Tampereen ammattikorkeakoulu

Tuulen suunta ja nopeus

Iiro Hämäläinen,

Julian Levä ja

Daniel Schmidt

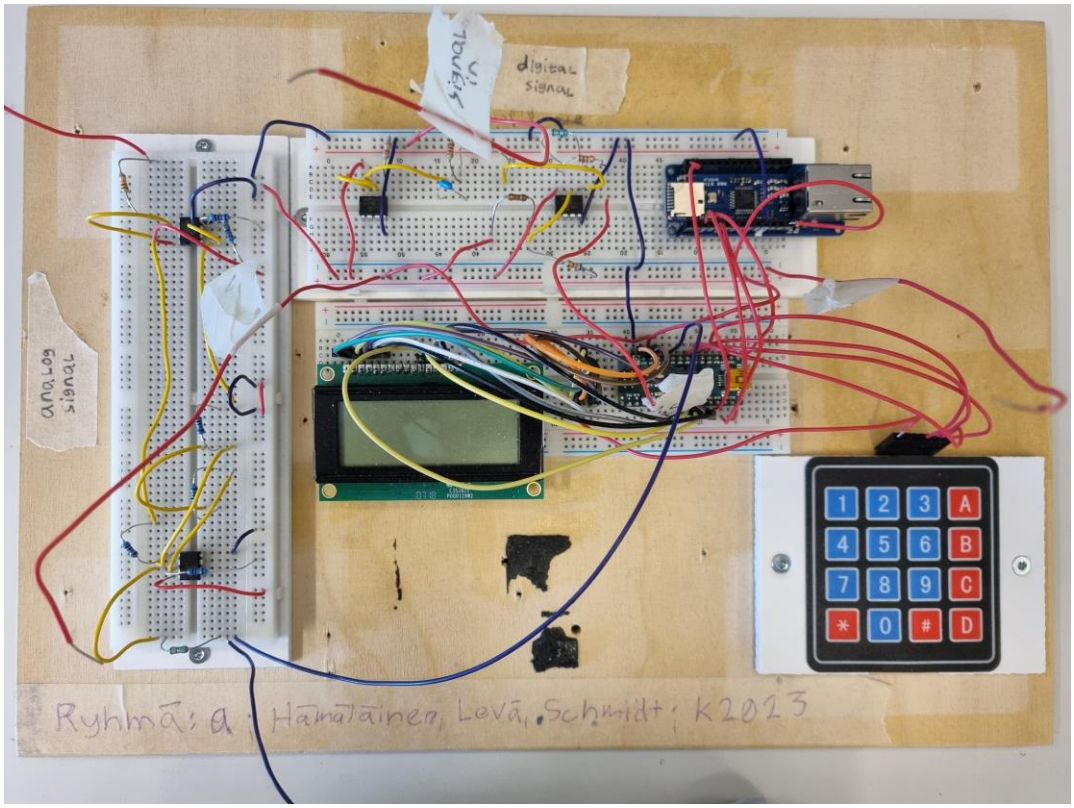
Tekninen dokumentti
Toukokuu 2023

Tietotekniikan tutkinto-ohjelma

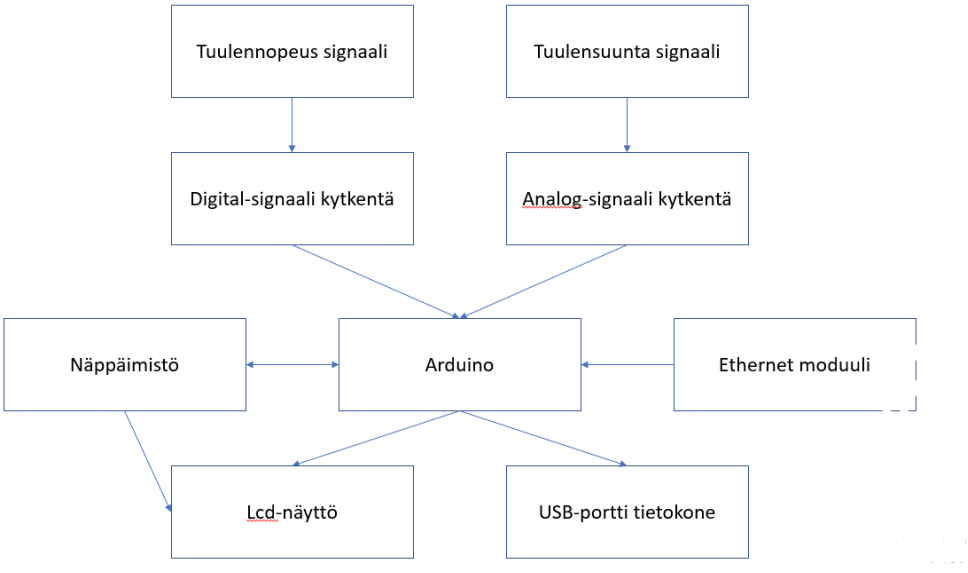
SISÄLLYS

1 Hardware

1.1 Kytkentä



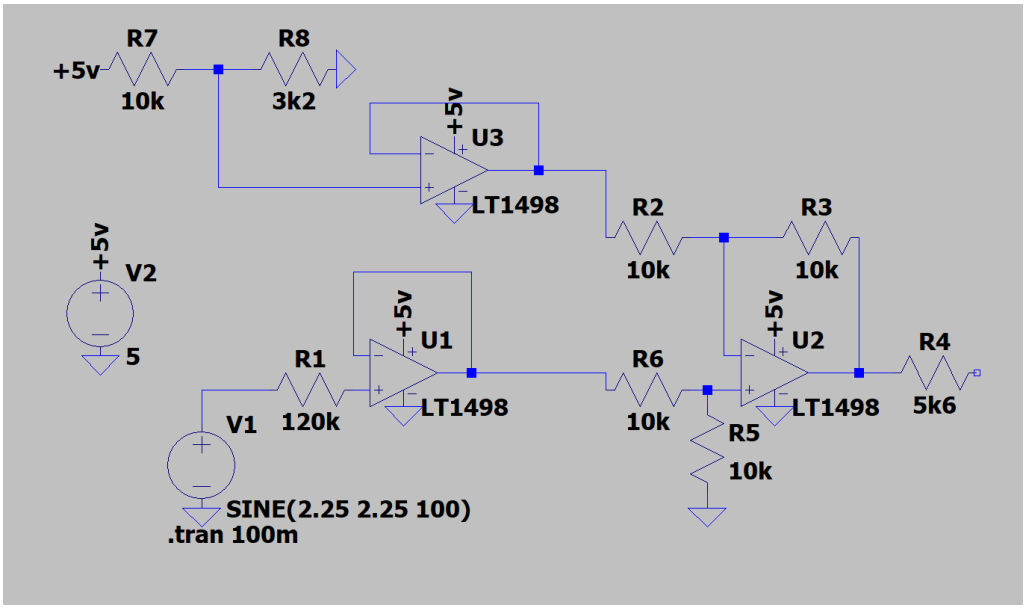
Kuva 1: Valmis kytkentä levyllä.



Kuva 2: Kaavio kytkennästä.

Sääasemalta tulee signaali levylle, jossa se kiinnitetään digital- tai analog-signaali kytkentään riippuen signaalista. Signaalia muokataan kytkennöissä halutulla tavalla, josta se viedään arduinolle. Arduino tulostaa sen näytölle näppäimistön kautta valitulla tavalla. Ethernetmoduuli mahdollistaa datan lähettämistä eteenpäin verkon avulla.

1.2 Simulointi



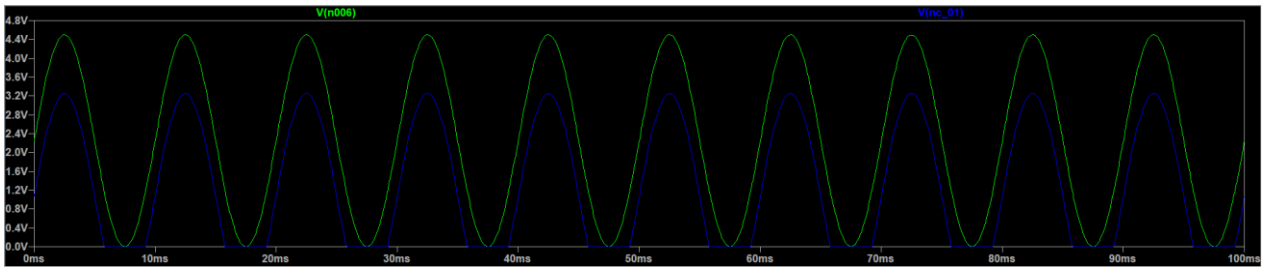
