ICT projekti – Ilmanpaine

(signaali 32)

Tekninen dokumentaatio

Olli Laurila

Ari Savolainen

Valdemar Tuominen

SISÄLLYS

[1 JOHDANTO 4](#_Toc405990433)

[2 TEKNISEN DOKUMENTIN TAVOITTEET, RAJAUS JA TULOKSET 5](#_Toc405990434)

[3 OHJELMIEN ARKKITEHTUURIN KUVAUS 6](#_Toc405990435)

[3.1. LCD-näytön testiohjelma 6](#_Toc405990436)

[3.2. Arduino ohjelma 6](#_Toc405990436)

3[.3. MQTT C++ ohjelma 6-7](#_Toc405990440)

3[.4. Raspberry ohjelma 7-9](#_Toc405990440)

[4 FUNKTIOT 10-12](#_Toc405990445)

[5 RAJAPINNAT 13](#_Toc405990446)

[6 KAAVIOT, PIIRRUSTUKSET JA KUVAT 14](#_Toc405990447)-24

[7 LIITTEET 25](#_Toc405990447)

# JOHDANTO

Tämän dokumentaation tekemiseen johti ICT-kurssi, jonka opetuksesta osa toteutetaan projektiopetuksena johon tämä dokumentaatio liittyy.

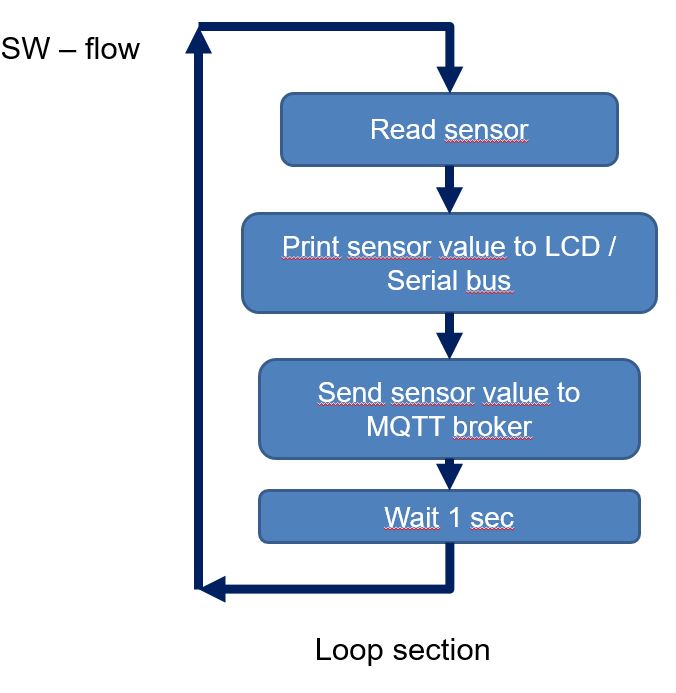
# DOKUMENTAATION TAVOITTEET, RAJAUS JA TULOKSET

Projektin tavoitteena on aikaan saada ilmanpaineanturin tietojen siirtyminen oikeassa muodossa TAMK:in serverille ja että ilmanpainetta pystytään lukemaan numeerisesti sekä graafisesti web-selaimella vaaditussa aikataulussa ja tekniset vaatimukset täyttäen.

Sekä oppia miten toimitaan työelämän projektityöympäristössä tehden yhteistyötä eri sidosryhmien kanssa, tiimin vetäjänä eli vastuuhenkilönä ja jäsenenä. Tämä tekninen dokumentaatio sisällytetään osaksi projektisuunnitelmaa.

Dokumentaatio on rajattu edellä mainittuihin projektin tavoitteisiin.

## 3. OHJELMIEN ARKKITEHTUURIN KUVAUS



|  |  |
| --- | --- |
| **3.1** | LCD-näytön testiohjelma |
| Kuvaus: | (Ks. koodikommentit liitteistä.)  #include <LiquidCrystal.h>  LiquidCrystal lcd(8, 7, 6, 5, 4, 3);  void setup() {  // put your setup code here, to run once:  lcd.begin(16, 2);  lcd.clear();  }  void loop() {  // put your main code here, to run repeatedly:  lcd.setCursor(0,0);  lcd.print("TEST"); |
|  |  |
|  |  |
| **3.2** | Arduino ohjelma |
| Kuvaus:  paine\_aika.ino  Hae IP  BMP.begin  BMP.read\_pressure  printTime  laheta\_pyynto  Connect\_MQTT\_server();  send\_message(paine); | (Ks. koodi liitteistä.)   * Arduino W5100 EtherCard ICT 2016 alkukoodit - edit 15.1.2016 KN * // Kytkeydytään TAMK:n verkkoon ja pyydetään IP-osoite. * Luodaan yhteys ilmanpainesensoriin. * // Adafruit kirjaston funktio joka palauttaa painedatan pascaleina. * // Kutsuu lähetä-pyyntö funktiota ja tulostaa sen LCD-näytölle muodossa hh:mm:ss * //Parsii ajanhakupyynnöstä saadun datan tulostettavaan aikamuotoon. * //Lähettää NTP-pyynnön palvelimelle josta palautuu EPOC-aika. * // Otetaan yhteys MQTT serveriin. * // Ottaa parametrina painelukeman ja lähettää sen MQTT serverille |

|  |  |
| --- | --- |
| **3.2** | MQTT C++ ohjelma |
| Kuvaus: | (Ks. myös koodikommentit liitteistä) |
| main.cpp | Vaateita 3kpl:tta.  MQTT luku.  LähetysTAMK:in serverille.  Ajastin ajan lähettämiseksi. |
| mqtt\_arduino.cpp | Arduinon lähettämän mittausdatan lukeminen ja MQTT protokollan käyttö.  Keskustelun kuuntelu : outTopic  Vastaanotetun datan tallentaminen globaaliin datapöytään table: data[40]  Kaikkien signaalien mittausdata jaotellaan ja tallentuu A ja B luokan datana. |
| SendTamk.cpp | Kaiken signaalidatan lähettäminen serverille yhtä aikaa.  Kaikki mittausdata lähetetään yhteen URL-osoitteeseen.  Mittausdatan lähetys A-luokan ja B-luokan tietokantaan.  CURL-kirjaston käyttö GET-pyynnön lähettämiseksi serverille.  1. Data kopioituu globaalista datapöydästä // data[] // paikalliseen datapöytään // apudata[] //  2. URL ylätunnisteen luonti // string url\_id //  3. Jokainen datasignaali lisätään // url\_id ) //:hen.  4. URL lähetetään // curl\_easy\_perform(hnd) // käskyllä.  5. // float ltoflo(int index) // Luetaan data paikallisesta datapöydästä // apudata[] //  6. Muutetaan pitkä integer data-arvo floatiksi, indeksinä on signaalinumero (=tiimi 32).  7. URL-lähetys A- ja B-luokka tietokantoihin.    8. Lähetysvaateet odottavat MQTT viestejä ennen [URL:in](file:///C:\Users\Saint\Dropbox\ICT-Projektit\in) lähettämistä serverille. |
|  |  |
|  |  |
| **3.3** | Raspberry ohjelma |
| Kuvaus: | (Ks. koodikommentoiti liitteistä.) |
|  |  |
|  | #define IP\_tamk  //#define IP\_koti  //////////////////  //// Raspberryn IP Koti  #ifdef IP\_koti  #define IP\_1 192  #define IP\_2 168  #define IP\_3 1  #define IP\_4 7  #endif    //// Raspberryn IP TAMK  #ifdef IP\_tamk  #define IP\_1 192  #define IP\_2 168  #define IP\_3 112  #define IP\_4 23  #endif  // apu\_sek = asek % 120; // apu\_sek 0-120, 0-120 ....  // apu\_sek = asek % osam;    // apu\_sek = apu\_sek / 5; // 120/5 = 24 apu\_sek 1-24 , 1-24, .... // apu\_sek = apu\_sek / jakaja;  // apu\_sek=asek % 15; // testausta varten nopeampi paivitys vali  //  // Testi moodi /////////////  //#define osam 15 // testi moodi 0-15, 0-15, o-15... sek  //#define jakaja 1 // testi moodi  // Mittaus ////////////////  #define osam 120 // mittaus moodi 0-120, 0-120, o-120... sek  #define jakaja 3 // mittaus moodi / 120/3 = 40 apu\_sek 1-40 , 1-40, ....  /////////////////////////////////////////////////////////////////////////////  #ifdef IP\_koti  char website[] = "time1.mikes.fi";  #endif  #ifdef IP\_tamk  char website[] = "ntp.tamk.fi";  #endif  //char website[]PROGMEM = "1.us.pool.ntp.org";  ///char website[]PROGMEM = "ntp2d.mcc.ac.uk";  //char website[]PROGMEM = "ntp2c.mcc.ac.uk";  //char website[]PROGMEM = "ntp.exnet.com";  //char website[]PROGMEM = "ntp.cis.strath.ac.uk";  //char website[]PROGMEM = "clock01.mnuk01.burstnet.eu"; |

