TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Emil Marvin Mikk 193313IAIB

NUTIKELLA TARKVARA ARENDAMINE

Bakalaureusetöö

Juhendaja: Peeter Ellervee
PhD

Tallinn 2023

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Emil Marvin Mikk

22.05.2023

Annotatsioon

Viimastel aastatel on nutikellad järjest populaarsust kogunud. Suuresti on see tingitud mugavusest, mida nutikellad pakuvad. Mugavuse suurenemine toob tihtipeale endaga kaasa privaatsuse vähenemise ja seda saab öelda ka nutikellade puhul.

Käesoleva lõputöö eesmärgiks oli luua eelnevalt autori poolt tehtud nutikella riistvarale vabavaraline tarkvara, mille üle on kasutajal täielik kontroll ning mille andmeid ei edastata kolmandale osapoolele. Eesmärkides püstitati tarkvarale järgmised peamised nõuded: nutitelefoni meediapleieri juhtimine, teavituste kuvamine ning kellaaja sünkroniseerimine, sammude lugemine, juhtmevaba tarkvarauuendamine ja aku kestvus vähemalt kaks nädalat.

Töö teoreetilises osas analüüsiti olemasolevaid lahendusi ning valiti välja sobilik alglaadur, operatsioonisüsteem, graafikateek ja BLE teenused. Töö praktilises pooles vaadeldi arendatud lahendust, testimist ja tulemusi ning edasiseid arendusvõimalusi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 6 peatükki, 16 joonist, 3 tabelit.

Abstract

Development of Smartwatch Software

In recent years, smartwatches have become increasingly popular, which is largely due to the convenience they offer. However, increase in convenience often comes at the expense of privacy and this can also be said about smartwatches.

The goal of this thesis was to create an open-source software for a smartwatch hardware, previously developed by the author, over which the user would have full control and gathered data would not be sent to third parties. The software had several requirements related to smartphone integration, including displaying notifications, synchronizing time, over-the-air software updates, and media player control. In addition to smartphone integration requirements, it also had to count steps and have at least two weeks of battery life.

The theoretical part of this thesis analyzes already existing open-source solutions and selects suitable bootloader, operating system, graphics library, and BLE services. The practical part goes into the implementation details, testing and results, as well as further development possibilities.

All the specified requirements were successfully met during the development of the software and battery life of up to two months was achieved, surpassing the initial two weeks requirement. The complete implementation, including software and hardware, is openly available on GitHub.

The thesis is written in Estonian and is 27 pages long, including 6 chapters, 16 figures and 3 tables.

Lühendite ja mõistete sõnastik

AMS	*Apple Media Service*, Apple arendatud meediapleieri juhtimiseks mõeldud BLE teenus
ANCS	*Apple Notification Center Service*, Apple arendatud BLE teenus iOS seadmete teavituste lugemiseks
API	*Application Programming Interface*, rakendusliides
BLE	*Bluetooth Low Energy*
CTS	*Current Time Service*, BLE teenus jooksva aja küsimiseks
DFU	*Device Firmware Update*, seadme tarkvarauuendus
I2C	*Inter-Integrated Circuit*, sünkroonne järjestiksuhtluse kommunikatsiooniprotokoll
LCD	*Liquid Crystal Display*, vedelkristallkuvar
RAM	*Random Access Memory*, operatiivmälu
SDK	*Software development kit*, tarkvaraarenduskomplekt
SPI	*Serial Peripheral Interface*, sünkroonse järjestiksuhtluse liidese standard
SWD	*Serial Wire Debug*, mikrokontrolleri programmeerimiseks ja debugimiseks mõeldud protokoll
UI	*User Interface*, kasutajaliides
UX	*User Experience*, kasutajakogemus

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

Nutikell on käe peal kantav seade, mis lisaks tavapärastele kella funktsionaalsustele suudab ka näiteks lugeda samme, mõõta pulssi ja unekvaliteeti ning mõned ennustavad isegi terviseprobleeme [1]. Tihtipeale on nutikelladel ka olemas Bluetoothi tugi, mis võimaldab neil ühenduda nutitelefoniga ja käituda selle laiendina. See annab muuhulgas nutikelladel võimaluse kuvada nutitelefoni teavitusi, võtta vastu kõnesid, juhtida meediapleierit ja jagada enda kogutud informatsiooni. Viimastel aastatel on nutikellad järjest populaarsust kogunud ja suuresti on see tingitud mugavusest, mida nutikellad pakuvad.

1.1 Probleem

Mugavuse suurenemine toob tihtipeale endaga kaasa privaatsuse vähenemise ja seda saab öelda ka nutikellade puhul. Nutikellad annavad firmadele ligipääsu veelgi enamatele personaalsetele andmetele, nagu näiteks tervisenäitajad, harjumused, asukoht ja tehtud ostud. Firmad on vägagi huvitatud nendest andmetest, sest neid saab ära kasutada enda toodete parendamiseks, personaliseeritud reklaamide näitamiseks või müüa neid andmeid edasi kolmandale osapoolele. Isegi kui andmeid ei kasutata/jagata isikustatult, vaid anonümiseeritud või koondatud kujul, siis on ikkagi üsnagi täpselt neid võimalik deanonümiseerida [2].

Üks lahendus sellele probleemile on kasutada vabavaralisi lahendusi, kuna see pakub võimalust näha ja muuta selle lähtekoodi, andes kasutajale täieliku kontrolli selle üle. Vabavaraline tarkvara võimaldab samuti näiteks vältida selliseid olukordi, kus tootja on süsteemi jõudlust kasutaja teadmata vähendanud [3] või pannud teenuse, mille riistvara on kasutaja juba ostnud, igakuise lisatasu taha [4]. Olemasolevatel vabavaraliste nutikellade riistvaradel oli samuti puudusi. Näiteks kasutati mitteoptimaalseid riistvarakomponente (peamiselt mikrokontrollereid), mis tarbivad liiga palju voolu ning ei ole autori arvates sobilikud nutikellade jaoks, mille aku mahtuvus on niigi väike.

1.2 Eesmärk

Käesoleva lõputöö eesmärgiks on luua nutikella tarkvara eelnevalt autori poolt tehtud riistvarale. Riistvaraga suhtlevad tarkvara kihid võiksid olla võimalikult ära abstraheeritud, et oleks vähese vaevaga võimalik tulevikus riistvara muudatusi teha või teistele platvormidele portida. Et hoida lõputöö skoop hallatavana, siis autor valis välja tema arvates olulisemad

nõuded:

- BLE ühendus nutitelefoniga
- aja sünkroniseerimine nutitelefoniga
- tarkvara uuendamine BLE kaudu
- nutitelefoni meediapleieri juhtimine
- nutitelefoni teavituste kuvamine
- nutitelefoni kõnede kuvamine ja vastu võtmine/lõpetamine
- sammude lugemine
- aku kestvus vähemalt paar nädalat
- vabavaraline

1.3 Metoodika

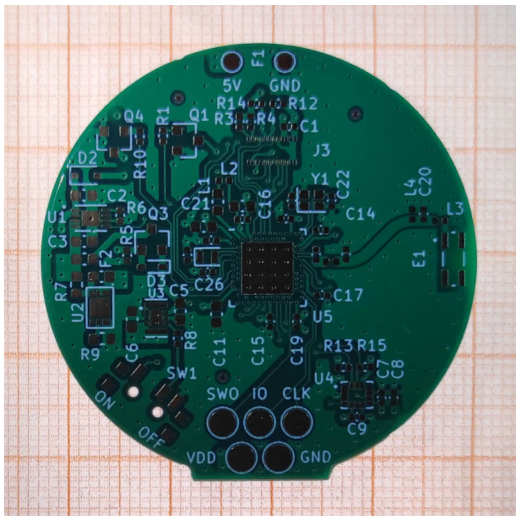
Eesmärkide saavutamiseks tuleb kõigepealt analüüsida olemasolevate vabavaraliste nutikellade tarkvaralisi lahendusi ning valida välja sobilik alglaadur, operatsioonisüsteem, graafikateek ja BLE teenused. Riistvaraga suhtlemiseks on vaja kirjutada draiverid puuteekraanile ja inertsiaalsensorile, mille kaudu saab lugeda tehtud sammude arvu. Aja sünkroniseerimiseks, nutitelefoni teavituste kuvamiseks ja meediapleieri juhtimiseks on vaja välja uurida, millised olemasolevad lahendused selle jaoks eksisteerivad ja valida välja kõige sobivam. Aku kestvuse maksimeerimise jaoks tuleb lugeda mikrokontrolleri ja teiste elektroonikakomponentide dokumentatsiooni, et välja selgitada, kuidas voolu tarbimist võimalikult madalal hoida. Aku kestvuse kontrollimiseks mõõdetakse voolu tarbimist erinevates olekutes: telefoniga ühendatult normaal- ja unerežiimil ja telefoniga mitte ühendatult normaal- ja unerežiimil. Nutikella kasutajasõbralikkuse ja funktsionaalsuste hindamiseks korraldatakse küsitlus.

Töö käigus valitakse välja sobilik alglaadur, operatsioonisüsteem, graafikateek ja BLE teenused. Valiku tegemiseks kasutatakse järgnevaid kriteeriume:

- mälu kasutus
- ajakulu õppimisele ja lahenduse implementeerimisele
- funktsionaalsused
- kasutamise keerukus
- toetatud platvormid
- dokumentatsioon ja kasutajaskonna tugi
- projekti küpsus

2. Taust

Nagu peatükis 1.2 oli mainitud, siis tarkvara hakatakse arendama autori poolt tehtud riistvarale. Riistvara on disanitud täielikult autori poolt, kuid komponentide valiku tegemisel, nagu näiteks mikrokontroller ja ekraan, uuris autor olemasolevaid lahendusi. Skeemide (lisad 2 ja 3) joonistamiseks kasutati vabavaralist programmi KiCad [5]. Kuna mikrokontrolleri tootja pakub tasuta skeemide ülevaatuse teenust ja autor ei ole väga kogenud elektroonika valdkonnas, siis kasutas autor seda teenust oma disainide kontrollimiseks. Pärast skeemide joonistamist ja kontrollimist telliti trükkplaadid (joonis 1) ja jootepasta stensiil JLCPCB'ist [6]. Trükkplaadi kokkupanemiseks kanti stensiiliga jootepasta plaadile, mille järel laoti kõik komponentid käsitsi oma kohtadele. Seejärel asetati trükkplaat rösterahju ja kuumutati seal kuni jootepasta oli ära sulanud (joonis 2). Pärast kokkupanemist kontrolliti, et plaadil ei oleks lühiseid.

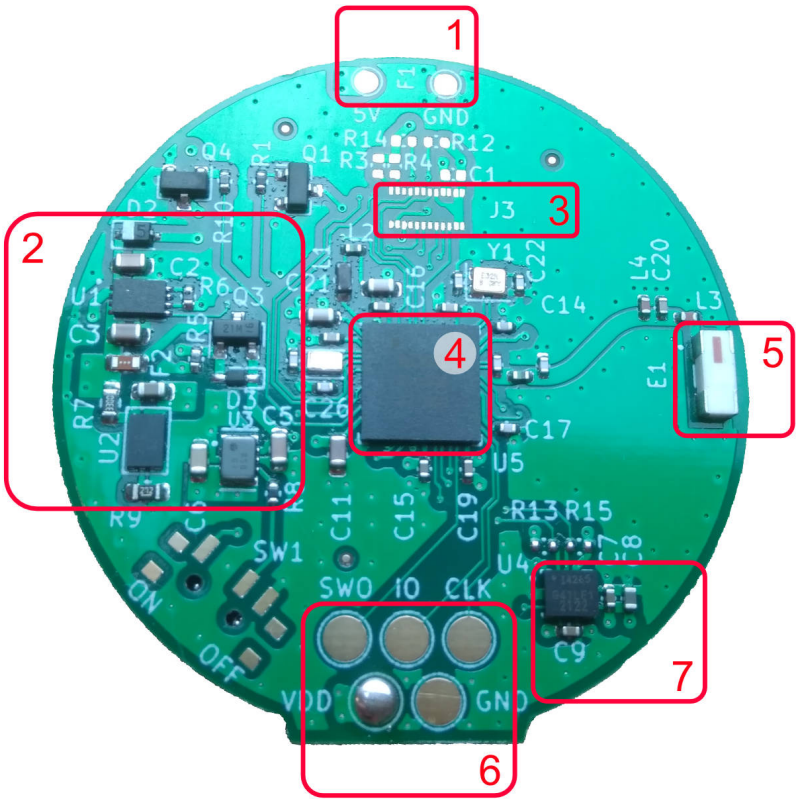


Joonis 1. Trükkplaat millimeeterpaberil.