Kuva 3: Analog -signaalin muokkaus kytkentä ltspicellä.

Vastukset R7 ja R8 muodostavat jännitejakaajan, jota tarvitsemme, että saamme referenssi jännitteen 1,2V U2 operaatiovahvistimelle. U3 operaatiovahvistin toimii erottamaa jännitejakaajan ja U2 operaatiovahvistimen, silloin ei tarvitse ottaa huomioon vastuksia R2 ja R3, kun lasketaan jännitejakoa.

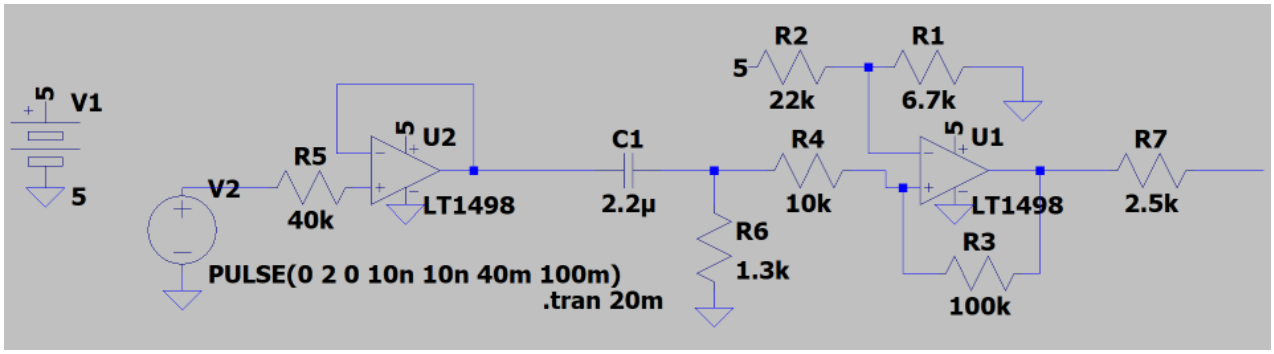
Vastukset R2, R3, R6, R5 ja operaatiovahvistin U2 muodostavat differentiaalivahvistimen, joka vahvistaa kahden tulosignaalin välisen erotuksen.

Vastus R1 toimii oikosulkusuojana eli estää virran kasvamista liian suureksi, jos oikosulku tapahtuu. Sen koko määräytynyt vaatimusten mukaan.
Operaatiovahvistin U1 toimii puskurina eli käytännössä estää kytkentää vaurioittavan virran pääsemistä kytkentään.