# 4 FUNKTIOT

Arduino: (Ks. koodikommentointi liitteistä)

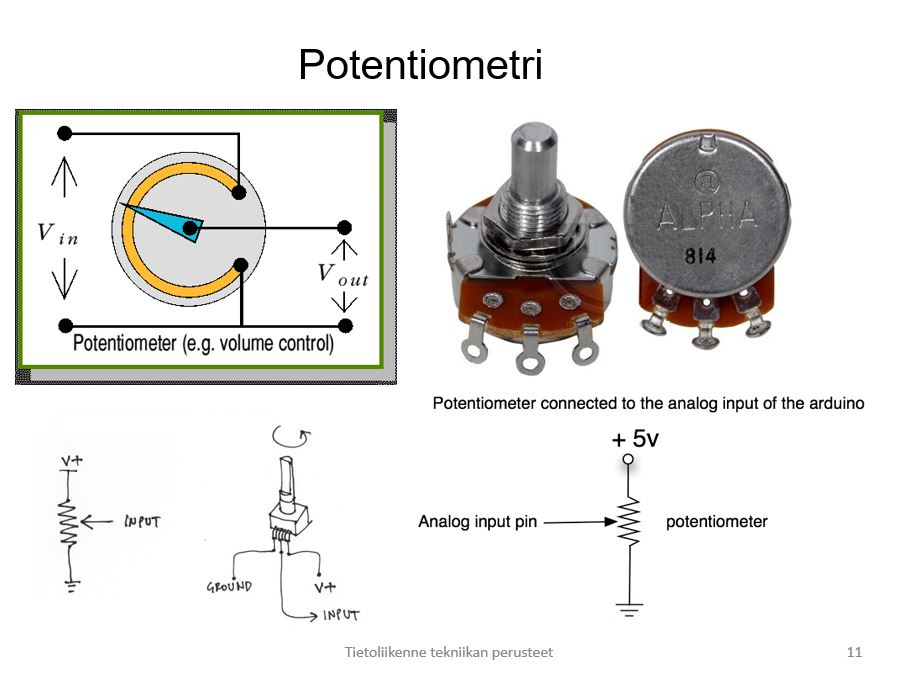
|  |
| --- |
| 1. // serial monitor on  2. // include LCD library  3. // incluide Ethernet library  4. // include MQTT library  5. // LCD pin wiring settings  6. // Ethernet configuration  7. // MAC nro  8. // Last mac number MSB mac numbers at ethernet\_mega.c  9. // Not relevat with Ethershield  10. // MQTT settings  11. // MQTT port number  12. // \* set your device id (will be the MQTT client username)  13. // \* set a random string (max 23 chars, will be the MQTT client id) |
| 14. // \* set your device secret (will be the MQTT client password) |
| 15. // MQTT Server settings  16. // HOME IP MQTT server address  17. // TAMK IP  18. // subscription callback for received MQTTT messages  19. // mqtt client  20. // MQTT topic names  21. // \* MQTT channel where data are received  22. // \* MQTT channel where data is send  23. // Used input pins  24. // pulse counting pin 19 = INT2  25. // AD input pin = A7  26. // AD input pin = A8  27. // AD input pin = A9  28. // AD input pin = A9  29. // AD input pin = A  30. // AD input pin = A9  31. // AD input pin = A  32. // AD input pin = A9  33. // AD input pin = A  34. // AD input pin = A9’  35. // volatile variables  36. // SETUP section  37. // put your setup code here, to run once:  38. // Serial monitor baudrate = 9600  39. // Counting Pin 2 as input  40. // Pull up resistors ON  41. // interrupt routine defination  42. // Display size defination 20 char 4 row  43. // set curosor to left upper corner  44. // print to lCD  45. // print to serial monitor  46. // initialize Ethernet connection  47. // get IP number from DHCP  48. // connect to MQTT server  49. LOOP section  50. // put your main code here, to run repeatedly:  51. // contious loop  52. // keep MQTT connection online  53. // Voltage measuring  54. // clear 2 rd row  55. // Voltage measuring  56. // clear 3 rd row  57. // Voltage measuring  58. // clear 4 rd row  59. // 1.23V = dark  60. // 4.5 V bright 4.5-1.23=3.27  61. // print scaled value to LCD  62. // print voltage to serial monitor  63. // SEND to MQTT server measured voltage  64. // 2.5V = 0 C  65. // 4.5V = 20 C  66. // print scaled value to LCD  67. // print voltage to serial monitor  68. // SEND to MQTT server measured voltage  69. // print scaled value to LCD  70. // print voltage to serial monitor  71. // SEND to MQTT server measured voltage  72. // print scaled value to LCD  73. // print voltage to serial monitor  74. // read AD converter value, put readed value to variable measure  75. // scale measure value to 0.0-5.0  76. LOOP END  77. //Read pulses with interrupt service  78. // time to count pulses (in milliseconds)  79. // start time from millisocond counter  80. // wait pulse counting time 1 sec  81. // Re connect if connection is lost  82. // keep MQTT connection online |
| 83. //input pin interrupt routine  84. //Interrupt service  85. // increment pulse counter each time when input pin moves from "0" to "1"  86. // Ethernet routines  87. // GET IP number from DHCP server  88. //MQTT Routines  89. // Send message  90. // check sign  91. // create message with header and data  92. // Print message to serial monitor  93. // send message to MQTT server  94. // Re connect if connection is lost  95. // Re Connection  96. // wait for reconnecting  97. // Receive incoming MQTT message  98. // copy the payload content into a char\*  99. // copy received message to receiv\_string  100. // print reveived message to LCD  101. // MQTT server connection  102. // Print MQTT server IP number to Serial monitor  103. // Print MQTT server IP number to LCD  104. // check if allready connected  105. // connection to MQTT server  106. // Connection is OK  107. // subscript to in topic  108. // error in connection |

