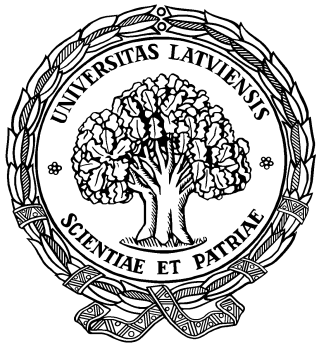


Latvijas Universitāte

Datorikas fakultāte

Modulārā viedās mājas sistēma

Kursa darbs

Autors:

Iļja Gubins

(st. apl. nr. ig11075)

Darba vadītājs:

Leo Seļāvo,

Prof. Dr. dat

Latvijas Universitāte

Rīga, 2014

Anotācija

Šī darba mērķis – analizēt un proektēt viegli paplašināmu modulāru viedās mājas sistēmu, kā arī izveidot prototipu. Mans darbs sastāv no šādām daļām:

Sistēmu projektēšana:

- Analizēt un salīdzināt esošo risinājumus
- Izvēlēties nepieciešamo funkcionalitāti
- Augsta līmeņa dizains

Prototipa izstrāde:

- Zema līmeņa dizains
- Tehnoloģiju salīdzinājums un izvēle
- Prototipa implementācija

Viedās mājas sistēma ļaus gāla lietotājiem ietaupīt gan laiku, gan naudu, automātiski (izmantojot iepriekš konfigurētus triggerus un limitus) vai manuāli (ar speciāli izveidoto mājas lapu) kontrolēt pieejamas mājas ierīces. Mājas lapā būs arī iespēja apskatīt vēstures datus, uzstādīt un konfigurēt pievienotas ierīces un norādīt triggerus un limitus.

Atslēgvārdi: viedās māja, bezvadu sensoru tīkli, sensor, aktuātors, monitorēšana, mājas vadība

Abstract

Main purpose of this work – to analyze and design easily extendable modular smart home system, aswell as create a prototype of it. My work consists of the following parts:

System design:

- Analyze and comparison of the existing solutions
- Choice of required functionality
- High level design

Prototype creation:

- Low level design
- Comparison and choice of the most fitting technologies
- Implementation of the prototype

Smart home system will allow end users to save both time and money, by automatically (via previously configurated triggers and thresholds) or manually (via special web-page) controlling connected home devices. Web-page will also allow to view history data, setup and configure connected devices and specify triggers and thresholds.

Keywords: smart home, wireless sensor network, sensor, actuator, monitoring, home management

Аннотация

Главная цель этой работы – проанализировать и спроектировать легко расширяемую, модулярную систему умного дома, а так же создать прототип такой системы. Моя работа состоит из следующих частей:

Проектирование системы:

- Анализ и сравнение существующих решений
- Выбор необходимого функционала
- Проектирование высокого уровня

Создание прототипа:

- Проектирование низкого уровня
- Сравнение и выбор подходящих технологий
- Имплементация прототипа

Система умного дома позволит конечным пользователям экономить время и деньги, управляя устроиствами дома в автоматическом режиме (посредством настроеных условий) и вручную, через удобную веб-страницу, на которой кроме управления будет так же доступны история показаний, настроики подключеных устроиств и управление условиями-триггерами.

Ключевые слова: умный дом, беспроводные сенсорные сеть, сенсор, актуатор, мониторинг, управление домом

Saturs

Apzīmējumu un jēdzienu apraksts

Viedās māja - dzīvokļa vai mājas automatizācijas virziens, ar kura palīdzību ir iespējams attālināti vadīt siltumapgādes sistēmu, elektrosistēmu vai atsevišķus elementus.

Mezglutīkls - Tīkls, kurā ir vismaz divi mezgli, kurus savieno divi vai vairāki ceļi.

Dēmons - Fona programma, kas vajadzības gadījumā tiek izmantota datoros, kuri darbojas UNIX vai kādas citas operētājsistēmas vidē. Dēmons veic tādus pakalpojumus kā, piem., elektroniskā pasta ziņojumu maršrutēšanu, un tas parasti darbojas bez lietotāja iejaukšanās.

Ievads

Šī darba mērķis – analizēt un proektēt viegli paplašināmu modulāru viedās mājas sistēmu, kā arī izveidot prototipu.

Pasaulē jau ir eksistējoši viedās mājas sistēmas risinājumi, un būtu negudri neapskatīt tos. Tāpēc, savā darbā es apskatīju vispopulārakus bezvadu protokolus, kā *Zigbee*, *Z-Wave,* un arī gan vadu, gan bezvadu protokulu *X10*.

Diemžēl, visi apskatīti protokoli ir proprietāri un ir diezgan dārgi. Tāpēc es nolēmu izveidot savu sistēmu. Pēc pirmo apskātu, es nolēmu ka tā dzīvotspējīga idēja, jo manuāli izveidot nepieciešamos komponentus ir līdz 10 reizes lētāk nekā pirkt gatavus ierīces. Lai to īstenot dzīvē, bija nepieciešami izveidot specifikācijas un saproektēt augsta līmeņu funkcionalitāti.

