UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matjaž Milanič

Nadzor pametnih naprav

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, 2016

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matjaž Milanič

Nadzor pametnih naprav

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

MENTOR: prof. dr. Marko Bajec

Ljubljana, 2016

To delo je ponujeno pod licenco *Creative Commons Priznanje avtorstva-Deljenje pod enakimi pogoji 2.5 Slovenija* (ali novejšo različico). To pomeni, da se tako besedilo, slike, grafi in druge sestavine dela kot tudi rezultati diplomskega dela lahko prosto distribuirajo, reproducirajo, uporabljajo, priobčujejo javnosti in predelujejo, pod pogojem, da se jasno in vidno navede avtorja in naslov tega dela in da se v primeru spremembe, preoblikovanja ali uporabe tega dela v svojem delu, lahko distribuira predelava le pod licenco, ki je enaka tej. Podrobnosti licence so dostopne na spletni strani creativecommons.si ali na Inštitutu za intelektualno lastnino, Streliška 1, 1000 Ljubljana.

Izvorna koda diplomskega dela, njeni rezultati in v ta namen razvita programska oprema je ponujena pod licenco *GNU General Public License*, različica 3 (ali novejša). To pomeni, da se lahko prosto distribuira in/ali predeluje pod njenimi pogoji. Podrobnosti licence so dostopne na spletni strani http://www.gnu.org/licenses.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Z razmahom interneta stvari je na voljo čedalje več naprav, ki so sposobne oddajanja in prejemanja podatkov, komunikacije prek interneta ter v posameznih primerih tudi obdelave podatkov in posledično odločanja glede na okoliščine. Scenariji, ki govorijo o napravah, ki so sposobne samouravnavanja ter sprožanja kontekstno odvisnih akcij (npr. hladilnik sam zazna, da primanjkuje mleka in ga doda na družinski seznam za nakup ali celo naroči prek interneta), ne zvenijo več futuristično, ampak so povsem izvedljivi. Z novimi tehnologijami pa kot običajno prihajajo tudi nove pasti in tveganja. Več odločanja, kot ga prepuščamo napravam in strojem, večja je nevarnost, da gre kaj hudo narobe. Stroj namreč nima intuicije in se v nedefiniranih okoliščinah lahko odzove povsem nepričakovano. S tem nastaja velika potreba po nadzornih sistemih, ki bodo omogočali stalno preverjanje stanja pametnih naprav, ki jih uporabljamo. V diplomskem delu preučite to področje ter razvijte nadzorni sistem za pregledovanje stanja pametnih naprav. Kot osnovo lahko uporabite eno izmed odprtokodnih IoT platform.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
OSGi	open services gateway initiative	pobuda za odprt servisni prehod
TSDB	time series database	podatkovna baza za časovna zaporedja
XML	extensible mark-up language	razširljivi označevalni jezik
UI	user interface	uporabniški vmesnik
DTO	data transfer object	objekt za prenašanje podatkov
PPPoE	point-to-point protocol over Ethernet	protokol za povezavo od točke do točke prek omrežja Ethernet
CPU	central processing unit	centralna procesna enota
SSH	secure shell	varna lupina
SNMP	simple network management protocol	enostaven protokol za upravljanje omrežja

Povzetek

Naslov: Nadzor pametnih naprav

Ljudje vedno več opravil prepuščamo pametnim napravam in tako postajamo vedno bolj odvisni od njihovega pravilnega delovanja. Pojavi se potreba po nadziranju teh naprav, ki je v poslovnem svetu že stalnica, kmalu pa bo začela postajati zanimiva tudi za domače uporabnike. Diplomska naloga opisuje postopek izdelave sistema za nadziranje pametnih naprav v domačem okolju od faze načrtovanja do končne implementacije. Vodilo razvoja je bilo narediti čimbolj enostaven in uporabniško prijazen sistem, hkrati pa izdelati karseda proaktiven nadzorni sistem, ki deluje na osnovi metode nadziranja prozorne škatle. Rešitev je osnovana na odprtokodnem sistemu za upravljanje in avtomatizacijo pametnih domov openHAB oziroma Eclipse SmartHome. Sistem nadzira delovanje vseh uporabnikovih naprav povezanih v okolje openHAB in v primeru morebitnih napak o tem obvesti uporabnika s pomočjo kontekstno obogatenih sporočil.

Ključne besede: diplomska naloga, nadzor, nadziranje, obveščanje, metrike, pametne naprave, internet stvari, openHAB, Eclipse SmartHome

Abstract

Title: Controlling smart devices

Smart devices make our lives easier. However, by delegating more and more tasks to smart devices our wellbeing becomes dependent on their correct functioning. This creates a need for monitoring home devices in a similar way as businesses already do. The thesis describes the steps required for building a monitoring system suited for smart home monitoring, from the initial design to the final implementation of the solution. The goal was to create a user friendly monitoring system, designed as much as possible as a proactive monitoring solution, based on the white-box monitoring approach. The solution is designed to work with the openHAB home automation software, which has been lately split and has become an extension to Eclipse SmartHome. The system monitors all devices connected with openHAB and generates context rich alerts whenever a device shows signs of faulty behaviour.

Keywords: BSc Thesis, monitoring, alerting, metrics, smart devices, internet of things, openHAB, Eclipse SmartHome

Poglavje 1 Uvod

Število pametnih naprav strmo narašča. Vedno več naprav je sposobnih komunicirati s sosednjimi napravami in avtonomno sprejemati odločitve glede na vhodne podatke iz okolice. Tako imamo na voljo pametna stikala, ki avtomatsko zaprejo okna, če so napovedane padavine, in pametni termostat, ki uravnava temperaturo prostora glede na to, ali je prostor v uporabi ali ne. Napravi za hranjenje hišnih ljubljenčkov in zalivanja travnika nam lahko izjemno olajšata vsakdanje delo. Vendar pametne naprave niso zanimive samo za domače uporabnike, temveč tudi za podjetja in zdravstvene storitve. Že danes poznamo pametne električne števce, ki dobavitelju električne energije v realnem času poročajo porabo posameznega odjemalca na daljavo [14]. Z razmahom interneta stvari bodo naprave razširile svoj radij povezljivosti od fizično bližnjih na svetovni splet, kar odpira neskončno aplikativnih možnosti. Hladilnik bi nam lahko v prihodnosti sam naročil hrano, pločnik pa nam pokazal pot do najbližjega prehoda za pešce [1].

S tem ko prepuščamo vedno več vsakdanjih opravkov pametnim napravam, postajamo tudi vedno bolj odvisni od njihovega pravilnega delovanja. V kolikor se nam naprava za avtomatsko zalivanje travnika pokvari in tega ne opazimo pravočasno, odpravljanje posledic lahko postane veliko breme. Podjetja že danes plačujejo ogromno denarja za nadzorne sisteme za odkrivanje napak in njihovo odpravljanje. Sčasoma bodo takšni sistemi postali upravičljivi tudi za domače uporabnike. Tako se je porodila ideja za izdelavo hišnega sistema za nadziranje pametnih naprav. Prvotna ideja je bila izdelati nadzorni sistem z uporabo profesionalnih orodij za nadziranje, kot na primer Microsoft System Center Operations Manager, vendar smo si po nasvetu mentorja prof. dr. Marka Bajca ogledali odprtokodni projekt za upravljanje in avtomatizacijo pametnih hiš openHAB. Vizija projekta openHAB se izjemno sklada z našo idejo, tako da smo se odločili realizirati nadzorni sistem, ki bi ga uporabniki lahko vključili v svoje obstoječe okolje in z njim nadzirali domače pametne naprave.

V nadaljevanju smo opisali, kaj je nadziranje in na kaj je potrebno biti pozoren pri zasnovi nadzornega sistema. Širši pojem nadziranja smo poskušali prenesti na pametne naprave in ugotoviti, kakšne so specifike njihovega nadziranja. Opisali smo vsa orodja, ki smo jih uporabili za izdelavo naloge, in navedli čemu služijo. Podrobno smo opisali postopek načrtovanja in implementacije nadzornega sistema, katere metode nadziranja smo izbrali in zakaj ter kako smo

sistem umestili v okolje openHAB. Za konec smo pripravili še en primer vezi za usmerjevalnike podjetja MikroTik, ki uporablja naš nadzorni sistem, da smo predstavili, kako naš sistem deluje v okolju openHAB.

Poglavje 2 Nadziranje

Nadziranje (*ang. monitoring)* je proces spremljanja prisotnosti in obsega sprememb stanj ali toka podatkov v sistemu. Nadziranje je prvi korak pri odkrivanju napak v informacijskem sistemu in njihovemu kasnejšemu odpravljanju. Ključni del vsakega nadzornega sistema je tudi sposobnost obveščanja uporabnikov o nezaželenih spremembah stanj v sistemu, bodisi prek grafičnega vmesnika, elektronskih sporočil ali katerekoli druge oblike sporočanja [2]. Bistvena naloga nadzornih sistemov je predstaviti podatke nabrane iz informacijskih sistemov in aplikaciji na način, ki je prijazen do končnih uporabnikov. S tem omogoča uporabnikom, da lažje sprejemajo kritične odločitve pri upravljanju informacijskega sistema [3].

Avtor J. Turnbull v svojem delu *The Art of Monitoring* deli nadziranje na tri nivoje glede na zrelost zasnove: ročno, odzivno oziroma reaktivno in proaktivno. Pri ročnem nadziranju uporabniki sami preverjajo performančne podatke in ročno poganjajo skripte, da bi opredelili, v kakšnem stanju je informacijski sistem. Odzivno nadziranje temelji na avtomatiziranem preverjanju stanja informacijskega sistema in obveščanju uporabnika o napakah. Proaktivni nadzorni sistemi pa poleg obveščanja uporabnikov poskušajo uporabnikom pomagati tudi pri odpravljanju napak. Poleg tega se ne omejijo na predstavitev surovih podatkov, ampak jih povezujejo v metrike, ki omogočajo uporabnikom razumevanje njihovega vpliva na celoten poslovni proces. Na osnovi trendov v informacijskem sistemu pa obveščajo uporabnika o morebitnih težavah še preden se te pojavijo oziroma prerastejo v kritične napake [3].

Nadzorni sistem je skupek programskih komponent, ki zbirajo podatke, jih hranijo za poznejšo obdelavo in interpretirajo njihov pomen [2]. Zasnova vsake komponente se lahko razlikuje odvisno od tega, kaj bomo nadzirali, v kakšno okolje bo umeščen nadzorni sistem in kdo ga bo uporabljal. Odgovori na vprašanja o zasnovi niso vedno črno-beli in potrebno je sprejemati kompromise. V nadaljevanju smo opisali štiri ključne točke, o katerih je potrebno razmisliti pri snovanju nadzornega sistema.