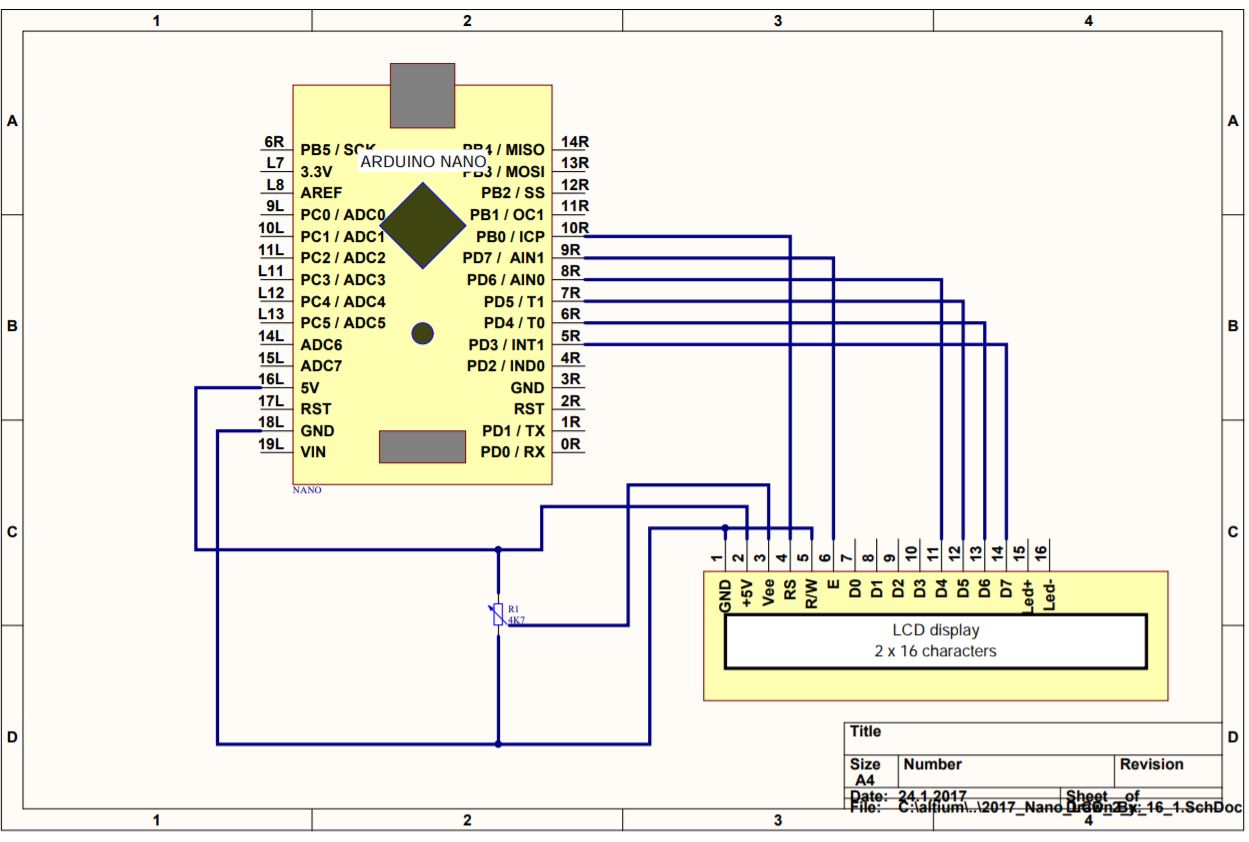
# 5 RAJAPINNAT

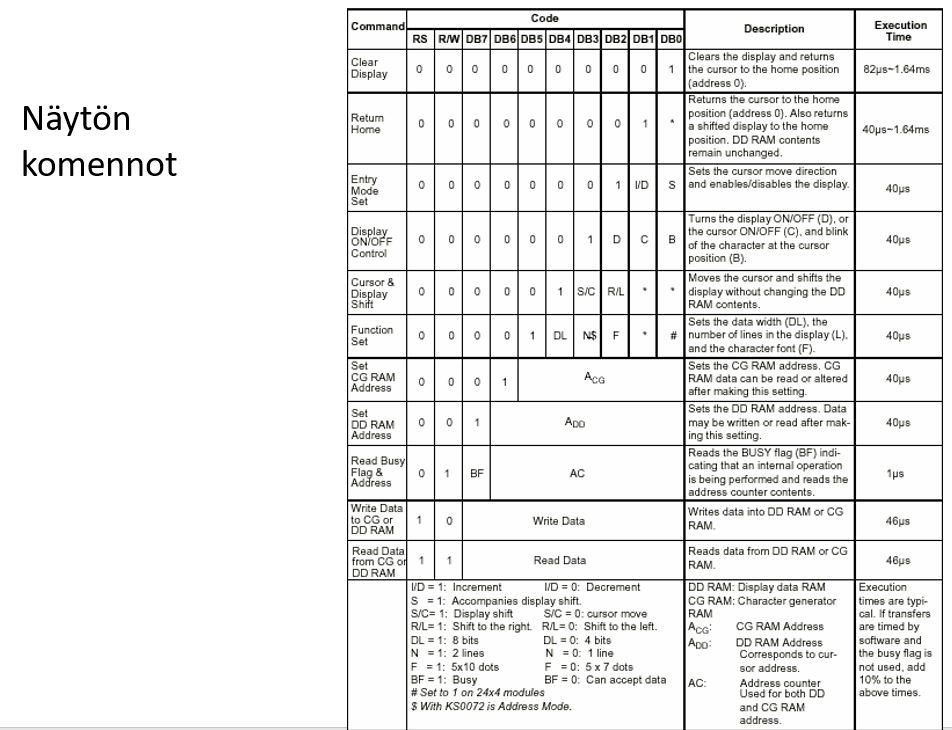
* SW
* Arduino
* Raspberry
* MQTT
* MySQL
* REST-API
* Web UI
* HTML
* PHP
* JS
* C++
* Ethernet moduuli
* HW Elektroniikka
* Anturi mittaamiseen (BMP180)
* Alusta jossa :
* Regulaattori (+8V, 5V, 3.3V)
* Arduino nano – ohjelma valmiina
* Ethernet moduuli
* Koekytkentäalusta kytkennän tekoon
* LCD-näyttö
* Näppäimistö

