SBNZ
Smart Home
Pametna Kuća

Članovi Tima

- Luka Šerbedžija SW-32-2018
- Miloš Panić SW-19-2018
- Luka Bjelica SW-21-2018

Motivacija

IoT tehnologije postaju sve popularnije i počinju da se primenjuju u svim aspektima ljudskog života. Oblast korišćenja na koju je stavljen fokus jesu pametne kuće, odnosno kućna automatizacija, čiji je cilj da korisnikove svakodnevne aktivnosti učine lakšim, bezbednijim i efikasnijim. Sistem se oslanja na skup senzora koji prikupljaju podatke i/ili mogu biti upravljani od strane korisnika ili sistema. Kako bi korisnici mogli da izvuku maksimum iz uređaja koji su instalirani, potrebno je da imaju dobar softver koji će im omogućiti efikasan rad sa njima.

Pregled problema

Kućna automatizacija omogućava korisniku pristup kontrolnim uređajima u kući sa bilo kog uređaja bilo gde u svetu. Preciznije opisuje domove u kojima je skoro sve - svetla, uređaji, električne utičnice, sistemi za grejanje i hlađenje - uključeno na mrežu kojom se može daljinski upravljati.

Pored upravljanja uređajima, sistem treba da omogući nadgledanje podataka koje senzori šalju, kako bi korisnik imao uvid u njih. S obzirom da se radi o velikoj količini podataka koji se periodično šalju (*time series data*), sistem treba da vrši agregaciju podataka u pozadini. Na taj način, korisnik bi mogao da ima uvid u tok podataka kroz vreme, što je njemu najbitnije.

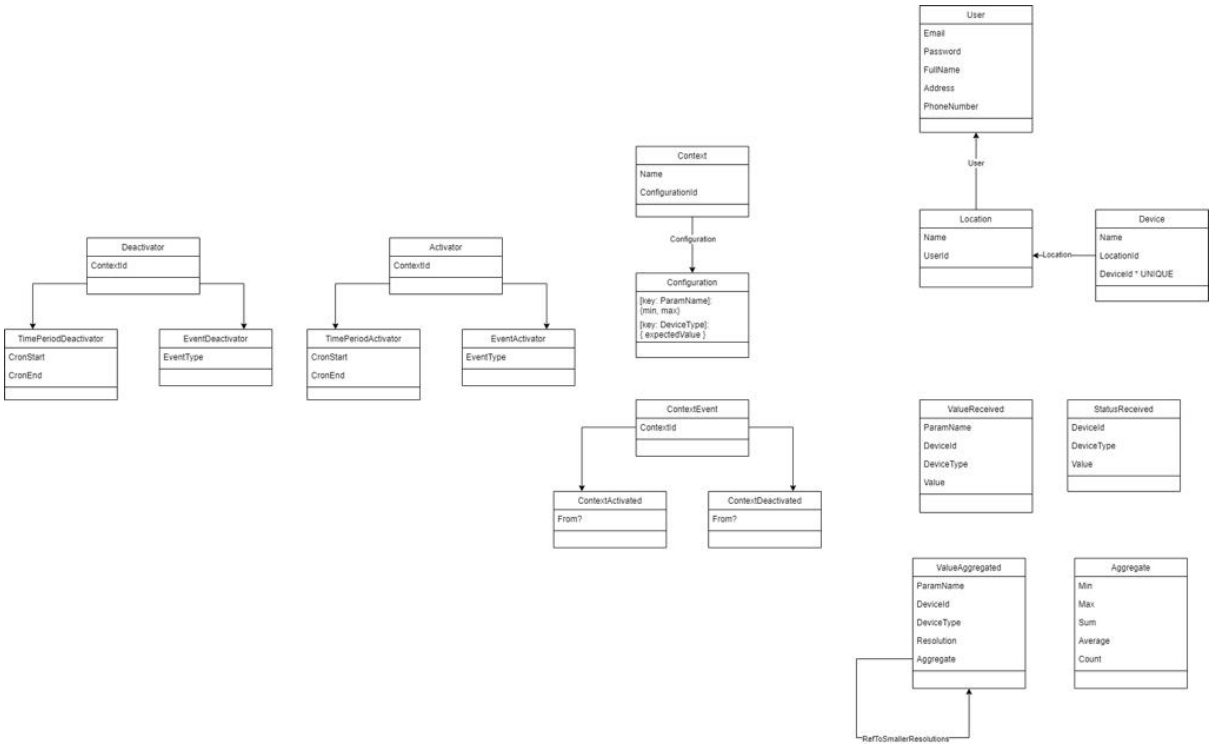
Iz perspektive kućne bezbednosti, ovo takođe uključuje postojanje alarmnog sistema, koji je povezan sa svim vratima, prozorima, bravama, detektorima dima, nadzornim kamerama i svim drugim senzorima koji su povezani sa njim.

