NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Graduate School of Business

Khabibulina Eleonora, Tonoyan Seda, Piskaeva Zlata, Panasenkov Vladimir

Report for 3-rd Option project
Manufacturing data Collection and Analysis

Degree program: Business Analytics and Big Data Systems

Moscow 2023

CONTENTS

Moscow 2023

1 VISUALISATION

1.1 Create web-page with test OPC UA server (any 3 dynamic parameters). Use node-red or create your own application. There should be a separate page for OPC UA. Try to make a nice and clear interface (Seda, Zlata, Vladimir)

In the Node red we visualized streaming data from OPC UA and MQTT. To visualize data from OPC UA the following charts were done:

1. To connect to OPC UA, UA expert was downloaded. The connection to the connection to the opc.tcp://opcua.demo-this.com:51210/UA/SampleServer server. From the dynamic folder 2 groups of data were used for visualization:
    a. AnalogArray
        i. Float
        ii. Double
        iii. Byte
        iv. Numerical
    b. AnalogScalar
        i. Float
        ii. Double
        iii. Numerical

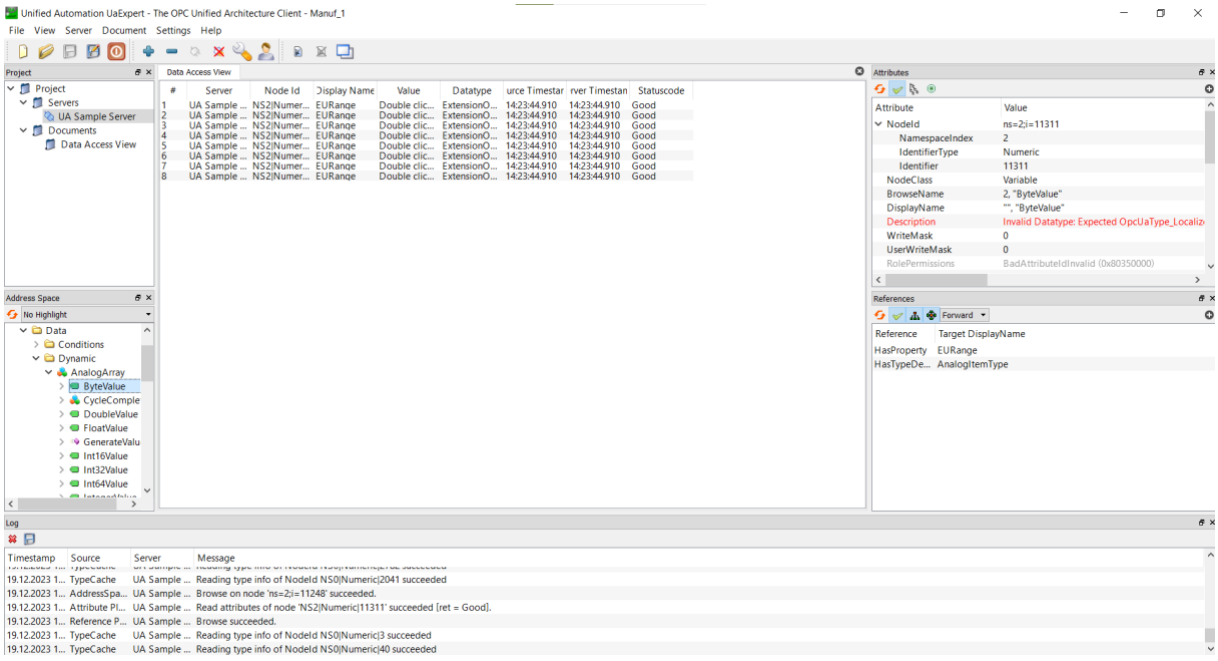


Figure 1. UA Expert setting (AnalogArray Bite attributes)

2. After setting the UA Expert, flows in Node-red were created:
    a. Inject with a 1 second cycle;
    b. OPC_UA Item with the respective information on the data (in Figure 2 Value: Byte ItemID – “ns=2;i=11311”)
    c. OPC_UA Client with the server link

