DCMonitoring for VGPIoT - Datacenter 2

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)

10.1.61.183

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/)

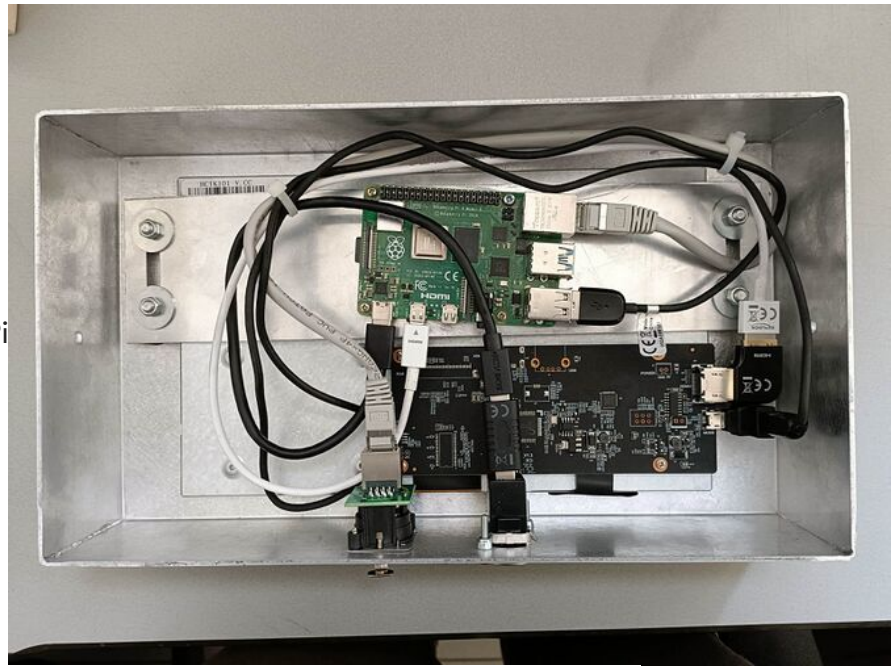
- 1x USB micro cable (https://www.allekabels.be/usb-micro-kabel/4911/3347008/micro-usb-kabel.html?gclid=Cj0KCQiA0oagBhDHARIsAI-BbgeQocB0V8z5fAg1dQyA6Zw-YzvmB9-JqaCSwGrbBt0ARon3FE8vU7s

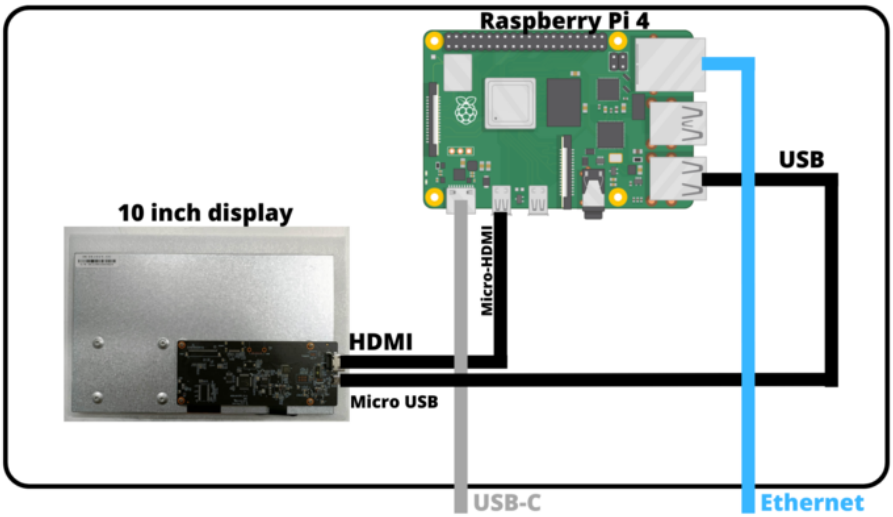
aAILnEALw_wcB)

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

- 6x 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://10.1.60.221:1880/ui/#!/0?socketid=n6QSITQjCdVDpDWRAA5h

Node-Red configuration: http://10.1.60.221:1880/

Node-Red Requirements

Node-red version: v3.0.2

Node.js version: v12.22.12

npm version: 6.14.16

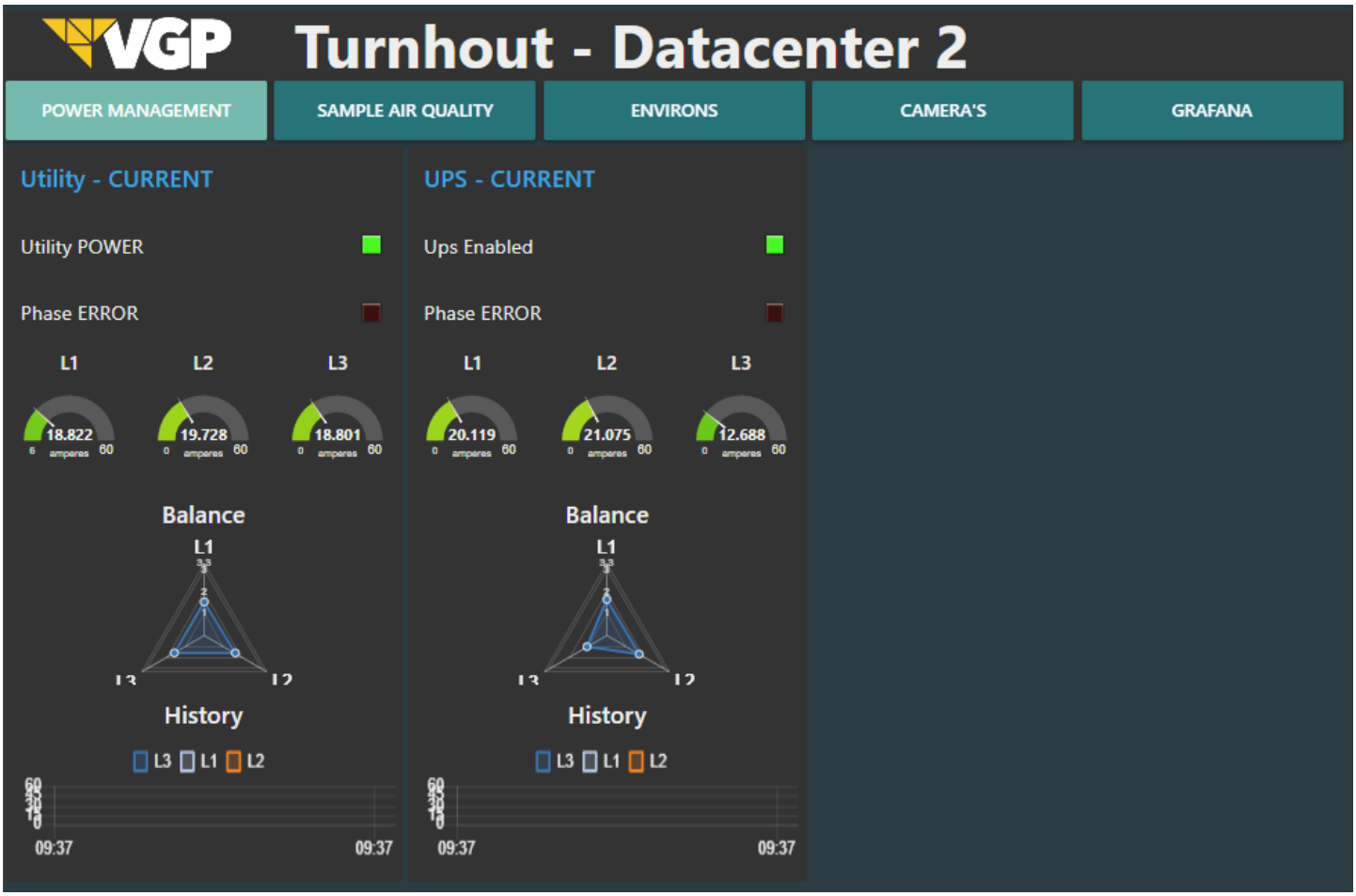
Power management

Utility current

The power that is currently being used by the servers.

