Haaga-Helia
ammattikorkeakoulu Oy

Prototyyppi haulikkoradan kiekonheittimien ohjaukseen

Jarno Wermundsen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2020

Tiivistelmä

Haaga-Helia ammattikorkeakoulu Oy

Tekijä(t)
Jarno Wermundsen

Koulutusohjelma
Tietojenkäsittely

Raportin/Opinnäytetyön nimi
Prototyyppi haulikkoammunnan ohjaukseen

Sivu- ja liitesivumäärä
35 + 22

Tämä opinnäytetyöprojekti käynnistettiin, koska Kauhajoen Sotkan haulikkoampumaradan käyttäjät toivoivat compak sporting -radalle harjoittelumuotoa, jossa kiekko lähtee satunnaisesta heittimestä. Työn ammunnanohjausjärjestelmä päätettiin toteuttaa langatonta viestintätekniikkaa käyttäen siten, että kierroksen ohjaaminen on mahdollisimman helppoa. Työ rajattiin protoversion kokoamiseen ja ohjelmointiin. Opinnäyteprojektin ulkopuolelle jätettiin laitteen kotelointi, käyttöympäristöön asentaminen ja käyttäjätutkimukset.

Protoversio koostuu kahdesta laitteesta, pääohjausyksiköstä ja heitinohjausyksiköstä. Heitinohjausyksikkö odottaa pääohjausyksiköltä itselleen kuuluvaa viestiä, antaakseen sitten kiekonheittimelle laukaisukäskyn. Pääohjausyksiköllä voidaan ohjata tavallista compak sporting -ammuntaa, satunnaisesta heittimisestä lähtevää harjoitusmuotoa ja lähettää näyttökiekkoja yksittäisistä heittimistä. Pääohjausyksikköön voidaan lisäksi suunnitella uusia compak sporting -kierroksia pääohjausyksikön ylläpitämässä WLAN-verkossa selaimen kautta.

Työ suoritettiin kevätlukukauden (tammi-toukokuu) aikana vuonna 2020.Työhön luotiin vesiputousmallin mukaiset määritelmät suurimmaksi osin compak sporting -sääntöjen pohjalta. Työn lopullinen tekniikka jätettiin, kuitenkin alkuvaiheessa avoimeksi ja tekniikoita kokeiltiin protovaiheelle tyypillisellä tavalla. Projektin ohjelmointi suoritettiin pienin mahdollinen askel kerrallaan.

Työn laitteistoksi valikoitui lopulta mikrokontrollerit ja langattomaksi viestintätekniikaksi LoRa-yhteys. Työssä saavutettiin tavoitteet ja aikaansaatua prototyyppiä aiotaan kehittää vielä eteenpäin.

Asiasanat
Ohjausjärjestelmät, langaton tiedonsiirto, mikrokontrollerit, haulikkoammunta

Sisällys

1 Johdanto

Tässä opinnäytetyöprojektissa tavoitteena on koota ja ohjelmoida protoversio langattomasta ohjausjärjestelmästä, joka soveltuu Kauhajoella Sotkan haulikkoradan compak sporting -ammuntaan. Compak sporting -ammunta on tarkemmin esiteltynä luvussa 1.2. Työssä toimeksiantajana on HM Service, jonka yrittäjä Heikki Marttunen on suunnitellut ja rakentanut radan nykyisen ohjausjärjestelmän. Projekti käynnistetään, koska osa Kauhajoen Sotkan haulikkoradan käyttäjistä on toivonut compak sporting -radalle harjoittelumuotoa, jossa kiekko lähtee satunnaisesta heittimestä.

Projekti oli alun perin suunniteltu toteutettavaksi käyttäen ohjelmoitavaa logiikkaa, mutta kuultuani projektista ehdotin käytettäväksi mikrokontrollereita ja langatonta viestintätekniikkaa. Langattomaksi viestinä teknologiaksi valikoitui kokeilujen jälkeen LoRa. Samalla ideoitiin, että olisi hyvä mahdollistaa ampumakierroksen suunnittelu uudelleen selaimen kautta. Jo alustavien laskelmien perusteella vaihtoehto näytti myös edulliselta toteuttaa. Projektin tavoite ja rajaus on esitelty tarkemmin luvussa 1.1.

Compak Sportingiin suunniteltuja järjestelmiä näyttää olevan markkinoilla hyvin vähän, joten järjestelmälle voisi olla kysyntää myös muilla haulikkoradoilla. Opinnäytetyössä valmistunut prototyyppi muuttaa Sotkan ampumaradalla toimintaa siten, että ammunnan ohjaajan ei tarvitse tietää, mistä heittimestä seuraava kiekko ammutaan. Järjestelmä pitää huolen kierroksen kulusta ja ammunnan ohjaajan työ helpottuu tämän vuoksi merkittävästi. Laitteisto voidaan pienin ohjelmallisin muutoksin myös muunnella muihin ampumalajeihin sopivaksi tai laajentaa yleisimpiin ammuntoihin sopivaksi yleisohjaimeksi.

Tässä opinnäytetyöraportissa on ensin alla esitelty projektin tavoite ja rajaus (luku 1.1), sekä keskeiset käsiteet (luku 1.2). Luvussa 1.3 on kerrottu työhön liittyvistä Compak sporting -ammunnan säännöistä. Laitteiden ja teknologian valikoitumisesta projektiin kerrotaan luvussa 2. Luku 3 toimii työn tietoperustana ja esittelee käytetyt laitteet, ohjelmistot ja ohjelmointikirjastot. Laitteiden tekniikan kuvaus on luvussa 4 ja ohjelmakoodin kuvaus luvussa 5. Luvussa 6 kerrotaan projektin etenemisestä ja haasteista, kun luvussa 7 käsitellään johtopäätökset työstä ja työn aikana tulleet uudet kehitysideat.

1.1 Projektin tavoite ja rajaus

Projektin tavoitteena on koota ja ohjelmoida langattomasti toimiva prototyyppi haulikkoradan kiekonheittimien ohjaukseen. Konkreettisena tuotos tavoitteena on heitinohjausyk-

sikkö ja pääohjausyksikkö. Heitinohjausyksikkö odottaa pääohjausyksiköltä itselleen kuuluvaa viestiä ja viestin saavuttua relettä käyttäen antaa kiekonheittimelle laukaisukäskyn. Pääohjausyksiköllä ammunnan ohjaaja hallitsee ammunnan kulkua myös tyypillisissä vikatilanteissa. Pääohjausyksikössä tulee olla vaihdettavissa ja hallittavissa oleva vakiokierros, sekä harjoitteluun soveltuva satunnaiskierros ja mahdollisuus kiekkojen näyttölaukaisuun. Ohjelmallisena tuotoksena on kiekonheitintä ohjaavan yksikön ohjelmakoodi, sekä pääohjausyksikön ohjelmakoodi. Tuotosten lisäksi kirjoitetaan tämä raportti, joka sisältää tietoperustan ja kuvaa prototyypin toiminnallisuuden.

Työstä on rajattu pois laitteiden lopulliset koteloinnit ja käyttöympäristön sähkötyöt eli laitteiden lopullinen asentaminen Sotkan ampumaradalle. Testausta ei tällä kertaa päästy COVID-19 -viruksen takia asetettujen rajoitteiden vuoksi suorittamaan käyttöympäristössä, vaikka se alun perin oli tarkoitus. Selaimessa toimivaan osuuteen ei tehdä käyttöliittymäsuunnittelua enempää kuin järkevän toimimisen kannalta on tarpeen. Käyttäjätutkimusta ei tämän projektin yhteydessä suoriteta.

1.2 Compak sporting -ammunta

Compak sporting on haulikkoammuntalaji, jossa haulikolla yritetään osua lentävään savikiekkoon. Savikiekot voidaan lähettää oikealta vasemmalle, vasemmalta oikealle tai suoraan poispäin ampujasta. Lisäksi voidaan käyttää suoraan ylös nousevia kiekkoja ja jäniskiekkoja. Compak sporting -radan lentokulmia ja kiekkoja muutetaan säännöllisesti. Compak sporting -rataan kuuluu kuusi heitintä ja viisi ammuntapaikkaa. Ammuntapaikat ovat suorassa linjassa noin 4 metrin päässä toisistaan. (Ampumaurheiluliitto 2020.)

Lajiin löytyy suomenkieliset säännöt Ampumaurheiluliiton kotisivuilta. Säännöistä käydään tarkemmin tässä luvussa vain kohdat, jotka ovat merkittäviä lähetyslaitteen ja sen ohjelmoinnin kannalta. Laukaisulaitteesta säännöissä (Ampumaurheiluliitto 2017, 86) kerrotaan kohdassa C.2.3 seuraavasti ”Heittimet voidaan laukaista manuaalisesti, kaukosäätimellä ja akustista järjestelmää käyttäen. Kiekot heitetään kilpailijan pyynnöstä 0..3 sekunnin viiveellä”.

Ammunnan aikana voidaan lähettää yksittäiskiekkoja tai kaksoiskiekkoja. Kaksoiskiekkoja on kahdenlaisia: kaksoiskiekko laukauksesta ja yhtäaikainen kaksoiskiekko. Kaksoiskiekko laukauksesta ohjataan siten, että ampuja pyytää ensimmäisen kiekon ja toinen kiekko lähetetään 0-3 sekunnin kuluessa, kun ensimmäistä on ammuttu. Kaksoiskiekko laukauksesta voidaan lähettää myös samasta heittimestä. Yhtäaikaisen kaksoiskiekon

molemmat kiekot lähetetään samanaikaisesti eri heittimistä kilpailijan pyynnöstä. (Ampumaurheiluliitto 2017, 87-88.)

Yksi kierros koostuu 25 kiekosta, mutta kilpailussa voi olla useita kierroksia. Kierroksen aikana jokaiselta viideltä paikalta ammutaan yhteensä viisi kiekkoa. Kiekot ammutaan joko viitenä yksittäiskiekkona, kolmena yksittäiskiekkona ja yhtenä kaksoiskiekkona tai yhtenä yksittäiskiekkona ja kahtena kaksoiskiekkona. (Ampumaurheiluliitto 2017, 92.)

Ampumaohjelmassa samalla radalla kaksoiskiekkojen on oltava joko laukauksesta lähteviä tai yhtäaikaisia. Yksittäiskiekot voivat olla vapaassa järjestyksessä, mutta kaksoiskiekkojen ensimmäisen kiekon tulee olla sama kuin edellisen paikan viimeisin kiekko. Mikäli ampujia on vähemmän kuin kuusi, ja väliin jää tyhjä paikka, on tyhjän paikan viimeinen kiekko näytettävä seuraavalle ampujalle. (Ampumaurheiluliitto 2017, 93-94.)

Kilpailu voidaan käydä läpi kuuden kilpailijan erinä tai jonojärjestelmällä. Ensimmäisellä kierroksella ensimmäiset viisi kilpailijaa asettuvat numerojärjestyksessä paikoille 1-5. Kuudes ampuja odottaa ensimmäisen ampumapaikan takana. Paikoilla 1-5 olevat kilpailijat ampuvat ensin vuorotellen ensimmäiset yksittäiskiekkonsa ja jatkavat ohjelman mukaisesti vuorotellen, kunnes paikan viisi ampuja on ampunut kaikki kiekkonsa. Paikan viisi ampuja siirtyy kiekot ammuttuaan paikalle kuusi odottamaan siirtymistä paikalle yksi. Muut ampujat siirtyvät yhden ampumapaikan eteenpäin. Kierros kaksi etenee kuten ensimmäinen kierros ja seuraa taas paikan vaihto. Näin jatketaan aina kierrokselle kuusi asti. Kierroksella kuusi ensimmäinen ampuja on ampunut kaikki kiekkonsa ja on valmis. Erämuotoisessa kilpailussa kierros kuusi käydään loppuun ja paikalle astuu seuraava erä. Jonojärjestelmällä käytävässä kilpailussa tässä kohtaa paikalle kuusi tulisi kilpailun seitsemäs kilpailija. Jos kilpailijoita on erässä vähemmän kuin kuusi, niin kilpailijat etenevät silti kierros kierrokselta samassa järjestyksessä, mutta väliin jää tyhjiä paikkoja. (Liite 2.) (Ampumaurheiluliitto 2017, s.94-96, 109 ja 111.)

Tuomari voi määrätä kiekkojen uudelleen lähettämisen ase- ja patruunahäiriöistä johtuvista tilanteista, sekä kiekkohäiriötilanteista. Ase- ja patruunahäiriöt ovat tilanteita, joissa patruuna ei laukea tai molemmat patruunat laukeavat samanaikaisesti. Kiekkohäiriöiksi taas luetaan erilaisia tilanteita, joista esimerkkeinä kiekon rikkoutuminen heitettäessä, tuomarin mielestä ampujaa on häiritty tai tuomari ei pysty varmuudella tuomitsemaan kiekkoa. (Ampumaurheiluliitto 2017, s.97-99.)

1.3 Keskeiset käsitteet

Käsitteet:

Aliohjelma on itsenäinen ohjelman osa, jota voidaan kutsua eri puolilta pääohjelmaa tai muista aliohjelmista. Tässä työssä on oikeastaan neljä pääohjelmaa, joten ne on nimetty aliohjelmiksi.

Bluetooth Low Energy (BLE) on lyhyen kantaman langaton yhteystekniikka tiedonsiirtoon. Toimii vapaalla 2,4GHz taajuusalueella. Käytetään tavallisesti kahden laitteen väliseen yhteyteen.

Flash-muisti on puolijohdemuisti, jossa ei ole liikkuvia mekaanisia osia. Toimii eri logiikalla kuin EEPROM ja on tyhjennettävä lohkoittain.

Funktio on pienen rajatun tehtävän suorittava ohjelman osa.

Hajaspektritekniikka on tekniikka, jossa radioviestin kaistanleveys on riippumaton lähettävästä signaalista ja huomattavasti suurempi, kuin signaalin lähettämiseen tarvittava kaistanleveys.

I/O linja on input/output linja. I/O linja on tiedonsiirtoväylä, jonka voi ohjelmallisesti määrittää toimimaan, joko sisäänpäin tai ulospäin menevän liikenteen hallintaan.

Järjestelmäpiiri (SoC) (System-on-Chip) mikropiiri, jossa yhdistyy useiden eri piirien toiminnot.

Kellotaajuus ilmaisee miten monta tilanvaihdosta suoritin suorittaa sekunnissa.

Kirjasto on ohjelmoinnin yhteydessä funktioiden kokoelma, jota voidaan käyttää kutsumalla ohjelmakoodissa funktioita kirjastosta.

Mikrokontrolleri on mikropiiri, jossa on mikroprosessori, muisti- ja liityntälohkoja.

MIT-lisenssi (MIT on lyhenne Massachusetts Institute of Technology) on vapaa ohjelmistolisenssi, joka sallii ohjelman käyttämisen, kunhan lisenssiteksti säilyy lähdekoodissa.

Ohjelmoitava logiikka on mikroprosessoripohjainen pieni tietokone, jota käytetään automaatioprosessien ohjauksessa.

Pinni tarkoittaa tässä opinnäytetyössä I/O linjan liittimen piikkiä, johon johdin voidaan liittää.

Pinnikartta kertoo mitkä ovat pinnien eli liittimien suunnitellut käyttötarkoitukset laitteessa.

Regulaattori on jännitteen tasaava tai säätävä laite.

Rele on sähköisesti ohjattava kytkin.

Sarjaporttiadapteri muuntaa mikrokontrollerilta tulevan sarjamuotoisen dataliikenteen tietokoneen USB-portille sopivaan muotoon.

Silmukka ohjelmoinnin toistorakenne, joka toistuu, kunnes määrätty ehto toteutuu.

Lyhenteet:
EEPROM (Electronically Erasable Programmable Read-Only Memory) sähköisesti ohjelmoitavaa haihtumatonta puolijohdemuistia, joka voidaan uudelleen kirjoittaa bitti kerrallaan.

GNU LGPL (GNU Lesser General Public License) on lisenssi, joka mahdollistaa linkityksen yhteen ei GPL-lisensoidun ohjelman osan kanssa.

I2C (Inter-Integrated Circuit) on väylästandardi kaksisuuntaisiin ohjaus- ja tiedonsiirtoväyliin.

I2S (Inter-IC Sound) on väylästandardi digitaalisen äänen siirtämiseen.

LoRa (Long Range) radioviestintäteknologia, jonka toiminta perustuu radioaallon LoRa-modulaatioon.

