Ville Pentikäinen

AUTOPILOTTI VENEESEEN

Opinnäytetyö
CENTRIA AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Huhtikuu 2013

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö Ylivieska	**Aika** Huhtikuu 2013	**Tekijä/tekijät** Ville Pentikäinen
Koulutusohjelma Tietotekniikka		
Työn nimi AUTOPILOTTI VENEESEEN		
Työn ohjaaja Hannu Puomio		**Sivumäärä** 29 + 8
Työelämäohjaaja Ritva Saviluoto		

Opinnäytetyön aiheena oli rakentaa rattikäyttöiseen moottoriveneeseen toimiva autopilotti, joka pitää veneen kurssin suorassa. Laitteen tarkoitus on helpottaa vetouistelua.

Työ toteutettiin käyttäen ohjelmoitavaa Arduino UNO alustaa, jolla ohjataan veneen rattiakseliin kytkettyä moottoria. Veneen suuntatiedon saamiseksi käytetään elektronista kompassia.

Työssä keskeisenä ajatuksena oli toteuttaa se mahdollisimman edullisesti.

Asiasanat
Arduino, autopilotti, elektroniikka, ohjelmointi

ABSTRACT

CENTRIA UNIVERSITY OF APPLIED SCIENCES	Date April 2013	Author Ville Pentikäinen
Degree programme Telecommunications Technology		
Name of thesis AUTOPILOT FOR MOTORBOAT		
Instructor Hannu Puomio		**Pages** 29+8
Supervisor Ritva Saviluoto		

Objective of this thesis was to build an autopilot for a motorboat, which keeps the boat's course straight. The device is intended to help trolling (a method of fishing).

The work was carried out using a programmable Arduino UNO platform, which controls a motor that is connected to boat's steering axle. Boat heading information is obtained from electronic compass.

The central idea of this work was to implement it as cheaply as possible.

Key words
Arduino, autopilot, electronics, programming

KÄSITTEIDEN MÄÄRITTELY

Arduino	Ohjelmoitava mikrokontrolleri-elektroniikka alusta
Mikrokontrolleri	Pieni ”tietokone”
I2C	Kaksisuuntainen tiedonsiirtoväylätekniikka
GPS	Global Positioning System, satelliittipaikannusjärjestelmä
Kommutaattori	Sähkövirran suunnankääntäjä
Ohmi (Ω)	Resistanssin, reaktanssin ja impedanssin mittayksikkö
C++	Ohjelmointikieli

KUVIOT

1 JOHDANTO

Vetouistelu on kalastusmuoto, jossa viehettä vedetään moottorikäyttöisellä veneellä tasaista vauhtia veneen perässä. Vieheitä voi olla veneen perässä useita samaan aikaan. Vavoille on yleensä rakennettu teline veneeseen, johon ne saadaan laitettua kätevästi paikoilleen vedon ajaksi. Kun vapoja lisätään tarpeeksi, joudutaan käyttämään apuna plaanareita. Ne ovat apuvälineitä, jotka levittävät siimoja kauemmas veneestä ja toisistaan, etteivät ne sotkeudu keskenään. Ongelmana usean vavan vetouistelussa on juuri siimojen sekoittuminen keskenään veneellä käännyttäessä sekä kalan tarttuessa vieheeseen. Kalan tarttuessa vieheeseen venettä ei voida pysäyttää ylösvedon ajaksi, sillä muut vieheet uppoavat tällöin pohjaan. Tarvitaan siis joku ohjaamaan venettä samaan aikaan.

Kaupallisia laitteita tähän tarkoitukseen on olemassa, mutta ne ovat todella kalliita. Tästä sain idean rakentaa mahdollisimman edullisesti vastaavan laitteen, joka voidaan kytkeä helposti napista päälle, ja joka ohjaa venettä suoraan automaattisesti. Tämä helpottaa usean vavan vetouistelua yksinään huomattavasti.

Työn pohjana päädyin käyttämään ohjelmoitavaa Arduino UNO mikrokontrollerialustaa muun muassa sen edullisen hinnan ja helppokäyttöisyyden takia. Jotta venettä voitaisiin ohjata halutulla tavalla, tarvitaan veneen kulkusuuntatieto mikrokontrollerille. Tähän tarkoitukseen käytin elektronista kompassipiiriä, josta saadaan suuntatieto I2C -väylää pitkin siirrettyä Arduinon mikrokontrollerille. Rattiakseliin tarvitaan moottori sitä kääntämään ja myös asentotunnistin, joka kertoo mikropiirille ratin senhetkisen asennon.

Kun laite kytketään päälle, mikrokontrolleri ottaa veneen senhetkisen suunnan muistiin kompassilta ja alkaa verrata nykyistä suuntaa siihen arvoon. Tämän perusteella ohjataan moottoria kääntämään rattia oikeaan suuntaan. Asentotunnistimen arvon perusteella ratti saadaan pysäytettyä haluttuun asentoon.

2 LAITTEEN RAKENTAMINEN JA KÄYTETYT KOMPONENTIT

Luvussa käydään läpi laitteen rakennuksessa käytetyt osat, sekä niiden toiminta ja kytkentä Arduino UNO alustalle. Luvussa kolme käydään läpi Arduino UNO mikrokontrolleri ja sen ohjelmointi.

2.1 Kompassi

Laitetta suunnitellessani päädyin käyttämään veneen suuntatiedon saamiseksi elektronista kompassipiiriä. Sen etuja esimerkiksi Global Positioning System (GPS) satelliittipaikannukseen verrattuna ovat nopeus ja suuntatiedon saaminen ilman liikettä. GPS-paikannin tarvitsee satelliittiyhteyden ja tietyllä nopeudella tapahtuvan liikkeen, jotta laite voi laskea kulkusuunnan.

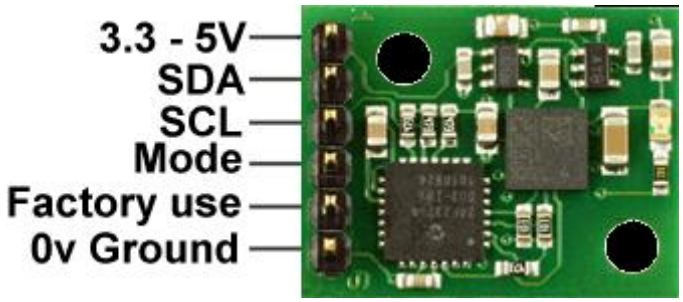
Kompassi toimii havaitsemalla maan magneettikenttiä. Maapallolla on rautasydän, joka on osaksi nestemäistä ja osaksi kiinteää. Uskotaan, että nestemäisen raudan liikkuminen ytimessä aiheuttaa maan magneettikentän. Kuten kaikissa magneettikentissä, myös maan magneettikentällä on kaksi napaa: pohjois- ja etelänapa. Nämä magneettiset navat ovat hieman erikohdassa maan pyörimisakselin navoista, mutta kuitenkin tarpeeksi lähellä, jotta yleisiä suuntia korjauksien avulla voidaan käyttää navigointiin. Näiden napojen välistä kulmaa kutsutaan erannoksi eli magneettiseksi deklinaatioksi. (Jessa 2010.)

Eranto on yleensä merkitty karttoihin. Koska tässä työssä sillä ei ole merkitystä, sitä ei käsitellä enempää. Tavallisessa kompassissa on magnetisoitu neula, joka pyörii vapaasti keskitapin ympärillä. Koska magneetin eri navat vetävät toisiaan puoleensa, kääntyy neulan negatiivinen puoli magneettista pohjoista kohden.

Elektroninen kompassi koostuu sensoreista, jotka mittaavat magneettikentän voimakkuuksia. Se on siis magnetometri, jonka sensorien avulla lasketaan magneettikentän horisontaalinen suunta.

Normaalissa kaksi sensorisessa magnetometripiirissä mitataan magneettikentän x- ja y-komponentteja. Näiden avulla saadaan laskettua suunta suhteessa magneettiseen pohjoiseen. Ongelmaksi muodostuu se, ettei piiri toimi kuin vaakatasossa maanpintaan nähden. Veneessä ollessaan kaksi sensorinen piiri ei käytännössä toimi luotettavasti, koska aallokko liikuttaa venettä myös pystysuunnassa. Tarvitaan siis kolme sensorinen magnetometri, joka mittaa x- ja y-komponenttien lisäksi myös pysty komponenttia z. Näiden kolmen sensorin arvojen avulla voidaan laskea piirin asennon aiheuttama virhe pois ja saada todellinen magneettikentän suunta selville.