Vastus R4 toimii arduinon oikosulkusuojana eli estää oikosulkutilanteessa liian suuren virran pääsemistä arduinoon. Sen koko määräytynyt vaatimusten mukaan.



Kuva 4: Analog-signaalin simulointi. Vihreä on signaali sisään ja sininen on signaali ulos

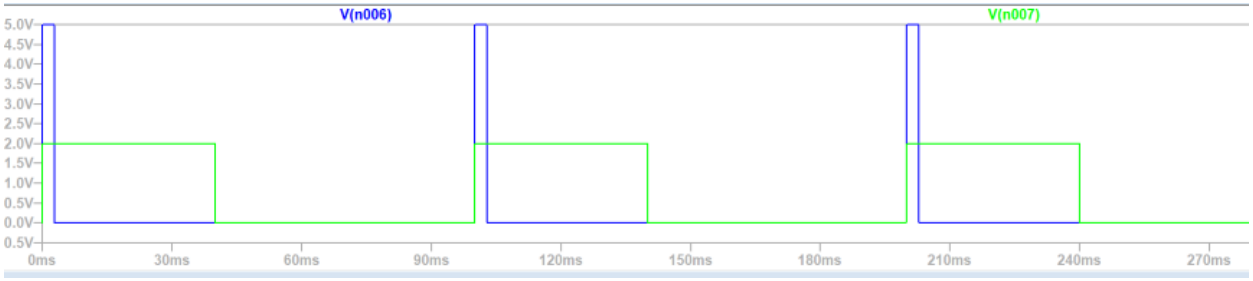


Kuva 5: Digitaali- signaalin muokkauskytkentä lt-spicellä.

Vastus R5 toimii oikosulkusuojana eli estää virran kasvamista liian suureksi, jos oikosulku tapahtuu. Sen koko määräytynyt vaatimusten mukaan.
Operaatiovahvistin U2 toimii puskurina eli käytännössä estää kytkentää vaurioittavan virran pääsemistä kytkentään.
Kondensaattori C1 ja vastus R6 toimivat signaalin muokkaajina. Niiden suhteella määritetään, kuinka kauan signaali on ylätilassa vaatimusten mukaisesti.
Operaatiovahvistin U1, vastukset R1, R2, R3 ja R4 muodostavat komparaattori kytkennän, jonka avulla määritetään, milloin kytkennän tila on 5V ja milloin 0V.
Vastukset R1 ja R2 muodostavat vertailutason, johon sisään tulevaa signaali jännitettä verrataan, josta komparaattori pystyy määrittämään, onko tila 5V vai 0V. Vastukset R3 ja R4 muodostavat hystereesin, joka poistaa kohinan kytkennästä. Siitä saadaan kynnysjännite, joka on myös ylitettävä ennen kuin tila voi

muuttua.

Vastus R7 toimii arduinon oikosulkusuojana eli estää oikosulkutilanteessa liian suuren virran pääsemistä arduinoon. Sen koko määräytynyt vaatimusten mukaan.



Kuva 6: Simuloitu signaali. Vihreä on signaali sisään ja sininen on signaali ulos.

1.3 Verifiointi

Suunnitelma

Rakennettu kytkentä verifioidaan, jotta tiedetään että se toimii niin kuin projektin vaatimuksien mukaan. Tämä suoritetaan mittaamalla kaikki mahdolliset vaatimukset.

Mitattavat vaatimukset

Digitaali-signaalin verifiointi

Vaatimus:
Loogista tilaa ”0” vastaava jännite pitää olla alle 0.5V
Kriteeri: alle 0.5V
Mittaus:
Mitataan lähtöjännite, joka vastaa loogista tilaa ”0”
Mitattu arvo: alle 0.5V
Mittaus tulos kertoo meille että loogista tilaa ”0” oleva jännite arvo on alle 0.5V

Vaatimus:
Loogista tilaa ”1” vastaava jännite pitää olla yli 4.0V

Kriteeri: yli 4.0V

Mittaus:

Mitataan lähtöjännite, joka vastaa loogista tilaa ”1”.

Mitattu arvo: noin 5V

Mittaus tulos kertoo meille että loogista tilaa ”1” oleva jännite arvo on yli 4.0V

Vaatimus:

Ei signaalin värähtelyä.

Kriteeri: Signaali ei värähtele.

Mittaus:

Katsotaan, että signaali ei värähtele oskilloskoopilla.

Mitattu arvo: Ei värähtele.

Signaalissa ei ollut huomattavaa värähtelyä.

Vaatimus:

Signaalin nousu ja lasku aika on < 1.0ms

Kriteeri: < 1.0ms

Mittaus:

Mitataan oskilloskoopilla signaalin nousu ja lasku aika

Mitattu arvo: 1 µs

Signaalin nousu ja lasku aika on noin 1 µs.

Vaatimus:

Kytkennän kuormitus signaali boxille pitää olla <50µA

Kriteeri: <50µA

Mittaus:

Virta mittaus.

Mitattu arvo: <10µA

Virta mittari ei voinut näyttää paljon virtaa kulki, koska virtaa kulkee niin vähän.

Vaatimus:

Arduino sisääntulo ei saa aiheuttaa yli 2mA oikosulkua

Kriteeri: <2mA

Mittaus:

Virta mittaus.

Mitattu arvo: 1.8mA

Kytkentä oikosulussa aiheuttaa 1.8mA virran.

Vaatimus:

Kytkennän sisään tulosignaali ei saa aiheuttaa yli 0.1 mA oikosulku virtaa.

Kriteeri: <0.1mA

Mittaus:

Virta mittaus.

