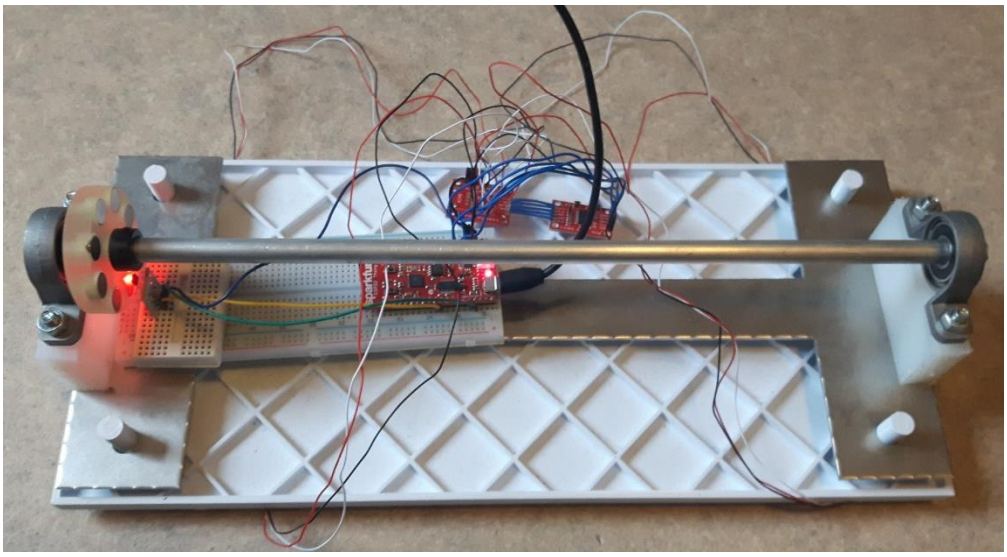
Samuel Kuikka

Sensorien kehitys älykkääseen foam roller -applikaatioon



Insinööri (AMK)

Tieto- ja viestintätekniikka

Syksy 2020

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Samuel Kuikka

Työn nimi: Sensorien kehitys älykkääseen foam roller -applikaatioon

Tutkintonimike: Insinööri (AMK), tieto -ja viestintätekniikka

Asiasanat: painesensori, hall-sensori, Bluetooth Low Energy, foam roller, Arduino, pyörimisnopeus

Tämän insinöörityön aiheena oli sensorien kehitys älykkääseen foam roller -applikaatioon, joka nimettiin Highroller Smart -applikaatioksi. Foam roller on usein vaahtomuovista tehty putkirulla, jota voidaan käyttää ennen ja jälkeen treenaamista purkamaan lihasjännitystä. Työn tilaajana oli kajaanilainen Kaswe Oy, joka tarjoaa yrityksille laite- ja ohjelmistosuunnittelua.

Highroller Smart -sovelluksen ideana oli lisätä tavalliseen foam rolleriin toiminnallisuutta ohjeistamalla käyttäjää laitteen käytössä mittaamalla käyttäjän laitteeseen aiheuttama painon ja rullan pyörimisnopeus. Applikaation käyttöliittymä toteutettiin Android-mobiilialustalle, jossa sensoridataa hyödynnettiin opastamalla käyttäjää laitteen käytössä pelimäisessä käyttöympäristössä.

Sensoridata tuli lähettää mobiililaitteeseen Bluetooth Low Energyn avulla. Niinpä sensoreiden ohjaukseen käytettiin Arduino-pohjaista ESP32 Thing -mikroprosessoripiiriä, jossa on sisäänrakennettu BLE-moduuli. Painon mittaamisessa tuli ottaa huomioon, että käyttäjä saattaa painaa akselia epäsymmetrisesti. Jotta paino saatiin mitattua luotettavasti, päädyttiin painon mittaamiseen neljällä painesensorilla wheatstonen siltakytkennässä.

Rullan pyörimisnopeuden mittaaminen toteutettiin hall-sensorin avulla. Rullan päätyyn kiinnitettiin 3D-tulostettu kiekko, johon pystyttiin kiinnittämään maksimissaan kahdeksan magneettia. Opinnäytetyössä tuli selvittää optimaalisin magneettimäärä pyörimisnopeuden mittaamiseen. Kokeen perusteella saatiin selville, että kahdeksalla magneetilla saatiin mitattua tarkin pyörimisnopeusarvo.

Insinöörityön tuloksena syntyi HighRoller Smart -laitteesta ensimmäinen prototyyppi, jonka avulla pystyttiin testaamaan BLE-yhteyden muodostusta ja sensoriarvojen lähetystä Android-applikaatioon.

KAMK • University of Applied Sciences

Abstract

Author: Samuel Kuikka

Title of the Publication: Sensor Development for Intelligent Foam Roller Application

Degree Title: Bachelor of Engineering, Information and Communications Technology

Keywords: pressure sensor, hall sensor, Bluetooth Low Energy, foam roller, Arduino, rotation speed

The subject of this Bachelor's thesis was development of sensors for an intelligent foam roller application, which was named as Highroller Smart application. A foam roller is usually a tubular roller made of foam that can be used before and after training to relieve muscle tension. The supervisor of this work was Kaswe Oy from Kajaani, which offers companies hardware and software design.

The main idea of this application was to add functionality to a standard foam roller by instructing the user in the use of the device by measuring weight and rotation speed caused by the user. User interface was planned to be implemented on the Android mobile platform, where the sensor data is utilized to guide the user in a game-like environment.

Sensor data was meant to be sent to a mobile device using Bluetooth Low Energy. Thus, an Arduino-based ESP32 Thing microprocessor circuit with a built-in BLE module was used to control the sensors. When measuring the weight, it should be noted that the user may press the shaft asymmetrically. To measure the weight reliably, it was decided to measure the weight with four pressure sensors in the Wheatstone bridge circuit.

Rotation speed measurement was implemented with a hall sensor and a magnet disc. A 3D-printed disc was attached to the end of the roll, to which a maximum of eight magnets can be attached. One part of this project was to find out the optimal number of magnets to be used. Based on the experiment, it was found that the most accurate rotation speed value was measured with eight magnets.

As a result of this thesis, the first prototype of the HighRoller Smart device was created, which was used to test BLE connection and the transmission of sensor values to the Android application.

Sisällys

1 Johdanto 1

2 Vaatimusmäärittely 2

3 Työn suoritus 3

3.1 Painon mittaus TAS606-painesensorilla 5

3.2 Painon mittaus neljällä SEN-10245-painesensorilla 8

3.3 Pyörimisnopeuden mittaus 13

3.3.1 Magneetin havaitseminen hall-sensorilla 13

3.3.2 Magneettien lukumäärän valinta 15

3.3.3 Pyörimisnopeuden mittaaminen 16

3.4 Yhdistetty sensorimittauslevy 17

4 Työn tulokset 21

4.1 Pyörimisnopeuden mittaustulokset 21

4.2 Painon mittaustulokset 21

4.3 Tulosten analysointi 22

5 Yhteenveto 23

Lähteet 24

Liitteet 1 Arduino master -koodi

Lyhenteet ja määritelmät

BLE	Bluetooth Low Energy, lyhyen kantaman langattoman tiedonsiirron protokolla
VDD	Integrated circuit power pin, Logiikan jännitelinja
VCC	Voltage Common Collector, Kytkennän käyttöjännitepinni
DAT	Data pin, Datalinja
CLK	Clock pin, Kellopulssi
GND	Ground pin, Maadoituspinni
RPM	Revolutions per minute, pyörimisnopeuden mittayksikkö
USB	Universal Serial Bus, tiedonsiirtoliitin

1 Johdanto

Tämän insinöörityön aiheena oli sensorien kehitys älykkääseen foam roller -applikaatioon, joka nimettiin Highroller Smart -applikaatioksi. Foam roller on usein vaahtomuovista tehty putkirulla, jota voidaan käyttää ennen tai jälkeen lihasharjoitusta purkamaan lihasjännitystä. Esimerkiksi selän lihaksia voidaan hieroa makaamalla lattialla ja liikkumalla edestakaisin foam rollerin päällä.

Työn tilaajana oli kajaanilainen Kaswe Oy, joka tarjoaa yrityksille laite- ja ohjelmistosuunnittelua. Highroller Smart -sovelluksen ideana oli kiinnittää hierontarulla telineeseen, josta pystyttiin mittaamaan laitteen käytön aikana siihen kohdistuva paino ja rullan pyörimisnopeus. Mitattujen sensoriarvojen avulla voidaan käyttäjää neuvoa laitteen käytössä Android-sovelluksessa. Tämän projektin tavoitteena oli toteuttaa ensimmäinen testattava prototyyppi.

