MEHILÄISPESÄN YMPÄRISTÖSEURANTA ARDUINOILLA

Jenny Jalo

Sisällysluettelo

Projektin Esittely

Tämä projekti keskittyy mehiläisten hyvinvoinnin seurantaan hyödyntäen Arduino-mikrokontrolleria. Tavoitteena on mitata mehiläispesän lämpötilaa, kosteutta ja painoa reaaliajassa, mikä mahdollistaa tarkemman ymmärryksen mehiläisyhteisön terveydestä ja ympäristöolosuhteista. Projektin avulla voimme paremmin seurata mehiläisten tilaa ja reagoida nopeasti, jos ilmenee poikkeavia muutoksia.

Laitteisto:

Arduino-mikrokontrolleri (esim. Arduino Uno)

Lämpötila- ja kosteusanturi (esim. DHT22)

Painosensori (esim. HX711)

Langaton tiedonsiirto (esim. WiFi-moduuli)

Virtalähde (paristot tai virtalähde)

Tarvittavat liittimet, johdot ja kotelointi

Toimintaperiaate:

Lämpötila ja kosteus: DHT22-anturi mittaa mehiläispesän sisälämpötilaa ja ilmankosteutta. Anturin tiedot luetaan Arduino-mikrokontrollerilla, joka lähettää ne langattomasti eteenpäin esimerkiksi WiFin avulla.

Paino: HX711-painosensori asennetaan mehiläispesän alustan alle. Sensori mittaa pesän painoa ja antaa tämän tiedon Arduino-mikrokontrollerille. Painon muutokset voivat viitata mehiläisten poistumiseen tai saapumiseen pesään, hunajan kerääntymiseen tai muuhun aktiviteettiin.

Tiedonsiirto ja näyttö: Arduino kerää kaikki tiedot lämpötilasta, kosteudesta ja painosta. Nämä tiedot voidaan lähettää langattomasti pilvipalveluun tai tietokoneelle. Tarvittaessa voidaan myös liittää näyttö, joka näyttää reaaliaikaiset tiedot suoraan mehiläistarhurille.

Tietojen analysointi: Kerättyjä tietoja voidaan analysoida ja visualisoida. Esimerkiksi pilvipalvelussa voi seurata graafisesti mehiläispesän olosuhteita ja niiden muutoksia ajan mittaan. Tämä auttaa havaitsemaan mahdolliset ongelmat tai poikkeavuudet, kuten lämpötilan äkilliset muutokset.

Hyödyt:

Parempi mehiläisten terveyden seuranta: Reaaliaikainen ympäristön seuranta auttaa havaitsemaan stressitekijöitä, kuten liian korkeita lämpötiloja tai kosteuden vaihteluja, jotka voisivat vaikuttaa mehiläisten hyvinvointiin.

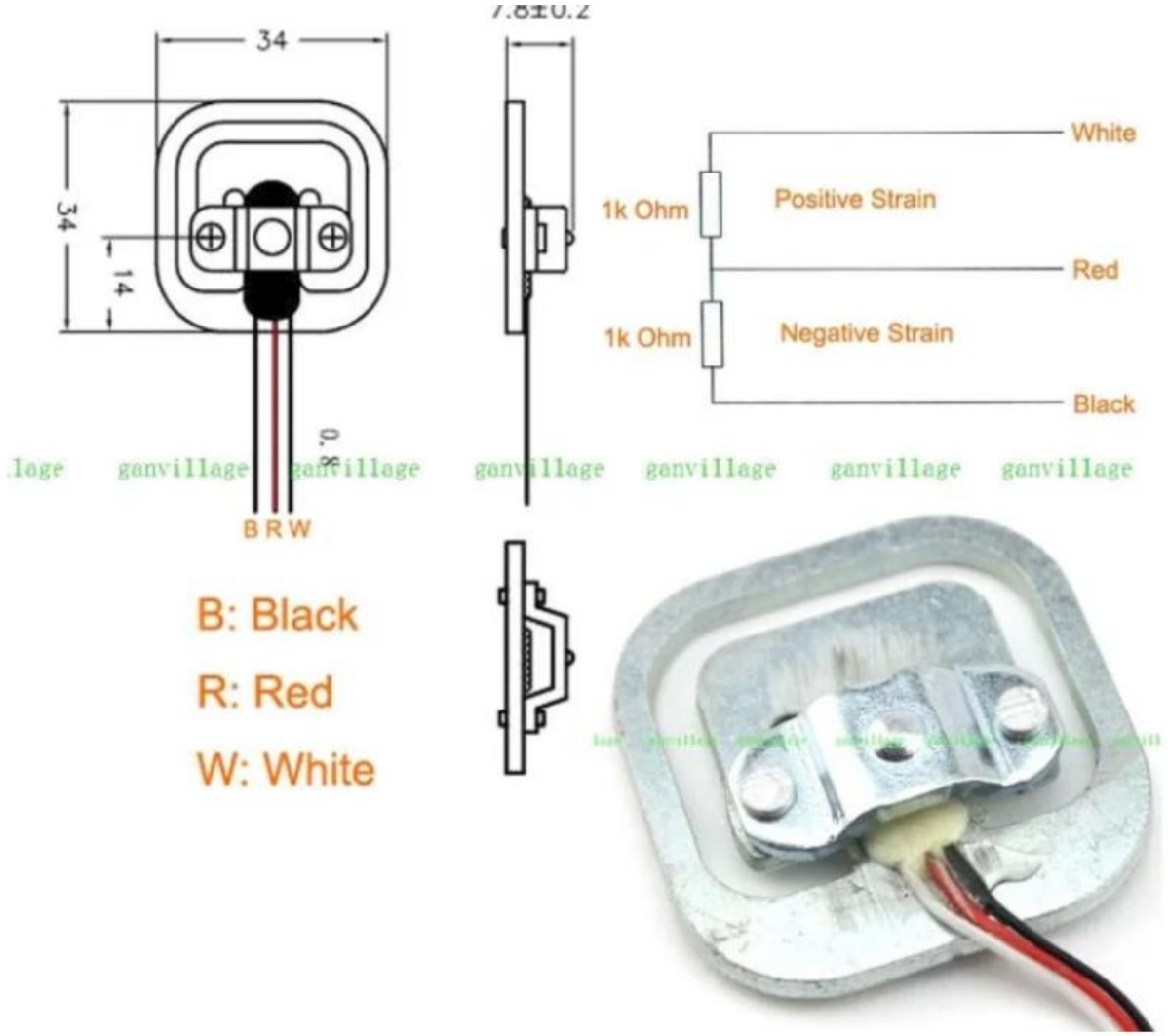
Tarkempi hunajan keruun ajoitus: Painon mittaus auttaa arvioimaan hunajan kerääntymistä ja mehiläispesän vahvuutta. Tämä voi ohjata mehiläistarhaajan päätöksiä hunajan sadonkorjuun ajoittamisessa.

Reaaliaikainen tiedonsiirto: Langaton tiedonsiirto mahdollistaa tiedon saamisen ilman tarvetta fyysisesti tarkastaa pesää jatkuvasti.

