OU00CV35 SAUNASENSORI

Nykyajan löylyvahti

19. TAMMIKUUTA 2021

HENNA KERÄNEN, MINNA NIEMINEN, NINA KOSKI, PASI REVONMÄKI
OAMK HEDSY20

Sisällysluettelo

1. Esipuhe

Saunassa on ajan saatossa alkuun pantu paljon elämää ja myös synnytetty sitä. Tämän **Saunasensori**:n ideakin on syntynyt alun perin saunassa. Mutta sen sijaan että tämä **Saunasensori** synnyttäisi elämää, on se valjastettu suojelemaan sitä.

Viime vuosina on mediassa noussut pari ikävää kohua liittyen saunaan ja sen aiheuttamiin tapaturmiin. Toisessa tapauksessa pyörätuolissa ollut nainen kuoli kuumuuteen ja toisessa tapauksessa kuumaan saunaan sammunut juhlija sai vakavia vammoja.

Näitten siivittämänä saimme idean hyödyntää edullista teknologiaa nojaten mikrokontrollereihin ja pilvipalveluun ja luoda sovellus, sekä ohjelma, jotka osaavat varoittaa, jos kuumassa saunassa viivytään liian pitkään. SOTE-puolella tälle olisi tarvetta esimerkiksi sellaisten asiakkaiden parissa, jotka tarvitsevat automaattista valvontaa, mutta joiden itsenäistä asumista ja tässä tapauksessa omatoimista saunomista, pyritään tukemaan mahdollisuuksien mukaan. Ja mainoksen sanoin, siitä se idea sitten lähti…

https://www.iltalehti.fi/kotimaa/a/3d6c4396-6b2b-47e8-b598-8598db4253b5

https://www.iltalehti.fi/kotimaa/a/2c656788-1f01-4715-a11c-36cc11bf5253

Projektin evoluutio

Jaoimme projektin osiin alkuperäisessä ajattelussa näin:

- Idea ja onko se toteutettavissa tässä harjoitustyössä
- Miten se liittyy SOTE-puoleen
- Mikrokontrollerin valinta
- Mikrokontrollerin ja oheissälän, kuten sensoreiden hankinta
- Tutustuminen mikrokontrollerin sielunelämään
- Mikrokontrollerin koodaaminen ja sensoreiden asennus siihen
- Sen tuottaman datan saaminen tietokoneelle luettavaksi (sarjaportti oli alkuun se luontevin väylä)
- Datan saaminen ja käsittely **Visual Studiossa vb.netin Visual Basic-kielellä** luodulla sovelluksella
- Datan saaminen mikrokontrollerista langattomasti
- Sähköpostitoiminnallisuuden luominen
- Hälytysten logiikan tarkempi määritteleminen ja toteutus
- Käyttöliittymän hiominen
- Loppumietteet, kehityskohteet

2. Alkutoimet ja hankinnat

Meillä ei ollut juurikaan aiempaa kokemusta mikrokontrollereista, mitä nyt pintapuolisesti tunsimme pari yleisintä nimeltä, eli **Raspberry Pin** ja **Arduinon** ja sen mihin niitä voisi ehkä hyödyntää.

Asian havainnollistamiseksi ja Pasin yleisestä mielenkiinnosta johtuen hän päätyi tilaamaan **Arduinon** ja siihen oheistarvikkeita.

Tarkalleen ottaen mikrokontrolleriksi hankittiin **Arduinon** Wi-Fi-versio tai itse asiassa kokonainen kitti ja siihen vielä lisätarvikkeita.

Eli paketti oli **Arduino MKR IoT Bundle**, jossa perustana on **Arduino MKR1000**.

https://store.arduino.cc/arduino-mkr-iot-bundle

Sensoripaketti:

Arduino Sensor Kit – Base

https://store.arduino.cc/sensor-kit-base

Verkkokaupasta lisäksi vielä **HC-SR04-ultraäänisensori**.

https://www.verkkokauppa.com/fi/product/5217

3. Lataukset, asennukset, perusohjeitten kartoitus

Tähän **Arduino**-pakettiin oli hyvä lähteä tutustumaan tämän sivun kautta:

https://www.arduino.cc/en/Guide/MKR1000

Asensimme ensiksi *Arduino Software (IDE)n*, eli ***I**ntegrated **D**evelopment **E**nvironmentin*. Sen saimme täältä:

https://www.arduino.cc/en/software

Valitsimme **Windows**ille soveltuvan exe-asennuspaketin:

Latasimme ja tutustuimme yleensäkin **Arduinoon** ja tuon *IDE*:n asennusprosessiin täällä:

https://www.arduino.cc/en/Guide

Valitsimme sieltä tässä tapauksessa *Windows*-ohjeet

Siellä pyydettiin asentamaan nimenomaan exe-paketti, koska se asentaa automaattisesti myös ajurit. Sitten menimme oletuksilla, hyväksyimme kyselyt ja asennuksen jälkeen palasimme aloitussivulle ja valitsimme käytettävän alustan oikealla olevasta valikosta. Tässä tapauksessa se oli siis **MKR1000**.

Ja nyt olimme jo aiemmin mainitulla sivulla https://www.arduino.cc/en/Guide/MKR1000 ,jonka ohjeita seuraamalla jatkoimme tästä.

Ohjeen tässä vaiheessa kehotettiin poistamaan alustaa suojaava vaahtomuovin palanen ja näin myös teimme.

4. Arduinon hallintatavat

Arduinon IoT-versiolle on olemassa kolme päähallintatapaa ja niihin kaikkiin oli hyvä tutustua aluksi. Tavat ovat:

Pilvipalvelu, eli **Arduino IoT Cloud**

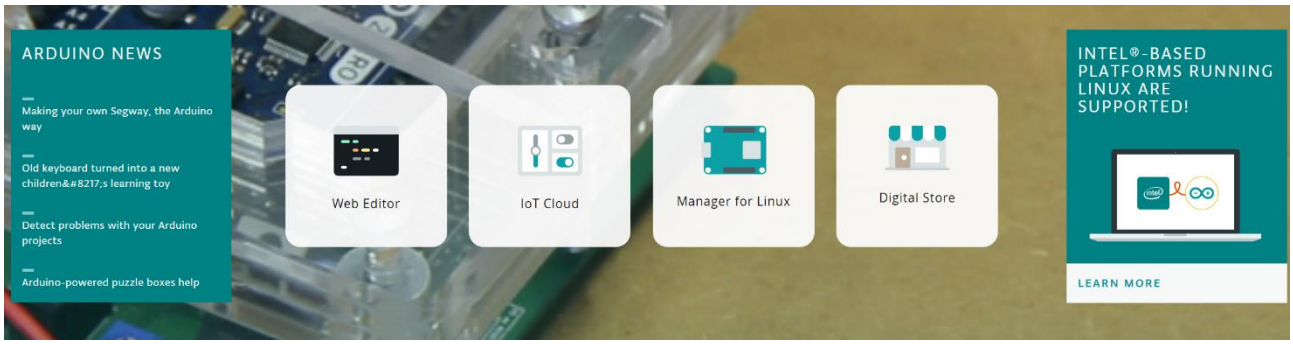
Web-editori, eli **Arduino Web IDE**

Työpöytähallintaliittymä, eli **Arduino Desktop IDE** paikalliseen hallintaan.

Pilvipalvelu, eli Arduino IoT Cloud

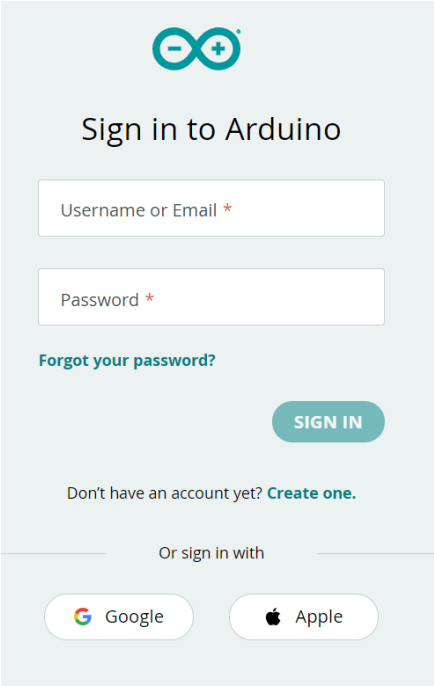
Tässä, kuten **Web Editorissakin**, tulee olla tili **Arduinon palvelussa**. Palvelu löytyy osoitteesta:

https://create.arduino.cc/



Noista valittiin nyt **IoT Cloud**

Ja sitten valitsimme **Create one** , koska sitä tiliä ei ollut ennestään.



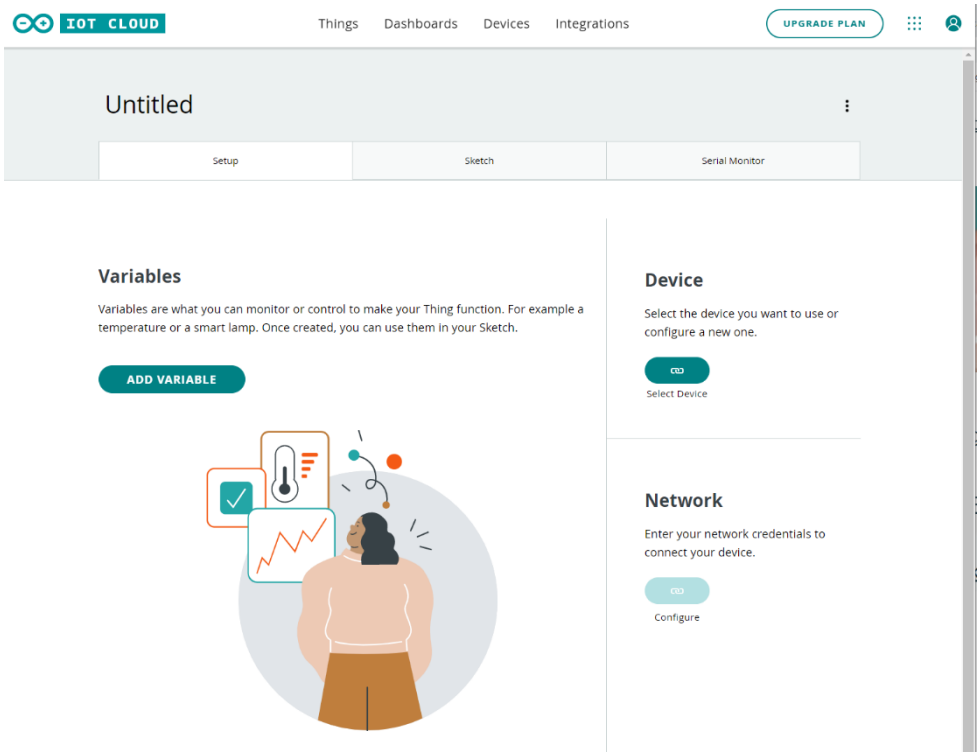
Ja tilin tekemisen ja sen vahvistamisen jälkeen pääsi kirjautuneena sitten aloitussivulle:

Create your first Thing

A Thing is a connected device that can communicate with the cloud. You can make your Things interact with other Things or anything else in the physical world.

CREATE THING

Ja tämän jälkeen oltiinkin heti jo itse asian äärellä.



Tässä kohtaa oli hyvä pysähtyä ja miettiä, sekä tutustua mitä kaikkea on tarjolla. Lupaavasti *Variablesin* kohdalla jo puhutaan lampuista ja lämpötiloista. Aikomuksenahan oli liittää **Arduinoon** ainakin ultraäänisensori, lämpötila-anturi, painike, liiketunnistin ja led.

Tutustuminen kannatti. Demosimme onnistuneesti (lopulta) pilvipalvelussa ledin ja painonapin asennuksen ja siinä samalla meille kirkastui, ettei pilvipalvelu tällä kertaa ollut toteutuksemme perusta. Toisaalta tietoturvasyistä ja toisaalta aikomus oli siirtää dataa suoraan PC:lle langattomasti ja lopullinen hallinta ilman kolmansia osapuolia. Tästä opettavaisesta harharetkestämme voi lukea liitteestä yksi, joka löytyy raportin lopusta.

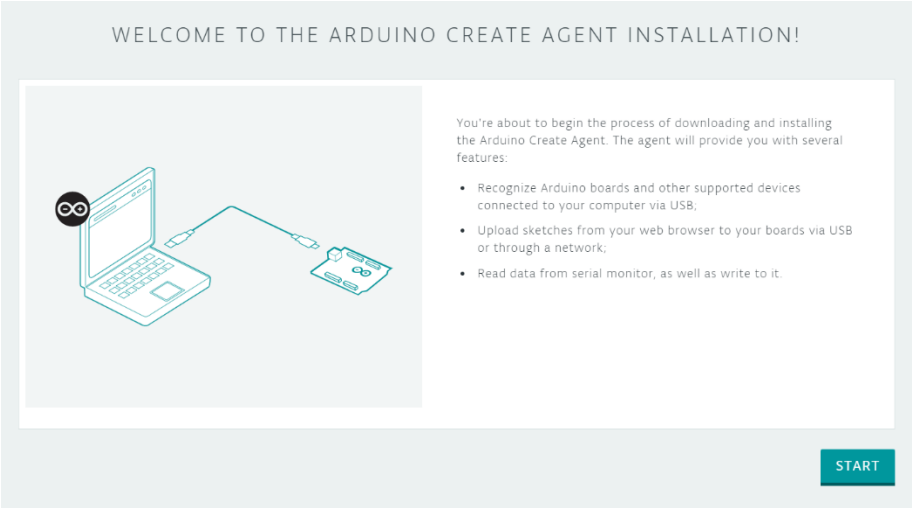
Siirryimme siis seuraavaksi tutustumaan **Web Editoriin**.

Web Editori, eli Arduino Web IDE

Tätä ennen kuitenkin ohjeessa oli asentaa *Arduino Create* -lisäosa joten asensimme sen ennen kuin se unohtui.

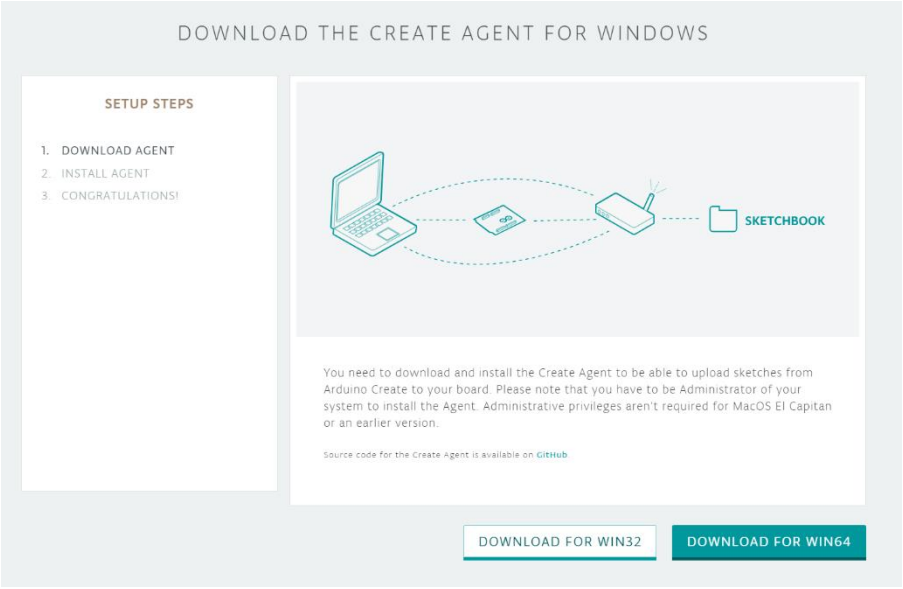
Se löytyi osoitteesta:

https://create.arduino.cc/getting-started/plugin/welcome

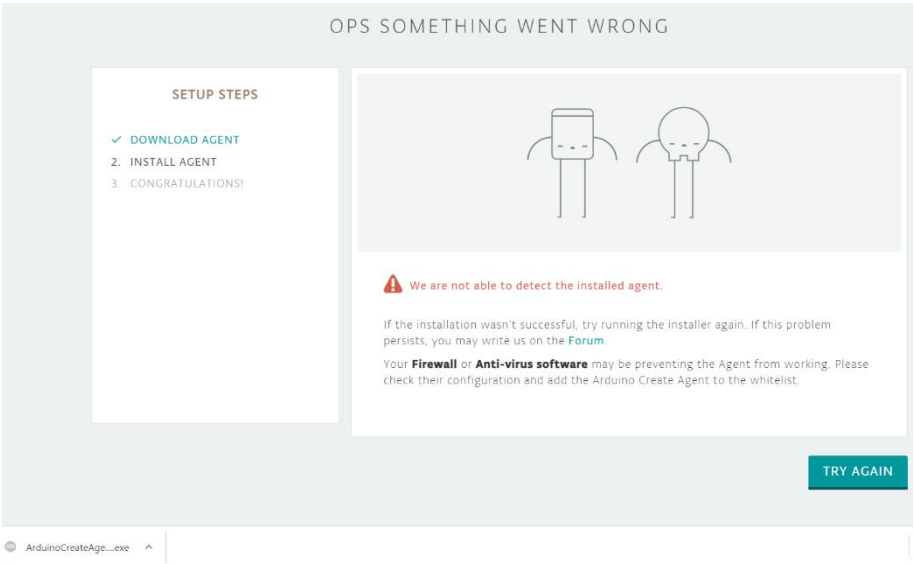


Kuten kuvan tekstistä kävi ilmi, niin tämä lisäosa mahdollisti monet kaipaamamme toiminnot meidän ja **Arduinon** välille. Klikkasimme siis *START*

Koska käytimme apuna *Windows 10* työasemaa, niin seuraava kysymys koski käyttämämme käyttöjärjestelmän bittisyyttä. Tämä kyseinen oli nykytyyliin 64-bittinen, joten valitsimme sen *(DOWNLOAD FOR WIN64)*.

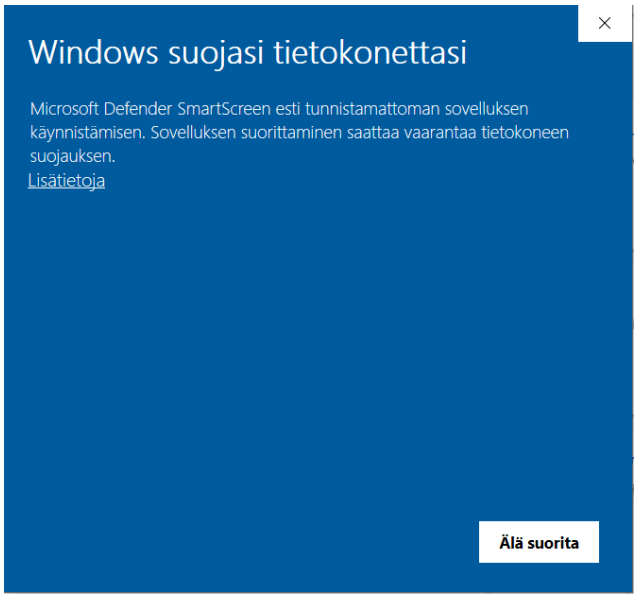


Asennusvelho oli tässä kohtaa hieman hassu. Se antoi virheilmoituksen, ettei asennettua agenttia löydy.

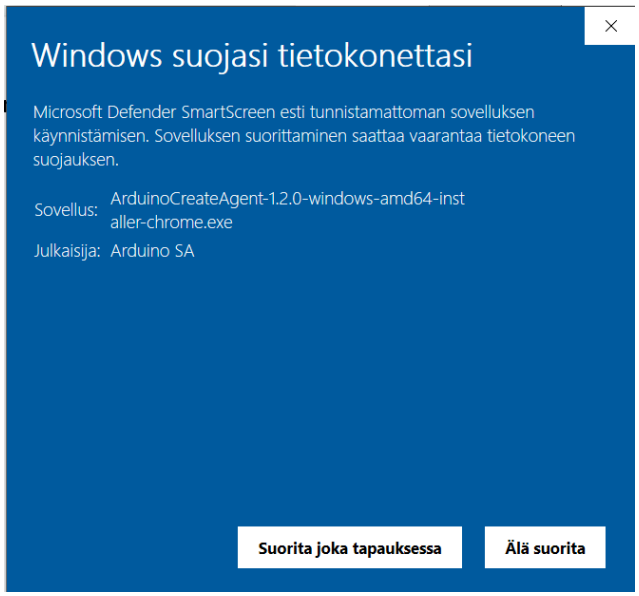


No ei löydy ei, koska asennuspakettihan vasta latautui ja sen merkkinä käyttämämme **Chrome**-selaimen vasempaan alakulmaan tuli latauspaketin nimi. Klikkasimme tätä ja lähdimme asentamaan agenttia.

Heti tuli seuraava kanto kaskeen:

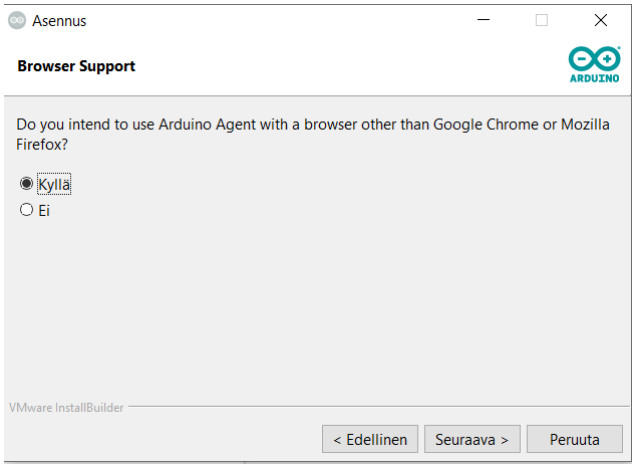


Tästä pääsimme eteenpäin klikkaamalla *Lisätietoja*, jolloin saimme mahdollisuuden suorittaa asennusohjelma.



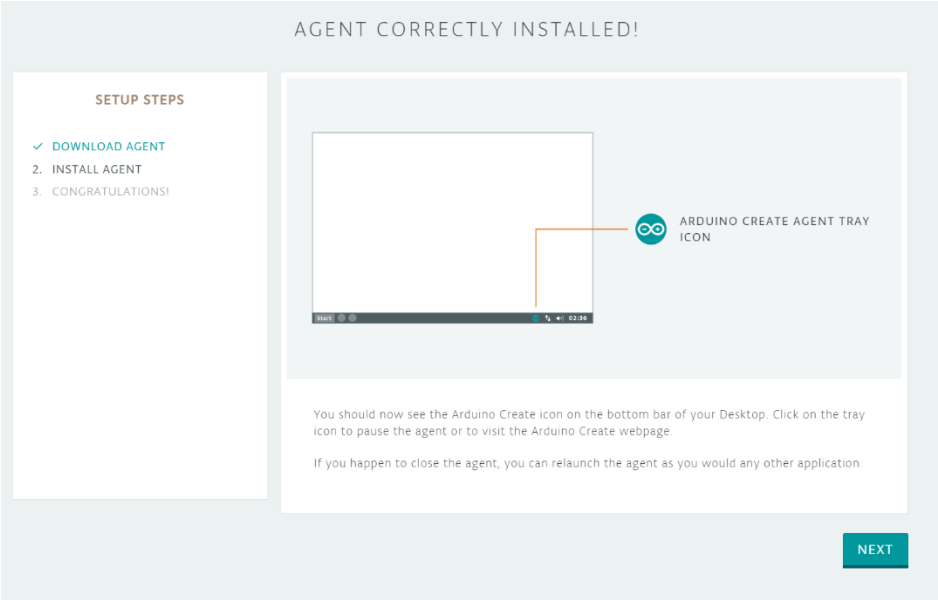
Klikattiin siis *Suorita joka tapauksessa*

Annoimme suorituslupia ja valitsimme asetuksia (mentiin oletuksin), mutta selaintuen kysymykseen vaihdoimme valinnan ja asensimme selaintuen. Tämä voisi olla kätevä.

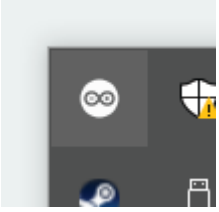


Hyväksyimme varmenteen ja palomuurin avauksen ja nyt asennus meni läpi.

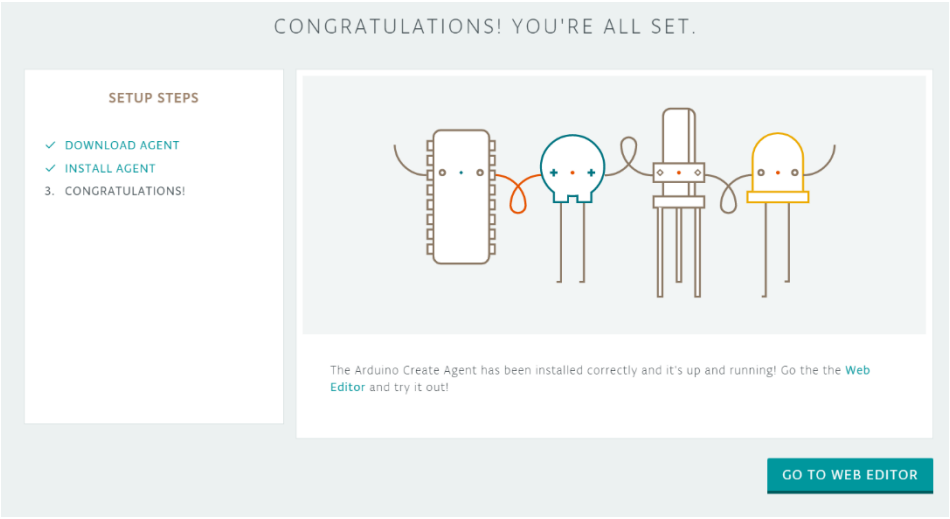
Nyt saatoimme jatkaa siitä ensimmäisen virheilmoituksen kohdasta eteenpäin.



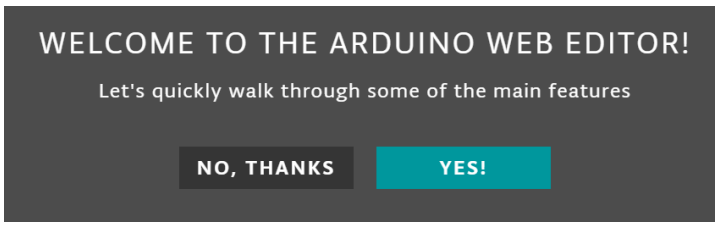
Jep, kuvake ilmestyi esimerkin mukaisesti, eli agentti pyöri nyt **Windowsin** taustalla.



Klikkasimme *Next* ja saimme vahvistuksen onnistumisestamme:

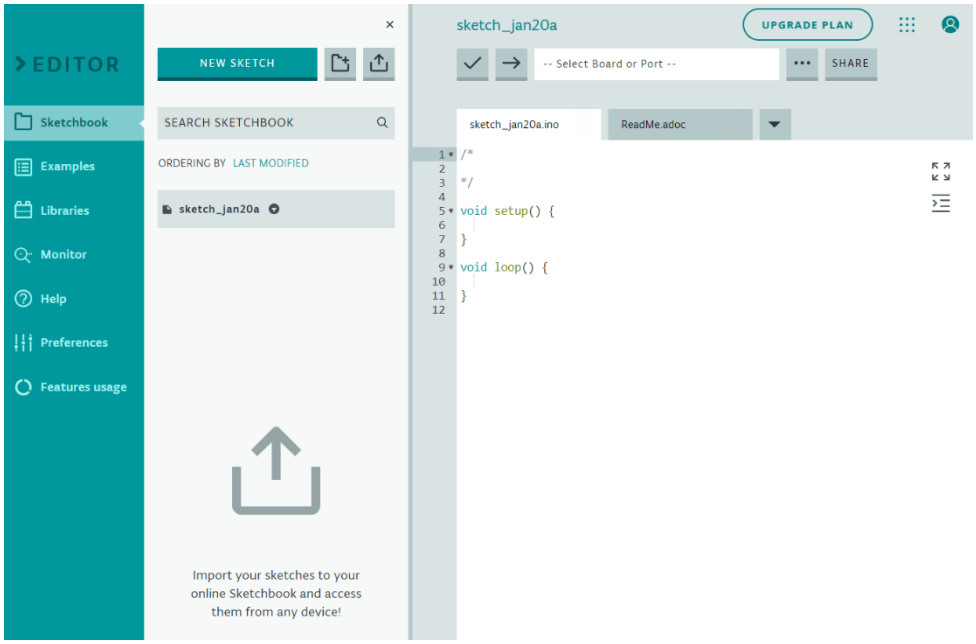


Klikkasimme *GO TO WEB EDITOR* ja kävimme läpi tutustumiskierroksen.



Ja näin meillä oli **Web-editori** auki. Näkymästä päätellen tulisimme pian väistämättä tutuksi ohjelmointikieli **C++**:n perusteiden kanssa, joten uuden oppimista olisi siltäkin osin tulossa. No, oppia ikä kaikki.

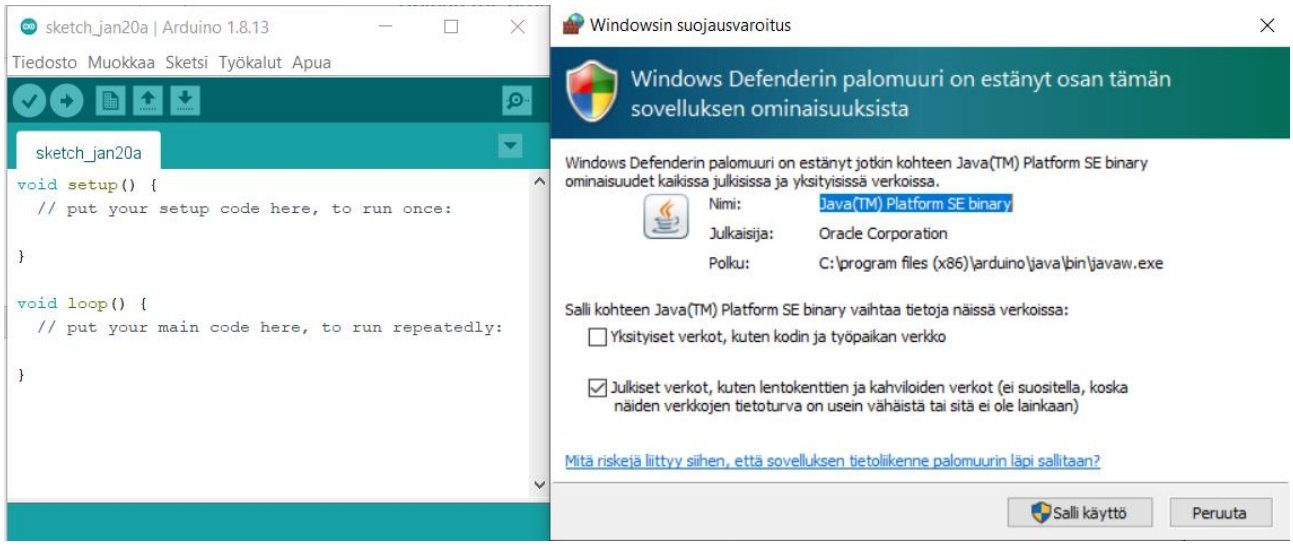
https://fi.wikipedia.org/wiki/C%2B%2B



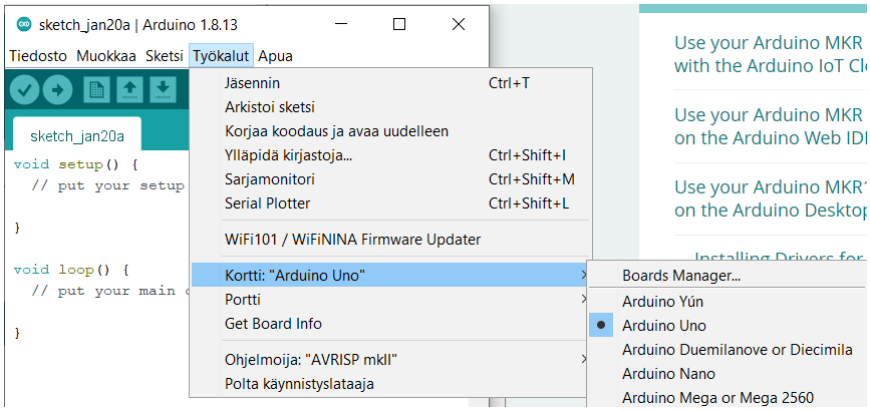
Työpöytähallintaliittymä, eli Arduino Desktop IDE

Kolmas tapa on se työpöytäsovellus ja yhteys kaapelilla. Tämä ei tarvitse internetyhteyttä. Tämä tarvitsi sen *Arduino Desktop IDE:n* ja lisäksi vielä *Atmel SAMD Coren*.