Bija nolēmts, ka sistēmai ir jābūt bez vadiem, ar vislētakiem komponentiem, bet pietiekāmi drošā lai strādāt reālos dzīvokļa apstākļos. Sistēmai jābūt viegli pieejamo lietotāju saskarne, kas atbalstis arīmobīlās ierīces.

1. Sistēmu projektēšana

Šajā nodaļā es gribu apskatīt teoriju aiz viedas mājas sistēmas: esošo risinājumi, funkcionalitātes izvēle un manas sistēmas augstākas līmeņas dizains.

1.1. Esošo risinājumu analīze

Šajā apakšnodaļā es gribu apskatīt esošo risinājumus un izveidot salīdzinājumu tabūlu kur būs apvienota visa informācija par protokoliem.

1.1.1. X10

X10 ir viens no pirmājiem protokoliem. X10 protokols bija izveidots 70os gādos. Pirmā protokola versija bija tikai caur mājās elektrolīniju un tikai pēc tam, galu galā gāja bezvadu. X10 ir zināms ka protokols ar vismazāko ātrumu un vismazāk iespējam. X10 tehnoloģija jau novecoja un ir ļoti iesakāms instalēt kaut ko saderīgāko ar jaunākiem bezvadu standartiem, jo X10 ar katru gādu ir grūtāk un grūtāk uzstādīt. Bet X10 ir joprojām de facto standarts.

1.1.3. Z-Wave

Z-Wave ir bezvadu mājas automatizācijas protokols, kas strāda uz 908.42MHz frekvenču joslā. Tas ir salīdzinoši jauns mājas automatizācijas protokols, bet ir pieaudzis diezgan strauji pēdējos gados. Grupa aiz tā protokola, Z-Wave Alliance, kas tagad apvieno vairāk nekā 1000 dažādām ierīcēm. Viena no galvenajām iezīmēm Z-Wave ir to ka to izmanto mezglutīklu, kas būtībā nozīme, ka viens Z-Wave produkts sūt informāciju caur citu, un tā tālāk līdz tas sasniedz paredzēto galamērķi. Tam arī ir ļoti zema enerģijas paterīņa, kas ir ideāli piemērots ierīcēm, kas balstās uz akumulatora enerģiju.

1.1.3. Zigbee

ZigBee ir 802 bezvadu sakaru IEEE standarts. Tā patērē ļoti mazu elektroenerģijas daudzumu un izmanto mezglutīklu, kā Z-Wave. Nedaudz lētāks nekā Z-Wave un vairāk

1.1.5. Salidzinājums

Ipašība	X10	Zigbee	Z-Wave
Komunikācijas metods	Bezvadu, 433 MHz	Bezvadu, 2.4Ghz (IEEE 802.15.4)	Bezvadu, 868.42 MHz
Gāds	1975	2005	2008
Protokols	Izmanto vienkāršu bitu struktūru lai pārsūtīt komandas. Nav pareizības pārbaudes.	Izmanto data paketes lidzīgas Ethernetam. Katram paketem ir datu pareizības pārbaude.	
Ātrums	1x	250kbit/s	100kbit/s
Instalācija	Papilda aparatūra nav vajagdzīga priekš uzstādīšanai. Sistēmu var uzstādīt gala lietotājs.		
Maks. ierīces vienā tiklā	256	~64000	232
Traucējumu kompensācija	Nav	Ir	Ir
Uzticamība	Pārraidītam komandām nav apstiprinājumu. Ja komanda tiek zaudēta, ne sūtītājs, ne saņēmējs nav onfirmēts par palaistu garām komandu.	Visam komandām ir saņēmšanas apstiprinājumi. Ja tiek konstātēts komandas zaudējums, signāla avots atkārotit mēģinās atsūtīt komandu. Ja tiek konstātēts ierīces zaudējums vienā signālas maršrutā, komanda var bū novadīta caur citiem ierīcem.	
Mezglutīkls	Nav	Ir	Ir
Drošība	Nav	Ir	Ir
Iebuvētā enerģijas taupīšana	Nav	Ir	Ir
Ierīces status apprasīšana	Nav	Ir	Ir
Sākuma komplekta cena	160 €	350 €	400 €

1.2. Funkcionalitātes analīze

Šajā apakšnodaļā es gribu apskatīt iespējamo gudras mājas iespējas un kādi ir vajagdzīgi manām projektam.

1.2.1. Sensoru rādījumi

Tiem jābūt vērtības-agnostic, t.i. pieņem jebkurus vērtības, jebkādā formāta – temperatūra, gaisa mitrums, apgaismības, utml. Sensori.

1.2.2. Aktuatori

Jābūt iespēja atsūtīt signālu kas ieslēgs, izslēgs vai citādi modificēs ierīces apakšsistēmu – piemēram ja mezglam būs pievienots relēja ar "gudru" lampu, jābūt iespējai ieslēgt, izslēdz un arī mainīt gaismas spožumu.

1.2.3. Notikumi

Sistēmai jāģenerē notikumus kas ir izdoti kad kaut kāds iepriekš definēts nosacījums tiek izpildīts. Piemērām, ja temperatūra tiek pārsniegta 26C, ieslēgts gaismas kondicionieru.

1.2.4. Vestūre

Visiem notikumiem un sensoru rādītajiem jābūt saglabātiem, ar visiem laika zīmogiem.