UPS (Uninterruptable Power Supply) current

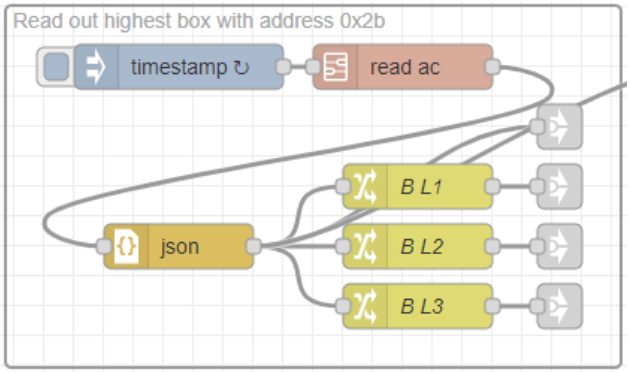
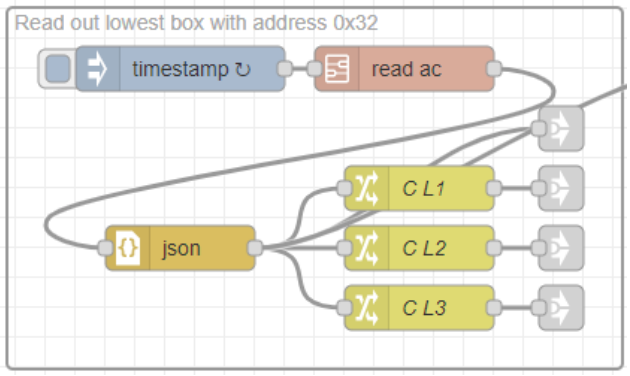
An **uninterruptible power supply** (UPS), also known as a battery backup, provides backup power when your regular power source fails or voltage drops to an unacceptable level. An **UPS** allows for the safe, orderly shutdown of a computer and connected equipment. The size and design of a UPS determines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



Flows

AC read

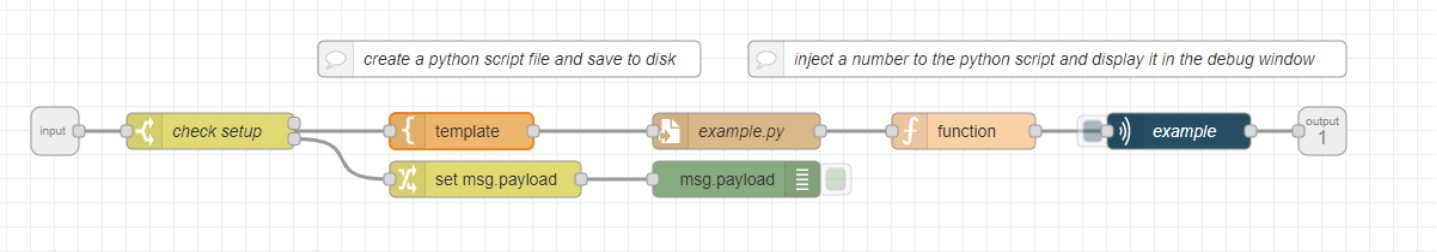
This flow reads the active current that is flowing in the cables of the different sources with **I2C**.

Flow	Power source	I2C address (hex)	I2C address (dec)
	Utility current	0x2b	43
	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