2.1 Zbiranje podatkov

Obstajata dve metodi za zbiranje podatkov iz nadzorovanih objektov. Prva, tako imenovana metoda prozorne škatle, temelji na ideji, da nadzorovani objekti samodejno sporočajo svoje

stanje in ostale zbrane podatke nadzornemu sistemu. Pri uporabi te metode je praviloma potrebno nadzorovane objekte nastaviti vnaprej, da se lahko potem uspešno povežejo z nadzornim sistemom. Metoda tudi predvideva, da sta se nadzorovani objekt in nadzorni sistem sposobna sporazumevati, bodisi prek protokola, kot na primer z uporabo pasti pri protokolu SNMP, bodisi s pomočjo agentov. Agenti so aplikacije, ki jih namestimo na nadzorovane objekte z namenom, da nabirajo in pošiljajo podatke nadzornemu sistemu. Druga metoda je metoda črne škatle. Ta pa predvideva, da nadzorni sistem periodično poizveduje pri nadzorovanih predmetih o njihovem stanju. Primeri slednje so http poizvedbe za preverjanje dosegljivosti spletnih strani ali preverjanje stanja gostitelja v omrežju s pomočjo ukaza *ping*.

Prva metoda ima številne prednosti pred drugo in bi se je morali po mnenju avtorja snovalci nadzornih sistemov posluževati vedno, ko je to mogoče. Glavna prednost metode prozorne škatle je, da se z njeno uporabo razbremeni centralni nadzorni sistem, saj je velik del preračunavanja prepuščen odjemalcem. To omogoča bistveno večjo obvladljivost rasti števila odjemalcev v sistemu. Poleg tega so zaradi zasnove zbiranja podatkov na osnovi posredovanja stanj s strani odjemalcev ti sposobni takoj obvestiti centralni sistem ob pojavu napake [3].

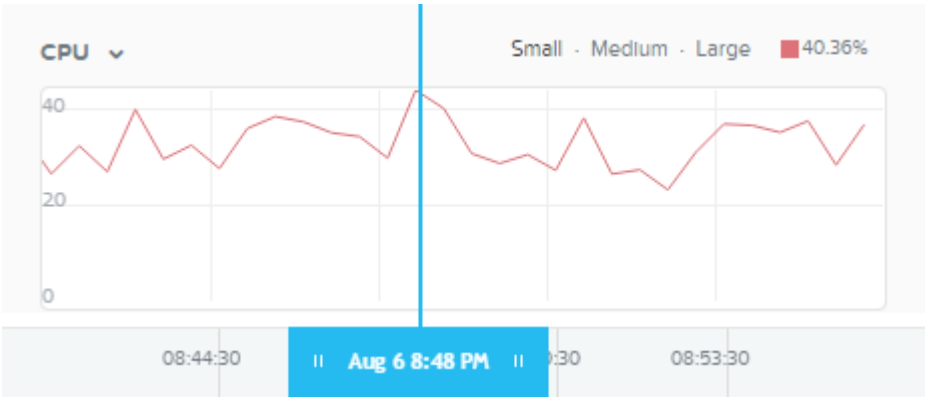
Pri načrtovanju zbiranja podatkov je potrebno biti pazljiv na možne učinke, ki jih proces nadziranja lahko ima na delovanje nadzorovanih naprav. Agenti, v kolikor jih uporabljamo, potrebujejo za delovanje določeno količino virov gostitelja. Poleg tega je treba biti pri načrtovanju še posebej pazljiv na tako imenovani učinek opazovalca. Ta se pojavi, ko nadzorni sistem prepogosto poizveduje po stanju objekta, bodisi zaradi neoptimalnih nastavitev bodisi zaradi uporabnikove želje po vedno svežih podatkih [3]. V takih primerih lahko poizvedovanje zavzame velik kos razpoložljivih virov in posledično botruje netočnim meritvam; v skrajnih primerih lahko privede celo do nedelovanja nadzorovanega sistema.

2.2 Hranjenje in obdelava podatkov

Nabrane podatke je potrebno smiselno združiti v metrike. Metrika je podatkovna struktura, ki je optimizirana za hranjenje in branje podatkov. Metrike tvorimo z združevanjem podatkov v smiselno celoto z namenom, da bi nam ti podatki predstavili jasnejše stanje sistema [2]. Tipičen primer metrike sestavljata dva podatka: vrednost nekega parametra in čas poizvedbe. Tako osnovano metriko lahko uporabimo, da začrtamo graf vrednosti nekega parametra, kot na primer zasedenost centralnega procesorja v odvisnosti od časa.

Pri shranjevanju podatkov je potrebno izbrati pravo vrsto metrike, le tako bomo lahko iz zbranih podatkov dobili najjasnejšo predstavo o stanju sistema. Poznamo več vrst metrik, opisali smo le najpogostejše.

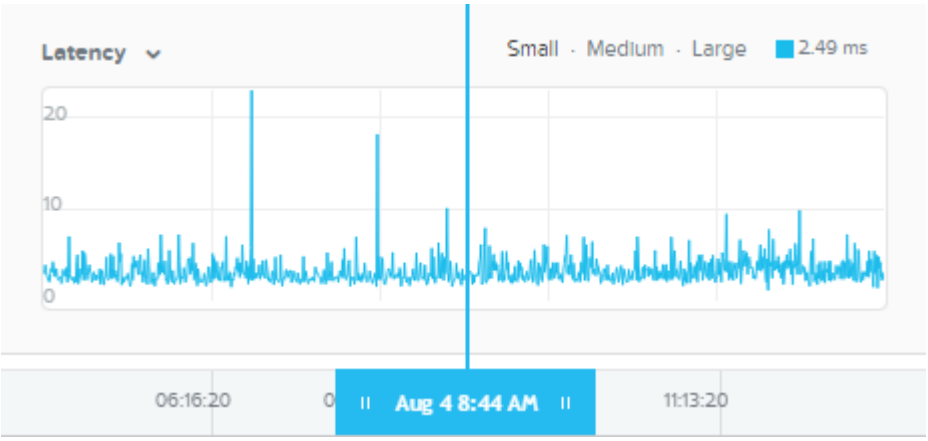
Merilec je najpogostejši tip metrike in predstavlja številčne vrednosti, ki se spreminjajo skozi čas. To metriko uporabljamo za predstavitev slike sistema oziroma vrednosti nekega kazalnika v sistemu, v določenem trenutku [3]; na primer poraba centralnega procesorja v sistemu, kot prikazuje slika 2.1.



Slika 2.1: Graf obremenjenosti centralnega procesorja v odvisnosti od časa.

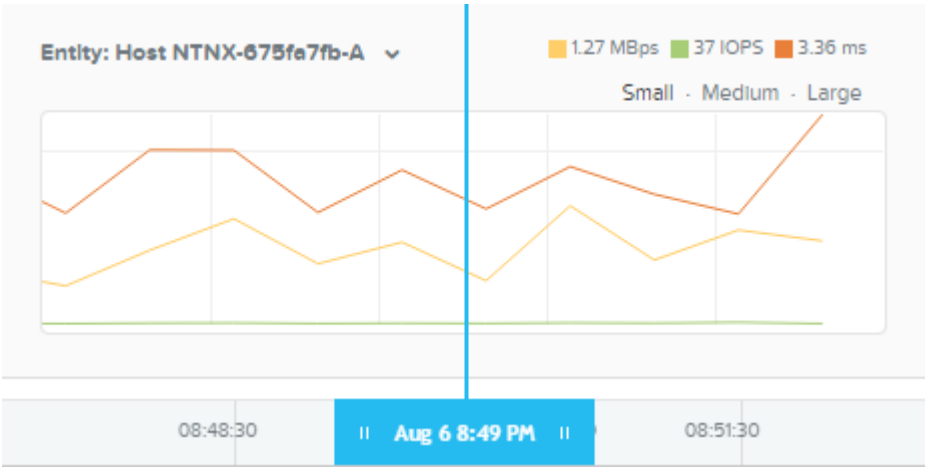
Drugi tip predstavljenih metrik so števci. Števci so števila, ki sčasoma samo naraščajo. Njihova vrednost se nikdar ne zmanjša, razen v primeru, ko jih ponastavimo na nič [3]. Primer takega števca je ura, ki meri čas delovanja neke naprave od trenutka zadnjega zagona, drugi primer je zaporedna številka transakcije v podatkovni bazi.

Tretja vrsta metrik so časovniki, ki nam povedo, koliko časa je neka stvar trajala [3]. Na primer z merjenjem časa med izstavitvijo ukaza podatkovnemu sistemu in njegovo potrditvijo lahko merimo latenco shranjevalnega sistema.



Slika 2.2: Graf latence shranjevalnega sistema.

Metrike lahko združujemo, kadar želimo prikazati njihovo medsebojno odvisnost ali kadar bi radi videli, kako več metrik vpliva na delovanje sistema. Primer takega pogleda je graf, ki prikazuje gibanje dostopnega časa v sistem v odvisnosti od števila prijavljenih uporabnikov. Drugi primer je graf vpliva podatkovnega pretoka in števila vhodno-izhodnih operacij na latenco, kot je prikazano na sliki 2.3.



Slika 2.3: Graf vpliva podatkovnega pretoka in števila vhodno-izhodnih operacij na latenco diskovne enote.

2.3 Interpretacija podatkov

Zbrane podatke je potrebno pravilno interpretirati, če želimo ugotoviti, ali je nadzorovani sistem izven območja pravilnega delovanja in ali je potrebno o tem obvestiti uporabnika. Pri interpretaciji metrik se poslužujemo različnih numeričnih metod za njihovo obdelavo. Izbira metode je odvisna od tipa metrike in konteksta njene uporabe. Tako lahko pri interpretaciji upoštevamo večje število meritev ali izračunamo povprečno vrednost neke metrike v določenem časovnem obdobju. Ta metoda je posebej priročna, če se želimo ogniti lažnim alarmom pri nadziranju merilcev, katerih vrednosti se zelo dinamično gibljejo. Nekatere metrike lažje primerjamo med seboj, če jih pretvorimo v odstotke. Možnosti je veliko, cilj pa je iz kopice podatkov izluščiti samo tiste, ki so za uporabnika zanimivi.

Za določitev kritičnosti dogodka sta pomembna dva dejavnika: vpliv, ki ga bo dogodek imel na sistem, in sposobnost sistema, da si sam opomore po dogodku. Pri pojavu dogodka je potrebno določiti njegov časovni, cenovni in materialni vpliv. Zelo pomemben je tudi podatek, ali je stanje po dogodku popravljivo ali ne [2]. Ko poznamo odgovore na vsa zgoraj našteta vprašanja, lahko določimo, ali gre za izjemni dogodek, ki zahteva proženje sistema za obveščanje, in nivo kritičnosti dogodka.

2.4 Obveščanje

Obvestila so primarni način za javljanje stanja nadzornega sistema uporabnikom. Uporabnike lahko obveščamo preko uporabniških vmesnikov, zvočnih efektov, elektronskih sporočil, SMS sporočil ali na kakršenkoli drug način. Pri izbiri oblike sporočanja je bistveno to, da sporočilo doseže želeni cilj – uporabnika.