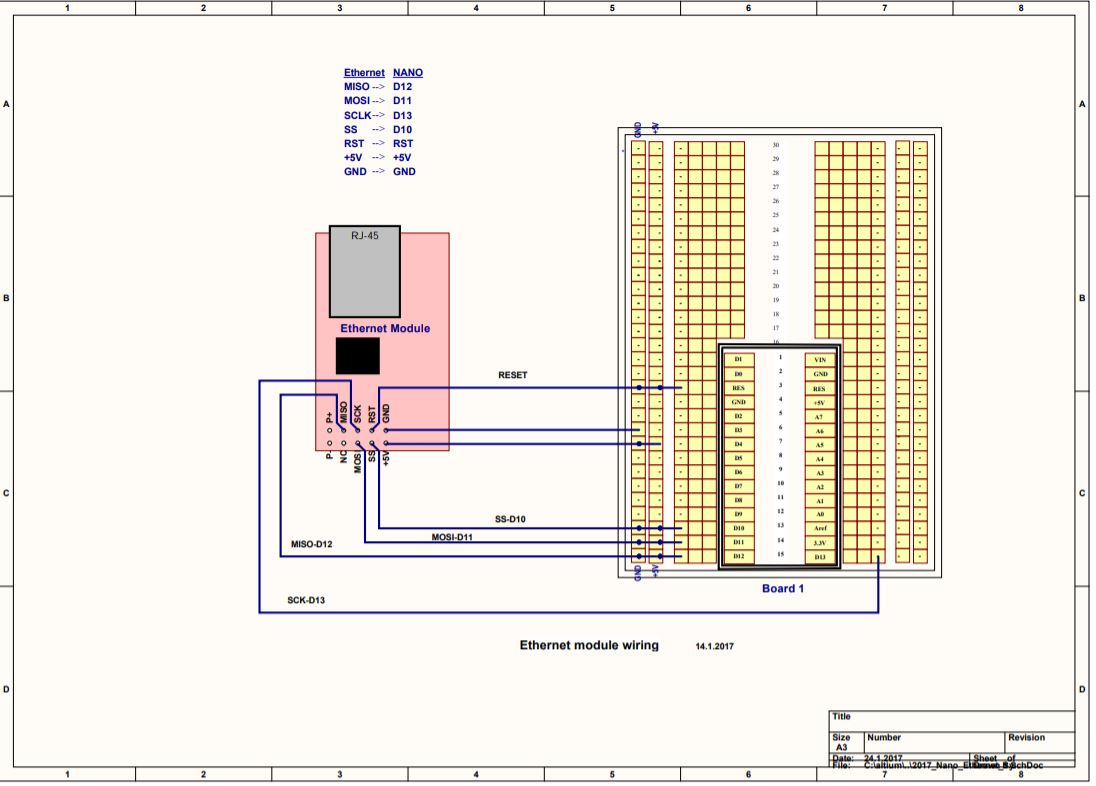
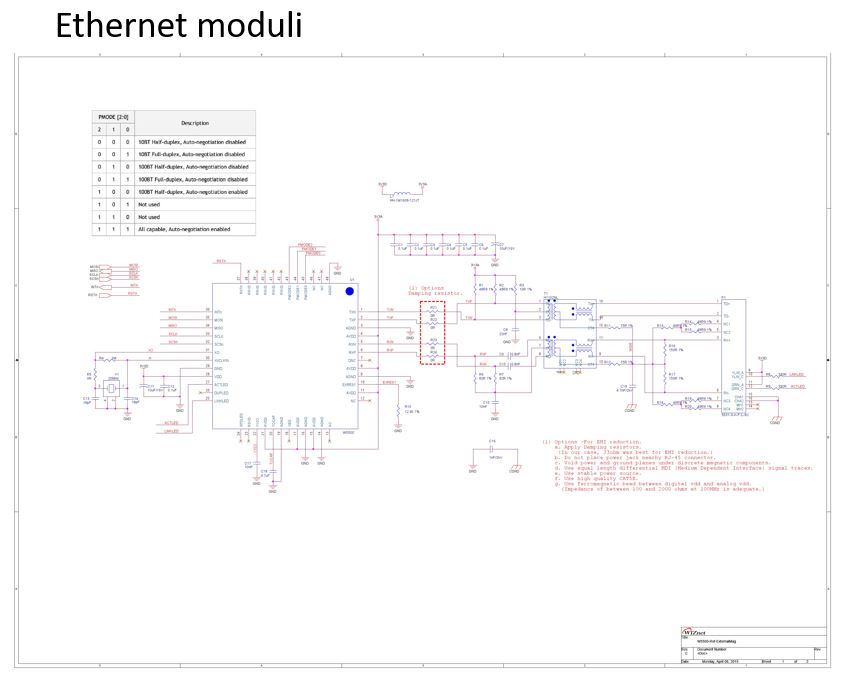
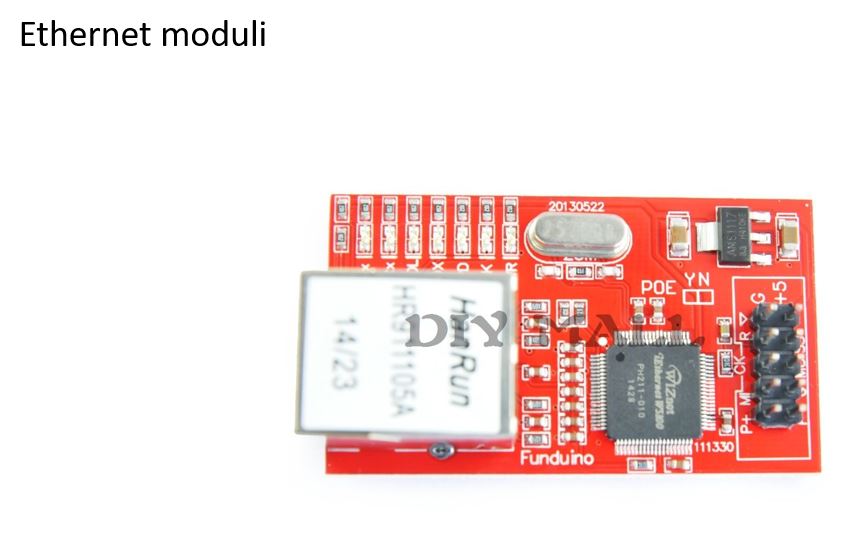
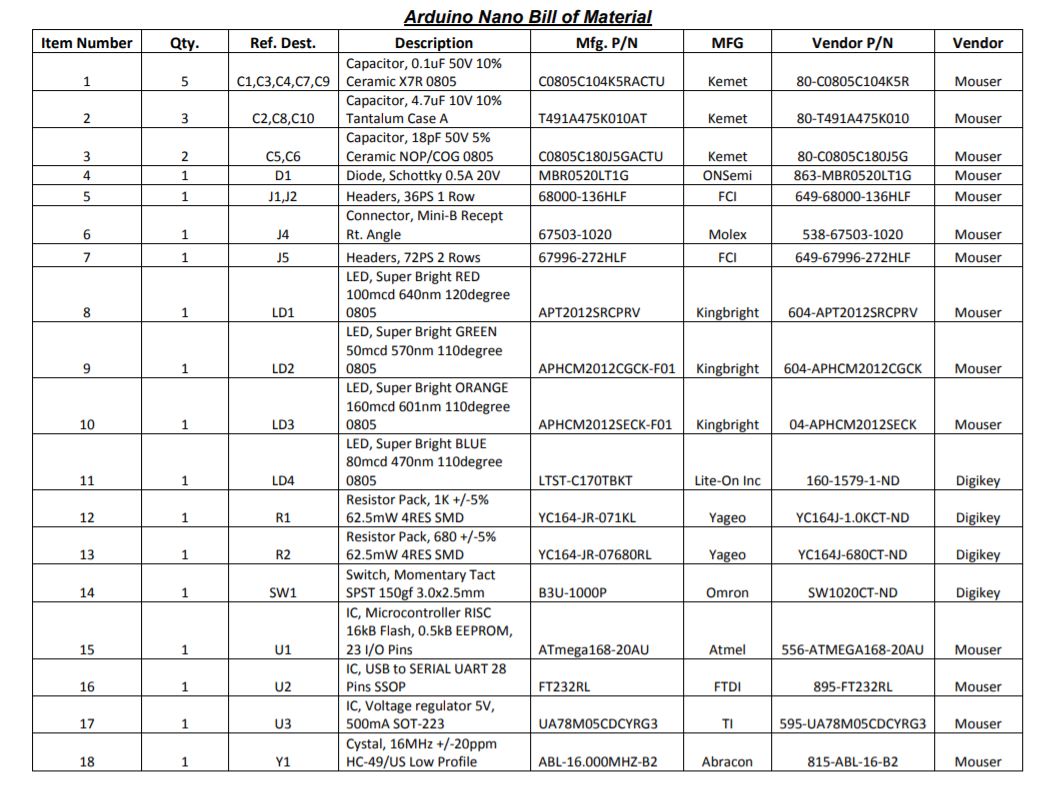
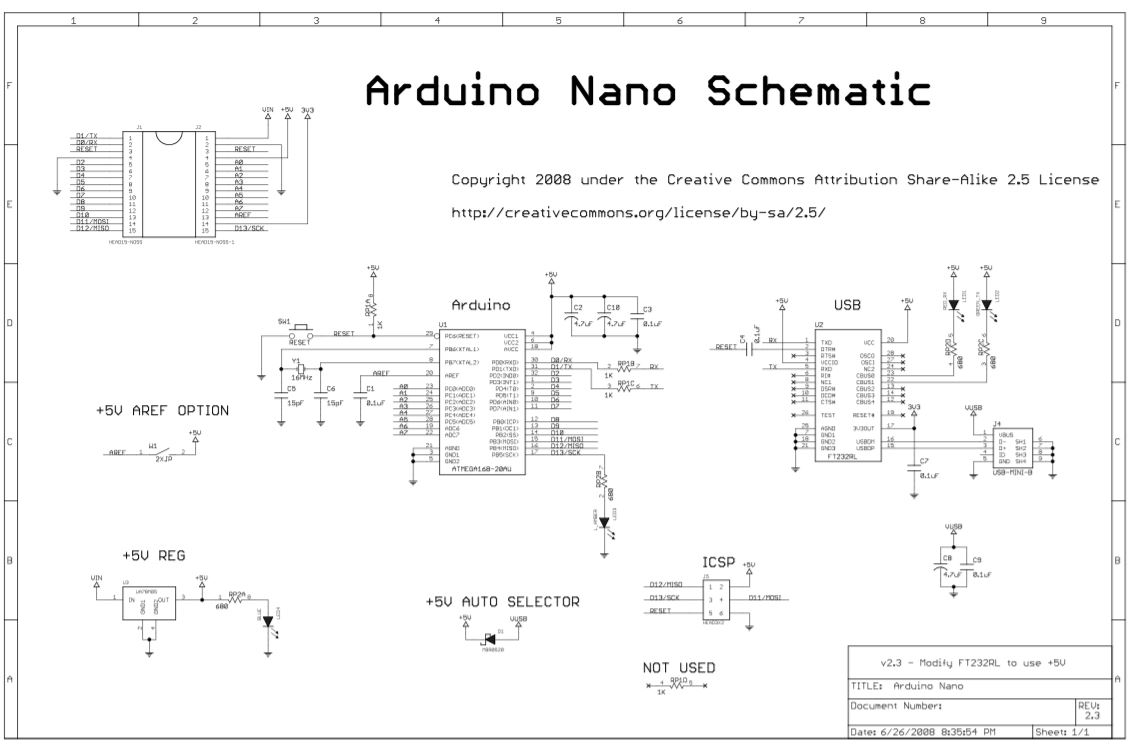
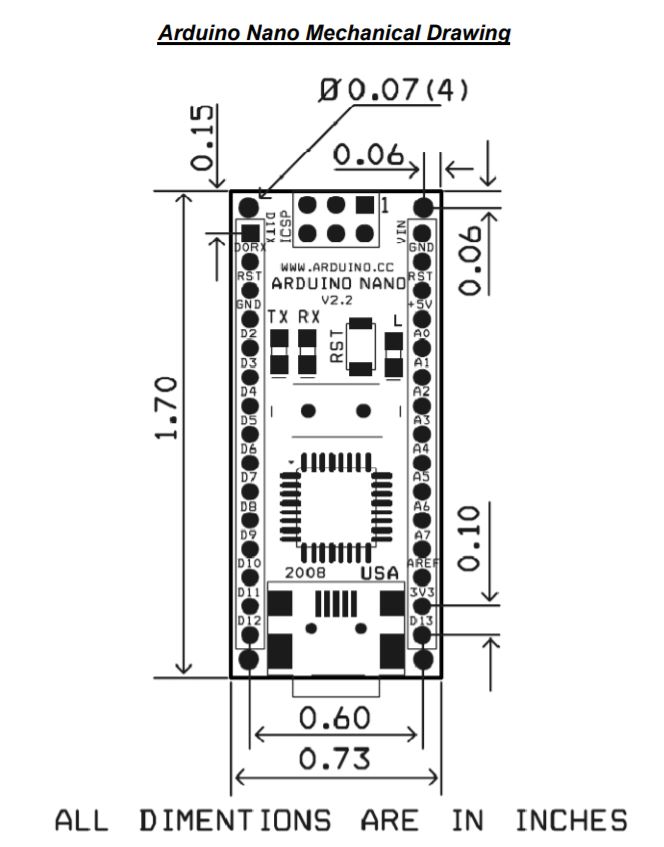
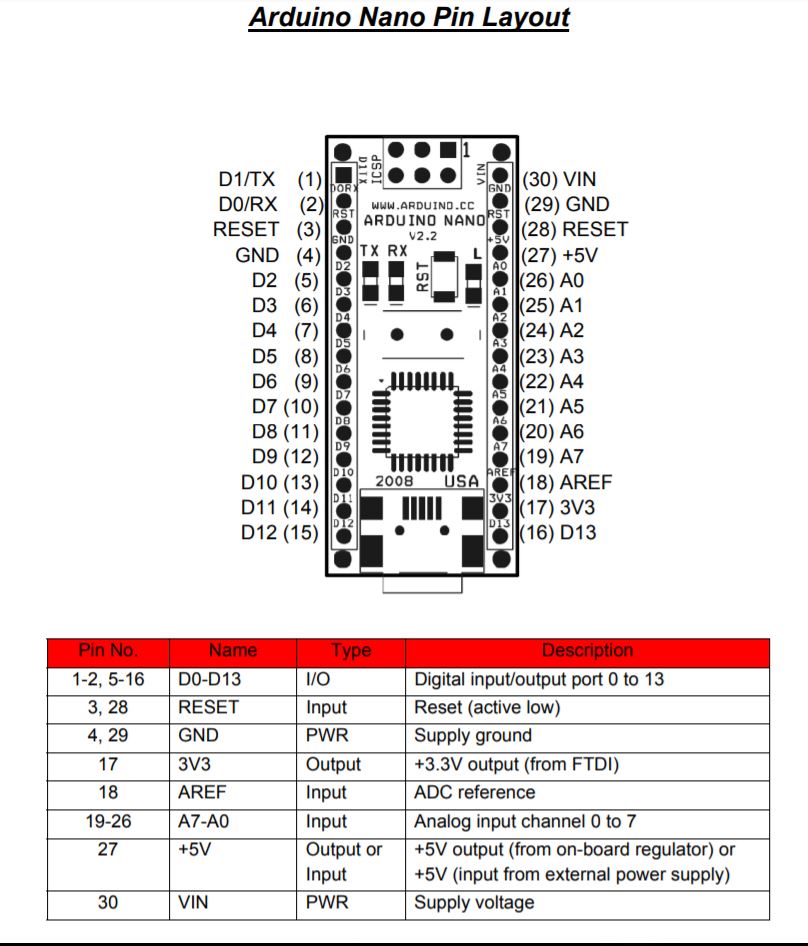
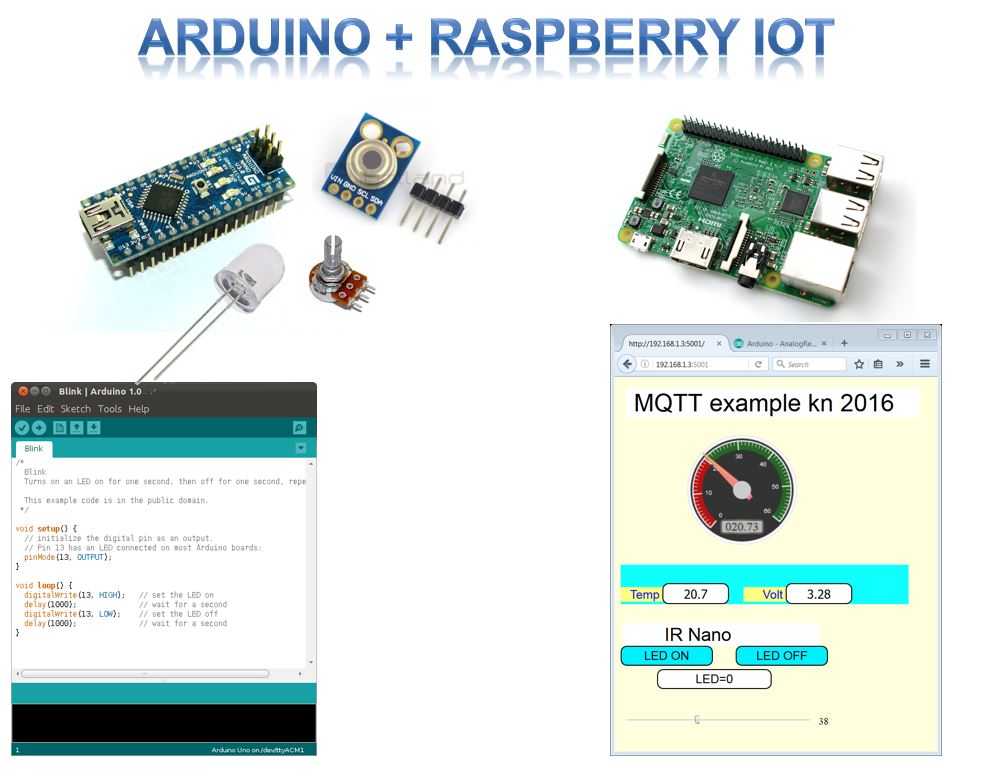
# 6 KAAVIOT, PIIRRUSTUKSET JA KUVAT

