DCMonitoring for VGPIoT - Datacenter 1

General

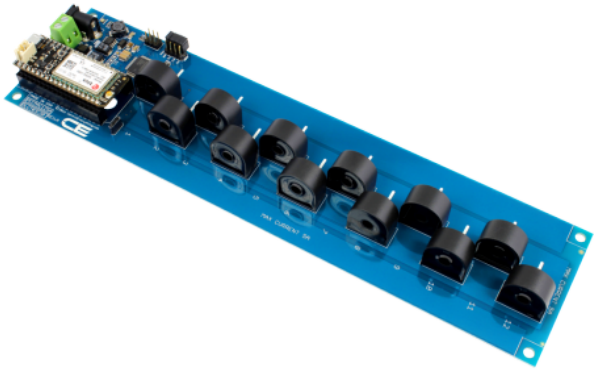
In this scope I will be describing the parameters that are currently being monitored. A screen is installed in each datacenter and shows a dashboard.

The monitoring is done in 4 big sections: power management, sample air quality, cameras and Grafana.

Datacenter 1 and **Datacenter 2** are located in Turnhout. The layout of these centers can be found here: Datacenter Locations - Turnhout (http://wiki01.prd.priv.vangenechten.com/mediawiki/index.php/Datacenter_Locations_-_Turnhout)

Components

- 1x PECMAC125A current monitoring module (https://store.ncd.io/product/12-channel-on-board-95-accuracy-20-amp-ac-current-monitor-with-iot-interface/)



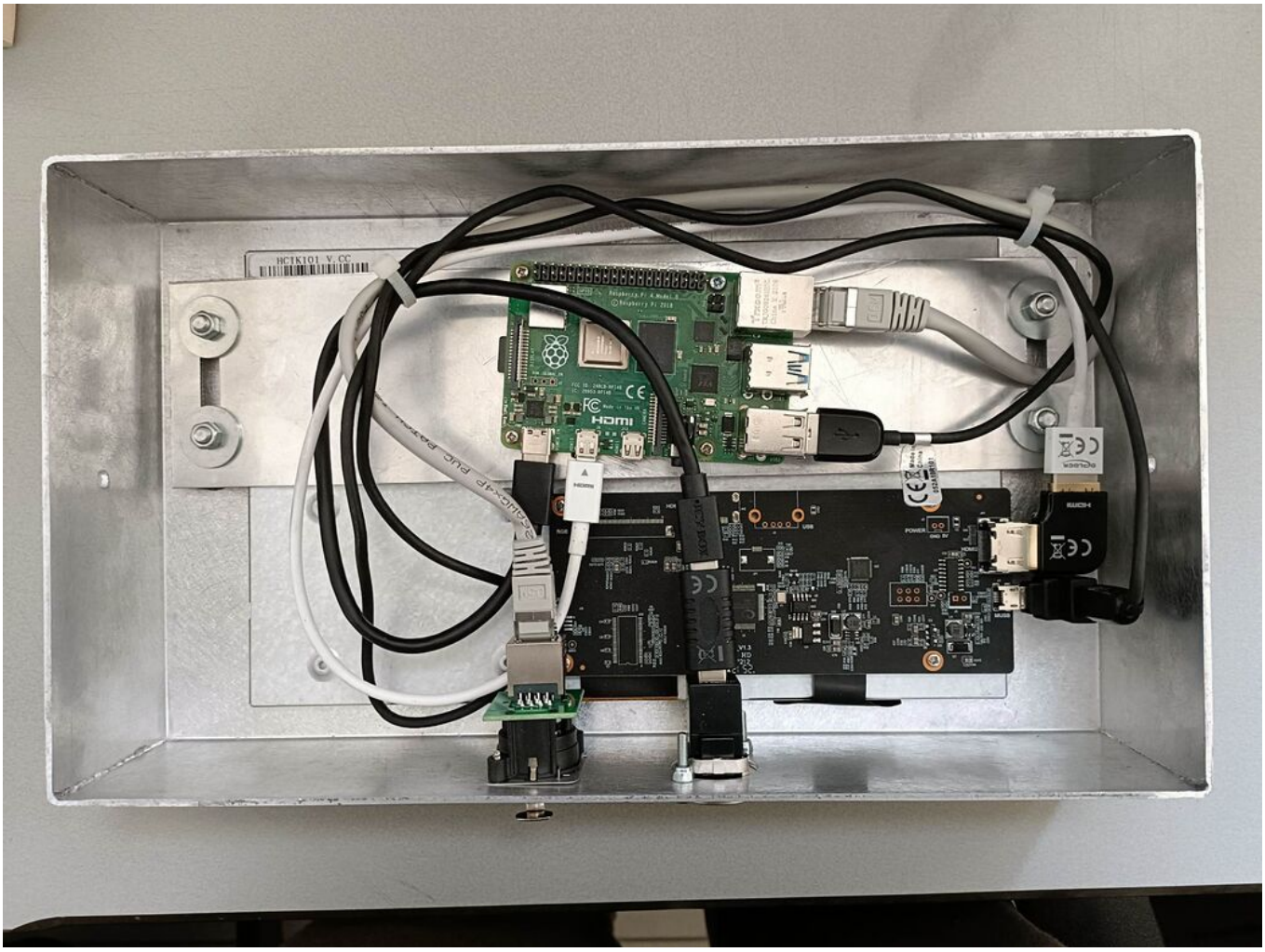
- 1x NCD wireless usb modem (https://store.ncd.io/product/industrial-wireless-usb-modem/)

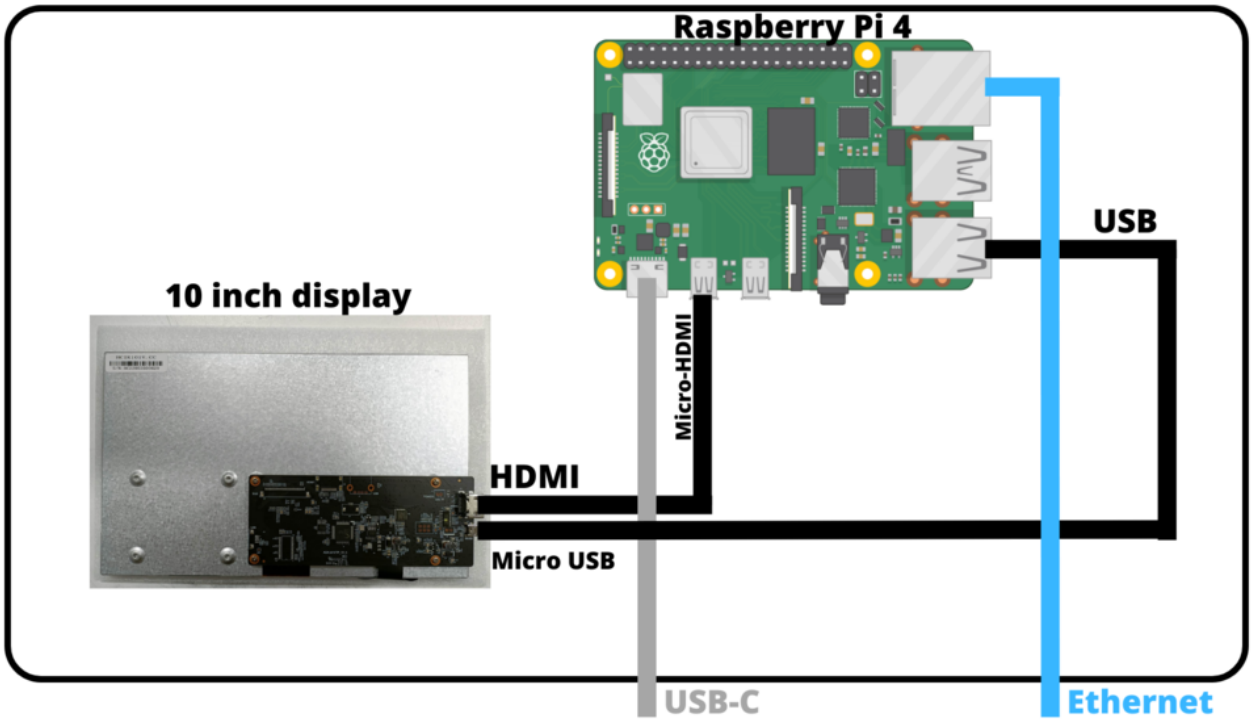
- 1x USB micro cable (https://www.allekabels.be/usb-micro-kabel/4911/3347008/micro-usb-kabel.html?gclid=Cj0KCQiA0oagBhDHARIsAI-BbgeQocB0V8z5fAg1dQyA6Zw-YzvmB9-JqaCSwGrbBt0ARon3FE8vU7saAlLnEALw_wcB)

- 1x NCD air quality sensor (https://store.ncd.io/product/industrial-iot-wireless-air-quality-co2-temperature-humidity-particulate-matter-sensor/)

- 4x UniFi cameras (https://eu.store.ui.com/collections/unifi-protect-cameras)

- 1x display for Raspberry Pi





Dashboard

A general view of each **dashboard** is made in **Node-Red** and can be found here: http://10.1.60.158:1880/ui/#!/0?socketid=s01kD71LtTKKazfmAA9r