Čeprav je obveščanje preprost koncept, lahko z napačno izbiro nastavitev postane popolnoma nefunkcionalno. Napačna izbira ponora obvestila, načina ter frekvence obveščanja lahko privedejo do stanja, ko je uporabnik prenasičen z obvestili in jim neha posvečati pozornost [3]. Enako velja za vsebinsko slabo oblikovana sporočila. Vsebina sporočila mora bita oblikovana tako, da poleg vzroka proženja vsebuje tudi druge kontekstne informacije, ki bodo omogočile uporabniku boljše razumevanje bistva problema. Primeri podatkov, ki sporočila kontekstno obogatijo, so:

- datum in ura nastalega dogodka,
- postopki in ukrepi, ki jih uporabnik lahko izvede za rešitev problema,
- graf metrike, ki je prožila dogodek,
- podatki o možnih vplivih dogodka na ostale dele sistema, na katere mora biti uporabnik še posebej pozoren.

Poglavje 3 Nadzor pametnih naprav

Pametne naprave so del našega vsakdana. Računalniki, telefoni, televizorji so samo najbolj razpoznavni primeri, ki vsi spadajo v skupino potrošniške elektronike. Med pametne naprave štejemo tudi pametne merilce porabe električne energije in vse druge omrežene naprave z določeno sposobnostjo komuniciranja in avtonomnega ravnanja [12][13]. Z uresničevanjem vizije interneta stvari bo vedno več naprav dobilo svoj internetni naslov.

Internet stvari je koncept, ki predpostavlja vsesplošno navzočnost stvari in objektov v okolju, ki so med seboj povezani prek spleta z drugimi stvarmi in objekti z namenom sodelovanja in doseganja skupnega cilja [16]. Objekti bodo vedno in povsod lahko dostopali do podatkov ostalih senzorjev in naprav ali oblačnih storitev ter na podlagi teh podatkov sprejemali bolj kontekstno smiselne odločitve.

Razmah naprav s procesno močjo in sposobnostmi komuniciranja odpira nov trg nadziranja pametnih naprav. Če se omejimo na nadziranje po metodi črne škatle, lahko nadziramo tako rekoč vsako napravo. Za vsako omreženo napravo lahko kadarkoli preverimo, če je dosegljiva, vendar ima tak pristop negativne posledice na življenjsko dobo baterij oddaljenih naprav ZigBee in Z-Wave. Za kakovostno nadziranje teh naprav se je nujno čim več zanašati na metode nadziranja prozorne škatle in čakati na posodobitve stanj, ki jih naprava sama oddaja.

Pri nadziranju pametnih naprav ne bi bilo smotrno, če bi se omejili le na preverjanje delovanja naprave. Pametne naprave oddajajo tudi kopico obvestil in opozoril o storitvah, ki jih nudijo. Tako je smiselno vključiti tudi ta sporočila med podatke, ki jih lahko ponudimo uporabniku. Pristop ponujanja vseh informacij v enem samem oknu oziroma na eni sami napravi *(ang. single pane of glass approach)* je nujno potreben, če želimo, da bo uporabnik sposoben slediti poplavi opozoril, ki jih bo deležen ob razmahu pametnih naprav.

Uresničitev vizije interneta stvari bo imela ogromne posledice tudi na področju nadziranja naprav. Koncept zbiranja podatkov o napravah bo dobil nove razsežnosti. S pomočjo podatkov, kot so lokacija naprave ter identiteta njej prostorsko bližnjih objektov in objektov, s katerimi je naprava komunicirala pred izrednim dogodkom, bomo lahko še natančneje sklepali o vzroku napake. S povečanjem količine podatkov pa bo postala pravilna interpretacija podatkov še toliko pomembnejša v izogib poplavi lažnih alarmov.

Domači uporabniki imajo že danes na voljo veliko pripomočkov za upravljanje in avtomatizacijo svojih domov. Te naprave, imenovane krmilniki ali bazne postaje, se povežejo z oddaljenimi napravami prek brezžičnih protokolov z majhno porabo energije in omogočajo njihovo upravljanje. Naprave so primerne tudi za nadziranje sistemov z uporabo oblačne storitve IFTTT, ki je okrajšava za *ang. if this, then that* [15]. Storitev omogoča izvajanje kompleksnejših zaporedij ukazov in omogoča tudi obveščanje prek mobilnih telefonov.

Zgoraj navedene rešitve za nadziranje pametnih domov pa niso ustrezne za vse uporabnike. En razlog, zakaj bi posegli po alternativni rešitvi, je, da želimo pridružiti v naše okolje tudi druge naprave, ki jih zgoraj navedeni krmilniki ne podpirajo. Drugi možen razlog je, da nočemo biti odvisni od oblačnih storitev oziroma na splošno od internetne povezljivosti, kajti če je oblačni ponudnik storitev nedosegljiv, potem ne bomo prejemali nikakršnih obvestil. Tretji razlog je varovanje osebnih podatkov in posledično problemi, ki lahko nastajajo pri deljenju vseh informacij, ki jih razni senzorji nabirajo v našem domu z oblačno storitvijo.

Rešitev teh problemov je uporaba krmilnika v navezi z okoljem openHAB, ki je opisan v naslednjem poglavju.

Poglavje 4 Orodja

V nadaljevanju je predstavljen nadzorni sistem, ki je osnovan na sledečih odprtokodnih rešitvah.

4.1 openHAB in Eclipse SmartHome

Naš sistem smo implementirali kot dodatek za okolje openHAB. openHAB je odprtokodni programski paket za avtomatizacijo pametnih domov, osnovan na okolju Java [4]. S svojo odprtostjo omogoča povezovanje naprav različnih proizvajalcev z uporabo poljubnih tehnologij.

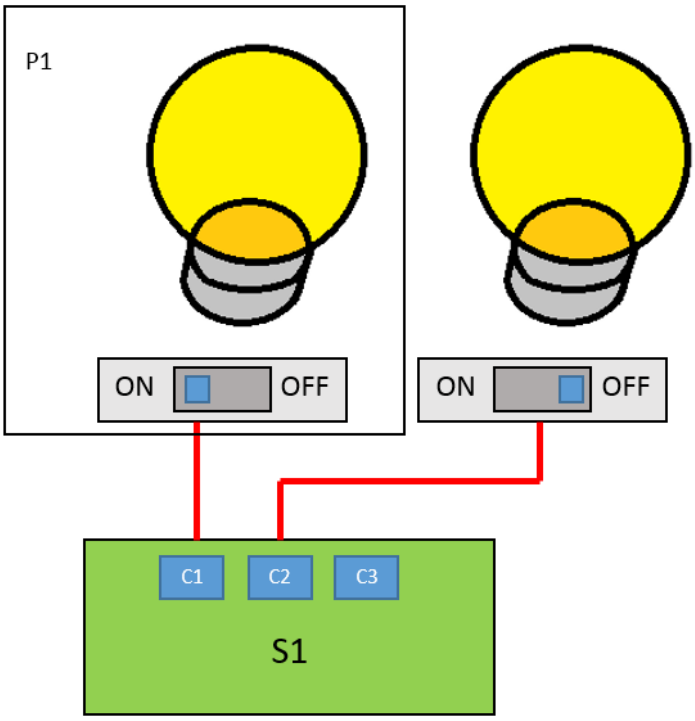
Za izdelavo naše komponente smo izbrali drugo verzijo paketa, ki je še v testni fazi in je dostopen zgolj kot predogledna različica. Pri prehodu na drugo verzijo se je projekt ločil na dva dela. Jedro ogrodja je zaživelo kot nov samostojni projekt pod okriljem ustanove Eclipse, imenovan Eclipse SmartHome, openHAB pa je sedaj na voljo kot razširitev tega produkta. Ločitev in prehod pod okrilje ustanove Eclipse omogoča lažje zbliževanje projekta s proizvajalci strojne opreme [5]. Pozitivne učinke uspešnega zbliževanja bodo občutili predvsem kupci naprav, saj bo to izjemno botrovalo k manjši fragmentaciji trga pametnih naprav z različnimi povezovalnimi standardi.

4.1.1 Osnovni koncepti

Eclipse SmartHome ločuje pogled na sistem povezanih naprav na dva nivoja: stvarni in funkcionalni. Stvarni pogled predstavlja objekte kot skupek fizičnih povezav in je namenjen dodajanju naprav v sistem ter njihovi konfiguraciji. V funkcionalnem smislu pa so sposobnosti objektov predstavljene kot predmeti, s katerimi lahko uporabniki interagirajo [7]. V ta namen Eclipse SmartHome vpeljuje naslednje pojme: stvar *(ang. thing),* (komunikacijski) kanal *(ang. channel),* povezava *(ang. link)* in predmet *(ang. item).*

Stvari so fizične enote povezane v sistem, ki prinašajo določeno število funkcionalnosti. Stvari ne predstavljajo nujno fizičnih naprav, ampak so to lahko tudi procesi ali storitve, do katerih želimo dostopati. Funkcionalnosti, ki jih stvari prinašajo, so izpostavljene prek komunikacijskih kanalov. Vsaka stvar ima lahko več kanalov, ki ponujajo različne funkcionalnosti. Predmeti

nudijo te funkcionalnosti v grafičnem vmesniku in imajo svoje stanje ter lahko sprejemajo ukaze. Predmete lahko povežemo s kanali naprav prek povezav in tako izpostavimo funkcionalnosti, ki jih naprava ponuja končnemu uporabniku.



Slika 4.1: Primer sistema povezanega v okolje Eclipse SmartHome. Preklopno stikalo (S1) predstavlja *stvar,* ki nudi tri komunikacijske *kanale* (C1, C2 in C3). Kanala C1 in C2 sta *povezana* z dvema *predmetoma* (P1 in P2). Predmeta uporabniku prinašata funkcionalnost prižiganja in ugašanja luči na napravi.

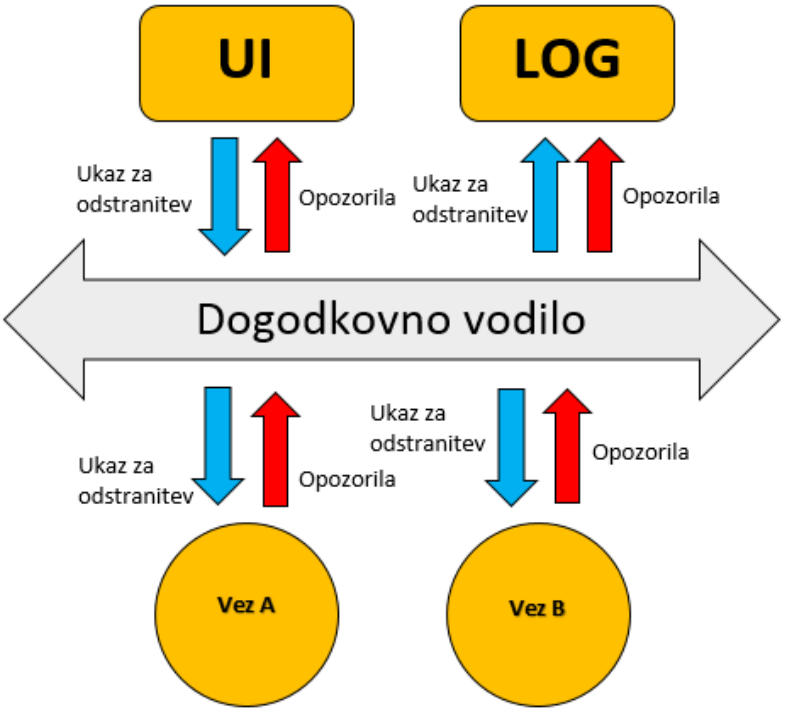
4.1.2 Arhitektura

Eclipse SmartHome je osnovan kot kopica svežnjev OSGi (okrajšava za *ang. Open Services Gateway Initiative)* [8]. Svežnji OSGi so Javanski paketi, ki vsebujejo programsko opremo in meta podatke v obliki datotek XML. Zaradi zasnove potrebuje Eclipse SmartHome za delovanje navidezni stroj Java.