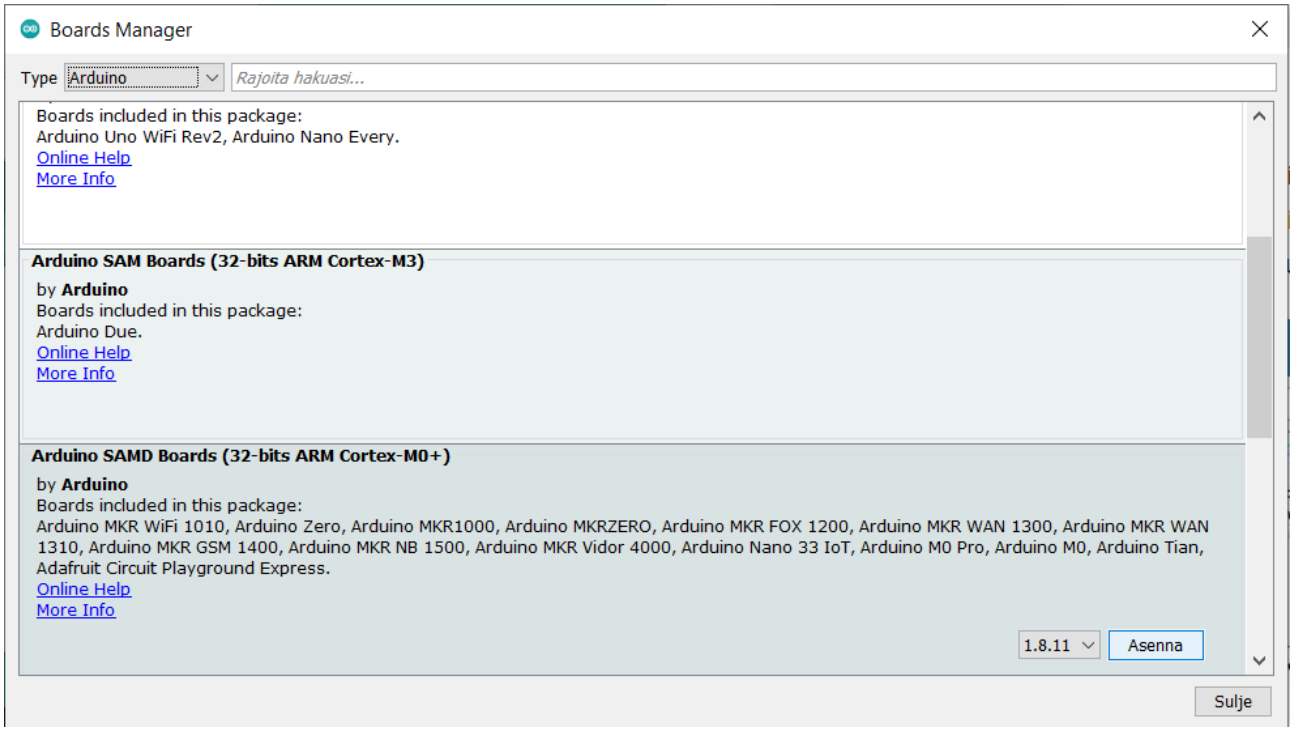
Kun *Arduino*-sovelluksen käynnisti ensimmäisen **Windows 10** PC:ssä, niin ensimmäiseksi velho kyseli palomuurilupia ja ne myös annoimme.



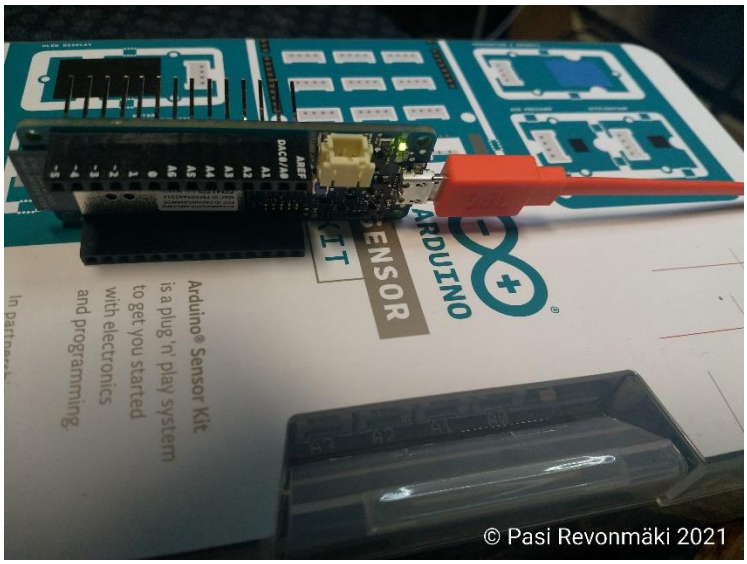
Nyt ohjeessa kehotettiin menemään *Boards Manageriin…*



jotta saisimme asennettua tuo *Atmel SAMD Coren.*

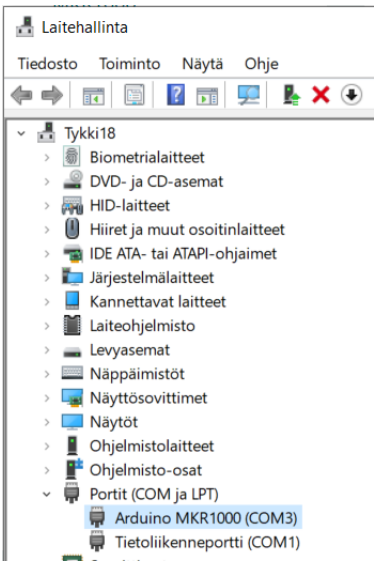


Listalla oli mainittuna käyttämämme alusta, eli **MKR1000**, hyvä! Klikkasimme siis *Asenna* ja annoimme luvat ajureiden asennukselle. Tämän jälkeen olimmekin sitten jännän äärellä, koska nyt kytkimme **Arduinomme** tietokoneeseen ja käynnistimme sen ihka ensimmäistä kertaa!

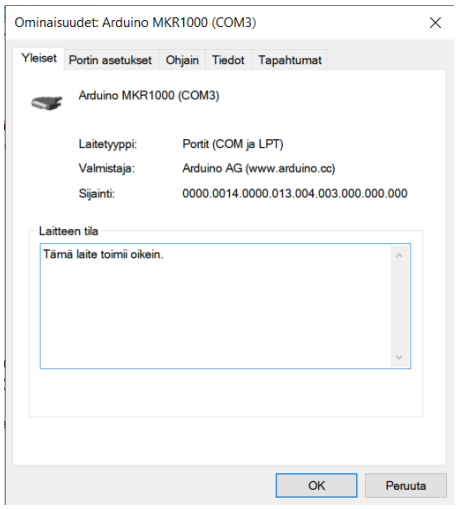


Hienoa, ledvalo syttyi, joten eloa oli.

Ohje pyysi nyt käynnistämään laitehallinnan (varsin hankalalla tavalla tosin), joten starttasimme sen simppelimmin, eli ***Windows-nappi + R*** ja avautuvaan ikkunaan kirjoitimme *devmgmt.msc* ja enteriä perään. Nyt olimme suoraan laitehallinnassa:

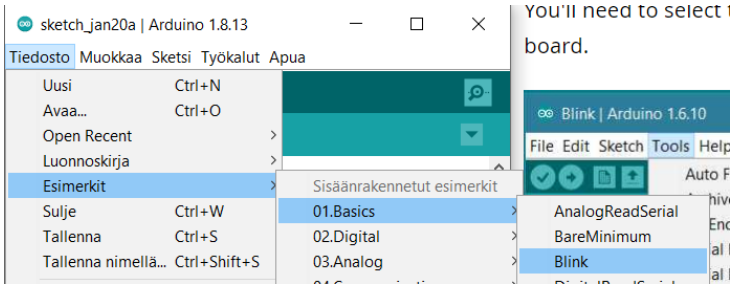


Haimme sieltä vinkatun **MKR1000**:n ja klikkasimme sitä hiiren toissijaisella ja valitsimme *Ominaisuudet* ja tarkistimme, että aiempi ajuriasennus oli onnistunut.

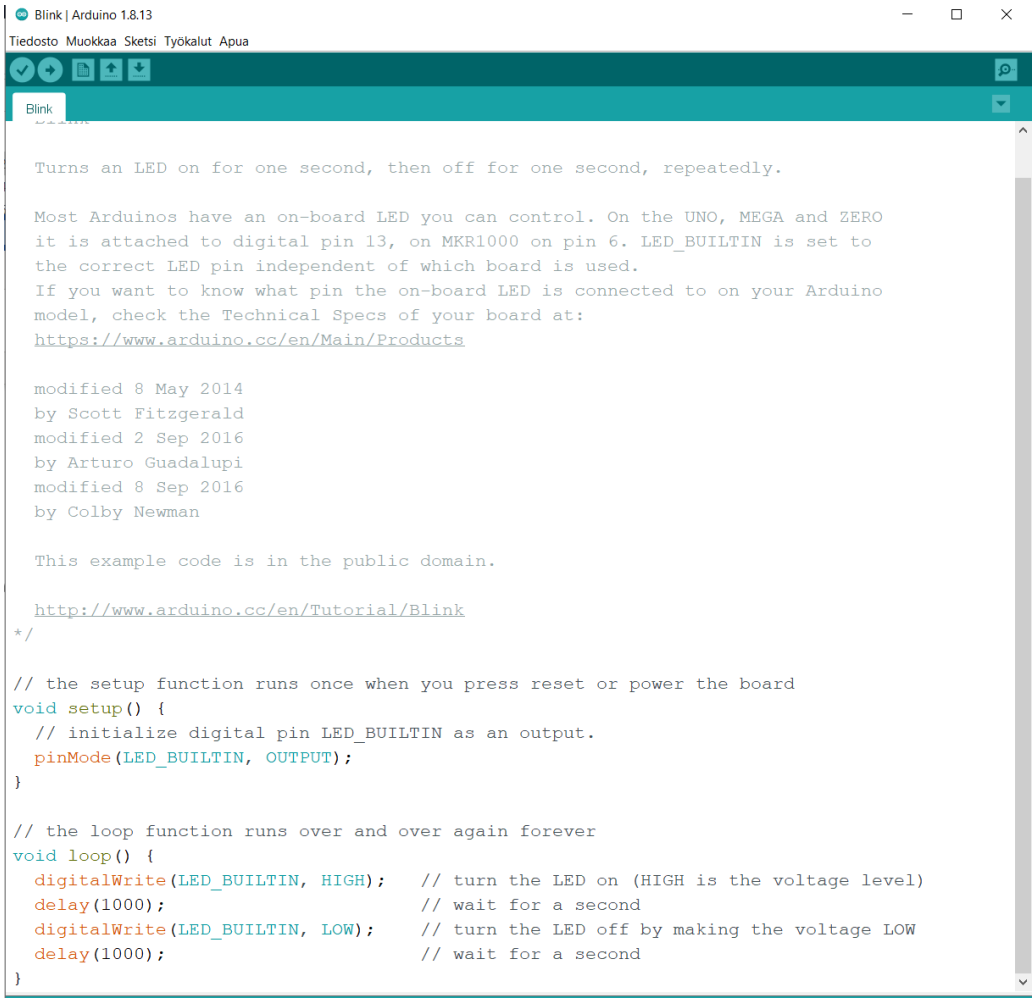


Kyllä vain, *Tämä laite toimii oikein*

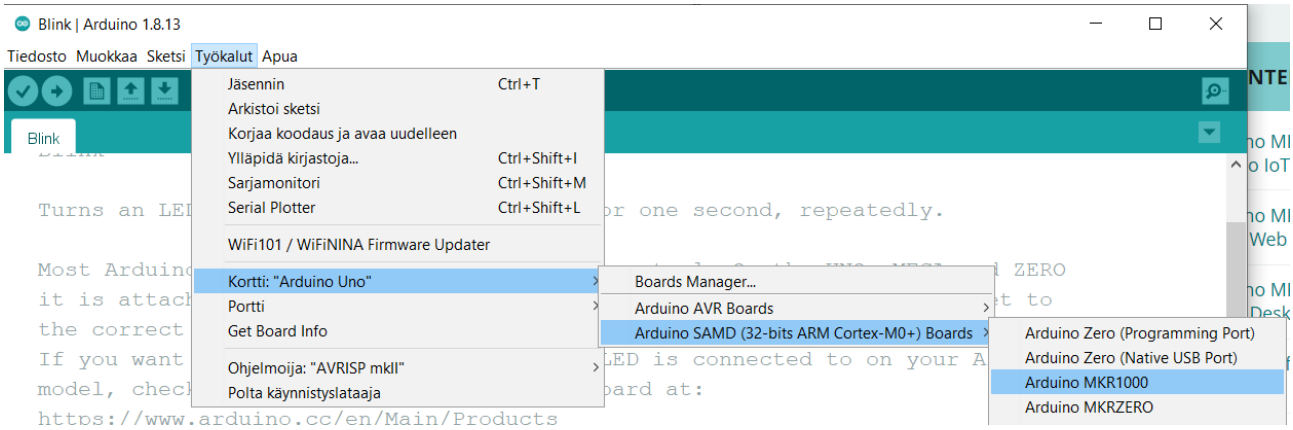
Testasimme toimintaa vielä ensimmäisellä esimerkkitoiminnolla. Avasimme esimerkeistä ledin vilkutuksen.



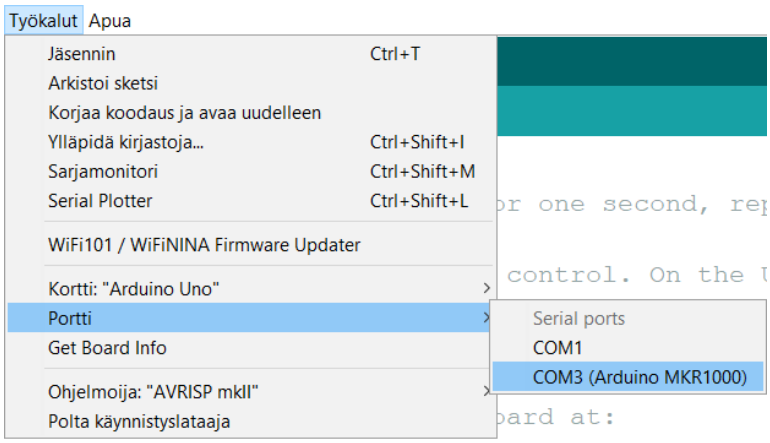
Koodi näytti tältä:



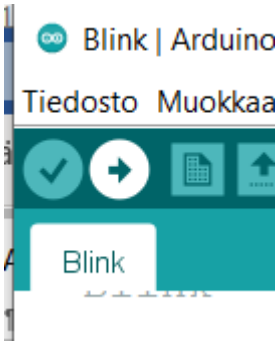
Sitten valitsimme käyttämämme alustan:



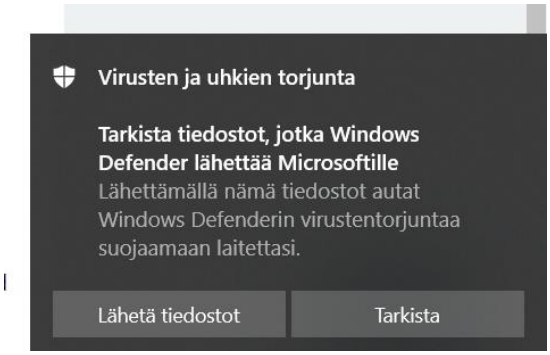
Sitten valitsimme portin, joka meidän tapauksessamme oli nyt *COM3*



Sitten vain latasimme esimerkkiohjelman **Arduinoomme** lataa-napilla.



Samalla piti **Windows Defenderille** antaa lupa lähetykseen:



Ja nyt meidän esimerkkiohjelmamme oli asennettuna **Arduinoon** ja led vilkkui, joten testi onnistui.

5. Saunasensorin sarjaporttiyhteys ja vb.netin Visual Basic

Olimme aiemmin käyttäneet ohjelmointiin **vb.netin Visual Basicia**, joten päätimme selvittää olisiko se mahdollista myös **Arduinon** kanssa ja myös langattomasti ilman yhteyttä tietokoneeseen.

Yhteys vb.net Visual Basicilla (Visual Studio 2017) ←→ Arduino ja esimerkkiohjelman tutkailu

Tähän vaikutti helpoin ja paras konsti olevan **FirmataVB**:

http://www.acraigie.com/programming/firmatavb/

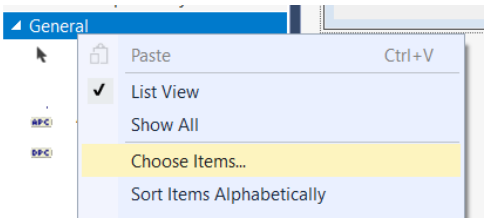
Se koostuu kolmesta dll-tiedostosta, jotka laitetaan projektikansioon.

Kopioimme esimerkkiprojektit ja lähdimme tutkailemaan ensimmäistä, eli *ArduinoFirmataVB:tä.*
Käynnistimme sen **Visual Studio 2017**:een (tämä on **Microsoftin** ohjelmankehitysympäristö ja se on käytettävissä ilmaiseksi).

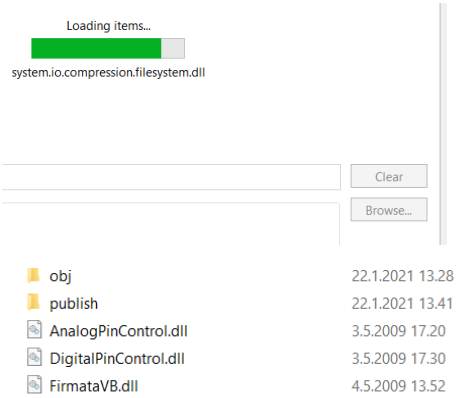
https://fi.wikipedia.org/wiki/Microsoft_Visual_Studio

https://visualstudio.microsoft.com/free-developer-offers/

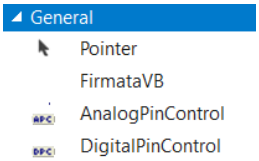
Tämän jälkeen kun halusimme käyttää niitä dll-tiedostoja, niin ensiksi ne lisätään työkaluihin ottamalla *Toolboxissa* hiiren toissijaisella halutussa valikossa ja valitsemalla *Choose items…*



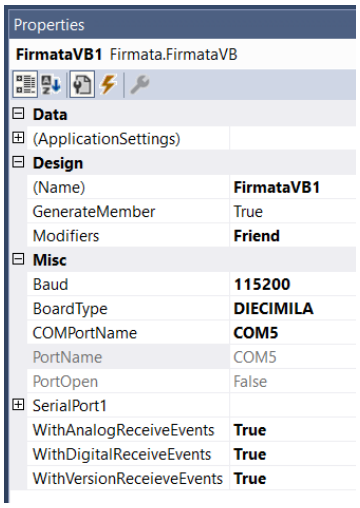
Sitten käydään selaamassa tiedostojen sijaintiin ja valitsemalla haluttu kohde *Browse…* -takaa:



Käydään kaikki kolme näin valitsemassa ja saadaan ne työkaluihin.

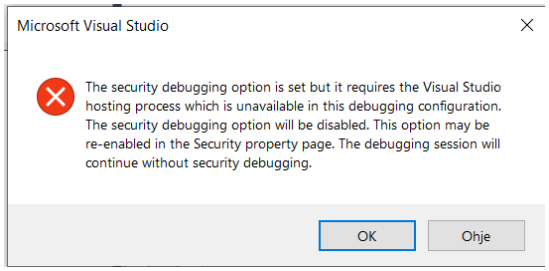


Seuraavaksi muutimme portin arvon esimerkkiohjelman lomakkeelle laitetusta komponentista *FirmataVB1* klikkaamalla sitä hiirellä ja vaihtamalla *COMPortNamen COM4*:sta *COM5*:ksi, koska laitteemmehan on kiinni viitosportissa.



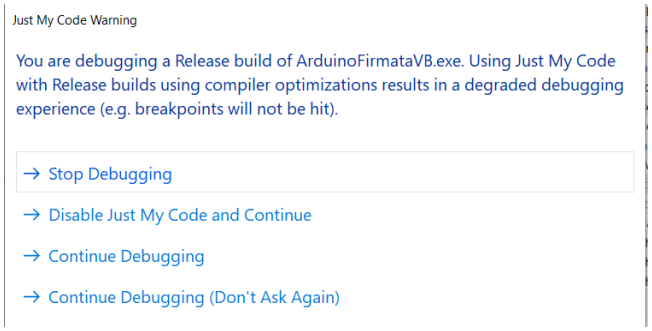
~~Tämän jälkeen käynnistimme ohjelman *Visual Studiosta* käsin *Start*-napilla.~~

~~Saimme virheilmoituksen, mutta emme antaneet sen häiritä:~~



~~Klikkasimme vain *OK*~~

~~Saimme seuraavan valintaikkunan:~~



~~Valitsimme *Disable Just My Code and Continue*~~

Tämä oli umpikuja. Syynä varmaankin se, että projekti on päättynyt jo 2011 ja **MKR1000** on paljon sitä uudempi malli. **Firmatan** kuviosta lisää lopussa liitteessä kaksi.

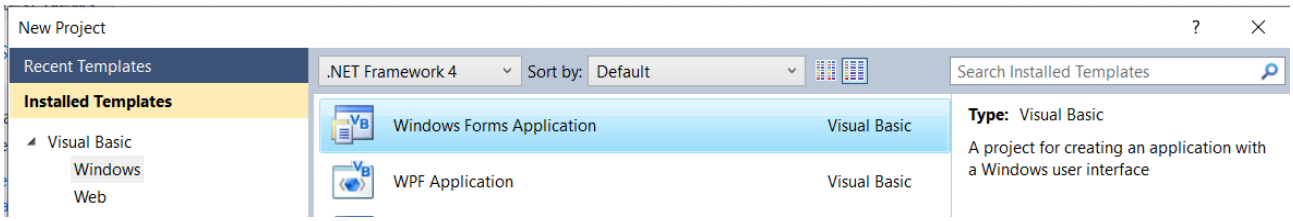
Seuraavaksi lähdimme koettamaan näillä ohjeilla:

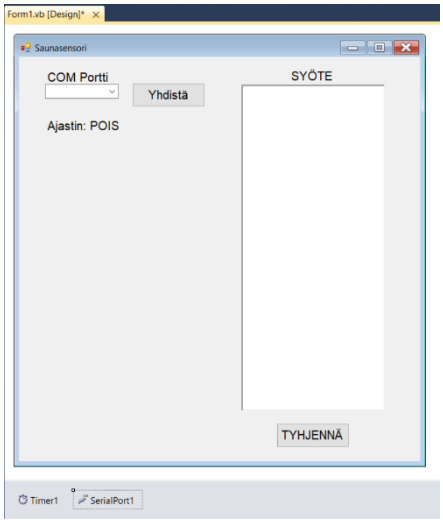
http://www.martyncurrey.com/arduino-and-visual-basic-part-1-receiving-data-from-the-arduino/

Tämäkin vaikutti toimivalta. Toisaalta tämän tekniikan käyttö oli paljon työläämpää, mutta toisaalta ei tarvittu lisäkirjastoja ja ratkaisu oli paljon yleispätevämpi.

Päätimme seurata aluksi tiukasti ohjeita, emmekä ladanneet valmista projektia, vaikka sekin olisi ollut ladattavissa. Halusimme päästä kunnolla jyvälle tästä ratkaisusta.

Loimme **Visual Studiolla** projektin nimellä **ArduinoVB** ja siihen lomakkeen ja tarvittavat kontrollit. Käytimme **Visual Studio 2010 Ultimatea**, joka oli ollut käytössä jo aiemmissa projekteissa.





Esimerkistä muokkaamamme koodi näytti tältä:

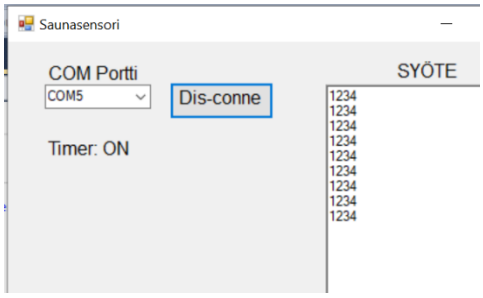
Form1.vb* | Form1.vb [Design]*

Saunasensori | (Declarations)

```
Imports System
Imports System.IO.Ports
Public Class Saunasensori
    Dim comPORT As String
    Dim receivedData As String = ""

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Enabled = False
        comPORT = ""
        For Each sp As String In My.Computer.Ports.SerialPortNames
            CBComPortti.Items.Add(sp)
        Next
    End Sub

    Private Sub CBComPortti_SelectedIndexChanged(sender As Object, e As EventArgs) Handles CBComPortti.SelectedIndexChanged
        If (CBComPortti.SelectedItem <> "") Then
            comPORT = CBComPortti.SelectedItem
        End If
    End Sub

    Private Sub ButtonYhdistä_Click(sender As Object, e As EventArgs) Handles ButtonYhdistä.Click
        If (ButtonYhdistä.Text = "Connect") Then
            If (comPORT <> "") Then
                SerialPort1.Close()
                SerialPort1.PortName = comPORT
                SerialPort1.BaudRate = 9600
                SerialPort1.DataBits = 8
                SerialPort1.Parity = Parity.None
                SerialPort1.StopBits = StopBits.One
                SerialPort1.Handshake = Handshake.None
                SerialPort1.Encoding = System.Text.Encoding.Default
                SerialPort1.ReadTimeout = 10000

                SerialPort1.Open()
                ButtonYhdistä.Text = "Dis-connect"
                Timer1.Enabled = True
                LabelAjastin.Text = "Timer: ON"
            Else
                MsgBox("Select a COM port first")
            End If
        Else
            SerialPort1.Close()
            ButtonYhdistä.Text = "Connect"
            Timer1.Enabled = False
            LabelAjastin.Text = "Timer: OFF"
        End If

    End Sub


    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        receivedData = ReceiveSerialData()
        RTBInput.Text &= receivedData
    End Sub

    Function ReceiveSerialData() As String
        Dim Incoming As String
        Try
            Incoming = SerialPort1.ReadExisting()
            If Incoming Is Nothing Then
                Return "nothing" & vbCrLf
            Else
                Return Incoming
            End If
        Catch ex As TimeoutException
            Return "Error: Serial Port read timed out."
        End Try

    End Function


    Private Sub clear_BTN_Click(sender As Object, e As EventArgs) Handles ButtonClear.Click
        RTBInput.Text = ""
    End Sub
End Class
```

Sitten laadimme sketsin esimerkin mukaisesti:

Lähetimme sen **Arduinoon** ja ajoimme ohjelman tämän jälkeen **Visual Studiosta.**

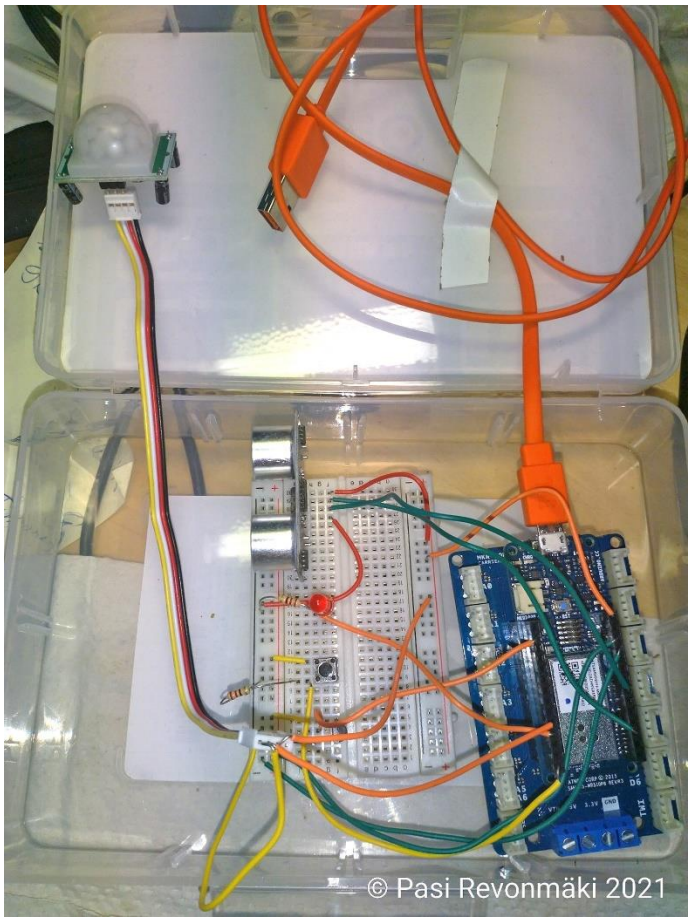


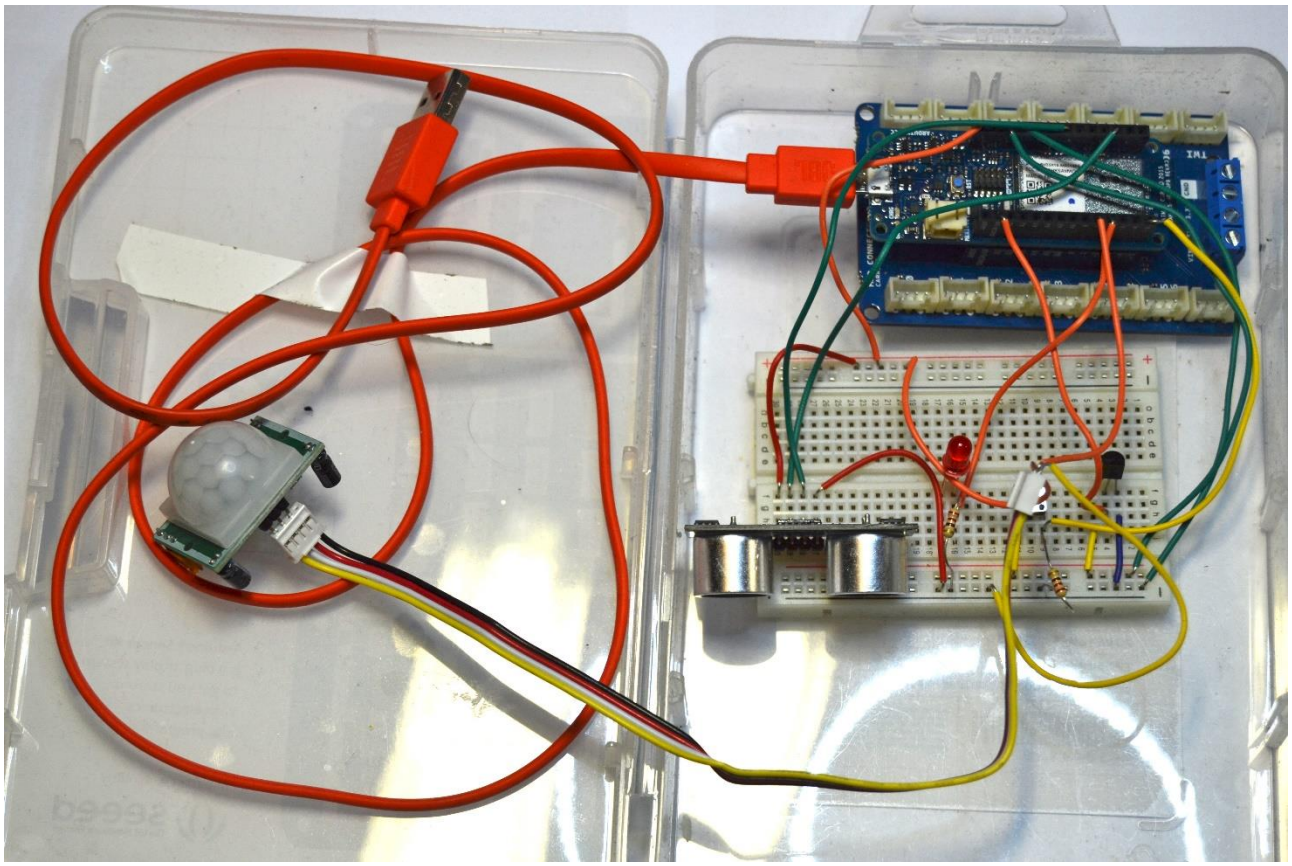
Toimii!