Node-Red configuration: http://10.1.60.158:1880/

Node-Red Requirements

Node-red version: v3.0.2

Node.js version: v12.22.12

npm version: 6.14.16

Power management

This view shows the current that is flowing in three different power sources

Utility current

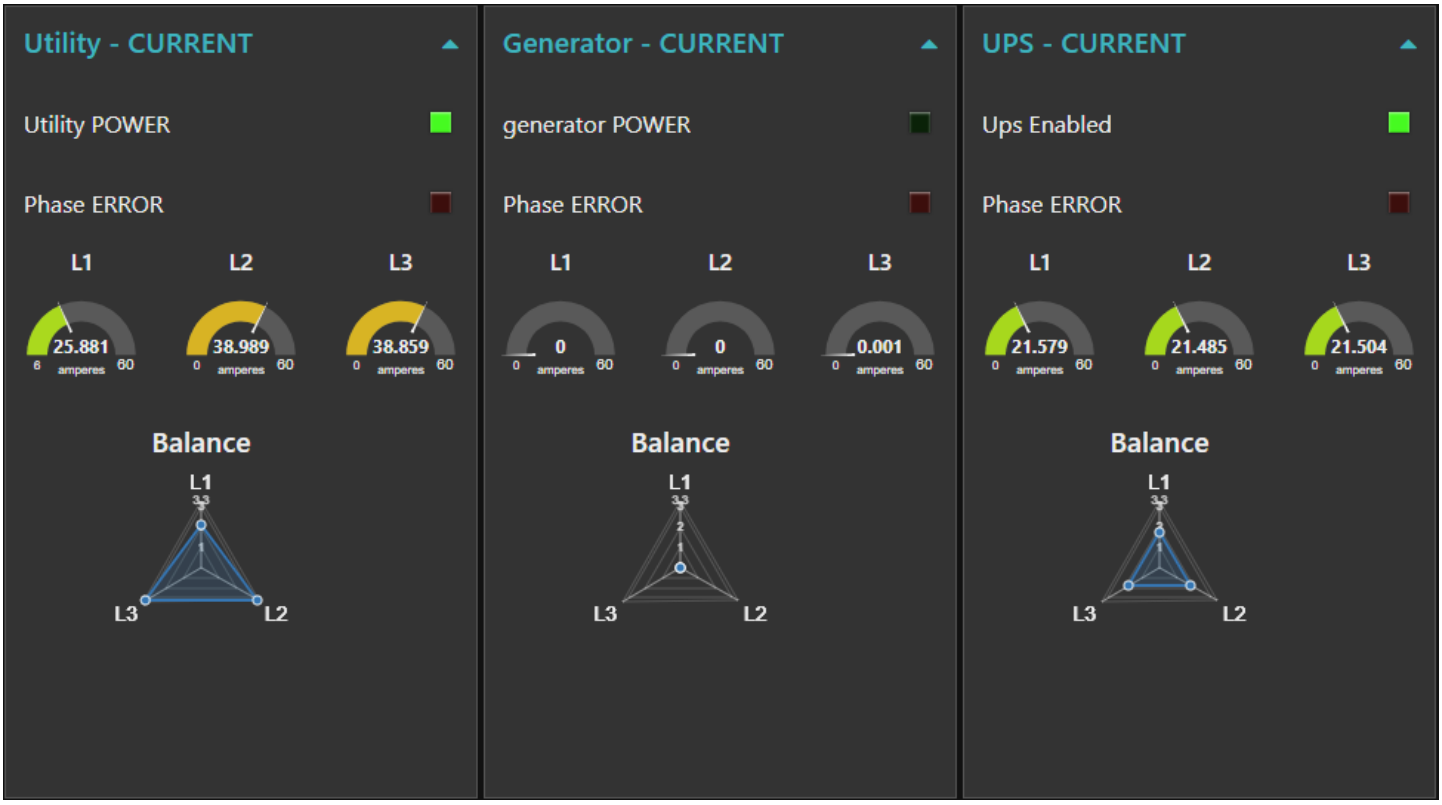
The power that is currently being used by the servers.

Generator current

The power that must be turned on in case the power grid is down.

UPS (Uninterruptable Power Supply) current

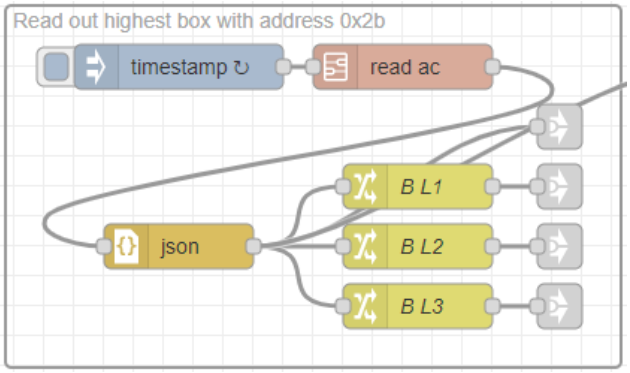
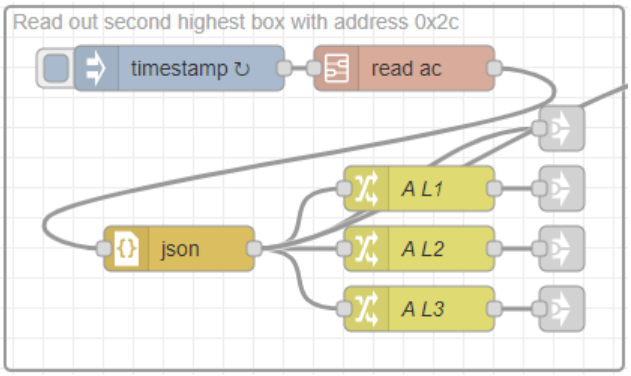
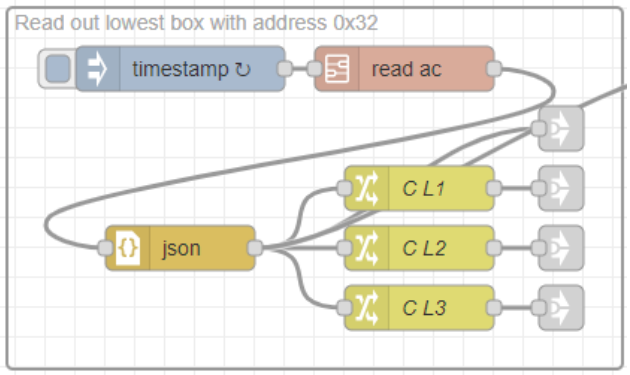
An **uninterruptible power supply** (UPS), also known as a battery backup, provides backup power when your regular power source fails or voltage drops to an unacceptable level. An **UPS** allows for the safe, orderly shutdown of a computer and connected equipment. The size and design of a UPS determines how long it will supply power.



Flows

AC read

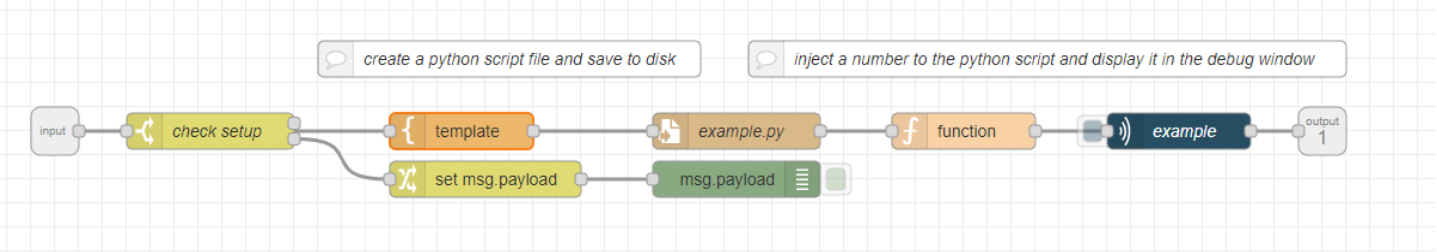
This flow reads the active current that is flowing in the cables of the different sources with **I2C**.

Flow	Power source	I2C address (hex)	I2C address (dec)
	Utility current	0x2b	43
	Generator current	0x2c	44
	UPS current	0x32	50

The "**read ac**" subflow needs some variables to read the **AC** current.

- address (i2c address that determines the source that the current comes from, this is the only variable that changes in the subflow.)
- header_1 (146, 0x92)
- header_2 (106, 0x6A)
- min_range (starting range)
- offset_read (85, 0x55)
- nr_channel (amount of channels to read from)

Their data values are stored in a JSON object and is then separated in the 3 different lines from a 3-phase electric supply: **L1, L2 and L3**.



Python script

The template node has a python script. This script uses **I2C** to read ac current.

The code to read this data is based on a Github repository. However it is not the same. We modified it to work with Node-Red. It does the following:

1. Import necessary libraries
2. Get variables from subflow properties
3. Write a command to get i2c bus
4. Write to tell i2c to read data
5. Read data
6. Convert bytes -> data -> ampere -> dictionary -> JSON object

Original code:
https://github.com/ControlEverythingCommunity/PECMAC/blob/master/Python/PECMAC125A.py

