NTNU
Kunnskap for en bedre verden

MANUAL

AgriBot

Author:
Mario Ruiz, Henrik Hauge, Kasper Gaup Madsen, Lars-Kristian Borchgrevink Mathisen, Jarle Nummedal, Martin Fuglum and Elias Aarekol

This is a complete manual for AgriBot. The manual includes a combination of manuals written by previous students who have worked on AgriBot.

8th July 2025

Contents

Abbreviations

AI Artificial Intelligence. iv

DLI Daily Light Integral. iv

EC Electrical Conductivity. iv, 4

Fe Iron. iv

HACS Home Assistant Community Store. iv, 3

HSI Hyperspectral Imaging. iv

HVAC Heating, Ventilation, and Air Conditioning. iv, 3, 4

IoT Internet of Things. iv

IP Internet Protocol. iv

K Potassium. iv

LAN Local Area Network. iv, 3

Mg Magnesium. iv

N Nitrogen. iv

P Phosphorus. iv

PAR photosynthetically active radiation. iv, 4

PID Proportional–Integral–Derivative. iv, 4

PPF Photosynthetic Photon Flux. iv

PPFD Photosynthetic Photon Flux Density. iv

RH Relative Humidity. iv

SWIR Short-Wave Infrared. iv

UI User Interface. iv

VF Vertical Farming. iv

VNIR Visible to Near-Infrared. iv

VPD Vapor Pressure Deficit. iv

Chapter 1

User manual for Agribot

NTNU
Kunnskap for en bedre verden

AgriBot user manual

8th July 2025

Contents

1.1 Introduction

This manual summarizes important information regarding the operation and maintenance of the Agribot hydroponics and Heating, Ventilation, and Air Conditioning (HVAC) system for vertical farming research. As the system is still under development this manual will be due to change.

1.2 Connecting to the system

Before interacting with the system, ensure the power and ethernet cables are connected. The system will turn on once the ground fault circuit breaker is switched on. Check that the switch is able to communicate over the ethernet by inspecting the switch LEDs. Make sure that the electrical and mechanical breakers are switched on.

The system uses a virtual Local Area Network (LAN) to facilitate communication between edge, server and user. Currently, zerotier is used to setup the virtual LAN. To be able to access the system dashboard remotely, one therefore needs to download zerotier and join the network using the network id. For windows and macos, connecting is easily done through the zerotier application, for linux the following command can be used:

```
zerotier-cli join ################
```

Where the hashtags represent the networks 16-digit hexadecimal id. Once connected to the network, the dashboard can be accessed via the following address:

```
http://172.26.45.193:8123/
```

Once connected, the dashboards are displayed in the side panel. Home Assistant Community Store (HACS) provides access to additional plugins and themes. The *Debug* dashboard lists all existing entities in Home Assistant, including basic information such as the status of switches and sensor readings. The main dashboard for the farm, *kybfarm*, serves as the central hub where all key views and controls are located. The *kybfarm* hub also provides an overview of the other views and explains their functionalities.

If this configuration, including themes and other elements, is not displayed, check the backup settings. Information about backup schedules is available in Home Assistant under:

```
Settings - System - Backups
```

1.3 Operating the system

1.3.1 The "kybfarm" Dashboard

For normal operation, the *kybfarm* dashboard serves as the main hub for controlling and monitoring the system. It includes dedicated views for all major farm operations:

- **Hub View** — Provides an overview of the available views and explains their functions.
- **Sensor Data** — Displays real-time readings from all connected sensors, including temperature, humidity, pH, Electrical Conductivity (EC), photosynthetically active radiation (PAR), and CO_2. Historical graphs help with trend analysis and system diagnostics.
- **Light Cycles** — Lets you control and monitor the photoperiod settings for the Heliospectra Elixia grow lamps. Adjust channel intensities, set start/end times, and configure dawn/dusk transitions to simulate natural light cycles. Live PAR data provides immediate feedback on light conditions.
- **Calibration** — Allows manual calibration of pH and EC sensors to ensure accurate water and nutrient measurements. Supports multi-point calibration modes.
- **HVAC** — Monitor and manage the Heating, Ventilation, and Air Conditioning system. Adjust reference values for temperature, humidity, and CO_2, and check real-time sensor readings with integrated Proportional–Integral–Derivative (PID) control.
- **Hyperspectral Imaging** — Control and visualize the hyperspectral imaging system for detailed crop monitoring and stress analysis.
- **Grow & Tank Control** — Manage the nutrient solution and water levels across grow tanks. Operate pumps and valves, and monitor EC and pH levels to ensure healthy plant growth.

1.3.2 The "Debug" Dashboard

The *Debug* dashboard lists every Home Assistant entity in the system. Use it for detailed inspection, troubleshooting, and configuration debugging of sensors, relays, controllers, and automation logic.

1.3.3 Standard Operation Procedure

During normal operation, the system should run autonomously. After setting reference values for EC, temperature, humidity, and lamp timing, the controllers can maintain desired conditions. Some settings may still require edits in AppDaemon configuration files, depending on the development stage.

To run the HVAC automatically, set the temperature and humidity references and enable the PID controllers for cooling and heating. Manual override is also possible

by setting voltage levels for the valve and fan between 0–10 V.

1.4 Before you grow

1.4.1 Filling and emptying

Filling the system is done by first filling the mixing tank and grow tanks, then the freshwater tank. Start by connecting a water hose with a female gardena connection to the male connector closest to the back-wall. Open the two entry valves and the mixing tank circulation valve. Turn on the water and wait until the mixing tank is full. Once full, close the right-most entry valve and activate the "5K water pump" in the dashboard and fill the grow tanks using the solenoids valves toggles in the dashboard. If the water level falls below the pump in the mixing tank, turn it off and start filling the mixing tank again. Repeat this until the growing tanks are full. Turn off the water supply, while keeping the entry valves open to release the pressure inside the hose. Then, connect the hose to the connector farthest from the wall and turn on the water. Wait until the freshwater tank is full, and turn off the water supply.

Emptying the system can be somewhat more time-consuming. Start by positioning the system close to a sink in the floor. Use the siphon pumps to empty the grow tanks one by one, then remove the growing tanks and empty the mixing tank and the freshwater tank using the same method. Once the tanks are close to empty it might be difficult to remove the last parts of the water. There should be a vacuum pump available in the workshop to ease the process.

1.4.2 Cleaning the system

Before every new growth experiment, the entire system should be thoroughly emptied, cleaned, and disinfected to remove residual nutrients, biofilm, and pathogens. The mixing tank is designed to hold up to 350 L and each grow tank up to 100 L. Follow these recommended steps:

1. **Ensure the system is empty:** Drain all water from the freshwater tank, mixing tank, and grow tanks using the siphon pumps. Remove the grow tanks if needed to access all surfaces.
2. **Prepare a cleaning solution for manual cleaning:** Mix clean water and 33% hydrogen peroxide to make a spray solution targeting a concentration of 2000 ppm. This equals approximately 580 mL hydrogen peroxide per 100 L of water.
3. **Clean tanks and accessible surfaces:** Use the spray solution to wet all internal surfaces of the mixing tank, grow tanks, and accessible piping. Wipe thoroughly using disposable paper towels or a soft cloth to remove deposits and biofilm. Wear PPE: gloves, mask, and eye protection.

4. **Fill and disinfect the mixing tank:** Refill the mixing tank with clean water and add 33% hydrogen peroxide to reach the target 2000 ppm concentration. Fill the mixing tank up to its working volume as needed.
5. **Recirculate the disinfecting solution:** In Home Assistant, open the 5K water pump valve and the grow tank solenoid valves to circulate the cleaning solution through the system for at least 15 minutes. This ensures the circulation lines and valves are flushed. Make sure to put something under the grow tank 2 pipe since this valve does not lead to the mixing tank when the grow tank is not in place.
6. **Drain and rinse:** After circulation, drain the mixing tank and any remaining solution in the system. Refill with clean water to rinse thoroughly, then drain completely.
7. **Reinstall and inspect:** Reinstall the grow tanks securely. Inspect all connections, pumps, and valves to ensure they are clean and free of leaks. The system is now ready for refilling and sensor calibration.

Important: Always wear proper PPE when handling hydrogen peroxide. Avoid running pumps dry during draining and filling.

1.4.3 Calibrating the sensors

The EC sensors naturally experience drift, therefore they need to be regularly calibrated. It is recommended to calibrate them at least before each grow period.

Each EC sensor needs to be calibrated by submerging the tip of each sensor in two different control liquids. The temperature of the liquid needs to be at 25 degrees celcius, if it is too cold it can be heated up using the heat gun in the workshop. Once the liquid reaches 25 degrees, the sensor is calibrated by writing to the correct register using the modbus protocol (See datasheet). Currently, this is can be done automatically in the home assistant debug dashboard for the mixing tank and one of the grow tank sensors. This is done by switching the calibration box from "Normal operation" to "Register calibration for EC #######" Where the hashtags denote the EC of your current control liquid.

1.5 Starting to grow

1.5.1 Germination

Before the plants can be placed in the AgriBot system, they need to be properly germinated. The plants will grow in foam-based growing mediums, typically rockwool, which are fitted into floating bases. Both the rockwool blocks and floating bases should be available in the lab.

The recommended germination procedure is as follows:

- Use rockwool blocks approximately 9 cm × 9 cm × 2.5 cm in size. Rip or cut pieces to fit each hole in the grow base.
- Insert the rockwool gently into each hole. Do not compress the material too tightly, as it needs to remain porous.

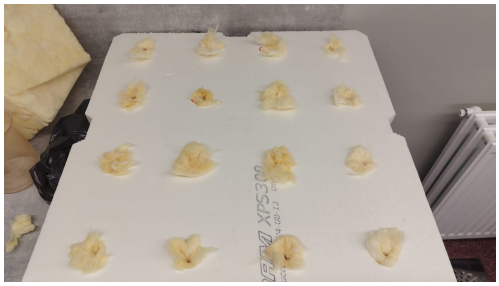


Figure 1.1: Foam growing mediom with rockwool

- Place around 5–6 seeds in each hole, ensuring the seeds are well enclosed within the rockwool to maintain contact with moisture.
- Pour clean water slowly to fully saturate the rockwool. The medium should be completely wet but not submerged. Seeds typically require about 90% relative humidity to germinate effectively.
- Place the seeded grow base in a dark, humid environment. This can be done by placing the entire base inside a large black bin bag or by storing it inside the grow chamber with the curtains closed and lights turned off.
- If using a bin bag, add a small amount of water inside the bag to help maintain high humidity around the rockwool.
- Check daily to ensure the rockwool remains moist. For plants such as lettuce, germination usually takes 3–4 days. Once seeds have sprouted and all holes show healthy seedlings, the growing base can be carefully transferred into the system.

Tip: Always handle the seedlings gently to avoid damaging young roots or dislodging seeds during transfer.

1.5.2 Growing

This has to be done manually, as the new automatic tank and nutrient control is currently under development. For this, you have two practical options:

- **Option 1: Manual dosing with peristaltic pumps.** Open the peristaltic pumps to dose nutrients into the mixing tank while monitoring the EC value in the dashboard. Once the target EC is reached, recirculate the solution to the grow tanks by opening the 5K valve and the corresponding solenoid in Home Assistant. Monitor the water level carefully to prevent overflow, and always close the solenoid before the grow tank reaches full capacity.

Then drain excess water from a grow tank, close the 5K valve and open the solenoid alone to discharge the liquid back to the mixing tank and repeat until the desired EC in the grow tank.

- **Option 2: Manual mixing with a graduated container.** Use a measuring jug or beaker to prepare the nutrient solution in advance. For example, for a grow tank of 100 L, if a 1 mL/L concentration is desired for each nutrient, add 100 mL of each nutrient to the water. Pour the solution carefully into the grow tank and ensure it mixes evenly.

Use the following table for nutrient reference:

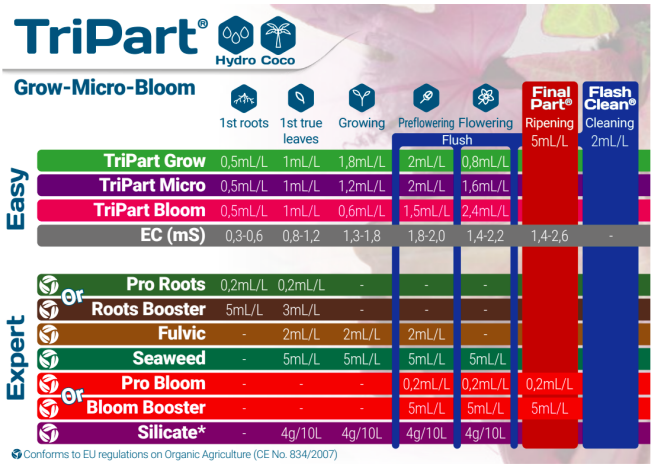


Figure 1.2: Nutrient reference table

Once the desired EC is stable, set the grow light amplitudes, on-off times, and reference values for temperature and humidity in the dashboard. Turn on the mixing tank and grow tank controllers. These will maintain circulation between the mixing tank and the grow tanks to help keep the EC levels balanced.

Important: Always keep the air pump running when there is water in the system to avoid stagnant water and unwanted microbe growth.

While the system is running under these settings, minimal further adjustment should be needed.

1.6 Guide for further development

1.6.1 Software overview

The system consists of two computers, the edge and the server. The edge is a Rasberry Pi located on the farm, and the server is a free standing computer located

in the D block. The edge directly reads and controls the sensors and actuators, on request by the server. The communication between the server and the edge is facilitated with MQTT. The server dashboard, data logging and other automations are handLED by Home assistent. All logged data is logged to an InfluxDB database. Finally, python scripts can be run using Appdeamon.

1.6.2 Developing on the system

When developing on the system, you will regularly need to access both the edge computer and the server. Both can be physically accessed, but the recommended approach is to connect via SSH. For convenient remote development, it is highly recommended to use the SSH plugin in Visual Studio Code (VSCode).

Before connecting, ensure that your machine is connected to the correct ZeroTier virtual network so that both the edge and server devices are reachable.

- **Server:** Connect via SSH using the following:
  `kybfarm@172.26.45.193`

  Use the password `tomato1000` when prompted.
- **Edge computer:** Connect via SSH using:
  `user1@172.26.171.33`

  Use the password `labanx` when prompted.

Always store passwords securely and do not share them outside the development team.

1.6.3 Adding sensors and actuators

When adding new sensors or actuators to the system, follow this protocol to ensure they are properly integrated with both the edge computer and the server.

It is recommended to use Modbus-compatible sensors to simplify integration. No standard pipeline currently exists for other sensor types.