1.2.5. Vairāki lietotāji

Sistēmai jābūt vairākiem lietotāju limēniem – administrātors, kas var mainīt, piemērām, sensoru rādījumu intervalu un administrēt sistēmas mezglus; lietotājs, kas var apskātīt vestūre un arī izmantot mezglus; viess, kas var apskātīt tikai tiekošo situāciju un tikai nekritiskiem elementiem.

1.2.6. Viegla jauno ierīču instalācija

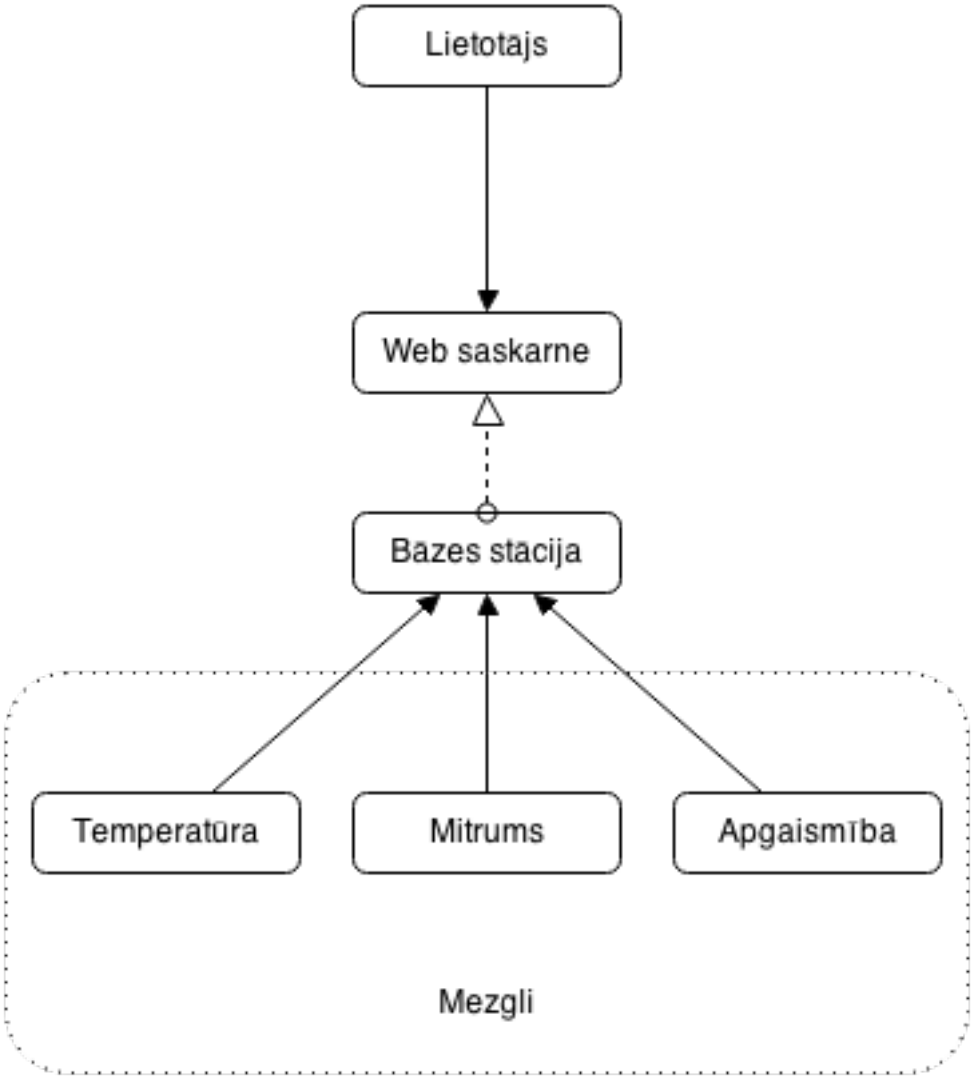
Jauno ierīču pievienošanai jābūt vieglam un ātram, optimāli – ieslēgt ierīce, iekš saskarne apstiprināt ka tas ir tavs mezgls un sakonfigurēt to pēc vajadzībām.

1.2.7. Sistēma pieejama attālināti

Visiem darbībam jābūt pieejamiem arī attālināti, ne tikai kaut kāda vienā vietā vai datorā. Piemērām, internētā, privātā serverī, vai uz bāzes stācijam. Sistēmai arī jābūt ērti

pieejamai no telefonā – vai nu speciālā lietojumprogrammatūrā, vai caur pārlūkprogrammai.