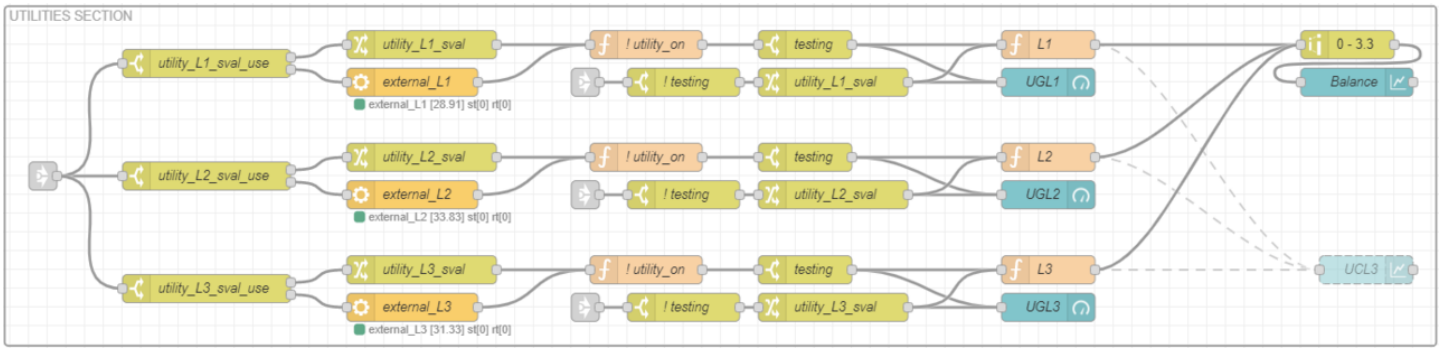
Modified code

Copy Code

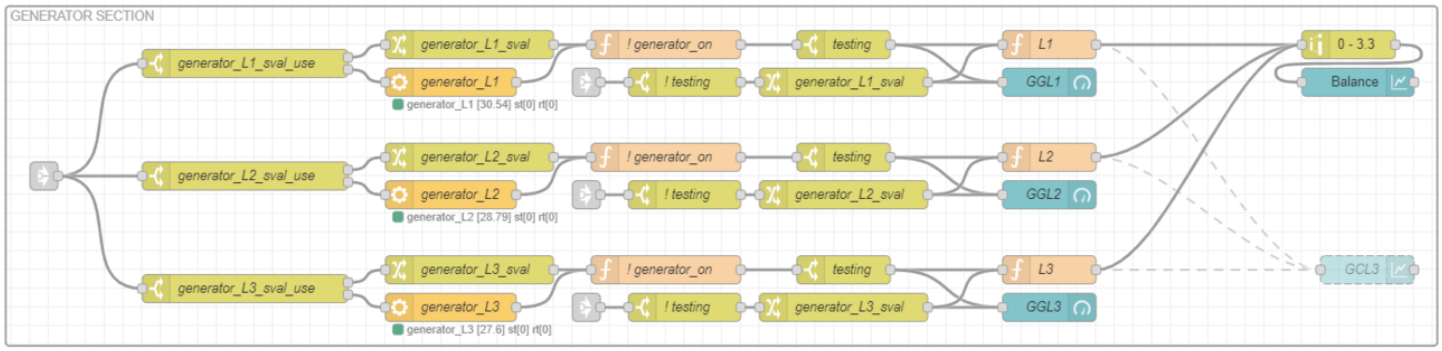
```
# Distributed with a free-will license.
# Use it any way you want, profit or free, provided it fits in the licenses of its associated works.
# PECMAC125A
# This code is designed to work with the PECMAC125A_DLCT03C20 I2C Mini Module available from ControlEverything.com.
# https://www.controleverything.com/content/Current?sku=PECMAC125A_DLCT03C20#tabs-0-product_tabset-2

import smbus
import time
import sys
import functools
import json

address = int(sys.stdin.readline())
header_1 = int(sys.stdin.readline())
header_2 = int(sys.stdin.readline())
min_range = int(sys.stdin.readline())
offset_read = int(sys.stdin.readline())
nr_channel = int(sys.stdin.readline())

write_command = 0x01
max_range = min_range + nr_channel -1
reserved_1 = 0x00
reserved_2 = 0x00


commands = [header_2, write_command, min_range, max_range, reserved_1, reserved_2]

foldl = lambda func, acc, xs: functools.reduce(func, xs, acc)
```

```
checksum = foldl(lambda x, y: x + y, header_1, commands) % 256
commands.append(checksum)


# Get I2C bus
bus = smbus.SMBus(1)

# PECMAC125A address, 0x2A(42)
# Command for reading current
# 0x6A(106), 0x01(1), 0x01(1),0x0C(12), 0x00(0), 0x00(0) 0x0A(10)
# Header byte-2, command-1, start channel-1, stop channel-12, byte 5 and 6 reserved, checksum
bus.write_i2c_block_data(address, header_1, commands)

time.sleep(0.5)

# PECMAC125A address, 0x2A(42)
# Read data back from 0x55(85), No. of Channels * 3 bytes
# current MSB1, current MSB, current LSB
data1 = bus.read_i2c_block_data(address, offset_read, nr_channel*3 + 2)
sumation = 0x00
ret = ""
# Convert the data
diction = {}
for i in range(0, nr_channel) :
        msb1 = data1[i * 3]
        msb = data1[1 + i * 3]
        lsb = data1[2 + i * 3]
        sumation += msb1 + msb + lsb
        # Convert the data to ampere
        current = (msb1 * 65536 + msb * 256 + lsb) / 1000.0

        # Output in a dictionary
        diction["L" + str(i + 1)] = str(current)

if(sumation % 256 == data1[3 * nr_channel]):
    print(json.dumps(diction))
```

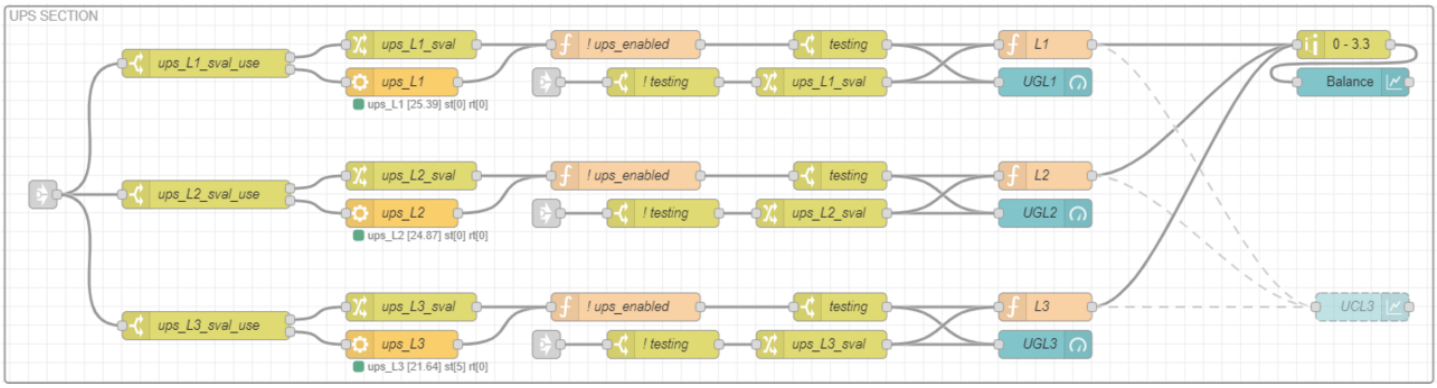
Utility



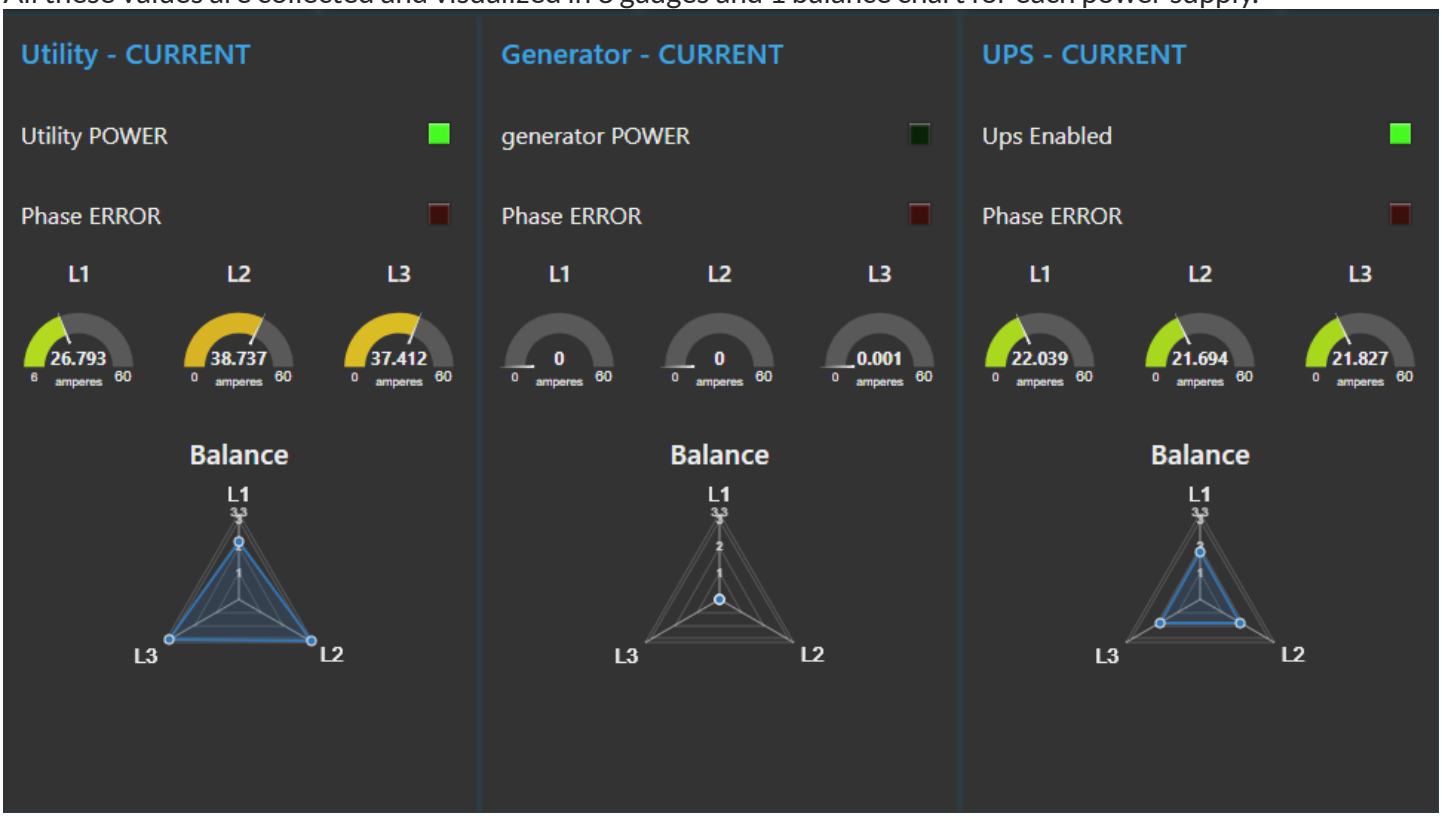
Generator



UPS



All these values are collected and visualized in 3 gauges and 1 balance chart for each power supply.

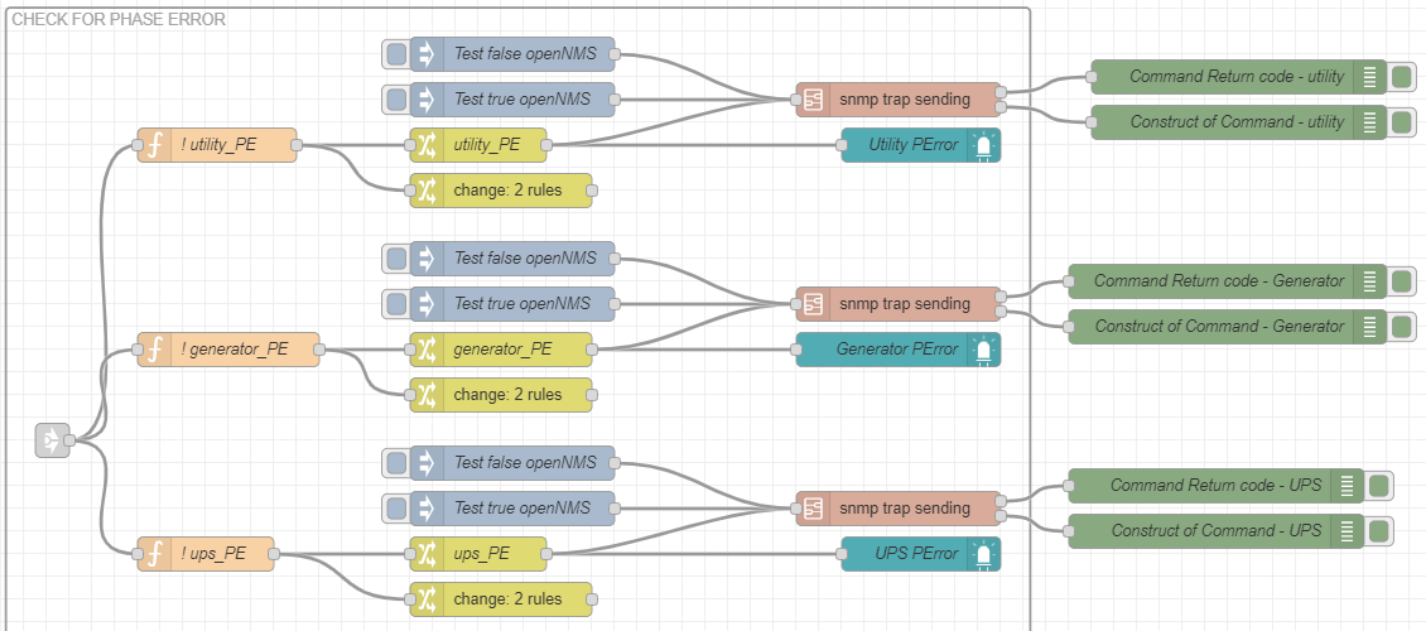


SNMP traps

Monitoring is done by sending **SMNP** traps. An alert will be send when there is a certain power error. This table is here to help you to know which **oid**'s are in use. These oid's are used to send a message from a smartbox to **OpenNMS**. So that Mario can decide whether people should be **alerted** or not. Currently simple traps tell the state of the machine.