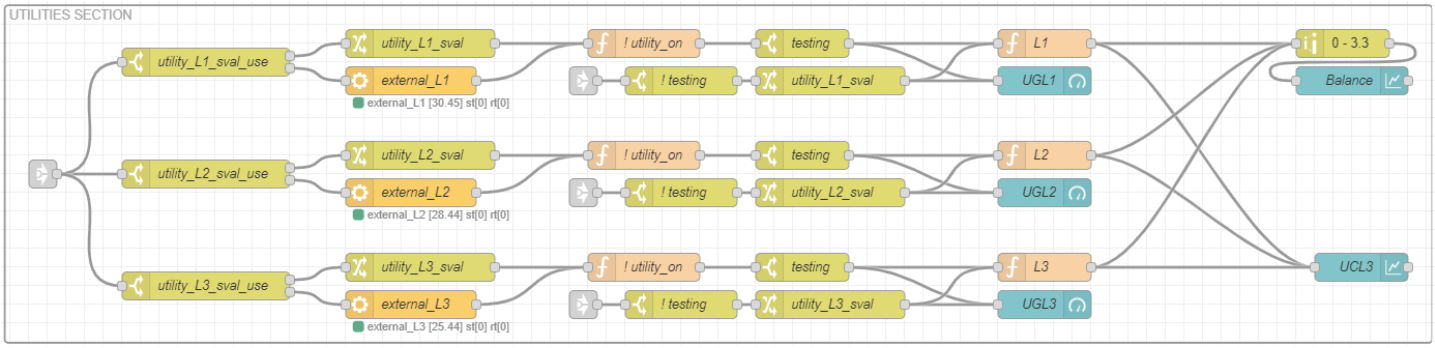
Modified code

Copy Code

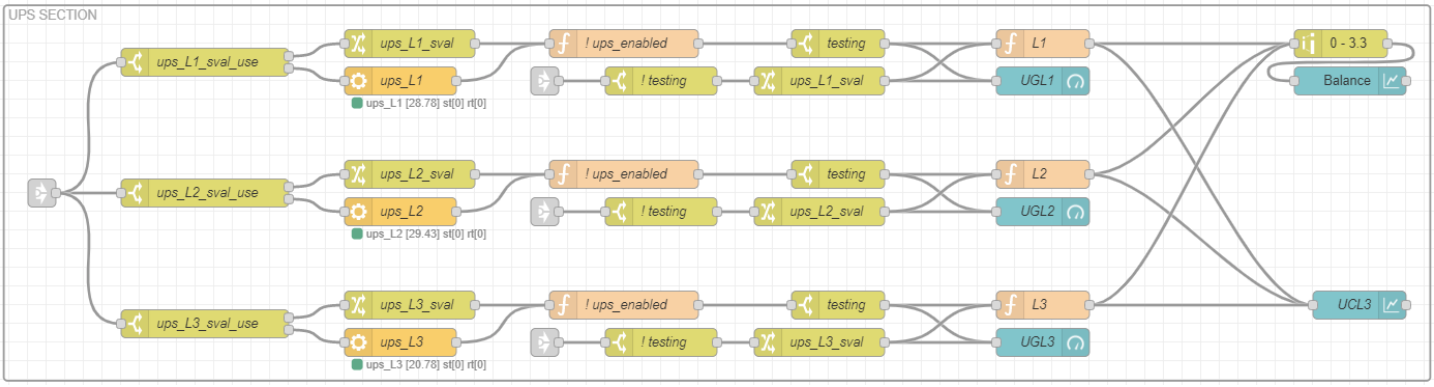
```
# Distributed with a free-will license.
# Use it any way you want, profit or free, provided it fits in the licenses of its associated
works.
# PECMAC125A
# This code is designed to work with the PECMAC125A_DLCT03C20 I2C Mini Module available from
ControlEverything.com.
# https://www.controleverything.com/content/Current?sku=PECMAC125A_DLCT03C20#tabs-0-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



UPS



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.

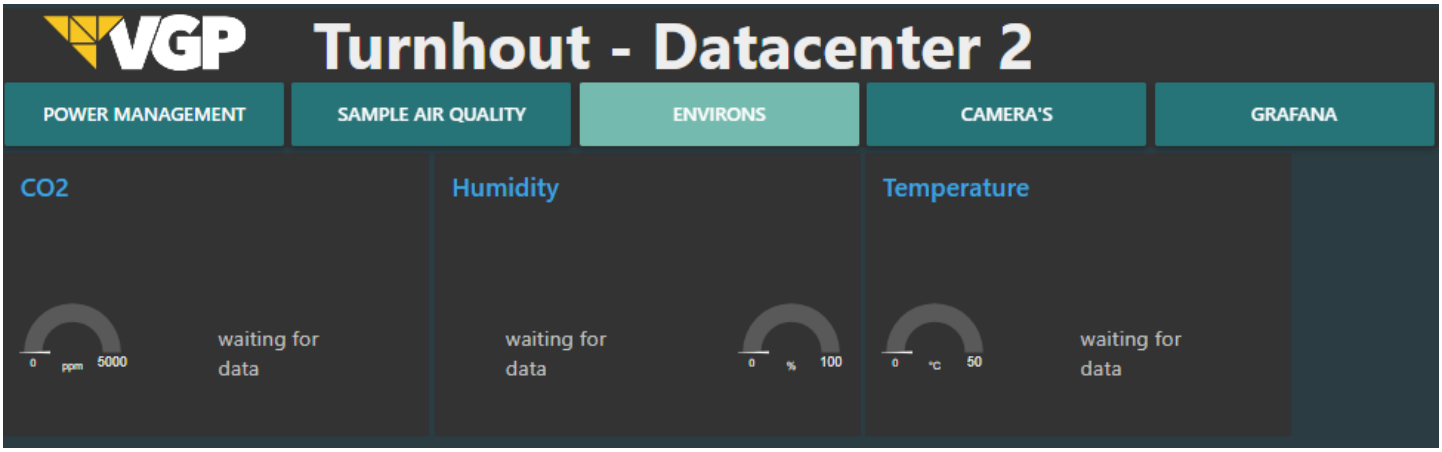
Temperature in this room should remain around 15 to 24 degrees **Celsius**.

Humidity

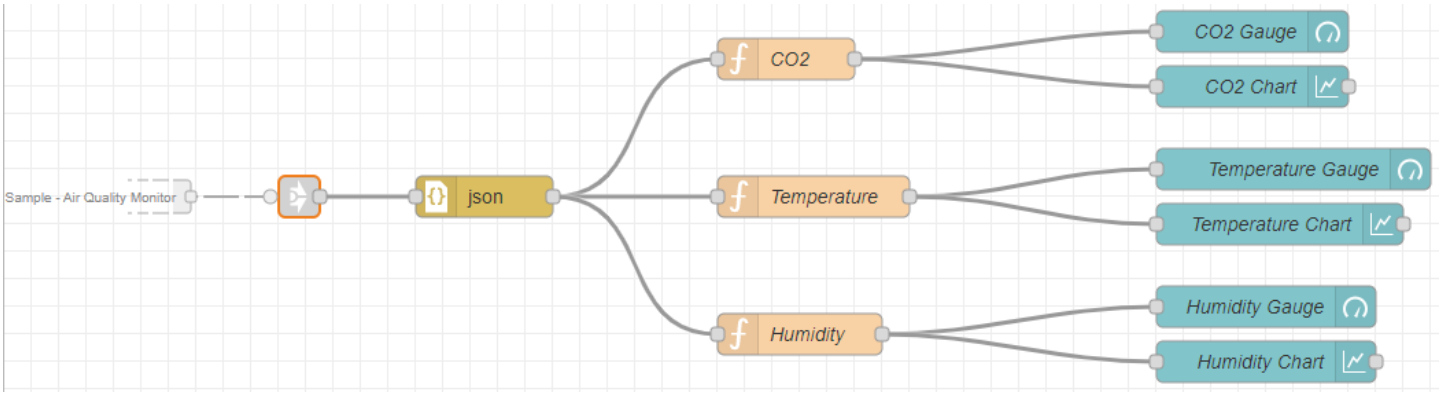