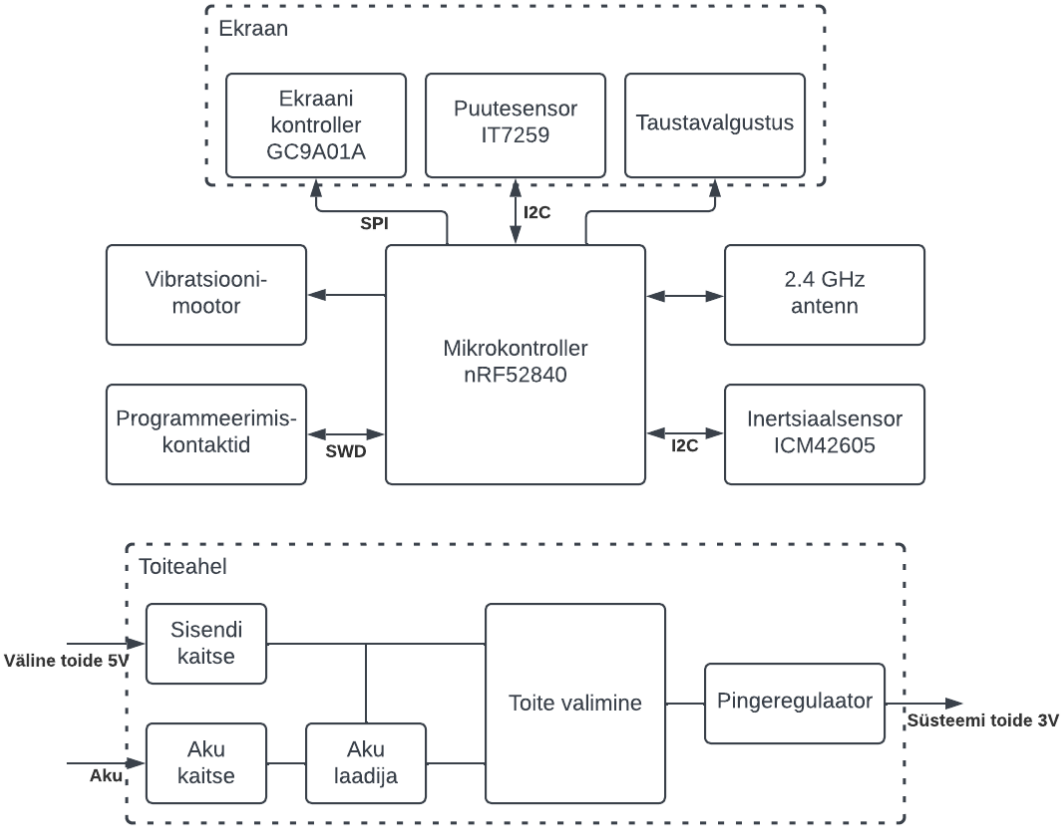
Joonis 2. Komponentidega trükkplaat rösterahjus.

Nutikella (joonis 3 ja 4) juhib Nordic Semiconductor'i madala voolutarbega nRF52840 mikrokontroller, millel on 256 kB operatiivmälu, 1024 kB säilmälu ja 64 MHz 32-bitine ARM Cortex-M4 protsessor. Samuti on nRF52840 mikrokontrolleril sisseehitatud BLE toega 2.4 GHz transiiver, mida kasutatakse nutitelefoniga suhtlemiseks. Seadmel ei ole nuppe ja juhtimine käib 1.28 tollise LCD puuteekraaniga, mille resolutsioon on 240x240 pikslit. Kasutajale teavitustest informeerimiseks ja puuteekraani vajutuste tagasiside andmiseks on seadmel vibratsioonimootor. Sammude lugemiseks on olemas inertsiaalsensor, mis suudab samuti tuvastada, mis asendis seade on. Seda saab kasutada süsteemi üles äratamiseks, kui kasutaja on seadme lugemiseks üles tõstnud või magama panna, kui seade on ära pandud.

Süsteem saab toite 200 mAh liitiumpolümeerakult. Kuna liitiumpolümeerakud on väga ohtlikud ja nende valesti käitlemine võib halvimal juhul lõppeda aku plahvatamisega, siis on seadmel olemas aku laadimis- ja kaitsekiibid, mis kaitsevad akut ülepinge, alapinge ja lühiste eest ning laevad seda ettenähtud viisil.



Joonis 3. Kokkupandud trükkplaat.
1 - laadimiskontaktid, 2 - toiteahel, 3 - ekraani pistik, 4 - mikrokontroller, 5 - 2.4 GHz antenn, 6 - programmeerimiskontaktid, 7 - inertsiaalsensor.



Joonis 4. Riistvara plokkskeem.

3. Analüüs

Järgnevates peatükkides analüüsitakse olemasolevaid lahendusi ja valitakse välja sobivad tehnoloogiad.

3.1 Olemasolevad lahendused

Kuigi vabavaralisi nutikellasid on mitmeid, siis järgnevalt vaadatakse lähemalt kahe nutikella (PineTime [7] ja Watchy [8]) tarkvaralisi lahendusi.

PineTime'l on olemas nii kiirendusandur kui ka pulssoksümeeter, mistõttu on võimalik seda kella kasutada pulsinäidu ning sammude lugemiseks. Kui kell on telefoniga ühendatud, saab see näidata telefoni teavitusi ja kõnesid, navigeerimise informatsiooni, juhtida meediapleierit, sünkroniseerida kellaaega. Samuti on võimalik seadme tarkvara uuendada. Lisaks on PineTime'l palju teisi olemasolevaid rakendusi nagu näiteks mängud, stopper, alarm ja muudetavad seadme sätted. Võrreldes PineTime'ga on Watchy üsna piiratud funktsionaalsusega. Lisaks kellajale ja sammude lugemisele saab Watchy tarkvara uuendada BLE kaudu. Kui kell on Wi-Fi'ga ühendatud siis on võimalik ka kellaaega sünkroniseerida ning hetke ilma kuvada.

Watchy tarkvara vaadates on näha, et kogu äriloogika kiht on kõik ühes failis, sisaldab pikki funktsioone, palju maagilisi numbreid ning korduvaid koodiridu, mis teeb projekti lugemise ja haldamise keeruliseks. Samuti ei ole riistvaraga tegelevad kihid hästi ära abstraheeritud ja on näha, et äriloogika fail tegeleb ka riistvara seadmete konfigureerimise ning juhtimisega. Kuna kogu äriloogika on ühes failis, siis tundub, et uute rakenduste lisamine on ka üsna keeruline. Kellal saab korraga olla peale laaditud ainult üks kella sihverplaat, kuid enda sihverplaadi tegemine tundub lihtne. PineTime on ahritektuurilisest poolest palju parem ja loetavam. Kellale on lihtne lisada oma rakendusi ja sihverplaate ning kogu selle jaoks vajaminev tegevus on hästi dokumenteeritud. Samuti on riistvaraga tegelevad kihid rohkem ära abstraheeritud kui Watchy'l, kuid tundub, et riistvara muudatused nõuavad siiski suuri koodi muudatusi ning ei ole võimalik mitut erinevat riistvaraversiooni korraga toetada.

PineTime'l on palju parem funktsionaalsus kui Watchy'l, kuid arvestades käesoleva töö mahtu, valis autor nendest välja enda arvates olulisemad. Üks asi mida autor arvab, et suudab parendada, on tarkvara arhitektuur ja riistvara abstraheerimine.

3.2 Tehnoloogiate valik

Selles peatükis analüüsitakse ja valitakse välja sobiv operatsioonisüsteem, alglaadur, graafikateek ja BLE teenused. Valiku tegemisel vaadeldakse järgnevaid kriteeriume:

- mälu kasutus
- ajakulu õppimisele ja lahenduse implementeerimisele
- funktsionaalsused
- kasutamise keerukus
- toetatud platvormid
- dokumentatsioon ja kasutajaskonna tugi
- projekti küpsus

3.2.1 Operatsioonisüsteem

Operatsioonisüsteemi valimine on oluline osa sardsüsteemide projekteerimisest. Mikrokontrollerites kasutavaid operatsioonisüsteeme nimetatakse ka RTOS'ideks (*Real-time operating system*) ehk reaalaja operatsioonisüsteemideks. Peamine erinevus reaalaja operatsioonisüsteemide ja tavapäraste personaalarvutite operatsioonisüsteemide vahel on see, et reaalaja operatsioonisüsteemid peavad tagama kiire reageerimise reaalaja sündmustele. Lihtsamatel sardsüsteemidel tihtipeale ei ole vaja operatsioonisüsteemi ja kogu tegevus toimub ühes protsessis, mis kutsub ise välja vastavaid süsteemi osi. See lähenemine aga ei ole mõistlik keerukamate süsteemide nagu nutikellade puhul, kuna see teeb projekti haldamise oluliselt keerulisemaks. Operatsioonisüsteem laseb jagada süsteemi osad erinevatesse protsessidesse, mis teeb süsteemi modulaarsemaks ja lihtsamini testitavamaks. RTOS aitab ka lihtsamini optimeerida seadme voolutarbimist, kasutades kõige madalama prioriteediga protsessi süsteemi magama panemiseks.

FreeRTOS [9] on 2003. aastal alguse saanud reaalaja operatsioonisüsteemi tuum, mille arenduse võttis 2017. aastal üle Amazon. Kuna FreeRTOS on ainult operatsioonisüsteemi tuum, siis on see ka üsnagi piiratud funksionaalsusega ja sisaldab peamiselt protsesside haldamise, sünkroniseerimise ning protsesside vahelise suhtluse jaoks mõeldud funktsionaalsusi. FreeRTOS'il on aga väga hea dokumentatsioon, väike ja lihtsasti mõistetav API ning autor on eelnevalt sellega kokku puutunud, mistõttu selle õppimine ei võta väga kaua aega. Arendamiseks tuleb kasutada Nordic Semiconductor'i nRF5 SDK'd [10], mis sisaldab erinevaid draivereid, teeke ja BLE kihti. Seda aga ei arendata enam aktiivselt ja uusi funktsionaalsusi sinna enam ei lisata [11]. Ehk siis, kui tulevikus on plaanis võimsamat Nordic'u mikrokontrollerit kasutada, nagu näiteks nRF53 või nRF91, siis need ei ole enam

toetatud. On küll võimalik arendada ilma SDK'ta, kuid ainuüksi BLE kihi tegemine on kordades suurema mahuga kui lõputöö ette näeb.

Zephyr [12] on reaalaja operatsioonisüsteem, mis sai alguse 2016. aastal. Kuna tegemist on suhteliselt noore projektiga, on palju API'sid, mis on veel ebastabiilsed ja aktiivses arengufaasis [13]. Hoolimata sellest on Zephyr märkimisväärselt populaarsust kogunud ning on toetatud suurte fimade poolt nagu näiteks Google, Intel ja Meta. Lisaks reaalaja tuumale on Zephyr'is ka palju erinevaid sisseehitatud teeke, draivereid, BLE tugi, failisüsteem jne. Kuna Zephyr kuulub Linux Foundation'isse, siis on nii mõnigi asi Linux'ist üle tulnud Zephyr'isse, nagu näiteks Kconfig [14] ja Devicetree [15]. Kconfig on mõeldud tarkvara konfigureerimiseks, millega on võimalik näiteks lülitada tarkvara mooduleid sisse või välja ja muuta nende parameetreid. Devicetree on aga riistvara kirjeldamiseks mõeldud andmestruktuur, mida operatsioonisüsteem kasutab draiverite initsialiseerimiseks. Kuna Zephyr'i tarkvara kihid on väga hästi abstraheeritud, siis riistvara muudatuse tegemine ja erinevate platvormide toetamine on üsna lihtne. Kui draiverid on olemas, siis muudatuste tegemine nõuab tihtipeale ainult Devicetree muutmist. Lisaks on Nordic Semiconductor viinud oma SDK Zephyr'i alla ja nad soovitavad uute toodete puhul kasutada nRF5 SDK asemel Zephyr'it [11]. Miinusteks Zephyr'i puhul on: projekti noorus, mille tõttu ei ole see veel kõige stabiilsem; dokumentatsiooni on palju, kuid see on osati puudulik või aegunud ning parema ülevaate saamiseks peab ise koodi lugema; kasutatakse palju C keele makrosid, mis teevad vigade otsimise keerulisemaks.