Events: https://onms.priv.vangenechten.com/opennms/event/list?sortby=id&acktype=unack&limit=20&filter=node%3D1273&filter=nodelocation%3DDefault

Flows



Utility error check

function *! utility_PE*

```
if ( flow.get("utility_on") && ( flow.get("utility_L1_sval") == 0.00 ||
flow.get("utility_L2_sval") == 0.00 || flow.get("utility_L3_sval") == 0.00 ) )  {

    flow.set("utility_PE", true );
} else {
     flow.set("utility_PE", false );
}


return msg;
```

Copy Code

Generator error check

function *! generator_PE*

```
if ( flow.get("generator_on") && ( flow.get("generator_L1_sval") == 0.00 ||
flow.get("generator_L2_sval") == 0.00 || flow.get("generator_L3_sval") == 0.00 ) )  {

    flow.set("generator_PE", true );
} else {
     flow.set("generator_PE", false );
}
```

Copy Code

```
return msg;
```

UPS error check

function *! ups_PE*

```
if ( flow.get("ups_enabled") && ( flow.get("ups_L1_sval") == 0.00 || flow.get("ups_L2_sval") == 0.00 || flow.get("ups_L3_sval") == 0.00 ) )  {

    flow.set("ups_PE", true );
} else {
     flow.set("ups_PE", false );
}


return msg;
```

SNMP List

Name source	Object identifier main	Object identifier argument	Value
Utility	.1.3.6.1.4.1.30290.3.**3**	.1.3.6.1.4.1.30290.3.3.**1**	up
Generator	.1.3.6.1.4.1.30290.3.**5**	.1.3.6.1.4.1.30290.3.5.**1**	not in use
UPS	.1.3.6.1.4.1.30290.3.**7**	.1.3.6.1.4.1.30290.3.7.**1**	not in use
Utility	.1.3.6.1.4.1.30290.3.**4**	.1.3.6.1.4.1.30290.3.4.**1**	down
Generator	.1.3.6.1.4.1.30290.3.**6**	.1.3.6.1.4.1.30290.3.6.**1**	in use
UPS	.1.3.6.1.4.1.30290.3.**8**	.1.3.6.1.4.1.30290.3.8.**1**	in use

SNMP trap sending

Properties

This **shell script** sends an "snmpinform" command that gets environment variables from the properties.

```
#!/bin/bash

sudo snmpinform -v3 -u monitor_read -a SHA -A '{{auth_pass}}'  -x AES -X '{{privacy_pass}}'
{{ip_to}} '' {{main_oid}}  {{oid_argument}} s  "{{name_source}} {{value_1}}"
```

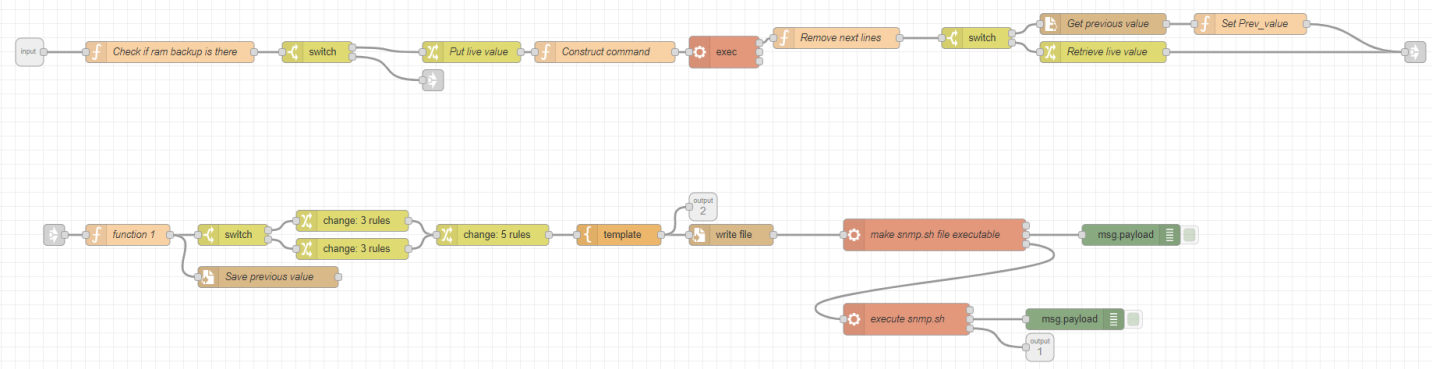
Copy Code

When an error is **TRUE**, the SNMP script will write **TRUE** to the according snmp_trap_xxx.txt file.

```
snmp_trap_generator.txt
snmp_trap_netstroom.txt
snmp_trap_ups.txt
```

By **default**, every .txt file has the value: **false.** This means that the power is **UP** and **RUNNING.**

Subflow



Environs

The quality of the air is monitored by 4 different variables. **CO2**, **temperature**, **humidity**, **particulate matter**.

CO2

Carbon dioxide, or **CO2**, is a nontoxic, noncombustible air pollutant and a colorless greenhouse gas. It's a natural product of human and animal respiration and can be created naturally through processes like volcanic eruptions. However, most carbon dioxide emissions are human produced through human activity.

To minimize the risk of airborne transmission of viruses, CO2 levels should be as possible in all indoor spaces. It is recommended to stay close to **400 ppm**.

CO2 (ppm)	Air quality
< 600	Excellent
600 -> 1000	Good
1000 -> 1500	Mediocre
1500 <	Bad

Temperature

Temperature is a physical quantity that expresses quantitatively the perceptions of hotness and coldness.

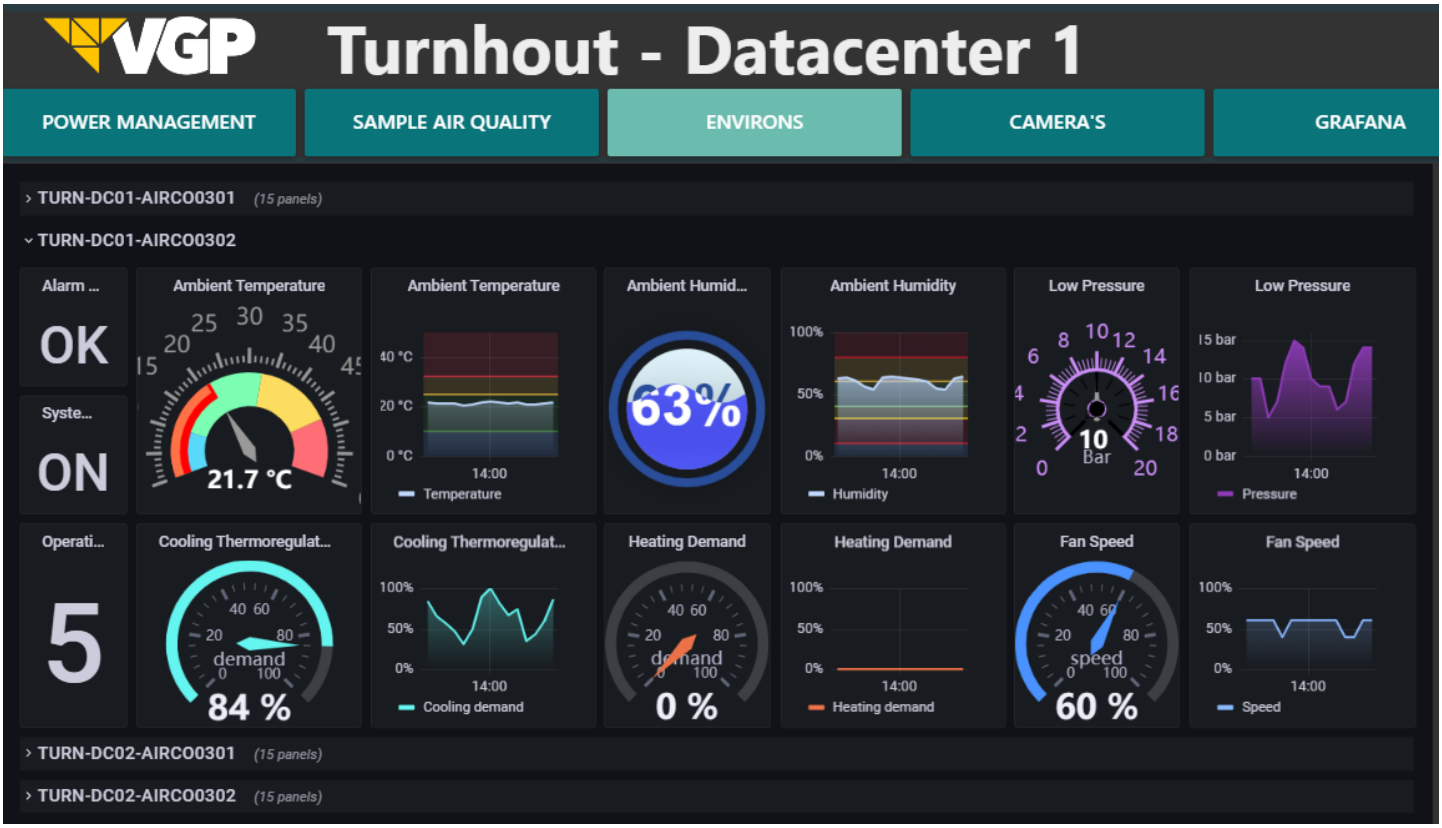
Temperature in this room should remain around 15 to 24 degrees **Celsius**.

Humidity

Humidity is **the amount of water vapor in the air**. If there is a lot of water vapor in the air, the humidity will be high.

