Petri Mikkonen

# Arduinon käyttö teollisuudessa

Opinnäytetyö

Insinööri (AMK)

Sähkö- ja Automaatiotekniikka

2021

| Tekijä/Tekijät | Tutkintonimike | Aika |
|---|---|---|
| Petri Mikkonen | Insinööri (AMK) | Toukokuu 2021 |
| **Arduinon käyttö teollisuudessa**<br><br>Arduinon käyttäminen kunnossapidon ennakoivaan huoltoon | | 29 sivua<br>4 liitesivua |
| **Toimeksiantaja**<br><br>Sinebrychoff Oy / P & H Mikkonen Oy | | |
| **Ohjaaja**<br><br>Jyrki Liikanen | | |


**Tiivistelmä**

Tämä opinnäytetyö käsittelee Arduinon käyttöä kunnossapidon työkaluna teollisuudessa. Teollisuudessa kunnossapidon vastuulla on tuotantolaitteiden ylläpito ja vikatilanteiden selvitys. Vikatilanteet aiheuttavat tuotantokatkoksia ja taloudellisia menetyksiä.
Työkalun tarkoituksena oli tallentaa mitattuja arvoja talteen tuotantolaitteilta. Tallennettuja arvoja voitiin vertailla aiempiin mittauksiin mahdollisten muutosten havaitsemiseksi.

Vikatilanteiden ja laiterikkojen ennakoinnin kehittäminen häiriöttömään tuotantoon vaatii erillisen tiedonkeruulaitteiston, mikä on erillään tuotannon järjestelmistä. Tuotantolaitteiden toimiessa moitteettomasti saadaan vertailudataa häiriötilanteiden ennakointiin. Laitteella seurataan muutosta laitteiston kunnossa. Käytöksen muuttuessa voidaan ennakkohuoltosuunnitelmaa parantaa ja kehittää oikeaan suuntaan.

Laitteisto tässä opinnäytetyössä seurasi kuljetinlinjojen momenttikäyttäytymistä sekä lämpötilamittauksia laitteistoissa. Mittauksen perusteena oli milliampeeriviestin hyödyntäminen taajuusmuuttajalta momenttia varten sekä lämpötila-anturin arvojen tallentamista sd-kortille. Käyttäytymistä seurataan projektissa pitkään, jotta vartailudataa saadaan eri tuotteille. Häiriötilanteista saadun datan perusteella voidaan tulkita, olisiko häiriö ollut estettävissä.

Jo kerätyllä aineistolla on havaittu hyötyjä ongelmakohtien parantamisessa. Historia-ainestoa on kerättävä talteen, jolloin muutoksien havaitseminen on helpompaa.
Tulevaisuudessa kerätyllä datalla voidaan luoda ennakkohälytyksiä laiterikkojen estämiseksi. Arduino osoittautui soveltuvan hyvin projektiin sekä lisäsi mahdollisuudet käyttää laitetta laajemminkin tuotannossa.



**Asiasanat**

arduino, teollisuus, ennakkohuolto, kunnossapito

Kaakkois-Suomen ammattikorkeakoulu

| Author (authors) | Degree | Time |
|---|---|---|
| Petri Mikkonen | Bachelor of Engineering | May 2021 |

**Arduino in industrial environment**

Application of Arduino in predictive maintenance

29 pages
4 pages of appendices

**Commissioned by**

Sinebrychoff Oy / P & H Mikkonen Oy

**Supervisor**

Jyrki Liikanen


**Abstract**

The objective of this thesis was to study how Arduino can be utilised as a maintenance tool to help and prevent breakdowns in industry. Maintenance is responsible for fixing malfunctions and maintaining devices and production machines. Breakdowns increase costs and time loss in the production. Arduino is used to record measurements from production machines.

In order to prevent breakdowns and improve preventive maintenance, separate tracking system is needed to follow the condition of production machines. When production is undistorted, collected data will show machines' base level. When data will show changes, preventive maintenance can be adjusted to the right level and do the correct actions to prevent breakdown.

Developed device of this thesis tracks a behaviour of torque in product conveyers. Temperatures are tracked. Basic idea is to use current loop information from frequency converter. Collected information is stored to SD-card. Long term data collection is necessary to get more information on torque when production is undistorted. In the future, collected breakdown data can be used to create preventive alarms, thus avoiding large breakdowns.



**Keywords**

Arduino, data logger, maintenance tool

**Esipuhe**

Opintojen suorittaminen osittain poikkeusoloissa toi mukavasti haastetta opintoihin.  Elektroniikka ja sähkötekniikka on kiehtonut yläasteikäisestä lähtien, joten selkeästä urapolusta sähkön parissa voidaan puhua.
Haluan kiittää ammattikouluopettajaani Eero Liuskaa, osaamispohjan rakentamisesta. Hänen opettamat vianhakumenetelmät ja elektroniikan perusteet on auttanut tämänkin opinnäytetyön teossa. Ammattikorkeakoulussa osaaminen vahvistui entisestään. Monista hyvistä ideoista ja opeista kiitos opinnäytetyön ohjaajalle Jyrki Liikaselle.
Aikuisena opiskelu on antanut mahdollisuuksia syventää osaamista, sekä avannut uusia ovia uralla. Suosittelen lämpimästi monimuoto-opintoja, vaikkakin opinnot ja niiden suorittaminen vie lähes kaiken vapaa-ajan.
Iso kiitos perheelle ja erityisesti puolisolle opiskelujen mahdollistamisesta.

Haarajoella
10.1.2021

**SISÄLLYS**

1 JOHDANTO ..... 6

2 KUNNOSSAPITO TEOLLISUUDESSA ..... 6

2.1 Kunnossapidon haasteet ..... 7

3 ARDUINON KÄYTTÖ KUNNOSSAPIDON TYÖKALUNA ..... 7

3.1 Arduinon historiaa ..... 8

3.2 Arduinon perusteita ..... 8

3.2.1 Nano ..... 10

3.3 Arduino NANOn liitännät ..... 10

3.3.1 Arduinon liitäntöjen toiminta ..... 11

3.4 Tiedonkeruulaitteiston komponentteja ja tekniikoita ..... 12

3.4.1 Milliampeeriviesti ..... 13

3.4.2 Lämpötila-anturit ..... 13

3.4.3 MicroSd- ja RTC-lisäkortti ..... 15

3.4.4 Johdotusalusta ..... 16

4 TIEDOKERUULAITTEISTOJEN MÄÄRITTELY ..... 17

4.1 Kuljetinlinjan moottorin väännön tallennus ..... 17

4.2 Laitteiden lämpötilojen seuranta ..... 19

5 TIEDONKERUULAITTEEN TOIMINTA ..... 20

5.1 Ohjelma milliampeeriviestin tallentamiseen ..... 20

5.2 Tiedokeruulaitteen kytkennät ..... 24

6 TIEDONKERUUN TULOKSIA ..... 24

7 POHDINTA ..... 26

LÄHTEET ..... 28

KUVALUETTELO ..... 29

LIITTEET

Liite 1. Arduino Nanon kytkentä taajuusmuuttajaan

Liite 2. Ohjelma lämpötilan tallentamiseen

# 1 JOHDANTO

Opinnäytetyössä tutkitaan Aduino-mikrokontrollerin käyttöä kunnossapidon apuna, seurataan ja kartoitetaan tuotannossa olevien laitteiden vikaantumista sekä mahdollisuuksia laiterikkojen ennakointiin. Ennakoinnilla saavutetaan hyötyjä tuotannon tehokkuutteen ja laiterikkojen aiheuttamien häiriöiden pienentämiseen.

Aiemmin laitteiden seurantaan tarkoitetut laitteet ovat olleet kalliita, siitä johtuen seurantalaitteiden määrä rajallinen. Pienet ja modulaariset mikrokontrollerit tarjoavat loistavan mahdollisuuden toteuttaa laiteseurantaa kustannustehokkaasti.

Arduinon kehitystyö on helppoa ja komponenttien edullisuus mahdollistaa kokeilemisen ilman merkittävää taloudellista panostusta. Lisäksi Arduinon pieni koko ja olematon virrankulutus luo useita mahdollisuuksia sijoittelun suhteen. Uskon tämän opinnäytetyön avaavan uusia mahdollisuuksia mikrokontrollereiden ja erityisesti Arduinon potenttiaaliin kunnossapidon työkaluna.

# 2 KUNNOSSAPITO TEOLLISUUDESSA

Kunnossapito sisältää mekaanisen ja sähköautomaation osaamisalat. Kunnossapito vastaa laitteen kunnosta ja ylläpidosta laitteiston elinkaaren aikana. Kunnossapito  suorittaa laitevalmistajan ohjeistuksen mukaan suunniteltua ennakkohuolto-ohjelmaa. Hyvästä ennakkohuollon suunnittelusta huolimatta odottamattomia laiterikkoja tapahtuu. Laiterikkoja aiheuttavien komponenttien valvonta on usein haastavaa ja mahdotonta kattavasti toteuttaa.