Prototyypin rakenteen suunnittelussa ongelmallista oli painesensorin kiinnitys hieromalaitteen alustaan. Aluksi painon mittaaminen oli suunniteltu toteutettavaksi yhdellä painesensorilla, asettamalla se hieromarullan akselin keskipisteen alapuolelle. Ongelmana oli, että tämä jätti rakenteen kiikkeräksi ja rullaan kohdistuma paino pystyttiin kohdistamaan epäsymmetrisesti akselin keskipisteeseen nähden. Tämän takia painon mittaaminen päätettiin toteuttaa neljällä painesensorilla asettamalla ne alustan kulmiin. Paino saatiin siten mitattua neljän sensorin summana ja laitteen rakenne saatiin myös tukevammaksi.

Rullan pyörimisnopeuden mittauksessa hyödynnettiin hall sensoria, joka asennettiin lähelle akselin päätyyn kiinnitettyä magneettikiekkoa. Projektissa tuli selvittää kiekon sopivin magneettimäärä eli tutkia magneettien määrän vaikutus pyörimisnopeuden resoluutioon. Magneettien asettelussa tuli myös ottaa huomioon magneettikenttien mahdollinen päällekkäisyys, jos magneetit asetetaan liian lähelle toisiaan.

Sensoridata tuli lähettää mobiililaitteeseen Bluetooth Low Energyn avulla. Niinpä sensoreiden lukemiseen käytettiin Arduino-pohjaista ESP32 Thing -mikroprosessorikorttia, koska siinä on sisäänrakennettu BLE-moduuli. Sensoriarvot tuli keskiarvoistaa ja datapaketit tuli lähettää sekunnin välein Android-applikaatioon BLE:n kautta.

2 Vaatimusmäärittely

Tässä insinöörityössä tuli noudattaa Kaswen laitteelle asettamia vaatimusmäärityksiä. HighRoller Smartin kaksi sensoria tuli mitata rullan pyörimisnopeutta (rpm) sekä harjoittelun aikana rullaan kohdistuvaa painoa (kg). Mitattu data tuli lähettää Android-mobiililaitteeseen langattoman Bluetooth Low energy -protokollan avulla ja sensorimittaukset tuli toteuttaa Arduino-ympäristössä. Käyttäjää tuli ohjeistaa Android-mobiiliapplikaatiossa lähettämällä sensoreilla kerätty data mobiililaitteelta tietokantaan, jossa datasta analysoitiin käyttäjän kannalta oleellinen tieto.

HighRoller Smart -laitteen kohderyhmäksi rajattiin urheilijat tai kuntoilijat, jotka haluavat seurata omaa lihashuoltoharjoitteluaan. Laitetta ei suunniteltu käytettäväksi ulkona tai kosteassa/likaisessa ympäristössä, vaan sisäkäytössä, esimerkiksi kuntosaleilla tai kotiympäristössä.

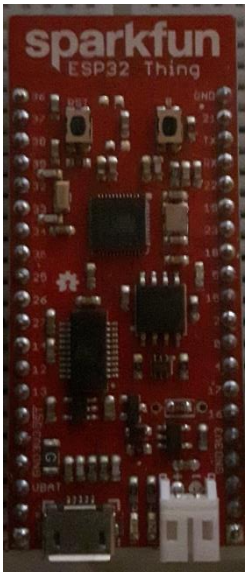
3 Työn suoritus

HighRoller Smart -sensoreiden kehitys jaettiin kahteen osaan: painesensorin testaukseen ja pyörimisnopeuden mittaamiseen. Aluksi siis paineen ja pyörimisnopeuden mittaukset suoritettiin eri koekytkennöissä ja lopuksi ne yhdistettiin samalle piirikortille. Tämän lisäksi insinöörityössä tuli myös perehtyä BLE-yhteyden muodostukseen Android-applikaation kanssa.

Tämän insinöörityön alkumittausten mikroprosessorina toimi kuvassa 1 esitelty Arduino Uno, joka pohjautuu ATmega 328P -mikroprosessoriin. Tässä kortissa on yhteensä 6 analogista ja 14 digitaalista signaalin sisääntuloporttia, siis riittävästi tämän projektin sensoreiden kalibrointikytkentöihin. Käyttöjännitteenä on 5 V tasajännite sekä 5 V:n ulostulojännite, joka sopi työssä käytettyjen komponenttien käyttöjännitteeksi. [1.]

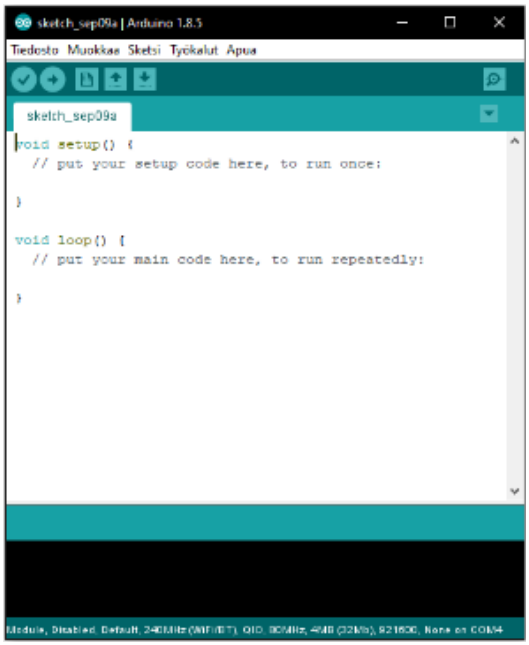
Kuva 1. Arduino Uno mikroprosessorikortti

Projektin edetessä Arduino Uno korvattiin kuvassa 2 näkyvällä SparkFun ESP32 Thing -mikroprosessorikortilla, koska se sisältää tarvittavan BLE-moduulin.



Kuva 2. Esp32 Thing -mikroprosessorialusta

Arduino Unon ohjelmointi tapahtui kuvassa 3 näkyvässä Arduino Software IDE:ssä, joka tukee C- ja C++-ohjelmointikieliä. Arduino IDE:tä voi käyttää Windows-, macOS- ja Linux-ympäristöissä.



Kuva 3. Arduino IDE -käyttöliittymä Windows-koneella

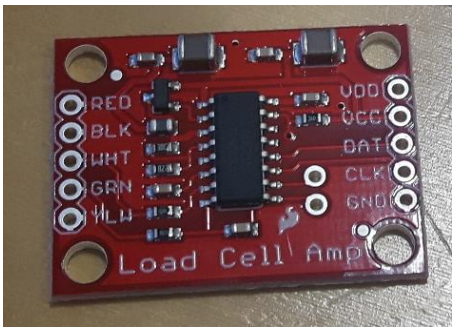
3.1 Painon mittaus TAS606-painesensorilla

Tässä insinöörityössä käytettyjen painesensorien toiminta perustuu venymäliuskan toimintaan. Paine aiheuttaa sensorin sisällä oleva metalliliuskan taipumisen ja samalla venymiseen, jolloin sen resistanssi muuttuu.

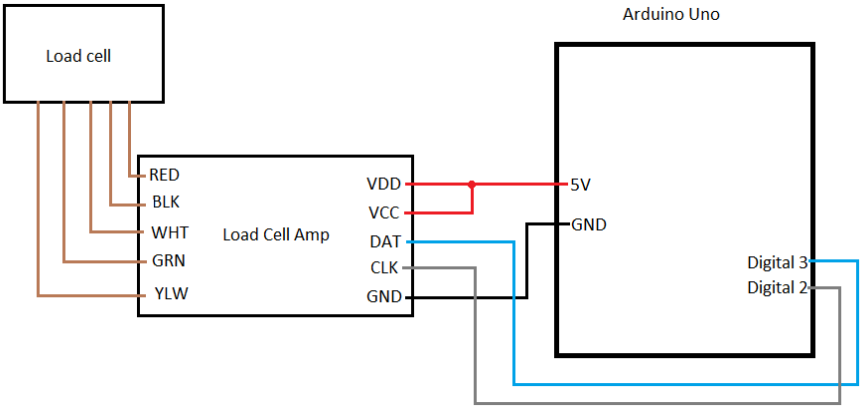
Painon mittaukseen valittiin aluksi kuvassa 4 näkyvä TAS606-painesensori, koska sillä pystyttiin mittaamaan tarkasti aina 200 kg:aan asti. TAS606-painesensori on tyypiltään kiekko, jonka sisälle on rakennettu neljä venymäliuska-anturia wheatstonen siltakytkennässä. [2.]

Kuva 4. TAS606-painesensori

Koekytkennässä ja ohjelmoinnissa käytettiin apuna SparkFun-ohjetta [3], jonka ohjeistamana painesensorin ja Arduinon välille kytkettiin kuvassa 5 näkyvä HX711-vahvistinpiirikortti vahvistamaan signaali kuvan 6 osoittamalla tavalla.