Činjenica da postoji veliki broj slučajeva primene sistema, kao i veliki broj senzora koji se mogu koristiti, poželjno je da sistem bude konfigurabilan kako bi mogao pokriti svaki od njih.

Postoji veliki broj rešenja ove prirode, poput “Thinger.io” i “ThingsBoard”, ali su oni uglavom ograničeni na rad sa malim skupom senzora koji su proizvedeni od strane određenog proizvođača. Takvi sistemi uglavnom podrazumevaju i ograničenje na korišćenje njihovog softverskog rešenja, bez mogućnosti integracije sa drugima.

Cilj je da rešenje koje razvijemo bude višenamensko i proširivo na rad sa proizvoljnim skupom senzora, konfigurabilno kako bi podržalo rad u različitim kontekstima primene i bazirano na pravilima, što povećava stepen automatizacije i doprinosi lakoći daljeg razvoja sistema.

Metodologija Rada



Postoje dve vrste ulaza u sistem, real-time podaci koji stižu sa senzora i podaci koje korisnik unosi. Od podataka sa senzora očekivani su sledeći parametri:

- temperatura
- vlažnost vazduha
- udeo ugljen dioksida i kiseonika u vazduhu
- statusi uređaja
- prisutnost u prostoru
- jačina zvuka
- koordinate
- i dodatne specifične informacije sa određenih uređaja.

Od uređaja kojima se može upravljati, i koji predstavljaju izlaze is sistema, očekivani su:

- sijalice
- prekidači
- utičnice
- emiteri zvuka alarma
- bojleri
- frižideri
- rerne
- klima uređaji
- radijatori
- lock sistemi (pametne brave)
- veš mašine
- mašine za sudove
- mikrotalasne
- pametni prozori

- kamere
- pametna ogrlica za ljubimce

Kako bi sistem bio upotrebljiv u različitim uslovima, potrebno je omogućiti konfiguraciju pragova vrednosti za svaki parametar za koji se definišu pravila. To se rešava uvođenjem koncepta Konteksta, koji predstavlja određenu konfiguraciju sistema. Kontekst može biti ručno ili automatski uključen, odnosno isključen. Kontekst se može aktivirati određenim pravilom, odnosno reakcijom na određeni događaj, kao i definisanim vremenskim periodom važenja. Sledeći parametari, odnosno ulazi u sistem, se mogu konfigurisati za određeni kontekst:

- min max potrošnja struje
- min max temperatura
- min max vlažnost
- min max CO2
- min max O
- status uređaja (određenog) On/Off
- min max intenziteti pokreta (definicija prisutnosti/odsustva)
- min max jačina zvuka

Postojaće predefinisani konteksti:

- Away From Home, u daljem tekstu AFH
- At Home, u daljem tekstu AH
- Eco
- Winter
- Spring
- Summer
- Autumn
- Daily
- Nightly
- Custom

Baza znanja

Legenda:

- K - Kontekst,
- C - Konfiguracija, [K] jedna definicija za specifican Kontekst - K
- Konfiguracija(<parametar>) - vrednost parametra iz konfiguracije konteksta
- naziv() u when delu - drools query funkcija
- naziv() u then delu - akcija
- dispatch(eventName) - ispaljivanje eventa

Pravila:

Pomoćna pravila:

when EventOccured
then insert EventStarted

on every 10 sec
when EventStarted
then change Refresh of EventStarted

when EventStarted($eventId) not EventOccured(id == $eventId)
then delete EventStarted

priority small
when ValueReceivedCopy
then delete ValueReceivedCopy, change State

priority small
when StatusReceived
then delete StatusReceived, change State

Pravila vezana za događaje:

when (K = AFH or K = Night) and outdoorCamDetectedPerson()
then informUser() and startRecording() and dispatch(PotentialRoberyMovement)

when K = AFH and C[K](max_int_pokret) < int_pokret
then informUser() and startRecording() and dispatch(PotentialRoberyMovement)

when K = AFH and C[K](max_zvuk) < zvuk
then informUser() and startRecording() and dispatch(PotentialRoberySound)

when PotentialRoberyMovement and PotentialRoberySound
then activateAlarmSystem() and informUserOfAlarm()

when K = AFH and DoorOrWindowsStatusChanged
then activateAlarmSystem() and informUserOfAlarm()

when any(K) and C[K](min_temp) > temp
then informUser() and dispatch (TemperatureTooLow)

when K = Summer and K = Nightly and TemperatureTooHigh
then openWindows()

when K = Summer and K = Daily and TemperatureTooHigh
then turnOnAC()

when K = Summer and K = Daily and TemperatureTooLow and isACOn()
then turnOffAC()

when K = Summer and K = Daily and TemperatureTooLow and !isACOn()
then openWindows()

when K = Summer and isACOn() and areWindowsOpen()
then closeWindows()

when K = Winter and TemperatureTooHigh
then turnOffHeating()

when K = Winter and TemperatureTooLow
then turnOnHeating()

when K = Winter and isHeatingOn() and areWindowsOpen()
then closeWindows()

when K = AH and C[K](int_prisutnost_kupatilo) < prisutnost_kupatilo and NotInKupatilo
then turnOffBoiler() and turnLightsOn() and dispatch(InKupatilo)

when K = AH and C[K](int_prisutnost_kupatilo) > prisutnost_kupatilo and InKupatilo
then turnOnBoiler() and turnLightsOff() and dispatch(NotInKupatilo)

when K = Nightly and K = Eco and WashingMachineOff
then turnOnWashingMachine() and dispatch(WashingMachineOn)

when C[K](int_prisutnost_ulazna_vrata) < prisutnost_ulazna_vrata
then informUser() and dispatch(MovementOnTheFrontDoor)

when C[K](int_prisutnost_ulazna_vrata) > prisutnost_ulazna_vrata
then dispatch(NothingOnTheFrontDoor)

when K = Nightly and !areFrontDoorLightsOn() and MovementOnTheFrontDoor
then turnFrontDoorLightsOn()

when K = Nightly and areFrontDoorLightsOn() and MovementOnTheFrontDoor
then turnFrontDoorLightsOff()

when K = AFH and doorbellRings()
then informUser()

when K = AH and doorbellRings()
then informUser() and turnVideoCallOn() and dispatch(VideoCallOn)

when NothingOnTheFrontDoor and VideoCallOn
then turnVideoCallOff() and dispatch(VideoCallOff)

when K != PetWalk and petBraceletOutOfRange(C[K](home_radius))
then informUser() and dispatch(PotentialPetMissing)

when K != PetWalk and PotentialPetMissing and petBraceletOutOfRange(C[K](neighborhood_radius))

then informUser() and dispatch(PetMissing)

when K == AFH and PetMissing
then alarmUser() and startSendingRealTimeLocation()

when PetMissing and petBraceletInRange(C[K](neighborhood_radius))
then informUser() and dispatch(PotentialPetCameBack)

when PotentialPetCameBack and petBraceletInRange(C[K](home_radius))
then informUser() and dispatch(PetCameBack)

Pored pravila koja su upravo definisana, postoji i pregršt pravila koja služe za određena čišćenja radne memorije. Potreba za time se javila zbog toga što su sesije dugotrajne i svi podaci se upisuju u istu sesiju ograničenu na lokaciju.