d. 2 types of visualization: gauge and chart located in the OPC tab

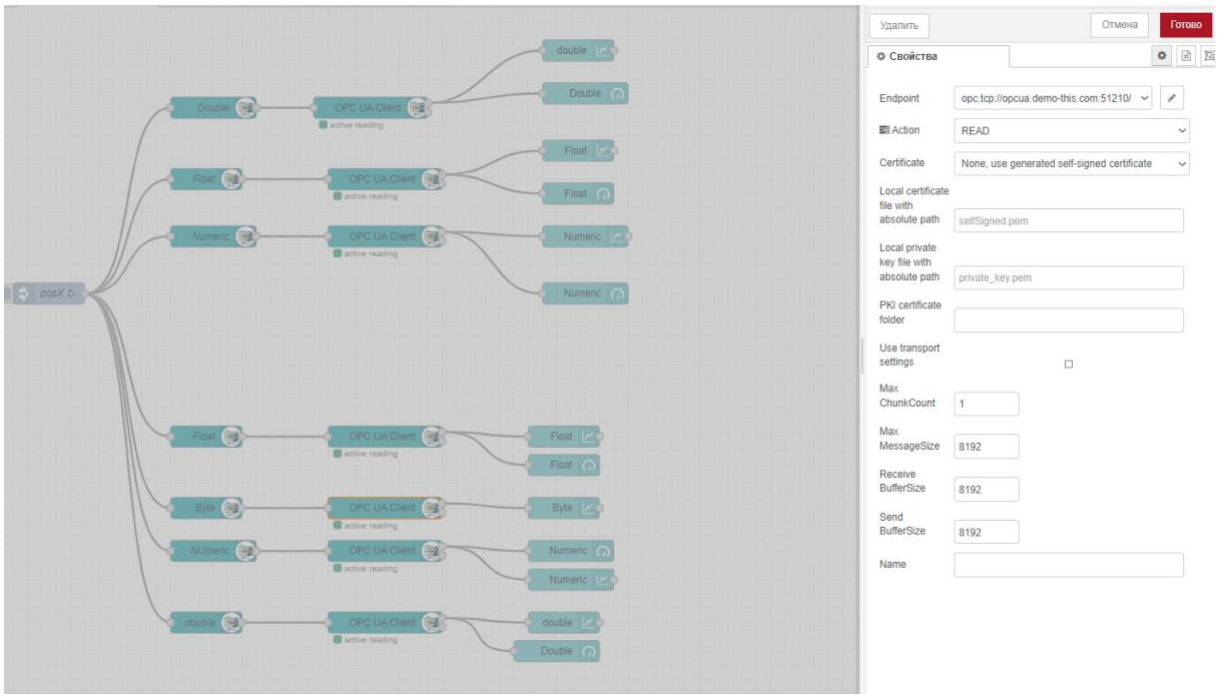


Figure 2. Node-red OPC_UA visualization flow

3. As a result, the OPC UA tab was created with the visualizations of the OPC UA data.

