
Jani Piri

# Käsiproteesin ohjain – prototyypin kehitystyö

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

31.5.2022

# Tiivistelmä


| | |
|---|---|
| Tekijä: | Jani Piri |
| Otsikko: | Käsiproteesin ohjain – prototyypin kehitystyö |
| Sivumäärä: | 46 sivua + 3 liitettä |
| Aika: | 31.5.2022 |
| Tutkinto: | Insinööri (AMK) |
| Tutkinto-ohjelma: | Tieto- ja viestintätekniikka |
| Ammatillinen pääaine: | Hyvinvointi- ja terveysteknologia |
| Ohjaaja: | Lehtori Sakari Lukkarinen |


---


Tämä insinöörityö on prototyypin tutkimus- ja kehitystyö. Insinöörityön aihe rajattiin alkuperäisestä ideasta rakentaa kokonainen robottiproteesiraaja, joka mahdollistaisi amputaatiossa raajansa menettäneelle samat käden liikkeet kuin biologinen raaja. Työn lähtökohtana on kehittää proteesiraajan ohjaimen prototyyppi, joka mittaa käsivarren lihaksesta sen tuottaman sähköisen impulssin. Signaalin tarkoitus on muuttaa servomoottorin pyörimissuuntaa kättä koukistaessa ja ojentaessa rannenivelestä.

Työssä perehdytään käsivarren ja luustolihasten anatomiaan sekä lihasten fysiologiaan. Tarkoituksena on luoda ymmärrys prototyypin kehitykseen vaikuttavista luonnollisista tekijöistä. Lisäksi perehdytään elektromyografiaan, jolla luodaan ymmärrys, miten tuo lihaksessa tapahtuva sähköinen impulssi saadaan digitaalipiirille muutetuksi.

Prototyypin kehitysalustana toimii Raspberry Pi Pico -mikro-ohjain, jota ohjelmoidaan MicroPython-ohjelmointikielellä. Prototyypin tulee toimia itsenäisesti ilman tietokonetta sen käyttötarkoituksen mukaan eli ihmisen mukana kulkeva proteesiraajana. Pico suoriutui sille vaadituista tehtävistä. Sen rajoitteetkin tuotiin esille, sillä lihasta mitatessa dataa tuotetaan lyhyessä ajassa paljon.

Työn tuloksena oli käsiproteesiohjaimen prototyyppi. Ranteen koukistus- ja ojennusliikkeen lisäksi kehityksen aikana rinnalle tuotettiin nyrkin puristuksen havaitseminen. Prototyypin lisäksi kehitettiin EMG:n mittaamiseen ja tallentamiseen tarkoitettu MicroPython-ohjelma.



Avainsanat: elektromyografia, EMG, proteesi, MicroPython, signaalinkäsittely, prototyyppi, Raspberry Pi Pico mikro-ohjain

# Abstract


| | |
|---|---|
| Author: | Jani Piri |
| Title: | Prosthetic Arm Controller – Prototype Development |
| Number of Pages: | 46 pages + 3 appendices |
| Date: | 31 May 2022 |
| Degree: | Bachelor of Engineering |
| Degree Programme: | Information Technology |
| Professional Major: | Health Technology |
| Supervisor: | Sakari Lukkarinen, Senior Lecturer |


---


This bachelor thesis is a prototype research and development work. The original idea for study was to develop a robotic prosthetic arm, so a person having lost the limb in amputation could get a substitute bionic limb that performs as well as the biological one. The scope was limited to the development of the logic for a microcontroller, which measures the electrical impulses produced by the forearm muscle. The signal is used to change the rotation of a servomotor when the hand is bent and extended from the wrist joint.

The anatomy of the arm and skeletal muscles as well as the physiology of the muscles are introduced. The aim is to provide an understanding of the natural factors influencing the prototype development. The study focuses on electromyography, which grants knowledge of how the electrical impulses in a muscle can be converted into a digital circuit.

The development platform for the prototype is the Raspberry Pi Pico microcontroller programmed in the MicroPython programming language. The prototype must operate independently without a computer for its intended purpose – a prosthetic limb attached to a human. Pico performed the required tasks; however, its limitations became clear as a large quantity of data is produced in a brief time when measuring the muscle.

The result of the study was a prototype of a prosthetic arm controller. In addition to wrist flexion and extension movements created for the controller, a clench of a fist motion was also added. As a side product a MicroPython program for EMG data collection was developed.



Keywords: electromyography, EMG, prosthesis, MicroPython, signal processing, prototype, Raspberry Pi Pico microcontroller

# Sisällysluettelo

## Lyhenteet

| | |
|---|---|
| A/D | Analogia-digitaali. |
| ADP | Adenosiinidifosfaatti. Energiaa sisältävä molekyyliyhdiste, joka syntyy ATP:n pilkkoutuessa lihaksessa. |
| ATP | Adenosiinitrifosfaatti. Energiaa sisältävä molekyyliyhdiste. |
| $Ca^+$ | Kalsiumioni. |
| EMG | Elektromyografia eli lihassähkökäyrä. |
| Hz | Hertsi. |
| $K^+$ | Kaliumioni. |
| $Na^+$ | Natriumioni. |
| Pico | Raspberry Pi Pico -mikro-ohjain. |
| PWM | Pulssinleveysmodulaatio, eng. Pulse-Width Modulation. |
| RMS | Neliöllinen keskiarvo, eng. Root Mean Square. |
| AVR | Tasasuunnatun signaalin liukuva keskiarvo, eng. Average Rectified Value. |
| $V_{out}$ | A/D-muuntimen mittaama jännite. |
| $V_{ref}$ | Jännitereferenssi. |

# 1 Johdanto

Ideana on kehittää prototyyppi laitteesta, joka ottaa vastaan lihassähkökäyrän (elektromyografia, EMG) ja signaalin avulla vaihtaa servomoottorin pyörimissuuntaa rannetta ojentaessa ja koukistaessa. Käsivarteen kiinnitettävä pintaelektrodi on yhteydessä piiriin, jossa EMG-signaali käsitellään. Piirillä muutetaan tämä luustolihaksessa tapahtuva sähköimpulssi servomoottorin liikkeeksi. Työn pääpaino on mikro-ohjaimen logiikan tuottamisessa.

Inspiraatio tälle työlle on alkuperäinen idea rakentaa robotisoitu proteesiraaja, joka mahdollistaisi amputaatiossa raajansa menettäneelle samat toiminnot, joita biologinenkin käsi tarjoaa. Aihe on tarkkaan rajattu tästä ideasta. Tämän insinöörityön tarkoitus on luoda perustat tuolle robottikädelle, josta sitä voi lähteä jatkokehittämään. Työ keskittyy ihmisen ja piirin rajapintaan.

Vaikka kokonaista kättä ei lähdetä rakentamaan, on se käsitteenä vahvasti mukana työtä tehdessä. Työssä perehdytään yläraajan anatomiaan ja fysiologiaan, jolla hankitaan ymmärrys niistä biologisista tekijöistä, jotka vaikuttavat prototyypin kehitykseen. Työn lopputuloksena tulisi olla toimiva ja testattu prototyyppi laitteesta, joka ottaa vastaan EMG-signaalin, prosessoi signaalin digitaalipiirillä sekä näiden signaalien muuttuessa liikuttaa servomoottoria halutulla tavalla.

Insinöörityön aloituspalaverin yhteydessä ohjaajani kanssa kävi ilmi, että Metropolian hyvinvointi- ja terveysteknologian opetussuunnitelmaa uudistetaan syksystä 2022 alkaen. Tulevaisuudessa opiskelijat tulevat tekemään samankaltaisia projekteja hyödyntäen Raspberry Pi Picoa kehitysalustana, johon liitetään erilaisia bioantureita. Tätä raporttia kirjoittaessa mielessäni ovatkin nuo tulevat opiskelijat, ja sen sisältö on pyritty laatimaan sellaiseen muotoon, että asiaan perehtymätönkin ymmärtäisi sen. Työn teen Metropolia Ammattikorkeakoululle, jolta saan projektiin vaadittavat osat lainaan. Vastaavasti he saavat vastineeksi tästä työstä tuotetun materiaalin.

# 2 Anatomia ja fysiologia

Anatomia ja fysiologia -luku on rajattu työn luonteen mukaisesti. Tarkoituksena on luoda ymmärrys niistä vaikuttavista tekijöistä, jotka ovat prototyypin suunnittelu- ja kehitystyön kannalta relevantteja. Eli miten lihas tuottaa mitattavan sähköisen impulssin sekä miten käden liikkeet syntyvät.

## 2.1 Kalvojännite ja aktiopotentiaali

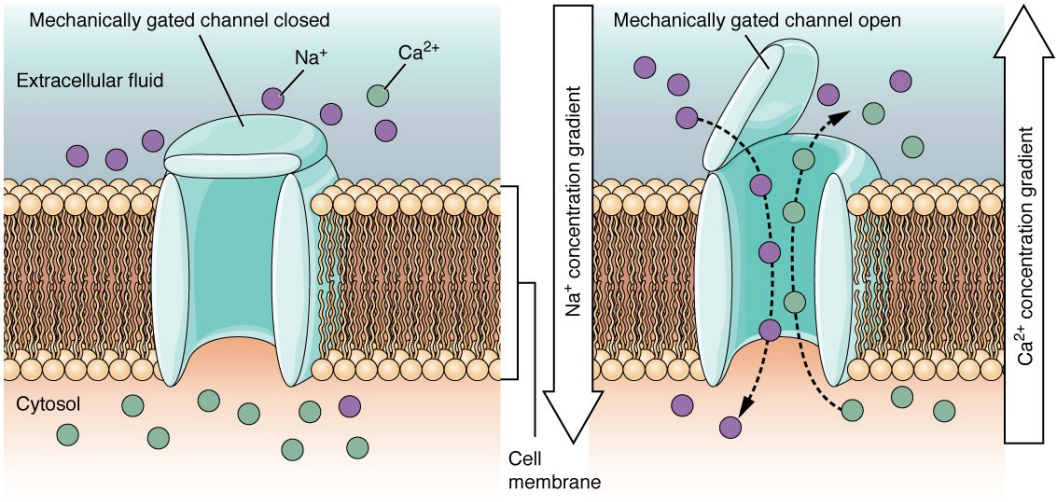
Elimistön solujen soluliman ja ulkoisen nesteen välillä vallitsee jännite-ero, jota kutsutaan kalvojännitteeksi (kuva 1). Jännite-erot hermo- ja lihassoluissa voivat olla lähes 0,1 volttia solukalvon ollessa vain vajaa 10 nm paksu. Senttimetrin paksuinen kalvo tuottaisi peräti 100 000 V/cm. Näillä vahvoilla sähköstaattisilla voimilla on oleellinen toiminto kuljettaa viestejä ja ravinteita, kun positiivisesti ja negatiivisesti varautuneet molekyylit pyrkivät soluun sisälle ja ulos. [1, s. 46.]


Kuva 1. Kalvojännite. Solun sisäinen varaus on negatiivinen ja ulkopuolisen nesteen varaus positiivinen.

Hermoimpulssit ja lihassupistukset perustuvat nopeisiin ja lyhytaikaisiin kalvojännitteen muutoksiin, joita kutsutaan aktiopotentiaaliksi. Kalvojännite muuttuu negatiivisesta varauksesta positiiviseksi 0,5 millisekunnissa. Aktiopotentiaalit välittävät viestejä hermostossa jopa yli 100 m/s nopeudella. Hermostoa pitkin lähtevää viestiä säädellään aktiopotentiaalien lukumäärillä ja niiden välisellä ajalla. [1, s. 49.]

Solun sisällä ja sitä ympäröivässä nesteessä on tyypillisesti kalium- ($K^+$), natrium- ($Na^+$) ja kalsiumioneita ($Ca^+$). Näiden väliset pitoisuudet vaihtelevat solun

ja nesteen välillä. Ionit pyrkivät suuremmasta pitoisuudesta pienempään pitoisuuteen. Solukalvolla on ioniportteja (kuva 2), jotka avautuvat ja sulkeutuvat kalvojännitteen muutoksista aiheutuvien sähköstaattisten voimien avulla. Näiden avulla solu säätelee ionien pitoisuuksia sisällään, sillä sen pitoisuuksien määrät ovat vakio.




Kuva 2. Solukalvolla oleva ioniportti, joka säätelee solun ionien pitoisuuksia. [2]

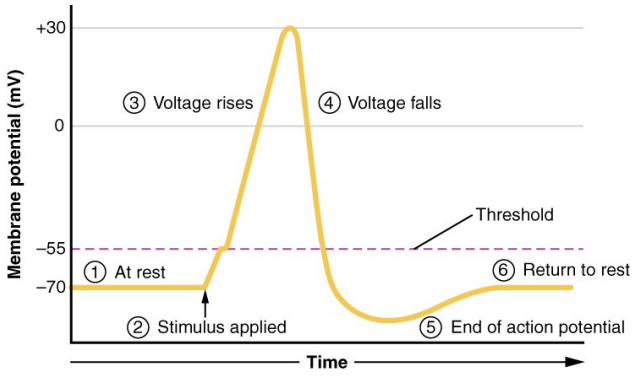
Usein solun sisäinen kaliumionipitoisuus on suurempi kuin sitä ympäröivässä nesteessä. Tyypillisesti solulta vuotaa passiivisesti $K^+$-ioneita solukalvon läpi sen ulkopuolella vallitsevaan pienempi pitoisempaan nesteeseen, sillä $K^+$-ionit läpäisevät solukalvon ilman aktiivisia ioniportteja. Tällöin solun varaus muuttuu negatiiviseksi, jota kutsutaan hyperpolarisaatioksi.

Ionien pitoisuuksien ollessa vakio, solun pitää korvata tämä passiivisesti tapahtuva ionivuoto. Ulkopuolisessa nesteessä puolestaan on suurempi määrä positiivisesti varautuneiden natrium- ja kalsiumionien pitoisuus kuin solussa, jolloin ne pyrkivät soluun sisälle. Solukalvolla olevat ioniportit estävät näiden pääsyä niin kauan, kunnes $K^+$-ioneita on vuotanut riittävästi ulos muuttaen samalla solun varausta. Silloin ioniportit aukeavat ja päästävät $Na^+$- ja $Ca^+$-ionit sisään. Solun varaus muuttuu positiiviseksi eli depolarisoituu, jolloin samalla syntyy aktiopotentiaali. Operaatio toistuu, koska nyt solulla on liikaa $Na^+$- ja $Ca^+$-

ioneita sen tasapainoon nähden, joten se pumppaa niitä ulos tuoden samalla sisään $K^+$-ioneita. Jälleen varaukset muuttuvat ja syntyy aktiopotentiaali. [1, s. 46–49.]

Aktiopotentiaalin (kuva 3) syntyyn vaaditaan, että tuleva hermoärsyke ylittää kalvojännitteen kynnysarvon, joka on noin -60 millivolttia. Solun levätessä sen lepojännite on noin -70 mV. Hermosolu stimuloi sähköllä solua joitakin millisekunteja ja muuttaa sen varausta positiivisemmaksi. Mikäli solun jännite ei ylitä kynnysarvoa, niin aktiopotentiaalia ei synny. Kynnysarvon ylittyessä syntyy aktiopotentiaali. Kalvojännite nousee jyrkästi ja nopeasti positiivisen puolelle, noin 0,5 millisekunnissa noin +20 millivolttiin.

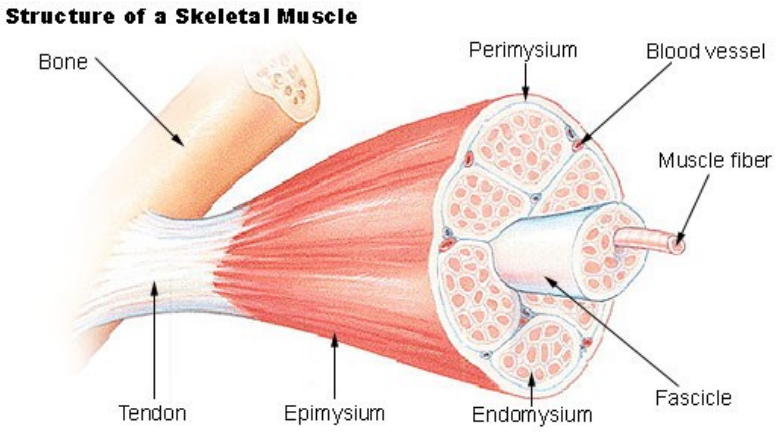
Tämän jälkeen solussa ei voi tapahtua uutta aktiopontetiaalia pariin millisekuntiin sen toipuessa, jota kutsutaan absoluuttiseksi refraktaariajaksi. Aktiopotentiaalin jälkeen solun varaus on joitakin millisekunteja lepoarvostaan alempana. Tässä tilassa solulta vuotaa $K^+$-ioneita helpommin muuttaen sen varausta enemmän negatiiviseksi, jota kutsutaan jälkihyperpolarisaatioksi. Tässä jälkihyperpolarisaatiotilassa solun varausta voidaan kyllä nostaa kynnysarvoonsa, mutta se vaatii voimakkaamman ärsykkeen. Tätä kutsutaan suhteelliseksi refraktaariajaksi. [1, s. 49–51]




Kuva 3. Aktiopotentiaali ja sen vaiheet. 1. Solu lepojännitteessä. 2. Hermosolu stimuloi solua, kunnes raja-arvo ylittyy. 3. Jännite nousee jyrkästi huippuunsa. 4. Jännite putoaa kohti lepoarvoa. 5. Jälkihyperpolarisaatiotila. 6. Jännite palautuu lepoarvoonsa. [3]

## 2.2 Luustolihaksista ja jänteistä

Yläraajan lihakset koostuvat luustolihaksista (kuva 4). Nämä lihakset ovat kiinnittyneet jänteiden avulla luuhun kiinni ja supistuessaan tuottavat raajan liikkeet. Yksittäinen lihas koostuu useasta lihassyykimpusta. Lihassyykimppu koostuu monista lihassyistä eli poikkijuovaisista lihassoluista. Lihassolut koostuvat useista pitkittäissuuntaisista myofibrileistä, jotka ovat lihaksen supistuva osa. Myofibrileissä on ketjussa useita lihaksen toiminnallisia yksiköitä eli sarkomeereja. Sarkomeeri koostuu aktiini- ja myosiinifilamenteista, jotka ovat lomittuneet toistensa lomaan. Näiden toiminnasta kerrottaan tarkemmin luvussa 2.3 Luustolihasten toiminta. [1, s. 189–191.]




