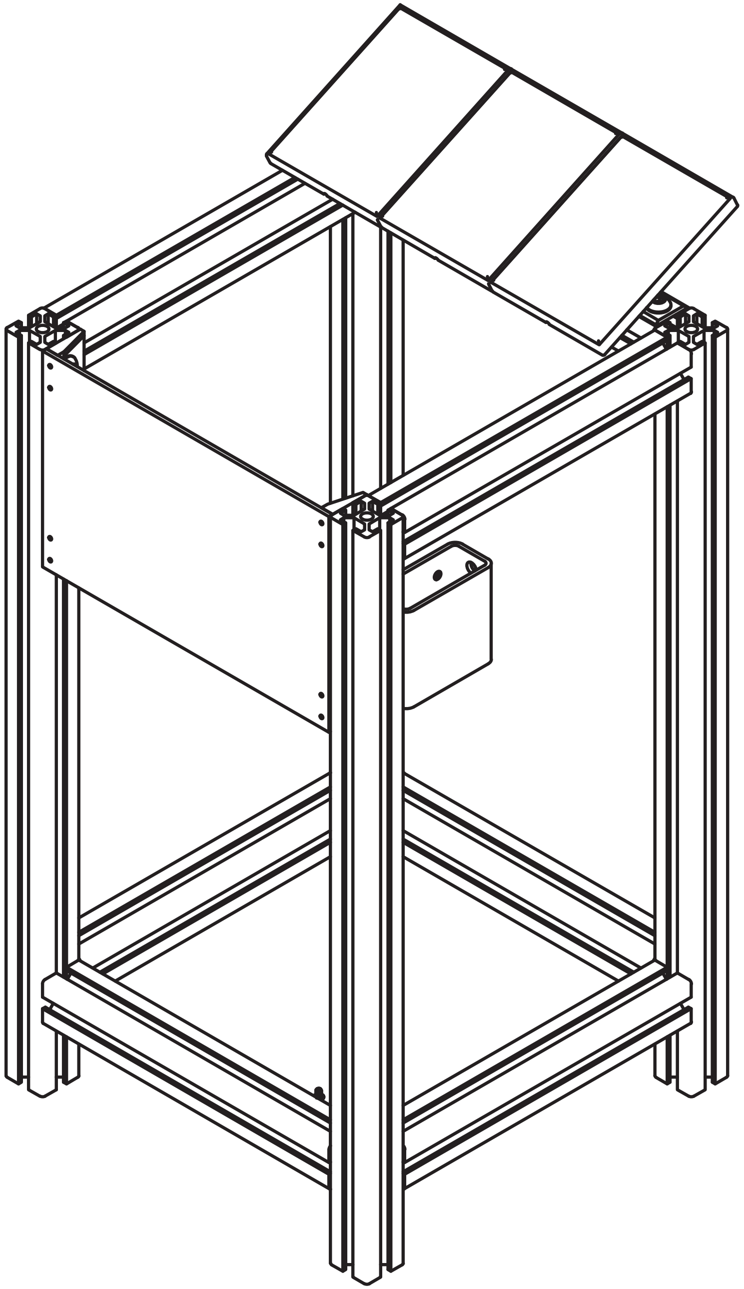


Thread Dripper V1

By Corner Crew

Lukas, Jakob, Philine

Table of Contents

Project Overview

Smart home objects are designed for seamless comfort. They are ubiquitous black boxes that promise convenience, obscuring their true workings and environmental costs. This course will question the nature of “smart” objects through deliberate acts of inconvenience and revelation. What emerges when we strip away optimization to expose networks, environments, usage, and change?

We will explore the foundations of physical computing and create objects across multiple dimensions: physical objects, open-source designs, and as parts of a whole. We will develop deliberately inconvenient objects that challenge assumptions about comfort, automation, and the relationship between interaction and the spatial environments that allow them. Our projects will have minimal resource use, allow repairability, cultivate exploration, and ideation. Our objects will be starting points for interactive ecologies that can be cultivated, modified, and grown by others and for others.

The course will combine workshops with online guest lectures. Students will learn the foundations of networks, sensors, basic machine learning, and 3D parametric design to examine always-on connectivity’s environmental and social implications. Through hands-on prototyping, we’ll explore how to inconvenience and cultivate connection using permacomputing principles.

Context

At the beginning of the project, we questioned the general areas of application of smart homes. To what extent have they been implemented as support in the everyday lives of different groups of people and how successful have they been in application? Which areas of the original smarthome are still relevant today and in which areas is the supposed smarthome ultimately not as intelligent as it seems? All computing applications consume a certain amount of energy. How much do we need for our application and how much is really necessary? Our aim is to work as energy-efficiently as possible and to measure our energy consumption for all actions. This is not only related to the current energy consumption of our applications, but also as a reminder of the general need for digital improvements for the optimization of objects and processes in domestic areas. Where is the interface between the energy efficiency of electronic processes and previously physical processes? This raises the question: is it really necessary to technically optimize everyday actions?

Ideation

In everyday life, there are a number of repetitive processes that are a natural part of our daily routine. These include cleaning reusable objects such as dishes and pots, maintaining hygienic conditions in our environment through superficial cleaning and care work on fellow human beings, animals and other living beings such as plants.