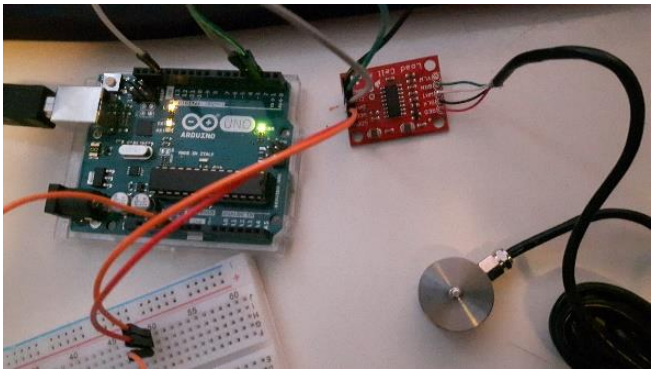


Kuva 5. HX711-vahvistinpiirikortti



Kuva 6. TAS606-painesensorin kytkentäkaavio

TAS607-painesensorin eriväriset johtimet tuli kytkeä vahvistinpiirikortin vastaaviin pinneihin. Koska Arduino Uno käyttää 5 voltin logiikkaa, pystyttiin VDD- ja VCC-linjat kytkemään 5 voltin jännitteeseen. Vahvistinpiirin signaalijohdin (DAT) kytkettiin Arduinon digitaaliporttiin 3 ja kello-pulssi (CLK) porttiin 2. Lopuksi GND eli maajohdin kytkettiin Arduinon maahan ja tuloksena saatiin kuvassa 7 näkyvä painon mittauskytkentä.



Kuva 7. Painon mittauskytkentä TAS607-painesensorilla

Paineanturin kalibroinnissa käytettiin hyväksi HX771-kirjastoa [4] ja Sparkfun-esimerkkikoodia [3]. Painesensori pystyttiin kalibroimaan muuttamalla "calibration_factor” -arvoa reaaliajassa näppäimistön – ja + -näppäimillä, kunnes paino vastasi ennalta tiettyä kappaleen painoa. Testissä kytkentä saatiin näyttämään 8 kg punnuksen oikeaa arvoa muuttujan arvolla 16000. Kuva 8 esittää tilannekuvan kalibroinnista, jossa sensorin mittaamaa arvoa tarkkailtiin serial monitorin kautta.

```
#include "HX711.h"
#define DOUT  3
#define CLK  2

HX711 scale;
float calibration_factor = 17050; //-7050 worked for my 440lb max scale setup

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press + or a to increase calibration factor");
  Serial.println("Press - or z to decrease calibration factor");

  scale.begin(DOUT, CLK);
  scale.set_scale();
  scale.tare(); //Reset the scale to 0

  long zero_factor = scale.read_average(); //Get a baseline reading
  Serial.print("Zero factor: "); //This can be used to remove the need to tare the scale. Useful in permanent scale projects.
  Serial.println(zero_factor);
}

void loop() {

  scale.set_scale(calibration_factor); //Adjust to this calibration factor

  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 1);
  Serial.print(" kg"); //Change this to kg and re-adjust the calibration factor if you follow SI units like a sane person
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

  if(Serial.available())
  {
    char temp = Serial.read();
    if(temp == '+' || temp == 'a')
      calibration_factor += 10;
    else if(temp == '-' || temp == 'z')
      calibration_factor -= 10;
  }
}
```

```
Reading: 0.1 kg calibration_factor: 16000.00
Reading: 0.1 kg calibration_factor: 16000.00
Reading: 0.1 kg calibration_factor: 16000.00
Reading: 0.3 kg calibration_factor: 16000.00
Reading: 0.5 kg calibration_factor: 16000.00
Reading: 0.8 kg calibration_factor: 16000.00
Reading: 1.5 kg calibration_factor: 16000.00
Reading: 2.4 kg calibration_factor: 16000.00
Reading: 3.3 kg calibration_factor: 16000.00
Reading: 4.0 kg calibration_factor: 16000.00
Reading: 4.7 kg calibration_factor: 16000.00
Reading: 5.3 kg calibration_factor: 16000.00
Reading: 5.8 kg calibration_factor: 16000.00
Reading: 6.2 kg calibration_factor: 16000.00
Reading: 6.6 kg calibration_factor: 16000.00
Reading: 6.8 kg calibration_factor: 16000.00
Reading: 7.1 kg calibration_factor: 16000.00
Reading: 7.2 kg calibration_factor: 16000.00
Reading: 7.3 kg calibration_factor: 16000.00
Reading: 7.4 kg calibration_factor: 16000.00
Reading: 7.4 kg calibration_factor: 16000.00
Reading: 7.4 kg calibration_factor: 16000.00
Reading: 7.4 kg calibration_factor: 16000.00
Reading: 7.4 kg calibration_factor: 16000.00
Reading: 7.5 kg calibration_factor: 16000.00
Reading: 7.7 kg calibration_factor: 16000.00
Reading: 7.7 kg calibration_factor: 16000.00
Reading: 7.8 kg calibration_factor: 16000.00
Reading: 7.8 kg calibration_factor: 16000.00
Reading: 7.8 kg calibration_factor: 16000.00
Reading: 7.9 kg calibration_factor: 16000.00
Reading: 7.9 kg calibration_factor: 16000.00
Reading: 7.9 kg calibration_factor: 16000.00
Reading: 7.9 kg calibration_factor: 16000.00
Reading: 7.9 kg calibration_factor: 16000.00
Reading: 7.9 kg calibration_factor: 16000.00
Reading: 7.9 kg calibration_factor: 16000.00
Reading: 7.9 kg calibration_factor: 16000.00
Reading: 8.0 kg calibration_factor: 16000.00
Vieritys
```

Kuva 8. TAS606-painesensorin kalibrointi

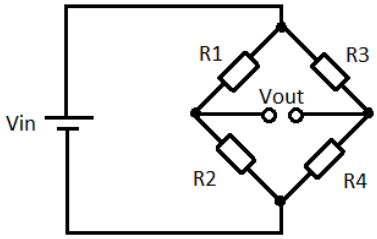
TA606-painesensorilla ei tehty enempää mittauksia, koska huomattiin, että yksi painesensori jättää hieromalaitteen rakenteen kiikkeräksi. Painon mittaus ei myöskään onnistunut luotettavasti, jos paino kohdistui jompaankumpaan akselin päätyyn. Jotta mittaustulos saataisiin tarkaksi riippumatta painon kohdistumisesta, otettiin käyttöön painon mittaus neljällä SEN-10245-painesensorilla.

3.2 Painon mittaus neljällä SEN-10245-painesensorilla

Jotta painon mittaus saatiin luotettavaksi riippumatta rullaan kohdistuvan painon kohdasta, vaihdettiin TAS606-sensorin tilalle neljä SEN-10245-sensoria. Kuvassa 9 näkyvä paineanturi mittaa vain 50 kg:aan asti, mutta neljällä voidaan mitata aina 200 kg:aan asti, koska paino jakautuu neljän anturin kesken.

Kuva 9. SEN-10245-painesensori

Wheatstonen siltakytkentä koostuu neljästä vastuksesta, joihin syötetään tietty tasajännite kuvan 10 osoittamalla tavalla.

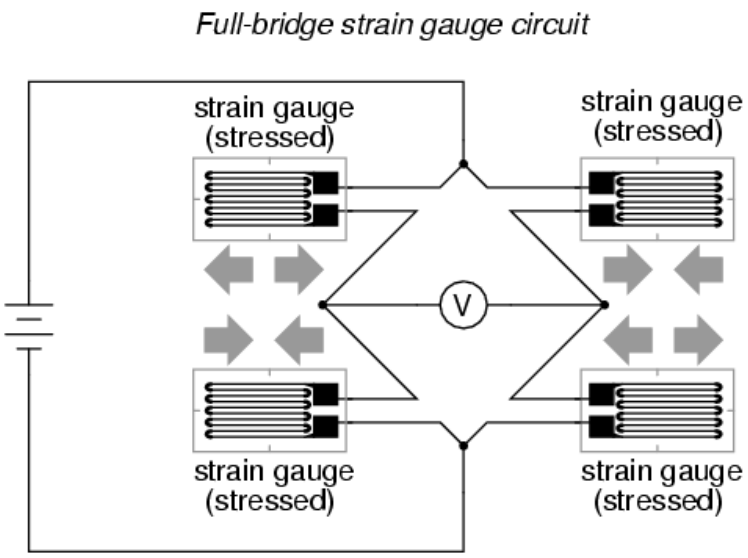


Kuva 10. Wheatstonen siltakytkentä

Kuvassa 10 oleva Vin tarkoittaa syöttöjännitettä, jonka suuruus on tiedossa. Vout on lähtöjännite, jota on tarkoitus mitata. Jos kytkentä on tasapainossa, eli R1/R2 = R3/R4, niin Vout on 0 V. Jos taas jonkin vastuksen arvo muuttuu, saadaan jännitemuutos laskettua Ohmin lain avulla seuraavasti:

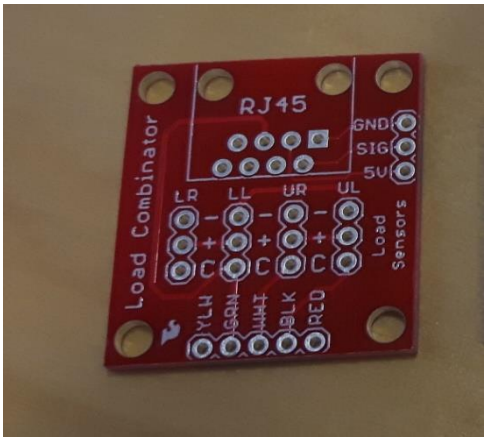
$$Vout = [(R3/(R3 + R4) – R2/(R1 + R2))] * Vin$$

Siltakytkennän vastukset korvattiin paineantureilla kuvan 11 osoittamalla tavalla.