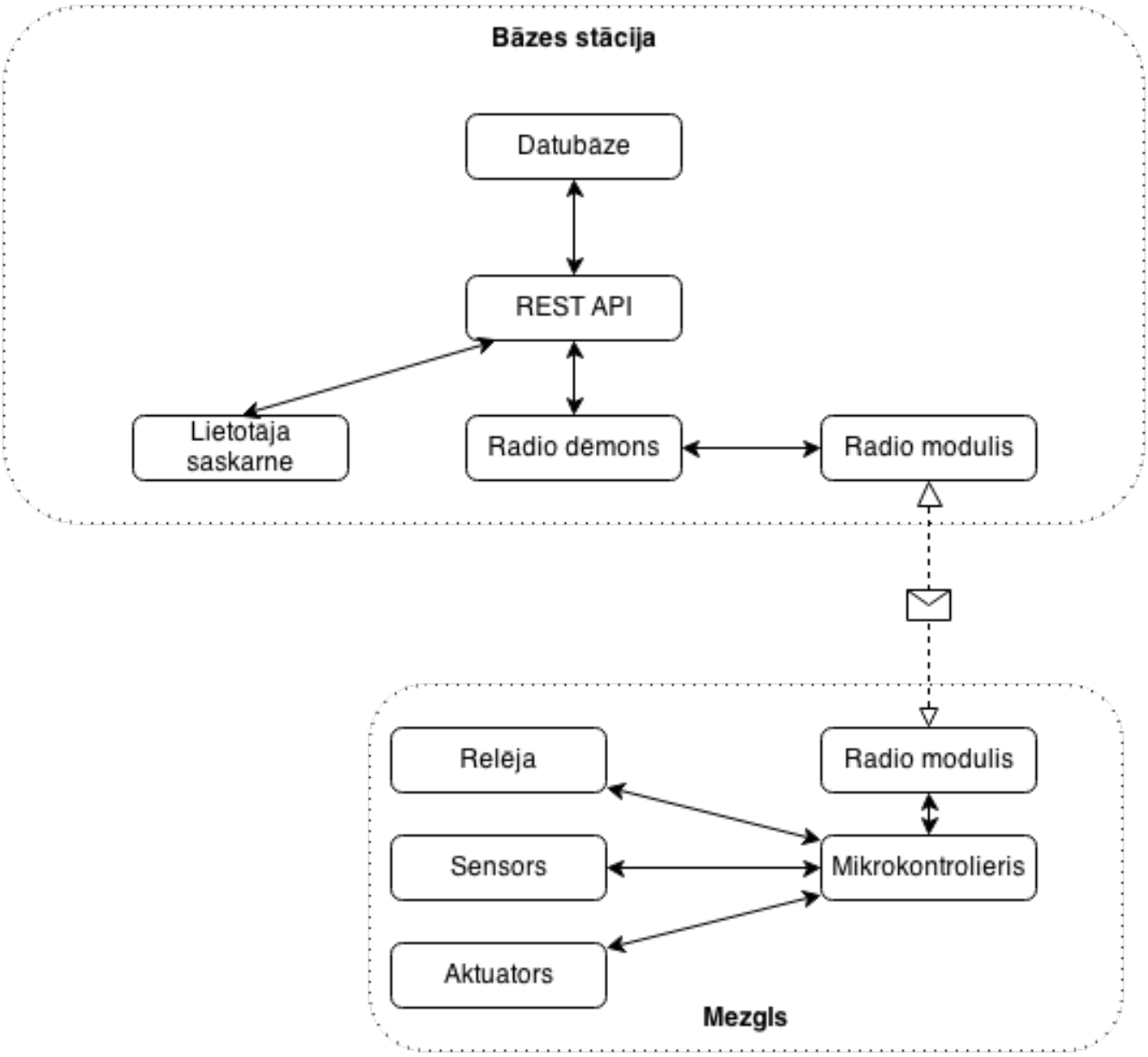
1.3. Augstākas līmeņas sistēmas dizains



2. Prototipa izstrāde

Šajā nodaļā es gribu apskatīt teoriju aiz viedas mājas sistēmas: esošo risinājumi, funkcionalitātes izvēle un manas sistēmas augstākas līmeņas dizains.

2.1. Zema līmeņa dizains



2.2. Tehnoloģiju salīdzinājums un izvēle

Šajā apakšnodaļā es gribu apskatīt iespējamo datoraparatūras tehnoloģijas kas varētu būt izmantoti manā implementācijā un argumentācija kāpēc tas tiek izvēlets.

2.2.1. Datoraparatūras tehnoloģijas

2.2.1.1. Bāzes stācija

Kā jau bija definēts līdz šīm, bāzes stācijai jābūt pietiekāmi jaudīgai lai atbalstīt internēta pieslēgumu, datubāze, datubāzes API, lietotāja saskarne serveri un jābūt iespējas paplašināt tādu iekārtu ar radio moduli, kaut kāda veidā. Protams, var izmantot tradicionālu personālu datoru, bet tas ir ļoti neefektīvi, gan no enerģijas paterīņa, gan no resursu viedokļa. Visloģiskāk liekas izmantot vienas plates ARM arhitektūras datoru, kur ir pieejāmi paplašinājuma saskarne, vai nu USB, vai GPIO.

