Puutarhan seurantajärjestelmä

Järjestelmän kuvaus

Järjestelmällä seurataan kasvimaan ja kasvihuoneen maan sekä ilman lämpötila- ja kosteustietoja. Järjestelmällä voidaan myös käynnistää pumppu kastelua varten. Järjestelmään kuuluu ilman lämpötilaa ja kosteutta seuraavia antureita sekä maan kosteutta ja lämpötilaa seuraavia antureita. Lisäksi kasteluyksikköön kuuluu päällekytkentärele sekä jokivedenpinnan korkeutta tarkkaileva anturi. Anturit on kytketty kolmeen NodeMcu -yksikköön. Tiedot lähetetään pilvipalvelimelle ja keskusyksikkönä toimivalle raspberry Pi 3 yksikölle. Järjestelmä sijaitsee n. 50 kilometrin etäisyydellä kodista, joten järjestelmän toimintaa pitää voida tarkkailla ja huoltaa etäyhteyden kautta. Sitä varten tarvitaan nopea verkkoyhteys, joka toteutetaan 4G verkkoyhteydellä. Projektin ensimmäisessä vaiheessa toteutetaan ympäristötiedon mittaus ja tietojen lähettäminen pilvipalveluun. Myöhemmin toteutettavassa toisessa vaiheessa automatisoidaan kastelu ja kasvihuoneen ilmanvaihtoon liittyvät toiminnot.

Järjestelmän mittaamat ympäristötiedot ovat ThingSpeakissa.

Linkki: https://thingspeak.com/channels/268204

Tarvittavat komponentit

1 4G langaton reititin ja wlan yksikkö

1 raspberry pi -yksikkö tiedonkeruuseen ja tiedonvälitykseen.

3 NodeMcu -yksikköä mittaustiedon keruuta ja toimintojen ohjausta varten.

2 maan kosteutta mittaavaa anturia (FC-28-D).

3 maan lämpötilaa mittaavaa anturia (DIY DS18B20).

2 ilman lämpötilaa ja kosteutta mittaavaa anturia (dht11).

1 ultraäänianturi mittaamaan vedenpinnan korkeutta joessa (Hc -Sr04).

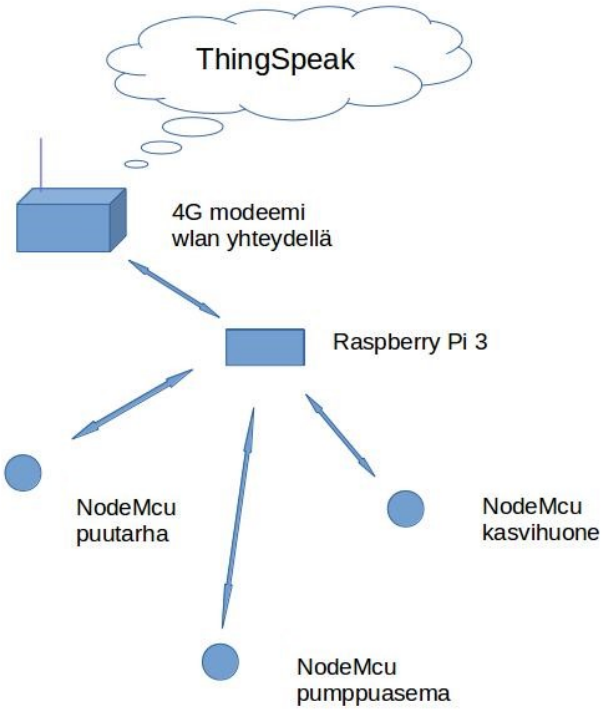
1 vedenpaineen mitta-anturi kasteluletkuun.

1 ohjattava kytkentärele pumpun käynnistämiseen.

3 jännitteenkasvatusvälinettä.

1 liitäntäjohtosetti. Universaali uros-naaras, uros-uros, naaras-naaras

Järjestelmäkaavio



Ohjemistokuvaus

NodeMcu-yksiköt

Puutarha:

Kasvimaalla oleva NodeMcu -yksikkö mittaa maan kosteuden, maan lämpötilan, lämpötilan 20 cm:n korkeudella maasta sekä lämpötilan ja ilman kosteuden 1,5 m:n korkeudella maasta. Tiedot luetaan ja lähetetään Raspberry Pi 3:lle jatkokäsittelyä ja kasteluun liittyviä automatiotoimintoja varten mqtt viestinä puolen tunnin välein. Tiedot lähetetään myös suoraan ThingSpeakiin, josta niitä voi helposti seurata.