Komponenttien vikojen seuraamiseen soveltuvat seurantalaitteet liittyvät lämmön, värähtelyn ja erilaisten sähköisten suureiden mittaukseen. Lämmön muutoksen seurannalla on mahdollista havaita alkavat viat laakereissa, vaihteistoissa ja muissa komponenteissa. Värähtelyä seuraamalla on mahdollista valvoa laakereiden kuntoa ja tehdä tarkkoja ennusteita vaihdon tarpeesta. Sähköisillä suureilla tarkoitetaan virran, jännitteiden ja niiden perusteella laskettavien arvojen seurantaa.

Vaikka teknisillä ratkaisuilla on mahdollista seurata tarkoin laittoistojen kuntoa on visuaalinen tarkastelu tärkeä osa kunnossapidollisia toimia edelleenkin, Robert Lusser /1/ onkin todennut aikoinaan seuraavasti:

*” Useista osista koostuvan järjestelmän luotettavuus on osien luotettavuuksien tulo”*

Tämä tulee ottaa huomioon kunnossapidon strategiaa laatiessa. Kattavalla ennakkohuoltosuunnittelulla ja kattavalla seurannalla saadaan tuotannosta häiriöttömämpää ja tehokkaampaa.

### 2.1 Kunnossapidon haasteet

Tuotannon kunnossapidossa suurimmat ongelmat ja haasteet johtuvat laiterikoista. Laiterikkojen tapahtuessa tuotanto pysähtyy ja aiheuttaa taloudellisia menetyksiä. Tuotannon kannalta on tärkeää saada linjasto takaisin käyntiin mahdollisimman nopeasti. Tämä aiheuttaa kunnossapidolle haasteen tehdä parempaa suunnitelmaa laiterikkojen ehkäisyyn. Laiterikkojen ehkäisyn ennakointiin on ennakkohuoltosuunnittelulla tärkeä rooli. Ennakkohuolto perustuu usein aikaperusteisesti suoritettavaksi huolloksi, missä laitevalmistajan määrittelemät komponentit tarkistetaan ja tarvittaessa vaihdetaan. Huolimatta hyvästäkin ennakkohuollosta odottamattomia laiterikkoja tapahtuu. /2./

Laiterikot oireilevat usein ennen rikkoutumista, kuten ääntä pitämällä lämpenemällä tai sammumalla odottamattomasti. Ongelmien havaitseminen ajoissa vaatisi paljon henkilöresursseja kunnossapidolta, joten jokaisen kriittisen komponentin jatkuva seuraaminen on mahdotonta. Kunnossapidon ennakoivaan huoltoon tarvittavan historiatiedon kerääminen on tärkeää laitteiden kunnossapidon ja tuotannon optimoimiseksi.

## 3 ARDUINON KÄYTTÖ KUNNOSSAPIDON TYÖKALUNA

Arduinoa pystyy käyttäminen tehokkaasti apuna vikatilanteissa, jotka ovat hetkellisiä kestoltaan tai vaativat pitkän aikavälin seurantaa. Näissä tilanteissa voidaan Arduinoa käyttää tiedonkeruulaitteistona. Tiedonkeruu yksinkertaisuu-

dessaan tarkoittaa lukemalla haluttua anturia ja tallentamalla luettu arvo talteen. Tallennusmediana voidaan käyttää joko sd-korttia tai erillistä verkkopalvelinta.

Tiedonkeruun voi myös ohjelmoida keräämään luettu data jatkuvasti talteen, mutta kerättyä tietoa on analysoitava jälkikäteen enemmän. Mahdollista on myös, että anturilta luettu arvo käsitellään ehtolauseiden avulla jo ennen kortille tai tietokantaan tallentamista, jolloin tallennettu tieto on jo valmiiksi suodatettua.

Kerättävälle tiedolle on yleistä, että se sisältää kellonajan, päivämäärän ja halutun anturin arvon. Tiedonkeruun aikaväli on täysin ohjelmoitavissa tarpeen mukaan. Rajoituksena on sd-kortille tapahtuvan kirjoitukseen kuluva aika, mahdollinen anturin mittausviive sekä Arduinoon ladatun ohjelman kiertoaika.

### 3.1 Arduinon historiaa

Arduino perustuu 2003 Italiassa Hernando Barragánin tohtorinväitöskirjaan Wiring, jonka ideana oli tuoda markkinoille edullinen kehitysalusta. Kehitystyötä 2005 jatkoi Massimo Banzi ja David Mellis suunnittelemalla piiri tukemaan edullisemman Atmel:in kehittämän ATmega-piirisarjan ympärille. Nimeksi tuli Arduino. Arduino tarjoaa mahdollisuuden luoda uusia innovaatioita helposti. /3./

ATmel:in ATmega-iirisarja kuuluu AVR-tuoteperheeseen, jonka norjalaiset opiskelijat Alf-Egil Bogen ja Vegard Wollan kehittivät. AVR-nimi tulee heidän nimikirjaimistaan eli **A**lf:in ja **V**egard:in **R**isc prosessori. Alkuperäinen piiri esiteltiin yleisölle vuonna 1997. Se on 8-bittinen, ensimmäisiä flash-muistia sisältävä RISC-prosessoripiiri. Tuohon aikaan yleisesti oli käytössä mikrokontrollerin lisänä erillisellä ROM-, EPROM- ja EEPROM-piireillä toteutettu ohjelmamuisti. Melkein kaikissa AVR-mikrokontrollereissa on sisäänrakennettu EEPROM-muisti tiedon tallentamista varten. /3./

### 3.2 Arduinon perusteita

Arduinon mallisarjasta löytyy laajasti useampaan käyttötarkoitukseen soveltuvia malleja. Käytännön erot tulevat lähinnä I/O-liittimien määrästä, suorittimen

kellotaajuuden ja muistin koosta. Arduinon kyky siirtää dataa on hieman rajallinen, esimerkiksi videokuvan siirtoon arduino ei sovellu. Tosin Arduinolla ja ulkoisella näytöllä voi näyttää graafista sisältöä. Lisäksi malleja yhdistävä tekijä on, että ne ohjelmoidaan Arduino IDE:ä käyttämällä. Arduino IDE on avoimen lähdekoodin ilmaisohjelma perustuen GNU/GPL-lisenssiin. Avoimuus järjestelmällä on lisännyt sen suosiota laajasti.

Ohjelmiston avulla laaditaan tarvittava koodi, ohjelmointikielenä käytetään supistettuja toimintoja ja komentoja C- ja C++ -kielistä. Kirjoitettu koodi käännetään Arduinon ja muidenkin mikroprosessorilaitteiden ymmärtämäksi heksadesimaalikonekieleksi.

Ohjelmoinnissa käytetään kirjastoja, mitkä ovat valmiiksi ohjelmoituja ”ajureita” ja tukevat erilaisten antureiden ja väylien käyttöä kortilla. Valmis kirjasto mahdollistaa Arduinon kehitysalustan käytön suppeammalla käyttäjän tekemällä koodilla. Tarvittavia ja haluttuja ominaisuuksia kutsutaan ohjelmassa käytetyn kirjaston ohjeiden mukaisesti. Kirjastot ovat myös avoimen lähdekoodin lisenssin alaisia, joten kirjastoon voidaan tehdä muutoksia tarpeen tullessa.

Ohjelmointinäkymä näkyy kuvassa 1, se koostuu kolmesta osiosta, ylimpään lohkoon määritellään kirjastot ja muuttujat. Seuraavaksi on setup-lohko, mikä luetaan ja suoritetaan ainoastaan käynnistysvaiheessa. Osiossa määritellään sisääntulot ja alustetaan sarjaliikenne. Main-lohkossa pyörii ohjelma loopissa, sitä toistetaan uudelleen ja uudelleen.

Kuva 1. Arduino IDE -ohjelmointinäkymä (P.Mikkonen 2021)

Ohjelmoinnin omaksuminen ja harjoittelu on mahdollista ohjelmalla itsessään, osiosta löytyy valmiita malleja ohjelmista, mitä muokkaamalla päästään mukavasti alkuun ohjelmoinnissa. Yhdistelemällä koodeja saadaan rakennettua monimutkaisiakin ohjelmia vähäisellä ohjelmointikokemuksella.

Ohjelman avulla koodi tarkistetaan virheiden varalta ennen kortille latausta. Samalla tarkistetaan koodin soveltuvuus valitulle kehitysalustalle. Tarkistuksessa havaitut virheet havaitaan helposti, ohjelman tarkistus pysähtyy riville, josta virhe on havaittu.