Tulevikukindlust ja arendamise aega silmas pidades, otsustas autor Zephyr'i kasuks. Zephyr'i Devicetree ja abstraktsioonikihid võimaldavad kergema vaevaga tulevikus riistvara muudatusi teha. Zephyr'i õppimine võtab FreeRTOS'i õppimisest küll kauem aega, kuid see-eest on võimalik kasutada sisseehitatud teeke, mis teevad arendamise palju kiiremaks.

3.2.2 Alglaadur

Alglaadur on esimene programm, mida jooksutatakse seadme käivitamisel ja mille peamiseks ülesandeks on põhiprogrammi või operatsioonisüsteemi käivitamine. Lisaks põhiprogrammi käivitamisele kuulub alglaaduri ülesannete hulka tihtipeale ka riistvara initsialiseerimine, põhiprogrammi autentimine, dekrüpteerimine ja selle tõstmine mälusse, kust mikrokontroller seda käivitada saab. Alglaadur on vägagi oluline ka tarkvarauuenduste jaoks, kuna see aitab kontrollida põhiprogrammi korrasolekut ja autentsust.

Antud projekti jaoks on alglaadur vajalik just tarkvarauuenduste jaoks. Töökindla tarkvarauuenduse jaoks peab seadmes olema kaks põhiprogrammi pesa. Tarkvarauuenduse ajal kirjutatakse uus tarkvara mitteaktiivsesse pesasse ja taaskäivitatakse seade. Seadme

käivitamisel kontrollib alglaadur uue tarkvara korrektsust. Kui uus tarkvara on korrektne, muutub see aktiivseks ja järgnevatel käivitumistel kasutatakse seda tarkvara. Kui uus tarkvara ei ole korrektne, kasutatakse vana tarkvara edasi. Zephyr'isse on sisse ehitatud kaks erinevat alglaadurit: nRF Secure Immutable Bootloader [16] ja MCUboot [17].

nRF Secure Immutable Bootloader, edaspidi NSIB, on Nordic Semiconductori poolt arendatud alglaadur. Nagu nimigi ütleb, on tegemist alglaaduriga, mida ei ole võimalik pärast pealelaadimist enam uuendada. NSIB toetab nii avalike võtmete tühistamist kui ka keelab vanemate tarkvarade pealelaadimist, mis on mõlemad vajalikud süsteemi turvalisuse tagamiseks. Esimene kaitseb olukordade eest, kui tarkvara autentimiseks kasutatavad privaatvõtmed on lekkinud. Teine aga vanade tarkvarade eest, millel võisid olla turvaaugud. Suureks miinuseks NSIB puhul on see, et ta ei toeta veel tarkvarauuendusi BLE kaudu, mis tuleb ise implementeerida. See aga on väga ajamahukas nii õppimise kui ka lahenduse väljaarendamise koha pealt. Dokumentatsioon on samuti puudulik ja selle kohta on väga vähe informatsiooni.

MCUboot on riistvara ja operatsioonisüsteemi agnostiline alglaadur. Vastupidiselt NSIB'ile on MCUboot alglaadurit võimalik uuendada. See alglaadur toetab BLE tarkvarauuendusi ja kaitseb samuti vanemate tarkvaraversioonide eest. Teoorias peaks MCUboot'i Zephyr'isse lisamine koos tarkvarauuendustega olema üsnagi lihtne, nõudes minimaalseid koodimuudatusi. Samuti on MCUboot'i dokumentatsioon heal tasemel. Puuduseks on aga see, et ei ole võimalik võtmeid tühistada.

Kuna NSIB ei ole BLE tuge, mille arendamine võtaks liiga kaua aega, otsustas autor MCUboot'i kasuks. MCUboot'i ainsaks miinuseks NSIB ees on see, et sellel ei ole võtmete tühistamise funktsionaalsust. Kuna selle implemeteerimine on aga väiksema mahuga kui, kui BLE toe NSIB lisamine, sai see ka otsustavaks faktoriks.

3.2.3 Graafikateek

Kasutajasõbraliku kasutajaliidese tegemiseks on vaja välja valida sobilik graafikateek. Kuna seadmel on puutetundlik LCD ekraan, siis graafikateek peab toetama värvilisi puutetundlikke ekraane. Selles peatükis võrreldakse kolme sardsüsteemidele mõeldud graafikateeki: uGFX [18], LVGL [19] ja Qt [20].

uGFX on piiratud ressurssidega sardsüsteemidele mõeldud platvormi agnostiline graafikateek. uGFX on modulaarse disainiga, millel on võimalik vastavalt projekti nõuetele mooduleid sisse ja välja lülitada, saavutades seeläbi väga madala mälu kasutuse. Graafikateek toetab monokroomseid, värvilisi, puutetundlikke ja mitte puutetundlikke ekraane.

Samuti on olemas audio tugi ja palju erinevaid sisseehitatud graafika elemente. uGFX on tasuta hobi- ja õppe-eesmärkide jaoks, kuid äriliste eesmärkide jaoks peab ostma kommertslitsentsi [21]. Lisaks on uGFX olemas töölauarakendus, kus on võimalik visuaalselt koodi kirjutamata disainida kasutajaliideseid. Miinuseks on see, et projekti arendus ja foorum ei tundu olevat väga aktiivsed. Viimane uGFX koodimuudatus oli 2022. aasta juunis.

LVGL on väikese mälukasutusega ja platvormi agnostiline sardsüsteemidele mõeldud graafikateek. LVGL on MIT litsenstiga, mis on üks kõige väiksemate piirangutega litsents. Minimaalsed mälunõuded LVGL jooksutamiseks on 64 kB säilmälu ja 16 kB operatiivmälu. Toetab kaht ekraanipuhvrit, mis teeb ekraanipildi uuendamise kiiremaks, kuna on võimalik samaaegselt kirjutada ühte ekraanipuhvrisse ja saata teisest ekraanipuhvrist infot ekraanile. LVGL'l on üle 30 sisseehitatud graafika elemendi, alustades nuppudest, piltidest ja lõpetades kalendri, graafikute ja klaviatuuriga. Elementidel on võimalik muuta kõiki nende stiiliomadusi, nagu näiteks taustavärv, serva raadius ja varjud. Kuna LVGL on juba Zephyr'isse sisse ehitatud, siis selle projekti lisamine peaks üsna lihtne olema, nõudes ainult Devicetree ja Kconfigi muudatusi. LVGL'l on minimaalne lihtsasti kasutatav riistvara abstraktsioonikiht, mis vajab ainult kahe funkstiooni implementeerimist: üks, mis saadab LVGL puhvri ekraanile ja teine, mis tagastab viimati puudutatud ekraanikoordinaadid. Õppimiseks on LVGL'l heal tasemel dokumentatsioon, palju koodinäiteid, aktiivne foorum ja tasuline kuuetunnine kursus. Autor on ka eelnevalt selle teegiga kokku puutunud, mistõttu õppimine ei võta väga kaua aega. Samuti on LVGL kood lihtsasti loetav ja sellel on intuitiivne API. 2022. aastal tuli sellele samuti välja töölauarakendus SquareLine Studio [22] visuaalseks kasutajaliidese disainimiseks. SquareLine Studio kasutamine mitteärilistel eesmärkidel on tasuta, kuid teatud piirangutega.

Qt on kõige populaarsem ja võimekam mainitud graafikateekidest. Sarnaselt eelnimetatud graafikateekidega, on Qt'l samuti olemas töölauarakendus, millega on võimalik lihtsa vaevaga kasutajaliideseid disainida. Suurteks miinusteks Qt puhul on see, et sellel on ainult kommertsiaallitsents [23] ja see kasutab palju rohkem mälu kui teised graafikateegid. Miinimum Qt mälunõuded on 42 kB operatiivmälu ja 500 kB säilmälu [24]. Kuna mikrokontrolleril on 1024 kB säilmälu ja tarkvarauuenduste jaoks on vaja hoida mälus kahe tarkvara jagu ruumi, siis võtaks Qt teek ära kogu võimaliku säilmälu. Nende miinuste tõttu ei ole Qt'd kahjuks võimalik kasutada.

Võrreldud graafikateekidest otsustas autor LVGL kasuks. Qt graafikateek osutus mittevalituks suure mälukasutuse tõttu, mis mikrontrollerisse lihtsalt ei mahu. LVGL ja uGFX tundusid mõlemad väga võimekad graafikateegid ja neil mõlemal on olemas kasutajaliidese disainimiseks mõeldud töölauarakendus. Peamisteks otustavateks faktoriteks oli aga autori

eelnev kogemus, dokumentatsioon ja projekti aktiivsus, mis olid LVGL projekti puhul palju paremad. Valikule aitas kaasa ka LVGL palju leebem litsents.

3.2.4 BLE teenused

BLE teenus on kogum ühe funksionaalsusega seotud karakteristikuid, mis omakorda sisaldavad väärtusi, mida on kliendil võimalik lugeda või üle kirjutada. Näiteks üks standardis defineeritud teenus on *Heart Rate Service* [25], millel on kaks karakteristikut. Üks sisaldab südamelöökide sagedust ja teine sensori asukohta.

Projekti eesmärkides oli viis nutitelefoniga suhtlemisega seotud nõuet: aja sünkroniseerimine, tarkvara uuendamine, meediapleieri juhtimine, teavituste kuvamine ja kõnede kuvamine ning kontrollimine. Nende nõuete täitmiseks on vaja välja valida sobilikud teenused või nende puudumisel need ise välja mõelda. Tarkvarauuenduste tegemiseks on MCUboot'il olemas enda BLE teenus. Õnneks on teiste nõuete jaoks standardis defineeritud sobilikud teenused ning neid ei pea ise välja mõtlema. Aja sünkroniseerimiseks on standardis defineeritud CTS (*Current Time Service*) [26], meediapleieri juhtimiseks MCS (*Media Control Service*) [27], teavituste kuvamiseks ANS (*Alert Notification Service*) [28] ja kõnede jaoks TBS (*Telephone Bearer Service*) [29]. Lisaks BLE standardi teenustele on Apple välja töötanud oma BLE teenused: ANCS (*Apple Notificaion Center Service*) [30], mida saab kasutada teavituste ja kõnede kuvamiseks ning AMS (*Apple Media Service*) [31], mille kaudu saab meediapleierit juhtida. Eelnimetatud Apple BLE teenused ja CTS jooksevad iga iOS seadme taustal ning ühildunud seadmel on neid võimalik kasutada. Erinevalt iOS seadmetest ei jookse Androidi seadmete taustal BLE teenused ja seadmega ühildumiseks ning suhtlemiseks on vaja arendada eraldi rakendus.

Autor otsustas CTS, ANCS ja AMS teenuste kasuks. Peamiseks faktoriks oli see, et need teenused jooksevad iOS seadmetel taustal. See võimaldab nutikellal ilma rakenduseta suhelda iOS seadmetega, hõlbustades seeläbi ka nutikella arendamist, kuna ei pea samaaegselt nutitelefonile rakendust arendama. Samuti on Zephyr'is olemas kõikide väljavalitud teenuste tugi.

4. Lahenduse väljatöötamine

Järgnevates peatükkides räägitakse arenduskeskkonnast, nutikella arhitektuurist, draiverite arendamisest, nutitelefoniga liidestamisest, kasutajaliidesest ning sellest, kuidas optimeerida aku kestvust.