Mitattu arvo: 50µA

Kytkentä oikosulussa aiheuttaa 50µA virran.

Vaatimus:

Digitaali signaali saa olla ”1” tilassa vain 2ms-6ms

Kriteeri: 2ms-6ms

Mittaus:

Mitataan ja katsotaan oskilloskoopilla, että signaali on ”1” tilassa 2ms-6ms

Mitattu arvo: 1Hz ylhäällä 3.79ms ja 30Hz ylhäällä 3.77ms

Signaali on ylätilassa 1Hz 3.79ms ja 30Hz 3.77ms. Signaali on ylätilassa realistisissa tilanteissa ylhäällä 2ms-6ms.

Analog-signaalin verifiointi

Vaatimus:

Analog signaali sisääntulo arduinolle pitää olla 0V < 5V

Kriteeri: 0V<5V

Mittaus:

Mitataan oskilloskoopilla sisään tulo arduinolle on yli 0V alle 5 V

Mitattu arvo: Sisään tulo arduinolle on yli 0V ja alle 5V

Sisään tulo signaali arduinolle on yli 0V ja alle 5V.

Vaatimus:

Kytkennän kuormitus signaali boxille pitää olla <50µA

Kriteeri: <50µA

Mittaus:

Virta mittaus.

Mitattu arvo: <10µA

Virta mittari ei voinut näyttää paljon virtaa kulki, koska virtaa kulkee niin vähän.

Vaatimus:

Arduino sisääntulo ei saa aiheuttaa yli 2mA oikosulku

Kriteeri: <2mA

Mittaus:

Virta mittaus.

Mitattu arvo: 0.9mA

Arduinon sisääntulo aiheuttaa 0.9mA virran oikosulussa.

Vaatimus:

Kytkennän sisään tulosignaali ei saa aiheuttaa yli 0.1mA oikosulku virtaa

Kriteeri: <0.1mA

Mittaus:

Virta mittaus.

Mitattu arvo: 42µA

Kytkennän sisään tulosignaali aiheuttaa 42µA oikosulku virran.

Vaatimus:

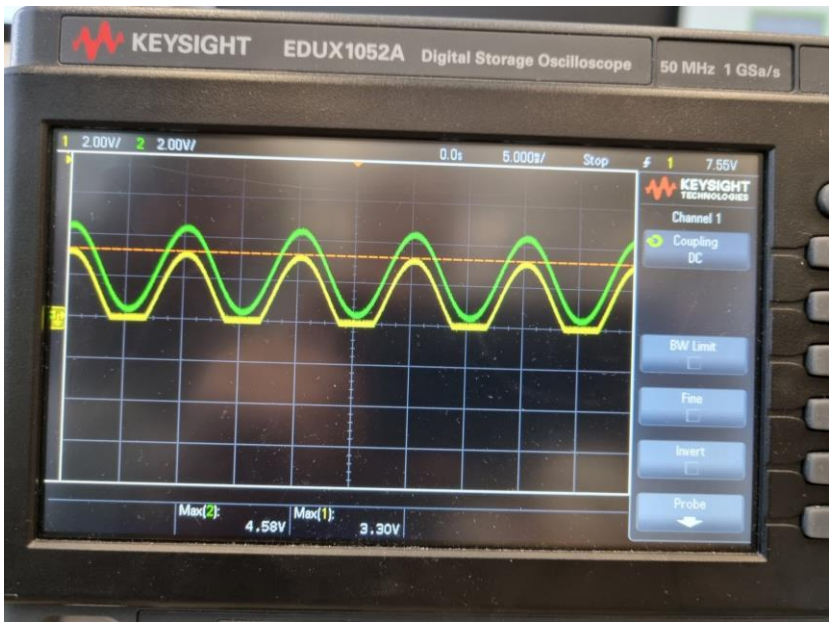
Analog signaali ulostulo 0V-3.33V

Mittaus:

Mitataan oskilloskoopilla, että analog signaali ulostulo 0V-3.33V

Mitattu arvo: 0V-3.3V

Analog signaali ulostulo on 0V-noin 3.3V välissä.



Kuva 7: Kuvaa verifiointi prosessista.

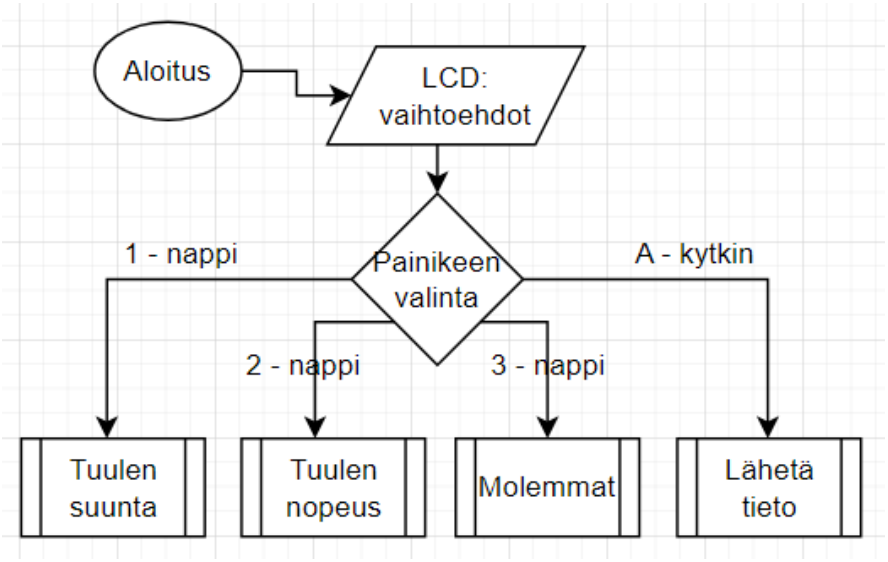
2 Software

2.1 Koodi

Ohjelmakoodin voi toiminnallisuuksiensa mukaan jakaa karkeasti seuraaviin ryhmiin:

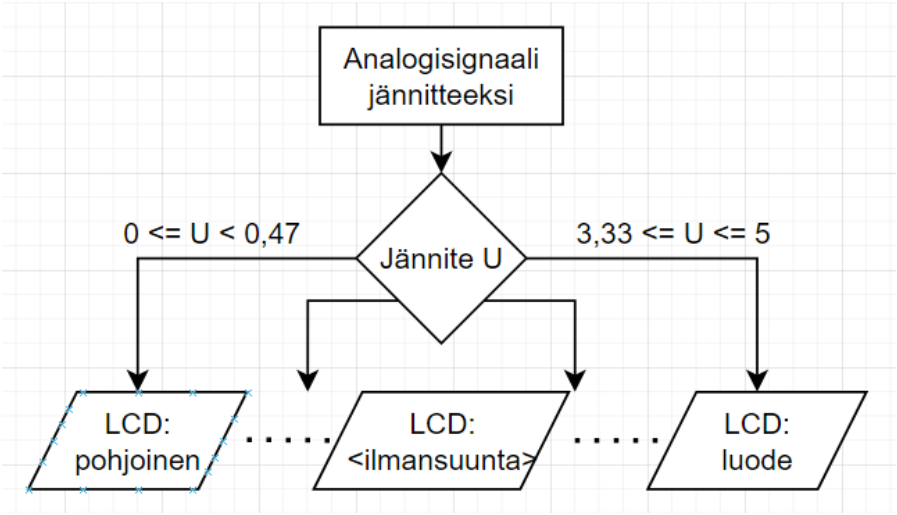
- Alustus: ulkoiset kirjastot liitetään osaksi ohjelmaa ja yhteiset muuttujat alustetaan
- Käyttöönotto: tilannetiedot päivitetään näyttöpäätteille, Arduinon pinnit aktivoidaan, Ethernet-moduulille haetaan IP-osoite, jonka jälkeen se yhdistetään MQTT-palvelimeen
- Käyttö: tulkitaan käyttäjän syötettä painikkeilta, jonka perusteella ohjelma tulostaa LCD-näytölle tuulen suunnan ja nopeuden taikka molemmat. Tiedon palvelimelle lähettämistä varten on erillinen painike.

Kuvassa on havainnollistettu ohjelman toimintaa käyttäjän näkökulmasta. Aluksi käyttäjälle tulostetaan ohjeet valinnoista LCD-näytölle. Painikkeet 1-3 toimivat vain painettuina, mutta painike A toimii binäärisen kytkimen tavoin. Painikkeiden tila tulkitaan 1,5 s välein.



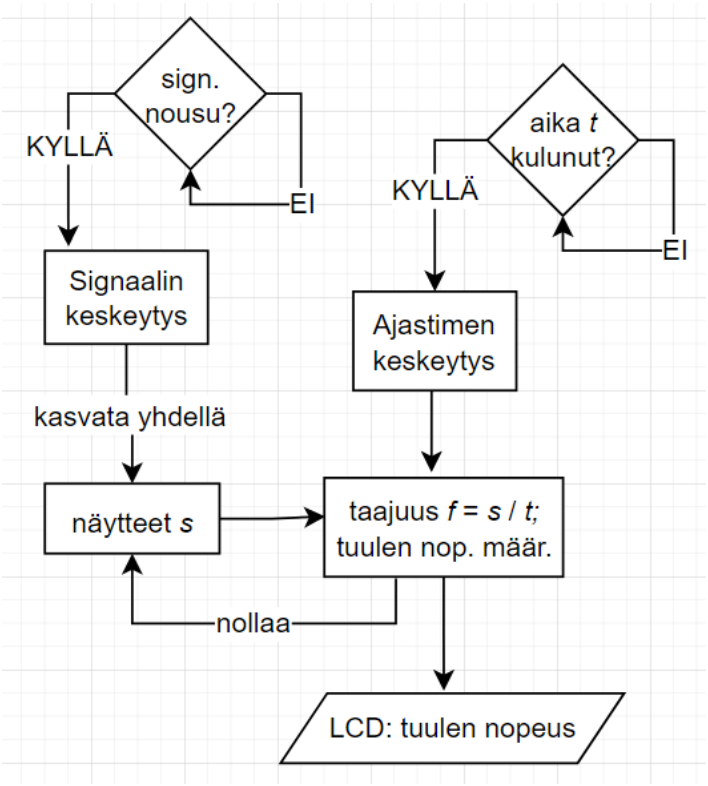
Kuva 8: Vuokaavioesitys: ohjelman toiminta käyttäjän näkökulmasta

Tuulen suunta esitetään LCD-näytöllä ilmansuuntana ja asteina. Arduinon analogisignaalin arvo suhteutetaan oletettuun viiden voltin toimintajännitealueeseen. Laskennallisen jännitearvon perusteella määritetään suunnan astelukema asteen tarkkuudella, muunnoskaavan avulla. Tulostettavaksi valitaan kulloistakin jännitettä vastaavan jännitealueen mukainen ilmansuunta, yhdessä astelukeman kanssa. Kuvassa on tarkemmin esitetty jännitteen ääriarvoja vastaavat esimerkkitilanteet.