Osrednja komponenta okolja Eclipse SmartHome je dogodkovno vodilo. Prek tega vodila poteka vsa komunikacija med različnimi komponentami. Komunikacija je osnovana na asinhronem pošiljanju in prejemanju dogodkov [10]. V osnovi obstajata dve vrsti dogodkov: ukazi, ki sprožajo akcije ali spremembe stanj, in obvestila o posodobitvi stanj.

Podpora novim zunanjim napravam je omogočena z uporabo vezi (*ang. binding).* Vezi so programski paketi za povezovanje naprav s sistemom openHAB. Služijo opisovanju stvari in

dodajajo funkcionalnosti, ki jih različni kanali v stvari nudijo. Namestitev novih vezi v sistem openHAB je zelo enostavna, saj zadostuje prenos paketa v za to določeno mapo. Sistem sam zazna nov paket in ga poveže s potrebnimi komponentami s pomočjo meta podatkov definiranih po standardu OSGi.



Slika 4.2: Poenostavljena arhitektura sistema openHAB.

Velika novost v drugi verziji paketa openHAB je podpora samodejnemu odkrivanju stvari in kanalov. Ta funkcionalnost je ključnega pomena za delovanje našega nadzornega sistema in hkrati doprinese k boljši uporabniški izkušnji.

4.1.3 Uporabniški vmesniki

Sistem openHAB ponuja uporabnikom na voljo tri grafične vmesnike. Prvi, *Paper UI*, je namenjen konfiguraciji stvari, kanalov in pripadajočih vezi. Drugi, *Basic UI*, je bolj okrnjen in je namenjen pregledovanju vsebine na mobilnih napravah. Tretji, *Classic UI*, pa je osrednji grafični vmesnik za končne uporabnike. Z vidika potreb našega projekta je največja pomanjkljivost predogledne različice ta, da osrednji grafični vmesnik trenutno še ne podpira prikazovanja dinamično odkritih naprav in predmetov.

Poleg zgornje omejitve pa ima openHAB trenutno še eno slabost – zelo slabo podporo grafični predstavitvi podatkov. Prikazovanje grafov je v osnovi podprto, vendar uporabnik nima možnosti nikakršnih nastavitev, kot so na primer izbira skale, označevanje osi in podobno. Tudi privzeta podatkovna baza *RRDTool,* ki jo grafi uporabljajo za prikazovanje podatkov, ni primerna za naše potrebe, saj je *round-robin* tip podatkovne baze [17]. To pomeni, da lahko hrani le omejeno količino podatkov. Sami smo zato za prikazovanje podatkov izbrali alternativno rešitev.

4.2 InfluxDB

InfluxDB spada v skupino podatkovnih baz TSDB, to so podatkovne baze optimizirane za hranjenje časovno zaporednih podatkov (*ang. time series data)* [18]. Časovno zaporedni podatki [6] so merilci – metrike, pri katerih je ena komponenta vedno čas. Merilci pa so najpogostejša oblika metrik, ki jih zbiramo pri nadziranju sistemov. Orodje openHAB ima privzeto vključeno vez za to podatkovno bazo, tako da nam konfiguracija ni povzročala težav.

4.3 Grafana

Za prikazovanje podatkov smo uporabili Grafano. Grafana je programski paket za ustvarjanje naprednih nadzornih plošč (*ang. dashboard)*, v katere lahko vstavljamo grafe in druge elemente. Poleg lepega izgleda jo krasi tudi zelo enostavna uporaba in hipna konfiguracija. Grafana privzeto podpira podatkovne baze InfluxDB kot podatkovni vir, tako da tudi v tem primeru nismo imeli problemov pri konfiguraciji.

Poglavje 5 Načrtovanje nadzornega sistema

Izdelali smo nadzorni sistem za nadziranje pametnih naprav v pametnih hišah. Sistem je prvotno namenjen domačim uporabnikom in njegova glavna naloga je obveščanje uporabnikov o izjemnih dogodkih in, kjer je to smiselno, prikazovanje zbranih podatkov iz naprav.

Naše poglavitno vodilo načrtovanja je bilo narediti čimbolj enostavno zasnovo. Sistem mora biti uporabniško prijazen in kar se da transparenten, saj bi kakršne koli kompleksne nastavitve odvrnile uporabnike od uporabe. Zagotoviti je potrebno nemoteno delovanje vezi naprav, ki bodo uporabljale naš nadzorni sistem, tudi če ta ni nameščen. Sistem mora služiti kot razširitev funkcionalnosti obstoječih vezi in njegova prisotnost ne sme biti predpogoj za njihovo namestitev. Taka zasnova omogoča, da se bodo izdelovalci naprav in razvijalci vezi dosti lažje odločili za vključevanje naše rešitve v svoje produkte.

5.1 Arhitekturna zasnova nadzornega sistema

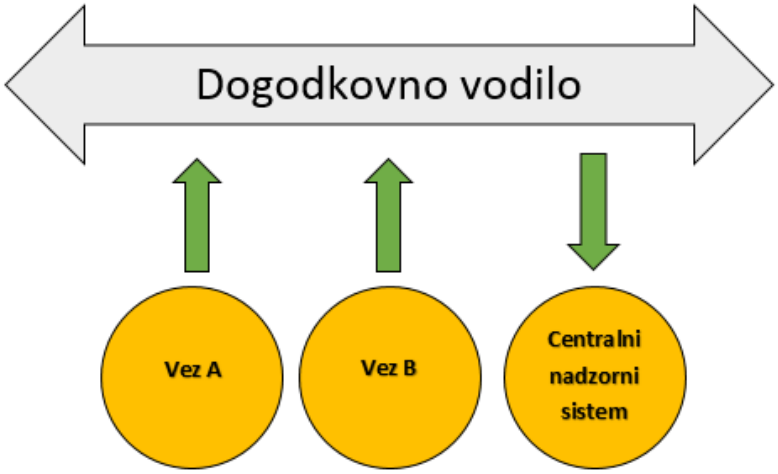
Z vidika našega nadzornega sistema so pametne naprave črne škatle. Ni mogoče podpreti vseh možnih protokolov, ki jih naprave uporabljajo za komunikacijo, in tudi nameščanje agentov na naprave ne pride v poštev. Brez povezave z napravo pa nam je onemogočeno kakršnokoli zbiranje podatkov o napravi. Tako bi v osnovi morali implementirati nadzorni sistem, ki deluje po metodi črne škatle. Toda zasnova okolja Eclipse SmartHome nam omogoča komuniciranje z napravami prek njihovih vezi, kar smo izkoristili in izdelali hibridni nadzorni sistem.

Nadzorni sistem je zasnovan tako, da vezi naprav delujejo kot agenti. Vezi so zadolžene za zbiranje podatkov o napravi in obveščanje centralnega nadzornega sistema o izjemnih dogodkih. Tak pristop zbiranja podatkov pa spada po definiciji med metode prozorne škatle. To seveda pomeni, da morajo pisci vezi sami poskrbeti, da zbrane podatke iz naprav aktivno preverjajo in po potrebi tvorijo opozorila. Prepustitev zasnove nadzornikov piscem vezi posamezne naprave je po našem mnenju edina smotrna rešitev, saj so pisci vezi edini, ki imajo vpogled v napravo in poznajo njene specifike. Z vključitvijo naše rešitve pa naprave pridobijo dostop do naprednega sistema obveščanja z avtomatskim tvorjenjem opozoril in prikazovanjem teh na enem mestu v uporabniškem vmesniku.

Ob pojavu izrednih dogodkov morajo vezi naprav periodično tvoriti opozorila, dokler napaka ni odpravljena. To reši problem osveževanja stanj kanalov, do katerega bi lahko prišlo, ker

programski vmesnik okolja Eclipse SmartHome ne dovoljuje branja trenutnih stanj kanalov, temveč le njihovo nastavljanje. Vnovično procesiranje že obstoječih opozoril tako opravljala funkcijo osveževanja predmetov, ki tvorijo opozorila.

Opozorila se prenašajo prek osrednjega dogodkovnega vodila okolja Eclipse SmartHome. Vsaka vez, ki želi pošiljati opozorila prek vodila, se mora najprej registrirati kot tvorec dogodkov. Centralni nadzorni sistem je registriran kot poslušalec na vodilu in prestreza vse dogodke tipov *SmartAlertEvent* in *ThingStatusInfoChangedEvent.*



Slika 5.1: Prikaz arhitekture sistema obveščanja prek dogodkovnega vodila.

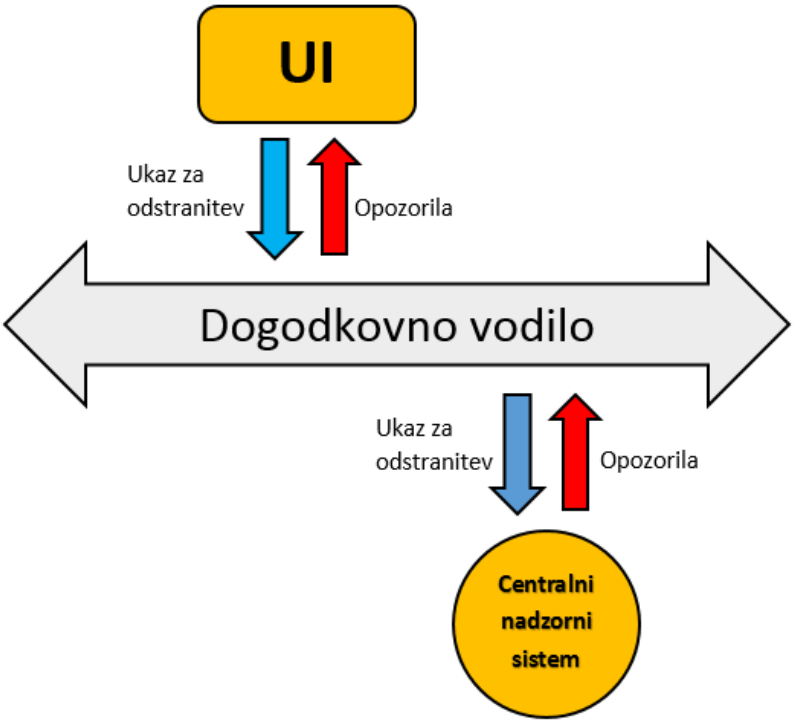
SmartAlertEvent smo namensko definirali za posredovanje podatkov o izrednih dogodkih. Sestavljajo ga naslednji podatki:

- stopnja kritičnosti dogodka – možne vrednosti so: informativni dogodek, opozorilni dogodek ali kritični dogodek;
- izvor dogodka – to je unikatni ključ stvari ali kanala, ki je tvorila dogodek;
- opis dogodka – načeloma je to ime metrike, ki je prestopila dovoljeni prag;
- datum in ura proženja dogodka;
- sporočilo dogodka, ki naj jasno definira vzrok za proženje dogodka;
- opis možnih ukrepov, ki jih uporabnik lahko izvede za rešitev problema.

