.1...
== Info: Connected to localhost (127.0.0.1) port 8181 (#0)
=> Send header, 0000000160 bytes (0x000000a0)
0000: 50 55 54 20 2f 77 69 73 65 2d 62 61 63 6b 75 70 PUT /wise-backup
                                ...
0090: 31 30 30 2d 63 6f 6e 74 69 6e 75 65 0d 0a 0d 0a 100-continue....
<= Recv header, 0000000023 bytes (0x00000017)
0000: 48 54 54 50 2f 31 2e 31 20 31 30 30 20 43 6f 6e HTTP/1.1 100 Con
0010: 74 69 6e 75 65 0d 0a                            tinue..
=> Send data, 0000001010 bytes (0x000003f2)
0000: 1f 8b 08 00 1b 3f 06 60 00 03 ed 56 4b 6f db 38 .....?.`...VKo.8
                                ...
03e0: 07 07 07 07 07 07 87 3d f8 17 cf 3d 4b 8d 00 28 .......=...=K..(
03f0: 00 00
== Info: We are completely uploaded and fine
                                ...
<= Recv data, 0000000000 bytes (0x00000000)
== Info: Closing connection 0
[2021-01-19 03:08:27.434] I [31534] [32] [GenericDevice] Backup file sent.
[2021-01-19 03:08:27.435] D [31534] [32] [GenericDevice] Messenger: Sending event: {"params":{
                          "bid":822262754},"name":"backupFinished"}
```

Слика 62 Комуникација уређаја са централним контролером током процеса чувања резервне копије података

Процес чувања резервне копије података успешно је завршен, као и отпремање датотеке на централни контролер. Након процеса, у базу података је смештена информација о том процесу (Слика 63).



Слика 63 Информација о успешно завршеном процесу смештена у базу података

Након процеса чувања, са клијентске апликације је покренут процес реконструкције резервне копије података. *WISE* паметна утичница је успешно преузела датотеку са централног контролера и процес је успешно завршен.

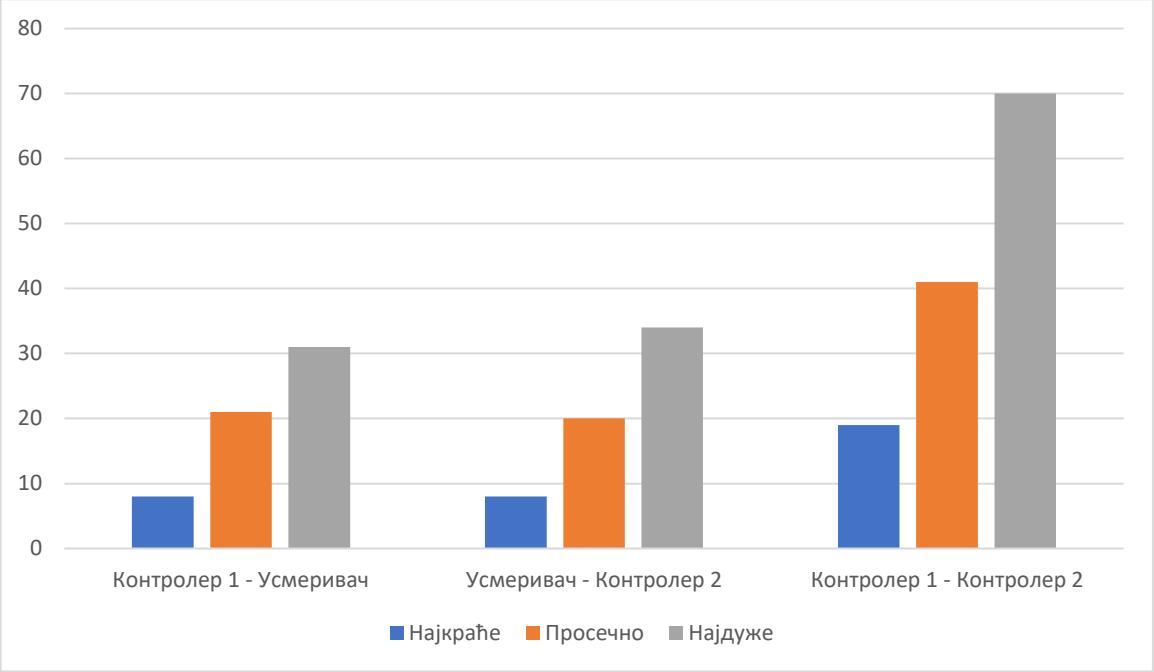
5.5 Тестирање перформанси

Подешен је систем описан у поглављу за тестирање *WISE* усмеривача (Поглавље 5.2) и њему је додата *WISE* паметна утичница. Написана је скрипта, помоћу *JMeter* алата, која на сваких сат времена иницира надоградњу управљачког софтвера, чување резервне копије и реконструкцију резервне копије. Такав систем остао је активан седам дана.