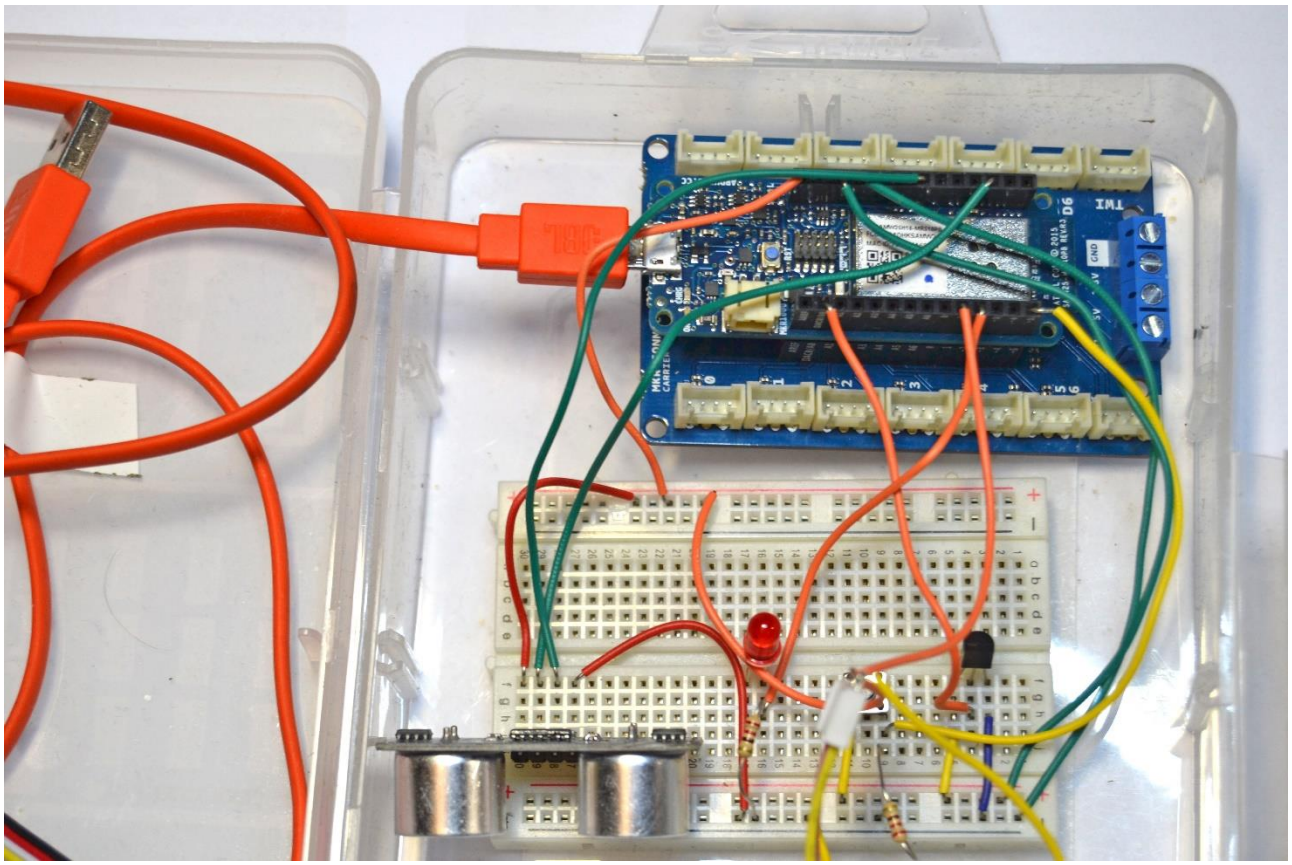
Nyt siis saimme ensimmäisen kerran onnistumaan tiedonsiirron **Arduinon** ja **Visual Studion** (**vb.net**) välillä. Eli toiveita oli, että pääsisimme koodaamaan PC:ltä ajettavan ohjelman **vb.netillä**.

6. Sensoreiden kytkeminen koekytkentäalustalle

Tämän jälkeen kokeilimme useita kokoonpanoja ja lopullinen testaustilanteemme (kun hankimme lisäksi liiketunnistimen, eli *PIR*rin) näytti sitten tältä (**Arduinon** alla oleva laajennusosa on vain fyysistä vakauttamista varten):


© Pasi Revonmäki 2021





Lämpötila-anturin **TMP35** kytkentäkaavion ja koodia **Arduino MKR1000:een** löysimme täältä:

https://www.hackster.io/foxlab/temperature-monitoring-with-arduino-mkr1000-and-artik-cloud-84c351

Ultraäänisensorin **HC-SR04:n** kytkentäkaavion ja koodia **Arduino MKR1000:een** löysimme täältä:

https://create.arduino.cc/projecthub/jat/drs-oil-container-ebc7a7

Perästä tilasimme vielä liiketunnistimeksi **HC-SR501**, siitä myöhemmin tarkemmin.

Meillä oli lopulta käytössä siis:

- Koekytkentäalusta
- **Arduino MKR1000** mikrokontrolleri
- USB-kaapelilla 5 V tulovirta
- 16 x hyppyjohto
- 4 johto kaapeli (3 käytössä)
- **HC-SR04-ultraäänisensori**
- Painokytkin (nappi)
- Punainen led
- **TMP36** lämpötilasensori
- **HC-SR501** liiketunnistin
- 2 x 10 kΩ vastus
- 5 V virtalähde (USB:n kautta)

Ultraäänisensorin pinnit 1 maa, 2 echo, 3 trig ja 4 VCC (virta 5V). Vastaavat pinnit **Arduinossa** 1 → GND, 2 → *12*, 3 → *8* ja 4 → *5V*

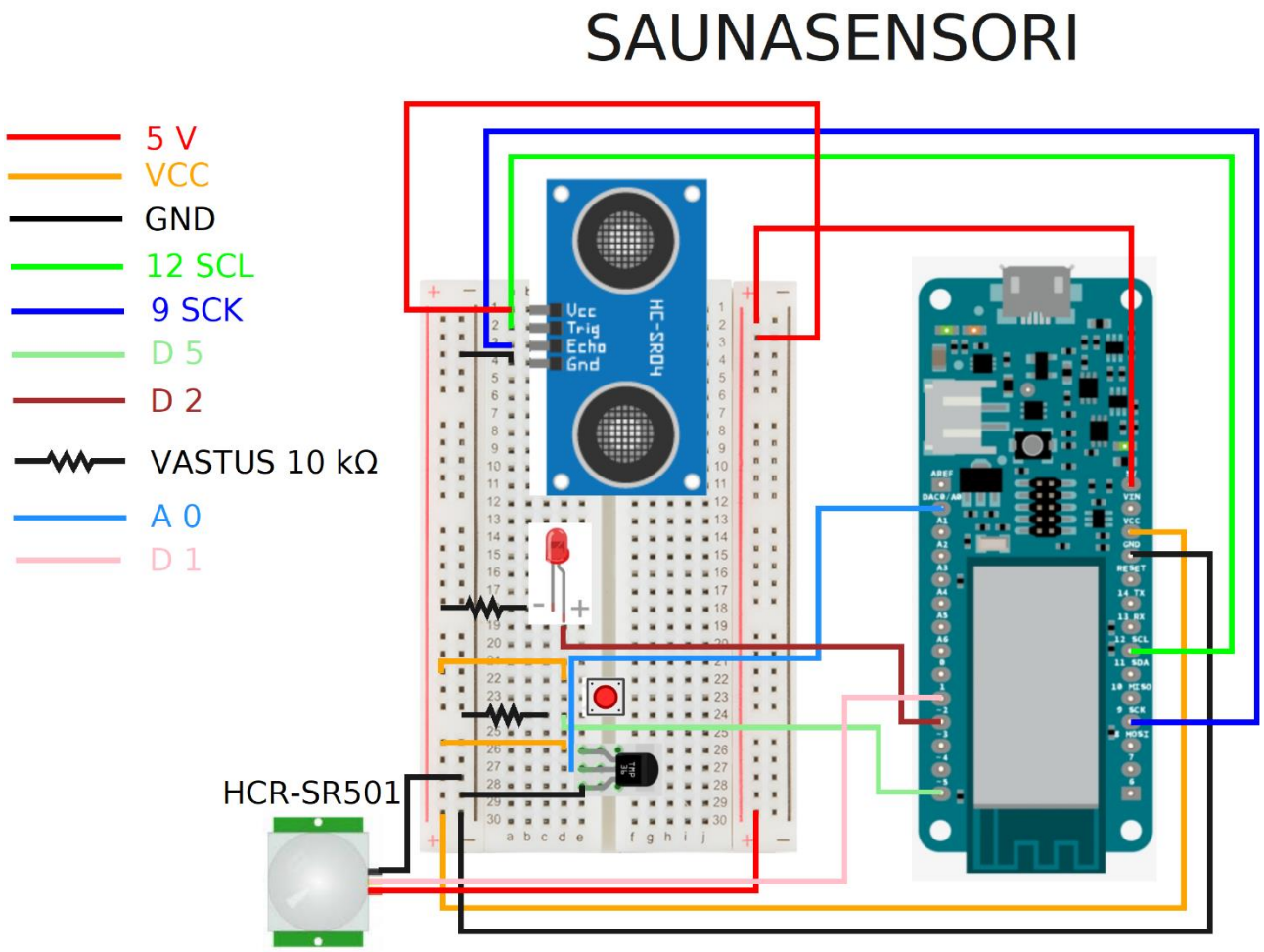
Ledin maa tuli **Arduinon** maasta, mutta välissä oli 10 k_Ω vastus ja sen virta tuli **Arduinon** pinnistä *2*

Painokytkimen (napin) maa tuli **Arduinon** maasta, mutta välissä oli 10 k_Ω vastus ja sen virta tuli **Arduinon** pinnistä *5*.

Lämpötilasensorin pinni 1 oli maa, pinni 2 oli lämpötilalukemapinni ja 3 oli virta. Vastaavat pinnit **Arduinossa** 1 → GND, 2 → *A0*, 3 → *VCC*, eli **MKR1000**:ssa 3,3 V.

Liiketunnistimen pinni 1 oli maa, pinni 2 oli triggerpinni ja 3 oli virta. Vastaavat pinnit **Arduinossa** 1 → GND, 2 → *1*, 3 → *5 V*

Kaaviona näin:



Näitä hyödyntäen saimme kytkettyä kaikki **Saunasensorin** vaatimat sensorit kytkentäalustalle. Kytkennöillä ei ollut tässä muuta funktiota kuin todistaa että ratkaisu toimi ja koodi ajoi asiansa. Virtansa laite saa USB:n kautta ja käyttää verkkovirtamuuntajaa.

Virrankulutus

MKR1000 120 mA (https://www.arduino.cc/en/Tutorial/MKR1000BatteryLife)

HC-SR04 6 mA (https://forum.arduino.cc/index.php?topic=216079.0)

Led 2,85 mA jatkuva (https://maker.pro/arduino/tutorial/how-to-reduce-arduino-power-consumption)

TMP36 0,005 mA (https://learn.adafruit.com/tmp36-temperature-sensor)

HC-SR501 3 mA (https://forum.arduino.cc/index.php?topic=493366.0)

Yhteensä siis **132 mA**. Eli jatkuvassa 24/7 käytössä ja käyttötarkoituksessa verkkovirtavaatimus on itsestään selvä.

7. Sensoreiden koodaaminen, painonappi ja lämpötila-anturi

Hyödynsimme tässä dokumentissa aiemmin tehtyä koodia, joten **Arduinon** puolelta poistimme koodista potentiometrin, koska sitä ei meillä ole tässä käytössä ja korjasimme painikkeen porttinumeron neljästä viideksi.

PC:n puolella lisäsimme lomakkeeseen puuttuvat kontrollit ja nimesimme ne toistaiseksi samoin kuin esimerkissä. Koodin puolella poistimme potentiometriosuudet ja korjasimme kontrollien nimet tarvittavin osin. Käänsimme osan teksteistä.

Sitten ajoimme **Arduinon** koodin paikalleen ja starttasimme **Visual Studiosta** ohjelman.

https://youtu.be/xsy-3bpGm9A

Nyt kulki siis jo painonapin data ja lämpötila-anturin tieto.

Sensoreiden koodaaminen, led

Seuraavaksi piti saada tietoa siirtymään **vb.netillä** tehdystä ohjelmasta ledille. Hyödynsimme seuraavia ohjeita:

http://www.martyncurrey.com/arduino-and-visual-basic-part-3-controlling-an-arduino/

Lisäsimme ledin koodia, sekä **Arduinoon** että **vb.netiin**

Sensoreiden koodaaminen, ultraäänisensori

Käyttämämme ultraäänisensorin kytkentä ja koodi **MKR1000:lle** kävi ilmi täältä:

https://www.mathworks.com/help/thingspeak/prototyping-with-sonar-proximity-sensor.html

Lähdimme lisäilemään **Arduinon** puolelle sensorin tarvitsemaa koodia.

Laitoimme ensiksi alkumääritykset:

```
//ultraäänisensori
const int trigPin = 8;
const int echoPin = 12;
long duration;
int distance;
unsigned int oldDist = 0;
unsigned int newDist = 0;
long lastUpdateTimeU = 0;
int points = 7;
float distanceThreshold = 0;
const unsigned long postingInterval = 60L * 1000L;   // Dataa pilveen kerran minuutissa
const unsigned long sporttiUInterval = 1L * 1000L;   // Dataa sarjaporttiin kerran sekunnissa
//
```

Pinnien moodit asetuksiin:

```
void setup()

{
//ultraäänisensori


    pinMode(trigPin, OUTPUT); // Asettaa triggerpinnin moodiksi lähtö
    pinMode(echoPin, INPUT); // Asettaa echopinnn moodiksi paluu
  // UÄ LOPPU
```

Sitten suoritukseen luuppiin.

```
void loop()
{
 //ultraäänisensori
float  distance=0;
    for (uint16_t loops = 0; loops < points; loops++){
     distance += getDistance(trigPin,echoPin);  //mittailee, tallentaa mittausten summan
     delay(5);
    }
    distance = distance/points;
   if (millis() - lastUpdateTimeU >=  sporttiUInterval) {
     lastUpdateTimeU = millis();
     Serial.println("Etaisyys: "+ String(distance)+ " cm");
     }
   delay(10);    // Tauko mittausten välillä. Pienensin tätä, oli 500 ms alkujaan
   //UÄ LOPPU
```

Tarvittava funktio etäisyyden laskemiseksi:

```
 float getDistance(int tPin,int ePin) //UÄ-sensorille
 {

  long duration, distance;

  digitalWrite(tPin, LOW);  // Resetoi pinnin.
  delayMicroseconds(2);
  digitalWrite(tPin, HIGH);  // Aloittaa mittauksen.
  delayMicroseconds(10); //
  digitalWrite(tPin, LOW);   // Lopettaa pulssin.
  duration = pulseIn(ePin, HIGH);  // Odottaa heijastepulssia.
  distance = (duration/2) / 29.1;     // Laskee etäisyyden käyttäen arvioitua äänennopeutta.

  // Virheenetsintää tarvittaessa sensorille.
/*
  if (distance >= 200 || distance <= 0){
    Serial.println("Out of range");
  }
  else {
   Serial.print(distance);
   Serial.println(" cm");
  }
  */
  return distance;
}
 //UÄ loppu
```

Sensoreiden testaaminen

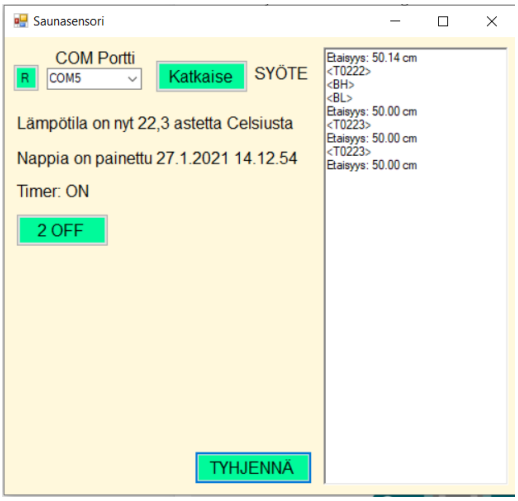
Tässä vastusteli taas ja otti aikansa selvittää missä kolmesta ongelmat piilivät, eli **Arduinossa**, **vb.netin** koodissa vai kytkennöissä. Jossain vaiheessa kaikissa näissä kolmessa oli säädettävää. Esimerkiksi alkuun UÄ-sensori näytti nollaa:

Distance: 0
Distance: 0
Distance: 0
Distance: 0
<BH>
Distance: 0
<BL>
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
Distance: 0
<T0225>
Distance: 0
Distance: 0

Tuo korjaantui sensorin liitosten muuttamisella.

Pääongelma oli saada ajastukset ja eri datojen lähetysfrekvenssit kohdalleen koodiin. Jos näytteitä lähetettiin liian herkästi, niin painonapin triggeröinti ei välittynyt sarjaliitäntään, vaan hukkui lämpödatan alle ja etäisyystieto jäi lämpödatan alle ja toisinpäin.

Lopulta saimme kuitenkin kokonaisuuden toimimaan halutusti ja lisäilimme ohjelman *GUI:hin* **Visual Studiolla** ja **vb.netin** koodilla toimintoja ja siistimme hieman ulkoasua. Ajossa näytti tältä:



Videolla homma näytti sitten tältä:

https://youtu.be/2Z2GcAjhZu4

8. Liiketunnistimen lisääminen, Arduinon pään osuus

Jos raportissa tuntuu olevan hieman epäjatkuvuutta, niin se johtuu siitä, että jouduimme hankkimaan lisää **Arduino**-tavaraa kesken projektin. Mm. tämä liiketunnistin saapui vasta siinä vaiheessa, kun langaton puoli oli jo pitkälti valmis. Mutta asiaan.

Vinkkiä liiketunnistimemme käytöstä **Arduinolle** saimme täältä:

https://www.makerguides.com/hc-sr501-arduino-tutorial/

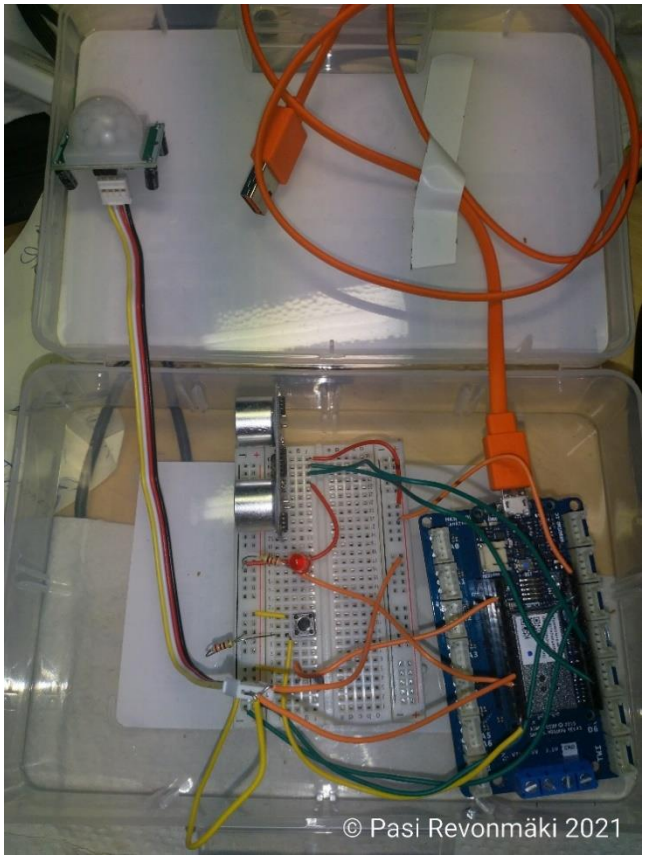
Liiketunnistimen kytkentä

Meillä oli siis käytettävissämme liiketunnistin *HC-SR501*

https://www.spelektroniikka.fi/p23758-liiketunnistin-hc-sr501-pir-moduuli-arduino-sovelluksiin-jne--fi.html

Siinä oli itsessään pinnit, joten emme voineet kytkeä sitä suoraan hyppylangoilla. Samoin siinä itsessään oli fyysiset säätimet (potentiometrit) herkkyydelle ja ajastukselle. Otimme käyttöön yhden nelijohtoisen kaapelin, jossa oli liittimet molemmissa päissä. Tarvitsimme tosin vain kolme johtoa. Ohjeen kytkentä oli **UNOlle**, mutta logiikkahan oli sama. Kytkeä 5 V, maa ja digidatapinni.

Kytkimme virran viiteen voltin ulostuloon **Arduinossa**, maan maahan ja digipinnin **Arduinon** ykkösdigipinniin, kun se sattui olemaan digipinneistä vapaana **Arduinossa** . Koko hökötys näytti siis kertauksen vuoksi nyt tältä (*PIR* ylhäällä vasemmalla).



Huomatkaa että vedimme koekytkentälevylle 5 V toisen puolen plussaan, koska nyt meillä oli jo kaksi sensoria (UÄ ja *PIR*), jotka olivat 5 V vailla.

Liiketunnistimen tarvitseman koodin lisääminen Arduinoon

Määrityksiin laitoimme näin:

```
//Liiketunnistin
#define pirPin 1
#define ledPin 2
int val = 0;
bool motionState = false; // Starttaamme liiketunnistin levossa
//Liiketunnistin loppu
```

Eli digidatapinni siis se numero yksi ja meidän ledimmehän oli ennestään digidatapinnissä kaksi.

Asetuksiin laitoimme näin:

```
  //Liiketunnistin
  // Datapinni määritellään sisääntuloksi ja ledipinni määritellään ulostuloksi
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);
  //Liiketunnistin loppu
```

Luuppiin laitoimme näin:

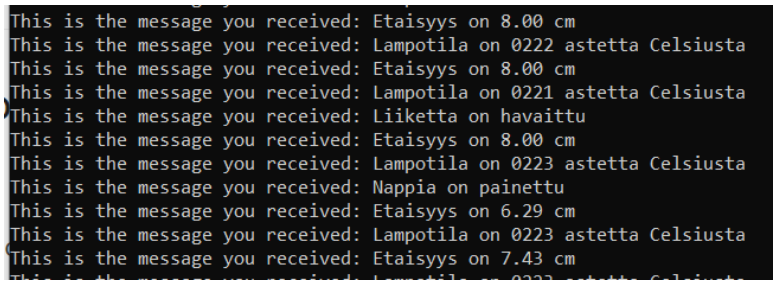
```
//Liiketunnistin
  // Lukee pirPin arvon ja tallentaa sen muuttujaan val:
val = digitalRead(pirPin);
// Jos liikettä, niin (pirPin = HIGH) ja tehdään seuraavaa:
if (val == HIGH) {
  digitalWrite(ledPin, HIGH); // Led päälle.
  // Laitetaan liiketunnistin päälletilaan
  if (motionState == false) {
    Serial.println("Liiketta on havaittu ");
    motionState = true;
    //  UDP-osuus
      strcpy(packetBuffer,"Liiketta on havaittu ");
      Udp.beginPacket(remoteIP,remotePort);
      Udp.write(packetBuffer,strlen(packetBuffer));
      Udp.endPacket();
  }
}
// Jos liikettä ei ole havaittu, niin (pirPin = LOW) ja tehdään seuraavaa:
else {
  digitalWrite(ledPin, LOW); // Led sammutetaan.
  // Ei liikettä:
  if (motionState == true) {
    Serial.println("Liike loppui ");
    motionState = false;
  }
}
//Liiketunnistin loppu
```

Ja kuten koodista näkyy, niin laitoimme sinne jo sarjaportin määritysten lisäksi myös *UDP*-osuuden. Mallia oli helppo katsoa nappiosuudesta. Ja tarkkasilmäisimmät huomaavat, että olimme tässä välissä myös siistineet koodia ja nyt vastapään IP ja portti määritellään muuttujalla ainoastaan kerran määrityksissä ja luupissa käytetään muuttujia *remoteIP* ja *remotePort*.

```
IPAddress remoteIP = IPAddress(192,168,1,10);
int remotePort = 2390;
```

Ja vaikka määrittelimme IP:n ja portin heti aluksi, niin luupissa oli yhä mahdollista muuttaa niitä. Tämä tapahtuisi simppelisti lähettämällä mikä tahansa teksti **Arduinoon** *UDP*:llä, eli sillä jo aiemmin **vb.netillä/Visual Studiolla** luomallamme *GUI*:lla. Siitä **Arduino** nappaa lähettäjän IP:n ja portin ja käyttää jatkossa niitä (niin kauan kuin **Arduino** on virroissa, eikä sille anneta uutta koodia).

Latasimme uuden koodin **Arduinoon** ja nyt vastaanottopäämme näytti tältä, kun teimme liikettä ja painoimme nappia.



Pelaa hienosti!

Seuraavassa tarkempi selostus langattomasta yhteydestä, *UDP*:stä, koodaamisesta **vb.netillä/Visual Studiolla**.

9. Langaton yhteys Arduino ←→ PC:llä oleva Visual Studio

Lähettäminen *UDP*:lla Arduinoon

Dokumentaatiota yhteydelle löysimme täältä:

https://www.open-electronics.org/lets-program-arduino-with-microsoft-visual-studio-2017/

Arduinon esimerkki auttoi alkuun **Arduinon** päässä ja **vb.netille** tsekkasimme täältä:

https://www.codeproject.com/Articles/8877/UDP-Send-and-Receive-using-threads-in-VB-NET

Ensimmäinen tehtävä oli lisätä kirjastot projektiin:

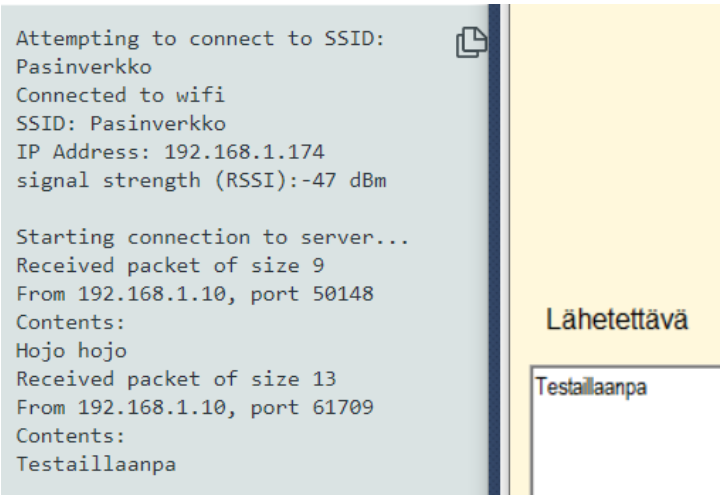
Imports System.Net.Sockets

Imports System.Net

```
Imports System.Net
Imports System.Net.Sockets
```

Määrittelimme Arduinon käyttämän portin ja IP:n. Sisäverkon IP:n osalta kannattaa tässä kohden mainita, että tuo IP kannattaa määritellä reitittimeen kiinteäksi *MAC*-osoituksella (ja mieluusti käytetyn sarjan loppupäästä), jolloin yhteys ei katoa reitittimen jakaman IP-osoitteen *leasetimen* tullessa vastaan ja osoitteen mahdollisesti muuttuessa.

Sitten laitoimme esimerkistä koodia ja **Web Editorin** sarjaportin monitoroinnissa ja *GUI*ssamme näytti nyt tältä:



Lähetys siis toimi PC:ltä **Arduinolle**.

Vastaanotto *UDP*:lla Arduinosta

Arduinon puoli

Tämä paljastuikin sitten olevan astetta hankalampi rasti. Harhailimme aikamme ja huomasimme, että oikotietä ei tälläkään kertaa ollut, vaan ongelma on ratkottava pala kerrallaan ja nollasta. Projektin ongelmana oli, että siinä oli kaksi erillistä liikkuvaa osaa ja kun koetti ratkoa molempia kerralla, niin ei tiennyt missä kohtaa mikäkin tökki. Eli oli **Arduino** ja sen koodi ja PC/**vb.net** ja sen koodi.

Löysimme esimerkkiprojektin, josta saimme *UDP*-kuuntelijan PC:n **vb.netille**.

https://www.codeguru.com/columns/vb/communication-using-udp-and-visual-basic.html

Näin saimme eliminoitua PC-puolen ongelmat tässä vaiheessa. Ainoat muutokset koodiin teimme määrittelemällä socketin.

Arduinon lähettäjäesimerkin saimme täältä:

https://forum.arduino.cc/index.php?topic=176669.0

Siellä oleellista tässä vaiheessa oli tämä:

It should go something like this if the UDP listener is 192.168.0.2 on port 8888.