We took a closer look at the process of caring for and watering plants. When caring for plants, there are a number of factors that should not be ignored in order to ensure the health of the plant. These factors include:

Watering that is adapted to the plant species and the time of year.
Exposure to light and the right amount of light to create ideal conditions for growth.
Humidity, especially to supply certain plant species with aerial roots.
The water level in the pot to prevent waterlogging and thus root rot.
The temperature, as it influences the soil moisture and thus the plant's water supply, both at low and high values.

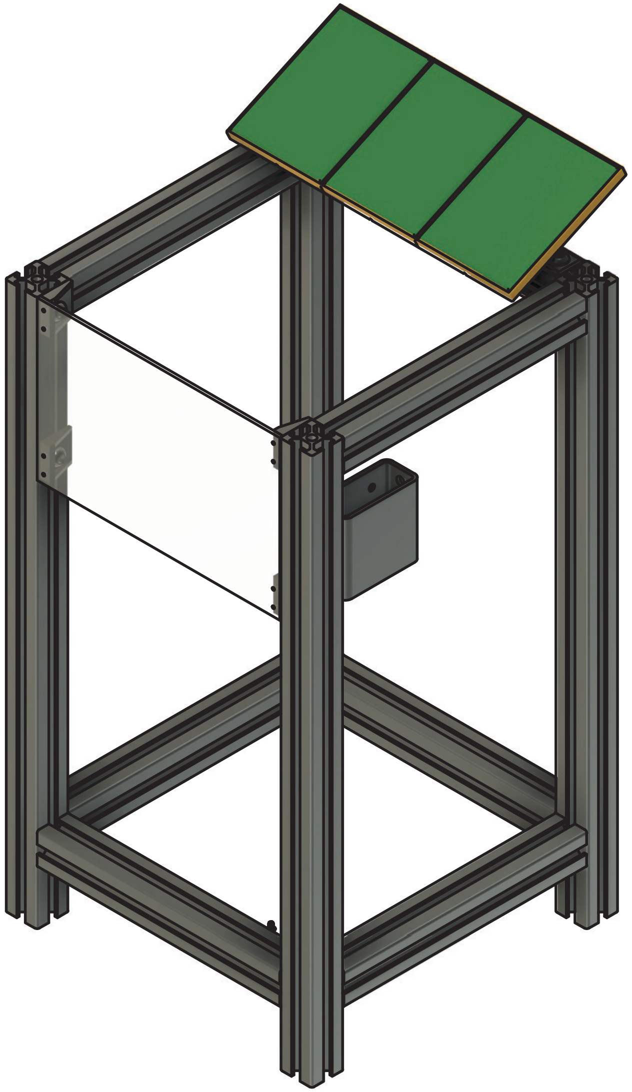
Most experienced plant owners can intuitively recognize all of these conditions. No additional technical devices are required.
This is where our idea comes in:
To what level of abstraction can we raise these diverse measurements with the help of technical modules? Can we automate the observation and care of different plant species so that they can be cared for without human intervention? How far would technical measurement systems have to go in order to monitor all these aspects and execute them autonomously through an autonomously programmed system?

And the question that led us to implement this project is:
How far must a technical construction be oversized and technologically exaggerated in order to exaggerate the simple act of watering plants in such a way that it critically reflects the current trend towards supercomputing and digital maximalism - and thus provides food for thought for a consumption-critical society?