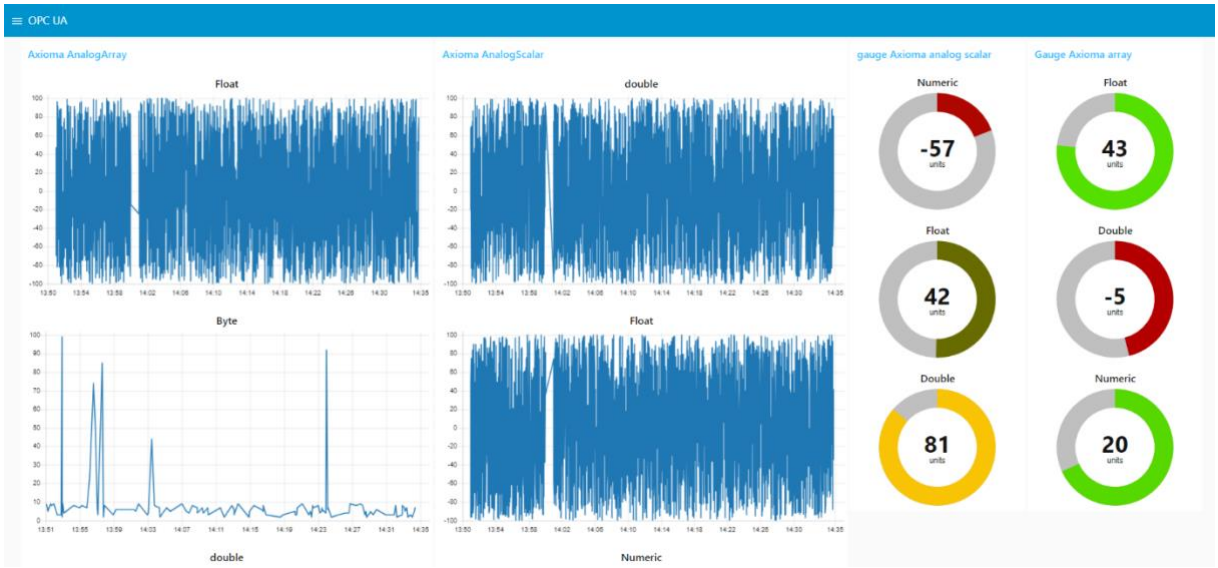


Figure 3. Node-red OPC_UA Tab

4. In the tab there are 4 groups of visualizations:
    a. Axioma AnalogArray,
        i. Float

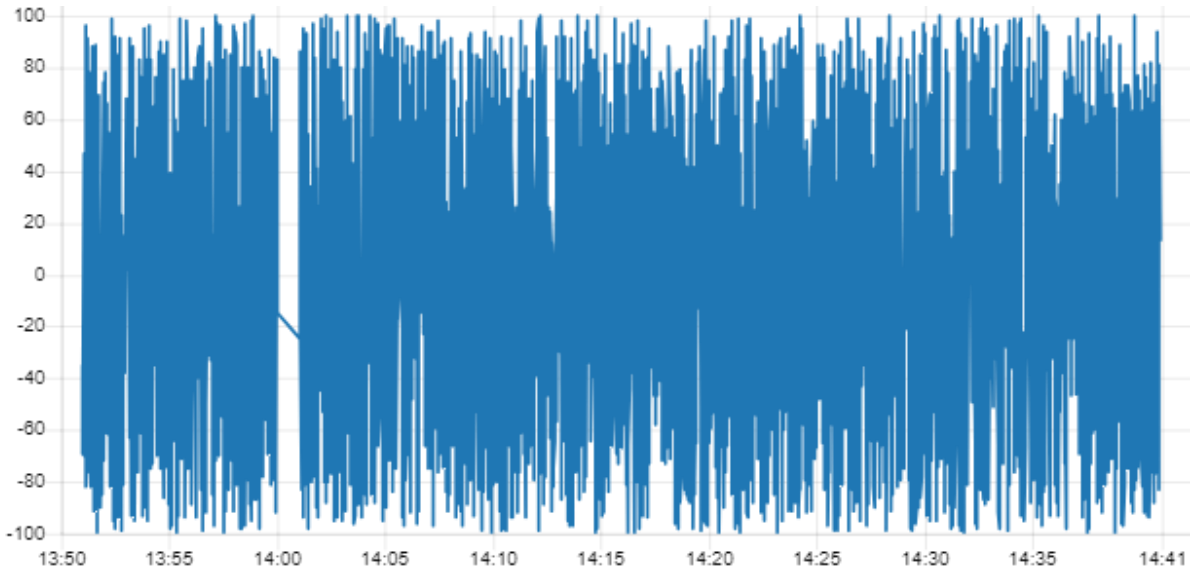


