OULUN AMMATTIKORKEAKOULU

Mikko Mäkelä

LANGATON VALVONTAJÄRJESTELMÄ

LANGATON VALVONTAJÄRJESTELMÄ

Mikko Mäkelä
Opinnäytetyö
Kevät 2017
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Tekijä: Mikko Mäkelä
Opinnäytetyön nimi: Langaton valvontajärjestelmä
Työn ohjaajat: Lauri Pirttiaho ja Jussi Heikkinen
Työn valmistumislukukausi ja -vuosi: Kevät 2017
Sivumäärä: 51 + 2 liitettä

Tämä opinnäytetyö tehtiin Bittium Wireless Oy:n toimeksiannosta ja sen tavoitteena oli suunnitella konsepti ja rakentaa esittelyversio langattomia tekniikoita hyödyntävästä valvontajärjestelmästä, joka täyttää sille asetetut vaatimukset. Teknologiavaatimuksia olivat mm. VoIP ja Bluetooth Low Energy. Tehtävään sisältyi sekä laite- että ohjelmistopuolen suunnittelu ja toteutus.

Työ eteni oppimisprojektina, jonka etenemistä seurattiin kahden viikon välein projektin toteutusvaiheessa yhdessä toimeksiantajan kanssa. Luonteensa vuoksi työ muutti useita kertoja suuntaansa projektin edetessä, kun kokemusta aihepiirin kanssa kertyi lisää. Työssä käytetyt ohjelmointikielet olivat Python ja C++.

Tuloksena syntyi Genuino 101- ja Raspberry Pi 3 -alustojen pohjalle rakennettu järjestelmä, joka toteuttaa sille asetetut vaatimukset muutamaa poikkeusta lukuun ottamatta. Konsepti ja esittelyversio esiteltiin demotilaisuudessa toimeksiantajan tiloissa projektin päätyttyä. Lopputuloksena syntynyt konsepti antoi hyvät edellytykset jatkaa järjestelmän kehitystä, mutta lopulliseksi tuotteeksi sillä on vielä pitkä matka.

Asiasanat: kaukovalvonta, langaton tiedonsiirto, Bluetooth Low Energy, Asterisk, SIP

ABSTRACT

Oulu University of Applied Sciences
Information technology

Author: Mikko Mäkelä
Title of thesis: Wireless surveillance system
Supervisors: Lauri Pirttiaho and Jussi Heikkinen
Term and year when the thesis was submitted: Spring 2017
Pages: 51 + 2 appendices

The subject of this thesis was to design a concept and build a demo of a wireless surveillance system, which fulfils the demands set by the assigner, Bittium Wireless Ltd. Among others, Bluetooth Low Energy and VoIP were set as technologies to be used in the scope of this project.

The project was a learning process, which led to multiple design overhauls during the execution phase as more experience had been accumulated. The main programming languages were C++ and Python. The project's progress was examined bi-weekly in collaboration with the assigner.

The end product was a Genuino 101 and Raspberry Pi 3 based system, with nearly all of the demands met as requested. The proof of concept was showcased to the assigner to close the project.

The concept proved to be applicable with multiple development paths to follow going forward, but to further develop it into a final product would require a lot of work still.

Keywords: Wireless surveillance, Bluetooth Low Energy, Asterisk, SIP

ALKUSANAT

Haluan kiittää Bittium Wireless Oy:n Kajaanin yksikköä erittäin antoisasta ja innostavasta opinnäytetyön aiheesta. Erityiskiitokset Jussi Heikkiselle, Toni Tammelanderille ja Teemu Lappalaiselle ajastanne, ohjauksestanne ja palautteestanne projektin aikana.

Kiitos kuuluu myös työn toiselle ohjaajalle, Oulun ammattikorkeakoulun yliopettaja Lauri Pirttiaholle, joka antoi matkan varrella arvokasta palautetta ja ohjeistusta työn tueksi. Kiitos myös kielentarkistuksesta vastanneelle lehtori Tuula Hopeavuorelle erinomaisista neuvoista tekstin viimeistelyssä.

Suurin kiitos kuuluu kuitenkin vaimolleni Miialle, joka lähes neljä vuotta työn ohella jatkuneiden opintojeni ajan on jaksanut antaa tukea ja kannustaa minua kohti valmistumista.

Kajaanissa 9.4.2017

Mikko Mäkelä

SISÄLLYS

SANASTO

AAC	Advanced Audio Coding, äänidatan häviölliseen tiedonpakkaukseen kehitetty standardi.
AGI	Asterisk Gateway Interface, komentosarjakieli omien laajennusten ohjelmointiin Asterisk-puhelinvaihdeohjelmistolle.
API	Application Programming Interface, ohjelmistorajapinta. Käytetään kahden ohjelmistokomponentin väliseen kommunikointiin.
Asterisk	Asterisk PBX (Private Branch Exchange), Digiumin kehittämä, avoimeen lähdekoodiin perustuva erittäin monipuolinen puhelinvaihdeohjelmisto.
bcrypt	Blowfish-pohjainen salausfunktio tiivisteen muodostamiseen annetusta merkkijonosta, kuten salasanasta.
BLE / BTLE	Bluetooth Low Energy, Bluetooth 4.0 ja uudemmat versiot. Langaton likiverkkotekniikka elektronisten laitteiden väliseen kommunikointiin.
CSS	Cascading Style Sheets, WWW-dokumenttien tyylimäärittelyihin käytetty menetelmä.
Flask	Python-pohjainen verkko-ohjelmistokehys, jonka pohjana ovat Jinja2 ja Werkzeug.
Genuino	Suositun Arduino-mikrokontrollerialustan Yhdysvaltojen ulkopuolella käytetty markkinanimi.
Hall-anturi	Hall-ilmiön avulla magneettikentän havaitseva anturi, jota yhdessä magneetin kanssa käytetään esimerkiksi ovien tai ikkunoiden avoimuuden tunnistamiseen.

HTML	Hypertext Markup Language, WWW-sivunkuvauskieli.
IoT	Internet of Things, esineiden Internet.
ISM-taajuusalue	Industrial, Scientific and Medical. Avoin taajuusalue, jonka käyttö ei vaadi erillistä lupaa.
JSON	JavaScript Object Notation, avain-arvopareina esitetyn datan tiedonvälitykseen käytetty muoto.
MP4	MPEG-4-standardissa määritelty kääre ääni- ja video-datalle.
MPEG-4	Moving Picture Experts Group. MPEG-4-standardi on kokoelma äänen ja videon pakkaamistapoja ja -koodekkeja.
NAT	Network Address Translation, IP-osoitteiden muunnosprotokolla julkisen ja paikallisen IP-osoitteen yhdistämiseen.
nRF24	Lyhyen kantaman radioliikenneprotokolla, joka toimii vapaalla 2,4 GHz:n taajuusalueella.
PAN	Personal Area Network, likiverkko. Verkko, jossa henkilökohtaiset laitteet kommunikoivat keskenään. Teknologioita ovat esimerkiksi Bluetooth ja ZigBee.
PBX	Private Branch Exchange, puhelinvaihdejärjestelmä.
PIR	Pyroelectric (tai Passive) Infrared Sensor, liiketunnistukseen käytetty sensori, joka mittaa infrapuna- eli lämpösäteilyä.
Raspberry Pi	Eräs yhden piirilevyn tietokone, erityisesti harrastelijoiden suosima alusta elektroniikkaprojekteihin.

RBSS	Remote Burglar Surveillance System, rakennetusta järjestelmästä käytetty työnimi.
RTP	Real-time Transport Protocol, reaaliaikaisen datan siirtoprotokolla. Esimerkiksi VoIP-puhelun ääni ja kuva siirretään RTP:n avulla.
SFTP	SSH File Transfer Protocol, tietoliikenneprotokolla salattuun tiedonsiirtoon SSH-protokollan yli.
SIP	Session Initiation Protocol, tietoliikenneprotokolla multimediaviestintäyhteyden luomiseen.
SSD	Solid State Disk, uudelleenkirjoitettavaan ja säilyvään Flash-muistiteknologiaan perustuva massamuisti, missä ei perinteiseen HDD-levyyn verrattuna ole liikkuvia osia.
SSH	Secure Shell, salattuun komentorivipohjaiseen etäkäyttöön tarkoitettu protokolla.
SSL	Secure Sockets Layer, tiedonsiirron salaukseen internetin yli käytetty protokolla.
TLS	Transport Layer Security, SSL-protokollan seuraaja.
URI	Uniform Resource Identifier, tiedon sijaintiin osoittava merkkijono. Esimerkiksi sähköpostin URI on ”mailto:käyttäjänimi@verkkotunnus.tld”.
UUID	Universally Unique Identifier, 128-bittinen, lähes yksilöllinen merkkijono jonkin asian yksilöintiin.
VoIP	Voice over Internet Protocol, kattotermi tekniikoille, jotka välittävät ääntä IP-verkkojen yli.

VPS	Virtual Private Server, virtuaalinen palvelintietokone, joka on käytännössä asiakkaalle vuokrattava osuus isäntälaitteen resursseista.
WLAN	Wireless Local Area Network, langaton lähiverkkotekniikka elektronisten laitteiden väliseen tiedonsiirtoon.

1 JOHDANTO

Tämän opinnäytetyön tavoitteena oli suunnitella konsepti ja rakentaa esittelyversio langattomasta valvontajärjestelmästä, joka hälytyksen saatuaan käynnistää videotallennuksen ja soittaa VoIP-puhelun käyttäjälle. Työn toimeksiantajana oli Bittium Wireless Oy.

Työn monialaisen luonteen vuoksi ratkaisuissa piti mahdollisuuksien mukaan tukeutua valmiiden ohjelmistokirjastojen tarjontaan, sillä aikataulu olisi muutoin käynyt liian tiukaksi. Jokainen järjestelmän kolmesta pääosa-alueesta olisi tarjonnut riittävästi työtä kokonaiseksi projektiksi, joten kompromisseja on jouduttu tekemään monessa kohtaa ajan, oman osaamisen ja käytettävissä olevan laitteiston suhteen.

Järjestelmää rakentaessa on yhtenä ohjaavana tekijänä ollut sen laajennettavuus. Yksittäiseen järjestelmään voidaan lisätä sensoriyksiköitä tarpeen mukaan ja samalla järjestelmällä voi olla useita käyttäjiä. Vastaavasti yhdellä käyttäjällä voi myös olla monta valvontajärjestelmää hallinnassaan. Yleisellä tasolla voidaan ajatella yhtä järjestelmää esimerkiksi kodin valvontajärjestelmänä, jossa on haluttu määrä sensorein valvottuja ja ennalta haavoittuvaksi valittuja tiloja. Käyttäjät tässä tapauksessa voivat olla kaikki kiinteistön asukkaat, joilla voi olla hallinnassaan myös esimerkiksi kesämökin valvontajärjestelmä.

Vastaavanlaisia ratkaisuja on jo olemassa, ja työn tarkoituksena olikin uusien menetelmien oppiminen ja toimivan järjestelmän rakentaminen harjoitusmielessä. Työssä käytettiin myös muutamia ennalta valittuja ja vaadittuja teknologioita, kuten SIP ja BLE. Työn vaatimuksia on tarkemmin esitelty luvussa 2.

2 JÄRJESTELMÄN VAATIMUKSET

Vaatimusmäärittelyn ja koko järjestelmän suunnittelun pohjana toimi toimeksiantajan tuottama kaaviokuva (liite 1). Esittelyversioon rakennettavaan järjestelmään tuli myös mahdollisuuksien mukaan käyttää ennalta hankittuja komponentteja.

Vaatimusmäärittely jätti suunnittelulle reilusti liikkumavaraa ja konseptia rakentaessa uusia näkökulmia oli mahdollista yhdistää alkuperäiseen määrittelyyn. Alla mainittujen lisäksi varsinaisen työprosessin vaatimuksena olivat versionhallinnan käyttö ja säännölliset palaverit toimeksiantajan kanssa.

2.1 Teknologiavaatimukset

Keskeisiä teknologiavaatimuksia projektille olivat

- Linux-pohjainen ratkaisu
- langaton tiedonsiirto (Bluetooth Low Energy)
- VoIP
- PJSIP
- Asterisk PBX.

2.2 Toimintavaatimukset

Työn tavoitteena oli rakentaa järjestelmä, joka tunnistaa ympäristöstään hälytyksen arvoiset tapahtumat, kuten lämpötilan nousun tai ihmisen liikkeen, ja ilmoittaa siitä VoIP-puhelulla käyttäjälle. Järjestelmän tuli hälytyksen yhteydessä käynnistää kuva- ja äänitallennus. Hälytysten selaamiseen piti tehdä selainpohjainen käyttöliittymä, jonka kautta tallenteita oli mahdollista selata.

2.3 Muutokset projektin aikana

Työn edetessä alkuperäisiä suunnitelmia jouduttiin muuttamaan useita kertoja. Aikataulupaineen lisäksi muutoksia aiheuttivat valittujen tekniikoiden yhteensopimattomuus ja ennakoimattomien ominaisuuksien lisääminen. Jälkeenpäin tarkasteltuna koko projektia voidaan pitää suunnitteluprosessina, jonka tulosten

perusteella varsinaista esittelyversiota voisi lähteä rakentamaan. Projekti olikin enimmäkseen oppimisprojekti, jonka teknologioista ja menetelmistä suurin osa oli ennestään tuntemattomia. Tämä puolestaan johti siihen, että suunnittelu piti toteuttaa käytännön kautta.