Potenciāli: Raspberry Pi un BeagleBoard Black. Apskatīsīm tos vienā tabūlā:

Ipašība	Raspberry Pi	BeagleBoard Black	Komentāri
Procesors	Broadcom BCM2835 ARM11 @ 700 MHz	TI Sitara AM3359AZCZ100 Cortex A8 @ 1GHz	Neskatoties uz vienu un to pašu frekvenci, A8 ir apmērām 60% jaudigāk nekā ARM11
GPU	VideoCore IV	PowerVR SGX530	GPU īsti netiek izmantots, bet tomēr VideoCore IV ir daudz jaudigāk, nekā PowerVR.
RAM	512 MB SDRAM @ 400 Mhz	512 DDR3L @ 400 Mhz	DDR3L izmanto māzāk elektroenerģijas nekā citi
Datu krātuve	SD kartas slots	2GB eMMC + microSD slots	eMMC ir tik pats ātrs kā 10. klasses SD
Ethernet	10/100M	10/100M	-
USB	2	2	-
Paplašinājumi	12 GPIO, USART, SPI, I2C, CSI, DSI	65 GPIO, SPI, I2C	Iekš BBB ir daudz vairāk GPIO pinus, bet Pi atbalsta vairākus pereferijas formātus
OS	ARMv6 operetājsistēmas, oficiāli – Debian un Arch Linux ARM	Jebkuras ARM operetājsistēmas, oficiāli – Ubuntu un Angstrom	Principā, visērtāk būtu izmantot tīros variantus kā Debian, nevis forkus Ubuntu. Debian nav tik user-frienly, bet vairāk iespējas
Sabiedrības aktivitāte	Ļoti aktīva sabiedrība, vispopulārakais ARM vienas plātes dators	Aktīva sabiedrība, ļoti daudz profesionālus	Šeit Raspberry Pi, neapšaubami, ir līderis

Īpašība	Raspberry Pi	BeagleBoard Black	Komentāri
Cena	$35	$200	Gandrīz x6 starpība!
Pieejamība LV	Ir	Ir	-

Neiskaitot to ka priekš Raspberry Pi vēl ir vajadzīga ātrā SD kārte (10. klasses), joprojām sanāk ļoti liela starpība cenā. Bija nolēmts, ka pat BBB ir nedaudz labāk atsevišķos gadījumos, Pi vairāk atbilst iespējam un kursa darbas kompetencei.

Izvēle: Raspberry Pi

2.2.1.2. Mezgli

Mezglam jābūt pietiekāmi energoefektīvam, pietiekāmi jaudīgam lai izmantot radio moduļu, ar pietiekāmi daudz pinus lai pieslēgt sensorus vai aktuātorus. Visērtāk būtu izmantot Arduino vai kaut kādu Arduino klonu. Veiksmīgi, man jau ir Arduino Uno, kas labi atbilst visam prasībam.

Izvēle: Arduino Uno

2.2.1.3. Radio modulis

Radio modulim jāizmantot brīvo frekvenci, kas ir atļāuti visiem; jābūt pievinojam pie izvēlēto ierīcem – Arduino Uno un Raspberry Pi. Sanāk tā, ka man jau bija pieejāmi moduli – nRF24L01. Laimīgi, modulim arī ir gatava bibliotēka. Tāpēc,

Izvēle: nRF24L01

2.2.2. Programmatūras tehnoloģijas

Šajā apakšnodaļā es gribu apskatīt iespējamo programmatūras tehnoloģijas kas varētu būt izmantoti manā implementācijā un argumentācija kāpēc tas tiek izvēlets.

2.2.2.1. Datubāze

Datubāze ir ļoti svarīga daļa, kas ir pamatelements gudras mājas sistēmai. Tā kā ir ļoti daudz datubāzes un gandrīz visi var būt izmantoti, bija nolēmts vispirms izveidot prasības:

- Publish – Subscribe modēļa atbalsta, priekš reāllaika notikumiem un izmaiņam

- Raspberry Pi (ARMv6) atbalsta

Pēc apspriedumiem, bija izvēleti sekojoši datubāzes varianti: Redis, MongoDB, PostgreSQL.

Ipašība	Redis	MongoDB	PostgreSQL
Metamodelis	NoSQL	NoSQL	SQL
ARMv6 Linux atbalsta	Ir	Vajag manuāli kompilēt	Ir
Rezerves kopēšana	Ir	Ir	Ir
Pub/Sub pattern	Ir	Ir	Ir (nedaudz modificēts un zināms kā NOTIFY/LISTEN)
Oficiāli adaptēri	Gandrīz visam valodam		
Iss apraksts	Redis nav datubāze savā pilnā nozīmē, tas drīzāk ir key-value krātuve	MongoDB katrs ieraksts ir JSON objekts. Ir viegli mainīt iekšejo struktūru, bet joprojām var validēt datus, bet ietvara līmenī	Ļoti advansēts, daudz datu tipus un stored procedūras iespējas, bet nav ļoti populārs un ir diezgan lēns, salidzinājot ar citiem SQL un NoSQL datubāzem

Izvēle: MongoDB

2.2.2.2. REST API un lietotāja saskarne

REST API būs "līme" starp visam darbībam un datubāze. Kā to izdarīt ir ļoti daudz iespējas, bet vajag arī domāt par resursu pateriņu, jo Raspberry Pi nevar, piemērām, izmantot pīlno Java virtuālo mašinu, tikai Embedded versiju kā arī uz Linux nevar izmantot .NET. Potenciāli tehnoloģijas: PHP, Go, Node.js.

Go ir jauna programmēšanas valoda kas mani stripri interēsē, bet tomēr tā ir pārāk jauna un nestabīla. Ļoti daudz būs jāraksta pa jaunam, jo vēl nevisas bibliotēkas ir portēti uz Go.