Humidity is **the amount of water vapor in the air**. If there is a lot of water vapor in the air, the humidity will be high.

Although you can't see it, it's still there. The ideal relative humidity for health and comfort is somewhere between 30-50% humidity. This means that the air holds between 30-50% of the maximum amount of moisture it can contain.

User interface



Flow



Sample 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:

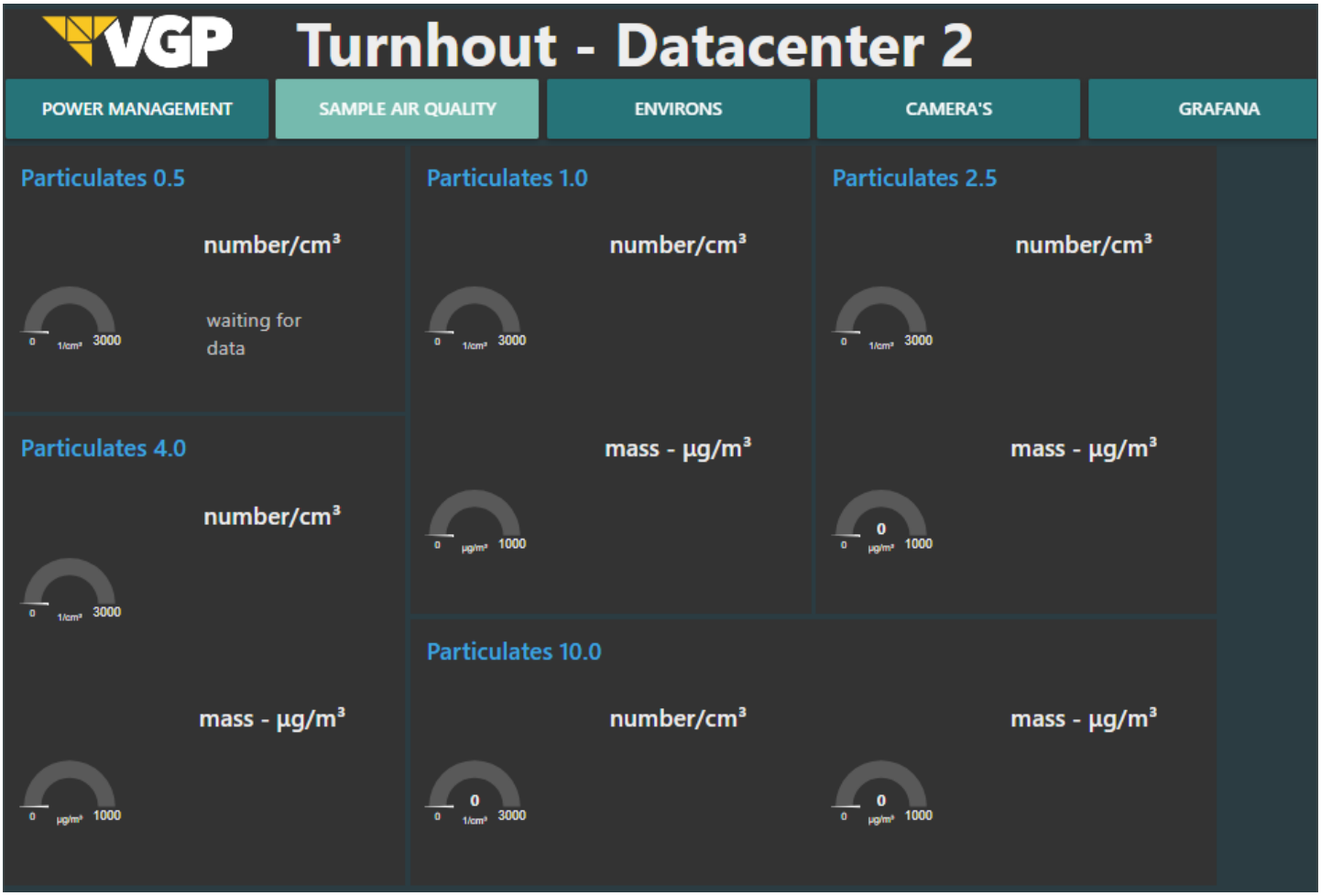
- Dust
- Dirt
- Soot
- Smoke
- Drops of liquid

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.

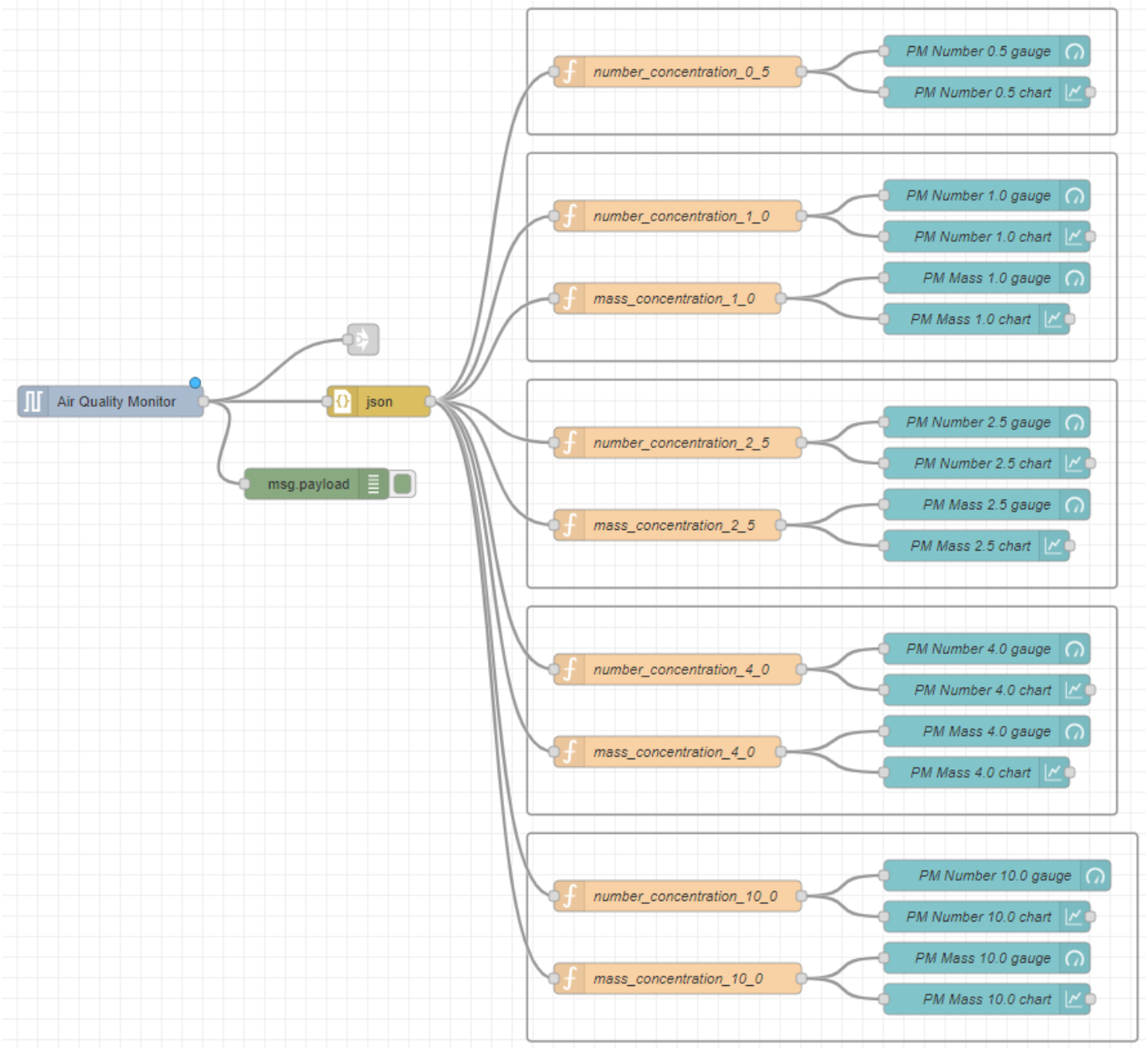
This **Particulate Matter Sensor** is designed to measure **PM0.5, PM1.0, PM2.5, PM4.0 and PM10.0**, providing a particle count for each size indicated.

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