Videopuhelutoiminnan korvaaminen

Alkuperäisenä vaatimuksena oli, että hälytyspuheluun olisi liitetty reaaliaikainen kuva ja ääni. Tämän toiminnallisuuden toteuttaminen kuitenkin jäi jatkoprojektiin, sillä projektille määritelty aikataulu ei olisi riittänyt ominaisuuden rakentamiseen. Työn esittelyversiossa tämä toiminta on korvattu toisistaan erillisillä videotallennus- ja puhelutoiminnoilla.

Useamman asiakkaan arkkitehtuuri

Vaatimusmäärittelyssä ei erikseen edellytetty usean järjestelmän näkökulmaa, mutta rakenne ohjautui tähän suuntaan heti projektin alkuvaiheessa. Käytännön vaikutukset näkyvät järjestelmän osien sijoitteluratkaisuissa ja tarkemmalla tasolla myös moduulien sisällä. Esimerkiksi Asterisk-puhelinvaihdeohjelmiston sijoittelulle oli muutamia vaihtoehtoja, mutta julkiselle palvelimelle asennettuna sillä voidaan palvella useita valvontajärjestelmiä samanaikaisesti.

Laajennettavuus ja palvelunäkökulma ohjasivat projektin etenemistä lopulta niin voimakkaasti, että sen voidaan katsoa olevan yksi lisävaatimus työlle, vaikka se ei tilaajalta tässä tapauksessa tullutkaan. Tämä näkökulma toisaalta laajensi projektia huomattavasti ja moni aiheeseen liittyvä kehitysidea jäi toteuttamatta aikataulun vuoksi. Tätä ja muita mahdollisia jatkoprojektin aiheita on käsitelty luvussa 6.

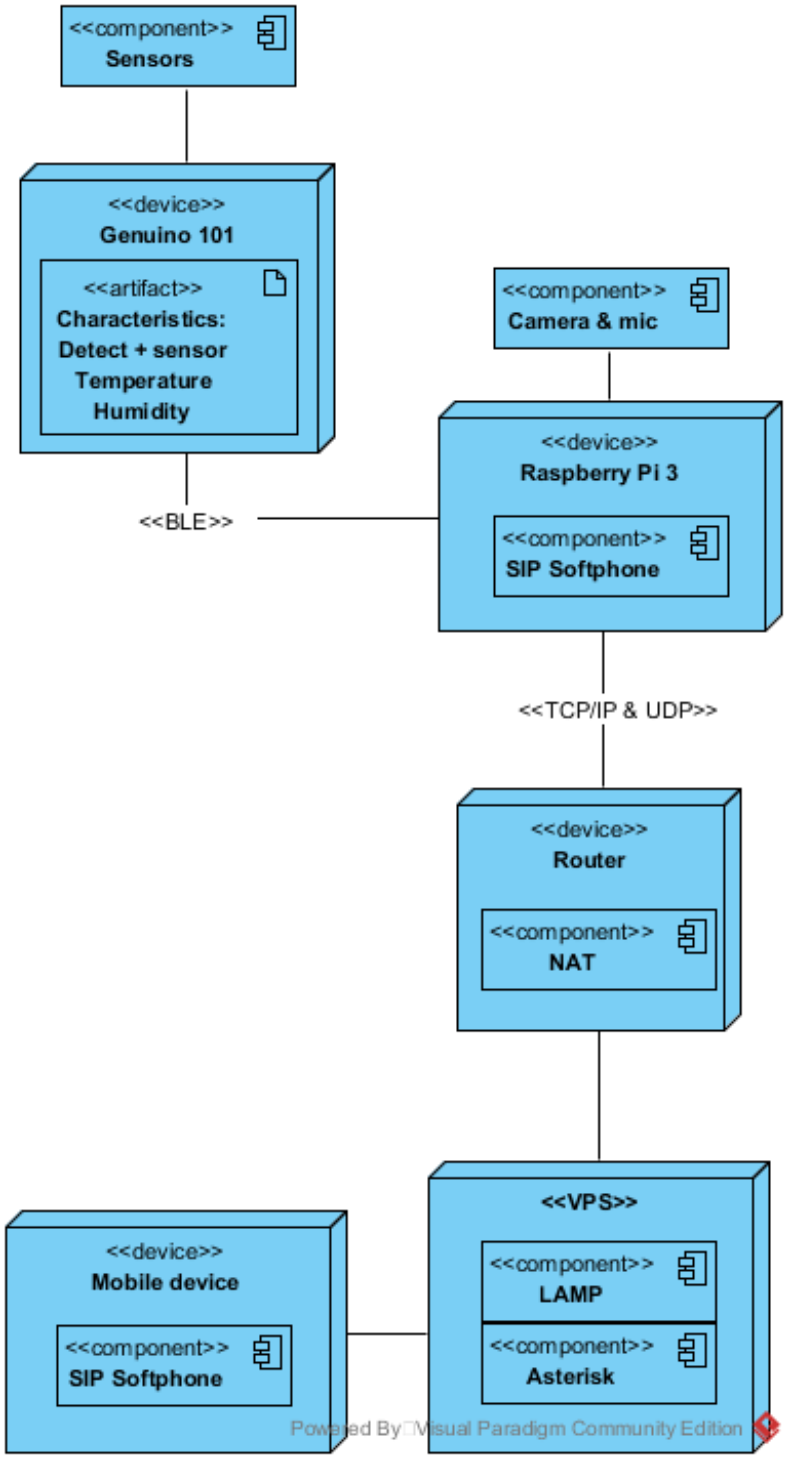
3 LAITERAKENNE

Työn tuloksena valmistunut järjestelmä voidaan jakaa kolmeen erilliseen osaan, jotka toistuvat sekä laite- että ohjelmistorakenteen esittelyssä: **sensorisolmu** (Genuino 101), **keskussolmu** (Raspberry Pi) ja **palvelin**.

Ensimmäinen vedos rakenteesta (liite 2) perustui kokonaisuudessaan komponentteihin, jotka olivat saatavilla jo valmiiksi. Oleellisin ero lopulliseen ratkaisuun on paikallisessa radioliikenteessä, joka alun perin oli tarkoitus toteuttaa nRF24-lähetin-vastaanottimilla. nRF24-piirien haasteena ovat kuitenkin sen vaatimukset virransyötölle (1; 2), jotka olisivat korostuneet komponenttien halpaversioiden kanssa.

Ensimmäisen version rakennetta katsottiin tarpeelliseksi muokata vaihtamalla sensorisolmujen alustat uudempaan ja ominaisuuksiltaan monipuolisempaan Genuino 101 -ohjainalustaan. Perusteena muutokselle olivat nRF24-piirien edellä mainitut ongelmat ja integroidun BLE:n hyödyt suositun PAN-teknologian tuoreena versiona.

Toteutusvaiheessa myös palvelin siirtyi julkiseen Internet-verkkoon, jotta yhdellä palvelimella on mahdollisuus palvella useita järjestelmiä samanaikaisesti. Järjestelmän lopullinen komponenttisijoittelu (kuva 1) noudattaa usean asiakkaan ja järjestelmän mallia, joka on laajennettavissa tarpeen mukaan.

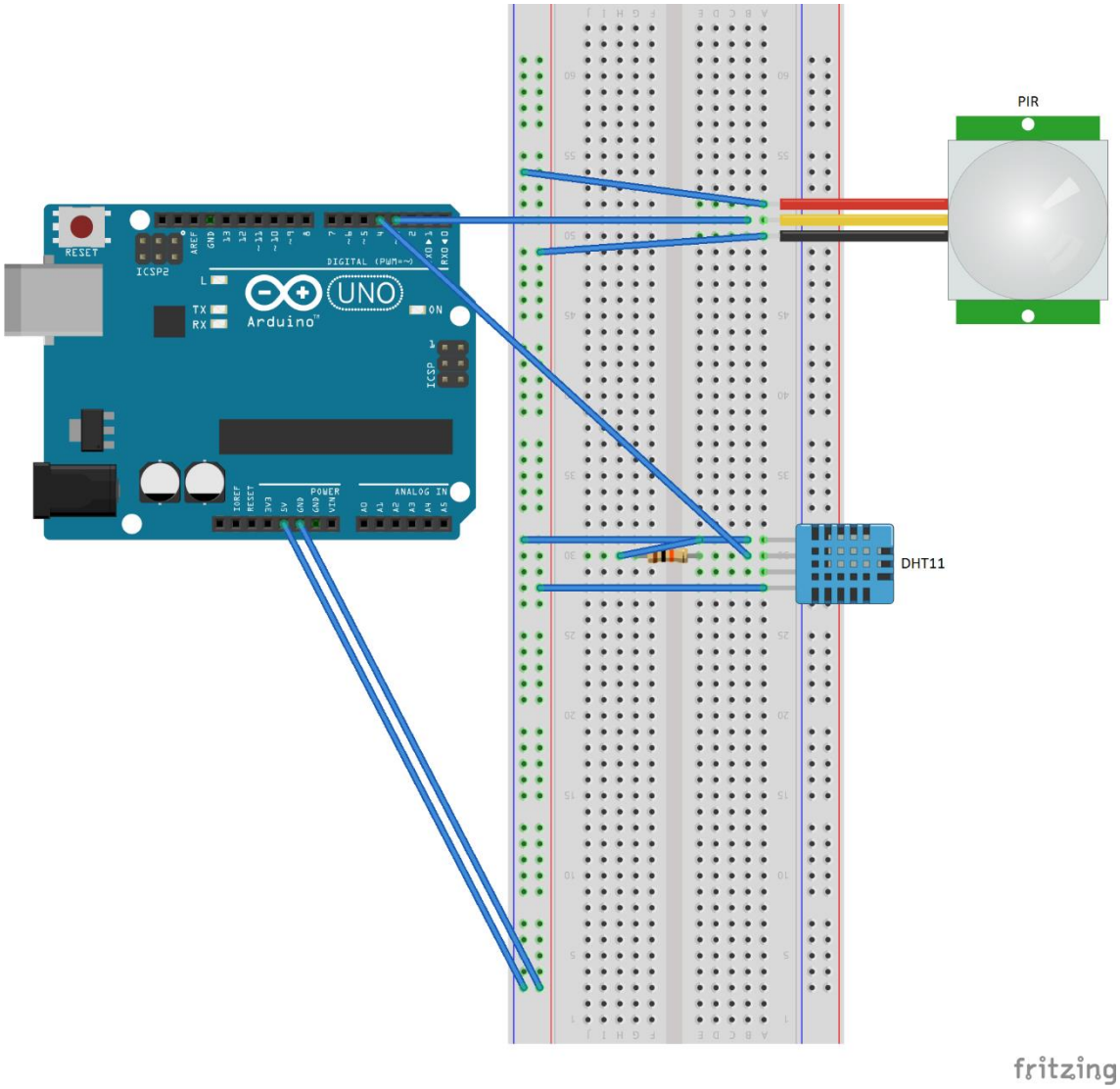


KUVA 1. Järjestelmän sijoittelukaavio.

3.1 Sensorisolmu

Genuino 101-pohjainen sensorisolmu on järjestelmän osa, joka tarkkailee ympäristöään sensorien avulla ja laukaisee hälytyksen, kun jokin asetettu raja-arvo

ylittyy. Esittelyversion laitteisto tarkkailee sekä liikettä (PIR-sensori) että lämpötilaa (DHT11-sensori). (Kuva 2.)



KUVA 2. Sensorisolmun kytkentä. Kuvassa Genuino 101:n sijaan Arduino Uno, joka on profiililtaan lähes vastaavanlainen. (3.)

3.1.1 Alusta

Mikrokontrollerialustaksi valikoitui lopulliseen kokoonpanoon Genuino 101, jonka sisäänrakennettu ominaisuusvalikoima nosti sen vanhempien Arduino Uno- ja Nano -alustojen yläpuolelle. Genuino 101 perustuu Intelin Curie-moduuliin, johon on integroitu mm. Bluetooth Low Energy -tiedonsiirtotekniikka ja kiihtyvyysanturi. (4; 5, s. 9.)

Sensorisolmun konseptissa myös virrankulutus on merkityksellinen, sillä solmun olisi tietyissä sijoituspaikoissa tarpeen toimia myös akkukäyttöisenä. 101 on virrankulutusmittauksessa huomattavasti edeltäjäänsä parempi: 68 mA (101) ja 144mA (Uno R3) (6). Testi ei kuitenkaan ota huomioon tietoliikenteen vaikutusta virrankulutukseen kummallakaan alustalla. Genuino 101 olisi siis myös akkukäyttöiseen sensorisolmuun Uno R3:a kannattavampi vaihtoehto.

3.1.2 Sensorit

Esittelyversiossa sensoreita on kaksi, liiketunnistin (PIR) ja lämpötila- ja ilmankosteussensori. Sensorivalikoimaa on kuitenkin mahdollista jatkossa laajentaa tarpeen mukaan esimerkiksi Hall-anturilla tai Genuino 101:n sisäisellä kiihtyvyysanturilla.