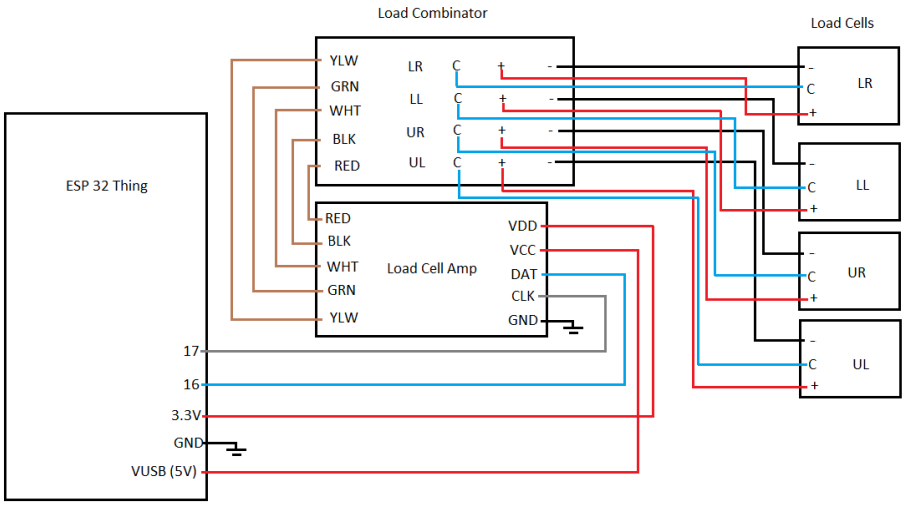


Kuva 11. Wheatstonen siltakytkentä neljällä venymäliuska-anturilla [13]

Painesensorit kytkettiin kuvassa 12 näkyvällä BOB-13878 ROHS painesensorikoontilevyllä wheatstonen siltakytkentään kuvan 13 osoittamalla tavalla. Tällä tavalla kokonaispaino saatiin neljän sensorin summana, jolloin paino pystyttiin mittaamaan luotettavasti, keskittyipä paino mihin kohtaan akselia tahansa.



Kuva 12. BOB-13878 ROHS -painesensorikoontilevy



Kuva 13. Painon mittauskytkentäkaavio neljällä SEN-10245-painesensorilla

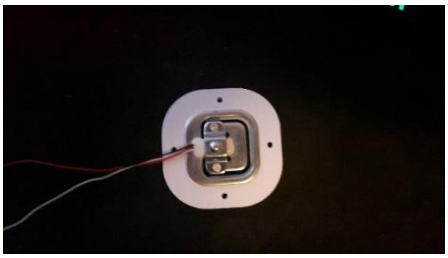
Toisin kuin Arduino Unossa, esp32 Thingissä on käyttö- ja lähdejännitteenä 3,3 volttia. Kytkennän sensorit tarvitsevat kuitenkin 5 volttia. Niinpä sensoreiden käyttöjännite kytkettiin VUSB-porttiin, joka saa virran USB:n kautta. Koska ESP32 Thing käyttää 3,3 V logiikkaa, tuli vahvistinpiirin VDD-linja kytkeä 3,3 volttiin.

Painesensorikytkentää tehdessä tuli erityisesti ottaa huomioon painesensoreiden johdinten väritys. Jännitejohdin on valkoinen, signaali punainen ja maajohdin on musta. Paineanturien johdinten värikoodi voi vaihdella, joten varminta on mitata vastukset kolmen johtimen kesken. Signaali- ja maajohtimen välillä oleva vastus on noin kaksi kertaa suurempi kuin maajohtimen ja jännitejohtimen välillä. Mikäli signaali- ja virransyöttöjohtimet kytketään väärinpäin, heittelehtivät sensoriarvot. Jos puolestaan maa- ja virransyöttölinjat kytketään ristiin, ovat sensoriarvot negatiivisia. [3.]

Painesensorit kalibroitiin kuntosalilla asettamalla sensorit henkilövaa'an jalkojen alle. Sensorien asettelussa tuli noudattaa kuvassa 13 esitettyä järjestystä. Esimerkiksi LR (lower right) -anturi tuli asettaa puntarin alaoikeaan nurkkaan. Puntarin päälle asetettiin eripainoisia punnuksia ja painesensorikytkennän arvoja verrattiin puntarin arvoihin. Kuvissa 14 ja 15 on näkyvillä tilannekuva painesensorien kalibroinnista kuntosalilla.

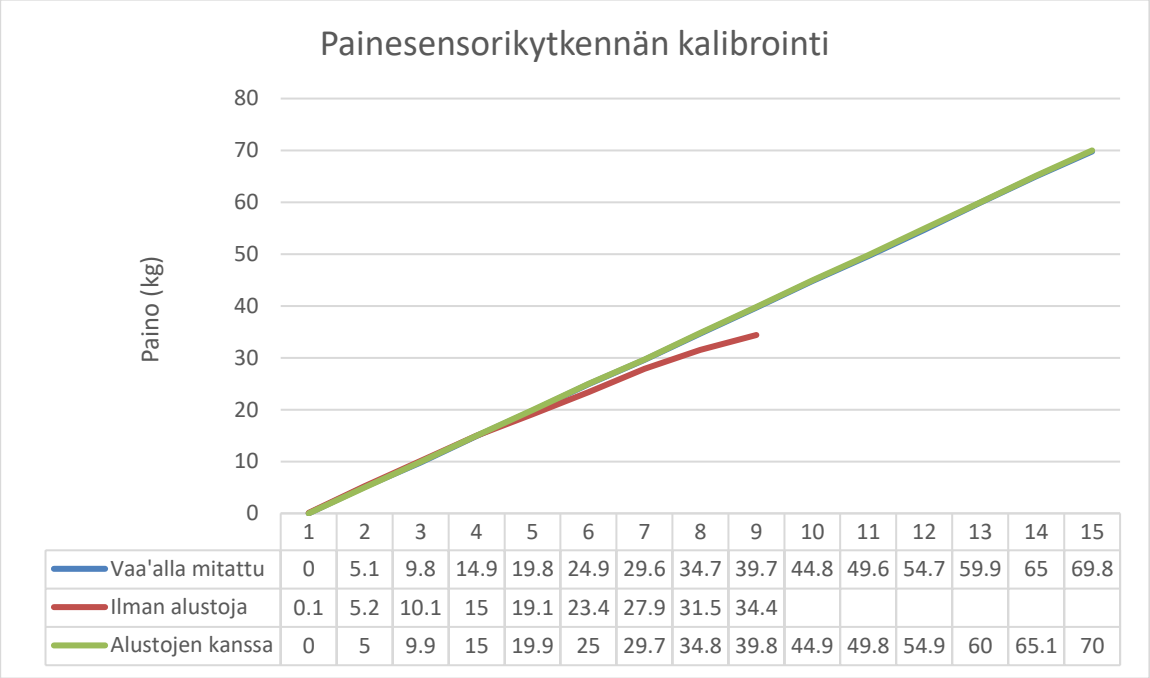
Kuvat 14 ja 15. SEN-10245-paineanturikytkennän kalibrointi kuntosalilla

Mittausten edetessä huomattiin, että painoarvot alkoivat heittämään puntarin arvoista aina enemmän, mitä suurempi paino oli kyseessä. Tämä johtui siitä, että paineanturin keskiosa ei päässyt taipumaan, vaan sensorin reunan alle täytyi laittaa koroketta. Niinpä sensorien alle tilattiin kuvissa 16 ja 17 näkyvät 3D-tulostetut alustat, jotka mahdollistivat sensorin keskiosan taipumisen kuorman alla. [5]