Kasvihuone:

Kasvihuoneessa oleva Node Mcu -yksikkö mittaa maan kosteuden, ilman lämpötilan ja ilman kosteuden ja lähettää tiedot Raspberry Pi 3:lle mqtt viestinä jatkokäsittelyä ja kastelun ja ilmanvaihdon automatisointia varten. Tiedot lähetetään myös suoraan ThingSpeakiin, josta niitä on helppo seurata.

Kastelupumpun kontrolliyksikkö:

Koska jokiveden pinta voi laskea niin alas, että vettä ei ole, on pumpun käynnistyminen

kuivana voitava estää. Tätä varten kontrolliyksikkö mittaa vedenpinnan korkeuden ja estää pumpun käynnistymisen, jos vedenpinta on asetetun raja-arvon alapuolella. Pumpun etäkäynnistystä varten kontrolliyksikössä on ohjattava rele. Veden pinnan korkeustieto lähetetään myös ThingSpeakiin, josta sitä on helppo seurata.

Raspberry Pi 3

Raspberry Pi 3:n tehtävänä on ohjata automaatioon liittyviä toimintoja, kuten kastelua ja kasvihuoneen ilmanvaihtoa. Tietoa voidaan myös kerätä tarpeen mukaan enemmän, kuin pivipalveluun lähetetään. Tässä on kyse lähinnä pilvipalvelun kustannuksista, eli paljonko dataa voi lähettää ilman kustannuksia.

NodeMcu -kykentäkaaviot

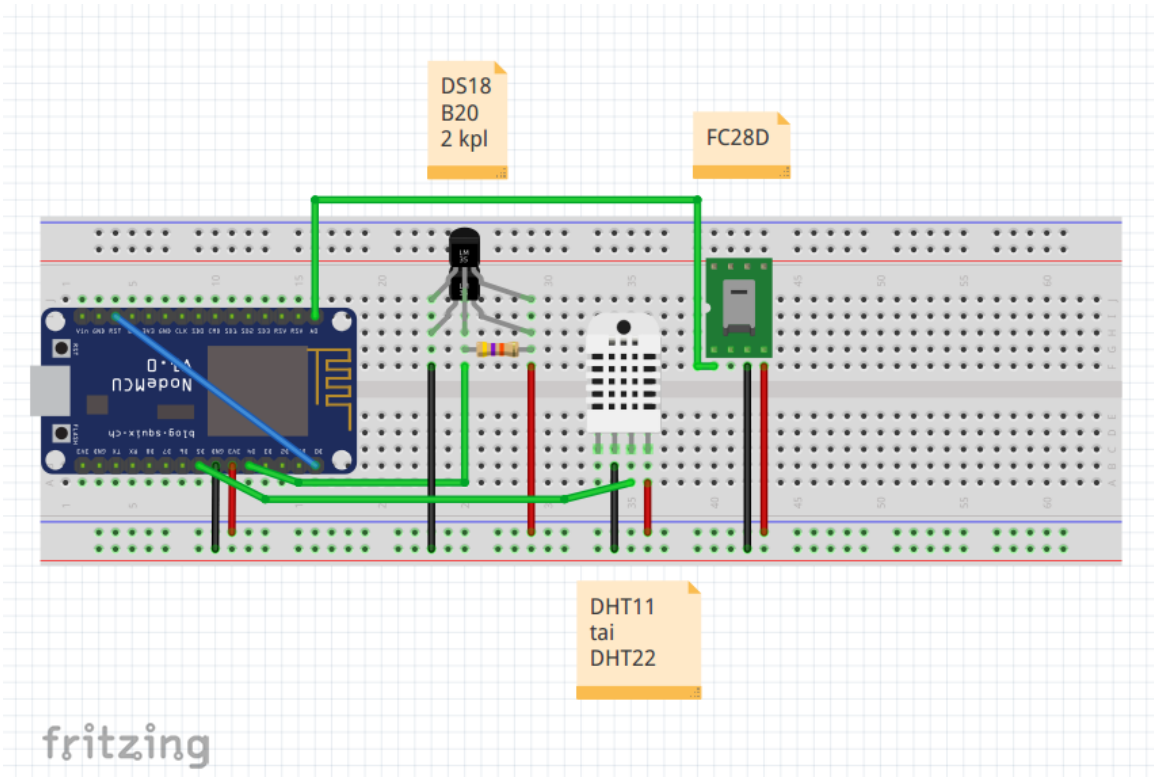
Lämpätilan ja kosteudenmittausyksikköjen kytkennät

Maan kosteudenmittausyksikön sensori A0 kytketään NodeMCU:n liitimeen A0.