PHP ir skriptēšanas valoda, kas redzēja ļoti daudz, bet viņai ir mīnusi, kas būs pārāk problēmatiski apiet, piemērām, lai izmantot Websockets uz PHP, vajag veidot atsevišķu procesu kas būs atbildīgs par Websocket savienojumiem. PHP arī nevar strādāt ar savienojumiem pa taisni, vajag vēl vienu procesu, kas būs atbildīga par to, piemērām, Apache vai Nginx.

Node.js, manuprāt, ir ideāla vīdē. Tā ir pietiekāmi jauna lai natīvi atbalstīt jaunākas tehnoloģijas (piemērām, Websockets, kas man ir vajadzīgi); tām ir ļoti ērta pakešu sistēma – NPM, kas atvieglo dzīvi un atļauj viegli un ātri instalēt jaunus apakšmoduļus; Node.js sabiedrība ir ļoti aktīva un ļoti daudz entuziastus kas veido daudz jaunus moduļus un ietvarus; Node.js izmanto tikai vienu pavedienu savam procesām un izmanto ļoti mazs resursu (kas arī ir ļoti lietdērīgs uz Raspberry). Tāpēc bija nolēmts izmantot Node.

Bet tas nav viss, priekš Node vajag arī izvēlēties ietvaru. Kā vispiemērotākais ietvars bija izvēletis SailsJS.

ExpressJS ir diezgan minimāls ietvars. Tas nav slīkti, jo tas atļauj ļoti lokanu programmatūras attīstību, bet tomēr tagad ir vajadzīgs kaut kas cits, kur jau ir vairākas iespējas. Express bija izmests no izvēle ļoti ātri, gandrīz uzreiz pēc izvēlēšanas sakuma, jo viņas konkurentiem ir daudz vairāk iespējas un tie ir vairāk piemēroti priekš REST API.

SailsJS atļauj vienā komandā izveidot resursu, kas būs pieejāms uzreiz ar visam pareiziem REST darbībam (scaffolding). SailsJS ir adapteris uz MongoDB, kas natīvi atbalst Pub/Sub. SailsJS arī atļauj izveidot lietotāja saskarne, ne tikai

Izvēle: Node.JS un SailsJS ietvars

2.2.2.3. Radio modulis

Laimīgi, izvelētam radio modulim nRF24L01 jau ir izveidota stabīla bibliotēka – RF (github.com/maniacbug/RF24). Tajam pašam lietotājam arī ir cita bibliotēka – RF24Network

(github.com/maniacbug/RF24Network), kas pieļauj izveidot mezglutīklu, bet tagad, pirma prototipa versijā, es izmantošu vienkāršo variantu – RF24. Alternatīvas priekš RF24 ir, bet diezgan nestabīli un nav izturīgi.

Izvēle: RF24

2.2.2.4. Radio dēmons

RF24 bibliotēka ir rakstīta uz C++, bet viņa bija portēta arī uz citam valodam. Tomēr tieši C++ ir vislabāk piemērots darbai ar zēmas līmeņas iekartam ka nRF24L01.

Izvēle: C++ ar standartam bibliotēkam + RF24

2.3. Prototipa implementācija

Šajā apakšnodaļā es gribu apskatīt gudras mājas implementācijas aspektus.

2.3.1. Radio pakešu specifikācija

Tiem jābūt vērtības-agnostic, t.i. pieņem jebkurus vērtības, jebkādā formāta – temperatūra, gaisa mitrums, apgaismības, utml. sensori, kā arī aktuātoru zīņas. Paketem ir JSON struktūra, jo tā ir vieglāk apstrādāt. Piemērs radio paketam:

```
{
    fromId: 'mezgluId',
    timestamp: '123456789',
    value: '36.6'
}
```

Kāda tipa mezgls jau ir definēts mezgla instalācijas laikā (un var būt izmainta caur lietotāja saskarne), tāpēc vajag tikai vērtību, laika zīmogu un sūtītāja identifikātoru.