Primer Forward Chaining-a:

when any(K) and C[K](max_temp) < temp
then informUser() and dispatch(TemperatureTooHigh)

when any(K) and C[K](max_CO2) < CO2
then informUser() and dispatch(CO2TooHigh)

when any(K) and C[K](min_humid) > humid
then informUser() and dispatch(HumidTooLow)

==========

when TemperatureTooHigh and CO2TooHigh and HumidTooLow
then dispatch(PotentialFire)

==========

when PotentialFire and isWindowOpen()
then closeWindow()

when PotentialFire and isACOn()
then turnOffAC()

when PotentialFire
then activateSprinklers() and dispatch(SprinklersActivated)

==========

when SprinklersActivated and isAnyElectricDeviceOn()
then turnOffAllElectricDevices()

when SprinklersActivated over: window(3 min)

then notifyFireStation()

Agregacija podataka (Primer CEP-a):

Podaci sa senzora će biti predstavljeni u obliku događaja. U realnim IoT sistemima, ovakvih događaja ima previše jer senzori mogu slati vrednosti pa čak i na nivou sekundi ili milisekundi. Tolika količina podataka nije pogodna za dalju statičku analizu, i uglavnom nam ne trebaju u ovom obliku.

Događaje ćemo agregirati uz pomoć sum (suma), average (srednja vrednost), min (minimalna vrednost), max (maksimalna vrednost), count (ukupan broj pojava) funkcija na nivou 5, 15, 30, 60 minuta, 1, 2, 4, 8, 24h. Kako bi postupak bio najoptimalniji, rezultati petominutnih agregacija (događaji) će se koristiti za 15 minutne agregacije (događaj) i tako ulančano do 24h.

Veoma je bitno da su vremena agregacija zaokružena, npr. da je 5 minuta na 15:00, 15:05, 15:10 a ne 15:08:42, 15:13:42, 15:18:42. Osim toga, bitno je da vreme podataka ne zavisi od vremena kada je sistem pokrenut pa tako, 15 min agregacija se neće izvršiti prvi put kad se agregiraju tri petominutna podatka što bi značilo da se 15 min agregacija može izvršiti u 15:20 ako je sistem pokrenut u 15:05 nego će se izvršiti u vreme deljivo sa 15 i tako dalje za svaku veću rezoluciju.

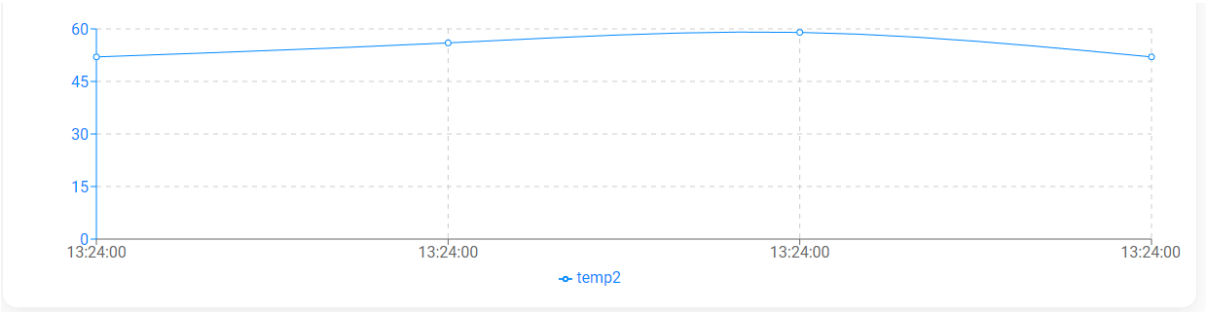
Agregirani podaci će biti organizovani u obliku strukture stabla. Čvor će predstavljati 24h agregacija dok će listove predstavljati 5min agregacija. Ovakvo stablo se može koristiti za backward chaining.