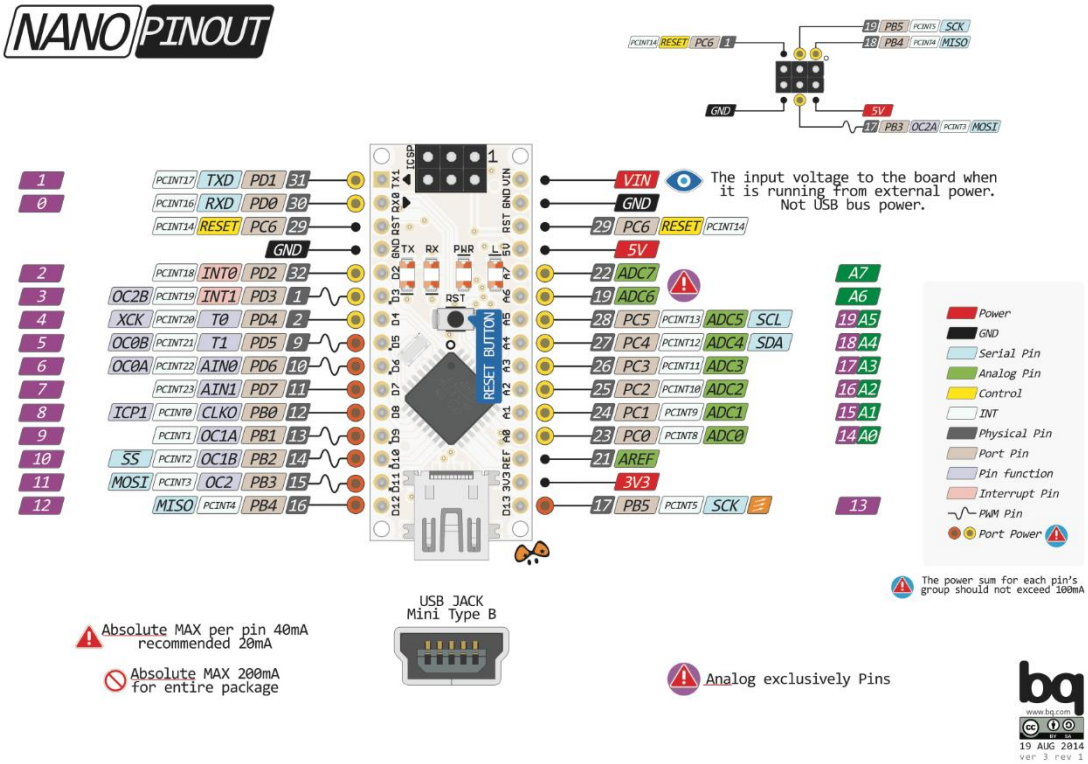
Ohjelmaa voidaan käyttää myös debug-ominaisuudessa käyttämällä sarjaliikenneikkunaa. Sarjaliikenneikkunan hyödyntäminen on yksinkertainen ja toimiva menetelmä, kun kytkennän antureiden ja kytkimien tilaa halutaan tarkkailla. Käytettäessä sarjaliikennemonitoria on Arduino oltava kytkettynä usb:lla ohjelmointilaitteeseen. /4./

### 3.2.1 Nano

Arduino-mallisarjan pienin kortti soveltuu koon puolesta hyvin tiedonkeruuseen, sekä on kustannuksiltaan edullinen. Mallissa on Micro-USB-liitäntä ohjelmointia ja jännitesyöttöä varten, toki jännite voidaan tuoda kortille myös erilliseen nastaan. Lisäksi laitteen pieni virrankulutus tukee laitteen käyttämistä pitkäkestoisissa tiedonkeruujärjestelmissä, ilman komponenttien poistoa piirilevyltä, päästään n 15,5 mA virrankulutukseen. /5./

## 3.3 Arduino NANOn liitännät

Arduinon käyttöjännite on 5V, analogisia I/O-liitäntöjä on kuusi kappaletta (A0-A5), digitaalisia I/O-liitäntöjä 14 kappletta, joista kuusi mahdollistaa PWM-ulostulon. Arduino Nanossa on Mini USB-liitin, jonka kautta ohjelmointi tehdään. USB-liittimen kautta Arduino saa käyttöjännitteen, jos sitä ei tuoda VIN-liittimeen. /6./




Kuva 2. Nanon liitännät

Kuvasta 2 nähdään Arduino Nanon liitännät. Suurin osa liittimistä on suoraan kuvasta tulkittavissa, alla on muutamia erikoisuuksia listattuna:
MOSI = Master Out Slave in, väylässä olevan masterin lähettämän datan sisääntulo
MISO = Master In Slave Out, väylässä olevan orjan lähettämä data masterille
SCLK = Serial Clock, Master laitteen generoima kellopulssi
SS/CS = Slave Select / Chip Select, pinnillä kerrotaan orjalle, onko master lähettämässä vai vastaanottamassa dataa. Yleensä asetetaan LOW-tilaan, jotta orja tietää master lähettää tai odottaa dataa.

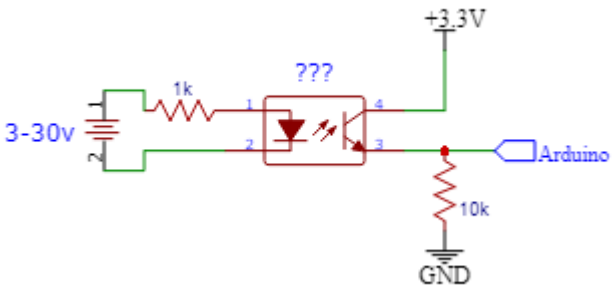
### 3.3.1 Arduinon liitäntöjen toiminta

Arduinon I/O:t toimivat joko analogiselle tai digitaaliselle signaalille. Analogista sisääntuloa käytetään, kun arvo on vaihteleva 0-5v välillä. Analoginen pinni mahdollistaa lukea tai kirjoittaa A/D-muuntimen läpi muunnetun arvon 0-1023, eli 0 tarkoittaa 0v ja 1023 tarkoittaa 5v tasoa. A/D-muuntimen tarkkuus on 10 bittiä. Arvon lukeminen analogisesta pinnistä onnistuu helpoiten käyttämällä ulkoista vastusta jännitteen mittaukseen, kun tallennetaan esimerkiksi virta-viestiä, ulkoisen laitteen lähettämästä virta-arvosta 4–20mA. Liittimiin lähetetty virta-arvo sopivasti mitoitetun vastuksen läpi, antaa vastuksen yli jännitteen

alueella 0V–5V. Monista automaatiossa käytetystä komponentista saadaan virtaviesti valittua sekä suodatettua tieto raja-arvoihin tai hälytyksiin perustuen. Digitaalista sisääntuloa käytetään, kun tallennuksessa käytetään ns. tosi / epätosi-valintaa. Digitaalista sisääntuloa hyödynnetään, kun tallennettavaan tilanteeseen sisältyy esimerkiksi käyttäjän tekemiä valintoja kytkimillä tai ohjelmallisesti.

### 3.4 Tiedonkeruulaitteiston komponentteja ja tekniikoita

Perinteisten antureiden käyttäminen Arduinossa onnistuu jännitesovituksen jälkeen. Perinteisesti antureiden käyttöjännite on 24VDC, jolloin signaalijännite on sovitettava Arduinolle sopivaksi. Käyttöjännitteen ollessa kytkettynä erillisestä jännitelähteestä jää anturin antaman signaalin taso liian korkeaksi. Suositeltava tapa on käyttää opto-erotinta, jolloin signaalijännite on galvaanisesti erotettu arduinon sisääntulosta. Periaate esitetty kuvassa 3.




Kuva 3. Arduino optoerotin

Lämpötilojen seurantaan löytyy hyviä perusantureita, joiden tarkkuus riittää hyvin laitteistojen ja komponenttien lämpötilan seurantaan. Lisäksi käyttämällä esimerkiksi Dallasin DS18B20-anturia voidaan yhteen sisääntuloon kytkeä väylämaisesti useita antureita.

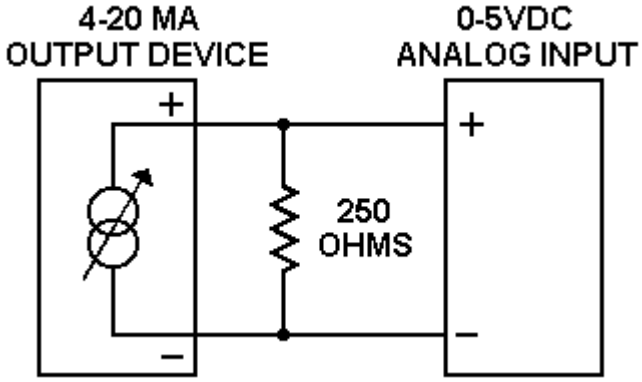
Edellä mainittuja sovelluksia on mahdollista yhdistellä tarpeen mukaan. Yhdistelemällä antureita saadaan kattavasti tallennettua tilatietokaappaus tietyllä ajanhetkellä. Saatu tieto yksinkertaisimmillaan voidaan tuoda Exceliin ja esittää helppolukuisina kaavoina.