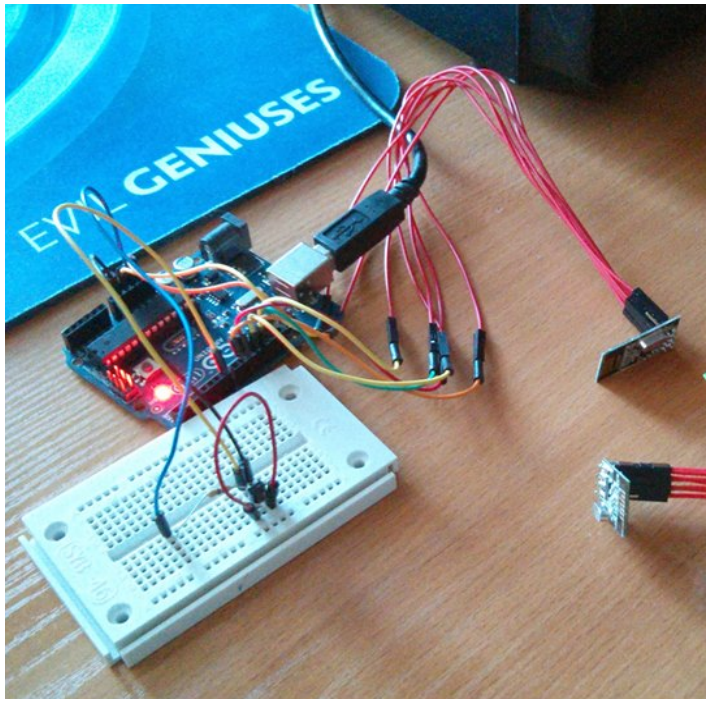
2.3.2. Pieejāmi galapunkti

Galapunkts	Apraksts
GET /devices	Atgriež HTML, kur var apskatīt visus ierīces sistēmā.
GET /devices/id	Atgriež HTML, kur var apskatīt noteiktu ierīce.
POST /devices/found	Jauni, nepievienoti sistēmai ierīces atsūta šeit informāciju par sevi. Atgriež status kodu, veiksme ja bija pievienots jauna ierīce, neveiksme ja jauna ierīce nebija pievienots.
POST /devices/pairing	Atgriež status kodu, veiksme ja slēpenais ID kas bija ievadīts ir pareizs, neveiksme pretējā gadījumā.
GET /devices/setup/id	Atgriež HTML, kur var iestādīt jauno ierīce sistēma. Tājai vispirms jābūt pievienotam caur /devices/found.
POST /devices/setup/id	Novirz atpakaļ uz tādu pāšu GET.
GET /devices/edit/id	Atgriež HTML, kur var rediģet ierīces konfigurāciju.
POST /devices/edit/id	Novirz atpakaļ uz tādu pāšu GET.
GET /devices/delete/id	Dzēst ierīce no sistēmas. Novirz uz GET /devices.
WS /subscribe/devices	Pieejāms tikai ar WebSockets. Abonē websocket notikumus kas saistīti ar ierīcem.
WS /subscribe/incoming	Pieejāms tikai ar WebSockets. Abonē websocket notikumus kas saistīti ar jauniem ienakošiem atskaites no ierīcem.
POST /report	Galapunkts priekš radio dēmonu. Uz šejieni tas sūt visu informāciju saņemtu no mezgliem, piem. Sensora vērtības. Atgriež status kodu,

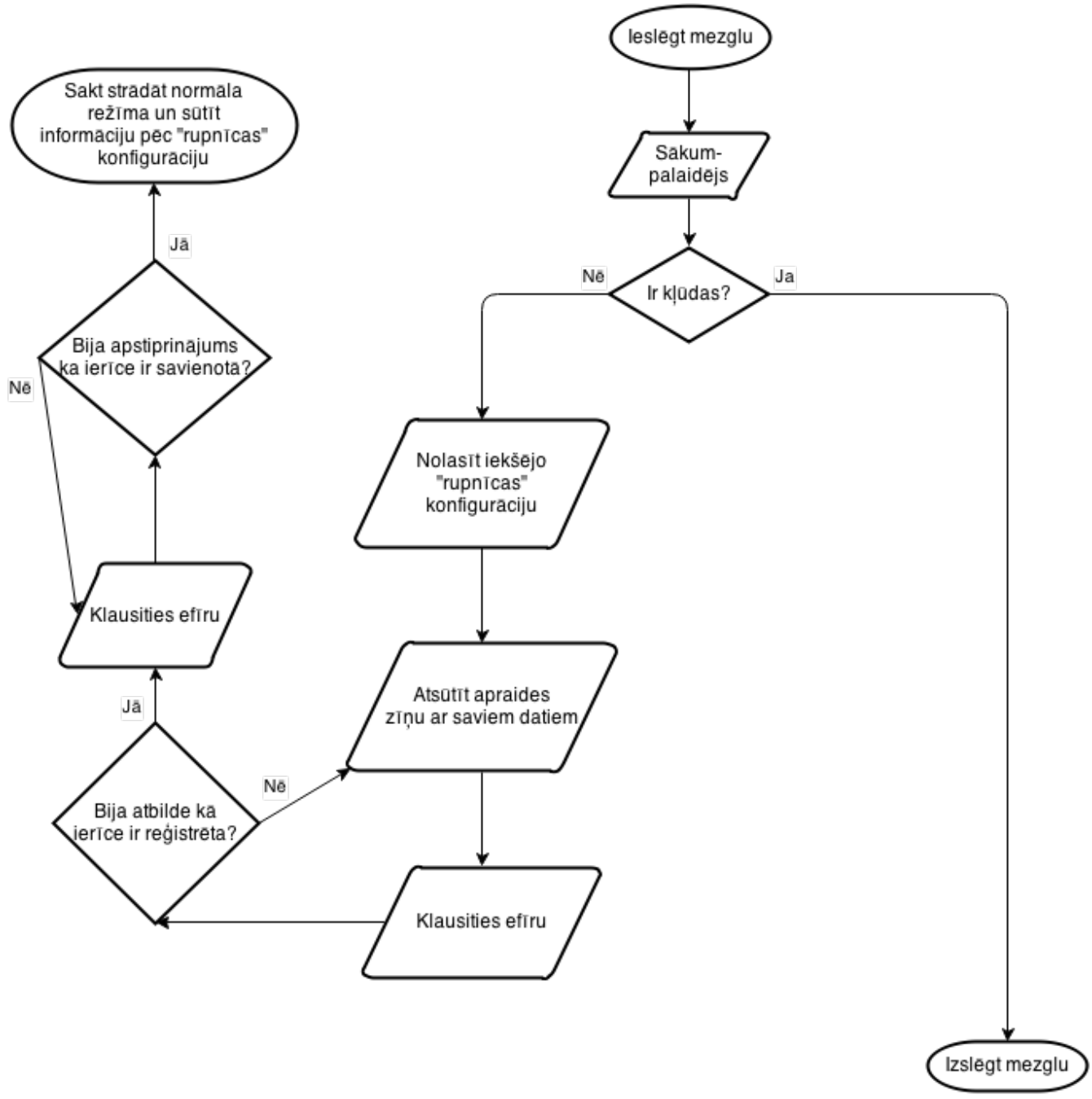
Galapunkts	Apraksts
	veiksme ja vērtība bija pievienota, neveiksme prētējā gadījumā.
GET /history	Atgriež HTML, kur var apskatīt visus pēdējus notikumus sistēmā.
GET /history/id	Atgriež HTML, kur var apskatīt kādu noteiktu notikumu sistēmā.
GET /triggers/	Atgriež HTML, kur var apskatīt visus pievienotus trigerus sistēmā.
GET /triggers/id	Atgriež HTML, kur var apskatīt kādu noteiktu trigeru sistēmā.
POST /triggers/id	Novirz atpakaļ uz tādu pāšu GET.
POST /triggers/add	Novirz uz GET /triggers/id
POST /triggers/delete/id	Dzēst trigeru no sistēmas. Novirz uz GET /triggers/.
GET /dashboard	Atgriež HTML, kur var apskatīt tiekošo kopējo statusu sistēmai un arī ātri izmantot kādus populārakus aktuātorus, piem. ieslēgt gaismu.
POST /dashboard/use/	Šeit ir atsūtīta informācija ar to kuru aktuatoru un ko tieši vajag izdarīt. Novirz uz GET /dashboard