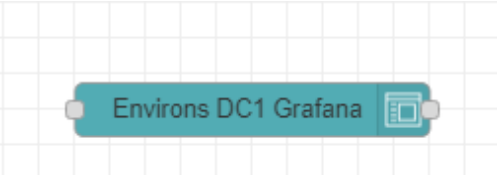
Although you can't see it, it's still there. The ideal relative humidity for health and comfort is somewhere **between 30-70% humidity**. This means that the air holds between 30-70% of the maximum amount of moisture it can contain.

User interface



Flow

Link: https://observer0302.priv.vangenechten.com:4443/d/P1zimBE4k/airco-turn-dc?orgId=1&refresh=1m&kiosk



Air quality

Particulate matter

Particle pollution — also called particulate matter (PM) — is made up of **particles** (tiny pieces) of solids or liquids that are in the air. These particles may include:

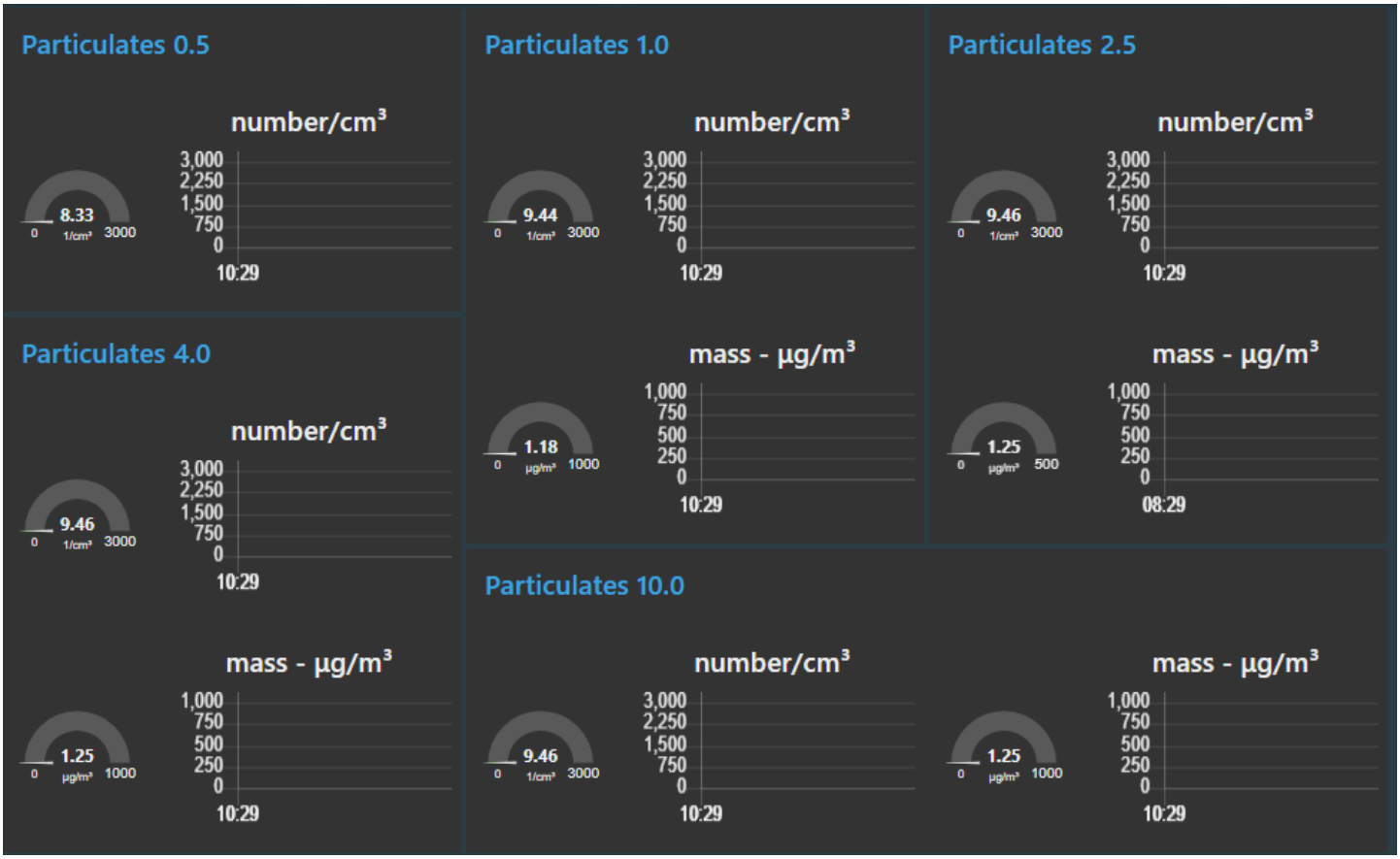
- Dust
- Dirt
- Soot
- Smoke
- Drops of liquid

Some particles are big enough (or appear dark enough) to see — for example, you can often see smoke in the air. Others are so small that you can't see them in the air.

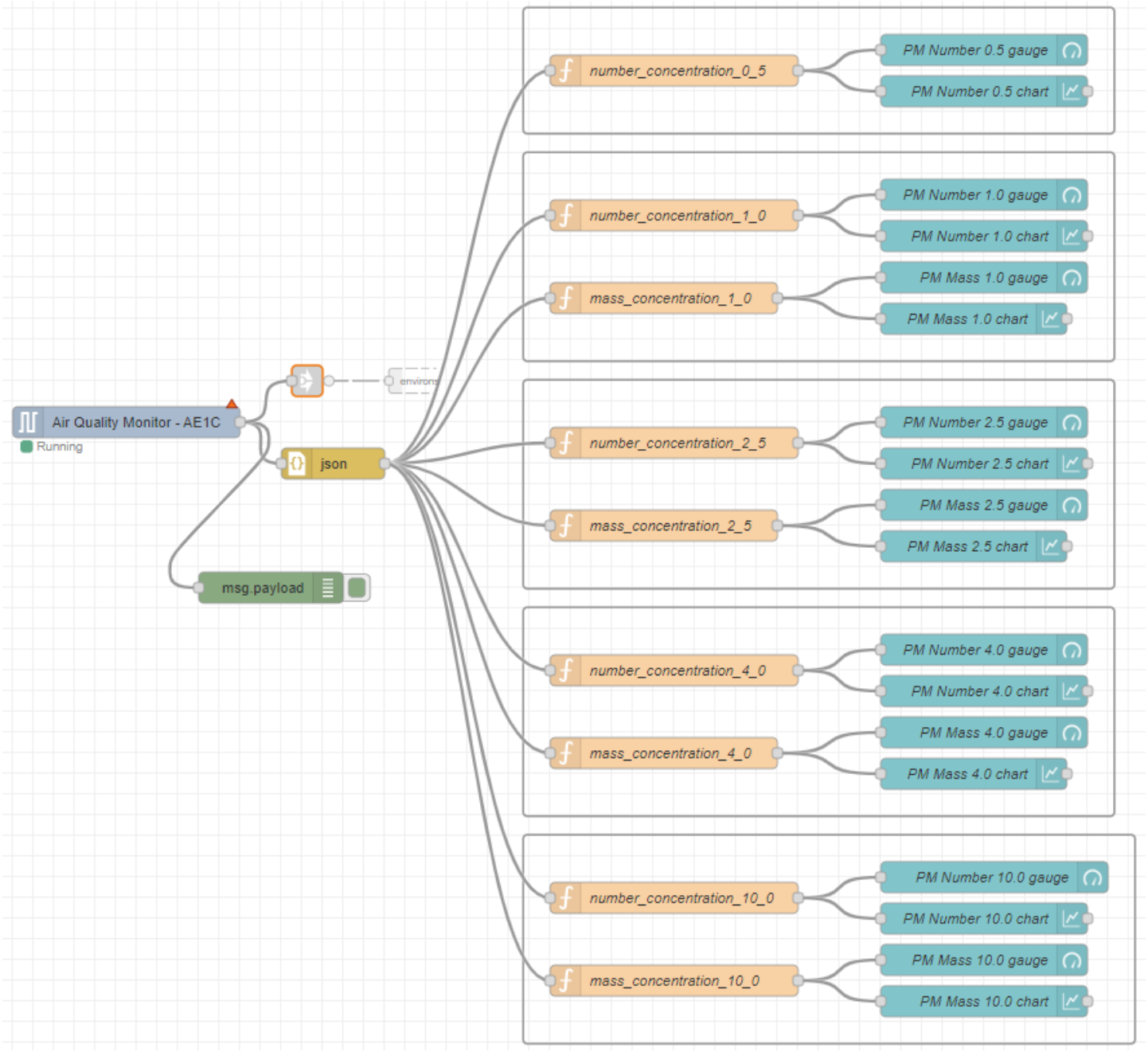
This **Particulate Matter Sensor** is designed to measure **PM0.5, PM1.0, PM2.5, PM4.0 and PM10.0**, providing a particle count for each size indicated.

PM2.5 and PM10 refer to particulate matter with particle diameter up to 2.5 microns and 10 microns respectively, which are among the most **dangerous air pollutants**. Due to their small size, PM2.5 particles can travel deep into the human lung and cause a variety of health issues.

User interface


Particulates 0.5
number/cm³
8.33
0
1/cm³
3000
3,000
2,250
1,500
750
0
10:29
Particulates 1.0
number/cm³
9.44
0
1/cm³
3000
3,000
2,250
1,500
750
0
10:29
Particulates 2.5
number/cm³
9.46
0
1/cm³
3000
3,000
2,250
1,500
750
0
10:29
Particulates 4.0
number/cm³
9.46
0
1/cm³
3000
3,000
2,250
1,500
750
0
10:29
mass - µg/m³
1.25
0
µg/m³
1000
1,000
750
500
250
0
10:29
mass - µg/m³
1.18
0
µg/m³
1000
1,000
750
500
250
0
10:29
mass - µg/m³
1.25
0
µg/m³
500
1,000
750
500
250
0
08:29
Particulates 10.0
number/cm³
9.46
0
1/cm³
3000
3,000
2,250
1,500
750
0
10:29
mass - µg/m³
1.25
0
µg/m³
1000
1,000
750
500
250
0
10:29

Flow



NCD sensors

Mac address	Description	Type	Sensor readings
00:13:a2:00:41:d7:ae:1c	CO2 Temperature Humidity PM Sensor	53	CO2, Humidity, Celcius, PM
00:13:a2:00:41:d7:ac:fc	Wireless USB Modem		

Wireless air quality sensor.

2 **NCD** devices are configured to collect all these values. A **sensor** and a **modem**.