### 3.4.1 Milliampeeriviesti

Teollisuuslaitteiden ja anturien välinen kommunikointi tapahtuu käyttämällä 4…20mA:n virtaviestiä. Etuja virtaviestin käyttämiselle on pitkien asennusvälimatkojen mahdollistaminen, virtaviesti ei vaimene, kuten jänniteviesti pitkillä johdinpituuksilla. Mahdollisia etäisyyksiä johtoteitse prosessiantureille on 5 m – yli 100 m. Anturilähettiminen säätämä virta säilyy samana koko kyseisessä virtasilmukassa. /7./

Mahdolliset johtokatkokset löydetään helposti, koska katkoksessa virta on 0mA. Alhaisella virralla voidaan järjestelmä ohjelmoida antamaan häiriöilmoitus katkoksesta. Jänniteviestillä katkennut kaapeli voi toimia antennina elektromagneettisille häiriöille, jolloin johtimessa kulkee jännitettä, mikä näkyy virheellisinä mittaustuloksina. Virtaviesti sietää paljon paremmin EMI-häiriöitä. Monissa ohjelmoitavissa antureissa voidaan anturivian sattuessa ohjelmoida anturi säätämään virta joko korkeammaksi esim. 23mA tai jokin muu alueen ulkopuolelta <0mA >4mA.

Anturin käyttäminen Arduinossa on mahdollista käyttämällä erillistä vastusta virtaviestilinjassa mittaamalla käytetyn vastuksen yli olevaa jännitettä. Vastus mitoitetaan siten, että 20mA arvolla vastuksen yli oleva jännite on 5V. Kuvassa 4 periaate kytkennästä ja mitoitetusta komponentista. /7./




Kuva 4. Milliamppeeriviestin mittaus

### 3.4.2 Lämpötila-anturit

Yleisesti käytetään Dallasin valmistamaa DS18B20-anturia, mikä on luotettava ja edullinen anturi. Anturia saa perinteisessä TO-92- tai SO-kotelossa, lisäksi

myös vesitiiviinä anturina, jossa on valettuna TO-92-koteloitu anturi, putken sisään (kuva 5).

Anturin toiminta perustuu analogisen lämpötila-arvon muuttamiseen digitaaliseksi, konvertointi kestää n. 750ms 12-bit ja 93.75ms 9-bit resoluutiolla. Konvertointiin kuluva aika on syytä ottaa huomioon mittauksen näytteenoton väliä suunnitellessa. Kyseinen anturi toimii -55 C - +125C alueella, tarkkuudella +-5 %. Tämä riittääkin useimpiin sovelluksiin. Resoluutio on käyttäjän asetelta-vissa 9–12 bit, vakioasetus on 12-bit, mikä tarkoittaa 0,0625C. Käyttöjännite on 3V-5.5V, virrankulutus on 1mA mittauksen aikana lepovirran ollessa 0,8uA. Vähäinen virrankulutus ja käyttöjännitealue tekeekin siitä ihanteellisen 18650 akuilla varustettuihin tiedonkeruulaitteistoihin.

Anturi käyttää 1-Wire-protokollaa lämpötilatiedon siirtämiseen, väylä tarvitsee vain kahta johtoa: maa ja data. Väylä toimii isäntä/orja-periaatteella yhden laitteen ohjatessa muita väylään kytkettyjä laitteita. Jokaisella anturilla on 64-bittinen uniikki id-numero, mikä mahdollistaa useamman anturin käyttämisen samassa dataväylässä. Käytettävä kirjasto arduinossa määrittelee väylässä olevien antureiden maksimimäärän, esimerkissä oleva kirjasto tukee 32 anturia. Anturit on löydettävä väylästä ennen niiden käyttöä, mikä tapahtuu isäntälaitteen reset-pulssilla väylään, jolloin orja-laitteet ilmoittavat olemassa olostaan isännälle. Itse anturissa on kolme johdinta kuvan 5 mukaisesti.


Kuva 5. Dallas DS18B20 -anturi

Anturia käytettäessä 4.7k ylösvetovastus tarvitaan väylään, jotta tiedonsiirto on luotettavaa. Arduinossa anturia käytettäessä tarvitaan kaksi kirjastoa One-Wire-väylää varten ja DallasTemperature. OneWire-kirjasto toimii muidenkin kyseistä väylää käyttävien laitteiden kanssa. /8./

### 3.4.3 MicroSd- ja RTC-lisäkortti

Tiedonkeruun tärkein komponentti on luotettava tallennusmedia, aikoinaan käytettiin paperille piirtäviä piirtureita keräämään dataa talteen. Nykyään tekniikan kehittyessä on mahdollisuudet laajentuneet huomattavasti tiedon ja mittaustulosten tallentamiseen. Yleisimpiä medioita tallennukseen on SD-kortit, joiden kapasiteetti on useita gigatavuja. Mittausdata koostuu yleensä yksinkertaisesta ja pienestä datamäärästä, jolloin SD-kortin kapasiteetti riittää tallennukseen vuosiksi.

Projekteissa yleisesti käytetty logger-kehityskortti sisältää reaalikellon ja sd-kortinlukijan. Kortti on tarkoitettu Arduino nanon pinnitykselle sopivaksi. Kuvassa 6 näkyy kortti, vamistaja Deek-Robot malli 8105. Arduinossa sd-kehityskortin käyttämiseen tarvitaan käyttöjännite 3.3V–5.5V sekä 4 dataväylää ja tietenkin maapotentiaali. Kyseinen kortti käyttää SPI- kommunikointia väylässä, mikä tarkoittaa Serial Parallel Interface, eli synkronoitu data siirretään kellopulssin tahdissa. Kyseinen kommunikointiväylä ei ole hyvä pitkille etäisyyksille, mutta toimii väylän ollessa alle 10m. Sd-kortti käyttää jännitteenä 3.3V, joten jännitetason muutos 5V:sta 3.3V:iin käytetään 74LVC125A-piiriä logiikkatason muuntimena.

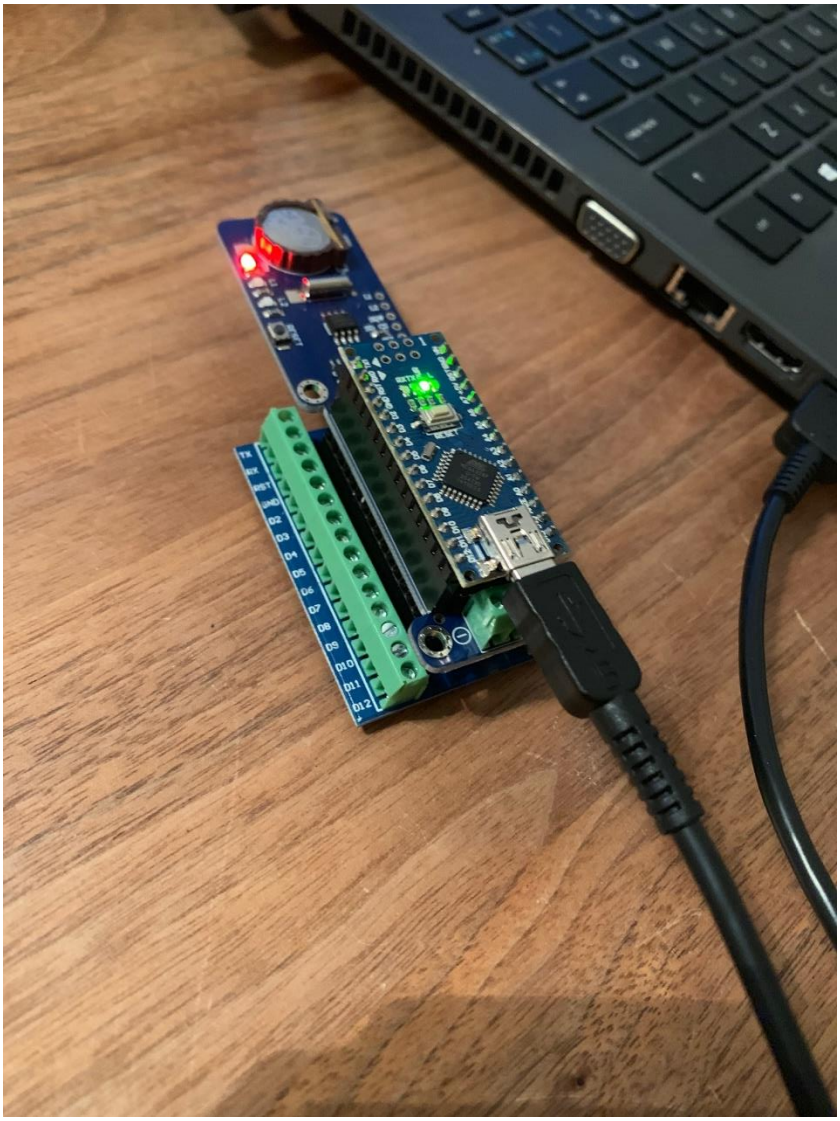
Käytössä on MOSI, MISO, SCLK ja SS/CS. Lisäksi käyttöjännite ja maa.