Työhön valitsin kolmesensorisen CMPS10 magnetometrimoduulin, josta löytyy tilt-kompensointi sisäänrakennettuna. Pienikokoinen kompassimoduuli toimii viidellä voltilla ja siitä saadaan I2C väylää pitkin suuntatieto 0-3599 asteikolla ulos. Tilt korjaus toimii 60 asteeseen asti, joka riittää hyvin tähän käyttöön. Vene voi siis olla 60 asteen pystykulmassa ilman, että kompassin antama suunta häiriintyy. Kompassipiiri tulee veneen keulaan vedenpitävään koteloon mahdollisten sähkölaitteista syntyvien häiriöiden minimoimiseksi.



KUVIO 1. CMPS10 kompassi moduuli (Robot-Electronics 2012)

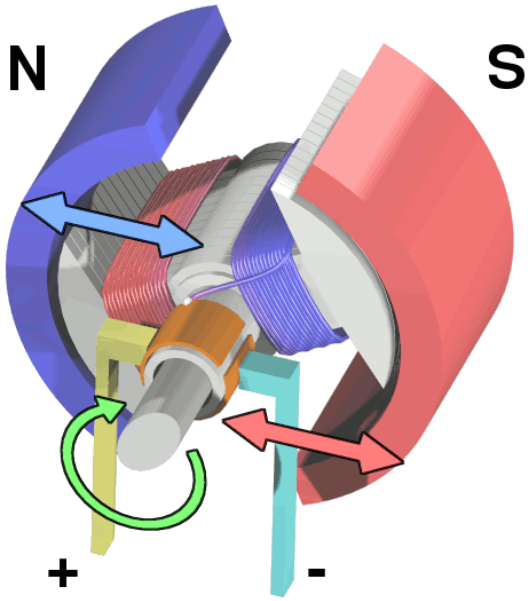
2.2 Kääntömoottori

Veneen kääntämiseksi haluttuun suuntaan tarvitaan sähkömoottori kääntämään rattia. Kyseisen veneen ohjaus on toteutettu suoraan vaijerilla moottorilta ratille, mikä tekee siitä suhteellisen raskaan käännettävän. Tämän vuoksi moottorin täytyy olla tehokas, jotta se jaksaa kääntää rattia kaikissa olosuhteissa.

Ensimmäisenä vaihtoehtona moottoriksi suunnittelin askelmoottoria. Askelmoottori on sähkömoottori, joka ei pyöri vapaasti, vaan sitä ohjataan askelittain. Moottorin yksi kierros jaetaan siis useampaan askeleeseen, joiden mukaan moottori voi pyöriä. Askelmoottoria on ohjattava erillisellä ohjauspiirillä, joka säätelee mihin suuntaan ja kuinka paljon moottori pyörii. Askelmoottorin toiminta perustuu sen sisällä oleviin sähkömagneetteihin. Niiden magneettikenttää muuttelemalla saadaan moottori liikkumaan. Kuhunkin magneettiin vuorollaan johdetaan virtaa, jolloin se vetää puoleensa moottorin sisällä pyörivää roottoria. (Kompo2010 2010.)

Askelmoottorin huono puoli on heikko vääntömomentti. Tarpeeksi tehokkaat askelmoottorit olisivat olleet todella kalliita ja vaikeasti saatavilla. Näistä syistä päädyin käyttämään työssä normaalia tasasähkömoottoria. Tasasähkömoottorin toiminta perustuu magneettisten samanpolaaristen napojen hylkivään ja eripolaaristen napojen toisiaan puoleensa vetävään voimaan. Sähkömagneetin, joka on käytännössä käämi, napojen polaarisuutta voidaan vaihtaa virran kulkusuuntaa muuttamalla, mahdollistaen jatkuvan liikkeen. (Kompo2010 2010.)

Yleensä tasavirtamoottorissa ankkurikäämin kommutaattorille johdetaan sähkövirta hiilien kautta. Tasavirtamoottori saadaan pyörimään toiseen suuntaan kääntämällä virransyötön napaisuus toisinpäin.

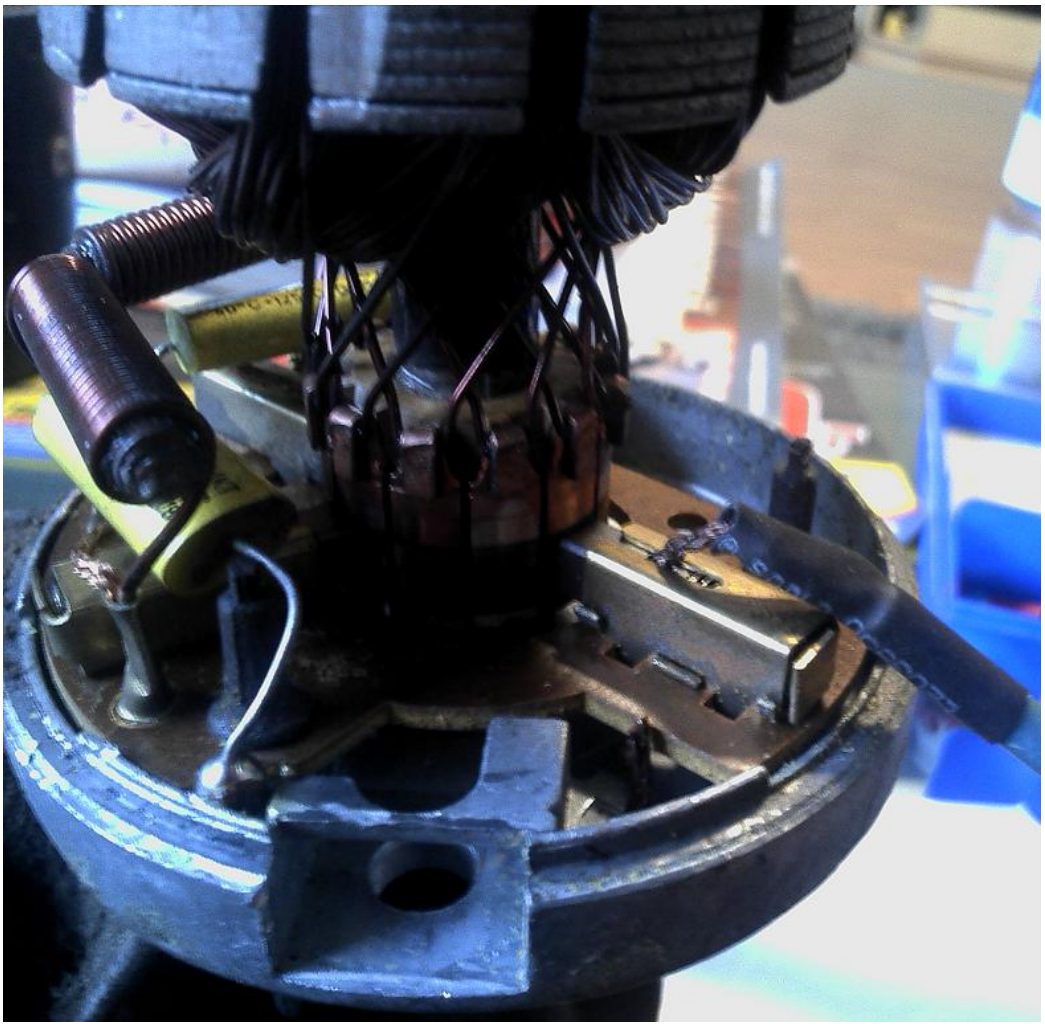


KUVIO 2. Tasavirtamoottorin toimintaperiaate (D&D Motor Systems 2007)

Tarpeeksi tehokkaan, edullisen ja sopivankokoisen tasavirtamoottorin sain henkilöauton tuulilasinpyyhkijän moottorista. Moottorissa on kolme hiiltä, joilla se saadaan pyörimään kahta eri nopeutta riippuen siitä, mitä hiilien etäisyyttä käytetään.

