Project: Smart Greenhouse

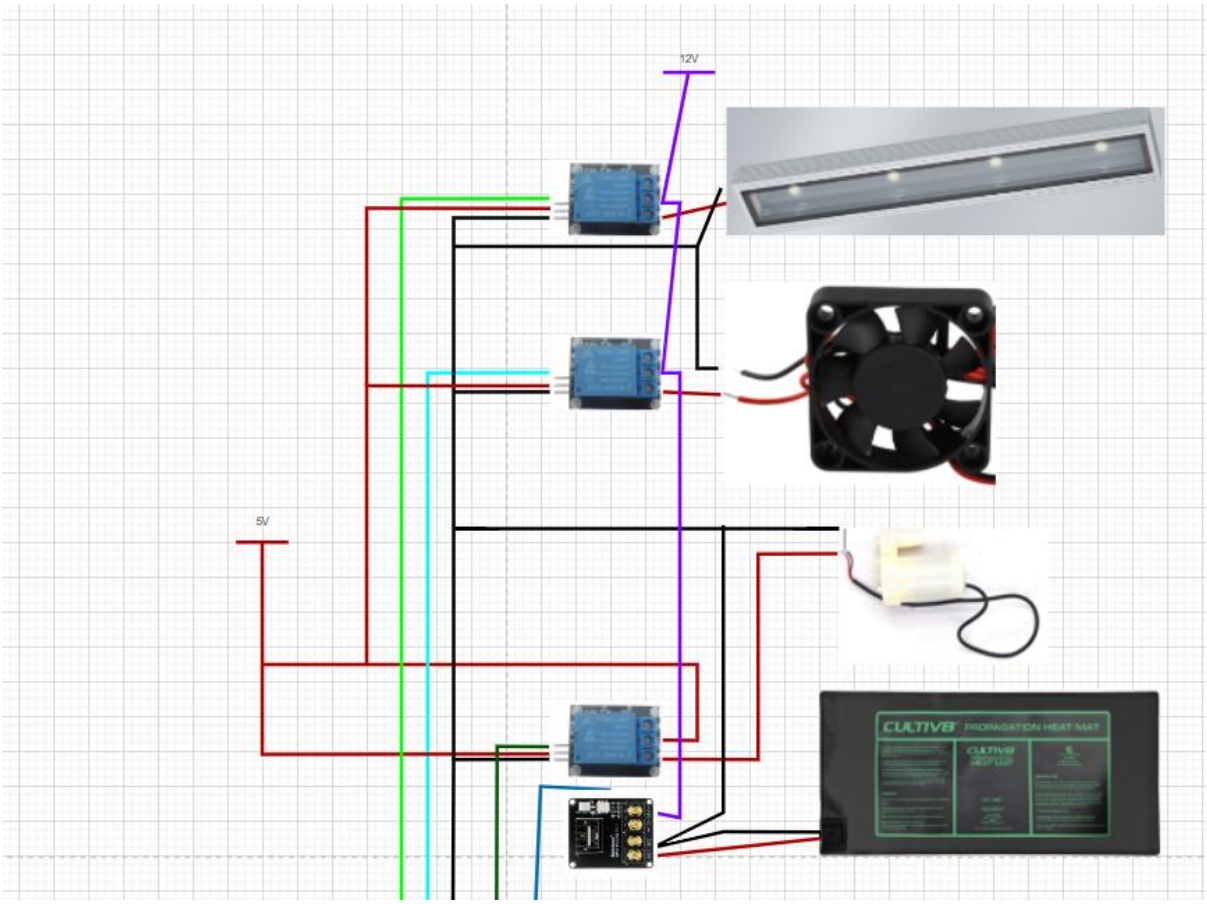
Doel van project:

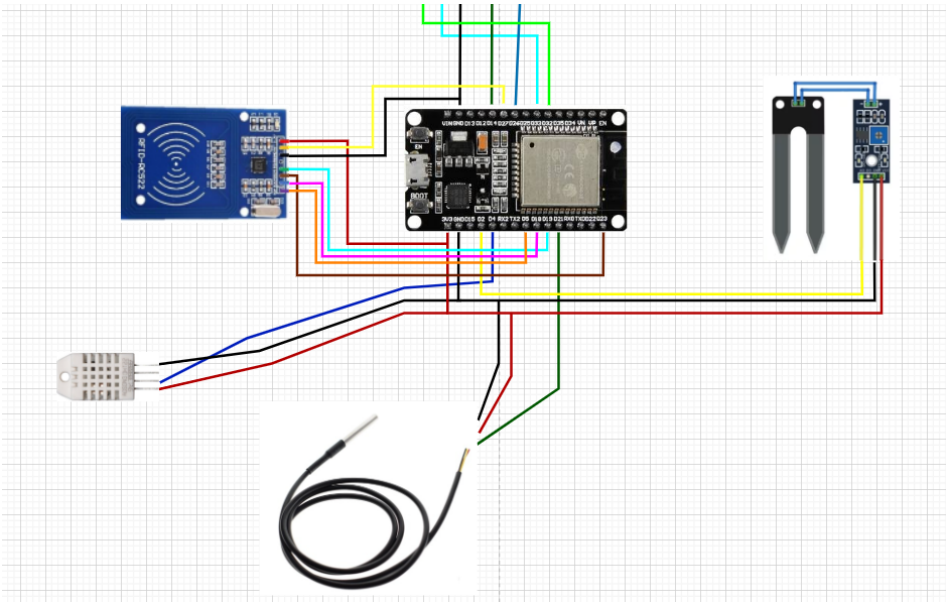
Het doel van het project is om een greenhouse te bouwen dat zichzelf reguleert aan de hand van welke (kruid)plant er in staat door middel van ID. De greenhouse moet de juiste sensoren hebben om alles te monitoren en de hardware hebben om deze factoren te regelen.

De kruidenplant moet kunnen overleven zonder dat er enige ingrijpen moeten worden gedaan door de gebruiker. Behalve het waterreservoir vol te houden natuurlijk.

Er moet een RFID reader geplaatst worden zodat de gebruiker kan instellen welke kruidenplant hij erin wilt bewaren (voor elke plant moet er een andere RFID tag ingesteld worden).