RTC eli reaaliaikakello on paristovarmennettu kello. Paristo pitää kellon ajassa, vaikka ulkopuolinen käyttöjännite häviää. RTC:n käyttö tulee kyseeseen, kun tarvitaan tarkkaa ja pidempää kellonaikatietoa. Arduinon oma millis(), toimii ainoastaan arduinon päälle kytkentähetkestä. Laskuri nollaantuu, kun käyttöjännite häviää. Lisäksi arduinon millis() ei ymmärrä, onko keskiviikko vai torstai, ainoastaan kuinka monta millisekuntia kytkentähetkestä on. RTC-kello perustuu Dallasin DS1307-piiriin. Piiri on edullinen, toimintavarma ja nappiparistolla varmennuttuna toimii vuosia. RTC-kello perustuu kideoskillaattoriin (TCXO), joka on lämpötilakompensoitu. Kideoskillaattori on suunniteltu toimimaan vakaasti lämpötila-alueella -10 - +60, alueen ulkopuolella värähtelyn tarkkuus heikkenee jyrkästi. Kideoskillaattori antaa 1 Hz pulsseja, joista määritellään ajalle sekunnin pulssi. Pulssien perusteella määritellään

ajan kulku piirissä. Piiriä käyttöön ottaessa määritellään käyttöönottoajankohta, yleensä määritys tapahtuu ohjelman lataushetken mukaan. /9./

### 3.4.4 Johdotusalusta

Kuvassa 6 on erittäin monipuolinen kytkentäliittimin varustettu kortti nanolle, kortin monipuolisuus tulee esille kytkentöjä tehdessä. Kohteissa, missä ilmaantuu tärinää, on ruuviliittimet kestävin ja luotettavin ratkaisu. Kustannuksiltaan kytkentäkortti on todella edullinen.



Kuva 6. Arduinon johdotusalusta ( P.Mikkonen 2021)

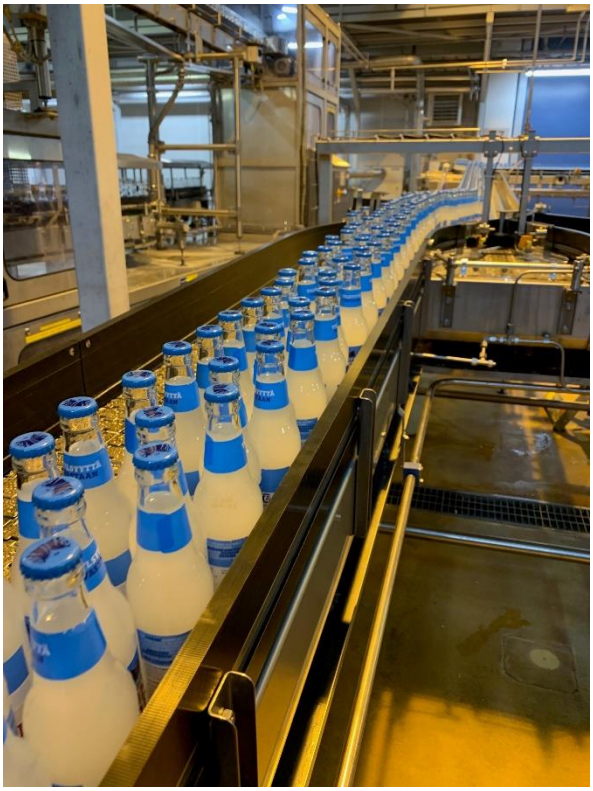
# 4 TIEDOKERUULAITTEISTOJEN MÄÄRITTELY

Laitteistoille mitattaviksi parametreiksi valikoitui moottorin vääntö ja ympäröivän ilman lämpötila. Mittauskohteiden monimuotoisuudesta päädyttiin tekemään kaksi erillistä mittauslaitteistoa. Mittauslaitteistolle ominaista projektin alkuvaiheessa on prototyyppimäinen ulkoasu. Prototyypillä on tarkoitus hankkia kokemuksia ja ennen kaikkea kerätä tietoa.

Laitteistolla kerättävä tieto on tärkeää, jotta saadaan kattavasti arvoja talteen järjestelmien toimiessa moitteettomasti. Laiterikkojen tapahtuessa on kerätystä tiedoista mahdollista havaita mahdolliset muutokset, jotka johtavat rikkoutumiseen. Tällöin voidaan määritellä tiedokeruulaitteiston ohjelmaan hälytysulostulo, joka pysäyttää tai aiheuttaa hälytyksen.

## 4.1 Kuljetinlinjan moottorin väännön tallennus

Tuotantolinjalla valmiiden tuotteiden kuljettamiseen käytetään kuljettimia, kuvassa 7. Kuljettimien lamellien rikkoutuessa tuotanto pysähtyy ja aiheuttaa häiriöitä tuotantoon. Lamellin ja kuljetinlinjan kunnon tarkkailu on tärkeää, koska kuljetinlinjan katkeaminen aiheuttaa tuotannon pysähtymisen.



Kuva 7. Kuljetinlinja tuotannossa (P.Mikkonen 2021)

Rakennetulla laitteistolla seurataan kuljetinlinjan moottorin momenttiarvoja taajuusmuuntajalta. Sähkökaapissa olevalla taajuusmuuntajalla voidaan määritellä ulostulo momentille virtaviestinä, kuten kuvassa 8.


Kuva 8. Taajuusmuuttaja ja tiedokeruulaitteisto (P.Mikkonen 2021)

Tuotannon ollessa käynnissä voidaan seurata eri tuotteiden vaikutusta käytettyyn momenttiin. Eroavaisuudet tulevat tuotteen painon muutoksista. Mitattavalla kuljettimella ajetaan lasipulloja.

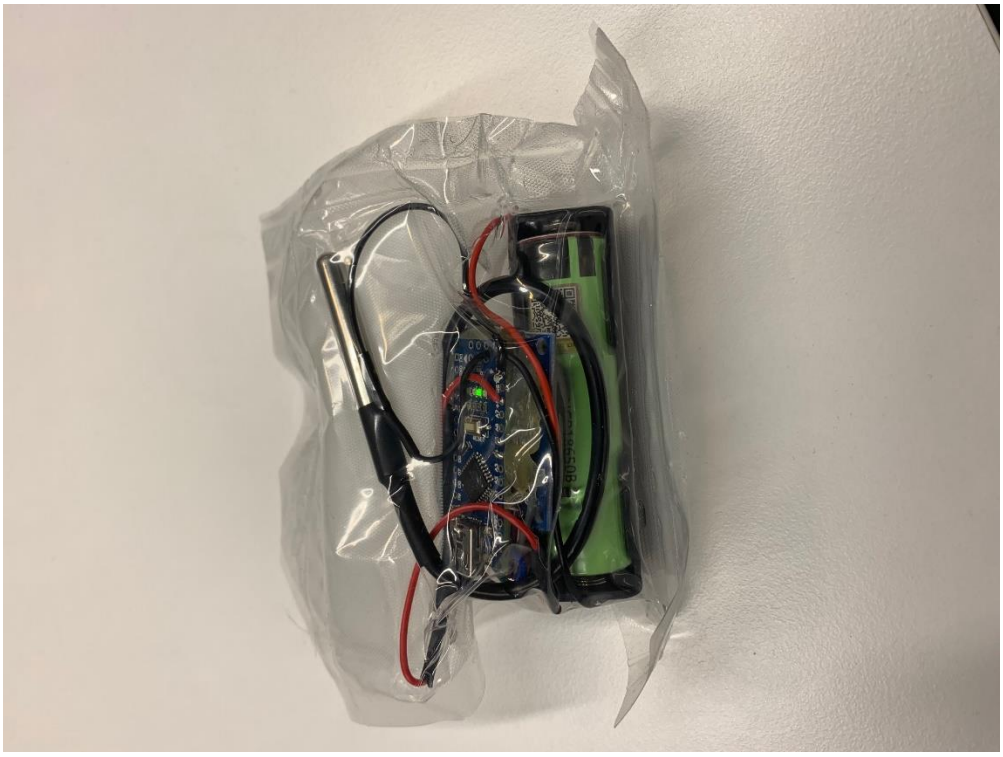
Mittauksella saavutetaan nominaaliarvot momenteille kuljettimien ollessa kunnossa. Kuljettimet pyörivät eri täyttöasteilla, joten kattavasti saadaan tallennettua dataa. Seurantajakson aikana sattuneet rikkoutumiset kirjataan tarkasti talteen.