Cameras

Datacenter 1 has **4 UniFi** cameras. These are installed in case of a physical data breach or if you want to check the inside from a remote location. If the background is **green**, it means the camera is turned on and showing live feed.

Camera configuration: Installing a Unifi Camera with Node-Red

Camera	RTSP_URL
CAM-301	rtsp://10.1.16.9:7447/61ea91fdc2dcbcf8de67fdf2_2
CAM-302	rtsp://10.1.16.9:7447/61ea91a0c2dcbcf8de67fdf0_2
CAM-303	rtsp://10.1.16.9:7447/61ea9146c2dcbcf8de67fdee_2
CAM-304	rtsp://10.1.16.9:7447/61ea9114c2dcbcf8de67fdec_2

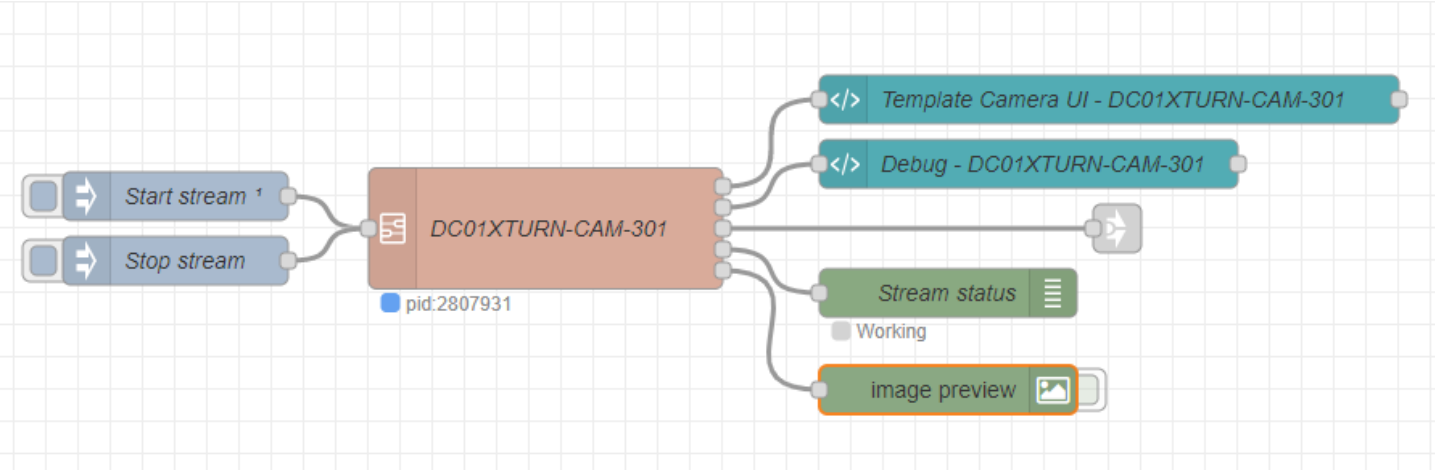
Flows

Every camera has the same flow to capture live feed and stream it to the UI. The only difference between each camera is the **RTSP_URL**.

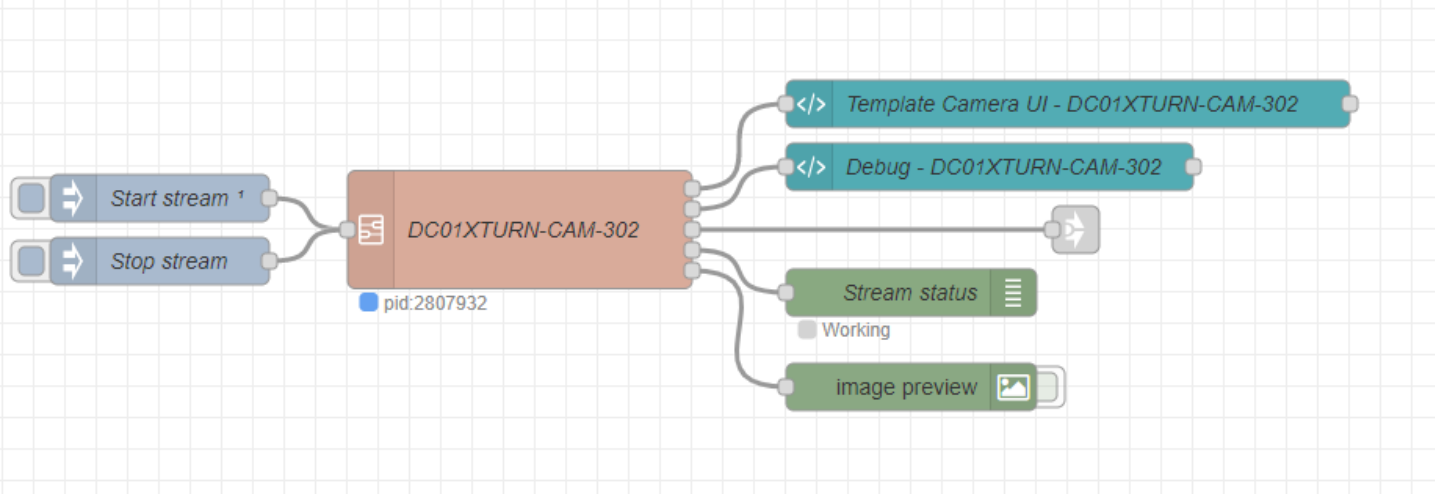
The **Real Time Streaming Protocol** (**RTSP**) is an application-level network protocol designed for multiplexing and packetizing multimedia transport streams (such as interactive media, video and audio) over a suitable transport protocol. RTSP is used in entertainment and communications systems to control streaming media

servers. The protocol is used for establishing and controlling media sessions between endpoints. Clients of media servers issue commands such as *play*, *record* and *pause*, to facilitate real-time control of the media streaming from the server to a client (video on demand) or from a client to the server (voice recording).