Flow



NCD sensors

Mac address	Description	Type	Sensor readings
00:13:a2:00:41:d7:ad:34	CO2 Temperature Humidity PM Sensor	53	CO2, Humidity, Celcius, PM
00:13:a2:00:41:b5:e1:24	Wireless USB Modem		

Wireless air quality sensor.

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

Cameras

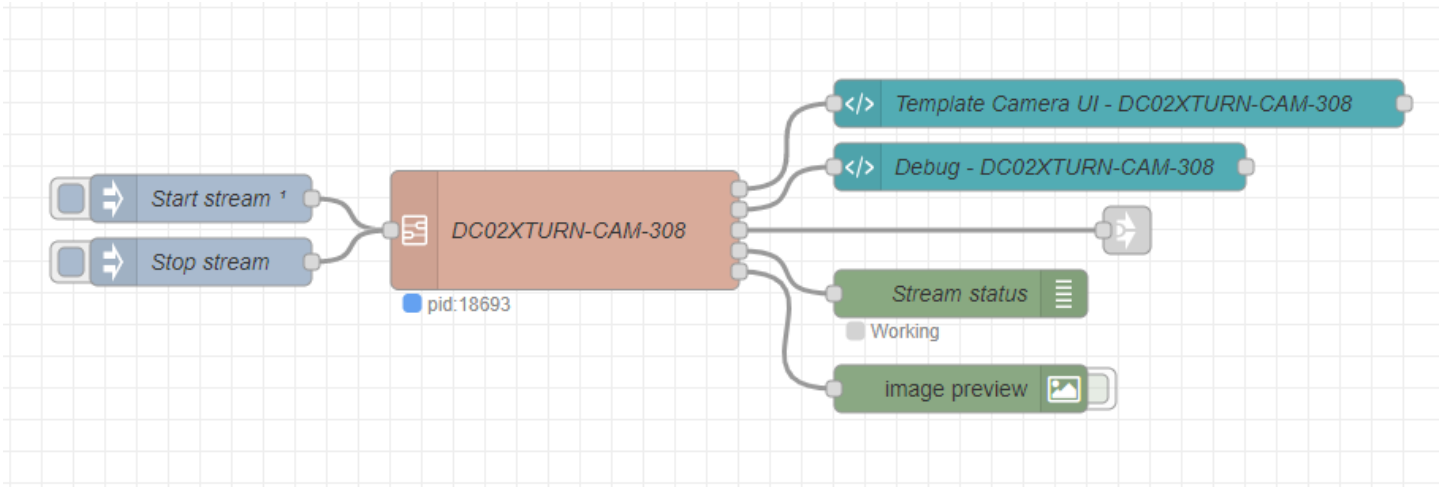
Datacenter 2 has **6 UniFi** cameras. These are installed in case of a physical data breach. If the background is **green**, it means the camera is turned on and showing live feed.

Camera configuration: Installing a Unifi Camera with Node-Red (http://wiki01.prd.priv.vangenechten.com/mediawiki/index.php/Node-Red%20Live%20camera)

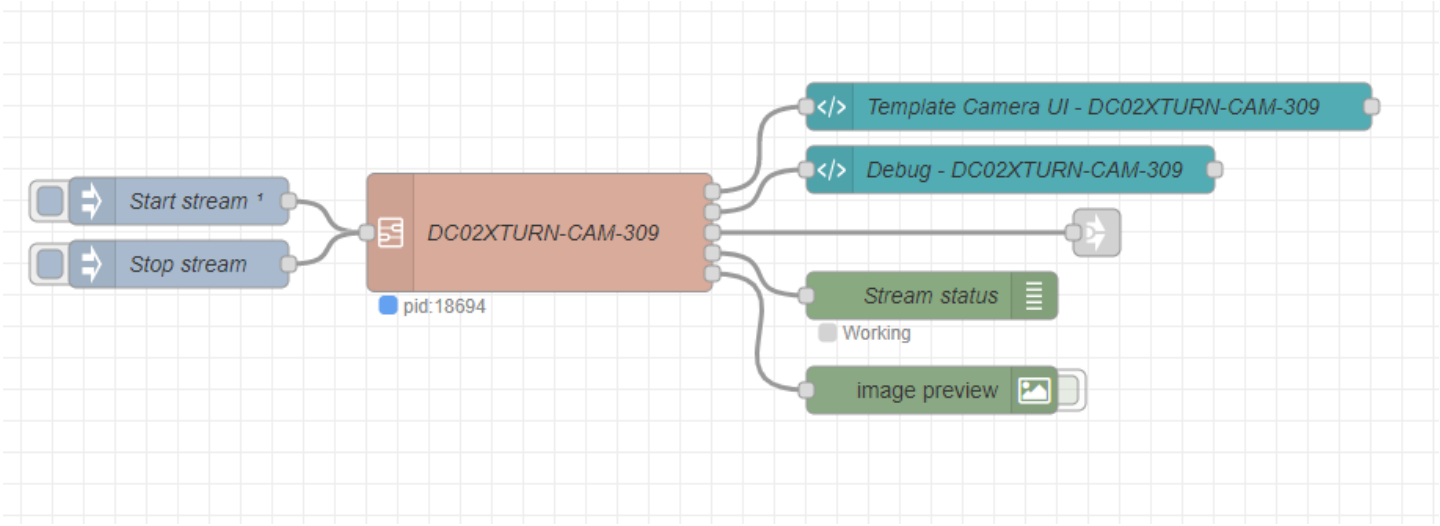
Camera	RTSP_URL
CAM-308	rtsp://10.1.16.9:7447/61ea905fc2dcbcf8de67fdea_2