SPI (Serial Peripheral Interface) sarjaliikenteen väylä, joka toimii neljällä signaalilla.

UART (Universal Asynchronous Receiver Transmitter) Asynkroninen vastaanotinlähetin, joka hoitaa laitteen bittien tulkinnan.

USB (Universal Serial Bus) on sarjaväyläarkkitehtuuri, jolla oheislaitteet liitetään tietokoneeseen.

Wi-Fi on WLAN-tuotteiden kaupallinen nimitys ja Wi-Fi Alliancen tavaramerkki. Nimitystä käytetään hyvin yleisesti WLAN-verkkojen ja siihen liittyvien ratkaisujen yhteydessä.

WLAN (wireless local area network) on langaton lähiverkko.

2 Käytettävien teknologioiden valinta

Vaihtoehtoina langattoman yhteyden muodostamiseksi pääohjausyksikön ja heitintä ohjaavan yksikön välille olivat vahvimmin Wi-Fi ja LoRa. Wi-Fi-yhteys on monissa laitteissa valmiiksi, mutta Jansonsin ja Dorinsin tekemät testit osoittavat yhteyden olevan herkän esteille ja kantaman olevan hyvissä olosuhteissa noin 300 metriä (Jansons & Dorins 2012, 30). Sotkan compak sporting -radalla Wi-Fin kantama todennäköisesti riittäisi, mutta joillakin ampumaradoilla kiekonheittimet ovat osittain maan alla tai vallin takana, jolloin Wi-Fin kuuluvuus helposti heikkenisi riittämättömäksi ja tarvittaisiin vahvistimia. Ensimmäiseksi valinnaksi tuli pitkille matkoille suunniteltu LoRa, jonka kanssa projekti käynnistettiin. Projektin aikana kuitenkin selvisi, että LoRan käyttämän taajuuden käytöstä on liikenne- ja viestintäviraston määräys, joka rajoittaa taajuutta käyttävän laitteen lähetysaikaa (Traficom 2019, 8). Lisäksi LoRan kantama voi olla, jopa 10 kilometriä, joten tähän käyttötarkoitukseen kantama on turhankin pitkä. Tämän vuoksi työssä kokeiltiin myös LRF24L01-moduulia, joka toimii radiolaitteena vapaalla 2.4GHz taajuudella. Prototyyppivaiheessa päätettiin kuitenkin käyttää LoRaa, joka oli jo valmiiksi tuttua ja käytettävissä tämän prototyypin osalta viestintäviraston säätämien rajoitusten mukaisesti.

Kierroksen suunnitteluun puhelimella vaihtoehtoina oli Bluetooth-yhteydellä toimiva sovellus, selainkäyttöliittymä internetissä ja selainkäyttöliittymä sisäisessä WLAN-verkossa. Puhelinsovellus karsiutui, koska jatkossa olisi otettava huomioon puhelimien eri käyttöjärjestelmät ja mahdollisesti käyttöjärjestelmäversiot. Selainkäyttöliittymä internetissä olisi vaatinut laitteelle jonkin internet liittymän. Internet liittymästä taas olisi tullut ylimääräistä kustannusta, kun kierroksen suunnittelu missä tahansa ei ole tarpeen. Projektissa päädyimme tekemään pääohjainyksiköstä WLAN-tukiaseman, joka pyörittää www-palvelinta. Tällöin kierroksen suunnittelija yhdistää oman laitteensa pääohjainyksikön ylläpitämään WLAN-verkkoon ja avaa puhelimen selainsovelluksessa kierroksen suunnittelusivun.

Pääohjausyksiköksi oli ehdolla Raspberry Pi 3 Model B+ tai jokin ESP-mikrokontrolleriin pohjautuva kehitysalusta. Prototyypin tekemiseen ESP-kehitysalusta soveltuu oikein hyvin, koska Arduino IDE helpottaa koodaamista ja virheiden selvittelyä. Koin asian henkilökohtaisesti, kun yritin saada käyttämäni LoRa-moduulin toimimaan Raspberry Pin kanssa. Usean päivän ja usean kirjaston kokeileminen ei tuottanut toivottuja tuloksia, kun ESP32-kehitysalustan ja Arduino IDE ohjelmointiympäristön kanssa toimiva kirjasto löytyi kohtuullisen nopeasti. Jatkossa Raspberry Pi tai vastaava yhden piirilevyn tietokone voisi kuitenkin tulla kysymykseen, sillä järjestelmään voisi kehittää vielä ajantasaisen tulosnäytön. Yhden piirilevyn tietokone tarjoaisi helpomman tavan tallentaa suuriakin määriä tietoja (esim. useamman valmiin kierrosvaihtoehdon ja vanhat tulokset).

Näytöksi valikoitui 2 rivin ja 16 merkin sininen nestekidenäyttö I2C-kommunikatioadapterilla, koska sellainen oli valmiiksi käytettävissä. Jälkikäteen tämä on osoittautunut toimivaksi, mutta hieman pieneksi. Isompi näyttö mahdollistaisi suuremman infomäärään näyttämisen kerralla ja pitkien suomenkielisten sanojen käyttämisen näytöllä.

Painikkeiden kohdalla käytettäväksi valikoitui teollisuudessa yleisesti käytetty halkaisijaltaan 22 millimetrin painonappi. Tällaista painonappia on helppo käyttää, vaikka hanskat kädessä. Lisäksi painikkeet ovat ulkokäyttöön soveltuvaa IP-luokitusta. Painikkeita prototyypissä on käytössä kolme kappaletta, joilla valintojen teko onnistuu jokaisessa tilanteessa. Tulosten talteen ottamiseksi laitteeseen mahdollisesti lisätään myöhemmin neljäs painike.

Virtalähteenä pääohjausyksikölle voidaan käyttää mitä tahansa kuluttajille myytävää varavirtapankkia soveltuvalla USB-johdolla. Näin akun ei tarvitse mahtua koteloinnin sisään ja sen vaihtaminen on helppoa. Heitintä ohjaavat yksiköt saavat sähkönsyötön heitinten akuista, joiden jännite pienennetään yhdeksään volttiin regulaattoreilla. Paikoissa, joissa kiekonheitin on kytkettynä verkkovirtaan, on mahdollista käyttää tavanomaista puhelimen laturia laitteen virransyöttöön.

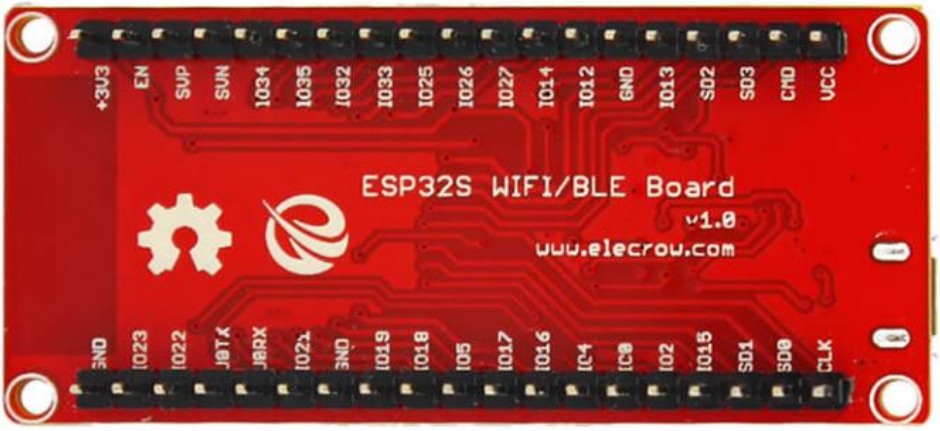
3 Kehitysympäristö

Tämä luku toimii tietoperustana projektille. Esittelen tässä luvussa valitsemieni komponenttien ominaisuuksia (3.1 - 3.4), ohjelmointialustaa (3.5) ja ohjelmointiin käytettäviä valmiita kirjastoja (3.5.1 - 3.5.3).

3.1 ESP32 kehitysalusta

ESP32 on kiinalaisen Espressif Systemsin suunnittelema mikrokontrolleri, joka tarjoaa sisäänrakennetun WLAN-yhteyden mahdollisuuden. ESP32n massatuotanto alkoi vuoden 2016 lopulla. ESP32-mikrokontrollerista on tehty useita moduuleita, joita voidaan käyttää eri kehitysalustoilla. Kehitysalustan avulla lisälaitteiden kytkeminen mikrokontrolleriin on huomattavasti helpompaa. (Kolban 2018, 60-63.)

Projektin pääohjausyksikkö on rakennettu Elecrown valmistaman ESP32S WiFi/BLE kehitysalustan päälle (kuva 1). Kuvassa yksi näkyy hyvin laitteen pohjassa olevat pinnit ja niiden nimet. Kyseiselle laitteelle ei valmistajan sivuilta löytynyt varsinaista pinnikarttaa, mutta kehitysalustan ytimenä toimii Espressif Systemsin valmistama ESP32-WROOM-32 -mikrokontrollerimoduuli. Espressif Systems on tehnyt mikrokontrollerimoduulille kattavan manuaalin, josta myös pinnien kaikki toiminnot selviävät.



Kuva 1. Elecrown ESP32S WiFi/BLE board (Elecrow 2020a)

ESP32-WROOM-32 -mikrokontrollerimoduuli sisältää langattoman tiedonsiirron mahdollistavan WLAN-yhteyden, sekä klassisen ja pienienergisen (BLE) Bluetooth-yhteyden. ESP32-WROOM-32 -mikrokontrollerimoduulin ytimenä toimii ESP32-D0WDQ6 -mikrokontrolleri, jossa on kaksi prosessoria, joiden kellotaajuutta voidaan säätää 80MHz ja 240MHz välillä. Lisäksi mikrokontrollerimoduulissa on sisäänrakennettu mahdollisuus SPI-, UART-, I2S- ja I2C-tiedonsiirtotavoille ja 4 megatavua SPI Flash-muistia. (Esperssif 2019, 1-2.)

Elecrow on kehitysalustallaan lisännyt laitteeseen USB-sarjaporttiadapterin, integroidun LiPo-akkulaturin, ohjelmointi- ja reset-painikkeet, sekä jännitteensäätimen, joka takaa järjestelmäpiirille tasaisen 3.3v jännitteen. (Elecrow 2020b.)

3.2 Nano 3.0 kehitysalusta

Käyttämäni GREATZT Nano 3 on Arduino Nano 3 -kehitysalustaan pohjautuva mikrokontrollerialusta, jonka ytimenä toimii ATmega328p-mikrokontrolleri. Nanossa on valmiina 22 pinniä, virtavalo, reset-painike ja mini-B USB-portti. Pienen kokonsa ansiosta tämä alusta sopii hyvin pienellekin reikälevylle ja tekee prototyyppi kokeiluista helpompaa. (AliExpress 2020a, Nussey 2013, 26.)

Arduinojen suunnittelu on alkanut Interaction Design Institute Ivreassa (IDII) Italiassa 2000-luvun alkupuolella Wiring-alustaan pohjautuen. Ensimmäinen Arduino kehitysalusta esiteltiin vuonna 2005. Kehitysalustan tarkoitus oli auttaa muotoiluopiskelijoita luomaan toimivia prototyyppejä fyysisen ja digitaalisen maailman yhdistämiseksi. Arduinoa on alusta alkaen kehitetty ihmisille, joilla ei ole aiempaa kokemusta elektroniikasta tai mikrokontrollerien ohjelmoimisesta. Arduino kehitysalustat perustuvat lisäksi avoimeen lähdekoodiin (The Creative Commons licenses) ja ovat siten muidenkin valmistajien muokattavissa ja valmistettavissa. Arduino-nimen käyttöä on kuitenkin rajoitettu siten, että vain Arduinon valmistamissa tuotteissa nimeä Arduino voidaan käyttää. (Arduino 2020a, Arduino2020b.)

ATmega328p mikrokontrolleri kuuluu AVR-arkkitehtuurisarjaan, jonka alun perin suunnittelivat norjalaiset opiskelijat Alf-Egil Bogen ja Vefard Wollan. ATmega328p on 8-bittinen mikrokontrolleri 32-kilotavun ohjelmoitavalla Flash-muistilla, kilotavun EEPROM-muistilla ja kahden kilotavun SRAM muistilla. Mikrokontrollerissa on 23 ohjelmoitavaa I/O linjaa. (Wheat 2011, Atmel 2015.)

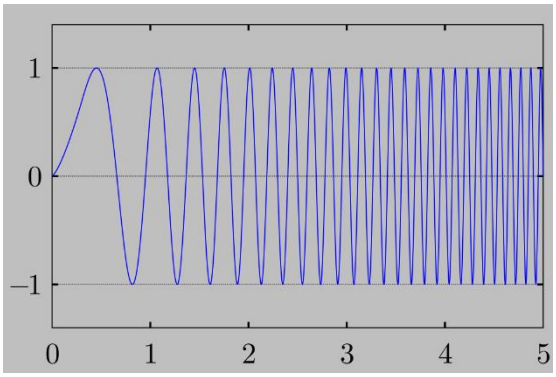
3.3 LoRa

LoRa (Long Range) on Semtechin kehittämä ja patentoima radioviestintäteknologia, jonka toiminta perustuu radioaallon LoRa-modulaatioon. LoRa tarjoaa pitkän kantaman tiedonsiirron pienille datamäärille, energiatehokkaasti ja turvallisesti. (Seneviratne 2019.)

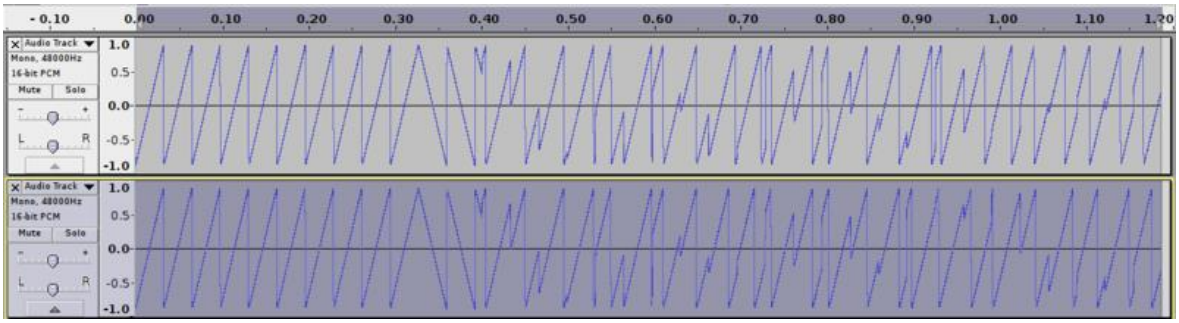
LoRa Alliancen ylläpitämästä käyttötaajuustaulukosta voidaan Suomen kohdalla todeta, että LoRa laitteita voidaan Suomessa käyttää taajuusalueilla 433.05-434.79 MHz ja 863-

873 MHz (LoRa Alliance 2020, 13). Projektissa käytetyn SX1276 LoRa-moduulin käyttötaajuudeksi on myyjä ilmoittanut 868 MHz (AliExpress 2020b). Tällä 868 megahertsin taajuusalueella on viestintäviraston määräyksen 15 mukaan toimintasuhde oltava enintään yksi prosentti. Prosentin toimintasuhde tarkoittaa, että laite saa olla lähetystilassa enintään 36 sekuntia tunnissa. Muuten taajuuden käyttö on radioluvasta vapaata aluetta laitteille, joiden efektiivinen säteilyteho on enintään 25 mW. (TrafiCom 2019, 6-8)

LoRa-modulaatiotekniikka on CSS-tyyppinen (Chirp Spread Spectrum) hajaspektritekniikka, jossa radiotaajuus joko alenee suhteessa aikaan (down-chirp) tai kasvaa suhteessa aikaan (up-chirp), kuten kuvassa 2. LoRa-modulaatiotekniikka käyttää laitteiden välisessä tiedonsiirrossa molempia, sekä kasvavia että alenevia taajuusvaihteluita, kuten kuvassa 3 suurin piirtein kohdassa 0,325. Jokainen lähetettävä bitti jaetaan osiin (chip). Näiden osien lukumäärää bittiä kohti kutsutaan hajotuskertoimeksi (SF, Spreading factor). LoRan käyttämässä CSS-hajaspektritekniikassa voidaan käyttää hajotuskertoimia 7-12. Pienemmällä hajotuskertoimella voidaan lyhyemmässä ajassa lähettää enemmän tietoa, mutta vastaavasti kantama pienenee. Suuremmalla hajotuskertoimella laite vaatii pidemmän ajan tiedon lähettämiseen, mutta kantama on pidempi. Kuvassa 3 näkyy myös kohdan 0,35 jälkeen, miten varsinainen data on säännöllisyyden poikkeamia. (Seneviratne 2019)



Kuva 2.Kasvava taajuus eli up-chirp (Wikipedia 2010)



Kuva 3. LoRa datavirtaa (RevSpace 2016)

3.4 I2C-Nestekidenäyttö ja painikkeet

Projektissa käytetään sinitaustaista 2*16 nestekidenäyttöä (LCD eli Liquid-Crystal display). Näytön numeroinnissa kaksi tarkoittaa, että näytössä saa tekstiä kahdelle riville ja 16 tarkoittaa, että yhdelle riville saa 16 merkkiä. Projektissa käytettyyn näyttöön on valmiiksi juotettu I2C-kommunikatioadapteri. I2C-kommunikaatioadapterin avulla näyttöä voi käyttää ESP32-kehitysalustassa vain neljällä johtimella. (Joy-it 2017.)