For Modbus sensors:

- Ensure each sensor has a control box clearly labeled with a unique letter. Mark the same letter on the corresponding cables to avoid confusion.
- Add the sensor's Python interface script (e.g., `sensor_instance.py`) to the `edge/src/sensor_interfaces` directory.
- To avoid address conflicts, disconnect other sensors temporarily.
- Connect to the edge computer, navigate to the project folder:

```
cd kybfarm/edge
```

Import the new instance in your Python script and run:

```
python sensor_registration.py
```

Ensure you specify the correct port (`/dev/ttySC0` or `/dev/ttySC1`) in the code.

- The registration script will prompt you to reset, change, or leave the Modbus address. When changing a slave address, power cycle the sensor afterwards to apply the new address. Once the address is unique, reconnect any other sensors.

Integration steps:

1. Add the MQTT topic, sensor address, and port to the `.env` file in the `edge` directory. Update the related template if needed.
2. Import the sensor instance in `edge_computer_main.py` and follow the standard pattern for:
   - Adding the MQTT call
   - Activating the sensor
   - Implementing the `on_message` callback
   - Writing the request-response logic
3. On the server side, declare the sensor entity in `Server/homeassistant/config/include/mqtt_senso`
4. Add the corresponding automation in `Server/homeassistant/config/include/automations/senso`
5. Add the polling or data request routine in `Server/appdaemon/config/apps/sensor_request_loop.y`
6. Finally, restart the Docker containers to apply all changes:

```
docker compose -f kybfarm-docker-compose.yaml up --build -d
```

Tip: Always verify that new sensors do not conflict with existing addresses and test communication thoroughly before reconnecting the full sensor network.

The process for integrating actuators should follow a similar pipeline. However, instead of adding it to the mqtt_sensor_calls.yaml it should be added to the automations in an appropriate folder or file, following one of the other entities sintax.

1.7 Common issues

1.7.1 Can not connect to server

Sometimes the server might become unavailable through SSH or the dashboard. Normally, when connecting to the server for the first time in a couple of hours it might take a minute or two before it becomes available. If connection problems continue after waiting a while, it could be caused by the following:

Docker crashed/stalled The docker deamon has been known to crash on some occasions. Note that this will only cause the dashboard to be inaccessible, SSH should still be possible. Docker needs to be restarted, the easiest solution if possible is to just reboot the computer

1.7.2 Can not connect to edge

The edge can on occasion become unavailable through ssh or MQTT. Some common causes and fixes are listed bellow:

Ethernet cable unplugged Plug in the ethernet cable again :)

Zerotier disconnected The edge is known to disconnect from the zerotier network. Zerotiers website makes it possible to check for connected computers, if the raspi can not be identified, it needs to be re-connected. This can be done by following the instructions in section 2, on the edge. Remeber to update the zerotier id of the edge in the server config.

1.7.3 Lamp does not work

Often caused by the lamp getting its dynamic IP address changed. To fix this, download helioCONNECT on your laptop, connect with an ethernet cable from your laptop to the electrical box switch. Through helioCONNECT you should be able to identify the IP address of your lamp. Update the ip address in the edge computer config.

1.8 Safety in the lab

1.8.1 Electrical

Several parts of the system run on 230 volt AC, therefore you should take the necessary precautions before interacting with the electrical system. Short periods of system downtime is generally not an issue, therefore the ground fault protection should always be switched off when working with the electronics.

1.8.2 Moving actuators

There are several moving actuators in the setup, which are the solenoids and the peristaltic pumps. The solenoids closed systems and generally pose little threat of harm, the peristaltic pumps can, when the cover is off, cause bodily harm. It can be wise to turn off the electronic breakers controlling the peristaltic drivers, or the ground fault protection before physically interacting with the pumps. **Always remember to remove the peristaltic tubes from the nutrient bottles when taking of the pump cover.** If not the tubing will experience a siphoning effect which starts to empty the nutrient bottles.

1.8.3 Eye protection

To reduce eye stress from the grow lamps, it is recommended to either turn off the grow lamps while working on the physical system. If for some reason this is not possible use the orange safety glasses at the lab.

1.8.4 Hydrogen peroxide

The hydrogen peroxide used to clean the system is acidic and should be handled with precaution. Use PPE gear for both eyes, hands and mouth.

1.8.5 Moving the setup

When the system is filled with water it weighs over half a ton. Therefore it is unwise to attempt to move it during an ongoing experiment.

1.8.6 Water acidity

The nutrient mix is slightly acidic, causing the water in the mixing and grow tanks to also be acidic. Therefore, when interacting with the water in the system, the use of gloves is recommended. Prolonged exposure will cause a degradation of the skin.

1.8.7 Pumps

Water pumps can take damage from being run dry. There is not implemented failsafe for the pumps, meaning the user has to be vary not to run the pumps dry. This is relevant to the pumps in the mixing tank and freshwater tank. If you are cleaning the system, always pay attention to the pumps and turn them off when they start to run dry.

1.9 System constants

The following table summarizes constants and properties of the system such as flow rates and time constants

Property	Value	Unit	Comment
Mixing tank pump flow rate	5000	L/s	Nominal
Freshwater tank pump flow rate	350	L/s	Nominal
Peristaltic pump 1 flow rate	Value 3	L/s	Measured
Peristaltic pump 2 flow rate	Value 3	L/s	Measured
Peristaltic pump 3 flow rate	Value 3	L/s	Measured
Grow tank time constant	Value 3	s	Measured with only one solenoid open
Mixing tank size	350	L	
Freshwater tank size	50	L	
Grow tank size	100	L	
Nutrient to EC ratio	value	value	Measured

Table 1.1: Table of system constants

Chapter 2

IoT Platform for Vertical Farming: Software Manual

The Kybfarm Embed repository provides an open-source Internet of Things (IoT) platform framework for vertical farming. This document serves as a user manual for detailed guidance.

The repository is available at:
`https://github.com/mfuglum/kybfarm/`

Contents

2.1 Introduction

The design of the Kybfarm Embed framework enables the management and automation of vertical farming systems. It leverages containerization for modularity and scalability at the server-side, while specific interfaces enable control over actuators and data acquisition from sensors on the edge-side. The repository comprises the software for both server and edge, allowing one environment file for common parameters. Dedicated directories contain the server- and edge-specific code. Code listing 2.1 outlines the repository structure:

Code listing 2.1: Outline of repository structure for Kybfarm Embed.

```
kybfarm/
|-- edge/
|   |-- src/
|   |   |-- actuator_instances/
|   |   |   |-- grow_lamp_elixia_initialization.py
|   |   |   ‘-- relay_devices_initialization.py
|   |   |-- sensor_interfaces/
|   |   |   |-- sensor_BMP280_I2C.py
|   |   |   |-- sensor_SCD41_I2C.py
|   |   |   |-- sensor_SEC01_modbus.py
|   |   |   |-- sensor_SLIGHT01_modbus.py
|   |   |   |-- sensor_SPAR02_modbus.py
|   |   |   |-- sensor_SPH01_modbus.py
|   |   |   ‘-- sensor_SYM01_modbus.py
|   |   ‘-- utils/
|   |       |-- grow_lamp_elixia.py
|   |       ‘-- relay_device.py
|   |-- auto_launch_at_reboot.sh
|   |-- edge_computer_main.py
|   |-- reboot_config_example.sh ( update and rename to -> reboot_config.sh )
|   ‘-- requirements.txt
|-- server/
|   |-- appdaemon/config/apps/
|   |             ‘-- dummy_control.py
|   |-- homeassistant/config/
|   |                 ‘-- ( generated after running docker-compose )
|   |-- mosquitto/config/
|   |             ‘-- mosquitto.conf
|   ‘-- src/config_generator/
|           |-- config_generator.py
|           |-- dockerfile
|           |-- requirements.txt
|           |-- appdaemon_templates/
|           |   ‘-- appdaemon_apps_template.yaml
|           ‘-- homeassistant_templates/
|               |-- automations_template.yaml
|               |-- configuration_template.yaml
|               |-- influxdb_template.yaml
|               |-- input_boolean_template.yaml
|               |-- input_number_template.yaml
|               |-- input_select_template.yaml
|               ‘-- input_text_template.yaml
|-- README.md
|-- env_template.env ( update and rename to -> .env )
‘-- kybfarm-docker-compose.yaml
```

2.2 Getting Started

2.2.1 Prerequisites

Server:

- Linux terminal (e.g., The native terminal for Linux-based systems or Windows Subsystem for Linux) available on the installation machine.
- Docker and Docker Compose installed.
- Ensure the IP address is available from the edge by configuring the network or installing Tailscale or ZeroTier.

Edge:

- A Raspberry Pi or similar device for edge computing.
    - Tested edge computer configuration: Raspberry Pi 4B with Rasbian OS installed.
- Sensors and actuators compatible with the platform.
    - See section on RS485 adapter configuration for Modbus communication.
- raspi-gpio installed for General Purpose Input/Output (GPIO) configuration on the edge computer.
- Ensure the server IP address is available by configuring the network or installing Tailscale or ZeroTier.

2.2.2 Configuration

Place updated `.env`-file in repository root: Configuration is managed through an environment file (`.env`) and template files with placeholder values. The environment file should be placed in the repository's root. The environment file includes parameters such as the MQTT broker address, device identifiers, and other settings specific to your deployment. The file `env_template.env` provides an example `.env` file and should be replaced with the actual parameters.

2.2.3 Docker Desktop as a tool during installation

Docker Desktop provides useful functionality for setup and debugging during the installation process as well as after deployment. After creating the containers using `docker compose`, check out the following features of Docker Desktop:

- Open the *Containers* overview.
- Extend by clicking on the arrow beside `kybfarm`.
- Network *Ports* are displayed and hyperlinked, enabling fast access to the container's interface.
- For debugging, click on a containers name and:

- *Logs*: Useful for identifying configuration errors or simply verifying proper functionality.
- *Exec*: Provides a terminal to operate easily within the container (e.g. for creating files).
- *Files*: Provide overview of file system. It is possible to verify content and modify or delete files without modifying permission flags (which is required if you access a container's files from an external interface).

2.3 Installation

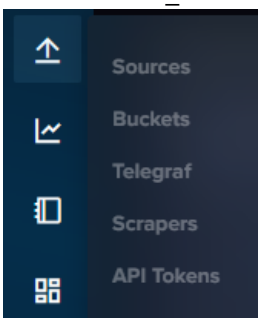
2.3.1 Server Setup

1. **Clone the repository:**

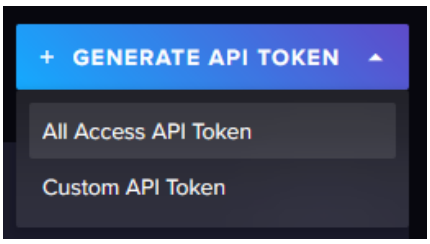
```
git clone https://github.com/mfuglum/kybfarm.git
cd kybfarm
```

2. **Build and start the containers using Docker Compose:**

```
docker compose -f kybfarm-docker-compose.yaml up --build -d
```

3. **Onboard to Home Assistant:**
   a. Access the Home Assistant dashboard at: `http://localhost:8123`.
   b. Create a new user and complete the onboarding process.
   c. The entered credentials are valid for the local instance created inside the recently created container.
4. **Configure communication between InfluxDB and Home Assistant:**
   a. Generate an all-access API token in InfluxDB (as outlined in the following steps): `https://docs.influxdata.com/influxdb/cloud/admin/tokens/create-token/#create-an-all-access-token`.
   b. Access InfluxDB at: `http://localhost:8086`.
   c. Log-in credentials are provided in the `.env` file as DOCKER_INFLUXDB_INIT_USERNAME and DOCKER_INFLUXDB_INIT_PASSWORD.
   d. 

   Locate the arrow symbol in the left pane, and click the *API Tokens* option.

e. 

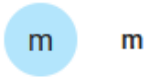
Click *GENERATE API TOKEN* and choose the *All Access API Token* option.

f. Add this token to the `.env` file as `INFLUXDB_TOKEN`.

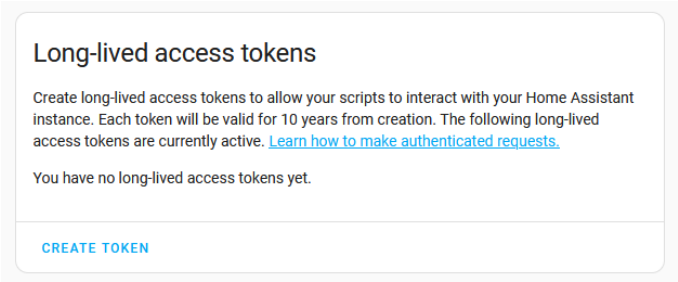
5. **Configure communication between AppDaemon and Home Assistant:**

a. Generate a long-lived access token in Home Assistant (as outlined in the following steps): `https://developers.home-assistant.io/docs/auth_api/#long-lived-access-token`.

b. Access the Home Assistant dashboard at: `http://localhost:8123`.

c. 

Click on the profile icon (*m* in the example figure) in the left pane of the dashboard.

d. 

Locate the token section in the *Security* tab, and create a *Long-lived access token*.

e. Add this token to the `.env` file as `TOKEN`.

6. **Update IP address fields of `.env` file**

a. `HOST_IP`: For the integration of InfluxDB and Home Assistant.

b. `INFLUXDB_URL`: For *optional* dashboard integration for InfluxDB in Home Assistant.

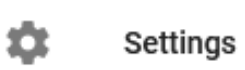
c. `HA_URL`: For the integration of AppDaemon and Home Assistant.

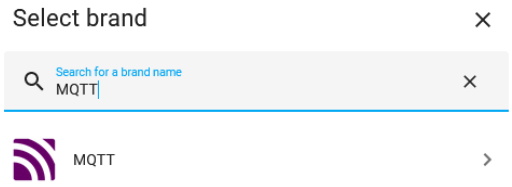
d. `APPDAEMON_URL`: For *optional* dashboard integration for AppDaemon and Home Assistant.

e. `MOSQUITTO_BROKER_IP`: The broker IP for the server for edge to connect (Tailscale IP for server).