Schema:



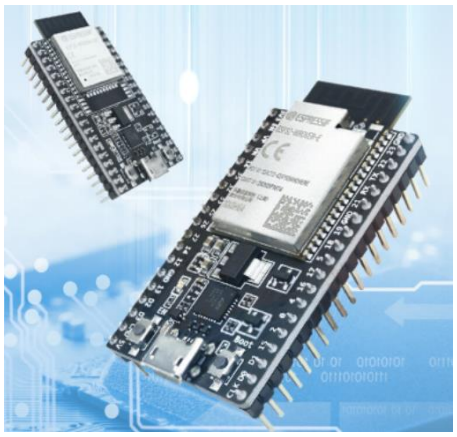


Hardware:

- ESP32
- Raspberry Pi 4
- DHT22
- RFID Reader (een tag voor elke plant)
- LCD
- Moisture Sensor
- Ground temperature sensor
- 3 relays
- Heatbed controller
- 12V fan
- 12V led strip
- 5V pump
- Power Supply (3.3V, 5V, 12V)

ESP32 Dev Module:

Een ESP32 is een kleine microcontroller met ingebouwde bluetooth en wifi modules. Makkelijk programmeerbaar met de Arduino IDE en een micro USB kabel. De ESP is ook breadbord friendly waardoor prototyping heel makkelijk is.



Raspberry Pi 4:

De raspberry Pi is een kleine computer waar je een besturingsysteem kan opzetten via USB-stick of micro SD kaart. Je kan een scherm, toetsenbord en muis aan de raspberry Pi hangen om het te gebruiken als een gewone desktop computer. Met 8GB max RAM geheugen kan deze computer heel veel hetzelfde als een normale werkcomputer. De raspberry Pi 4 support 2 micro-HDMI poorten, het kan ook een ethernet verbinding maken met een netwerk. De raspberry Pi wordt gevoed door een USB-C adapter (5V en 3A).

DHT22:

Een DHT22 is een temperatuur en humidity sensor, verbeterde versie van de DHT11. De DHT22 kan de temperatuur meten met 0.5 precisie. Tegenover de DHT11 die enkel afrond tot 1.

RFID Reader:

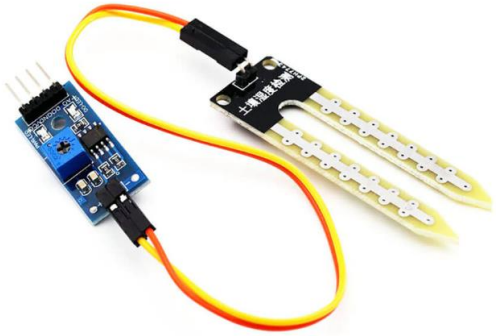
Een RFID reader is een sensor die door middel van een magnetisch veld chips in tags kan snannen. Elke Tag heeft zijn eigen ID en met deze ID kan je je code aanpassen, en zo kan je deze gebruiken als activatie voor je doeleinden.

LCD:

Een LCD scherm is een liquid crystal display. Het scherm dat ik gebruik is 2x16, het scherm heeft dus 2 lijnen waar 16 characters elk op kunnen. Mijn LCD heeft een I²C module aan de achterkant waardoor het makkelijker is om te gebruiken via een library.

Moisture sensor:

Een moisture sensor om de vochtigheid van het water te controlleren. Door middel van de extra module kan je deze waarde digitaal en analoog inlezen. Ik map het analoog signaal naar een percentage om hiermee dan te werken.



Ground temperature sensor:

Dit is een grond temperatuur sensor, deze sensor wordt niet gewoon analoog ingelezen. Deze sensor gebruikt het OneWire protocol, de naam verwijst naar het feit dat deze communicatie door 1 kabel gaat. OneWire is hetzelfde als I²C maar met 1 draad, tragere communicatie en power supply.

Relay:

Een relay is een soort switch dat bedoelt is om hardware aan te sturen dat een hoger voltage nodig heeft dan de controller kan geven.

In dit project gebruik ik 2 dual relays. Met deze relays kan je een externe voeding gebruiken waardoor zelfs het ingang signaal van de controller minder power nodig heeft.

Heatbed Controller:

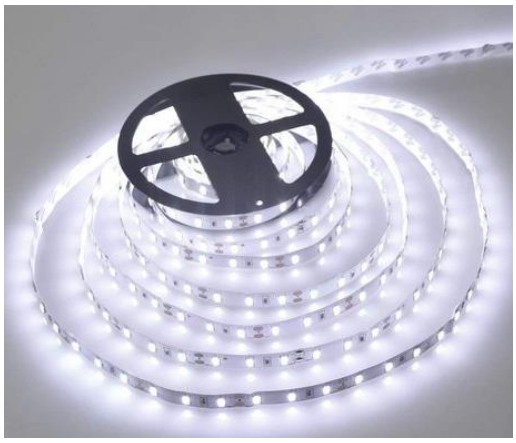
Als warmte element in het project gebruik ik een Heatbed voor 3D printers. Een heatbed kan tot 90 graden Celcius gaan dus hiervoor gebruik ik geen relay. Ook omdat het heatbed een hele hoge amperage nodig heeft, gebruik ik een heatbed controller, deze controller kan tot 25A doorsturen wat nodig is omdat het heatbed minstens 14A nodig heeft en de relay die ik voorginds heb laten zien kan maximum 10A doorgeven. Het goede aan een heatbed controller is ook dat als ik een PWM signaal doorstuur, dat ik de temperatuur van het heatbed kan regelen. Zodat deze niet naar 90 graden gaat want dat zou heel slecht zijn voor een plant.

Fan:

In dit project gebruik ik een fan voor koeling, normaal gezien ging ik een 12V fan gebruiken maar deze is gebroken. Daarom gebruik ik een 24V fan maar zet ik hem op 12V omdat ik geen extra voeding ga plaatsen voor alleen de fan

Led Strip:

Voor verlichting gebruik ik een gewone 12V ledstrip. De amperage nodig voor een ledstrip verandert aan de hand van hoelang de ledstrip is die je wilt voeden. In dit project gebruik ik Cold White Led Strip.