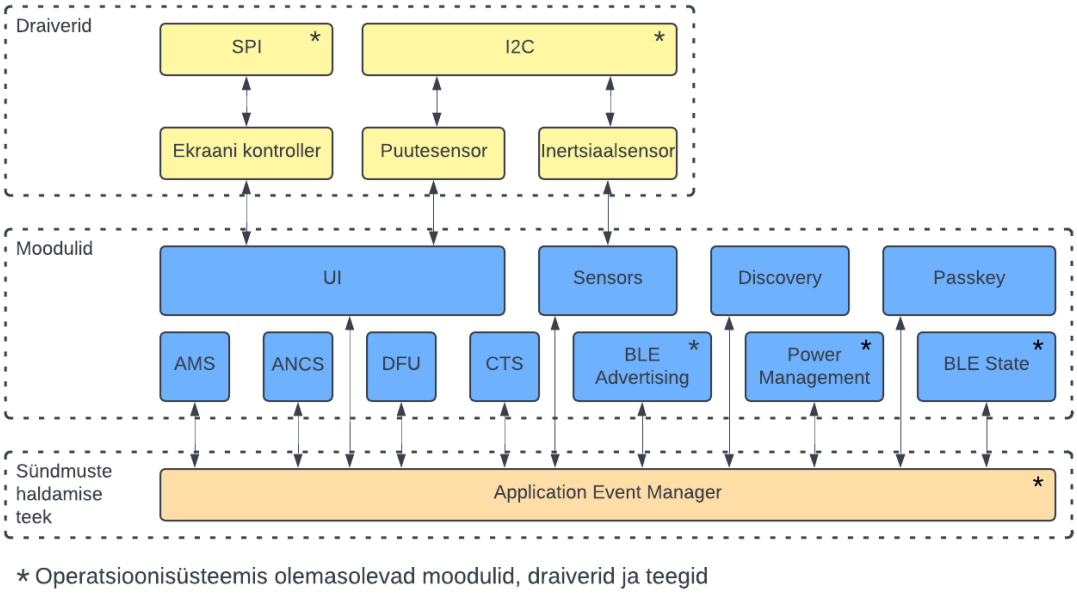
4.1 Arenduskeskkond

Nutikella tarkvara arendamiseks kasutati Visual Studio Code tekstiredaktorit peamiselt seetõttu, et autor on kõige tuttavam selle tekstiredaktoriga. Samuti on Nordic Semiconductor kirjutanud sellele laiendid Devicetree ja Kconfigi lihtsamaks konfigureerimiseks ning seadme debugimiseks. Riistvara programmeerimine ja debugimine käib SWD kontaktide kaudu. Seadet on võimalik debugida nii teksti välja printimisega kui ka interaktiivse debuggeriga, kuid arendamise käigus tuli välja, et interaktiivset debuggerit ei saa väga hästi kasutada, kui BLE kiht nõuab kindlat ajastamist.

4.2 Arhitektuur

Nutikella äriloogika kiht kasutab sündmustepõhist arhitektuuri (joonis 5), kus süsteemi funktsionaalsused on jaotatud erinevate moodulite vahel ning suhtlemine käib sündmuste kaudu. Iga moodul tegeleb ühe kindla funktsionaalsusega ning on isoleeritud teistest moodulitest, teadmata teiste moodulite olemasolust. Sündmuste haldamiseks ja edastamiseks kasutati *Application Event Manager* [32] teeki.

Võrreldes arhitektuuriga, kus moodulid suhtlevad otse omavahel, on sündmustepõhisel arhitektuuril paar eelist. Esiteks võimaldab see kõikidel moodulitel lugeda saadetud sündmusi, mis teeb triviaalseks sündmuse edastamise mitmele süsteemi osale korraga. Näiteks võib süsteem saata toiterežiimi muutuse sündmuse ning kõik moodulid saavad seda pealt kuulata, ilma et süsteem peaks teadma kõiki süsteemi osi, mis on toiterežiimi muutustest huvitatud. Teiseks, kuna moodulite vaheline suhtlus käib ainult sündmuste kaudu, siis moodulid ei sõltu üksteisest ega pea teadma üksteise funktsionaalsusi, mis muudab süsteemi testimise ja moodulite samaaegse arendamise lihtsamaks.



Joonis 5. Tarkvara arhitektuuri plokkseem.

- UI: kasutajaliidese moodul
- Sensors: edastab sensoritelt tulevaid andmeid
- Power Management: süsteemi toiterežiimi eest vastutav moodul
- AMS: juhib nutitelefoni meediapleierit
- ANCS: edastab nutitelefoni teavitusi sündmustena
- DFU: tarkvarauuendustega tegelev moodul
- Passkey: edastab nutitelefoniga ühendamiseks vajaliku parooli sündmusena
- Discovery: vastutab BLE teenuste avastamise eest
- BLE Advertising: algatab BLE reklaamipakettide saatmise
- BLE State: initsialiseerib BLE ja edastab BLE staatuse sündmusi

Neid mooduleid vaadeldakse lähemalt järgnevates peatükkides.

4.3 Draiverid

Selleks, et mikrokontroller oskaks ülejäänud seadmetega suhelda, oli vaja kirjutada draiverid järgnevatele seadmetele: ekraani kontroller, puutesensor ja intertsiaalsensor.

LCD ekraani juhtimise eest vastutab ekraani kontroller. Ekraani kontrolleriga suhtlemine käib peamiselt kolme funktsiooni kaudu, millest kaks on taustavalgustuse juhtimiseks ja üks, mis saadab ekraanipuhvri ekraani kontrollerile. Ekraanipuudutuste teadasaamiseks on ekraanil olemas puutesensor. Et puutesensori draiver ei peaks seadet konstantselt pollima,

saab puutesensorit konfigureerida nii, et see muudaks üht oma väljundi nivood, kui ekraani on puudutatud. Nivoo muutudes pärib draiver puutesensorilt ekraanipuudutuse koordinaadid ja saadab selle informatsiooni edasi draiveri kasutaja tagasikutse funktsioonile. Ekraani kontrolleri ja puutesensori draiveritega suhtlemiseks on *ui* moodul.

Nutikellal on intertsiaalsensor, millel on 3-teljeline kiirendusandur ja 3-teljeline güroskoop, mille väärtusi on kasutajal võimalik lugeda. Lisaks toorandmete lugemisele on seadmel ka sisseehitatud signaaliprotsessor, mis kasutab kiirendusanduri andmeid, et arvutada välja näiteks tehtud sammude arvu ja seadme asendit. Kuna seadmel on intertsiaalandur just sammude lugemiseks ja süsteemi üles äratamiseks, kui kasutaja on seadme üles tõstnud, ei pea toorandmeid ise töötlema ning saab kasutada ainult signaaliprotsessorist tulevaid andmeid. Inertsiaalsensori draiveriga suhtlemiseks on *sensors* moodul, mis küsib konfigureeritud aja tagant draiverilt tehtud sammude arvu ja edastab selle informatsiooni *sensors_event* sündmusena.

Draiverite testimiseks ja vigade leidmiseks kasutati Saleae loogikaanalüsaatorit.

4.4 BLE

Järgnevates peatükkides vaadeldakse kuidas nutitelefoniga ühendus luuakse, aega sünkroniseeritakse, meediapleierit juhitakse, teavitusi kuvatakse ning tarkvara uuendatakse.

4.4.1 Ühendamine

GAP (*Generic Access Profile*) on BLE protokolli kiht, mis defineerib, kuidas BLE seadmed üksteist üles leiavad, ühendus luuakse ja üksteise teenuseid tuvastavad. GAP defineerib neli rolli, kuid selle projekti raames on olulised ainult kaks nendest: *Central* (keskne seade) ja *Peripheral* (perifeerne seade). Perifeerne seade, selle projekti puhul nutikell, on seade millega ühendutakse. Et keskne seade leiaks perifeerse seadme üles, peab perifeerne seade saatma teatud intervalli tagant reklaamipakette, mis sisaldavad näiteks seadme nime, teenuseid ja mis tüüpi seadmega on tegu. Keskne seade kuulab perifeersete seadmete saadetud reklaamipakette ja õige seadme leidmisel üritab sellega ühenduda. Pärast ühendumist juhib keskne seade seadmetevahelist suhtlust.

Nutitelefoniga ühendamiseks oli võimalik kasutada *Application Event Manager*'i mooduleid *BLE state module* [33] ja *BLE advertising module* [34]. *BLE state module* initsialiseerib BLE kihi, määrab turvataseme ja edastab BLE sündmusi (näiteks seadmega ühenduse loomise, ühenduse katkemise või ühenduse parameetrite muutumise sündmus). *BLE ad-*

vertising module kuulab *BLE state module* sündmusi ja nende põhjal alustab või lõpetab reklaamipakettide saatmise. *BLE advertising module*'l on võimalik ka konfigureerida millise aja tagant reklaamipakette saadetakse, mida vaadeldakse lähemalt peatükis 4.6.

Lisaturvalisuse jaoks on võimalik ühendamisel näidata ühel seadmel parooli, mida peab kasutaja teise seadmesse sisestama. Selle funktsionaalsuse lisamiseks oli vaja teha *passkey* moodul, millel on BLE kihi tagasikutse funktsioon, mida kutsutakse, kui seade peab näitama parooli. See parool saadetakse edasi *passkey_event* sündmusena, mida kasutajaliidese kiht kasutab parooli kuvamiseks.

Pärast ühendumist peab nutikell tuvastama, mis teenused nutitelefonil on. Teenused, mida nutitelefon peab leidma, on järgmised: CTS, ANCS ja AMS. Kuna korraga on võimalik avastada ainult üht teenust, siis nende teenustega tegelevad moodulid ei saa oma teenuseid ise avastada. Muidu võib tekkida olukord, kus moodulid üritavad teenuseid samal ajal avastada. Selle probleemi lahendamiseks oli vaja luua eraldi *discovery* moodul, mis üritab ükshaaval neid teenuseid avastada ja õnnestumise korral edastab need *discovery_event* sündmustena.

4.4.2 Aja sünkroniseerimine

Aja sünkroniseerimiseks oli vaja teha *cts* moodul, mis suhtleb nutitelefoniga CTS teenuse kaudu. Seadme käivitumisel peab see moodul esiteks süsteemi kellaaja initsialiseerima, kuna enne selle tegemist ei ole võimalik kellaaega lugeda. Pärast kellaaja initsialiseerimist jääb *cts* moodul ootama *discovery_event* sündmust. Sündmuse ilmnedes loeb moodul nutitelefonilt kellaaja ja kasutab seda süsteemi kellaja uuendamiseks.

Lisaks on võimalik *cts* moodulil sisse lülitada aja varukoopiate tegemine, et pärast taaskäivitamist oleks võimalik kellaaeg taastada. Kui varukoopiate tegemine on sisse lülitatud, siis *cts* moodul salvestab seadistatud aja tagant kellaaja muutujasse ja taaskäivitamisel kasutatakse seda kellaaja initsialiseerimisel. Et seda muutujat seadme käivitumisel üle ei kirjutataks, on vaja see panna kindlasse operatiivmälu sektorisse. Tagamaks, et varukoopia muutuja sisaldab korrektset kellaaega, pannakse samasse sektorisse samuti ka selle kontrollimiseks mõeldud muutuja. Kui kontrollimiseks mõeldud muutujas on õige konstant, siis on teada, et varukoopia muutujas olev kellaaeg on samuti õige. Miinuseks sellise lahenduse puhul on aga see, et olenevalt varukoopiate tegemise sagedusest, jääb taastatud kellaaeg vähesemal või rohkemal määral maha reaalsest kellaajast. Kuna varukoopiat hoitakse operatiivmälus, siis ei suuda selline lahendus samuti hoida kellaega, kui mikrokontrollerilt võetakse toide ära.

4.4.3 Meediapleieri juhtimine

Nutitelefoni meediapleieriga suhtlemiseks kasutati AMS teenust, millel on kolm karakteristikut: *Remote Command*, *Entity Update* ja *Entity Attribute*.

Remote Command karakteristikusse kirjutades on võimalik juhtida meediapleierit. Samuti kasutab AMS server seda karakteristikut, et anda kliendile teada, mis käske meediapleier toetab.

AMS defineerib kolm erinevat objekti, mille atribuute on võimalik kliendil lugeda. Esimene objekt on *Player*, aktiivne meediapleieri rakendus, mille atribuutideks on rakenduse nimi, volüüm ja staatus. Järgmine on *Queue*, lugude järjekord, mille atribuutideks on järjekorras olevate lugude arv ja mitmes lugu hetkel mängib. Viimane objekt on *Track*, aktiivne lugu, mille atribuutideks on artisti, albumi ja loo nimi ning loo pikkus. Et server hakkaks kliendile informatsiooni nende objektide kohta saatma, peab klient kõigepealt kirjutama *Entity Update* karakteristikusse, mis objektidest ja atribuutidest ta huvitatud on. Pärast seda hakkab AMS server tulevastest atribuutide muutustest teada andma *Entity Update* karakteristiku kaudu.

On võimalus, et mingi objekti atribuudi väärtus ei mahu täielikult *Entity Update* karakteristikusse ära. Sellisel juhul on kliendil võimalik pärida seda atribuuti uuesti *Entity Attribute* karakteristiku kaudu.

AMS teenusega suhtlemiseks on *ams* moodul. Moodul ootab *discovery_event* sündmust, mille kätte saades kirjutab *Entity Update* karakteristikusse objektide atribuudid, millest süsteem on huvitatud. Atribuudi väärtuse kättesaades saadetakse see edasi vastavalt atribuudile kas *media_player_event* või *media_track_event* sündmusena. Toetatud käsud, ehk kui server kirjutab *Remote Command* karakteristikusse, edastatakse *media_capabilities_event* sündmusena. Meediapleieri juhtimiseks saavad moodulid saata *media_command_event* sündmuse.