Drugi dogodek, ki ga prestrezamo, se imenuje ThingStatusInfoChangedEvent. Ta dogodek se proži vsakič, ko se spremeni stanje katerekoli stvari v sistemu. Vsakič ko prestrežemo tak tip dogodka, preverimo, ali je naprava, ki je dogodek prožila, postala nedosegljiva in če je, preverimo vzrok. V kolikor je vzrok katerakoli napaka, o tem obvestimo uporabnika. Ta metoda nadziranja spada v kategorijo nadzorovanja črnih škatel. Na ta način je sistem sposoben vsaj delno nadzorovati stvari, katerih vezi niso prilagojene za sporočanje stanja naprave našemu sistemu.

5.2 Zasnova uporabniškega vmesnika

Naloga centralnega nadzornega sistema je, da ob prejemu opozorila o tem obvesti uporabnika. Opozorila so predstavljena kot skupek predmetov, ki prikazujejo vsebino dogodka SmartAlert. Ob razrešitvi problema lahko uporabnik s pritiskom na stikalo sprožil ukaz za odstranitev dogodka iz sistema. V kolikor problem na stvari ni razrešen in vez znova sproži dogodek, se bo ta spet pojavil med aktivnimi opozorili. Nadzorni sistem nikoli sam ne počisti obstoječih opozoril, tudi če tvorec dogodka preneha obveščati sistem o nastali napaki.



Slika 5.2: Komunikacija med nadzornim sistemom in grafičnim vmesnikom.

5.3 Predstavitev podatkov

Sistem je v osnovi reaktiven, saj je njegova osnovna funkcija zbiranje podatkov in obveščanje uporabnikov o napakah. Taka zasnova najverjetneje zadostuje večini uporabnikov. Z uporabo naprednejših analitičnih orodij pa je z našim sistemom možno doseči tudi proaktivno nadziranje.

Za naprednejše uporabnike našega nadzornega sistema, ki jim osnovna ponudba prikazovanja podatkov v programskem paketu openHAB ne zadošča, je predstavljena uporaba programskega paketa Grafana v kombinaciji s podatkovno bazo InfluxDB.

Poglavje 6 Implementacija nadzornega sistema

Pri implementaciji smo uporabili sledeča orodja:

- Git
- Eclipse IDE (verzija Neon)
- openHAB Development Kit
- Eclipse SmartHome Designer (verzija 0.8.0)
- Apache Maven (verzija 3.1)
- Oracle JDK 8

6.1 Centralni nadzorni sistem

Centralni nadzorni sistem je tudi sam implementiran kot vez, ki opisuje le eno stvar, in v osnovi nima priključenega nobenega komunikacijskega kanala, ampak se kanali dinamično dodajajo, ko v sistem prispejo dogodki SmartAlert.

Za opis in prikaz enega dogodka je potrebnih šest kanalov. Za lažje rokovanje s kanali smo uporabili metodo grupiranja kanalov. Identifikacijski ključ kanalom nastavimo na sledeč način: <ime skupine>#<zaporedna številka kanala>. Definirali smo naslednje skupine:

- *alertRoot* – te kanale bomo povezali s stikalom v uporabniškem vmesniku, da bomo omogočili uporabnikom odstranjevanje sporočil iz sistema;
- *alertSummary* – ti kanali bodo hranili podatek o izvoru dogodka;
- *alertSeverity* – te kanale bomo uporabili za hranjenje informacije o kritičnosti napak;
- *alertDatetime* – to bodo kanali koledarskega tipa in bodo hranili podatke o času nastanka napak;

- *alertMessage* – ti kanali bodo hranili sporočilo o napakah, ki ga bomo prebrali iz dogodkov;

- *alertResolution* – ti kanali bodo hranili opis možnih ukrepov iz dogodka.

Zaporedno številko kanala izračunamo spotoma, tako da vzamemo prvo prosto število v zaporedju od ena do N.

Sveženj nadzornega sistema je sestavljalen iz treh komponent:

- SmartAlertEventFactory

  Prva komponenta definira dogodek SmartAlert in metode za njegovo pošiljanje. Za definiranje novega tipa dogodka v okolju Eclipse SmartHome smo najprej vsebino dogodka opisali v pripadajočem razredu SmartAlert DTO *(ang. data transfer object)* in implementirali razred *SmartAlertEvent* kot razširitev razreda *AbstractEvent.*

  Nato smo spisali razred za tvorbo dogodkov SmartAlertEvent imenovan *SmartAlertEventFactory*, ki je razširitev razreda *AbstractEventFactory.* V tem razredu so definirane tri javne metode, ki jih pisci vezi lahko uporabijo pri pošiljanju naših dogodkov: *createInfoEvent*, *createWarningEvent* in *createCriticalEvent*.

  Na koncu smo spisali še dokument XML, ki po standardu OSGi registrira razred SmartAlertEventFactory kot storitev tipa *org.eclipse.smarthome.core.events.EventFactory.*

- SmartAlertEventSubscriber

  Druga komponenta skrbi za spremljanje dogodkov na dogodkovnem vodilu in njihovo obravnavo. Spisali smo razred *SmartAlertEventSubscriber* kot implementacijo vmesnika *EventSubscriber*, ki obravnava povratne klice iz dogodkovnega vodila okolja openHAB.

  Ob inicializaciji se omejimo na prestrezanje dogodkov tipa SmartAlertEvent in ThingStatusInfoChangedEvent. Dogodke tipa ThingStatusInfoChangedEvent pošljemo v metodo ThingStatusInfoMonitor, ki preveri, ali sprememba stanja zahteva tvorbo opozorila. Opozorila tvorimo v primeru prehoda stvari iz stanj *ONLINE* ali *INITIALIZING* v stanji *OFFLINE* in *UNINITIALIZED*. V kolikor nam nadzornik javi, da dogodek zahteva tvorbo opozorila, ga pretvorimo v tip SmarAlertEvent. Tako

dobljene dogodke skupaj z ostalimi dogodki tipa SmartAlertEvent dodamo v vrsto dogodkov, ki čakajo na obdelavo v razredu *SmartAlertHandler*.

Za končno delovanje smo definirali še dokument OSGi za predstavitev razreda SmartAlertEventSubscriber kot storitev org.eclipse.smarthome.core.events.EventSubscriber.

- SmartAlertHandler

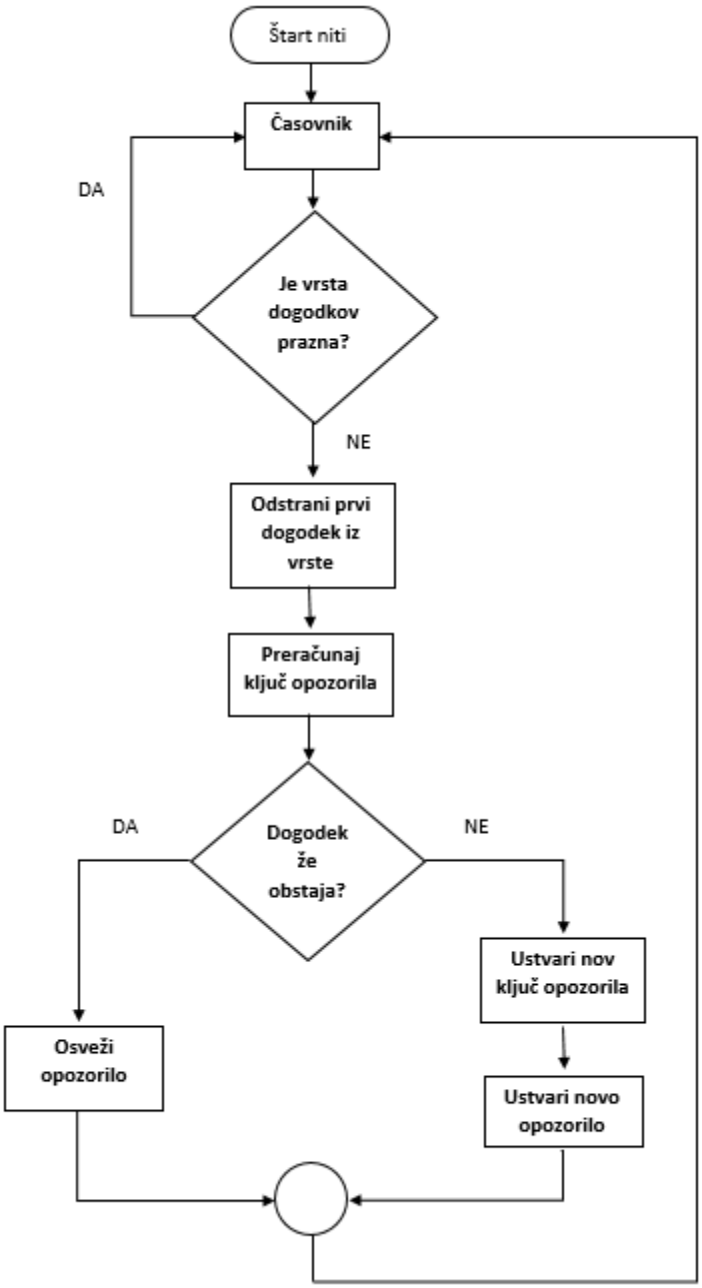
  Naloga tretje komponente je razčlenitev dogodkov in obveščanje uporabnikov. Komponenta se imenuje SmartAlertHandler in je razširitev razreda *BaseThingHandler*. Jedro komponente je nit, ki se požene ob inicializaciji objekta in periodično preverja stanje vrste dospelih dogodkov. Dogodki se posredujejo metodi *parseAlert* in ta najprej primerja nivo kritičnosti dogodka z minimalnim nivojem za tvorbo opozoril, ki ga je uporabnik določil v nastavitvah vezi. Če nivo kritičnosti dosega ali presega nastavljeni nivo, potem nadaljujemo z obdelavo dogodka. Kreira se instanca zgoščene tabele, v kateri so vsi kanali tipa alertSummary. Ključ posameznega vnosa je sestavljen iz izvornega dogodka.

  V kolikor zgoščena tabela že vsebuje vnos za trenutni dogodek, se sproži ukaz za osvežitev. Pri osvežitvi je potrebno najprej ugotoviti zaporedno številko kanala, ki jo preračunamo iz identifikacijskega ključa kanala, in nato ponastavimo stanje vsem šestim kanalom z dano zaporedno številko.