Code: [Select]

```
IPAddress remoteIP(192,168,0,2);
int remotePort = 8888;

strcpy(packetBuffer,"This is a test");

Udp.beginPacket(remoteIP,remotePort);
Udp.write(packetBuffer,strlen(packetBuffer));
Udp.endPacket();
```

Lisäsimme tuon koodinpätkän **Arduinon** luuppiin samaan suoritukseen UÄ-sensorin etäisyysilmoituksen kanssa, niin saimme helposti toistoa esimerkillemme. Kuitenkin niin että vaihdoimme näytettävän tekstin, PC:mme IP:n ja portin eli socketin.

Starttasimme ensin kuuntelun PC:ssä ja sitten tuuppasimme muunnetun koodin **Arduinoon**.

Nyt kuuntelija alkoi suoltaa tällaista:

```
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
This is the message you received: T??m??n l??het??mme Arduinosta
```

Hyvä! Se toimii, tosin merkistö on pielessä *UTF8* vs. *ANSI* varmaan, mutta se ei ole tässä oleellista.

Seuraavaksi ajattelimme koettaa saada **Arduino** lähettämään ne etäisyys-, lämpötila- ja painiketiedot *UDP*:lla.

Etäisyystieto

Ja kun parin tunnin päästä saimme ensi kertaa jotain tolkullista etäisyysdataa *UDP*:lla näytölle tuntui se varsinaiselta työvoitolta. Tässä lyhyt selostus tämän ongelman ratkonnasta.

Huomasimme heti, ettei homma ole lähellekään yhtä simppeli kuin sarjaportin kanssa. Siinähän riitti, kun laittoi komennolla *println* aiemmin saatu desimaalinen etäisyysarvo (muuttujassa) ja halutut tekstit liitettynä menemään tyyliin:

Serial.println("Etaisyys: "+ String(distance)+ " cm");

Suoraviivaista ja selkeää. Mutta *UDP*:n kanssa, ehei. Saimme siis menemään tekstiä lainausmerkkien sisältä, mutta tuo sarjaporttityyli ei toimi.

Loimme ensin taulukon ***packetBuffer2[],*** johon meillä oli tarkoitus saada se etäisyystieto, vaikka sitten tekstinä. Saimme muutaman sata eri virheilmoitusta yhtä monella yrittämällä, koska **C++:n** logiikka poikkeaa täysin **vb.netin** vastaavasta. Mikään konsti ei tuntunut purevan. Mutta lopulta löysimme yrityksen ja erehdyksen kautta varsin lupaavan esimerkin:

https://arduino.stackexchange.com/questions/26832/how-do-i-convert-a-float-into-char

*There is a function in the standard **Arduino** library called **dtostrf()**. I think of it as "Decimal to String Float". You pass in the float, how wide you want the whole number to be (if it will fit), the number of decimals of precision - and the buffer you want it to fill.*

Ei kun kokeilemaan. Vaan ei, ilmoitti ettei tunne koko komentoa. Piti siis löytää vielä komennolle kirjasto. **Google** kertoili parin yrityksen jälkeen tällaista:

apparently dtostrf is available only on avr, I've added a function to emulate it, it will be available on the next IDE release by including <avr/dtostrf.h>.

Joten lisäsimme tuon kirjaston *<avr/dtostrf.h>* ja johan alkoi lyyti kirjoittamaan.

Laitoimme koodiin siis rivin:

```
dtostrf(distance,4,0,packetBuffer2);
```

Distance on se aiemman koodin antama desimaalilukema etäisyydelle.

Sitten data piti saada *packetBufferiin*. Ja tämäkin rassasi tovin aikaa. Löytyi tällainen uusi hieno komento kuin *strcpy()*

https://stackoverflow.com/questions/15286014/segmentation-fault-strcpy-c

Sen avulla saimme taas luotua rivin lisää koodia:

```
strcpy(packetBuffer, packetBuffer2);
```

Ja lopulta itse asiaan:

```
Udp.beginPacket(remoteIP,remotePort);
Udp.write(packetBuffer,strlen(packetBuffer2));
Udp.endPacket();
```

Ja nyt!

```
This is the message you received:   58
This is the message you received:   59
This is the message you received:   58
This is the message you received:   58
This is the message you received:   62
This is the message you received:   58
This is the message you received:   58
This is the message you received:   64
```

Teksti tuli sieltä **vb.netin** koodista, mutta lukema etäisyysanturilta **Arduinosta**, jes.

Etäisyys sai olla oikeastaan tarkempikin lukema, joten lisäsimme desimaalit:

```
dtostrf(distance,4,2,packetBuffer2);
```

Kerralla nappiin, oho:

```
This is the message you received:   58
This is the message you received:   61
This is the message you received: 61.29
This is the message you received: 57.29
This is the message you received: 57.57
```

Tuo tulisi tarvitsemaan tunnisteita, jotta ei mene lämpötilan ja painikkeen kanssa sekaisin. Mutta katsomme sen myöhemmin.

Lämpötilatieto

Nyt meillä oli valmis sapluuna, joten lämpötilatiedon saaminen langattomasti ei oletuksemme mukaan pitänyt olla enää iso ongelma.

Lähdimme siis luomaan **Arduinoon koodia** luuppiin lämpötilasensorin kohdalle.

Ihan pikkuisen harha-askeleen jälkeen saimme koodin toimimaan. Meitä hämäsi, että tilanne oli nyt helpompi, muunnoksia ei tarvittu!

Laitoimme lämpötila-anturin sarjaporttikoodissa lasketun *numberString*-merkkitaulukon karvoineen kaikkineen suoraan lähetykseen ja toimi laakista.

```
Udp.beginPacket(remoteIP,remotePort);
Udp.write(packetBuffer,strlen(numberString));
Udp.endPacket();
```

PC:lle tuli kamaa nyt näin:

```
This is the message you received: 57.00
This is the message you received: 0222
This is the message you received: 66.86
This is the message you received: 0221
This is the message you received: 67.71
This is the message you received: 0221
This is the message you received: 67.14
This is the message you received: 0220
This is the message you received: 67.57
This is the message you received: 0222
```

Eli joka toinen rivi etäisyyttä ja toinen lämpötilaa. Noille arvoille pitää tosiaan kehitellä fiksausta jompaankumpaan päähän, parempi olisi ehdottomasti **Arduinon** pää.

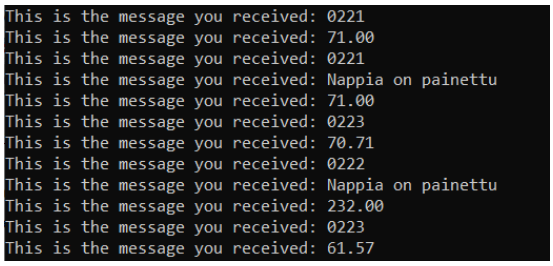
Painonapin tila

Sitten vielä napin painallus. Lähdimme tsekkaamaan sen koodin ja kuinka siihen saadaan *UDP*-komponentti mukaan.

Tämä oli helppo, meillehän riitti, että lähetetään ennalta valittu teksti, kun trigger laukeaa ja tämä triggerhän oli valmiina sarjaporttiosuuden koodissa, joten lisäsimme sen sinne, esimerkkikoodihan meillä oli jo alkujaan:

```
if (newButtonSwitchState != oldButtonSwitchState)
  {
    oldButtonSwitchState = newButtonSwitchState;
    if (oldButtonSwitchState == true)
    {
      Serial.print("<BH>");
      IPAddress remoteIP(192,168,1,10);
      int remotePort = 2390;
   strcpy(packetBuffer,"Nappia on painettu ");
   Udp.beginPacket(remoteIP,remotePort);
   Udp.write(packetBuffer,strlen(packetBuffer));
   Udp.endPacket();
    }
    else
    {
      Serial.print("<BL>");
    }
    if (DEBUG) { Serial.println("");  }
  }
```

Ja sitten testaamaan, näytti tältä:



Pelasi kerrasta, huolestuttavaa.

Yhteenvetoa tilanteesta. Nyt toimivat siis kaikkien sensoreiden lähetys **Arduinosta** langattomasti lähiverkossa PC:lle ja lukemat saatiin **vb.netillä** tehdyllä ohjelmalla myös näkymään. Jo aiemmin oli saatu lähetettyä tekstimuotoista dataa langattomasti Arduinoon.

Arduinon pään lähettämän datan muodon virittäminen

Saimme nätisti lisättyä koodia, jotta ulostuonti saatiin tolkulliseksi. Lämpötilalukema oli muodoltaan hassu, mutta se olisi helposti korjattavissa PC:n/**vb.netin** päässä.

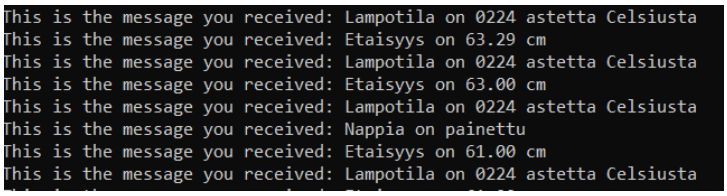
Ultraäänisensori:

```
//UDP-osuus

    dtostrf(distance,4,2,packetBuffer2);

    IPAddress remoteIP(192,168,1,10);
    int remotePort = 2390;
    Udp.beginPacket(remoteIP,remotePort);
    strcpy(packetBuffer,"Etaisyys on ");
    Udp.write(packetBuffer,12);
    strcpy(packetBuffer, packetBuffer2);
    Udp.write(packetBuffer,strlen(packetBuffer2));
    strcpy(packetBuffer," cm");
    Udp.write(packetBuffer,3);
    Udp.endPacket();
```

Lämpötilasensori:

```
//UDP-osuus

    IPAddress remoteIP(192,168,1,10);
    int remotePort = 2390;
    Udp.beginPacket(remoteIP,remotePort);
    strcpy(packetBuffer, "Lampotila on ");
    Udp.write(packetBuffer,13);
    strcpy(packetBuffer, numberString);
    Udp.write(packetBuffer,strlen(numberString));
    strcpy(packetBuffer," astetta Celsiusta");
    Udp.write(packetBuffer,18);
    Udp.endPacket();
```

Ja kuuntelijan päässä näkymä nyt:



10. Tavoitteet tähän saakka

Nyt olimme saavuttaneet taas yhden tavoitteen, eli **Arduinon** pää oli periaatteessa valmis. Se teki seuraavat asiat:

- mittasi etäisyyksiä ja reagoi, eli käytännössä huomasi, jos joku ylitti tietyn linjan (UÄ-sensori)
- reagoi ledin sytyttämisellä ja sammuttamisella haluttuihin asioihin (led)
- reagoi napin painamiseen (painonappi)
- mittasi lämpötiloja ja lähetti arvon eteenpäin (lämpötilatunnistin)
- valvoi liikettä ja reagoi siihen (*PIR*-sensori)
- lähetti datan sarjaportin kautta
- liittyi langattomaan lähiverkkoon suojatulla *WPA2*-salauksella
- lähetti kaikista näistä sensoreista informaation langattoman lähiverkon kautta *UDP*-protokollalla

Nämä olivat kaikki tarvittava tieto siihen, että saattaisimme tehdä loppuun käyttöliittymän **vb.netillä**, joka osaisi hälyttää ja toimia annettujen raja-arvojen puitteissa. Saunasensorimme oli nousemassa kuin Feeniks-lintu tuhkasta, vaikka välissä meinasi mennä jo epätoivon puolelle pään seinään lyömisessä.

Meille helpompi ja tutumpi osuus olisi nyt edessä. Ainoa oikeastaan uusi asia olisi saada ne *UDP*-viestit graafiselle puolelle esitettävään muotoon komentoikkunan sijaan.

11. *GUI*:n rakentamista Visual Studiolla ja vb.netillä (PC)

Sensorien lähettämän datan tuominen formiin tekstikenttään (*RichTextBox*)

Seuraavaksi piti saada **Arduinon** lähettämä *UDP*-liikenne näkymään formissa moduulista komennetun komentoikkunan (*console.write*) sijaan.

Tähän sai sitten etsiä hetken käypästä esimerkkiä. Sellainen löytyi lopulta saksankieliseltä sivulta. Vaikka saksa ei juurikaan taivu, niin koodi oli samaa tuttua. Esimerkki löytyi täältä:

https://foren.activevb.de/archiv/vb-net/thread-94837/beitrag-94841/Re-UDP-Client-Einfache-Variante/

Tämä koodi toimi omassa formissaan halutusti ja seuraavaksi implementoimme sen meidän olemassa olevaan **vb.net**-koodiimme.

Saimme **Arduinon** lähetteen näkymään formissa ok:

Arduinon koodin automaattiajamisongelma

Seuraavaksi huomasimme, että armas **Arduinommepa** ei alakaan kanssamme tanssiin, jos emme käy sitä aina erikseen kutsumassa. Eli jos tökkäämme USB:n vain virransaantiin (ilman sarjakaapeliyhteyttä), niin sketsi ei starttaakaan automaattisesti. Ja eipä se startannut automaattisesti, vaikka oli sarjakaapeliyhteydessäkin. Vasta kun starttasi sarjakaapeliyhteydessä **Web Editorin** monitoroinnin, alkoivat lukemat liikkua langattomasti.

Hetki miettimistä ja silmämääräistä **Arduinon** koodin tutkimista, niin löysimme todennäköisen paikan mikä ongelman aiheuttaisi. Eli kohta, jossa odoteltiin sitä sarjakaapeliyhteyden avautumista ennen verkkoon yhdistämistä. Laitoimme sen tässä vaiheessa kommenteille:

```
//  while (!Serial) {

 //   ; // wait for serial port to connect. Needed for native USB port only
 // }
```

Ja kas, nyt starttaa langattoman *UDP*-yhteyden ihan ilman kyyneleitä ja sarjakaapeliyhteyttä ja automaattisesti, kun virrat laitetaan **Arduinoon** päälle. Kuva sarjaporttiliikenteestä.

Lämpötilalukeman tuominen omaan tekstikenttään

Tämä asia oli meille uutta ja tahdoimme oppia sen esimerkkiä paremmin. Tiivistettynä selviteltävä asia oli tuoda taustalta *UDP*:lla **vb.netillä** vastaanotettu data näkyväksi kontrolliin *RichTextBox*. Komennot olisivat *Invoke* ja *AppendText*

Täältä löysimme logiikan:

https://riptutorial.com/vb-net/example/6235/performing-thread-safe-calls-using-control-invoke--

Vanhin käyttämämme **Visual Studio** oli 2010 versio, joten yläosan ohje tuli meille kyseeseen.

Teimme GUI:hin kontrollin eli tekstilaatikon (*TextBox*) lämpötilalukemalle ja sen koodin laitoimme samaan subiin millä tulostetaan kaikki viestit erilliseen *RichTextBoxiin*.

```
If lampotilaTB.InvokeRequired Then
    Dim wd As New DelegateWriteRtf(AddressOf WriteRtf)
    lampotilaTB.Invoke(wd, s)
Else
    lampotilaTB.AppendText(s & vbNewLine)
End If
```

Toimi odotetusti, mutta vielä pitäisi suodattaa mukaan vain lämpötilarivit ja niistäkin vain lukema, niin saamme ääkkösetkin mukaan ilman eri kikkailuja.

Eli laittelimme ehtoja ja säätöjä seuraavasti:

```
Sub WriteRtf(s As String)
    If vastRTB.InvokeRequired Then
        Dim wd As New DelegateWriteRtf(AddressOf WriteRtf)
        vastRTB.Invoke(wd, s)
    Else
        vastRTB.AppendText(s & vbNewLine)
        If Mid(s, 1, 9) = "Lampotila" Then
            lampotilaTB.Text = ""
            If Mid(s, 14, 1) = 0 Then 'alle sata astetta
                lampotilaTB.AppendText(Mid(s, 15, 2) & "," & Mid(s, 17, 1))
            Else 'yli sata astetta
                lampotilaTB.AppendText(Mid(s, 14, 3) & "," & Mid(s, 17, 1))
            End If
        End If
    End If
End Sub
```

Toimi hienosti:

https://youtu.be/Il_STpxuUsI

Napinpainalluksen tuominen omaan tekstikenttään

Tässä oli nyt hyvä monistaa ylempää ratkaisua ja lisäsimme koodia:

```
Sub WriteRtf(s As String)
    If vastRTB.InvokeRequired Then
        Dim wd As New DelegateWriteRtf(AddressOf WriteRtf)
        vastRTB.Invoke(wd, s)
    Else
        vastRTB.AppendText(s & vbNewLine)
        'Lämpötila
        If Mid(s, 1, 9) = "Lampotila" Then
            lampotilaTB.Text = ""
            If Mid(s, 14, 1) = 0 Then 'alle sata astetta
                lampotilaTB.AppendText(Mid(s, 15, 2) & "," & Mid(s, 17, 1))
            Else 'yli sata astetta
                lampotilaTB.AppendText(Mid(s, 14, 3) & "," & Mid(s, 17, 1))
            End If
        End If
        'Napinpainallus
        If Mid(s, 1, 6) = "Nappia" Then
            nappiTB.Text = ""
            nappiTB.AppendText(Now)
        End If
    End If

End Sub
```

Näkyi *GUI*:ssa napinpainalluksen jälkeen näin:

Etäisyyslukeman tuominen omaan tekstikenttään

Seuraavaksi monistimme ratkaisua etäisyyslukemaan:

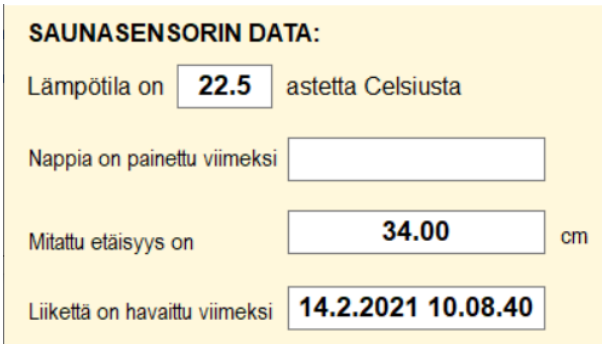
```
'Etäisyyssensori
If Mid(s, 1, 8) = "Etaisyys" Then
    etaisyysTB.Text = ""
    'Dim etaisyys2 As Double = CDbl(Val(etaisyysTB.Text))
    If Mid(s, 14, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 1) & "." & Mid(s, 15, 2)) ' < 10cm
    If Mid(s, 15, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 2) & "." & Mid(s, 16, 2)) ' > 10 cm < 100 cm
    If Mid(s, 16, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 3) & "." & Mid(s, 17, 2)) ' >= 100 cm
    Call UA() 'asettaa aikaleiman UÄ-sensorin ohitukselle
    Call lahtoSeuranta() 'määritellään onko löylyssä ketään, läsnäolo/poistuminen
End If
```

PIR-tiedon tuominen omaan tekstikenttään

Nyt lisäsimme *GUI*:hin liiketunnistimen hälytykset ja lisäsimme koodia *PIR*:rin osalta **vb.netin** koodiin. Koko osuus näytti nyt tältä:

```
'Lämpötilasensori
If Mid(s, 1, 9) = "Lampotila" Then 'kyseinen rivi koskee lämpötilaa ja mennään sen mukaiseen määrittelyyn
    lampotilaTB.Text = "" 'tyhjätään tekstikenttä, jotta jottei lado vain arvoja peräkkäin tekstikenttään
    If Mid(s, 14, 1) = "0" Then
        lampo = CDbl(Val(Mid(s, 15, 2))) 'jos alta sata astetta, hetkellinen lämpötila asteina muuttujaan
        lampotilaTB.AppendText(Mid(s, 15, 2) & "." & Mid(s, 17, 1)) 'alle sata astetta, arvo tekstikenttään
    Else
        lampo = CDbl(Val(Mid(s, 14, 3))) 'sata astetta tai yli, hetkellinen lämpötila asteina muuttujaan
        lampotilaTB.AppendText(Mid(s, 14, 3) & "." & Mid(s, 17, 1)) 'yli sata astetta, arvo tekstikenttään
    End If
    Call LampoAikaSeuranta() 'mennään lämpötilaseurannan subiin
End If

'Napinpainallus
If Mid(s, 1, 6) = "Nappia" Then
    nappiTB.Text = ""
    nappiTB.AppendText(Now)
End If

'Etäisyyssensori
If Mid(s, 1, 8) = "Etaisyys" Then
    etaisyysTB.Text = ""
    'Dim etaisyys2 As Double = CDbl(Val(etaisyysTB.Text))
    If Mid(s, 14, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 1) & "." & Mid(s, 15, 2)) ' < 10cm
    If Mid(s, 15, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 2) & "." & Mid(s, 16, 2)) ' > 10 cm < 100 cm
    If Mid(s, 16, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 3) & "." & Mid(s, 17, 2)) ' >= 100 cm
    Call UA() 'asettaa aikaleiman UÄ-sensorin ohitukselle
    Call lahtoSeuranta() 'määritellään onko löylyssä ketään, läsnäolo/poistuminen
End If

'Liiketunnistin
If Mid(s, 1, 8) = "Liiketta" Then
    liikeTB.Text = ""
    liikeTB.AppendText(Now)
    Call PIR() 'liikettä ollut, aikaleima ylös, jos myös UÄ reagoinut, hälytys jos tarkiste TOSI liian kovalle lämpötilalle
    Call maarittelyt_saunassaoloaikaehdolle() 'jos eka liike ja UÄ on havainnut tulon, niin käväistään laittamassa aikaleima muuttujalle
End If
```

Ja *GUI*:ssa tältä:



Alkavat olla osaset jo hyvällä mallilla. Kaikkien sensoreiden data saadaan nyt järkevästi näkymään *GUI*:ssa.

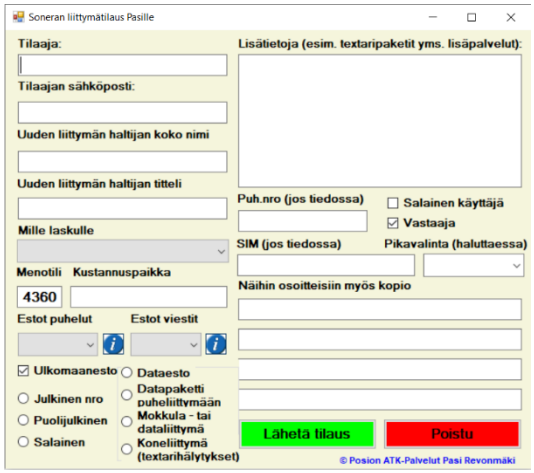
https://youtu.be/7yDhf2CVGlI

Seuraavaksi voisimme alkaa miettiä, kuinka tästä datasta saadaan lähtemään hälytys sähköpostiin.

12. Sähköpostilähetys

Tässä hyödynsimme surutta Pasin aiemmin luomaa koodia, jota hän hyödyntää aina sähköpostiominaisuuksia tekemiinsä ohjelmiin kaivatessaan.

Nappasimme subin matkapuhelinliittymätilausohjelmasta. Sillä on tilattu liittymät Posion kunnalle jo vuosia:



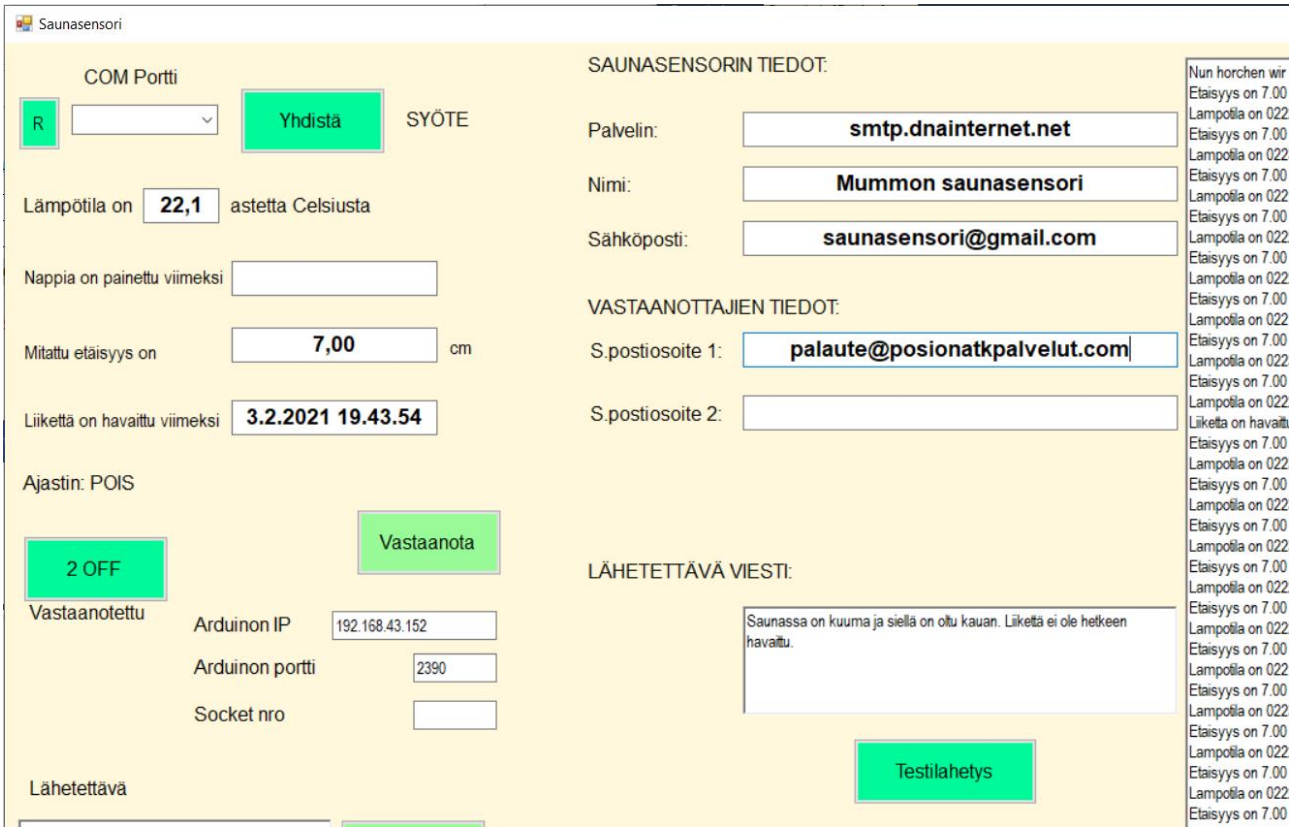
Ensin määrittelimme pari kirjastoa:

```
Imports System.IO
Imports System.Net.Mail
```

Ja sitten aloimme muokata subin koodia tähän tilanteeseen sopivaksi.

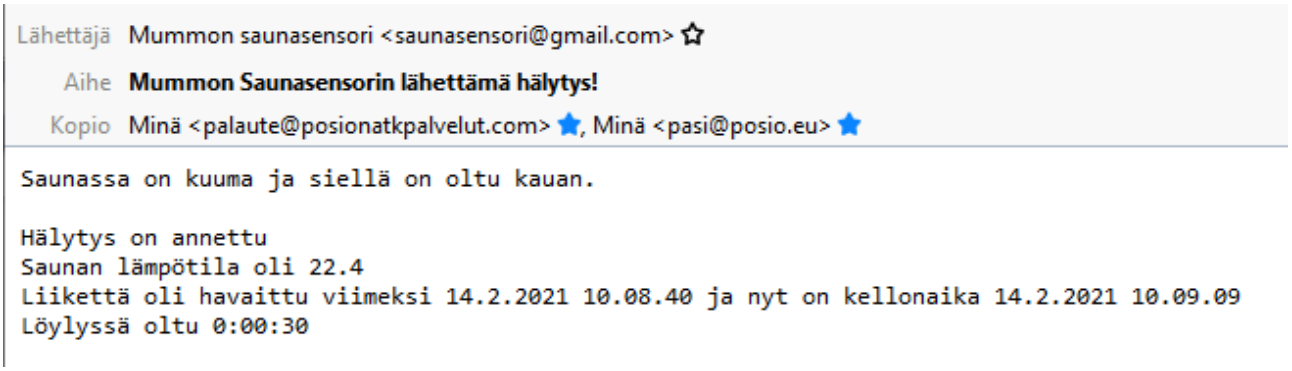
```
    Sub koostettu_viesti() 'Handles testiBTN.Click

        Dim s As String
        Dim smtp As New SmtpClient(palvelinTB.Text)

        teksti1 = ardnimiTB.Text
        teksti2 = arduinomeiliTB.Text
        s = viestiRTB.Text & vbCrLf & vbCrLf _
        & "Hälytys on annettu " & nappiTB.Text & vbCrLf _
        & "Saunan lämpötila oli " & lampotilaTB.Text & vbCrLf _
        & "Liikettä oli havaittu viimeksi " & liikeTB.Text & " ja nyt on kellonaika " & Now & vbCrLf _
        & "Löylyssä oltu " & SaunassaOloAikaRajaTB.Text

        Dim mail As New MailMessage()

        'mail.[To].Add("pasi@posio.eu") tähän voi laittaa osoitteita joihin aina halutaan meilin menevän, myös piilokopiot

        Try
            mail.From = New MailAddress(arduinomeiliTB.Text, ardnimiTB.Text)
        Catch exc As Exception
            MessageBox.Show("Antamasi Arduinon sähköpostiosoite ei ole kelvollinen.") : arduinomeiliTB.Focus() : GoTo loppu
        End Try
        If vastmeili1TB.Text <> "" Then
            Try
                mail.[CC].Add(vastmeili1TB.Text)
            Catch exc As Exception
                MessageBox.Show("Antamasi eka sähköpostiosoite ei ole kelvollinen.") : vastmeili1TB.Focus() : GoTo loppu
            End Try
        End If
        If vastmeili2TB.Text <> "" Then
            Try
                mail.[CC].Add(vastmeili2TB.Text)
            Catch exc As Exception
                MessageBox.Show("Antamasi toka sähköpostiosoite ei ole kelvollinen.") : vastmeili2TB.Focus() : GoTo loppu
            End Try
        End If

        mail.Subject = "Mummon Saunasensorin lähettämä hälytys!"
        mail.Body = s
        smtp.EnableSsl = True
        Try
            smtp.Send(mail)
        Catch exc As Exception
            MessageBox.Show("Nyt meni lähetys pieleen! Tarkista internet-yhteys ja yritä uudelleen.")
        End Try
loppu:

    End Sub
```

Tämän jälkeen muokkasimme käyttöliittymää ja lisäilimme muutaman kontrollin ja annoimme osalle niistä oletustekstit:



Starttasimme ohjelman ja lisäsimme tuon ensimmäisen vastaanottajan sähköpostiosoitteen (yllä täytettynä). Klikkasimme testilähetyspainiketta ja odotimme sydän syrjällään, että kilahtaako posti.

No kilahtihan se, hyvä! Tältä näytti saatu posti:



Sähköpostin lähetys siis toimii toivotusti. Seuraavaksi ajattelimme lähteä lisäilemään käyttöliittymään kontrolleja, joilla säädellään lämpötilan raja-arvoa ja ajastinta, kuinka heti hälytys annetaan, kun ehdot täyttyvät.