4.4.4 Teavituste ja kõnede kuvamine

Nutitelefoni teavituste ja kõnede kättesaamiseks kasutati ANCS teenust, millel on kolm karakteristikut: *Notification Source*, *Control Point* ja *Data Source*.

Notification Source karakteristikut kasutatakse uutest teavitustest, teavituste muutustest või teavituste eemaldamisest teada andmiseks. Karakteristik sisaldab järnevaid välju:

- *EventID*: kas teavitus lisati, muudeti või eemaldati
- *EventFlags*: teavituse indikaatorbitid, näiteks mis tegevusi teavitus lubab ja kas tegemist on tähtsa teavitusega
- *CategoryID*: teavituse kategooria, näiteks e-kiri, sissetulev kõne või sotsiaalmeedia
- *CategoryCount*: mitu aktiivset *CategoryID* teavitust eksisteerib
- *NotificationUID*: teavituse identifikaator

Teavituste kontrollimiseks ja lisainformatsiooni saamiseks kasutatakse *Control Point* ja *Data Source* karakteristikuid. Teavituse kontrollimiseks peab nutikell kirjutama *Control Point* karakteristikusse teavituse identifikaatori ja kas soovitakse teha negatiivset või positiivset tegevust. Mida need tegevused täpsemalt teevad sõltub teavitusest, kuid ANCS server tagab, et tegevus ei üllata kasutajat. Näiteks sissetuleva kõne puhul võib positiivne tegevus tähendada kõne vastuvõtmist ja negatiivne kõnest keeldumine. Et teada saada, millised tegevusi teavitus lubab, tuleb seda lugeda *EventFlags* väljast. Teavituse lisainformatsiooni ehk atribuutide saamiseks tuleb sarnaselt teavituste kontrollimisele kirjutada *Control Point* karakteristikusse teavituse identifikaator ja loend atribuutidest, mida soovitakse lugeda. Vastus sellele päringule saadetakse *Data Source* karakteristikusse. Võimalikud teavituse atribuudid on näiteks teavituse pealkiri, sisu, kuupäev ja rakenduse nimi, mis teavituse saatis.

ANCS teenusega suhtlemiseks on nutikellal *ancs* moodul. ANCS teenuse kasutamise teeb keeruliseks see, et täieliku teavituse informatsiooni saamiseks peab lugema kaht erinevat karakteristikut. See tähendab ka seda, et uus teavitus võib tulla enne, kui vana teavituse atribuudid on kätte saadud. Samuti ei saa algatada uute atribuutide pärimist enne, kui vanale päringule on vastus tulnud. Selle probleemi lahendamiseks on *ancs* moodulil järjekord, kuhu pannakse kõik *Notification Source*'st tulnud uued teavitused. Kui teavitusele on kätte saadud kõik tema atribuudid, saadetakse see edasi *notification_event* sündmusena, eemaldatakse järjekorrast ja päritakse järgmise teavituse atribuute. Teavituste kontrollimiseks on *notification_action_event*, mille moodul saadab edasi *Control Point* karakteristikusse.

4.4.5 Tarkvara uuendamine

Tarkvarauuenduste tegemiseks kasutati MCUboot alglaadurit ja MCUmgr. MCUmgr on sardüsteemide manageerimise teek, mis antud projekti puhul paneb nutitelefonilt saadud uue tarkvara mitteaktiivsesse põhiprogrammi pesasse ja taaskäivitab süsteemi. Pärast taas-

käivitumist kontrollib MCUboot uue tarkvara autentsust ja korrektsust ning õnnestumise korral vahetab selle aktiivseks põhiprogrammiks.

MCUboot'i ja MCUmgr integreerimiseks oli vaja teha muudatusi Kconfig failis ning Devicetree konfiguratsioonis. Devicetree'sse oli vaja lisada neli säilmälu partitsiooni: üks MCUboot alglaaduri jaoks, kaks põhiprogrammi pesade jaoks ja viimane, mida kasutatakse põhiprogrammide vahetamiseks. Viimaseks oli vaja teha *dfu* moodul, mis süsteemi käivitumisel initsialiseerib MCUmgr teegi ja lisab sellele tagasikutse funktsiooni. Tagasikutse funktsiooni kutsutakse perioodiliselt välja, kui nutitelefon saadab tarkvarauuendust. Funktsiooni argumentideks on tarkvara suurus ja palju sellest on kohale jõudnud. Kasutades neid argumente, arvutab *dfu* moodul välja, mitu protsenti uuest tarkvarast on kohale jõudnud ja edastab selle *dfu_event* sündmusena.

4.5 Kasutajaliides

Projekti nõuete täitmiseks tuli kasutajaliidese poole peal teha järgmised vaated:

- paroolivaade – nutitelefoniga ühendamisel parooli näitamiseks
- põhivaade – kellaaja/kuupäeva ja tehtud sammude kuvamiseks
- tarkvarauuenduse vaade – näitab tarkvarauuenduse progressi
- meediapleieri vaade – saab juhtida nutitelefoni meediapleierit ja kuvab hetkel mängiva loo informatsiooni
- nutitelefoni teavituste vaade
- kõnede vaade – näitab sissetulevaid/aktiivseid kõnesid ning kust saab neid vastu võtta/katkestada

Vaadeldes neid lähemalt, on võimalik jagada need vaated kahte kategooriasse: rakendused (*apps*) ja hüpikaknad (*popups*). Hüpikaknad on vaated, mis annavad kasutajale teada mingi sündmuse toimumisest, nagu näiteks tarkvarauuenduse ja teavituse vaated. Hüpikaknaid ei ole võimalik kasutaja poolt avada, ainult sulgeda (viibates ekraanil üles). Rakendused on vaated, mille vahel saab kasutaja vabalt ringi käia (viibates ekraanil vasakule või paremale), nagu näiteks kellaaja ja meediapleieri rakendus.

Oluline kasutajaliidese arhitektuuri puhul oli see, et uute vaadete lisamine oleks võimalikult lihtne ja nõuaks minimaalseid muudatusi UI arhitektuuri kihis. Selle eesmärgi saavutamiseks disainiti järgnev lahendus.

Uue vaate tegemisel tuleb see esiteks lisada vastavalt vaate tüübile kas rakenduste või hüpikakende enumisse. Enumi väärtuste järjekord on samuti oluline, kus rakenduste puhul

vastab see rakenduste järjekorrale ja hüpikakende puhul määrab see nende prioriteedi. Kui hüpikaknad eksisteerivad, on alati aktiivne kõige suurema prioriteediga hüpikaken. Selle sulgemisel otsitakse prioriteedi poolest järgmist hüpikakent. Kui selline leidub, siis avatakse see. Vastasel juhul naasetakse rakenduse vaate peale.

Pärast väärtuse enumisse lisamist tuleb vaatele defineerida kaks funktsiooni: *init* ja *deinit*. *Init* funktsioon kutsutakse välja, kui vaade muutub aktiivseks. Selles funktsioonis initsialiseeritakse vaate graafikaelemendid ja teised ressursid, nagu näiteks taimerid. *Deinit* funktsioon kutsutakse välja, kui vaade muutub mitteaktiivseks. Selles funktsioonis vabastatakse *init* funktsioonis allokeeritud ressursid, välja arvatud graafikaelemendid, kuna nende vabastamisega tegeleb graafikateek.

Viimaseks tuleb vastavalt vaate tüübile kasutada *UI_APP_DEFINE(type, api)* või *UI_POPUP_DEFINE(type, api)* makrot, kus *type* on vaate enumi väärtus ja *api* on struktuur, mis sisaldab eelnevalt defineeritud funktsioone. Need makrod defineerivad vaate andmestruktuuri ja panevad selle kindlasse mälusektorisse, kust *ui* moodul need süsteemi initsialiseerimisel üles leiab.

Vaadete disainimiseks kasutati LVGL SqureLine Studio töölauarakendust, mis tegi arendamise lihtsamaks ja kiiremaks, kuna muudatuste nägemiseks ei pidanud tarkvara kompileerima ja seadmele programmeerima.

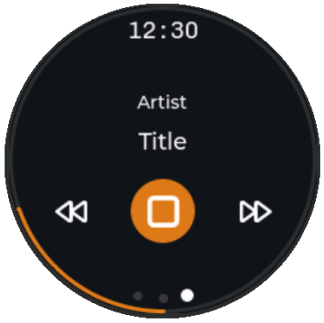
4.5.1 Rakendused

Kellaaja ja kuupäeva näitamisega tegeleb põhirakendus (joonis 6). Lisaks sellele kuvab põhirakendus ka tehtud samme, mida ta loeb *sensors_event* sündmustest. Tulevikus saab sellele ekraanile veel lisada näiteks aku protsendi ja pulsinäidu.

Joonis 6. Põhirakendus.

Nutitelefoni meediapleieri juhtimiseks on kellal meediapleieri rakendus (joonis 7). See

rakendus kuulab *media_player_event* ja *media_track_event* sündmusi ning kasutab nendest sisalduvat informatsiooni, et kuvada loo kestvust, nime ja artisti. Loo tööle/pausile panemiseks ning loo vahetamiseks on rakendusel kolm nuppu, mille vajutamisel saadetakse vastav tegevus edasi *media_action_event* sündmusena.



Joonis 7. Meediapleieri rakendus.

Paremaks orienteerumiseks rakenduste vahel joonistab *ui* moodul enne rakenduse *init* funktsiooni välja kutsumist vaate alla rakenduste arvu jagu indikaatoreid, kus aktiivse rakenduse indikaator on teistest suurem ja teise värviga. Et kasutaja ei peaks iga kord naasma põhirakendusse, kui soovib teada kellaaega, siis *ui* moodul joonistab samuti iga rakenduse ülaserva kellaaja.

4.5.2 Hüpikaknad

Nutitelefoniga ühendamisel parooli näitamiseks on parooli hüpikaken (joonis 8). *Passkey_event* sündmuse kätte saades kuvab vaade sündmuses sisaldava parooli. Pärast nutitelefonil parooli sisestamist või parooli sisestamise aja möödumist, kuvatakse kas ühendamine õnnestus või mitte.

(a) Ühendamisel.

(b) Ühendamine õnnestus.

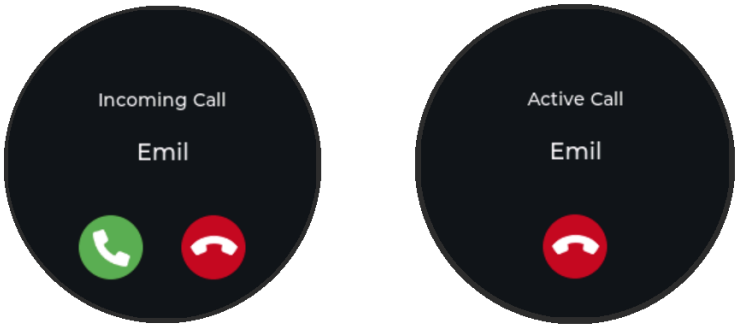
(c) Ühendamine ebaõnnestus.

Joonis 8. Parooli hüpikaken.

Tarkvarauuenduste progressi näitamiseks on kellal tarkvrauuenduste hüpikaken (joonis 9). See hüpikaken kuulab *dfu_event* sündmusi, et uuendada vastavalt sellele vaatel olevat progressiriba.

Joonis 9. Tarkvarauuenduse hüpikaken.

Sissetulevate ja aktiivsete kõnede jaoks on olemas kõnede hüpikaken (joonis 10), mis kuulab *notification_event* sündmusi ning kui sissetulev teavitus on sissetulevate või aktiivsete kõnede kategooriast, kuvatakse see. Kõnede vastu võtmiseks või ära panemiseks lisatakse vaatele ka nupud, mille vajutused edastatakse *notification_action_event* sündmustena.



(a) Sissetulev kõne. (b) Aktiivne kõne.

Joonis 10. Kõnede hüpikaken.

Nutitelefoni teavituste näitamiseks on teavituste hüpikaken (joonis 11). Teavituste hüpikaken kuulab *notification_event* sündmusi ning kuvab kõik, välja arvatud sissetulevate või aktiivsete kõnede kategooriast pärit teavitused. Vaate alaservas kuvatakse olemasolevate teavituste arv ja millist teavitust hetkel näidatakse. Teavituste vahel on võimalik liikuda, viibates ekraanil vasakule või paremale. Teavitused, mida kasutaja on näinud, kustutatakse hüpikakna sulgemisel ära.