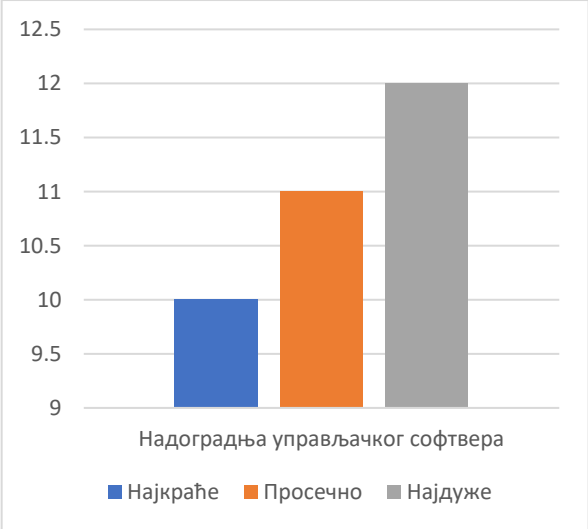
За *WISE* усмеривач, подаци су показали да је кашњење у поруци за промену стања укупно четири стотинке између два централна контролера. Просечно, време протекло од слања поруке са једног контролера до пријема са стране усмеривача је две стотинке, као што је и време протекло од слања поруке са усмеривача до пријема на други централни контролер исто две стотинке. Време протекло од слања поруке са једног контролера до пријема поруке на другом контролеру је варирало од најмање две, до највише седам стотинки. Ни након дугорочног тестирања, није се појавило губитак порука за овај тестни систем. Графички приказ добијених резултата налази се на слици испод (Слика 64).

Надоградња управљачког софтвера, као и чување и реконструкција резервне копије података, за тестни систем, нису пријављивали грешке. У овим, контролисаним, условима, надоградња управљачког софтвера, чување и реконструкција резервне копије података је сваки пут извршена успешно.

Надоградња управљачког софтвера трајала је, у просеку, једанаест секунди, од најмање десет секунди, до највише дванаест секунди. Графички приказ добијених резултата налази се на слици испод (Слика 65).

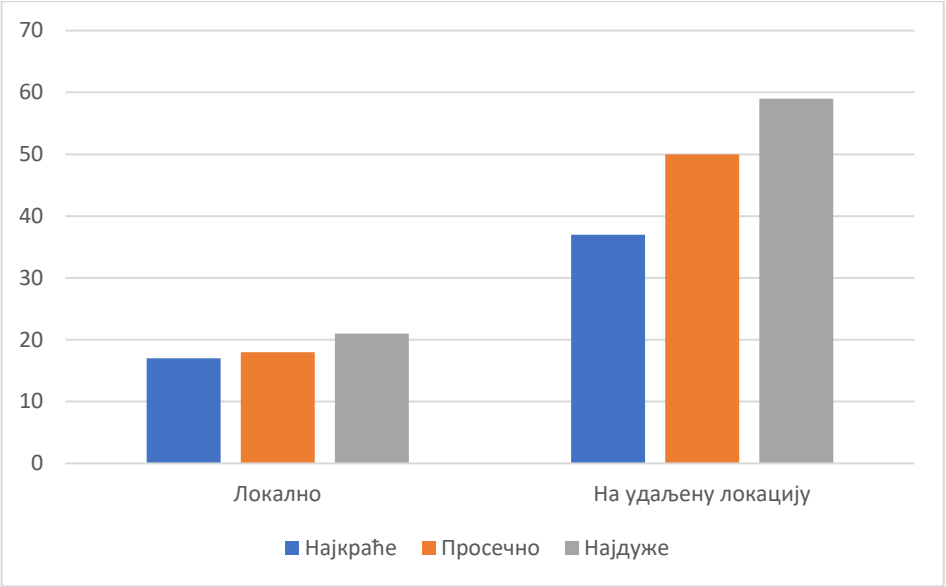


Слика 64 Време протекло од слања поруке до обраде у милисекундама



Слика 65 Време потребно за надоградњу управљачког софтвера у секундама

Чување резервне копије просечно је трајало педесет секунди за чување на удаљену локацију, од најмање тридесет и седам секунди, до највише педесет и девет секунди. За локално чување, потребно је свега осамнаест секунди у просеку, јер нема отпремања датотеке на удаљену локацију. Враћање резервне копије показало је исте просечне брзине као и чување. Графички приказ добијених резултата налази се на слици испод (Слика 66).