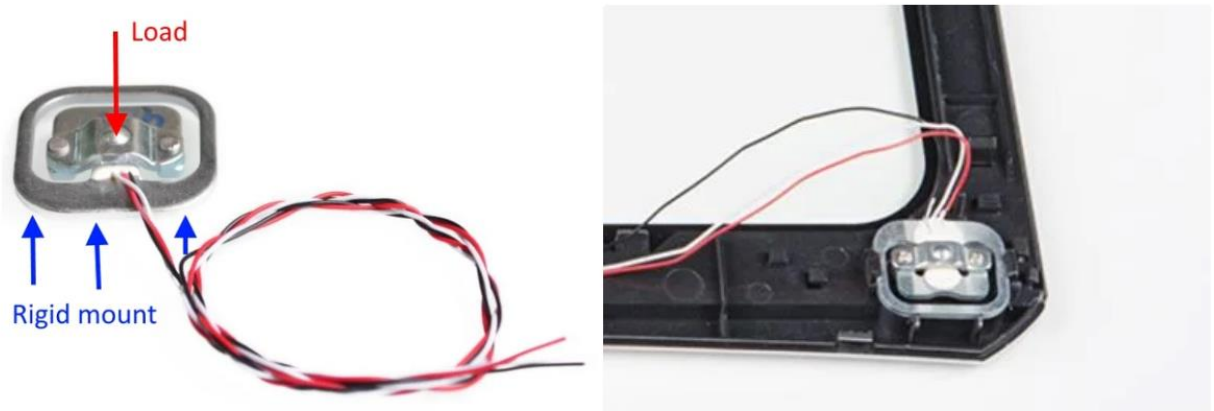
Johtopäätös:

Mehiläispesän ympäristöseurantaprojekti Arduino-mikrokontrollerilla on erinomainen tapa hyödyntää teknologiaa mehiläistarhauksen tehostamiseksi ja mehiläisten hyvinvoinnin parantamiseksi. Reaaliaikainen seuranta antaa arvokasta tietoa mehiläispesän tilasta ja ympäristöolosuhteista, mikä auttaa mehiläistarhaajaa tekemään parempia päätöksiä ja varmistamaan mehiläisten menestyksen.

Suunnittelu



Kuvassa loadcell jota tulemme käyttämään projektissa.



Kuvassa esitellään toimintaperiaatetta, load kuvaa painoa joka asetetaan vaa¨alle, rigid mount on laitteen runko. Oikeanpuoleinen kuva havainnollistaa, kuinka kyseinen loadcell löytyy myös kotona vaa'asta

DS18B20 on digitaalinen lämpötila-anturi, joka perustuu Dallas Semiconductorin (nykyään Maxim Integrated) 1-Wire-tekniikkaan. Se on suosittu anturi, joka tarjoaa tarkkoja ja tarkastettuja lämpötilamittauksia digitaalisessa muodossa yhden johdon kautta.

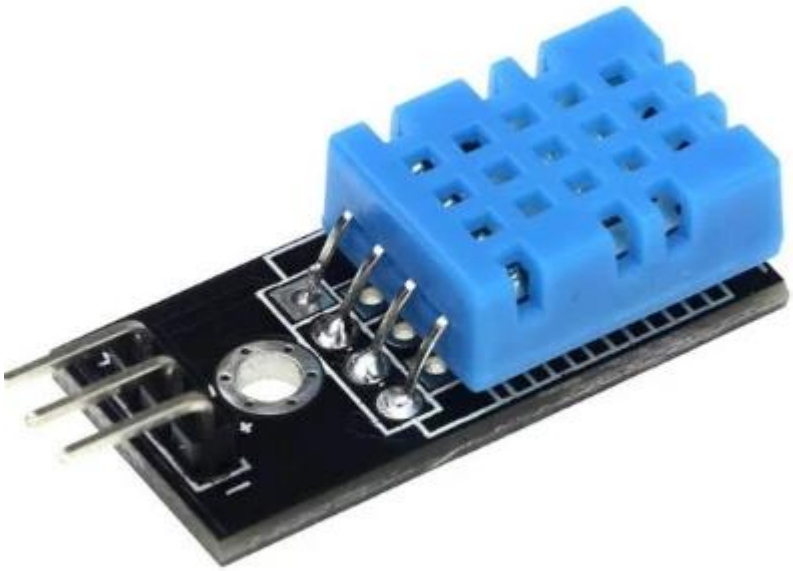
Yksijohdintekniikka: DS18B20 käyttää 1-Wire-väylää sekä tiedonsiirtoon että virtalähteenä, mikä tekee sen asentamisesta ja käytöstä yksinkertaista. Yksi lanka riittää sekä anturin että ohjaimen kytkemiseen.

Digitaalinen lähtö: Anturi tuottaa digitaalista lähtöä, mikä helpottaa mittausten tulkintaa ja käsittelyä mikro-ohjaimilla ja tietokoneilla.

Tarkkuus: DS18B20 on tarkka ja tarjoaa yleisesti ottaen korkeaa tarkkuutta lämpötilamittauksissa. Tarkkuus voi vaihdella eri versioiden välillä, mutta useimmat DS18B20-anturit ovat riittävän tarkkoja monenlaisiin sovelluksiin.

Ohjelmoitavuus: DS18B20-antureilla on erilaisia konfiguroitavia parametreja, kuten lämpötilan resoluutio, mikä antaa käyttäjälle joustavuutta mittauksen tarkkuuden ja nopeuden välillä.

Monipuolisuus: DS18B20-antureita on saatavana erilaisissa pakkausmuodoissa, mukaan lukien pinnikiinnitteiset ja vesitiiviit versiot. Tämä tekee niistä sopivia erilaisiin sovelluksiin, kuten sisätilojen ympäristöseurantaan tai ulkona olevien olosuhteiden mittaamiseen.



DHT11 on suosittu ja edullinen digitaalinen anturi, joka on suunniteltu mittaamaan ympäristön lämpötilaa ja kosteutta. Se tarjoaa helpon ja kätevän tavan seurata ympäristöolosuhteita erilaisissa sovelluksissa, kuten kodin automaatiossa, sääasemissa, kasvihuoneissa ja monissa muissa projekteissa. DHT11-anturin yksinkertaisuus ja suhteellisen edullinen hinta tekevät siitä houkuttelevan vaihtoehdon monille harrastajille ja elektroniikan harrastajille.

Ominaisuudet:

Lämpötila- ja kosteusmittaus: DHT11-anturi pystyy mittaamaan ympäristön lämpötilan ja suhteellisen ilmankosteuden. Se antaa digitaalisen lähtösignaalin, joka kertoo lämpötilan ja kosteuden arvot.

Tarkkuus: Anturi tarjoaa kohtuullista tarkkuutta peruskäyttöön. Lämpötilamittauksissa virhe voi olla noin ±2°C, kun taas kosteusmittauksissa virhe voi olla noin ±5%.

Digitaalinen lähtö: DHT11 tuottaa digitaalisen lähtösignaalin, mikä tekee sen helppokäyttöiseksi ja yhteensopivaksi mikrokontrollerien kanssa. Anturi lähettää tiedot yhden datapisteen kerrallaan, joten tietojen lukeminen mikrokontrollerilla vaatii vain yhden signaalipinnin.

Lähtömuoto: DHT11-anturin tuottama data on binäärimuodossa, joka on helppo lukea ja tulkita mikrokontrollerilla. Esimerkiksi lämpötila voidaan esittää Celsius-asteina ja kosteusprosentti yksinkertaisesti luvulla.

Virransyöttö: Anturi toimii yleisesti 3.3V tai 5V jännitteellä, mikä tekee siitä helposti integroitavan erilaisiin projekteihin. Se kuluttaa vain vähän virtaa, mikä on erityisen tärkeää, kun käytetään akkukäyttöisiä järjestelmiä.

DHT11-lämpötila- ja kosteusanturi tarjoaa edullisen ja helppokäyttöisen ratkaisun ympäristön seurantaan monissa erilaisissa sovelluksissa. Sen digitaalinen lähtö ja suhteellisen yksinkertainen liitäntä tekevät siitä houkuttelevan valinnan niin aloittelijoille kuin kokeneillekin harrastajille, jotka haluavat seurata ympäristöolosuhteita tarkasti ja vaivattomasti.

Arduino Ethernet Shield on laajennuskortti (shield), joka mahdollistaa Arduinon liittämisen Ethernet-verkkoon. Tämä tarjoaa mahdollisuuden luoda yhteyksiä internetiin, paikallisiin verkkojärjestelmiin ja palvelimiin. Ethernet Shield laajentaa Arduino-alustan käyttömahdollisuuksia verkkopohjaisten sovellusten luomiseen, tietojen lähettämiseen ja vastaanottamiseen sekä etäohjaukseen.