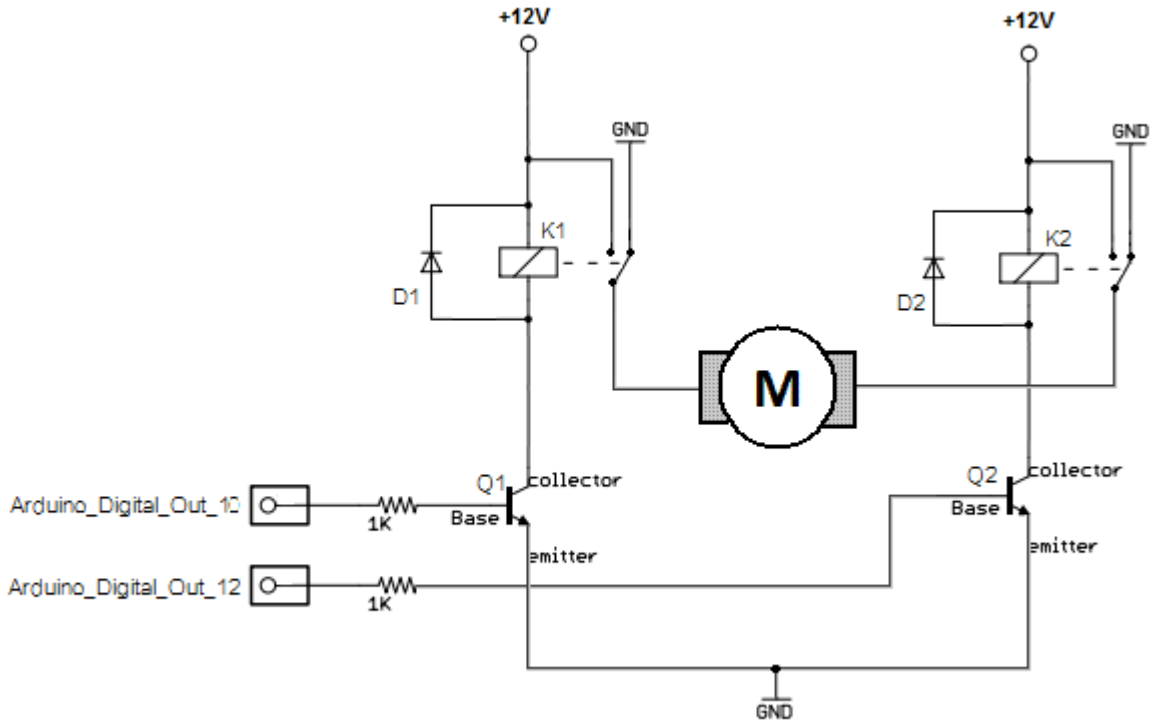
KUVIO 3. Tuulilasinpyyhkijän moottori

Moottorin yksi hiili oli kytketty suoraan moottorin runkoon, koska autossa runko on kytketty akun miinusnapaan. Tässä käyttötarkoituksessa kyseinen kytkentä aiheuttaa oikosulun moottoriin syötettävän sähkön napaisuutta muutettaessa. Jouduin irrottamaan hiilen moottorin rungosta ja jatkamaan sen johdolla kotelon ulkopuolelle.



KUVIO 4. Tuulilasinpyyhkijän moottorin muokkausta

Moottorin ohjauksen toteutin kahdella vaihtokosketinreleellä, joilla saadaan moottorin syötön napaisuus käännettyä ja moottori pyörimään haluttuun suuntaan. Moottori toimii 12 voltilla, joten sen ohjaukseen käytin 12 volttisia vaihtokosketinreleitä. Arduinolla ei voida suoraan ohjata releitä, koska sen ”output” -portista saadaan maksimissaan 5 voltin jännite. Releiden ohjaukseen käytin kahta transistoria, joita ohjataan kytkemällä transistorin kantaan liitetty Arduinon ”output” portti ylös. Porttiin tulee tällöin 5 voltin jännite, joka kytkee transistorin johtavaan tilaan toimien niin sanotusti kytkimenä releelle.



KUVIO 5. Moottorin kytkentäkaavio

Kun molemmat Arduinon ”output” portit (KUVIO 5) ovat ”0” –tilassa, transistorien kautta ei kulje virtaa. Releet eivät tällöin vedä ja moottorin molemmat navat ovat kytkettynä maahan. Kun Arduinon ”Out_10” portti ohjataan ”1” -tilaan, transistori Q1 alkaa johtaa ja rele K1 vetää. Tällöin moottorin toiseen napaan tulee +12V toisen ollessa edelleen maissa. Moottori alkaa tällöin pyöriä vasemmalle. Vastaavasti ”Out_10” portin ollessa tilassa ”0” ja ”Out_12” portin tilassa ”1” moottori pyörii oikealle napaisuuden ollessa käänteinen. Mikäli molemmat portit ajetaan tilaan ”1”, moottori ei pyöri molempien napojen ollessa kytkettynä +12V tuloon. Arduinon porttien ja transistorien välissä käytin 1kΩ (ohmi) kokoisia vastuksia sekä releiden yli 1N4004 suojadiodeja. Transistoreina riittäisi tehonkestoltaan esimerkiksi TIP102 tai vastaava NPN transistori. Tässä käytin ylimääräisenä löytyneitä tehonkestoltaan isompia TIP41C tehotransistoreita, jotka toimivat kytkennässä kuitenkin samoin.

2.3 Magneettikytkin

Moottorin asentamisen veneen rattiakseliin suunnittelin toteuttavani ketjulla ja hammaspyörillä. Autopilotin ollessa pois päältä venettä on oltava mahdollista ohjata käsin normaalisti. Tuulilasinpyyhkijän moottori on kuitenkin käytännössä jumissa, kun siihen ei syötetä sähköä. Tämä johtuu moottorissa olevasta alennusvaihde hammastuksesta, jolla saadaan moottoriin enemmän vääntöä. Mikäli moottoriin liittää hammaspyörän ja kytkee sen suoraan ketjulla veneen rattiakseliin, rattia ei voi kääntää käsin.

Ongelmaan ratkaisuksi purin henkilöauton ilmastoinnin kompressorista hihnapyörän magneettikytkimen. Henkilöautossa ilmastoinnin kompressorille saadaan pyörimisliike moottorilta hihnalla. Kompressorin hihnapyörässä on magneetilla toimiva kytkin. Ilmastoinnin ollessa pois päältä hihnapyörän ulkokehä pyörii tyhjää laakerin avulla sisäosan pysyessä paikoillaan. Kun ilmastointi kytketään autossa päälle, syötetään hihnapyörän magneettiin sähkövirta, jolloin magneetti lukitsee pyörän ulkokehän kiinni sisäosaan. Tällöin kompressori alkaa pyöriä hihnan mukana.