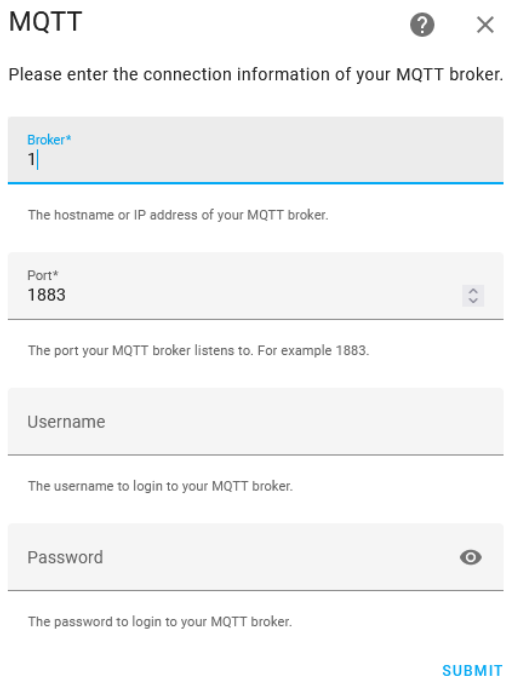
7. **Rebuild containers with new parameters and tokens:**

```
docker compose -f kybfarm-docker-compose.yaml up --build -d
```

8. **Integrate MQTT broker with Home Assistant:**
   a. Access the Home Assistant dashboard at: `http://localhost:8123`.
   b. 

      Click the *Settings* option on the left pane of the dashboard.
   c. Click *Devices & Services*.
   d. Click *Add Integration*.
   e. 

      Search for *MQTT*, and choose *MQTT* twice.

   f. 

      Provide `MOSQUITTO_BROKER_IP` as *Broker*, and `MOSQUITTO_BROKER_PORT` as *Port* if modified.
   g. Save the broker settings, and configured MQTT devices appear automatically.

2.3.2 Edge Computer Setup

1. **Clone the repository on the edge computer:**

```
git clone https://github.com/mfuglum/kybfarm.git
cd kybfarm/edge
```

2. **Set up a Python virtual environment:**

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

3. **Configure addresses**
   a. **For Modbus RTU sensors:** Provide the correct Modbus RTU address for each sensor instance in `edge_computer_main.py`
   b. **For HTTP interfaced grow lamp:** Provide the lamps IP address in the `.env` file.
4. **Configure crontab for auto-launch at boot on Rasbian system:**

   The script for auto-launch is in the file `auto_launch_at_reboot.sh`. It configures all GPIO ports to avoid unwanted relay states and utilizes `raspi-gpio` for this. To make sure the script doesn't run too early, it waits for proper ping with the MQTT broker.

   Provide `MOSQUITTO_BROKER_IP` in `reboot_config.sh`:

   The IP for the MQTT broker must be provided in the file or in a separate `reboot_config.sh` in the same folder. An example file is provided as `reboot_config_example.sh`. Rename the file and replace the value for the `mqtt_broker_ip`.

   To make the auto-launch script executable, enter the following command in a terminal in the /edge folder:

```
chmod +x auto_launch_at_reboot.sh
```

   Then, a cron-job must be scheduled for every reboot. Open a terminal and enter:

```
crontab -e
```

   If this is the first time using crontab, choose your preferred editor.

   Add the following line to the crontab-file (replace `/path` with the actual path to kybfarm, e.g., `/home/username/`):

```
@reboot /path/kybfarm/edge/auto_relaunch_at_reboot.sh
```

   When added, save and exit.

2.4 Running the Platform

2.4.1 Server Setup

After setting up the configuration, start the platform using Docker Compose:

```
docker compose -f kybfarm-docker-compose.yaml up --build -d
```

This will initialize all containers and start the services.

2.4.2 Edge Computer Setup

For automatic launch, reboot the system from GUI or terminal:

```
sudo reboot
```

To manually activate the virtual environment and start the script:

```
source /path/to/kybfarm/edge/venv/bin/activate
python /path/to/kybfarm/edge/edge_computer_main.py
```

2.5 Server Components

Docker-Compose orchestrates all of the listed components based on the content of the file `kybfarm-docker-compose.yaml`.

2.5.1 MQTT Broker

- **Purpose:** Facilitate communication between the server and edge devices.
- **Implementation:** Uses the Eclipse Mosquitto Docker Image.
- **Configuration:**
  - Port: 1883
  - Configured for "allow anonymous": This is enabled for simplicity while networking is end-to-end encrypted. If the IP address is available publicly, this should be changed.

2.5.2 Home Assistant

- **Purpose:** Manage devices, data logging, automation, and provide a user interface.
- **Implementation:** Deployed using the Home Assistant Docker Image. The "config_generator" provides all configuration and functionality when executed.
- **Configuration:**
  - Port: 8123
  - Integrates with InfluxDB for data storage
  - Integrates with AppDaemon for control and processing

2.5.3 InfluxDB

- **Purpose:** Store time-series data.
- **Implementation:** Uses the InfluxDB Docker Image.
- **Configuration:**
  - Port: 8086
  - User details and bucket names are specified in the environment file

2.5.4 AppDaemon

- **Purpose:** Host Python scripts for data processing, modeling, and control.
- **Implementation:** Uses the AppDaemon Docker Image.
- **Configuration:**
  - Port: 5050
  - Scripts can access sensor data from InfluxDB, communicate via MQTT, and interact directly with entities in Home Assistant.

2.5.5 Configuration Generator: `config_generator`

- **Purpose:** Implement Architecture as Code by generating configuration files with parameters from the environment file.
- **Implementation:** Executes in a container. It replaces placeholder values in the Home Assistant and AppDaemon configuration templates with actual parameters from the `.env` file using Jinja2.
- **Configuration:**
  - The YAML file for Docker Compose mounts relevant Home Assistant and AppDaemon directories so that configuration can be written from this container.
- **Usage:**
  - Add or extend configuration templates in the `config_generator/` directory.
  - Ensure the placeholders in the templates match the variable names in the `.env` file.
  - The `config_generator` container will automatically generate the final configuration files on startup if provided with the appropriate location.

2.6 Edge Components

2.6.1 Sensors

- **Purpose:** Gather data for the vertical farming system.
- **Implementation:** Most sensors use Modbus Remote Terminal Unit (RTU) protocol. Implemented in Python using minimalmodbus.

- **Table 2.1 lists the implemented sensor interfaces:**

Table 2.1: Implemented sensor interfaces. Datasheets are linked in the model names.

Ambient Conditions				
Sensor Type	**Model Name**	**Manufacturer**	**MPN**	**QTY**
Photosynthetically Active Radiation	S-PAR-02	Seeed Studio	314990735	1
Light Intensity	S-LIGHT-01	Seeed Studio	314990740	1
Temperature and Leaf Wetness	S-YM-01	Seeed Studio	314990738	1
Hydroponics				
Temperature and Barometric Pressure	BMP280	Adafruit	2651	6
pH and Temperature	S-PH-01	Seeed Studio	101990666	1
EC, Total Dissolved Solids and Temperature	S-EC-01	Seeed Studio	314990634	1
System Monitoring				
Duct Conditions				
CO_2, Humidity and Temperature	Thermokon LK+ CO2 100 temp_rH RS485 Modbus	Thermokon	670579	2
Humidity and Temperature	Seeed S-TH-01	Seeed Technology Co., Ltd	101990882	1

2.6.2 Actuators

Relay controlled devices

- **Purpose:** Relays enable control of actuators in the vertical farming asset.
- **Implementation:** Controlled via GPIO pins using the RPi.GPIO package, implemented as a relay device class.

HTTP controlled Grow Lamp

- **Purpose:** An ethernet-interfaced grow lamp provides light in the vertical farming asset.
- **Implementation:** Controlled via HTTP using the requests package, implemented as a lamp device class.
- Table 2.2 details the lamp.

Table 2.2: Actuators Interfaced

Actuator type	Model Name	Links
	HTTP protocol	
Grow lamp	Heliospectra Elixia LX 601C R4B	Datasheet User Manual

2.6.3 Communication Controller

The main script on the edge, `edge_computer_main.py`, serves as a communication controller.

- **Purpose:** Enable central control and data acquisition.
- **Implementation:** Written in Python, utilizing paho-mqtt for MQTT communication.
- **Key Functions:**
  - Load parameters from the environment file
  - Initiate MQTT client and connect to the broker
  - Subscribe to topics and handle data/command requests

2.7 MQTT Topic Convention

A well-defined MQTT topic structure ensures efficient communication:

- **Data Acquisition:**
  - Request: `dt/location/device-identifier/req`
  - Response: `dt/location/device-identifier/res`
- **Control and Configuration:**
  - Request: `cmd/location/device-identifier/req`
  - Response: `cmd/location/device-identifier/res`

Embed response topics in the payload to avoid hardcoding.

2.8 Development Guidelines

2.8.1 Recommended Development Flow

When adding new sensors or actuators to the system, follow this protocol to ensure they are properly integrated with both the edge computer and the server.

It is recommended to use Modbus-compatible sensors to simplify integration. No standard pipeline currently exists for other sensor types.

For Modbus sensors:

- Ensure each sensor has a control box clearly labeled with a unique letter. Mark the same letter on the corresponding cables to avoid confusion.
- Add the sensor's Python interface script (e.g., `sensor_instance.py`) to the `edge/src/sensor_interfaces` directory.
- To avoid address conflicts, disconnect other sensors temporarily.
- Connect to the edge computer, navigate to the project folder:

  ```
  cd kybfarm/edge
  ```

  Import the new instance in your Python script and run:

  ```
  python sensor_registration.py
  ```

  Ensure you specify the correct port (`/dev/ttySC0` or `/dev/ttySC1`) in the code.
- The registration script will prompt you to reset, change, or leave the Modbus address. When changing a slave address, power cycle the sensor afterwards to apply the new address. Once the address is unique, reconnect any other sensors.

Integration steps:

1. Add the MQTT topic, sensor address, and port to the `.env` file in the `edge` directory. Update the related template if needed.
2. Import the sensor instance in `edge_computer_main.py` and follow the standard pattern for:
   - Adding the MQTT call
   - Activating the sensor
   - Implementing the `on_message` callback
   - Writing the request-response logic
3. On the server side, declare the sensor entity in `Server/homeassistant/config/include/mqtt_senso`
4. Add the corresponding automation in `Server/homeassistant/config/include/automations/senso`
5. Add the polling or data request routine in `Server/appdaemon/config/apps/sensor_request_loop.y`
6. Finally, restart the Docker containers to apply all changes:

   ```
   docker compose -f kybfarm-docker-compose.yaml up --build -d
   ```

Tip: Always verify that new sensors do not conflict with existing addresses and test communication thoroughly before reconnecting the full sensor network.

The process for integrating actuators should follow a similar pipeline. However, instead of adding it to the mqtt_sensor_calls.yaml it should be added to the automations in an appropriate folder or file, following one of the other entities sintax.

2.8.2 Outline of Home Assitant .yamls

All of the configuration files described below are located in `kybfarm/server/homeassistant/config/` and its `include` subfolder. These YAML files define the core structure for automations, sensor integration, device control, and user interface helpers in Home Assistant.

- **Automations** This folder includes YAML files such as `desired_values.yaml`, `device_control.yaml`, `HSI.yaml`, `relays.yaml`, `sensor_calibrations.yaml`, `sensor_calls.yaml`, and `solid_state_relay.yaml`. These files define the logic for automations that control sensor polling, device actuation, hyperspectral imaging routines, relays, and sensor calibration workflows.
- **GUI_helpers** This folder contains helper definitions for the Home Assistant dashboards: `input_boolean.yaml`, `input_button.yaml`, `input_datetime.yaml`, `input_number.yaml`, `input_select.yaml`, and `input_text.yaml`. These files provide user-adjustable controls such as toggle switches, numeric inputs, text fields, and time settings.
- **mqtt_sensor_topics.yaml** Defines the MQTT topics for each sensor integrated into the system, mapping data channels for Home Assistant to subscribe to real-time readings from the edge computer.
- **influxdb.yaml** Contains configuration settings for connecting Home Assistant to the InfluxDB time-series database for logging sensor data and system states.
- **configuration.yaml** The main entry point that ties together all included YAML files, merging them into the final Home Assistant setup. This file loads all automations, helpers, MQTT sensors, and database connections.

These templates make it easier to modify or extend the system by editing only the relevant sections without changing the core structure.

2.9 Development and Debugging Tips

2.9.1 On Edge / Raspberry Pi

Stop Current Cronjob

1. Check the status of cron jobs:

   ```
   systemctl status cron
   ```

2. Identify the process under "cron.service" (e.g., `edge_computer_main.py`) and its Process ID (PID).
3. Kill the process by entering the following command replacing `PID` with the actual number:

   ```
   kill PID
   ```

   This ensures the cron job does not conflict with manually started scripts.

Error during initialization

Occasionally, during reboot or while starting the program, some sensors might not initialize correctly. The might look like this:

```
STH01_1, data fetch error: name 'sensor_STH01_1' is not defined
```

This can happen when multiple sensors are occupying the bus and causes collions. To remedy this, follow these steps.

- Stop the program: CTRL + C, or kill the cronjob, see section 2.9.1.
- Start the program again
- Keep doing the above steps until they initialize correctly

2.9.2 On Server

Debugging with MQTT

1. Navigate to: `Settings > Devices & Services > MQTT > Configure`
2. Use the built-in tool to debug communication:
    - Detect if a packet is not published from the edge or server.
    - Publish packets to identify if the issue is on the edge computer or in the Home Assistant configuration.

Typical errors include incorrect formatting or wrong topics.

Debugging with Docker

1. Open Docker Desktop.
2. Select the container you are having trouble with:
    - Access files and logs in the container.
    - Inspect files that might be incorrect or simply read errors in the log.

Debugging with Home Assistant Graphical User Interface (GUI)

Common errors occur when configuring relays and calibration GUI tools (Helpers). These errors often relate to entity IDs. The following bullet points can be useful in such situations:

- Follow the naming convention of lowercase and underscore-separated names.
- If automations linked to helpers don't trigger as expected, an ID error is likely.
- To identify this, simply click on the helper and check its entity ID in the GUI, and see if it deviates from the one provided in the environment file.

2.10 Miscellaneous

2.10.1 Raspberry Pi 4B with RS485 Adapter

This section details the configuration of Raspberry Pi 4B as an edge computer with the RS485 adapter:

- Waveshare 2-CH RS485 HAT

The product wiki provides all the necessary information at:
`https://www.waveshare.com/wiki/2-CH_RS485_HAT`

In summary, the following steps must be conducted to deploy the adapter with Raspberry Pi 4B for Modbus RTU communication:

- DIP switch configuration: Set the utilized channels to Full-auto mode.
- Load driver:
  - Open the file `/boot/config.txt`.
  - Add the following line and save:

    ```
    dtoverlay=sc16is752-spi1, int_pin=24
    ```

  - Reboot and the RS485 channels are available at the ports:
    - `ttySC0`
    - `ttySC1`

2.11 Glossary

GPIO : General Purpose Input/Output

GUI : Graphical User Interface

IoT : Internet of Things

MPN : Manufacturer Part Number

PID : Process ID

RTU : Remote Terminal Unit

Chapter 3

IoT Platform for Vertical Farming: Hardware Manual

As part of the development of an Internet of Things (IoT) platform framework for vertical farming (VF), an electronic control unit (ECU) was designed and implemented to serve as a subsystem in a VF prototype unit. This document serves as a user manual for detailed guidance.