Ominaisuudet:

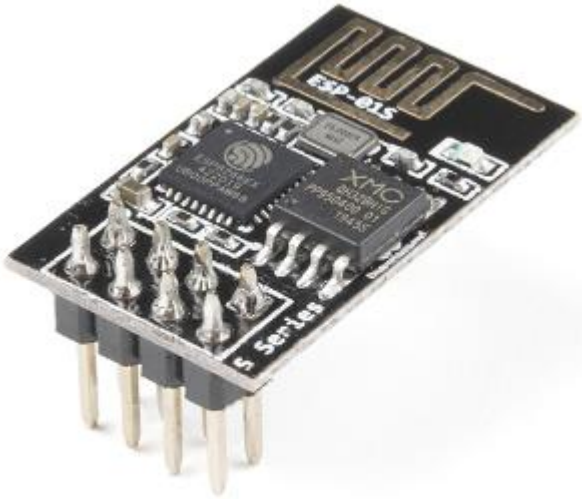
Ethernet-liitäntä: Arduino Ethernet Shield sisältää Ethernet-liitännän, joka mahdollistaa yhteyden muodostamisen lähiverkkoon (LAN) tai internetiin. Tätä voidaan käyttää tiedon siirtoon, etäohjaukseen tai etävalvontaan.

Mikrokontrollerin suojaus: Ethernet Shield suojaa Arduinon mikrokontrolleria ylikuormituksilta ja sähköisiltä häiriöiltä, jotka voivat aiheutua Ethernet-verkkoon liittymisestä.

Liitännät: Kortissa on liitännät sekä Ethernet-kaapelille että SD-kortille. Tämä mahdollistaa tiedonsiirron ohella myös datan tallentamisen SD-kortille.

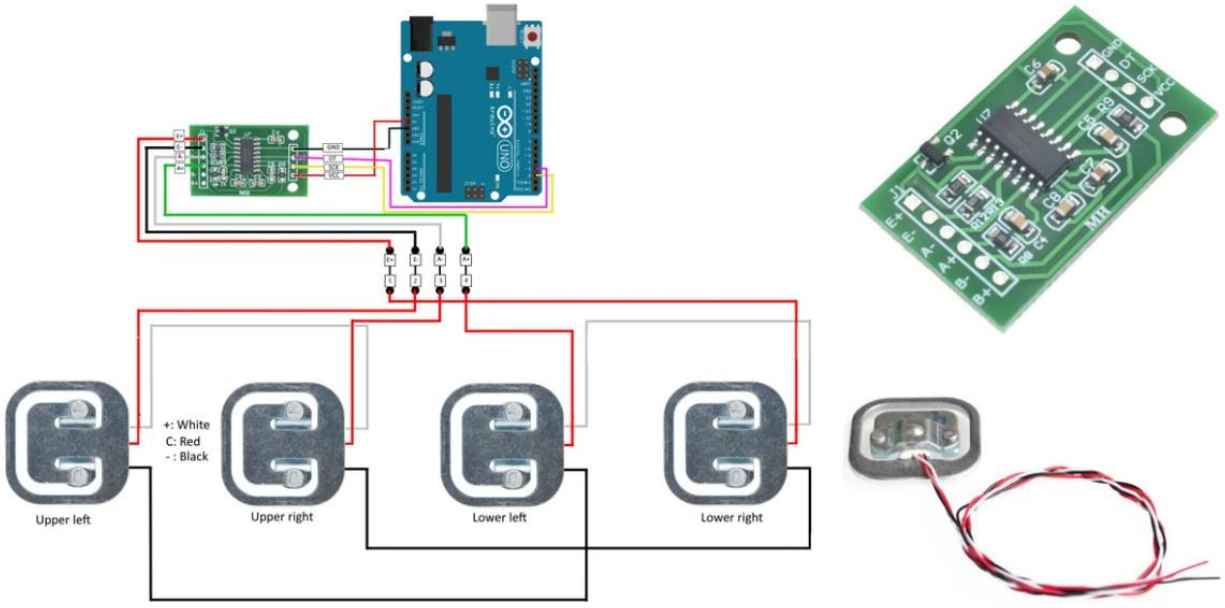
Ohjelmointi: Ethernet Shieldin kanssa käytetään Ethernet-biblioteekkia, joka tarjoaa valmiita toimintoja verkkoyhteyksien hallintaan. Käyttämällä tätä kirjastoa, voit lähettää HTTP-pyyntöjä, avata socket-yhteyksiä ja kommunikoida muiden verkkolaitteiden kanssa.

Arduino Uno on yksi suosituimmista ja laajimmin käytetyistä mikrokontrollerimalleista Arduino-perheessä. Se tarjoaa monipuolisen ja helppokäyttöisen alustan elektroniikan harrastajille, kehittäjille ja opiskelijoille, jotka haluavat luoda erilaisia projekteja ja prototyyppejä. Arduino Unon avulla voi toteuttaa lukemattomia erilaisia sovelluksia aina yksinkertaisista LED-ohjauksista monimutkaisiin automaatiojärjestelmiin.



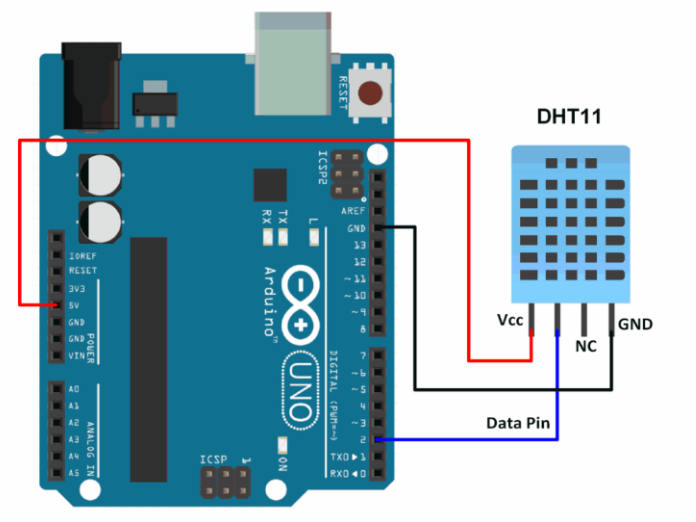
ESP8266WIFI moduuli.

Kytkentäkaaviot

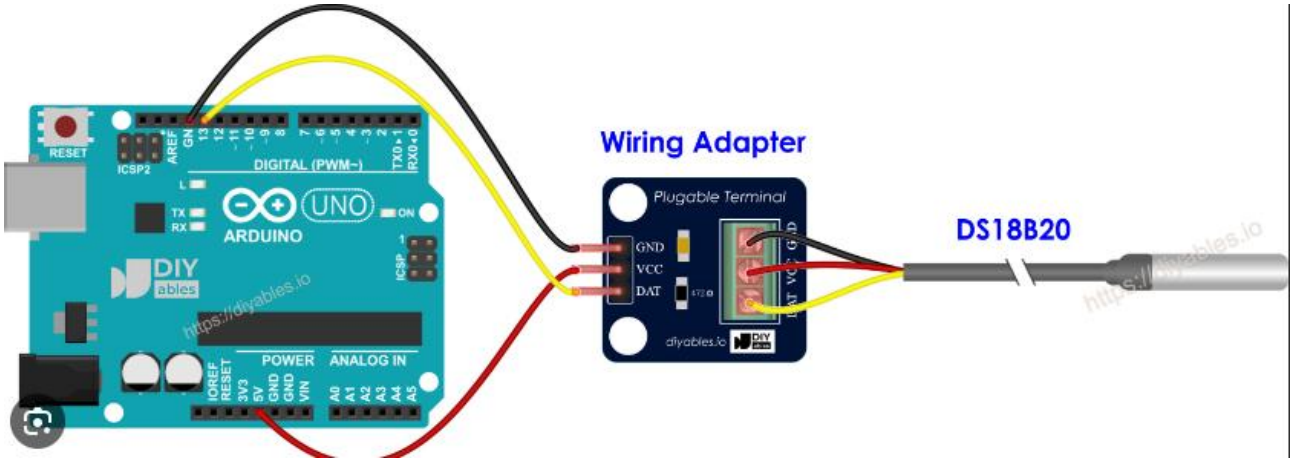


Kuvassa on nähtävillä kytkentäkaavio, sekä oikealla ylhäällä on loadcellin piirilevy joka laskee painomuutokset. Koska käytämme 4 kpl loadcellejä, joudumme laskemaan piirilevyllä jokaisen loadcellin