Lämpötila ja ilmankosteus

DHT11-sensori mittaa sensorisolmun ympäristön lämpötilaa ja ilmankosteutta digitaalisesti. Sensori on sekä tarkkuudeltaan että mittausalueeltaan hyvin heikko. Suhteellisen ilmankosteuden mittaustarkkuudeksi on ilmoitettu ±5 % ja vastaavasti lämpötilan ±2 °C. Mittausresoluutio on 1, joten parhaimmillaankin sensori jättää toivomisen varaa. Mittausalueet ovat 0–50 °C ja 20–90 %. (7.)

Testikäytön aikana kävi ilmi, että DHT11 toimi hyvin ailahtelevaisesti. Luenta epäonnistui toistuvasti tai saatu mittaus ei vastannut todellisuutta. Tämä saattaa kuitenkin johtua viallisesta yksilöstä, sillä varakappaletta ei ollut saatavilla.

Puutteistaan huolimatta sensori oli kuitenkin riittävä projektin tarpeisiin. Heikko mittaustarkkuus ei ollut rasite, kun lämpötilamittausta käytettiin vain hälytysrajojen hakuun, jossa yhden asteen resoluutio kahden asteen mittaustarkkuudella oli riittävä.

Lämpötilan mittaustarkkuuden toteamiseksi sensoria testattiin ennen käyttöönottoa ennalta tunnetuissa ympäristöissä, kuten jääkaapissa. Testausten perusteella muodostui käsitys kiinteästä, n. +4 °C:n poikkeamasta, joka piti kompensoida ohjelmistossa vastaluvulla.

Liiketunnistus

Liikkeentunnistukseen käytetty PIR-sensori toimi luotettavasti koko projektin ajan. Sensorin tarkkaa mallia ei ole tiedossa, mutta malliltaan ja kytkennöiltään se mukailee Adafruitin PIR-sensoria (8). Digitaalinen ulostulosignaali antaa 3V jännitteen, kun liikettä havaitaan. Sensorin herkkyys, viive ja uudelleenlaukaisuasetus ovat säädettävissä piirilevyn potentiometreillä.

Adafruitin sensorille luvataan kuuden-seitsemän metrin toimintasäde (8), mutta tarkkoja käytännön testejä ei esittelylaitteen sensorille ole tehty. Käyttötestauksen perusteella liiketunnistus toimii ainakin kahden-kolmen metrin etäisyydeltä toivotulla tavalla.

3.2 Keskussolmu

Genuino 101:n valitsemisen jälkeen myös keskussolmun valintaa oli syytä tarkastella uudelleen. Koska tiedonsiirtoprotokollaksi sensorisolmun ja keskussolmun välillä valikoitui BLE, oli Raspberry-pienoistietokoneen kolmas versio sisäänrakennetun Bluetooth-tuen vuoksi sopivin valinta keskussolmuksi. Raspberry Pi 3 -versioon on lisätty myös WLAN-tuki. Langattoman lähiverkon myötä laitteen sijoituspaikka on vapaammin valittavissa, koska langallinen verkkoinfrastruktuuri ei rajoita sijaintia. (9.)

Kameran lisäksi Raspberryyn ei ole liitetty muita laitteita. Laite on yhteydessä Internetiin langattoman lähiverkon kautta ja tarvittaessa siihen saa yhteyden SSH-yhteydellä palvelimen kautta. Raspberryn ja palvelimen välillä pidetään pysyvästi avoimena SSH-tunnelia, jota voidaan käyttää reitittimen NAT-muunnoksen ohittamiseen.

Kamera-mikrofoni

Keskussolmuun on liitetty kuvan ja äänen tallennusta varten perinteinen verkkokamera. Kamera on malliltaan Logitech C270 ja siinä on myös mikrofoni. Vaikka valmistaja ei lupaa toimivuutta Linux-ympäristössä, toimi se kuitenkin ongelmitta eikä vaatinut kytkennän jälkeen konfigurointia. (10.)

3.3 Palvelin

Palvelimena järjestelmässä on käytetty virtuaalipalvelinta (VPS), jonka isäntäkone sijaitsee Amsterdamissa palveluntarjoajan (DigitalOcean) tiloissa. Virtuaalikoneelle on varattu yksi suoritin, 512 MB keskusmuistia ja 20 GB SSD-levytilaa. Palvelin on julkisen IP-osoitteen takana ja siihen on kytketty verkkotunnus.

4 OHJELMISTORAKENNE

Tässä luvussa käsitellään järjestelmän ohjelmistorakennetta ja esitellään työssä käytetyt kirjastot. Ohjelmisto on laiterakenteen tavoin jaettu kolmeen samanlaiseen osaan.

Versionhallintaohjelmistona projektissa oli Git. Genuinon ohjelmaa lukuun ottamatta editorina on käytetty Atom-tekstieditoria Python-laajennuksella. Genuinon ohjelma on kirjoitettu ja käännetty Arduinon omalla editorilla.

Genuinon ohjelma on kirjoitettu C++-kielellä, keskussolmun ohjelma Python-kielellä ja palvelimen käyttöliittymäohjelmoinnissa on käytetty sekä Pythonia että JavaScriptiä.

Ohjelmiston rakentaminen eteni kokeiluluonteisesti, mikä johti useisiin ennakoimattomiin tilanteisiin projektin aikana. Jossain määrin tämä vaikutti myös järjestelmän toiminnallisuuteen, mikä osittain poikkeaa vaatimusmäärittelystä siten, että esimerkiksi videopuhelu ei ole mahdollinen.

4.1 Vaaditut teknologiat

Tämä luku esittelee kaksi työn kannalta merkittävää teknologiaa, Bluetooth Low Energyn ja SIP:n. Sekä BLE että SIP olivat myös työn vaatimusmäärittelyssä teknologiavaatimuksina.

4.1.1 Bluetooth Low Energy

BLE on Bluetooth 4.0 -versioon lisätty vähävirtainen versio perinteisestä Bluetooth-tekniikasta. Nimensä mukaisesti BLE on suunniteltu käyttämään vähemmän virtaa perinteiseen Bluetooth-tekniikkaan verrattuna (11; 12). Tällä muutoksella BLE on edeltäjiään houkuttelevampi ratkaisu myös IoT-toteutuksiin, joissa vähävirtaisuus ja sen myötä pitkät toiminta-ajat ovat ensisijaisen tärkeitä.

Periaatteeltaan BLE vastaa perinteistä Bluetooth-tekniikkaa ja sen keskeisimmät ominaisuudet ovat edelleen samat. BLE toimii 2,4 GHz:n ISM-taajuudella ja käyttää taajuushyppelytekniikkaa tiedonsiirrossa. Merkittävin ero on kuitenkin

yhteyden ylläpidossa ja sen myötä radiorajapinnan suoritustehossa. BLE on lepotilassa aina kun tietoa ei siirretä, joten tiedonsiirtonopeudet jäävät korkeintaan 100 kbit/s:n tasolle. Näistä syistä Bluetooth Low Energy soveltuu erityisen hyvin satunnaiseen radioliikenteeseen, jossa siirrettävät tietomäärät ovat pieniä. (13.)

BLE-yhteys muodostetaan keskuslaitteen (central) ja oheislaitteen (peripheral) välille. Oheislaite tarjoaa BLE-arkkitehtuurissa palveluita (service), jotka on jaettu edelleen pienempiin kokonaisuuksiin (characteristic). Keskuslaite voi lukea haluamansa characteristic-arvot oheislaitteen tarjonnasta tai tilata niihin tehdyt muutokset. (13.) Tässä toteutuksessa oheislaitteena toimii Genuino-sensorisolmu, joka tarjoaa palvelun Raspberry-keskuslaitteelle.

Tässä ratkaisussa characteristic-arvojen tilaustoiminto ei ole käytössä, vaan keskuslaite lukee arvot ennalta määritetyillä aikaväleillä ja kirjoittaa tarvittaessa niille uudet arvot. Characteristic-arvojen muutoksiin perustuva ohjelma olisi ollut sopivampi vaihtoehto kokonaisuutta ajatellen, sillä sen avulla radioliikennettä ja ohjelman vaatimaa prosessoriaikaa olisi saatu tältä osin vähennettyä huomattavasti. Tämä on merkittävä seikka erityisesti siksi, että yhtenä jatkokehitysmahdollisuutena on järjestelmän akkukäyttöisyys ja tilausominaisuudella olisi virrankulutukseen suora vaikutus. Tilausominaisuuden lisääminen projektin keskivaiheilla olisi kuitenkin aiheuttanut melkoisesti uudelleensuunnittelua erityisesti keskussolmun ohjelmarakenteessa, ja tästä syystä characteristic-arvojen muutostilaustoiminto on jätetty jatkokehitysmahdollisuudeksi.

4.1.2 SIP

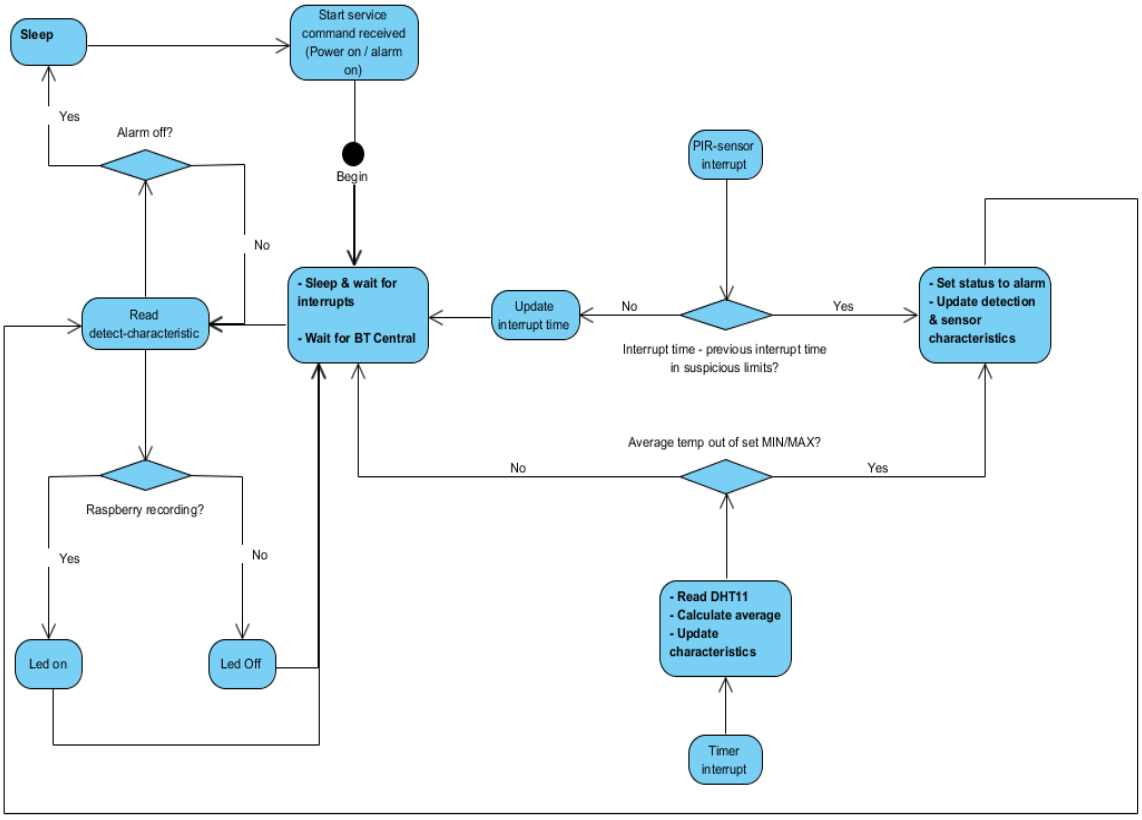
Session Initiation Protocol on tietoliikenneprotokolla, jota käytetään esimerkiksi VoIP-puheluiden muodostamiseen ja hallintaan. SIP-liikenteessä ei siirretä varsinaista dataa lainkaan, vaan ainoastaan hallinnoidaan osapuolien välisiä yhteyksiä. Puheludatan siirtoon luodaan erillinen RTP- tai RTSP-tiedonsiirtoyhteys, mitä ei tässä projektissa ole videopuhelun hylkäämisen vuoksi käytetty lainkaan. (14, s. 8–9.)

SIP-protokollassa on laajennusten myötä myös useita pikaviestimistä tuttuja toimintoja, kuten yhteyshenkilön tilatiedon (15) välittäminen ja pikaviestit (16). Näitä ominaisuuksia on hyödynnetty myös tässä projektissa.

Yhteys muodostetaan kontaktin SIP-URI:n avulla. URI on muodoltaan ”sip:user@domain.com”, jossa domain.com on VoIP-palveluntarjoajan verkko-osoite. Tässä tapauksessa palveluntarjoajana toimii projektissa käytetty palvelin.

4.2 Sensorisolmu

Genuinon ohjelma on hyvin yksinkertainen kokonaisuus. Ohjelmassa suoritetaan sensorien luku ja BLE:n characteristic-arvojen kirjoitus ja tarkkaillaan keskussolmun kirjoittamia muutoksia tilakoneeseen (esimerkiksi hälytys päälle tai pois). Genuinon ohjelmaa voidaan kuvata tavallisella vuokaaviolla (kuva 3).