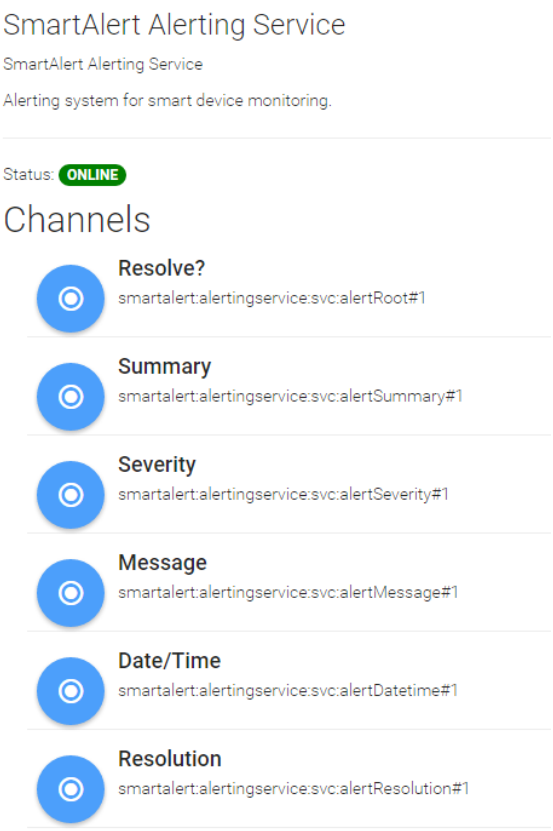
  V primeru, da gre za nov dogodek, pa ustvarimo novo opozorilo. Postopek izdelave opozorila je podoben osvežitvi, le da za zaporedno številko vzamemo prvo prosto število iz množice vseh obstoječih kanalov in to številko uporabimo pri izdelavi skupine potrebnih kanalov.

  Poleg ravnanja z dogodki ima komponenta še eno nalogo – brisanje opozoril iz sistema. V ta namen smo definirali metodo za ravnanje z ukazi iz uporabniškega vmesnika. V našem primeru je to samo ukaz za odstranitev določenega opozorila iz spiska, ki je naslovljen na kanale v skupini alertRoot.

  Ker okolje Eclipse SmartHome ne ponuja metode za brisanje kanalov iz sistema, je pri brisanju posameznega kanala potrebno najprej sestaviti seznam vseh obstoječih kanalov in iz njega odstraniti neželene kanale. Dobljen spisek kanalov nato nastavimo kot privzet spisek kanalov za našo stvar.



Slika 6.1: Diagram prikazuje ravnanje komponente SmartAlertHandler s prispelimi dogodki.



Slika 6.2: Pogled na stvar, ki jo definira naša vez, razkriva skupino kanalov, ki opisujejo eno proženo opozorilo.

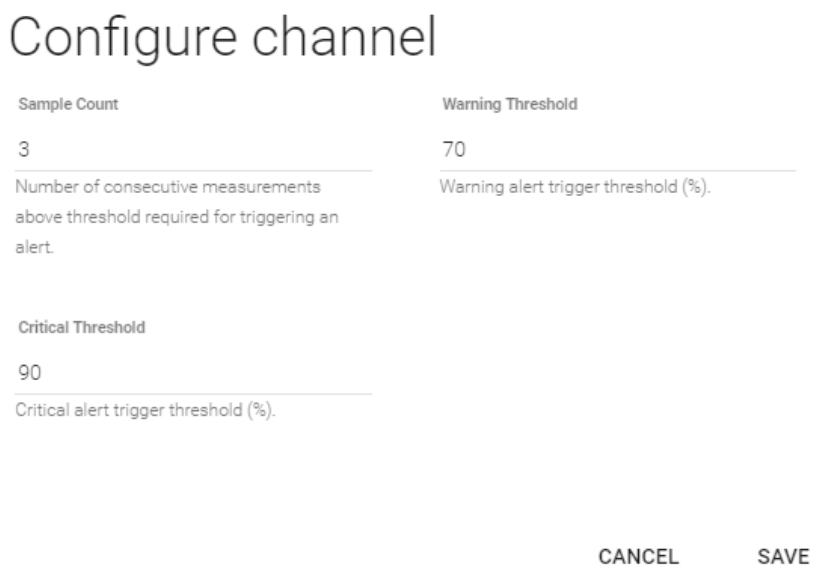
6.2 Vez za usmerjevalnike MikroTik

Za prikaz delovanja našega nadzornega sistema smo napisali vez za usmerjevalnike podjetja MikroTik, ki so osnovani na operacijskem sistemu RouterOS. Z uporabo skripte *create_openhab_binding_skeleton.cmd,* ki je del razvojnega okolja za openHAB, smo izdelali osnovno programsko strukturo vezi, ki nam je služila za nadaljnji razvoj.

Vez definira eno stvar z različnimi kanali. Kanali so namenjeni zbiranju podatkov o operacijskem sistemu, centralni procesni enoti, spominu in zunanjim širokopasovnim povezavam usmerjevalnika. V sistem obveščanja so vključeni trije kanali:

- cpu#load – kanal za merjenje zasedenosti centralnega procesorja;
- memory#usage – kanal za merjenje porabe spomina z naključnim dostopom usmerjevalnika;
- pppoe#status – kanal, ki nadzira stanje širokopasovne povezave.

Vsak od naštetih kanalov ima možnost nastavitve, pri katerih vrednostih naj se prožijo opozorilna oziroma kritična obvestila. Kanaloma za merjenje zasedenosti procesorja in porabe spomina je mogoče nastaviti tudi število meritev, ki morajo presegati prag, preden se tvori izredni dogodek.



Slika 6.3: Primer nastavitve opozorilnih pragov za kanal v Paper UI.

Poleg zgoraj naštetih kanalov velja izpostaviti še naslednje tri kanale, za katere so zbrani podatki prikazovani v grafih:

- system#uptime – števec sekund od zadnjega zagona;
- pppoe#rxRate – merilec porabe pasovne širine prejetega prometa;
- pppoe#txRate – merilec porabe pasovne širine oddanega prometa.

Usmerjevalniki MikroTik so osnovani na operacijskem sistemu RouterOS [9]. To je namensko za usmerjevalnike razvit operacijski sistem, ki je osnovan na jedru Linux. Ima lasten nabor ukazov, ki jih uporabniki lahko vnašajo prek ukazne vrstice. Komunikacija z usmerjevalnikom poteka prek protokola SSH. Za odjemalca SSH na sistemih Microsoft Windows smo uporabili program Plink, ki je del programskega paketa Putty, medtem ko je na sistemih Linux odjemalec SSH privzeto prisoten. Uporabnik mora pri konfiguraciji vezi v Paper UI vnesti potrebne podatke za uspešno povezavo na napravo in prijavo vanjo. Po opravljeni konfiguraciji bo nit v vezi periodično preverjala stanje naprave in osveževala vrednosti kanalov. Če na katerem od

nadzorovanih kanalov prebrane vrednosti prestopijo nastavljen prag, vez tvori dogodek tipa SmartAlert in ga pošlje na dogodkovno vodilo.

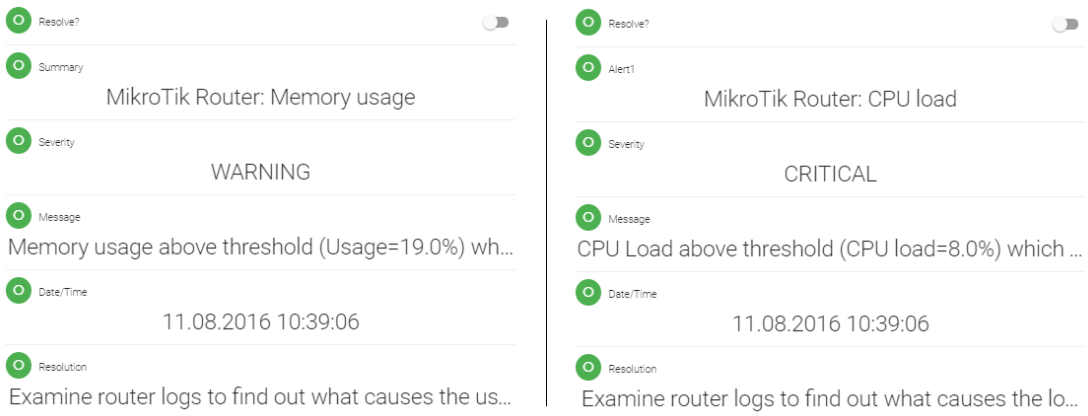
Do podatkovnega vodila dostopamo tako, da se s pomočjo deklaracije OSGi vežemo na storitev *org.eclipse.smarthome.core.events.EventPublisher*. Ogrodje OSGi nam nato v času izvajanja programa samo priskrbi instanco objekta tipa *EventPublisher,* preko katerega lahko pošiljamo dogodke.

Za kreiranje dogodkov SmartAlert moramo uvoziti paket Java iz projekta SmartAlertEventFactory. Pri načrtovanju nadzornega sistema smo si zastavili zahtevo, da naša rešitev ne bo predstavljala pogoja za namestitev in delovanje vezi naprav. To zahtevo smo izpolnili tako, da smo v datoteko OSGi manifest dodali vrstico *org.openhab.binding.smartalert.eventfactory;resolution:=optional*. Ogrodje OSGi bo tako navedene pakete vključilo le, če bodo na voljo v času izvajanja programa.

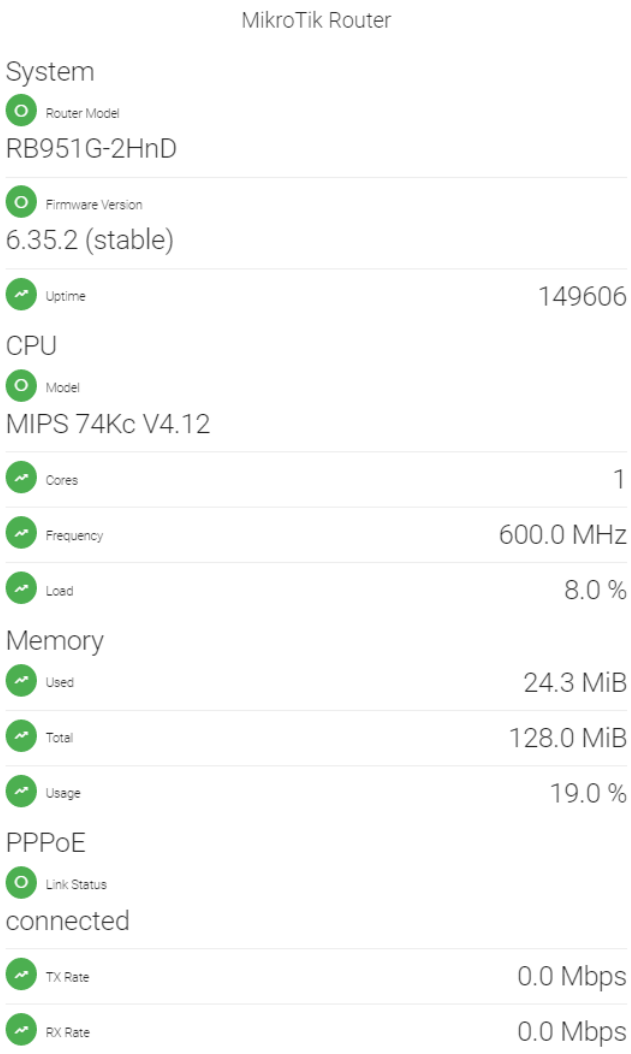
6.3 Obveščanje uporabnikov

Uporabnike bomo obveščali prek grafičnih vmesnikov. Vmesnik Paper UI avtomatsko prikazuje vsa novo nastala obvestila v oknu *control*. Vmesnik je v osnovi namenjen konfiguraciji sistema openHAB in zelo nazorno prikazuje stanje kanalov raznih naprav.