Kuva 4. Luustolihaksen rakenne. [4]

Lihassyyn halkaisija vaihtelee 0,01–0,1 mm välillä ja pituutta sillä voi olla muutamasta senttimetristä jopa 30 senttimetriin riippuen lihaksesta. Lihassyitä on kahdenlaisia: nopeita ja hitaita. Jaottelu perustuu siihen, kuinka nopeasti myosiini pilkkoo adenosiinitrifosfaattia (ATP), joka on myosiinifilamentin toiminnan kannalta ainoa energianlähde.

Nopeat lihassyyt ovat tarkoitettu lyhytkestoiseen kovaan lihastyöhön. Ne kuluttavat enemmän energiaa hitaisiin verrattuna ja väsyvät nopeammin. Hitailla lihassyillä on hallitseva osuus kohtalaisessa lihastyössä, kuten kehon asennon ylläpidossa. Näiden nopeiden ja hitaiden lihassyiden määrät vaihtelevat riippuen

lihaksesta ja sille tarkoitetusta työstä. Yläraajoissa nopeat lihassyyt ovat enemmistö kädellä tehtävien asioiden takia kuten painonnosto ja tavaroiden heittäminen. [1, s. 189, s. 198.]

Jokaista yksittäistä lihassyytä, lihassyykimppua ja lihasta kokonaisuudessaan ympäröi sidekudoskalvot. Näiden kalvojen molemmista päistä lähtevät kollageenisäikeet, jotka ovat liitoksissa jänteiden kollageenisäikeisiin. Jänteillä lihas kiinnittyy luustoon luoden saumattoman ja vahvan liitoksen lihaksen ja luun välille.

Jänteiden pituudet vaihtelevat millimetreistä useaan kymmeneen senttimetriin. Esimerkiksi sormien jänteet ovat pitkiä, sillä niitä ohjaavia lihaksia löytyy myös käsivarresta. Lihasten sijaitseminen käsivarressa mahdollistaa ohuet ja ketterät sormet. Paikoitellen jänteitä verhoaa jännetuppi, joka voitelee jännettä vähentäen kitkaa. Jännetuppea ympäröi nivelside, joka pitää pitkiä jänteitä lähellä luustoa mahdollistaen voimansiirrot luiden ja nivelien muodostamassa vipuvarsijärjestelmässä. [1, s. 170, s. 172–173, s.175, s. 200–201.]

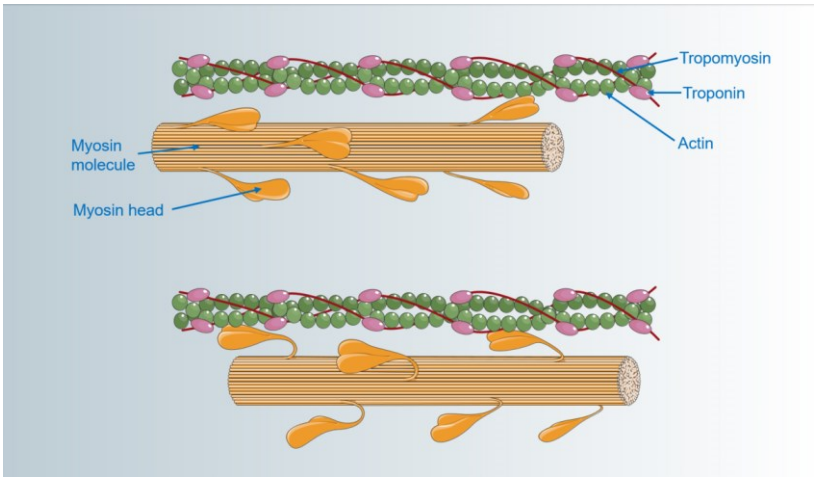
## 2.3 Luustolihasten toiminta

Lihakset tuottavat voimaa ainoastaan supistumalla. Yksittäinen lihas ei voi itsestään venyä vaan lihakset toimivat aina pareittain; toinen supistuu, niin toinen venyy. Näitä lihaspareja on joka puolella kehoa. [1, s. 199.]

Lihaksen supistumista on kahdenlaista, isotoninen ja isometrinen lihassupistus. Isotonisessa supistuksessa lihaksen pituus muuttuu, kuten raajaa liikuteltaessa. Isometrisessä supistuksessa lihas tuottaa voimaa, muttei muuta sen pituutta kuten painavaa esinettä nostaessa. Kuitenkin aina ennen isotonista supistusta edeltää isometrinen supistus. [1, s. 194–195.]

Luustolihaksen supistumisen mekanismina toimii lihassyissä olevien sarkomeerien lyhentyminen. Sarkomeerin sisällä olevat myosiinifilamentit tarttuvat väkäsillään kiinni aktiinifilamentteihin (kuva 5), jonka jälkeen myosiinifilamentti

taipuu taaksepäin. Se liukuu syvemmälle aktiinifilamentteihin ja vetää samalla viereistä sarkomeeria lähemmäksi itseään, jolloin lihas lyhenee tai sen jännitystila säilyy. Tämän ”tarraa ja koukistu” -mekanismin saa aikaan, kun ATP-molekyyli sitoutuu myosiinin kanssa ja pilkkoutuu adenosiinidifosfaatiksi (ADP) ja fosfaatiksi. Tästä vapautuva energia varastoituu myosiiniväkäseen ja se tarttuu aktiiniin. Prosessissa syntyy samalla lämpöä. [1, s. 191–193.]




Kuva 5. Myosiini- ja aktiinifilamentti. [5]

Myosiini voi tarttua aktiiniin vain, jos sarkomeerin kalsiumpitoisuus on tarpeeksi koholla. Kalsiumia on varastoituneena sarkoplasmakalvolla, joka peittää myofibrillejä. Keskushermostolta saatu hermoimpulssi luo lihassoluun kiinnittyneessä hermosolussa ärsykkeen, joka synnyttää lihaskalvolle aktiopotentiaalin. Aktiopotentiaali kulkee syvälle lihassoluihin, jolloin sarkoplasmakalvoilta vapautuu kalsiumioneita. Sarkomeerien ympäristön kalsiumpitoisuus alkaa kohota, mahdollistaen myosiinifilamenttien toiminnan. Myosiinifilamentit tarttuvat aktiinifilamentteihin niin kauan, kunnes sillä on tarpeeksi ATP:lta saatua energiaa ja kalsiumpitoisuus pysyy koholla.

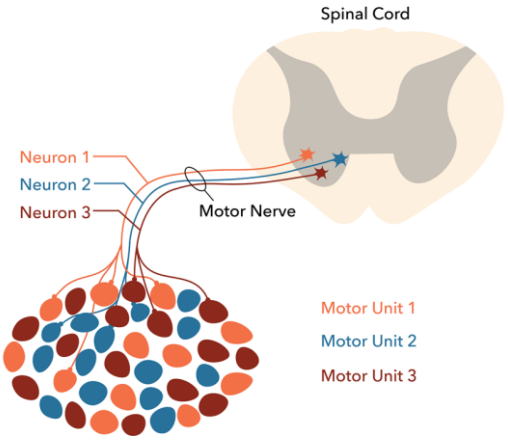
Aktiopotentiaalin tapahtuessa on pieni viive itse lihassupistukseen, kun sähkö etenee lihaksen syvimpiin osiin ja kalsiumpitoisuudella kestää hetki kohota. Viiveen kesto riippuu lihassyytyypistä. Nopeassa lihassyyssä supistusvoima on suurimmillaan 10 ms:n kuluttua ja hitaassa yli 100 ms:n kuluttua.

Aktiopotentiaalin päättyessä sarkoplasmakalvolta ei vapaudu lisää kalsiumioneita. Pitoisuus alkaa laskea eikä myosiini voi olla enää tarttuneena aktiiniin. Väkäset irtoavat ja suoristuvat, jolloin lihas veltostuu, muttei veny itsestään. [1, s. 191–194.]

Lihaksen pysyessä supistuneena pidempiä aikoja johtuu aktiopotentiaalien sarjatulesta. Yksittäinen lihassupistuksen kesto on huomattavasti pidempi kuin aktiopotentiaalin kesto (1–2 ms). Uusi aktiopotentiaali voi syntyä jo ennen kuin lihas veltostuu, mahdollistaen voimakkaamman yhtäjaksoisen supistuksen. Ilmiötä kutsutaan tetaniseksi lihassupistukseksi eli tetanisaatioksi.

Kalsiumioneilla kestää aina hetki aktiopotentiaalin jälkeen palautua takaisin sarkoplasmakalvolle, jolloin uusi impulssi pumppaa ioneita lisää sarkomeerille kohottaen sen pitoisuutta edelleen. Tällöin myosiinifilamentit lomittuvat yhä syvemmälle aktiinifilamentteihin, ja lihasvoima suurenee jatkuvassa ärsytyksessä. [1, s. 195–196.]

Lihasvoimaa säätelevät eri motoristisen yksiköiden aktivoituminen riippuen tehtävästä työstä (kuva 6). Lihaksessa on kiinnittyneenä useita keskushermostolta tulevia motoneuroneja eli liikehermosoluja. Yksi motoneuroni voi olla yhteydessä muutamasta lihassyystä useaan tuhanteen. Kuitenkin yhdellä lihassyyllä voi olla ainoastaan yksi hermo-lihasliitos. Yksi motoneuroni ja siihen kiinnittyneet lihassyyt muodostavat yhden motorisen yksikön. [1, s. 194.]




Kuva 6. Motoneuronien muodostamia motorisia yksiköitä. [6]

Pienissä liikkeissä, jotka vaativat vähän voimaa, aktivoituvat kaikkein pienimmät yksiköt. Mikäli voimaa tarvitaan enemmän, aktivoituvat isot yksiköt tuottaen isomman supistusvoiman. Nopeissa liikkeissä isot yksiköt voivat aktivoitua, vaikka pienet eivät aktivoituisi.

Supistusvoima on riippuvainen myosiini- ja aktiinifilamenttien päällekkäisyydestä. Niillä on optimipituus, missä voima on suurimmillaan, kun kaikki myosiiniväkäset ovat tarttuneena aktiiniin. Venyneessä sarkomeerissa filamenttien päällekkäisyys on vain osittaista, jolloin osa väkäsistä ovat kiinnittyneenä. Vastaavasti liikaa supistuessaan alkavat vastakkaiset aktiinifilamentit lomittua toistensa päälle, mikä hankaloittaa myosiiniväkästen tarttumista. [1, s. 196.]

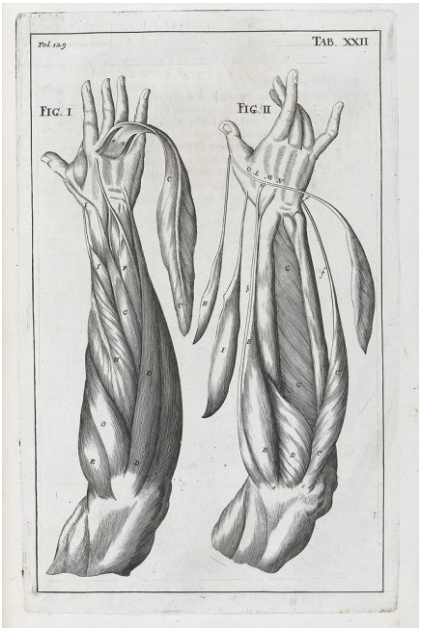
## 2.4 Kyynärvarren ja käden lihakset

Kyynärvarressa on yli 20 lihasta monenlaiseen tehtävään (kuvat 7 ja 8). Osa lihaksista osallistuu kyynärnivelen koukistukseen. Suurin osa niistä osallistuu ranteen ja sormien liikkeisiin. Värttinäluun ja kyynärluun välissä kulkee lihaksia, joiden tehtävänä on kyynärvarren ulko- ja sisäkierto.

Useiden rannetta liikuttavien lihasten pitkät jänteet ulottuvat sormiin asti, jolloin ne toimivat myös sormien pitkinä koukistaja- ja ojentajalihaksina. Kämmenpuolen lihakset koukistavat rannetta ja sormia, ja kämmenselän puoleiset puolestaan ojentavat niitä. Sormia voidaan koukistaa ilman, että ranne koukistuu, koska kämmenselkään menevien jänteiden lihakset supistuvat samaan aikaan, jolloin ne estävät ranteen koukistuksen.

Jänteet kulkevat ajoittain jännetuppien sisällä, joiden tehtävänä on voidella ja ohjata jännettä mahdollistaen sulavat liikkeet. Kyynärvarren lihasten sidekudoskalvo on kehittynyt erityisen vahvaksi nivelsiteeksi ranteessa, joka pitää jänteet paikallaan mahdollistaen voimansiirrot vipuvarsijärjestelmässä. [1, s. 212–213.]


Kuva 7. Kämmenpuolen lihaksia. [7]




Kuva 8. Kämmenselän puoleisia lihaksia. [8]

Kämmenessä on useita pieniä lihaksia, joiden tehtävänä ovat täsmällisen tarkat ja pienet liikkeet. Kämmenluiden väliset pienet lihakset levittävät sormia haaralle ja tuovat niitä yhteen. Peukalon päkiälihas ja pikkurillin vastapäkiälihas koostuvat useammasta lihaksesta, jotka liikuttavat niitä. Näiden lihasten jänteet yhdistyvät lähellä rannetta, jolloin ne muodostavat ranteen koukistajien pidäkesiteen. [1, s. 213.]

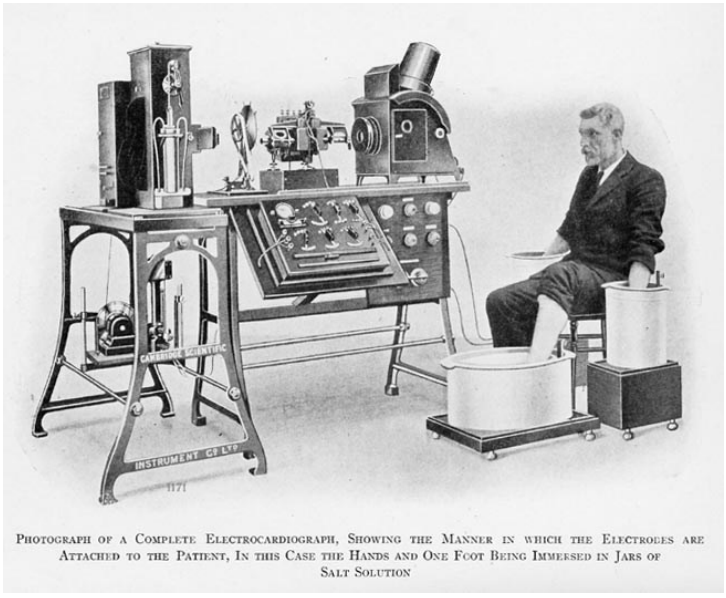
# 3 Elektromyografia

Elektromyografialla (EMG) tarkoitetaan lihaksen sähköisten ilmiöiden mittaamista. Termi elektromyografia juontuu kolmesta sanasta: elektro, jolla tarkoitetaan sähköistä aktiivisuutta, myo on kreikan kielestä johdettu sana, jolla viitataan lihakseen ja grafialla tarkoitetaan tiedon tallentamista. Ensimmäinen tallennettu EMG-mittaus on vuodelta 1907, jonka suoritti Hans Piper. [9, s. 217.]

Kokeessaan hän huomasi eroavaisuuksia signaalin rytmissä tahdonalaisessa lihassupistuksessa. Mittalaitteena hänellä oli galvanometri. Laitteeseen syötettiin elektrodien avulla heikko lihassähköimpulssi, jossa se vahvistettiin.

Laitteessa oleva herkkä lanka alkoi värähtelemään jännitteen muuttuessa. Lampun avulla heijastettiin langan liikkeet kamerafilmille, jolla tieto voitiin tallentaa. Kamerafilmille piirtyi luettavissa oleva lihassähkökäyrä. [10.]

Havainnollistamiseksi vanhasta teknologiasta on laitettu kuva (9) vuonna 1911 Cambridge Scientific Instruments -yhtiön kaupalliseen käyttöön julkaisemasta sydänsähkökäyrämittarista, jonka mittausteknologia perustuu galvanometriin.



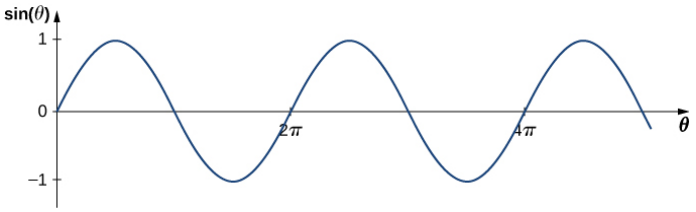

Kuva 9. Sydänsähkökäyrämittari vuodelta 1911, jonka mittausteknologia perustuu galvanometriin. [11]

Noista päivistä lähtien elektromyografia on ollut tärkeässä roolissa monissa lihaksia ja hermostoa rappauttavien sairauksien tutkimuksissa. EMG-signaalin avulla voidaan tutkia lihaksen motoristen yksiköiden aktivoitumista, voimaa ja väsymistä. Tätä tietoa voidaan hyödyntää diagnosoinnissa ja kuntoutuksessa.

Apuvälinetekniikassa EMG-signaaleilla voidaan ohjata sähköpyörätuoleja, kehoa tukevia ulkoisia tukirankoja ja proteesiraajoja. Suun lihaksia mittaamalla voidaan tulkata henkilön puhetta, kun ääntä ei synny. Käden liikkeistä pystytään tunnistamaan kirjoittamista sekä liikkeitä voidaan mallintaa virtuaalimaailmassa. [9, s. 226.] [12.]