KUVA 3. Genuino-ohjelman vuokaavio.

4.2.1 Kirjastot

Genuinon ohjelmassa on hyödynnetty CurieBLE- ja CurieTimerOne-kirjastoja tiedonsiirtoon ja ajastukseen. Näiden lisäksi käytössä on Adafruitin DHT-kirjasto lämpötilasensorin lukemiseen.

CurieBLE ja CurieTimerOne ovat Intelin avoimen lähdekoodin kirjastoja (17). CurieBLE:ssä on korkean tason funktiot ja luokat Bluetooth Low Energy-yhteyden muodostamiseen. CurieTimerOne puolestaan on ajastinkeskeytyksiin käytettävä kirjasto.

4.2.2 BLE-rajapinta

Sensorisolmu tarjoaa yhden BLE-palvelun, jossa on kuusi characteristic-arvoa (taulukko 1). Jokainen characteristic vaatii oman UUID:nsä, jotka on johdettu koko järjestelmälle generoidusta UUID:stä kasvattamalla viimeistä numeroa yhdellä. Tällä tavoin UUID:n perusteella voidaan päätellä, kohdistuuko se koko-naiseen järjestelmään vai johonkin sen alaosista.

TAULUKKO 1. Sensorisolmun BLE-palvelun characteristic-valikoima.

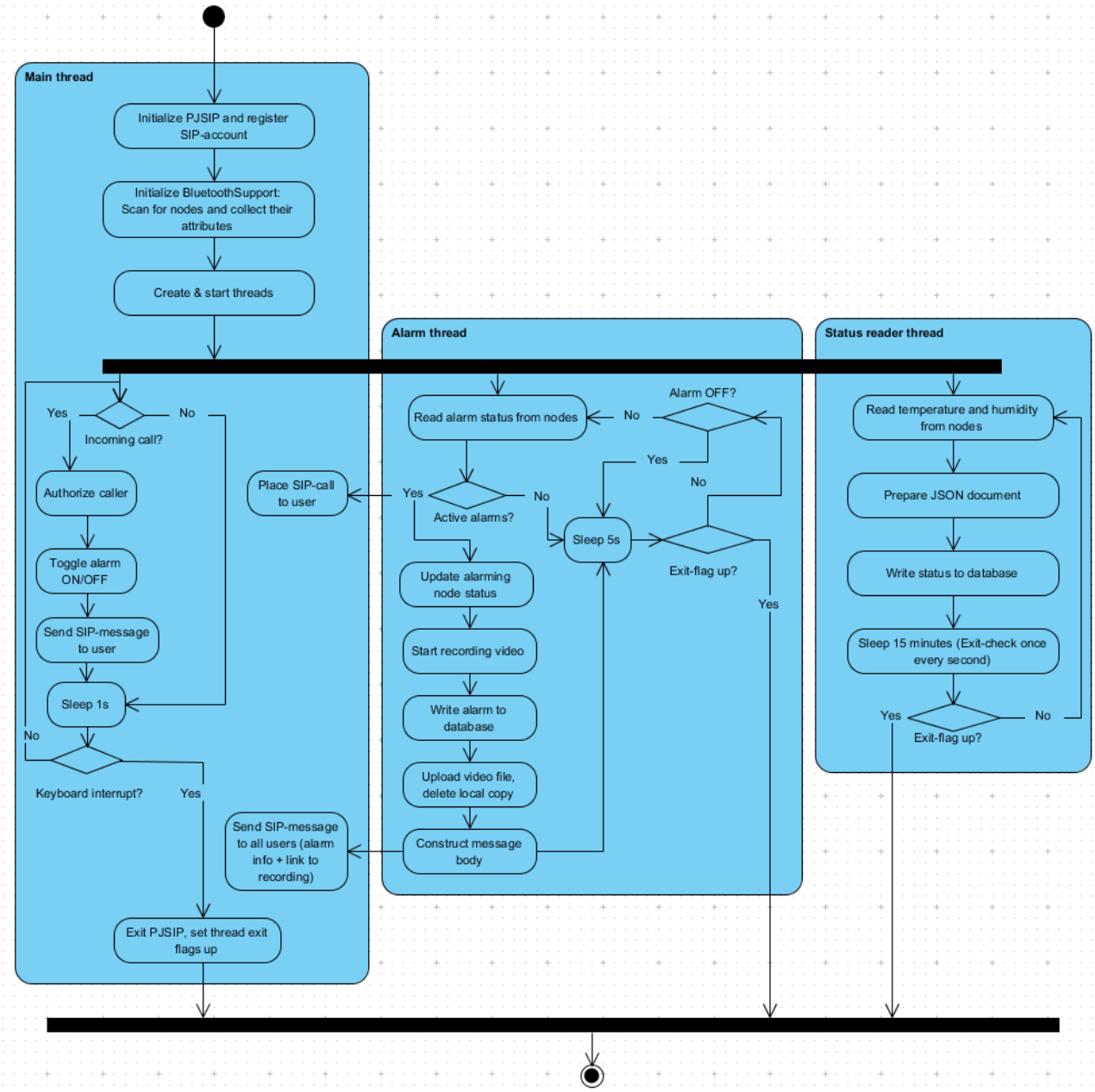
Nimi	Tyyppi	Tietotyyppi	Käyttötarkoitus
Detection	Luku, kirjoitus	Kokonaisluku	Hälytystila (0 = ON, 1 = Kuittaamaton hälytys, 2 = Keskussolmu tallentaa, 3 = OFF)
Sensor	Luku	Kokonaisluku	Hälyttävän sensorin numero
Temp	Luku	Liukuluku	Uusin lämpötilalukema
TempAvg	Luku	Liukuluku	Lämpötilamittausten keskiarvo
Humid	Luku	Liukuluku	Uusin kosteusmittauslukema
Record	Luku, kirjoitus	Merkki	Keskussolmun tallennuksen indikaattori

4.3 Keskussolmu

Keskussolmulle tehty ohjelma on koko järjestelmän mittakaavassa monimutkaisin kokonaisuus. Karkealla tasolla kuvattuna ohjelman toiminta voidaan esittää listana toimintoja. Keskussolmun ohjelma esittelyversiossa

- etsii järjestelmään kuuluvat sensorisolmut ja yhdistää niihin
- lukee sensorisolmujen tarjoamat lämpötila- ja ilmankosteustiedot 15 minuutin välein
- lukee sensorisolmujen hälytystilan 5 sekunnin välein
- kirjoittaa sensorisolmujen lämpötila- ja ilmankosteustiedot ja keskussolmun järjestelmäresurssitilanteen tietokantaan palvelimella
- reagoi hälytyksiin käynnistämällä videotallennuksen ja soittamalla käyttäjälle VoIP-puhelun. Videotiedosto siirretään päättyneen tallennuksen jälkeen palvelimelle ja hälytyksestä kirjoitetaan merkintä tietokantaan. Käyttäjälle lähetetään lisäksi SIP-pikaviesti, jossa annetaan hälytyksen tiedot ja linkki tallenteeseen.

Keskussolmun ohjelmarakenne aktiviteettikaaviona on esitetty kuvassa 4.



KUVA 4. Keskussolmun ohjelmarakenteen aktiviteettikaavio.

4.3.1 Kirjastot

Keskussolmun ohjelma hyödyntää kahta merkittävää ohjelmakirjastoa: BlyePy ja PJSUA Python. BluePy tarjoaa BLE-toiminnot ja PJSUA Python puolestaan PJSIP-kirjaston.

BluePy-kirjasto on korkean tason BLE-rajapinta Python-kielelle Linux-ympäristössä (18). Kirjasto on edelleen kehitysvaiheessa, joten kaikkia BLE:n ominaisuuksia ei projektin tekovaiheessa ollut vielä tarjolla.

Avoimeen lähdekoodiin perustuvan PJSIP-kirjaston Python-rajapinta on nimeltään PJSUA Python (19). Kirjasto mahdollisti helpon VoIP-toiminnallisuuden integroinnin ohjelmaan SIP-tietoliikenneprotokollaa hyödyntäen. Myös PJSIP-oppaan esimerkit antoivat hyvän lähtökohdan soveltaa kirjastoa tähän projektiin.

4.3.2 Moduulit

Keskussolmun ohjelma koostuu seitsemästä Python-moduulista:

- Alarmhandler: Hälytysten käsittely
- Auto: Pääohjelma, alustukset ja säikeiden käynnistys
- BLE: Bluetooth LE-toiminnot
- Caller: SIP-viestintä, puhelut, tilatiedot
- DBops: Tietokantaoperaatiot
- Logger: Lokimerkinnät
- Worker: Säikeissä ajettavat tehtävät.

Näiden moduulien toimintaa on avattu tarkemmin moduulikohtaisten otsikoiden alla, pääohjelmasta (*Auto*) aloittaen.

Auto

Keskussolmun pääohjelma on kolmivaiheinen kokonaisuus. Ohjelman käynnistyksen yhteydessä luodaan ja alustetaan ohjelman toiminnan kannalta keskeinen *BluetoothSupport*-olio. Ohjelman alussa myös luodaan ja käynnistetään ajastukseen vaadittavat säikeet.

Ajon aikana pääohjelma vastaa säikeiden välisestä viestinnästä. Pääohjelma tarkkailee viestilippujen tilaa ja reagoi niihin tarvittaessa. Esimerkiksi hälytyskäsittelijän ilmoittama puhelutarve kulkee viestilippujen kautta säikeiden välillä.

Ohjelman suoritus keskeytetään näppäimistökeskeytyksellä Ctrl+C, jonka havaittuaan pääohjelma huolehtii säikeiden pysäyttämisestä ja yhdistämisestä

prosessin pääsäikeeseen. Myös *Caller*-moduulin käyttämä PJSIP-olio vaatii hallitun sulkemisen, joka tehdään pääohjelman kutsumalla metodilla.

Alarmhandler

Hälytystapahtumat käsitellään *Alarmhandler*-moduulissa. Hälytyksistä ylläpidetään keskussolmukohtaista juoksevaa numerointia, joka tallennetaan tekstitiedostoon ohjelman hakemistossa. Hälytyksen järjestysnumero yhdistettynä keskussolmukohtaiseen UUID-tunnukseen muodostavat jokaiselle hälytykselle yksilöllisen tunnisteen, jota käytetään mm. hälytykseen liittyvän videotiedoston nimenä.

Videon tallentamiseen kameralta käytetään aliprosessina ajettavaa avconv-työkalua, joka annetuilla parametreilla muodostaa MPEG-4-pakatun MP4-videotiedoston. Avconv-työkalulla muodostetaan myös aikaleima jokaiseen kuvaan. Työkalu tallentaa myös äänen, joka pakataan AAC-koodauksella.

Alarmhandler-moduulissa on vain yksi funktio, jota kutsutaan, kun hälytystarkastuksia ajava säie on havainnut hälytyksen joltain sensorisolmulta. Koska funktio suoritetaan aina rinnakkaisessa säikeessä, pitää *Caller*-moduulin soitto-ominaisuuksia käytettäessä viedä viesti pääohjelman kautta säikeestä toiseen.

Moduulin toiminta hälytyshetkestä alkaen voidaan esitellä kronologisessa järjestyksessä. Kun puhelutarpeesta on välitetty viesti *Caller*-moduulille, kirjoitetaan hälytyksen laukaisseelle sensorisolmulle uusi tilatieto. Uusi tila kertoo keskussolmun havainneen hälytyksen ja käynnistäneen sen käsittelyn.

Videon tallennukseen käytetty aliprosessi käynnistetään ja tallennusta jatketaan ennalta määritellyn ajan. Tallennuksen päätyttyä videotiedosto siirretään palvelimelle SFTP-protokollan avulla ja paikallinen tiedosto poistetaan, koska keskussolmun tallennustila on hyvin rajallinen.

Hälytyksestä kirjoitetaan tietokantaan dokumentti, jossa kerrotaan hälyttävän sensorisolmun UUID, hälyttävä sensori, hälytyksen juokseva järjestysnumero ja videotallenteen tiedostonimi. Onnistuneen tietokantatallennuksen jälkeen juokseva järjestysnumero päivitetään myös paikalliseen tekstitiedostoon.

Hälytystapahtuman viimeisenä toimenpiteenä on lähettää kaksi SIP-pikaviestiä kaikille järjestelmään liitetyille käyttäjille. Viestien teksti muodostetaan järjestelmän osien selväkielisistä nimistä. Nimet kuvaavat järjestelmää, sen sensorisolmuja ja niihin kytkettyjä sensoreita. Näiden pohjalta viestistä muodostetaan teksti, joka on käyttäjille helppolukuinen. Toinen käyttäjille lähetettävä viesti sisältää linkin videotiedostoon palvelimella. Viestien esimerkkimuoto on nähtävillä viidennessä luvussa kuvassa 12.