Слика 66 Време потребно за чување резервне копије података

6. Закључак

Систем паметне куће описан у овом раду има подршку за више различитих комуникационих протокола као што су *ZigBee*, *Z-Wave*, *UPnP* и *WISE*. Да би систем подржао све те комуникационе протоколе, потребно је да контролер креира логичке магистрале за сваки подржани протокол. Свака магистрала заузима одређену меморију у систему и користи велики, често варијабилан, број програмских нити. То не представља проблем за персоналне рачунаре, али за мале уграђене системе то постаје итекако значајно.

Осим заузећа ресурса, многи протоколи захтевају посебне физичке чипове за комуникацију, што приморава произвођаче контролера да троше више новца на сам хардвер и самим тим повећају цену за крајње кориснике.

Подршком за уређаје за конверзију протокола омогућава се произвођачима контролера да смање трошкове производње контролера, па самим тим и цене коју плаћају корисници, као и да омогући инжењерима да прилагоде софтвер за много већи број уређаја, па би можда једног дана и телевизор, као централни уређај у објекту са константним напајањем, могао имати функцију централног контролера.

У овом раду је представљен концепт уређаја за конверзију протокола као уређај који омогућава *WISE* контролеру да чита информације са *ZigBee* уређаја и управља истим. Како се у тестирању овај уређај показао успешним, следећи корак у развоју овакве врсте уређаја би могао бити имплементација могућности конверзије протокола на саме крајње уређаје са константним напајањем, као на пример *ZigBee* или *Z-Wave* паметна утичница, па би корисницима било могуће да, уместо да купују одвојене уређаје за конверзију протокола, могли само да купе једну верзију паметне утичнице која би уз мало већу цену имала могућност конверзије протокола и усмеравала сав *ZigBee* или *Z-Wave* саобраћај на интернет протокол контролер. Или би можда произвођач контролера имплементирао уређаје за

конверзију протокола, које уређаји корисника највише користе, и тако омогућио кориснику да одреди које протоколе је потребно да његов систем подржава.

Уз то, могуће је повезати један или више централних контролера, као што је и сам усмеривач уређај (Поглавље 4.2) радио, па тако омогућио да се са једног контролера контролишу уређаји у јако великим кућама, зградама, хотелима или великим производним постројењима, у којима због домета мреже не би било могуће имати само један контролер.

На крају, имплементирана су и две основне функције уређаја: подршка за надоградњу управљачког софтвера и подршка за чување резервне копије података. Међутим, ове карактеристике омогућавају *WISE* протоколу да, са првом, постане протокол по функционалности раван са осталим протоколима коришћеним у паметним кућама, а са другом се уздигне изнад других комуникационих протокола и тако, нудећи корисницима отпорност уређаја на грешке, искористи потенцијал пропусног опсега своје основе, интернет протокола.

7. Литература

[1] *MQTT* протокол за комуникацију између апликација, http://mqtt.org/, јануар 2021.

[2] Спецификација *HTTP* протокола, https://tools.ietf.org/html/rfc2616, јануар 2021.

[3] *cURL* документација алата и библиотека, https://curl.se/docs/, јануар 2021.

[4] *JavaScript Object Notation* документација, https://www.json.org/json-en.html, јануар 2021.

[5] Протокол за комуникацију *IP* уређаја са централним контролером паметне куће, Игор Стефановић, Универзитет у Новом Саду, Факултет техничких наука, 2018.

[6] Спецификација *ZigBee* протокола, https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf, јануар 2021.

[7] Спецификација *Z-Wave* протокола, https://www.silabs.com/wireless/z-wave/specification, јануар 2021.

[8] Спецификација *UPnP* протокола, https://tools.ietf.org/html/rfc6970, јануар 2021.

[9] *POCO* колекција библиотека, https://github.com/pocoproject/poco, јануар 2021.

[10] Систем за управљање базом података коришћен у систему, https://www.sqlite.org/index.html, јануар 2021.

[11] Спецификација *Bluetooth* протокола, https://www.bluetooth.com/specifications/protocol-specifications/, јануар 2021.

[12] Алат за тестирање *JMeter*, https://jmeter.apache.org/, јануар 2021.