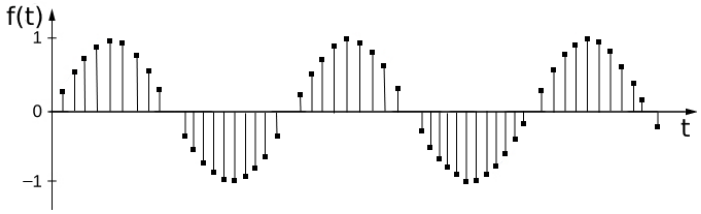
## 3.1 EMG-signaalinkäsittely

Luonteeltaan mitattava lihassähköimpulssi on monen jatkuvan siniaallon summa (analoginen signaali). Teoriassa siniaaltojen summa voi sisältää ääretömän määrän eri taajuuksia. Sillä ei ole vain yhtä diskreettiä amplitudiarvoa ajassa, vaan arvot voivat olla mitä tahansa reaalilukuja. Siniaalto jatkuu ajassa äärettömyydestä äärettömyyteen, mutta tyypillisesti sen amplitudi on äärellinen ja sillä on tietty värähtelyn jakso (hertsi, Hz).




Kuva 10. Jatkuva siniaalto [13]

Haasteena on tuoda analoginen signaali digitaaliseen ympäristöön, joka operoi diskreetisti äärellisellä joukolla arvoja. Vasta ottaessa näyte tietyllä ajan hetkellä saadaan analogisesta signaalista diskreetti signaali (kuva 11), jossa X-akselin (aika) arvo on diskreetti, mutta Y-akselin (signaalin amplitudi) arvo on edelleen jatkuva. Diskreetti signaali muutetaan digitaaliseksi signaaliksi, jossa ajan lisäksi amplitudin arvo muuttuu diskreetiksi. [9, s. 3–8, s. 15–16, s. 27–28.]




Kuva 11. Diskreetti signaali, jossa näytteitä on otettu tietyllä ajan hetkellä. Y-akselilla signaalin amplitudi ja X-akselilla aika.

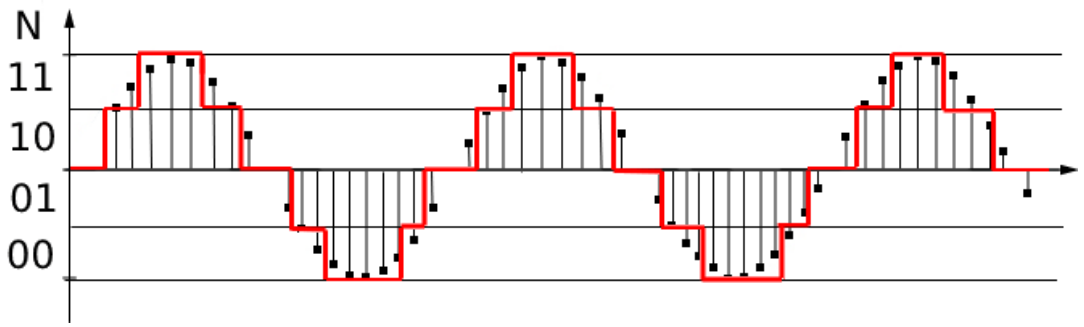
Tähän väliin tarvitaan analogia-digitaalimuunnin (A/D-muunnin), joka havaitsee signaalissa tapahtuvat jännitemuunnokset. A/D-muuntimen resoluutio ja näytteenottotaajuus määrittää, kuinka herkästi se havaitsee signaalin jännitteessä

tapahtuvat muutokset. Resoluution määrittävät A/D-muuntimessa käytetyt bittien määrät. Mitä suurempi on käytetty bittien määrä, niin sen tarkemmin luotu digitaalinen signaali vastaa alkuperäistä mitattua jatkuvaa siniaaltoa.

Lisäksi Nyquistin teoreemaan mukaan näytteenottotaajuuden (näytteitä per sekunti) tulee olla vähintään kaksinkertainen kuin mitattavan analogisen signaalin korkein taajuus ($2f_{max}$), jotta voidaan luoda alkuperäistä signaalia vastaava digitaalinen signaali. [9, s. 26–27.] [14.]

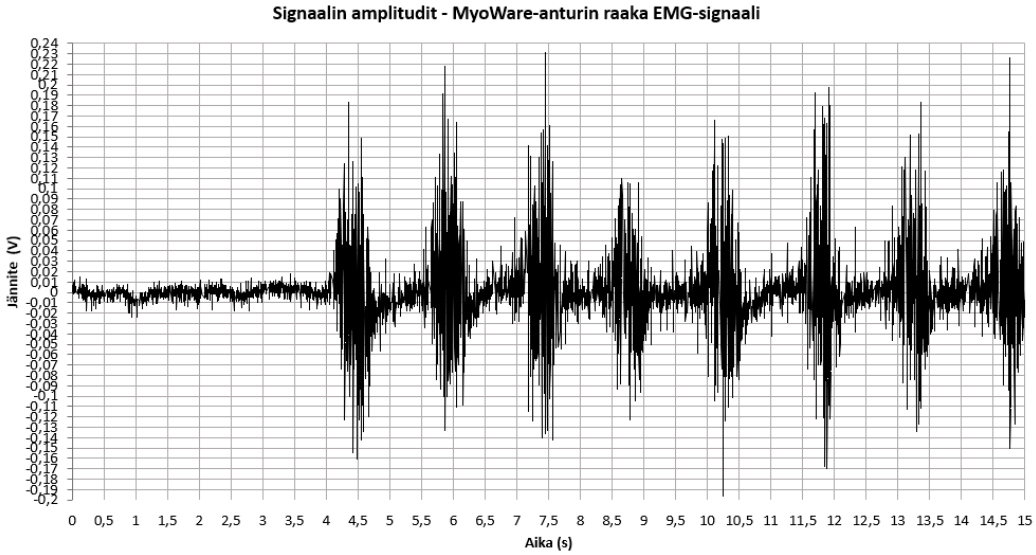
A/D-muuntimen ollessa 2-bittinen sen resoluutio on neljä ($2^2$). Se tasoittaa tulevan diskreetin signaalin neljälle tasolle kuvan 12 mukaisesti. Jokainen taso vastaa A/D-muuntimen jänniteherkkyyttä ja tasoille on digitaalinen arvo binaarimuodossa. Mikäli A/D-muuntimessa käytetty jännitereferenssi ($V_{ref}$) on 3,3 volttia, saadaan herkkyys jakamalla $V_{ref}$ bittien määrällä ($V_{ref}$ / $2^{bitti}$). Näin ollen 2-bittisen muuntajan herkkyys on 0,825 V (3,3 V / $2^2$). Siispä se pystyy havaitsemaan jännitemuutokset 0,825 voltin välein jännitereferenssin ollessa 3,3 V. Vastaavasti 12-bittisen resoluutio on 4096 ja se havaitsee $V_{ref}$:stä noin 0,8 mV:n muutokset (3,3 V / $2^{12}$).

A/D-muunnin muuntaa diskreetin signaalin jännitteen lähimpään sitä vastaavaan digitaaliseen arvoon. Mikäli todellinen mitattu jännite olisi 3 V, niin arvo pyöristyy 3,3 volttiin, jolloin kvantisointivirhe on 0,3 volttia. Kaksibittinen A/D-muunnin antaa silloin digitaalisen arvon binaarimuodossa 11 (lue yksi yksi), joka vastaa arvoa kolme. [9, s. 6–7.] [14.]




Kuva 12. Diskreetistä signaalista (mustat pisteet) luotu digitaalinen signaali (punainen viiva), jonka A/D-muunnin on 2-bittinen.

Todellisuudessa mitattavan lihasaktivaation oleellinen tieto on 20–500 Hz:n taajuusalueella, josta saadaan EMG-signaalin ominainen aallonmuoto (kuva 13). Taajuusalueen alemmat ja ylemmät taajuudet voidaan suodattaa pois. Samalla päästään eroon liikeartefaktien tuottamasta kohinasta, joiden taajuusalue on 0–20 Hz. Liikeartefaktien synty johtuu elektrodien, kaapeleiden ja lihasten tahattomista liikkeistä, jotka ovat EMG-signaalin merkittävin kohinan aiheuttaja. [9, s. 230–231.]




Kuva 13. Esimerkki EMG-signaalin aallonmuodosta. Y-akselilla signaalin amplitudit (V), joiden voimakkuudet vaihtelevat positiivisten ja negatiivisten jännitteiden välillä. X-akselilla kulkee aika (s).

Kokonaisessa aallonmuodossa lihasaktivaation aikana amplitudien arvot vaihtelevat positiivisten ja negatiivisten jännitteiden välillä. Signaalin tehollisarvon analysoimiseksi eräs yleisesti keino on neliöllisen keskiarvon laskeminen (RMS) (kaava 1), joka samalla kuvastaa lihasaktivaation fyysisiä ominaisuuksia kuten voimaa.

$$RMS = \sqrt{\frac{x_1^2 + x_2^2 + x_3^2 \cdots x_N^2}{N}} \quad (1)$$

x on amplitudin arvo

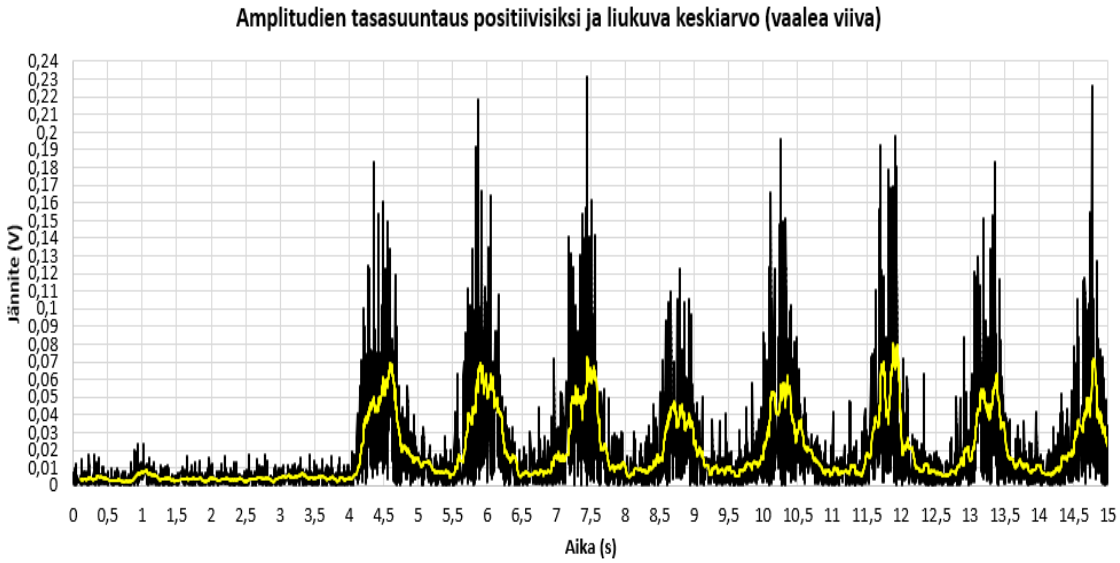
N on näytteiden lukumäärä

Toinen signaalin analysoimiseen käytettävä kaava on tasasuunnatun signaalin liukuva keskiarvo (AVR) (kaava 2). Tasasuunnatussa signaalissa negatiiviset jännitteet muutetaan positiiviksi. Digitaalisessa signaalissa amplitudeista lasketaan itseisarvot, jolloin negatiiviset arvot muuttuvat positiiviksi (kuva 14). AVR kuvastaa signaalin voimakkuutta, eikä sillä ole erityistä merkitystä lihaksen fyysisten ominaisuuksien tutkimiseen. [9. s. 231–232.] [15.]

$$AVR = \frac{|x_1|+|x_2|+|x_3|\cdots|x_N|}{N} \quad (2)$$

|x| on tasasuunnatun amplitudin arvo

N on näytteiden lukumäärä




Kuva 14. Amplitudit suunnattuna positiivisiksi ja näistä laskettu liukuva keskiarvo (keltainen viiva).

## 3.2 EMG-signaaliin vaikuttavia tekijöitä

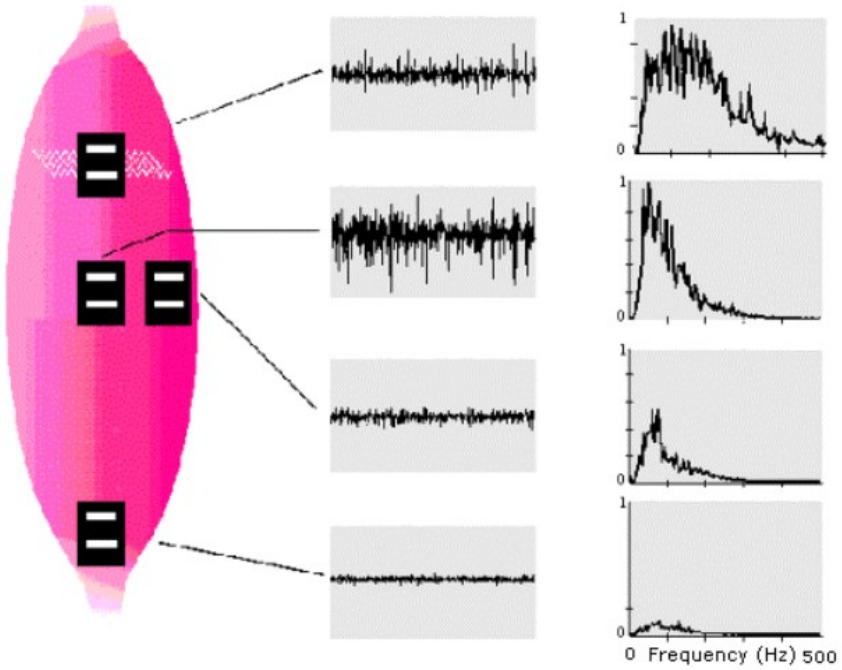
Lähtökohtaisesti EMG-signaali on hyvin häiriöistä. Siihen vaikuttavat monet tekijät: elektrodien sijainti, liikeartefaktit, verkkovirrassa ja piirin sisällä tapahtuva kohina sekä langattomien verkkojen aiheuttama kohina. Myös sydämen ja mitattavan lihaksen viereisten lihaksien sähköimpulsseilla voi olla vaikutusta signaaliin.

Nämä signaaliin vaikuttavat tekijät voidaan jaotella kahteen ryhmään, ulkopuolelta tuleviin muuttujiin ja luonnollisiin sisäisiin muuttujiin. Ulkopuoliset muuttujat liittyvät ihon ulkopuolelle tulevan elektrodin konfigurointiin, johon voidaan vaikuttaa. Elektrodien pinta-alalla ja muodolla on väliä siihen, kuinka monta lihassyytä osuu mittausalueelle.

Kuvassa 15 on havainnollistettu EMG-signaalin amplitudia ja aallonmuotoa elektrodin sijoitettuna eri kohtiin lihaksessa. Jotta saadaan vahva ja selkeä signaali tulee elektrodit sijoittaa oikeaan kohtaan lihaksessa, suurin piirtein keskelle lihaksen paksuimpaan kohtaan ja lihassyiden suuntaisesti.

Liian lähelle jänteitä sijoitettu elektrodi johtaa signaalin vaimenemiseen. Lihaksen reunaan sijoitetulla elektrodilla on puolestaan riskinä mitata ei-haluttuja signaaleita viereisistä lihaksista. Myös elektrodien etäisyydellä toisistaan on merkitystä, sillä liian lähekkäin voi ihon kostuessa jännite hypätä elektrodista toiseen tiputtaen jännitettä merkittävästi.

Kuvan ylin elektrodi kuvastaa kohtaa, jossa on hermo-lihasliitos. Keskimmäinen elektrodi on optimissa kohtaa ja käyrässä näkyy selkeä huippu lihaksen aktivoitumisesta. Keskelle lihaksen reunaan asetetun elektrodin mittausalueella on vähemmän lihassyitä, joten signaali heikkenee huomattavasti. Alin elektrodi on kiinnitetty kohtaan, josta alkaa jänne, ja signaali on lähes kokonaan kadotettu. [15.]




Kuva 15. Elektrodien kiinnityskohdat lihaksessa ja niiden vaikuttavuus EMG-signaalin aallonmuotoon ja amplitudiin. Keskellä lihasta paksuimmassa kohtaa on optimisijainti. [15.]

Luonnolliset sisäiset muuttujat ovat mitattavan henkilön anatomisia ja fysiologisia, joihin ei pystytä vaikuttamaan. Signaalin amplitudiin ja aallonmuotoon vaikuttavat aktivoituvien motoristen yksiköiden määrät ja kuinka syvällä lihaksessa ne ovat suhteessa pintaelektrodiin. Joillain henkilöillä aktivoituvat lihassyyt voivat olla suoraan pintaelektrodin alapuolella, kun toisilla ne ovatkin syvemmällä lihaksessa. Tuloksena voi olla, että mitattu fyysinen voima pysyy samana, mutta signaalin voimakkuuksissa on eroja. Lihassyiden halkaisijalla on vaikutusta sähkönjohtavuuteen ja sen myötä mitattuun amplitudiin.

Nopeiden ja hitaiden lihassyiden suhde lihaksessa vaikuttaa aallonmuotoon, joiden määrät ovat yksilöllisiä. Nopeiden lihassyiden aktivoituminen näkyy aikajanalla pienempänä viiveenä, jolloin graafiin muodostuva huippu on enemmän vasemmalle vino. Vastaavasti hitailla lihassyillä syntyy viivettä enemmän, ja graafin huippu on enemmän oikealle vino sekä usein loivempi, kun sitä tarkastellaan aikajanalla.

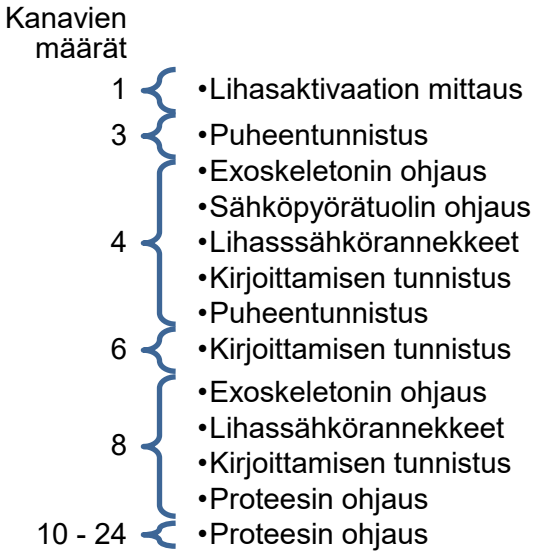
Näillä nopeiden ja hitaiden lihassyillä on myös vaikutusta lihaksen pH-arvon muutoksiin. Nopeat lihassyyt pilkkovat ATP-molekyylejä hitaita nopeammin, mutta myös väsyvät nopeammin. Samalla muodostuu laktaattia, mikä heikentää sähkönjohtavuutta. Verenkierrolla on vaikutusta, miten nopeasti aineenvaihdunta toimii lihaksessa ja siten lihaksen toimintaan ja EMG-signaalin muutoksiin. [15.]