Izveštaji (Primer Backward chaining-a):

Izveštaji su sastavni deo svakog sistema. Na osnovu njih se može ispratiti trend u podacima, statički analizirati i doneti određen zaključak. Izveštaji služe za prikaz agregiranih podataka, koji se dodatno mogu filtrirati. Kako korisnici ne bi morali svaki put zadavati parametre filtriranja izveštaja, uveden je koncept predefinisanih izveštaja. Korisnik predefiniše parametra izveštaja, koje potom može videti na posebnoj stranici.

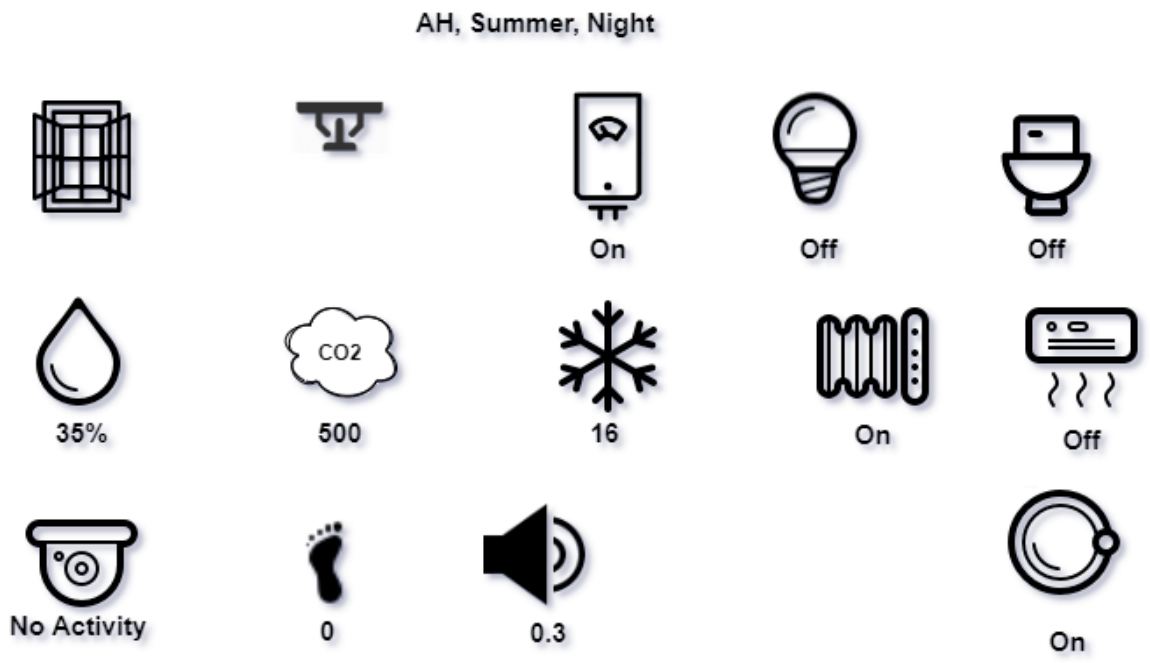
U sistemu postoje tri vrste izveštaja:

1. Normal izveštaj - filtrira podatke zadavanjem rezolucije, datuma i ostalih parametara.
2. AtSomeTime izveštaj - filtrira podatke zadavanjem rezolucija, datuma i ostalih parametara. Za razliku od Normal izveštaja, kod kojih podatak zadate rezolucije mora ispunjavati filtere, dovoljno je da bilo koji deo perioda zadate rezolucije ispunjava filtere. To znači da će podatak 8h rezolucije ispunjavati filtere ako neki od njegovih 4h perioda ispunjavaju uslove.
3. MaxPeriod izveštaj - kod kojeg se, za razliku od prethodna dva izveštaja, zadaju samo period i filteri. Ovaj izveštaj će pokušati da nađe najduže periode koji ispunjavaju zadate parametre. Ovaj izveštaj bi se mogao koristiti za traženje optimalnih perioda.