CAM-309	rtsp://10.1.16.9:7447/61ea9023c2dcbcf8de67fde8_2
CAM-310	rtsp://10.1.16.9:7447/61ea8fdec2dcbcf8de67fde6_2
CAM-311	rtsp://10.1.16.9:7447/61ea8fa0c2dcbcf8de67fde4_2
CAM-313	rtsp://10.1.16.9:7447/61ea8f2bc2dcbcf8de67fde0_2
CAM-314	rtsp://10.1.16.9:7447/61ea8f6fc2dcbcf8de67fde2_2

Flows

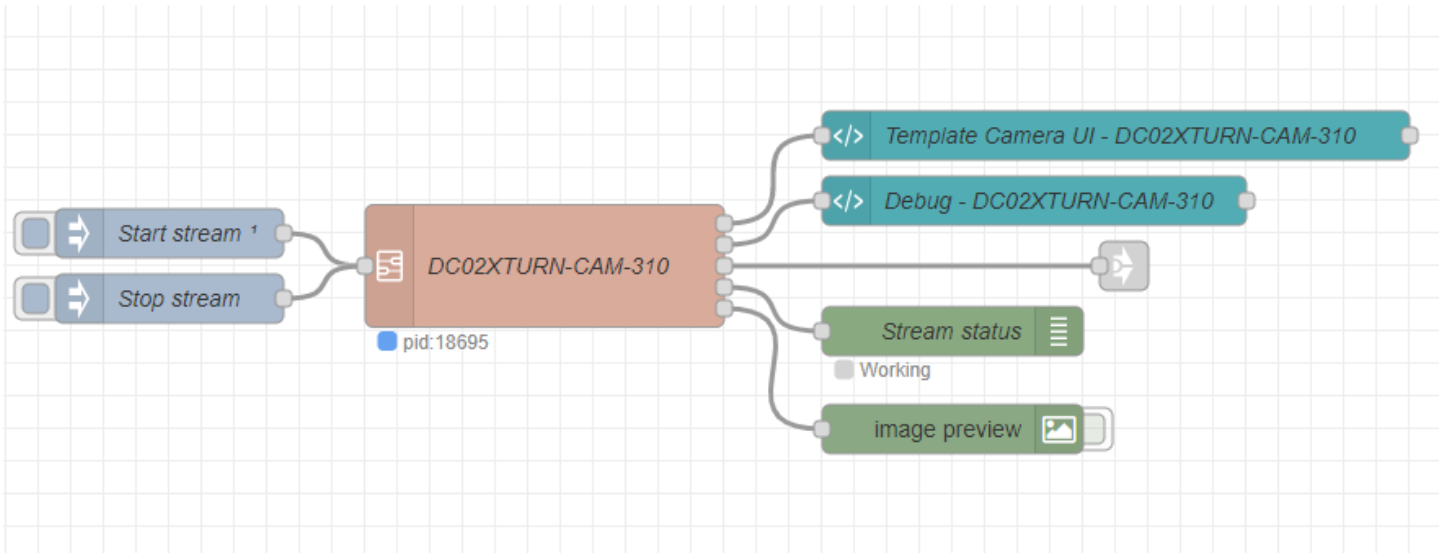
CAM-308



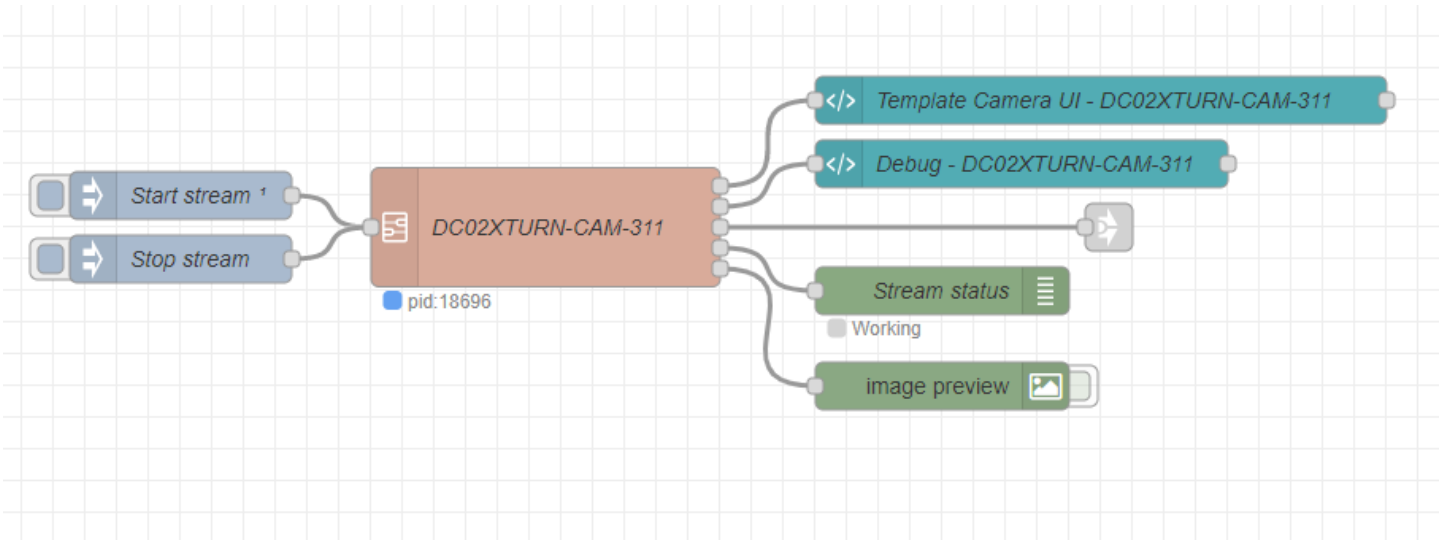
CAM-309



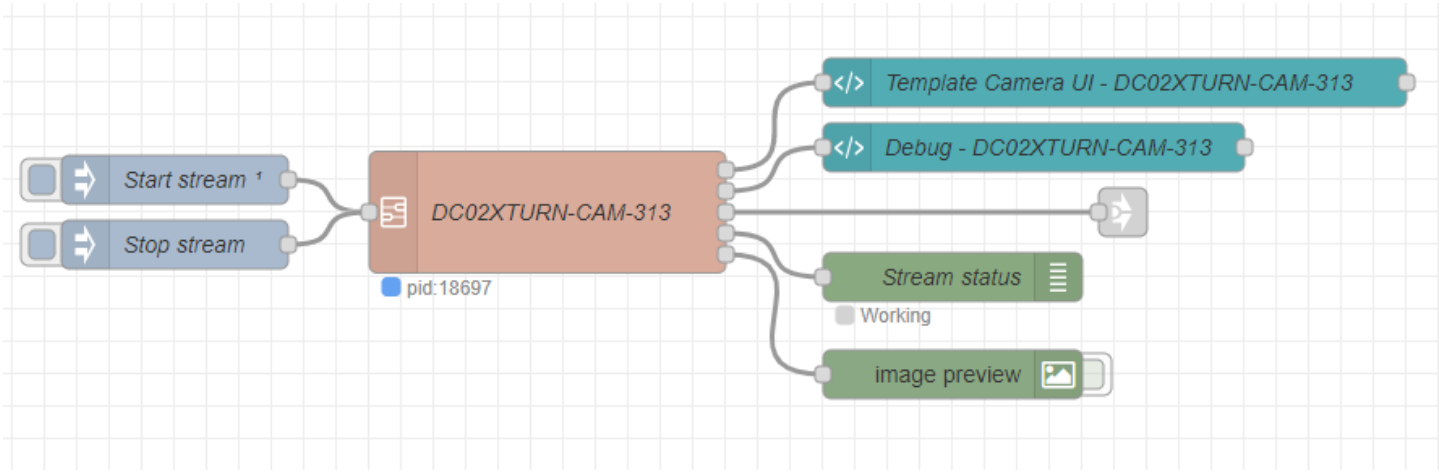
CAM-310



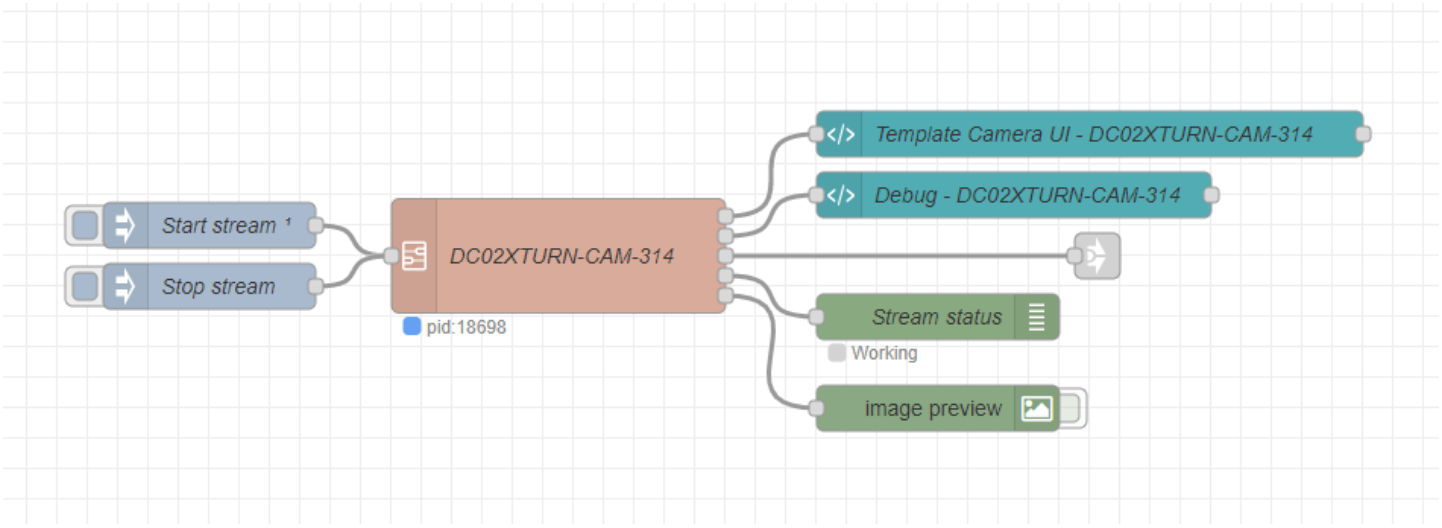
CAM-311



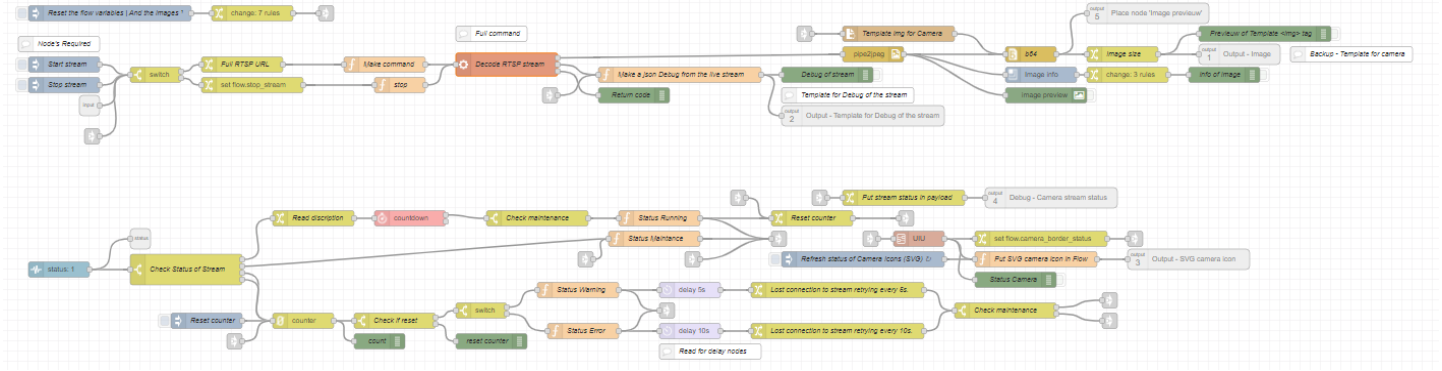
CAM-313



CAM-314



Live - IP Camera subflow



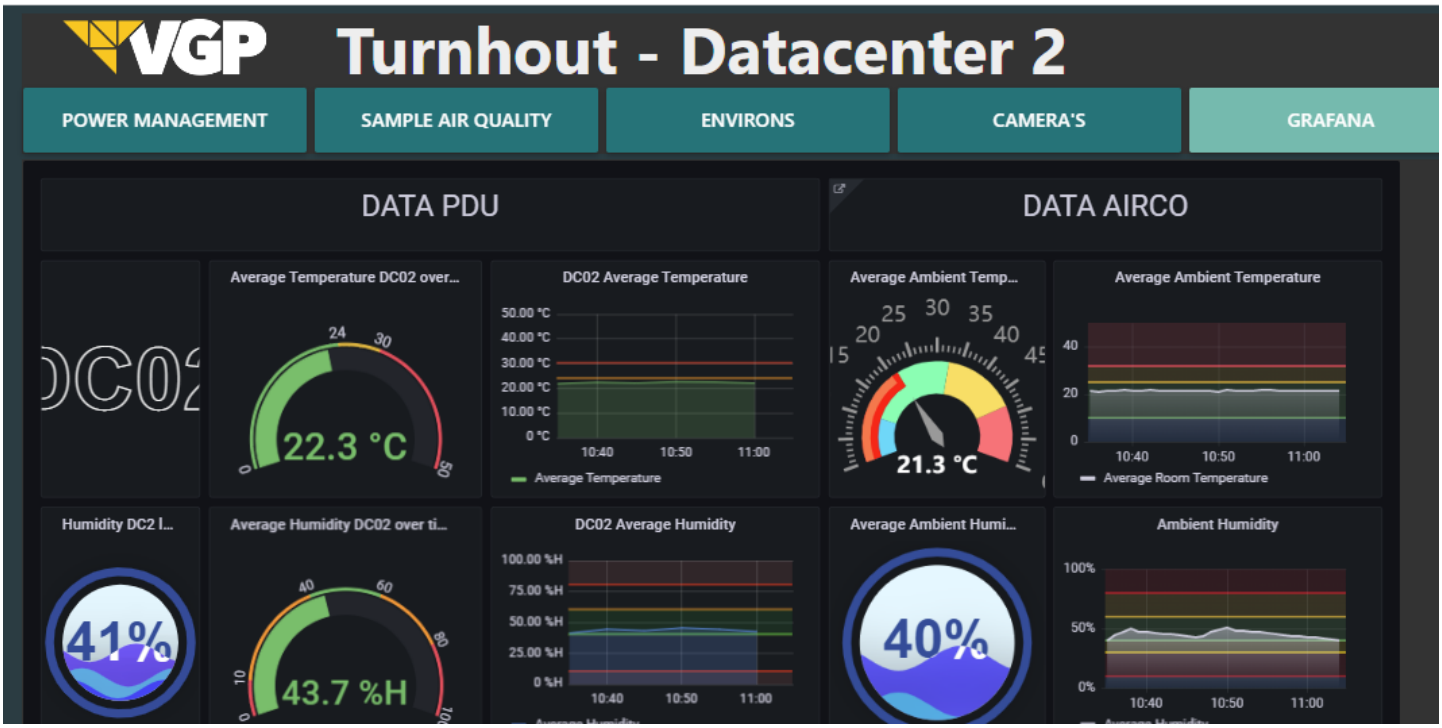
Grafana

https://grafana.com/

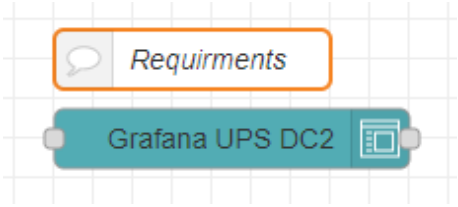
Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data-driven culture