Zaradi dejstva, da je Paper UI trenutno edini vmesnik okolja openHAB, ki podpira dinamično dodajanje in odstranjevanje objektov, je zelo primeren tudi za prikazovanje obvestil. Opozorila so prikazana v vrsti, tako kot so definirani kanali. Uporabnik ima možnost počistiti opozorilo iz sistema s pritiskom na stikalom *Resolve.*



Slika 6.4: Primer prikazovanja dveh opozoril v vmesniku Paper UI.



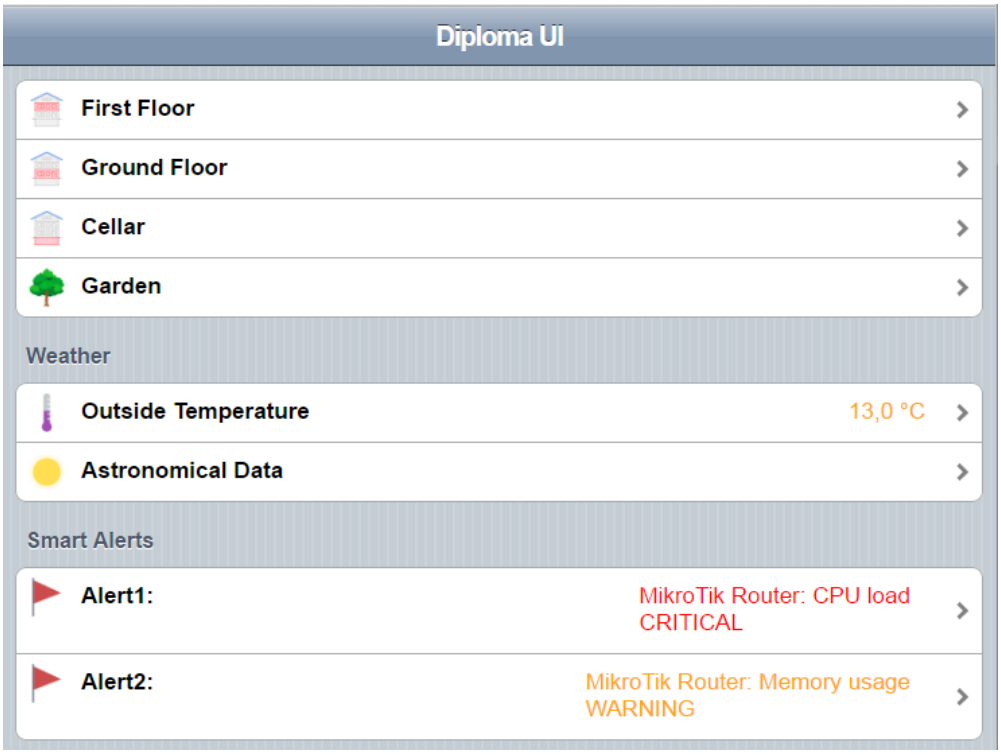
Slika 6.5: Stanje kanalov usmerjevalnika MikroTik.

Drugi grafični vmesnik, Classic UI, pa žal še ne podpira dinamičnega upravljanja z objekti. Vse predmete, ki jih želimo prikazati, je potrebno vnaprej navesti v tekstovni datoteki. Tak način rokovanja z objekti seveda ni primeren za prikazovanje naključno generiranih opozoril.

Problem smo zaobšli tako, da smo definirali deset skupin statičnih predmetov, ki nam služijo za prikazovanje opozoril. Namesto da predmete dinamično dodajamo in odstranjujemo, jim programsko nastavljamo vidljivost. Druga pomanjkljivost uporabniškega vmesnika je, da ne omogoča dinamičnega nastavljanja oznak. Tako smo bili primorani predmete označiti s statičnimi nizi znakov od Alert1 do Alert10, kar negativno vpliva na končni izgled vmesnika.

Opozorila so prikazana v ločenem okvirju na dnu domače strani. Organizirana so v hierarhično strukturo. Vrhnji predmet prikazuje opis opozorila in njen nivo kritičnosti. Tekst je obarvan v

odvisnosti od nivoja kritičnosti opozorila. Ob pritisku na vrstico z opozorilom se odpre podstran s podrobnostmi.

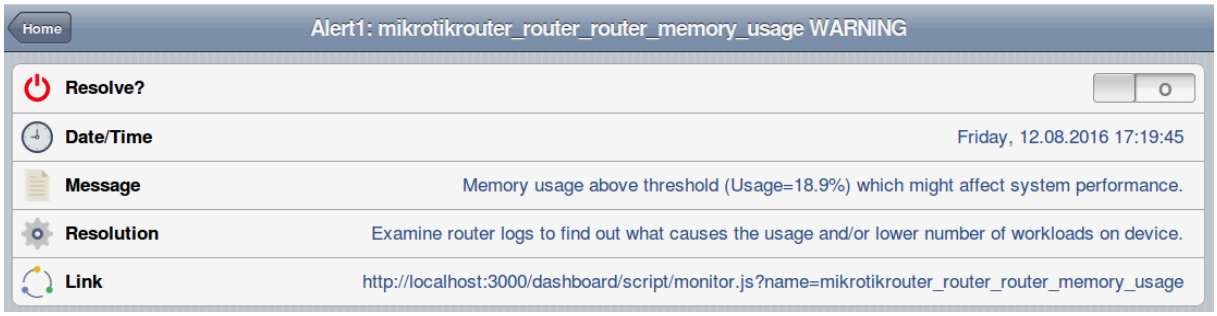


Slika 6.6: Demonstracijsko domačo stran okolja openHAB smo nadgradili z okvirjem za prikaz aktivnih opozoril.

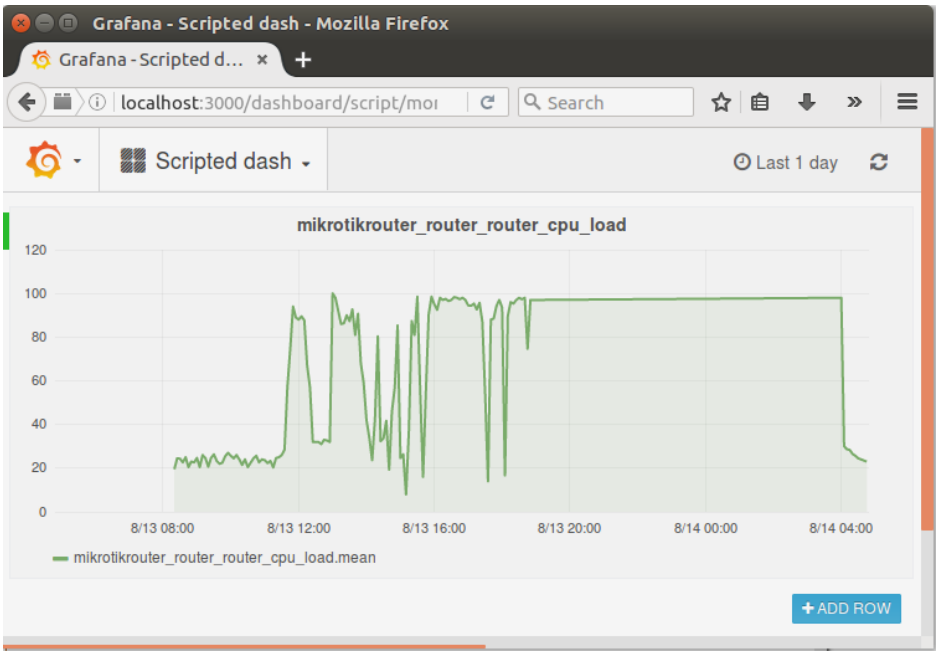
Stran s podrobnostmi opozorila prikazuje datum in uro proženja opozorila, sporočilo in napotke za odpravo opozorila, ki se prenesejo prek dogodka SmartAlert. Uporabnik ima na voljo stikalo za odstranitev opozorila iz sistema, vendar se zaradi zasnove vmesnika le-to ne počisti iz domače strani, čeprav opozorilo v sistemu ni več prisotno.

Zadnja vrstica prikazuje naslov povezave, kjer lahko uporabnik preveri graf s stanjem gibanja metrike pred proženjem dogodka. Za sestavo naslova smo napisali novo pravilo *(ang. rule),* ki je skripta v programskem jeziku *Java Xtend*. Naslov povezave pa kaže na skripto spisano v jeziku *JavaScript*, ki gostuje na strežniku Grafana in služi dinamičnemu kreiranju nadzornih plošč s tehniko *Grafana scripted dashboards.* Skripta za vhodni parameter sprejme ime metrike, ki je enako unikatnemu ključu dotičnega kanala.

Znova smo morali zaradi zastarelosti zasnove grafičnega vmesnika sprejeti kompromis in prikazati samo naslov povezave, namesto dejanske slike grafa.



Slika 6.7: Podrobnosti obvestila o izjemnem dogodku v uporabniškem vmesniku Classic UI.



Slika 6.8: Primer avtomatsko generiranega grafa ob pojavu napake.