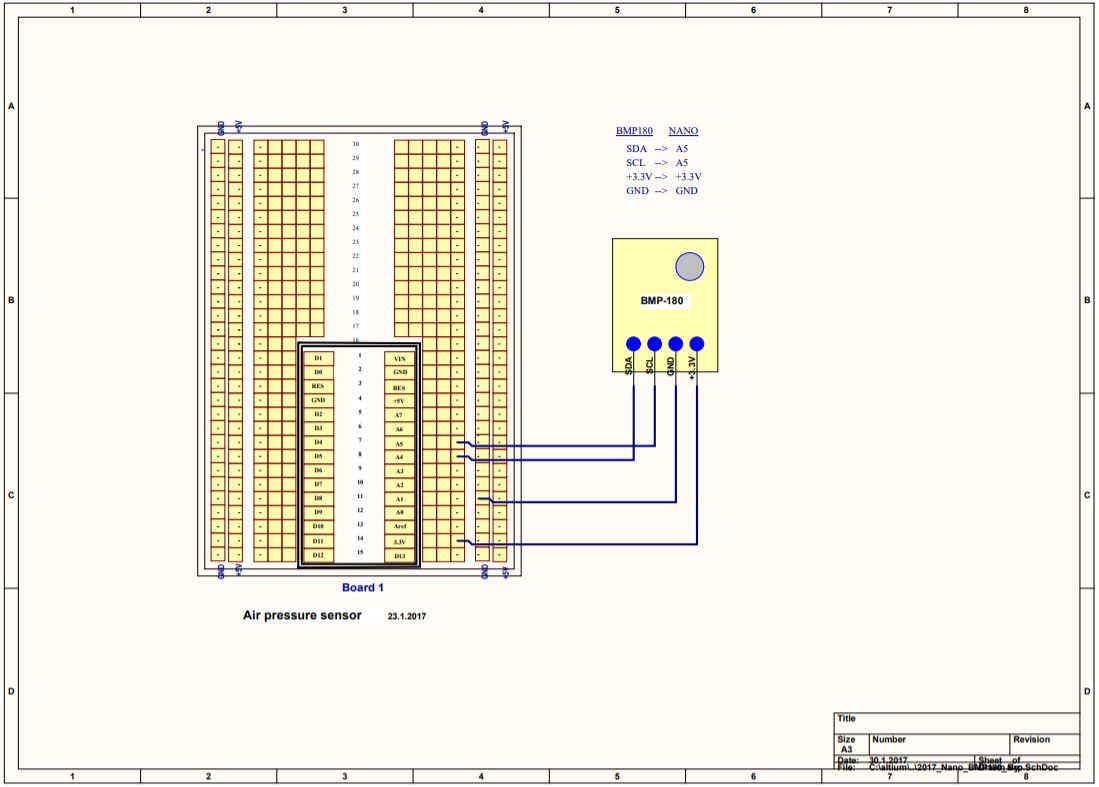
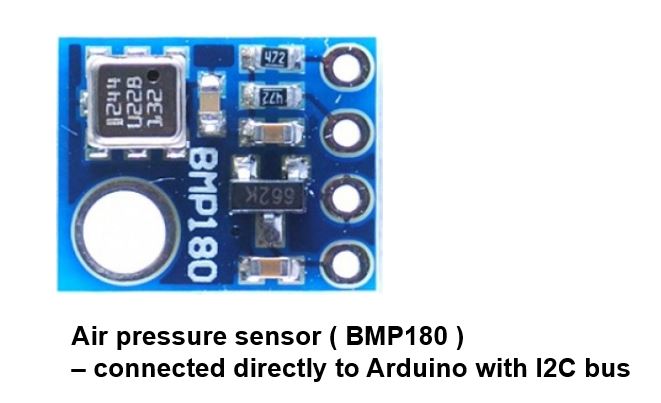