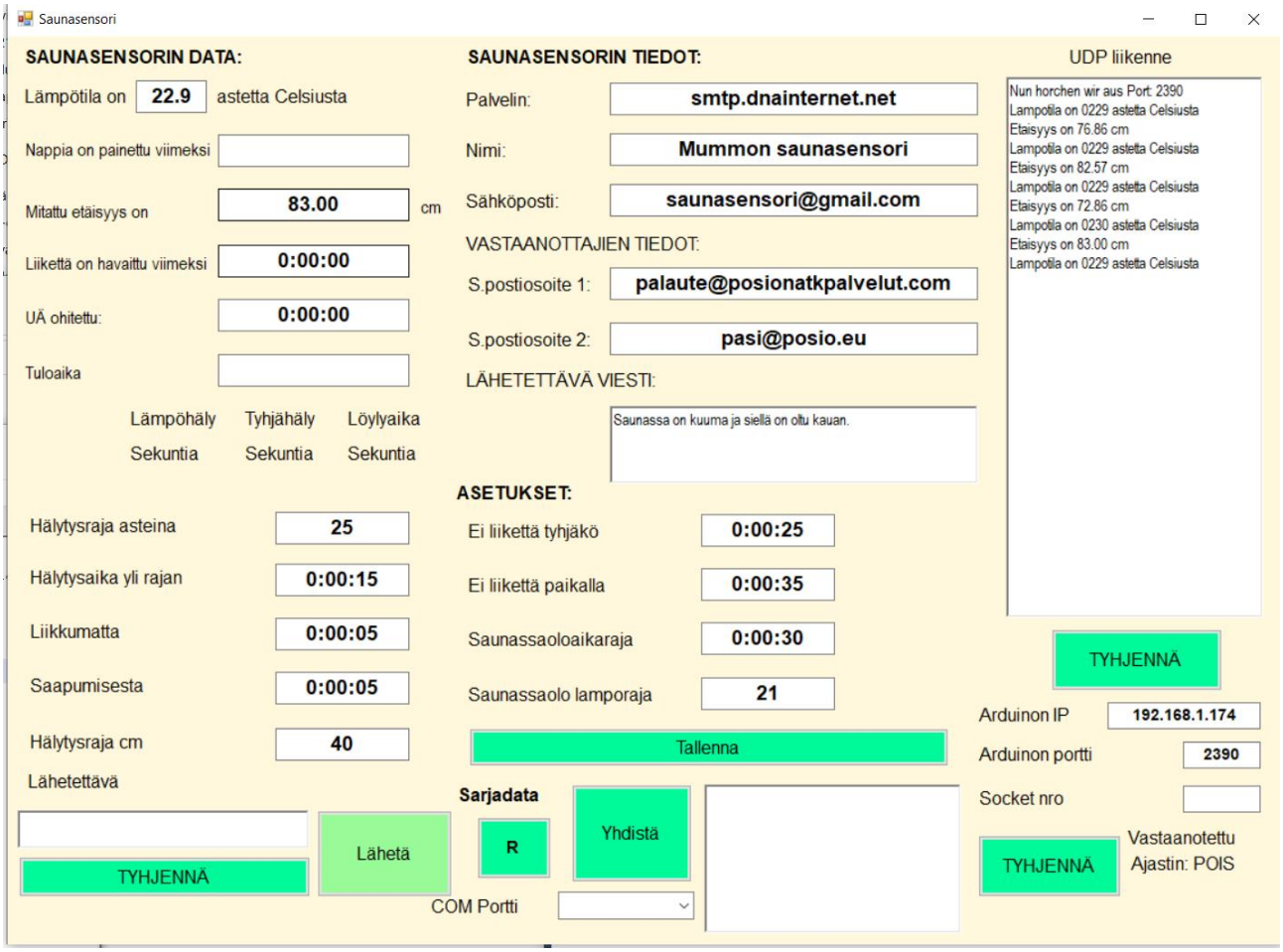
13. Raja-arvojen kontrollit käyttöliittymään ja koodi hälytyksille ja nollaamisille

Ajastimen toiminta

Lähdimme liikkeelle siitä, että saisimme laukaistua toiminnon, kun hälytyksestä olisi kulunut *GUI*:ssa määrätty aika.

http://www.vb-helper.com/howto_2005_countdown_timer.html

Tämä edellytti, että teimme ensin tarvittaville raja-arvoille muutaman uuden kontrollin. Samalla teimme jatkoa varten muutaman muunkin samalla kertaa. Nyt näytti tältä:



Saimme esimerkkikoodin toimimaan, joten seuraavaksi lähdimme koodaamaan seuraavaa ehtoa.

Logiikka hälytykselle

Tilanne 1.

Nyt piti miettiä, että mitkä reunaehdot tulisi täyttyä, jotta hälytys annetaan.

Lämpötilarajaylityksen ja sen kestoehdon tulee ensin täyttyä. Toisakseen löylyyn on pitänyt jonkun tulla ja se on tulkittu UÄ-sensorilla ja liiketunnistimella.

UÄ-anturi siis laukeaa saunaan tullessa ensin ja muutoksesta merkitään aikaleima muuttujaan. Kun sitten löylyhuoneessa havaitaan liikettä (*PIR*), niin katsotaan josko tuore aikaleima UÄ:ltä ja tästä tieto toiseen aikaleimaan ja näiden vertailusta päätellään että löylyyn on joku todella saapunut. Tällä eliminoidaan liiketunnistimen väärä hälytys.

Jos siis lämpötila- ja aikaraja ylittyy ja löylyssä on yhä joku, niin annetaan hälytys, mikäli läsnäoloehto täyttyy. Löylyssäolo päätellään *PIR*in havaitsemasta tai sitten jos liikettä ei havaita määrätyssä ajassa, niin katsotaan UÄ:n anturin hälytyksen viimeistä aikaleimaa (se ei saa olla sama kuin tuloleima) ja sen perusteella päätellään onko löylyssä enää ketään.

Jos löylyssä on siis yhä joku, niin annetaan hälytys.

Tilanne 2.

Lämpimään saunaan ja löylyyn on tultu, mutta sieltä ei ole poistuttu (UÄ-sensorin aikaleima) aikarajan puitteissa.

Tilanne 1. lämpötilaehto ja aikaraja sille

Otimme käsittelyyn lämpötilaehdon ja aikarajan sille. Funktion rakenteesta johtuen emme voineet käyttää lämpötilan hetkellisessä arvossa enää tekstikentän arvoa, joten määrittelimme sille kokonaislukumuuttujan *lampo* ja sille määrityksen seuraavasti:

```
'Lämpötilasensori
If Mid(s, 1, 9) = "Lampotila" Then 'kyseinen rivi koskee lämpötilaa ja mennään sen mukaiseen määrittelyyn
    lampotilaTB.Text = "" 'tyhjätään tekstikenttä, jotta jottei lado vain arvoja peräkkäin tekstikenttään
    If Mid(s, 14, 1) = "0" Then
        lampo = CDbl(Val(Mid(s, 15, 2))) 'jos alta sata astetta, hetkellinen lämpötila asteina muuttujaan
        lampotilaTB.AppendText(Mid(s, 15, 2) & "." & Mid(s, 17, 1)) 'alle sata astetta, arvo tekstikenttään
    Else
        lampo = CDbl(Val(Mid(s, 14, 3))) 'sata astetta tai yli, hetkellinen lämpötila asteina muuttujaan
        lampotilaTB.AppendText(Mid(s, 14, 3) & "." & Mid(s, 17, 1)) 'yli sata astetta, arvo tekstikenttään
    End If
    Call LampoAikaSeuranta() 'mennään lämpötilaseurannan subiin
End If
```

Koodasimme ensin ajastimen, että se käynnistyy, mikäli asetettu lämpötilaraja ylittyy ja samalla määritettiin aika kuinka kauan ajastin juoksee ennen kuin se laukeaa ja aiheuttaa tarvittaessa hälytyksen.

Toisessa subissa sitten juoksuttaa nollaan ja tekee hälytyksen tältä osin, mikäli yhä tarpeen. Koodimme näytti nyt tältä:

```
Private Sub LampoAikaSeuranta()

    Dim lampo2 As Double = CDbl(Val(rajalampoTB.Text)) 'muutetaan tekstikentän, jossa lämpötilaraja, arvo tekstistä desimaaliluvuksi

    If rajalampoAjastin.Enabled = False Then
        Dim kesto As TimeSpan = TimeSpan.Parse(halyaikaTB.Text)
        raja = Now.Add(kesto)
        If lampo >= lampo2 Then rajalampoAjastin.Enabled = True 'laitetaan lämpötila-ajastin tikittämään
    End If

End Sub
```

```
Private Sub tmrWait_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles rajalampoAjastin.Tick

    Dim lampo2 As Double = CDbl(Val(rajalampoTB.Text))
    Dim jaljella As TimeSpan = raja.Subtract(Now)
    jaljella = New TimeSpan(jaljella.Hours, jaljella.Minutes, jaljella.Seconds)
    If jaljella.TotalSeconds < 0 Then jaljella = TimeSpan.Zero
    jaljellaLBL.Text = jaljella.ToString
    If jaljella.TotalSeconds <= 0 Then
        If lampo >= lampo2 Then
            rajalampoAjastin.Enabled = False
            halyLampoAikaTrk = True
            'MsgBox("Hälytys annetaan, on yhä liian kuumaa ja on oltu aikarajaa kauemmin löylyssä!")
        Else
            rajalampoAjastin.Enabled = False
            halyLampoAikaTrk = False
            'MsgBox("Lämpötila ehti laskea tässä välissä rajan alle, ei hälytystä")
        End If
    End If

End Sub
```

https://youtu.be/1V8A2A3R214

Kuten videosta näkyy (videossa *GUI*:n vanhempi versio), niin ei startannut, jos lämpötila oli alle hälytysrajan, ei antanut hälytystä, jos lämpötila ehti laskea rajan alle ja antoi hälytyksen, jos lämpötila oli vielä yli rajan, kun aikaraja tuli vastaan.

Tuloehto UÄ- ja liikesensori

Lähdimme liikkeelle luomalla parit muuttujat:

```
Dim aikaUA As Date
Dim aikaPIR As Date
```

Sitten lähdimme luomaan omat subit sensoreille:

```
Private Sub UA()
    If etaisyysTB.Text <> "" Then
        Dim etaisyys As Double = CDbl(Val(etaisyysTB.Text))
        If etaisyys < rajacmTB.Text Then aikaUA = Now : uaTB.Text = aikaUA 'laitetaan ylös milloin on UÄ-sensorin ohi tultu
    End If
End Sub
```

```
Private Sub PIR() 'PIR on huomannut liikettä ja jos lämpötsekkaus on myös TOSI, niin hälytys
    Dim liike As DateTime = DateTime.Parse(liikeTB.Text)
    If aikaUA <> Nothing And liike > aikaUA Then 'on yleensäkin tultu ohi UÄ-sensorin
        If halyLampoAikaTrk = True Then Call koostettu_viesti() ':MessageBox.Show("HÄLYTYS! HÄLYTYS!", "HÄLYTYS")
        halyOliKovaLampoTrk = True
        'nollataan_muuttujat()
    End If
End Sub
```

Eli UÄ-sensori toteaa että joku meni ohi ja laittoi aikaleiman ylös. Sitten *PIR*-sensori näkee liikettä, toteaa että UÄ on nähnyt myös ja päättelee, että nyt on löylyyn joku saapunut. Seuraavaksi koodit vain piti saada

ajettua järkevään aikaan. Pidimme helpoimpana tapana kytkeä ne aiempien tekstikenttien laukaisuihin, koodissa näin:

```
'Etäisyyssensori
If Mid(s, 1, 8) = "Etaisyys" Then
    etaisyysTB.Text = ""
    'Dim etaisyys2 As Double = CDbl(Val(etaisyysTB.Text))
    If Mid(s, 14, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 1) & "." & Mid(s, 15, 2)) ' < 10cm
    If Mid(s, 15, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 2) & "." & Mid(s, 16, 2)) ' > 10 cm < 100 cm
    If Mid(s, 16, 1) = "." Then etaisyysTB.AppendText(Mid(s, 13, 3) & "." & Mid(s, 17, 2)) ' >= 100 cm
    Call UA() 'asettaa aikaleiman UÄ-sensorin ohitukselle
    Call lahtoSeuranta() 'määritellään onko löylyssä ketään, läsnäolo/poistuminen
End If
```

Testissämme näytti tältä (vanhempi GUI):

https://youtu.be/q5m1-joQZUU

Hyvin toimii, kiva.

Tilanne 1. ehtojen yhdistäminen

Seuraavaksi yhdistimme nämä kaksi vaatimusta. Ensimmäiseksi huomasimme tarvitsevamme yhden TOSI/EPÄTOSI -tarkisteen yhdistämistä varten. Tämä tarkiste muuttuu todeksi, kun lämpö- ja aikaraja yhdessä ylitetään. Ja kun sitten myös liikettä havaitaan, niin mikäli tarkiste on tosi, niin hälytys annetaan. Tässä kohtaa piti siis saada lämpötila- ja aikaseuranta juoksemaan myös koko ajan taustalle. Kytkimme senkin siis samaan tapaan sinne tekstikentän muutosseurantaan, muutimme samalla subin nimen kuvaavammaksi:

```
'Lämpötilasensori
If Mid(s, 1, 9) = "Lampotila" Then 'kyseinen rivi koskee lämpötilaa ja mennään sen mukaiseen määrittelyyn
    lampotilaTB.Text = "" 'tyhjätään tekstikenttä, jotta jottei lado vain arvoja peräkkäin tekstikenttään
    If Mid(s, 14, 1) = "0" Then
        lampo = CDbl(Val(Mid(s, 15, 2))) 'jos alta sata astetta, hetkellinen lämpötila asteina muuttujaan
        lampotilaTB.AppendText(Mid(s, 15, 2) & "." & Mid(s, 17, 1)) 'alle sata astetta, arvo tekstikenttään
    Else
        lampo = CDbl(Val(Mid(s, 14, 3))) 'sata astetta tai yli, hetkellinen lämpötila asteina muuttujaan
        lampotilaTB.AppendText(Mid(s, 14, 3) & "." & Mid(s, 17, 1)) 'yli sata astetta, arvo tekstikenttään
    End If
    Call LampoAikaSeuranta() 'mennään lämpötilaseurannan subiin
End If
```

Seurantakoodi oli siis nyt näin:

```
Private Sub LampoAikaSeuranta()

    Dim lampo2 As Double = CDbl(Val(rajalampoTB.Text)) 'muutetaan tekstikentän, jossa lämpötilaraja, arvo tekstistä desimaaliluvuksi

    If rajalampoAjastin.Enabled = False Then
        Dim kesto As TimeSpan = TimeSpan.Parse(halyaikaTB.Text)
        raja = Now.Add(kesto)
        If lampo >= lampo2 Then rajalampoAjastin.Enabled = True 'laitetaan lämpötila-ajastin tikittämään
    End If

End Sub
```

```
Private Sub tmrWait_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles rajalampoAjastin.Tick

    Dim lampo2 As Double = CDbl(Val(rajalampoTB.Text))
    Dim jaljella As TimeSpan = raja.Subtract(Now)
    jaljella = New TimeSpan(jaljella.Hours, jaljella.Minutes, jaljella.Seconds)
    If jaljella.TotalSeconds < 0 Then jaljella = TimeSpan.Zero
    jaljellaLBL.Text = jaljella.ToString
    If jaljella.TotalSeconds <= 0 Then
        If lampo >= lampo2 Then
            rajalampoAjastin.Enabled = False
            halyLampoAikaTrk = True
            'MsgBox("Hälytys annetaan, on yhä liian kuumaa ja on oltu aikarajaa kauemmin löylyssä!")
        Else
            rajalampoAjastin.Enabled = False
            halyLampoAikaTrk = False
            'MsgBox("Lämpötila ehti laskea tässä välissä rajan alle, ei hälytystä")
        End If
    End If

End Sub
```

```
Private Sub UA()
    If etaisyysTB.Text <> "" Then
        Dim etaisyys As Double = CDbl(Val(etaisyysTB.Text))
        If etaisyys < rajacmTB.Text Then aikaUA = Now : uaTB.Text = aikaUA 'laitetaan ylös milloin on UÄ-sensorin ohi tultu
    End If
End Sub
```

```
Private Sub PIR() 'PIR on huomannut liikettä ja jos lämpötsekkaus on myös TOSI, niin hälytys
    Dim liike As DateTime = DateTime.Parse(liikeTB.Text)
    If aikaUA <> Nothing And liike > aikaUA Then 'on yleensäkin tultu ohi UÄ-sensorin
        If halyLampoAikaTrk = True Then MessageBox.Show("HÄLYTYS! HÄLYTYS!", "HÄLYTYS")
    End If
End Sub
```

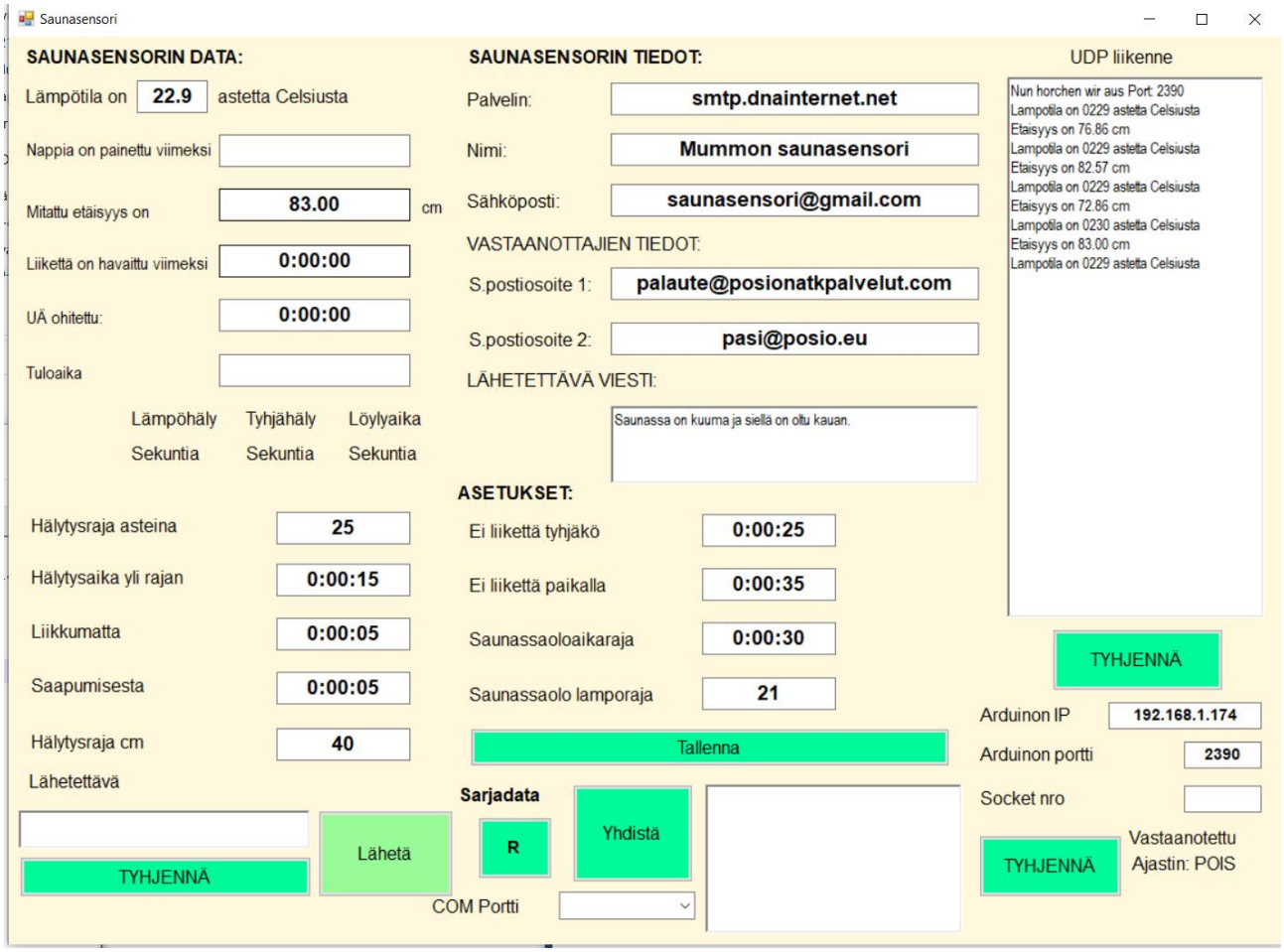
Testasimme ja saimme tällaista (vanha GUI):

https://youtu.be/cglHSnGBB9I

Pelasi, hyvä.

Logiikka milloin löylyhuone on tyhjentynyt

Tässä ajattelimme seuraavanlaista kuviota. Kun liikesensori hälyttää, niin laitetaan ajastin juoksemaan ja kun määrätyn ajan kuluttua tarkistetaan *PIR*rin huomaaman viimeisen liikkeen aikaleima, niin sen tulee olla aikaleimaltaan sama koko ajastimen *lahtoAjastin* juoksun aikana ja sen on oltava vanhempi kuin UÄ:n (kulkusuuntaindikaattorimme). Jos näin on, niin nollataan kaikki arvot alkutilanteeseen. Tämä vaati taas yhden uuden kontrollin ja arvon tauluun ja teimme samaan syssyyn toisetkin, koska myös tilanne 2 vaatisi yhden lisää molempia.



Lähtöehto koodiin, UÄ ja *PIR*

Laitetaan uusi ajastin *lahtoAjastin*

Lisäillään tarvittava koodi. Se oli taas vain pieni askel mille tahansa kunnalle ja puoli sivua koodia, mutta luojaltaan taas paljon hikeä, muttei onneksi kyyneleitä. Koodimme kommentit kertokoon tyhjentävästi mistä oli kyse ja miten se toteutettiin. Tulikohan koodaajalle henkilökohtainen ennätys sisäkkäisissä *If*-lauseissa.

```
Private Sub lahtoSeuranta() 'tänne tullaan kun UÄ-sensori hälyttää
    'If liikeTB.Text <> "" Then
    Dim liike2 As DateTime = DateTime.Parse(liikeTB.Text)
    If etaisyysTB.Text <> "" Then
        paikallaTrk = True 'asetetaan paikallaolomuuttujan arvo todeksi koska liikettä ollut löylyhuoneessa ja UÄ-sensorin mukaan ohi menty
        Dim etaisyys2 As Double = CDbl(Val(etaisyysTB.Text))
        If lahtoAjastin.Enabled = False Then 'jos lähtöajastin ei ole käynnissä niin silloin jatketaan
            If etaisyys2 < rajacmTB.Text Then 'UÄ-anturi hälytysrajan alle, eli jokin meni sensorin ohi
                If aikaUA > liike2 Then 'jos lähtöaika on suurempi kuin viimeksi havaittu liike niin jatketaan
                    aikaPIR = liike2 'laitetaan muuttujaan viimeisen liikkeen havannointiaika, jotta osataan verrata
                    Dim kesto2 As TimeSpan = TimeSpan.Parse(AikaTyhjakoTB.Text) 'määritetään kontrollista, että kauanko pitää löylyhuoneen olla liikkeettä, jotta tulkitaan tyhjäksi
                    raja2 = Now.Add(kesto2) 'lisätään määritelty aika tähän hetkeen
                    lahtoAjastin.Enabled = True 'laitetaan lähtökello tikittämään
                End If
            End If
        End If
    End If
    'End If
End Sub
```

```
Private Sub lahto_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles lahtoAjastin.Tick
    ' If liikeTB.Text <> "" Then 'jotta ei koeteta parsia datatyyppiä ennen kuin on arvo
    Dim liike2 As DateTime = DateTime.Parse(liikeTB.Text)
    Dim jaljella2 As TimeSpan = raja2.Subtract(Now) 'laskurin jäljellä oleva aika, kun se startataan
    jaljella2 = New TimeSpan(jaljella2.Hours, jaljella2.Minutes, jaljella2.Seconds)
    If jaljella2.TotalSeconds < 0 Then jaljella2 = TimeSpan.Zero
    EiLiikettäLBL.Text = jaljella2.ToString 'näytetään jäljelläoleva aika labelissa
    If jaljella2.TotalSeconds <= 0 Then 'kun laskuri menee nollaan, niin mitä tehdään
        If liike2 = aikaPIR Then 'jos viimeinen liikkeen aikaleima on sama myös testausajan päättyessä, niin tulkitaan löylyhuone tyhjäksi
            lahtoAjastin.Enabled = False 'laitetaan ajastin pois päältä
            'MsgBox("Löylyhuone on tyhjä") 'kerrotaan tilanne
            Call nollataan_muuttujat() 'tyhjennetään arvot, eli palautetaan alkutilanne
        Else
            lahtoAjastin.Enabled = False 'laitetaan ajastin pois päältä
            'MsgBox("Liikettä oli tässä välissä, löylyhuone ei ollut tyhjä")
        End If
    End If
    'End If
End Sub
```

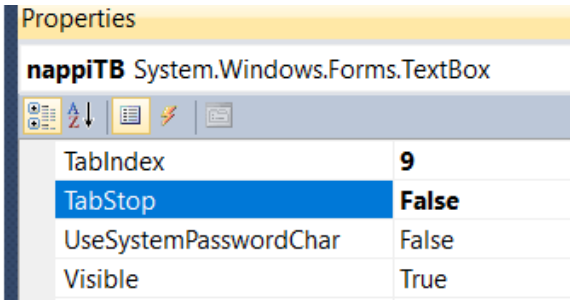
Tilanne 2. löylyssä ollaan liian pitkään

Loimme jälleen pari kontrollia lisää, jotta maksimiaika ja -lämpö löylyssäololle ilman hälytystä saadaan *GUI*:ssa asetettua. Ja näille taulukkoon omat solunsa *(3, 2) & (3, 3)* ja näiden huomioiminen tallennuksessa ja lataamisessa. Koodiin määrittelimme saunaoloaikaehdolle seuraavasti. Koodin kommentit puhukoot puolestaan:

```
Sub maarittelyt_saunassaoloaikaehdolle()

    If liikeTB.Text <> "" Then 'jotta ei koeteta parsia datatyyppiä ennen kuin on arvo
        If liikeEkaTulo = Nothing Then liikeEkaTulo = DateTime.Parse(liikeTB.Text) : TuloaikaTB.Text = liikeEkaTulo 'merkataan ensimmäinen saapuminen ihan omaan muuttujaan,
        'jotta saadaan kokonaislöylyssäoloaika laskettua, mutta arvoja asetettaessa huomioitava että onko monta saunojaa peräkkäin, jolloin kannattaa laittaa pois päältä (pitkä aika)
        Dim kesto As TimeSpan = TimeSpan.Parse(SaunassaOloAikaRajaTB.Text)
        raja3 = Now.Add(kesto)
        kauankoSaunassaAjastin.Enabled = True
    End If

End Sub
```

```
Private Sub kauanko_saunassa_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles kauankoSaunassaAjastin.Tick

    Dim liike3 As DateTime = DateTime.Parse(liikeTB.Text)
    Dim jaljella3 As TimeSpan = raja3.Subtract(Now) 'saunassaoloaikarajan laskurin jäljellä oleva aika, kun se startataan
    jaljella3 = New TimeSpan(jaljella3.Hours, jaljella3.Minutes, jaljella3.Seconds)
    If jaljella3.TotalSeconds < 0 Then jaljella3 = TimeSpan.Zero
    SaunassaOloAikaHalyLBL.Text = jaljella3.ToString 'näytetään jäljelläoleva aika labelissa
    If jaljella3.TotalSeconds <= 0 Then 'kun laskuri menee nollaan, niin mitä tehdään
        If paikallaTrk = true Then 'jos yhä löylyssä liikettä, niin annetaan hälytys
            kauankoSaunassaAjastin.Enabled = False 'laitetaan ajastin pois päältä
            'MsgBox("Lämpimässä saunassa on oltu liian kauan") 'kerrotaan tilanne
            halyOliPitkaAikaTrk = True
            Call koostettu_viesti()
            Call nollataan_muuttujat() 'tyhjennetään arvot, eli palautetaan alkutilanne
        Else
            kauankoSaunassaAjastin.Enabled = False 'laitetaan ajastin pois päältä
            MsgBox("Löylyhuone tyhjä")
        End If
    End If
End Sub
```

TabStopit ja *TabIndexit* kohdalleen *GUI*:ssa



Laittelimme *GUI*:ssa *TabSopit* pois päältä niihin kontrolleihin, joihin emme tarkoittaneet käyttäjän koskevan. Tässä kannatti huomata, että kaikkien noiden kenttien arvon saattoi laitella kerralla *Falseksi*, kunhan ensin nokki kaikki ne valintaan hiirellä ja *CTRL*-napilla. Sama päto muihin yhteisiin asetuksiin ja olimmekin jo aiemmin viilailleet kenttien ulkoasua tuolla tavalla.

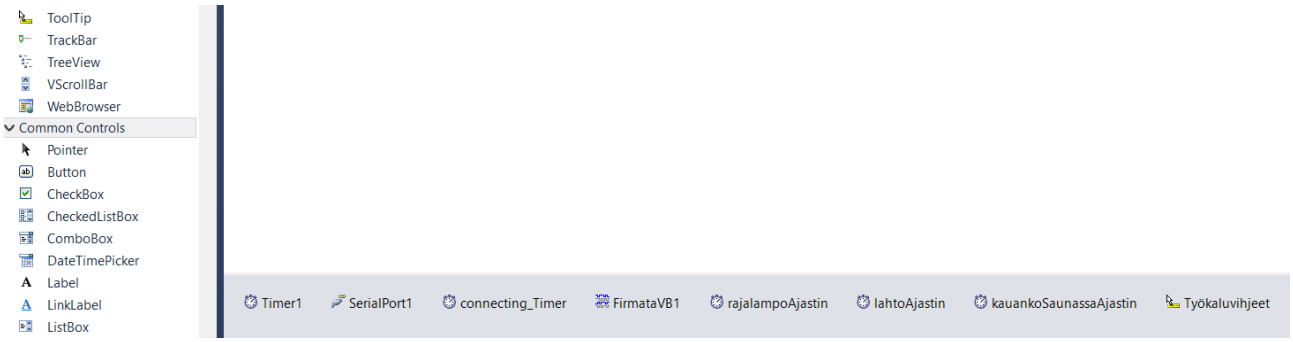
TabIndexit laittelimme siihen järjestykseen, että tabulaattorilla järjestys on käyttäjälleen looginen. Meillä oli aiempaa kokemusta tilanteesta, jossa näihin ei oltu kiinnitetty huomiota ja vasta käyttöönotossa huomattiin kuinka järjestys pomppi sinne sun tänne ja käyttäjä pääsi kenttiin, joihin ei ollut tarkoitus. Mitään emme myönnä.

Viilailumme jälkeen testi meni näin:

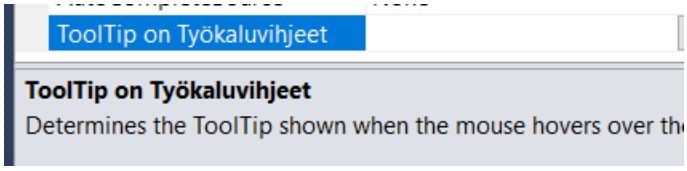
https://youtu.be/1ZZujcPsULw

Työkaluvihjeet (ToolTips) *GUI*:hin

Ensiksi meidän tuli lisätä tämä työkalu lomakkeeseen. Nimesimme sen kuva mukaisesti *Työkaluvihjeet*



Nyt saimme kontrollien ominaisuuksin kentän mihin syöttää vihjeet:



Teimme jokaiseen kontrolliin mihin käyttäjä pääsee syöttämään tai käyttämään, oman vihjeensä.