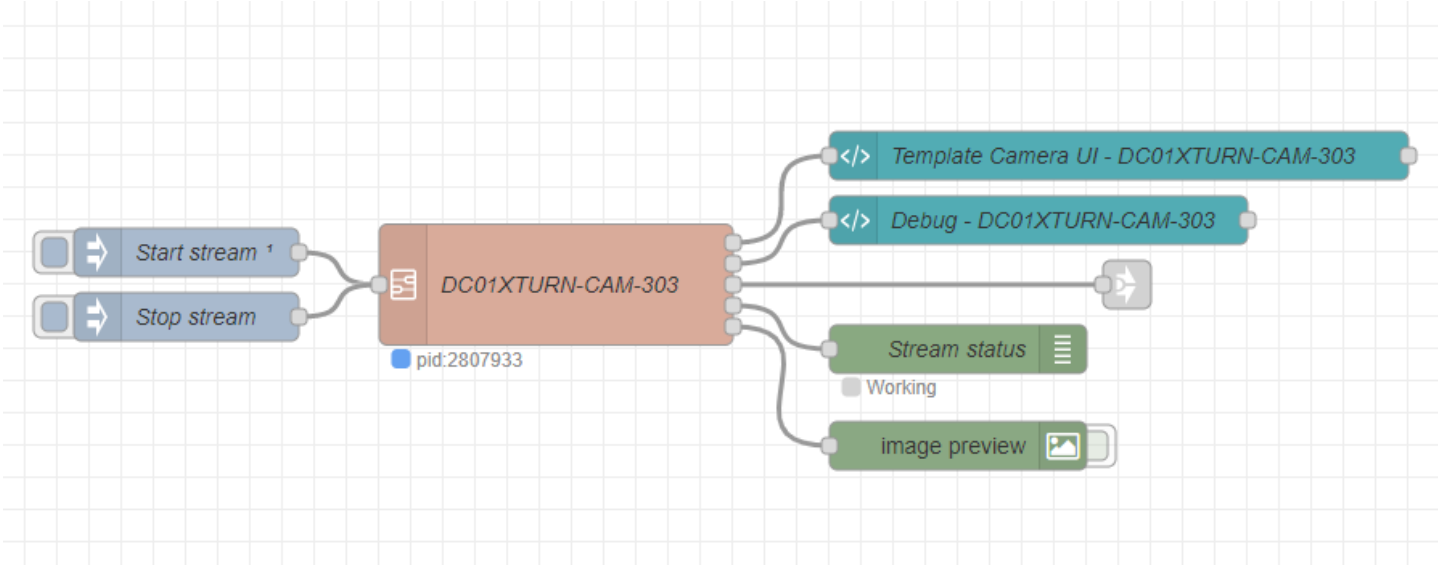
CAM-301



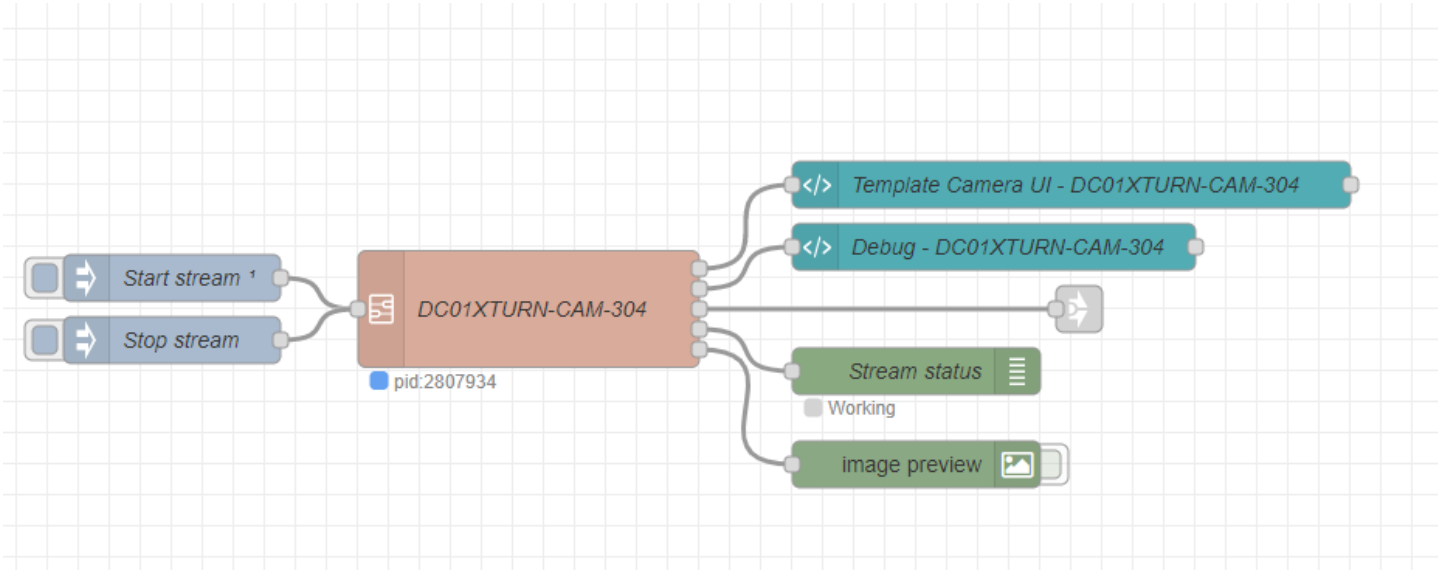
CAM-302



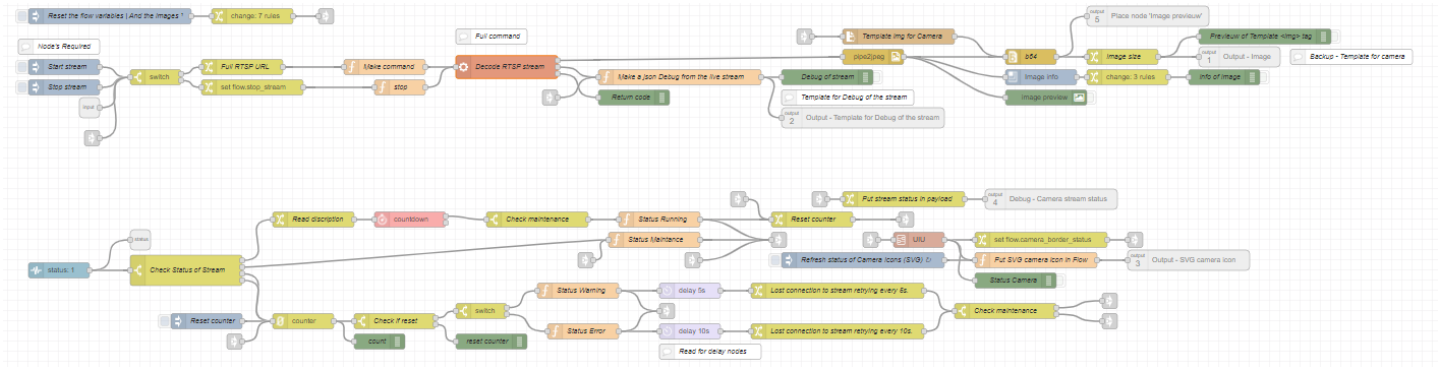
CAM-303



CAM-304



Live - IP Camera subflow



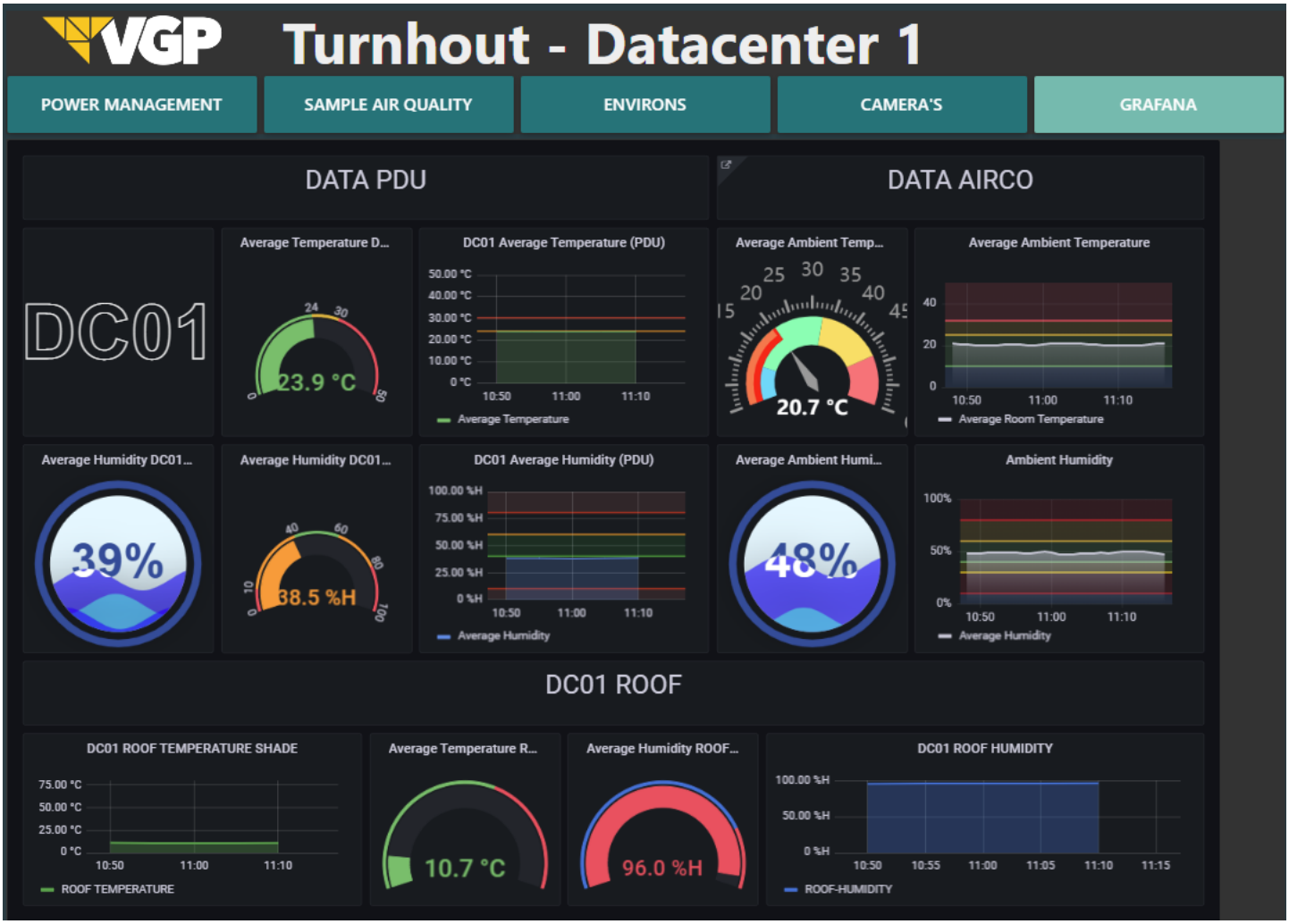
Grafana

https://grafana.com/

Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data-driven culture

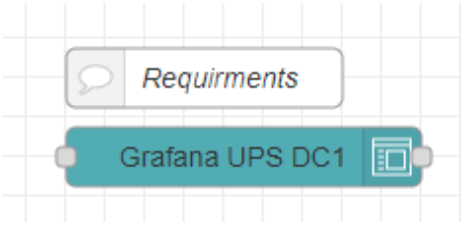
This grafana dashboard shows more details about **DC1**.

Link: https://observer0302.priv.vangenechten.com:4443/d/crF6C4J7z/turnhout_dc01_termohydrograph?orgId=1&refresh=1m&kiosk



Flow

The flow is an iframe with the grafana url.

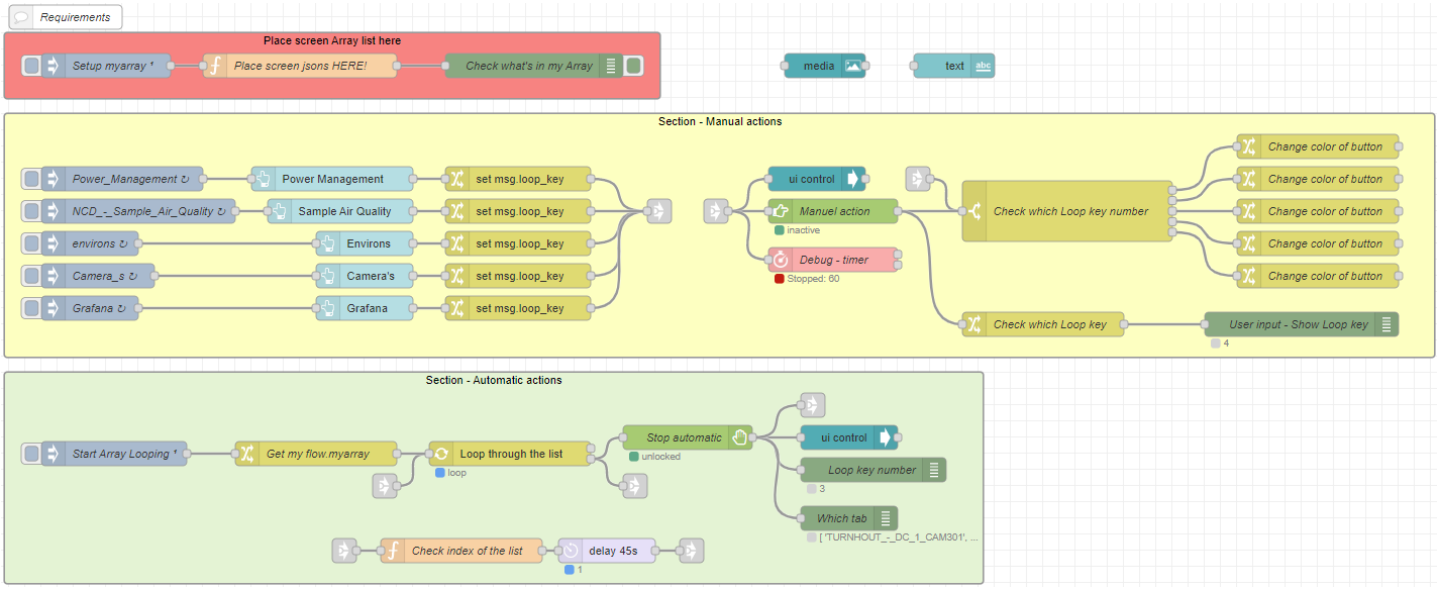


AutoSwitch - Dashboard Tabs

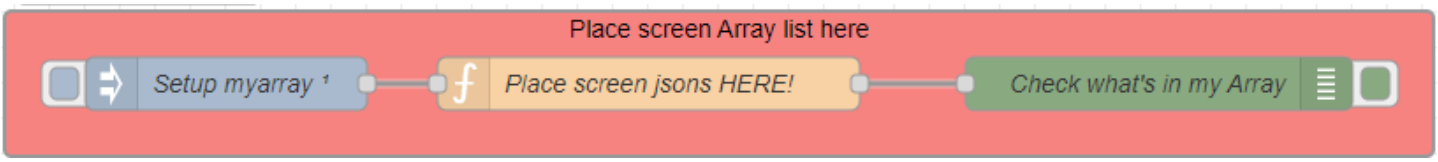
Main flow

The main purpose of this flow is to make an automatic and manual control panel for selecting the desired tab. When you press the tab "**Camera's**" the loop key is set to 3 but the loop key number is set to 4.