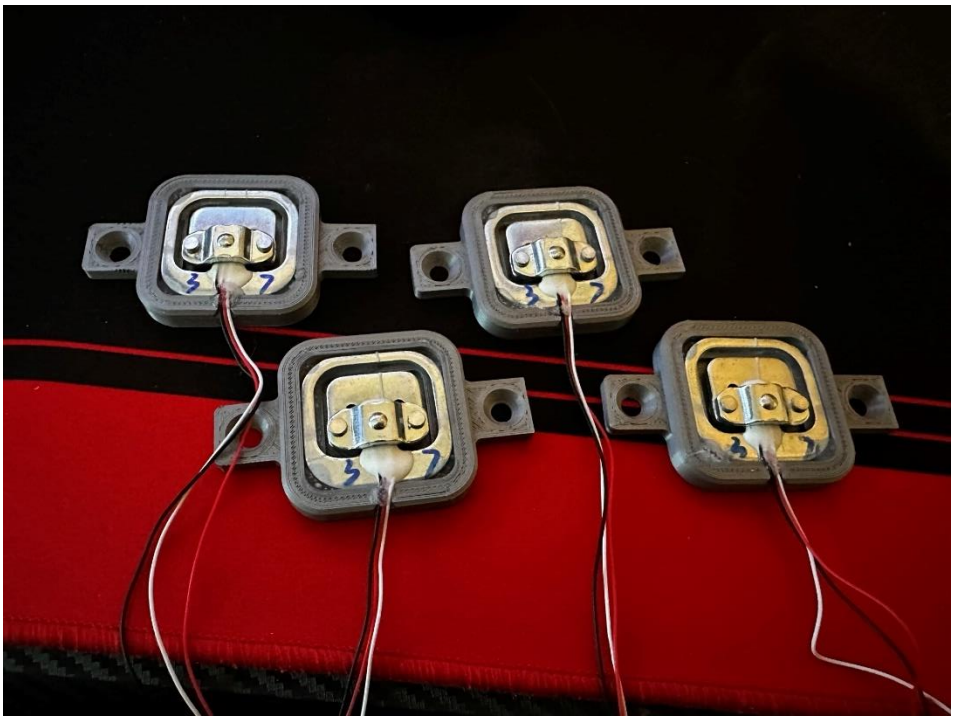
tulevan painon ja näistä muodostaa kokonaispaino. Joka välittää laskennan jälkeen arduinolle saadun tuloksen.



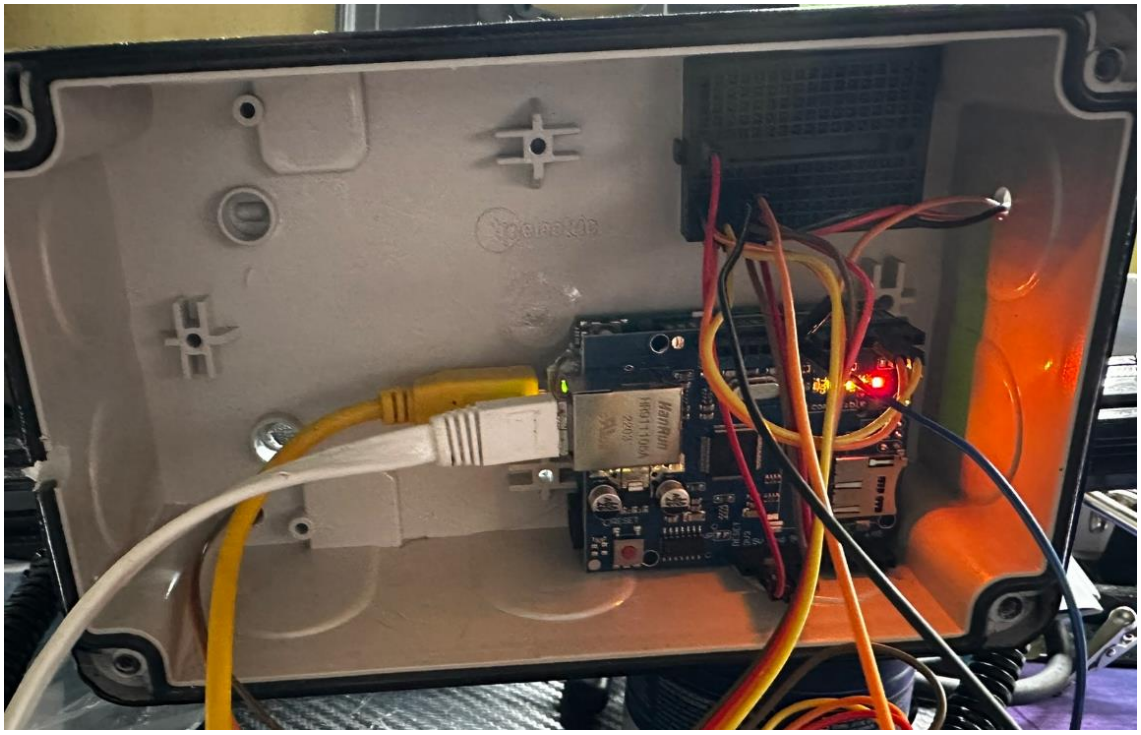
Kuvassa kytkentäkaavio DHT11 anturille.



DS18B20 kytkentäkaavio



Olen 3D tulostanut loadcelleille omat kiinnikkeet johon nämä voi sijoittaa siististi. Jotka loppujenlopuksi maalattiin mustaksi, sopimaan runkoon. Runko on rakennettu alumiiniprofiilista toistaiseksi. Mikäli tämä projekti olisi tarkoitus kaupallistaa, tulisi rungon materiaali muuttaa hieman siistiimpään materiaaliin. Sekä myös kotelot, sun muut.



Kun kaikki on kytketty kytkentäkaavioiden mukaan, voimme siirtyä ohjelmointivaiheeseen.

Ohjelmointi

Lämpötila ja kosteus

paaohjelmatemphumid.ino

```ino
#include <DHT.h>

#define DHTPIN 9         // Pin mistä luetaan
#define DHTTYPE DHT11    // Anturi mitä käytetään

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHT sensor reading program");
  dht.begin();
}

void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Tarkistetaan onko virheitä aiheutunut
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
  } else {
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.print(" °C\t");

    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
  }

  delay(2000); // Odotetaan hetki, ja luetaan lukemat uudestaan
}
```

```
19    if (isnan(humidity) || isnan(temperature)) {
20      Serial.println("Virhe lukiessa dataa!");
```

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on 'COM4')

```
Temperature: 25.70 °C   Humidity: 60.00 %
Temperature: 25.80 °C   Humidity: 60.00 %
Temperature: 25.80 °C   Humidity: 60.00 %
Temperature: 25.80 °C   Humidity: 60.00 %
Temperature: 25.80 °C   Humidity: 60.00 %
Temperature: 25.80 °C   Humidity: 60.00 %
Temperature: 25.80 °C   Humidity: 60.00 %
Temperature: 25.80 °C   Humidity: 60.00 %
Temperature: 25.80 °C   Humidity: 66.00 %
Temperature: 25.80 °C   Humidity: 71.00 %
Temperature: 26.10 °C   Humidity: 75.00 %
Temperature: 26.40 °C   Humidity: 77.00 %
Temperature: 26.60 °C   Humidity: 78.00 %
Temperature: 26.90 °C   Humidity: 79.00 %
Temperature: 27.20 °C   Humidity: 79.00 %
Temperature: 27.50 °C   Humidity: 79.00 %
Temperature: 27.80 °C   Humidity: 79.00 %
Temperature: 28.00 °C   Humidity: 78.00 %
Temperature: 28.20 °C   Humidity: 70.00 %
```