Joonis 11. Nutitelefoni teavituste hüpikaken.

4.6 Aku kestvuse optimeerimine

Aku pealt töötavatel seadmetel, nagu seda on nutikell, on voolutarbimise optimeerimine oluline osa seadme arendusest. See ei ole mitte ainult oluline mugavuse koha pealt, kuna seadet peab harvemini laadima, vaid ka aku eluea seisukohalt. Kuna akudel on kindel arv laadimistsükleid, mis jääb liitiumpolümeerakude puhul vahemikku 300-500 tsüklit [35], siis mida harvemini akut laaditakse, seda kauem peab aku ka vastu. Projekti eesmärgiks oli, et nutikella aku kestvus peab olema vähemalt kaks nädalat. Selle nõude täitmiseks on vaja keskmine voolutarbimine alandada 600 µA'ni. Voolutarbimise optimeerimiseks on vaja välja selgitada, milliste riistvarakomponentide voolutarbimist on tarkvaras võimalik vähendada, millal toiterežiimi muudetakse ja kuidas sellest erinevatele süsteemi osadele teada anda.

Toiterežiimi muutustest teada andmiseks kasutati *Application Event Manager*'i olemasolevat moodulit *Power Manager Module* [36]. *Power Manager Module* on lihtne moodul, mis saadab konfigureeritud aja möödudes *power_down_event* sündmuse, mille kätte saades peavad moodulid voolutarbimist vähendama. Unerežiimist välja tulemiseks või unerežiimi minemise taimeri lähtestamiseks saavad moodulid saata *wake_up_event* sündmuse. Seda sündmust saadetakse, kui seadmele tuleb teavitus, ekraani puudutatakse või seade tõstetakse lugemiseks üles. *Power Manager Module*'iga on võimalik toiterežiimi muutusest anda teada ainult teistele moodulitele. Et moodulid saaksid seda informatsiooni draiveritele edastada, kasutati *Device Runtime Power Management* [37] teeki. *Device Runtime Power Management* tööpõhimõte sarnaneb sellele, kuidas dünaamilise mälu koristamine töötab mõnes programmeerimiskeeles, nagu näiteks Python. Igal draiveril või nii-öelda ressursil on muutuja, mis viitab sellele, mitu kasutajat sellel ressursil on. Kui muutuja väärtus on suurem kui null, siis on teada, et ressurss on kasutusel. Kui muutuja on null, siis ressursil kasutajaid ei ole ja võib voolutarvet piirata. Seega peavad draiveritega suhtlevad moodulid ka *wake_up_event* sündmuse saades andma teada, et kasutavad draiverit ja *power_down_-*

event saades teada andma, et on lõpetanud selle kasutamise. Samuti on vaja *Device Runtime Power Managment* sisse lülitada draiveri poole pealt ja implementeerida funktsioon, mida kutsutakse välja, kui toiterežiim on muutunud. Vältimaks olukordi, kus draiver pannakse magama samal ajal, kui tal on seadmega suhtlus pooleli, tuleb iga transaktsioon ümbritseda ka *pm_device_busy_set* ja *pm_device_busy_clear* funkstioonidega.

Seadme kõige suurem voolutarbija on ekraan, tarbides umbes 18 mA, samal ajal kui ülejäänud süsteem tarbib umbes 2 mA. Ainuüksi ekraani voolutarbimise optimeerimisega on võimalik aku kestvust suurendada poolelt päevalt nelja päevani. Ekraan koosneb taustavalgustusest, puutesensorist ja ekraani kontrollerist. Taustavalguse juhtimine on nendest kõige lihtsam ja vajab lihtsalt ühe mikrokontrolleri väljundi nivoo muutmist. Ekraani kontrolleril on olemas kaks käsklust: üks unerežiimi minekuks ja teine sealt välja tulemiseks. Unerežiimis lülitakse välja pingeregulaatorid ja ekraani uuendamine, aga ekraani mälu hoitakse alles ning pärast üles äratamist ei ole seda vaja uuesti saata. Sarnaselt ekraani kontrollerile on puutesensoril samuti käsklus unerežiimi minekuks, aga sellest välja tuleb sensor alles pärast ekraani puudutamist.

Järgmine voolutarbimise poolest on mikrokontroller. Üks kõige suuremat mõju avaldav faktor on see, millist pingeregulaatorit mikrokontroller kasutab: lineaarset pingeregulaatorit või pinge impulss-stabilisaatorit. Kasutades lineaarse pingeregulaatori asemel pinge impulss-stabilisaatorit, on võimalik vähendada energiakasutust kuni 35% [38]. Kuna nutikella riistvara juba toetab pinge impulss-stabilisaatori kasutamist, siis selle tarkvaras sisse lülitamiseks on vaja ainult Kconfig'i muuta.

Mikrokontrolleril endal on kaks erinevat unerežiimi: *System OFF* ja *System ON*. *System OFF* on kõige madalama voolutarbimisega režiim, kus mikrokontrolleri põhifunktsionaalsused on välja lülitatud. Sellest režiimist on võimalik väljuda ainult siis, kui süsteem tuvastab ettenähtud sisendi nivoo muutuse, mille tagajärjel süsteem taaskäivitatakse. *System ON* on unerežiim, kus protsessor ja perifeeriad jäetakse tööle, mistõttu on sellel ka suurem voolutarve. Erinevalt *System OFF*'ist, saab sellest režiimist väljuda ilma taaskäivitamiseta. Lisaks kahele unerežiimile on võimalik ka plokkide kaupa operatiivmälu välja lülitada.

Tabel 1. nRF52840 voolutarve unerežiimides [39].

Unerežiim	**Voolutarbimine (µA)**
System OFF, RAM'i säilitamiseta	0.40
System OFF, RAM'i säilitamisega	1.86
System ON, RAM'i säilitamiseta	0.97
System ON, RAM'i säilitamisega	2.35

Nagu tabelist 1 näha on, siis nende unerežiimide maksimaalne vahe on ainult 2 µA. See

minimaalne voolutarbe võit aga ei olnud autori arvates väärt lisakeerukust, mida see endaga kaasa toob. Seetõttu otsustas autor kasutada ainult *System ON* režiimi, millesse Zephyr ka automaatselt läheb, kui ühelgi protessil ei ole tööd teha.

Suurt rolli mängivad ka BLE reklaamimise ja ühenduse parameetrid. Need parameetrid olenevad küll suuresti tootest, kuid Apple'l on kindlad juhised [40], mida tuleks järgida parameetrite valimisel, et tagada kiire ühendumine ja stabiiline side. Parema ülevaate saamiseks on Nordic Semiconductor'il olemas ka voolutarbimise kalkulaator, millega saab arvutada eeldatavat voolutarbmist erinevate BLE parameetritega.

Reklaamipakettide saatmisel on peamine parameeter saatmissagedus. Mida sagedamini pakette saadetakse, seda kiiremini leitakse seade üles ja ühendatakse, kuid see-eest tarbitakse ka rohkem voolu. Apple soovitab alguses saata reklaamipakette iga 20 ms tagant ja kui seadmega ühendumist ei ole toimunud 30 sekundi jooksul, siis suurendada saatmisintervalli üheks võimalikuks kindlaksmääratud üheksast väärtusest vahemikus 152.5 - 1285 millisekundit. Autor valis aeglasemaks intervalliks 417.5 ms, millel võrreldes 152.5 ms on kolm korda väiksem voolutarbimine, kuid ka piisavalt kiire ühildumiskiirus.

Ühenduse voolutarbimist mõjutavad peamised parameetrid on ühenduse sagedus ja perifeerse seadme latentsus. Ühenduse sagedus näitab, kui tihti keskne seade võtab perifeerse seadmega ühendust. Sarnaselt reklaamipakettidele saab voolutarvet vähendada ühenduspakettide sagedust vähendades. See aga ei ole ideaalne, kuna see viib ühenduse kiiruse ja jõudluse alla. Perifeerse seadme latentsus näitab, mitu korda võib perifeerne seade ignoreerida keskse seadme pakette, kui tal ei ole midagi uut saata. Latentsuse suurendamine on hea viis voolutarbe vähendamiseks ja samas ei kannata selle tõttu ühenduse kiirus. Miinuseks suurel latentsusel on aga see, et kesksel seadmel läheb kauem aega ühenduse katkemise tuvastamiseks. Et mitte kaotada reageerimiskiirust ja läbilaskevõimet, otsustas autor kasutada suurt ühenduse sagedust ja voolutarbe madaldamiseks tõsta latentsust.

Viimane seade, mille voolutarbimist saab tarkvaras muuta, on inertsiaalsensor. Inertsiaalsensori voolutarbimist mõjutab see, mis sensorid on sisse lülitatud ja nende diskreetimissagedus ehk see, kui tihti sensorite väärtusi loetakse. Olemasolevatest sensoritest kasutatakse ainult kiirendusandurit, mida on vaja sammude lugemiseks ja saamaks teada, kui kasutaja on kella üles tõstnud, et süsteem üles äratada. Seega ei ole võimalik kiirendusandurit välja lülitada, kuna sellest sõltub kella toimimine. Samuti ei ole diskreetimissagedust võimalik alandada, kuna see on juba kõige väiksema sageduse peal, millega sammude lugemine veel töötab. Lisaks on seadmel ka olema unerežiim, aga kuna selles režiimis ei tööta kiirendusandur, siis ei saa seda kasutada.

5. Testimine, tulemused ja edasised arendusvõimalused

Järgnevates peatükkides räägitakse lähemalt lahendusele seatud nõuete testimisest, testimise tulemustest ja edasistest arendamise võimalustest.

5.1 Testimine ja tulemused

Meediapleieri, teavituste, kõnede ja aja sünkroniseerimise testimiseks kasutati iOS seadet. Aja sünkroniseerimise testimiseks ühendati iOS seade nutikellaga ja kontrolliti, kas nutikella aeg muutub õigeks. Samuti kontrolliti seda, et kui seade on juba ühendatud ja iOS seadmel muutub kellaaeg, kajastuks see ka nutikellal. Meediapleieri testimiseks kasutati Spotify ja YouTube rakendusi ning kõnede ja teavituste testimiseks Discord'i. Teavituste testimisel leiti, et kui teavituse sõnumi pikkus oli suurem kui nutikell päris, lõppes saadud sõnum baitidega e2 80, mis ei vasta ühelegi ASCII nähtavale sümbolile. Tuli välja, et ANCS teek küsis ühe baidi rohkem kui pidi ja sõnum lõppes tegelikult baitidega e2 80 a6, mis vastab mõttepunktide sümbolile. Kuna C keeles sõnede lõppu tähistatakse null baidiga, siis ANCS teek asendas viimase baidi sellega. Probleemi lahendamiseks viidi parandus sisse ANCS teegis.

Tarkvarauuenduste testimiseks kasutati nRF Connect Device Manager rakendust nii Androidi kui ka iOS seadmel. Androidi seadmel töötas tarkvarauuendamine iga kord, kuid iOS seadmega ei olnud selle alustamine väga töökindel ja seda pidi mitu korda proovima. Laadima hakkamisel suutis ta selle aga edukalt lõpetada. Praegune hüpotees on see, et tarkvara laadimise algatamisel hakkab MCUmgr tarkvarapesa kustutama ning sel ajal ei vastata päringu algatajale. Saatja poole peal on päringu aegumise taimer, mis iOS rakenduses on palju lühem kui Androidi omas ja seetõttu aegub iOS rakendusel saatmispäring enne tarkvarapesa kustutamise lõppu. Seda hüpoteesi toetavad ka logid, kust on näha, et pärast tarkvarauuenduse algatamist saadetakse sõnum mälu kustutamisest pika viitega. Hüpoteesi aga ei jõutud täielikult tõestada, kuna selle testimiseks oleks vaja arendada oma mobiilirakendus.

Aku kestvuse prognoosimiseks mõõdeti voolutarbimist erinevates režiimides (tabel 2).

Tabel 2. Nutikella voolutarbimine erinevates režiimides.

Režiim	**Voolutarbimine (mA)**	**Aku kestvus (päeva)**
Ärkvel, BLE reklaamimine	19.97	0.4
Ärkvel, BLE ühendatud	19.97	0.4
Unerežiimis, BLE reklaamimine	0.143	58
Unerežiimis, BLE ühendatud	0.125	66

Nagu tabelist näha, on nutikella aku kestvus ärkvelolekus alla päeva, kuid unerežiimil on see umbes kaks kuud. Kui eeldada, et seade on ärkel päevas 10 minutit, siis peab see vastu umbes kuu aega, mis on kaks korda pikem, kui projekti eesmärkides sai määratud.