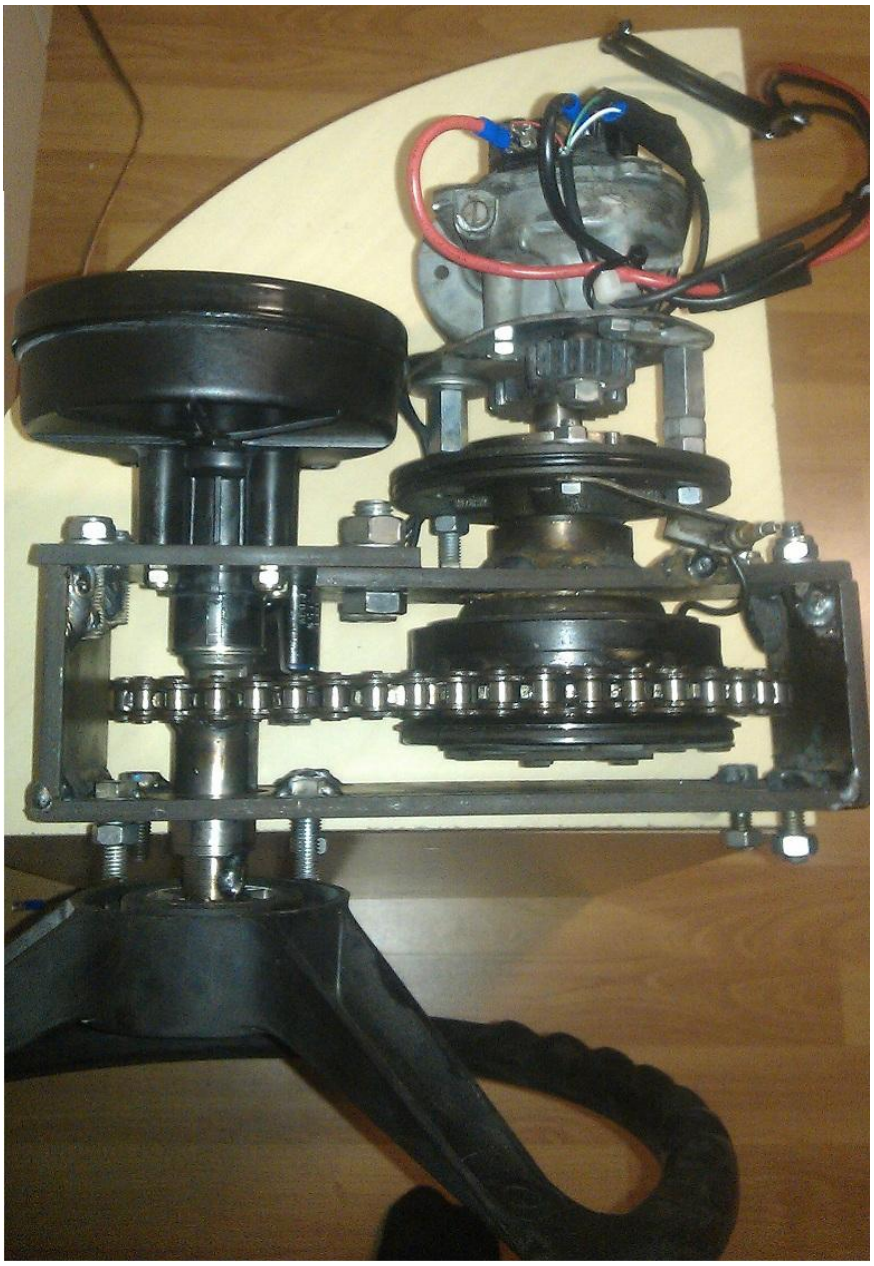
KUVIO 6. Magneettikytkin purettuna

Magneettikytkimen hihnapyörän ulkokehälle täytyi hitsata hammastus, jotta sitä voidaan käyttää ketjulla. Kun magneettiin ei johdeta sähköä, rattia voidaan kääntää liikkeen välittymättä moottoriin. Kytkettäessä autopilotti päälle, ohjataan magneettikytkimeen samalla +12 voltin jännite, jolloin akselit lukittuvat toisiinsa ja ratti pyörii moottorin avulla.

Haittapuolena kyseisessä magneettikytkimessä on sen suuri koko. Kytkintä ei voi laittaa veneen rattiakseliin suoraan kiinni, koska se ei mahtuisi kojelaudan sisään tilan ollessa suhteellisen ahdas. Magneettikytkimelle jouduttiin tekemään oma akseli, josta se yhdistettiin veneen rattiakseliin ketjulla.

KUVIO 7. Ohjauslaite edestä

Moottorin päähän on lisätty hammasratas, jolla se pyörittää magneettikytkimen akselia siihen hitsatun hammaspyörän avulla. Laitteelle on tehty runko lattaraudasta ja se on rakennettu sopimaan veneen alkuperäisen ratin paikalle. Koko laite menee veneen kojelaudan sisään piiloon, eikä näkyville jää kuin ratti. Lattarunko tulee kiinni kojelautaan pulteilla.



KUVIO 8. Ohjauslaite päältä

2.4 Ratin asentoanturi

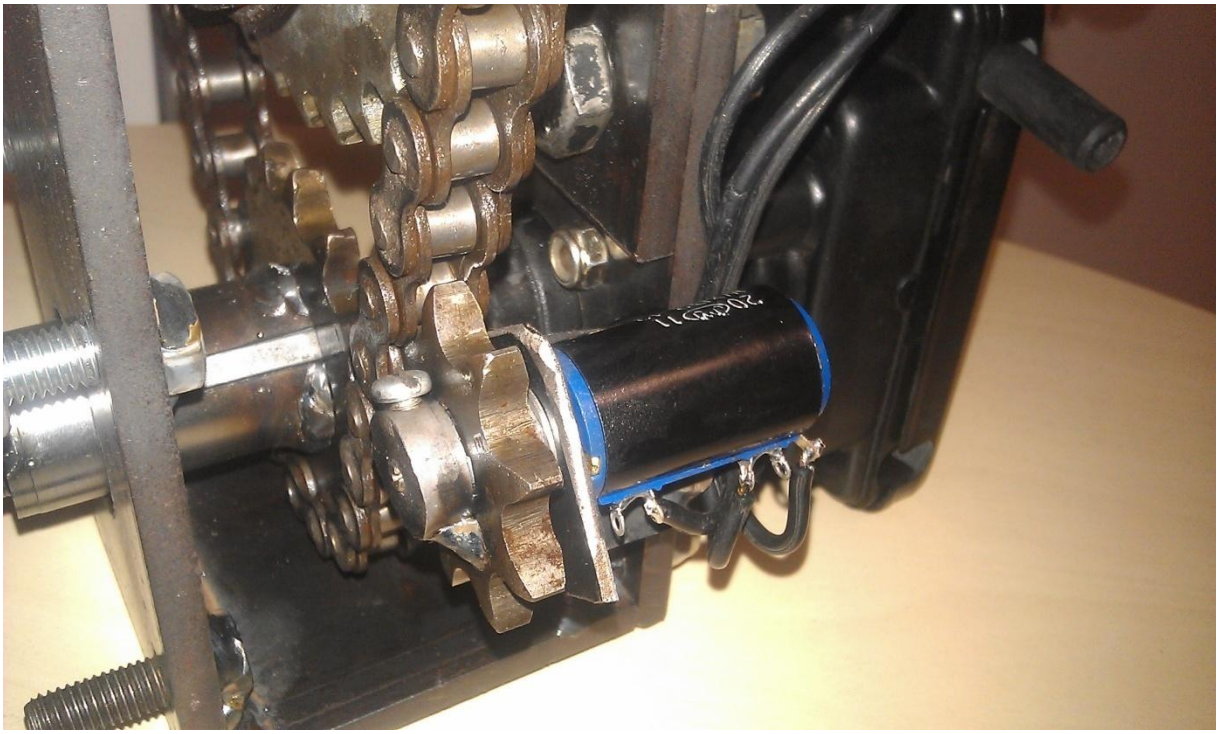
Autopilotin ollessa päällä Arduino tarvitsee tiedon ratin asennosta. Tähän tarkoitukseen käytin lineaarista monikierroksista potentiometria (KUVIO 9). Potentiometri on portaattomasti säädettävä vastus, jonka vastusarvo muuttuu sen akselia pyöritettäessä.

Syöttämällä potentiometrin vastuksen päihin 5V jännite voidaan Arduinolla mitata sen keskijalasta analogiportin kautta säätövastuksen arvo kussakin akselin kohdassa. Arduino mittaa potentiometrin arvon väliltä 0-1023, mikä on suoraan verrannollinen porttiin tulevaan jännitteeseen (Arduino 2013).

KUVIO 9. Potentiometri

Potentiometrissä on kymmenen kierrosta ja resistanssin arvo vaihtelee välillä 0-1kΩ. Kyseisessä veneessä ratti kääntyy laidasta laitaan vain muutaman kierroksen, joten potentiometri riittää hyvin. Ratin ollessa keskellä potentiometrin resistanssiarvo on 500Ω.

Potentiometrin päähän laitetun hammasrattaan avulla se on laitettu mittaamaan ratin asentoa ketjusta (KUVIO 10).