## 4.2 Laitteiden lämpötilojen seuranta

Arduinolla on mahdolista seurata mittaamalla lämpötiloja laitteista ja komponenteista tuotannon aikana. Laitteistojen lämpenemisen ja tuotannon häiriöiden välillä on usein yhteys, ja Arduinon avulla se voidaan todentaa. Osalla laitteita saatetaan huomata häiriöiden ilmaantuvan varsinkin pitkien tuotantosarjojen aikana. Yleensä syy on, että jokin komponentti lämpenee tuotannon aikana, aiheuttaen häiriöitä tuotantoon. Kuvassa 9 olevalla laitteella seurattiin sähkökaapin jäähdytyksen toimintaa.


Kuva 9. Lämpötilan seuranta AP9 kotelossa (P.Mikkonen 2021)

Lämpötilojen tallentamisen edut ovat energiatehokkuuden seurannassa, esimerkiksi tuotteen optimaalinen lämpötila tuotannon aikana. Kokeiluissa sijoitettiin tiedonkeruulaite testitölkkiin. Tiedonkeruulaite pakattiin vakuumipakettiin ja asetettiin vedellä täytettyyn tölkkiin, kuvassa 10.



Kuva 10. Arduino nano vakuumissa (P.Mikkonen 2021)

# 5 TIEDONKERUULAITTEEN TOIMINTA

Ohjelmallisesti Arduinon käyttö tapahtuu luvun 6.1 mukaisesti. Kelloa otettaessa käyttöön ensimmäistä kertaa on ohjelmallisesti määriteltävä kellonaika ja päivämäärä. Asetus tehdään vain kerran, jonka jälkeen paristo pitää huolen, että kello pysyy ajassa. Lisäohjeet ajan asettamiseen löytyvät käytetyn RTC-kirjaston malleista. Yleisesti kellonaika muodostuu hetkestä, jolloin ohjelma ladataan Arduinolle. Ohjelma listaus lämpötilan tallennukseen on liitteessä 2.

## 5.1 Ohjelma milliampeeriviestin tallentamiseen

Ohjelman Arduinon käyttämiseen, soveltuu suoraan käytettäessa aiemmin mainittuja lisäkortteja ja kytkentää kuvan 11 mukaisesti. SD-kortti voidaan alustaa Windows käyttöjärjestelmälle toimivaksi, jolloin se toimii myös tiedokeruu laitteiston kanssa.

Ohjelmiston prosenttilasku kaavaa joudutaan soveltamaan tapauskohtaisesti. Parhaimpaan lopputulokseen päästään fyysisellä mittauksella. Apuna kaavan tarkistukseen voidaan käyttää sarjaliikennemonitoria.

Koodissa itsessään on ” dataFile.flush(); ” komento, joka tallentaa tiedoston kortille jokaisen päivityksen jälkeen, jolloin mitatut arvot sailyvät tallessa vaikka Arduinon käyttöjännite sammuu.

Koodi:

```
#include <Wire.h>    //kirjasto, väylä anturien käytön esim. DS18B20
#include <TimeLib.h> //kirjasto, RTC-piirin tarjoaman tiedon käytön astronomi-
sesti
#include <DS1307RTC.h> // kirjasto, RTC-piirin käytön, vaatii TimeLib.h kir-
jaston
#include <SD.h> //SD-kortinlukijan kirjasto

int apin = A0;   //analogisen liitännän liittäminen muuttujaan apin

float torque = 0;  //Määritellään muuttujan torque arvoksi nolla ja tyypiksi float,
eli desimaaleja sisältävä muuttuja

float prosentti = 0; //Määritellään muuttujan prosentti arvoksi nolla ja tyypiksi
float, eli desimaaleja sisältävä muuttuja

const int chipSelect = 4; //Määritellään SD kortin cS-pinni
File dataFile; //Tiedoston kutsuminen

void setup() {   //Ohjelman setup osio


 Serial.begin(9600); //Sarjaliikenteen alustus nopeudella 9600
    pinMode(SS, OUTPUT); //SS liitin toimii ulostulona”


 // Kortti paikalla tarkistus, seuraava ehtolauseke tarkistaa sd-kortin tilan
 if (!SD.begin(chipSelect)) {
   Serial.println("Kortti puuttuu!!!");

   while (1) ;
```

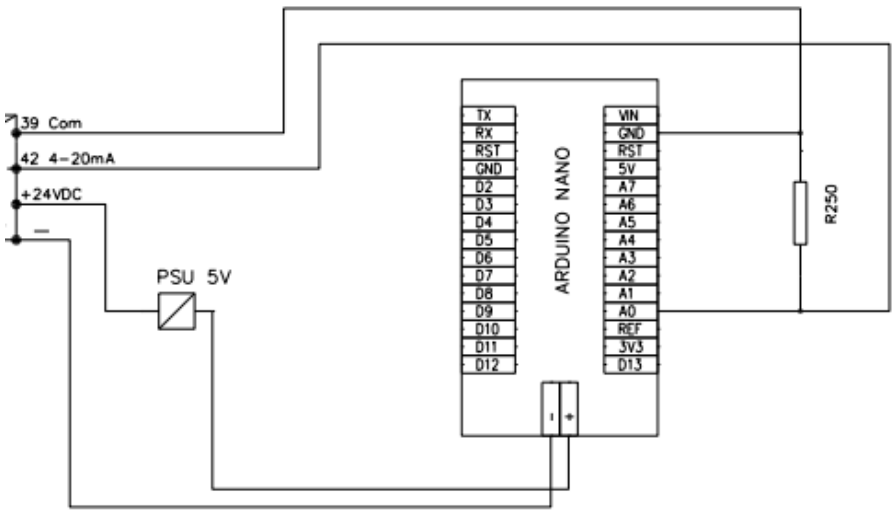
```
  }
  Serial.println("Kortti alustettu."); //kirjoittaa sarjamonitoriin, kun sd-kortti val-
mis


  // Tiedoston avaus ja luonti
  dataFile = SD.open("datalog.txt", FILE_WRITE);
  dataFile.print("Torque logger ");


  dataFile.println("(c)Petri Mikkonen");
//Seuraava ehtolauseke tarkistaa jos tiedostoa ei voida avata
  if (! dataFile) {
    Serial.println("logitiedostoa ei voi avata");

    while (1) ;
  }

}

void loop() {

// Kelloon liittyvät määritykset

tmElements_t tm;
(RTC.read(tm)); //RTC piirin lukeminen

//luetaan analoginen liitin 0 ja lasketaan vääntöprosentti
  torgue = analogRead(apin);
  prosentti = (((torque -658) / 1023) * 100);
 //Alla olevalla kirjoitetaan halutut tiedot sd-kortille
  Serial.print(prosentti);
  dataFile.print(prosentti);
  Serial.print(" %");
  dataFile.print("   :   ");
Serial.print("/");
```

```
Serial.print("   Aika = ");
Serial.print("/");
   print2digits(tm.Hour);
   dataFile.print(tm.Hour);

   Serial.write(':');
   dataFile.print(':');

   print2digits(tm.Minute);
   dataFile.print(tm.Minute);

   dataFile.print(':');
   Serial.write(':');

   print2digits(tm.Second);
   dataFile.print(tm.Second);

   Serial.print(" / ");
    dataFile.print('/');

   Serial.print(tm.Day);
    dataFile.print(tm.Day);

   Serial.write('.');
    dataFile.print('.');

   Serial.print(tm.Month);
    dataFile.print(tm.Month);

   Serial.write('.');
    dataFile.print('.');
   Serial.print(tmYearToCalendar(tm.Year));
    dataFile.print(tmYearToCalendar(tm.Year));
   Serial.println();
   dataFile.println();
```

```
  dataFile.flush();  // Tämä komento tallentaa tiedoston pysyvästi sd-kortille
 delay(1000);


}
//alla olevalla määritellään kaikki numerot kahdelle numerolle
void print2digits(int number) {
 if (number >= 0 && number < 10) {
   Serial.write('0');
 }
 Serial.print(number);
}
```

## 5.2 Tiedokeruulaitteen kytkennät

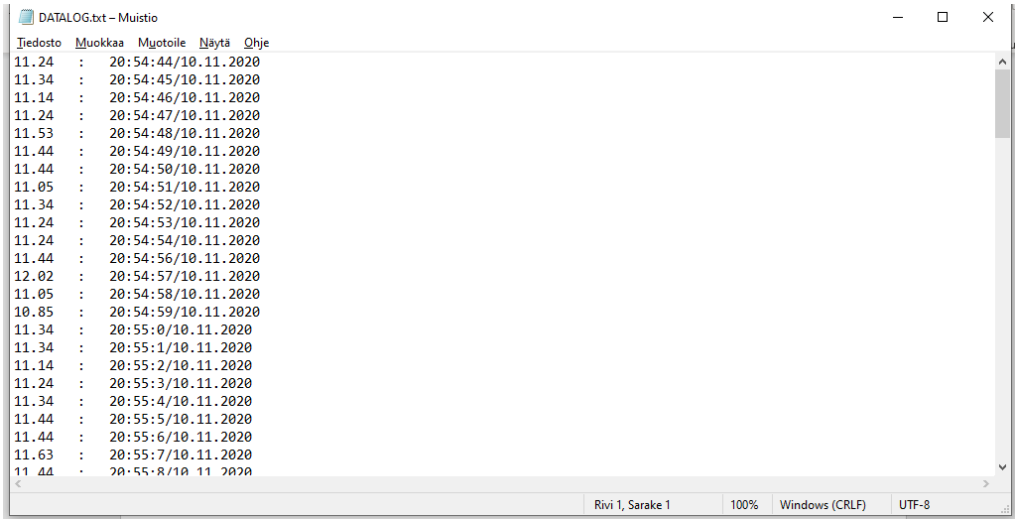
Arduinon kytkennät taajuusmuuttajan lähettämän milliampeeriviestin tallennukseen kuvan 11 mukaisesti. Kuvassa olevat johdotukset taajuusmuuttajan ja Arduinon väliset kytkennät on kuvattu kokonaisuudessaan liitteessä 1.