## 3.3 Elektromyografian teknisiä vaatimuksia

EMG-signaalia voidaan hyödyntää lukuisissa erilaisissa käyttötarkoituksissa. Tuore kirjallisuuskatsaus [12] vuodelta 2020 käy läpi 30 erilaista projektia liittyen elektromyografiaan. Näistä projekteista tutkimuksen tekijät koostivat toteutuksien teknisiä ominaisuuksia.

Projekteina olivat muun muassa lihasten patologisten poikkeamien tutkiminen, puheen ja kirjoituksen tunnistaminen sekä kehon ulkoisten tukirankojen, sähköpyörätuolien ja proteesien ohjaaminen. Käyttötarkoituksen mukaan se asettaa vaatimuksia laitteistolle, kuten A/D-muuntimen tarkkuudelle ja käytettyjen elektrodien määrälle.

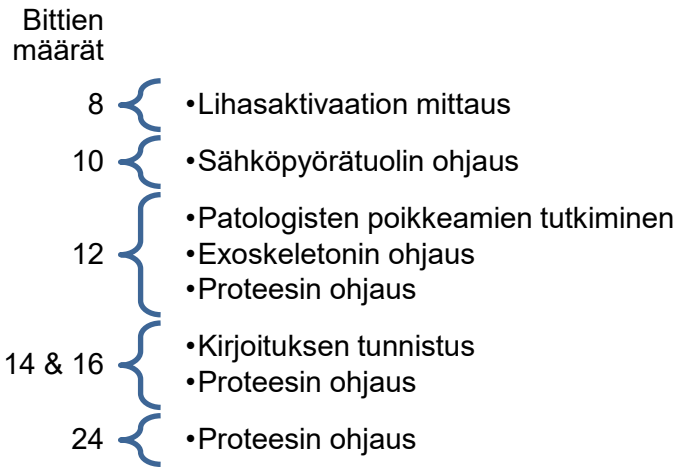
Mitattavien lihasten määrällä on vaikutusta tarvittavien kanavien määrään. Kanavia voi olla käytössä yhdestä useaan kymmeneen, kun mukana on antureita, jotka tuottavat tietoa ympäristöstä. Muun muassa neljällä kanavalla voidaan mitata yksinkertaisia käden liikkeitä, joita voidaan hyödyntää muun muassa sähköpyörätuolia ohjatessa. Raajan patologisia poikkeamia etsiessä jopa 10 kanavaa voi olla tarpeellista, kun mukana on usean signaalin hahmontunnistusta. Proteesien ohjauksessa on käytetty 8–24 kanavaan riippuen, kuinka tarkasti sitä ohjataan. Kuvassa 16 on listattu kanavien määrät projektien käyttötarkoitusten osalta. [12.]




Kuva 16. Käytettävien kanavien määrät käyttötarkoituksen mukaan. [12]

Mitattavien signaalien taajuusalueet vaihtelivat 2–750 Hz:n välillä. Tyypillisimmät mitatut alataajuudet olivat 10–20 Hz ylätaajuuden ollessa 500 Hz. Riippuen käyttötarkoituksesta A/D-muuntimen ominaisuudet vaihtelivat eri projektien välillä. Tyypillisimmät käytetyt näytteenottotaajuudet olivat 1000–1500 näytettä sekunnissa.

Projekteissa oli käytetty 8-, 10-, 12-, 14-, 16- ja 24-bittistä A/D-muunninta, joista yleisimmät olivat 12-, 14- ja 16-bittinen A/D-muunnin. Yksinkertaisissa lihasaktivaatiota mittaavissa sovelluksissa pienempi bittimäärä on riittävä, kun monimutkaisemmassa proteesin ohjaamisessa suurempi bittimäärä on tarpeen, jotta voidaan havaita raajan hienovaraiset liikkeet. Kuvassa 17 on kategorisoitu projektien käyttötarkoitukset niissä käytettyjen A/D-muuntimien resoluution mukaan. [12.]




Kuva 17. A/D-muuntimessa käytetty bittien määrä käyttötarkoitusten mukaan. [12]

# 4 Prototyyppi

Prototyypin tarkoitus on käsiproteesin ohjaaminen. Kehitystyö keskittyy mikro-ohjaimen logiikan tuottamiseen. Ideana on mitata käsivarresta lihaksen tuottama EMG-signaali, kun kättä koukistetaan ja ojennetaan ranneniveleestä. Signaalilla on tarkoitus saada servomoottorin pyörimissuunta muuttumaan koukistus- ja ojennusliikkeiden mukaisesti. Oikean käden ranteen koukistuksessa servomoottori pyörisi vasemmalle ja ojennuksessa oikealle.

Tämän prototyypin ideana on luoda perustat ihmisen ja piirin rajapinnasta, jota voi lähteä jatkokehittämään lisäämällä antureita ja sen myötä tutkia signaaleista kaavamaisuutta eri käden liikkeistä enemmän.

## 4.1 Prototyyppilaitteen määrittely

Prototyypin määrittelyvaatimuksena on, että laitteen tulisi toimia itsenäisesti sen käyttötarkoituksen mukaan eli proteesiraajaan asennettava piiri, joka kulkee ihmisen mukana. Piirin tulee suoriutua toiminnoista ilman, että siihen on kytkettynä erillistä tietokonetta. Kytkentä lihakseen tehdään pintaelektrodilla sen ei-invasiivisen luonteen takia.

Alun perin ideana oli käyttää yhtä elektrodia yhdessä lihaksessa ja sillä ohjata yhden servomoottorin suuntausta rannetta ojentaessa ja koukistaessa. Anatomiasta opittiin se, että lihakset toimivat pareittain, joten toteutusta varten tarvitaan kaksi lihasanturia. Ranteen koukistus- ja ojennuslihakset osallistuvat nyrkin puristukseen, joten syntyi idea lisätä myös toinen servomoottori, joka reagoi nyrkkiä puristaessa.

Ennakkotietona työhön lähtiessä oli, että logiikka tullaan tekemään Python-ohjelmointikielellä sen yleisen käytön takia. Kehitysalustana oli alun perin pohdinnassa Raspberry Pi tai Arduino, kunhan piiri tukee Pythonia. Valinnan ratkaisi koulun puolesta lainaan saadut komponentit, joten prototyypin kehitysalustana toimii Raspberry Pi Pico -mikro-ohjain (Pico). Picoa ohjelmoidaan MicroPythonilla. Se on kevyempi versio Pythonista, joka on tarkoitettu mikro-ohjaimille, joissa laskentatehot ja muistien määrät ovat rajalliset.

## 4.2 Komponentit

### 4.2.1 Raspberry Pi Pico

Prototyypin koodia ajetaan Raspberry Pi Pico -mikro-ohjaimella. Picon mikro-ohjaimena toimiva RP2040-piiri on Raspberry Foundationin ensimmäinen julkaistu mikro-ohjain vuodelta 2021. Piiri on ohjelmoitavissa MicroPythonin lisäksi myös C/C++ ja assembly -ohjelmointikielillä. Myyntiesitteissä he mainitsevat korkean suorituskyvyn ja helposti lähestyttävän piirin pieneen hintaan.

Suorituskyvyn osalta Pico tuntuisi sitä olevan. Noin 5 euroa maksavasta kehitysalustasta löytyy kaksiytiminen ARM Cortex-M0+ -prosessori, jonka kellotaajuus voidaan säätää 133 megahertsiin asti, 264 kilotavun välimuisti, 4 megatavun sisäinen muisti, joka on laajennettavissa 16 MB:n ulkoisella Flash-muistilla sekä 30 ohjelmoitavaa pinniä, joita voidaan käyttää signaalin vastaanottajana tai lähettäjänä. Niiden avulla Picoon voidaan liittää useita lisälaitteitta.

Näistä 30 pinnistä tähän projektiin oleellisimmat ovat pulssinleveysmodulaatioon (PWM) kykenevät pinnit, joita on 16 kappaletta. PWM-pinneillä pystytään ohjaamaan servomoottoreita.

Toinen oleellinen piiriltä löytyvä porttityyppi on A/D-muuntimeen kytketyt pinnit, joita on 4 kappaletta. Näistä 3 kanavaa on vapaasti käytettävissä analogisen signaalin mittaamiseen. Neljännellä kanavalla voidaan muuttaa A/D-muuntimen jännitereferenssiä ulkoisella jännitelähteellä tarvittaessa. Analogiaporttien kautta syötetään piirille lihasanturien mittauslukemat.

RP2040-piirillä on yksi 12-bittinen A/D-muunnin. Se pystyy muuntamaan tulevan signaalin 4096:lle ($2^{12}$) tasolle. Piirin sisäinen $V_{ref}$ on oletuksena 3,3 V. Tällöin pienin havaittavissa oleva jännitemuutos on noin 0,8 mV (3,3 V / $2^{12}$).

A/D-muunnin pystyy ottamaan 500 000 näytettä sekunnissa sen kellotaajuuden ollessa 48 MHz. Yhteen näytteenottoon sillä kestää 96 kellosykliä, jolloin 96 * (1 / 48*$10^6$) Hz = 2 mikrosekuntia per näyte.

Useaa kanavaa käyttäessä A/D-muunnin käy kanavat läpi vuorotellen. Se ottaa ensiksi yhdestä kanavasta näytteen ja muuntaa arvon digitaaliseen muotoon. Sen jälkeen vaihdetaan toiseen kanavaan. [16.]

Tärkeää on huomioida RP2040-piiriä ohjelmoitaessa MicroPythonilla, että koodissa 12-bittinen bittiset arvot on muutettu 16-bittiseen muotoon tarkoittaen digitaalisen signaalin arvojoukon koostuvan 0–65 535 ($2^{16}$) väliltä. Tätä kehittäjät perustulevat sillä, että tällä tavalla RP2040-piiri käyttäytyy samalla tavalla kuin muiden MicroPythonia tukevien mikrokontrollerien A/D-muuntimet. [17.]

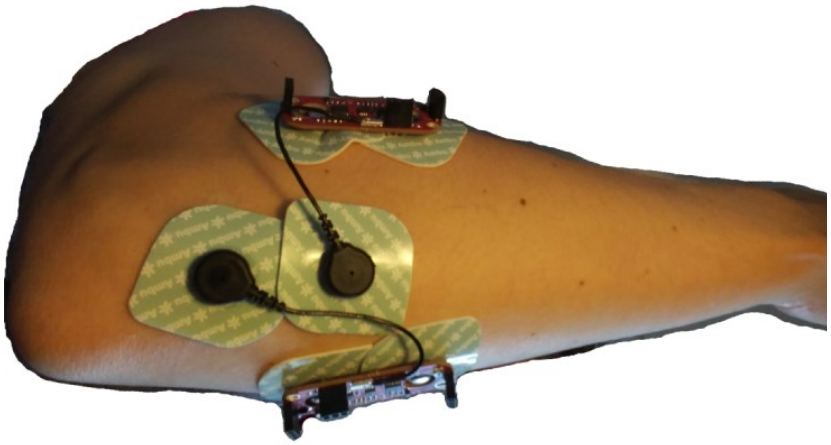
Näillä tiedoilla Pico sopii prototyypin kehitysalustaksi. Lihasantureita on käytössä kaksi kappaletta, joten A/D-muuntimen kanavien määrät ovat tässä vaiheessa riittävät. PWM-pinnit riittävät myös, kun servomoottoreita tulee kaksi.

A/D-muuntimen noin 0,8 mV:n herkkyys ja 500 000:n näytteenottotaajuuden mahdollisuus ovat riittäviä prototyypin kannalta. Lisäksi näytteenottotaajuus

täyttää Nyquistin teoreeman vaatimukset, jossa näytteenottotaajuuden tulee olla vähintään kaksinkertainen kuin signaalin korkein taajuus, joka EMG-signaalissa on tyypillisesti 500 Hz.

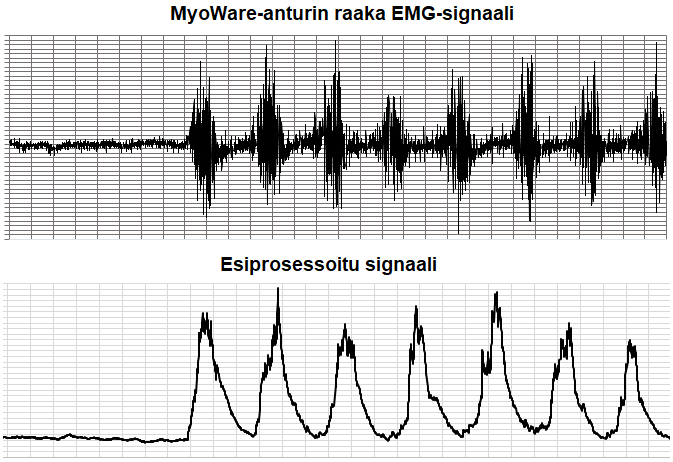
### 4.2.2 Advancer Technologies MyoWare EMG -anturi

Prototyypin EMG-sensoreina toimivat Advancer Technologiesin kehittämä MyoWare-lihasanturi. Anturi kiinnittyy suoraan iholle pintaelektrodien avulla (kuva 18). Sillä on mahdollisuus mitata raakaa EMG-signaalia tai valmiiksi käsitelty signaali, jossa on tasasuuntaus positiiviseksi ja signaalin verhokäyrä on laskettu käyttäen kaavaa 2 (AVR) (kuva 19). Anturi toimitetaan ilman liittimiä, jotka pitää erikseen juottaa piirille. Saatavilla on anturiin kytkettävä lisäosa, joka mahdollistaa elektrodien kytkennän lihakseen EMG-kaapelin avulla. [18.]



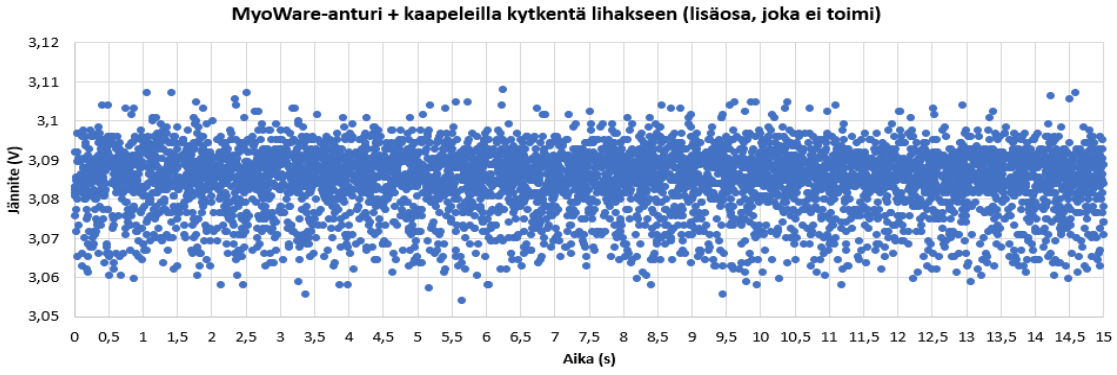
Kuva 18. Anturien sijainti käsivarressa. Ylempi oikean käden ranteen ojentajalihaksessa, alempi koukistajalihaksessa. Mustat referenssielektrodit kyynärtaipeen luisiin kohtiin.

Raa'an EMG-signaalin, jossa on niin positiivisia kuin negatiivisia amplitudeja. Sensori tuottaa laskemalla keskikohdan jakamalla kahdella sille annetusta käyttöjännitteestä (Picon 3,3 V / 2 = 1,65 V). Mitatut jännitteet, jotka ovat suurempia kuin 1,65 V ovat signaalin positiivisia amplitudeja ja sitä pienemmät negatiivisia. Todellisuudessa käyttöjännite oli noin 3,1 V, jolloin mitatussa raa'assa EMG-signaalissa keskikohdaksi tuli tyypillisesti noin 1,55 volttia. [18.]

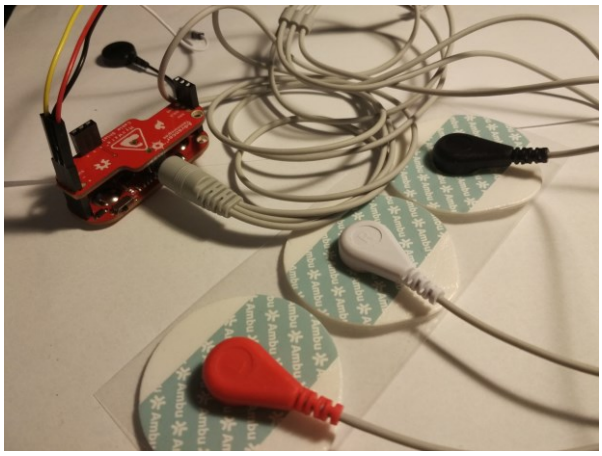



Kuva 19. MyoWare-anturin tuottamat signaalit. Ylempänä on raaka EMG-signaali. Alempana on anturin valmiiksi käsittelemä signaali.

Anturille hankittiin kaapelikytkennän mahdollistava lisäosa (kuva 21) Tämä ei kuitenkaan toiminut ja sillä mitatut lukemat olivat satunnaisia. Kuvassa 20 on tallennettu 15 sekunnin testiajon data kaapelilisäosaa käyttäen. Testissä on puristettu kättä nyrkkiin, jota ei erota kohinasta.




Kuva 20. MyoWare-anturin tuottama data, kun käytössä on kaapeliliitännän lihakseen mahdollistava lisäosa.



Kuva 21. MyoWare-anturi, johon liitetty EMG-kaapelikytkennän mahdollistava lisäosa.

### 4.2.3 Osalista ja hinnat

Prototyyppiä varten tilatut komponentit ja niiden hinnat ovat listattuna taulukkoon 1. Prototyypin ydinkomponentit koostuivat Pico-mikro-ohjaimesta, kahdesta EMG-anturista ja kahdesta servomoottorista. Näiden osien lisäksi tarvittiin liitäntöjä varten kaapeleita ja juotettavia pinnejä.