Kuvasta voimme todeta että tietoja ilmestyy. Kosteusprosentti ja lämpötilojen heittoja huomioiden, olemme aika lähellä todellisia lukuja. Hetken pitäessä anturia kädessä, voimme myös todeta että lämpötila sekä kosteusprosentti lähtee nousuun.

```
Temperature: 28.80 °C   Humidity: 58.00 %
Temperature: 28.90 °C   Humidity: 57.00 %
Temperature: 28.90 °C   Humidity: 56.00 %
Temperature: 28.90 °C   Humidity: 55.00 %
Temperature: 28.90 °C   Humidity: 55.00 %
Temperature: 28.90 °C   Humidity: 55.00 %
Temperature: 28.90 °C   Humidity: 54.00 %
Temperature: 28.90 °C   Humidity: 54.00 %
Temperature: 28.90 °C   Humidity: 54.00 %
Temperature: 28.90 °C   Humidity: 54.00 %
Temperature: 28.90 °C   Humidity: 54.00 %
Temperature: 28.80 °C   Humidity: 54.00 %
Temperature: 28.70 °C   Humidity: 54.00 %
Temperature: 28.70 °C   Humidity: 54.00 %
Temperature: 28.60 °C   Humidity: 54.00 %
Temperature: 28.60 °C   Humidity: 54.00 %
Temperature: 28.50 °C   Humidity: 54.00 %
```

Kädestä pois ottaessa, lukemat myös lähtevät laskemaan alkuperäisille tasoille.

Loadcell calibrointi

calibrointi_copy_20230809141343.ino

```ino
#include "HX711.h"

#define LOADCELL_DOUT_PIN  3
#define LOADCELL_SCK_PIN  2

HX711 scale;

float calibration_factor = -3280; // Säädä tämä arvo oikein kilogrammoille

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 kalibrointi ohjelma");
  Serial.println("Poista kaikki paino vaa'alta");
  Serial.println("Kun lukemat alkavat, laita tiedossa oleva paino vaa'alle");
  Serial.println("Paina + tai a lisätäksesi kalibrointikerrointa");
  Serial.println("Paina - tai z vähentääksesi kalibrointikerrointa");

  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale();
  scale.tare(); // Nollaa vaa'an lukema

  long zero_factor = scale.read_average(); // Hae peruslukema
  Serial.print("Nollakerroin: "); // Tätä voi käyttää tarvittaessa vaakalukemien nollaamiseen. Hyödyllistä pysyvissä vaakaprojekteissa.
  Serial.println(zero_factor);
}

void loop() {
  scale.set_scale(calibration_factor); // Säädä kalibrointikerroin

  Serial.print("Lukema: ");
  Serial.print(scale.get_units(), 1);
  Serial.print(" kg"); // Näytä paino kilogrammoina
  Serial.print(" kalibrointikerroin: ");
  Serial.print(calibration_factor);
  Serial.println();

  if (Serial.available()) {
    char temp = Serial.read();
    if (temp == '+' || temp == 'a')
      calibration_factor += 100; // Lisää 100 kalibrointikerrointa tarkemmin säätämiseen kilogrammoille
    else if (temp == '-' || temp == 'z')
      calibration_factor -= 100; // Vähennä 100 kalibrointikerrointa tarkemmin säätämiseen kilogrammoille
  }
}
```

Yksinkertaisesti koodin tarkoitus on kalibroida loadcellit oikeaan lukemiin. Tähän tarvitaan joku paino joka tunnetaan tarkalleen mikä se on. Jotta voimme löytää oikean calibrointikertoimen pääohjelmaamme

Pääasiassa ohjelma kuitenkin toimii ja lukemia tulee.

```
29
30    Serial.print("Lukema: ");
31    Serial.print(scale.get_units(), 1);
32    Serial.print(" kg"); // Näytä paino kilogrammoina
33    Serial.print(" kalibrointikerroin: ");
34    Serial.print(calibration_factor);
35    Serial.println();
36
```

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on 'COM4')

```
Lukema: 0.0 kg kalibrointikerroin: -4580.00
Lukema: 0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: 0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -4580.00
Lukema: -0.0 kg kalibrointikerroin: -458
```

Voimme kuitenkin todeta, että lukemamme on negaatiivinen, mikä tarkoittaa sitä että calibrointi kerroin ei ole oikein. Voimme siis muuttaa kalibrointikerrointa ohjelman mukaisesti näppäin painalluksilla.

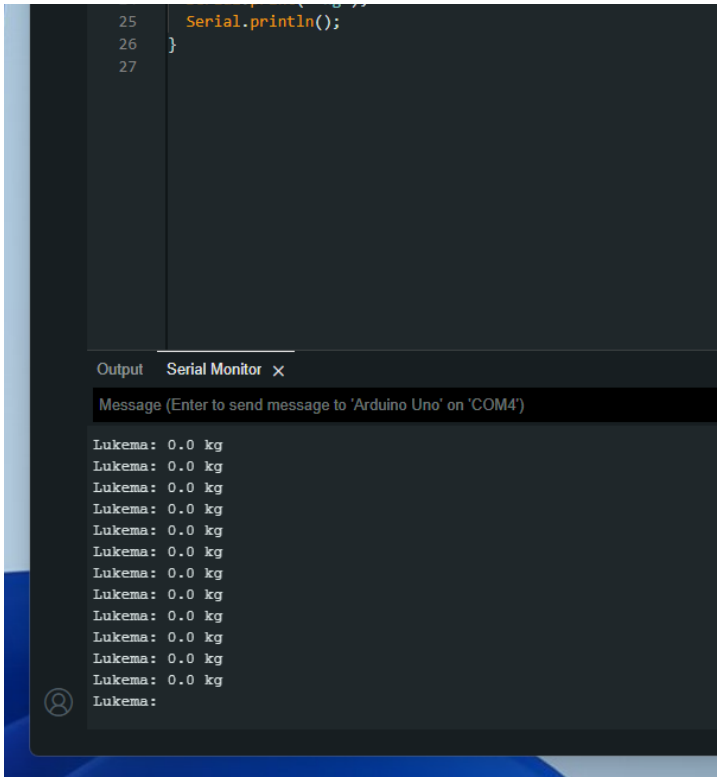
Tarkoitus on saada lukema juuri vastaamaan, meidän esimerkki painoa jonka painon tunnemme.

Kun paino on saatu juuri oikeaksi, otamme ylös kalibrointikertoimen luvun jota käytämme seuraavassa pääohjelmassa.

Painon mittauksen pääohjelma

```ino
#include "HX711.h"

#define calibration_factor -7050.0 // Tämä arvo saadaan käyttämällä calibraatio -ohjelmaa

#define LOADCELL_DOUT_PIN  3
#define LOADCELL_SCK_PIN  2

HX711 scale;

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 vaakademotila");

  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale(calibration_factor); // Tämä arvo saadaan käyttämällä calibraatio -ohjelmaa
  scale.tare(); // Olettaen, että vaaka on tyhjä käynnistyessä, nollaa vaaka

  Serial.println("Mittaukset:");
}

void loop() {
  Serial.print("Lukema: ");
  Serial.print(scale.get_units(), 1); // scale.get_units() palauttaa desimaaliluvun
  Serial.print(" kg");
  Serial.println();
}
```

Esimerkissä on lyhyt ohjelma, jossa todetaan laitteen toimivuus johon on syötetty kalibrointiohjelmasta saatu arvo.