BLE

BLE-moduuliin on koottu kaikki Bluetooth Low Energy -rajapintaa käyttävät funktiot ja metodit *BluetoothSupport*-luokan alle. *BluetoothSupport*-olion avulla ohjelman eri moduulit voivat käsitellä yhteistä radiorajapintaa, joka liittää sensorisolmut järjestelmään. Kaikki tässä luvussa esitellyt toiminnot toteutetaan ohjelmassa *BluetoothSupport*-luokan kautta.

InitialConnect() on ohjelman käynnistyksen yhteydessä ajettava *BluetoothSupport*-olion alustusmetodi. Metodissa skannataan ympäristön BLE-laitteet ja yhdistetään sensorisolmuiksi tunnistettuihin oheislaitteisiin. Sensorisolmuilta kerätään niiden tarjoamien palveluiden characteristic-tiedot olion tietojäseniin talteen.

Luokassa on omat metodit lämpötilojen ja hälytysten lukuun sensorisolmuilta. Molemmat metodit toimivat samalla periaatteella:

- Kaikki olioon tallennetut sensorisolmut käydään läpi.
- Jokaisesta sensorisolmusta tunnistetaan haluttu characteristic, ja sen arvo luetaan muuttujaan.
- Metodi palauttaa avain-arvopareina sensorisolmun UUID:n ja halutun characteristic-arvon.

Hälytyskuittaukseen on oma metodinsa, jonka toimintaperiaate on lähestulkoon sama kuin lukumetodeissa:

- Etsitään haluttu sensorisolmu UUID:n perusteella.
- Etsitään haluttu characteristic UUID:n perusteella.

- Kirjoitetaan tilatieto characteristicin arvoksi, jolloin sensorisolmu voi todeta hälytyksen olevan kuitattu.

Hälytyskuittauksen kanssa samaan tapaan toimii hälytyksen tilanvaihtometodi, joka kirjoittaa kaikille olioon tallennetuille sensorisolmuille järjestelmän muuttuneen tilan (kytketty/poiskytketty). Tämän tiedon pohjalta sensorisolmut kytkevät toimintoja tarpeen mukaan päälle ja pois päältä.

BLE-moduulista puuttuu alkuperäisestä suunnitelmasta poiketen kaksi metodia, jotka voidaan siirtää jatkokehityksen piiriin. Ensimmäinen on BLE-rajapinnan uudelleenskannaus ja löytyneiden laitteiden listan päivitys. Toinen puolestaan todentaisi sensorisolmujen toimivuuden tietokantaan tehtävää tilatietokirjausta varten. Esittelyversiossa tilatiedoksi ilmoitetaan aina ”OK”, vaikka mitään tarkistusta järjestelmän osille ei tehdä.

Caller

Caller-moduulissa ovat kaikki PJSUA Python -kirjastoa käyttävät funktiot, metodit ja alaluokat. Moduulissa on kolme luokkaa, jotka perivät PJSIP-kirjaston vaatimusten mukaisesti kirjaston yliluokat:

- *AccountCallback*
- *BuddyCallback*
- *CallCallback*

Ensimmäisellä ajokerralla kirjasto alustetaan, määritellään käytettävät tietoliikenneportit ja rekisteröidytään SIP-palvelimelle.

Moduulissa on ohjelman käyttöön seuraavat palvelut:

- seurattavan käyttäjän tilatiedot (SIP-presence)
- hälytysviestit SIP-pikaviestinä kaikille järjestelmän käyttäjille
- SIP-pikaviestit yhdelle käyttäjälle valinnaisella tekstillä
- SIP-puhelutoiminnot ilman varsinaista RTP-dataa
- hälytysten päälle- ja poiskytkentä saapuvien puheluiden perusteella.

DBops

Kaikki keskussolmun tekemät tietokantaoperaatiot on koottu yhteiseen moduuliin *DBops*. Sekä tila- että hälytystiedon kirjoitus tehdään *DBops*-moduulin funktioilla palvelimella ajettavaan tietokantaan.

Moduulissa kerätään myös tilatietokirjausta varten keskussolmun resurssitilanne. Tiedot noudetaan suoraan käyttöjärjestelmän ajonaikaisesta tilasta Linuxin proc-hakemistosta.

Tietokantamerkinnät tehdään MongoDB-tietokantaan JSON-muodossa, joka Pythonissa rakennetaan dictionary-tietotyypin avulla.

Logger

Logger-moduuli tekee kootusti koko ohjelmanlaajuiset lokimerkinnät käyttöjärjestelmälokiin (syslog). Moduulissa on vain yksi funktio *log*, jolle annetaan parametreina moduulinimi, josta lokimerkintä tehdään ja varsinainen lokikirjaus.

Worker

Worker-moduulissa muodostetaan ohjelmalle rinnakkaisrakenne säikeiden avulla. Moduulissa on luokka *Worker*, jonka oliot muodostavat omat säikeensä. Ohjelman tarvitsemat ajastetut toiminnot (tilatiedon luku ja hälytysten luku) on toteutettu säikeiden avulla.

Ajastus on rakennettu for-silmukan avulla, jossa säie nukutetaan sekunniksi kerrallaan niin monta kertaa, kuin ajastuksen kannalta halutaan. Esimerkiksi 15 minuutin ajastus on toteutettu ajamalla sekunnin nukutussilmukka 900 kertaa. Tällä rakenteella säie pystyy reagoimaan muutoksiin kerran sekunnissa, toisin kuin 900 sekunnin yhtäjaksoisella nukutuksella. Mahdollisia muutoksia ovat esimerkiksi ohjelmasta poistuminen ja hälytyksen poiskytkentä. Varsinainen ajastettu toiminta suoritetaan ennen silmukkaan palaamista.

Hälytystapahtuman yhteydessä ongelmaksi muodostuu prosessin pääsäikeen ja hälytysluentaa tekevän säikeen ero. *Caller*-moduulin soittotoimintoja ei voida käyttää rinnakkaisesta säikeestä, vaan kutsun on tultava samasta säikeestä.

Tämä rajoitus on PJSUA Python -kirjastossa ja ominaisuus on ohjelmassa kierretty globaalilla lippumuuttujalla, joka viestii soittotarpeen kahden säikeen välillä.

Hälytyslukua ajastavassa säikeessä tarkkaillaan myös globaalia muuttujaa, joka viestittää tarpeen muuttaa hälytysjärjestelmän tilan päälle tai pois päältä. Mikäli ohjelman tila on muuttumassa (käyttäjä on kytkenyt järjestelmän pois päältä tai päälle), ajaa säie *BluetoothSupport*-olion *switchAlarm()*-metodin, joka kirjoittaa uuden tilan kaikille sensorisolmuille.

4.4 Palvelin

Palvelimelle on koottu järjestelmän komponenteista ne, joiden sinne loogisesti voidaan ajatella kuuluvan. Poikkeuksena on Asterisk-puhelinvaihdeohjelmisto, jonka sijoituksessa oli useita vaihtoehtoja. Asterisk vastaa järjestelmässä SIP-rekisteröinnistä, tilatietojen päivittämisestä ja puheluiden ja pikaviestien välityksestä. Asterisk-ohjelmiston sijoituksen perusteena on useamman asiakkaan arkkitehtuuri, jolloin yhtä palvelinta ja Asterisk-konfiguraatiota voidaan käyttää järjestelmän kaikkien asiakkaiden palvelemiseen tiettyyn pisteeseen asti. Mikäli Asterisk olisi samassa lähiverkossa keskussolmun kanssa, vaatisi jokainen asiakasjärjestelmä myös oman Asterisk-asennuksensa.

Järjestelmässä on kaksi erillistä tietokantaa, jotka on fyysisesti sijoitettu palvelimelle. Toiseen tietokantaan tallennetaan järjestelmän lähettämä tilatieto ja hälytykset, toinen on käyttäjien ja järjestelmän osien ylläpitoon tarkoitettu tietokanta.

Käyttöliittymän toiminnallisuus on muodostettu Flask-ohjelmistokehyksen avulla ja ulkoasu on viimeistelty valmiilla Bootstrap-mallilla.

4.4.1 Asterisk IP PBX

Asterisk on Digiumin kehittämä avoimen lähdekoodin puhelinvaihdeohjelmisto (20). Projektin aikana kävi ilmi ohjelmiston monipuolisuus ja muokattavuus sekä positiivisessa että negatiivisessa mielessä. Asterisk sisältää perusasennuksellaankin laajan valikoiman toimintoja, joita voidaan entisestään muokata omilla

AGI-laajennuksilla. Vaikka järjestelmä oli jälkikäteen tarkasteltuna varsin yksinkertainen asentaa ja ottaa käyttöön, oli mahdollisuuksien paljous vähäistenkin muutosten kanssa monesti rasitteena. Lisäksi erikoisempien toimintojen, kuten pikaviestivälityksen, lisääminen aiheutti projektin aikana monta epätoivon hetkeä.

Esittelyversiossa konfiguraatioon on lisätty kaksi SIP-käyttäjää: keskussolmu ja järjestelmän varsinainen käyttäjä. Käyttäjät rekisteröityvät päätelaitteillaan palvelimelle omilla käyttäjätunnus-salasanayhdistelmillään, jonka jälkeen muut SIP-toiminnot ovat käytössä. Asteriskin konfiguraatiossa on määritelty VoIP-puheluiden välityksen lisäksi pikaviestien välitys ja käyttäjien tilatietojen päivitys.

4.4.2 Tila- ja hälytystietokanta (MongoDB)

Tila- ja hälytystietokantaohjelmistona on dokumenttipohjainen MongoDB, johon tietoa tallennetaan JSON-dokumentteina. MongoDB on perinteiseen relaatiotietokantaan verrattuna erittäin nopea ottaa käyttöön, sillä se ei vaadi alustusvaiheessa tarkkaa mallia tallennettavan tiedon muodosta. Dokumentit tallennetaan kokoelmiin, joita voi perustaa tarpeen mukaan ilman muutoksia tietokannan ohjelmistoon. MongoDB voidaan tarvittaessa rakentaa hajautetuksi tietokannaksi. (21.)

Järjestelmän tila- ja hälytysdokumenteille on omat kokoelmansa tietokannassa. Tilatiedossa on mukana myös järjestelmän testaukseen käytettävää informaatiota Raspberryn resurssitilanteesta. Tiladokumentissa tietokantaan tallennetaan kaikilta järjestelmään liitetyiltä sensorisolmuilta

- aikaleima
- sensorisolmun yksilöllinen UUID
- Raspberryn resurssitilanne:
    - käytettävyysaika
    - muistin tila
    - prosessorikuormat 1, 5 ja 15 minuutin ajalta
- lämpötila
- suhteellinen ilmankosteusprosentti.

Kuvassa 5 on esimerkki tietokantaan tallennetusta JSON-dokumentista.

```
{"_id" : ObjectId("58a02f5de1d45d5eff81bb6f"),
"status" : "OK",
"Uptime" : NumberLong(1975633),
"MemTotalkB" : 947732,
"uuid" : "a699b6ce-3b10-401a-834f-d1cee3000001",
"FifteenMinLoad" : 0.01,
"timestamp" : "12-02-2017 11:48:13",
"humidity" : 19,
"OneMinLoad" : 0.13,
"MemFreekB" : 623816,
"MemAvailablekB" : 800000,
"FiveMinLoad" : 0.03,
"temperature" : 20}
```


KUVA 5. Tilamerkintä MongoDB-tietokannassa JSON-muodossa. "_id" on MongoDB:n muodostama tunnus dokumentille.

Hälytyskirjaukset tehdään niin ikään dokumenttimuodossa (kuva 6). Hälytyksestä kirjataan:

- Aikaleima
- Hälyttävän sensorisolmun UUID
- Hälyttävän sensorin numero
- Hälytyksen paikallinen järjestysnumero
- Tiedostonimi videotallenteelle (UUID + järjestysnumero)


```
{ "_id" : ObjectId("58a07871e1d45d61d4acb2cb"),
"uuid" : "a699b6ce-3b10-401a-834f-d1cee3000001",
"timestamp" : "12-02-2017 17:00:01",
"alarmId" : 2,
"recording" : "a699b6ce-3b10-401a-834f-d1cee3000001_2.mp4",
"sensor" : 1 }
```


KUVA 6. Hälytysmerkintä tietokannassa.