Kokonaissummaksi tuli yhteensä 140,09 euroa. Osalistalta voi jättää pois toimimattomiksi todetut antureiden kaapelilisäosat, EMG-kaapelit ja tarvittavat Sparkfun liitäntäpinnit (8 kpl), jolloin loppusummaksi jää 103,09 euroa. Pintaelektrodeita ei ole hankinnoissa huomioitu, sillä niitä oli koululla valmiina.

Taulukko 1. Prototyypin osaluettelo ja hinnat (mouser.fi).

| # | Tuote | Tuotekuva | Määrä | Hinta (kpl) |
|---|---|---|---|---|
| 1 | Raspberry Pi Pico -mikro-ohjain |  | 1 | 4,62 € |
| 2 | Juotospinnit 20-pinninen. Picon reunoille. |  | 2 | 2,32 € |

| | | | | |
|---|---|---|---|---|
| 3 | Seeed Studio Grove Shield -lisäosa Picolle |  | 1 | 3,67 € |
| 4 | Seeed Studio Grove servomoottori |  | 2 | 4,97 € |
| 5 | Seeed Studio Grove 4-pinnin uroskaapelit |  | 2 | 2,02 € (Viiden kaapelin paketti) |
| 6 | Advancer Technologies MyoWare lihasanturi |  | 2 | 38,16 € |
| 7 | MyoWare-anturin kaapelilisäosa |  | 2 | 4,50 € |
| 8 | Olimex EKG/EMG-kaapeli |  | 2 | 12,12e |
| 9 | Sparkfun 3-pinnin uros- & naarasliitos (anturreille) |  | 12 | 0,47 € |
| | | | **Kokonaissumma** | **140,09 €** |

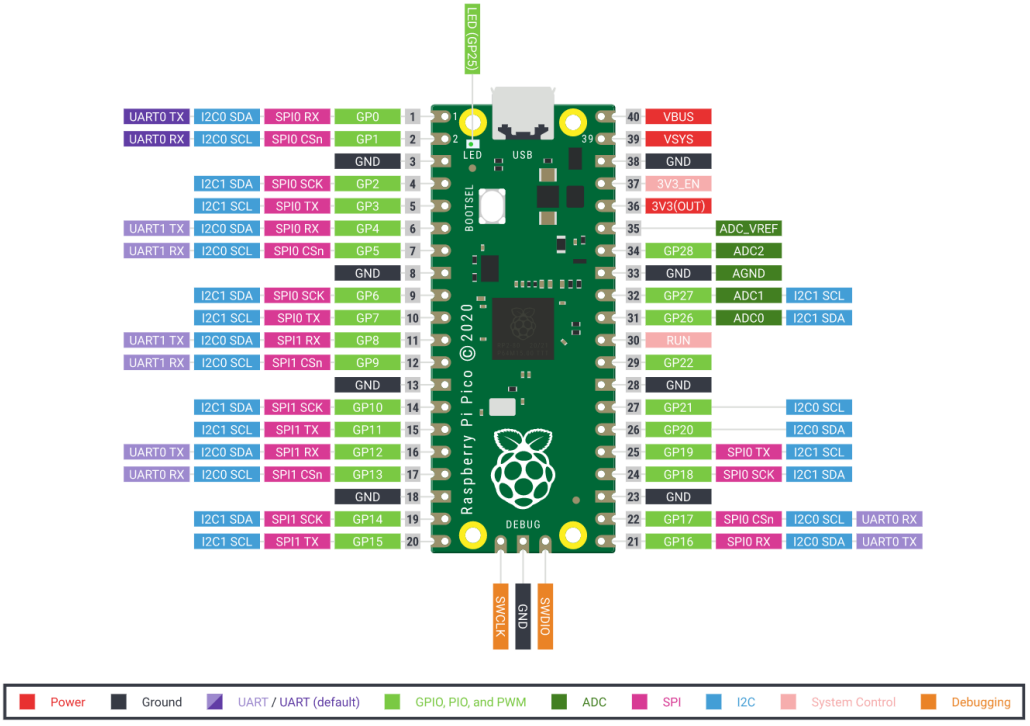
## 4.3 Toteutus

### 4.3.1 Komponenttien kytkentä

Picon pinnien tulkinta on tehty helpoksi. Picon kummallakin reunalla on 20 numeroitua pinniä per puoli, jotka tukevat eri tiedonsiirtomuotoja. Kuvassa 20 pinnit on numeroitu ja niihin on merkattu, mitä tiedonsiirtomuotoja pinni tukee.

Tulkitsemalla kuvasta löytyvät A/D-muuntimeen kytketyt pinnit #31 ADC0 (GP26), #32 ADC1 (GP27) ja #34 ADC2 (GP28). A/D-muuntimen maadoitus on pinnissä #33. Pinnin #35 A/D-muuntimen ulkoista jännitereferenssiä ei käytetty, vaan kaikki perustuu piirin sisäiseen jännitteeseen.

Vaalealla vihreällä merkityt GP(n)-pinnit tukevat pulssinleveysmodulaatiota, joilla ohjataan servomoottoreita. Ottaessa pinnin käyttöön koodissa sen tunniste on GP:n jälkeinen numero. Esimerkiksi ADC0-pinni otetaan käyttöön ”sensori = ADC(26)”. [16.] [17.]

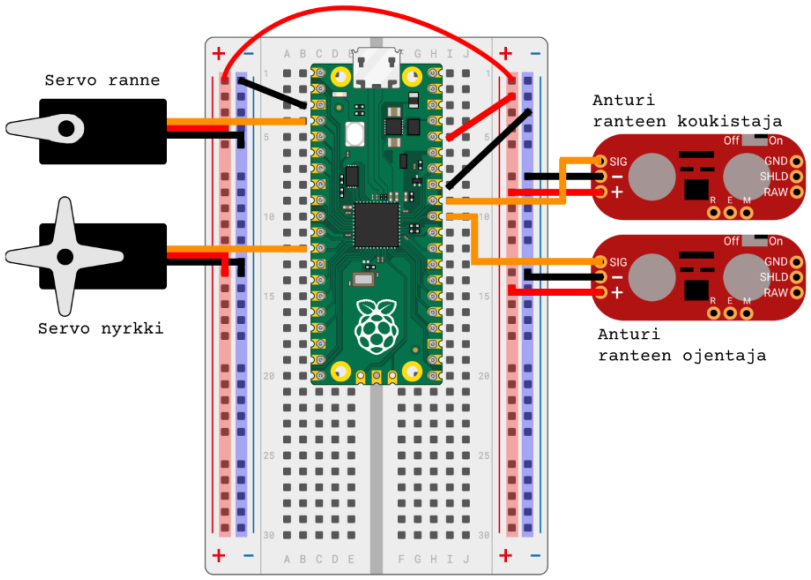



Kuva 20. Picon pinnien kartta. [20].

Prototyypin kehityksen aikana kytkentöjä tehtiin koekytkentälevylle ja Seeed Studion kehittämällä Pico Shield -lisäosalla, joka mahdollistaa kytkennät pikaliittimien avulla. Kuvassa 21 on havainnollistettu komponenttien kytkentää koekytkentälevyllä.

Virta viedään levyn virtakiskoille pinnistä #36, joka antaa 3,3 V jännitteen. Servomoottoreiden signaalikaapelit mihin tahansa GP(n)-pinniin.

Antureiden signaalikaapelit joko anturin ”RAW”- tai ”SIG”-liittimeen riippuen mitataanko raakaa EMG-signaalia vai valmiiksi käsiteltyä signaalia. Käsivarressa sijaitsevan ranteen koukistajalihakseen kiinnitettävän anturin signaalikaapeli on pinniin #31 ADC0 (GP26). Ranteen ojentajalihaksen anturi on pinniin #32 ADC1 (GP27). Anturien yhteinen maadoitus on pinniin #36 AGND.



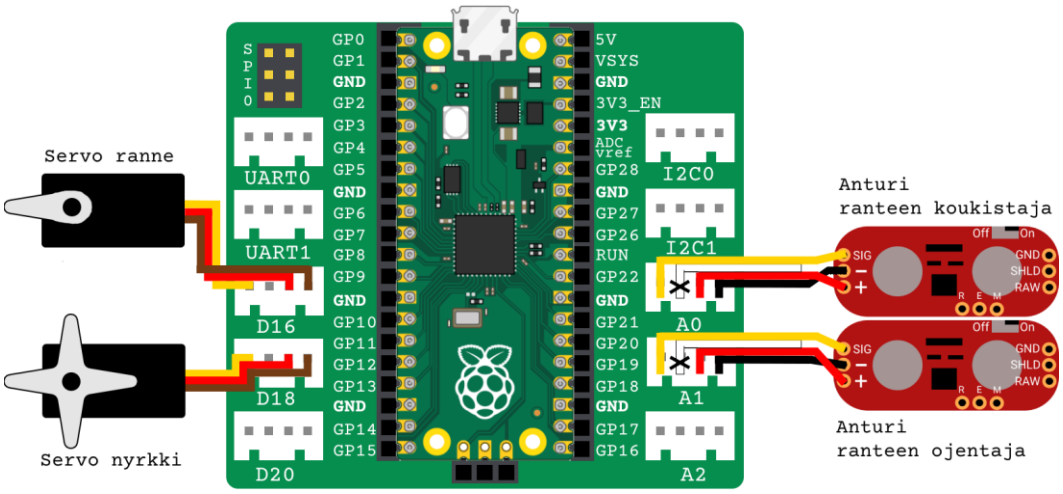

Kuva 21. Komponenttien kytkentä koekytkentälevyllä. Punainen väri on virta, musta on maadoitus ja oranssi on signaali. [20]

Prototyypille hankittiin Seeed Studion valmistama Pico Shield -lisäosa. Siinä on Picon 40 pinnille naarasliittimet, johon sen voi helposti kiinnittää. Lisäosa tarjoaa 10 kappaletta Grove-portteja, joihin kytkeminen toimii pikaliittimien avulla. Muun muassa A/D-muuntimen kaikille kolmelle kanavalle on omat Grove-portit. Shieldiä käyttäessä päästään yksittäisistä kaapeleista eroon, joita on koekytkentälevyä käyttäessä. Picon yksittäiset pinnit ovat silti edelleen käytettävissä.

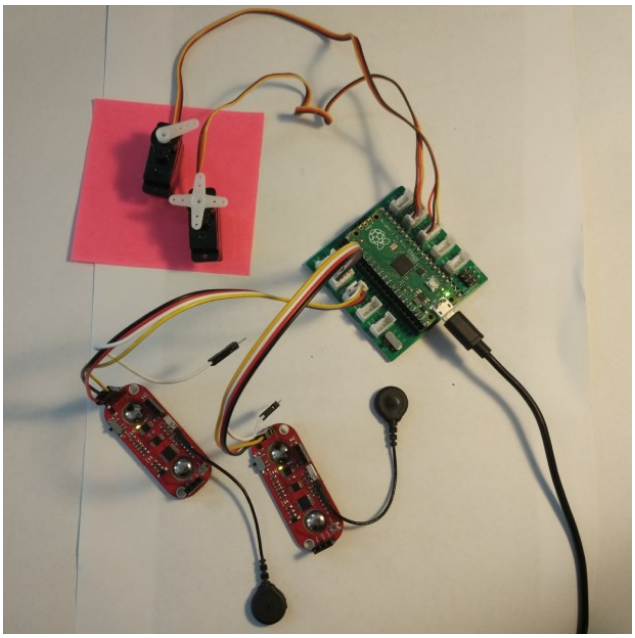
Kuvassa 22 on kytkentäkaavio hyödyntäen Pico Shieldiä ja sen alapuolella olevassa kuvassa 23 on prototyyppi koottuna fyysisesti. Ranteen koukistajalihakseen kiinnitettävä anturi kytkettiin Shieldin A0-porttiin käyttäen nelipinnistä Grove-uroskaapelia. Signaalin keltainen kaapeli kytkettiin Myoware-anturin ”SIG”-liitäntään, maadoituksen musta kaapeli anturin miinusmerkkiseen liitäntään ja punainen virtakaapeli kytkettiin anturin plusmerkkiseen liitäntään.

Grove-kaapelin valkoiseen piuhaan ei tullut kytkentää. Ranteen ojentajalihakseen kiinnitettävä anturi kytkettiin porttiin A1 samoilla värikoodauksilla.

Servomoottoreissa oli jo valmiiksi Grove-liitin, jossa keltainen kaapeli on signaali, punainen virta ja ruskea maadoitus. Servomoottori, jonka lavan suuntaus muuttuu ranteen koukistuksen ja ojennuksen mukaan, kytkettiin porttiin D16. Nyrkinpuristukseen reagoiva servomoottori kytkettiin porttiin D18.




Kuva 22. Kytkentäkaavio käyttäen Seeed Studion Pico Shieldiä. [20]



Kuva 23. Prototyyppi koottuna.

### 4.3.2 Tuotetut MicroPython-ohjelmat

Prototyypin kehityksen aikana MicroPython-ohjelmia syntyi kaksi, jotka molemmat esitellään tässä luvussa. Toinen on tarkoitettu anturin tuottaman datan tallentamiseen, ja sen lähdekoodi on liitteessä 3. Toinen on käsiproteesiohjaimen demokoodi, jonka lähdekoodi on liitteessä 2. Lähdekoodit on julkaistu MIT-lisenssillä ja sen ehdot on kirjoitettu lähdekoodin alkuun.

Koodit on pyritty kirjoittamaan mahdollisimman ymmärrettävään muotoon, jotta sitä ymmärtäisi henkilö, jolla ei ole paljoa kokemusta koodauksesta. Kommentointia on käytetty runsaasti ja muuttujien nimet ovat mahdollisimman informatiivisia. Käytäntö ei ole ihan optimi mikro-ohjainympäristössä, jossa laskentatehoja on rajallisesti.

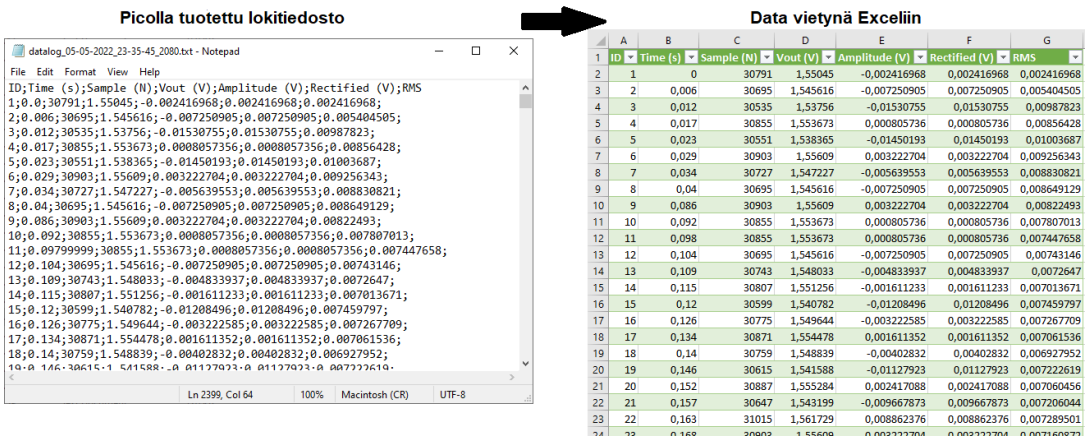
MicroPythonin kirjoittamista varten käytettiin Thonny-ohjelmointiympäristöä, jota ajettiin Windows 10 -käyttöjärjestelmässä. Thonny:sta löytyy suoraan tuki Picolle. Asennukseen löytyvät ohjeet löytyvät thonny.org-sivuilta. Asennus on tehty suoraviivaiseksi ja Pico saatiin keskustelemaan tietokoneen kanssa ongelmitta.

#### Anturin datan tallennusohjelma

Anturin datan tallennusohjelmaa kehittäessä huomattiin Pythonin ja MicroPythonin erot. Aluksi sitä kehitettiin Pythonilla ilman, että Pico oli kytkettynä tietokoneeseen. Koodin suorittaessa Picolla syntyi virhekoodeja. Karsitusta MicroPythonista ei löytynyt samanlaista valmista funktioita kuin Pythonista, joka mittaa ohjelman ajon aikana kulunutta aikaa. Koodi piti siirtää MicroPythonille sopivaksi. MicroPythonista on myös karsittu monia tärkeitä statistiikkaan tarvittavia funktioita, josta ohjelman kannalta oleellinen oli mediaanin laskeminen taulukosta. Funktio löytyi ulkopuolisen kehittäjän toimesta. Kehittäjään on viitattu lähdekoodissa.

Ohjelman tarkoituksena on mitata lihasanturia tietyn ajanjakson ja käsitellä A/D-muuntimelta saatua signaalia digitaalisesti. Signaalinkäsittelyn eri vaiheista

saadut arvot kirjataan lokitiedostoon tekstimuotoisena (.txt), jotta arvot voidaan siirtää Microsoft Exceliin tarkasteltavaksi sekä graafien luontia varten. Lokitiedosto tallentuu Picon sisäiseen muistiin. Kuvassa 24 on vasemmalla ohjelman tuottama lokitiedosto ja oikealla näkymä, kun se on viety Exceliin.




Kuva 24. Picon tuottama tekstitiedosto vasemmalla, jossa on ohjelman tuottamia arvoja. Oikealla näkymä lokitiedoston vietynä Exceliin.

Ohjelman alussa on kolmen sekunnin kestoinen aloitussekvenssi, jossa mitataan anturia ja samalla ilmoitetaan käyttäjälle testin aloituksesta. Indikaattorina testin aloituksesta toimii Picon oma ledivalo, joka vilkkuu ensiksi hitaammin kahden hertsin taajuudella.

Anturilta luetut arvot laitetaan taulukkoon, josta saadaan laskettua signaalin keskikohta (mediaani), jota tarvitaan laskettaessa signaalin positiiviset ja negatiiviset amplitudit. Sekvenssin loputtua taulukko nollataan, ettei se vie turhaan välimuistia.