Figure 4. Axioma AnalogArray Float chart

ii. Byte

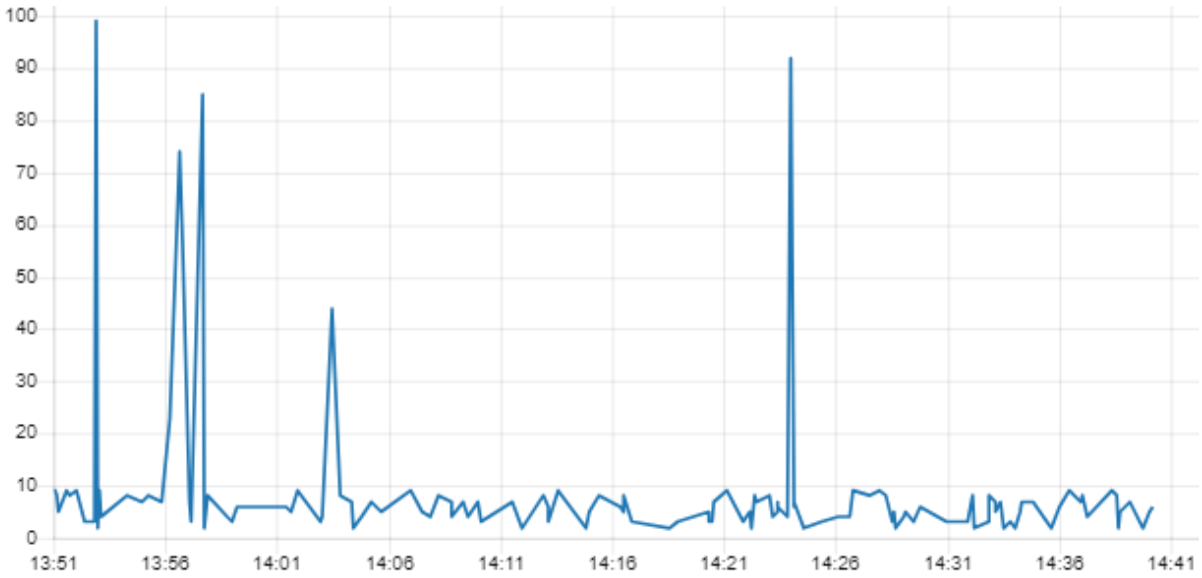


Figure 5. Axioma AnalogArray Byte chart

iii. Double

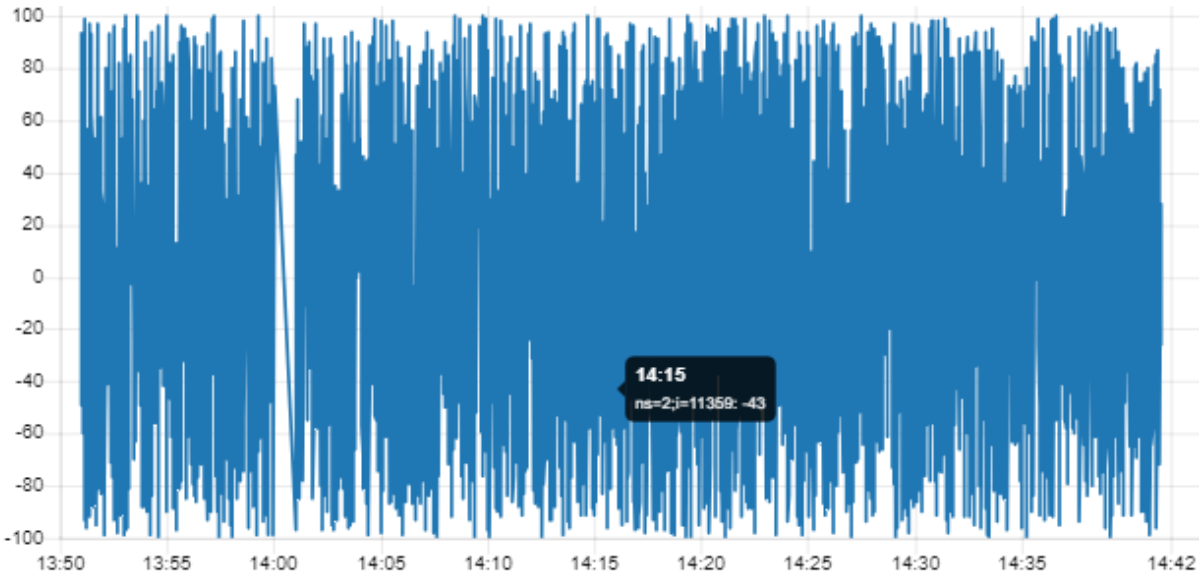


Figure 6. Axioma AnalogArray Double chart

iv. Numeric.