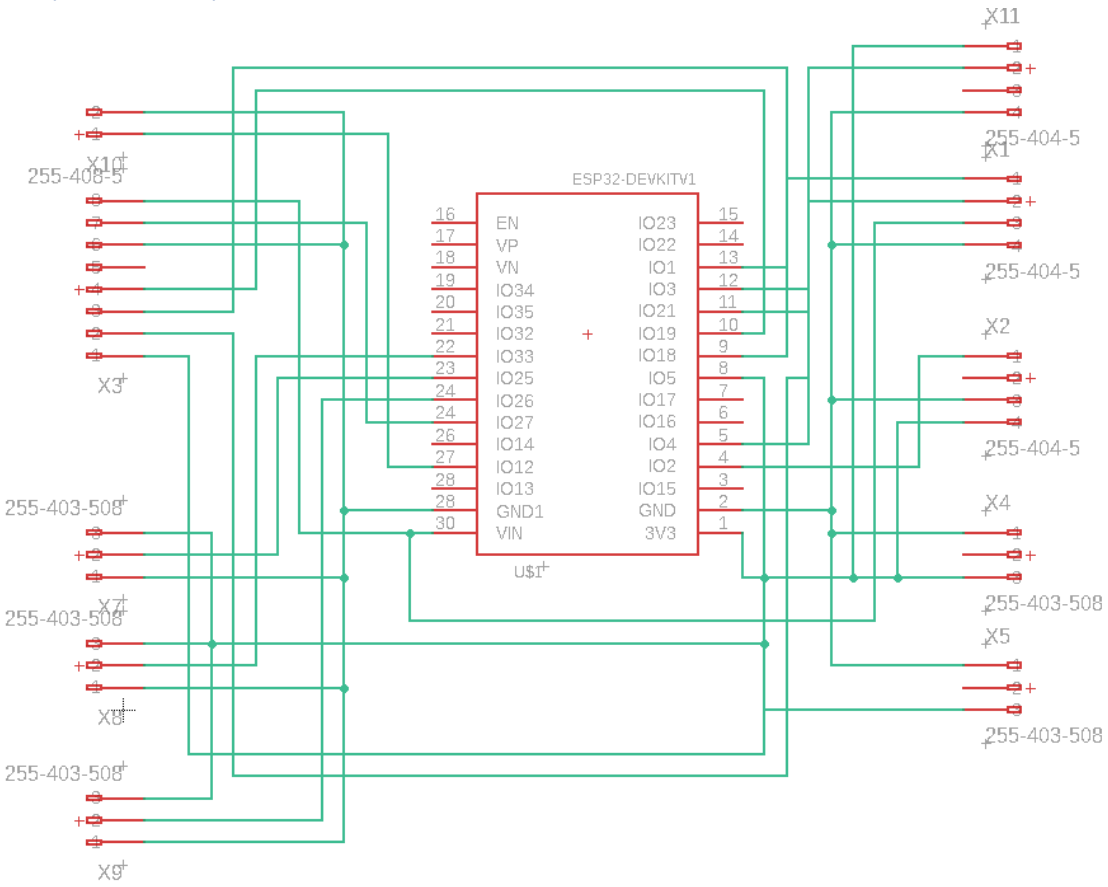
Pump:

Voor het watergeven van de plant gebruik ik een 5V waterpomp die ik met een relay aanstuur en softwarematisch laat ik hem 3 seconden lopen om de 30 seconden als de plant water zou nodig hebben.

Power supply:

Als power supply gebruik ik een oude server voeding. Deze voeding kan enorme amperages voeden op 3.3V, 5V en 12V. Je moet wel weten dat als je een server/computer voeding gebruikt voor een project ga je 1 bepaalde draad vast hangen aan de ground, anders zal de voeding geen stroom voeden. Deze draad is verschillend voor de voeding.

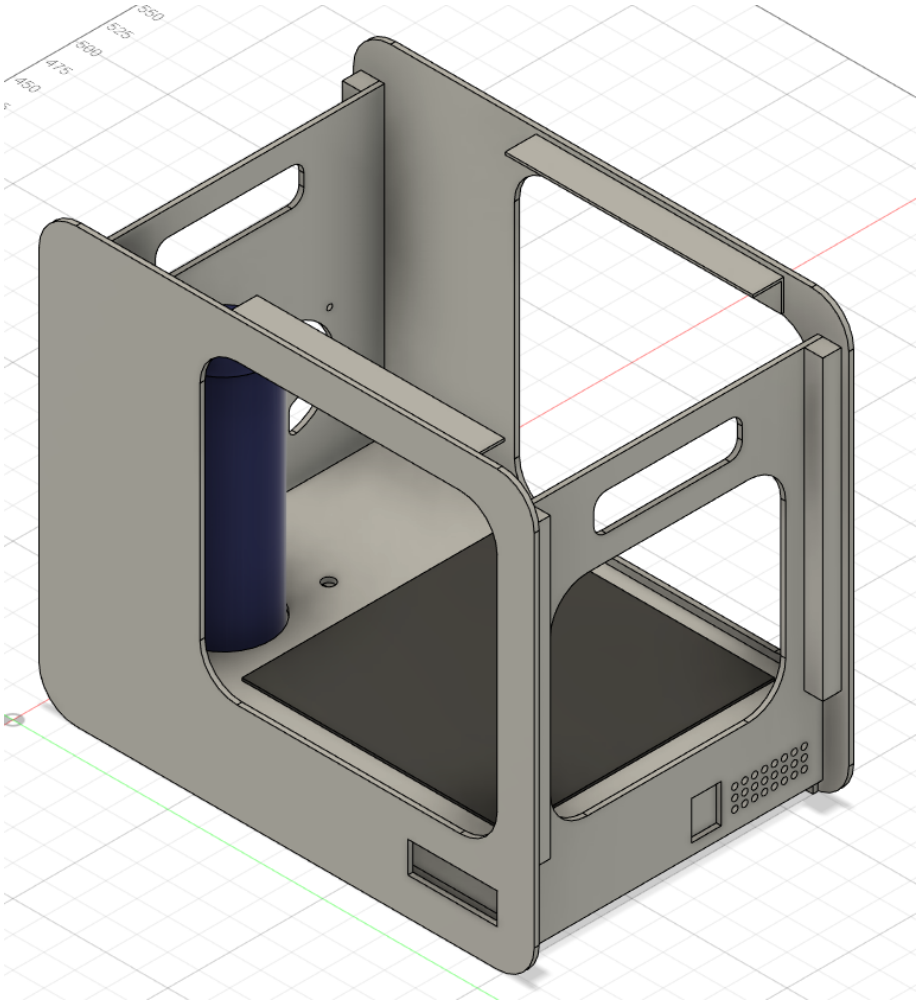
PCB (in EAGLE):