Figure 3.1: The VF prototype unit, with the electronic control unit integrated into the box on the short end.

Contents

3.1 Introduction

This manual provides implementation details relevant to the usage and development of the electronic control unit supporting the IoT platform for deployment in the VF prototype unit. Figure 3.1 depicts the VF prototype unit with the electronic control unit deployed in the enclosure on the short end.

3.2 Electrical Documentation Outline

The electrical documentation provides all relevant information about the assembly and associated components. Figure 3.2 depicts the assembled electronic control unit. The following points offer an outline of the documentation:

- **Component Diagram:** Figure 3.3 provides a visual representation of all components.
- **Electric Wire Diagram:** chapter 6 provides detailed wiring diagram.
- **Connection Schedules:**
  - chapter 8 provides interfacing details for the terminal blocks.
  - chapter 9 provides interfacing details for the Raspberry Pi module.
- **Component List:** chapter 7 provides a comprehensive list of components used in the electronic control unit.

Figure 3.2: Assembled ECU deployed in the VF prototype unit and powered.

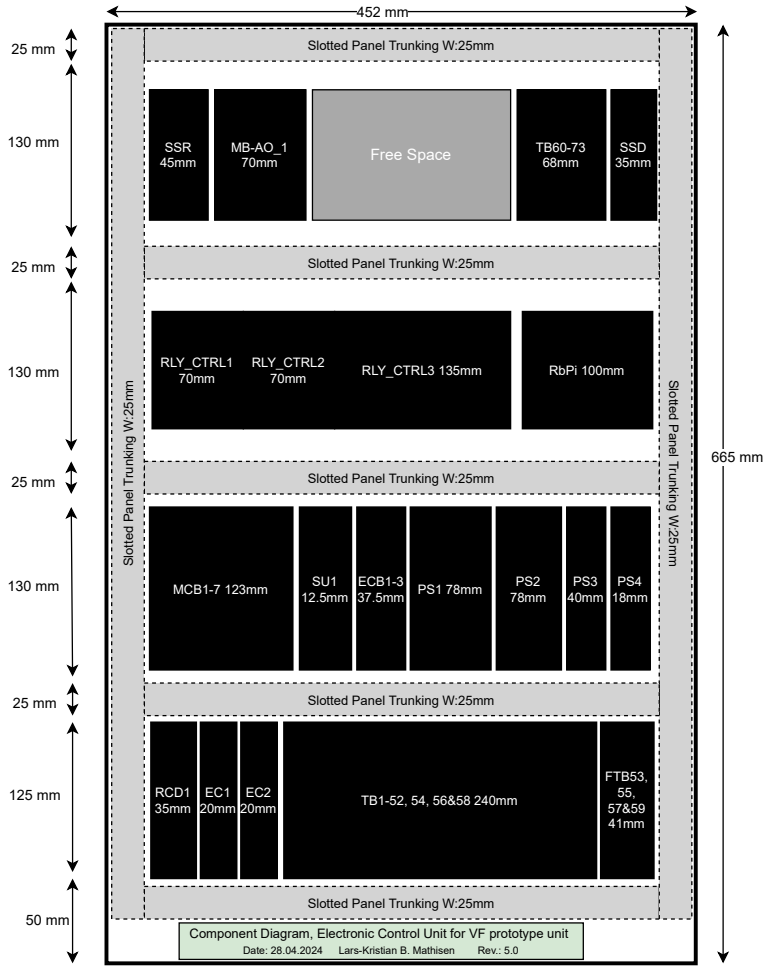


Figure 3.3: The component diagram for the ECU for the VF prototype unit.

The components on each row, counting from the one furthest down, can be summarized as follows (See chapter 7 for legend):
First row: From the left, the ground fault protection interfaces the external power supply. Next are the energy counters, one measuring the total power consumption and the other measuring the grow lamp circuit. Beside them, there is space for two additional energy counters for more discrete measurements. Next are all the terminal blocks, providing the interface between all components in the control box and the sensors and actuators in the asset.
Second row: From the left, miniature circuit breakers protect most actuator circuits from overload. Next to them is a supply unit, followed by three electronic circuit breakers protecting the driver circuits for peristaltic pumps. Next are power supply units (PS). Two 24VDC PSs supply the peristaltic pumps, and one PS supplies the sensor networks, water valve and CO_2 solenoid. A 5VDC PS supplies the edge computer and the relay control boards.
Third row: From the left, relay control boards interfaced with the edge computer to the right. The edge computer is specifically a Raspberry Pi 4B stacked with adapters.
Fourth row: From the left, the Solid State Relay for the heating battery and Modbus Analog Output module for different 0-10V control signals. Moreover, some free space for further development of the ECU. Lastly, to the right, the newest Terminal Blocks to interface with outside the ECU, and the SSD for the Raspberry Pi.

3.3 Safety

Always cut the power completely when working with exposed conductors during maintenance or mounting of devices.

All circuit breakers, except the ground fault protection, are single-poled. Beware that a device connected to a single pole circuit breaker is still dangerous to touch when the circuit breaker is in the open position as the voltage potential remains.

3.4 Usage: Power and Connectivity

This section describes how to power and connect the electronic control unit.

Circuit Breakers

Before powering the unit, be aware of the circuit breakers' state. The safest power-up procedure is to start with all open and then close them individually. This approach should be considered if new development is deployed.

Power Supply: The electronic control unit provides power to all subsystems within the VF prototype unit. Use the dedicated 230VAC connector as depicted in Figure 3.4 for powering up the units.

Ethernet: Use the dedicated RJ-45 connector for ethernet connection depicted in Figure 3.4. Provide a separate cable to the Edge Computer within the enclosure.

Figure 3.4: External 230VAC power is connected with the dedicated connector in the center of the picture. The RJ-45 connector is located directly above it. The connectors underneath the electronic control unit enclosure allow for the complete detachment of external wires when moving the VF prototype unit.

3.5 Sensors

This section describes procured and obtained sensors, the constitution of the sensor network and the connector pinout, sensor calibration, and fuse replacement for the tank level sensors.

3.5.1 Procured and obtained Sensors

Table 3.1 provides a list of all planned sensors, specifying the type, model name (with a linked datasheet), manufacturer part number (MPN), and quantity (QTY) obtained.

Ambient Conditions				
Sensor Type	**Model Name**	**Manufacturer**	**MPN**	**QTY**
Photosynthet -ically Active Radiation	S-PAR-02	Seeed Studio	314990735	
Light Intensity	S-LIGHT-01	Seeed Studio	314990740	1
Temperature, and Leaf Wetness	S-YM-01	Seeed Studio	314990738	1
Hydroponics				
Sensor Type	**Model Name**	**Manufacturer**	**MPN.**	
pH, and Temperature	S-PH-01	Seeed Studio	101990666	1[1]
Electric Conductivity, Total Dissolved Solids, and Temperature	S-EC-01	Seeed Studio	314990634	1
Dissolved Oxygen	S-RJY-01	Seeed Studio	314990633	0[2]
Tank Level	RSF70 Float Switch	RS PRO	174-8419	4
System Monitoring				
Sensor Type	**Model Name**	**Manufacturer**	**MPN.**	
Energy Counter	Energy Counter	Carlo Gavazzi	EM111DIN -AV81XS1X	2 [5]
Duct Conditions				
Sensor Type	**Model Name**	**Manufacturer**	**MPN.**	
CO_2, Humidity, and Temperature	LK+ CO2 100 temp_rH RS485 Modbus	Thermokon	670579	2[3]
Humidity and Temperature	S-TH-01	Seeed Technology Co., Ltd	101990882	1[3]
Air Flow	EE671	E+E Elektronik Ges	T15J3 HV25P1	1[4]

Table 3.1: Obtained Modbus RTU sensors.

[1] A second S-PH-01 sensor is in place, but not yet implemented.

[2] The dissolved oxygen sensor has not yet been implemented in the VF system.

[3] Due to delivery issues for the LK sensor there has only been implemented one. To replace the missing sensor, an additionally S-TH-01 sensor is used.

[4] The sensor is not prioritized in the HVAC-system for this work period, but is procured to be implemented for further development.

[5] The energy sensors are currently not connected with Modbus, see section 3.5.2 for more details.

3.5.2 Sensor Network Details

Two RS485 networks interface all Modbus Remote Terminal Unit (RTU) compatible sensors in the VF prototype unit.

Physical Components

The currently deployed networks consists of the following physical components:

- **Ethernet Cable** **MPN:** XS6W-6LSZH8SS1000CM-Y Datasheet
- **SP13/P5 Circular connectors** **MPN:** 144-0630 Datasheet
- **RS-485 splitters** **MPN:** 206001060 Datasheet

Pinout Sensor Networks

All the Modbus RTU sensors share the same pinout, described in Table 3.2. Consult this pinout when mounting new connectors and cables to extend the networks or assembling a new sensor with cable and connector.

Table 3.2: Modbus RTU sensor and connector pinout

PIN	Wire Color	Signal
1	RED	24VDC
2	BLUE	-
3	YELLOW	RS-485 A
4	WHITE	RS-485 B
5	BLACK	GND

The energy counter sensors are currently not connected at all. To support future implementations, it is necessary to ensure that the sensors integrate into one of the

excising sensor networks. To achieve this, follow the pinout provided in Table 3.3. For more details, refer to the datasheet linked in Table 3.1.

Table 3.3: Energy Counter sensor pinout

PIN	Function
5	Termination (on outermost sensor)
6	RS-485 B
7	GND
8	RS-485 A

3.5.3 Sensor Calibration

Refer to the individual sensor datasheets linked in Table 3.1 for calibration procedures. Regular calibration ensures accurate readings and optimal performance.

For the S-PH-01 and S-EC-01 sensors, Table 3.4 lists calibration reference solutions procured for the S-PH-01 and S-EC-01 sensors.

A suggested calibration routine is to conduct calibration at least before every new growth cycle, but it should be reevaluated based on accuracy test results utilizing the reference solutions.

Table 3.4: Calibration Reference Solutions

Calibration Type	MPN	Datasheet Link
pH Calibration 4.01	HI70004P	Datasheet
pH Calibration 7.00	HI70007P	Datasheet
pH Calibration 10.01	HI-70010P	Datasheet
EC Calibration 1413	HI70031P	Datasheet
EC Calibration 12880	HI7030L	Datasheet

3.5.4 Float Switch Sensors Maintenance

For safety reasons, the RSF70 Float Switch sensors are interfaced with the Raspberry Pi's 3V3 via fused terminal blocks.

The fused terminal blocks use cartridge fuses of type *G / 5 x 20*. Figure 3.5 demonstrates the fuse replacement procedure and can be summarized as follows:

1. Release the fused terminal block from the DIN rail.
2. Open the fuse housing.
3. Remove the existing fuse.
4. Place a new fuse in the plastic bracket in the housing lid and close it.
5. Mount the fused terminal block back onto the DIN rail.

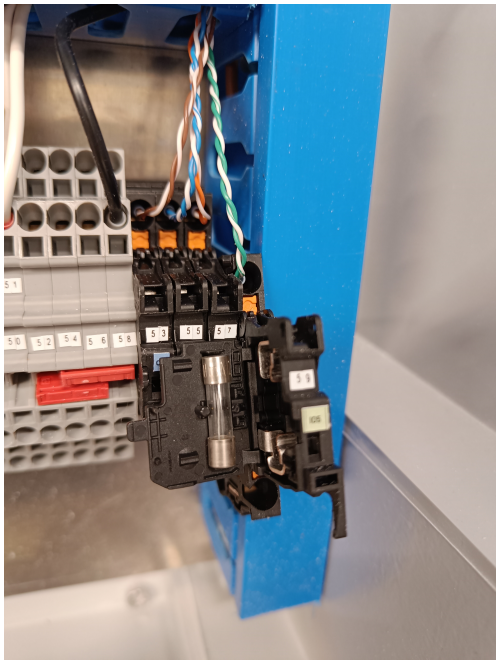


Figure 3.5: Fuse replacement for fused terminal blocks.

3.6 Edge Computer Hardware

The edge computer ensures interfaces to all devices in the VF prototype unit, and communication with the server as illustrated in Figure 3.6. The edge computer is referred to as the Raspberry pi module in the electrical documentation.

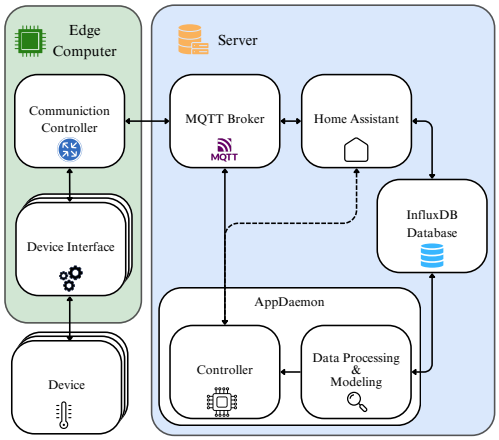


Figure 3.6: The entire pipeline for the IoT platform.

This section details the edge computer components and how to mount and detach them from the electronic control unit.

3.6.1 Components Overview

The following components constitute the Raspberry Pi module in the electrical documentation:

- **Raspberry Pi 4 Model B** (8GB memory model), datasheet.
- **RS485 Adapter**: Waveshare 2-CH RS485 HAT, datasheet.
- **GPIO Screw Terminal Hat**: Seeed Studio GPIO Screw Terminal Hat for Raspberry Pi, datasheet.
- **SSD**: Cepter 512 GB for Booting, Operating System and Storage the Raspberry Pi, Link

The according datasheets are linked.

3.6.2 Connecting and Disconnecting the Edge Computer

This section provides detailed instructions on how to connect and disconnect the Edge computer, specifically the Raspberry Pi 4 Model B stack, which includes an RS485 adapter and a General Purpose Input/Output (GPIO) Screw Terminal Hat. The procedure ensures that all wires to the relay control boards and power supply can remain connected while the Raspberry Pi and RS485 adapter can be detached for development and prototyping.

Connecting the Edge Computer

Follow these steps to connect the Raspberry Pi stack to the control box:

Step 1: Prepare the Components

1. Ensure that the power supply to the control box is disconnected to avoid any electrical hazards.
2. Gather all necessary components: Raspberry Pi, RS485 adapter, and GPIO Screw Terminal Hat.