KUVIO 10. Potentiometri paikoillaan

3 ARDUINO MIKROKONTROLLERI

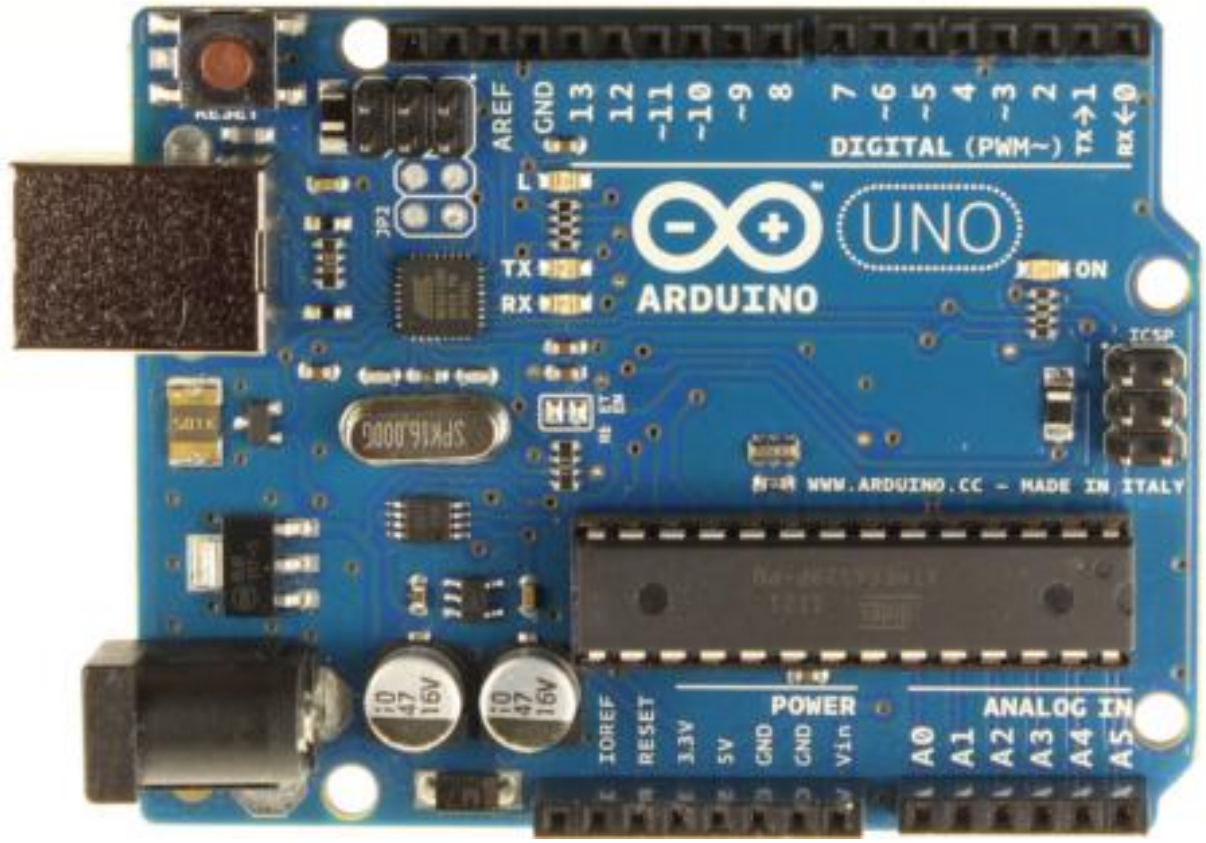
3.1 Yleistä

Arduino on ohjelmoitava mikrokontrolleri-elektroniikka alusta. Se on suunniteltu lähinnä harrastekäyttöön helpottamaan monimutkaistenkin ohjelmointia tarvitsevien projektien rakentamista. Laitteen idea on lähtöisin Italiasta opiskelijoille suunnatusta projektista, jonka tarkoituksena oli rakentaa edullinen sekä helppokäyttöinen kytkentä-ohjelmointiympäristö opiskelu tarkoituksiin. Laitetta valmisti aluksi Smart Projects niminen yhtiö, mutta sen perustuessa avoimeen laitteistoon, sitä on myöhemmin valmistanut useampi yritys. (Lahart 2009).

Arduino on 8-bittisellä Atmel AVR suorittimella toimiva piirilevy, jossa oleviin kytkentänastoihin voidaan kytkeä haluamiaan komponentteja. Digitaalisia nastoja voidaan käyttää sisään- tai ulostuloina. Levyltä löytyy niin digitaalisia kuin analogisia portteja. Laitteiston ohjelmointikieli perustuu C++ -kieleen. (Arduino 2013).

Laitteesta löytyy useita eri versioita, jotka eroavat toisistaan lähinnä muistin sekä kytkentäpinnien määrällä. Ensimmäisissä Arduino versioissa ohjelmointi tapahtui sarjaportin kautta, mutta nykyisissä laitteissa se on korvattu USB -väylällä. Arduinon oma ohjelmointiympäristö löytyy lähes kaikille käyttöjärjestelmille.

Tässä työssä käytin Arduino Uno versiota laitteesta (KUVIO 11). Arduino Unosta löytyy tarvittavat ominaisuudet ja hinta oli edullisempi kuin sen isommissa versioissa.



KUVIO 11. Arduino Uno (Arduino 2013)

Laitteen Tiedot:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

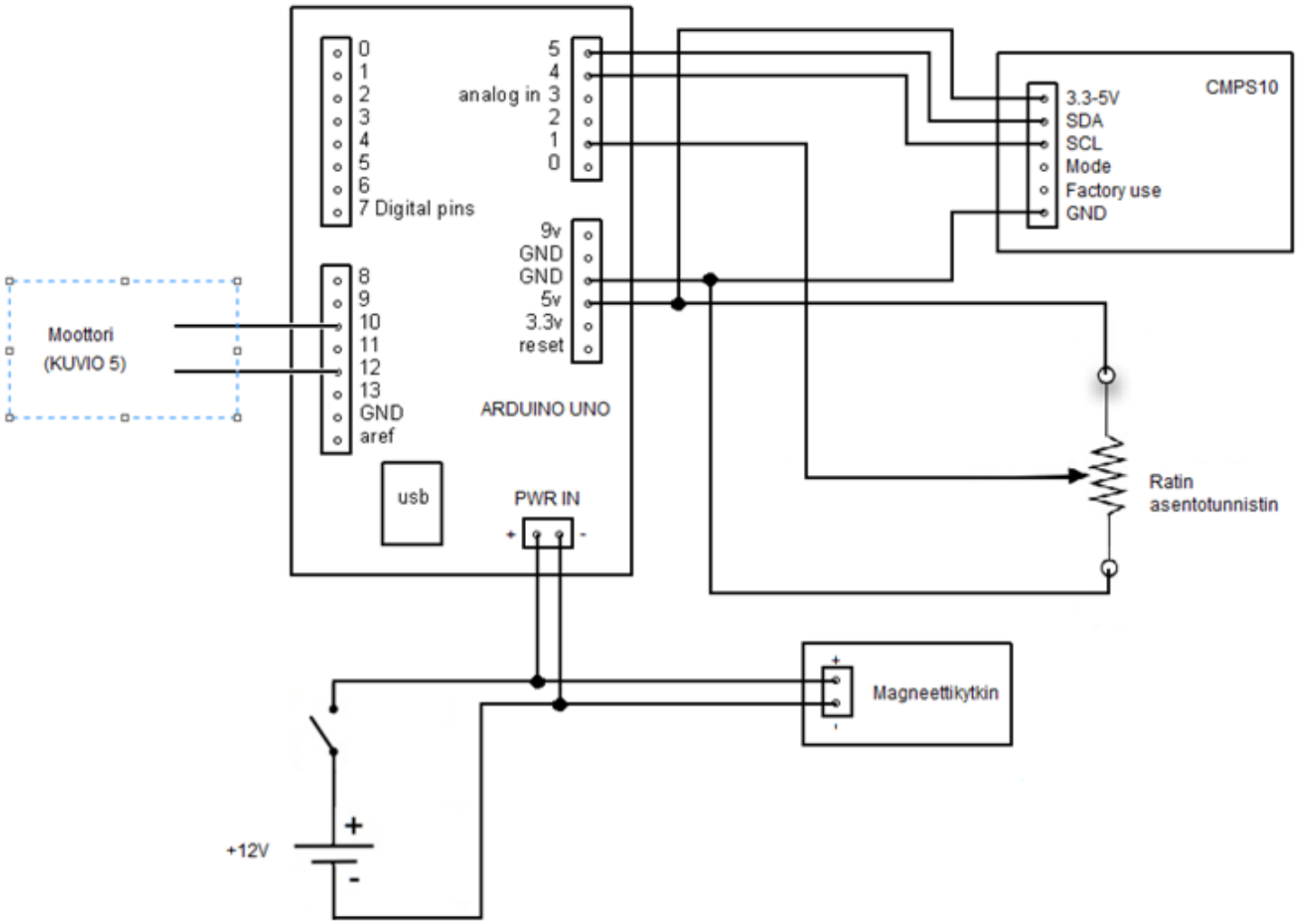
Arduino Uno levyltä löytyy digitaalisia I/O pinnejä neljätoista kappaletta ja analogisia sisääntuloja kuusi kappaletta. Työssä releiden ohjaus hoituu digitaalisilla pinneillä kytkemällä portti päälle tai pois. Kompassimoduuli ja asentotunnistin potentiometri kytketään analogisiin pinneihin. Kompassin käyttämä I2C -väylä tarvitsee Serial Data Line (SDA) sekä Serial Clock (SCL) signaalit. Arduinosta löytyy 5 voltin syöttö, josta virta riittää kompassille ja potentiometrille. Veneen kojetauluun tuleva päävirtakytkin käynnistää Arduinon ja magneettikytkimen samaan aikaan.