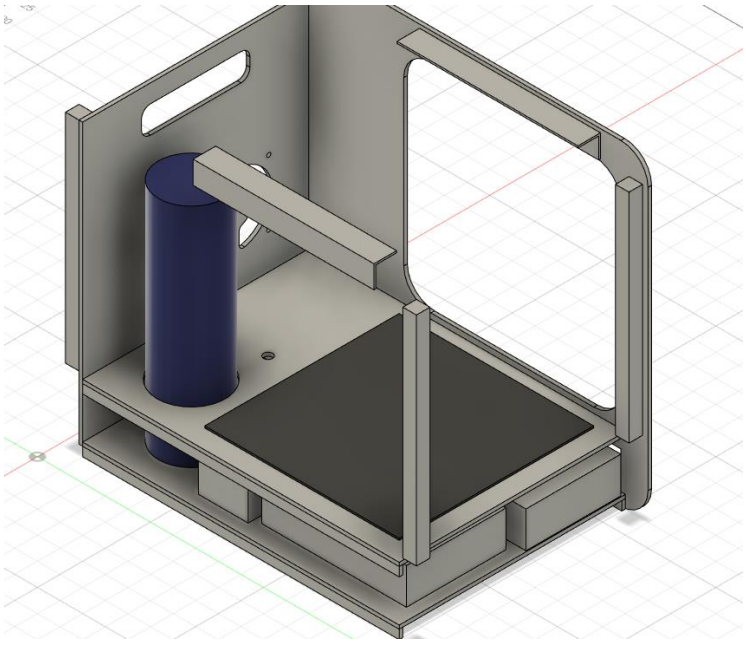
PCD file zit bij de github

Design Greenhouse (in Autodesk Fusion):

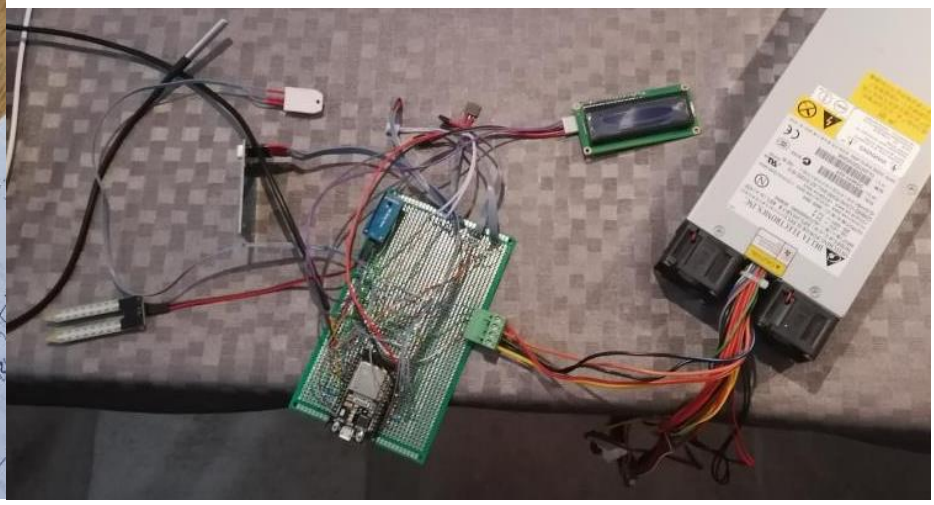
De dimensie van de serre is 40 cm breedt, 25cm diep en 35cm hoog



Er zijn 2 grondplaten, tussen deze twee platen liggen alle elektronika.



De blauwe cilinder is de water tank die ik gebruik (een oude drinkbus). De twee L balken zijn van metaal en hierop zijn de Led strips gemonteerd.

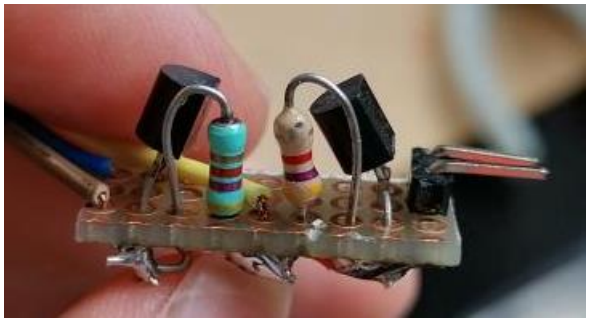


Dit is de elektronika die erin zit. De breedte van de bak is op maat van de voeding (sinds dat de voeding heet grootste is) en de lengte is door het heatbed.

De gaten vanonder in het design zijn voor de fans van de voeding en het gat in de voorkant is voor het lcd scherm.

De RFID reader zit naast het LCD en heeft geen gat nodig omdat deze erdoor kan scannen(heb wel geduld voor de scanner)

Er zit nog 1 component tussen het pwm signaal en de heatbed controller.



Dit zijn 2 transistors gelinkt met 12V en 2 4.7k resistors. Dit zorgt ervoor dat het 3.3V PWM signaal verandert wordt naar een 12V PWM signaal, de rede dat er 2 transistoren gebruikt worden is omdat als je er 1 gebruikt, is het geinverteerd.

Code ESP32:

Dit is de code dat in de ESP32 moet. Deze file zit ook in de github.

```
// Includes

#include <DHT.h>

#include <SPI.h>

#include <WiFi.h>

#include <MFRC522.h>

#include <OneWire.h>

#include <Arduino.h>

#include <PubSubClient.h>

#include <LiquidCrystal_I2C.h>

#include <DallasTemperature.h>


// definitions WiFi

const char* ssid = "MSI 6691";

const char* password = "123456789";


// definitions MQTT

const char* mqttServer = "192.168.137.64";

const int mqttPort = 1883;

const char* mqttUser = "Arnoud";

const char* mqttPassword = "raspberry";

const char* clientID = "client_greenhouse"; // MQTT client ID
```

```
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long publishPeriod = 10000;
unsigned long startPublish;

// Ground temperature sensor
#define SOILPIN  14
OneWire oneWire(SOILPIN);
DallasTemperature DS18B20(&oneWire);

float GroundTemp = 0;

// Temperature and humidity sensor (DHT22)
#define DHTPIN 4  // Pin connected to the DHT sensor
#define DHTTYPE DHT22  // DHT11 or DHT22
DHT dht(DHTPIN, DHTTYPE);

float Temp = 0;
float TempTarget = 0;
float TempRange = 0;
float Humidity = 0;

// Relays
#define RelayPump 26
bool pumpState;
bool pumpInCycle;
unsigned long timeOnPump;
unsigned long timeOffPump;
unsigned long pumpOnPeriod = 3000;
unsigned long pumpOffPeriod = 30000;
```

```
#define RelayFan 25
bool fanState;
#define RelayLights 33
bool lightsState;

// Heat Bed Controller
#define BedController 13
bool heatbedState;
#define PWM_Freq  1000

int DutyCycle = 0;

// Moisture sensor
#define MoisturePin 32

int Moisture = 0;
int MoistureMaxRange = 0;
int MoistureMinRange = 0;
int MoistureTarget = 0;

// RFID reader
#define RST_PIN    2
#define SS_PIN     5

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

byte readCard[4];
String tagID = "";
String Basilicum = "D3FCC119";
String Rozemarijn = "9CDF738";
```

```
String Plant = "Unknown";

// LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);
unsigned long lcdInfoPeriod = 4000;
unsigned long timeLCD;
unsigned long lcdInfoIdx = 0;

void setupWifi() {
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println("Connected to the WiFi network");

  client.setServer(mqttServer, mqttPort);
}

void clearLCD() {
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
}

void splashLCD() {
  clearLCD();
```

```
  lcd.setCursor(0, 0);
  lcd.print("Smart Greenhouse");
  lcd.setCursor(0, 1);
  lcd.print("Greenhouse Smart");
}

void infoPlantLCD() {
  String line1, line2;
  char buffer[40];

  clearLCD();

  sprintf(buffer,"Plant:%s",Plant.c_str());
  line1 = String(buffer);
  sprintf(buffer,"TagID:%s",tagID.c_str());
  line2 = String(buffer);

  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
}

void infoMeasuresLCD() {
  String line1, line2;
  char buffer[40];

  clearLCD();

  sprintf(buffer,"T:%+2.1f H:%+2.1f",Temp,Humidity);
  line1 = String(buffer);
```

```
  sprintf(buffer,"M:%+5d G:%+2.1f",Moisture,GroundTemp);
  line2 = String(buffer);

  // line1 = " T:" + String(Temp, 1) + "  H:" + String(Humidity, 1);
  //line2 = " M:" + String(Moisture) + "  G:" + String(GroundTemp, 1);

  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
}

void infoActuatorsLCD() {
  String line1, line2;

  clearLCD();

  String strFan = "Off";
  if (fanState) strFan = "On ";
  String strLights = "Off";
  if (lightsState) strLights = "On ";
  String strPump = "Off";
  if (pumpState) strPump = "On ";
  String strHeatbed = "Off";
  if (heatbedState) strHeatbed = "On ";

  line1 = " Fa:" +  strFan + "  Ls:" + strLights;
  line2 = " Pu:" +  strPump + "  Hb:" + strHeatbed;

  lcd.setCursor(0, 0);
```

```
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
}

void startPump() {
  digitalWrite(RelayPump, LOW);
  pumpState = true;
  timeOnPump = millis();
}

void stopPump() {
  digitalWrite(RelayPump, HIGH);
  pumpState = false;
  timeOffPump = millis();
}

void cyclePump() {
  unsigned long timeNow = millis();

  if (pumpInCycle) {
    if (pumpState == true && (timeNow - timeOnPump >= pumpOnPeriod)) {
      stopPump();
    }

    if (pumpState == false && (timeNow - timeOffPump >= pumpOffPeriod)) {
      pumpInCycle = false;
    }
  }
  else {
    startPump();
```

```
    pumpInCycle = true;
  }
}

void startFan() {
  digitalWrite(RelayFan, LOW);
  fanState = true;
}

void stopFan() {
  digitalWrite(RelayFan, HIGH);
  fanState = false;
}

void startLights() {
  digitalWrite(RelayLights, LOW);
  lightsState = true;
}

void stopLights() {
  digitalWrite(RelayLights, HIGH);
  lightsState = false;
}

void startHeatbed(int dutycycle) {
  ledcWrite(0, map(dutycycle, 0, 100, 0, 256));
  heatbedState = true;
}

void stopHeatbed() {
  ledcWrite(0, 0);
```

```
  heatbedState = false;
}

void getMeasurements() {
  // Sensor Readings
  DS18B20.requestTemperatures();       // send the command to get temperatures
  float tmpTemp1 = DS18B20.getTempCByIndex(0);  // read temperature in °C
  if (tmpTemp1 > -126.5f) GroundTemp = tmpTemp1;

  float tmpTemp2 = dht.readTemperature();
  if (!isnan(tmpTemp2)) Temp = tmpTemp2;

  float tmpHumidity = dht.readHumidity();
  if (!isnan(tmpHumidity)) Humidity = tmpHumidity;

  Moisture = map(analogRead(MoisturePin), 1024, 4095, 100, 0);
}

boolean getID() {
  // Getting ready for Reading PICCs
  if ( ! mfrc522.PICC_IsNewCardPresent()) { //If a new PICC placed to RFID reader continue
    Serial.println("No new card");
    return false;
  }

  if ( ! mfrc522.PICC_ReadCardSerial()) { //Since a PICC placed get Serial and continue
    Serial.println("Couldn't read card");
    return false;
  }

  tagID = "";
```

```
  for ( uint8_t i = 0; i < 4; i++) { // The MIFARE PICCs that we use have 4 byte UID
    tagID.concat(String(mfrc522.uid.uidByte[i], HEX)); // Adds the 4 bytes in a single String variable
  }
  Serial.println("Card Read");
  tagID.toUpperCase();
  mfrc522.PICC_HaltA(); // Stop reading

  return true;
}

void MqttSendAllData() {
  // Check if WiFi still present
  if (WiFi.status() != WL_CONNECTED) {
    setupWifi();
  }

  if (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect("ESP32Client", mqttUser, mqttPassword )) {
      Serial.println("connected");
    } else {
      Serial.print("failed with state ");
      Serial.print(client.state());
      return;
    }
  }

  if (Plant == "Basilicum") {
    client.publish("GreenHouse/Bak1/GroundTemp", String(GroundTemp).c_str());
    client.publish("GreenHouse/Bak1/Temperature", String(Temp).c_str());
    client.publish("GreenHouse/Bak1/Humidity", String(Humidity).c_str());
```

```
   client.publish("GreenHouse/Bak1/Moisture", String(Moisture).c_str());
   client.publish("GreenHouse/Bak1/Lights", String(lightsState).c_str());
   client.publish("GreenHouse/Bak1/Fan", String(fanState).c_str());
   client.publish("GreenHouse/Bak1/HeatBed", String(heatbedState).c_str());
   client.publish("GreenHouse/Bak1/Pump", String(pumpState).c_str());
  }

  if (Plant == "Rozemarijn") {
   client.publish("GreenHouse/Bak2/GroundTemp", String(GroundTemp).c_str());
   client.publish("GreenHouse/Bak2/Temperature", String(Temp).c_str());
   client.publish("GreenHouse/Bak2/Humidity", String(Humidity).c_str());
   client.publish("GreenHouse/Bak2/Moisture", String(Moisture).c_str());
   client.publish("GreenHouse/Bak2/Lights", String(lightsState).c_str());
   client.publish("GreenHouse/Bak2/Fan", String(fanState).c_str());
   client.publish("GreenHouse/Bak2/HeatBed", String(heatbedState).c_str());
   client.publish("GreenHouse/Bak2/Pump", String(pumpState).c_str());
  }

  Serial.println("Sent Data");
}

void dump() {

  Serial.print("now ");
  Serial.println(millis());
  Serial.print("pumpOn ");
  Serial.println(timeOnPump);
  Serial.print("PumpOff ");
  Serial.println(timeOffPump);
}
```

```
// Main setup function
void setup() {
 // init LCD display
 clearLCD();

 // Splash LCD
 splashLCD();

 // Setup WiFi
 setupWifi();

 // Setup Serial
 Serial.begin(115200);

 // Init ground temperature sensor (DS18B20)
 DS18B20.begin();    // initialize the DS18B20 sensor

 // Init Temperature and humidity sensor (DHT22)
 dht.begin();

 // Init RFID reader
 SPI.begin();      // Init SPI bus
 mfrc522.PCD_Init();   // Init MFRC522
 delay(4);       // Optional delay. Some board do need more time after init to be ready, see Readme
 mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD - MFRC522 Card Reader details

 // pinmodes
 pinMode(MoisturePin, INPUT);
 pinMode(RelayLights, OUTPUT);
 pinMode(RelayFan, OUTPUT);
 pinMode(RelayPump, OUTPUT);
```

```
  // PWM init
  ledcAttachPin(BedController, 0);
  ledcSetup(0, PWM_Freq, 8);

  // Init actuators to default state
  stopLights();
  stopFan();
  stopHeatbed();
  stopPump();

  startPublish = millis();
  timeLCD = millis();
  lcdInfoIdx = 0;

  timeOnPump = millis();
  timeOffPump = timeOnPump;
  pumpInCycle = false;

  Plant = "Unknown";
  TempTarget = 0;
  TempRange = 100;
  MoistureTarget = 0;
  MoistureMaxRange = 100;
  MoistureMinRange = 100;

}

// Main loop function
void loop() {
```

```
unsigned long timeNow = millis();


// Check if we need to set a new plant
if (getID())
{
 // If a new plant is requested, set new parameters
 if (tagID == Basilicum && Plant != "Basilicum") {
  startLights();
  TempTarget = 20;
  TempRange = 2;
  MoistureTarget = 50;
  MoistureMaxRange = 10;
  MoistureMinRange = 15;
  Plant = "Basilicum";
 }
 else if (tagID == Rozemarijn && Plant != "Rozemarijn") {
  startLights();
  TempTarget = 25;
  TempRange = 2;
  MoistureTarget = 50;
  MoistureMaxRange = 10;
  MoistureMinRange = 5;
  Plant = "Rozemarijn";
 }
 else {
  if (Plant != "Unknown") {
   stopLights();
   TempTarget = 0;
   TempRange = 100;
   MoistureTarget = 0;
   MoistureMaxRange = 100;
```

```
      MoistureMinRange = 100;
      Plant = "Unknown";
    }
  }
 }

 getMeasurements();

 // Compare mean(groundtemperature, temperature) with desired value, if lower than desired,
activate heatbed
 float meanTemp = (GroundTemp + Temp) / 2.0f;
 Serial.println(meanTemp);
 Serial.println(TempTarget);
 Serial.println(TempRange);
 if (meanTemp < TempTarget - TempRange) {
  Serial.println("cold");
  stopFan();
  startHeatbed(50);
 }
 else if (meanTemp > TempTarget + TempRange) {
  Serial.println("hot");
  stopHeatbed();
  startFan();
 }
 else if ( meanTemp < TempTarget + TempRange && meanTemp > TempTarget - TempRange) {
  Serial.println("perfect");
  stopHeatbed();
  stopFan();
 }

 // Compare moisture with desired value and activate pump if too low.
```

```
 if (Moisture < MoistureTarget - MoistureMinRange) {
  cyclePump();
 }

 if (timeNow - timeLCD >= lcdInfoPeriod) {
  switch (lcdInfoIdx) {
   case 0:
    infoPlantLCD();
    break;
   case 1:
    infoMeasuresLCD();
    break;
   case 2:
    infoActuatorsLCD();
    break;
   default:
    break;
  }
  lcdInfoIdx++;
  if (lcdInfoIdx > 2) lcdInfoIdx = 0;
  timeLCD = timeNow;
 }

 if (timeNow - startPublish >= publishPeriod) {
  MqttSendAllData();
  startPublish = timeNow;
 }
 delay(50);
}
```