Aloitussekvenssissä luodaan myös tekstitiedosto ja sen ensimmäiselle riville kirjoitetaan taulukon solujen otsikot muodossa "ID;Time (s);Sample (N);Vout (V);Amplitude (V);Rectified (V);RMS\r", jossa puolipiste toimii Excelin solujen erottelijana. Komento ”\r” aloittaa tiedostoon uuden rivin.

Aloitussekvenssin kesto oli aluksi viisi sekuntia, mutta taulukon koko kasvoi sen aikana liian isoksi eikä Picon välimuisti riittänyt sille. Ohjelma kaatui antaen

virheilmoituksen muistin riittämättömyydestä. Vika korjattiin ottamalla vähemmän näytteitä taulukkoon ja lyhentämällä sekvenssin kestoa. Virhekoodin lainaus alla.

```
MemoryError: memory allocation failed, allocating
54592 bytes
```

Aloitussekvenssin jälkeen alkaa ohjelman varsinainen osuus, jossa suoritetaan näytteenottoja anturilta. Picon ledivalo alkaa vilkkua tiheästi 20 Hz:n taajuudella ilmoittamaan käyttäjälle varsinaisen osuuden alkamisesta. Testijakson kestoa voi muuttaa lähdekoodista tarpeen mukaan. Tyypillisin käytetty kesto oli 15 sekuntia.

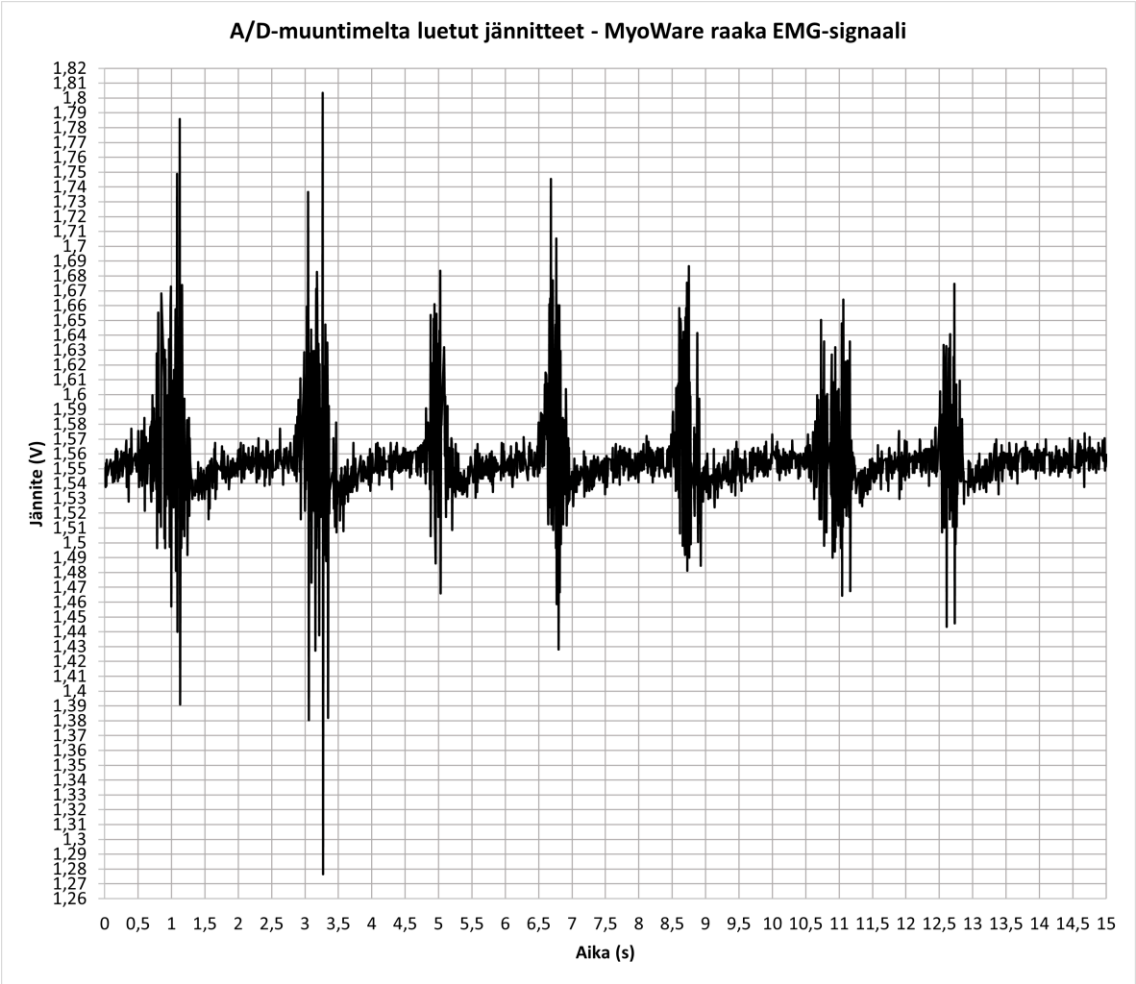
Testijakson aikana voidaan suorittaa erilaisia lihasaktivaation testejä. Tyypillisin testi oli käden puristamista nyrkkiin tasaiseen tahtiin, jonka jälkeen käsi hetkeksi rennoksi ja jälleen nyrkin puristamista. Tarkoitus on havaita yhden anturin toimivuutta, eli näkyykö signaalista eroavaisuudet lihasaktivaatiosta verrattuna lihaksen ollessa rentona. Näytteenoton ajankohdat tallennetaan lokitiedoston taulukon soluun ”Time (s)”.

Anturilta luetut tiedot ovat A/D-muuntajalta saatuja lukemia, joka on 12-bittinen. MicroPythonissa lukujoukko esitetään 16-bittisessä muodossa antaen lukemia väliltä 0–65 535. Lukemat tallennetaan soluun ”Sample (N)”.

Lukemat muutetaan ymmärrettävämpään muotoon volteiksi kaavan 3 [14] mukaisesti ja tallennetaan soluun ”Vout (V)”. Datan vietynä Exceliin siitä syntyy kuvan 25 mukainen graafi, jossa Y-akselilla on A/D-muuntimen havaitsemat jännitteiden muutokset ja X-akselilla aika.

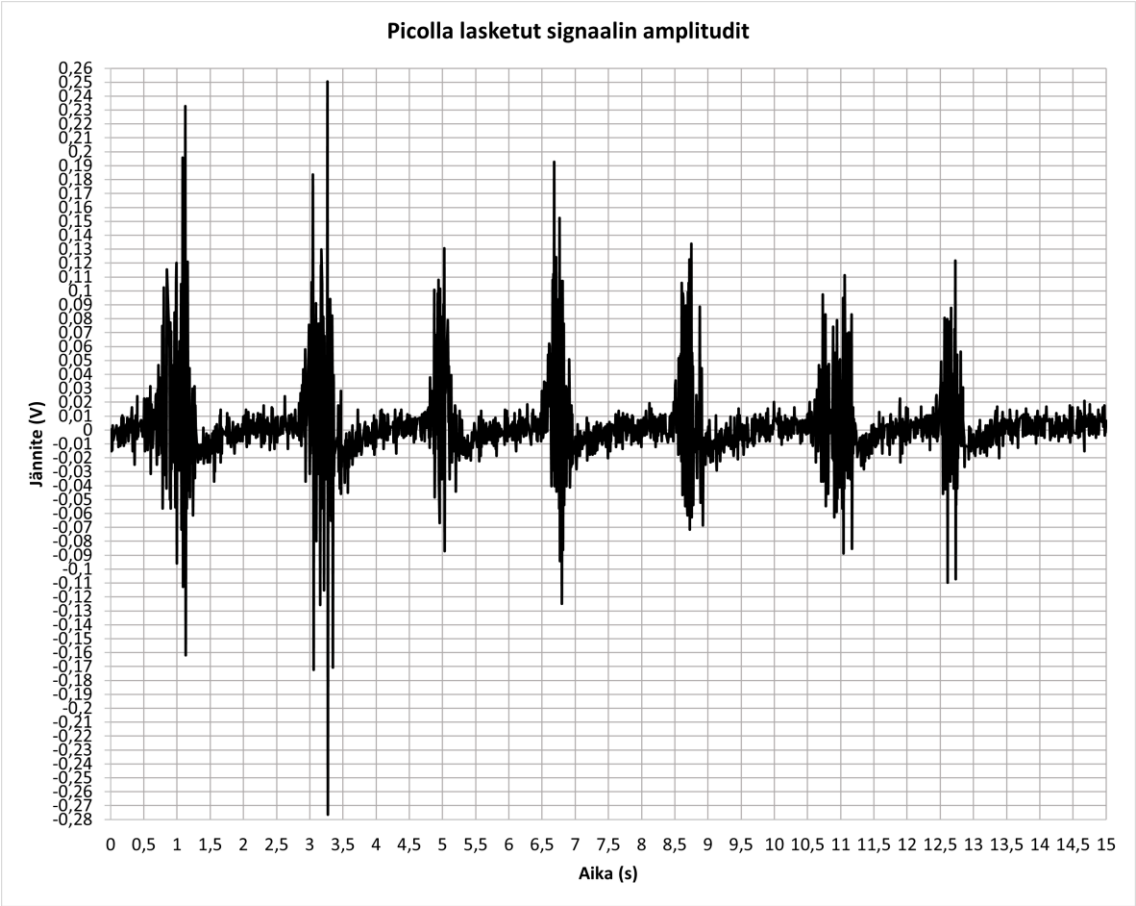
$$V_{out} = \frac{N * V_{ref}}{2^{bit}} \tag{3}$$

N on A/D-muuntimelta saatu arvo
$V_{ref}$ on jänniterefenssi



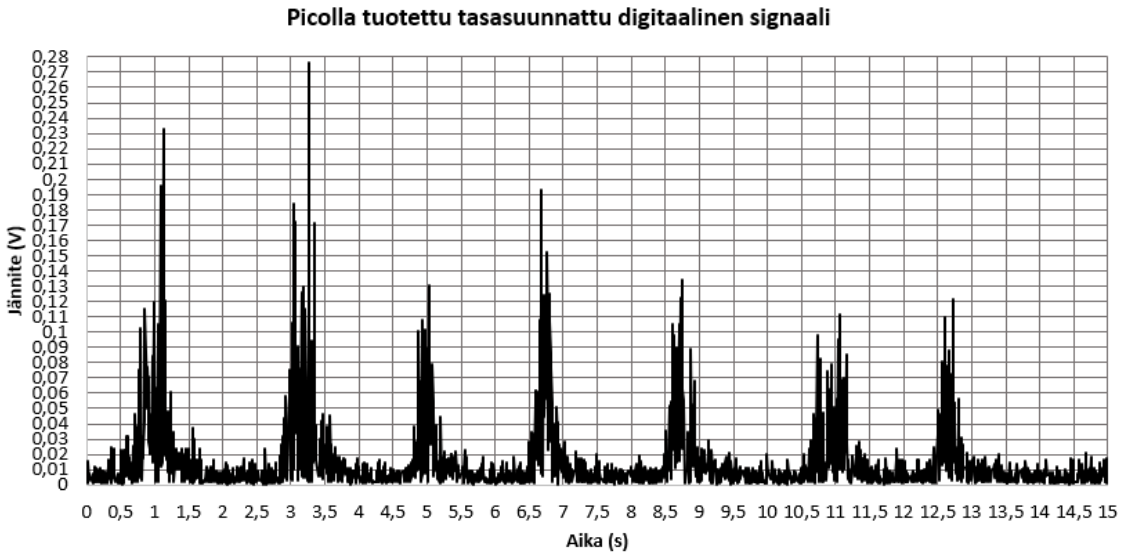

Kuva 25. A/D-muuntimen mittaamia jännitteitä.

Näytteenotossa saaduista jännitteistä lasketaan signaalin positiiviset ja negatiiviset amplitudit vähentämällä arvosta signaalin keskiarvo ($V_{out}$ – mediaani), jonka arvo on laskettu aloitussekvenssin aikana. Signaalin keskikohdaksi muodostuu lukema nolla, jolloin amplitudien positiiviset ja negatiiviset voimakkuudet saadaan eroteltua (kuva 26). Amplitudin arvot tallennetaan taulukon soluun ”Amplitude (V)”.




Kuva 26. Signaalin amplitudit, jotka laskettu Picon piirillä.

Signaalin kokonaisesta aallonmuodosta, jossa on niin negatiivisia kuin positiivisia jännitteitä, saadaan tasasuuntaus positiiviseksi laskemalla amplitudeista itseisarvot. Tällöin negatiiviset jännitteet muuttuvat positiiviksi ja muodostuu puoliaalto, jossa on säilytetty kaikki mittauspisteet (kuva 27). Näiden tiedot tallennetaan soluun ”Rectified (V)”.

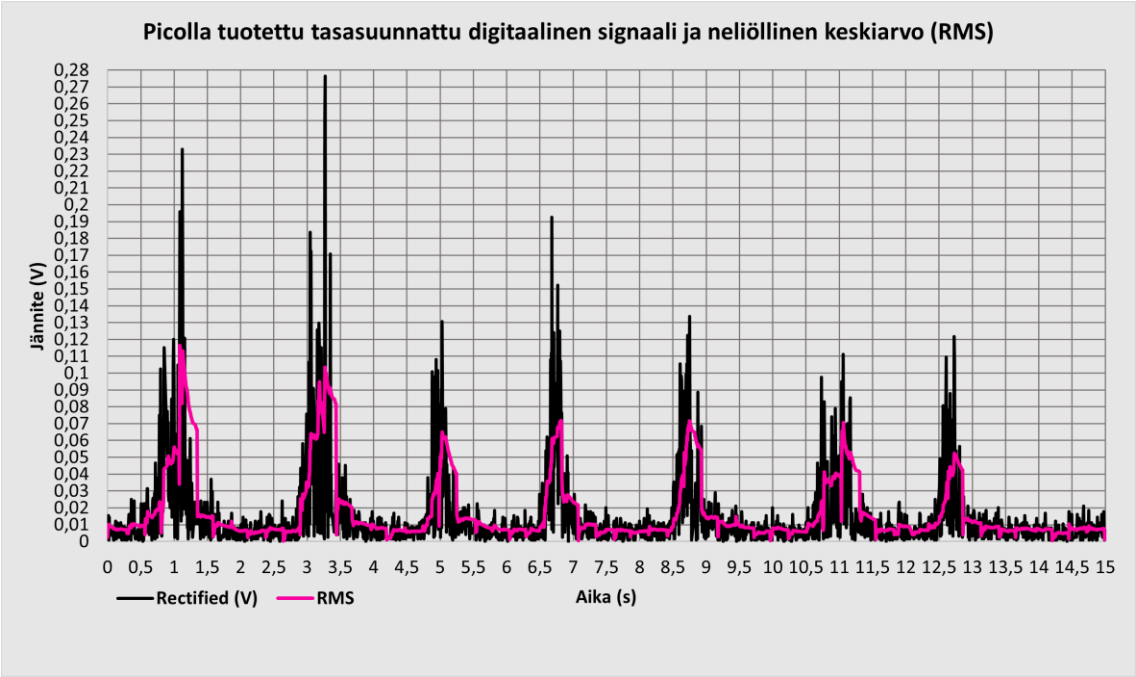



Kuva 27. Picolla tuotettu tasasuunnattu digitaalinen signaali, jossa kaikki mittauspisteet on säilytetty ja amplitudit ovat positiivisia.

Tasasuunnatussa signaalissa on edelleen jännitteen vaihtelua. Yhden lihasaktivaation aikana jännitteen arvot voivat käydä lähellä nollaa. Jännitteen vaihtelun eliminoimiseksi ja signaalin verhokäyrän laskentaan käytettiin neliöllistä keskiarvoa, jonka kaava (2) esitettiin luvussa 3.1 (kuva 28). Laskenta RMS:lle tapahtuu reaaliaikaisesti ohjelmaa ajettaessa. Näiden arvot tallennetaan soluun ”RMS”. RMS:n käyrä näkyy kuvaajassa purppuran värisellä viivalla.

Mallissa on vielä hiomista, sillä laskentaa tehdään 42 näytteen jaksoissa. Näytteiden lukumäärän saavuttaessa 42 kappaletta laskettu keskiarvo nollataan. Amplitudien summat ja näytteiden lukumäärä palautetaan alkutilaan. Uudessa jaksossa amplitudien summa alkaa jälleen nollasta ja näytteiden lukumäärä yhdestä. Tällöin voi käydä tilanne, jossa kesken lihasaktivaation laskenta alkaa alusta ja keskiarvo putoaa liian alhaiseksi. Tällainen keskiarvon putoaminen keskellä lihasaktivaatiota näkyy kuvaajasta noin 5 ja 11 sekunnin kohdilla.

Pienemmillä jaksoilla keskiarvojen putoamisia tapahtui useammin, mutta samalla viive on pienempi. Suurempi jakso aiheuttaa lasketulle signaalille viivettä. Tässäkin mallissa viivettä huomataan, kun RMS:n käyrä on enemmän oikealle vino suhteessa mitattuun lihasaktivaation (musta alue).




Kuva 28. Picolla luotu tasasuunnattu signaali, josta on laskettu neliöllinen keskiarvo (purppura viiva). Ajan hetkillä 5 ja 11 näkyy signaalin putoaminen. Lisäksi signaalissa on viivettä suhteessa mitattuun signaaliin.

Kun kaikki arvot (ID, Time (s), Sample (N), Vout (V), Amplitude (V), Rectified (V), RMS) ovat laskettu ja lisätty taulukkoon, se viedään funktiolle. Funktio käy silmukassa läpi taulukon alkiot ja kirjoittaa niiden sisältämän tiedon tekstitiedostoon uudelle riville. Ohjelma suorittaa itseään, kunnes asetettu aikaraja täyttyy.

Lopuksi tekstitiedosto suljetaan ja Picon ledivalo jää jatkuvasti päälle indikoimaan käyttäjälle testiohjelman päättymisestä. Kaikki data katoaa, mikäli ohjelma keskeytyy testiajon aikana eikä tekstitiedostoa suljeta oikein.

Käytännössä ei ole järkevää käsitellä signaalia digitaalisesti Picon piirillä, sillä sen laskentatehot tulevat rajoitteeksi. Ensimmäisessä datan tallennusohjelman versiossa tallennettiin pelkästään ajan, A/D-muuntimen tuottaman digitaalisen arvon ja siitä muunnetun jännitteen arvon. Datapisteitä syntyi lokitiedostoon noin 5500 kappaletta 15 sekunnin testiajon aikana.