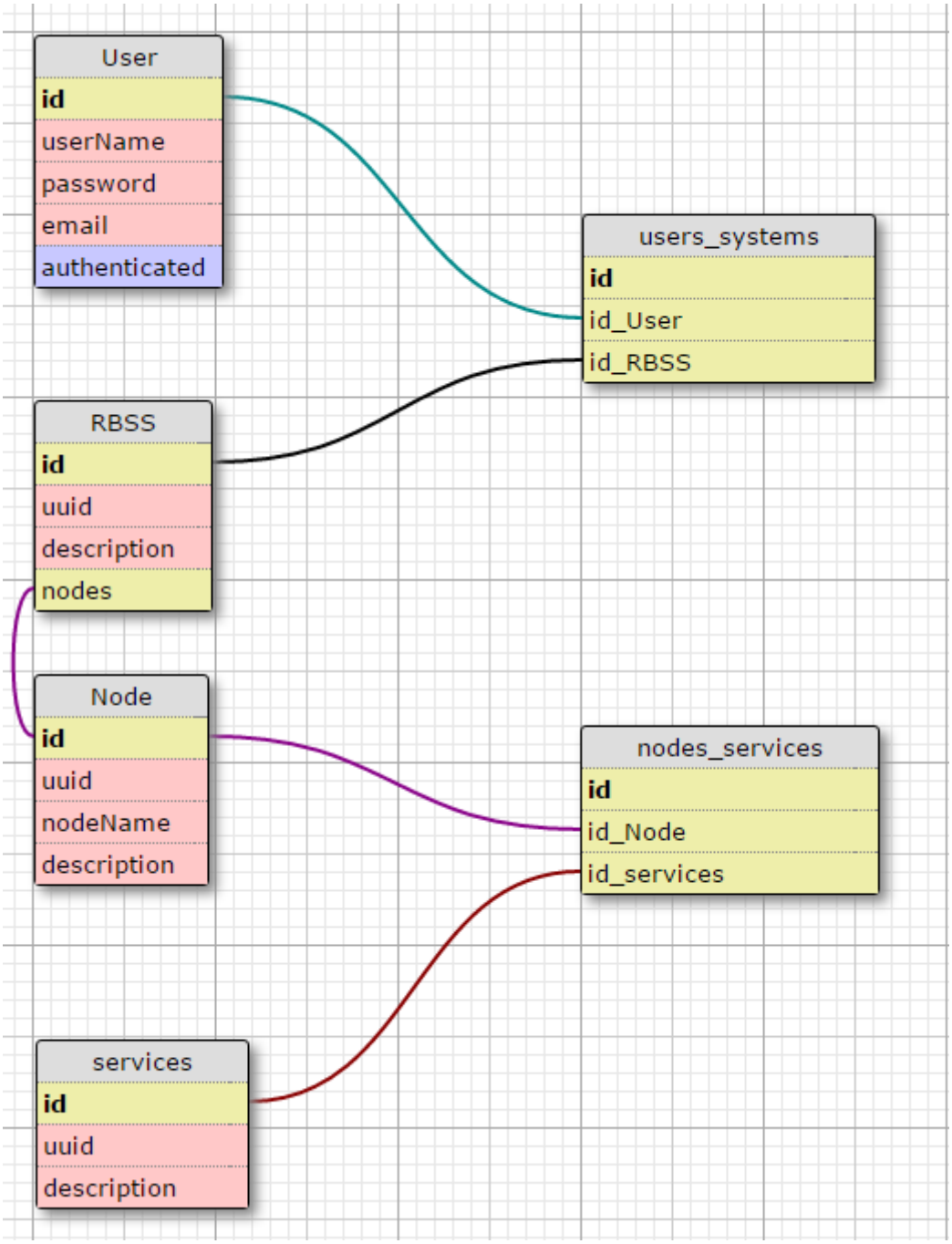
4.4.3 Ylläpitotietokanta (SQLite)

Ylläpitotietokannassa pidetään kirjaa järjestelmistä, niihin liitetyistä sensorisolmuista ja käyttäjistä. Esittelyversion ylläpitotietokantana on kevyt relaatiotietokanta SQLite. Tietokannan ja varsinaisen ohjelmiston välissä on Flask-ohjelmistokehyksen liitännäinen SQLAlchemy, jonka avulla varsinaiset tietokan-

taoperaatiot tehdään olioviittauksilla. Tällä menetelmällä SQLAlchemy hoitaa siis varsinaiset alimman tason tietokantaoperaatiot ja peittää ne luokkapohjaisella abstraktiolla, jolloin ohjelmoija voi käsitellä tietokannan rivejä kuten mitä tahansa ohjelman olioita.

Ylläpitotietokanta koostuu kuudesta taulusta (kuva 7):

- Käyttäjä (User)
- Järjestelmä (RBSS)
- Aputaulu käyttäjien ja järjestelmien yhdistämiseen monen suhde moneen -relaatiolla (users_systems)
- Sensorisolmu (Node)
- Palvelu (Service)
- Aputaulu sensorisolmujen ja palveluiden yhdistämiseen monen suhde moneen -relaatiolla (nodes_services).



KUVA 7. Ylläpitotietokannan malli.

Aputaulujen tarkoitus on yhdistää taulut, joilla on keskenään monen suhde moneen -relaatio. Tämän tietokannan tapauksessa esimerkiksi yhdellä käyttäjällä voi olla oikeus useaan järjestelmään ja yhdellä järjestelmällä voi olla useita käyttäjiä.

Käyttäjä

Järjestelmään lisättävästä käyttäjästä tallennetaan tietokantaan seuraavat tiedot:

- yksilöllinen pääavain (id)
- käyttäjänimi (userName)
- salasana bcrypt-salattuna (password)
- sähköpostiosoite (email)
- totuusarvomuuttuja sisäänkirjautumiselle (authenticated).

Käyttäjien salasanoja ei hyvän tavan mukaisesti (22) tallenneta selkokielisenä, vaan niistä muodostetaan tiiviste bcrypt-funktiolla, ennen kuin salasanat tallennetaan tietokantaan. Kirjautumistapahtuman yhteydessä käyttäjän syöttämä salasana muutetaan tiivisteeksi, jota verrataan tietokantaan tallennettuun tiivisteeseen. Selkokielistä salasanaa ei vertailla missään vaiheessa.

Järjestelmä

Järjestelmästä tallennetaan

- yksilöllinen pääavain (id)
- yksilöllinen UUID (uuid)
- sanallinen kuvaus (description)
- järjestelmään liitetyt sensorisolmut.

Järjestelmän UUID on pohjana sensorisolmujen ja palveluiden UUID-tunnisteille.

Sensorisolmu

Sensorisolmusta tietokantaan tallennetaan

- yksilöllinen pääavain (id)
- yksilöllinen UUID (uuid)
- kuvaava nimi (nodeName)
- sanallinen kuvaus (description).

Palvelu

Palveluista tietokantaan tallennetaan

- yksilöllinen pääavain (id)
- yksilöllinen UUID (uuid)
- sanallinen kuvaus (description).

4.4.4 Palvelinohjelmisto

Käyttöliittymän palvelinpuolen ohjelmisto on kirjoitettu Flask-ohjelmistokehystä hyödyntäen. Internet-palvelun toteutus oli projektin aiheista vähiten tuttu, ja tarkoitukseen piti löytää ratkaisu, joka olisi riittävän yksinkertainen toteuttaa ja tarjoaisi siltä toivotut ominaisuudet. Flask täytti kriteerit tarjoten riittävästi valmiita moduuleita eri toiminnoille ja Python-kielisen ohjelmointiympäristön.

Flask-ohjelmiston oleellisin moduuli on *Views. Views*-moduuli on vastuussa sivuston eri näkymien, kuten kirjautumissivun muodostamisesta ja dynaamisesta sisällöstä. Sivuston näkymiin viitataan reittinä eli osoitteena ohjelman /-juuresta johdettuna. Esimerkiksi kirjautumissivun reitti on ”/login”. Jokaiselle reitille on *Views*-moduulissa määritelty Python-funktio, joka muodostaa ja palauttaa selaimeen HTML-sivun halutuilla parametreilla. Parametrina sivulle tuodaan esimerkiksi tietokannasta haettua dataa.

Sivuston kirjautumista vaativat näkymät merkitään Flask-koodissa *@login_required*-tunnisteella ennen näkymän palauttavan funktion koodia. Käyttäjän tunnistamista vaativan sivuston rakentaminen oli Flaskin valmiiden moduulien ansiosta erittäin yksinkertaista ja loogista.

Myös hälytys- ja tilatietokannan luku- ja kirjoitusoperaatiot ovat *Views*-moduulissa. Lukuoperaatioita käytetään tietojen hakemiseen tietokannasta ja ne tuodaan HTML-sivun palautuksen yhteydessä selaimessa ajettavan JavaScript-ohjelman käsiteltäviksi. Tietokantojen lukemia käsittelee Google Charts API, jolla niistä muodostetaan taulukoita ja kaavioita. Kirjoitusoperaatiota käytetään hälytyksen poistamiseen tietokannasta. Poistofunktion kutsu tulee selaimelta, kun käyttäjä valitsee poistettavan hälytysrivin hälytyssivulla näytettä-

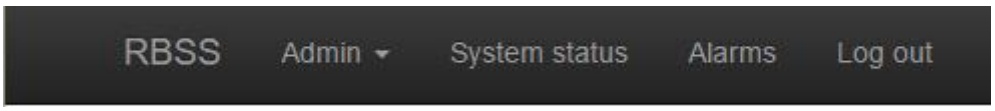
västä taulukosta. Tietokantaoperaation yhteydessä myös hälytykseen liitetty videotiedosto poistetaan palvelimelta.

Views-moduulin ohella *Models* on toinen ohjelman tärkeimmistä moduuleista. *Models* on linkkinä ohjelman ja ylläpitotietokannan välillä. Flaskin SQL-Alchemy-moduulia hyödyntävä *Models* muuttaa tietokannan taulut ohjelmassa käsiteltäviksi luokiksi, joiden olioihin viittaamalla tietokannan rivejä voidaan lukea ja manipuloida. Tällä menetelmällä tietokannan käsittely ohjelmaa kirjoittaessa on luontevaa ja lopullinen ohjelma on luettavuudeltaan parempi kuin suoraan tietokantaan tehtävillä operaatioilla.

4.4.5 Käyttöliittymä

Käyttöliittymä rakentuu Flaskin Views-moduulin palauttamista HTML-malleista, jotka muodostetaan dynaamisesti tarpeen mukaan. Sivuilla on myös jonkin verran selaimessa ajettavia JavaScript-toimintoja, joista merkittävimpänä on tietokantadatan visuaalisesta taulukoinnista vastaava Google Charts API. Ulkoasu on muodostettu Bootstrap-mallien avulla.

Käyttäjälle näkyvä sivu muodostetaan kahdesta osasta: kaikille sivuille yhteisestä ylävalikosta ja varsinaisesta sisältösivusta valikon alapuolella. Flask muodostaa molemmat dynaamisesti tilanteen mukaan. Esimerkiksi kirjautuneelle käyttäjälle palautetaan erilainen ylävalikko (kuva 8) kuin kirjautumattomalle (kuva 9).



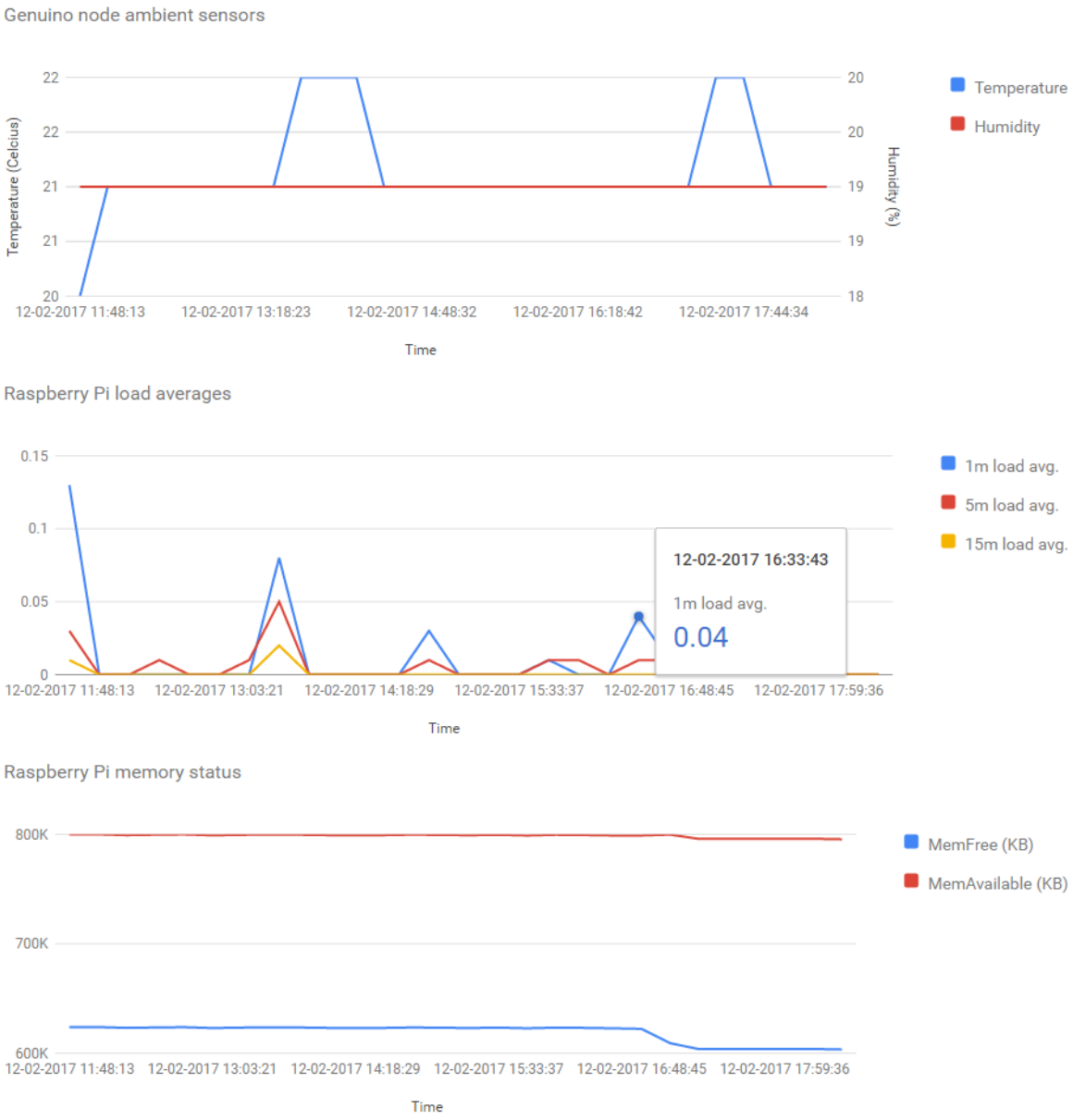
KUVA 8. Ylävalikko kirjautuneelle Admin-käyttäjälle.