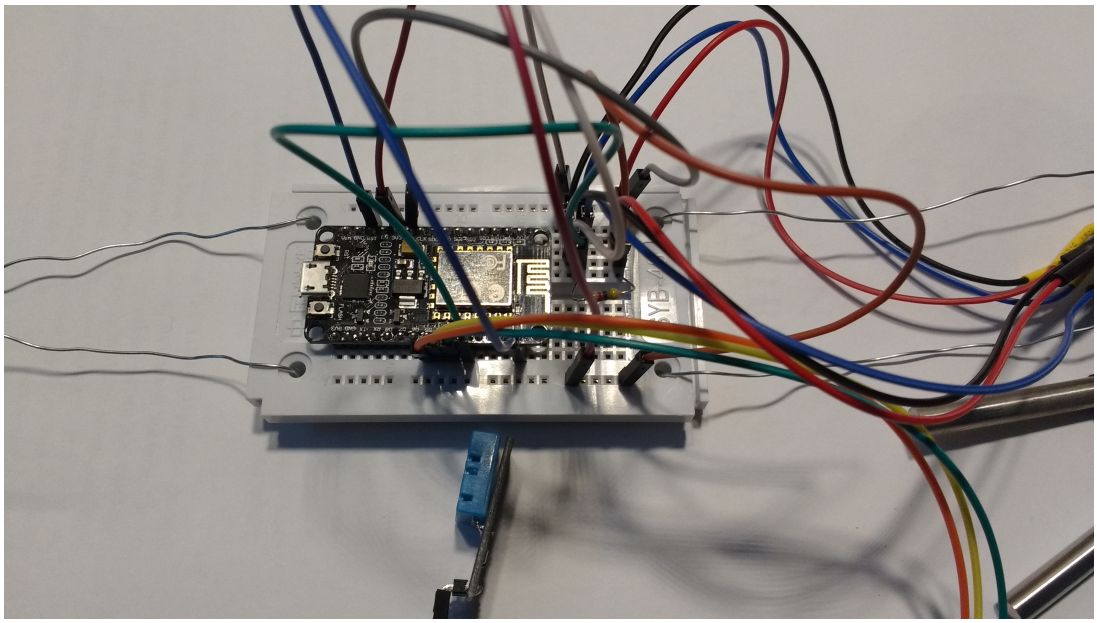
Lämpötilanmittausyksiköiden out-liitin kytketään NodeMCU:n digitaaliseen liittimeen D4.

Lämpötilan- ja kosteudenmittausyksikön out liitin kyketään NodeMCU:n digitaaliseen liittimeen D5.

Deep Sleep -moden herätystoimintoa varten D0 ja reset liittimet yhdistetään.



Kuva 1: Kaavio kytkennöistä kytkentälevyllä.