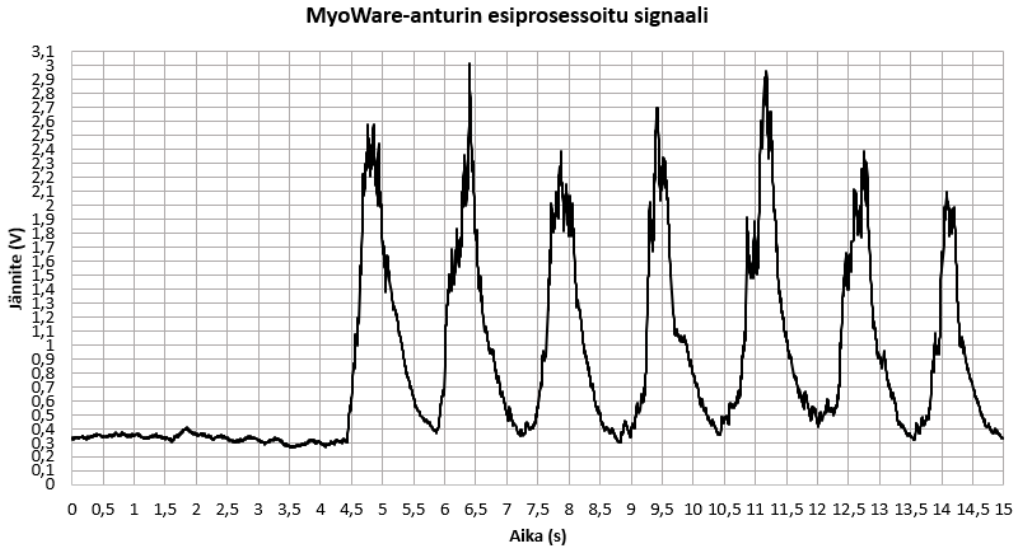
Toisessa versiossa mukaan tuli amplitudien laskeminen ja signaalin tasasuuntaus sekä näiden arvojen kirjoitus lokitiedostoon. Datapisteiden määrät putosivat noin 4400:aan 15 sekunnin testiajolla. Viimeisimmässä versiossa, jossa on jokainen signaalinkäsittelyn vaihe sekä neliöllisen keskiarvon jatkuva laskenta, datapisteitä syntyi noin 2400 kappaletta.

Näytteenotto heikentyi noin 56 prosentilla verrattuna ensimmäisen version tuottamaan dataan, Picon käyttäessä enemmän tehoja RMS:n laskentaan. Datan tallennusohjelmassa mitataan yhtä anturia, joten toisen anturin lisättäessä ja molempien antureiden signaalien laskeminen kuluttaisi tehoja vielä enemmän. Lisäksi lokitiedoston koko kasvoi noin 20 prosenttia 120 kilotavusta 145 kilotavuun.

### Käsiproteesiohjaimen demo-ohjelma

Alkuperäinen idea oli toteuttaa käsiproteesiohjaimen logiikka hyödyntäen digitaalisesti käsiteltyä signaalia. Kuten anturin datan tallennusohjelmasta ilmeni, niin signaalinkäsittely digitaalisena suoraan mikro-ohjaimella ei ole kannattavaa. Näytteenotossa hävitään merkittävä määrä dataa Picon laskentatehojen tullessa rajoitteeksi.

Prototyypin demokoodissa hyödynnettiin MyoWare-anturin valmiiksi käsittelemää signaalia (kuva 29). Tällä kevennettiin Picolle asettamaa kuormaa. Lisäksi valmiiksi käsitellyssä signaalissa ei tapahdu samanlaista keskiarvon putoamista kesken lihasaktivaation kuin kehitetyllä digitaalisella mallilla. Anturin tuottamien arvojen skaala on 0–3,1 volttia, mikä selkeyttää huomattavasti lihasaktivaatioiden raja-arvojen miettimistä, joiden ylittyessä servomoottorit liikkuvat.

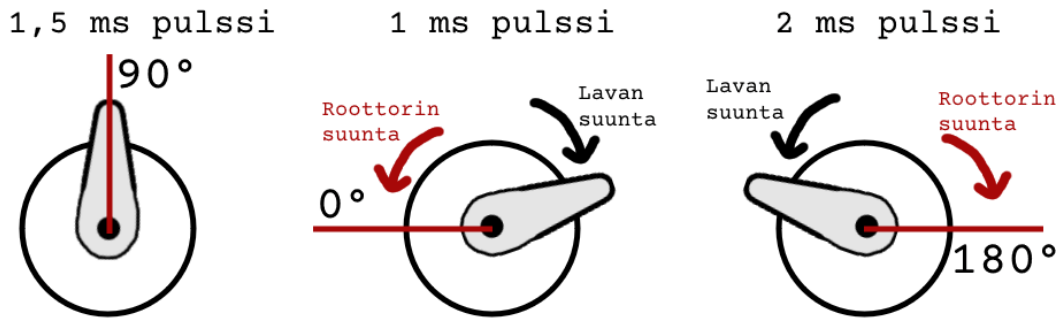



Kuva 29. MyoWare-anturin valmiiksi käsittelemä signaali, jossa on tasasuuntaus positiiviseksi ja verhokäyrä. Y-akselin jännitteen skaala 0–3,1 V. X-akselilla aika (s).

Logiikan vuokaavio on liitteenä 1, joka visuaalisesti kertoo ohjelman toiminnan. Sen tarkoitus on olla tukena ohjelman toiminnasta kirjoitettua tekstiä lukiessa. Ensiksi määritellään koodissa käytettävien pinnien sijainnit. Ranteen koukistajalihakseen tuleva pinni kytkettiin ADC0-pinniin ja ojentajalihaksen anturi ADC1-pinnii. Ranteen liikkeisiin reagoivaa servomoottori kytkettiin GP16-pinniin ja nyrkin puristukseen reagoiva GP18-pinniin.

Alussa määritellään myös servomoottoreiden käytetty taajuus ja pulssinleveydenkestot. Pulssien kestot ovat millisekunteja, mutta koodissa ne määritellään nanosekunteina.

Servomoottori ottaa pulsseja 20ms:n välein, joten taajuudeksi tulee 50 Hz. Servomoottorilla on 180 asteen liikkuvuus (kuva 30). Pulssinleveyden kestäessä 1,5 ms:n servon roottori pyörähtää 90 asteeseen eli keskipisteeseen, jolloin siinä nähtävissä oleva lapa on myös keskellä. Pulssinleveyden kestäessä 1 ms roottori pyörähtää 0 asteeseen eli roottorin keskipisteestä vasemmalle. Tällöin nähtävissä oleva lapa liikahtaa oikealle servon sisällä olevien rattaistojen takia. Pulssin kestäessä 2 ms roottori pyörähtää 180 asteeseen eli keskipisteestä oikealle, jolloin lavan suunta on vasemmalle. [20.]




Kuva 30. Servomoottorin ja lavan pyörimisliikkeiden suhde.

Ohjelman varsinaisessa osiossa on loputon silmukka, joka keskeytyy ainoastaan Picon sammuttamalla tai keskeyttämällä ohjelma Thonnyssa. Silmukan sisällä luetaan ensiksi molemmista antureista signaalit. A/D-muuntajalta saadut digitaaliset arvot (0–65 535) annetaan funktiolle, joka muuttaa arvot ymmärrettävään muotoon volteiksi kaavan 3 mukaisesti.

Seuraavassa vaiheessa päästään kolmeen ehtolauseeseen, joilla tarkistetaan, kumpi lihasantureista on aktiivisena vai ovatko molemmat. Ehtojen raja-arvoina on käytetty anturin tuottaman signaalin lukemia (skaala 0–3,1 V).

Oikean käden puoleisen ranteen koukistajalihaksen anturin ollessa aktiivisena ja ojentajalihaksen ollessa rentona liikutetaan servon lapaa vasemmalle. Vastaavasti ojentajalihaksen anturin aktivoituessa ja koukistajalihaksen ollessa rentona liikutetaan servon lapaa oikealle.

Jos molemmat anturit ovat aktiivisena nyrkkiä puristaessa, niin nyrkin puristukseen reagoivaa servomoottoria liikutetaan. Mikäli mikään ehdoista ei täyty, annetaan pelkästään nyrkkiservolle 1,5 ms:n pulssi, joka palauttaa sen asennon keskelle. Ehtojen ja servomoottorien liikuttamisen jälkeen käsketään A/D-muuntajaa nukkumaan yhden millisekunnin ajan, josta silmukka alkaa taas alusta.

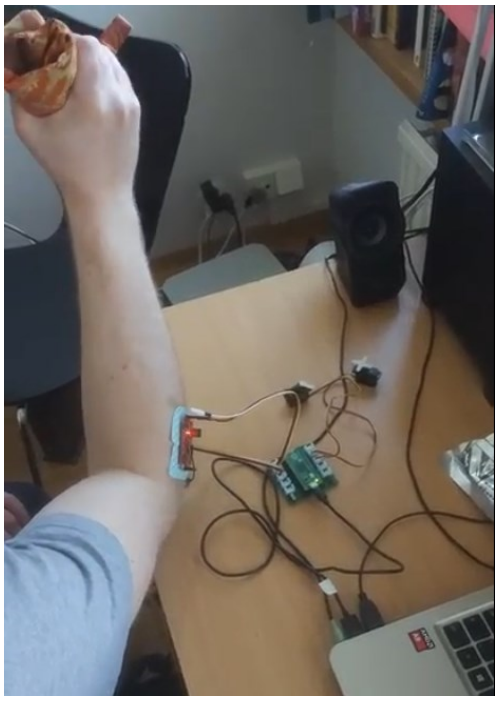
### 4.3.3 Testaus

Prototyyppiä testattiin yhdellä henkilöllä ja tilannetta havainnoitiin. Tilanne videoitiin tallenteeksi (kuva 31). Testattavalta pyydettiin tunnustelemaan oikean käden käsivarresta lihasten paksuimmat kohdat, jotka tuntuivat ranteen koukistus- ja ojennusliikkeillä sekä kättä puristaessa nyrkkiin. Testattavalle liimattiin anturit käsivarteen pintaelektrodeilla ranteen koukistaja- ja ojentajalihakseen siten, että anturi on lihassyiden suuntaisesti ja toinen elektrodeista tulee lihaksen paksuimpaan kohtaan. Referenssielektrodit kytkettiin kyynärtaipeen luisiin kohtiin. Oikeat kohdat antureille löytyi ensimmäisellä yrittämällä, sillä testauksen aikana niitä ei tarvinnut liimata uusiksi toiseen kohtaan.

Testissä pyydettiin testattavaa puristamaan kättä nyrkkiin, jolloin nyrkinpuristukseen reagoiva servomoottori pyöri. Havaintona oli, että ohjelmassa asetetut nyrkin puristuksen raja-arvot olivat kuitenkin liian korkealla, jolloin kättä puristaessa tyhjänä servomoottori ei aina pyörähtänyt puristuksen tapahtuessa. Testattavalle laitettiin käteen patalappu puristeltavaksi, joka lisäsi käteen kohdistuvaa vastusta. Vastuksen ansiosta servomoottori reagoi jokaiseen puristukseen.

Rannetta koukistaessa ranteen servomoottorin lapa pyörähti vasemmalle. Rannetta ojentaessa lapa pyörähti oikealle. Havaintona oli myös, että ranteen servomoottorilla oli satunnaisesti tapahtuvia liikkeitä, vaikka ranne ei liikkunut.

Testin perusteella prototyypin tuottamat liikkeet toimivat kuten on tarkoituskin, eli se reagoi ranteen ojennus- ja koukistusliikkeisiin sekä nyrkin puristukseen. Ehtolauseiden raja-arvoja tulisi muuttaa tapauskohtaisesti, jotta ne ovat sopivia mitattavalle lihakselle.



Kuva 31. Tilannekuva testauksesta.

### 4.3.4 Prototyypin puutteet

Testissä tehdyt havainnot tukevat kehityksen aikana tehtyihin havaintoihin. Ajaessa demokoodia eri hetkinä ehtolauseiden raja-arvot eivät ollutkaan sopivat kuin edellisellä kerralla. Välillä servomoottorit reagoivat hyvinkin herkästi tuottaen ei-haluttuja liikkeitä. Raja-arvoja piti muuttaa lähdekoodissa useasti.

Testejä ei suoritettu ihmisellä, jolle on tehty amputaatio. Käyttötarkoituksen näkökannalta tämä olisi oleellinen testitapaus. MyoWare-anturi tarvitsi toimiakseen optimin sijainnin lihaksessa, jotta mahdollisimman iso määrä lihassyitä osuu anturin alle tuottaen vahvan signaalin. Lisäksi pitäisi löytää keino saada demokoodin yhteyteen lokitiedostojen tuottaminen, jotta testejä voisi tehdä systemaattisesti tuottaen tietoa eri kokoisista lihaksista.

Ulkoisella varavirtalähteellä (Trust Primo 10000) Picon sai käynnistymään, mutta hetken kuluttua varavirtalähde ja Pico sammuivat. Todennäköisesti tämä johtuu varavirtalähteen piiristä, ettei Pico vedä siitä tarpeeksi virtaa, jolloin se sammuttaa itsensä. Joten prototyypillä ei vielä tässä vaiheessa ole todennettu

käyttötarkoituksen mukaista käyttötapausta, että se kulkisi ihmisen mukana. Toiminnon prioriteetti ei ollut kuitenkaan tässä vaiheessa kehitystä korkealla, joten muita virransyöttömahdollisuuksia ei tutkittu. Määrittelyvaatimuksen ehto prototyypin toimivuudesta ilman tietokonetta kuitenkin täyttyi, sillä prototyyppi toimii itsenäisesti käyttäen yleistä USB-laturia kytkettynä verkkovirtaan.

### 4.4 Jatkokehitysideoita

Yhtenä ajatuksena työtä aloittaessa oli, että servomoottori matkisi käden liikettä enemmänkin. Mikäli rannetta liikuttaisi vain hieman, niin servomoottori liikahtaisi vain pari astetta. Toimiakseen tällä tavalla tulisi pohtia skaalautuva malli, missä signaalin voimakkuudella ja lihasaktivaatioiden kestoilla on vaikutusta servomoottoreille syötettyjen pulssinleveysmodulaation arvoihin.

Lisäksi servomoottorien pyörimiseen vaadittujen raja-arvojen laskemiseen olisi hyvä saada skaalautuva malli, ettei arvoja tarvitse muuttaa erikseen lähdekoodista. Koodi tunnistaisi eri käyttäjien tuottamista eri vahvuuksista signaaleista sopivat raja-arvot suoraan.

## 5 Yhteenveto

Työn lähtökohtana oli saada yksi servomoottori liikkumaan yhdellä anturilla rannetta koukistaessa ja ojentaessa. Yhdellä anturilla se ei kuitenkaan ole mahdollista lihasten toimiessa pareittain. Rinnalle tuli toinen anturi, joka mahdollisti liikkeet. Sen ansiosta prototyypille saatiin lisäksi nyrkin puristukseen reagoiva servomoottori. Joten lähtökohtaan nähden työ ylitti tavoitteet. Vielä on kuitenkin pitkä matka siihen, että amputaatiossa raajansa menettänyt henkilö saisi yhtä hyvän, ellei paremman bionisen raajan menettäneensä tilalle.

Pico osoittautui sopivaksi kehitysalustaksi prototyypille. Pieneksi piiriksi se suoriutui tehtävistään hyvin ja taipuu moneen asiaan. Se täyttää lihassähkökäyrän mittaamisen tekniset vaatimukset. Sen A/D-muuntimen resoluutio ja näytteenottotaajuus ovat riittävät havaitsemaan luonnossa tapahtuvat lihaksen

sähköimpulssit. Prototyypin lisäksi työssä tuotettiin kattava EMG-signaalin digitaaliseen käsittelyyn ja tallentamiseen tarkoitettu MicroPython-ohjelma.

Picon rajoitteitakin ilmaantui tässä työssä, jossa dataa tuotetaan lyhyessä ajassa paljon, etenkin jos laskentaa tehdään piirillä. Joten digitaalista signaalia hyödyntävä käsiproteesiohjaimen logiikka hylättiin. Välillä tallennustila loppui kesken, kun lokitiedostoja alkoi kasaantua. Välimuistikin saatiin kerran täyteen liian suurella määrällä dataa väliaikaiseen taulukkoon tallennettuna.

MicroPythonilla tuotos ideasta valmiiseen prototyyppiin syntyi luontevasti sen helposti ymmärrettävän syntaksin ja hyvien dokumentaatioiden ansiosta. Ennen kaikkea työn onnistumiseen vaikutti eri valmistajien laatimat kattavat dokumentaatiot sekä Raspberry Pi:n ympärille kasvanut laaja käyttäjäkunta.

## Lähteet


1 Bjålie, Jan G. ym. 2007. Ihminen: fysiologia ja anatomia. Helsinki: WSOY.

2 File:1217 Mechanically-gated Channels-02.jpg. 2016. © OpenStax. <https://commons.wikimedia.org/wiki/File:1217_Mechanically-gated_Channels-02.jpg> CC BY 4.0 licence <https://creativecommons.org/licenses/by/4.0/legalcode > Luvun 2.1 kuvalähde. Luettu 7.5.2022.

3 File:1222 Action Potential Labels.jpg. 2016. © OpenStax. <https://commons.wikimedia.org/wiki/File:1222_Action_Potential_Labels.jpg> CC BY 4.0 licence <https://creativecommons.org/licenses/by/4.0/legalcode >. Luvun 2.1. kuvalähde. Luettu 7.5.2022.

4 File:Illu muscle structure.jpg. Public Domain. Luvun 2.2 kuvalähde. <https://commons.wikimedia.org/wiki/File:Illu_muscle_structure.jpg> Luettu 7.5.2022.

5 File:Actin-myosin.png. 2017. © Jeff16 <https://commons.wikimedia.org/wiki/File:Actin-myosin.png> CC BY-SA 4.0 licence <https://creativecommons.org/licenses/by-sa/4.0/legalcode>. Luvun 2.3 kuvalähde. Luettu 7.5.2022.