This grafana dashboard shows more details about **DC2**.

Link: https://observer0302.priv.vangenechten.com:4443/d/yCdMq4J7z/turnhout_dc02_termohydrograph?orgId=1&refresh=1m&kiosk



Flow

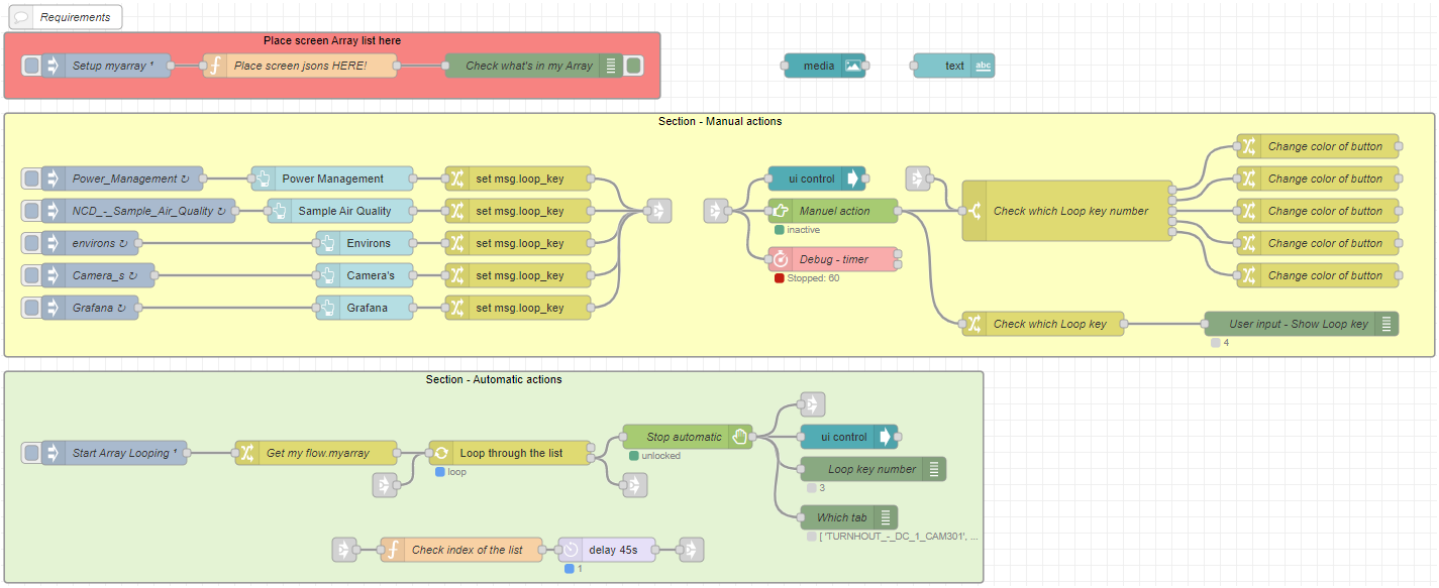


AutoSwitch - Dashboard Tabs

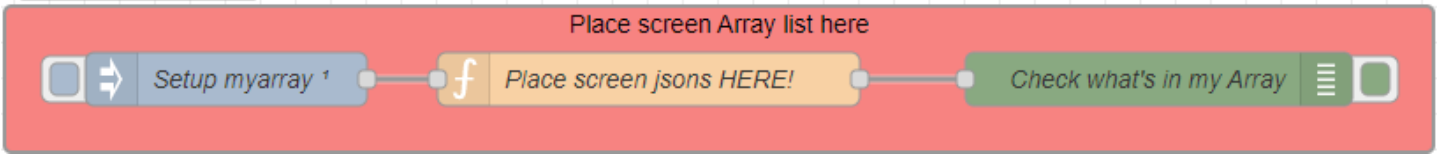
Main flow

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_2_Power_-_STATUS",
            "TURNHOUT_-_DC_2_Utility_-_CURRENT",
            "TURNHOUT_-_DC_2_Generator_-_CURRENT",
            "TURNHOUT_-_DC_2_UPS_-_CURRENT",
            "TURNHOUT_-_DC_2_Extra_-_CURRENT"
        ],
        "hide": [
            "TURNHOUT_-_DC_2_Particulates_0.5",
            "TURNHOUT_-_DC_2_Particulates_1.0",
            "TURNHOUT_-_DC_2_Particulates_2.5",
            "TURNHOUT_-_DC_2_Particulates_4.0",
```