KUVA 9. Ylävalikko ennen sisäänkirjautumista.

Google Charts API

Käyttöliittymässä esiintyvien taulukoiden muodostamiseen on hyödynnetty Googlen Charts-rajapintaa, jossa on kattava valikoima taulukoita ja kaavioita datan visualisointiin (23).

Charts-rajapinnan avulla käyttöliittymässä esitetään sensoreilta kerätyn lämpötila- ja ilmankosteusdatan lisäksi Raspberryn resurssitilanne ja hälytystietokannan sisältö. Kaaviot on muodostettu tietokantakyselyiden pohjalta, ja niiden sisältämistä arvoista saa tarkempaa tietoa viemällä kursorin kuvaajan tai selitteiden päälle (kuva 10). Hälytysrivejä näyttävä taulukko sisältää linkit videotallenteisiin ja hälytyksien poistamisfunktioon.



KUVA 10. Käyttöliittymän kaavionäkymä. Prosessorikuorman (Raspberry Pi load averages) kohdalla on näkyvissä taulukon reaktiivisuus, kun kursori on vietty kuvaajan päälle.

Bootstrap

Käyttöliittymän ulkoasun siistimiseen käytettiin Bootstrap-edustakehystä, joka on avoimen lähdekoodin kokoelma käyttöliittymän ulkoasukomponentteja CSS- ja HTML-pohjineen. Bootstrap-malli tekee sivusta samalla automaattisesti mukautuvan, jolloin esimerkiksi puhelimen pienemmällä näytöllä sivun ylävalikko piilotetaan valikkopainikkeen taakse (kuva 11).

KUVA 11. Bootstrapin tiivistetty yläpalkkinäkymä.

5 KÄYTTÖTAPAUSKUVAUKSET

Järjestelmälle on mahdollista kuvata useita käyttötapauksia, esimerkiksi asentajan ja loppukäyttäjän näkökulmista. Tässä käyttötapauskuvauksessa keskitytään vain loppukäyttäjän käyttötapaukseen. Kaikkia käyttötapauksessa kuvattuja ominaisuuksia ei esittelyversiossa ole toteutettu, mutta suurin osa on toteutettu vähintäänkin simuloimalla lopullista toimintaa.

5.1 Hälytyksen asetustapahtuma

Käyttäjä soittaa järjestelmään SIP-puhelusovelluksella (esimerkiksi CSipSimple Androidilla) valitsemalla järjestelmän SIP-URI:n, joka on muotoa *käyttäjänimi@palvelinosoite.* Järjestelmä käsittelee saapuvan puhelun ja vertaa soittajan SIP-URI:tä listaan sallituista käyttäjistä. Mikäli puhelu tulee sallitulta käyttäjältä, järjestelmän tila muutetaan käänteiseksi (inaktiivinen/aktiivinen) ja uusi tila ilmoitetaan käyttäjälle SIP-pikaviestillä (kuva 12).

Hälytyksen poiskytkentä ei katkaise tilatiedon keruuta, vaan lämpötila- ja ilmankosteustiedot ja keskussolmun resurssit kirjoitetaan tietokantaan myös hälytysjärjestelmän ollessa poiskytkettynä. Myös lämpötilan muutokseen perustuva hälytystoiminto on päällä muiden sensoreiden ollessa poiskytkettynä.

5.2 Järjestelmän tilatarkastelutapahtuma

Käyttäjä kirjautuu selainpohjaiseen ylläpitokäyttöliittymään omilla käyttäjätunnuksillaan ja hyväksytyn kirjautumisen jälkeen pääsee tarkastelemaan tunnukselleen liitettyjen valvontajärjestelmien tilatietoja.

Tilatietonäkymässä käyttäjä voi tarkastella lämpötilojen, ilmankosteuksien ja järjestelmäresurssien tiloista viidentoista minuutin välein kerättyä dataa kuvaajamuodossa.

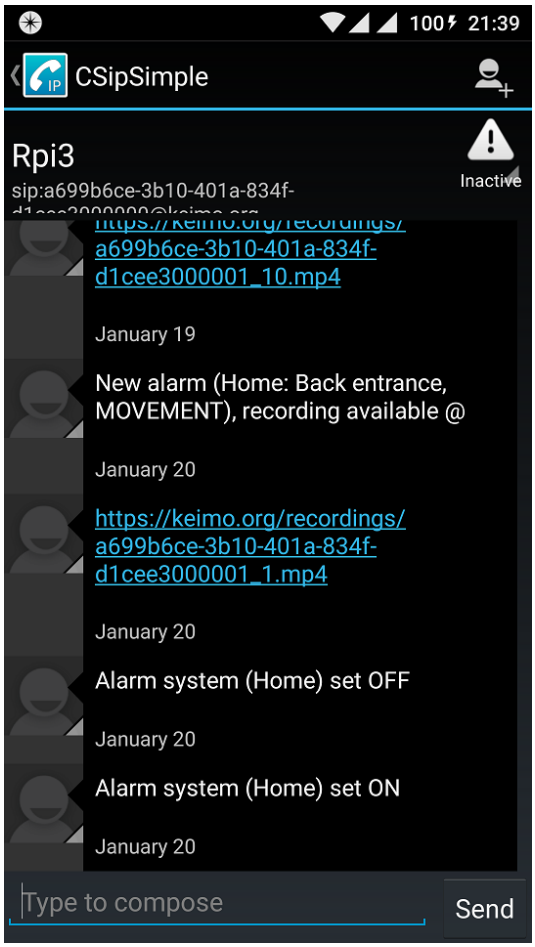
Hälytysnäkymässä käyttäjä näkee taulukkona valitun järjestelmän hälytyshistorian. Taulukossa on kerrottu hälytyksen laukaissut sensorisolmu, tarkka sensori, aikaleima ja linkki videotallenteeseen. Hälytyksien poistoon taulukossa on kuvake, jota painamalla käyttäjältä varmistetaan hälytysrivin ja videotallenteen

poisto. Mikäli käyttäjä hyväksyy operaation, poistetaan hälytysmerkintä tietokannasta ja hälytykseen liittyvä videotiedosto palvelimen tiedostojärjestelmästä.

5.3 Hälytystapahtuma

Hälytyksen havaittuaan järjestelmä soittaa pääkäyttäjälle SIP-puhelun ilmoitukseksi havaitusta hälytyksestä. Samaan aikaan käynnistetään videotallennus, jonka päätyttyä videotiedosto siirretään palvelimelle ja tietokantaan kirjataan merkintä hälytyksestä. Kaikille järjestelmän käyttäjille lähetetään lopuksi kaksi SIP-pikaviestiä, joista ensimmäisessä on hälytyksen tiedot ja toisessa linkki videotiedostoon palvelimella (kuva 12).

Hälytysviestin teksti on muodostettu järjestelmän osille annettujen, kuvaavien nimien pohjalta. Kuvan 12 tapauksessa järjestelmäkokonaisuudelle on annettu kuvaus ”Home” (koti), sensorisolmulle ”Back entrance” (takaovi) ja PIR-sensorille ”Movement” (liikettä).



KUVA 12. CSipSimplen SIP-pikaviestinäkymä. Tammikuun 20. päivä saapuneista viesteistä kaksi ensimmäistä liittyvät hälytystapahtumaan ja kaksi viimeistä hälytyksen kytkentään pois tai päälle.

6 JATKOKEHITYSMAHDOLLISUUDET

Projektin aikana pois jätettyjen ominaisuuksien lisäksi jatkokehitysmahdollisuuksia on erittäin paljon. Suurin osa näistä liittyy konseptin laajentamiseen palveluksi ja edelleen liiketoiminnaksi.

Koska työssä rakennettiin vain konseptijärjestelmä, on ohjelmistossa todella paljon parannettavaa. Ohjelmisto vaatisi uudelleensuunnittelua ja järkeistämistä, mikä projektin aikana hankitun kokemuksen myötä olisi aiempaa helpompaa.

6.1 Laitekannan suunnittelu

Projektin esittelylaitteisto rakennettiin kokonaisuudessaan kuluttajille suunnatuilla harrastekomponenteilla, jotka eivät vastaa massatuotantoon tarkoitetun järjestelmän tarpeita. Esimerkiksi Genuino-alustan käyttämää Curie-moduulia voisi hyvin hyödyntää solmun suunnittelussa, mutta piirilevyratkaisu olisi syytä rakentaa tarpeeseen sopivaksi ja helposti laajennettavaksi esimerkiksi sensorivalikoimaltaan.

6.2 Sensorivalikoiman laajennettavuus

Sensorisolmun luonteen vuoksi sensoritarve määräytyy täysin lopullisen sijoituspaikan mukaan. Tästä syystä valikoiman tulisi olla tarpeiden mukaan valittavissa, jolloin esimerkiksi sisäänkäyntiä valvova solmu tunnistaisi myös oven aukeamisen. Tämä olisi yksinkertaisimmillaan toteutettavissa Hall-anturilla ja magneetilla.

6.3 Käyttöönotto ja ylläpito

Mikäli järjestelmää ajatellaan laajemmin levitettäväksi, pitää myös käyttöönoton olla asentajalleen yksinkertainen ja mahdollisimman vaivaton. Yksi käyttöönottoon ja ylläpitoon liittyvä kehitysidea tuli toimeksiantajan suunnalta: konfiguraatiotiedostojen etälataus esimerkiksi JSON-muodossa.

Ylläpidon näkökulmasta oleellisia kehityskohteita ovat esimerkiksi uusien laitteiden, järjestelmien ja käyttäjien lisäys- ja muokkaustoiminnot. Nämä tietokanta-

operaatiot on tarkoituksella jätetty esittelyversion käyttöliittymästä pois tietoturvasyistä. Esittelyversiossa ei ole TLS/SSL-salausta käytössä, joten kaikki liikenne palvelimen ja selaimen välillä liikkuu selkokielisenä.

6.4 Käyttäjäkokemus

Loppukäyttäjän käyttömukavuutta voidaan lisätä myös monin tavoin. Esittelyversion mobiilipäätelaitteen ohjelmistona toimi tavallinen SIP-puhelusovellus, mutta jatkokehityksenä myös oman mobiilisovelluksen suunnittelu voisi tulla kysymykseen. Sama sovellus tarjoaisi kevyemmän version loppukäyttäjän hallintamahdollisuuksista ja esimerkiksi hälytyksien kytkennän päälle ja pois.

7 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnitella konsepti ja rakentaa esittelyversio tilanvalvontajärjestelmästä. Työssä oli suunniteltava sekä laitteisto- että ohjelmisto-osuus, ja kaikkien osapuolien yhteinen näkemys oli, että aihe oli lähtökohtaisesti erittäin laaja kokonaisuus. Heti alkuvaiheessa tulikin selväksi, ettei aikataulu mahdollistanut kaikkien ominaisuuksien toteuttamista. Lopputuloksena syntyneen kokonaisuuden voidaan katsoa olevan kompromissi resurssin, osaamisen ja käytettävissä olevan laitteiston kesken.

Jälkikäteen tarkasteltuna moni asia projektissa tulisi tehtyä toisin, jos sitä alettaisiin tehdä uudelleen nykyisillä tiedoilla. Projekti sisälsi monta ennestään tuntematonta tekniikkaa, joiden opiskelun vuoksi työ eteni kokeilemalla ja käytännön kautta sopivia menetelmiä etsimällä. Tämä lähestymistapa aiheutti projektin lopputulokseen suhteutettuna enemmän työtunteja, kuin voisi ulkopuolisen tarkastelun perusteella olettaa. Jälkikäteen arvioituna työmäärän ja lopputuloksen suhde voi vaikuttaa jopa tehottomalta, mutta henkilökohtaisen kehittymisen kannalta käytettyjen tuntien vaikutus on kuitenkin ollut arvokas.

Oppimisprojektina tarkastellessa aihe oli erinomainen ja lähtökohdat huomioon ottaen lopputulokseen voi olla kokonaisuutena tyytyväinen. Vaikka osa ominaisuuksista jäikin toteuttamatta, esittelyversiossa ovat silti järjestelmän tärkeimmät toiminnot ja se osoittaa, että konseptilla on runsaasti jatkokehitysmahdollisuuksia. Konseptista tuotteeksi asti työstäminen edellyttäisi kuitenkin mittavaa työpanosta ja sekä laitteiston että ohjelmiston uudelleensuunnittelua.