I2C-kommunikaatioadapteri tarvitsee virta- ja maajohtimen lisäksi johtimet vain sarjan tiedonsiirrolle (SDA), sekä sarja kellolle (SCL). Yhdistettyjen laitteiden välille tulee orjaisäntä suhde, jossa isännällä on jokaiselle orjalle oma yksilöllinen osoitteensa. Useamman laitteen ollessa yhteydessä I2C-väylä pystyy törmäyksen havaitsemiseen datan korruptoitumisen estämiseksi. Esimerkiksi nestekidenäytön kohdalla voimme käyttää 12 johtimen sijaan neljää johdinta, jolloin kytkeminen on helpompaa ja muille laitteille jää enemmän tilaa. (NXP Semiconductors 2014, 3-4.)

Painonapiksi työssä valikoitui teollisuudessa yleisesti käytössä oleva halkaisijaltaan 22 millimetrin painike Harmony XB5. Painikkeesta käytetään kolmea eri väriä, jotta ohjaimen käyttäminen olisi helpompaa. Painikkeen käyttölämpötilaksi on ilmoitettu -40 - +70 celsiusastetta ja siinä on IP-67 suojaluokitus, joka riittää ulkokäyttöön. (SE 2020.)

3.5 Arduino IDE ja ohjelmakirjastot

Arduino on vapaan lähdekoodin kehitysalusta, joka käsittää sekä helppokäyttöisen laitteiston että helppokäyttöisen ohjelmiston. Arduino IDE on integroitu kehitysympäristö (Integrated Development Enviroment), joka on kehitetty, jotta näiden kehitysalustojen ohjelmointi olisi helpompaa. Arduino IDE -ohjelmointiympäristössä käytetään omaa ohjelmointikieltä, joka perustuu Processing-ohjelmointikieleen. Kehitysalustalle lähetettäessä ohjelma tulkitsee kirjoitetun koodin C-ohjelmointikielelle ja tulkinta välitetään avr-gcc-kääntäjällä mikroprosessorin ymmärtämäksi (Banzi 2011, 20). Arduinon ohjelmointikieli muistuttaa hyvin paljon C-ohjelmointikieltä ja useimmat C-ohjelmointikielen ohjeet toimivat myös Arduino IDE ohjelmointiympäristössä. (Arduino 2020c.)

Arduino ohjelmointiympäristön toimivuutta voi laajentaa kirjastojen avulla, kuten useimmissa ohjelmointiympäristöissä. Kirjastot tarjoavat ohjelmointiin toiminnallisuuksia, joilla voidaan ohjata erilaisia laitteistoja, sensoreita tai vaikkapa manipuloida dataa. Arduino IDE ohjelmiston mukana tulee useita valmiita kirjastoja ja niitä voi lisätä tarpeen mukaan. Arduinon sivuilta löytyvillä ohjeilla kirjastoja voi tehdä omatoimisesti lisää. Projektissa käytetyt kirjastot on esitelty alla paremmin. (Arduino 2020d.)

3.5.1 ESP32-kirjasto

Arduino IDE ohjelmointiympäristöön on saatavilla Espressif Systemsin ylläpitämä kirjasto, joka mahdollistaa ESP32-kehitysalustan ohjelmoinnin Arduino IDE ohjelmointiympäristöllä. Kirjasto on lisensoitu GNU Lesser General Public License (GNU LGPL) -lisenssillä. Kirjasto on oikeastaan kokoelma usean eri tekijän yksittäisiä kirjastoja. (Github 2020a.)

Tässä projektissa on tarpeen käyttää Espressif Systemsin ylläpitämästä kirjastosta WiFi-kirjaston ja EEPROM-muistikirjaston ominaisuuksia. WiFi-kirjastosta tarkemmin osia WiFi.h, WiFiClient.h ja WiFiAP.h. Ivan Grokhotkovin joulukuussa 2014 muokkaama ESP32 WiFi.h perustuu Arduinon vastaavaan WiFi Shield -kirjastoon. Tästä Grokhotkovin kirjastosta löytyy kaikki perusominaisuudet, joilla WLAN-yhteyksiä perustetaan. WiFiClient.h on lisensoitu Adrian McEwen toimesta vuonna 2011 ja tällä kirjastolla voidaan hallita mm. palvelinyhteyspyyntöjä. WiFiAP.h joka myös perustuu Arduino WiFi Shieldin WiFi.h kirjastoon on Ivan Grokhotkovin joulukuussa 2014 muokkaama, mutta jonka on uudelleentyöstänyt Markus Sattler joulukuussa 2015. WiFiAP-kirjaston avulla ESP32-kehitysalustasta voidaan tehdä WLAN-tukiasema. (Github 2020b, Github 2020c, Github 2020d.)

EEPROM-muisti (Electronically Erasable Programmable Read-Only Memory) on sähköisesti pyyhittävää ohjelmoitavaa lukumuistia. EEPROM-muistissa tieto pysyy, vaikka laitteesta sammutettaisiin virta (Li, Yu & Hao 2011, 170). Muistia käytetään pääohjainlaitteessa vakiokierroksen tallentamiseen. Tämän muistin käyttämiseksi on kirjaston alun perin tehnyt Ivan Grokhotkov ESP8266-laitteille. Tämän on kääntänyt ESP32-laitteille Paolo Becchi ja muokannut Elochukwu Ifediora. (Github 2020e.)

3.5.2 I2C-nestekidenäytön kirjasto

Projektissa I2C-kommunnikaatioadapterilla varustetun nestekidenäytön ohjaukseen käytettiin kirjastoa, johon on merkitty tekijäksi Frank de Brabander ja ylläpitäjäksi Marco Schwartz. Frank de Brabander on omalla Github sivullaan jakanut kirjaston ja merkinnyt, että kirjasto perustuu DFROBOTin tekemään vastaavaan kirjastoon (Github 2020d). Mihinkään näistä kirjastoista ei ole merkitty lisenssiä, joten käyttöoikeuden varmistamiskeksi käyttölupaa kysyttiin sähköpostitse käytetyn kirjaston ylläpitäjäksi merkityltä Marco Schwartzilta ja alkuperäisen kirjaston lisenssiä kysyttiin kirjaston tekijän DFROBOTin verkkosivuilta. Verkkosivulla kysymykseen lisenssistä ei toistaiseksi ole vastattu, mutta si-

vuston nykyinen nestekidenäyttöjen kirjasto johtaa käyttämääni Marco Schwartzin ylläpitämään kirjastoon. Marco Schwartz vastasi sähköpostiini ja antoi luvan kirjaston käyttöön työssä (liite 1). (Github 2020c.)

3.5.3 LoRa kirjasto

Toimivan LoRa-kirjaston löytäminen oli työn onnistumisen kannalta kriittisen tärkeää. Ilman kirjastoa ei olisi pystynyt SX1276 LoRa-moduulia käyttämään työssä. Tällaisen toimivan kirjaston on MIT-lisenssillä (Massachusetts Institute of Technology) tehnyt Sandeep Mistry (Github 2020e). Kirjaston mukana tulee lisäksi käytännön esimerkkejä, joiden avulla kirjaston toimintoihin voi tutustua.

4 Arkkitehtuuri

Kokonaisuus muodostuu teoriassa kahdesta laitteesta: pääohjausyksiköstä ja heitinohjausyksiköstä. Kiekonheittimiä pitää kuitenkin sääntöjen mukaisella kierroksella olla vähintään kuusi (luku 1.2). Tämä määrittää, että käytännössä heitinohjausyksiköitä tulee olla myös kuusi. Jos kaksi heitintä on hyvin lähellä toisiaan, voidaan samalla heitinohjausyksiköllä ohjata kahta relettä eli myös kahta kiekonheitintä.

Pääohjausyksikön tehtävä on käskystä lähettää LoRa-viesti, jossa kerrotaan minkä heitinohjausyksikön tulee antaa kiekonheittimelle viesti kiekon laukaisemiseksi. Pääohjausyksikössä on valittavana neljä toimintoa: vakiokierros, satunnaiskierros, näyttökierros ja suunnittelu.

Vakiokierros on Compak Sporting -sääntöjen mukainen kierros, jonka kulku voidaan suunnitella suunnittelutoiminnossa. Kierros toimii 1-6 ampujalle sääntöjen mukaisessa ampumajärjestyksessä. Kiekonheittimellä sattuva vikatilanne on otettu huomioon siten, että kiekon/kiekot voi uusia minkä tahansa lähetystilanteen jälkeen. Heitinhäiriön sattuessa kaksoiskiekon jälkimäisellä kiekolla, uusitaan sääntöjen mukaisesti molemmat kiekot (luku 1.2). Mikäli ampujan ase rikkoutuu tai ampuja poistuu muusta syystä kesken kierroksen, niin voidaan tämän ampujan kiekot ohittaa lähettämättä niitä.

Satunnaiskierros on kierros, jossa kiekko lähtee satunnaisesta kiekonheittimestä. Kierros on prototyypissä koodattu muotoon kolme yksittäiskiekkoa ja yksi kaksoiskiekko. Ampujia voi kierrokselle osallistua 1-6 ja ammunta etenee samassa järjestyksessä kuin compak sporting -säännöissä. Satunnaiskierroksella on samat kiekon uusinta- ja ohitusmahdollisuudet kuin vakiokierroksella.

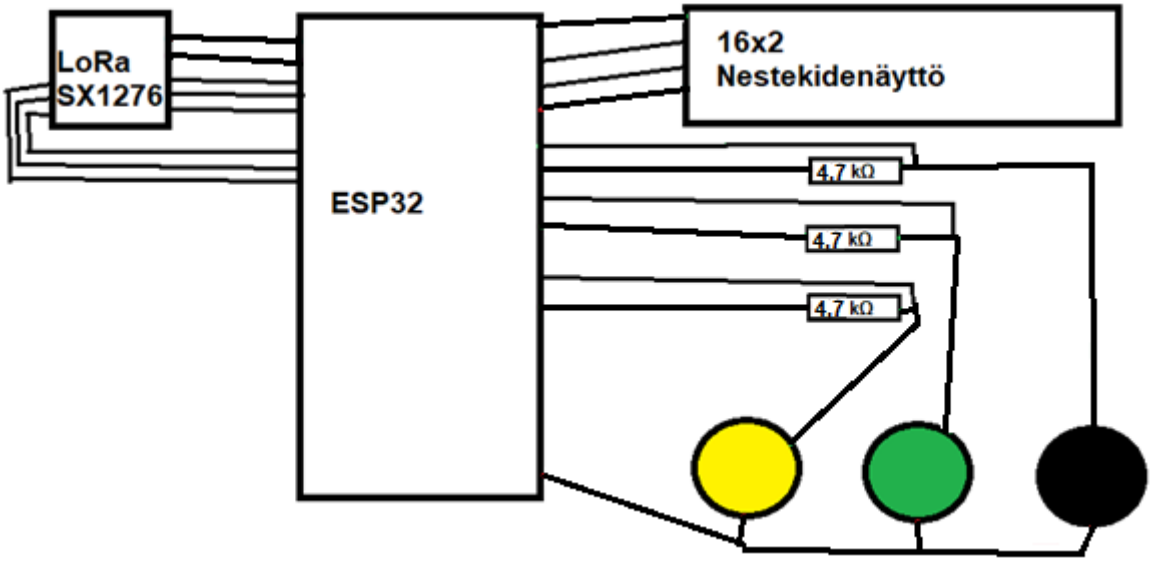
Näyttökierros mahdollistaa näyttökierroksen järjestämisen tai yksittäisten kiekkojen harjoittelemisen. Kierroksella on mahdollista valita mille heitinohjausyksikölle lähetetään viesti kiekon laukaisemiseksi. Tämän ansiosta näyttökierroksella on hyvä testata LoRa-yhteyttä, jos siinä ilmaantuu häiriötä.

Suunnittelun valitseminen tekee pääohjainyksiköstä www-palvelimen, jonka luomalla internetsivulla on mahdollista suunnitella vakiokierroksen kulku. Prototyypissä on mahdollista tehdä kierrossuunnitelma myös compak sporting -sääntöjen vastaisesti. Säännöissä on kirjattuna kohdassa C.4.2, että kaksoiskiekon ensimmäinen kiekko on oltava sama kuin ohjelman edellinen kiekko (Ampumaurheiluliitto 2017, 93). Tästä säännöstä poikkeamista

ei suunnittelussa ole poissuljettu, vaan oletetaan suunnittelijan tekevän poikkeuksen sääntöön tarkoituksellisesti.

4.1 Pääohjausyksikkö

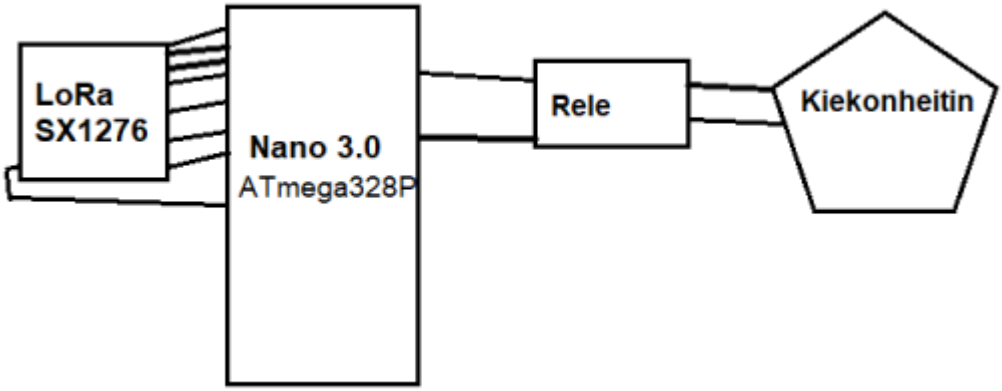
Pääohjausyksikön ytimenä toimii Elecrown valmistama ESP32S WiFi/BLE kehitysalusta (luku 3.1). ESP32n tehtävänä on myös toimia WLAN-tukiasemana ja www-palvelimena kierroksen suunnittelussa. Tähän kehitysalustaan on liitetty SX1276 LoRa-moduuli (luku 3.3), sinitaustainen 2x16 nestekidenäyttö (luku 3.4) ja kolme eriväristä Harmony XB5-sarjan painonappia (luku 3.4). Kokonaisuudessa jokaiselle kolmelle painonapille on yksi 4,7:n kilo-ohmin vastus. Kuvassa 4 on pääohjausyksikön laitteisto. SX1276 LoRa-moduuli toimii lähettimenä, joka lähettää viestit heitinohjausyksiköille. Nestekidenäytölle annetaan ammunnan ohjaamiseen ja valitsemiseen tarvittava tieto. Painikkeilla suoritetaan valinnat eri tilanteissa.



Kuva 4. Pääohjausyksikkö

4.2 Heitinohjausyksikkö

Heitinohjausyksikön ytimenä toimii GREATZT Nano 3.0 mikrokontrollerialusta (luku 3.2). SX1276 LoRa-moduuli toimii pääohjausyksikön lähettämien viestien vastaanottajana. Mikrokontrolleri tarkistaa saapuneesta LoRa-viestistä kuuluuko viesti mikrokontrolleriin kytketylle kiekonheittimelle. Viestin kuuluessa kyseiselle heittimelle mikrokontrolleri antaa releen kautta kiekonheittimelle laukaisukäskyn. Heitinohjausyksiköillä on siis normaalitilanteessa jokaisella nimetty kiekonheitin, jolle kuuluviin viesteihin ne reagoivat. Poikkeustapauksessa yhteen heitinohjausyksikköön voidaan kytkeä useampi kiekonheitin, jolloin heitinohjausyksikkö antaa laukaisukäskyn sille releelle jolle viesti kuuluu. Kuvassa 5 on heitinohjausyksikön laitteisto kytkettynä releen kautta kiekonheittimeen.