Saamme painolukemia kokoajan, kun laitteellamme ei ole minkäänlaista painoa. Tulee arvo olla 0

```
Lukema: 2.7 kg
Lukema:
```

Kun olemme lisäneet painon, tulee arvo nousta. Jonka toteamme kuvasta että paino on noussut

EthernetShieldin ohjelmointi

sketch_aug16a.ino

```ino
#include <Ethernet.h>

// Määritä Ethernetin MAC-osoite
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

void setup() {
  // Käynnistä sarjaportti viestintää varten
  Serial.begin(9600);

  // Käynnistä Ethernet-kirjasto ja määritä MAC-osoite
  Ethernet.begin(mac);

  // Tulosta IP-osoite sarjaportille
  Serial.print("IP-osoite: ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  // Ohjelman pääsilmukka
}

```

Voimme ottaa todella lyhyellä koodilla käyttöön ethernet palikan. Koodissa tulee huomioida mac osoite, joka tulee vastata ethernet shieldin mac osoitetta. Tulostamme IP osoitteen monitoriin, testataksi toimivuuden.

DS18B20 ohjelmointi

```
#include <OneWire.h>
#include <DallasTemperature.h>

// Data-piiri on kytketty pinniin 8
#define ONE_WIRE_BUS 8

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(9600);
  sensors.begin();
}

void loop() {
  sensors.requestTemperatures();  // Lähetetään komento lämpötilamittausten pyytämiseksi

  float temperatureC = sensors.getTempCByIndex(0);  // Haetaan lämpötila celsius-asteina

  Serial.print("Lämpötila: ");
  Serial.print(temperatureC);
  Serial.println(" °C");

  delay(1000);  // Viive sekunnin ajan ennen seuraavaa mittauskertaa
}
```

Koodi on kyseiselle sensorille, testausta varten että kytkennät ja koodi on ok. Joka tulee implentoida pääohjelmaan.

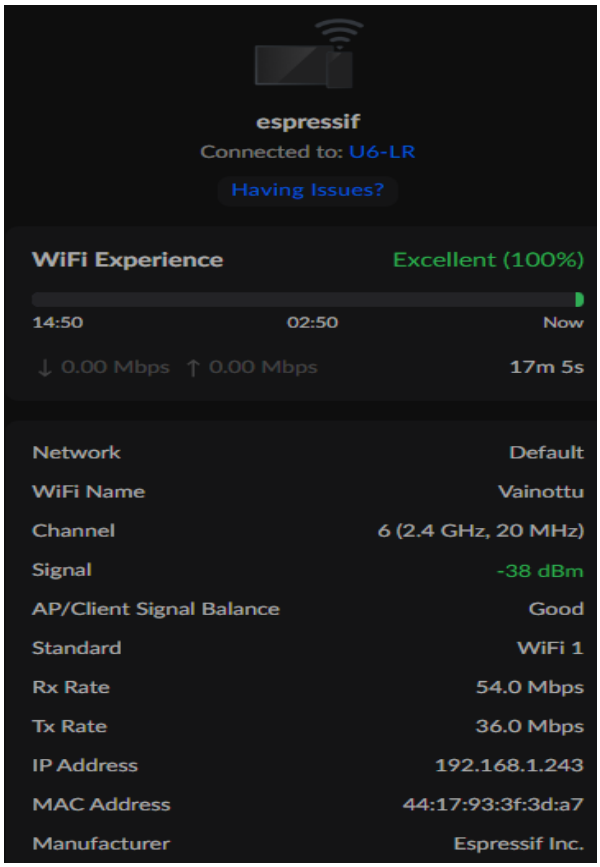
ESP8266 WIFI

Huomasin, että moduuli on access point moodissa vakiona. Joka tulee muuttaa, että se voi yhdistää internettiin.

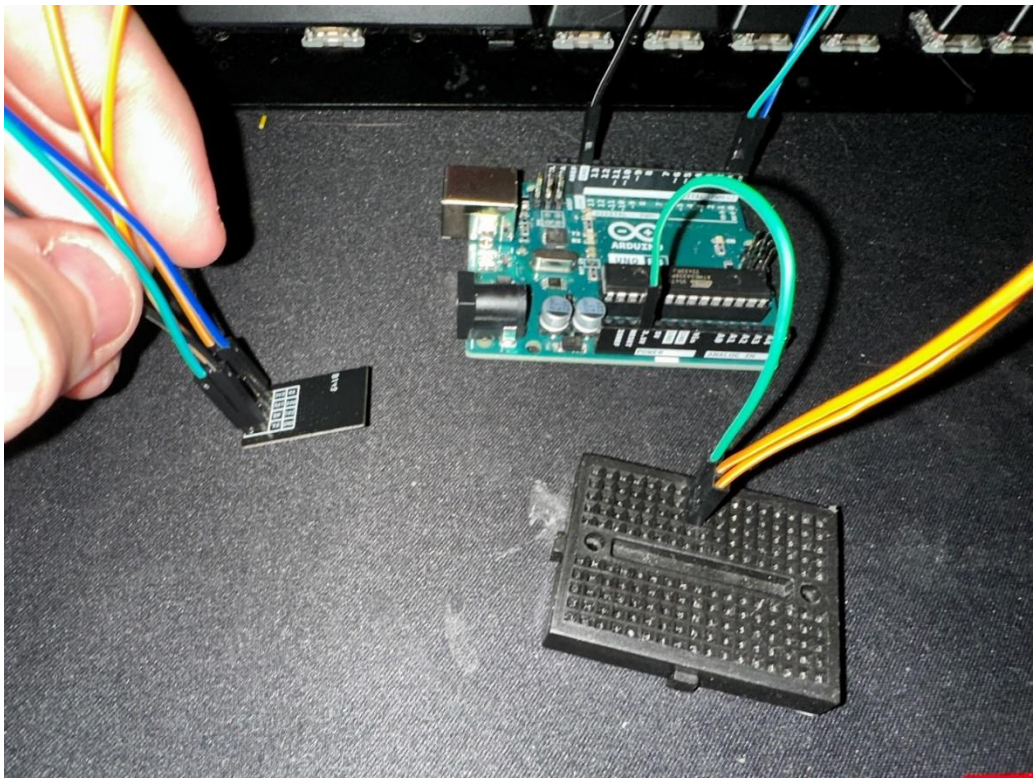
AT+CWMODE=1 (Muuttaa laitteen access point moodista > Station moodiin.

AT+CWJAP="your_ssid","your_password" (Asettaa wifin ja sen salanan, jonka jälkeen yrittää yhdistää siihen.

```
WIFI CONNECTED
WIFI GOT IP

OK
```



Jotta kytkentä toimisi pääohjelmassa varayhteytenä, tarvitsisimme jonkinlaisen mikropiirin joka seuraa liikettä kun ethernet yhteys katkeaisi, ja siirtyisi se senjälkeen suorittamaan wifin kautta. Tätä nyt ei ole mahdollista hankkia, joten jätämme wifi kokeilun toistaiseksi tähän.