Kuvat 16 ja 17. SEN-10245-painesensorin 3D-tulostettu alusta

Mittaukset toistettiin 3D-tulostettujen alustojen kanssa, jolloin saatiin hyvin tarkkoja arvoja. Mittausten 1 ja 2 tulokset koottiin alla olevaan kuvaan 18.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vaa'alla mitattu	0	5.1	9.8	14.9	19.8	24.9	29.6	34.7	39.7	44.8	49.6	54.7	59.9	65	69.8
Ilman alustoja	0.1	5.2	10.1	15	19.1	23.4	27.9	31.5	34.4						
Alustojen kanssa	0	5	9.9	15	19.9	25	29.7	34.8	39.8	44.9	49.8	54.9	60	65.1	70

Kuva 18. Painesensorikytkennän kalibrointi ilman 3D-tulostettua alustaa ja mittaukset alustan kanssa

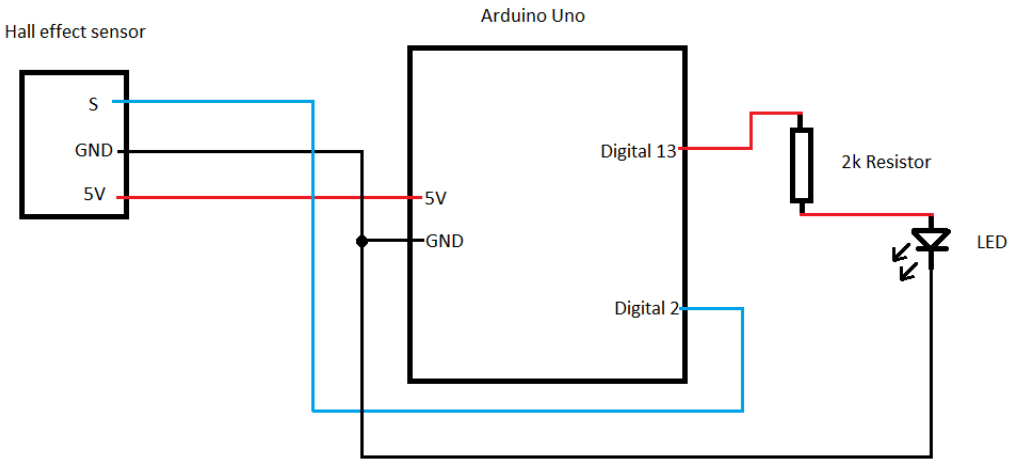
3.3 Pyörimisnopeuden mittaus

Hall-anturin toiminta pohjautuu hall-ilmiöön, jossa magneettikenttä aiheuttaa elektronien kulkusuunnan muutoksen johtimessa, joka tuodaan lähelle magneettia kohtisuoraan magneettikenttään nähden. [6.]

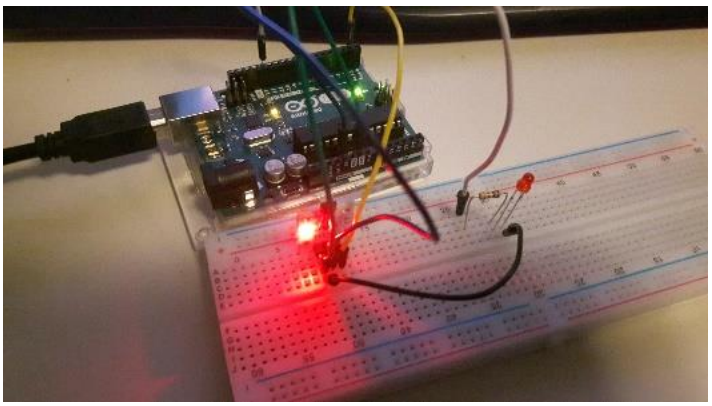
Projektissa käytettiin VMA313-hall-sensorimoduulia. Hall-sensorin toimintaan perehdyttiin ensiksi tekemällä koekytkentä, jossa ledin syttymistä ohjattiin hall-anturin avulla. Seuraavaksi pyrittiin määrittämään paras magneettien lukumäärä pyörimisnopeuden mittaamiseen. Lopuksi tehtiin varsinainen kytkentä pyörimisnopeuden mittaamiseen edellisten kokeiden perusteella.

3.3.1 Magneetin havaitseminen hall-sensorilla

Ensimmäisen testin päämääränä oli magneetin havaitseminen Arduino Unolla. Testikytkennässä led sytytettiin aina, kun magneetti asetettiin hall-sensorin lähelle. [7.] Hall-anturin signaalijohdin (S) kytkettiin Arduino Unon digitaaliporttiin 2 kuvan 19 osoittamalla tavalla. Lisäksi porttiin 13 kytkettiin led 2k ohmin virranrajoitusvastuksen kanssa ja tuloksena syntyi kytkentä, joka on näkyvissä kuvassa 20.

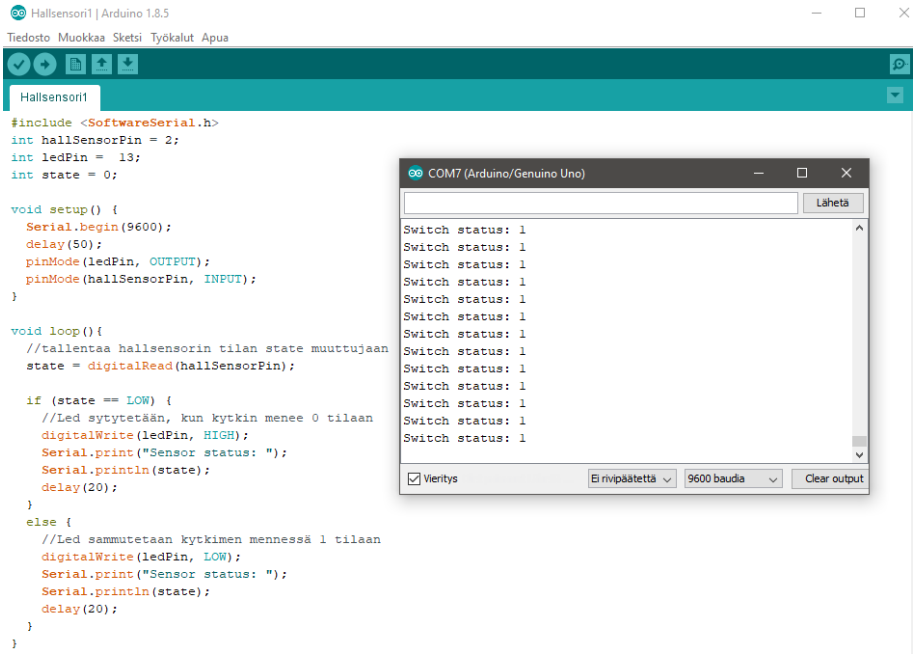


Kuva 19. Hall-sensorin testikytkentäkaavio



Kuva 20. Hall-anturin testikytkentä

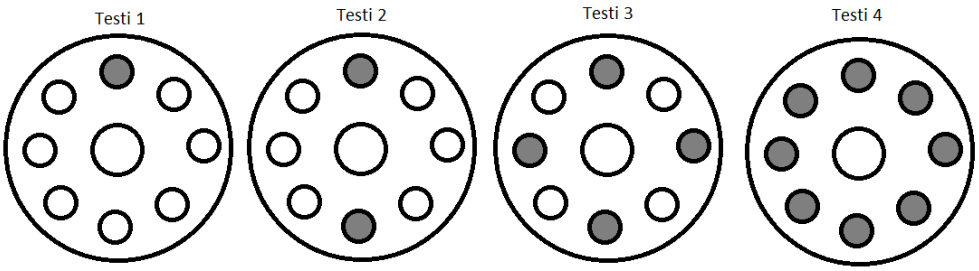
Hall-anturi toimi kytkimenä ledille, joka syttyi aina, kun magneetti tuotiin lähelle hall-anturia. Lisäksi kuvassa 21 olevassa testikoodissa tulostetaan hall-anturin tila. Kytkin on ykköstilassa, jos magneettia ei ole havaittu (led on pois päältä). Kun magneetti havaitaan, menee kytkin nollatilaan ja led sytytetään.



Kuva 21. Hall-sensorin testikoodi

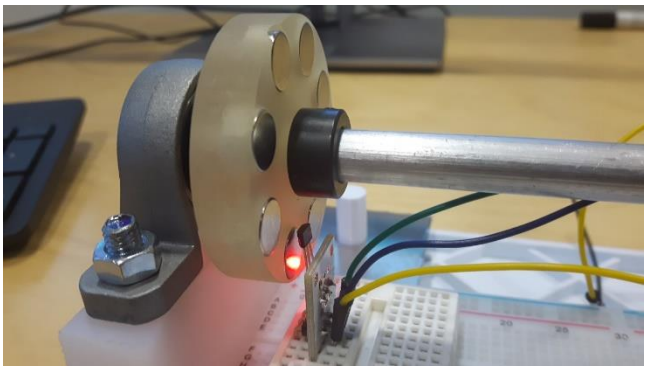
3.3.2 Magneettien lukumäärän valinta

Seuraavaksi pyrittiin selvittämään 3D-tulostettuun kiekkoon kiinnitettyjen magneettien määrän vaikutus mittaustulokseen. Tässä kokeessa hall-sensori toimi edelleen kytkimenä ledille ja kytkentä oli sama kuin edellisessä testissä. Kiekkoon lisättiin magneetteja aina kiekon vastakkaisille puolille ja kytkimen toiminnan luotettavuutta tarkkailtiin magneettien määrän lisääntyessä. Kokeita tehtiin yhteensä neljä kappaletta, joissa magneetit aseteltiin kuvan 22 osoittamalla tavalla. Kaikki magneetit tuli asettaa samoin päin hall-sensoria kohden.