Realtime podaci:

Blagovremena reakcija na pristigle događaje je od krucijalne važnosti u IoT sistemima i baš iz tog razloga korisnici moraju imati brz uvid u trenutno aktuelne podatke. S tim ciljem, potrebno je implementirati dijagram, koji će prikazivati poslednje podatke za svaki uređaj. Takođe, podatke bi trebalo prikazati i u obliku realtime izveštaja kako bi se mogao ispratiti neki trend. Uz sve to, sva obaveštenja korisnika trebaju ići u realtime.



Domenski specifični jezik:

Najveći problem saradnje programera i eksperta u polju je taj da ekspert izrazito retko zna programske jezike. To spečava samog eksperta da direktno implementira pravila, nego pravila implementira programer uz konsultacije sa ekspertom gde može doći do problema u komunikaciji i samim tim pogrešne implementacije. Način da se ovaj problem premosti je DSL, koji enkapsulira kompleksnost i time znatno olakšava proces pisanja, kao i korišćenje *drools* sintakse.

U nastavku će biti naveden deo sintakse DSL-a:

- Da li je (ili nije) uređaj u određenom stanju

    - *{device} has (doesn't have) status {status}*
    - *{device} doesn't have status {status}*
- Da li se (ili nije) desio događaju u prethodnih n vremenskih jedinica
    - *in the last {time[ms]} {event} has occurred*
    - *in the last {time[ms]} {event} has not occurred*
- Da li se događaj (ili dva događaja) desio (nije desio) (uz opciono definisanje parametra)
    - *def $param={event} has occurred*
    - *{event} has occurred*
    - *{event} has not occurred*
    - *def $param={event1} or {event2} have occurred*
    - *{event1} or {event2} have occurred*
- Da li događaj traje duže od n minuta
    - *{event} lasts more than {time}min*
- Korišćenje predefinisanih *query*
    - *are|is|has|aren't|isn't|hasn't {query}*
- Rad sa kontekstima
    - *def $contextParam=getAnyContextIfActive()*
    - *def $contextParam=getContextIfActive({context})*
    - *def $contextParam= getContextIfActive({context1} or {context2})*
    - *ensureContextNotActive({context})*
- Dobavljanje parametra iz konfiguracije konteksta
    - *def $param=C*[*$contextParam*][(*max*|*min*|*expected*)_(*eg.* temperature*, humidity*)]
- Poređenje vrednosti dobijenih sa senzora
  curr_(eg. temperature, humidity)
    - - manje od {param}
    - - manje ili jednako sa {param}
    - - vece od {param}
    - - vece ili jednako sa {param}
    - - jednako sa {param}
    - - razlicito od {param}
- Obaveštavanje korisnika
  informUser()
  with "{param}": "{value}"
  ===>

Proširivost pravila (Template):

Jedan od problema postojećih rešenja je njihova proširivost. Određeni sistemi imaju mogućnost proširivanja logike uz pomoć koda određenog programskog jezika, najčešće su to JavaScript, Python ili C# ali takvo rešenje nije user friendly. Naš sistem omogućava korisniku da proširi postojeći code base, koristeći naš DSL jezik koji se piše nalik engleskom.

Korisnici su u mogućnosti da definišu templejte, koje mogu da aktiviraju kreiranjem instanci templejta sa konkretnim parametrima. Kreirane instance templejta su automatski primenjene bez potrebe da se sistem restartuje.

Templejt se definiše zadavanjem *when* i *then* dela pravila, importi se ne navode i dinamički se generišu u zavisnosti od korišćenih tipova (podržani tipovi su svi tipovi iz *java util*-a, *java time* kao i naše tipovi koji su pravljeni u okviru projekta).