Kuva 9: Signaalin arvon perusteella valitaan ilmansuunta ja lasketaan suunnan astelukema. Arvot tulostetaan näytölle

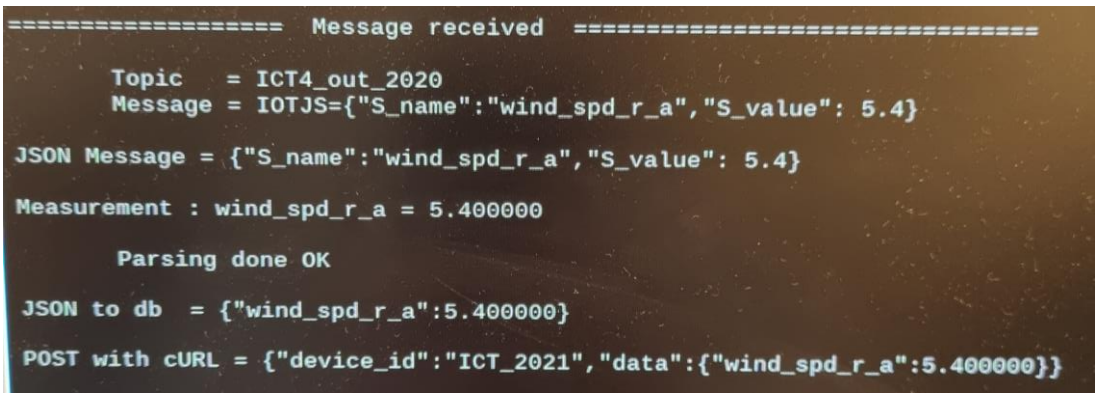
Tuulen nopeus tulostetaan LCD-näytölle metreinä sekunnissa. Tätä ennen se on laskettava signaalitaajuudesta muunnoskaavan avulla. Taajuuden määrittämistä varten pääprosessia on keskeyttämässä anturin noususignaali ja ajastimen ”aika kulunut” -signaali. Taajuus on kerättyjen näytteiden lukumäärän ja keräämiseen käytetyn ajan osamäärä. Kuten kuvasta ilmenee, kerätään näytteitä *s* ensin ajan *t* (3 s) verran, ja tämän jälkeen näytteiden lukumäärä nollataan seuraavaa sykliä varten.



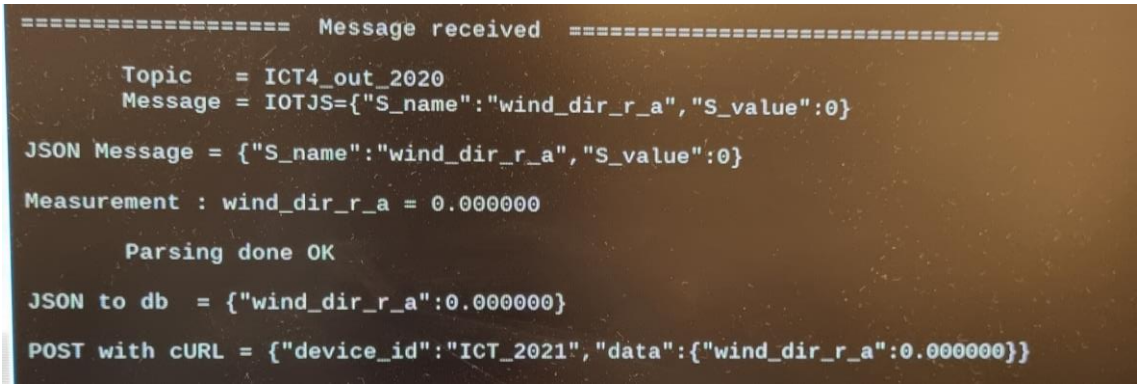
Kuva 10: Taajuus määritetään näytteiden lukumäärän ja käytetyn ajan avulla, lopuksi tuulen nopeus tulostetaan LCD-näytölle

2.2 Verifiointi

Kuvista 11 ja 12 ilmenee, että viestit lähetettiin palvelimelle JSON-muotoisessa tietorakenteessa. Viestien purku palvelimella hyväksyttiin.



Kuva 11: "Tuulen nopeus" -viestin lähetys onnistunut.



Kuva 12: "Tuulen suunta" -viestin lähetys onnistunut.

LIITTEET

LIITE 1. Kytkennän verifiointipöytäkirja

			Ryhmä: A Nimet: Iiro, Daniel, Julian				
Test Case	pvm	Testaaja	Vaatimus joka todennetaan	Mittaus	Mitattu arvo	Hyväksyntä raja	OK/NOK
			Digitaali signaalin				
1.	16.2.2023	A	Loogista tilaa "0" vastaava jännite pitää olla alle 0.5V	Mitataan lähtöjännite, joka vastaa loogista tilaa "0"	<0.5V	<0.5V	
2.	16.2.2023	A	Loogista tilaa "1" vastaava jännite pitää olla yli 4.0V	Mitataan lähtöjännite, joka vastaa loogista tilaa "1"	5V	>4.0V	
3.	16.2.2023	A	Ei signaalin värähtelyä	Katsotaan, että signaali ei värähtele		Signaali ei värähtele	
4.	16.2.2023	A	Signaalin nousu ja lasku aika on < 1.0ms	Mitataan oskilloskoopilla signaalin nousu ja lasku aika	1us	<1.0ms	
5.	16.2.2023	A	Analog signaali sisääntulo arduinolle pitää olla 0v < 5v	mitataan oskilloskoopilla sisään tulo arduinolle		0v<5v	
6.	16.2.2023	A	Kytkennän kuormitus signaali boxille pitää olla <50uA	virta mittaus	<10uA	<50uA	
7.	16.2.2023	A	Arduino sisääntulo ei saa aiheuttaa 2mA oikosulku	virta mittaus	1.8mA	<2mA	
8.	16.2.2023	A	Kytkennän sisään tulosignaali ei saa aiheuttaa 0.1mA oikosulku virtaa .	virta mittaus	50uA	<0.1mA	
9.	16.2.2023	A	Kytkennän kuormitus signaali boxille pitää olla <50uA	virta mittaus	<10uA	<50uA	
10.	16.2.2023	A	Arduino sisääntulo ei saa aiheuttaa 2mA oikosulku	virta mittaus	0.9 mA	<2mA	
11.	16.2.2023	A	Kytkennän sisään tulosignaali ei saa aiheuttaa 0.1mA oikosulku virtaa .	virta mittaus	42uA	<0.1mA	
12.	16.2.2023	A	Digitaali signaali saa olla 1 tilassa vain 2ms-6ms	mitataan ja katsotaan oskilloskoopilla, että signaali on 1 tilassa 2ms-6ms	1Hz=3.79ms 30Hz = 3.77ms	2ms>6ms	
13.	16.2.2023	A	Analog signaali ulostulo 0V-3.33V	mitataan oskilloskoopilla, että analog signaali ulostulo 0V-	noin 0V-3.3V		