Step 2: Attach the RS485 Adapter

1. Place the Waveshare 2-CH RS485 HAT on top of the Raspberry Pi 4 Model B, aligning the GPIO pins with the headers on the adapter.
2. Press down gently to ensure a secure connection between the Raspberry Pi and the RS485 adapter.

Step 3: Connect channels for the Modbus RTU network

1. Connect the RS485 A and RS485 B signal wires from the channel 1 and 2 cables to the corresponding terminals on the RS485 adapter.

2. For both channel 1 and channel 2, verify the signal wire colors with the ones connected to the terminal block as scheduled in chapter 8.

Step 4: Attach the GPIO Screw Terminal Hat

1. Place the Seeed Studio GPIO Screw Terminal Hat on top of the RS485 adapter, again aligning the GPIO pins.
2. Press down gently to secure the connection. However, a slight mismatch in the header pin configuration between the adapters persists, and the stack becomes skewed when mounted until a proper configuration is resolved.

Step 5: Connect the Wires

1. If not connected, connect the wires from the electronic control unit to the GPIO Screw Terminal Hat. Make sure each wire is securely fastened to the appropriate terminal.
2. Refer to the wiring diagram in chapter 6 and the connection schedule in chapter 9 for correct wire placement.

Step 6: Power Up the System

1. Once all connections are secured, reconnect the power supply to the control box.
2. Verify that the Raspberry Pi and connected components are functioning correctly.

Disconnecting the Edge Computer

Follow these steps to disconnect the Raspberry Pi stack for development and prototyping safely:

Step 1: Power Down the System

1. Shut down the Raspberry Pi properly using the command:
   `sudo shutdown now`
2. Disconnect the power supply to the electronic control unit.

Step 2: Remove the GPIO Screw Terminal Hat

1. Gently lift the GPIO Screw Terminal Hat off the RS485 adapter.
2. If desired: Carefully disconnect all wires from the GPIO Screw Terminal Hat, noting their positions for reassembly.

By following these detailed steps, you can efficiently manage the connection and disconnection of the Edge computer while ensuring the integrity and functionality of the electronic control unit.

3.7 Grow Lamp Configuration

This section provides guidance configuring the grow lamp Heliospectra Elixia LX 601C R4B deployed in the VF prototype unit. Refer to the user manual link in the model name for detailed instructions. The lamp and the Raspberry Pi are connected to a network switch with ethernet, which must be connected to the internet for communication between the edge and the server.

For communication to a new lamp (or if the lamp IP address is unknown), utilize the Heliospectra software helioCONNECT. To discover the lamp, **admin rights to the machine is needed** to allow the correct firewall settings and an ethernet connection to the network switch where the lamp is connected.

Figure 3.7 shows the helioCONNECT user interface after a lamp is discovered. Follow these steps to obtain a connected lamp IP address accordingly:

1. Download the helioCONNECT on the employed machine.
2. Allow all firewall and network options requested when starting the application.
3. Click *Scan* and wait for the lamp to appear.
4. Select the detected lamp.
5. *Open Network ethernet configuration* to obtain the IP address.
6. Update the respective value in the `.env` file on the Raspberry Pi with the obtained IP.

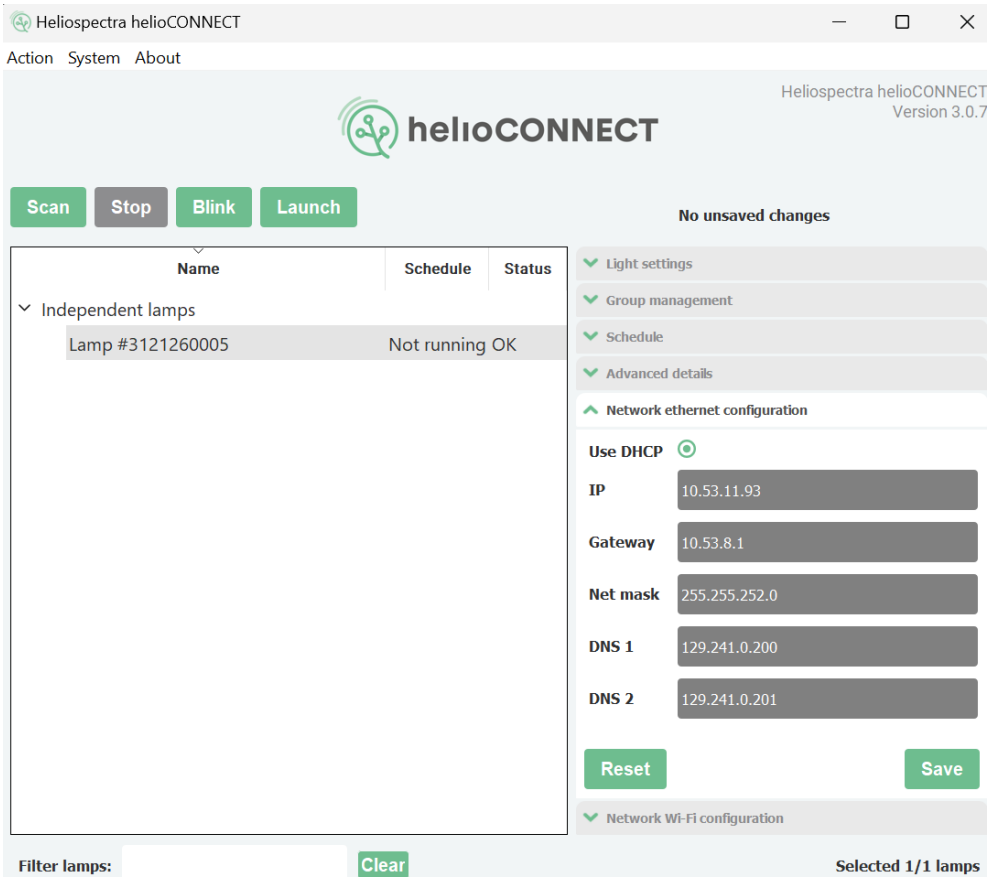


Figure 3.7: The helioCONNECT application for identifying connected Heliospectra lamps.

3.8 Further Development

As it occurs in Figure 3.2 and Figure 3.3, the upper row in the enclosure of the electronic control unit is not used and available for further development.

The whole ECU system is mounted on an aluminum plate, which enhances maintenance. The aluminum plate and all ECU components can be detached if reordering and rewiring components are desired.

3.9 Glossary

GPIO General Purpose Input/Output

IoT : Internet of Things

VF : Vertical Farming

ECU : Electronic Control Unit

MPN : Manufacturer Part Number

QTY : Quantity

RTU : Remote Terminal Unit

SSD : Solid-State Drive

Chapter 4

Rasperry Pi 4 system recovery and booting configuration

Contents

4.1 Introduction

This section will provide the documentation for the booting and operating system of the Raspberry Pi 4 used in KybFarm Embedded. This can be used in case of system failure, hardware replacement or further development.

4.2 Backup

The OS in the Raspberry Pi is located on the KybFarmEmbedded server as an IMG-file. It can be accessed both physically and remotely. The backup is located under the file path:

```
/home/kybfarm/kybfarm/RPI_OS_backup
```

Additionally, Akhil S. Anand has a microSD card containing the operating system for the Raspberrry Pi.

4.3 Retrieve the Backup

4.3.1 Remote Access

1. Navigate to the backup directory to verify the file path. It should be something similar to this:
   `/home/kybfarm/kybfarm/RPI_OS_backup`

2. Use proper software to transfer/copy the file to your local machine. An option is `scp` through terminal with the following prompt, and remember to open the terminal as an administrator:
   ```
   scp kybfarm@172.26.142.218:/home/kybfarm/kybfarm/RPI_OS_backup/RPIOSbackup.img
   /local/file/path
   ```

3. After the file is successfully transferred to your local computer, move on to section 4.4.

4.3.2 Physical Access

1. Connect a USB drive or SSD to the machine that stores the backup.
2. Log in to the system.
3. Navigate to the directory:
   `/home/kybfarm/kybfarm/RPI_OS_backup`

4. Copy the backup called:
   `RPIOSbackup.img`

5. After successfully copying the backup, disconnect the USB/SSD. Then connect it to a system where it can be flashed and move on to section 4.4.

4.4 Flash IMG-file

To flash the IMG-file, use appropriate disk imaging software. A recommended option is Win32DiskImager, which can be downloaded for free. Make sure to select the correct source image and target device. Pay close attention to whether you are reading or writing the disk/device, since incorrect settings may result in loss of data.

After successfully flashing the IMG-file to the new booting and storage device, it can be inserted into the Raspberry Pi inside the Electrical Control Unit. Finally, the system can be rebooted to verify that the Raspberry Pi boots correctly.

4.5 Raspberry Pi 4 Boot Configuration

The Raspberry Pi currently installed in the Electrical Control Unit has been configured properly to support a SSD as a booting device. It is important that the Raspberry Pi supports USB booting, which Raspberry 4B does. The attached guide has been used for the configuration.

> URL guide: `https://www.raspberrystreet.com/learn/how-to-boot-raspberrypi-from-usb-ssd` [1]

The above guide will provide all necessary steps of how to boot from a SSD device, or just simple booting configuration. In the case of full replacement of the Raspberry Pi in the Electrical Control Unit, it is essential to configure the EEPROM bootloader by following the steps in the attached guide(or something similar).

Chapter 5

Hyperspectral Imaging Scanner

This appendix describes the design, installation, and usage of the integrated Hyperspectral Imaging (HSI) scanner added to the Kybfarm platform.

5.1 Overview

The HSI scanner enables non-invasive, high-resolution spectral analysis of plant health. It provides pixel-level data across the visible to near-infrared (VNIR) spectrum, supporting research on stress detection, nutrient deficiencies, and early disease identification. The unit is mounted inside the grow chamber and connected to the edge computer for automated image capture.

5.2 Mechanical Integration

The camera is mounted on a custom-designed mechanism using a Prusa mk3 i3 3D printer. It is installed in the growing tank chamber above the plants. Mounting hardware includes adjustable supports to ensure proper focus distance and alignment with the plants.

5.3 Connectivity and Control

The HSI unit and its associated printer are both controlled by a dedicated Raspberry Pi 4B, which handles camera communication, image capture, and any local printing tasks. The Pi runs the acquisition routine that can trigger scans either on demand or at scheduled intervals. Captured spectral data is saved in an organized folder structure on the edge device and can be sent to the server for storage and further processing.

To access the Raspberry Pi remotely for configuration or troubleshooting, use SSH with X11 forwarding enabled. This allows you to run graphical user interface

(GUI) applications from the Pi on your local machine.

Connect using:

```
ssh -Y kfspectra@172.26.5.180
```

Use the password: `lettuce450`.

The `-Y` flag enables trusted X11 forwarding, allowing any camera configuration tools or image viewers that require a GUI to display properly on your local computer. Make sure your local machine has an X server installed (e.g., XQuartz for macOS or Xming for Windows) if connecting from a non-Linux system.

Acquisition Routine

A dedicated Python module handles:

- Initial camera connection and configuration
- Triggering scans at defined times
- Storing raw spectral images and metadata

The main script that manages the MQTT connection to the server is located at `KFSpectra/edge/main.py`. This script is designed to start automatically at system reboot, but if it is not running for any reason, it can be manually launched by navigating to KFSpectra/edge and running:

```
uv run main.py
```

5.4 Usage Instructions

Before starting a scan:

1. Ensure the chamber is free of obstacles and plants are positioned correctly.
2. Check that the scanner mount is secure and that the lens is clean.
3. Use the Home Assistant dashboard or command line to trigger the scan or set up scheduled scans.
4. Monitor the storage path to ensure there is sufficient disk space.

5.5 Data Management

Captured hyperspectral data files are stored in an organized directory, typically:

```
/kybfarm/edge/data/hyperspectral/
```

But data should be also uploaded automatically to the server in kybfarm/server-/homeassitant/config/HSI. File names include timestamps and scan parameters

for easy traceability. Follow best practices for backup and storage to avoid accidental data loss.

5.6 Future Improvements

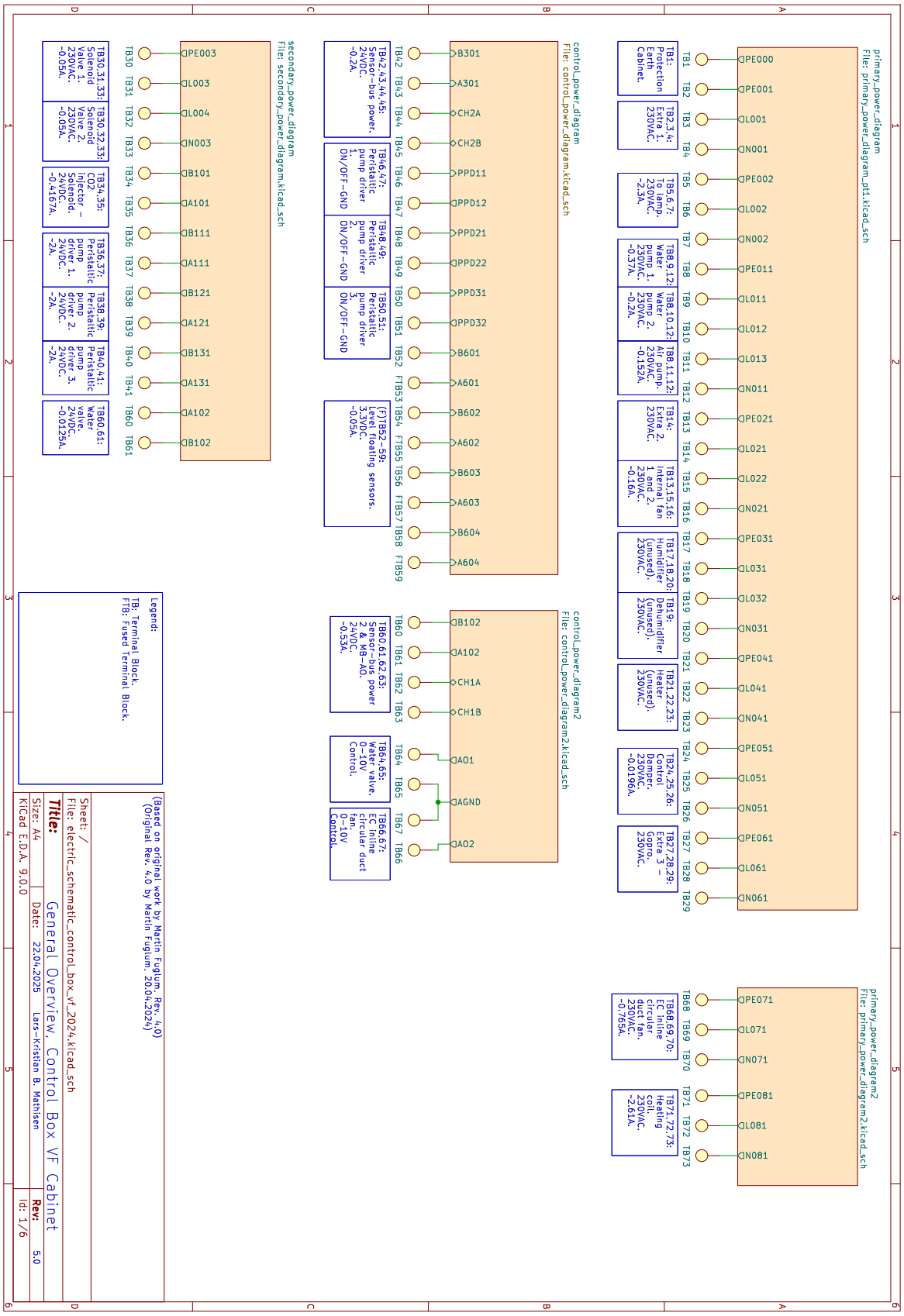
The current version serves as a prototype. Planned improvements include:

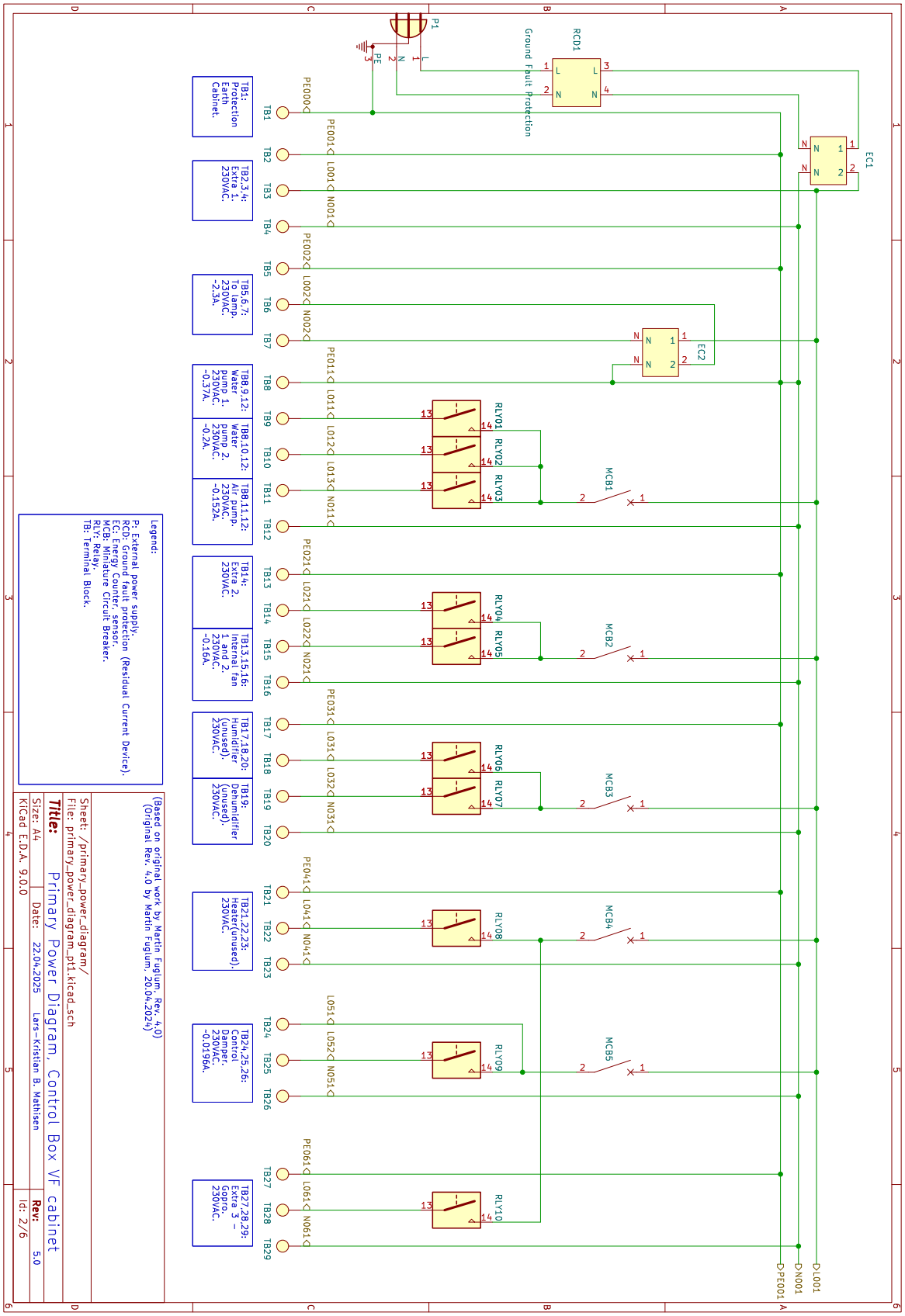
- Integration with sensor-based feedback for closed-loop control
- Development of preprocessing pipelines for spectral feature extraction

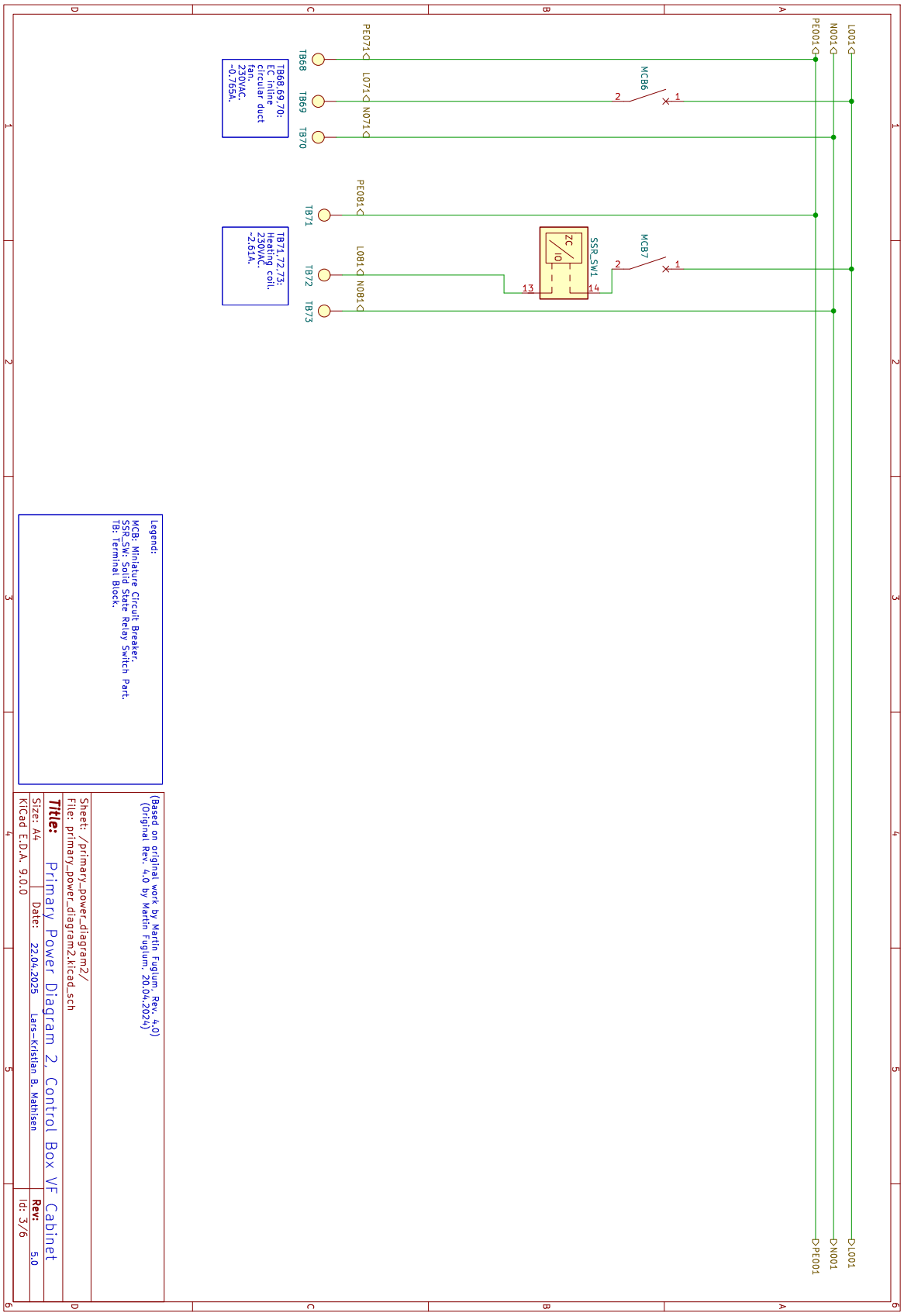
Chapter 6

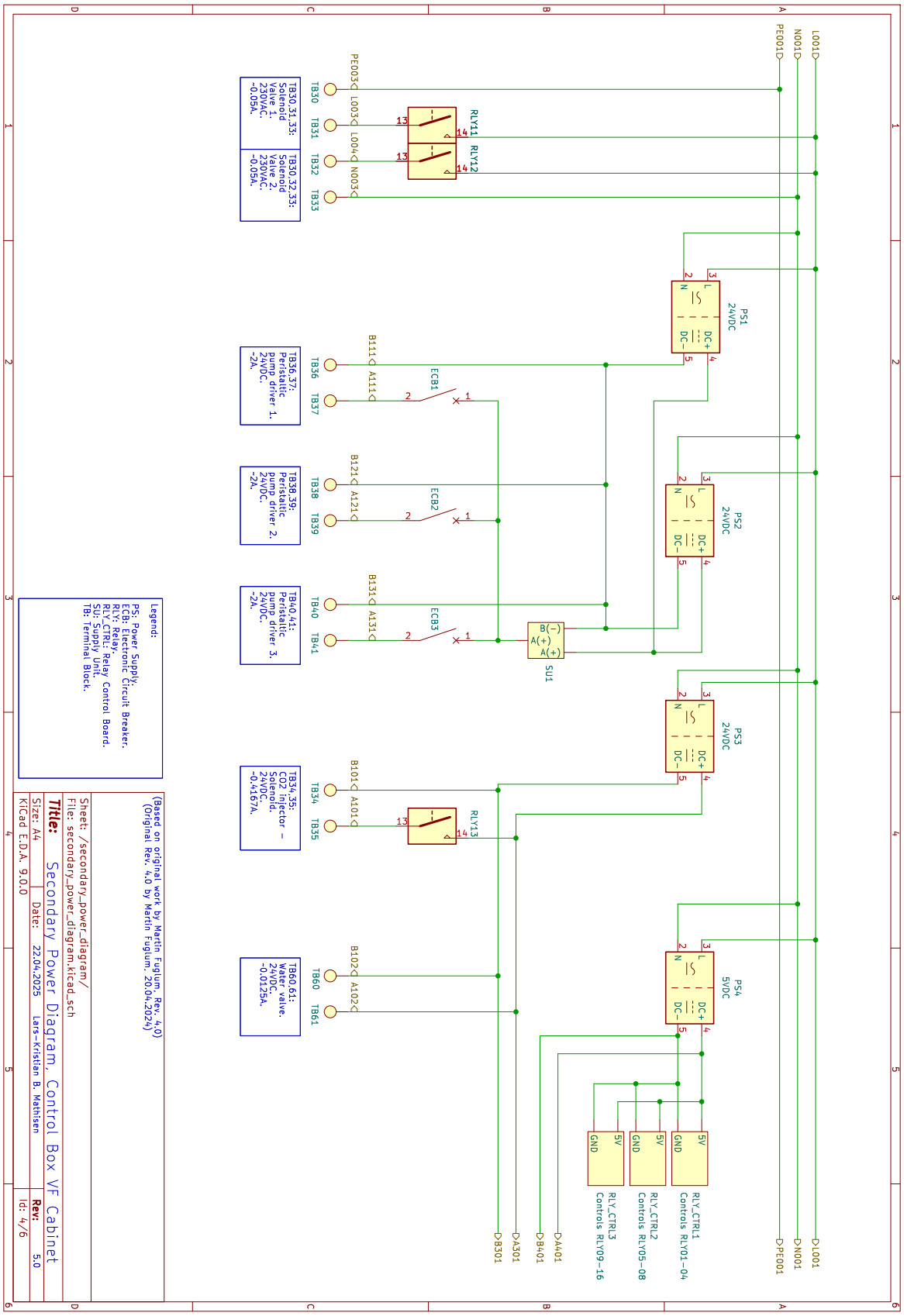
Electronic Control Unit: Electric Wire Diagram