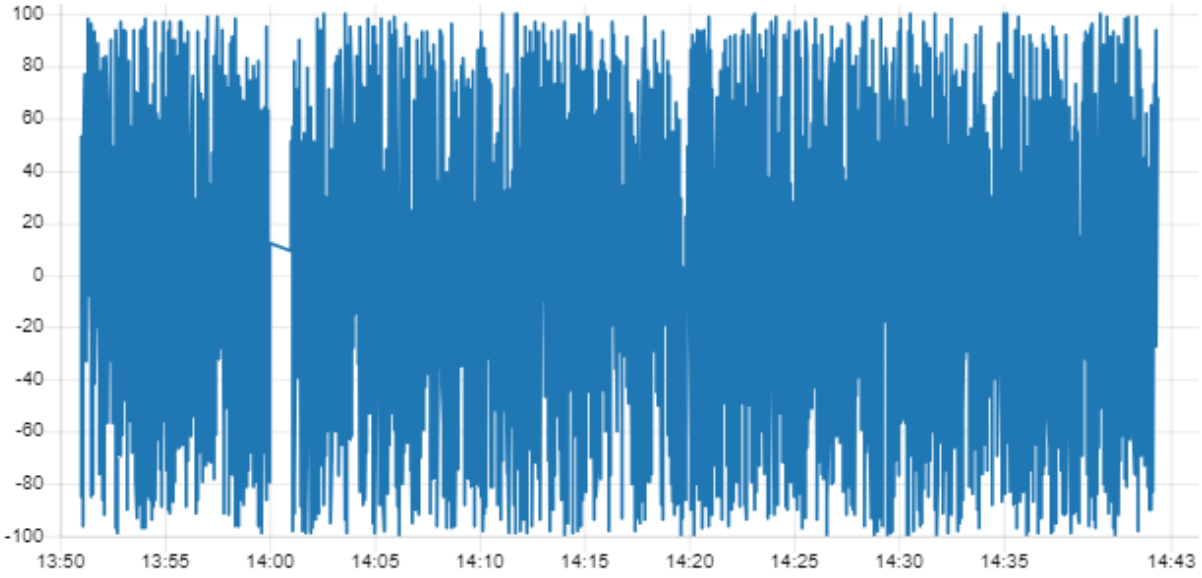


Figure 7. Axioma AnalogArray Numeric chart

b. Axioma AnalogScalar,

i. Float

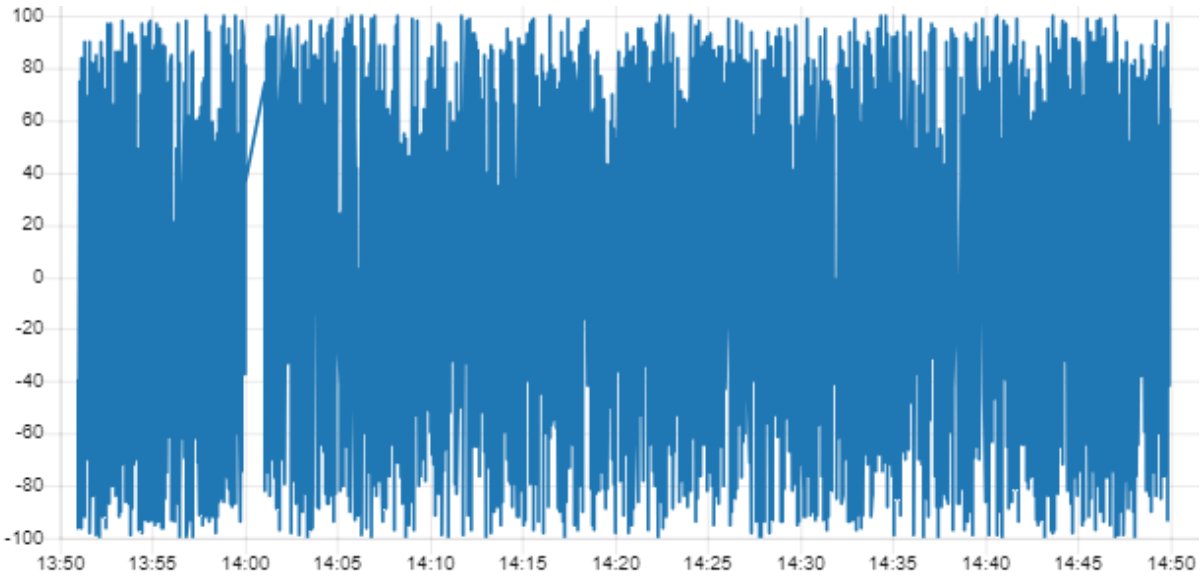


Figure 8. Axioma AnalogArray Float chart

ii. Numeric

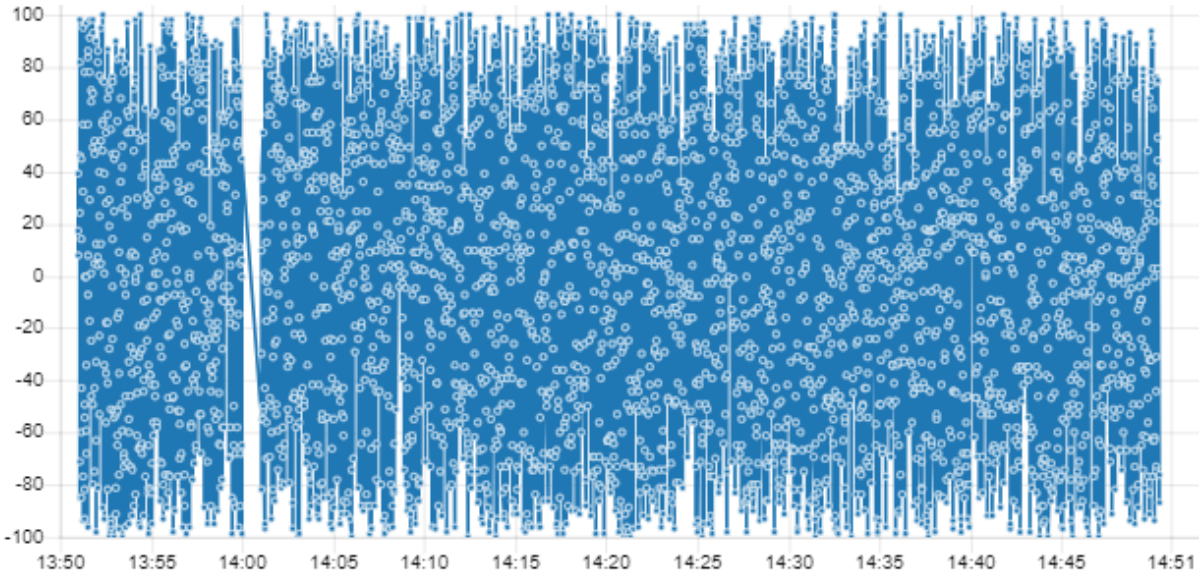


Figure 9. Axioma AnalogScalar Numeric chart

iii. Double

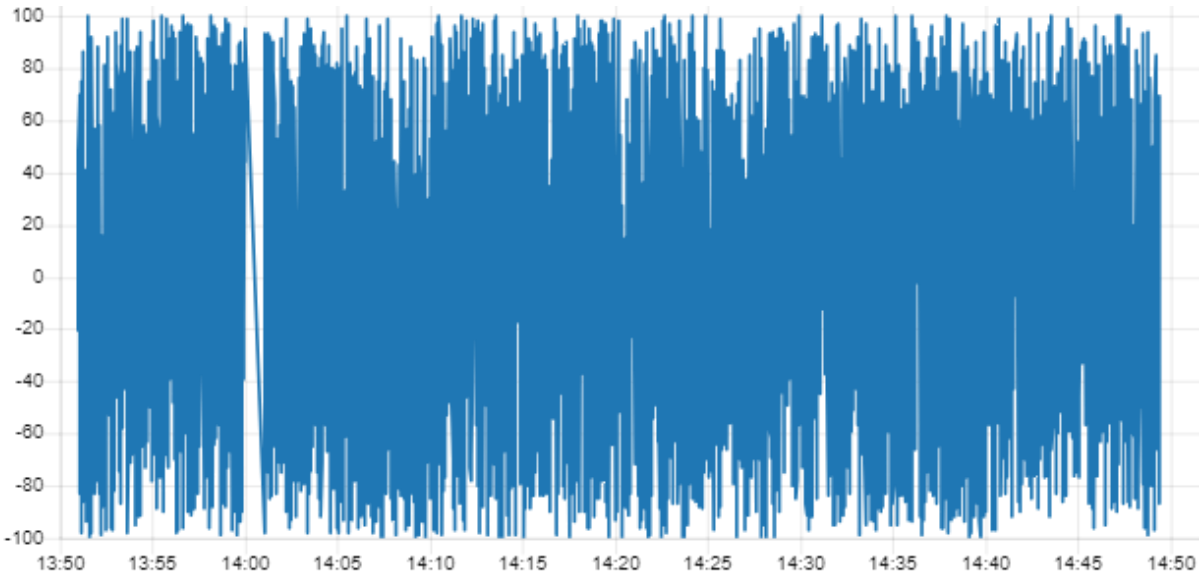


Figure 10. Axioma AnalogScalar Double chart

c. Gauge Axioma analog Scalar (Figure 11),