2.3.3. Sensora mezgls

Fotografijā ir redzāms sensoru mezgls – uz maketēs plates ir temperatūras sensors; ar sarkaniem vadiem ir pievienots radiomodulis; ar lielo melnu – elektrības vads.



2.3.4. Mezglu iestatīšanas protokols



Secinājumi

Savā kursa darba rezultātā, es:

- noprojektēju viedās mājas sistēmu: noanalizēju un salīdzināju esošo risinājumus, izvēlējos nepieciešamo funkcionalitāti, izveidoju augsta līmeņa dizainu;
- izveidoju prototipu tādāi sistēmai: sākot no zema līmeņa dizainu, tehnoloģiju salīdzinājumu un izvēlei un izveidoju pāšu prototipu.

Atkarība no esošiem risinājumiem, prototips varbūt nav tik profesionāli izskātams un varbūt nav tik labi notestēts, bet pašizmaksa tādai sistēmai ir ļoti māza, salidzīnājot ar gataviem risinājumiem – uz visu sistēmu bija izmantots apmērām 60 eiro. Par vislētāku tādu pašu sistēmu būtu jāmaksa vismaz 100 eiro vairāk.

Prototips labi pārāda ka tāda koncepcija strāda un pēc vairāk darbu uz šo, tāda sistēma varbūt izmantota, piemērām manā mājā un saglabāt gan laiku, gan naudu un dod pārliecību ka mājās viss ir labi. Savu darbu rezultātu es gribu turpināt izstrādāt un izmantot savā mājā.

Pielikums 1. Programmatūras koda fragments

```
/**
 * SubscribeController
 *
 * @description :: Server-side logic for managing pub/sub events.
 */

module.exports = {

  /**
   * `SubscribeController.subscribeDiscoveredDevices()`
   * Allows clients to subscribe for publish events.
   * Used on index page - to monitor new discovered devices.
   */
  subscribeDiscoveredDevices: function(req, res) {
    if (req.isSocket) {
      async.parallel({
        nodes: function(cb) {
          Node.find().sort('added DESC')
            .exec(function(err, nodes) {
              cb(err, nodes);
            });
        },
        discoveredNodes: function(cb) {
          DiscoveredNode.find().sort('added DESC')
            .exec(function(err, discNodes) {
              cb(err, discNodes);
            });
        }
      }, function(err, results) {
        if (err) {
          console.log(err);
          return res.send(500);
        } else {
          Node.watch(req.socket);
          DiscoveredNode.watch(req.socket);

          return res.send(200);
        }
      });
    } else {
      return res.send(403);
    }

  }

};
```

Literatūras avotu saraksts

1. **Node.js dokumentācija**
2. **SailsJS dokumentācija**
3. **Z-Wave oficiāla lapa**
4. **X10 oficiāla lapa**
5. **ZigBee oficiāla lapa**
6. **X10, Z-Wave, Zigbee salidzinājums**
7. **Maniacbug's RF24 bibliotēka priekš nRF24L01**
8. **Socket.io dokumentācija**