Kuva 5. Heitinohjausyksikkö

5 Ohjelmakoodi

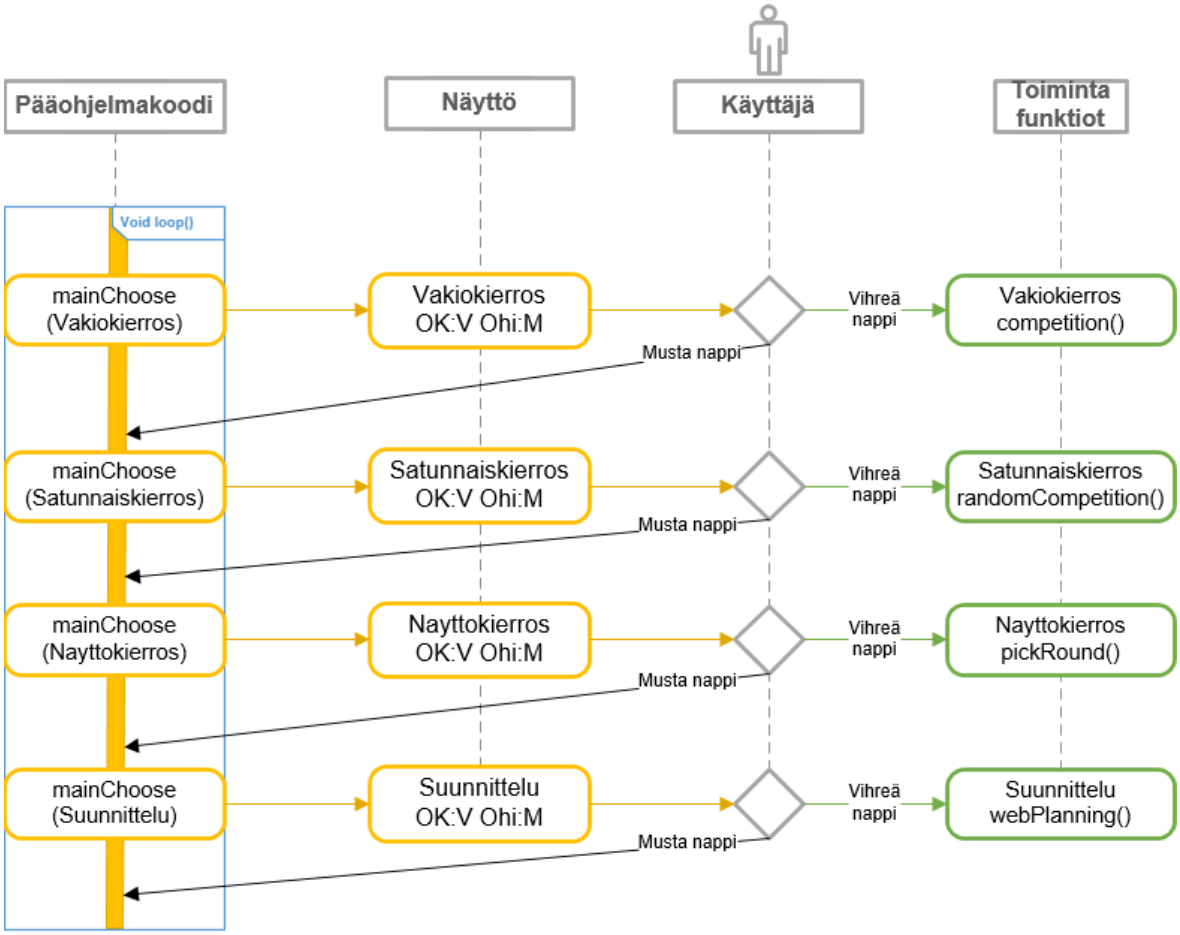
Ohjelman näkyvä toiminnallisuus rakentuu päävalikosta ja neljästä aliohjelmasta. Neljä aliohjelmaa ovat laitteen toimintavaihtoehdot eli vakiokierros, satunnaiskierros, näyttökierros ja suunnittelu. Ohjelma käyttää kuitenkin hyväkseen useita funktiota, joita käytetään useammassa aliohjelmassa ja päävalikossa. Tässä luvussa selitetään ensin mitä ohjelma tekee, kun laite käynnistyy. Toisena käydään läpi päävalikon ohjelmallinen toiminta. Alaotsikoiden alla selitetään neljän aliohjelman ohjelmallinen toiminta ja viimeiseksi heitinohjausyksikön ohjelmallinen toiminta. Funktioiden toimintaa selvitetään sitä mukaa, kun se muun selvityksen kannalta on oleellista.

Laitteen käynnistyttyä käyttäjän ensimmäinen näkymä on ohjelman päävalikko. Ohjelmassa ennen päävalikkoa tehdään kuitenkin laitteelle alkuasetuksia. Ensimmäisenä ohjelmassa on lueteltuna käytettävät kirjastot (luku 3.5). Seuraavaksi on määritelty globaalit muuttujat eli muuttujat, joita tarvitaan ohjelmassa useammassa eri aliohjelmassa. Tällaisia ovat ampujamäärä, laitteen IP-osoite ja kierroskulun taulukot. Globaaleiksi muuttujiksi on määritetty arvoja, joita tullaan mahdollisesti myöhemmin muuttamaan. Esimerkkejä mahdollisesti myöhemmin muuttuvista arvoista, ovat käytettävät pinnit, satunnaiskierroksen ampumaohjelma ja heitinten määrä, sekä nimeämiskäytäntö. Ensimmäisinä varsinaisina toimintoina voidaan kuitenkin pitää setup-nimisessä funktiossa tapahtuvia LoRan, EEPROM-muistin, WLAN-tukiaseman avauksia. Tässä setup-funktiossa asetetaan painonappien pinnit sisääntulopinneiksi ja luetaan EEPROM-muistista sinne tallennettu vakiokierros.

Ohjelman päävalikko on päättymätön silmukka, jossa laitteen käyttäjä voi valita aloitetaanko jokin neljästä aliohjelmasta. Päävalikon kohta, jossa valinnan voi tehdä, näkyy kuvan 6 vasemmassa reunassa. Aliohjelmat kuvan 6 oikeassa reunassa on kuvassa nimetty toiminta funktioiksi. Silmukan alussa luodaan totuusarvomuuttuja, jonka arvo kysytään jokaisen aliohjelman kohdalla funktiosta nimeltä mainChoose. Kun totuusarvomuuttuja saa arvokseen tosi, antaa päävalikon silmukka kohdalla olevalle aliohjelmalle suoritus komennon. Päävalikon funktiosta mainChoose-funktiolle lähetetään kysyttävän kierroksen nimi, joka kuvassa 6 on pääohjelmakoodin toimintalaatikoissa sulkeiden sisällä oleva teksti.

Funktio mainChoose toimii nestekidenäytölle näkyvän tekstin ohjaimena ja oikean totuusarvon palauttajana (kuvassa 6 kohdan näyttö toiminnot). Nestekidenäytölle näkyy päävalikon funktiolle toimittama teksti ja ohjeet, millä painikkeella hyväksytään tai ohitetaan toiminto. Painikeohjeiden näkyminen näytölle on protoamisvaiheessa ollut tarpeellista, mutta eriväristen painikkeiden ja myöhemmin tehtävän manuaalin myötä nämä todennäköisesti

poistetaan. Oikean totuusarvon palauttamiseksi mainChoose-funktion alussa luodaan kokonaislukumuuttuja, jonka arvo kysytään painikkeita hallitsevalta funktiolta nimeltä choose. Kuvassa 6 choose-funktion toiminta tapahtuu käyttäjän kohdalla. Painallukset tunnistava funktio palauttaa jokaisesta kolmesta painikkeesta eri kokonaisluvun. Näiden kokonaislukujen perusteella mainChoose-funktio osaa palauttaa tosiarvon vihreän painikkeen painalluksesta ja epätosiarvon mustan painikkeen painalluksesta. Keltaista painiketta painettaessa mainChoose-funktio tuo viideksi sekunniksi näytölle tekstin ”valitsit K”, kunnes palauttaa alkuperäisen tekstin ja painikeohjeet.



Kuva 6. Päävalikon toiminta

5.1 Vakiokierros

Yksi neljästä aliohjelmasta on vakiokierros, jolla onnistuu tavallisen compak sporting -kierroksen ohjaaminen 1-6 ampujalle. Tämä aliohjelma alkaa ohjaamiselle ampujamäärän selvittävään funktioon nimeltä howMany. Tämä funktio tulostaa näytön ensimmäiselle riville tekstin ”Ampujia” ja kokonaisluvun valittuna olevasta ampujamäärästä. Näytön toisella rivillä näkyy ohjeet painikeille eli mustalla vähennetään, keltaisella lisätään ja vihreällä vahvistetaan. Painikkeiden painallukset tunnistetaan taas funktiolla nimeltä choose,

jonka toimintaa kuvattiin edellä. Mikäli ampujia yritetään valita vähemmän kuin yksi pyöräyttää ohjelma ampujamäärän kuuteen. Painettaessa ampujamäärää lisäävää keltaista painiketta, ampujamäärän ollessa kuusi, ampujamäärä palaa takaisin yhteen.

Ampujamäärän selvittämisen jälkeen lähetetään tieto ampujamäärästä shootOrder nimiselle funktiolle. Tämä funktio tallentaa compak sporting -sääntöjen mukaisen ampujien kierron ampumapaikkojen taulukkoon ja ampujien taulukoon. Ampujien kiertoa on esitelty tarkemmin luvussa 1.3 ja liitteessä 2. Ohjelman mukainen lähetettävä kiekko lasketaan myöhemmin tämän paikkatiedon ja meneillään olevan ampumavuoron perusteella. Ampujien taulukkoa käytetään vuorossa olevan ampujan selvittämiseen, sillä kierros kulkee aina pieninumeroisimmasta paikasta alkaen eikä pieninumeroisimmasta ampujasta alkaen.

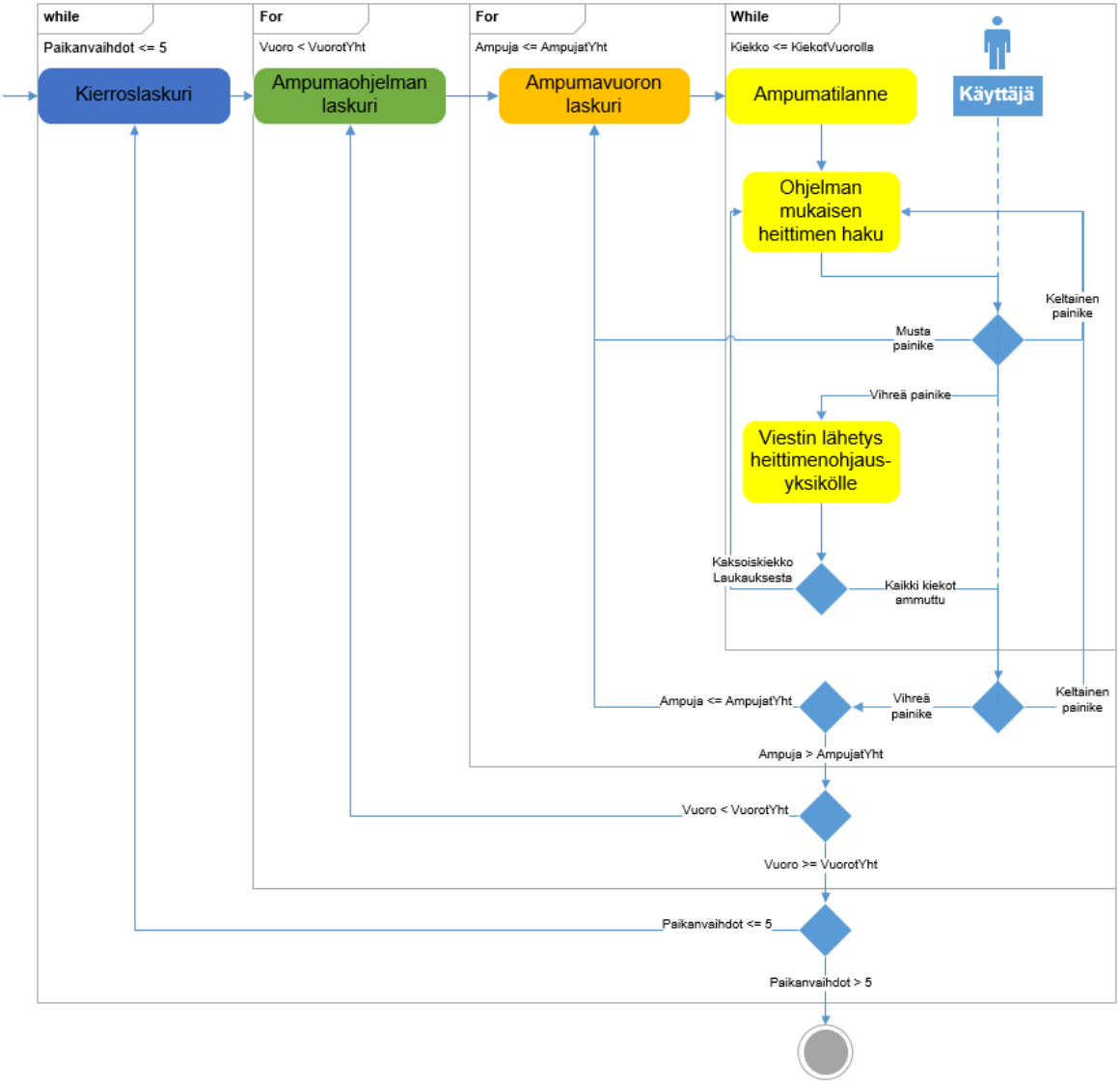
Vakiokierros-aliohjelman seuraavassa askeleessa lasketaan todellisten ampumavuorojen määrä, eli yksittäis- ja kaksoiskiekkojen määrää per paikka. Todellista ampumavuorojen määrää käytetään vakiokierroksen kulun ohjauksessa. Ampumavuorojen taulukko on määritelty viiden yksittäiskiekon mukaisesti viiden muuttujan kokoiseksi. Jokainen kaksoiskiekko kuitenkin vähentää ampumavuoroja ja muuttujien määrää yhdellä. Tämän vuoksi ampumaohjelman mukaisten ampumavuorojen määrä vaihtelee kolmesta viiteen ja vuorojen määrä tarkistettava laskemalla.

Varsinainen vakiokierros koostuu neljästä sisäkkäisestä silmukasta, kuten kuvassa 7 on havainnollistettu. Ensimmäisessä silmukassa (kuva 7 kierroslaskuri) lasketaan paikanvaihdot, eli säännöissä ja liitteessä 2 kuvatut kierrokset. Toisessa silmukassa (kuva 7 ampumaohjelman laskuri) lasketaan ampuohjelman mukaisia vuoroja. Kolmannessa silmukassa (kuva 7 ampumavuoron laskuri) lasketaan, että jokainen ampuja saa suorittaa tämän ampumaohjelman mukaisen vuoronsa. Neljäs silmukka (kuva 7 ampumatilanne) laskee ja lähettää ohjelman mukaiset kiekot.

Ensimmäisen silmukan, eli kierroslaskurin muuttuja kasvaa aina ammunnan edettyä vaiheeseen, jossa ampujat vaihtavat paikkaa. Tämän muuttujan kasvaessa yli viiden on vakiokierros suoritettu kokonaan ja palataan päävalikkoon, kuten kuvan 7 alin kärkineliö osoittaa. Lisäksi tämän silmukan lopussa muuttuja, joka laskee sijaintia ampumapaikkojen ja ampujien taulukossa siirretään seuraavalle kierrokselle. Tämä tehdään lisäämällä muuttujaan ampujien määrä.

Toisessa silmukassa lasketaan, monesko ampumaohjelman mukainen kiekkokierros on menossa. Viiden yksittäiskiekon kierroksella ampumavuoroja tulee jokaiselle viisi, kun kahden kaksoiskiekon ja yhden yksittäiskiekon kierroksella ampumavuoroja tulee kolme.