d. Gauge Axioma analog array (Figure 11).

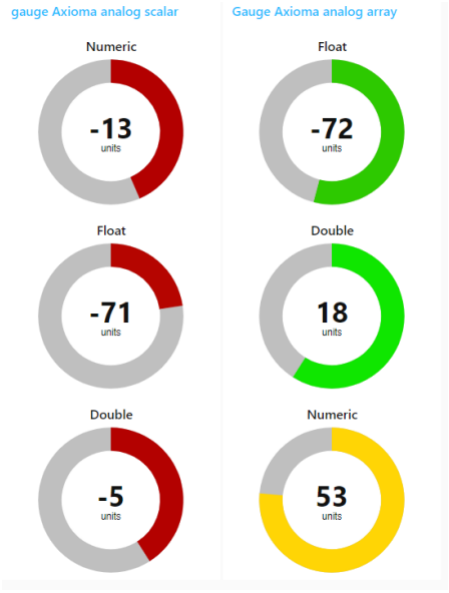


Figure 11. Axioma gauge groups

1.2 Create web-page in Node-Red (or visualization data, for example, python matplotlib or another) with data from real machine (Eleonora, Zlata, Seda, Vladimir)

1.2.1 Visualization in Python for CNC 3

In Python we visualized data from the CNC machines. To visualize data from CNC 3 the following plots were done:

1. Core Work Time Current (Время работы ядра (текущее)

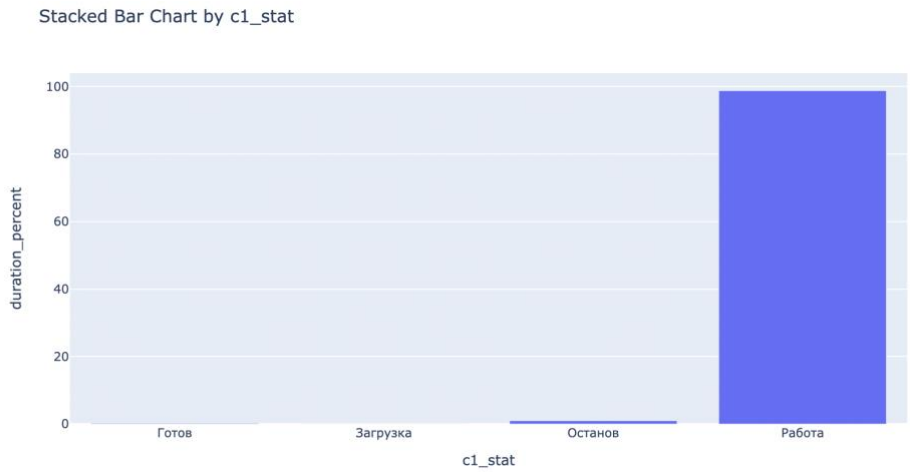
Data began to be recorded on December 13. The core worked stably on December 14, but on December 15 there was some " oddity" in work. Then the work resumed.

2. Channel Total Work Time (Время работы канала (всего))

Since the core was stagnant on December 15, the channel was also affected.

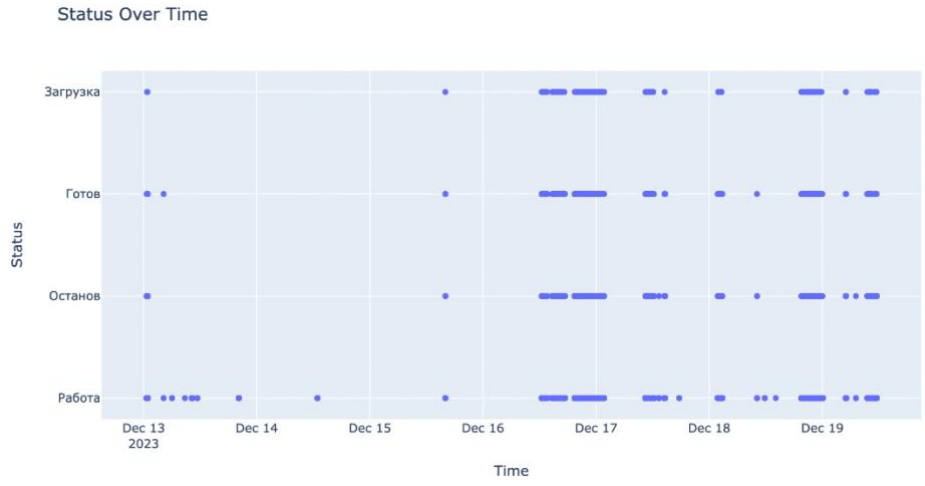
3. Status of channels (Статус канала)

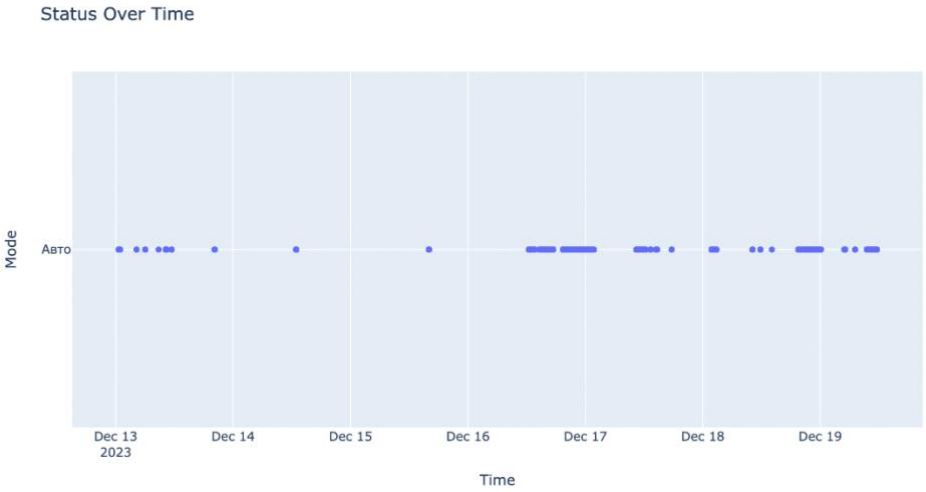
	c1_stat	duration
0	Готов	0 days 00:21:32.342000
1	Загрузка	0 days 00:04:55.860000
2	Останов	0 days 01:29:22.944000
3	Работа	6 days 08:58:17.308000



As can be seen from the table above, the channel was running most of the time (almost 6 days and 9 hours) and was left for about 1.5 hours. 21 minutes it was on standby and was in the process of loading for 5 minutes. This can also be seen in the graph below.