Kuva 11. Arduino-kytkennät (P.Mikkonen 2021)

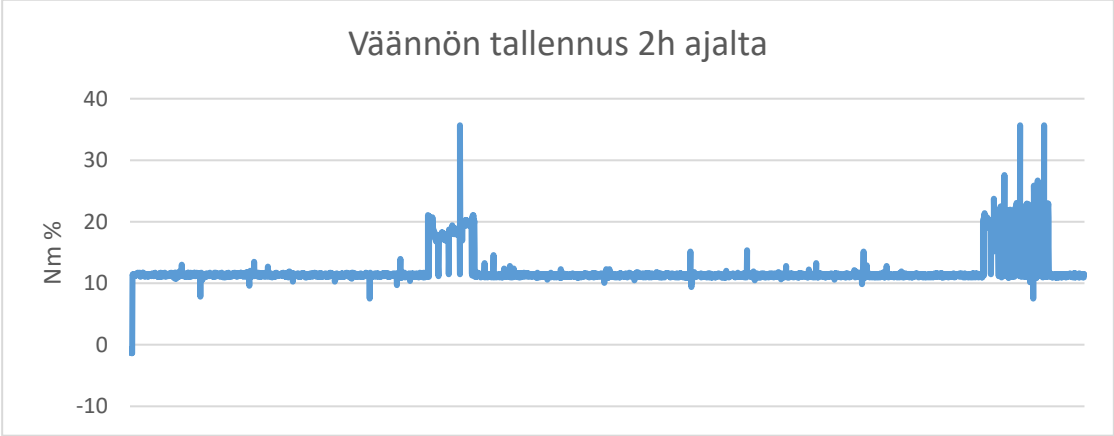
# 6 TIEDONKERUUN TULOKSIA

Tallennuksesta saatava tiedosto on kuvassa 12. Tiedosto on txt-muodossa, tiedot voidaan tuoda Exceliin ja muokata ne graafiksi.



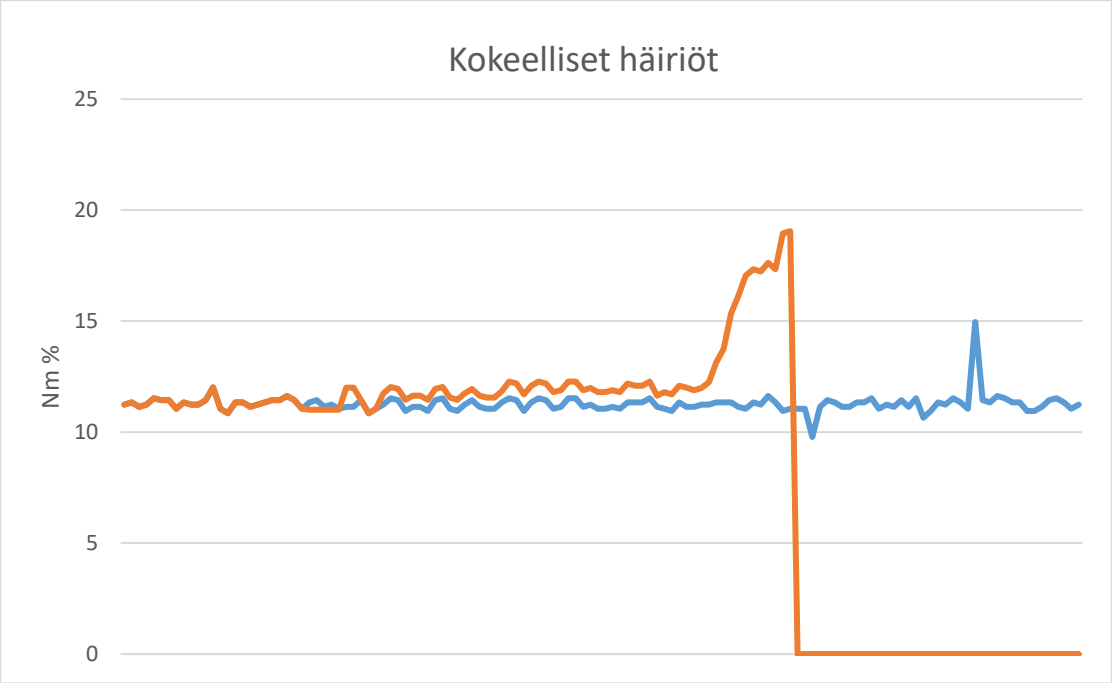

Kuva 12. Tallennettu tiedosto (P.Mikkonen 2021)

Tallennettujen tietojen perusteella voidaan kuljetinlijojen käyttäytyvän saman kaavan mukaisesti tuotannon aikana (kuva 13) kahden tunnin ajon aikana tapahtuvat momenttimuutokset. Kuvassa näkyvät piikit muodostuvat tuotteiden saapumisesta kyseiselle kuljetinosuudelle. Korkeimmat momenttiarvot kestävät noin, 3-4s, joten jaksollisesti pidempien ja korkeimpien momenttikäyrien aikana voidaan aktivoida hälytys operaattorille.




Kuva 13. Kahden tunnin väännön momenttikäyrä (P.Mikkonen 2021)

Kokeellisesti testattiin erillisellä moottorilla vierasesineiden vaikutusta momentin kasvuun. Kuvassa 14 on yhdistetty kaksi momenttikäyrää. Sininen käyrä kuvaa normitasoa, ja oranssi käyrä kuvaa vierasesineen joutumisen voimansiirtoon, lisäksi käyrässä näkyy hätäpysäytyksen painallus.




Kuva 14. Kokeellinen momenttikäyrä (P.Mikkonen 2021)

Tuloksien perusteella on selkeästi nähtävissä momenttikuormituksen muutokset häiriötilanteissa.

## 7 POHDINTA

Opinnäytetyön tavoitteena oli suunnitella ja valmistella laite, millä olisi mahdollistaa seurata ja kartoittaa laitteiden kuntoa tuotannossa. Idea mittauksien tallennuksesta on tarpeellinen, ongelmien jälkiselvittelyssä ja kuinka jatkossa voidaan estää vastaavat tapahtumat. Laitteen perustana ollut Arduino soveltui todella hyvin työhön sekä Arduinon monipuolisuus tuli todennettua, kun mittauskohteiden erillaisuus ja erityistarpeet oli helposti muokattavissa sopiviksi. Laitteen mahdollisuuksia on mahdollista hyödyntää muissakin tuotannon alueilla.

Laitteen kehitystyö onnistui helposti ja materiaalia ohjelmien hienosäätöön löytyi riittävästi. Samalla osaaminen erilaisten antureiden sijoittelusta lisääntyi. Laitteen jatkokehittelyyn on selkeä suunnitelma, lähinnä miten tallennetut tiedostot siirretään langattomasti verkkoon, jättäen samalla sd-kortille varmuuskopion tiedostosta.

Tuloksista voidaan kuitenkin jo nähdä muutoksia laitteen käyttäytymisessä, joiden perusteella voidaan kunnossapidon ennakkohuoltoohjelmaa parantaa.

Kerätyn tiedon perusteella on jo mahdollista määritellä tarvittavat kohteet huollolle ja mahdollisesti varoitusilmoitukset laitteille.

Projektina työni on vasta alussa, joten todelliset hyödyt saadaan myöhemmin selville. Kuitenkin jo saadut tulokset ovat lupaavia ja historiatiedon tallentaminen on aloitettu.

**LÄHTEET**


1. Luotettavuustekniikka. Wikipedia. WWW-dokumentti. Päivitetty 24.4.2020. Saatavissa: https://fi.wikipedia.org/wiki/Luotettavuustekniikka [viitattu 9.3.2021].