Code Raspberry Pi:

Dit is het script van de raspberry pi waar alle data die door de ESP verstuurt wordt via mqtt direct in de database wordt gezet. Als je dan de grafana service op de raspberry pi runned dan kan je op het IP address van de Pi grafana openen end e databank laten uitlezen.

```
import re
from typing import NamedTuple
import paho.mqtt.client as mqtt
from influxdb import InfluxDBClient
INFLUXDB_ADDRESS = '192.168.137.64'
INFLUXDB_USER = 'Arnoud'
INFLUXDB_PASSWORD = 'raspberry'
INFLUXDB_DATABASE = 'GreenHouse'

MQTT_ADDRESS = '192.168.137.64'
MQTT_USER = 'Arnoud'
MQTT_PASSWORD = 'raspberry'
MQTT_TOPIC = 'GreenHouse/+/+'
MQTT_REGEX = 'GreenHouse/([^/]+)/([^/]+)'
MQTT_CLIENT_ID = 'MQTTInfluxDBBridge'
influxdb_client = InfluxDBClient(INFLUXDB_ADDRESS, 8086, INFLUXDB_USER,
INFLUXDB_PASSWORD, None)
class SensorData(NamedTuple):
    location: str
    measurement: str
    value: float
def on_connect(client, userdata, flags, rc):
    """ The callback for when the client receives a CONNACK response from the
server."""
    print('Connected with result code ' + str(rc))
    client.subscribe(MQTT_TOPIC)
def _parse_mqtt_message(topic, payload):
```

```
    match = re.match(MQTT_REGEX, topic)
    if match:
        location = match.group(1)
        measurement = match.group(2)
        if measurement == 'status':
            return None
        return SensorData(location, measurement, float(payload))
    else:
        return None
def _send_sensor_data_to_influxdb(sensor_data):
    json_body = [
        {
            'measurement': sensor_data.measurement,
            'tags': {
                'location': sensor_data.location
            },
            'fields': {
                'value': sensor_data.value
            }
        }
    ]
    influxdb_client.write_points(json_body)
def on_message(client, userdata, msg):
    """The callback for when a PUBLISH message is received from the server."""
    print(msg.topic + ' ' + str(msg.payload))
    sensor_data = _parse_mqtt_message(msg.topic, msg.payload.decode('utf-8'))
    if sensor_data is not None:
        _send_sensor_data_to_influxdb(sensor_data)
def _init_influxdb_database():
    databases = influxdb_client.get_list_database()
    if len(list(filter(lambda x: x['name'] == INFLUXDB_DATABASE, databases))) == 0:
```