Kuva 22. Hall-sensoritestit yhdellä, kahdella, neljällä ja kahdeksalla magneetilla

Kokeista havaittiin, että kytkentä toimi luotettavasti aina kahdeksaan magneettiin asti, mutta kahdeksannen kohdalla tuli kiekko asettaa hyvin lähelle sensoria, noin kahden millimetrin päähän, jotta magneettikentät eivät sulaudu yhteen. Sensorin oikea asettelu on näkyvissä kuvassa 23.



Kuva 23. Hall-sensorin testaus kahdeksalla magneetilla

3.3.3 Pyörimisnopeuden mittaaminen

Kolmannen testin päämääränä oli kiekon pyörimisnopeuden mittaaminen. Riippuen käytettävien magneettien lukumäärästä havaitaan yhden kiekon pyörähdyksen aikana 1,2,4 tai 8 signaalia. Signaalipulssien avulla voidaan selvittää akselin kierrostaajuus laskemalla yhden sekunnin aikana havaitut signaalit yhteen ja jakamalla saatu tulos käytettyjen magneettien lukumäärällä. Lopuksi kierrostaajuus saadaan muutettua pyörimisnopeudeksi kertomalla tulos 60:llä. [7.] Alla olevassa kuvassa 24 on tilannekuva pyörimisnopeuden mittauksesta neljällä magneetilla.

RPM = havaitut magneetit sekunnissa * 60 / magneettien lukumäärä

```
// read RPM
volatile int rpmcount = 0;//see http://arduino.cc/en/Reference/Volatile
int rpm = 0;
unsigned long lastmillis = 0;

void setup(){
Serial.begin(9600);
attachInterrupt(0, rpm_fan, FALLING);//interrupt zero (0) is on pin two(2).
}

void loop(){

if (millis() - lastmillis == 1000){ /*Uptade every one second*/

detachInterrupt(0); //Disable interrupt when calculating

rpm = rpmcount * 15; /*Convert frecuency to RPM, note: *60* works for one interruption per full rotation.
                       For two interrups per full rotation use rpmcount * 30.*/

Serial.print("RPM =\t"); //print the word "RPM" and tab.
Serial.println(rpm); // print the rpm value.

rpmcount = 0; // Restart the RPM counter
lastmillis = millis(); // Uptade lasmillis
attachInterrupt(0, rpm_fan, FALLING); //enable interrupt
}
}

void rpm_fan(){ /* this code will be executed every time the interrupt 0 (pin2) gets low.*/
rpmcount++;
}
```

```
COM7 (Arduino/Genuino Uno)
RPM =   0
RPM =   0
RPM =   0
RPM =   0
RPM =   30
RPM =   120
RPM =   150
RPM =   150
RPM =   285
RPM =   390
RPM =   360
RPM =   315
RPM =   405
RPM =   315
RPM =   195
RPM =   15
RPM =   0
RPM =   0
RPM =   0
RPM =   0
RPM =   0
Vieritys   Ei rivipäätettä   9600 baudia
```

Kuva 24. Pyörimisnopeuden mittaus neljällä magneetilla

Kokeen tulosten perusteella havaittiin, että yhdellä magneetilla rpm saatiin 60 kertoimena, kahdella magneetilla 30 ja neljällä 15 kertoimena. Kahdeksalla magneetilla saatiin tarkin arvo 7,5 kertoimella. Pyörimisnopeuden mittaamiseen valittiin siis käytettäväksi kahdeksaa magneettia, jolloin lopullinen lakentakaava saatiin seuraavasti:

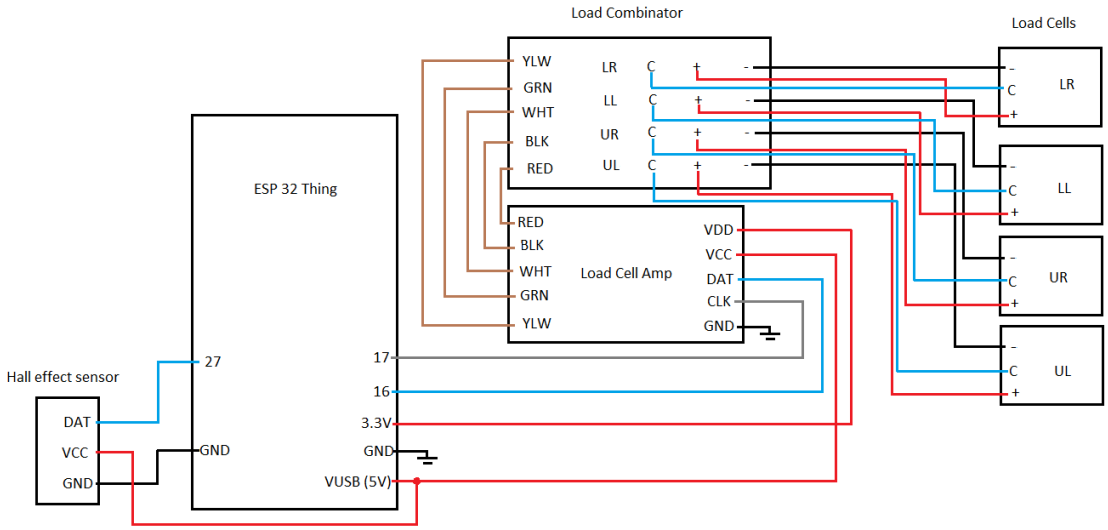
RPM = havaitut magneetit sekunnissa * 60 / magneettien lukumäärä

= havaitut magneetit sekunnissa * (60/8)

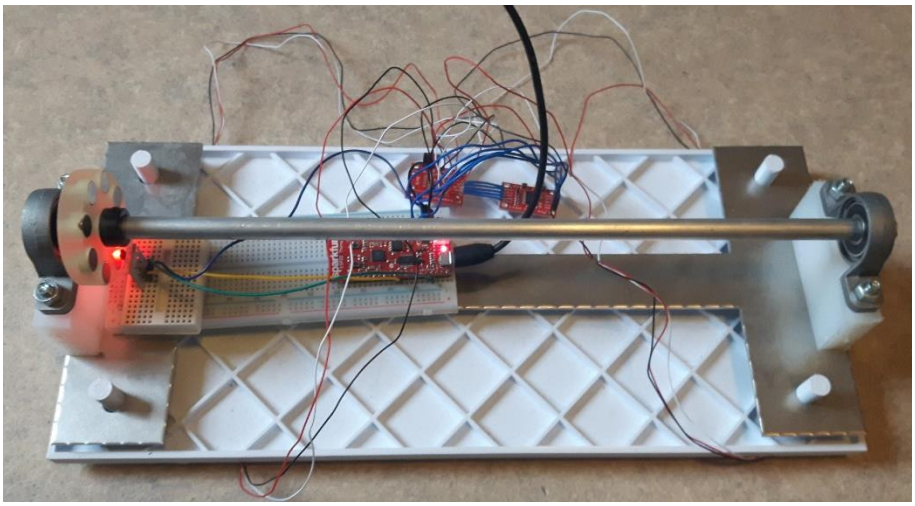
= havaitut magneetit sekunnissa * 7,5.

3.4 Yhdistetty sensorimittauslevy

Painon ja pyörimisnopeuden mittausalustana käytettiin SparkFun ESP32 Thing -alustaa, koska se on Arduino-pohjainen ja se sisältää sisäänrakennetun BLE-moduulin. Kuvassa 25 on näkyvillä yhdistetyn sensorimittauslevyn kytkentäkaavio ja kuvassa 26 puolestaan sensorimittauslevy, joka on asennettu HighRoller Smart -laitteen runkoon.