2. Aro, M. & Elovaara, J. Suurjännitetekniikka. Kunnossapidon strategiat s. 179-180. Helsinki: Otatieto Gaudeamus Oy 2015.

3. Arduino. About Us. WWW-dokumentti. Päivitetty 2021. Saatavissa: https://www.arduino.cc/en/Main/AboutUs [viitattu 10.5.2021].

4. Sami-Petteri Pukkila. Arduinon ohjelmointi. PDF-dokumentti. 2017. Saatavissa: https://mycourses.aalto.fi/pluginfile.php/665908/course/section/120801/arduino_ohjelmointi.pdf [viitattu 9.3.2021].

5. Arduino. Guide, using Arduino Nano for battery-powered projects. Ohje 2016 .WWW-dokumentti. Saatavissa: https://forum.arduino.cc/t/guide-using-arduino-nano-for-battery-powered-projects/402598 [viitattu 10.4.2021].

6. Massimo Benzi. Arduino perusteista hallintaan. Arduinon kehitysalusta. Saksa: Robomaa.com Oy. 2011

7. PR-Electronics. 4…20mA virtapiirien perusteet. WWW-dokumentti. Päivitetty 17.5.2011. Saatavissa: https://www.prelectronics.com/fi/the-fundamentals-of-4-20-ma-current-loops [viitattu 10.4.2021].

8. Datasheets. Datalehti DS18B20. PDF-dokumentti. 2019. Saatavissa: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf [viitattu 10.4.2021].

9. RTC clocks. Ohjeita kellopiirin käyttöön. WWW-dokumentti. Päivitetty 2021. Saatavissa: https://www.electronics-tutorials.ws/connectivity/real-time-clocks.html [viitattu 10.4.2021].

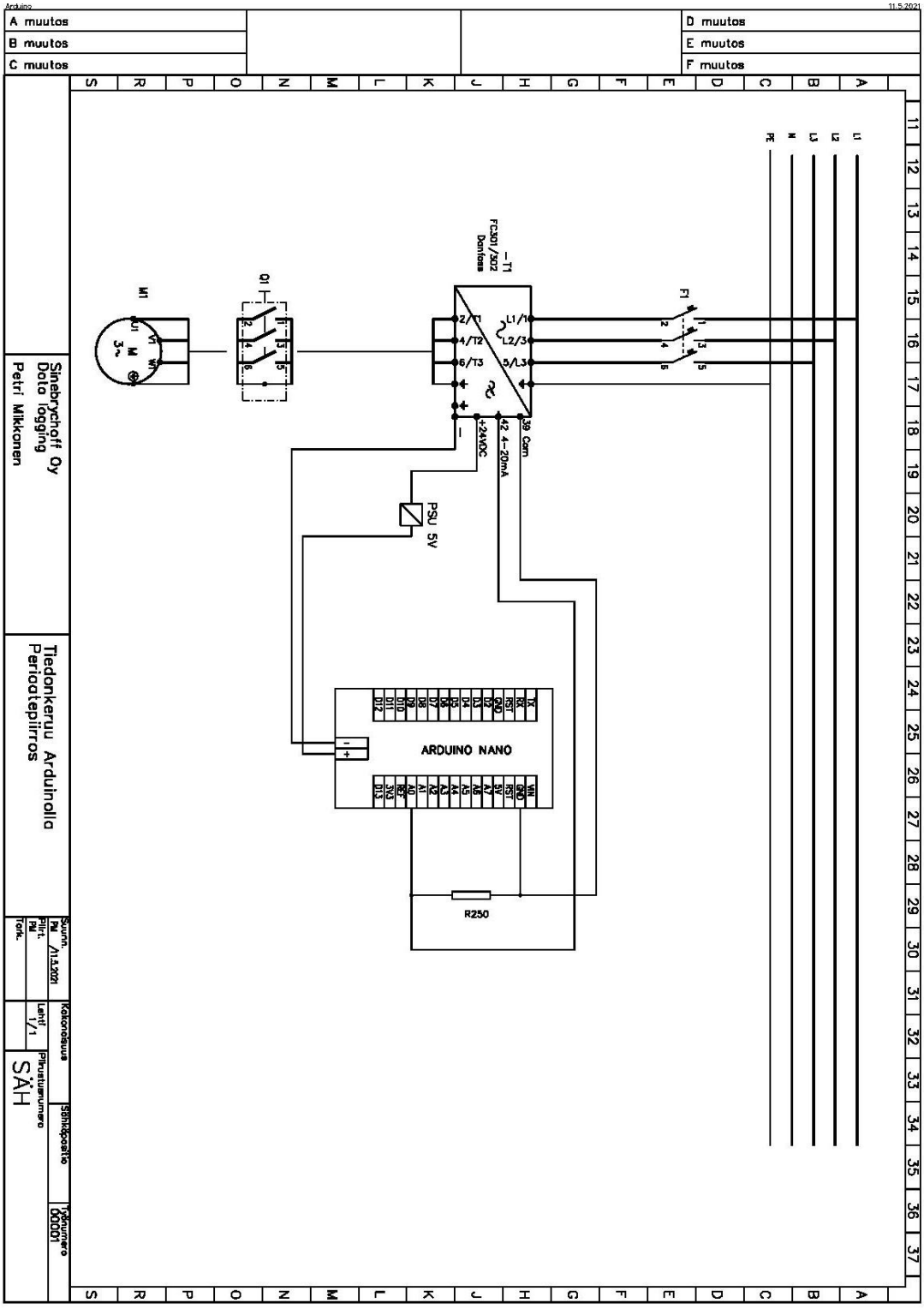
**KUVALUETTELO**

Kuva 2. Arduino nano pinout. Saatavissa: https://components101.com/microcontrollers/arduino-nano [Viitattu 10.4.2021].

Kuva 3. Opto coupler. Saatavissa: https://i.ibb.co/W2tD3QX/Diagram-for-forum-opto.png [Viitattu 8.4.2021].

Kuva 4. Milliampeeri viestin mittaus. Saatavissa: https://www.data-taq.com/blog/wp-content/uploads/2012/06/current-loop.png [Viitattu 10.4.2021].

Kuva 5. Dallas lämpötila-anturi. Saatavissa: https://www.14core.com/wp-content/uploads/2015/11/Temperature-Sensor-Dallas-Pinout-Diagram.jpg [Viitattu 1.4.2021].


Arduino
11.5.2021
A muutos
B muutos
C muutos
D muutos
E muutos
F muutos
Sinebrychoff Oy
Data logging
Petri Mikkonen
Tiedonkeruu Arduinolla
Periaatepiirros
Suunn.
PM
11.5.2021
Piirt.
PM
Tark.
Kokonaisaura
Lehti
1/1
Pihustuenumero
SÄH
Sähköposti:o
Työnumero
00001
L1
L2
L3
N
PE
F1
Q1
M1
3~
T1
FC301/302
Danfoss
L1/1
L2/3
5/L3
2/
4/T2
6/T3
+24VDC
42 4–20mA
39 Com
PSU 5V
ARDUINO NANO
R250

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SD.h>
#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 rtc;

#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

const int chipSelect = 4;
File dataFile;

void setup () {
    Serial.begin(57600);
     pinMode(SS, OUTPUT);



  if (!SD.begin(chipSelect)) {
    Serial.println("muistikorttia ei paikalla");

    while (1) ;
  }
  Serial.println("kortti valmis.");
   Wire.begin();
  rtc.begin();
  DateTime now = rtc.now();

  dataFile = SD.open("datalog.txt", FILE_WRITE);
```

```
  dataFile.print("Mitattu ");
  dataFile.print(now.year(), DEC);

    dataFile.print('/');
    dataFile.print(now.month(), DEC);
   dataFile.print('/');
   dataFile.print(now.day(), DEC);
    dataFile.print(' ');

  dataFile.println("(c)Petri Mikkonen");
  if (! dataFile) {
    Serial.println("tiedostoa datalog.txt ei voi avata");

    while (1) ;
  }


#ifdef AVR
  Wire.begin();
#else
  Wire1.begin();
#endif
  rtc.begin();
//rtc.adjust(DateTime(__DATE__, __TIME__));
  if (! rtc.isrunning()) {


    Serial.println("RTC ei toimi!");

  // rtc.adjust(DateTime(__DATE__, __TIME__));
}
}

void loop () {
    DateTime now = rtc.now();
```

```
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
      sensors.requestTemperatures(); // Send the command to get temperatures
PM
 dataFile.print(now.hour(), DEC);
    dataFile.print(':');
    dataFile.print(now.minute(), DEC);
    dataFile.print(':');
    dataFile.print(now.second(), DEC);
    dataFile.print(' ');
   dataFile.print(" , ");
  dataFile.print(sensors.getTempCByIndex(0));
  dataFile.println(" "); //Astetta
   Serial.println(sensors.getTempCByIndex(0));
   dataFile.flush();
  delay(500);

}
```