3.2 Virtalähde

Arduino UNO toimii viiden voltin jännitteellä. Virtaa laite tarvitsee siihen kytketyistä laitteista riippuen muutamia satoja milliampeereja. Laitteeseen voidaan syöttää virta USB -portin kautta tai piirilevyllä olevaan virtapistokkeeseen. Laite valitsee virransyötön automaattisesti. Virtapistokkeen kautta laitteeseen suositellaan syötettävän 7-12 voltin jännitettä, jotta jänniteregulaattori ei lämpene liikaa. Pienemmällä jännitteellä laitteen toiminta muuttuu epävakaaksi, eikä uloslähtöihin saada tarpeeksi jännitettä.

Veneestä löytyy 12 voltin akku, mutta moottorin ollessa käynnissä laturi nostaa akun jännitteen 14.4 volttiin. Virransyötön Arduinolle toteutin muokkaamalla henkilöauton tupakinsytyttimeen tulevan säädettävän virtalähteen kytkennälle sopivaksi juottamalla siihen Arduinoon sopivan virtapistokkeen ja veneen akkuun kytkettävät virransyöttökaapelit. Virtalähteessä on jänniteregulaattori, jolla jännitteen saa tasattua haluamaan arvoon. Totesin 9 volttia riittävän laitteen vakaaseen toimintaan.

3.3 Kytkentäkaavio



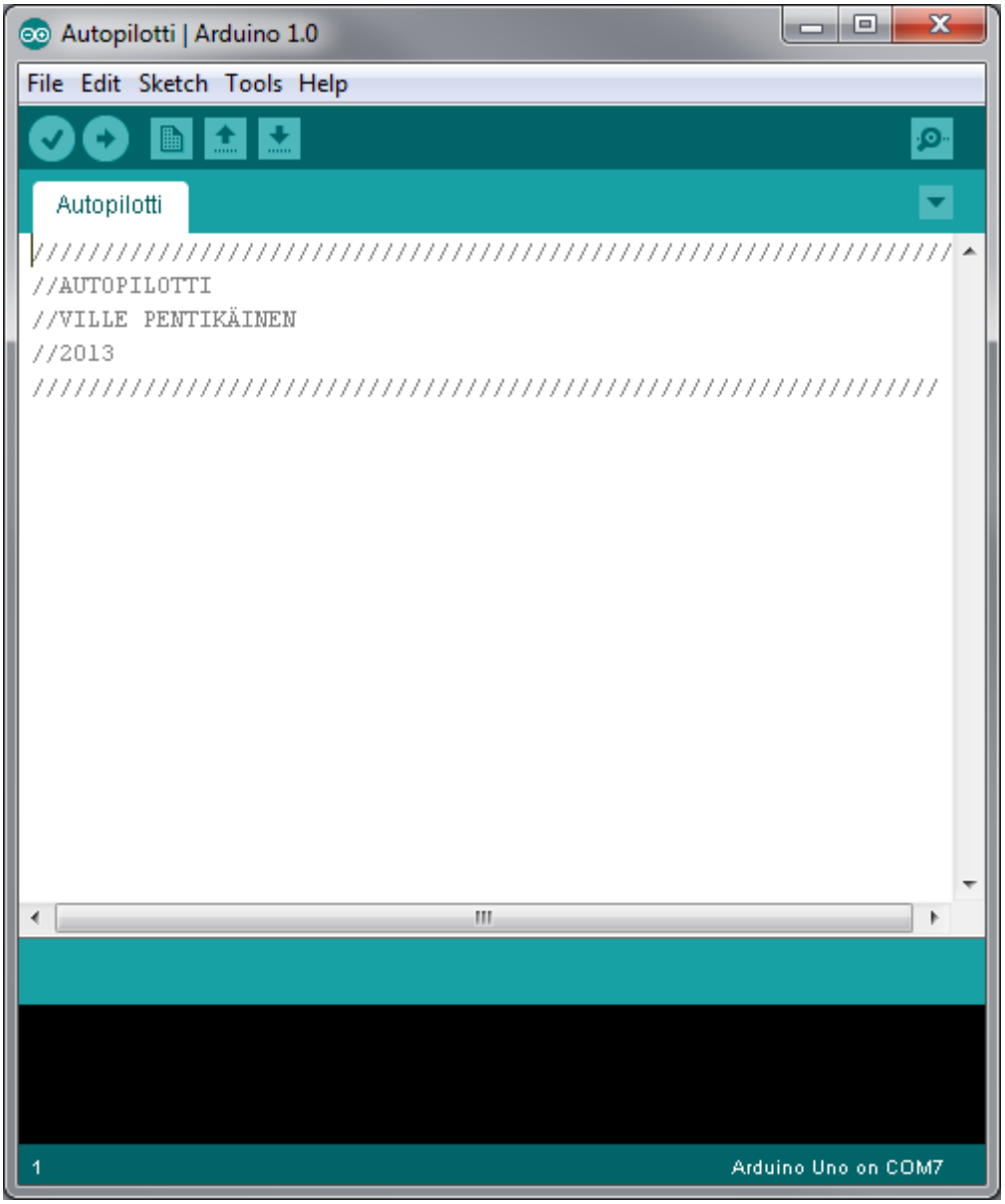
KUVIO 12. Arduino Uno kytkentäkaavio

Virtalähteen, Arduinon sekä moottorinohjauksen transistorit ja releet asensin roiskevedenpitävään kytkentärasiaan (KUVIO 13). Rasian saa veneessä kiinnitettyä vedeltä suojaan kojelaudan alle. Tulevaisuudessa työlle voisi harkita piirilevyn tekemistä siistimmän lopputuloksen saamiseksi. Arduinon ohjelmointi onnistuu USB -väylän kautta laitteen ollessa rasiassa. Rasiasta lähtee virtakaapelit sekä kompassin, potentiometrin ja moottorin johdot.

KUVIO 13. Ohjausyksikkö

3.4 Ohjelmointi

Arduino Unon ohjelmointi tapahtuu Arduinon kotisivuilta ladattavalla omalla ohjelmointiympäristöllä (KUVIO 14). Pienikokoinen ja kevyt ohjelma sisältää tekstieditorin koodin kirjoitukseen, viestikonsoli-ikkunan, työkalurivin ja valikot. Ohjelma tunnistaa USB -porttiin kytketyn Arduino-alustan ja koodin siirto onnistuu nappia painamalla tietokoneelta Arduinoon tai toisinpäin.



KUVIO 14. Arduino ohjelmointiympäristö

Ohjelmointikieli pohjautuu C++ -kieleen ja valmiita esimerkkikoodeja löytyy Arduinon kotisivulta runsaasti.