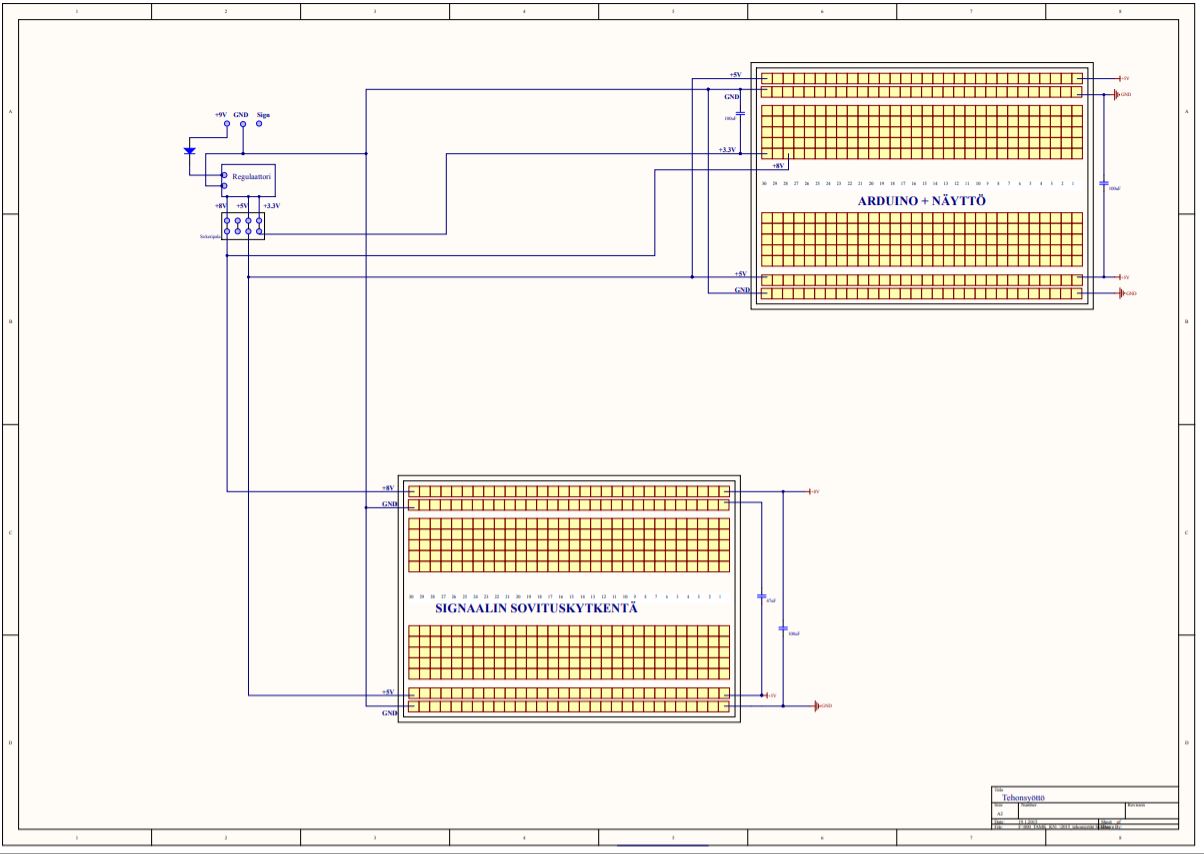
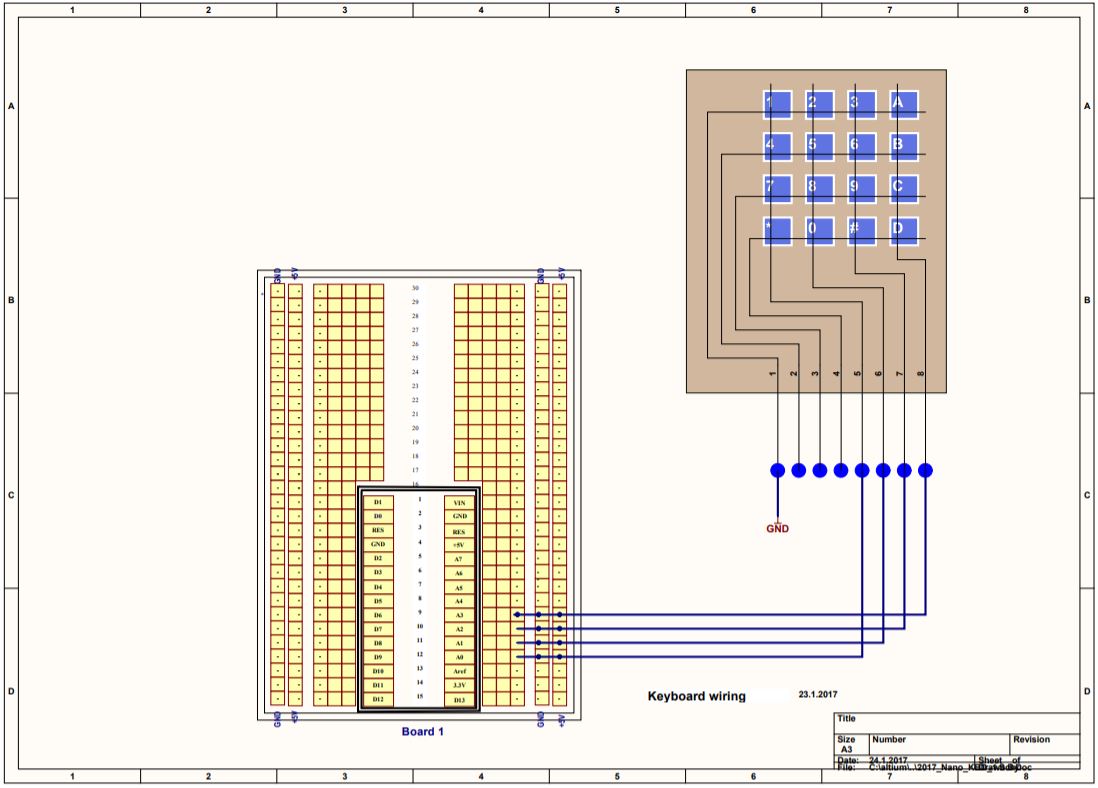