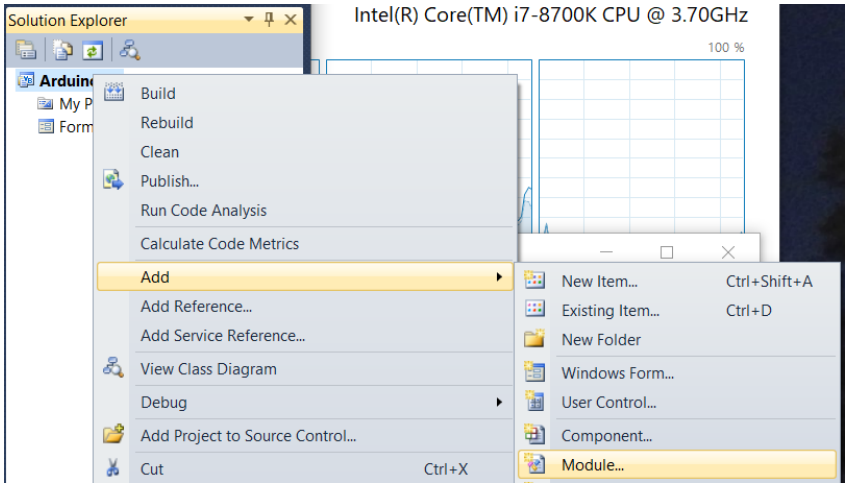
14. Käyttöliittymässä asetettujen arvojen tallentaminen

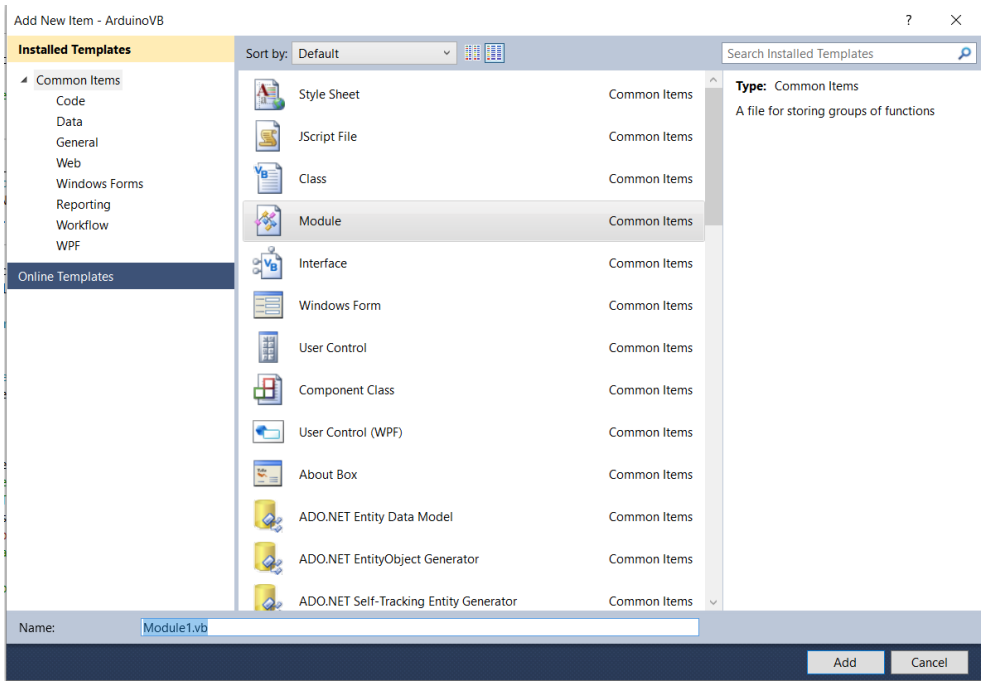
Perusrakenne

Tässä vaiheessa huomasimme, että olisi korkein aika saada tallennettua käyttöliittymään syötetyt arvot ja ladata ne aina käynnistettäessä ohjelma. Demossa arvot olisi ollut sinänsä helppo asettaa käsin kontrollereille suoraan koodiin, mutta oikeassa toteutuksessa se ei olisi ollut mahdollista.

Niinpä nappasimme Pasilta taas valmista koodia ja hyödynsimme sitä. Pasin aiemmissa ohjelmissa toteutus oli ollut pienimuotoisemmissa ohjelmissa sellainen, että arvot tallennettiin ja luettiin ihan taulukkomuotoisesta *plain text -tiedostosta*, eli puhtaasta tekstitiedostosta. Arvoihin viitattiin taulu/taulukkotyyppisesti ja erottimena toimi tolppamerkki.

Koodit ja funktiot olivat tyypillisesti *ModuleMain* -moduulissa, joten loimme sellaisen myös tähän ohjelmaan. Moduuleissa ollessaanhan koodit ja funktiot ovat käytettävissä kaikkialla projektissa.





Nimeksi ehdotteli *Module1.vb*, se sopii, sillä ei ole varsinaista merkitystä tässä. Sinne sitten laitoimme tarvitsemamme koodit. Lisäsimme muutamia muitakin pikku apuja kopioimastamme ohjelmakoodista. Salaamisen remmasimme siitä tarpeettomana ainakin tässä vaiheessa.

Module1

```
    Private Const KEYEVENTF_KEYUP As Byte = &H2

    Private Sub closeMsgbox(ByVal delay As Object)
        Threading.Thread.Sleep(CInt(delay) * 1000)
        'AppActivate(Me.Text)
        keybd_event(VK_RETURN, 0, KEYEVENTF_KEYDOWN, 0)
        keybd_event(VK_RETURN, 0, KEYEVENTF_KEYUP, 0)
    End Sub

    Public Sub msgboxautoclose(Optional ByVal Message As String = "Odota...", Optional ByVal Style As MsgBoxStyle = vbSystemModal, _
                               Optional ByVal title As String = Nothing, Optional ByVal delay As Decimal = 0.2)
        Dim t As New Threading.Thread(AddressOf closeMsgbox)
        t.Start(delay) '5 second default delay
        MsgBox(Message, Style, title)
    End Sub

    Public Function Lataa(ByVal ladattava As String, ByRef taulunnimi(,) As String _
                          , ByRef rivinumero As String, ByRef srknumero As String, Optional ByRef salaus As Boolean = False, _
                          Optional ByVal verrattavanro As Integer = 0, Optional ByVal verrattava As String = "", _
                          Optional ByRef valittu(,) As String = Nothing, Optional ByVal poista As Boolean = False, _
                          Optional ByVal kasvata As Integer = 0)
        Dim luuppi As Integer = 0
alku:
        Try

            Dim rivitieto()
            'System.Threading.Thread.Sleep(200)
            Dim kama As System.IO.StreamReader = New System.IO.StreamReader(ladattava, Encoding.UTF8)
            If File.Exists(ladattava) Then
                kama = File.OpenText(ladattava)
                rivitieto = kama.ReadToEnd().Split(Environment.NewLine)
                rivinumero = UBound(rivitieto)
                'POISTA If salaus = True Then rivitieto(0) = Pura(rivitieto(0))
                osatrivi = rivitieto(0).Split("|")
                srknumero = UBound(osatrivi)
                ReDim taulunnimi(rivinumero + 1 + kasvata, srknumero + 1)
                ReDim valittu(1, srknumero + 1)
                Dim x As Integer
                Dim y As Integer
                For x = 0 To rivinumero - 1
                    If rivitieto(x) <> "" Then
                        'If x > 0 And salaus = True Then rivitieto(x) = Pura(rivitieto(x))
                        osatrivi = rivitieto(x).split("|")
                        If x = 0 Then osatrivi(0) = Mid(osatrivi(0), 1, 400) Else osatrivi(0) = Mid(osatrivi(0), 2, 400)
                        For y = 0 To srknumero
                            If osatrivi(verrattavanro) <> verrattava Or poista = False Then 'Otetaan mukaan jos valitun rivin poistoa ei ole pyydetty tai rivin tiedot eivät matchaa valittuun
                                taulunnimi(x, y) = osatrivi(y)
                            End If
                            If osatrivi(verrattavanro) = verrattava Then 'Ajetaan taulukkoon valitun rivin tiedot ja rivinnumero
                                '  MsgBox("Tärppäs")
                                valittu(0, y) = taulunnimi(x, y)
                                dellaarivi = x
                            End If
                        Next y
                        'End If
                    End If
                Next x
            End If
            kama.Close()
            GC.Collect()
        Catch
            luuppi = luuppi + 1
            msgboxautoclose("Odota hetki", , "Odota...", 0.2)
            If luuppi < 10 Then GoTo alku
            MsgBox("tiedoston " & ladattava & " lukeminen epäonnistui! tarkista polku ja muut määritykset.")
        Finally
        End Try

    End Function

    Public Function Tallenna(ByVal taulunnimi(,) As String, ByVal tallennettava As String, ByVal rivinumero As String, Optional ByVal salaus As Boolean = False, Optional ByVal lisamerkit As String = "")

        GC.Collect()
        Dim luuppi As Integer = 0
alku:

        Try

            Venaa(200)
            Dim erotinmerkki As String = "|"
            Dim kirjoitettava As String

            Dim kama As New StreamWriter(File.Create(tallennettava), System.Text.Encoding.UTF8)
            rivinumero = UBound(taulunnimi, 1) + 1
            srknumero = (taulunnimi.Length / rivinumero) ' + lisaa_sarakkeita
            'MsgBox(srknumero)
            kirjoitettava = ""
            Dim x As Integer
            Dim y As Integer
            For x = 0 To rivinumero - 1
                kirjoitettava = ""
                For y = 0 To srknumero - 1
                    kirjoitettava += Replace(taulunnimi(x, y), erotinmerkki, "") & IIf(y < srknumero - 2, erotinmerkki, "")
                Next y
                Rivivex(kirjoitettava)
                If Replace(kirjoitettava, erotinmerkki, "") <> "" Then
                    kirjoitettava = kirjoitettava & lisamerkit 'Jos tarvitaan lisäsarakkeita
                    'MsgBox("Lisämerkit " & lisamerkit)
                    'POISTA If salaus = True Then kirjoitettava = Salaa(kirjoitettava)
                    kama.WriteLine(kirjoitettava)
                    'MsgBox(kirjoitettava)
                End If
            Next x

            kama.Flush()
            kama.Close()
            ''End Using
            GC.Collect()

        Catch ex As Exception

            luuppi = luuppi + 1
            ''Venaa(200)
            msgboxautoclose("Odota hetki", , "Odota...", 0.2)
            If luuppi < 10 Then GoTo alku

            If luuppi = 10 Then
                MsgBox("Tiedoston " & tallennettava & " tallentaminen takkuaa. Kokeillaan vielä kerran.")
                GoTo alku
            End If
            If luuppi > 10 Then
                MsgBox("Tiedoston " & tallennettava & " tallentaminen valitettavasti epäonnistui! Tarkista polku ja muut määritykset.")
            End If

        End Try

    End Function


    Sub Venaa(ByVal aikams As Integer)

        'Set interval to 30 minutes
        'tmr.Interval = TimeSpan.FromMinutes(30).TotalMilliseconds
        tmr.AutoReset = False
        tmr.Interval = aikams
        'tmr.Interval = 2000
        tmr.Start()

        'Exit Sub

        'Just wait.  This allows user input. Desired? I didn't know how
        'to just pause and wait without user interaction.

        Do While aaaa < 2

            'Console.ReadLine()
        Loop
        aaaa = 1
    End Sub

    Public Function Rivivex(ByRef rivitys As String)

        rivitys = Replace(rivitys, vbCr, "")
        rivitys = Replace(rivitys, vbLf, "")
        rivitys = Replace(rivitys, vbCrLf, "")

    End Function

    WithEvents tmr As New Timers.Timer
    Dim aaaa As Integer = 1
    Private Sub Tikittaa(ByVal sender As Object, ByVal e As System.Timers.ElapsedEventArgs) Handles tmr.Elapsed

        aaaa = aaaa + 1

    End Sub

End Module
```

Salaaminen

Jos kyseessä olisi ollut sensitiivinen data, niin *Salaa*-funktiolla olisi ollut mahdollista salata tiedostot vahvasti.

```
Public Function Salaa(ByVal kirjoitettava As String) As String

    Avain =                                          g
    Suola =
    Hash = "
    Iteraati
    IV = "@1
    Bitit =

    Dim initVectorBytes As Byte()
    initVectorBytes = Encoding.ASCII.GetBytes(IV)

    Dim saltValueBytes As Byte()
    saltValueBytes = Encoding.ASCII.GetBytes(Suola)

    Dim plainTextBytes As Byte()
    If kirjoitettava <> "" Then plainTextBytes = Encoding.UTF8.GetBytes(kirjoitettava)

    Dim password As PasswordDeriveBytes
    password = New PasswordDeriveBytes(Avain, saltValueBytes, Hash, Iteraatiot)

    ' Use the password to generate pseudo-random bytes for the encryption
    ' key. Specify the size of the key in bytes (instead of bits).
    Dim keyBytes As Byte()
    keyBytes = password.GetBytes(Bitit / 8)

    Dim symmetricKey As RijndaelManaged
    symmetricKey = New RijndaelManaged()
    symmetricKey.Mode = CipherMode.CBC
    Dim encryptor As ICryptoTransform
    encryptor = symmetricKey.CreateEncryptor(keyBytes, initVectorBytes)
    Dim memoryStream As MemoryStream
    memoryStream = New MemoryStream()
    Dim cryptoStream As CryptoStream
    cryptoStream = New CryptoStream(memoryStream, encryptor, CryptoStreamMode.Write)
    ' Start encrypting.
    If kirjoitettava <> "" Then cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length)
    ' Finish encrypting.
    cryptoStream.FlushFinalBlock()
    ' Convert our encrypted data from a memory stream into a byte array.
    Dim cipherTextBytes As Byte()
    cipherTextBytes = memoryStream.ToArray()
    ' Close both streams.
    memoryStream.Close()
    cryptoStream.Close()
    Dim salattu As String
    salattu = Convert.ToBase64String(cipherTextBytes)
    ' Return encrypted string.
    Salaa = salattu
End Function
```

Purkamisen osuus sitten seuraavasti:

```
Public Function Pura(ByRef salattu As String) As String

    Avain
    Suola
    Hash =
    Iteraa
    IV = "
    Bitit

    Dim initVectorBytes As Byte()
    initVectorBytes = Encoding.ASCII.GetBytes(IV)
    Dim saltValueBytes As Byte()
    saltValueBytes = Encoding.ASCII.GetBytes(Suola)
    ' Convert our ciphertext into a byte array.
    Dim cipherTextBytes As Byte()
    cipherTextBytes = Convert.FromBase64String(salattu)

    Dim password As PasswordDeriveBytes
    password = New PasswordDeriveBytes(Avain, saltValueBytes, Hash, Iteraatiot)

    Dim keyBytes As Byte()
    keyBytes = password.GetBytes(Bitit / 8)

    Dim symmetricKey As RijndaelManaged
    symmetricKey = New RijndaelManaged()
    symmetricKey.Mode = CipherMode.CBC
    Dim decryptor As ICryptoTransform
    decryptor = symmetricKey.CreateDecryptor(keyBytes, initVectorBytes)
    Dim memoryStream As MemoryStream
    memoryStream = New MemoryStream(cipherTextBytes)
    Dim cryptoStream As CryptoStream
    cryptoStream = New CryptoStream(memoryStream, decryptor, CryptoStreamMode.Read)
    Dim plainTextBytes As Byte()
    ReDim plainTextBytes(cipherTextBytes.Length)
    Dim decryptedByteCount As Integer
    decryptedByteCount = cryptoStream.Read(plainTextBytes, 0, plainTextBytes.Length)

    ' Close both streams.
    memoryStream.Close()
    cryptoStream.Close()

    ' Convert decrypted data into a string.
    ' Let us assume that the original plaintext string was UTF8-encoded.
    Dim kirjoitettava As String
    kirjoitettava = Encoding.UTF8.GetString(plainTextBytes, 0, decryptedByteCount)

    ' Return decrypted string.
    Pura = kirjoitettava
End Function
```

Käyttö

Nyt määrittelimme asetustiedoston nimen ja että se ladataan heti startissa.

```
Public filuAsetukset = "asetukset.dat"


Private Sub Form1_Load(ByVal sender As System.Object
    Lataa(filuAsetukset, tauluyht, riviyht, srkyht)
```

Sitten lähdimme miettimään asetustiedoston rakennetta. Tiedosto oli kätevin luoda **Notepad++**:lla.

https://notepad-plus-plus.org/

Se on todella monipuolinen editori muutenkin ja kannattaakin laittaa oletuseditoriksi **Windowsissa**.

Tämän tiedoston sijainti oli järkevintä asettaa ohjelmatiedoston alikansioon. Alikansion nimesimme ytimekkäästi *data*ksi ja teimme samalla sillekin alakansion *bu*, jonne tulevat tallentumaan varmuuskopiot asetuksista.

Katsoimme käyttöliittymää ja sen kenttiä, joihin tietoja voi syöttää.

Päädyimme alustavasti viiteen sarakkeeseen ja ainakin aluksi kolme riviä riittäisi. Kuvassa tekstiselitteet.

asetukset.dat

```
arduinon IP|arduinon portti|nettiyhteyden lähettävä palvelin|
lähettäjän nimi|lähettäjän sähköposti|eka vastaanottajan sähköposti|toka vastaanottajan sähköposti
lähetettävä viesti|hälytysraja asteina|hälytysaika yli minmilämmön sekunteina|hälytysaika liikkumattomalle sekunteina|saapumisesta montako sekuntia
```

Tuon jälkeen lisäillessämme toimintoja otimme käyttöön lisäksi:

(1, 4) = viesti kun lämpimässä saunassa oltu liian kauan

(3, 0) = ei liikettä, tyhjäkö

(3, 1) = ei liikettä, mutta joku paikalla

(3, 2) = saunassaoloaikaraja ennen kuin hälytetään

(3, 3) = lämpötilaraja saunassaoloajalle

Annetuilla arvoilla näytti sitten tältä:

```
192.168.1.174|2390|smtp.dnainternet.net||
Mummon saunasensori|saunasensori@gmail.com|palaute@posionatkpalvelut.com|pasi@posio.eu|Saunassa on kuuma ja siellä on oltu kauan.
40|25|0:00:15|0:00:05|0:00:05
0:00:25|0:00:35|0:00:30|21|
```

Arvot ovat siis testausta varten, jotta saadaan testaus nopeaksi. Lähettävä palvelin on oltava *ISP*:n lähettävä *smtp*-postipalvelin, niin saamme postit lähtemään ilman autentikointia.

Nyt sitten saatoimme koodissamme viitata noihin arvoihin.

Sitten sama homma muutosten tallennuksille. Tälle oli järkevää tehdä erillinen tallennusnappi. Samoin tallennuksen yhteyteen oli hyvä rakentaa asetusten varmuuskopiointi.

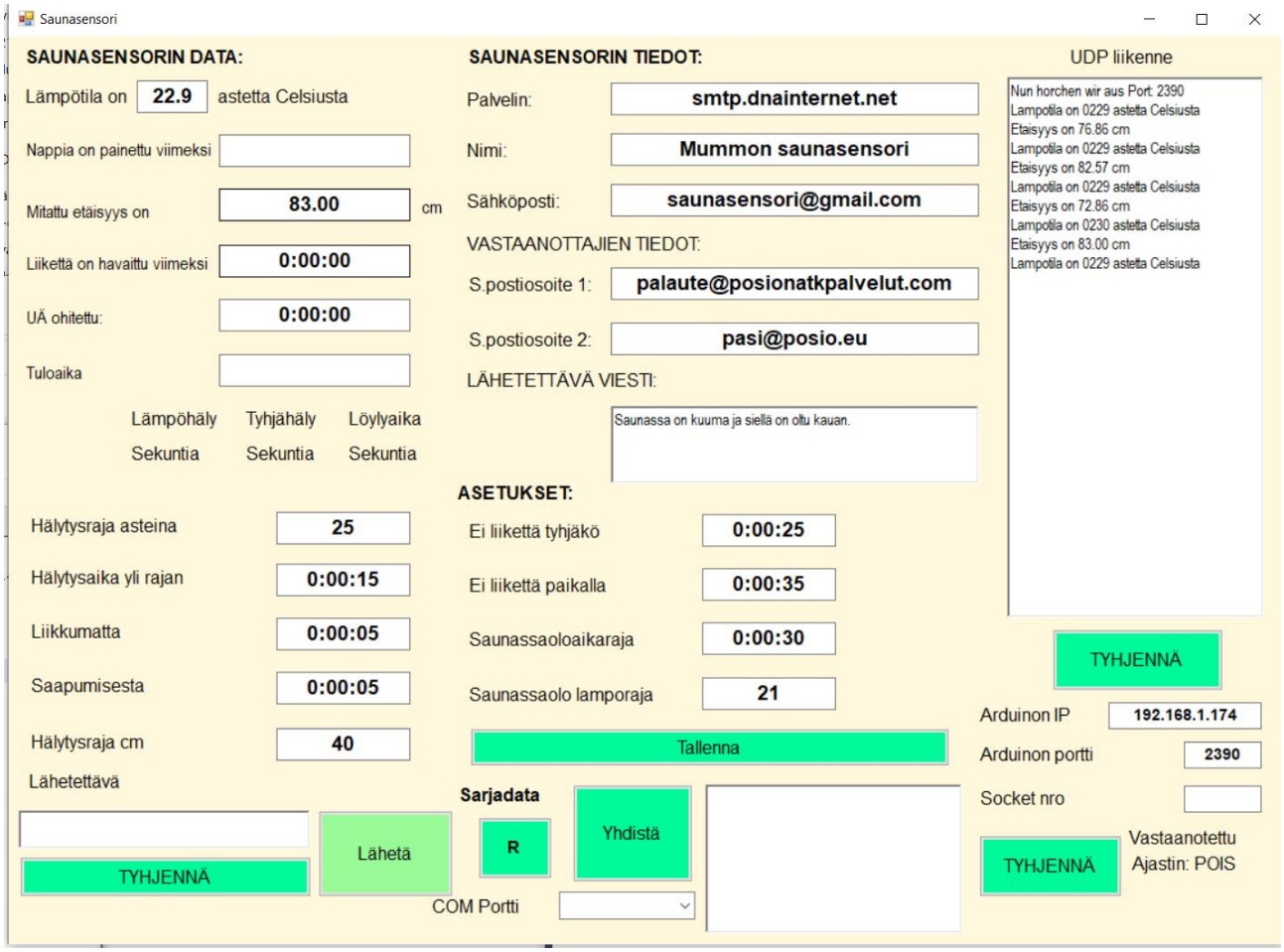
Tallennuksen koodi:

```
Private Sub tallennaBTN_Click(sender As System.Object, e As System.EventArgs) Handles tallennaBTN.Click

    tauluyht(0, 0) = ipTB.Text
    tauluyht(0, 1) = porttiTB.Text
    tauluyht(0, 2) = palvelinTB.Text
    tauluyht(1, 0) = ardnimiTB.Text
    tauluyht(1, 1) = arduinomeiliTB.Text
    tauluyht(1, 2) = vastmeili1TB.Text
    tauluyht(1, 3) = vastmeili2TB.Text
    tauluyht(1, 4) = viestiRTB.Text
    tauluyht(2, 0) = rajacmTB.Text
    tauluyht(2, 1) = rajalampoTB.Text
    tauluyht(2, 2) = halyaikaTB.Text
    tauluyht(2, 3) = liikkumattaTB.Text
    tauluyht(2, 4) = saapumisestaTB.Text
    tauluyht(3, 0) = AikaTyhjakoTB.Text
    tauluyht(3, 1) = EiLiikettaPaikallaTB.Text
    tauluyht(3, 2) = SaunassaOloAikaRajaTB.Text
    tauluyht(3, 3) = SaunassaOloLampoRajaTB.Text

    Dim luku As Integer
    luku = Int((10 * Rnd()) + 1) 'arvotaan varmuustiedoston nimi
    filuBu = "data\bu\asetukset" & luku & ".dat"
    Tallenna(tauluyht, filuAsetukset, riviyht)
    Tallenna(tauluyht, filuBu, riviyht)

    MessageBox.Show("Tallennus onnistui!", "Suoritettu")

End Sub
```

Ja lomakkeen lataukseen vastaavasti:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Lataa(filuAsetukset, tauluyht, riviyht, srkyht)
    ipTB.Text = tauluyht(0, 0)
    porttiTB.Text = tauluyht(0, 1)
    palvelinTB.Text = tauluyht(0, 2)
    ardnimiTB.Text = tauluyht(1, 0)
    arduinomeiliTB.Text = tauluyht(1, 1)
    vastmeili1TB.Text = tauluyht(1, 2)
    vastmeili2TB.Text = tauluyht(1, 3)
    viestiRTB.Text = tauluyht(1, 4)
    rajacmTB.Text = tauluyht(2, 0)
    rajalampoTB.Text = tauluyht(2, 1)
    halyaikaTB.Text = tauluyht(2, 2)
    liikkumattaTB.Text = tauluyht(2, 3)
    saapumisestaTB.Text = tauluyht(2, 4)
    AikaTyhjakoTB.Text = tauluyht(3, 0)
    EiLiikettaPaikallaTB.Text = tauluyht(3, 1)
    SaunassaOloAikaRajaTB.Text = tauluyht(3, 2)
    SaunassaOloLampoRajaTB.Text = tauluyht(3, 3)
    rajalampoAjastin.Enabled = False
    Timer1.Enabled = False
    populateCOMport()
    StartListener()
End Sub
```

Nyt näytti ohjelman startattua tältä:



15. Projektin tietoturvanäkökulma

Heikkouksia

Ongelma: **Arduinon** sarjaporttiliikenteelle ei tarvita tunnistautumista. Jos laitteeseen pääsee fyysisesti käsiksi, niin kaikki sarjaporttiliikenne on luettavissa.

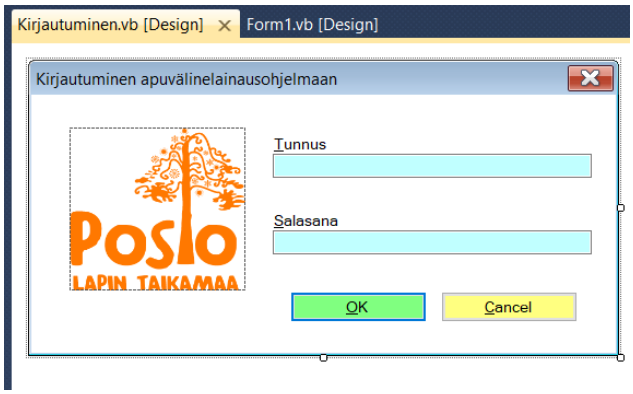
Ratkaisu: Jos siis halutaan tämä turvata, on fyysinen pääsy laitteeseen ja mahdollisen virta/datakaapelointiin estettävä muuten.

Ongelma: **Arduino**n *UDP*-liikenne ei ole tunnistautumisen takana. Jos lähiverkkoon on pääsy, on **Arduinoonkin**.

Ratkaisu: Lähiverkkoon pääsy estettävä vahvalla kirjautumissalasanalla ja protokollalla, verkkolaitteet fyysisesti turvaan.

Ongelma: PC:ssä ohjausohjelmaan pääsee ilman tunnistautumista.

Ratkaisu: Salataan ohjaustietokoneen levy ja laitetaan koneeseen kirjautuminen vahvan salasanan taakse. Ja tai luodaan ohjelmaan kirjautuminen. Tämä olisi ollut helppo implementoida Pasin aiemmista tuotoksista nappaamalla kirjautumislomake ja koodi vaikkapa hänen aikoinaan tekemästä apuvälineohjelmistostaan.