Silmukan lopussa muuttuja, joka laskee sijaintia ampumapaikkojen ja ampujien taulukossa siirretään takaisin kierroksen alkuun. Näin tilanne, jossa kuvan 7 toiseksi alimmassa kärkineliössä siirrytään vain seuraavalle kiekkokierrokselle (vasemmalle) pitää ampuja- ja paikkajärjestyksen ennallaan.



Kuva 7. Vakiokierroksen ohjelmakaavio

Kolmannessa silmukassa lasketaan, että kaikki mukana olevat ampujat saavat tällä kiekkokierroksella vuoronsa. Silmukan tässä kohdassa muuttujalla, joka laskee sijaintia ampumapaikkojen ja ampujien taulukossa haetaan tämänhetkinen ampuja ja ampumapaikka. Silmukan lopussa kuvan 7 käyttäjän alimmassa kärkineliössä varmistetaan, että pitääkö ampumatilannesilmukka ottaa uudelleen. Lopuksi siirrytään seuraavan ampujan vuoroon tai pois silmukasta sen mukaan, ovatko kaikki ampujat saaneet vuoronsa kiekkokierroksella.

Neljännessä silmukassa käydään läpi ampumatilanne. Silmukan alussa ampumapaikka ja ampumavuoro tietojen perusteella lasketaan heitin, jolle viesti kuuluu lähettää. Tämän jälkeen siirrytään shootSituation nimiseen funktioon, jolle lähetetään tieto vuorossa olevasta ampujasta, ampumapaikasta, heittimestä ja monennestako kiekosta tällä kertaa on kyse. Tämä shootSituation-funktio käyttää apunaan painikkeiden painallukset tunnistavaa choose-funktiota. Ensimmäisen kiekon aikana mustalla voi ohittaa ampujan vuoron ja vihreällä vahvistaa viestin lähettämisen heittimelle. Toisen kiekon aikana mustalla voi ohittaa ampujan vuoron, keltaisella voi uusia ensimmäisen kiekon ja vihreällä voi vahvistaa toisen kiekon lähettämisen. Viestin lähettämisestä heitinohjausyksikölle ja viestin lähettävästä message-funktiosta on kerrottu tarkemmin luvussa 5.5.

5.2 Satunnaiskierros

Satunnaiskierros on toiminnallisesti lähes identtinen vakiokierroksen kanssa. Kierroskulun, paikanvaihtojen ja ampumajärjestyksen osalta, on tarkoitus mennä samalla tavoin kuin vakiokierroksessakin. Satunnaiskierros eroaa vakiokierroksesta vain sillä, että ampuja ei ennakkoon tiedä mistä heittimestä kiekko lähtee. Kerron tässä luvussa sen vuoksi vain eroavaisuuksista vakiokierrokseen nähden.

Satunnaiskierroksen ampumaohjelma, eli yksittäis- ja kaksoiskiekkojen määrä kierroksella, on määritelty koodin alussa ja on muokattavissa vain ohjelmoimalla. Vakiokierroksessa ampumaohjelmaa voi muokata myös suunnittelussa. Toinen eroavaisuus vakiokierrokseen nähden löytyy kuvan 7 ampumatilanne silmukan ensimmäisestä kohdasta, jossa lukee ohjelman mukaisen heittimen haku. Satunnaiskierroksella ohjelman mukainen satunnainen heitin saadaan arpomalla sijainti heitinten taulukkomuuttujasta. Arpominen suoritetaan Arduinon oman ohjelmointikielen random-funktiolla.

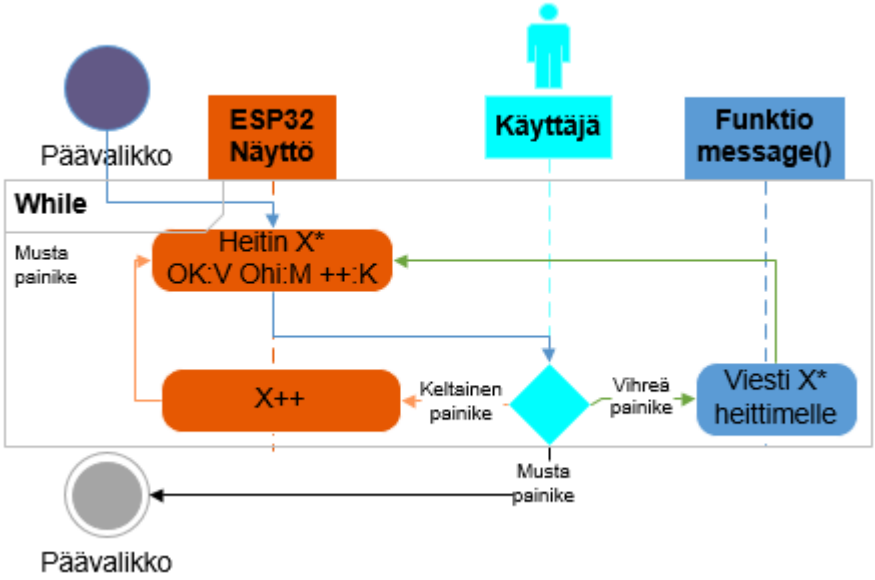
5.3 Näyttökierros

Näyttökierroksella voidaan esittää ampujille kiekkojen lentoradat. Näyttökierroksella voidaan valita haluttu heitin ja lähettää lähetyskäsky tälle kyseiselle heittimelle. Ominaisuutta voidaan lentoratojen esittelemisen lisäksi käyttää yksittäisten kiekkojen harjoittelussa ja heitinten toiminnan kokeilemisessa. Näyttökierroksen ohjelmakaavio on kuvassa kahdeksan.

Kun päävalikosta saavutaan näyttökierrokselle, tulee näytön ensimmäiselle riville teksti ”Heitin X”. X kuvaa heittimien taulukkomuuttujan ensimmäiseen sijaintiin tallennettua merkkiä, joka todennäköisesti on A. Näytön toiselle riville tulee toistaiseksi painikeohjeet ”OK:V Ohi:M ++:K”. Kuvan 8 mukaisesti valinta pyydetään käyttäjältä painike painallukset

tunnistavaa choose-funktiota apuna käyttäen. Käyttäjän valitessa keltaisen painikkeen mennään heittimien taulukkomuuttujassa seuraavaan sijaintiin. Jokaisella keltaisen painikeen painalluksella mennään seuraavaan sijaintiin, kunnes sijainnin sisältö on nolla eli tyhjä. Heitinten taulukkomuuttujan palauttaessa tyhjän arvon palataan uudestaan taulukon ensimmäiseen sijaintiin.

Painettaessa vihreää painiketta, pääohjausyksikkö lähettää viestin taulukkomuuttujan sijainnissa olevalle heittimelle eli tämän heitinohjausyksikölle. Viestin lähetys ja heitinohjausyksikön toiminta on kuvattu tarkemmin luvussa 5.5. Viestin lähettämisen jälkeen ollaan heitinten taulukkomuuttujassa samassa sijainnissa ja palataan käyttäjän valintaan. Käyttäjän painaessa mustaa painiketta siirrytään pois näyttökierroksen aliohjelmasta ja palataan päävalikkoon.



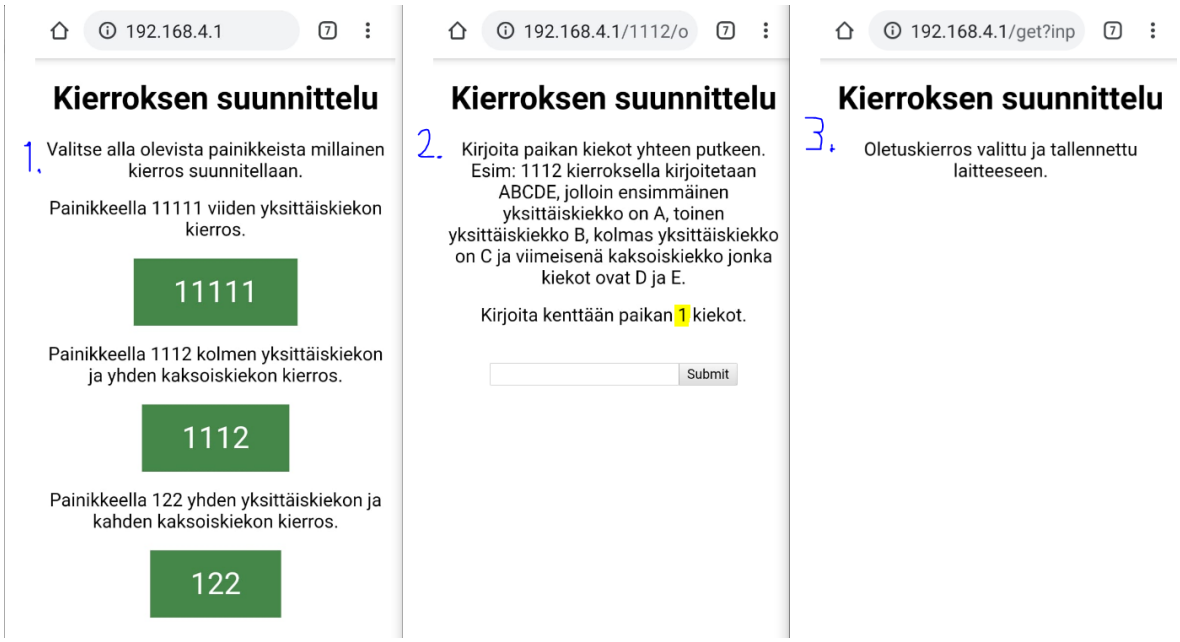
Kuva 8. Näyttökierroksen ohjelmakaavio

5.4 Suunnittelu

Suunnittelussa vakiokierroksen kulku voidaan suunnitella ja vaihtaa. Suunniteltu vakiokierros tallennetaan pääohjausyksikön EEPROM muistiin, jotta se on käytettävissä myös virrankatkaisun jälkeen. Suunnittelu on toteutettu selainpohjaiseksi pääohjausyksikön ylläpitämään sisäiseen verkkoon. Käyttäjän on siis ensin otettava yhteys pääohjausyksikön WLAN-verkkoon ja kirjoitettava selaimen osoitekenttään pääohjausyksikön IP-osoite. Kun pääohjausyksiköstä valitaan suunnittelu, niin IP-osoite näkyy myös pääohjausyksikön näytöllä. Suunnittelu voidaan tehdä millä tahansa laitteella, joka on yhdistettävissä WLAN-verkkoon ja jossa on verkkoselain.

Kuvassa 9 on kuvakaappaukset vakiokierroksen suunnittelu sivuista 1-3. Kun käyttäjä saapuu pääohjainyksikön suunnittelu sivulle, näkyy ensimmäisenä sinisellä ykkösellä numeroitu vasemmanpuoleisin sivu. Tällä ensimmäisellä sivulla valitaan haluttu ampumaohjelma kierrokselle, eli yksittäiskiekkojen ja kaksoiskiekkojen määrä. Ampumaohjelmavaihtoehtoja on kolme, painikkeiden nimessä ykköset ja kakkoset ovat ampumaohjelman mukaisia yksittäiskiekkoja ja kaksoiskiekkoja. Painikkeiden ampumaohjelmat ovat, myös selitettynä painikkeiden yläpuolella (kuva 9, vasen reuna).

Kun jokin kolmesta ampumaohjelmasta on valittu, sivu päivittyy sinisellä numerolla toiseksi merkittyyn vaiheeseen (kuvassa 9 keskellä). Sivussa on tekstikenttä, johon paikan kiekot kirjoitetaan heittojärjestyksessä. Paikka on kerrottu ennen tekstikenttää tekstissä ”kirjoita kenttään paikan X kiekot”. X kuvaa numeroa, joka kuvassa 9 on korostettu keltaisella värillä ja joka kasvaa aina yhdellä, kun painetaan Submit-painiketta. Kun paikan viisi kiekot on kirjoitettu tekstikenttään ja painettu Submit-painiketta, sivu päivittyy vaiheeseen kolme. Kolmannessa vaiheessa kerrotaan, että suunniteltu kierros on valittu ja tallennettu laitteeseen. Tämän näkyessä pääohjainyksikkö tallentaa suunnitelman EEPROM-muistiin, käynnistyy uudelleen ja aloittaa jälleen päävalikon alusta.



Kuva 9. Kuvakaappaukset suunnittelun etenemisestä

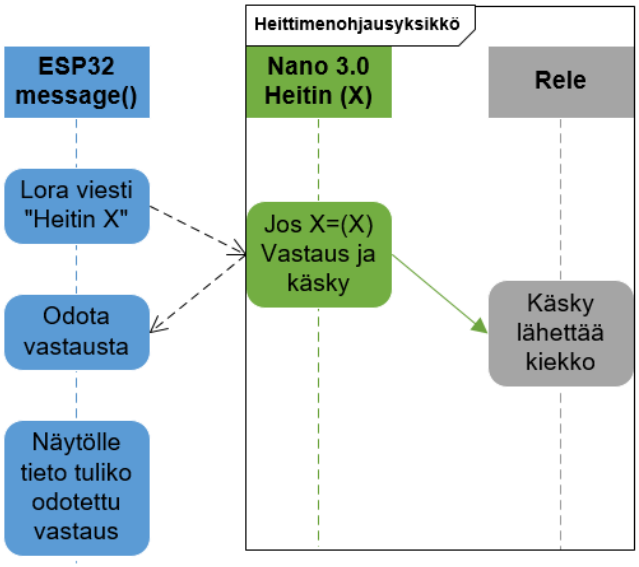
5.5 Heitinohjausyksikkö ja viestin lähetys

Heitinohjausyksikön toiminta alkaa pääohjausyksiköltä lähtevästä viestistä. Pääohjausyksikön message-funktio on vastuussa viestin lähettämisestä, ja siksi tässä yhteydessä kuvataan myös sen toiminta. Heitinohjausyksikkö odottaa pääohjausyksiköltä itselleen kuuluvaa viestiä ja sen saatuaan antaa omalle heittimelleen käskyn heittää kiekon. Kiekon

heittämisen lisäksi heitinohjausyksikkö vastaa pääohjausyksikölle, että viesti saapui. Tämä viestikuittausominaisuus auttaa selvittämään yhteysongelmaa vikatilanteissa.

Kun käyttäjä päättää vahvistaa missä tahansa aliohjelmassa kiekon heittämisen, niin pääohjausyksikkö siirtyy message-funktioon (kuvassa 10 sinisellä pohjalla). Message-funktio saa vahvistuksen yhteydessä sen heittimen kirjaimen, jolle viesti lähetetään. Message-funktio liittää kirjaimen lähetettävään viestiin ja pakkaa viestin LoRa-moduulin lähetettäväksi. Lähetyksen jälkeen pääohjausyksikön LoRa-moduuli siirretään pieneksi hetkeksi kuuntelevaan tilaan. Jokainen kuuluvuusalueella oleva heitinohjausyksikkö vastaanottaa pääohjausyksikön lähettämän viestin. Heitinohjausyksikkö muuntaa viestin merkkijonomuuttujaksi ja vertaa merkkijonoa nimekseen annettuun merkkijonoon (kuvassa 10 vihreällä pohjalla). Merkkijonon ollessa sama, kuin heitinohjausyksikön nimeksi annettu merkkijono, niin heitinohjausyksikkö antaa releelle käskyn lähettää kiekon (rele kuvassa 10 harmaalla pohjalla) ja lähettää pääohjausyksikölle viestin ”viesti ok”. Ellei viestinä saapunut merkkijono ole sama kuin nimeksi annettu merkkijono, niin heitinohjausyksikkö ei tee mitään ja jatkaa viestien kuuntelua.

Pääohjausyksikkö on viestin lähettämisen jälkeen kuuntelutilassa ja vastaanottaessaan viestin tarkistaa onko se odotettu ”viesti ok”-viesti. Mikäli viesti on odotettu viesti, niin pääohjausyksikkö näyttää kahden sekunnin ajan näytöllä tekstin ”viestikuittaus OK”. Jos viestiä ei saavu tai saapunut viesti on väärä, niin pääohjausyksikkö näyttää kahden sekunnin ajan näytöllä tekstin ”Viestikuittaus EI SAAPUNUT”. Nämä tekstit näkyvät näytön toisella rivillä. Meni viesti perille tai ei, niin ohjelma poistuu Message-funktiosta ja aliohjelmassa käyttäjä voi päättää miten edetään.