# 6 LIITTEET

* Ohjelmakoodit kommentteineen
* /\*
* \* Arduino W5100 EtherCard ICT 2016 alkukoodit - edit 15.1.2016 KN
* \*/
* #define DEBUG
* #define SERIAL\_ON
* #include <SPI.h>
* #include <Ethernet.h>
* #include <EthernetUdp.h>
* #include <PubSubClient.h>
* #include <TimerOne.h>
* #include <avr/interrupt.h>
* #include <avr/io.h>
* #include <LiquidCrystal.h>
* #include <Time.h>
* #include <TimeLib.h>
* #include <avr/pgmspace.h>
* //#include <raspIP.h>
* #include <Adafruit\_BMP085.h>
* #include <Wire.h>
* //
* //
* // Kanavanumerot
* //
* // A B
* //
* // Tuulen nopeus 1 1 21
* // Tuulen suunta 1 2 22
* // Lampotila 1 3 23
* // Sademaara 1 4 24
* // Kosteus sisa 5 25
* // Kostesus ulko 6 26
* // Tuulen suunta2 7 27
* // Tuulen nopeus2 8 28
* // Lampotila2 9 29
* // Sademaara2 10 30
* // Ilmanpaine1 11 31
* // Ilmanpaine2 12 32
* /////////////////////////////////////////////////////////////////////////////////////////////
* /// signaali ///
* /// ILmanpaine 1 ///
* ///////////////// ///
* /// Kanava nro ///
* ///
* #define kanava 32 // A ///
* #define kanava\_numero "32" ///
* ///
* //////////////// ///
* ///MAC nro // eka kanava MAC6 A luokka 0x21-0x32 ///
* // B luokka 0x41-0x52 ///
* #define mac\_6 0x52 ///
* ///
* /////////////////////////////////////////////////////////////////////////////////////////////
* Adafruit\_BMP085 bmp;
* #define mittaus\_m 1
* LiquidCrystal lcd(8,7,6,5,4,3);
* static uint8\_t mymac[6] = { 0x54,0x55,0x58,0x10,0x00,mac\_6 };
* unsigned int localPort = 8888; // local port to listen for UDP packets
* const int NTP\_PACKET\_SIZE = 48; // NTP time stamp is in the first 48 bytes of the message
* byte packetBuffer[ NTP\_PACKET\_SIZE]; //buffer to hold incoming and outgoing packets
* // A UDP instance to let us send and receive packets over UDP
* EthernetUDP Udp;
* #define inTopic "ict1\_in\_2018" // \* MQTT channel where lelylan updates are received
* #define outTopic "ict1\_out\_2018" // \* MQTT channel where lelylan updates are received
* char\* deviceId = "2016"; // \* set your device id (will be the MQTT client username)
* char\* deviceSecret = "tamk"; // \* set your device secret (will be the MQTT client password)
* char\* clientId = "1234567890"; // \* set a random string (max 23 chars, will be the MQTT client id)
* char website[] = "ntp.tamk.fi";
* //////////////////
* //// Raspberryn IP Luetaan tiedostoa raspIP.h
* //#define IP\_1 192
* //#define IP\_2 168
* //#define IP\_3 1
* //#define IP\_4 2
* //// Raspberryn IP TAMK ICT
* //#ifdef IP\_tamk
* #define IP\_1 10
* #define IP\_2 10
* #define IP\_3 206
* #define IP\_4 141
* //#endif
* /\* Server settings \*/
* byte server[] = { IP\_1, IP\_2, IP\_3,IP\_4 }; // MQTT serveri osoite