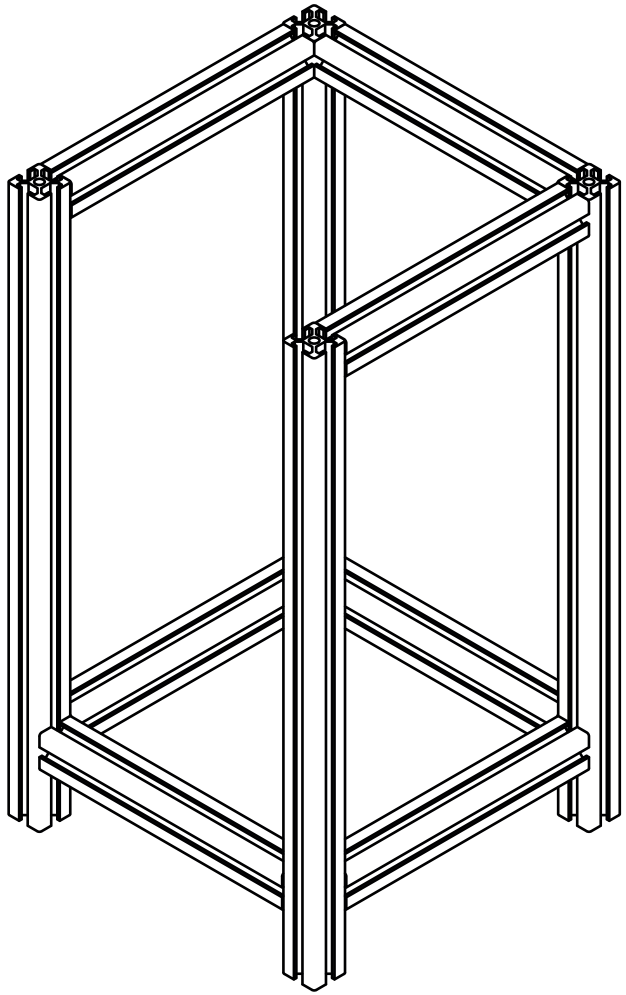
Permacomputing

“Permacomputing” is a manifesto for more sustainable and resilient approaches to computing, written by Marloes de Valk and Ville-Matias Heikkilä in 2022. It draws inspiration from permaculture — a regenerative, sustainable way of interacting with natural systems. Translated into the realm of technology, this means reducing waste, prioritizing repair, embracing long-term usability, and fostering adaptability. The text responds to the severe environmental consequences of our digital age — from high energy consumption and electronic waste to planned obsolescence — and offers a counter-narrative to Big Tech’s logic of short-term profit and perpetual growth.

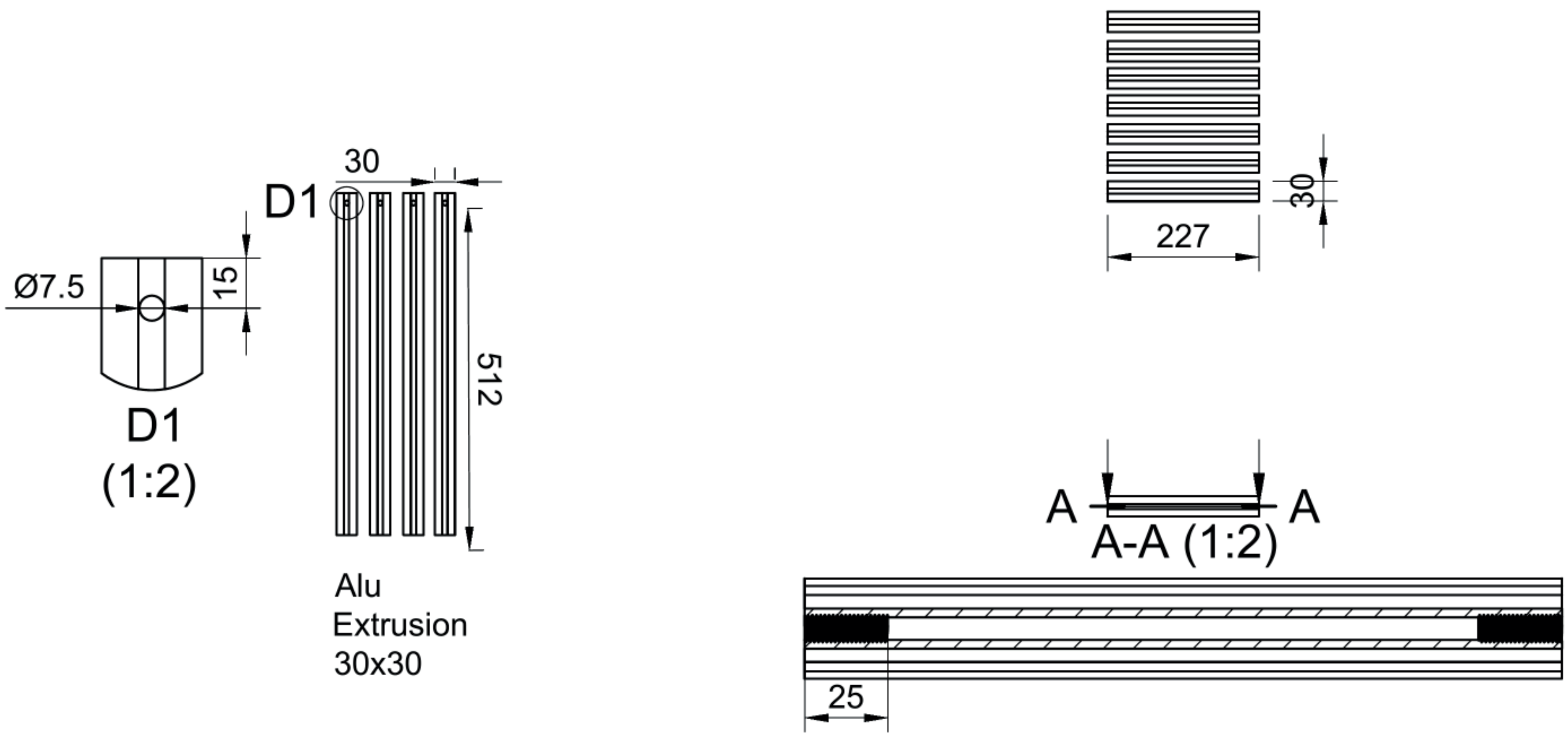
In a similar spirit, our project is designed to provoke by embracing exaggeration. Through the intentional use of excessive technical complexity, we push the idea of technological maximalism to an absurd extreme. This over-engineering becomes a tool for critique: by satirizing the excesses of consumer technology, we invite reflection on sustainability, necessity, and the values that shape our relationship with technology.