Kuva 10. Heitinohjausyksikkö ja viestin lähetys

6 Laitteen kokoaminen toimintaan

Projektiin liittyen tehtiin heti alustavia määrittelyjä ja pohdittiin huomioon otettavia seikkoja, kuten rikkinäisen heitinohjausyksikön mahdollisuus tai lähdössä hajoavan kiekon mahdollisuus. Nämä määrittelyt täsmentyivät projektin aikana ja ohjelmoitujen esimerkkien avulla. Laitteella on mahdollista suorittaa vakiokierros, satunnaiskierros, näyttökierros ja vakiokierroksen suunnittelu. Vakio ja satunnaiskierrokseen voi osallistua 1-6 ampujaa. Vakiokierroksella ja satunnaiskierroksella on mahdollista ottaa uudestaan yksittäiskiekot ja kaksoiskiekot, kaksoiskiekko myös heti ensimmäisen kiekon jälkeen. Vakio- ja satunnaiskierroksella on lisäksi mahdollisuus ohittaa ampujan vuoro, mikäli tämä poistuu tai poistetaan kesken kierroksen.

6.1 Osien testaaminen

Aluksi jokaiselle osalle tehtiin käyttökoe toiminnan selvittämiseksi. Kokeiltiin painonapin toimintaa ledin kanssa, ohjelmoitiin näytölle ”Hello World!” teksti ja tehtiin yksinkertainen yhteyskokeilu LoRa-moduuleilla. Pikkuhiljaa kokonaisuutta laajennettiin ja siihen lisättiin uusia toimintoja.

Kolmen painonapin lisäämisen kohdalla oli jonkin aikaa ongelmia, kun laite tuntui arpovan, mitä nappia painettiin. Aiemmin kahdella painonapilla koodi oli toiminut aivan hyvin. Kun tilannetta tutkittiin, niin todettiin laitteen saavan kaikille painonapeiksi määritellyille pinneille virtaa, kun yhtä painonappia painettiin. Tässä kohtaa syyksi selkeytyi vastusten puuttuminen ja ratkaisuna lisättiin vastukset painonappeihin.

Useamman LoRa-moduulin testeissä ongelmia ilmeni kahden LoRa-moduulin kanssa. Näiden kahden LoRa-moduulin juotokset purettiin ja tehtiin uudelleen. Näin toinen LoRa-moduuleista alkoi toimia moitteettomasti, mutta toinen LoRa-moduuli jäi rikkinäiseksi. Rikkinäinen LoRa-moduuli onnistui vastaanottamaan viestin, mutta yrittäessään lähettää viestiä se jäi jumiin.

6.2 Langaton yhteys

Projektin onnistumisen kannalta kaikkein kriittisin kohta oli langattoman viestiyhteyden luominen. Projektin ensimmäiset testit tehtiin LoRa-moduulien kanssa. LoRan käyttöön liittyvä toimintasuhteen rajoitusten selvittyä tehtiin kokeiluja myös muilla täysin vapailla taajuuksilla toimivilla radiolähettimillä. Kokeilut eivät tuottaneet toivotunlaista yhteyttä ja palattiin takaisin LoRan käyttöön.

LoRaa käytettäessä oli tehtävä laskelmat laitteen toimintasuhteesta. Sillä viestintäviraston määräyksen 15 mukaan tällaisen laitteen toimintasuhde voi olla korkeintaan yksi prosentti (tarkemmin luvussa 3.3). Lähettämisajan selvittämiseksi The Thing Networkin (Thethingsnetwork 2020) internetsivustolla on laskuri, jolla lähetysajan voi laskea. Tässä projektissa lähetettävä bittimäärä on kahdeksan, hajotuskerroin (SF) seitsemän, alue EU ja kaistanleveys 125kHz. Laskuri kertoo näillä tiedoilla, että yhden viestin lähettämiseen laite käyttää 56,6 millisekuntia. Tiedossa on, että prosentin toimintasuhteella lähetysaikaa voi tunnissa olla korkeintaan 36 sekuntia, joten tunnin aikana voidaan lähettää (36000/56,6) 636 viestiä. Tällöin viesti voidaan toimintasuhteen rajoissa lähettää noin kuuden sekunnin välein. Viiden ampujan ja (5*25) 125 kiekon kierros kestää kokeneilla ampujilla arvion mukaan noin 25-30 minuuttia. Tällöin kiekko täytyy keskimäärin lähettää (1500/125) 12-14 sekunnin välein.

6.3 Käyttö ja käyttöliittymä

Laitteen peruskäyttö eli kierrosten ohjaaminen on pyritty pitämään mahdollisimman yksinkertaisena. Painonapit on tarkoituksella valittu riittävän isoiksi, jotta niiden painaminen onnistuu myös talvikäsineillä. Painonappien suuri koko ja painovastus todennäköisesti vähentävät myös vahinkopainalluksia ja kaksoispainalluksia. Painonappien käyttötarkoitus on aina sama. Vihreällä painonapilla kaikki toiminnot hyväksytään tai vahvistetaan. Mustalla painonapilla vähennetään tai ohitetaan. Keltaisella painonapilla uusitaan ja lisätään.

Vakiokierroksen suunnittelun käyttöliittymä on hankalampi jo sen vuoksi, että on osattava yhdistää laite pääohjausyksikön ylläpitämään WLAN-verkkoon. Oletuksena on kuitenkin, että vakiokierroksen uudelleen suunnittelua tehdään sen verran harvoin, että se voi olla harvemman ihmisen hallittavissa. Selaimen puolella tehtävät toiminnot kuitenkin ovat helposti tehtävissä. Kompastuskiveksi voi muodostua se, että paikan kiekot pitää kirjoittaa kaikki peräkkäin. Jatkossa tämän voisi suunnitella siten, että tekstikenttiä tulee ampumaohjelman mukaisesti ja jokaiseen kenttään kirjoitetaan vain yhden kierroksen kiekot.

6.4 Toiminnallisuuden testaaminen

Määritellyt toiminnallisuudet on testattu viemällä kaikki toiminnot läpi neljän heitinohjausyksikön kanssa. Testit on COVID19-viruksen myötä tulleiden poikkeusolojen vuoksi suoritettu sisätiloissa ilman kiekonheittimiä siten, että heittokäskyn saatuaan heitinohjausyksikkö on sytyttänyt ledin. Testissä on testattu kiekkojen uusimiset ja ampujan vuoron ohittaminen. Laajemmat käyttötestit on tarkoitus suorittaa myöhemmässä vaiheessa Sotkan ampumaradalla.

7 Johtopäätökset

Projektiin lähdettäessä onnistumisen mahdollisuus tuntui hyvin epävarmalta, sillä olin tehnyt muutamia epäonnistuneita kokeiluja LoRan käyttämiseksi Raspberry Pi 3n kanssa ja palvelimen perustamisessa ESP8266 ESP-01llä GREATZT Nano mikrokontrollerimoduuliin. Tunsin vain vähän laitteiston toimintaa ja niiden ohjelmointia.

Toimivan prototyypin tekeminen projektissa onnistui ja projektille asetettuihin tavoitteisiin päästiin, vaikka käytännön testaus jää myöhemmäksi. Seuraava vaihe projektissa olisi käytännön testaaminen kahdeksan heitinohjausyksikön kanssa ja palautteen kerääminen käyttäjiltä. Oppimisen kannalta merkittävimpänä kokemuksena voidaan pitää projektin onnistunutta läpivientiä, sillä projektin alussa oli hyvin paljon epävarmuustekijöitä. Käytännön kokemusta kertyi eri laitteistoista ja niiden soveltamisesta, sekä niiden ohjelmoinnista.

7.1 Käytettävyys

Prototyypillä onnistuu compak sporting -sääntöjen mukainen kierroksen ohjaaminen muutaman poikkeustilanteen huomioiden. Laitteiston käyttämällä tekniikalla voitaisiin tehdä myös muihin haulikkoammuntalajeihin soveltuvat järjestelmät. Samaa tekniikkaa voidaan käyttää muissakin etäohjausta tarvitsevissa sovellutuksissa. Käänteisesti tekniikkaa käyttäen olisi mahdollista tehdä vaikkapa pyyntiloukkuihin hälytinjärjestelmä.

Nykyisessä järjestelmässä, jokaiselle heittimelle on oma nappinsa ja ampumaohjelma pitää olla ammunnanohjaajalla tiedossa. Prototyyppi muuttaa nykyisin käytössä olevaa järjestelmää siten, että ammunnanohjaajan ei tarvitse tietää miltä heittimeltä kiekko tulee lähettää. Nykyisessä järjestelmässä kauko-ohjaimen radiovastaanotin on pääkopissa, jossa vastaanottimeen liitetyistä releistä tieto lähtee johtimia pitkin heittimille. Prototyypissä radiovastaanottimet ja releet voidaan viedä suoraan heittimelle, jolloin tarvittavien johtimien määrä vähenee. Sotkan ampumaradalla ukkosmyrskyt ovat aiheuttaneet laiterikkoja ja tämän johdinten vähentämisen ansiosta laiterikkojen uskotaan vähenevän. Ainakaan kaikkien laitteiden ei pitäisi rikkoutua yhdestä salamasta, kun eivät ole enää johtimin kytkettyinä toisiinsa. Suunniteltujen kehityskohteiden ja käyttötestien perusteella tehtävän jatkojalostuksen myötä järjestelmän käytettävyys tulee paranemaan vielä paljon.

Prototyyppinä kehitetyn järjestelmän haittapuolena on, että osien saapuminen etenkin LoRa-moduulien saapuminen toimittajalta kestää kauan. Joten niitä on varattava riittävästi varaosiksi. Näyttökierroksella heitintaulukon eri päissä olevien heittimien peräkkäinen lähettäminen on suhteellisen hidasta, kun pitää selata koko taulukko läpi.

7.2 Kehitettävää

Kehitettävistä asioista sääntöjen kannalta tärkein on, että yhtäaikaisen kaksoiskiekon mahdollisuus lisätään järjestelmään. Painikkeiden määrää tullaan lisäämään neljään, jolloin käytettävyys paranee ja painikkeiden käyttötarkoituksia voidaan selkeyttää. LoRa-kommunikaatiota voisi kehittää pienentämällä lähetettävää viestiä ja lisäämällä kaistanleveyttä. LoRa-lähetyksen kuittausviesti voisi sisältää tiedon, miltä heittimeltä kuittaus tulee. Tieto auttaa havaitsemaan, jos bittivirheen vuoksi väärä heitin vastaa. Lisäksi, kun viestikuittausta ei tule, pääohjausyksikkö voisi automaattisesti lähettää toisen viestin ja odottaa taas kuittausta. Näiden LoRa-kommunikaatiomuutosten todellisen tarpeen havaitsee varmasti tarkemmin oikeassa käyttöympäristössä tehtävissä käytännön testeissä.

Jatkokehityksen osalta ensimmäisenä tehtävänä on osumien ja hutien merkkaaminen järjestelmään. Tämä mahdollistaisi kierroksen tuloksen tallentamisen ja näyttämisen erillisessä tulostaulussa. Tämän jälkeen olisi tarpeen mukaan mahdollista lisätä pidempiaikaista tulosten tallentamista tai vaikka ennätystuloksen tallentaminen ja näyttäminen. Kilpailutoimintaa silmällä pitäen järjestelmään pitäisi mahdollistaa päätuomarille oikeus muuttaa pääohjausyksiköstä annettua osuma/ohi tuomiota. Tämä tulosten tallentaminen olisi sellainen ominaisuus, joka on herättänyt kiinnostusta muillakin ampumaradoilla. Näyttökierrokselle voisi lisätä viimeiseksi satunnaisheittimen, jolloin sitäkin voisi harjoitella ja näyttää ilman koko kierroksen läpi käymistä.

Tekemääni ohjelmakoodia voisi korjata käytänteiden mukaiseksi ja parantaa rakenteellisesti. Ainakin osan funktioista voisi siirtää itse tehtyyn kirjastoon, jolloin mahdollisesti muuttuvat osat olisivat paremmin esillä ja ohjelmakoodi helpommin luettavissa. Poikkeustilanteiden huomioimista vielä laajemmalle täytyy lisätä, mutta tähän saa varmasti paremmin työkaluja muutamien käytännön testien jälkeen. Ainakin kierrokselta pitäisi pystyä poistumaan kesken kierroksen ilman koko kierroksen läpikäyntiä tai laitteen uudelleenkäynnistämistä. Kesken jääneen kierroksen kierrostilanteen tallentamista voisi harkita.

7.3 Riskien toteutuminen

Tunnistin projektin riskeiksi osaamiseen puutteet, ajankäytön hallitsemattomuuden, budjetin ylittymisen ja kriittisten osien rikkoutumiset. Kaikki ennalta tunnistetut riskit toteutuivat jossain määrin. Osaamisen puutteet aiheuttivat monessa kohtaa hidasteita. Tarkoitus oli toiminnan varmistamisen jälkeen tilata loput LoRa-moduuleista, mutta toiminnan varmistuessa selvisi samaan aikaan toimintasuhderajoitus, joka olisi ollut hyvä tietää ajoissa. Tä-

män vuoksi en tilannut tuossa kohtaa lisää LoRa-moduuleja vaan toisia LRF24L01-moduuleita, jotka osoittautuivat projektiin sopimattomiksi. Lopulta LoRa-moduuleita ei tämän vuoksi saapunut ajoissa tarpeeksi montaa kahdeksan heitinohjausyksikön tarpeisiin. Samasta syystä myös budjetti ylittyi suunnitellusta.

Suomeen julistettiin projektin aikana poikkeusolosuhteet COVID19-viruksen leviämisen estämiseksi. COVID19-viruksen vuoksi asetettuja liikkumisrajoituksia ja tapaamisrajoituksia en osannut ennakoida. Ajankäytön hallitsemattomuus toteutui projektissa siten, että COVID19-viruksen myötä lasta ei suositeltu vietäväksi päiväkotiin. Hoitaessani lasta kotona, en projektin loppuvaiheessa pystynytkään käyttämään suunniteltua tuntimäärää projektiin. Viruksen myötä tulleiden liikkumis- ja tapaamisrajoitusten myötä, myös testausmatka Kauhajoelle oli peruttava.

7.4 Projektin kulku ja oppiminen

Projektissa ajankäyttö jakautui puoliksi laitteiston parissa työskentelyyn ja lopullisen ohjelmakoodin kokoamiseen. Suurin osa ajasta laitteiston parissa kului toimivan esimerkin löytämiseen ja sen muokkaamisessa projektin tarpeisiin. Laitteiston kanssa työskennellessä noin kolmasosa ajasta kului manuaalien tutkimiseen. Oikean laitteiston hakemisessa kokeiltiin myös, vaihtoehtoja joita ei lopullisessa työssä otettu käyttöön. Ohjelmoinnin osalta hankalinta oli hahmottaa millä tavoin compak sporting -kierroksen kulku voidaan ohjelmallisesti toteuttaa. Alun perin yritettiin selvittää kaavaa millä kierroskulun järjestyksen olisi voinut laskea, mutta päädyttiin lopulta ampujamäärän mukaan valmiiksi laskettuihin taulukoihin. Itselleni tämä oli myös ensimmäinen itsenäinen ohjelmointityö ja siksi kirjoitin ohjelmaa muutamaan kertaan uudelleen sen selkeyttämiseksi.

Helpoimmin osoitettavissa oleva oppiminen on tapahtunut ohjelmointi taidon kehittymisessä. Samalla ohjelmointi on osa-alue, jossa edelleen on eniten opittavaa. Jos nyt aloittaisin ohjelmoimaan projektia uudelleen, niin ohjelmakoodista tulisi hyvin erilainen ja koittaisin tehdä omia kirjastoja yksinkertaisille toistuville ohjelmapätkille. Ohjelmakoodi olisi myös tiivistettävissä ja globaalien muuttujien määrää voisi edelleen karsia, jotta dynaamisen muistin käyttö vähenisi. Kun tätä prototyyppiä lähdetään kehittämään eteenpäin, voidaan määrittelyt tehdä ennakkoon ja ohjelmoida määrittelyjen pohjalta. Tässä projektissa en olisi vielä hahmottanut ohjelmakoodin kokonaisuutta valmiiden määrittelyjen pohjalta ja käytetty asteittain etenevä lähestymistapa sopi itselleni oikein hyvin. Seuraavassa ohjelmointiprojektissa osaan varmasti paremmin hahmottaa kokonaisuuden jo määrittelyvaiheessa ja voin valita ohjelmointiin jonkin tietyn lähestymistavan.