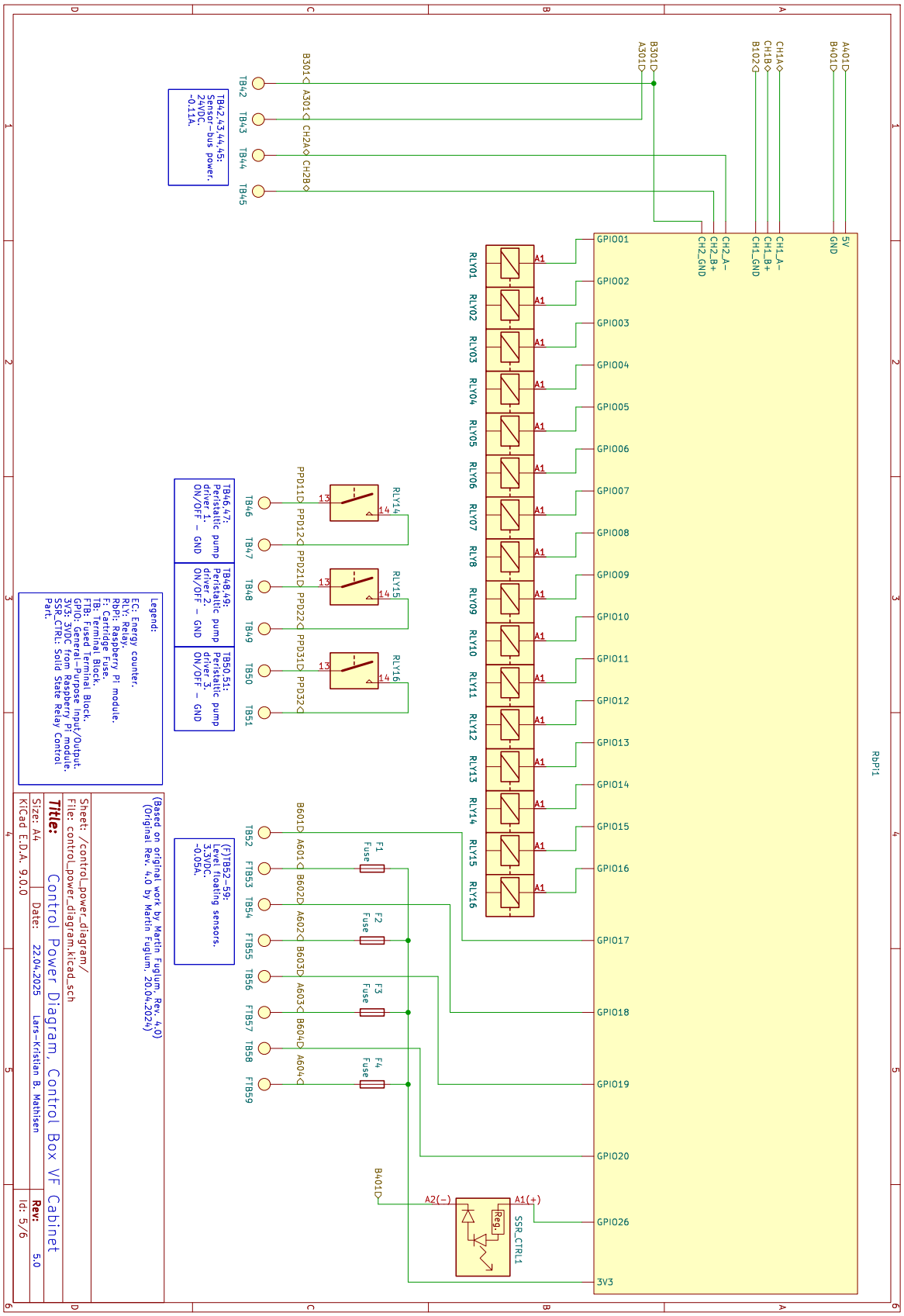
The electric wire diagrams in this appendix document the design and current implementation within the electronic control unit of the vertical farming prototype unit. These diagrams were drawn using KiCad 9.0, which is primarily known as a circuit board design tool. However, it is free to download and was an effective choice for rendering electric schematics in this context. Despite not being recognized as a traditional tool for electric schematics, its familiarity for the author, stemming from previous circuit design experience, facilitated the creation of comprehensive and accurate diagrams.

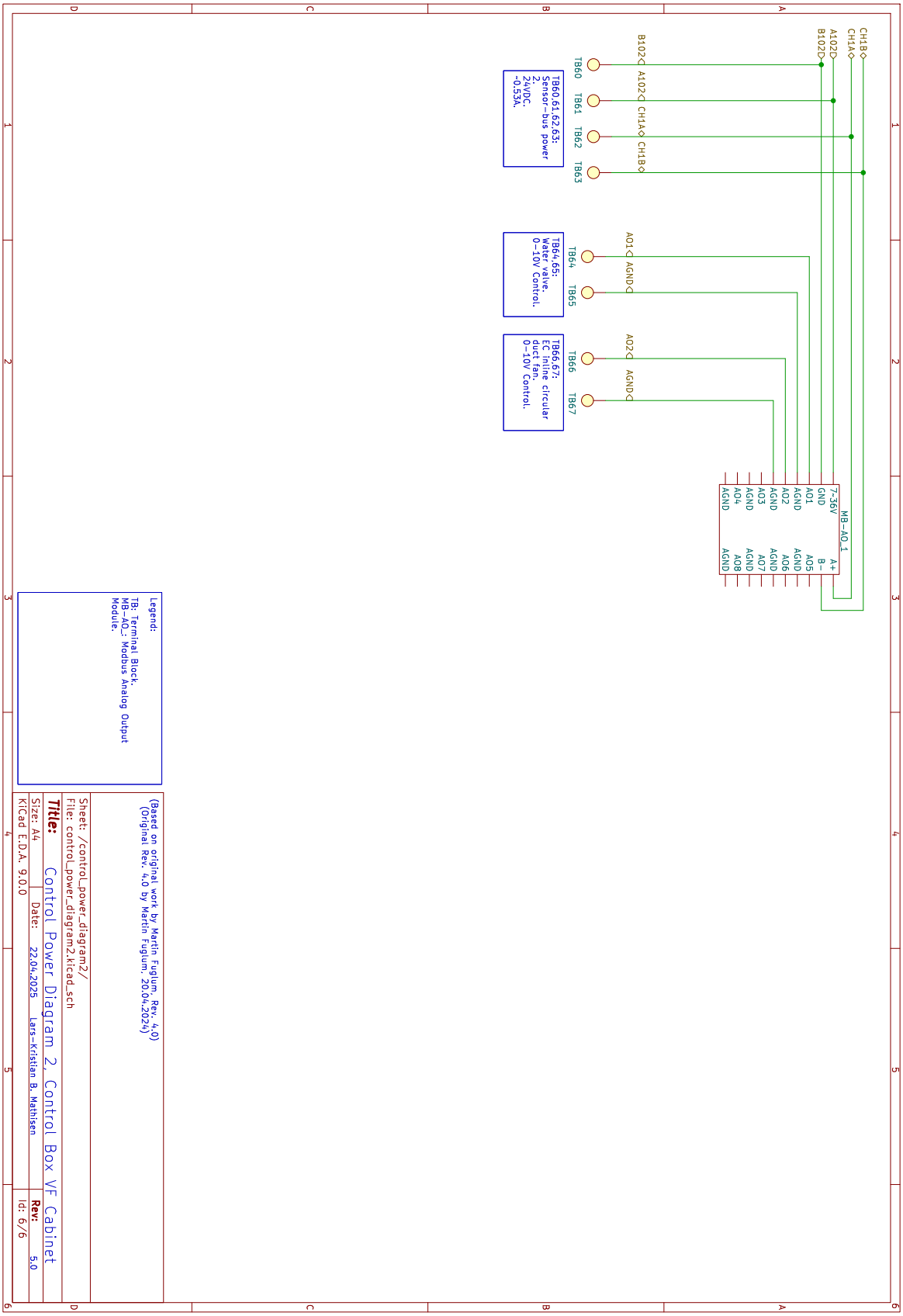

primary_power_diagram
File: primary_power_diagram_pt1.kicad_sch
TB1: Protection Earth. Cabinet.
TB2,3,4: Extra 1. 230VAC.
TB5,6,7: Light lamp. 230VAC. -2.3A.
TB8,9,12: Water pump 1. 230VAC. -0.37A.
TB8,10,12: Water pump 2. 230VAC. -0.2A.
TB8,11,12: Air pump. 230VAC. -0.152A.
TB14: Extra 2. 230VAC.
TB13,15,16: Internal fan 1. 230VAC. -0.16A.
TB17,18,20: Humidifier (unused). 230VAC.
TB19: Dehumidifier (unused). 230VAC.
TB21,22,23: Heater (unused). 230VAC.
TB24,25,26: Control Damper. 230VAC. -0.0196A.
TB27,28,29: Extra 3 - Control. 230VAC.
PE000
PE001
L001
N001
PE002
L002
N002
PE011
L011
L012
L013
N011
PE021
L021
L022
N021
PE031
L031
L032
N031
PE041
L041
N041
PE051
L051
N051
PE061
L061
N061
primary_power_diagram2
File: primary_power_diagram2.kicad_sch
TB68,69,70: EC inline circular fan. 230VAC. -0.765A.
TB71,72,73: Heating element. 230VAC. -2.61A.
PE071
L071
N071
PE081
L081
N081
control_power_diagram
File: control_power_diagram.kicad_sch
TB42,43,44,45: Sensor-bus power. 24VDC. -0.2A.
TB46,47: Peristaltic pump driver 1. ON/OFF-GND
TB48,49: Peristaltic pump driver 2. ON/OFF-GND
TB50,51: Peristaltic pump driver 3. ON/OFF-GND
(F)TB52-59: Level floating sensors. 3.3VDC. -0.05A.
B301
A301
CH2A
CH2B
PPD11
PPD12
PPD21
PPD22
PPD31
PPD32
B601
A601
B602
A602
B603
A603
B604
A604
control_power_diagram2
File: control_power_diagram2.kicad_sch
TB60,61,62,63: Sensor-bus power 2 & MB-AO. 24VDC. -0.53A.
TB64,65: Water valve. 0-10V Control.
TB66,67: EC inline circular duct fan. 0-10V Control.
B102
A102
CH1A
CH1B
AO1
AGND
AO2
secondary_power_diagram
File: secondary_power_diagram.kicad_sch
TB30,31,33: Solenoid Valve 1. 230VAC. -0.05A.
TB30,32,33: Solenoid Valve 2. 230VAC. -0.05A.
TB34,35: CO2 Injector - Solenoid. 24VDC. -0.1167A.
TB36,37: Peristaltic pump driver 1. 24VDC. -2A.
TB38,39: Peristaltic pump driver 2. 24VDC. -2A.
TB40,41: Peristaltic pump driver 3. 24VDC. -2A.
TB60,61: Water valve. 24VDC. -0.0125A.
PE003
L003
L004
N003
B101
A101
B111
A111
B121
A121
B131
A131
A102
B102
Legend:
TB: Terminal Block.
FTB: Fused Terminal Block.
(Based on original work by Martin Fuglum, Rev. 4.0)
(Original Rev. 4.0 by Martin Fuglum, 20.04.2024)
Sheet: /
File: electric_schematic_control_box_vf_2024.kicad_sch
Title:
General Overview, Control Box VF Cabinet
Size: A4
Date: 22.04.2025
Lars-Kristian B. Mathisen
Rev: 5.0
KiCad E.D.A. 9.0.0
Id: 1/6


P1
RCD1
Ground Fault Protection
EC1
EC2
MCB1
MCB2
MCB3
MCB4
MCB5
RLY01
RLY02
RLY03
RLY04
RLY05
RLY06
RLY07
RLY08
RLY09
RLY10
TB1: Protection Earth Cabinet.
TB2,3,4: Extra 1. 230VAC.
TB5,6,7: To lamp. 230VAC. -2.3A.
TB8,9,12: Water pump 1. 230VAC. -0.37A.
TB8,10,12: Water pump 2. 230VAC. -0.2A.
TB8,11,12: Air pump. 230VAC. -0.152A.
TB14: Extra 2. 230VAC.
TB13,15,16: Internal fan 1 and 2. 230VAC. -0.10A.
TB17,18,20: Humidifier (unused). 230VAC.
TB19: Dehumidifier (unused). 230VAC.
TB21,22,23: Heater(unused). 230VAC.
TB24,25,26: Control Damper. 230VAC. -0.019GA.
TB27,28,29: Extra 3 - Gopro. 230VAC.
Legend:
P: External power supply.
RCD: Ground fault protection (Residual Current Device).
EC: Energy Counter, sensor.
MCB: Miniature Circuit Breaker.
RLY: Relay.
TB: Terminal Block.
(Based on original work by Martin Fuglum, Rev. 4.0)
(Original Rev. 4.0 by Martin Fuglum, 20.04.2024)
Sheet: /primary_power_diagram/
File: primary_power_diagram_pt1.kicad_sch
Title: Primary Power Diagram, Control Box VF cabinet
Size: A4
Date: 22.04.2025
Lars-Kristian B. Mathisen
Rev: 5.0
KiCad E.D.A. 9.0.0
Id: 2/6


L001
N001
PE001
MCB6
MCB7
SSR_SW1
ZC
PE071
L071
N071
TB68
TB69
TB70
TB68,69,70: Electric circular duct fan. 230VAC. ~0.765A.
PE081
L081
N081
TB71
TB72
TB73
TB71,72,73: Heating coil. 230VAC. ~2.61A.
Legend:
MCB: Miniature Circuit Breaker.
SSR_SW: Solid State Relay Switch Part.
TB: Terminal Block.
(Based on original work by Martin Fuglum, Rev. 4.0)
(Original Rev. 4.0 by Martin Fuglum; 20.04.2024)
Sheet: /primary_power_diagram2/
File: primary_power_diagram2.kicad_sch
Title: Primary Power Diagram 2, Control Box VF Cabinet
Size: A4
Date: 22.04.2025
Rev: 5.0
KiCad E.D.A. 9.0.0
Lars-Kristian B. Mathisen
Id: 3/6


Legend:
PS: Power Supply.
ECB: Electronic Circuit Breaker.
RLY: Relay.
RLY_CTRL: Relay Control Board.
SU: Supply Unit.
TB: Terminal Block.
(Based on original work by Martin Fuglum, Rev. 4.0)
(Original Rev. 4.0 by Martin Fuglum, 20.04.2024)
Sheet: /secondary_power_diagram/
File: secondary_power_diagram.kicad_sch
Title: Secondary Power Diagram, Control Box VF Cabinet
Size: A4
Date: 22.04.2025
Lars-Kristian B. Mathisen
Rev: 5.0
KiCad E.D.A. 9.0.0
Id: 4/6
PE001
N001
L001
PE003
L003
L004
N003
TB30
TB31
TB32
TB33
TB30,31,33: Solenoid Valve 1. 230VAC. ~0.05A.
TB30,32,33: Solenoid Valve 2. 230VAC. ~0.05A.
RLY11
RLY12
PS1 24VDC
PS2 24VDC
PS3 24VDC
PS4 5VDC
B111
A111
B121
A121
B131
A131
TB36
TB37
TB38
TB39
TB40
TB41
TB36,37: Peristaltic pump driver 1. 24VDC. ~2A.
TB38,39: Peristaltic pump driver 2. 24VDC. ~2A.
TB40,41: Peristaltic pump driver 3. 24VDC. ~2A.
ECB1
ECB2
ECB3
SU1
B(-)
A(+)
A(+)
B101
A101
TB34
TB35
TB34,35: CO2 injector - Solenoid. 24VDC. ~0.4167A.
RLY13
B102
A102
TB60
TB61
TB60,61: Water Valve. 24VDC. ~0.0125A.
5V
GND
RLY_CTRL1 Controls RLY01-04
RLY_CTRL2 Controls RLY05-08
RLY_CTRL3 Controls RLY09-16
A301
B301
A401
B401
L001
N001
PE001