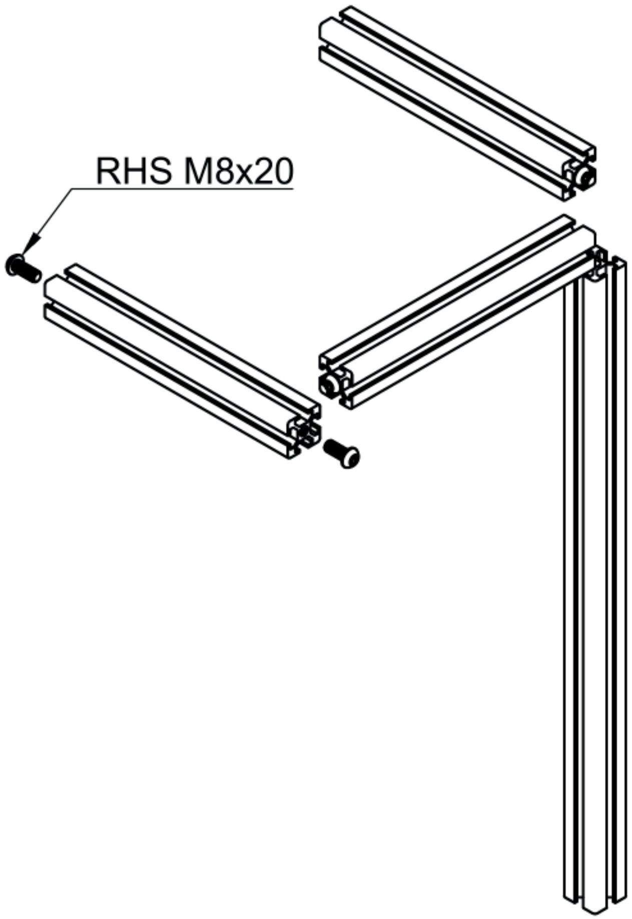


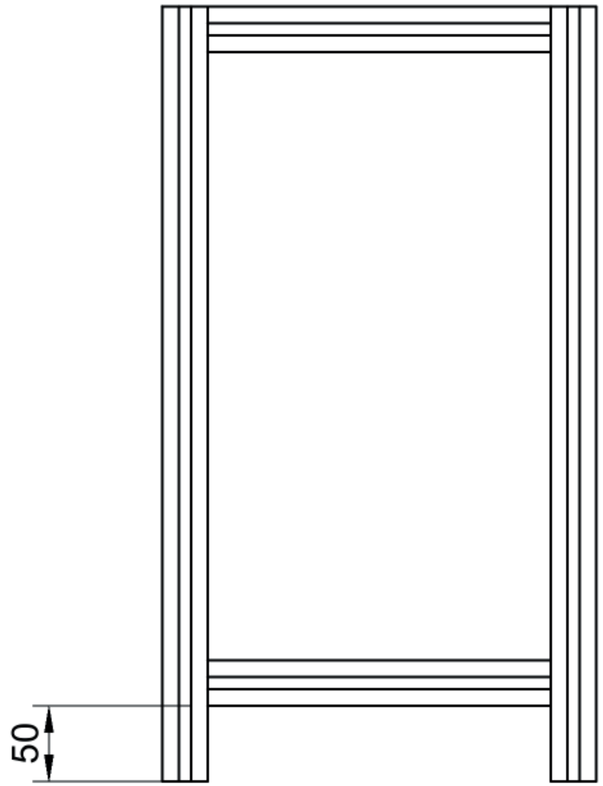
Technical Product Development

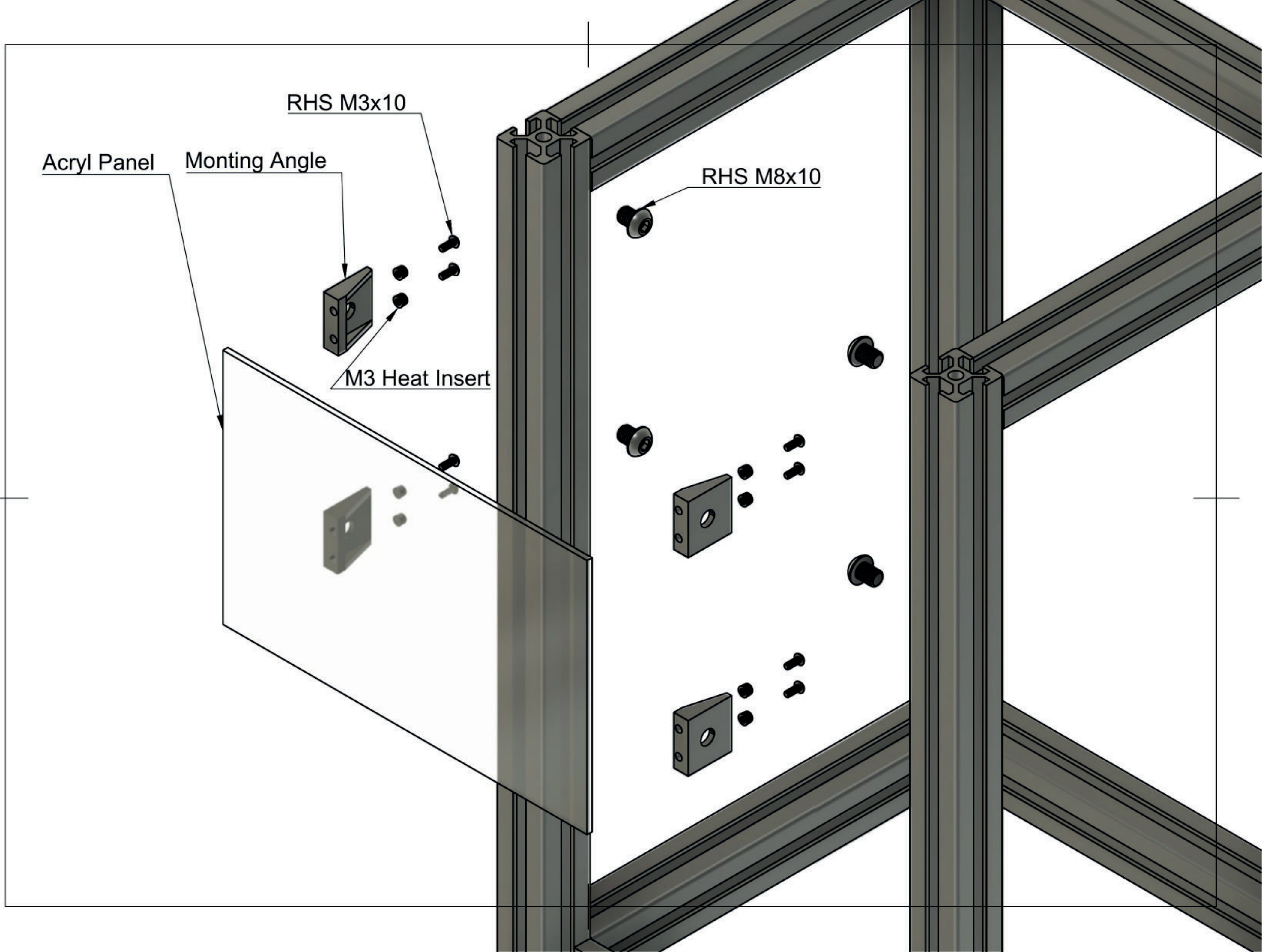


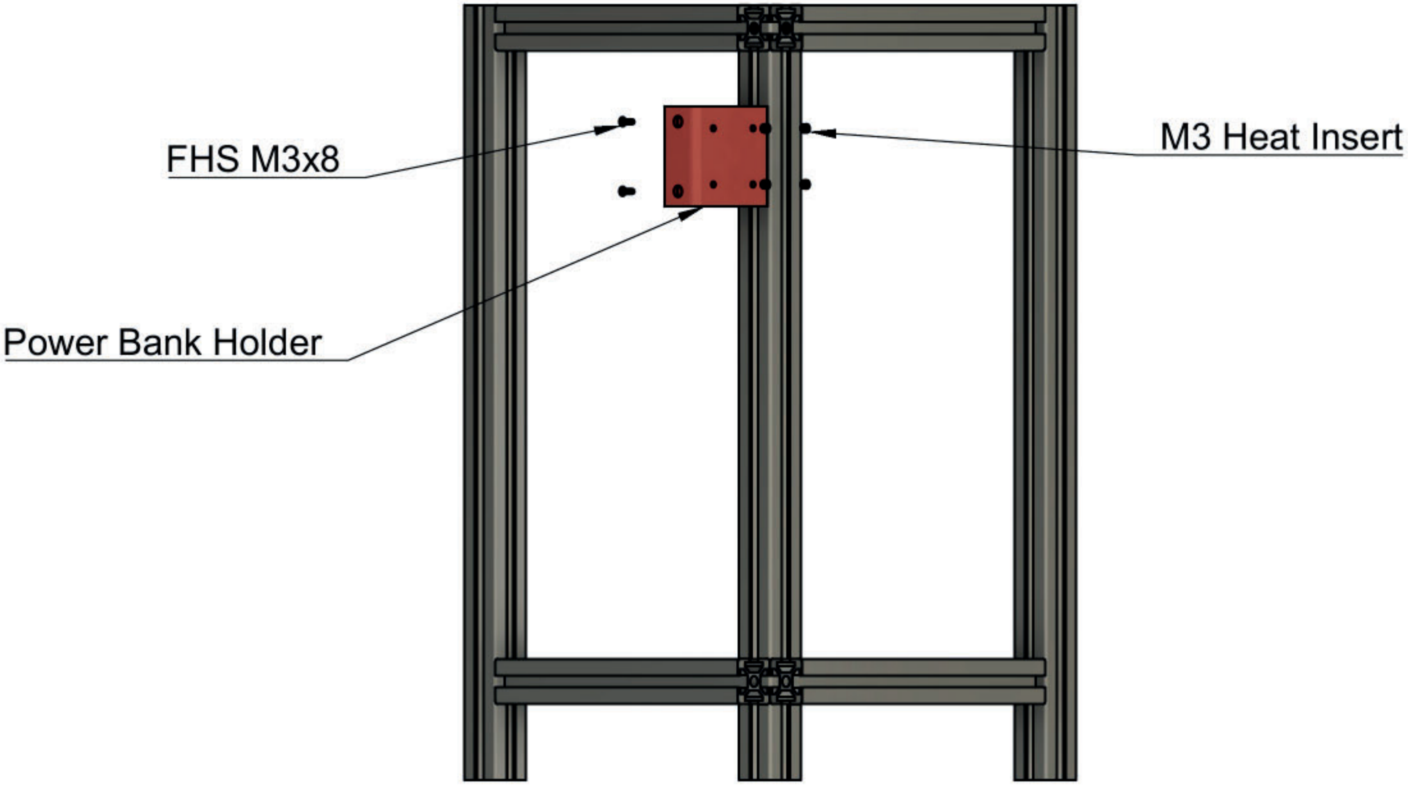
Building of the Frame

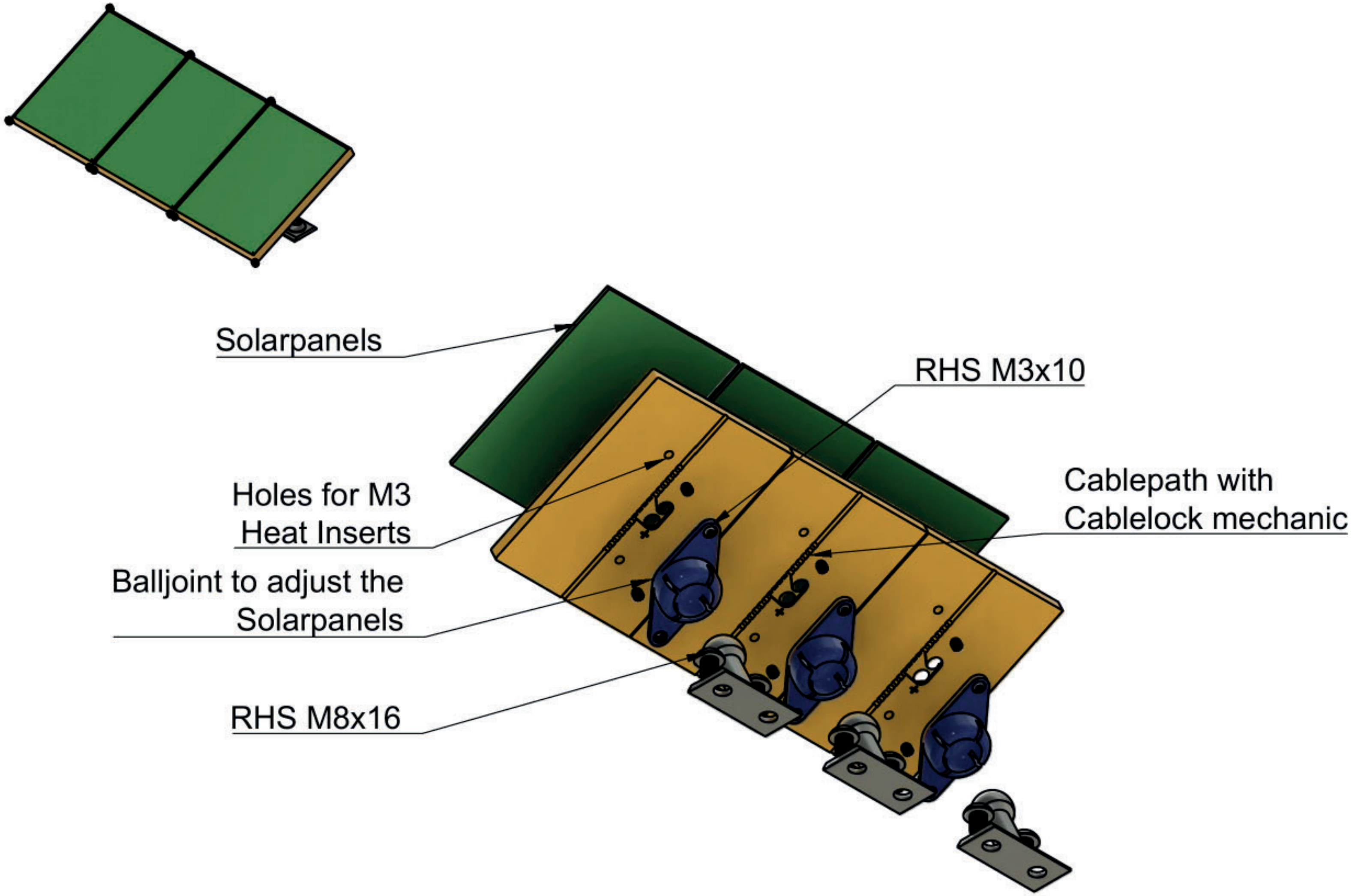

30
D1
Ø7.5
15
D1
(1:2)
512
Alu
Extrusion
30x30
30
227
A