Võrreldes eelnevalt mainitud vabavaraliste nutikelladega, on saavutatud aku kestvus neli korda pikem (tabel 3). Samas on aga autori poolt loodud nutikell mõõtmetelt suurem, mida on võimalik parendada parema korpuse ja trükkplaadi disainiga.

Tabel 3. Autori lahenduse võrdlus kahe olemasoleva vabavaralise nutikellaga [41]

	Autori lahendus	**Watchy**	**PineTime**
Mikrokontroller	nRF52840	ESP32-PICO-D4	nRF52832
Sensorid	Kiirendusandur + güroskoop	Kiirendusandur	Kiirendusandur + pulssoksümeeter
Laius/Pikkus	46/55	34/46	37.5/40
Paksus	13	9	11
Kaal	33	18	38
Aku kestvus	1 kuu	1 nädal	1 nädal
Ekraan	1.28” 240x240 LCD	1.54” 200x200 e-paber	1.3” 240x240 LCD
Puutetundlik	Jah	Ei	Jah
Nupud	0	4	1
Veekindlus	-	-	IP67
Vabavaraline riistvara	Jah	Jah	Skeemid
Ühenduvus	BLE	BLE + Wi-Fi	BLE
Operatiivmälu	256 KB	520 KB	64 KB
Säilmälu	1 MB	4 MB	512 KB + 4 MB

Nutikella kasutajasõbralikkuse ja funktsionaalsuste hindamiseks korraldati küsitlus pere ja tuttavate hulgas. Küsitluses osales kuus inimest vanuses 19–66 aastat, neli naist ja kaks meest. Küsitlusest ilmnes, et kaks küsitletavat omavad nutikella. Peamised põhjused, miks teistel nutikelli ei olnud, oli see, et need on liiga kallid ja ei ole vaja läinud. Järgmiseks tahtis autor teada, kui oluline oli küsitletute jaoks privaatsus ja kui kaua peaks aku minimaalselt kestma. Privaatsuse olulisust hinnati skaalal nullist viieni, kus null oli mitte

üldse ja viis väga oluline. Kaks vastajat hindas privaatsuse olulisust 5-ga, kaks 4-ga ja kaks 3-ga. Minimaalseks aku kestvuseks sooviksid viis inimest vähemalt nädal aega ja ühele piisaks päevast. Pärast algsetele küsimustele vastamist andis autor lõputöö käigus valminud nutikella küsitletavale ja palus sellega tutvuda. Pärast seadmega tutvumist paluti hinnata seadme kasutajasõbralikkust skaalal nullist viieni. Neli vastajat hindas kasutajasõbralikkust 4-ga, üks 5-ga ja üks 3-ga. Peamine murekoht, mis välja toodi oli, et ekraanipuudutused ei registreeru vahepeal. Viimaseks tahtis autor teada, mis funkstionaalsustest oleksid küsitletavad veel huvitatud. Välja pakuti unekvaliteedi monitoorimine, taimer, maksete sooritamine, navigatsiooni informatsiooni kuvamine, pulsi mõõtmine, ilma kuvamine, nutitelefoni üles leidmine ja teiste seadmete juhtimine, nagu näiteks kaamera ning televiisori.

Kõik eesmärkides püstitatud nõuded said täidetud, ning kogu projekt on üleval GitHub'is.

5.2 Edasised arendusvõimalused

Vaatamata sellele, et lõputöö raames said täidetud kõik eesmärkides püstitatud nõuded nutikellale, on arendamise võimalusi veel palju.

Üks olulisemaid tegevusi oleks arendada Androidi rakendus, mis suhtleks kellaga, kuna praegu töötab nutikell ainult iOS seadmetega. Samuti tuleks läbi mõelda seadme turvalisus ja kuidas süsteemitõrgete ilmumisel käitutakse. Tõrgete põhjuste leidmiseks oleks kasulik arendada süsteem, mis salvestab tõrke ilmumisel süsteemi seisundi ja järgmisel ühendumisel edastab selle informatsiooni nutitelefonile. Kuna autor ei ole kogenud kasutajaliideste disainimises, siis oleks see samuti üks asi, mida saaks parendada. Lisaks võiks proovida saada tarkvara tööle emulaatori peal, nagu näiteks Renode [42], mis oleks kasulik integratsioonitestide jaoks ning teeks arendamise ja debugimise lihtsamaks.

Tarkvaraarendamise käigus tuli nähtavale ka paar riistvara puudujääki ja arenguvõimalust. Esiteks oli ununenud lisada akupinge monitoorimisahel, mille tõttu praeguse riistvaraga ei ole võimalik aku protsenti kuvada. Samuti võiks lisada rohkem sensoreid, nagu näiteks pulssoksümeeter, magnetomeeter ja baromeeter. Kuna tarkvarauuenduste tugi võtab ära enam-vähem poole säilmälust, oleks kasulik lisada ka mikrokontrolleriväline säilmälu, kuhu saaks salvestada tarkvarauuendusi ning teisi ressursse nagu näiteks pildid ja fondid. Viimaseks oleks vaja disainida veekindel korpus ja samuti seda testida. Lisaks laadimiskontaktidele võiks korpusest välja tuua ka programmeerimiskontaktid, kuna praeguse lahenduse korral on seadme programmeerimine üsna tülikas.

6. Kokkuvõte

Antud lõputöö eesmärgiks oli luua vabavaraline nutikella tarkvara eelnevalt autori poolt tehtud riistvarale. Tarkvarale seati mitu nutitelefoniga suhtlemisega seotud nõuet, sealhulgas teavituste kuvamine, meediapleieri juhtimine, kellaaja sünkroniseerimine ja tarkvarauuendamine. Lisaks pidi seade kuvama tehtud samme ja aku kestvus olema vähemalt kaks nädalat.

Töö käigus analüüsiti olemasolevaid vabavaralisi nutikellade lahendusi ning valiti välja sobilik alglaadur, operatsioonisüsteem, graafikateek ja BLE teenused.

Eesmärkide saavutamiseks tuli õppida valitud tehnoloogiaid, kirjutada draiverid riistvaraseadmetele, mõelda välja äriloogikakihi arhitektuur, kirjutada moodulid nutitelefoniga suhtlemiseks ning kasutajaliides.

Nutitelefoniga suhtlemise nõuete tulemuste valideerimiseks kasutati nii Androidi kui ka iOS seadmeid. Aku kestvuse testimiseks mõõdeti seadme voolutarbimist erinevates režiimides.

Töö tulemusena valminud nutikella tarkvara täidab kõiki esialgseid nõudeid ning aku kestab kauem kui algselt oli planeeritud. Seadme ekraani tööloleku aeg on 10 tundi ning kui ekraan ei ole tööl, siis kestab aku umbes kaks kuud. Kuigi kõik nõuded said täidetud, on tulevikuks veel palju arendusvõimalusi. Üks olulisemaid oleks Androidi rakenduse kirjutamine nutikellaga suhtlemiseks, kuna enamik funktsionaalsusi töötab praegu ainult iOS seadmetega. Nii projekti kood kui ka riistvara disainifailid on üleval GitHub'is.

Kasutatud kirjandus

[1] *Heart health notifications on your Apple Watch*. [kasutatud: 11.02.2023]. URL: https://support.apple.com/en-gb/HT208931.

[2] Luc Rocher, Julien M. Hendrickx ja Yves-Alexandre de Montjoye. „Estimating the success of re-identifications in incomplete datasets using generative models“. *Nature Communications* 10.1 (juuli 2019), lk. 3069. ISSN: 2041-1723. DOI: 10.1038/s41467-019-10933-3. URL: https://doi.org/10.1038/s41467-019-10933-3.

[3] *Mr Justin Gutmann v Apple Inc.* [kasutatud: 13.02.2023]. URL: https://www.catribunal.org.uk/sites/cat/files/2022-09/2022.09.14_Gutmann_Apple_Summary_of_Claim_Final.pdf.

[4] *D. Carter, BMW asks consumers to pay subscription for features already installed in car, The Brussels Times, 2022*. [kasutatud: 13.02.2023]. URL: https://www.brusselstimes.com/257605/bmw-asks-consumers-to-pay-subscription-for-features-already-installed-in-car.

[5] *KiCad*. [kasutatud: 04.03.2023]. URL: https://www.kicad.org/.

[6] *JLCPCB*. [kasutatud: 04.03.2023]. URL: https://jlcpcb.com/.

[7] *PineTime*. [kasutatud: 12.02.2023]. URL: https://wiki.pine64.org/wiki/PineTime.

[8] *Watchy*. [kasutatud: 12.02.2023]. URL: https://watchy.sqfmi.com/.

[9] *FreeRTOS*. [kasutatud: 26.03.2023]. URL: https://www.freertos.org/.

[10] *nRF5 SDK*. [kasutatud: 26.03.2023]. URL: https://www.nordicsemi.com/Products/Development-software/nrf5-sdk.

[11] *Nordic Devzone - nRF Connect SDK and nRF5 SDK statement*. [kasutatud: 26.03.2023]. URL: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/nrf-connect-sdk-and-nrf5-sdk-statement.

[12] *Zephyr Project*. [kasutatud: 26.03.2023]. URL: https://zephyrproject.org/.

[13] *Zephyr Project - API Overview*. [kasutatud: 26.03.2023]. URL: https://docs.zephyrproject.org/latest/develop/api/overview.html.

[14] *Zephyr Project - Configuration System (Kconfig)*. [kasutatud: 26.03.2023]. URL: https://docs.zephyrproject.org/latest/build/kconfig/index.html.

[15] *Zephyr Project - Devicetree*. [kasutatud: 26.03.2023]. URL: https://docs.zephyrproject.org/latest/build/dts/index.html.

[16] *nRF Connect SDK - nRF Secure Immutable Bootloader*. [kasutatud: 26.03.2023]. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/samples/bootloader/README.html.

[17] *MCUboot*. [kasutatud: 26.03.2023]. URL: https://www.mcuboot.com/index.html.

[18] *uGFX*. [kasutatud: 26.03.2023]. URL: https://ugfx.io/.

[19] *LVGL*. [kasutatud: 26.03.2023]. URL: https://lvgl.io/.

[20] *Qt for MCU*. [kasutatud: 26.03.2023]. URL: https://www.qt.io/product/develop-software-microcontrollers-mcu.

[21] *uGFX - Pricing*. [kasutatud: 26.03.2023]. URL: https://ugfx.io/pricing.

[22] *SquareLine Studio*. [kasutatud: 26.03.2023]. URL: https://squareline.io/.

[23] *Qt for MCU - Licensing Model*. [kasutatud: 26.03.2023]. URL: https://doc.qt.io/QtForMCUs-2.2/qtul-licensing.html.

[24] *Qt for MCU - Requirements*. [kasutatud: 26.03.2023]. URL: https://doc.qt.io/QtForMCUs-2.2/qtul-overview.html#minimum-hardware-requirements.

[25] *Bluetooth Technology Website - Heart Rate Service*. [kasutatud: 26.03.2023]. URL: https://www.bluetooth.com/specifications/specs/heart-rate-service-1-0/.

[26] *Bluetooth Technology Website - Current Time Service*. [kasutatud: 26.03.2023]. URL: https://www.bluetooth.com/specifications/specs/current-time-service-1-1/.

[27] *Bluetooth Technology Website - Media Control Service*. [kasutatud: 26.03.2023]. URL: https://www.bluetooth.com/specifications/specs/media-control-service/.

[28] *Bluetooth Technology Website - Alert Notification Service*. [kasutatud: 26.03.2023]. URL: https://www.bluetooth.com/specifications/specs/alert-notification-service-1-0/.

[29] *Bluetooth Technology Website - Telephone Bearer Service*. [kasutatud: 26.03.2023]. URL: https://www.bluetooth.com/specifications/specs/telephone-bearer-service-1-0/.

[30] *Apple Notification Center Service*. [kasutatud: 26.03.2023]. URL: https://developer.apple.com/library/archive/documentation/CoreBluetooth/Reference/AppleNotificationCenterServiceSpecification/Specification/Specification.html.

[31] *Apple Media Service*. [kasutatud: 26.03.2023]. URL: https://developer.apple.com/library/archive/documentation/CoreBluetooth/Reference/AppleMediaService_Reference/Introduction/Introduction.html.

[32] *nRF Connect SDK - Application Event Manager*. [kasutatud: 07.04.2023]. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/libraries/others/app_event_manager.html.