Aluksi koodissa määritellään käytettävät kirjastot, sekä kytketyt laitteet ja niiden portit. Kirjasto ”Wire.h” mahdollistaa I2C väylän käytön. LEFT_PIN sekä RIGHT_PIN ovat releiden ohjausportit, joilla moottori saadaan pyörimään valittuun suuntaan. WHEEL_PIN portista luetaan ratin asentotunnistin potentiometrin arvo.

```
#include <Wire.h>
#define ADDRESS 0x60                    //Defines address of CMPS10
#define LEFT_PIN 12                     //Define PIN names
#define RIGHT_PIN 10
#define WHEEL_PIN 1

int wantedDir=NULL, currentDir;
int wheelpos=NULL;
```

Seuraavaksi määritellään väylän nopeus ja digitaaliporttien tyyppi

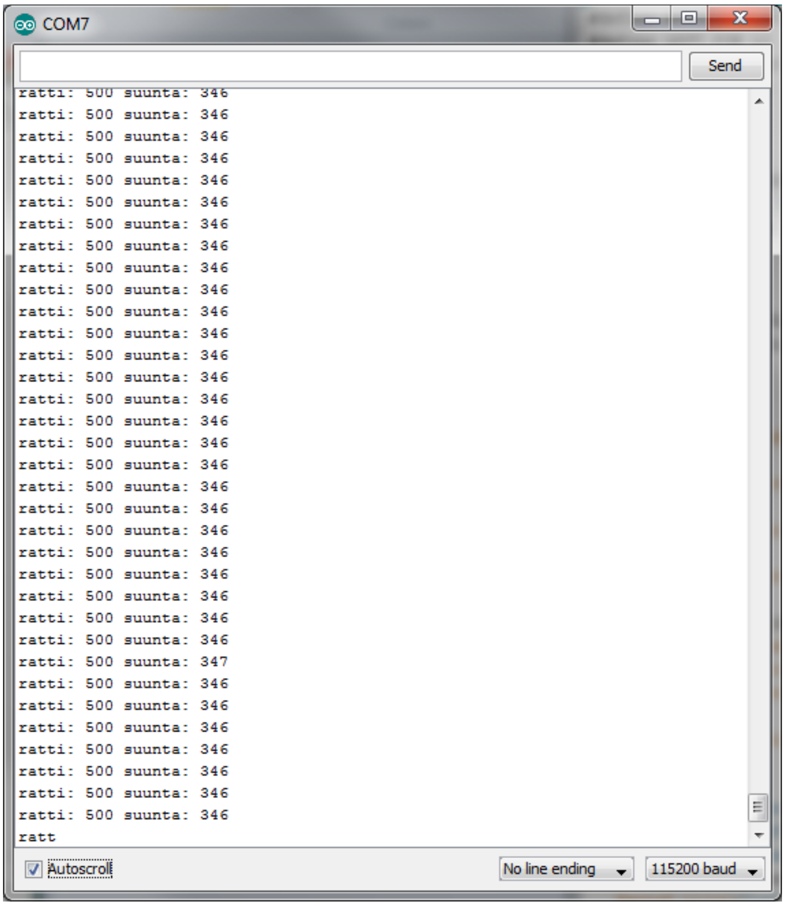
```
void setup() {

// Connects I2C
  Wire.begin();
  Serial.begin(115200);
  pinMode(LEFT_PIN, OUTPUT);
  pinMode(RIGHT_PIN, OUTPUT);

}
```

Ohjelmasta tein ikuisen silmukan, joka käynnistyy välittömästi, kun laite kytketään päälle. Käyttäjältä ei vaadita mitään muuta kuin laitteen kytkeminen päälle ja pois. Alussa on määritelty käytettävät muuttujat. Kun laite kytketään päälle, Arduino ottaa muistiin kompassilta veneen senhetkisen suunnan ja tallentaa sen muuttujaan ”wantedDir”. Sen jälkeen nykyistä suuntaa, joka tallentuu muuttujaan ”currentDir”, aletaan verrata ”wantedDir” arvoon. Ratin asentotunnistimen arvoa luetaan ”wheelpos” muuttujaan.

```
void loop(){

   wheelpos = analogRead(WHEEL_PIN);

   byte highByte, lowByte;    // highByte and lowByte store high and
                              // low bytes of the bearing and fine
                              // stores decimal place of bearing
   int bearing;               // Stores full bearing

//starts communication with CMPS10
    Wire.beginTransmission(ADDRESS);

//Sends the register we wish to start reading from
   Wire.write(2);
   Wire.endTransmission();

// Request 4 bytes from CMPS10
   Wire.requestFrom(ADDRESS, 4);

// Wait for bytes to become available
    while(Wire.available() < 4);
       highByte = Wire.read();
       lowByte = Wire.read();

// Calculate full bearing
   currentDir = ((highByte<<8)+lowByte)/10;
```

Ohjelman testausvaiheessa "Serial.print" komennolla saa tulostettua Arduinon serial monitor ikkunaan (KUVIO 15) haluttujen muuttujien arvot näkyviin, mitkä helpottavat ohjelmointia.

```
Serial.print("ratti: ");
Serial.print(wheelpos);
Serial.print(" suunta: ");
Serial.print(currentDir);
Serial.print("\n");
```



KUVIO 15. Arduino Serial Monitor

Suuntatieto tulee kompassimoduulilta arvoina 0-359, joten funktion “doSmartTurn” sisälle tuodut arvot pakotetaan näihin arvoihin.

```
    if(wantedDir == NULL){
       wantedDir = currentDir;
    }

   doSmartTurn(currentDir, wantedDir);

}
```

Halutusta suunnasta ja nykyisestä suunnan arvosta lasketaan erotus, jonka perusteella tehdään oikea kääntöliike. Ongelmaksi tässä kohtaa muodostui halutun ja nykyisen suunnan laskussa negatiivinen tulos. Suunnan sattuessa olemaan juuri 359 ja 1 asteen paikkeilla kompassin kääntyessä ohjelma meni sekaisin ja rupesi kääntämään rattia täyden ympyrän väärään suuntaan korjatakseen muutaman asteen muutoksen. Tähän ratkaisuksi muutoksesta täytyi laskea itseisarvo ”abs()”, jotta negatiivista arvoa ei pääse syntymään muuttujaan.

```
void doSmartTurn(unsigned int curDirection, unsigned int newDirection)
{

// force directions between 0..359
  curDirection %= 360;
  newDirection %= 360;

  int delta = newDirection - curDirection;

  if (abs(delta) <= 10 ){
    if (wheelpos < 490) {
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }

    if (wheelpos > 510) {
     digitalWrite(LEFT_PIN, HIGH);
     digitalWrite(RIGHT_PIN, LOW);
     delay(10);
```

```
    }
    if (wheelpos >= 490 && wheelpos <= 510) {
     digitalWrite(LEFT_PIN, LOW); digitalWrite(RIGHT_PIN, LOW);
    delay(10);
    }

// don't do small angles.
  return;
}

  if (delta > 180) turnLeft(360 - delta);
  else if (delta > 0) turnRight(delta);
  else if (delta > -180) turnLeft(-delta);
  else turnRight(360+delta);
}
```

Koska muutaman asteen muutoksella kurssissa ei ole mitään käytännön merkitystä, ohjelmasta ei kannattanut tehdä jatkuvasti korjaavaa. Käännöt toteutin kahdella erikokoisella kääntöliikkeellä. Pieni poikkeama kurssista kääntää rattia vähän ja iso poikkeama kääntää rattia enemmän. Kun vene alkaa palaamaan takaisin kurssiin isosta poikkeamasta, ohjelma kääntää ratin pienen käännön asentoon jne.

Käännöt vasemmalle ovat toteutettu koodissa seuraavalla tavalla:

```
void turnLeft(unsigned int Delta){

//small turn left
  if(Delta <= 20){
    if(wheelpos > 350){
      digitalWrite(LEFT_PIN, HIGH);
      digitalWrite(RIGHT_PIN, LOW);
      delay(10);
    }
    else if (wheelpos < 330){
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }
    else{
    digitalWrite(RIGHT_PIN, LOW);
    digitalWrite(LEFT_PIN, LOW);
    delay(10);
    }
  }

//big turn left
    if(Delta > 20){
    if(wheelpos > 220){
      digitalWrite(LEFT_PIN, HIGH);
      digitalWrite(RIGHT_PIN, LOW);
      delay(10);
    }
    else if (wheelpos < 200){
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }
    else{
    digitalWrite(LEFT_PIN, LOW);
    digitalWrite(RIGHT_PIN, LOW);
    delay(10);
  }
```

```
 }
}
```

Käännöt oikealle tapahtuvat samalla periaatteella:

```
void turnRight(unsigned int Delta){

//small turn right
  if(Delta <= 20){
    if(wheelpos > 670){
      digitalWrite(LEFT_PIN, HIGH);
      digitalWrite(RIGHT_PIN, LOW);
      delay(10);
    }

    else if (wheelpos < 650){
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }
    else{
    digitalWrite(RIGHT_PIN, LOW);
    digitalWrite(LEFT_PIN, LOW);
    delay(10);
    }
  }

//big turn right
    if(Delta > 20){
    if(wheelpos > 820){
      digitalWrite(LEFT_PIN, HIGH);
      digitalWrite(RIGHT_PIN, LOW);
      delay(10);
    }
    else if (wheelpos < 800){
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }
    else{
```

```
    digitalWrite(LEFT_PIN, LOW);
    digitalWrite(RIGHT_PIN, LOW);
    delay(10);
  }
}
}
```

Arduino ohjelmointiympäristöstä löytyy VERIFY -painike, jolla koodi saadaan tarkistettua. Valmiin koodin siirto USB -porttiin kytkettyyn Arduinoon tapahtuu UPLOAD -painiketta painamalla.