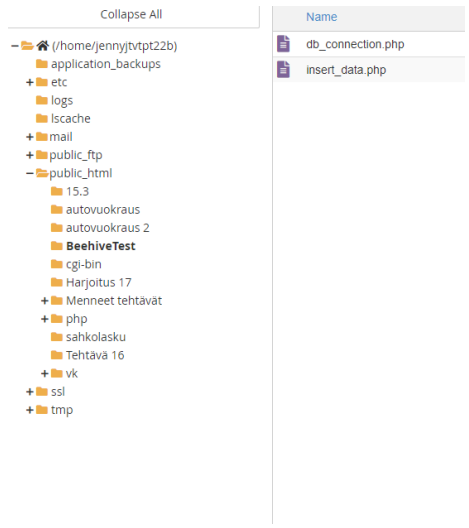


PHP ohjelma

```php
<?php
require_once "db_connection.php";

// Get values from the Arduino
$insideTemp = $_GET['insideTemp'];
$outsideTemp = $_GET['outsideTemp'];
$insideHumidity = $_GET['insideHumidity'];
$sound = $_GET['sound'];
$weight = $_GET['weight'];
$hiveID = $_GET['hiveID'];

// Prepare and bind the statement
$stmt = $conn->prepare("INSERT INTO Measurements (HiveID, TimeStamp, InsideTemp, OutsideTemp, InsideHumidity, Sound, Weight) VALUES (?, NOW(), ?, ?, ?, ?, ?)");
$stmt->bind_param("iddddd", $hiveID, $insideTemp, $outsideTemp, $insideHumidity, $sound, $weight);


// Execute the statement
if ($stmt->execute()) {
    echo "Data inserted successfully.";
} else {
    echo "Error inserting data: " . $stmt->error;
}

$stmt->close();
$conn->close();
?>

```

Meidän tulee luoda myös php ohjelma, jotta voimme kirjoittaa tietomme talteen palvelimelle internetyhteyden välityksellä. Luomme lyhyen ja tehokkaan koodin, jossa kutsutaan adruinolla php ohjelmaa joka sijaitsee vlab palvelimella. Koodia tulisi kehittää eteenpäin, jotta esimerkiksi tietoturvaaukot tulee korjattua.

Kyseisellä koodilla on mahdollista syöttää tietoja ilman Arduinoa, mikäli joku kyseisen osoitteen sattuisi löytämään.



Esimerkki kuvassamme olemme sijoittaneet tiedostot palvelimella. Db_connection.php sisältää käyttäjätunnukset ja salasanat yhteyden luomista varten.

SQL taulukkorakenne

C: > Users > jalop > Downloads > jennyjtvtpt22b_BeeHive.sql

```sql
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

CREATE TABLE `Hives` (
  `HiveID` int(11) NOT NULL,
  `UserID` int(11) DEFAULT NULL,
  `HiveName` varchar(100) DEFAULT NULL,
  `HiveLocation` varchar(100) DEFAULT NULL,
  `HiveType` varchar(50) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;


CREATE TABLE `Measurements` (
  `MeasurementID` int(11) NOT NULL,
  `HiveID` int(11) DEFAULT NULL,
  `TimeStamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `InsideTemp` float DEFAULT NULL,
  `OutsideTemp` float DEFAULT NULL,
  `InsideHumidity` float DEFAULT NULL,
  `Sound` float DEFAULT NULL,
  `Weight` float DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO `Measurements` (`MeasurementID`, `HiveID`, `TimeStamp`, `InsideTemp`, `OutsideTemp`, `InsideHumidity`, `Sound`, `Weight`) VALUES
(120, NULL, '2023-08-16 10:25:32', 26.4, NULL, 58, NULL, -85.08),
(119, NULL, '2023-08-16 09:38:16', 25.4, NULL, 61, NULL, 113.22),
(118, NULL, '2023-08-16 09:23:00', 24.9, NULL, 63, NULL, 97.18),
(117, NULL, '2023-08-15 09:02:45', 22.9, NULL, 65, NULL, 97.06),
(116, NULL, '2023-08-15 08:52:44', 22.9, NULL, 65, NULL, 97.07);


CREATE TABLE `Users` (
  `UserID` int(11) NOT NULL,
  `Username` varchar(100) DEFAULT NULL,
  `Password` varchar(100) DEFAULT NULL,
  `Email` varchar(100) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

ALTER TABLE `Hives`
  ADD PRIMARY KEY (`HiveID`),
  ADD KEY `UserID` (`UserID`);

ALTER TABLE `Measurements`
  ADD PRIMARY KEY (`MeasurementID`),
  ADD KEY `HiveID` (`HiveID`);

ALTER TABLE `Users`
  ADD PRIMARY KEY (`UserID`);

ALTER TABLE `Measurements`
  MODIFY `MeasurementID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=121;
COMMIT;
```

Jotta voimme tallentaa tietomme talteen palvelimelle, joudumme luomaa SQL taulun rakenteet. Esimerkkikuvassa on myös jotain dataa syötetty tauluun. Jotta voimme todeta taulujen toimivuuden, Erityisesti mittauksiemme.

Hive ja user taulut on jatkojalostusta varten, mikäli haluamme useille käyttäjille mahdollisuuden lisäämään omat laitteensa palvelimellemme.

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	Hives	Browse Structure Search Insert Empty Drop	0	MyISAM	latin1_swedish_ci	1.0 KiB	-
☐	Measurements	Browse Structure Search Insert Empty Drop	5	MyISAM	latin1_swedish_ci	4.7 KiB	1.5 KiB
☐	Users	Browse Structure Search Insert Empty Drop	0	MyISAM	latin1_swedish_ci	1.0 KiB	-
	3 tables	Sum	5	MyISAM	latin1_swedish_ci	6.7 KiB	1.5 KiB

Check all / Check tables having overhead With selected:

Taulukkorakenne visuaalisessa muodossa.

Pääohjelma

Nyt meillä on tähän asti kaikki komponentit, koodit luotu eri toiminnoille. Seuraavassa vaiheessa joudumme lyömään nämä kaikki yhteen ja luomaan itse pääohjelman.

```
#include <DHT.h>
#include <Ethernet.h>
#include "HX711.h"
#include <OneWire.h>
#include <DallasTemperature.h>

#define DHTPIN 9         // Pin mistä luetaan
#define DHTTYPE DHT11    // Anturi mitä käytetään

DHT dht(DHTPIN, DHTTYPE);

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // Ethernet shieldin mac
osoite
EthernetClient client;

const char *serverAddress = "jennyj.vlab.fi"; // Palvelimen osoite
const int serverPort = 80;
unsigned long previousMillis = 0;
const long interval = 600000; // 10 minuutin välein

#define calibration_factor -25050.0 // Muutetaan arvo joka saadaan calibrointi
ohjelmasta
#define LOADCELL_DOUT_PIN  3
#define LOADCELL_SCK_PIN  2

HX711 scale;

// OneWire configuration
#define ONE_WIRE_BUS 8
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(9600);
  Serial.println("DHT sensor reading program");

  dht.begin();
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale(calibration_factor);

  Ethernet.begin(mac);
  while (!Ethernet.begin(mac)) {
    Serial.println("Ethernet connection failed");
```

```
    delay(1000);
  }
  Serial.println("Connected to Ethernet");

  sensors.begin(); // Initialize OneWire temperature sensors
}

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Resetoidaan anturit ennen lukemista
    dht.begin();

    float humidity = dht.readHumidity();
    float insideTemperature = dht.readTemperature();

    // Katsotaan onko jokin failed
    if (isnan(humidity) || isnan(insideTemperature)) {
      Serial.println("Datan luku epäonnistu!");
    } else {
      Serial.print("Inside Temperature: ");
      Serial.print(insideTemperature);
      Serial.print(" °C\t");

      Serial.print("Humidity: ");
      Serial.print(humidity);
      Serial.print(" %");

      // Read outside temperature
      sensors.requestTemperatures();
      float outsideTemperature = sensors.getTempCByIndex(0);

      Serial.print(" Outside Temperature: ");
      Serial.print(outsideTemperature);
      Serial.print(" °C");

      float weight = scale.get_units(); // Haetaan painolukema

      Serial.print(" Weight: ");
      Serial.print(weight);
      Serial.println(" kg");

      sendSensorData(insideTemperature, humidity, outsideTemperature, weight);
    }
  }
```

```
  delay(2000); // Odotetaan ennen seuraavaa lukua
}

void sendSensorData(float insideTemperature, float humidity, float
outsideTemperature, float weight) {
  if (client.connect(serverAddress, serverPort)) {
    // luodaan http get pyyntö
    String url = "/BeehiveTest/insert_data.php"; // Url missä php scripti sijaitsee
    url += "?insideTemp=" + String(insideTemperature);
    url += "&insideHumidity=" + String(humidity);
    url += "&outsideTemp=" + String(outsideTemperature);
    url += "&weight=" + String(weight);

    // lähetetään GET pyyntö
    client.print("GET " + url + " HTTP/1.1\r\n" +
                 "Host: " + serverAddress + "\r\n" +
                 "Connection: close\r\n\r\n");
    delay(10);

    // Luetaan ja tulostetaan vastausta palvelimelta.
    while (client.available()) {
      String line = client.readStringUntil('\r');
      Serial.println(line);
    }

    client.stop();
  } else {
    Serial.println("Connection to server failed");
  }
}
```

Joudumme nyt kuviin pätkimään koodin useampaan osaan, johtuen ultrawide näytöstä ja kuvakaappauksen rajallisesta koosta.