LIITE 2. Arduinokoodi

```
//    Alyk  jatko course 2021       30.3.2020 KN
//                                  2021 SV, EK
//
#include <LiquidCrystal.h>
#include <String.h>
#include <Ethernet.h>                                  // incluide Ether-
net library W5100
#include <PubSubClient.h>                              // include MQTT li-
brary
#include <TimerOne.h>                                  // include timer
library

const int rs = 7, en = 8, d4 = 6, d5 = 5, d6 = 4, d7 = 3;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
float value;
float voltage;
String wind_dir;
int degrees;
int f;
volatile int number = 0;
int interval = 3000;
int samples;
float wind_speed;
int state = 0;
int state1 = 0;
int state2 = 0;
int state3 = 0;
bool is_state3 = false;

EthernetClient ethClient;                               // Ethernet ob-
ject var

static uint8_t mymac[6] = { 0x44,0x76,0x58,0x10,0x00,0x62 };

//  MQTT settings

unsigned int Port = 1883;                            //  MQTT port number
byte server[] = { 10,6,0,21 };                      // TAMK IP 10.6.0.21

char* deviceId     = "ryhma_a_dev";                    // * set your de-
vice id (will be the MQTT client username) *yksilöllinen*
char* clientId     = "ryhma_a";                        // * set a random
String (max 23 chars, will be the MQTT client id) *yksilöllinen*
char* deviceSecret = "tamk";                          // * set your device
secret (will be the MQTT client password) *kaikille yhteinen*

//  MQTT Server settings
```

```
void callback(char* topic, byte* payload, unsigned int length); // sub-
scription callback for received MQTT messages

PubSubClient client(server, Port, callback, ethClient);   // mqtt client

//  MQTT topic names

 #define inTopic    "ICT1B_in_2020"                    // * MQTT channel
where data are received
 #define outTopic   "ICT4_out_2020"                   // * MQTT channel
where data is send

//  SETUP section

void get_wind_dir();

void setup() {
    Serial.begin(9600);                                 // Serial monitor
baudrate  = 9600
    Serial.println("Start 19.3.2021");                  // print to se-
rial monitor
    pinMode(A5, INPUT);
    pinMode(14, INPUT_PULLUP);
    pinMode(15, INPUT_PULLUP);
    pinMode(16, INPUT_PULLUP);
    pinMode(17, INPUT_PULLUP);
    lcd.begin(20, 4);
    delay(500);
pinMode(2, INPUT);
attachInterrupt(digitalPinToInterrupt(2), pin_ISR, RISING);
Timer1.initialize(3000000);
Timer1.attachInterrupt (time_interrupt);
    fetch_IP();                                         // initialize
Ethernet connection
    Connect_MQTT_server();                              // connect to
MQTT server
}

void loop() {
  lcd.clear();
    Serial.println("Are we connected?");
    lcd.setCursor(0, 0);
    lcd.print("1. Wind_dir");
    lcd.setCursor(0, 1);
    lcd.print("2. Wind_spd");
    lcd.setCursor(0, 2);
    lcd.print("3. Both");
    lcd.setCursor(0,3);
    lcd.print("A. Send data");
    state = digitalRead(14);
    state1 = digitalRead(15);
```

```
    state2 = digitalRead(16);
    state3 = digitalRead(17);
if(state==0){
  lcd.clear();
get_wind_dir();
}
if(state1==0){
  lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Wind speed: ");
lcd.setCursor(0, 1);
lcd.print(wind_speed);
lcd.print(" m/s");
}
if(state2==0){
  lcd.clear();
  get_wind_dir();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(wind_speed);
  lcd.print(" m/s");
lcd.setCursor(0,1);
lcd.print(wind_dir);
lcd.print(" ");
lcd.print(degrees);
lcd.print(" degrees");
}
if(state3==0){
if(is_state3){
is_state3 = false;
}else{
  is_state3=true;
}
}
if(is_state3){
    lcd.clear();
  get_wind_dir();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(wind_speed);
  lcd.print(" m/s");
lcd.setCursor(0,1);
lcd.print(wind_dir);
lcd.print(" ");
lcd.print(degrees);
lcd.print(" degrees");
send_MQTT_message();
delay(1500);
}
send_MQTT_message();
delay(1500);
```

```
}

void get_wind_dir() {
  // put your main code here, to run repeatedly:
value = analogRead(A5);
voltage=(5*value/1023);
degrees = 94.57*voltage+0.1586;
lcd.setCursor(0, 0);
lcd.print("Voltage: ");
lcd.print(voltage);
lcd.setCursor(0, 1);
lcd.print("Wind dir: ");
lcd.setCursor(0, 2);
if(voltage>=0 && voltage<0.47 || voltage ==5){
  wind_dir = "N";
  lcd.print("N ");
  lcd.print(degrees);
  lcd.print(" degrees");
}
else if(voltage>=0.47 && voltage<0.95){
  wind_dir = "NE";
   lcd.print("NE ");
  lcd.print(degrees);
  lcd.print(" degrees");
}
else if(voltage>=0.95 && voltage<1.43){