[33] *nRF Connect SDK - CAF: Bluetooth LE state module*. [kasutatud: 06.04.2023]. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/libraries/caf/ble_state.html.

[34] *nRF Connect SDK - CAF: Bluetooth LE advertising module*. [kasutatud: 06.04.2023]. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/libraries/caf/ble_adv.html.

[35] *Grepow - The Charging Cycles of Lithium-ion Polymer Batteries*. [kasutatud: 06.04.2023]. URL: https://www.grepow.com/blog/charging-cycles-of-lithium-ion-polymer-batteries.html.

[36] *nRF Connect SDK - CAF: Power manager module*. [kasutatud: 06.04.2023]. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/libraries/caf/power_manager.html.

[37] *Zephyr Project - Device Runtime Power Management*. [kasutatud: 06.04.2023]. URL: https://docs.zephyrproject.org/latest/services/pm/device_runtime.html#pm-device-runtime.

[38] *Nordic Devzone - Optimizing Power on nRF52 Designs*. [kasutatud: 06.04.2023]. URL: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/optimizing-power-on-nrf52-designs.

[39] *nRF52840 Datasheet*. [kasutatud: 12.02.2023]. URL: https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.7.pdf.

[40] *Accessory Design Guidelines for Apple Devices*. [kasutatud: 06.04.2023]. URL: https://developer.apple.com/accessories/Accessory-Design-Guidelines.pdf.

[41] *Crowd Supply - Watchy*. [kasutatud: 16.05.2023]. URL: https://www.crowdsupply.com/sqfmi/watchy.

[42] *Renode*. [kasutatud: 13.04.2023]. URL: https://renode.io/.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

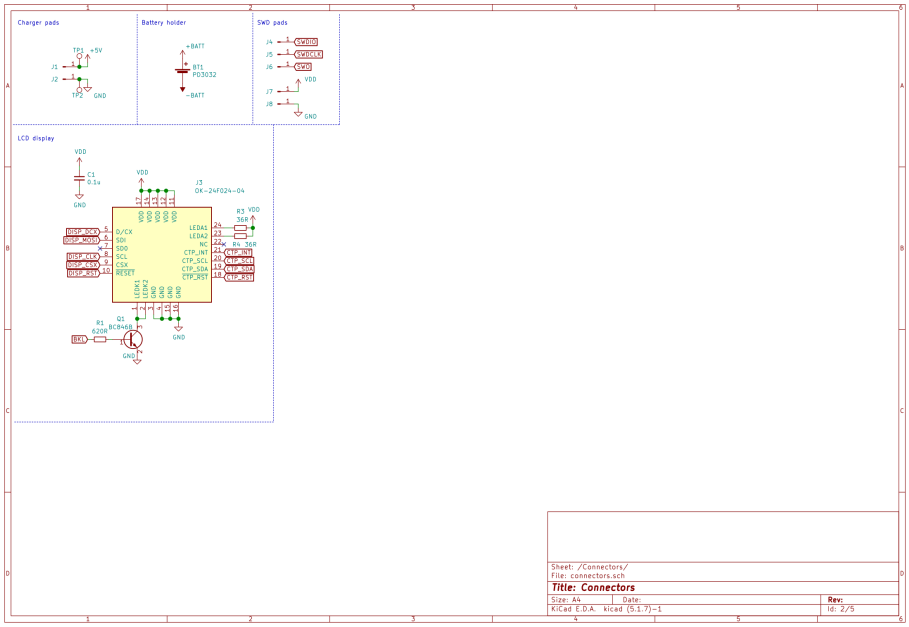
Mina, Emil Marvin Mikk

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Nutikella tarkvara arendamine", mille juhendaja on Peeter Ellervee
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

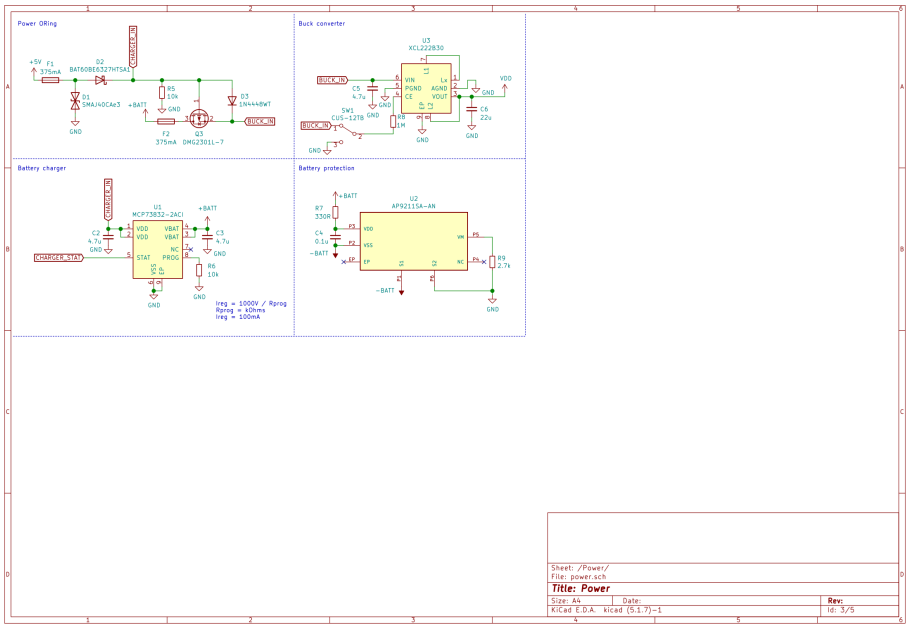
22.05.2023

[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

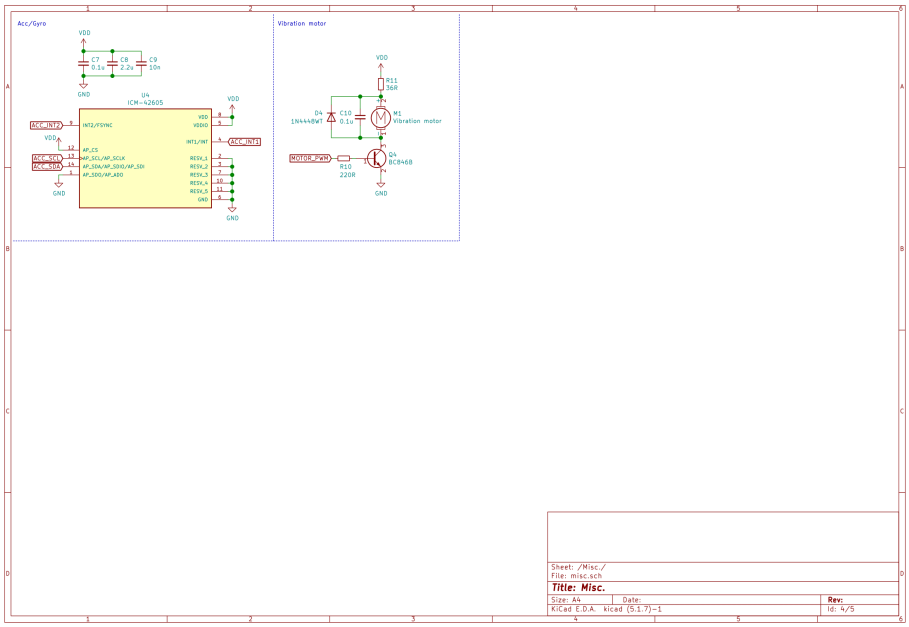
Lisa 2 – Trükkplaadi skeemid



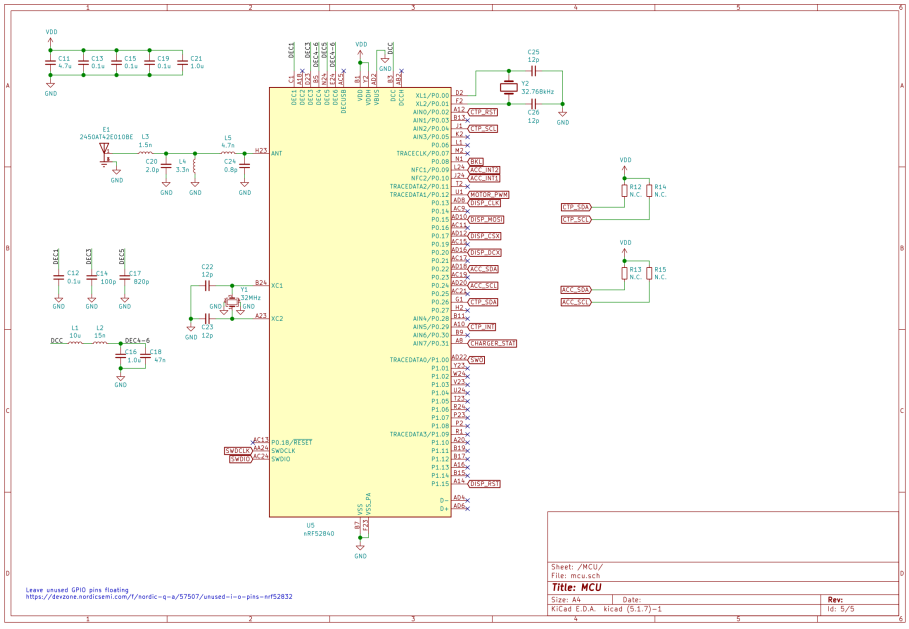
Joonis 12. Trükkplaadi pistikute skeem.



Joonis 13. Trükkplaadi toiteskeem.

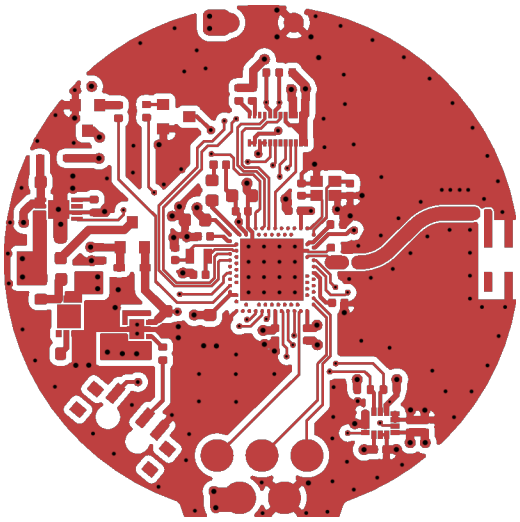


Joonis 14. Trükkplaadi sensorite skeem.



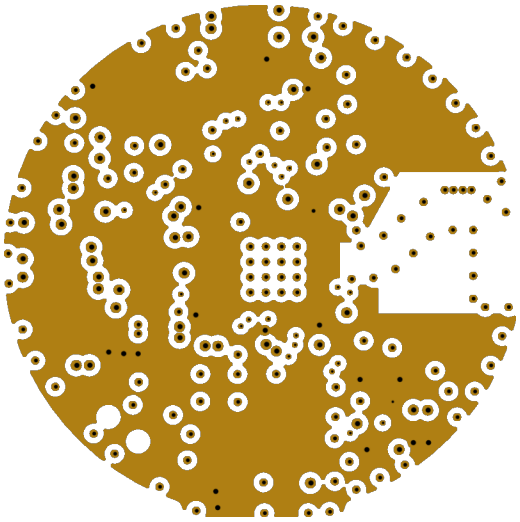
Joonis 15. Trükkplaadi mikrokontrolleri skeem.

Lisa 3 – Trükkplaadi disain

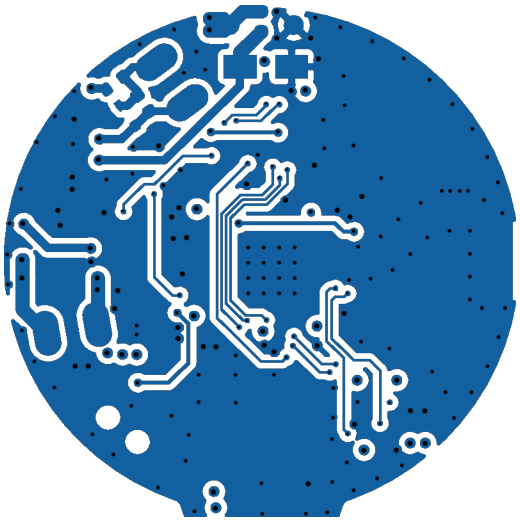


(a) Ülemine kiht.

(b) Esimene sisemine kiht.



(c) Teine sisemine kiht.



(d) Alumine kiht.

Joonis 16. Trükkplaadi kihid.