LÄHTEET

1. Big Dan, 2015. The, Very Frustrating, NRF24L01+. Saatavissa: https://bigdanzblog.wordpress.com/2015/02/21/the-very-frustrating-nrf24l01/. Hakupäivä 2.2.2017.

2. King, Terry 2011. RF24L01 2.4GHz Radio/Wireless Transceivers How-To. ArduinoInfo. Saatavissa: https://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo#PP. Hakupäivä 2.2.2017.

3. Back, Andrew 2016. Say Hello to Genuino 101. Saatavissa: https://www.rs-online.com/designspark/say-hello-to-genuino-101. Hakupäivä 6.2.2017.

4. Arduino board comparison. 2016. Arduino. Saatavissa: https://www.arduino.cc/en/Products/Compare. Hakupäivä 2.2.2017.

5. Intel Curie Module Datasheet. 2016. Intel. Saatavissa: http://www.intel.com/content/dam/support/us/en/documents/boardsandkits/curie/Intel_Curie_Module_Datasheet_V1.21.pdf. Hakupäivä 20.1.2017.

6. Lextrait, Thomas 2016. Arduino: Power consumption compared. Saatavissa: https://tlextrait.svbtle.com/arduino-power-consumption-compared. Hakupäivä 6.2.2017.

7. DHT11 Humidity & Temperature Sensor. 2010. D-Robotics UK. Saatavissa: http://www.micropik.com/PDF/dht11.pdf. Hakupäivä 2.2.2017.

8. Fried, Limor 2014. PIR Motion Sensor: Overview. Adafruit. Saatavissa: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview. Hakupäivä 6.2.2017.

9. Raspberry Pi 3 Model B. 2016. The Raspberry Pi Foundation. Saatavissa: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Hakupäivä 6.2.2017.

10. HD WEBCAM C270. 2017. Logitech. Saatavissa: http://www.logitech.com/fi-fi/product/hd-webcam-c270. Hakupäivä 6.2.2017.

11. How It Works: Bluetooth Low Energy. 2017. Bluetooth SIG. Saatavissa: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy. Hakupäivä: 8.2.2017.

12. Nilsson, Rolf – Saltzstein, Bill 2012. Bluetooth Low Energy vs. Classic Bluetooth: Choose the Best Wireless Technology For Your Application. Medical Electronics Design. Saatavissa: http://www.medicalelectronicsdesign.com/article/bluetooth-low-energy-vs-classic-bluetooth-choose-best-wireless-technology-your-application. Hakupäivä 8.2.2017.

13. CurieBLE Library. 2017. Arduino. Saatavissa: https://www.arduino.cc/en/Reference/CurieBLE. Hakupäivä 8.2.2017.

14. Rosenberg, J. et. al. 2002. RFC 3261 SIP: Session Initiation Protocol. The Internet Engineering Task Force (IETF). Saatavissa: https://tools.ietf.org/html/rfc3261. Hakupäivä 8.2.2017.

15. Rosenberg, J 2004. RFC 3856 SIP Presence. The Internet Engineering Task Force (IETF). Saatavissa: https://tools.ietf.org/html/rfc3856. Hakupäivä: 8.2.2017.

16. Campbell, B. et. al. 2002. RFC 3428 SIP Message Extension. The Internet Engineering Task Force (IETF). Saatavissa: https://www.ietf.org/rfc/rfc3428.txt. Hakupäivä 8.2.2017.

17. Corelibs-arduino101. Intel Open Source Technology Center. Saatavissa: https://github.com/01org?utf8=%E2%9C%93&q=arduino&type=&language=. Hakupäivä 16.2.2017.

18. Harvey, Ian. Bluepy. Saatavissa: https://github.com/IanHarvey/bluepy. Hakupäivä 16.2.2017.

19. PJSUA Python Module. 2017. PJSIP. Saatavissa: https://trac.pjsip.org/repos/wiki/Python_SIP_Tutorial. Hakupäivä 2.3.2017.

20. Asterisk. 2017. Digium Inc. Saatavissa: http://www.asterisk.org/. Hakupäivä 16.2.2017.

21. MongoDB Architecture. 2017. MongoDB. Saatavissa: https://www.mongodb.com/mongodb-architecture. Hakupäivä 8.2.2017.

22. Kovanen, Pasi 2013. Salasanojen tallentaminen. 67% - Ohjelmistokehityksen ostaminen. Vincit /blog. Saatavissa: https://www.vincit.fi/blog/salasanojen-tallentaminen/. Hakupäivä 13.2.2017.

23. Google Charts. 2017. Google Inc. Saatavissa: https://developers.google.com/chart/. Hakupäivä 13.2.2017.

TOIMEKSIANNON KAAVIOKUVA

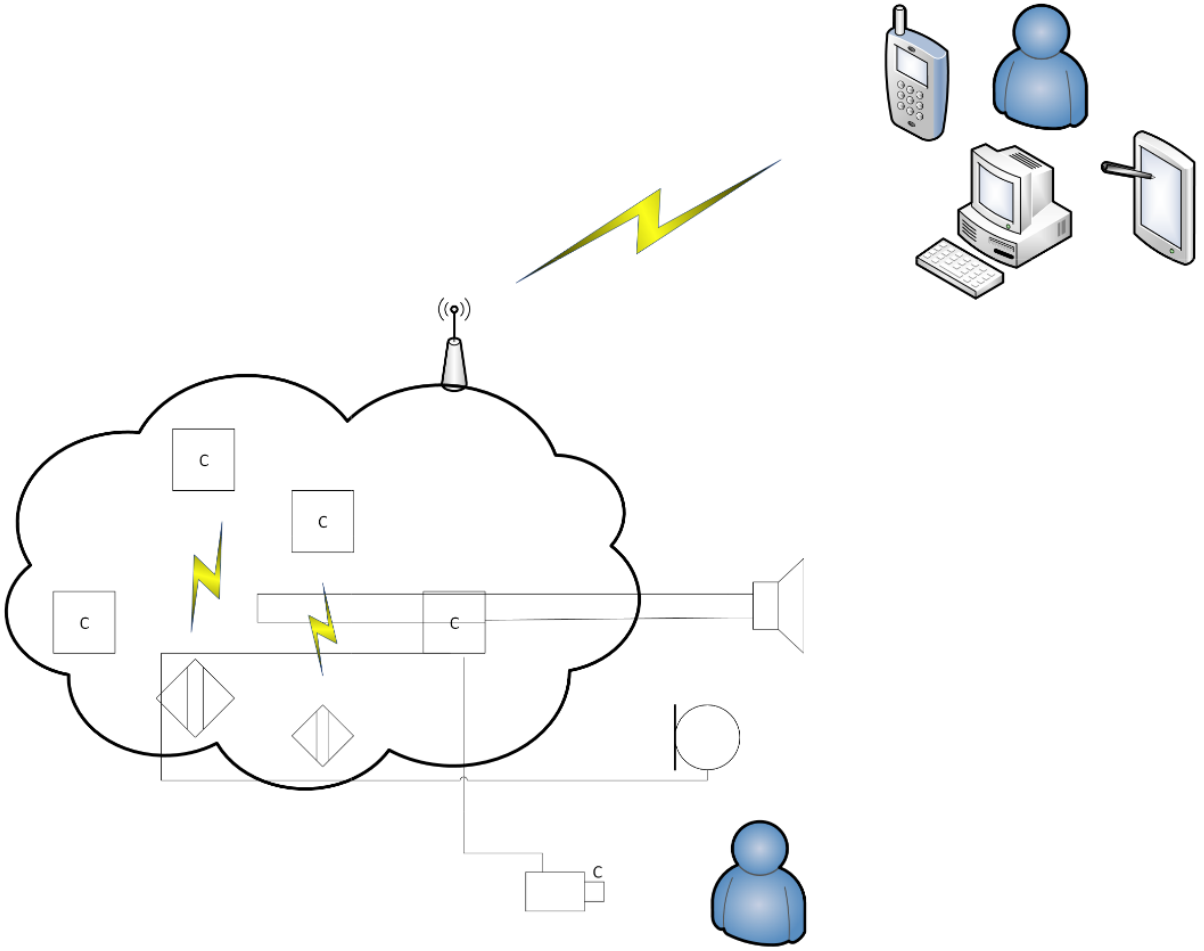
LIITE 1

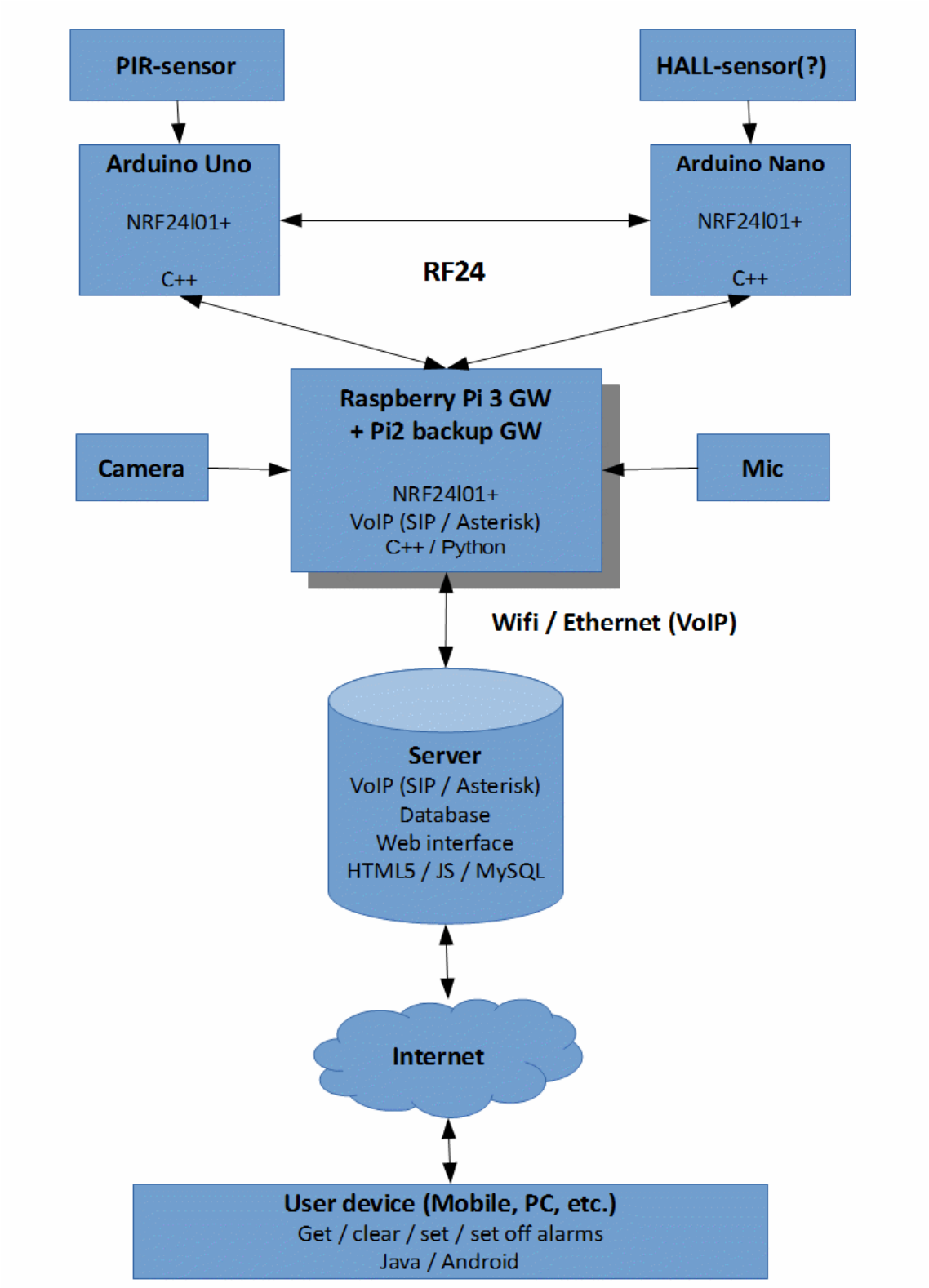
Remote burglar surveillance system (PoC):

Use cases:
- Some sensor detects something
=> sensor makes VoIP call to remote server PC
=> server starts voice and video recording for a “minute”
=> systemsSend alarm to “user”
=> User can listen/download recorded audio/video from web page
=> User can check alarm

Technologies etc:
- VoIP, PJSIP, Asterisk
- Networking: WiFi, BT, ...
- HW: e.g. Raspberry/Odroid/Arduino, camera, mic, sensor,...
- Linux platform, GIT, Web IF
- Python, C/C++, JS, HTML5, ...

ToDo:
- Concept
- Architecture description
- Basic implementation and demo
- Done! ☺




PIR-sensor
HALL-sensor(?)
Arduino Uno
NRF24l01+
C++
Arduino Nano
NRF24l01+
C++
RF24
Raspberry Pi 3 GW
+ Pi2 backup GW
NRF24l01+
VoIP (SIP / Asterisk)
C++ / Python
Camera
Mic
Wifi / Ethernet (VoIP)
Server
VoIP (SIP / Asterisk)
Database
Web interface
HTML5 / JS / MySQL
Internet
User device (Mobile, PC, etc.)
Get / clear / set / set off alarms
Java / Android