Tab	src variable	loop_key	loop key number
Power management	src1	0	1
Sample air quality	src2	1	2
environs	src3	2	3
Camera's	src4	3	4
Grafana	src5	4	5



Array List



JavaScript file

There are 5 buttons: Power management, Sample air quality, environs, Camera's and Grafana. Each button relates to their src-variable. Variables are put in an array. The variable is a JSON-object that shows/hides the groups that you want to show.

```
var src1 = {
    "group": {
        "show": [
            "TURNHOUT_-_DC_1_Power_-_STATUS",
            "TURNHOUT_-_DC_1_Utility_-_CURRENT",
            "TURNHOUT_-_DC_1_Generator_-_CURRENT",
            "TURNHOUT_-_DC_1_UPS_-_CURRENT",
            "TURNHOUT_-_DC_1_Extra_-_CURRENT"
        ],
        "hide": [

            "TURNHOUT_-_DC_1_Particulates_0.5",
            "TURNHOUT_-_DC_1_Particulates_1.0",
            "TURNHOUT_-_DC_1_Particulates_2.5",
            "TURNHOUT_-_DC_1_Particulates_4.0",
            "TURNHOUT_-_DC_1_Particulates_10.0",

            "TURNHOUT_-_DC_1_Environs_Grafana",

            "TURNHOUT_-_DC_1_CO2",
            "TURNHOUT_-_DC_1_Temperature",
            "TURNHOUT_-_DC_1_Humidity",

            "TURNHOUT_-_DC_1_CAM301",
            "TURNHOUT_-_DC_1_CAM302",
            "TURNHOUT_-_DC_1_CAM303",
            "TURNHOUT_-_DC_1_CAM304",
```

Copy Code

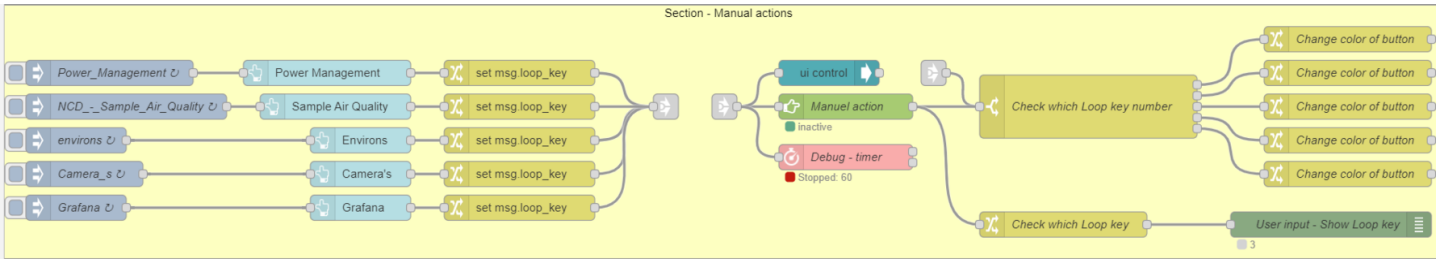
```
            "TURNHOUT_-_DC_1_Grafana"
        ]
    }
}


var src2 = {
    "group": {
        "show": [

            "TURNHOUT_-_DC_1_Particulates_0.5",
            "TURNHOUT_-_DC_1_Particulates_1.0",
            "TURNHOUT_-_DC_1_Particulates_2.5",
            "TURNHOUT_-_DC_1_Particulates_4.0",
            "TURNHOUT_-_DC_1_Particulates_10.0",

        ],
        "hide": [
            "TURNHOUT_-_DC_1_Power_-_STATUS",
            "TURNHOUT_-_DC_1_Utility_-_CURRENT",
            "TURNHOUT_-_DC_1_Generator_-_CURRENT",
            "TURNHOUT_-_DC_1_UPS_-_CURRENT",
            "TURNHOUT_-_DC_1_Extra_-_CURRENT",

            "TURNHOUT_-_DC_1_CO2",
            "TURNHOUT_-_DC_1_Temperature",
            "TURNHOUT_-_DC_1_Humidity",

            "TURNHOUT_-_DC_1_Environs_Grafana",

            "TURNHOUT_-_DC_1_CAM301",
            "TURNHOUT_-_DC_1_CAM302",
            "TURNHOUT_-_DC_1_CAM303",
            "TURNHOUT_-_DC_1_CAM304",

            "TURNHOUT_-_DC_1_Grafana"
        ]
    }
}

var src3 = {
    "group": {
        "show": [
            "TURNHOUT_-_DC_1_Environs_Grafana"
        ],
        "hide": [
            "TURNHOUT_-_DC_1_Power_-_STATUS",
            "TURNHOUT_-_DC_1_Utility_-_CURRENT",
            "TURNHOUT_-_DC_1_Generator_-_CURRENT",
            "TURNHOUT_-_DC_1_UPS_-_CURRENT",
            "TURNHOUT_-_DC_1_Extra_-_CURRENT",

            "TURNHOUT_-_DC_1_CO2",
            "TURNHOUT_-_DC_1_Temperature",
            "TURNHOUT_-_DC_1_Humidity",

            "TURNHOUT_-_DC_1_Particulates_0.5",
            "TURNHOUT_-_DC_1_Particulates_1.0",
            "TURNHOUT_-_DC_1_Particulates_2.5",
            "TURNHOUT_-_DC_1_Particulates_4.0",
            "TURNHOUT_-_DC_1_Particulates_10.0",
```

```
            "TURNHOUT_-_DC_1_CAM301",
            "TURNHOUT_-_DC_1_CAM302",
            "TURNHOUT_-_DC_1_CAM303",
            "TURNHOUT_-_DC_1_CAM304",

            "TURNHOUT_-_DC_1_Grafana"
        ]
    }
}


var src4 = {
    "group": {
        "show": [
            "TURNHOUT_-_DC_1_CAM301",
            "TURNHOUT_-_DC_1_CAM302",
            "TURNHOUT_-_DC_1_CAM303",
            "TURNHOUT_-_DC_1_CAM304"
        ],
        "hide": [
            "TURNHOUT_-_DC_1_Power_-_STATUS",
            "TURNHOUT_-_DC_1_Utility_-_CURRENT",
            "TURNHOUT_-_DC_1_Generator_-_CURRENT",
            "TURNHOUT_-_DC_1_UPS_-_CURRENT",
            "TURNHOUT_-_DC_1_Extra_-_CURRENT",

            "TURNHOUT_-_DC_1_CO2",
            "TURNHOUT_-_DC_1_Temperature",
            "TURNHOUT_-_DC_1_Humidity",

            "TURNHOUT_-_DC_1_Environs_Grafana",

            "TURNHOUT_-_DC_1_Particulates_0.5",
            "TURNHOUT_-_DC_1_Particulates_1.0",
            "TURNHOUT_-_DC_1_Particulates_2.5",
            "TURNHOUT_-_DC_1_Particulates_4.0",
            "TURNHOUT_-_DC_1_Particulates_10.0",


            "TURNHOUT_-_DC_1_Grafana"
        ]
    }
}

var src5 = {
    "group": {
        "show": [
            "TURNHOUT_-_DC_1_Grafana"
        ],
        "hide": [
            "TURNHOUT_-_DC_1_Power_-_STATUS",
            "TURNHOUT_-_DC_1_Utility_-_CURRENT",
            "TURNHOUT_-_DC_1_Generator_-_CURRENT",
            "TURNHOUT_-_DC_1_UPS_-_CURRENT",
            "TURNHOUT_-_DC_1_Extra_-_CURRENT",

            "TURNHOUT_-_DC_1_CO2",
            "TURNHOUT_-_DC_1_Temperature",
            "TURNHOUT_-_DC_1_Humidity",

            "TURNHOUT_-_DC_1_Environs_Grafana",

            "TURNHOUT_-_DC_1_Particulates_0.5",
```

```
            "TURNHOUT_-_DC_1_Particulates_1.0",
            "TURNHOUT_-_DC_1_Particulates_2.5",
            "TURNHOUT_-_DC_1_Particulates_4.0",
            "TURNHOUT_-_DC_1_Particulates_10.0",


            "TURNHOUT_-_DC_1_CAM301",
            "TURNHOUT_-_DC_1_CAM302",
            "TURNHOUT_-_DC_1_CAM303",
            "TURNHOUT_-_DC_1_CAM304"

        ]
    }
}


// If you add more src[numbers] also add them in the variable "Myarray"
var myarray = [src1, src2, src3, src4,src5]
// Here i put the "myarray in a flow"
flow.set('myarray', myarray)


// This is for debugging reasons
msg.payload = myarray
return msg;
```

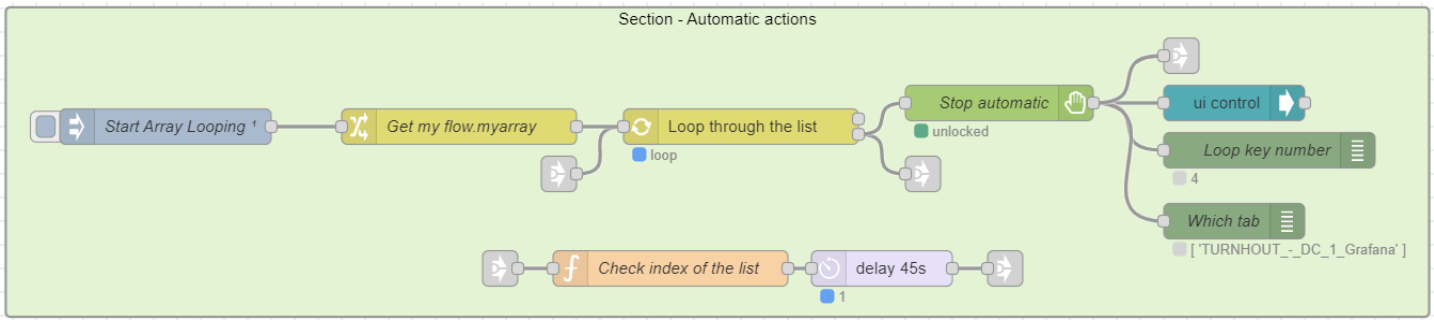
Manual actions

Flow for manual button selection.



Automatic actions

Flow for automatic loop through the buttons.



Back to: Projects Pages Overview - DCMonitoring for VGPIoT
Back to: Main Page

Status Open Priority B SMTTemplate SMTProject SMT Project Page VGPIoT DCMonitoring