Copy Code

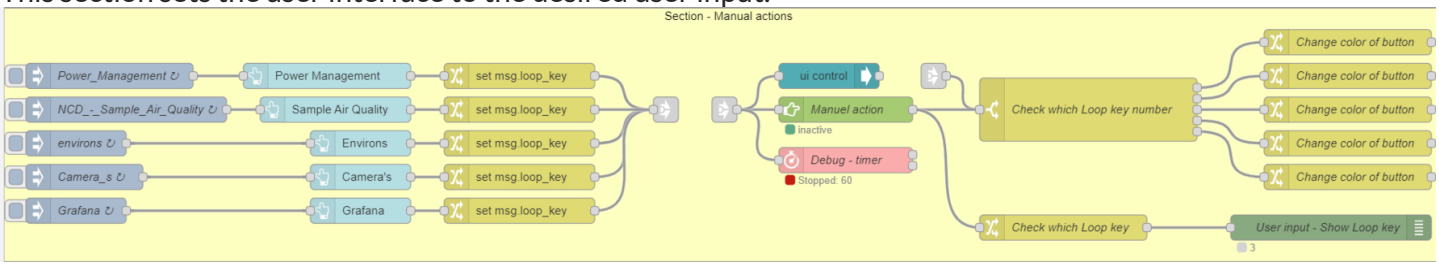
```
            "TURNHOUT_-_DC_2_Particulates_10.0",

            "TURNHOUT_-_DC_2_CO2",
            "TURNHOUT_-_DC_2_Temperature",
            "TURNHOUT_-_DC_2_Humidity",

            "TURNHOUT_-_DC_2_CAM308",
            "TURNHOUT_-_DC_2_CAM309",
            "TURNHOUT_-_DC_2_CAM310",
            "TURNHOUT_-_DC_2_CAM311",
            "TURNHOUT_-_DC_2_CAM313",
            "TURNHOUT_-_DC_2_CAM314",

            "TURNHOUT_-_DC_2_Grafana"

        ]
    }
}


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

            "TURNHOUT_-_DC_2_Particulates_0.5",
            "TURNHOUT_-_DC_2_Particulates_1.0",
            "TURNHOUT_-_DC_2_Particulates_2.5",
            "TURNHOUT_-_DC_2_Particulates_4.0",
            "TURNHOUT_-_DC_2_Particulates_10.0",

        ],
        "hide": [
            "TURNHOUT_-_DC_2_Power_-_STATUS",
            "TURNHOUT_-_DC_2_Utility_-_CURRENT",
            "TURNHOUT_-_DC_2_Generator_-_CURRENT",
            "TURNHOUT_-_DC_2_UPS_-_CURRENT",
            "TURNHOUT_-_DC_2_Extra_-_CURRENT",

            "TURNHOUT_-_DC_2_CO2",
            "TURNHOUT_-_DC_2_Temperature",
            "TURNHOUT_-_DC_2_Humidity",

            "TURNHOUT_-_DC_2_CAM308",
            "TURNHOUT_-_DC_2_CAM309",
            "TURNHOUT_-_DC_2_CAM310",
            "TURNHOUT_-_DC_2_CAM311",
            "TURNHOUT_-_DC_2_CAM313",
            "TURNHOUT_-_DC_2_CAM314",

            "TURNHOUT_-_DC_2_Grafana"
        ]
    }
}

var src3 = {
    "group": {
        "show": [
            "TURNHOUT_-_DC_2_CO2",
            "TURNHOUT_-_DC_2_Temperature",
            "TURNHOUT_-_DC_2_Humidity"
        ],
        "hide": [
            "TURNHOUT_-_DC_2_Power_-_STATUS",
            "TURNHOUT_-_DC_2_Utility_-_CURRENT",
```

```
            "TURNHOUT_-_DC_2_Generator_-_CURRENT",
            "TURNHOUT_-_DC_2_UPS_-_CURRENT",
            "TURNHOUT_-_DC_2_Extra_-_CURRENT",
            "TURNHOUT_-_DC_2_Particulates_0.5",
            "TURNHOUT_-_DC_2_Particulates_1.0",
            "TURNHOUT_-_DC_2_Particulates_2.5",
            "TURNHOUT_-_DC_2_Particulates_4.0",
            "TURNHOUT_-_DC_2_Particulates_10.0",

            "TURNHOUT_-_DC_2_CAM308",
            "TURNHOUT_-_DC_2_CAM309",
            "TURNHOUT_-_DC_2_CAM310",
            "TURNHOUT_-_DC_2_CAM311",
            "TURNHOUT_-_DC_2_CAM313",
            "TURNHOUT_-_DC_2_CAM314",

            "TURNHOUT_-_DC_2_Grafana"
        ]
    }
}

var src4 = {
    "group": {
        "show": [
            "TURNHOUT_-_DC_2_CAM308",
            "TURNHOUT_-_DC_2_CAM309",
            "TURNHOUT_-_DC_2_CAM310",
            "TURNHOUT_-_DC_2_CAM311",
            "TURNHOUT_-_DC_2_CAM313",
            "TURNHOUT_-_DC_2_CAM314"
        ],
        "hide": [
            "TURNHOUT_-_DC_2_Power_-_STATUS",
            "TURNHOUT_-_DC_2_Utility_-_CURRENT",
            "TURNHOUT_-_DC_2_Generator_-_CURRENT",
            "TURNHOUT_-_DC_2_UPS_-_CURRENT",
            "TURNHOUT_-_DC_2_Extra_-_CURRENT",


            "TURNHOUT_-_DC_2_Particulates_0.5",
            "TURNHOUT_-_DC_2_Particulates_1.0",
            "TURNHOUT_-_DC_2_Particulates_2.5",
            "TURNHOUT_-_DC_2_Particulates_4.0",
            "TURNHOUT_-_DC_2_Particulates_10.0",

            "TURNHOUT_-_DC_2_CO2",
            "TURNHOUT_-_DC_2_Temperature",
            "TURNHOUT_-_DC_2_Humidity",

            "TURNHOUT_-_DC_2_Grafana"
        ]
    }
}


var src5 = {
    "group": {
        "show": [
            "TURNHOUT_-_DC_2_Grafana"
        ],
        "hide": [
            "TURNHOUT_-_DC_2_Power_-_STATUS",
            "TURNHOUT_-_DC_2_Utility_-_CURRENT",
            "TURNHOUT_-_DC_2_Generator_-_CURRENT",
```

```
            "TURNHOUT_-_DC_2_UPS_-_CURRENT",
            "TURNHOUT_-_DC_2_Extra_-_CURRENT",


            "TURNHOUT_-_DC_2_Particulates_0.5",
            "TURNHOUT_-_DC_2_Particulates_1.0",
            "TURNHOUT_-_DC_2_Particulates_2.5",
            "TURNHOUT_-_DC_2_Particulates_4.0",
            "TURNHOUT_-_DC_2_Particulates_10.0",

            "TURNHOUT_-_DC_2_CO2",
            "TURNHOUT_-_DC_2_Temperature",
            "TURNHOUT_-_DC_2_Humidity",

            "TURNHOUT_-_DC_2_CAM308",
            "TURNHOUT_-_DC_2_CAM309",
            "TURNHOUT_-_DC_2_CAM310",
            "TURNHOUT_-_DC_2_CAM311",
            "TURNHOUT_-_DC_2_CAM313",
            "TURNHOUT_-_DC_2_CAM314"

        ]
    }
}


// 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;
```

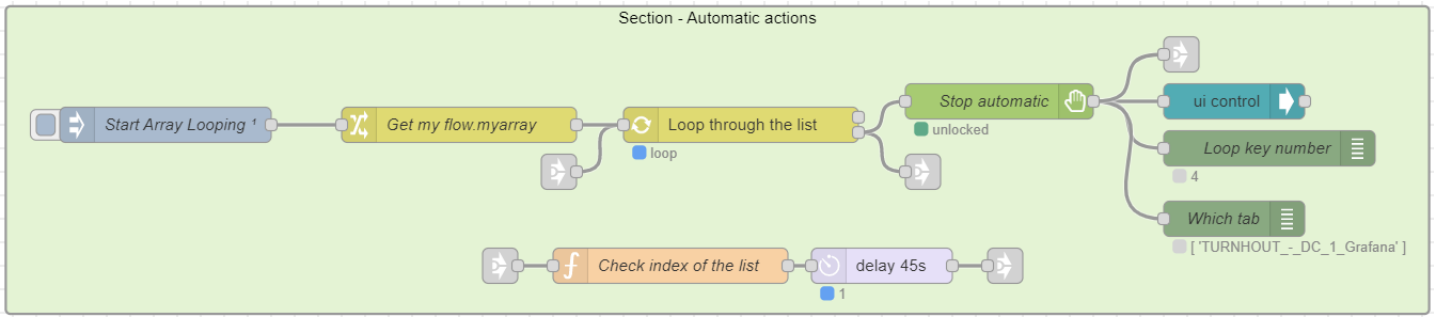
Manual actions

This section sets the user interface to the desired user input.



Automatic actions

This sections becomes active by default. It cycles through the input array list every 45 seconds.



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

Status Open Priority B SMTTemplate SMTProject SMT Project Page VGPIoT DCMonitoring