* /\* Ethernet configuration \*/
* EthernetClient ethClient;
* /\* MQTT communication \*/
* void callback(char\* topic, byte\* payload, unsigned int length); // subscription callback
* PubSubClient client(server, 1883, callback, ethClient); // mqtt client
* // Find list of servers at http://support.ntp.org/bin/view/Servers/StratumTwoTimeServers
* // Please observe server restrictions with regard to access to these servers.
* // luetaan tiedostosra raspIP.h
* //char website[]PROGMEM = "ntp2d.mcc.ac.uk";
* //char website[]PROGMEM = "ntp2c.mcc.ac.uk";
* //char website[]PROGMEM = "ntp.exnet.com";
* //char website[]PROGMEM = "ntp.cis.strath.ac.uk";
* //char website[]PROGMEM = "clock01.mnuk01.burstnet.eu";
* byte aika\_nro=0; // onnistuneiden ajan lukujen maara
* uint32\_t paine;
* uint32\_t timeLong;
* // Number of seconds between 1-Jan-1900 and 1-Jan-1970, unix time starts 1970
* // and ntp time starts 1900.
* void setup()
* {

* lcd.begin(16,2); // Näytön koon määrittely 16 merkkiä 2 riviä
* lcd.setCursor(0,0);
* lcd.print("14.3.2018 ");

* lcd.setCursor(0,1);
* // 1234567890123456
* lcd.print("Odotan IP nroa");
* #ifdef SERIAL\_ON
* Serial.begin(9600);
* Serial.println( F("EtherCard/Nanode NTP Client" ) );
* #endif
* hae\_IP(); // Kytkeydytään TAMK:n verkkoon ja pyydetään IP-osoite.
* if (!bmp.begin()) {
* Serial.println("Could not find a valid BMP085 sensor, check wiring!");
* }
* lcd.clear();
* } // Setup END
* ////////////////////////////////////////////////////
* ///////////////// LOOPIN ALKU
* ///////////////////////////////////////////////////
* void loop()
* {
* paine = bmp.readPressure(); // Adafruit kirjaston funktio joka palauttaa painedatan pascaleina.
* lcd.setCursor(0,0);
* lcd.print(float(paine) / 100);
* lcd.setCursor(8,0);
* lcd.print("mbar");
* Serial.print("Pressure = ");
* Serial.print(paine);
* Serial.println(" Pa");


* lcd.setCursor(0,1);
* printTime(); // Noutaa palvelimelta kellonajan ja tulostaa sen LCD-näytölle muodossa hh:mm:ss
* Connect\_MQTT\_server(); // Otetaan yhteys MQTT serveriin
* send\_message(paine); // Ottaa parametrina painelukeman ja lähettää sen MQTT serverille
* Serial.println();
* delay(500);
* } // Loop End
* void printTime()
* {
* laheta\_pyynto(); // Lähettää ajanhakupyynnön palvelimelle.
* time\_t aika = timeLong; //Parsii ajanhakupyynnöstä saadun datan tulostettavaan aikamuotoon.
* int tunnit = hour(aika) + 2;
* int minuutit = minute(aika);
* int sekunnit = second(aika);
* if (tunnit < 10)
* {
* lcd.print("0");
* lcd.print(tunnit);
* }
* else
* {
* lcd.print(tunnit);
* }
* lcd.print(":");
* if (minuutit < 10)
* {
* lcd.print("0");
* lcd.print(minuutit);
* }
* else
* {
* lcd.print(minuutit);
* }
* lcd.print(":");
* if (sekunnit < 10)
* {
* lcd.print("0");
* lcd.print(sekunnit);
* }
* else
* {
* lcd.print(sekunnit);
* }
* }
* ///////////////////////////////////////////
* //////// IP numeron haku
* //////////////////////////////////////////
* void hae\_IP(void)
* {
* byte rev=1;
* rev=Ethernet.begin( mymac);
* #ifdef SERIAL\_ON
* Serial.print( F("\nW5100 Revision ") );
* #endif
* if ( rev == 0){
* #ifdef SERIAL\_ON
* Serial.println( F( "Failed to access Ethernet controller" ) );
* #endif
* // 1234567890123456
* lcd.setCursor(0,1); lcd.print("FAIL E cont conn");
* }
* #ifdef SERIAL\_ON
* Serial.println( F( "Setting up DHCP" ));
* Serial.print("Connected with IP: ");
* Serial.println(Ethernet.localIP());
* #endif
* lcd.setCursor(0,0);
* lcd.print("IP=");
* lcd.print(Ethernet.localIP());
* }
* ///////////////////////////////////////////////
* ////////// ajan hakupyynto
* ///////////////////////////////////////////////
* void laheta\_pyynto(void)
* {
* byte DSNError=1;
* Udp.begin(localPort);
* sendNTPpacket(website); // send an NTP packet to a time server
* delay(200);
* if (Udp.parsePacket())
* {
* // We've received a packet, read the data from it
* Udp.read(packetBuffer, NTP\_PACKET\_SIZE); // read the packet into the buffer
* // the timestamp starts at byte 40 of the received packet and is four bytes,
* // or two words, long. First, extract the two words:
* unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
* unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
* // combine the four bytes (two words) into a long integer
* // this is NTP time (seconds since Jan 1 1900):
* unsigned long secsSince1900 = highWord << 16 | lowWord;
* // now convert NTP time into everyday time:
* // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
* const unsigned long seventyYears = 2208988800UL;
* // subtract seventy years:
* unsigned long epoch = secsSince1900 - seventyYears;
* timeLong=epoch;
* }
* else
* {
* delay(10000);
* Ethernet.maintain();
* }
* aika\_nro++; // ajan luku onnistui

* } /// ajan paivitys loppuuu
* // send an NTP request to the time server at the given address
* void sendNTPpacket(char\* address) {
* // set all bytes in the buffer to 0
* memset(packetBuffer, 0, NTP\_PACKET\_SIZE);
* // Initialize values needed to form NTP request
* // (see URL above for details on the packets)
* packetBuffer[0] = 0b11100011; // LI, Version, Mode
* packetBuffer[1] = 0; // Stratum, or type of clock
* packetBuffer[2] = 6; // Polling Interval
* packetBuffer[3] = 0xEC; // Peer Clock Precision
* // 8 bytes of zero for Root Delay & Root Dispersion
* packetBuffer[12] = 49;
* packetBuffer[13] = 0x4E;
* packetBuffer[14] = 49;
* packetBuffer[15] = 52;
* // all NTP fields have been given values, now
* // you can send a packet requesting a timestamp:
* Udp.beginPacket(address, 123); //NTP requests are to port 123
* Udp.write(packetBuffer, NTP\_PACKET\_SIZE);
* Udp.endPacket();
* }
* /////////////////////////////////////////////////////////////////////////////////
* //////////////// MQTT RUTIINIT
* ////////////////////////////////////////////////////////////////////////////////
* /// Laheta sanoma
* void send\_message(uint32\_t data)
* {
* char bufa[25];
* sprintf(bufa,"32:+%06lu",(unsigned long)data);
* #ifdef SERIAL\_ON
* Serial.println( bufa );
* #endif
* client.publish(outTopic,bufa);
* }
* /\* Receive MQTT message and confirm the physical change \*/
* void callback(char\* topic, byte\* payload, unsigned int length)
* {
* // copu the payload content into a char\*
* char\* json;
* json = (char\*) malloc(length + 1);
* memcpy(json, payload, length);
* json[length] = '\0';
* lcd.setCursor(0,0);
* lcd.print(json);
* #ifdef SERIAL\_ON
* Serial.println( json );
* #endif

* free(json);
* }
* /\* MQTT server connection \*/
* void Connect\_MQTT\_server()
* {
* // add reconnection logics
* if (!client.connected())
* {
* // connection to MQTT server
* if (client.connect(clientId, deviceId, deviceSecret))
* {
* lcd.setCursor(0,1);
* //0123456789012345//
* lcd.print("Connect ");
* #ifdef SERIAL\_ON
* Serial.println(" Connected" );
* #endif
* MQTTSubscribe(); // topic subscription
* }
* }
* }
* /\* MQTT subscribe \*/
* void MQTTSubscribe()
* {
* client.subscribe(inTopic);
* }