6 File:Motor unit.png. 2019. © Daniel Walsh ja Alan Sved. <https://commons.wikimedia.org/wiki/File:Motor_unit.png> CC BY-SA 4.0 licence <https://creativecommons.org/licenses/by-sa/4.0/legalcode>. Luvun 2.3 kuvalähde. Luettu 7.5.2022.

7 File:Anatomical illustrations showing muscles of the lower arm Wellcome L0046740.jpg. 2018. © Wellcome Images. < https://commons.wikimedia.org/wiki/File:Anatomical_illustrations_showing_muscles_of_the_lower_arm_Wellcome_L0046740.jpg> CC BY 4.0 licence <https://creativecommons.org/licenses/by/4.0/legalcode>. Luvun 2.4 kuvalähde. Luettu 7.5.2022.

8 File:Anatomical illustrations showing muscles of the lower arm Wellcome L0046744.jpg. 2018. © Wellcome Images. <https://commons.wikimedia.org/wiki/File:Anatomical_illustrations_showing_muscles_of_the_lower_arm_Wellcome_L0046744.jpg> CC BY 4.0 licence <https://creativecommons.org/licenses/by/4.0/legalcode>. Luvun 2.4. kuvalähde. Luettu 7.5.2022.

9 Najarian, Kayvan; Splinter, Robert. 2012. Biomedical and Image Processing, second edition. Florida: CRC Press.

10 Duchateau, Jacques; Enoka, Roger. 2011. Human motor unit recordings: Origins and insight into the integrated motor system. Verkkoaineisto. Researchgate. <https://www.researchgate.net/publication/51496178_Human_motor_unit_recordings_Origins_and_insight_into_the_integrated_motor_system> Luettu 10.3.2022.

11 File:Willem Einthoven ECG.jpg. Public Domain (PD-old, PD-US). <https://commons.wikimedia.org/wiki/File:Willem_Einthoven_ECG.jpg> Luvun 3 kuvalähde. Luettu 7.5.2022.

12 Rodríguez-Tapia, Bernabe. ym. 2020. Myoelectric Interfaces and Related Applications: Current State of EMG Signal Processing–A Systematic Review. Verkkoaineisto. IEEE Xplore. <https://ieeexplore.ieee.org/document/8949764> Luettu 9.3.2022.

13 File:CNX UPhysics 16 02 SineWave.png. 2016. © OpenStax University Physics. <https://commons.wikimedia.org/wiki/File:CNX_UPhysics_16_02_SineWave.png> CC BY 4.0 licence <https://creativecommons.org/licenses/by/4.0/legalcode>. Luvun 3.1 kuvalähde. Luettu 7.5.2022.

14 How the Arduino ADC measures an input voltage. 2022. Verkkoaineisto. Skillbank. < https://www.skillbank.co.uk/arduino/adc.htm> Luettu 6.5.2022.

15 Carlo J. De Luca. 1997. The Use of Surface Electromyography in Biomechanics. Verkkoaineisto. De Luca Foundation. <https://www.delucafoundation.org/download/bibliography/deluca/078.pdf> Luettu 8.2.2022.

16 Raspberry Foundation. 2020. RP2040 Datasheet – A microcontroller by Raspberry Pi. Verkkoaineisto. Raspberry Pi Trading Ltd. <https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf> Luettu 30.3.2022.

17 Raspberry Foundation. 2020. Raspberry Pi Pico Python SDK – A MicroPython environment for RP2040 microcontrollers. Verkkoaineisto. Raspberry Pi Trading Ltd. <https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-python-sdk.pdf> Luettu 1.4.2022.

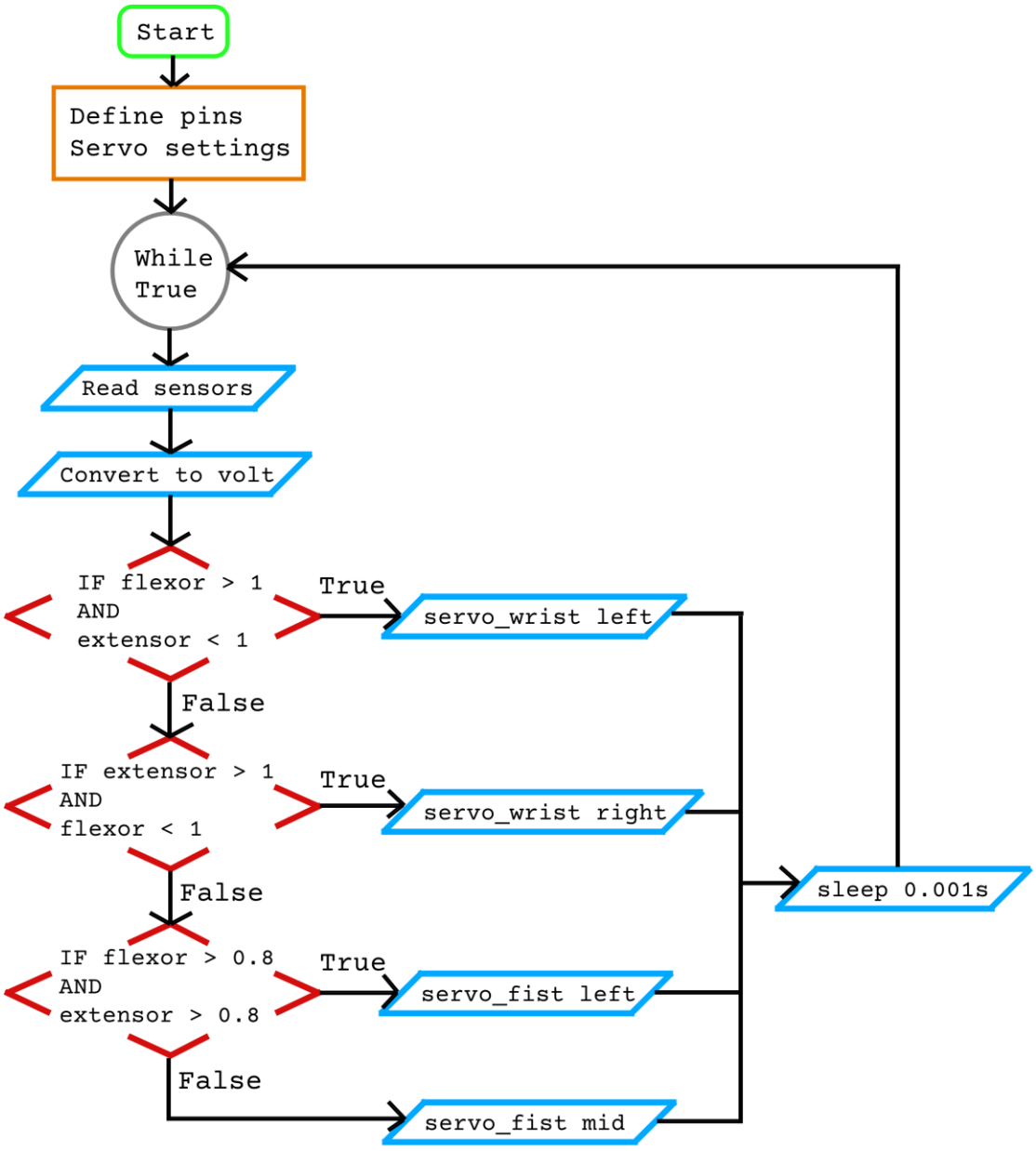
18 Advancer Technologies. 2015. MyoWare muscle sensor datasheet. Verkkoaineisto. Sparkfun. <https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MyowareUserManualAT-04-001.pdf> Luettu 26.4.2022.

19 Raspberry Pi Foundation. 2022. Getting started with Raspberry Pi Pico. Verkkoaineisto. Raspberry Pi Foundation.

<https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/0> Luettu 8.5.2022. Luvun 4.3.1 alkuperäisten kuvien lähde, joista tehty omat muokkaukset. Luvat kuvien käyttöön ja muokkaamiseen julkisesti tulevassa insinöörityössä on tarkistettu sähköpostitse suoraan valmistajalta. Käyttö ja muokkaaminen on sallittua opetustarkoituksiin.

20 Servo Motor control using MicroPython. 2020. Verkkoaineisto. Techawarey. < http://techawarey.com/programming/micropython/servo-motor-control-using-micropython/> Luettu 4.5.2022.

# Liite 1: Käsiproteesiohjaimen demokoodin vuokaavio



## Liite 2: Käsiproteesiohjaimen lähdekoodi

```
#---------------------------------#
# Prosthetic Arm Controller v. 0.1 #
# for RP2040-microcontroller       #
# Demo code                        #
#---------------------------------#
#The MIT License (MIT)
#
#Copyright © 2022 Jani Piri
#
#Permission is hereby granted, free of charge, to any per-
son obtaining a copy of this software and associated
documentation files (the “Software”), to deal in the Soft-
ware without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute,
sublicense, and/or sell copies of the Software, and to per-
mit persons to whom the Software is furnished to do so,
subject to the following conditions:
#
#The above copyright notice and this permission notice
shall be included in all copies or substantial portions of
the Software.
#
#THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS
OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

from machine import Pin,PWM,ADC,Timer
import time

#Define pins
led = Pin(25, Pin.OUT)
servo_wrist = PWM(Pin(16))
servo_fist = PWM(Pin(18))

#ADC pins on board: GP26 = ADC0, GP27 = ADC1, GP28 = ADC2
sensor_flexor = ADC(26) #Sensor to Wrist flexor muscles
GP26 pin (ADC0)
sensor_extensor = ADC(27) #Sensor to Wrist extensor muscles
GP27 pin (ADC1)
```

```
#Servo position constants, length of pulse to determine
orientation. Settings may wary.
SERVO_MID = 1500000 #1,5ms in nanosecs, servo rotor middle
SERVO_RIGHT = 1000000 #1ms, servo blade to right
SERVO_LEFT = 2000000 #2ms,  servo blade to left
servo_wrist.freq(50)
servo_fist.freq(50)

timer = Timer()
#Function to convert sampled analog-to-digital converted
number (N value) to Voltage(V)
def convert_volt(sample):
    sampled_volt = sample * 3.3 / 65536 #Calculate sampled
V from quantized number. 3,3V default reference volt, Max N
value = 65536 (16-bit ADC domain)
    return sampled_volt

#Function to blink on board led
def blink(timer):
    led.toggle()

timer.init(freq=30, mode=Timer.PERIODIC, callback=blink)
#Blink led fast to notify user that program is running

#Main program
while True:
    sample_flexor = convert_volt(sensor_flexor.read_u16())
#Read value from wrist flexor sensor and convert sample N
to volt
    sample_extensor = convert_volt(sensor_exten-
sor.read_u16()) #Read value from wrist extensor sensor and
convert sample N to volt

    if sample_flexor >= 1.3 and sample_extensor < 1:
#Flexor muscle active, rotate servo to left
        servo_wrist.duty_ns(SERVO_LEFT)
        servo_fist.duty_ns(SERVO_MID)
    elif sample_extensor >= 1.9 and sample_flexor < 1: #Ex-
tensor muscle active, rotate servo to right
        servo_wrist.duty_ns(SERVO_RIGHT)
        servo_fist.duty_ns(SERVO_MID)
    elif sample_flexor >= 1.6 and sample_extensor >= 1.2:
#Close fist motion when both muscles active
        servo_fist.duty_ns(SERVO_LEFT)
    else:
        servo_fist.duty_ns(SERVO_MID)
```

```
    time.sleep(0.001) #Wait until next sample. Faster times
gives more responsiveness on servos but Pico might crash
Pico faster
```

## Liite 3: Anturin datan tallennusohjelman lähdekoodi

```
# ------------------------------------------------------------
--------#
# Electromyography (EMG) logger for RP2040-microcontroller
v. 0.3 #
#------------------------------------------------------------
-------#
#The MIT License (MIT)
#
#Copyright © 2022 Jani Piri
#
#Permission is hereby granted, free of charge, to any per-
son obtaining a copy of this software and associated
documentation files (the “Software”), to deal in the Soft-
ware without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute,
sublicense, and/or sell copies of the Software, and to per-
mit persons to whom the Software is furnished to do so,
subject to the following conditions:
#
#The above copyright notice and this permission notice
shall be included in all copies or substantial portions of
the Software.
#
#THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS
OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
#--- Version comments --- #
# v. 0.3 includes real time Root Mean Square calculation
inside the chip - this created ~56 % less datapoints compa-
red to v. 0.1 since more computation capacity being used on
real time calculation.
# v. 0.2  signal amplitude calculation and rectifying the
signal.
# v. 0.1 included sensor reading and changing values to
volts. And writing "Time (s); Sample (N); Vout(V)" columns
to the datalog.

#Start
#Libraries
from machine import Pin,PWM,ADC,Timer
```

```
import time
import random
import math
import array

#Define pins
led = Pin(25,Pin.OUT) #On board led is GP25
#ADC pins on board: GP26 = ADC0, GP27 = ADC1, GP28 = ADC2
sensor = ADC(26) #Tell RP2040-chip that EMG-sensor is con-
nected to ADC0-pin

#Datetime
year, month, day, hour, mins, secs, weekday, yearday =
time.localtime()

timer = Timer() #Function

#--FUNCTIONS--#

#Function to blink on board led
def blink(timer):
    led.toggle()

#Function to convert sampled analog-to-digital converted
number (N value) to Voltage(V)
def convert_volt(sample):
    v_ref = 3.3 #Reference voltage of Pico 3,3V
    v_out = sample * v_ref / 65536 #Calculate sampled V
from quantized number
    return v_out

#Function to write new line in datalog.txt if datalogging
is enabled. Function takes in row array variable
def write_row(row):
    for row_item in row: #Loop the array for values
        file.write(str(row_item)) #Write individual values
to one line
        file.write(";") #Add semicolon separator between
values for MS Excel data import
    file.write("\r") #New line/"Press enter"

#Function to get median value - MIT Licence copyright (c)
2018 Roberto Colistete Junior - from https://git-
hub.com/rcolistete/MicroPython_Statistics/
def median(data):
    data = sorted(data)
    n = len(data)
    if n % 2 == 1:
        return data[n//2]
```

```
    else:
        i = n//2
        return (data[i - 1] + data[i])/2

#--- BOOT SEQUENCE ---#
#Notify user the recording is about to start, read sensor
w/o storing to file yet, calculate signal middle point and
create the datalog file

timer.init(freq=2, mode=Timer.PERIODIC, callback=blink)
#Start blink on board led twice a second to notify user
that programm is in boot sequence

#Create datalog file
log_date = "{:02d}-{:02d}-{}_{:02d}-{:02d}-{:02d}".for-
mat(day, month, year, hour, mins, secs) #Get current date
and time
random_id = str(random.randrange(9999)) #Add random number
to filename since Pico doesn't retain real time clock wit-
hout battery. Without computer the log date will be
01.01.2021
filename = str("datalog_"+log_date+"_"+random_id+".txt")
#Full filename e.g. datalog_27-01-2022_16-20-01_1234.txt
file = open(filename, "x") #Create new file data-
log_time_XXXX.txt
file.write("ID;Time (s);Sample (N);Vout (V);Amplitude
(V);Rectified (V);RMS\r") #Write first row with column
headers and add new line

boot_array = array.array("f") #Create array with floating
points to store sensor data

boot_start = time.ticks_ms() #Get current count of millise-
conds
boot_end = 3 #Give how many seconds does it take to boot
boot_elapsed = time.ticks_diff(time.ticks_ms(), boot_start)
#Get elapsed time from calculating difference from start to
running time

while boot_elapsed < boot_end: #Run as long as elapsed time
reaches to limit
    boot_elapsed = time.ticks_diff(time.ticks_ms(),
boot_start) / 1000 #Boot running time. Divide by thousand
to convert milliseconds to seconds
    value_EMG = sensor.read_u16() #Read EMG-sensor
    volt = convert_volt(value_EMG) #Call convert to volt -
function
    boot_array.append(volt)
    time.sleep(0.1)
```

```
sample_median = median(boot_array) #Get median value from
signal
boot_array = 0 #Clear boot array for not taking memory
anymore
#--- END OF BOOT SEQUENCE ---#

#--- MAIN PROGRAMM ---#
timer.init(freq=20, mode=Timer.PERIODIC, callback=blink)
#Blink led fast to notify user that programm is running
main part

start = time.ticks_ms() #Number of milliseconds passed to
this point
end = 15 #Seconds to run the sampling
elapsed_time = time.ticks_diff(time.ticks_ms(), start) #Get
starting point as 0 seconds

number_of_samples = 1 #RMS number of samples (N) value
starting from 1
sum_of_all_samples = 0 #RMS Sum of all samples (X's) value
starting from 0
sample_id = 1 #First row of datalog entry

while elapsed_time < end:
    elapsed_time = time.ticks_diff(time.ticks_ms(), start)
/ 1000 #Convert milliseconds to seconds
    sample_EMG = sensor.read_u16() #Read value from sensor
    sample_volt = convert_volt(sample_EMG) #Call con-
vert_volt(sample) function to return value in volts
    sample_amplitude = sample_volt - sample_median #Calcu-
late signal amplitudes
    rectify_sample = math.fabs(sample_amplitude) #All nega-
tive amplitudes to positive values (absolute value) to
rectify signal

    #--- Root Mean Square ---#
    #RMS = Squareroot(x^2 / N), where X is amplitude power
of two and N is number of samples
    sum_of_all_samples = sum_of_all_samples + pow(rec-
tify_sample,2) #x^2

    #Period length of RMS, after 42 samples, Start new pe-
riod and reset Sum of all samples
    if number_of_samples <= 42:
        RMS = math.sqrt(sum_of_all_samples / num-
ber_of_samples)
    else:
        number_of_samples = 1
```

```
        sum_of_all_samples = 0
    #--- End of RMS ---#

    row = [sample_id, elapsed_time, sample_EMG,
sample_volt, sample_amplitude, rectify_sample, RMS] #All
values to array
    write_row(row) #Send array to write function to add new
line with values to log file

    number_of_samples = number_of_samples + 1 #Increases
RMS N-value
    sample_id = sample_id + 1 #Id for next sample row in
datalog
file.close()
timer.deinit() #Stop led blinking
led.value(1) #Led stays on continuously to notify user that
programm is done
#End
```