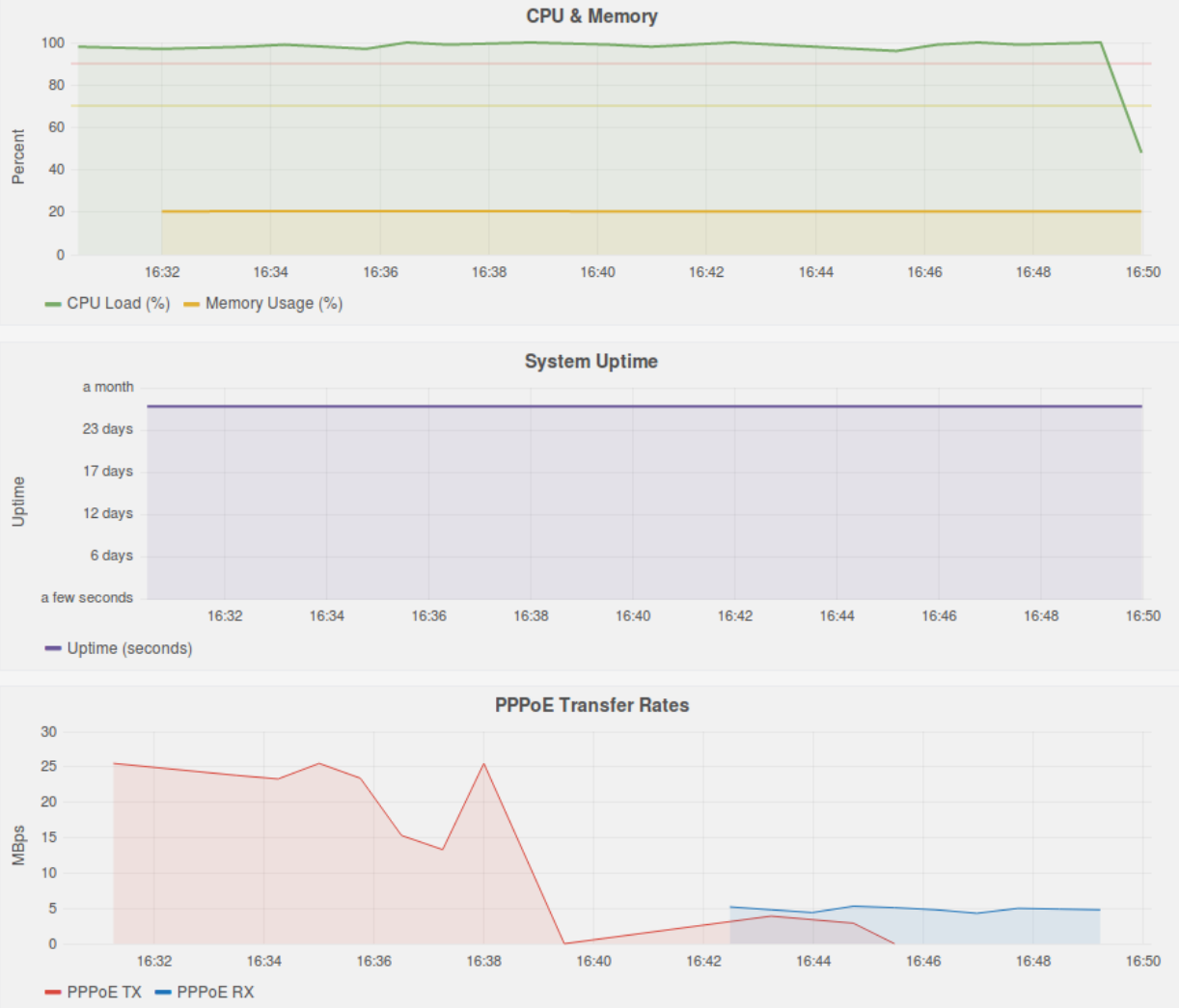
Modularna zasnova okolja openHAB omogoča tudi razširitev sistema obveščanja z uporabo vezi za pošiljanje elektronske pošte, SMS sporočil ali govornih ukazov napravi Amazon Echo [11]. Na ta način si lahko uporabniki nastavijo nadzorni sistem po svojih željah.

6.4 Napredno prikazovanje podatkov

Grafana je v kombinaciji s podatkovno bazo InfluxDB izvrstno orodje za prikazovanje med nadziranjem zbranih metrik. Uporabnikom omogoča ustvarjanje nadzornih plošč po meri in avtomatično prikazovanje grafov, kot smo prikazali v prejšnjem poglavju.

Dejstvo, da je vez za podatkovno bazo InfluxDB privzeto prisotna v okolju openHAB, nam je izjemno olajšalo delo pri postavitvi in konfiguraciji tega programskega paketa. Potrebno je bilo ustvariti le uporabnika in bazo, kamor se shranjujejo podatki. Vez za InfluxDB smo nastavili tako, da stanje objektov v sistemu shrani ob vsaki spremembi oziroma najmanj enkrat na minuto.

Tudi konfiguracija Grafane je potekala izjemno gladko. Orodje privzeto podpira povezave s podatkovno bazo InfluxDB, tako da posebnih nastavitev praktično ni.



Slika 6.9: Z orodjem Grafana narejena nadzorna plošča, ki prikazuje podatke nabrane z vezjo za usmerjevalnike MikroTik v okolju openHAB.

Poglavje 7 Sklepne ugotovitve

V diplomski nalogi smo raziskali tematiko nadziranja in pridobljeno znanje izkoristili za izdelavo sistema za nadziranje pametnih naprav v domačem okolju. Vodilo razvoja je bilo narediti čimbolj enostavno in uporabniku prijazno rešitev. Za izdelavo naloge smo uporabili kopico različnih orodij. Sistem temelji na odprtokodnem ogrodju openHAB, ki je sistem za upravljanje in avtomatizacijo pametnih domov. Z osnovanjem našega sistema na okolju openHAB smo rešili problem komuniciranja s pametnimi napravami različnih proizvajalcev. To nam je omogočilo razviti hibridni nadzorni sistem, ki opravlja nadziranje tako po metodi črne kot prozorne škatle. Sistem deluje tako, da prestreza sporočila o napakah na osrednjem dogodkovnem vodilu okolja openHAB in vsak prejeti izredni dogodek prevede v kontekstno izpopolnjeno sporočilo o napaki. Za boljši pregled nad domačim okoljem se ta sporočila prikažejo uporabniku v grafičnem vmesniku okolja openHAB na enem mestu.

Z vključitvijo našega sistema v njihovo okolje bodo uporabniki lahko nadzirali svoje domače pametne naprave in bodo avtomatsko obveščeni vsakič, ko bo kakšna njihova naprava prenehala delovati. Centralni sistem za obveščanje, ki smo ga razvili, jim bo omogočil, da prejmejo vsa opozorila o napakah na enem mestu in tako izboljšajo pregled nad delovanjem njihovega celotnega doma. Z vključitvijo podpore našega nadzornega sistema v vezi teh naprav pa bodo lahko koristili še napredni sistem obveščanja, ki ga naša rešitev ponuja. Tako bodo lahko prejemali opozorila o napakah s kontekstno dopolnjenimi informacijami o vzroku napake, možnih ukrepih in grafom, ki prikazuje gibanje problematičnih metrik pred napako.

Zaradi dejstva, da je ogrodje openHAB dostopno le v predogledni različici, smo imeli nemalo težav v fazi implementacije. Sploh pa nam je povzročala težave zastarela zasnova osrednjega grafičnega vmesnika Classic UI. Na forumih smo zasledili informacijo, da ta vmesnik čaka celovita prenova, ki se je izjemno veselimo. Ko bo nov vmesnik na voljo, bi bilo potrebno posodobiti izgled sporočil in odpraviti obstoječe pomanjkljivosti. Potreben bi bil prehod na sistem dinamično generiranih predmetov, ki bi odprl ogromno možnosti za izboljšave. Uredili bi lahko trenutno nedelujoče brisanje sporočil iz osrednjega vmesnika. Nadaljnje delo pa bi bilo osredotočeno na iskanje možnosti vključitve grafa metrike v samo sporočilo, saj obstoječa implementacija prikazuje samo naslov strani, kjer uporabnik lahko vidi dotični graf, kar tej funkcionalnosti znižuje uporabno vrednost. Potrebno bi bilo tudi nadgraditi skripto za

avtomatsko generiranje grafov Grafana, da bi bili ti bolj informativni. Grafom bi bilo potrebno dodati naslov, zaznamke in po možnosti prikazati nastavljene pragove za prožena opozorila. Preden pa lahko sistem ponudimo v uporabo končnim uporabnikom, bi bilo potrebno spisati še dokumentacijo. Brez dobre dokumentacije je rešitev le deloma uporabna, saj uporabniki ne bodo znali vzpostaviti naprednejših funkcionalnosti nadzornega sistema.

Ko bodo te pomanjkljivosti odpravljene, bomo lahko našo rešitev ponudili odprti skupnosti okolja openHAB in tako zapolnili eno vrzel, ki jo okolje openHAB ima še odprto – pomanjkanje centralnega nadzornega sistema. Zanimivo bo videti, ali v skupnosti obstaja interes za rabo naše rešitve.

Literatura

[1] If Sidewalks Could Talk: Crowdsourcing for a Walkable City *IEEE* [Online] Dosegljivo: http://iot.ieee.org/iot-scenarios.html?prp=oc-028552c8-abf5-4602-83be-98bb850806bd [Dostopano 5. 8. 2016]

[2] S. Ligus (2013). *Effective Monitoring and Alerting* [Online]. O'Reilly Media. Strani 1, 2, 5, 6, 48, 49. Dosegljivo: Amazon, https://www.amazon.com/Effective-Monitoring-Alerting-Web-Operations-ebook/dp/B00ADVPNNK/#nav-subnav [Dostopano 6. 8. 2016]

[3] J. Turnbull (2016). The Art of Monitoring [Online]. Version 0.12. Dosegljivo: Amazon, https://www.amazon.com/Art-Monitoring-James-Turnbull-ebook/dp/B01GU387MS/#nav-subnav [Dostopano 6. 8. 2016]

[4] openHAB. *Domača stran projekta* [Online]. Dosegljivo: http://www.openhab.org/ [Dostopano 9. 8. 2016]

[5] K. Kreuzer. openHAB 2.0 and Eclipse SmartHome. *Blog* [Online]. Desegljivo: http://kaikreuzer.blogspot.si/2014/06/openhab-20-and-eclipse-smarthome.html [Dostopano 9. 8. 2016]

[6] Time series database. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Time_series_database [Dostopano: 9. 8. 2016]

[7] Concepts. *Dokumentacija orodja openHAB* [Online]. Dosegljivo: http://docs.openhab.org/concepts/index.html [Dostopano: 10. 8. 2016]

[8] Architecture. *Opis orodja openHAB* [Online]. Dosegljivo: http://www.openhab.org/features/architecture.html [Dostopano: 10. 8. 2016]

[9] MikroTik RouterOS. *Katalog MikroTik RouterOS opracijskega sistema* [Online]. Dosegljivo: http://download2.mikrotik.com/what_is_routeros.pdf [Dostopano: 11. 8. 2016]

[10] Events. *Eclipse SmartHome documentation* [Online]. Dosegljivo: http://www.eclipse.org/smarthome/documentation/concepts/events.html [Dostopano: 11. 8. 2016]

[11] Amazon Echo. *Blog o uporabi naprave Amazon Echo v sistemu openHAB* [Online]. Dosegljivo: http://tinsley.io/2015/06/control-your-items-using-the-amazon-echo-openhab/ [Dostopano 13. 8. 2016]

[12] Smart Device. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Smart_device [Dostopano: 13. 8. 2016]

[13] Smart Meter. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Smart_meter [Dostopano: 13. 8. 2016]

[14] Smart Grid. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Smart_grid [Dostopano: 13. 8. 2016]

[15] IFTTT. *Domača stran storitve* [Online]. Dosegljivo: https://ifttt.com/ [Dostopano: 13. 8. 2016]

[16] O. Vermesan, P. Friess, P. Guillemin, H. Sundmaeker, M. Eisenhauer, K. Moessner, F. Le Gall, P. Cousin (2013). Internet of Things Strategic Research and Innovation Agenda. *V Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems* [Online]. River Publishers. Dosegljivo: http://www.internet-of-things-research.eu/pdf/Converging_Technologies_for_Smart_Environments_and_Integrated_Ecosystems_IERC_Book_Open_Access_2013.pdf [Dostopano: 13. 8. 2016]

[17] RRDTool. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/RRDtool [Dostopano: 13. 8. 2016]

[18] Time series database. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Time_series_database [Dostopano: 13. 8. 2016]