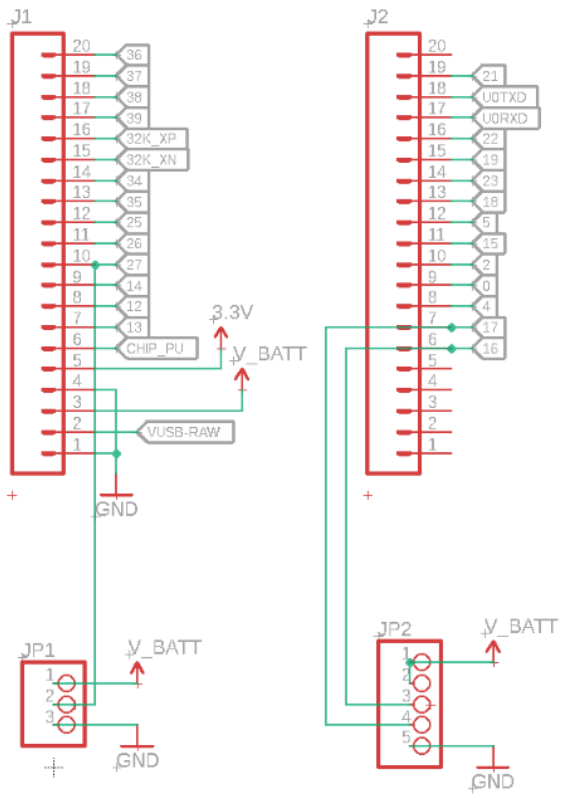
Kuva 25. Yhdistetyn sensorimittauslevyn kytkentäkaavio



Kuva 26. Yhdistetty sensorimittauslevy asennettuna HighRoller Smart -laitteen runkoon

Arduino master -koodi saatiin yhdistämällä aikaisempien kokeiden paine- ja pyörimisnopeudenmittauskoodit yhteen. Android-sovelluksen vaatimuksena oli, että sensoriarvot tuli keskiarvoistaa ja lähettää samassa muuttujassa puolipisteellä erotettuna BLE-yhteyden kautta sekunnin välein. BLE-yhteyttä testattiin ensin Nordicsemi Android -applikaatiossa [8] ja myöhemmin varsinaisessa HighRoller Smart -applikaatiossa.

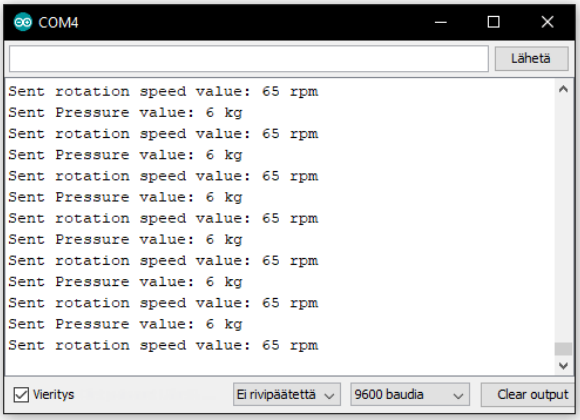
Sensorimittauslevy suunniteltiin myös Eaglessä, joka on Autodeskin kehittämä ohjelmisto piirilevyn suunnitteluun. Sensorilevy on esitettynä kuvassa 27, jossa J1 ja J2 ovat esp32 Thing -piirin jalkojen kytkentäportit. JP1 on liitin hall-anturille ja JP2 puolestaan painesensoreille. Koska esp32 Thing -kortin käyttöjännite antaa vain 3,3 volttia ja kytkennän sensorit vaativat 5 voltin käyttöjännitteen, tulee systeemiin kytkeä V_BATT-liittimen kautta akku. Kytkennän esp32 Eagle -tiedostot saatiin ladattua SparkFunin sivuilta [10].



Kuva 27. Eaglellä tehty piirikortti sensorimittauslevystä

BLE-yhteyden muodostukseen käytettiin esp32:n kirjastoa BLE2902.h [11] ja sensoriarvojen keskiarvoistukseen käytettiin Smoothed.h-kirjastoa [12]. Arduino master -koodi löytyy tämän insinöörityön liitteenä ja kuvassa 28 näkyy sensoriarvojen lähetys Android-applikaatioon.

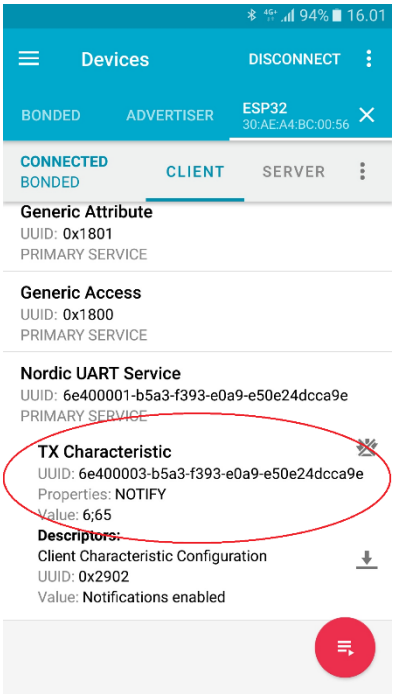
```
//BLE characteristics for pressure and rpm values
BLECharacteristic* pCharacteristic;

bool deviceConnected = false;
int pressureValue = 0;


// Create instance of the class to use
Smoothed <int> mySensor;

class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    };

    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};

void hallSensorSetup(){
  //hallsensor interrupt pin 27
  attachInterrupt(27, rpm_counter, FALLING);
```

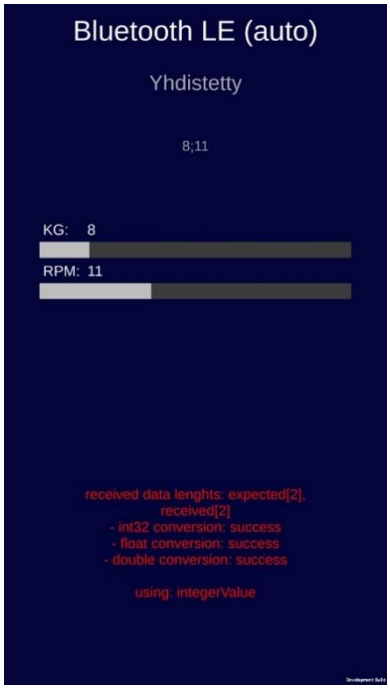


Kuva 28. Painon ja pyörimisnopeuden lähetys Android-applikaatioon

BLE-yhteyden koodin pohjana käytettiin Arduino IDE:stä löytyvää ”Ble_server”-esimerkkikoodia, joka latautui samassa esp32-kirjaston kanssa. BLE-tietoliikenteessä tieto liikkuu char-taulukoissa TX characteristiceissa. Aluksi sensoriarvot lähetettiin omissa characteristiceissa, mutta myöhemmin Android-sovellusta testatessa tuli kuitenkin ilmi, että applikaatio löysi vain ensimmäisen characteristicin. Tämän korjattiin lähettämällä molemmat sensoriarvot samassa characteristicissa puolipisteellä erotettuna. Datapakettien lähetyksen välille laitettiin 1000 ms viive, koska tieto haluttiin applikaation päässä vastaanottaa sekunnin välein.

Alla olevissa kuvissa 29 ja 30 näkyy BLE-yhteyden testaus aluksi Nordicsemi-sovelluksessa, jossa lähetetty painoarvo on 6 kg ja pyörimisnopeus 65 rpm. Kuvassa 28 on BLE-yhteyden automaattista yhdistystä varten tehty testiapplikaatio, jossa sensoriarvot on erotettu samasta muuttujasta ja sensoriarvojen muutos esitetään graafisesti palkeilla.





Kuvat 29 ja 30. BLE-yhteyden testaus nRF Connect for Mobile -sovelluksessa sekä HighRoller Smart -testisovelluksessa

4 Työn tulokset

Tämän insinöörityön tavoitteena oli selvittää keino painon mittaamiseen mahdollisimman luotettavasti paineantureilla sekä löytää keino pyörimisnopeuden mittaamiseen hall-sensorin avulla. Näihin tavoitteisiin pyrittiin pääsemään jakamalla työ ensin pienempiin osiin ja lopuksi tehtyjen havaintojen pohjalta työ koottiin yhdeksi kokonaisuudeksi.

4.1 Pyörimisnopeuden mittaustulokset

Pyörimisnopeuden mittaamiselle oli oleellista selvittää, miten magneettien määrä vaikuttaa mittaustulokseen, koska mittauksen resoluutio on sitä tarkempi, mitä enemmän magneetteja käytetään. Työssä käytettiin akselin päähän kiinnitettyä 3D-tulostettua kiekkoa, johon pystyi maksimissaan kiinnittämään kahdeksan magneettia. Jos käytettiin kaikkia kahdeksaa magneettia, niin magneettien väliin ei jäänyt paljoa tilaa. Niinpä tuli selvittää pystyykö hall-sensori erottamaan magneetit toisistaan, jos ne ovat näin lähekkäin.

Tämän kokeen tuloksena oli, että kaikkia kahdeksaa magneettia pysytettiin käyttämään, mutta hall-sensori tuli kiinnittää noin 1–2 millimetrin päähän kiekosta.