4 TULOKSET JA POHDINTA

Lopputuloksena autopilotti toimii niin kuin sen suunnittelinkin. Testikäytössä kytkemällä laitteeseen sähköt päälle magneettikytkin lukittuu, ohjelma ottaa senhetkisen suunnan muistiin sekä alkaa verrata kompassin antamaa arvoa muistissa olevaan arvoon. Kun kompassipiiriä kääntää vasemmalle tai oikealle, autopilotti kääntää rattia automaattisesti päinvastaiseen suuntaan suorittaakseen korjausliikkeen. Korjausliikkeen suuruus riippuu suunnan poikkeaman suuruudesta. Kompassin kääntyessä takaisin alkuperäiseen suuntaan laite suoristaa ratin. Tulevaisuudessa autopilotti (KUVIO 16) on tarkoitus asentaa veneeseen paikoilleen.

KUVIO 16. Autopilotti

Työn suunnittelu ja tekeminen oli mielestäni sopivan haastavaa. Opin työssä paljon Arduinon käyttömahdollisuuksista, sillä aikaisempaa kokemusta alustasta ei ollut. Työ oli monipuolinen sisältäen elektroniikkaa, ohjelmointia sekä tietenkin mekaanista rakentelua.

LÄHTEET

Arduino. 2013. Arduino UNO. Www-dokumentti. Saatavissa: http://arduino.cc/en/Main/ArduinoBoardUno. Luettu 12.8.2012

Arduino. 2013. FAQ. Www-dokumentti. Saatavissa: http://arduino.cc/en/Main/FAQ. Luettu 16.2.2013

Arduino. 2013. Reading a Potentiometer (analog input). Www-Dokumentti. Saatavissa: http://www.arduino.cc/en/Tutorial/Potentiometer. Luettu 17.2.2013

D&D Motor Systems, Inc. 2007. Simple 2 pole DC Motor. Www-dokumentti. Saatavissa: http://www.ddmotorsystems.com/CurrentRange.php. Luettu 23.9.2012

Jessa, T. 2010. How Does a Compass Work. Www-dokumentti. Saatavissa: http://www.universetoday.com/77072/how-does-a-compass-work/. Luettu 2.2.2013

Kompo2010. 2010. Askelmoottori. Www-dokumentti. Saatavissa: http://kompo2010.wikispaces.com/Askelmoottori. Luettu 22.9.2012

Kompo2010. 2010. Pienjännitesähkömoottori. Www-dokumentti. Saatavissa: http://kompo2010.wikispaces.com/Pienjännitesähkömoottori. Luettu 22.9.2012

Lahart, J. 2009. Taking an Open-Source Approach to Hardware. The Wall Street Journal. Www-dokumentti. Saatavissa: http://online.wsj.com/article/SB10001424052748703499404574559960271468066.html. Luettu 1.3.2013

Robot-Electronics. 2012. CMPS10. Www-dokumentti. Saatavissa: http://www.robot-electronics.co.uk/acatalog/Compass.html. Luettu 22.9.2012

Ohjelmakoodi

```
////////////////////////////////////////////////////////////////
//AUTOPILOTTI
//VILLE PENTIKÄINEN
//2013
////////////////////////////////////////////////////////////////

#include <Wire.h>
#define ADDRESS 0x60                          //Defines address of CMPS10
#define LEFT_PIN 12                           //Define PIN names
#define RIGHT_PIN 10
#define WHEEL_PIN 1

int wantedDir=NULL, currentDir;
int wheelpos=NULL;


void setup() {

// Connects I2C
  Wire.begin();
  Serial.begin(115200);
  pinMode(LEFT_PIN, OUTPUT);
  pinMode(RIGHT_PIN, OUTPUT);

}
```

```
void loop(){

   wheelpos = analogRead(WHEEL_PIN);

   byte highByte, lowByte;    // highByte and lowByte store high and //
                              low bytes of the bearing and fine
                              // stores decimal place of bearing
   int bearing;               // Stores full bearing

//starts communication with CMPS10
    Wire.beginTransmission(ADDRESS);

//Sends the register we wish to start reading from
   Wire.write(2);
   Wire.endTransmission();

// Request 4 bytes from CMPS10
   Wire.requestFrom(ADDRESS, 4);

// Wait for bytes to become available
    while(Wire.available() < 4);
       highByte = Wire.read();
       lowByte = Wire.read();

// Calculate full bearing
   currentDir = ((highByte<<8)+lowByte)/10;
```

```
   /* For testing purposes
      Serial.print("ratti: ");
      Serial.print(wheelpos);
      Serial.print(" suunta: ");
      Serial.print(currentDir);
      Serial.print("\n");
   */

   if(wantedDir == NULL){
      wantedDir = currentDir;
    }

   doSmartTurn(currentDir, wantedDir);

}
```

```
void doSmartTurn(unsigned int curDirection, unsigned int newDirection)
{

// force directions between 0..359
  curDirection %= 360;
  newDirection %= 360;

  int delta = newDirection - curDirection;

  if (abs(delta) <= 10 ){
    if (wheelpos < 490) {
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }

    if (wheelpos > 510) {
     digitalWrite(LEFT_PIN, HIGH);
     digitalWrite(RIGHT_PIN, LOW);
     delay(10);
    }
    if (wheelpos >= 490 && wheelpos <= 510) {
     digitalWrite(LEFT_PIN, LOW); digitalWrite(RIGHT_PIN, LOW);
    delay(10);
    }

// don't do small angles.
  return;
}
```

```
  if (delta > 180) turnLeft(360 - delta);
  else if (delta > 0) turnRight(delta);
  else if (delta > -180) turnLeft(-delta);
  else turnRight(360+delta);
}

void turnLeft(unsigned int Delta){


//small turn left
  if(Delta <= 20){
    if(wheelpos > 350){
      digitalWrite(LEFT_PIN, HIGH);
      digitalWrite(RIGHT_PIN, LOW);
      delay(10);
    }
    else if (wheelpos < 330){
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }
    else{
    digitalWrite(RIGHT_PIN, LOW);
    digitalWrite(LEFT_PIN, LOW);
    delay(10);
    }
  }
```

```
//big turn left
    if(Delta > 20){
    if(wheelpos > 220){
      digitalWrite(LEFT_PIN, HIGH);
      digitalWrite(RIGHT_PIN, LOW);
      delay(10);
    }
    else if (wheelpos < 200){
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }
    else{
    digitalWrite(LEFT_PIN, LOW);
    digitalWrite(RIGHT_PIN, LOW);
    delay(10);
  }
 }
}
```

```
void turnRight(unsigned int Delta){

//small turn right
  if(Delta <= 20){
    if(wheelpos > 670){
      digitalWrite(LEFT_PIN, HIGH);
      digitalWrite(RIGHT_PIN, LOW);
      delay(10);
    }

    else if (wheelpos < 650){
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }
    else{
    digitalWrite(RIGHT_PIN, LOW);
    digitalWrite(LEFT_PIN, LOW);
    delay(10);
    }
  }
```

```
//big turn right
    if(Delta > 20){
    if(wheelpos > 820){
      digitalWrite(LEFT_PIN, HIGH);
      digitalWrite(RIGHT_PIN, LOW);
      delay(10);
    }
    else if (wheelpos < 800){
      digitalWrite(RIGHT_PIN, HIGH);
      digitalWrite(LEFT_PIN, LOW);
      delay(10);
    }
    else{
    digitalWrite(LEFT_PIN, LOW);
    digitalWrite(RIGHT_PIN, LOW);
    delay(10);
  }
}
}
```