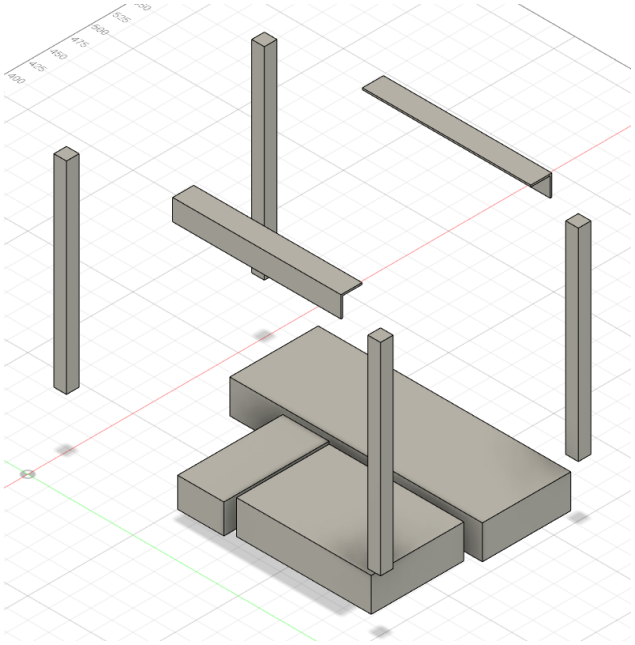
```
        influxdb_client.create_database(INFLUXDB_DATABASE)
    influxdb_client.switch_database(INFLUXDB_DATABASE)
def main():
    _init_influxdb_database()
    mqtt_client = mqtt.Client(MQTT_CLIENT_ID)
    mqtt_client.username_pw_set(MQTT_USER, MQTT_PASSWORD)
    mqtt_client.on_connect = on_connect
    mqtt_client.on_message = on_message
    mqtt_client.connect(MQTT_ADDRESS, 1883)
    mqtt_client.loop_forever()
if __name__ == '__main__':
    print('MQTT to InfluxDB bridge')
    main()
```

Build tutorial:

Om dit project te bouwen zal je eerst de componenten moeten kopen en de PCB laten maken.

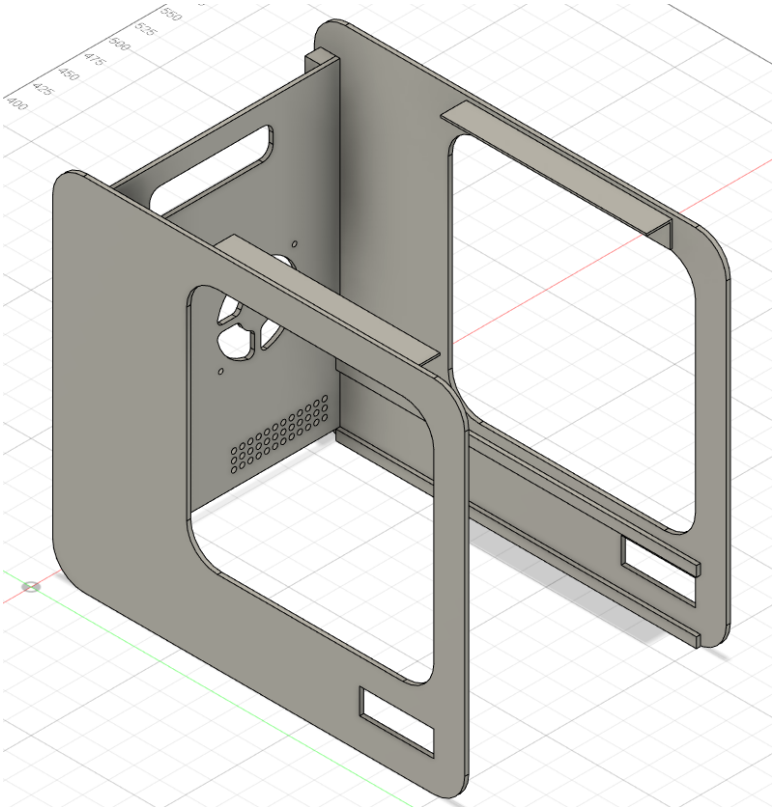
Je kan ook meteen de files van fusion laten snijden (de file van fusion en PCB zitten bij de github).

NOTE: de bak is gesneden uit MDF hout dat 4mm dik is. Als je dit wilt veranderen zal je zelf de nodige parameters moeten veranderen aan de fusion file (dit kan wel een tijdje duren als je niet ervaren bent met fusion).