A
A-A (1:2)
25


RHS M8x20


50


RHS M3x10
Acryl Panel
Monting Angle
RHS M8x10
M3 Heat Insert

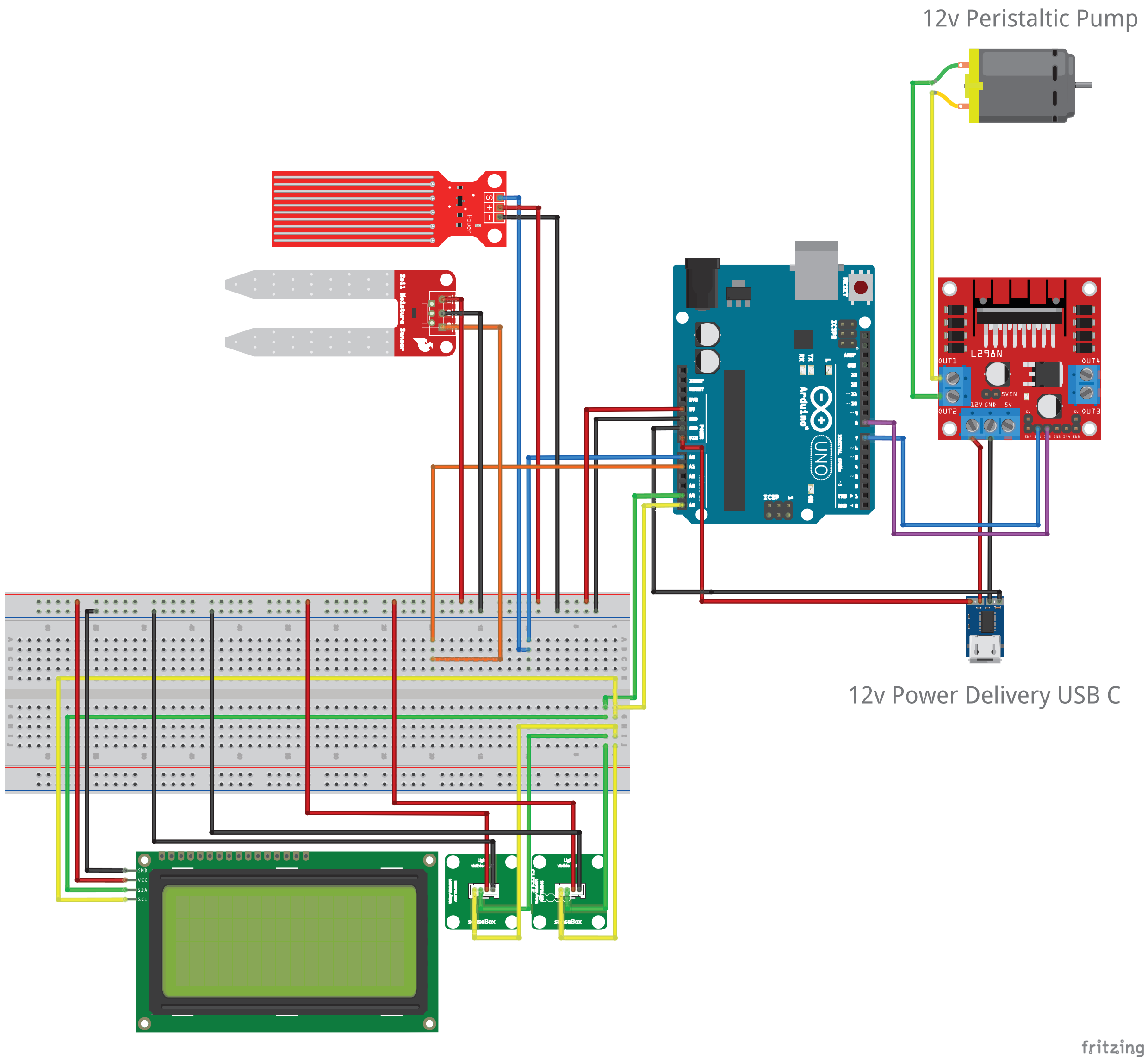

FHS M3x8
M3 Heat Insert
Power Bank Holder


Solarpanels
RHS M3x10
Holes for M3
Heat Inserts
Cablepath with
Cablelock mechanic
Balljoint to adjust the
Solarpanels
RHS M8x16

Electronics

Controllers, Drivers & Sensors

Category	Part	Notes
MCU	Arduino Uno R3	Primary microcontroller
Display	20 × 4 I^2C LCD	incl. driver module (check the I^2C address)
UV sensor	VEML6070	senseBox 1 component, two sensors are combined
Light sensor	TSL45315	senseBox 1 component, two sensors are combined
Temperature / Humidity	HDC1000	senseBox 2 component, two sensors are combined
Soil-moisture	Capacitive analog sensor	—
Water-level	Analog probe	—
Motor driver	L298N	—
Pump	Peristaltic pump	12 Volt
Solar panels	5 V 1.5 Watts	Not implemented yet


12v Peristaltic Pump
L298N
OUT1
OUT2
OUT3
OUT4
UNO
Arduino
12v Power Delivery USB C
fritzing