  wind_dir = "E";
  lcd.print("E ");
  lcd.print(degrees);
  lcd.print(" degrees");
}
else if(voltage>=1.43 && voltage<1.90){
  wind_dir = "SE";
  lcd.print("SE ");
  lcd.print(degrees);
  lcd.print(" degrees");
}
else if(voltage>=1.90 && voltage<2.38){
  wind_dir = "S";
  lcd.print("S ");
  lcd.print(degrees);
  lcd.print(" degrees");
}
else if(voltage>=2.38 && voltage<2.85){
  wind_dir = "SW";
  lcd.print("SW ");
  lcd.print(degrees);
  lcd.print(" degrees");
}
else if(voltage>=2.85 && voltage<3.33){
  wind_dir = "W";
```

```
  lcd.print("W ");
  lcd.print(degrees);
  lcd.print(" degrees");
}
else if(voltage>=3.33 && voltage<5){
  wind_dir = "NW";
  lcd.print("NW ");
  lcd.print(degrees);
  lcd.print(" degrees");
}
}

void pin_ISR(){
 samples++;
}
void time_interrupt(){
  number++;
  if(number<=interval){
     f = samples/3;
     wind_speed = -0.24 + f * 0.699;
     samples=0;
  }
}

//  GET IP number from DHCP server

void fetch_IP(void){
  byte rev=1;
  rev=Ethernet.begin(mymac);                    // get IP number
  Serial.print( F("\nW5100 Revision ") );

  if (rev == 0){
    Serial.println( F( "Failed to access Ethernet controller" ) );
  }

  Serial.println( F( "Setting up DHCP" ));
  Serial.print("Connected with IP: ");
  Serial.println(Ethernet.localIP());
  delay(1500);
  lcd.clear();
}

//  MQTT Routines

void send_MQTT_message(){                       // Send MQTT message
  char bufa[50];                               //  Print message to serial
monitor
  char bufb[50];
  char bufc[4];
  if (client.connected()){
if(state==0){
```

```
sprintf(bufa,"IOTJS={\"S_name\":\"wind_dir_r_a\",\"S_value\":%i}",de-
grees);              // create message with header and data
client.publish(outTopic,bufa);
}
if(state1==0){
  dtostrf(wind_speed, 4, 1, bufc);
  sprintf(bufb,"IOTJS={\"S_name\":\"wind_spd_r_a\",\"S_value\":%s}",bufc)
;
  client.publish(outTopic,bufb);                        // send message
to MQTT server                     // create message with header and data
}
if(state2==0){
  sprintf(bufa,"IOTJS={\"S_name\":\"wind_dir_r_a\",\"S_value\":%i}",de-
grees);              // create message with header and data
    dtostrf(wind_speed, 4, 1, bufc);
    sprintf(bufb,"IOTJS={\"S_name\":\"wind_spd_r_a\",\"S_value\":%s}",buf
c);              // create message with header and data
    Serial.println( bufa );
    client.publish(outTopic,bufa);                        // send message
to MQTT server
    client.publish(outTopic,bufb);                        // send message
to MQTT server
}
if(is_state3){
  sprintf(bufa,"IOTJS={\"S_name\":\"wind_dir_r_a\",\"S_value\":%i}",de-
grees);              // create message with header and data
    dtostrf(wind_speed, 4, 1, bufc);
    sprintf(bufb,"IOTJS={\"S_name\":\"wind_spd_r_a\",\"S_value\":%s}",buf
c);              // create message with header and data
    Serial.println( bufa );
    client.publish(outTopic,bufa);                        // send message
to MQTT server
    client.publish(outTopic,bufb);                        // send message
to MQTT server
}
  }
  else{                                                          //   Re
connect if connection is lost
    delay(500);
    Serial.println("No, re-connecting" );
    client.connect(clientId, deviceId, deviceSecret);
    delay(1000);                                              // wait for
reconnecting
  }
}

//  MQTT server connection

void Connect_MQTT_server(){
  Serial.println(" Connecting to MQTT" );
```

```
  Serial.print(server[0]); Serial.print(".");     // Print MQTT server IP
number to Serial monitor
  Serial.print(server[1]); Serial.print(".");
  Serial.print(server[2]); Serial.print(".");
  Serial.println(server[3]);
  delay(500);

  if (!client.connected()){                                  // check if
allready connected
    if (client.connect(clientId, deviceId, deviceSecret)){ // connection
to MQTT server
      Serial.println(" Connected OK " );
      client.subscribe(inTopic);                        // subscript to
in topic
    }
    else{
       Serial.println(client.state());
    }
  }
}

//  Receive incoming MQTT message

void callback(char* topic, byte* payload, unsigned int length){
  char* receiv_String;                                 // copy the payload
content into a char*
  receiv_String = (char*) malloc(length + 1);
  memcpy(receiv_String, payload, length);           // copy received mes-
sage to receiv_String
  receiv_String[length] = '\0';
  Serial.println( receiv_String );
  free(receiv_String);
}
```