```
Private Sub OK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK.Click

    Dim passuok As Boolean

    passuok = True

    If Tunnus.Text = "TUNNUS" And Passu.Text = "PASSU" Then
        passuok = True
    Else
        MessageBox.Show("Tunnus tai salasana väärä", "Käyttäjää ei tunnistettu")
        Application.Exit()
    End If
    Me.Close()
End Sub
```

```
Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel.Click
    MessageBox.Show("Perutaan kirjautuminen...", "Poistuminen")
    Me.Close()
    Application.Exit()
End Sub
```

Ongelma: Asetukset selkokielisinä *plain text*-tiedostossa.

Ratkaisu: Otetaan tiedoston salaus käyttöön. Pasilla jo valmis koodi tähän ja *Lataa* ja *Tallenna*-funktioissa parametri valmiina (käytössä muualla).

Vahvuudet

UDP-liikenne on vahvasti salattua, jos langaton liikenne on (kuten voisi olettaa).

Sähköpostiliikenne on vahvasti salattua (*SSL-suojaus*):

```
smtp.EnableSsl = True
```

Toiminnoissa ei hyödynnetä pilvipalveluita (**Arduino IoT Cloud**), joten data ei siltä osin mene kolmansien osapuolten kautta.

Ohjelma ei kerää henkilökohtaisia terveystietoja, eikä ole lääkinnällinen laite.

16. Mitä voisi kehittää, lisätä, muuttaa jatkossa

Arduinoon on asennettu ja koodattu ja PC-puolelle koodattuna painonappi, mutta sille ei ole määriteltynä funktiota. Sen voisi ottaa käyttöön vaikka löylyhuoneen seurannan ohituspainikkeena tai manuaalisena hälytyksenä. Voisi vaikka laittaa optiona ja asiakas saa valita käyttötarkoituksen.

Sensorien soveltuvuuteen kuumiin ja kosteisiin tiloihin voisi kiinnittää huomiota.

Laitteiden kotelointi ja suojaus.

Löylyhuoneen olosuhteiden rasitusten päättelyä saunojalle voisi kehittää lisäämällä ilmankosteussensorin ja luomalla monimutkaisemmat kaavat laskemaan lämmön ja kosteuden yhteisvaikutuksia. Kuuma ja kostea ilma on lämpörasitukseltaan paljon suurempaa kuin kuivassa ilmassa. Tämä olisi suht helposti lisättävissä (optiona, jos kaupallistetaan?).

Ohjelman voisi forkata **Androidille** ja **iOSille**, jolloin sitä voitaisiin käyttää myös älypuhelimilla ja tableteilla.

Arduinon käyttö internetin yli. Onnistuisi porttiohjauksella tai **Arduinon** julkisella IP:llä, mutta tämä toisi haasteita tietoturvaan, jotka olisi huolellisesti huomioitava.

Ultraäänisensorin vaihto tarkempaan ratkaisuun, esimerkiksi mikroaaltosensoriin. Tämä olisi projektissa jo toteutettu, mutta UÄ valikoitui ratkaisuksi, koska tämä sensori oli käsillä ja mikroaaltosensori ei. Ja halusimme tehdä oikean paketin, ei virtuaalista. Kun siis on lusikalla annettu, ei voi kauhalla vaatia. Sitten seuraavassa elämäss... ei kun projektissa sitten.

17. Loppumietteet

Saunasensori-projekti onnistui oikeastaan yli odotusten. Moni asia askarrutti alunperin, että onnistummeko tekemään ylipäätään, mutta kaikki vaikeudet tulivat voitetuiksi, mitään ei jäänyt hampaankoloon.

Projekti sytytti kipinän syvällisempäänkin mikrokontrolleriharrastukseen ja hankittuna onkin jo useamman sadan euron arvosta lisääkamaa omiin projekteihin.

Projektilla saattaisi olla jopa kaupallista käyttöä oikeiden suhteiden kautta, koska itse komponentit maksavat muutaman kympin, mutta kaupallistettuna kokonaisuudesta olisi mahdollista saada liikevaihtoa moninkertaisesti ja ehkä jopa palveluita olisi mahdollista liittää mukaan, jolloin olisi mahdollista saada

jatkuvaa tuloa per asiakas. Joskin pääpaino tässä kuumalla saunalla, joita ei monessa maassa ole, mutta luovasti ajatellen samaa ideaa olisi mahdollista sovittaa myös muihin olosuhteisiin, miksei vaikkapa pakastamoihin, ettei kukaan palellu kuoliaaksi.

Projekti osoittaa asiantuntijuuden arvon. Halvoille komponenteille on mahdollista saada huomattavaa lisäarvoa, kun niitä yhdistellään ja käytetään luovaa insinööritaitoa käyttäen ja mukana on monialaisen substanssin vaatimuksia.

Kerrassaan mielenkiintoinen ja mukanaan vievä kokonaisuus kaiken kaikkiaan, tosin panostus pelkkään harjoitustyöhön ajateltuna oli rankasti ylimitoitettu. Mutta oppi koskaan ojaan kaada ja vaikka tieto lisääkin tuskaa, niin kuten aiemmin sanottua, opittua on jo aikomus hyödyntää uusin innovaatioin.

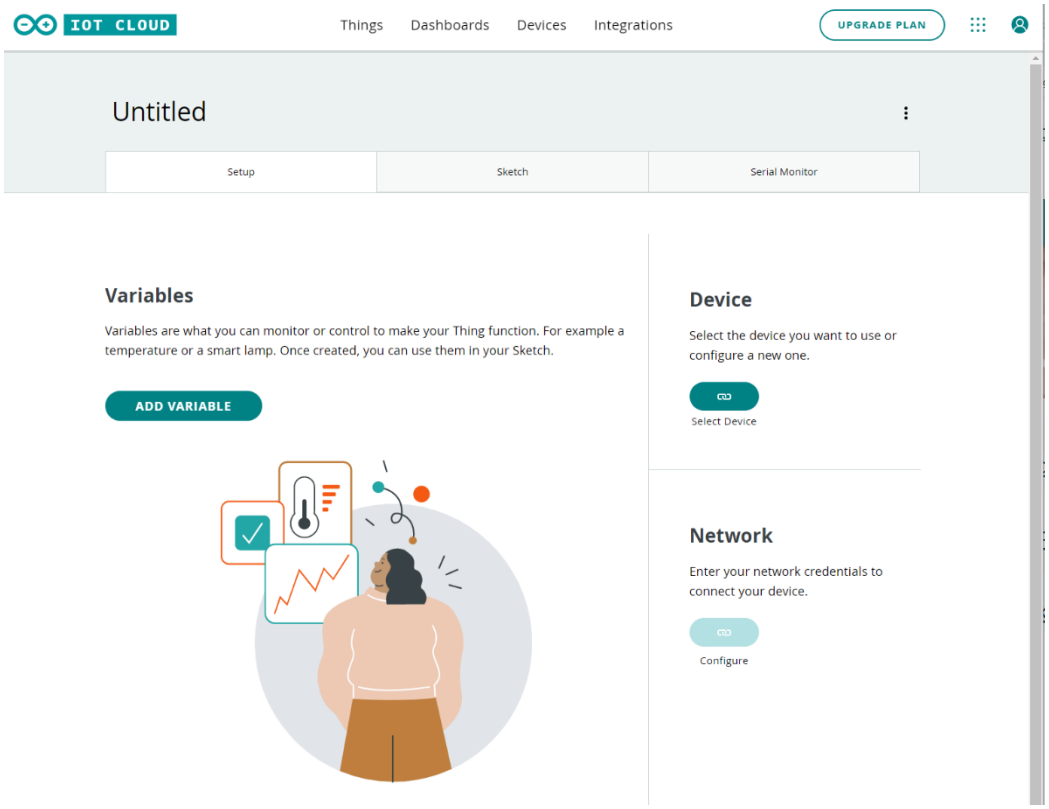
LIITTEET

1. Toimintojen teko ja hyödyntäminen pilvipalvelussa

Arduinon pilvipalveluun pääsee osoitteesta:

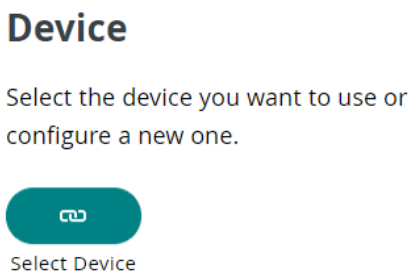
https://create.arduino.cc/iot/things

Alla pilviseikkailumme selostus. Pilviosuutta emme lopullisessa toteutuksessa hyödyntäneet lainkaan, joten siksi siirsimme tämän osuuden raportin loppuun.

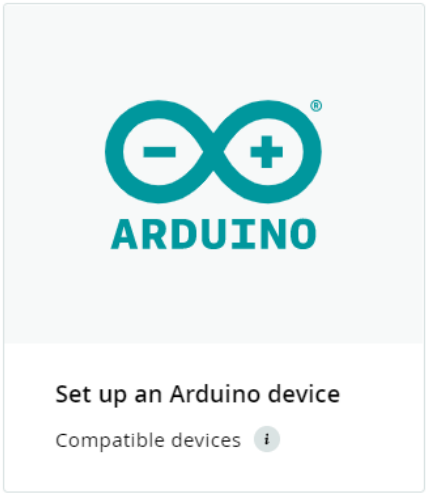


Laitteen lisääminen

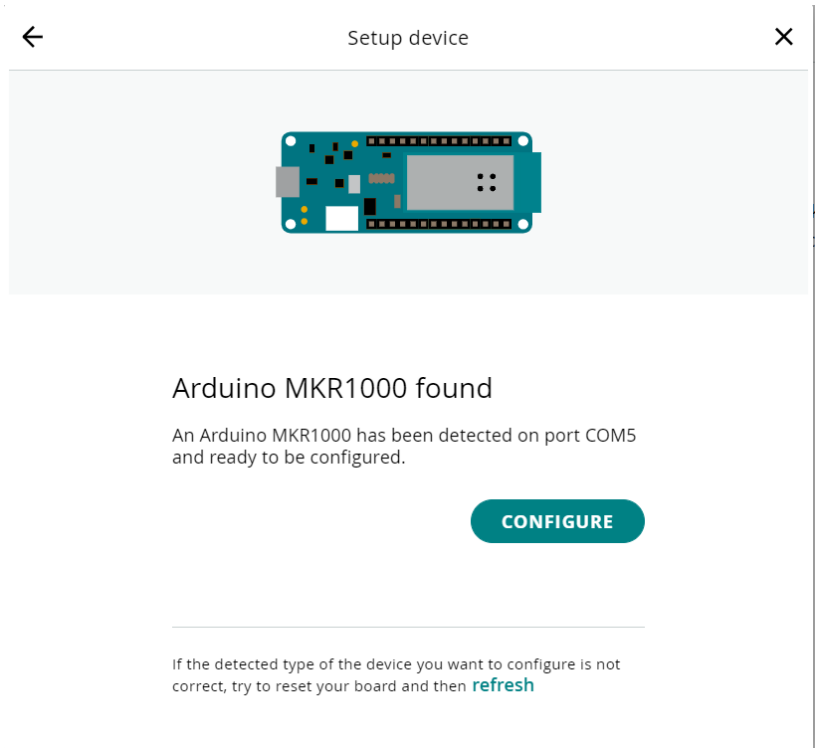
Lisätäänpä ensin vaikka **Arduinomme** kohdassa *Device,* klikataan siis *Select device*



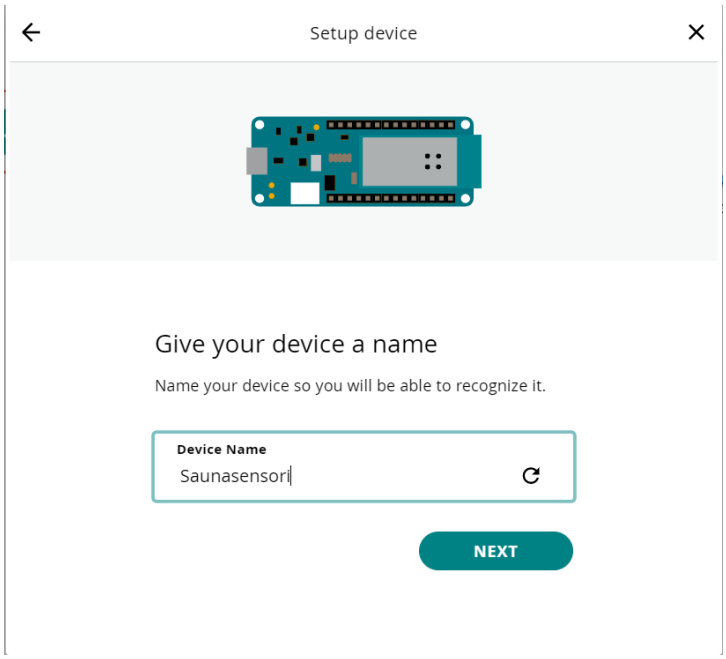
Valitaan *ARDUINO*



Nyt jos laite ei olisi kiinni tietokoneessa, niin se pitäisi lisätä. Mutta meillä se oli jo, joten hetken päästä tuli ilmoitus:

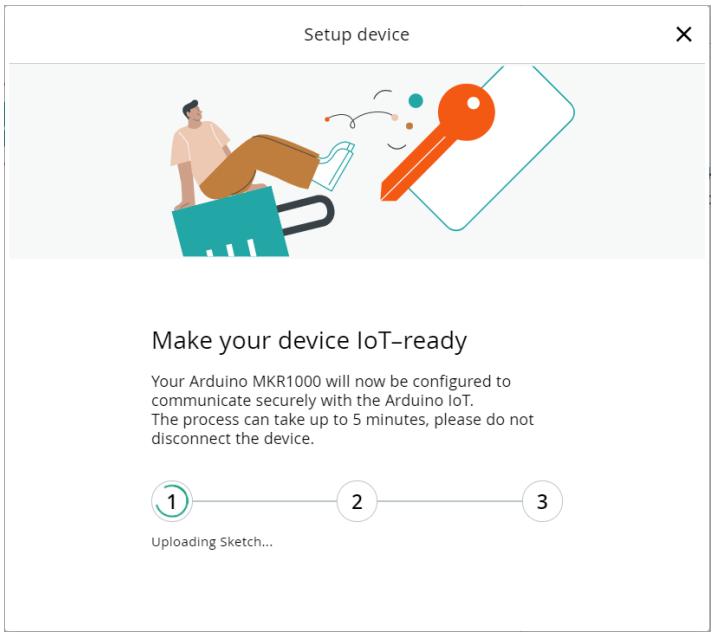


Huomatkaa, nyt porttinumero onkin nyt viisi (COM5). Klikataan *CONFIGURE*



Nimetään **Arduinomme Saunasensoriksi** ja klikataan *NEXT*

Nyt asennusvelho määrittelee erinäisiä asioita:



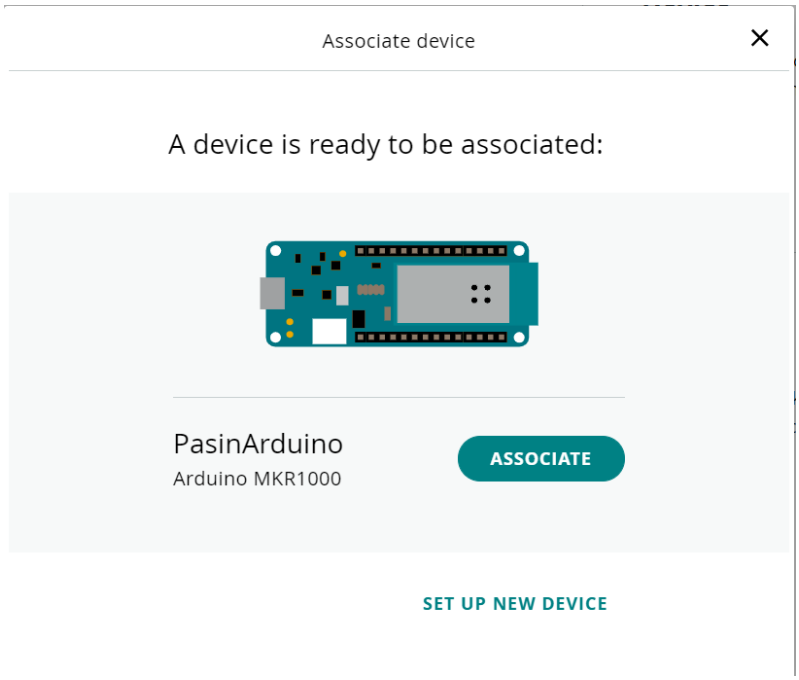
Ja tähän se sitten tökkäsi ja pahasti. Emme saaneet **Arduinoamme** asennettua pilveen, emme sitten millään.

Ja emme olleet ongelmamme kanssa yksin. Tässä yksi esimerkki keskusteluketjusta, jossa käsitellään samaa ongelma kuin meillä.

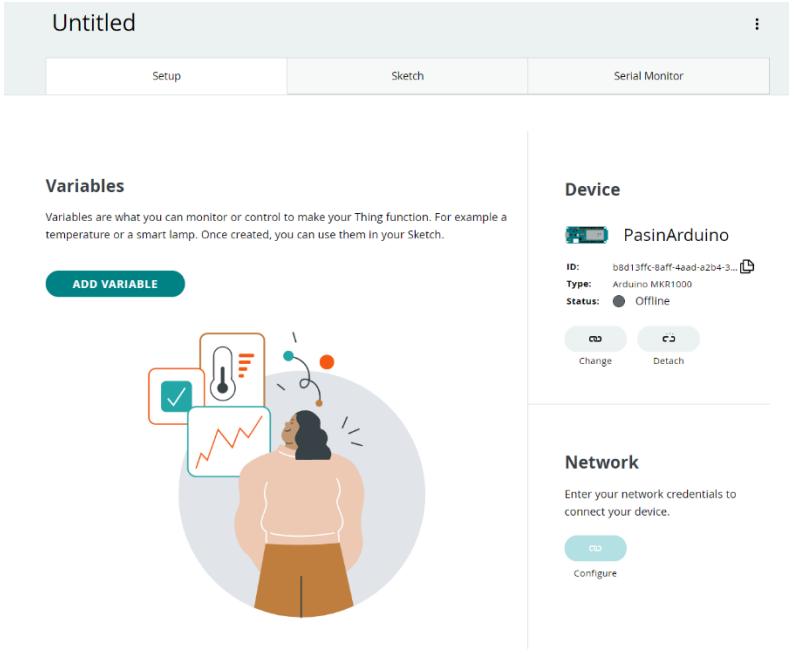
https://forum.arduino.cc/index.php?topic=699677.0

Onneksi IT-alalla on oppinut kärsivälliseksi ja lukuisten yritysten jälkeen laite lopulta suostui asentumaan. Tässä oli jotain tekemistä sillä, että kun yritti aina heti perään uudelleen, niin asennus sattui lopulta jäämään sopivasti johonkin välitilaan, joka hyväksyi jatkamisen ja läpimenon. Tämä arvaus tulee siitä, että tietokoneen kaiuttimista kuuli koko ajan merkkiääniä kuinka eri asennusyritysten ja vaiheiden aikana USB-laitetta irtikytkettiin ja kytkettiin toistuvasti, sekä se että laite näkyi aina hetken aikaa oikein pilvihallinnassa *Devices*-kohdan alla.

Niin tai näin, lopultakin **Arduinomme** oli pilvihallittavissa. Tai siis se piti ensiksi valita testiimme pilvihallinnassa. Nimi sattui sillä onnistuneella yrityksellä olemaan **Saunasensorin** sijasta **PasinArduino**, mutta tätä ei voi muuttaa kuin poistamalla laite ja asentamalla se uudelleen ja siihen lottoon emme enää halunneet uudestaan. Klikkasimme siis *ASSOCIATE*

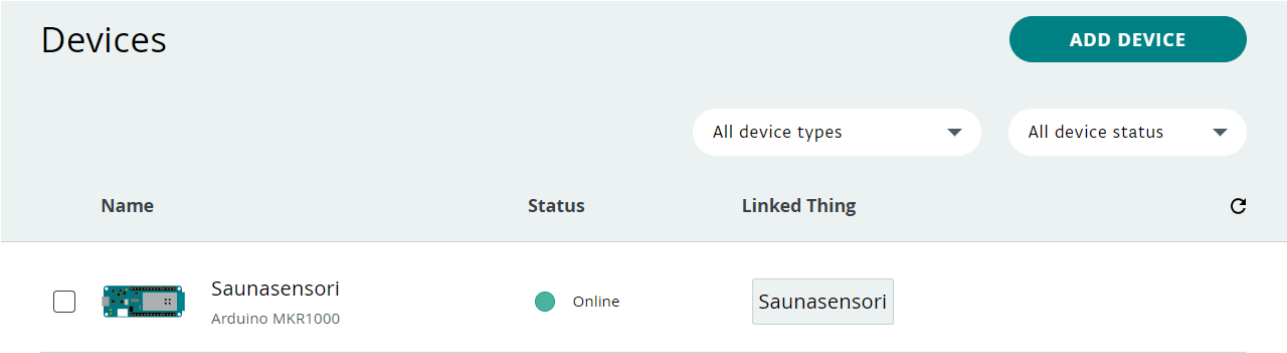


Ja nyt laite näkyi valittuna kuten kuuluu:



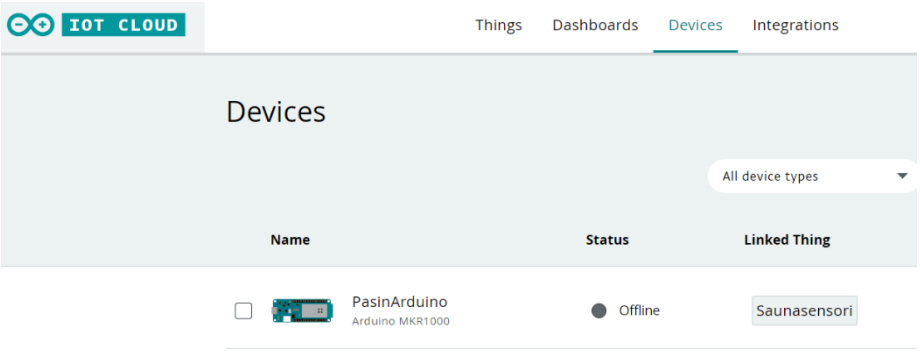
Nyt kun tarkemmin katsotaan, niin huomataan ettei se meidän asennus tainnut sittenkään onnistua. Statuksena lukee *Offline*. Tässä sitten taas jumpattiin tovi ja yhden kerran hetkellisesti laite kävi linjalla. Lopulta kuitenkaan ei auttanut muu kuin koettaa asentaa **Arduino** uudelleen huolimatta siitä, että tuo aiempi saattaisi kadota.

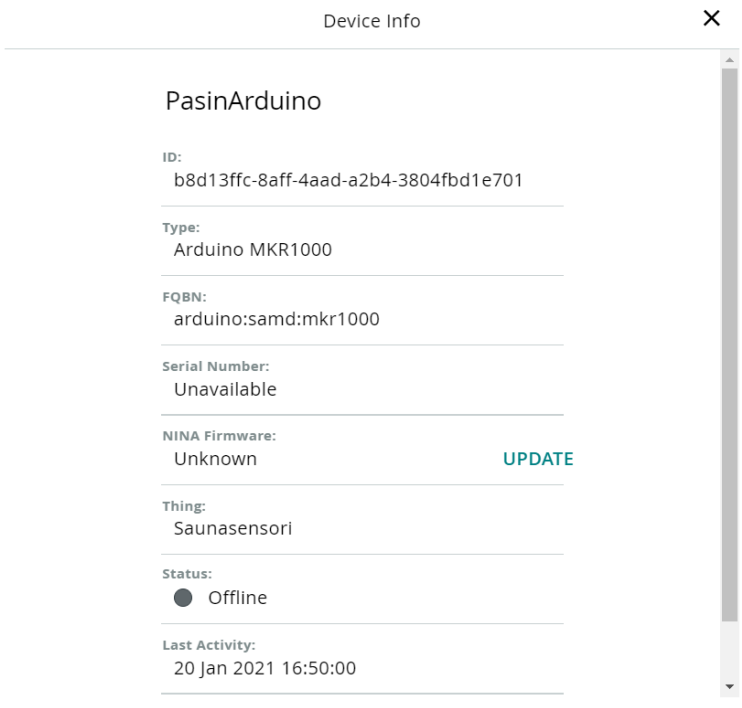
Nyt asennus meni onneksi heti läpi ja laite pysyi linjalla. Määrittelimme nimenkin tässä välissä halutusti eli **Saunasensorista** taas puhutaan.



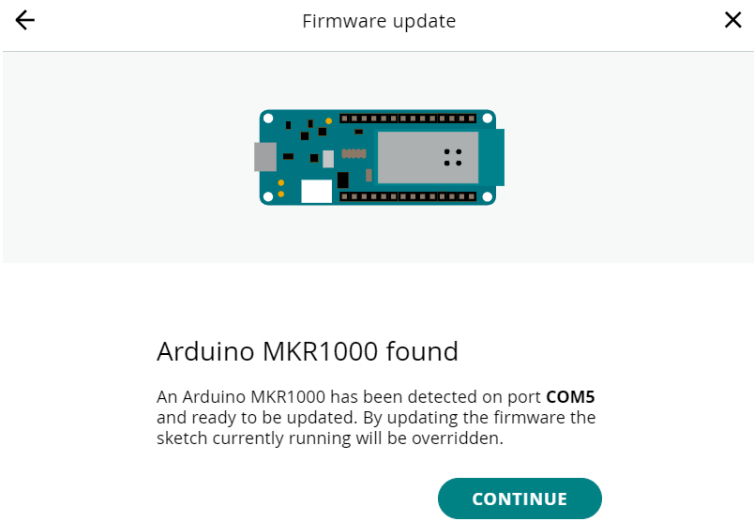
Laiteohjelmiston päivittäminen

Tätä ennen päätimme päivittää **Arduinomme** laiteohjelmiston pilvipalvelun avulla. Tämä onnistuu *Devices*-kohdan alta laitettamme klikkaamalla ja valitsemalla *Update*.



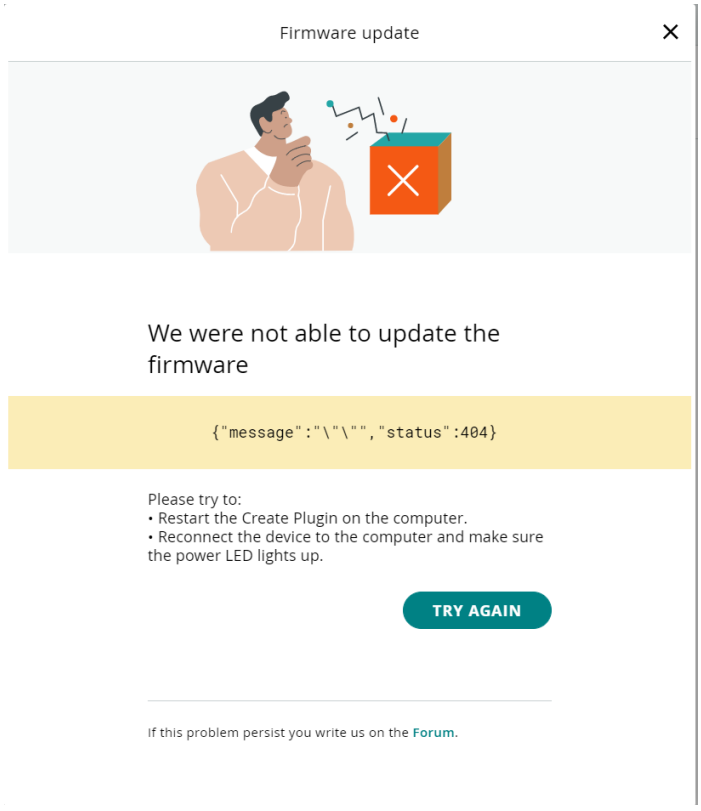


Tämän jälkeen tulee joko kehotus kytkeä laite tai sitten laite löydetään ja ilmoitetaan siitä:



Klikataan *CONTINUE*

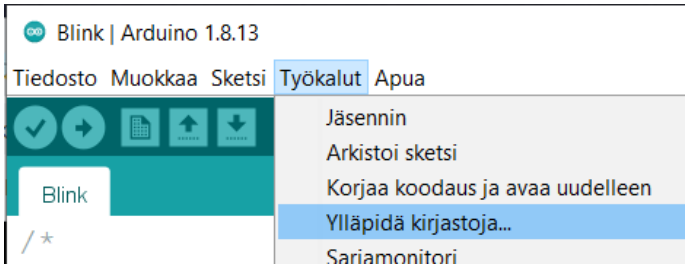
Ja taas törmäsimme ongelmiin:

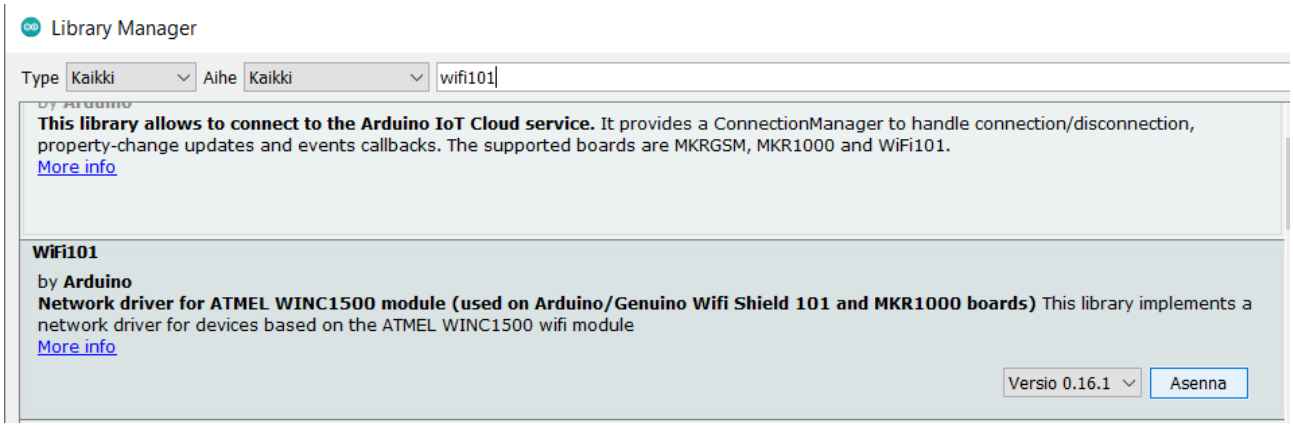


Vaikuttaa kovasti samalta kuin se aiempi kytkemisongelma. No päivitetään laiteohjelmisto sitten eri kautta, eli *IDE*:n avulla.

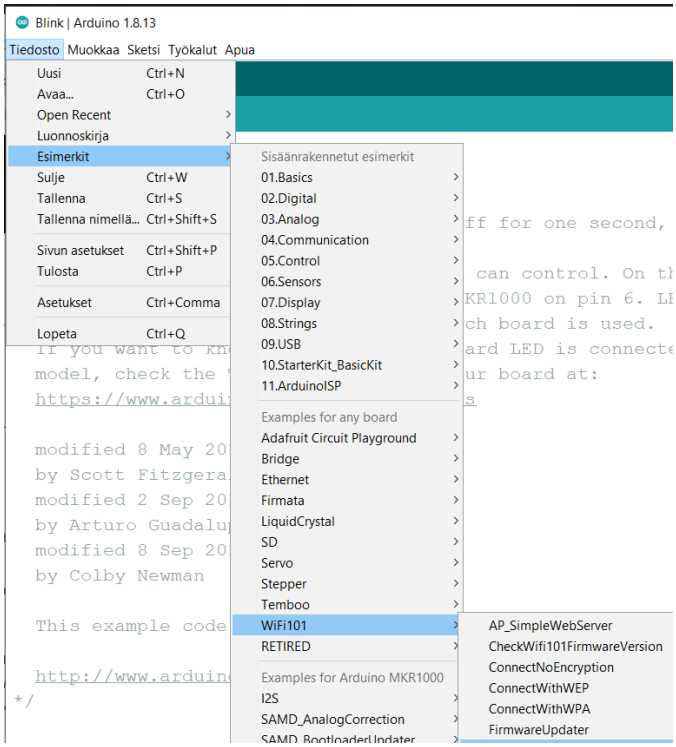
https://learn.mansteri.com/workshops/iout/mkr1000-firmware-update-and-adding-ssl-certificates/

Mennään ohjeen mukaan ja tarkistetaan ensin, että viimeisin *WiFi101* on asennettuna. Löytyy kirjastojen ylläpidosta:

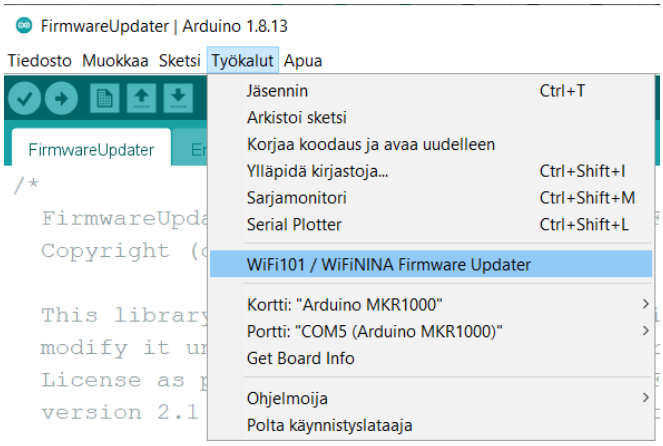


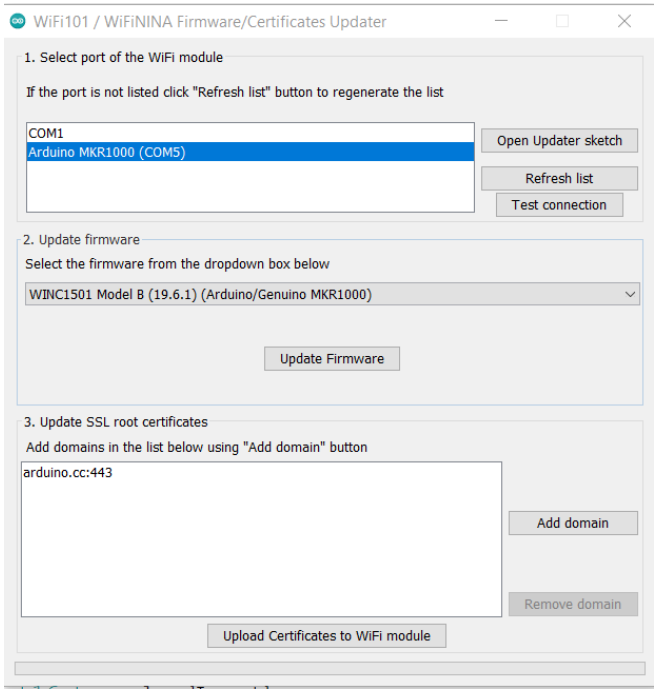


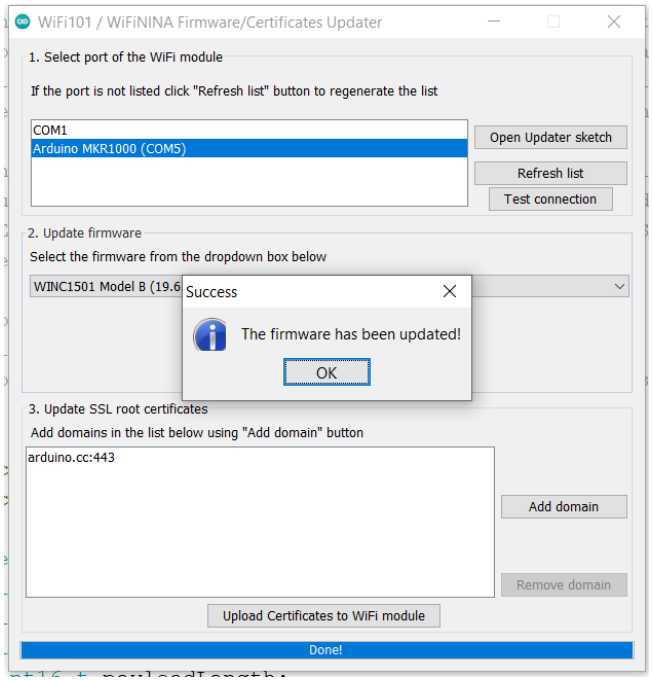
Asensimme siis ensin *WiFi101*:n ja sitten menimme *Esimerkkien* alta löytyvään *WiFi101:een* ja sieltä valitsimme *FirmwareUpdaterin* ja latasimme sen laitteeseen.

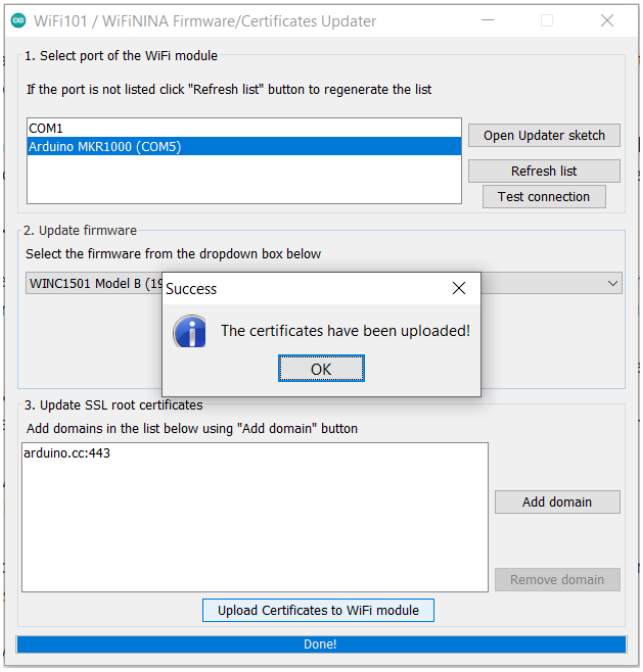


Sitten tarkistimme että oikea laite ja portti oli valittuna ja otimme *Työkaluista WiFi101 / WiFiNina Firmware Updaterin* käyttöön ja päivitimme sillä laiteohjelmiston, sekä sertifikaatit.


FirmwareUpdater | Arduino 1.8.13
Tiedosto Muokkaa Sketsi Työkalut Apua
Jäsennin Ctrl+T
Arkistoi sketsi
Korjaa koodaus ja avaa uudelleen
Ylläpidä kirjastoja... Ctrl+Shift+I
Sarjamonitori Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Kortti: "Arduino MKR1000"
Portti: "COM5 (Arduino MKR1000)"
Get Board Info
Ohjelmoija
Polta käynnistyslataaja
FirmwareUpdater


WiFi101 / WiFiNINA Firmware/Certificates Updater
1. Select port of the WiFi module
If the port is not listed click "Refresh list" button to regenerate the list
COM1
Arduino MKR1000 (COM5)
Open Updater sketch
Refresh list
Test connection
2. Update firmware
Select the firmware from the dropdown box below
WINC1501 Model B (19.6.1) (Arduino/Genuino MKR1000)
Update Firmware
3. Update SSL root certificates
Add domains in the list below using "Add domain" button
arduino.cc:443
Add domain
Remove domain
Upload Certificates to WiFi module


WiFi101 / WiFiNINA Firmware/Certificates Updater
1. Select port of the WiFi module
If the port is not listed click "Refresh list" button to regenerate the list
COM1
Arduino MKR1000 (COM5)
Open Updater sketch
Refresh list
Test connection
2. Update firmware
Select the firmware from the dropdown box below
Success
The firmware has been updated!
OK
3. Update SSL root certificates
Add domains in the list below using "Add domain" button
arduino.cc:443
Add domain
Remove domain
Upload Certificates to WiFi module
Done!



Toimintojen rakentaminen

Tässä palvelussa seuraava toimenpide oli luoda *Thing*. Näitä ilmaislisenssissä saa olla vain yksi, mutta sen alle voi tehdä sitten lukuisia toimintoja.

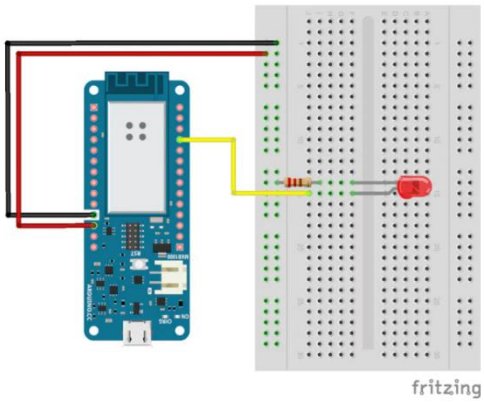
Seuraavassa selostuksessa hyödynsimme nyt sitten näitä ohjeita:

https://create.arduino.cc/projecthub/133030/iot-cloud-getting-started-c93255

Ohjeet olivat sikäli hieman puutteelliset, että ne oli tehty vanhemmalle pilviversiolle, joka sivulla kyllä kerrotaankin. Mutta koska sivun ohjeet olivat muuten hyvät ja selkeät, niin päätimme mennä niiden mukaan ja improvisoida missä tulisi tarvetta.

Ledin kytkeminen

Kytkimme ledin ohjeen mukaan



An LED connected to Digital Pin 2 of an Arduino MKR1000

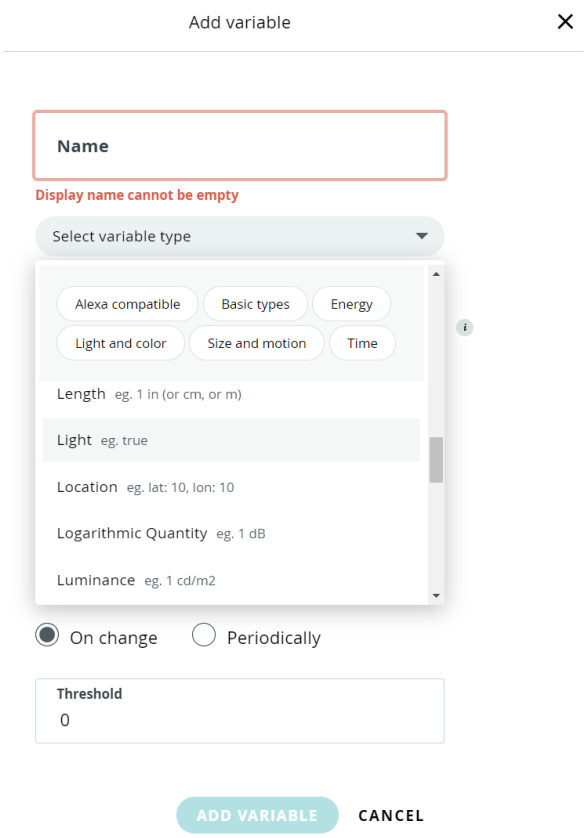
Eli tässä piti vetää kytkentäalustalle **Arduinolta** maakaapeli, virtakaapeli ja digitaalisesta pinni 2:sta kaapeli. Lisäksi piti laittaa oikean kokoinen vastus ja itse led. Tässä vaiheessa virtakaapelia ei vielä hyödynnetä, koska led saa virtansa pin 2:n kautta. Myöhemmin lisättävät komponentit tarvitsevat tuota kytkentäalustan jakamaa virtaa.

Thingin luominen

Loimme ohjatusti uuden thingin ja nimesimme sen **Saunasensoriksi**.

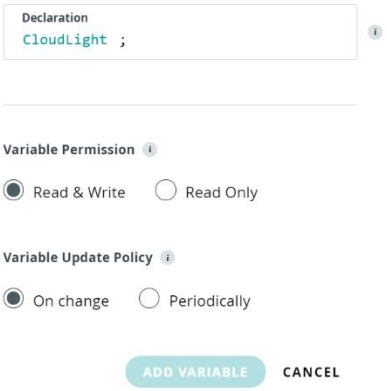
Toimintojen lisääminen

Sitten lähdimme lisäämään sinne ensimmäistä toimintoa *ADD* (lisää) -toiminnolla.



Annoimme sille ohjeen mukaisesti nimen light. Tällöin meidän ei tarvitse muuttaa sitä esimerkkikoodissamme.

Valitsimme kuvan mukaisesti tyypiksi *Light*



Tällöin valituksi tuli *Cloudlight*. Laitoimme sen luku- ja kirjoitustilaan, koska meidän on tarkoitus muuttaa arvoa pilvestä (päälle/pois). Jos pelkästään lukisimme arvoja laitteesta, niin silloin riittäisivät lukuoikeudet. Laitoimme vielä, että **Arduino** laittaa meille tiedon pilveen aina kun ledin tila muuttuu. Vaihtoehto olisi laittaa se tekemään ilmoitus halutuin väliajoin.

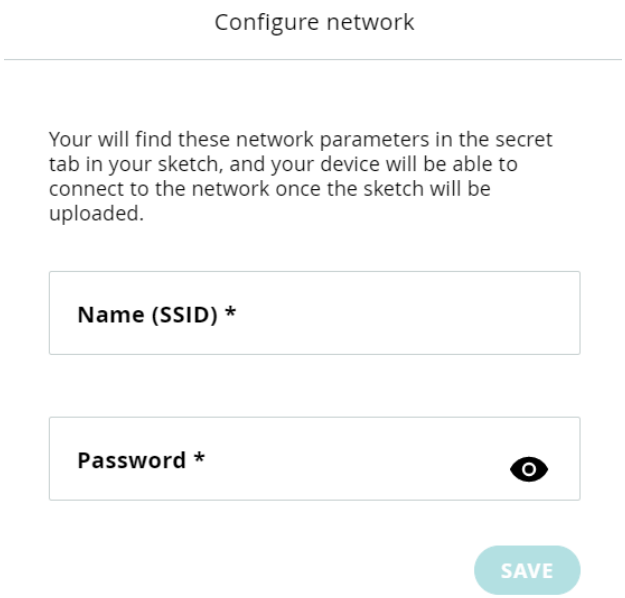
Lopuksi tallensimme toiminnon *ADD VARIABLE*-painikkeella.

Nyt meillä oli ensimmäinen toiminto tehtynä. Kytkimme sen (associate) vielä koskemaan tuota **Saunasensori**-laitettamme.

Verkkoasetusten syöttäminen

Nyt kannattaa huomata, että verkkoasetusten hallinta tuli tässä kohdin aktiiviseksi. Asetimme siis seuraavaksi verkkoasetukset kohdalleen, jotta saimme **Arduinomme** langattomasti lähiverkkoon ja sitä kautta internetiin ja pilvipalveluun ilman tietokoneen apua.

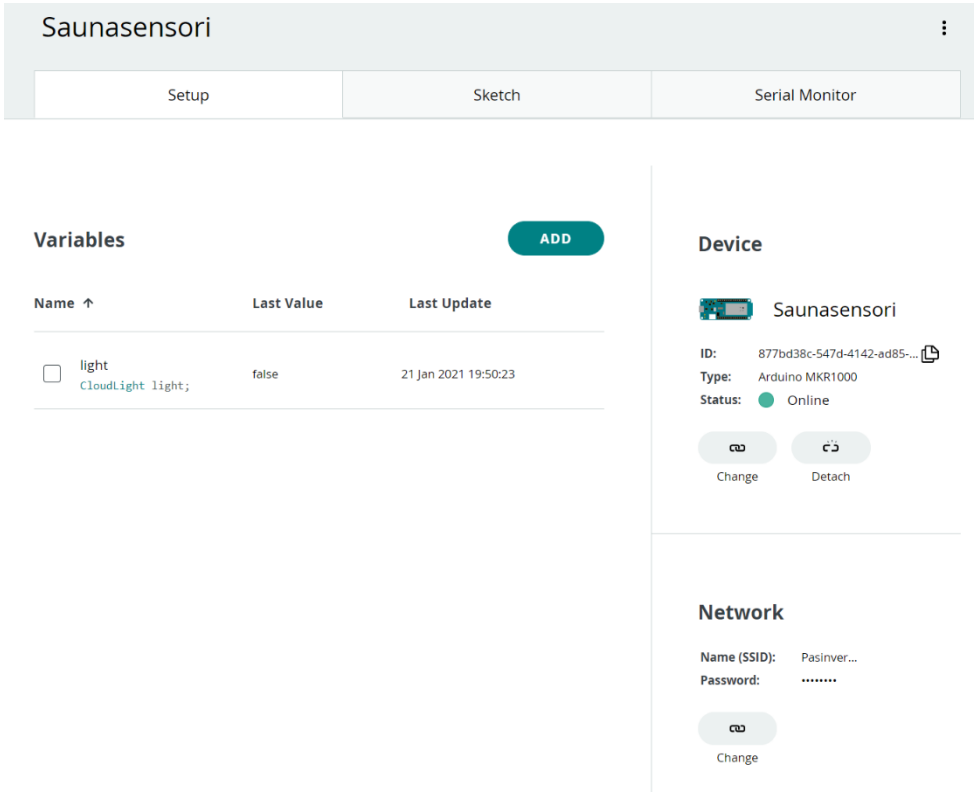
Verkkoasetusten hallintaikkuna näytti (klikkasimme *Configure*) tältä:



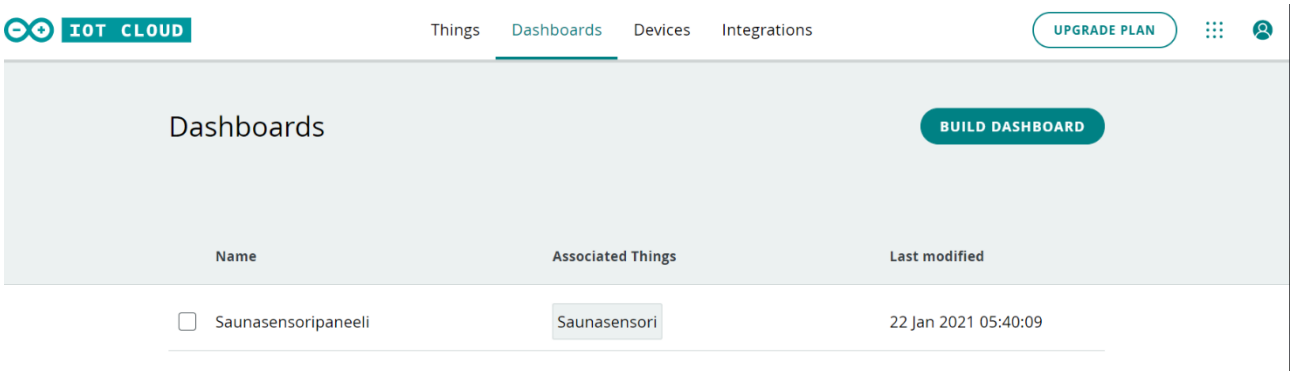
Laittelimme tuohon kelvolliset asetukset ja klikkasimme *SAVE*

Näkymät

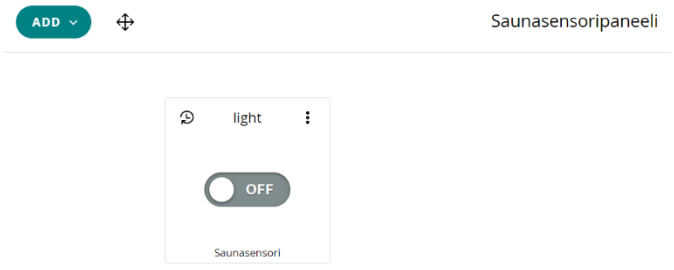
Nyt hallintapaneelissa näytti siis tältä:



Ja kun menimme *Dashboards* -välilehdelle niin näimme *Saunasensoripaneelimme*



Ja kun klikkasimme sen auki, niin näimme uuden hienon On/Off -säätimemme.



Koodin muokkaaminen Web-editorissa

Homma ei ollut kuitenkaan vielä selvä ledinkään osalta. Meidän piti seuraavaksi hieman lisäillä koodia, koska automaattisesti generoidussa ei ollut määriteltynä kaikkea tarvitsemaamme.

Otimme web-editoriin auki *Saunasensori_jan20a.ino* -tiedostoon ja teimme seuraavat lisäykset.

Määrittelimme että ledimme on kiinni **Arduinon** pinnissä numero 2.