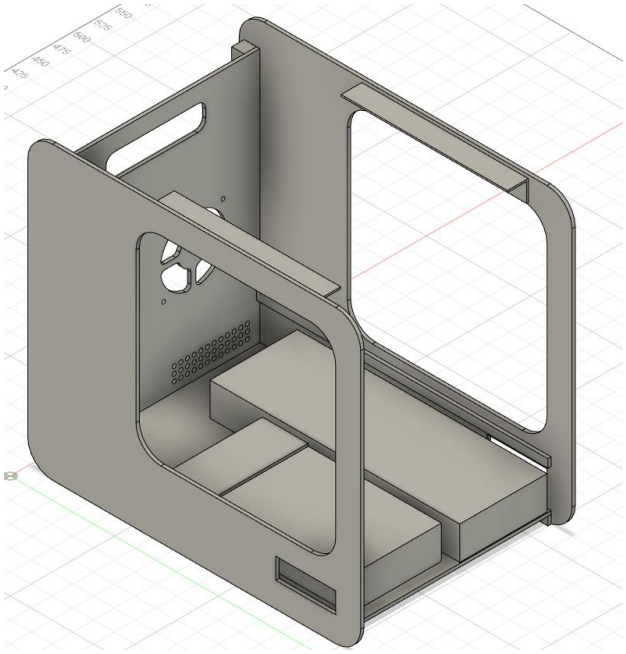


Deze onderdelen zijn niet uit hout. De blokken beneden zijn gewoon de elektronische componenten heel simpel uitgedrukt. De 2 L balken zijn metaal en voor de led strips. En de 4 verticale balkjes zijn ook van metaal (15x15mm hol alluminium), deze zijn voor de bouten om de bak in elkaar te zetten.

Wat je eerst wil doen als je deze bak wilt in elkaar steken, is al het soldeerwerk van de PCB, test zeker eerst alles op voorhand voor je de bak in elkaar steekt. Eens het werkt start je het opbouwen met dit:

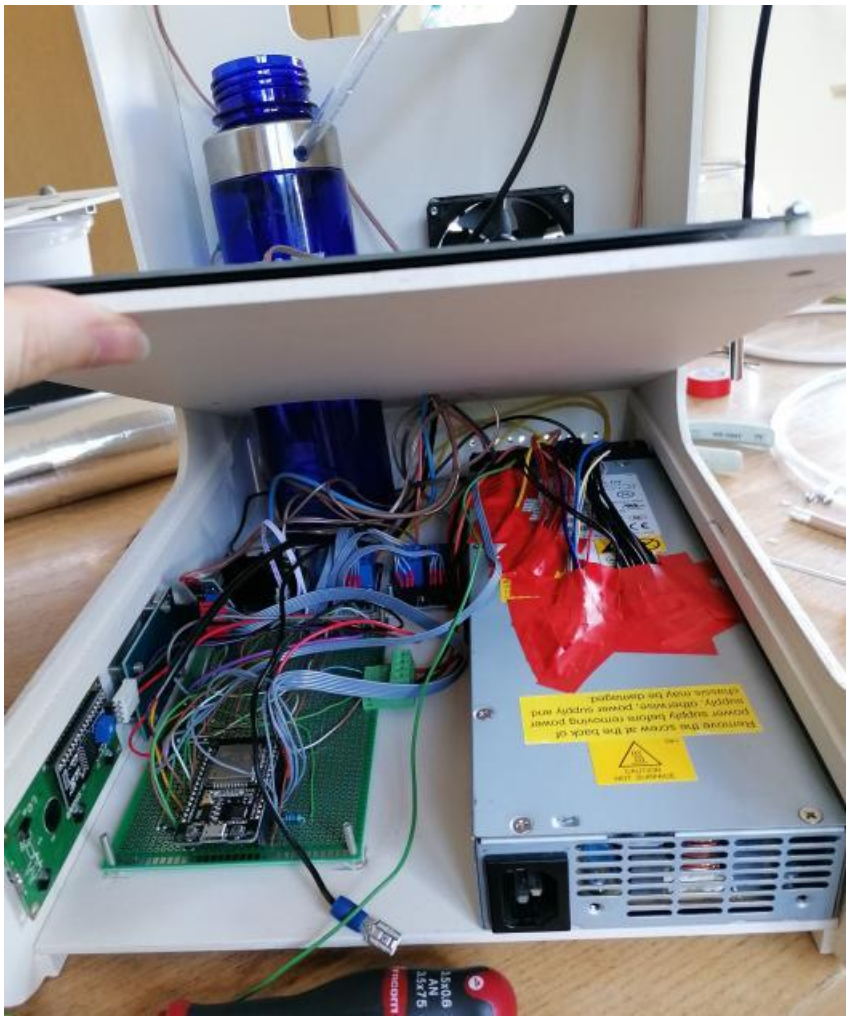


Gebruik 2 van de metalen balken em de wanden aan elkaar te connecteren. Je zal de gaten voor de bouten moeten maken in beide de balk en de wand, vergeet niet deftig te meten. De afmeting van de bouten is voor jezelf om te kiezen welke breedte je gebruikt, ga wel niet over een diameter van 8mm want anders gaan de alluminium balken niet stevig genoeg zijn voor het gewicht. Hierna kan je de vloer erin leggen, deze wordt niet vastgemaakt het werkt zoals lades in een kast steken de zijstrips houden ze tegen. Eens alles aan elkaar hangt zal het heel stevig zijn dus het is een case van ‘Trust the process’. Eens je de eerste vloer er insteekt, zet je alle elektronica in orde want sommige kabels gaan door de tweede vloer moeten gaan.

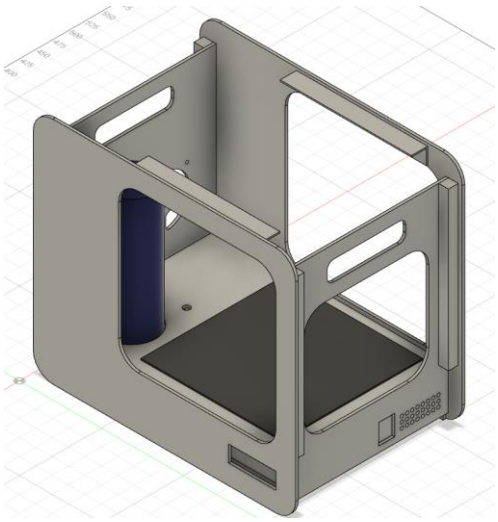


Test zeker nog eens de elektronika voor je verder gaat. Als alles werkt, zet je de tweede vloer erin.

NOTE: vergeet niet de kabels die door de tweede vloer moeten komen, deze kabels zijn de sensoren en de kabels die uit de relays komen. Het lastigste deel is het heatbed sinds die vast staat op de tweede vloer en zal je hem schuin moeten houden om hem in de controller te vijzen.



Eens alle elektronica er inzit en de tweede vloerplaat erin ligt, kan je de laatste wand monteren.



Hierna zet je de pot met je plant in, liefst niets dat geleid, ik heb een glazen pot gebruikt omdat deze makkelijker temperatuur regelbaar is. Eens je je pot hebt zet je de sensoren erin en de waterpomp tube en je start de greenhouse door gewoon de power cable in te steken.