Kuva 2: Testiasennus koekytkentälevyllä

Kuva 3: Mittauspiste paikoillaan pellolla

Node Mcu koodi

Olosuhteidenmittausyksikkö:

```
/* Tämä koodi mittaa lämpötilan, ilmankosteuden ja maan kosteuden arvoja ulkona. Mittaustiedot lähetetään
ThingSpeakiin, josta niitä voi seurata.*/

#include <PubSubClient.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <WiFiClient.h>
#include <DallasTemperature.h>
#include <Streaming.h>

// DHT Sensor connected to digital pin 5
#define DHTPIN D5
// Type of DHT sensor
#define DHTTYPE DHT11
// Deep sleep delay, in this case 5 minutes
#define SLEEP_DELAY_IN_SECONDS  300
// Waterproof temperature sensor
#define ONE_WIRE_BUS            D4      // DS18B20 pin
// Soil moisture sensor pin
int sensorPin = A0;

// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE);
//  your network SSID (name)
char ssid[] = "DNA-Mokkula...jne";
// your network password
char pass[] = "XXX.....";

// Initialize the Wifi client library
WiFiClient client;
// Initialize the PuBSubClient library
PubSubClient mqttClient(client);
// Define the ThingSpeak MQTT broker
const char* server = "mqtt.thingspeak.com";

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);

void setup() {
 // Begin serial transfer
 Serial.begin(115200);
 dht.begin();
 delay(10);
 // Connect to WiFi network
 Serial.println();
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, pass);
 while (WiFi.status() != WL_CONNECTED)
 {
  delay(500);
  Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");
 // Print the IP address
 Serial.println(WiFi.localIP());
```

```
 // Set the MQTT broker details
 mqttClient.setServer(server, 1883);
}

// Get temperature. One wire off the ground
float getTemperature() {
 Serial << "Requesting DS18B20 temperature..." << endl;
 float temp;
 do {
  DS18B20.requestTemperatures();
  temp = DS18B20.getTempCByIndex(0);
  delay(100);
 } while (temp == 85.0 || temp == (-127.0));
 return temp;
}

// Get temperature. One wire. Soil temperature.
float getTemperature2() {
 Serial << "Requesting DS18B20 temperature..." << endl;
 float temp;
 do {
  DS18B20.requestTemperatures();
  temp = DS18B20.getTempCByIndex(1);
  delay(100);
 } while (temp == 85.0 || temp == (-127.0));
 return temp;
}

// Get soil moisture
int getMoisture() {
 Serial.println("Requesting FC28D moisture...");
 int sensorValue = analogRead(sensorPin);
 Serial.println("Sensor value:");
 Serial.println(sensorValue);
 // Normal value limits are 966 (dry) and 337 (water) 202 (wet soil).
 // Transverse those to show from down to up by changing moist values to negative -(966+200), 200 for
precaution
 sensorValue = sensorValue-1266;
 Serial.println(sensorValue);
 // Divide those also to be approximately between 20-100,
 sensorValue = sensorValue/10;
 // Change those to positive
 sensorValue = sensorValue - sensorValue - sensorValue;
 return sensorValue;
}

void loop() {
 // Check if MQTT client has connected else reconnect
 if (!mqttClient.connected())
 {
  reconnect();
 }
 // Call the loop continuously to establish connection to the server
 mqttClient.loop();
  mqttpublish();
 delay(100);
 Serial << "Entering deep sleep mode for " << SLEEP_DELAY_IN_SECONDS << " seconds..." << endl;
 ESP.deepSleep(SLEEP_DELAY_IN_SECONDS * 1000000, WAKE_RF_DEFAULT);
 delay(500);   // wait for deep sleep to happen
}

void reconnect()
{
```

```
 // Loop until we're reconnected
 while (!mqttClient.connected())
 {
  Serial.print("Attempting MQTT connection...");
  // Connect to the MQTT broker
  if (mqttClient.connect("ESP8266Client"))
  {
   Serial.println("connected");
  } else
  {
   Serial.print("failed, rc=");
   // Print to know why the connection failed
   // See http://pubsubclient.knolleary.net/api.html#state for the failure code and its reason
   Serial.print(mqttClient.state());
   Serial.println(" try again in 5 seconds");
   // Wait 5 seconds before retrying to connect again
   delay(5000);
  }
 }
}

void mqttpublish() {
 // Read temperature from DHT sensor, Fahrenheit if dht.readTemperature(true)
 float t = dht.readTemperature();
 // Cut data length of all values. Last values in string are not othervise accepted.
 // Just cut the value. Rounding is not necessary in this context.
 String t_s = String(t, DEC);
 t_s = t_s.substring(0,7);

 // Read humidity from DHT sensor
 float h = dht.readHumidity();
 String h_s = String(h, DEC);
 h_s = h_s.substring(0,7);

 // Read from waterproof one wire temperature sensors
 float t2 = getTemperature();
 String t2_s = String(t2, DEC);
 t2_s = t2_s.substring(0,7);
 float t3 = getTemperature2();
 String t3_s = String(t3, DEC);
 t3_s = t3_s.substring(0,7);

 // Read soil moisture
 int moisture = getMoisture();

 // Create data string to send to ThingSpeak
 String data = String("field1=" + t_s + "&field2=" + h_s + "&field3=" + t2_s + "&field4=" + t3_s + "&field5="
+ String(moisture, DEC));
 // Get the data string length
 int length = data.length();
 char msgBuffer[length];
 // Convert data string to character buffer
 data.toCharArray(msgBuffer,length+1);
 Serial.println(msgBuffer);

 // Publish data to ThingSpeak. Replace <YOUR-CHANNEL-ID> with your channel ID and <YOUR-
CHANNEL-WRITEAPIKEY> with your write API key
 mqttClient.publish("channels/<CHANNEL-ID>/publish/<CHANNEL-WRITEAPIKEY>",msgBuffer);
 delay(100);
}
```

Jatkokehitystarpeita

Tavoittena on rakentaa Raspberry Pi 3 -pohjainen täysin automatisoitu järjestelmä, joka säätää kastelua tarpeen mukaan ja huolehtii kasvihuoneen ilmanvaihdosta säätämällä tuuletusluukkujen aukipitoa lämpötilan mukaan. Sään tarkkailua varten lisätään myöhemmin valon määrää, ilmanpainetta, tuulta ja salamoita havainnoivat anturit.