Jälkikäteen on helppo sanoa, että LoRan käyttämän radiotaajuuden käyttörajoituksiin olisi pitänyt tutustua aiemmin ja muitakin ongelmia olisi voinut välttää huolellisemmalla pohjatyöllä. Itselleni monet projektissa kohdatut aihealueet olivat kuitenkin täysin uusia, kuten nämä radiotaajuuksien käyttörajoitukset, joten en olisi osannut niitä ennakkoon selvittää kuitenkaan. Projektin aikana on kuitenkin oppinut taas paljon uusia asioita ja paikkoja joista tietoa saa hankittua. Seuraavaan projektiin lähdettäessä on taas hieman valmiimpi tekemään perusteellisempaa pohjatyötä.

Projektin ajankäyttö oli mielestäni hyvin hallinnassa, vaikka lopussa ajankäyttö rajoittui suunniteltua pienemmäksi COVID19-viruksen sivuseuraamusten myötä. Itselleni on kuitenkin aina ollut luontevaa suunnitella ajankäyttö siten, että aluksi tehdään mieluummin ylimääräistä, että ongelmiin törmätessä lopussa on varaa lisätä työmäärää tai kaiken mennessä hyvin tehdä huolellinen viimeistely. Työmäärän arviointi sujui myös yllättävän hyvin ja siksikin ajankäytön suunnitelmassa pysyttiin hyvin. Suunniteltu työjärjestys muuttui ohjelmoinnin osalta paljonkin, sillä oli luontevampaa edetä suunnitellusta poiketen pienimmästä vaiheesta suurimpaan. Näyttökierroksen ja satunnaiskierroksen työmäärä oli vakiokierrosta pienempi, joten suunnittelusta poiketen tein niiden ohjelmallisen toteutuksen ensin. Lisäksi näyttökierroksen ohjelmakoodia oli hyvä käyttää laitteiston testauksen apuna ja laitteiston yhteistoiminnan perusteiden haltuun ottamisessa.

Lähteet

AliExpress 2020a. GREATZT Nano 3.0 tuotekuvaus. Luettavissa: https://www.aliexpress.com/i/32804743935.html. Luettu: 14.2.2020

AliExpress 2020b. SX1276 LoRa Module tuotekuvaus. Luettavissa: https://www.aliexpress.com/item/32984655636.html?spm=a2g0o.product-list.0.0.5e126536Y2wnzl&algo_pvid=7a70428b-8345-4437-aa44-66f52fa1be03&algo_expid=7a70428b-8345-4437-aa44-66f52fa1be03-4&btsid=0be3769015867091288943499e3eac&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_ Luettu: 12.4.2020

Ampumaurheiluliitto 2020. Compak-sporting. Luettavissa: https://www.ampumaurheiluliitto.fi/haulikko/sporting/compak-sporting/. Luettu: 14.2.2020

Ampumaurheiluliitto 2017. Sportinglajien säännöt, s. 86-88, 92-96, 109 ja 111. Suomen ampumaurheiluliitto. Luettavissa: https://www.ampumaurheiluliitto.fi/wp-content/uploads/2016/09/Sporting-2017-netti-1.pdf. Luettu: 11.4.2020

Arduino 2020a. Arduino, AboutUs. Luettavissa: https://www.arduino.cc/en/Main/AboutUs. Luettu: 17.2.2020

Arduino 2020b. Arduino, Policy. Luettavissa: https://www.arduino.cc/en/main/policy. Luettu: 17.2.2020.

Arduino 2020c. Arduino, What is Arduino? Luettavissa: https://www.arduino.cc/en/guide/introduction. Luettu: 4.3.2020

Arduino 2020d. Arduino, Libraries. Luettavissa: https://www.arduino.cc/en/reference/libraries. Luettu: 4.3.2020

Atmel 2015. ATmega328P datasheet. Atmel Corporation. s. 1-2. Luettavissa: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. Luettu: 14.2.2020

Banzi, M. 2011 Arduino – perusteista hallintaan. 2. painos. Suom. Mäenpää, Y. Hämeenlinna: Robomaa.com Oy.

Elecrow 2020a. Verkkokaupan tuotekuva. Luettavissa: https://www.elecrow.com/esp32-wifi-ble-board.html. Luettu: 13.2.2020

Elecrow 2020b. Elecrow wikisivusto ESP32 kehitysalustalle. Luettavissa: https://www.elecrow.com/wiki/index.php?title=ESP32_WIFI/BLE_Board_v1.0. Luettu: 13.2.2020

Espressif 2019. ESP32-WROOM-32 Datasheet. Version 2.9, Espressif Systems, s. 1-2. Luettavissa: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Luettu: 13.2.2020

Github 2020a. Espressif Systemsin Arduino-esp32 kirjasto. Luettavissa: https://github.com/espressif/arduino-esp32. Luettu: 20.2.2020

Github 2020b. ESP32 WiFi kirjaston WiFi.h. Luettavissa: https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h. Luettu: 20.2.2020

Github 2020c. LiquidCrystal_I2C. Luettavissa: https://github.com/johnrickman/LiquidCrystal_I2C/blob/master/library.properties. Luettu: 6.3.2020

Github 2020d. Arduino-LiquidCrystal-I2C-library. Readme.md. Luettavissa: https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library. Luettu: 6.3.2020

Github 2020e. ESP32 EEPROM-kirjaston EEPROM.h. Luettavissa: https://github.com/espressif/arduino-esp32/blob/master/libraries/EEPROM/src/EEPROM.h Luettu: 13.4.2020

Github 2020f. Arduino-LoRa-kirjasto. Luettavissa: https://github.com/sandeepmistry/arduino-LoRa. Luettu: 13.4.2020

Jansons, J. & Dorins, T. 2012. Analyzing IEEE 802.11n standard: outdoor performance. International Conference on Digital Information Processing and Communications (ICDIPC), Klaipeda City. Luettavissa: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6257274. Luettu: 2.3.2020

Joy-it, 2017. I2c Serial 2.6“LCD Module, datasheet. Joy-it 8.9.2017. Luettavissa: http://anleitung.joy-it.net/wp-content/uploads/2016/09/SBC-LCD16x2-Datasheet.pdf. Luettu: 6.3.2020

Kolban, N. 2018. Kolban’s Book on ESP32, September 2018. s. 60-63. E-kirja.

Li, L. Yu. Z. & Hao. Y. 2011. "Estimates of EEPROM device lifetime," *Tsinghua Science and Technology*, vol. 16, no. 2, s. 170, April 2011. Luettavissa: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6077983. Luettu 13.4.2020

LoRa Alliance. 2020. RP002-1.0.1 LoRaWAN® Regional 40 Parameters. s.13. LoRa Alliance 20.2.2020, Fremont, CA. Luettavissa: https://lora-alliance.org/sites/default/files/2020-02/rp_2-1.0.1.pdf. Luettu: 12.4.2020

Nussey, J. 2013. Arduino for dummies. John Wiley & Sons, Ltd. West Sussex. E-kirja.

NXP Semiconductors 2014. UM10204, I 2C-bus specification and user manual. Rev. 6. NXP Semiconductors, 4.4.2014. Luettavissa: https://www.nxp.com/docs/en/user-guide/UM10204.pdf Luettu: 13.4.2020

RevSpace 2016. Kuvituskuva LoRa lowdatarate4.png Luettavissa: https://revspace.nl/File:Lora_lowdatarate4.png#filelinks. Luettu: 13.4.2020.

SE 2020. Schneider Electronics. Tuotekuvaus, Painike vihreä Harmony XB5. Luettavissa: https://www.se.com/fi/fi/product/XB5AA31/painike%2C-vihre%C3%A4%2C-no/. Luettu: 13.4.2020

Seneviratne P. 2019. Beginning LoRa Radio Networks with Arduino: Build Long Range, Low Power Wireless IoT Networks. Apress. E-kirja.

Thethingsnetwork 2020. LoRaWAN airtime calculator. The Things Network. Luettavissa: https://www.thethingsnetwork.org/airtime-calculator. Luettu: 25.4.2020

Traficom 2019. Määräys, luvasta vapaiden radiolähettimien yhteistaajuuksista ja käytöstä. 15 AO/2019 M. s. 6-8. Liikenne ja viestintävirasto, Traficom. Helsinki. Ladattavissa: https://www.finlex.fi/data/normit/44836/Maarays_15AO_FI.pdf. Ladattu: 4.3.2020

Wheat, D. 2011. Arduino Internals. Luku 3. E-kirja.

Wikipedia 2010. Kuvituskuva lineaarisesta taajuuden kasvamisesta suhteessa aikaan. Georg-Johann, 28.8.2010. Kuva lisensoitu CC BY-SA 3.0 lisenssillä. Luettavissa: https://en.wikipedia.org/wiki/Chirp_spread_spectrum. Luettu: 12.4.2020

Liitteet

Liite 1. Sähköposti liquidcristal-i2c kirjaston käytöstä

Lähettäjä: Marco Schwartz <marcolivier.schwartz@gmail.com>
Lähetetty: tiistai 18. helmikuuta 2020 22.04
Vastaanottaja: Jarno Wermundsen
Aihe: Re: LiquidCrystal_I2C

Hi Jarno,

Sure, feel free to use it! I basically just uploaded this library on GitHub from an old zip file so everybody can use
it.

Cheers,
Marco

On Thu, Feb 13, 2020 at 12:21 AM Jarno Wermundsen <jarno.wermundsen@gmail.com> wrote:
Hi,

I found recommendation in randomnerdtutorials.com for this library you made. I would use it in my thesis work, but there are no any license mentioned?
Is it possible to use this library?

Best regards
Jarno Wermundsen

Liite 2. Ammunnan kulku

Kierros 1	**Paikka 1**	**Paikka 2**	**Paikka 3**	**Paikka 4**	**Paikka 5**	**Paikka 6**
	Ampuja **1**	Ampuja **2**	Ampuja **3**	Ampuja **4**	Ampuja **5**	Ampuja **6**
Vuoro 1	C	E	A	F	D	
Vuoro 2	D+B	B+F	F+C	C+A	A+F	
Vuoro 3	F+A	A+D	D+E	E+B	B+C	
Kierros 2	**Paikka 1**	**Paikka 2**	**Paikka 3**	**Paikka 4**	**Paikka 5**	**Paikka 6**
	Ampuja **6**	Ampuja **1**	Ampuja **2**	Ampuja **3**	Ampuja **4**	Ampuja **5**
Vuoro 1	C	E	A	F	D	
Vuoro 2	D+B	B+F	F+C	C+A	A+F	
Vuoro 3	F+A	A+D	D+E	E+B	B+C	
Kierros 3	**Paikka 1**	**Paikka 2**	**Paikka 3**	**Paikka 4**	**Paikka 5**	**Paikka 6**
	Ampuja **5**	Ampuja **6**	Ampuja **1**	Ampuja **2**	Ampuja **3**	Ampuja **4**
Vuoro 1	C	E	A	F	D	
Vuoro 2	D+B	B+F	F+C	C+A	A+F	
Vuoro 3	F+A	A+D	D+E	E+B	B+C	
Kierros 4	**Paikka 1**	**Paikka 2**	**Paikka 3**	**Paikka 4**	**Paikka 5**	**Paikka 6**
	Ampuja **4**	Ampuja **5**	Ampuja **6**	Ampuja **1**	Ampuja **2**	Ampuja **3**
Vuoro 1	C	E	A	F	D	
Vuoro 2	D+B	B+F	F+C	C+A	A+F	
Vuoro 3	F+A	A+D	D+E	E+B	B+C	
Kierros 5	**Paikka 1**	**Paikka 2**	**Paikka 3**	**Paikka 4**	**Paikka 5**	**Paikka 6**
	Ampuja **3**	Ampuja **4**	Ampuja **5**	Ampuja **6**	Ampuja **1**	Ampuja **2**
Vuoro 1	C	E	A	F	D	
Vuoro 2	D+B	B+F	F+C	C+A	A+F	
Vuoro 3	F+A	A+D	D+E	E+B	B+C	
Kierros 6	**Paikka 1**	**Paikka 2**	**Paikka 3**	**Paikka 4**	**Paikka 5**	
	Ampuja **2**	Ampuja **3**	Ampuja **4**	Ampuja **5**	Ampuja **6**	
Vuoro 1	C	E	A	F	D	
Vuoro 2	D+B	B+F	F+C	C+A	A+F	
Vuoro 3	F+A	A+D	D+E	E+B	B+C	

Liite 3. Heitinohjausyksikön ohjelmakoodi

```
#include <SPI.h>
#include <LoRa.h>

// Releen pinni
int command = 4;

// Viestin string muuttujat
String packet;
String fullPacket;

// Heittimen nimeäminen
String iAm("Heitin A");

void setup() {
  LoRa.setPins(10, 9, 2);
  Serial.begin(115200);
  pinMode(command, OUTPUT);
  if (!LoRa.begin(866E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}

void loop() {
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    while (LoRa.available()) {
      packet = ((char)LoRa.read());
      fullPacket = fullPacket + packet;
    }
    if (fullPacket == iAm) {
      digitalWrite(command, HIGH);
      LoRa.beginPacket();
      LoRa.print("Viesti ok");
      LoRa.endPacket();
      digitalWrite(command, LOW);
    }
    fullPacket = "";
  }
}
```