Code

```
lux, [21.05.2025 16:09]
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_VEML6070.h>
#include <Makerblog_TSL45315.h>
#include <Adafruit_HDC1000.h>
#include <L298N.h>

// === LCD Setup ===
LiquidCrystal_I2C lcd(0x26, 20, 4);  //
I2C address 0x26, 20 columns, 4 rows

// === Sensor Objects ===
Adafruit_VEML6070 uvSensor;
Makerblog_TSL45315
lightSensor(TSL45315_TIME_M4); //
400ms integration time
Adafruit_HDC1000 hdcSensor =
Adafruit_HDC1000();  // Temperature
and humidity sensor

// === Sensor Values ===
int waterLevel = 0;
int soilMoisture = 0;
uint16_t uvIndex = 0;
uint32_t lightLux = 0;
float temperature = 0.0;
float humidity = 0.0;

// === Motor Pins ===
const unsigned int MOTOR_IN1 = 7;
const unsigned int MOTOR_IN2 = 8;
L298N motor(MOTOR_IN1, MOTOR_
IN2);  // Motor object using L298N

// === Condition Thresholds
(Configuration Section) ===
const float TEMP_THRESHOLD =
15.0;        // Minimum temperature
const float HUMIDITY_MAX = 50.0;
// Maximum humidity
const uint16_t UV_THRESHOLD = 0;
// Minimum UV index
const uint32_t LIGHT_THRESHOLD =
100;      // Minimum light level
const int SOIL_MOISTURE_MAX =
100;        // Maximum soil moisture
const int WATER_LEVEL_MIN = 500;
// Minimum water tank level

// === Motor Runtime Configuration
===
const unsigned long MOTOR_RUN_
TIME_MS = 500; // Motor runs for
0.5 seconds

// === Internal motor state tracking
===
bool motorIsRunning = false;
unsigned long motorStartTime = 0;

// === Check all environmental
conditions ===
bool all_conditions_met(float temp,
float hum, uint16_t uv, uint32_t lux,
int soil, int water) {
  return (temp > TEMP_THRESHOLD
&&
          hum < HUMIDITY_MAX &&
          uv > UV_THRESHOLD &&
          lux > LIGHT_THRESHOLD &&
          soil < SOIL_MOISTURE_MAX
&&
          water > WATER_LEVEL_MIN);
}

// === Control motor based on
condition ===
void control_motor(bool shouldRun)
{
  if (shouldRun) {
    motor.forward();                    //
(Re)start motor
    motorStartTime = millis();
// Reset timer
    motorIsRunning = true;
  }

  // Stop motor after the configured
duration
  if (motorIsRunning && (millis()
- motorStartTime >= MOTOR_RUN_
TIME_MS)) {
    motor.stop();
    motorIsRunning = false;
  }
}

// === Arduino Setup ===
void setup() {
  Serial.begin(9600);
  Wire.begin();

  lcd.init();
  lcd.backlight();
```

```
  uvSensor.begin(VEML6070_1_T);
  lightSensor.begin();
  hdcSensor.begin(); // Just init, no
fail handling here

  Serial.println("All sensors
initialized.");
}

// === Main Loop ===
void loop() {
  // === Read all sensor values ===
  waterLevel = analogRead(A0);
  soilMoisture = analogRead(A1);
  uvIndex = uvSensor.readUV();
  lightLux = lightSensor.readLux();
  temperature = hdcSensor.
readTemperature();
  humidity = hdcSensor.
readHumidity();

  // === Serial Monitor Output ===
  Serial.print("Water Level: "); Serial.
println(waterLevel);
  Serial.print("Soil Moisture: "); Serial.
println(soilMoisture);
  Serial.print("UV Index: "); Serial.
println(uvIndex);
  Serial.print("Light Lux: "); Serial.
println(lightLux);
  Serial.print("Temperature: "); Serial.
print(temperature); Serial.println("
C");
  Serial.print("Humidity: "); Serial.
print(humidity); Serial.println(" %");

  // === Check Conditions ===
  bool conditionsOK = all_conditions_
met(temperature, humidity, uvIndex,
lightLux, soilMoisture, waterLevel);

  // === LCD Output ===
  lcd.clear();

  if (conditionsOK) {
    // Display only soil moisture and
motor status
    lcd.setCursor(0, 1);
    lcd.print("Soil Moisture: ");
    lcd.print(soilMoisture);

    lcd.setCursor(3, 2); // Centered
(approx) for 20 columns
    lcd.print(">> MOTOR ON <<");
  } else {
    // Full sensor data display
    lcd.setCursor(0, 0);
    lcd.print("Water: ");
    lcd.print(waterLevel);
    lcd.setCursor(10, 0);
    lcd.print("Soil: ");
    lcd.print(soilMoisture);

    lcd.setCursor(0, 1);
    lcd.print("UV: ");
    lcd.print(uvIndex);
    lcd.setCursor(10, 1);
    lcd.print("Lux: ");
    lcd.print(lightLux);

lux, [21.05.2025 16:09]
lcd.setCursor(0, 2);
    lcd.print("Temp: ");
    lcd.print(round(temperature));
    lcd.print("C");
    lcd.setCursor(10, 2);
    lcd.print("Hum: ");
    lcd.print(humidity, 0);
    lcd.print("%");
  }

  // === Control Motor ===
  control_motor(conditionsOK);

  delay(1500); // Refresh every 1.5
seconds
}
```