RbPi1
GPIO01
GPIO02
GPIO03
GPIO04
GPIO05
GPIO06
GPIO07
GPIO08
GPIO09
GPIO10
GPIO11
GPIO12
GPIO13
GPIO14
GPIO15
GPIO16
GPIO17
GPIO18
GPIO19
GPIO20
GPIO26
3V3
5V
GND
CH1_A-
CH1_B+
CH1_GND
CH2_A-
CH2_B+
CH2_GND
A401D
B401D
CH1A
CH1B
B102
B301D
A301D
CH2A
CH2B
B301
A301
TB42
TB43
TB44
TB45
TB42,43,44,45: Sensor-bus power. 24VDC. ~0.11A.
RLY01
RLY02
RLY03
RLY04
RLY05
RLY06
RLY07
RLY8
RLY09
RLY10
RLY11
RLY12
RLY13
RLY14
RLY15
RLY16
A1
PPD11D
PPD12C
PPD21D
PPD22C
PPD31D
PPD32C
TB46
TB47
TB48
TB49
TB50
TB51
TB46,47: Peristaltic pump driver 1. ON/OFF - GND
TB48,49: Peristaltic pump driver 2. ON/OFF - GND
TB50,51: Peristaltic pump driver 3. ON/OFF - GND
B601D
A601C
B602D
A602C
B603D
A603C
B604D
A604C
TB52
FTB53
TB54
FTB55
TB56
FTB57
TB58
FTB59
F1 Fuse
F2 Fuse
F3 Fuse
F4 Fuse
(F)TB52-59: Low-side floating sensors. 3.3VDC. ~0.05A.
B401D
A2(-)
A1(+)
Reg.
SSR_CTRL1
Legend:
EC: Energy counter.
RLY: Relay.
RbPi: Raspberry PI module.
F: Cartridge fuse.
TB: Terminal block.
FTB: Fused terminal block.
GPIO: General-Purpose Input/Output.
3V3: 3VDC from Raspberry Pi module.
SSR_CTRL: Solid State Relay Control Part.
(Based on original work by Martin Fuglum, Rev. 4.0)
(Original Rev. 4.0 by Martin Fuglum, 20.04.2024)
Sheet: /control_power_diagram/
File: control_power_diagram.kicad_sch
Title: Control Power Diagram, Control Box VF Cabinet
Size: A4
Date: 22.04.2025
Lars-Kristian B. Mathisen
Rev: 5.0
KiCad E.D.A. 9.0.0
Id: 5/6


CH1B
CH1A
A102
B102
MB-AO_1
7-36V
GND
A+
B-
AO1
AGND
AO2
AGND
AO3
AGND
AO4
AGND
AO5
AGND
AO6
AGND
AO7
AGND
AO8
AGND
B102
A102
CH1A
CH1B
TB60
TB61
TB62
TB63
TB60,61,62,63:
Sensor-bus power
2.
24VDC.
-0,53A.
AO1
AGND
TB64
TB65
TB64,65:
Water valve,
0-10V Control.
AO2
AGND
TB66
TB67
TB66,67:
EC inline circular
duct fan.
0-10V Control.
Legend:
TB: Terminal Block.
MB-AO_: Modbus Analog Output
Module.
(Based on original work by Martin Fuglum, Rev. 4.0)
(Original Rev. 4.0 by Martin Fuglum, 20.04.2024)
Sheet: /control_power_diagram2/
File: control_power_diagram2.kicad_sch
Title: Control Power Diagram 2, Control Box VF Cabinet
Size: A4
Date: 22.04.2025
Lars-Kristian B. Mathisen
Rev: 5.0
KiCad E.D.A. 9.0.0
Id: 6/6

Chapter 7

Electronic Control Unit: Component List

Table 7.1 provides the component list for the Electronic control unit.

Table 7.1: Electronic Control Unit for VF prototype unit 2025 Component List

Label	Component type	MPN	Datasheets
EC1-2	Energy counter	EM111DINAV81XS1X	Datasheet
ECB1-3	Electronic Circuit breaker 2A 24V 1-channel	REX12-TA1-107-DC24V-2A	Datasheet
F1-4	Cartridge Fuse, 5 x 20mm 160mA	VBS USL 160mA 250V (05)	Datasheet
FTB	Fused terminal block	3211861	Datasheet
MCB1-2	Miniature Circuit Breaker 1A	2CDS251001R0014 S201-C1	Datasheet 1, Datasheet 2, Datasheet 3, Datasheet 4
MCB3-4	Miniature Circuit Breaker 2A	2CDS251001R0024 S201-C2	Datasheet 1, Datasheet 2, Datasheet 3
MCB5-6	Miniature Circuit Breaker 1A	2CDS251001R0014 S201-C1	See MCB1-2
MCB7	Miniature Circuit Breaker 3A	2CDS251001R0034 S201-C3	Datasheet 1
P1	External Power supply	-	-
PS1-2	Power supply 24V	DR-4524	Datasheet
PS3	Power supply 24V	MDR-60-24	Datasheet 1, Datasheet 2
PS4	Power supply 5V	1170954	Datasheet

Label	**Component type**	**MPN**	**Datasheets**
RbPi1	Raspberry Pi Module	-	-
RCD1	Ground fault protection	DS201M-B16	-
RLY01-04	Onboard Relay Control Card	TTL-RELAY04	Datasheet
RLY05-08	Onboard Relay Control Card	TTL-RELAY04	Datasheet
RLY09-16	Onboard Relay Control Card	TTL-RELAY08	Datasheet
RLY_CTRL1	Relay Control Card	TTL-RELAY04	Datasheet
RLY_CTRL2	Relay Control Card	TTL-RELAY04	Datasheet
RLY_CTRL3	Relay Control Card	TTL-RELAY08	Datasheet
SSR_SW1	Solid State Relay (Switch part)	RM1A23D25	Datasheet
SSR_CTRL1	Solid State Relay (Control part)	RM1A23D25	See SSR_SW1
SU1	Supply Unit	EM12-T01-001-DC24V-40A	Datasheet 1, Datasheet 2
TB	Terminal block double-level PV	3038464	Datasheet 1, Datasheet 2
TB	Terminal block double-level connected internally PV	3038477	Datasheet
TB	Terminal block double-level connected internally BU	3038493	Datasheet
TB	Terminal block double-level connected internally PE	3038480	Datasheet
MB-AO_1	Modbus RTU Analog Output 8CH	-	Datasheet
SSD	SSD Raspberry Pi Booting unit	-	Link

Chapter 8

Electronic Control Unit: Connection Schedule Terminal Blocks

Table 8.1 provides the connection schedule for the terminal blocks of the Electronic control unit.

Table 8.1: Connection Schedule Terminal Blocks

Terminal Block no.	**Internal connection**	**External connection in VF prototype unit**	**Signal**
TB1	P1	Protection earth for the cabinet	PE 230VAC
TB2	P1	Extra 1	PE 230VAC
TB3	EC1	Extra 1	L 230VAC
TB4	EC1	Extra 1	N 230VAC
TB5	P1	Lamp	PE 230VAC
TB6	EC2	Lamp	L 230VAC
TB7	EC2	Lamp	N 230VAC
TB8	P1	Water pump 5k, 350 and Air pump	PE 230VAC
TB9	RLY01	Water pump 5k l/hour	L 230VAC
TB10	RLY02	Water pump 350 l/hour	L 230VAC
TB11	RLY03	Air pump	L 230VAC
TB12	EC1	Water pump 5k, 350 and Air pump	N 230VAC
TB13	P1	Internal fan 1 and 2	PE 230VAC
TB14	RLY04	Extra 2	L 230VAC
TB15	RLY05	Internal fan 1 and 2	L 230VAC
TB16	EC1	Internal fan 1 and 2	N 230VAC
TB17	P1	Humidifier max. 450W	PE 230VAC

Terminal Block no.	**Internal connection**	**External connection in VF prototype unit**	**Signal**
TB18	RLY06	Humidifier max. 450W	L 230VAC
TB19	RLY07	Dehumidifier(unused)	L 230VAC
TB20	EC1	Humidifier(unused)	N 230VAC
TB21	P1	Heater(unused)	PE 230VAC
TB22	RLY08	Heater(unused)	L 230VAC
TB23	EC1	Heater(unused)	N 230VAC
TB24	EC1	Control damper	L 230VAC
TB25	RLY09	Control damper	L 230VAC
TB26	EC1	Control damper	N 230VAC
TB27	P1	Extra 3 - Gopro	PE 230VAC
TB28	RLY10	Extra 3 - Gopro	L 230VAC
TB29	EC1	Extra 3 - Gopro	N 230VAC
TB30	P1	Solenoid valve 1 and 2	PE 230VAC
TB31	RLY11	Solenoid valve 1	L 230VAC
TB32	RLY12	Solenoid valve 2	L 230VAC
TB33	EC1	Solenoid valve 1 and 2	N 230VAC
TB34	PS3	CO2 injector - Solenoid B	24V (GND)
TB35	RLY13	CO2 injector - Solenoid A	24V
TB36	PS1 and 2	Peristaltic pump driver 1	B 24V (GND)
TB37	ECB1	Peristaltic pump driver 1	A 24V
TB38	PS1 and 2	Peristaltic pump driver 2	B 24V (GND)
TB39	ECB2	Peristaltic pump driver 2	A 24V
TB40	PS1 and 2	Peristaltic pump driver 3	B 24V (GND)
TB41	ECB3	Peristaltic pump driver 3	A 24V
TB42	PS3	Sensor-bus 1	B 24V (GND)
TB43	PS3	Sensor-bus 1	A 24V
TB44	RbPi1	Sensor-bus CH2	RS-485 A-
TB45	RbPi1	Sensor-bus CH2	RS-485 B+
TB46	RLY14	Peristaltic pump driver 1	ON/OFF
TB47	RLY14	Peristaltic pump driver 1	GND
TB48	RLY15	Peristaltic pump driver 2	ON/OFF
TB49	RLY15	Peristaltic pump driver 2	GND
TB50	RLY16	Peristaltic pump driver 3	ON/OFF
TB51	RLY16	Peristaltic pump driver 3	GND
TB52	RbPi1	Level floating sensor 1	B 3V3 (GND)
FTB53	F1	Level floating sensor 1	A 3V3
TB54	RbPi1	Level floating sensor 2	B 3V3 (GND)
FTB55	F2	Level floating sensor 2	A 3V3
TB56	RbPi1	Level floating sensor 3	B 3V3 (GND)

Terminal Block no.	**Internal connection**	**External connection in VF prototype unit**	**Signal**
FTB57	F3	Level floating sensor 3	A 3V3
TB58	RbPi1	Level floating sensor 4	B 3V3 (GND)
FTB59	F4	Level floating sensor 4	A 3V3
TB60	PS3	Water valve and Sensor-bus 2	B 24V (GND)
TB61	PS3	Water valve and Sensor-bus 2	A 24V
TB62	RbPi1	Modbus Analog output and Sensor-bus 2 CH1	RS-485 A-
TB63	RbPi1	Modbus Analog output and Sensor-bus 2 CH1	RS-485 B+
TB64	AO1	Water valve	0-10V Control
TB65	AGND	Water valve	GND
TB66	AO2	EC inline circular duct fan	0-10V Control
TB67	AGND	EC inline circular duct fan	GND
TB68	EC1	EC inline circular duct fan	PE 230VAC
TB69	EC1	EC inline circular duct fan	L 230VAC
TB70	P1	EC inline circular duct fan	N 230VAC
TB71	SSR_SW1	Heating coil	PE 230VAC
TB72	EC1	Heating coil	L 230VAC
TB73	P1	Heating coil	N 230VAC

Chapter 9

Electronic Control Unit: Connection Schedule Raspberry Pi Module

Table 9.1 provides the connection schedule for the Raspberry Pi module of the Electronic control unit.

Table 9.1: Connection schedule Raspberry Pi module

Pin NO.	**Function Raspberry Pi**	**GPIO NO.**	**Application**	**Ext. Board Pin**	**GPIO Screw Terminal Hat label**
1	3.3V Power	-			3V3
2	5V Power	-	PS4 supply		5V
3	GPIO 2 (I2C SDA)	2	RLY1	TTL	SDA
4	5V Power	-			5V
5	GPIO 3 (I2C SCL)	3	RLY2	TTL	SCL
6	Ground	-	PS4 supply		GND
7	GPIO 4 (GPCLK0)	4	RLY3	TTL	IO4
8	GPIO 14 (UART TXD)	14	RLY4	TTL	TXD
9	Ground	-			GND
10	GPIO 15 (UART RXD)	15	RLY5	TTL	RXD
11	GPIO 17 (SPI1 CE1)	17	RLY6	TTL	IO17
12	GPIO 18 (PCM_CLK) (SPI1 CE0)	18	RS-485 Hat	CS	IO18
13	GPIO 27	27	RS-485 Hat	EN1	IO27
14	Ground	-			GND
15	GPIO 22	22	RS-485 Hat	EN2	IO22
16	GPIO 23	23	RLY7	TTL	IO23
17	3.3V Power	-			3V3
18	GPIO 24	24	RS-485 Hat	IRQ	IO24
19	GPIO 10 (SPI MOSI)	10	RLY8	TTL	MOSI
20	Ground	-			GND
21	GPIO 9 (SPI MISO)	9	RLY9	TTL	MISO
22	GPIO 25	25	RLY10	TTL	IO25
23	GPIO 11 (SPI SCLK)	11	RLY11	TTL	SLCK
24	GPIO 8 (SPI CE0)	8	RLY12	TTL	CE0
25	Ground	-			GND
26	GPIO 7 (SPI CE1)	7	Floating Sensor 1		CE1
27	GPIO 0 (I2C ID_SD)	0	Floating Sensor 2		IDSD
28	GPIO 1 (I2C ID_SC)	1	Floating Sensor 3		IDSC
29	GPIO 5	5	Floating Sensor 4		IO5
30	Ground	-			GND
31	GPIO 6	6	RLY13	TTL	IO6
32	GPIO 12 (PWM0)	12	RLY14	TTL	IO12
33	GPIO 13 (PWM1)	13	RLY15	TTL	IO13
34	Ground	-			GND
35	GPIO 19 (PCM_FS) (SPI1 MISO)	19	RS-485 Hat	MISO	IO19
36	GPIO 16	16	RLY16	TTL	IO16
37	GPIO 26	26	SSR_CTRL1	TTL	IO26
38	GPIO 20 (PCM_DIN) (SPI1 MOSI)	20	RS-485 Hat	MOSI	IO20
39	Ground	-	SSR_CTRL1		GND
40	GPIO 21 (PCM_DOUT) (SPI1 SCLK)	21	RS-485 Hat	SLCK	IO21

Bibliography

[1] R. S. Press, 'HOW TO: Boot Raspberry Pi 4 from USB SSD Drive,' 2.2021, [Online]. Available: https://www.raspberrystreet.com/learn/how-to-boot-raspberrypi-from-usb-ssd, (visited on 23/04/2025).