Kun ohjelma on ajettu suoraan arduinoon, laite tulisi toimia niin kuin sen on tarkoituskin.

Laite siis ottaa yhteyden ethernetshieldin kautta, lukee datan antureista jonka jälkeen kirjoittaa nämä php scriptiä käyttäen tietokantaan.

Tässäkohtaa on hyvä myös huomata, kun koodi on ajettu arduinolle että ohjelman muistin tilankäyttö on yllättävän suuri. Aikaisemmassa versiossa huomioin, että kun otettiin suorayhteys tietokantaan sql komennoilla ilman php koodia. Muistinkäyttö oli päälle 80%. Nykyinen versio on siis hieman tehokkaampi, kun käytetään php:tä myös.

```
Output   Serial Monitor
Sketch uses 23828 bytes (73%) of program storage space. Maximum is 32256 bytes.
Global variables use 984 bytes (48%) of dynamic memory, leaving 1064 bytes for local variables. Maximum is 2048 bytes.
```

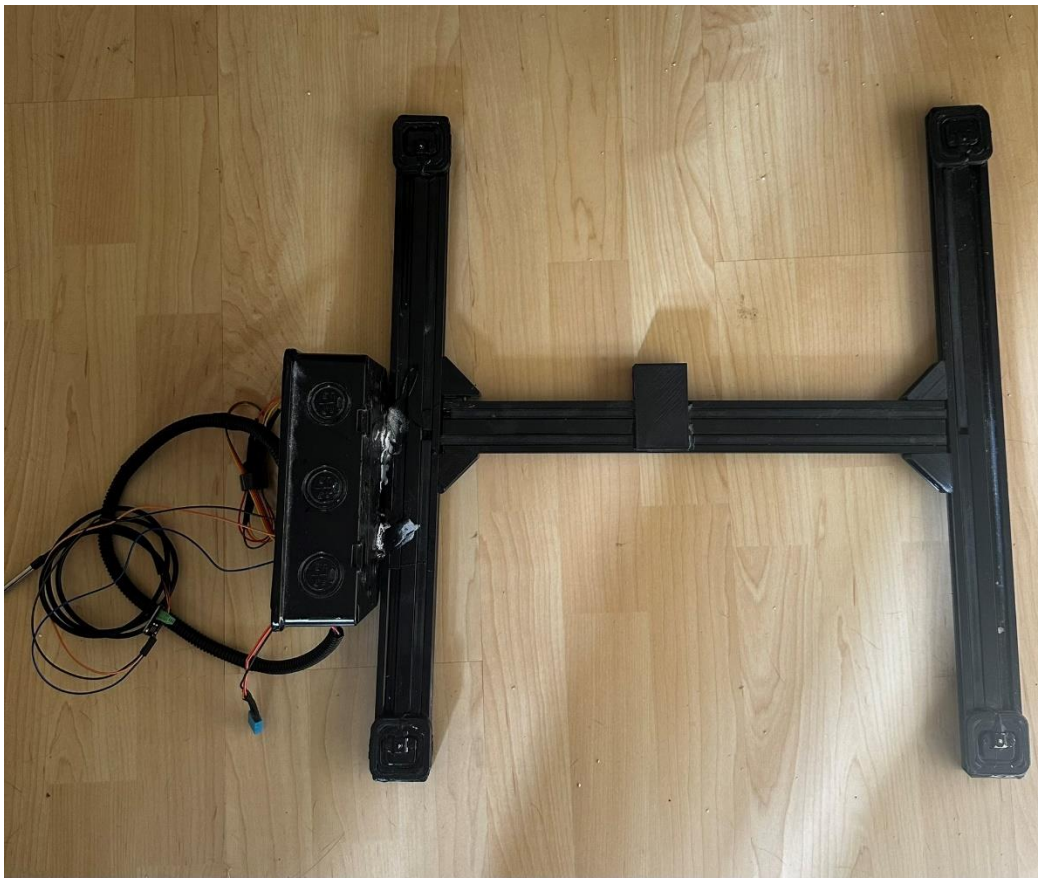
Mikäli tätä projektia jatkojalostettaisiin kaupallisempaan käyttöön, tulee koodia mahdollisuuksien mukaan lyhentää sekä tehostaa entisestään. Vaihtoehtoisesti hankkia tehokkaampaa arduino laitteita.

Kuvia ulkoa.

Kuvassa havainnollistamaan mitäkautta sisätilalämpöanturi on viety mehiläispesän sisälle.

Kuva sivulta.

Laitelaatikko. Virtajohto ja ethernet johdon on tarkoitus poistua valmiissa versiossa pois näkyviltä. Nyt kuva vain havainnollistamaan.



Itse laite odottamassa kiinteää asennnusta.

Lyhyt videoesittely:

https://youtu.be/qBLn-ccJ7vA

Itsearviointi

Tavoitteenasettelu ja projektin suunnittelu (5/5):

Projekti alkoi selkeästi määritellyllä tavoitteella: mehiläispesän ympäristön seuranta Arduino-alustalla.

Suunnittelu oli huolellinen ja kattava, ja se sisälsi tarvittavat komponentit ja niiden yhteensopivuuden.

Projektissa asetettiin selkeät mittarit ja tavoitteet ympäristötietojen keräämiseksi ja analysoimiseksi. Projektin tarkoitus on tuottaa dataa kotisohvalle, helpottamaan mehiläisten hoitoa.

Tekninen toteutus (5/5):

Arduino-alustaa käytettiin tehokkaasti ympäristötietojen keräämiseen.

Sensoreiden asennus ja liittäminen oli hyvin toteutettu.

Koodin ohjelmointi oli tehokasta ja toimivaa, ja se mahdollisti luotettavan datan keräämisen.

Jossa myös havaittiin muutamia korjausideoita kehitystyön edetessä, mm tehostaa resurssienkäyttöä.

Tietojen keruu ja analysointi (5/5):

Arduino onnistui keräämään laadukasta ympäristötietoa mehiläispesän ympäristöstä.

Arduino kerää dataa onnistuneesti ja tallentaa ne tietokantaa käyttäen php rajapintaa.

Dokumentaatio ja raportointi (5/5):

Projektista laadittiin selkeä ja kattava dokumentaatio, joka sisälsi projektin tavoitteet, suunnitelman, toteutuksen ja tulokset.

Innovatiivisuus ja lisäarvo (5/5):

Projekti osoittaa innovatiivisuutta käyttämällä Arduinoa mehiläispesien ympäristön seurantaan, mikä voi olla hyödyllistä mehiläishoitajan tehtävien helpottamiseksi.

Projektia on mahdolista jatkojalostaa haluamaan suuntaan.

Kokonaisuudessaan Arduino-mehiläispesän ympäristöseurantaprojektinne ansaitsee kiitettävän 5 arvosanan erinomaisesta suunnittelusta, toteutuksesta ja dokumentoinnista. Projektin innovatiivisuus ja sen mahdollinen vaikutus mehiläisten suojelemiseen ja työn helpottamiseen ovat erityisen huomionarvoisia.