4.2 Painon mittaustulokset

Painon mittaus oli suunniteltu aluksi toteutettavaksi vain yhdellä painesensorilla. Tämä kuitenkin havaittiin ongelmalliseksi laitteen rakenteelliselle vakaudelle, jolloin siirryttiin painon mittaamiseen neljän painesensorin summana. Painesensorikytkentä pyrittiin kalibroimaan asettamalla sensorit henkilövaa’an alle, jolloin painesensoriarvoja voitiin verrata vaa'an arvoihin.

Kokeen edetessä huomattiin sensoriarvojen alkavan heittää todellisesta mitä enemmän painoa käytettiin. Tämä korjattiin 3D-tulostetuilla sensorin alustoilla, jotka mahdollistivat paremman sensorin keskiosan taipumisen. Mittaukset suoritettiin uudelleen, jolloin saatiin täysin samoja painoarvoja kuin vaaka.

4.3 Tulosten analysointi

Tässä insinöörityössä mitattujen pyörimisnopeusarvojen paikkansapitävyyttä ei ehditty vertaamaan jollakin toisella mittausmenetelmällä saatuihin arvoihin. Painon mittaamiselle suoritettiin kalibrointimittaukset vertaamalla paineanturiarvoja henkilövaa'an arvoihin. Kalibroinnin jälkeen sensoriarvot vastasivat tarkasti vaa'an arvoja, kun käytettiin sensorin alustoja.

5 Yhteenveto

Tämän insinöörityön aiheena oli tutustua painon mittaukseen paineanturin avulla ja pyörimisnopeuden mittaamiseen hall-sensorilla sekä BLE-yhteyden muodostamiseen Android-applikaatioon. Projektin keskeinen tavoite saatiin täytettyä, eli projektin tuloksena syntyi ensimmäinen prototyyppi, jonka avulla voitiin testata HighRoller Smart -laitetta ja Android-sovellusta.

Jatkokehitysmahdollisuuksina ovat piirilevyn valmistus ja kotelointi, sensoreiden kiinnityksen miettiminen HighRoller Smart -laitteen alustaan sekä pyörimisnopeuden mittaamisen kalibrointi.

Lähteet

(1) Arduino.cc. Arduino Uno rev3. Saatavilla: https://store.arduino.cc/arduino-uno-rev3. Luettu 2.4.2019.
(2) SparkFun Electronics. Load Cell - 200kg, Disc (TAS606). Saatavilla: https://www.spark-fun.com/products/13332. Luettu 4.5.2019.
(3) SparkFun Electronics. Load Cell Amplifier HX711 Breakout Hookup Guide. Saatavilla: https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide. Luettu 5.5.2019.
(4) Github.com. HX711.h. Saatavilla: https://github.com/bogde/HX711/archster.zip. Luettu 6.4.2020.
(5) Thingiverse.com. Greg Powell. 50Kg Load Cell Bracket for Digital Scale. Saatavilla: https://www.thingiverse.com/thing:2274593. Luettu 10.4.2020.
(6) Wikipedia.org. Hall-ilmiö. Saatavilla: https://fi.wikipedia.org/wiki/Hall-ilmi%C3%B6 . Luettu 13.9.2020.
(7) Academicfox.com. Arduino hall-effect sensor measuring speed of rotation. Saatavilla: http://academicfox.com/arduino-hall-effect-sensor-measuring-speed-of-rotation/. Luettu 15.6.2019.
(8) Google Play. Nordic Semiconductor ASA. nRF Connect for mobile. Saatavilla: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=fi. Luettu 15.3.2020.
(9) esp32 pinout reference. Saatavilla: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/. Luettu 15.8.2019.
(10) Eagle files for esp32 Thing. Saatavilla: https://cdn.sparkfun.com/assets/learn_tutorials/5/0/7/esp32-thing-v10.zip. Luettu 15.9.2019.
(11) Github.com. Arduino core for ESP32. Saatavilla: https://github.com/espressif/arduino-esp32. Luettu 20.3.2020.
(12) Github.com.Smoothed.h library. Saatavilla: https://github.com/MattFryer/Smoothed. Luettu 27.3.2020.
(13) Getting Started with ESP32 Bluetooth Low Energy (BLE) on Arduino IDE. Saatavilla: https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/. Luettu 29.3.2020.
(14) E-book. Edward Raden. Hall-Effect Sensors. 2nd ed. ed. US: Newnes; 2006.

Liitteet

```
#include <BLEDevice.h>

#include <BLEServer.h>

#include <BLEUtils.h>

#include <BLE2902.h>

#include "HX711.h"//Pressuresensor pinout and library

#include <Smoothed.h> //For calculating smoothed rpm values https://github.com/MattFryer/Smoothed


//Pressure sensor pinout

#define DOUT 16

#define CLK  17


#define SERVICE_UUID        "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"

#define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"


//***************Hallsensor variables*************************

volatile int rpmcount = 0;

int rpm = 0;

int smoothedRPM;
```

```
//***************Pressuresensor variables*************************

HX711 scale;

float calibration_factor = 16000; //16000 worked for 8kg weight



//BLE characteristics for pressure and rpm values

BLECharacteristic* pCharacteristic;



bool deviceConnected = false;

int pressureValue = 0;



Smoothed <int> mySensor;



class MyServerCallbacks: public BLEServerCallbacks {

   void onConnect(BLEServer* pServer) {

     deviceConnected = true;

   };



   void onDisconnect(BLEServer* pServer) {

     deviceConnected = false;

   }

};
```

```
void hallSensorSetup(){

  //hallsensor interrupt pin 27

  attachInterrupt(27, rpm_counter, FALLING);



  // Initialise rpm value store. We want this to be the simple average of the last 2 values.

  // Note: The more values you store, the more memory will be used.

  mySensor.begin(SMOOTHED_AVERAGE, 2);

}



void rpm_counter() { // this code will be executed every time the interrupt pin gets low

  rpmcount++;

}



void pressureSensorSetup(){

  //pressure sensor settings

  scale.begin(DOUT, CLK);

  scale.set_scale();

  scale.tare(); //Reset the scale to 0



  long zero_factor = scale.read_average(); //Get a baseline pressure reading

}
```

```
void BLEServerSetup(){

  // Create the BLE Device

  BLEDevice::init("ESP32");



  // Create the BLE Server

  BLEServer *pServer = BLEDevice::createServer();

  pServer->setCallbacks(new MyServerCallbacks());



  // Create the BLE Service

  BLEService *pService = pServer->createService(SERVICE_UUID);



  // Create BLE Characteristic

  pCharacteristic = pService->createCharacteristic(

                    CHARACTERISTIC_UUID_TX,

                    BLECharacteristic::PROPERTY_NOTIFY

                  );

  //   https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.de-
scriptor.gatt.client_characteristic_configuration.xml

  // Create BLE Descriptors

  pCharacteristic->addDescriptor(new BLE2902());
```

```
  // Start the service
  pService->start();

  // Start advertising
  pServer->getAdvertising()->start();
  Serial.println("Waiting a client connection to notify...");
}

float getRPM(){
  //Disable interrupt when calculating
  detachInterrupt(27);

  rpm = rpmcount * 7, 5; /*Convert frecuency to RPM, note: *60* works for one interruption per full rotation. For two interrups per full rotation use rpmcount * 30.*/

  // Add the new value to sensor value stores
  mySensor.add(rpm);

  // Get the smoothed values
  smoothedRPM = mySensor.get();

  rpmcount = 0; // Restart the RPM counter
```

```
  attachInterrupt(27, rpm_counter, FALLING); //enable interrupt


  return smoothedRPM;
}


float getPressure(){
  scale.set_scale(calibration_factor);
  pressureValue = scale.get_units();


  return pressureValue;
}


float BLEClient(){
  if (deviceConnected) {
    char txPressureString[8];
    char txRPMString2[8];
    char txCombinedString3[8];


    String combinedSensors = String(pressureValue) + ";" + String (smoothedRPM);


    //conversion of txValues
    dtostrf(pressureValue, 1, 2, txPressureString);
```

```
    dtostrf(smoothedRPM, 1, 2, txRPMString2);


    combinedSensors.toCharArray(txCombinedString3,8);


    //Setting the value to the characteristic
    pCharacteristic->setValue(txCombinedString3);


    //Notify the connected client
    pCharacteristic->notify();


    Serial.print("Sent Pressure value: " + String(txPressureString));
    Serial.println(" kg");
    Serial.print("Sent rotation speed value: " + String(txRPMString2));
    Serial.println(" rpm");
    delay(1000); //1 sec delay, bluetooth stack will go into congestion, if too many packets are sent
  }
  else{
    Serial.println("Disconnected");
    delay(1000);
  }
}
```

```
void setup() {

  // put your setup code here, to run once:

  Serial.begin(9600);



  hallSensorSetup();

  pressureSensorSetup();

  BLEServerSetup();

}

void loop() {

  // put your main code here, to run repeatedly:

  getRPM();

  getPressure();

  BLEClient();

}
```