Liite 4. Pääohjausyksikön ohjelmakoodi

```
// Tarvittavat kirjastot
#include <WiFi.h>
#include <WiFiClient.h>
#include <SPI.h>
#include <LoRa.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>

#define EEPROM_SIZE 30

// Näytön osoite, merkki- ja rivimäärä
LiquidCrystal_I2C lcd(0x27, 16, 2);

//Wifi-yhteyksien maksimimäärä
int maxConnections = 1;

//Nappien pinnimääritys
int buttonR = 25;
int buttonL = 26;
int buttonC = 27;

//IP-osoitteen "olio".
IPAddress myIP;

//Web-palvelun portti
WiFiServer server(80);

//WLAN-verkon nimi ja salasana
const char *ssid = "**********";
const char *password = "**********";

// Heitinten taulukkomuuttuja
char throwers[10] = "BCDE";

// Ampumapaikkojen taulukkomuuttuja
int shootPlaces[6] = {1, 2, 3, 4, 5, 6};
```

```
// Paikkojen yhteismäärä eli ampumapaikkojen taulukkomuuttujan koko
const int totalPlaces = sizeof(shootPlaces) / sizeof(shootPlaces[0]);

// Muuttuja ampujien määrälle
int shooters;

// Taulukkomuuttuja ampujien vuoroille
int shooterTurns[(6 * totalPlaces)];

// Taulukkomuuttuja vuorossa oleville ampumapaikoille
int shooterPlaces[6 * totalPlaces];

// Satunnaiskierroksen ampumaohjelma
int randRounds[4] = {1, 1, 1, 2};

// Taulukkomuuttuja vakiokierroksen ampumaohjelmalle
int defRounds[5];

// Taulukkomuuttuja paikkojen kiekko ohjelmalle
char clayOrder[25];

// muuttuja EEPROM tallennus sijainnille
int eeCount = 0;

void setup() {
  //Nappi pinnien asetus tulopinneiksi
  pinMode(buttonL, INPUT);
  pinMode(buttonR, INPUT);
  pinMode(buttonC, INPUT);

  //Näytön asettaminen valmiuteen.
  lcd.init();
  lcd.backlight();

  //Sarjaportin avaus liikenteelle
  Serial.begin(115200);
  Serial.println();
```

```
  // Loran herättäminen valmiuteen.
  LoRa.setPins(5, 14, 15);
  Serial.println("Avataan LoRa");
  if (!LoRa.begin(866E6)) {
    Serial.println("LoRan avaus ei onnistu!");
    while (1);
  }

  // Oletus kierroksen hakeminen EEPROM muistista.
  if (!EEPROM.begin(EEPROM_SIZE))
  {
    Serial.println("failed to initialise EEPROM"); delay(1000000);
  }
  int x = 0;
  for (int i = 0; i < EEPROM_SIZE; i++)
  {
    if (i < 5) {
      defRounds[i] = EEPROM.read(i);
    }
    if (i >= 5) {
      clayOrder[x] = char(EEPROM.read(i));
      x++;
    }
  }

  // Tukiaseman asettaminen valmiuteen
  //(verkon nimi, salasana, kanava, verkon piilotus, sallittujen yhteyksien määrä)
  WiFi.softAP(ssid, password, 1, 0, maxConnections);
  myIP = WiFi.softAPIP();
  server.begin();
}

void loop() {
  bool pick;
  //Näytöllä kysytään otetaanko oletuskierros
  pick = mainChoose("Vakiokierros");
  if (pick) {
```

```
    competition();
  }
  //Näytöllä kysytään otetaanko satunnaiskierros.
  pick = mainChoose("Satunnaiskierros");
  if (pick) {
    randomCompetition();
  }
  //Näytöllä kysytään otetaanko valintakierros
  pick = mainChoose("Nayttokierros");
  if (pick) {
    pickRound();
  }
  //Näytöllä kysytään aloitetaanko suunnittelu
  pick = mainChoose("Suunnittelu");
  if (pick) {
    webPlanning();
  }
}

void pickRound() {
  int x = 0;
  int chosen;
  while (1) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Heitin ");
    lcd.print(throwers[x]);
    lcd.setCursor(0, 1);
    lcd.print("OK:V Ohi:M ++:K");
    chosen = choose();
    if (chosen == 2) {
      char c = throwers[x];
      message(c);
    }
    if (chosen == 3) {
      x++;
      if (throwers[x] == 0) {
        x = 0;
```

```
      }
    }
    if (chosen == 1) {
      break;
    }
  }
}

bool mainChoose(String showThis) {
  int chosen;
  while (1) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(showThis);
    lcd.setCursor(0, 1);
    lcd.print("OK:V Ohi:M");
    chosen = choose();
    if (chosen == 2) {
      return true;
    }
    if (chosen == 3) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Valitsit K");
      lcd.setCursor(0, 1);
      lcd.print("OK:V Ohi:M");
      delay(5000);
    }
    if (chosen == 1) {
      return false;
    }
  }
}

int choose() {
  while ((digitalRead(buttonL) == LOW) || (digitalRead(buttonC) == LOW) || (digi-
talRead(buttonR) == LOW)) {
    if (digitalRead(buttonL) == HIGH) {
```

```
      delay(100);
      while (digitalRead(buttonL) == HIGH) {
      }
      return 1;
    }
    if (digitalRead(buttonC) == HIGH) {
      delay(100);
      while (digitalRead(buttonC) == HIGH) {
      }
      return 2;
    }
    if (digitalRead(buttonR) == HIGH) {
      delay(100);
      while (digitalRead(buttonR) == HIGH) {
      }
      return 3;
    }
    else {
      // älä tee mitään
    }
  }
}

int howMany (String showThis) {
  int howMany = 1;
  int chosen;
  while (1) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(showThis);
    lcd.print(howMany);
    lcd.setCursor(0, 1);
    lcd.print("--:M OK:V ++:K");
    chosen = choose();
    if (chosen == 3) {
      howMany++;
      if (howMany > 6) {
        howMany = 1;
```

```
    }
   }
   if (chosen == 1) {
    howMany--;
    if (howMany < 1) {
     howMany = 6;
    }
   }
   if (chosen == 2) {
    return howMany;
   }
  }
}

void competition() {
 shooters = howMany("Ampujia ");
 shootOrder();
 int realRounds = 0;
 for (byte i = 0; i < (sizeof(defRounds) / sizeof(defRounds[0])); i++) {
  if (int(defRounds[i]) > 0) {
   realRounds++;
  }
 }
 int placeChance = 0;
 int y = 0;
 while (placeChance <= 5) {
  for (int t = 0; t < realRounds; t++) {
   int claysOnRound = int(defRounds[t]);
   int oneTwoTwo = 0;
   if (t == 2 && claysOnRound == 2) {
    oneTwoTwo = 1;
   }
   for (int sh = 1; sh <= shooters; sh++) {
    int clayN = 1;
    int shooter = shooterTurns[y];
    int place = shooterPlaces[y];
    if (place != 6) {
     while (clayN <= claysOnRound) {
```

```
            int variable = t + ((clayN - 1) + oneTwoTwo);
            char thrower = countThrower(place, variable);
            int isOk = shootSituation(shooter, place, thrower, (clayN));
            clayN++;
            if (isOk == 1) {
              clayN = 1;
            } else if (isOk == 3) {
              clayN = claysOnRound + 1;
            }
          }
          bool again = takeAgain();
          if (again == true) {
            sh--;
            y--;
          }
        } else {
          //Paikalla 6 ei ammuta
        }
        y++;
      }
      y = y - shooters;
    }
    placeChance++;
    y = y + shooters;
  }
}

char countThrower(int place, int clayNow) {
  switch (place) {
    case 1:
      return char(clayOrder[clayNow]);
    case 2:
      return char(clayOrder[clayNow + 5]);
    case 3:
      return char(clayOrder[clayNow + 10]);
    case 4:
      return char(clayOrder[clayNow + 15]);
    case 5:
```

```
    return char(clayOrder[clayNow + 20]);
  }
}

int shootSituation(int sh, int sP, char thrower, int clay) {
  int chosen;
  if (clay == 1) {
    while (1) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Ampuja ");
      lcd.print(sh);
      lcd.print("Paikka ");
      lcd.print(sP);
      lcd.setCursor(0, 1);
      lcd.print("Kiekko ");
      lcd.print(thrower);
      lcd.print(". OK:V");
      chosen = choose();
      if (chosen == 2) {
        message(thrower);
        return 2;
      }
      if (chosen == 3) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Valitsit K");
        lcd.setCursor(0, 1);
        lcd.print("OK:C");
        delay(2000);
      }
      if (chosen == 1) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Valitsit M");
        lcd.setCursor(0, 1);
        lcd.print("Seuraava ampuja.");
        delay(2000);
```

```
      return 3;
    }
  }
}
if (clay == 2) {
  while (1) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Kiekko ");
    lcd.print(thrower);
    lcd.print(". OK:V");
    lcd.setCursor(0, 1);
    lcd.print("Uudestaan K");
    chosen = choose();
    if (chosen == 2) {
      message(thrower);
      return 2;
    }
    if (chosen == 3) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Valitsit K");
      lcd.setCursor(0, 1);
      lcd.print("OK:V");
      delay(2000);
      return 1;
    }
    if (chosen == 1) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Valitsit M");
      lcd.setCursor(0, 1);
      lcd.print("Seuraava ampuja.");
      delay(2000);
      return 3;;
    }
  }
}
```

```
}

void randomCompetition() {
  shooters = howMany("Ampujia ");
  shootOrder();
  int randMax;
  for (byte i = 0; i < (sizeof(throwers) / sizeof(throwers[0])); i++) {
    if (throwers[i] != 0) {
      randMax++;
    }
  }
  int claysPerRound = 0;
  for (byte i = 0; i < (sizeof(randRounds) / sizeof(randRounds[0])); i++) {
    claysPerRound = claysPerRound + randRounds[i];
  }
  int clayTotal = claysPerRound * shooters * totalPlaces;
  int placeChance = 0;
  int y = 0;
  while (placeChance <= 5) {
    for (int t = 0; t <= (sizeof(randRounds) / sizeof(randRounds[0])) - 1; t++) {
      int clays = randRounds[t];
      for (int sh = 1; sh <= shooters; sh++) {
        int clayN = 1;
        int shooter = shooterTurns[y];
        int place = shooterPlaces[y];
        if (place != 6) {
          while (clayN <= clays) {
            int lottery = random(0, randMax);
            char thrower = throwers[lottery];
            int isOk = shootSituation(shooter, place, thrower, (clayN));
            clayN++;
            if (isOk == 1) {
              clayN = 1;
            } else if (isOk == 3) {
              clayN = clays + 1;
            }
          }
          bool again = takeAgain();
```

```
          if (again == true) {
            sh--;
            y--;
          }
        } else {
          //Paikalla 6 ei ammuta
        }
        y++;
      }
      y = y - shooters;
    }
    placeChance++;
    y = y + shooters;
  }
}

int takeAgain() {
  int chosen;
  while (1) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Kiekko/t OK: V");
    lcd.setCursor(0, 1);
    lcd.print("Uudestaan K");
    chosen = choose();
    if (chosen == 3) {
      return true;
    }
    if (chosen == 1) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Valitsit M");
      lcd.setCursor(0, 1);
      lcd.print("Paina K tai V");
      delay(2000);
    }
    if (chosen == 2) {
      return false;
```

```
    }
  }
}

void shootOrder() {
  // Ampumajärjestys
  switch (shooters) {
    case 1: {
      int tempor[6] = {1, 1, 1, 1, 1, 1};
      int temppl[6] = {1, 2, 3, 4, 5, 6};
      for (int i = 0; i < (shooters * totalPlaces); i++) {
        shooterTurns[i] = tempor[i];
        shooterPlaces[i] = temppl[i];
      }
    }
    break;
    case 2: {
      int tempor[12] = {1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 2, 1};
      int temppl[12] = {1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 1, 6};
      for (int i = 0; i < (shooters * totalPlaces); i++) {
        shooterTurns[i] = tempor[i];
        shooterPlaces[i] = temppl[i];
      }
    }
    break;
    case 3: {
      int tempor[18] = {1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 3, 1, 2, 2, 3, 1};
      int temppl[18] = {1, 2, 3, 2, 3, 4, 3, 4, 5, 4, 5, 6, 1, 5, 6, 1, 2, 6};
      for (int i = 0; i < (shooters * totalPlaces); i++) {
        shooterTurns[i] = tempor[i];
        shooterPlaces[i] = temppl[i];
      }
    }
    break;
    case 4: {
      int tempor[24] = {1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 4, 1, 2, 3, 3, 4, 1, 2, 2, 3, 4, 1};
      int temppl[24] = {1, 2, 3, 4, 2, 3, 4, 5, 3, 4, 5, 6, 1, 4, 5, 6, 1, 2, 5, 6, 1, 2, 3, 6};
      for (int i = 0; i < (shooters * totalPlaces); i++) {
```

```
        shooterTurns[i] = tempor[i];
        shooterPlaces[i] = temppl[i];
       }
      }
      break;
    case 5: {
       int tempor[30] = {1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 5, 1, 2, 3, 4, 4, 5, 1, 2, 3, 3, 4, 5, 1, 2, 2, 3,
4, 5, 1};
       int temppl[30] = {1, 2, 3, 4, 5, 2, 3, 4, 5, 6, 1, 3, 4, 5, 6, 1, 2, 4, 5, 6, 1, 2, 3, 5, 6, 1, 2,
3, 4, 6};
       for (int i = 0; i < (shooters * totalPlaces); i++) {
        shooterTurns[i] = tempor[i];
        shooterPlaces[i] = temppl[i];
       }
      }
      break;
    case 6: {
       int tempor[36] = {1, 2, 3, 4, 5, 6, 6, 1, 2, 3, 4, 5, 5, 6, 1, 2, 3, 4, 4, 5, 6, 1, 2, 3, 3, 4, 5,
6, 1, 2, 2, 3, 4, 5, 6, 1};
       int temppl[36] = {1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 1, 2, 3,
4, 5, 6, 1, 2, 3, 4, 5, 6};
       for (int i = 0; i < (shooters * totalPlaces); i++) {
        shooterTurns[i] = tempor[i];
        shooterPlaces[i] = temppl[i];
       }
      }
      break;
  }
}

void message(char x) {
  bool ack = false;
  String message = "Heitin ";
  message = message + x;
  LoRa.beginPacket();
  LoRa.print(message);
  LoRa.endPacket();
  for (int n = 0; n < 3000; n++) {
```

```
    int packetSize = LoRa.parsePacket();
    String packet;
    String fullPacket;
    String throwerIn = "Viesti ok";
    if (packetSize) {
      while (LoRa.available()) {
        packet = ((char)LoRa.read());
        fullPacket = fullPacket + packet;
      }
      if (throwerIn == fullPacket) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Viestikuittaus");
        lcd.setCursor(0, 1);
        lcd.print("OK");
        delay(2000);
        ack = true;
        break;
      }
    }
  }
  if (ack == false) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Viestikuittaus");
    lcd.setCursor(0, 1);
    lcd.print("EI SAAPUNUT");
    delay(2000);
  }
}

void webPlanning() {
  String take11111 = "off";
  String take1112 = "off";
  String take122 = "off";
  int counter = 1;
  int clayCounter = 0;
  lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("Suunnittelu:");
lcd.setCursor(0, 1);
lcd.print(myIP);
bool complete = false;
while (complete == false) {
  String header;
  WiFiClient client = server.available();
  if (client) {
    String currentLine = "";
    String valueString = String(5);
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        header += c;
        if (c == '\n') {
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();
            if (header.indexOf("GET /get?input=") >= 0) {
              int pos1 = header.indexOf('=');
              int pos2 = header.indexOf(' ');
              pos2 = header.indexOf(' ', pos2 + 1);
              valueString = header.substring(pos1 + 1, pos2);
              for (int i = 0; i < 5; i++) {
                clayOrder[clayCounter] = toupper(valueString[i]);
                clayCounter++;
              }
            } else if (header.indexOf("GET /11111/on") >= 0) {
              take11111 = "on";
              defRounds[0] = 1;
              defRounds[1] = 1;
              defRounds[2] = 1;
              defRounds[3] = 1;
              defRounds[4] = 1;
```

```
            } else if (header.indexOf("GET /1112/on") >= 0) {
              take1112 = "on";
              defRounds[0] = 1;
              defRounds[1] = 1;
              defRounds[2] = 1;
              defRounds[3] = 2;
              defRounds[4] = 0;
            } else if (header.indexOf("GET /122/on") >= 0) {
              take122 = "on";
              defRounds[0] = 1;
              defRounds[1] = 2;
              defRounds[2] = 2;
              defRounds[3] = 0;
              defRounds[4] = 0;
            }
            // Web-sivu määritykset ja ulkoasu
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-width, ini-
tial-scale=1\">");
            client.println("<meta charset=\"UTF-8\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            client.println("<style>html { font-family: Arial; display: inline-block; margin: 0px
auto; text-align: center;}");
            client.println(".button { background-color: #458748; border: none; color: white;
padding: 16px 40px;");
            client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor:
pointer;}</style></head>");
            // Sivuston sisältö alkaa tästä.
            client.println("<body><h1>Kierroksen suunnittelu</h1>");
            if (counter == 6) {
              int placeCounter = 0;
              client.println("<p>Oletuskierros valittu ja tallennettu laitteeseen.</p>");
              for (byte i = 0; i < (sizeof(clayOrder) / sizeof(clayOrder[0])); i++) {
                if (clayOrder[i] == 0) {
                  break;
                }
                if (placeCounter >= 5) {
                  placeCounter = 0;
```

```
      }
      placeCounter++;
    }
    for (byte i = 0; i < (sizeof(defRounds) / sizeof(defRounds[0])); i++) {
      EEPROM.write(eeCount, defRounds[i]);
      eeCount++;
    }
    for (byte i = 0; i < (sizeof(clayOrder) / sizeof(clayOrder[0])); i++) {
      EEPROM.write(eeCount, clayOrder[i]);
      eeCount++;
    }
    EEPROM.commit();
    complete = true;

  } else if (take11111 == "on" || take1112 == "on" || take122 == "on") {
    client.println("<p>Kirjoita paikan kiekot yhteen putkeen. Esim: 1112 kierroksella kirjoitetaan ABCDE, jolloin ensimmäinen yksittäiskiekko on A, toinen yksittäiskiekko B, kolmas yksittäiskiekko on C ja viimeisenä kaksoiskiekko jonka kiekot ovat D ja E.</p>");
    client.print("<p>Kirjoita kenttään paikan ");
    client.print(counter);
    client.println(" kiekot.</p>");
    client.println("</form><br><form action=\"/get\"><input type=\"text\" name=\"input\"><input type=\"submit\" value=\"Submit\" + &\"></form><br>");
    counter++;
  } else {
    client.println("<p>Valitse alla olevista painikkeista millainen kierros suunnitellaan.</p>");
    client.println("<p>Painikkeella 11111 viiden yksittäiskiekon kierros.</p>");
    client.println("<p><a href=\"/11111/on\"><button class=\"button\">11111</button></a></p>");
    client.println("<p>Painikkeella 1112 kolmen yksittäiskiekon ja yhden kaksoiskiekon kierros.</p>");
    client.println("<p><a href=\"/1112/on\"><button class=\"button\">1112</button></a></p>");
    client.println("<p>Painikkeella 122 yhden yksittäiskiekon ja kahden kaksoiskiekon kierros.</p>");
    client.println("<p><a href=\"/122/on\"><button class=\"button\">122</button></a></p>");
```

```
            }
            client.println("</body></html>");
            client.println();
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
      }
    }
    header = "";
    client.stop();
  }
 }
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("LAITE SAMMUU,");
 lcd.setCursor(0, 1);
 lcd.print("JA KAYNNISTYY");
 delay(5000);
 ESP.restart();
}
```