```
#include "thingProp

#define LED_PIN 2

void setup() {
```

Määrittelimme että se pinni on ulostuloa varten.

```
  delay(1500);
  pinMode(LED_PIN, OUTPUT);

  // Defined in thingPropert
```

Ja lopuksi määrittelimme, että mitä tehdään kun led laitetaan päälle ja pois (tila muuttuu).

```
void onLightChange() {
    digitalWrite(LED_PIN, light);
    Serial.print("The light is ");
    if (light) {
        Serial.println("ON");
    } else {
        Serial.println("OFF");
    }
}
```

Sketsin lähettäminen Arduinoon

Nyt kun koodi oli valmis, niin lähetimme sen web-editorilla **Arduinoomme** klikkaamalla nuolipainiketta.

Saunasensori-koodi lähetettiin **Arduinoomme** ja vieläpä onnistuneesti:

```
Success: Done Uploading Saunasensori_jan20a
CPU reset.
readWord(addr=0)=0x20007ffc
readWord(addr=0xe000ed00)=0x410cc601
readWord(addr=0x41002018)=0x10010305
writeWord(addr=0xe000ed0c,value=0x5fa0004)
Saunasensori_jan20a uploaded successfully on board Arduino MKR1000 (COM5)
```

Testaaminen

Otimme **Arduinon** irti tietokoneesta ja kytkimme sen virtalähteeseen USB-kaapelilla. Nyt kaikki tiedonsiirto siihen menisi lähiverkon kautta. Otimme pilvipalvelussa **dashboardin** auki ja vaihtelimme kytkimen asentoa.

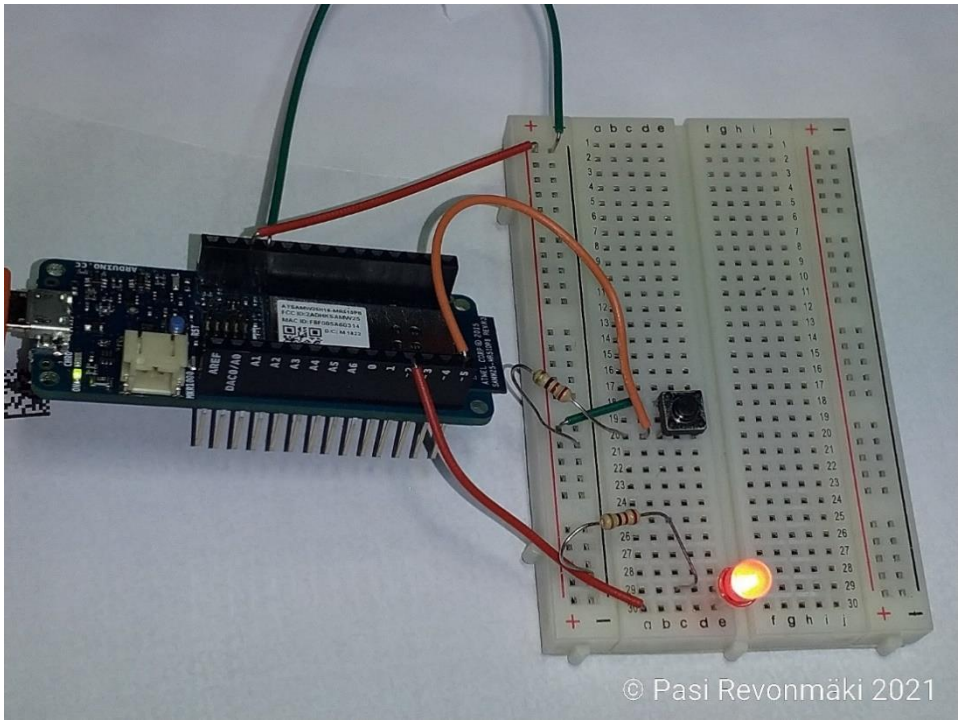
Totesimme ohjauksen toimivan, kuten oheinen video todistaa:

https://youtu.be/KX9FNKkabbM

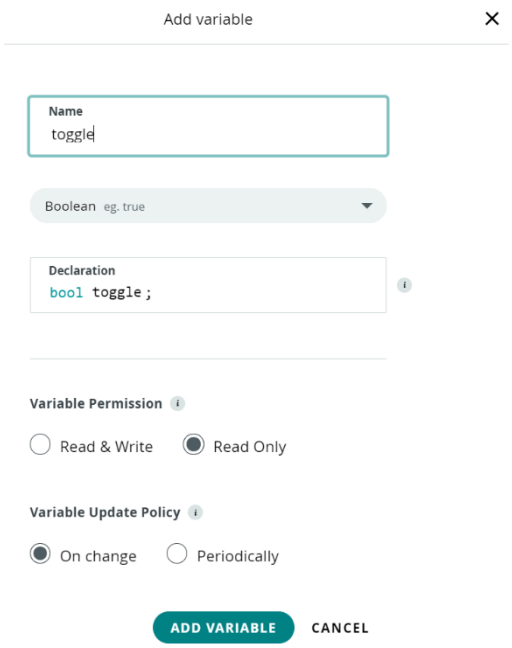
Painikkeen lisääminen

Seuraavaksi lisäsimme painikkeen. Esimerkkisivulla olevaa potentiometriä emme kokeilleet, koska tässä harjoitustyössä sitä ei tarvittu.

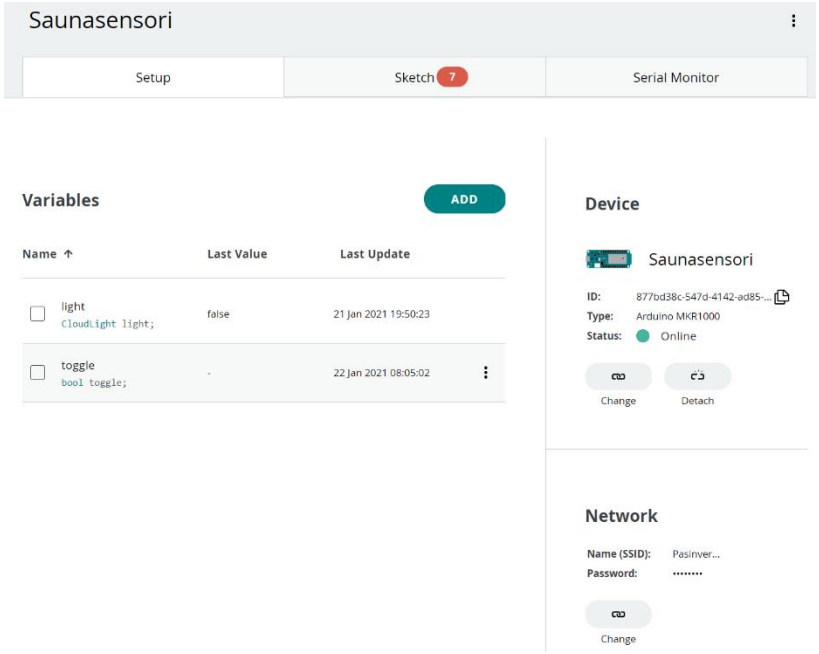
Kokoonpanomme näytti nyt tältä:



Seuraavaksi menimme *thingsiin* ja *ADD* -valinnalla lähdimme lisäämään *Saunasensorille* painike-toimintoa. Asettelimme ominaisuudet siihen näin:



Klikkasimme *ADD VARIABLE* ja nyt näytti tältä:



Seuraavaksi menimme taas web-editoriin ja lähdimme lisäämään koodia. Ensimmäiseksi kolme uutta riviä määrittelemään painikkeen pinninumero **Arduinossa** ja asettamaan painikkeen tila, jotta tila ilmoitetaan vain painettaessa, ei vapautettaessa:

```
#define LED_PIN 2
#define BUTTON_PIN 5
int btnState;
int btnPrevState = 0;
```

Ja nyt lopuksi laitellaan koodissa ensin input kohdalleen…

```
pinMode(LED_PIN, OUTPUT);
pinMode(BUTTON_PIN, INPUT);
```

…ja loput loopin loppuun:

```
void loop() {
  ArduinoCloud.update();
  // Your code here
  btnState = digitalRead(BUTTON_PIN);
if (btnPrevState == 0 && btnState == 1) {
 toggle = !toggle;
}
btnPrevState = btnState;

}
```

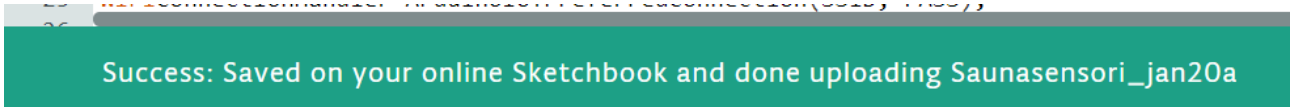
Ja taas homma meni seis. Tällä kertaa virheilmoitus ilmoitteli tallennettaessa, ettei tuota meidän *togglea* oltu määritelty. Ja tottahan se oli. Kun katsoimme *thingProperties.h* tiedostoa, niin siellä ei ollut jälkeäkään *togglesta,* vaikka se oli pilvessä joka paikassa kyllä määritelty *Saunasensoriin*. Automaattinen toiminto ei ollut toiminut, kuten ledin kohdalla.

Joten ei auttanut kuin lisätä tiedot tiedostoon käsin kolmeen eri kohtaan:

```
void onLightC
void loop();
```

```
CloudLight light;
bool toggle;

  ArduinoCloud.addProperty(toggle, READ, ON_CHANGE, loop);
}
```

Tämän jälkeen saatoimme ladata koodin **Arduinoomme**.



Testasimme ja molemmat, sekä ledin ohjaus, että nappi toimivat kuten videolta käy ilmi.

https://youtu.be/t9or3bEOwKs

1. Firmata

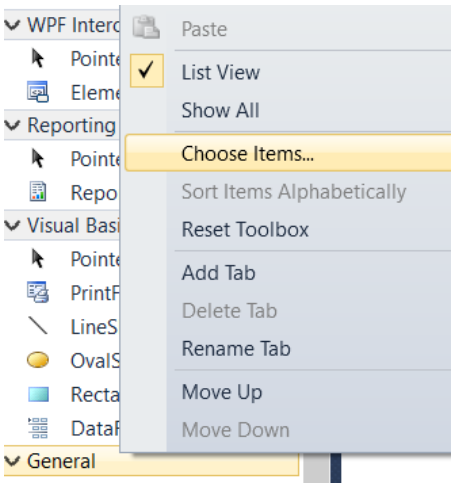
Muutama kirjasto **Arduinoon**, joiden avulla jatkossa voisi olla simppeliä lisäillä **Visual Studiossa** kontrollereita langatonta hyödyntämiseksi. Asensimme, mutta emme projektissa kuitenkaan hyödyntäneet. Alla selostus asennuksesta.

Työselostus

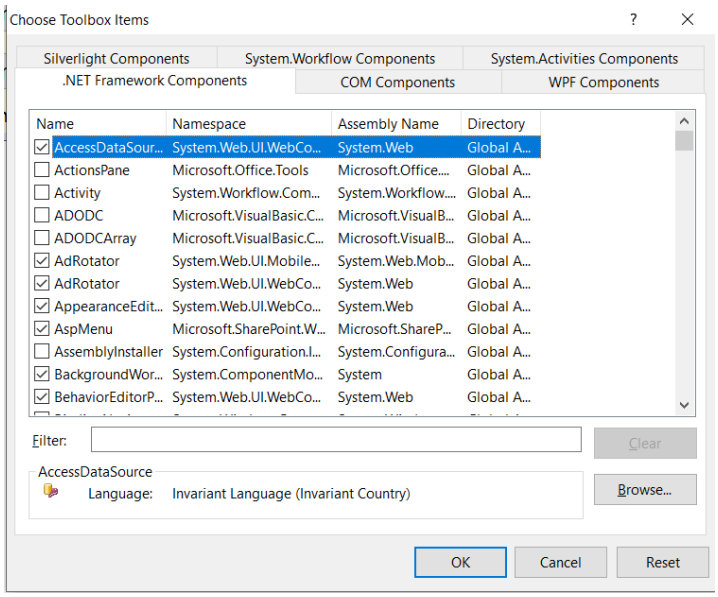
Tähän tarkoitukseen soveliaimmalta vaikutti *StandardFirmataWiFi*

https://github.com/firmata/arduino/tree/master/examples/StandardFirmataWiFi

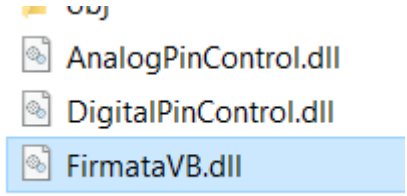
Lisäsimme ehdotetut kolme dll-tiedostoa projektikansioon ja seuraavaksi piti saada työkalut haluttuun valikkoon. Klikkasimme *Generalin* päällä hiiren toissijaisella ja valitsimme ensin *Choose Items…*



Ja sitten *Browse…*

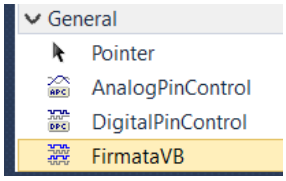


Ja sitten kävimme yksitellen hakemassa ne kolme tiedostoa…

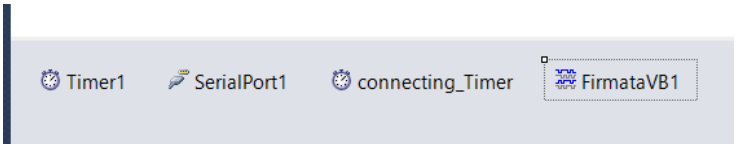


Ja lisäsimme tuohon listaan ja klikkasimme *OK*.

Nyt meillä oli työkaluissa valittavissa nuo kolme komponenttia.



Lisäsimme sen saman tien lomakkeelle:



Olisi sitten käytettävissä tarvittaessa, kun hätä on suurin.