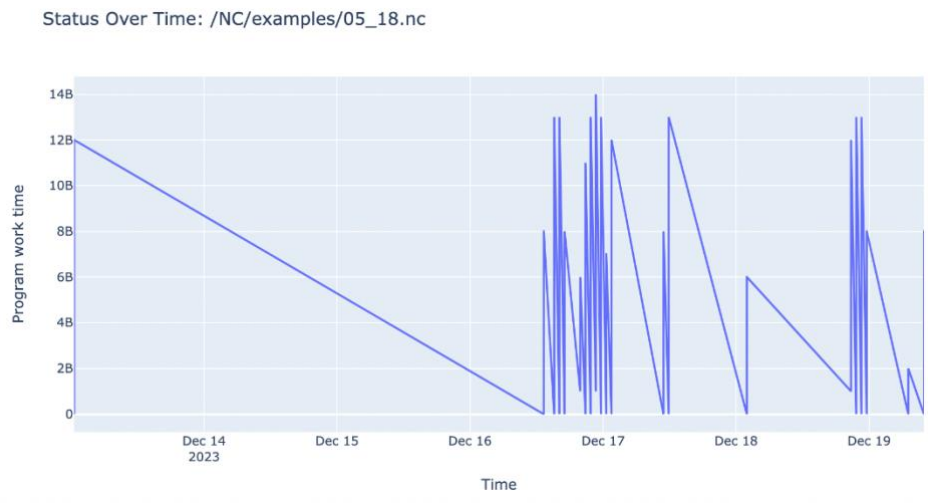
4. Channel mode (Режим канала)

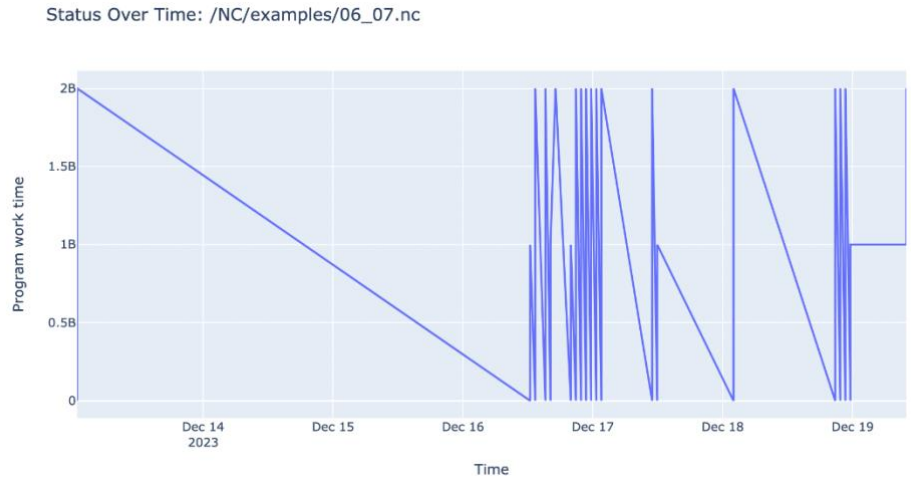


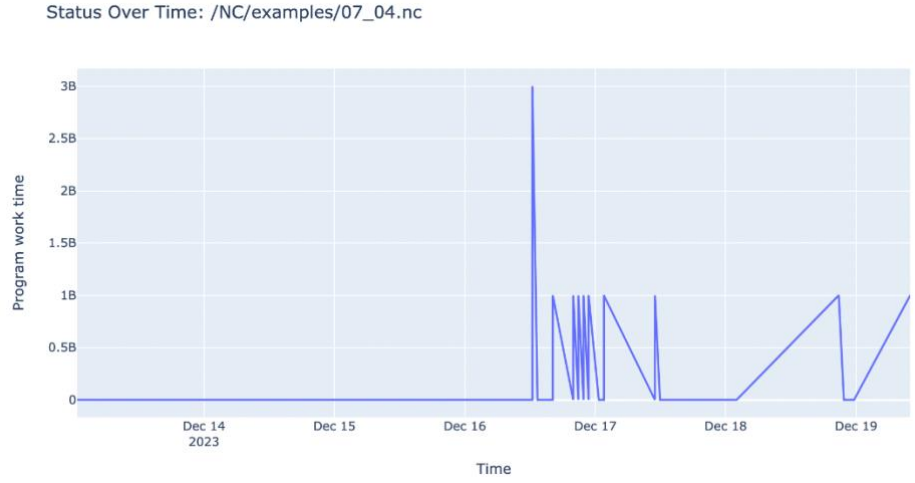


The channel was in auto mode the entire time and ran smoothly.

5. Work Time of Programm (Время работы программы)

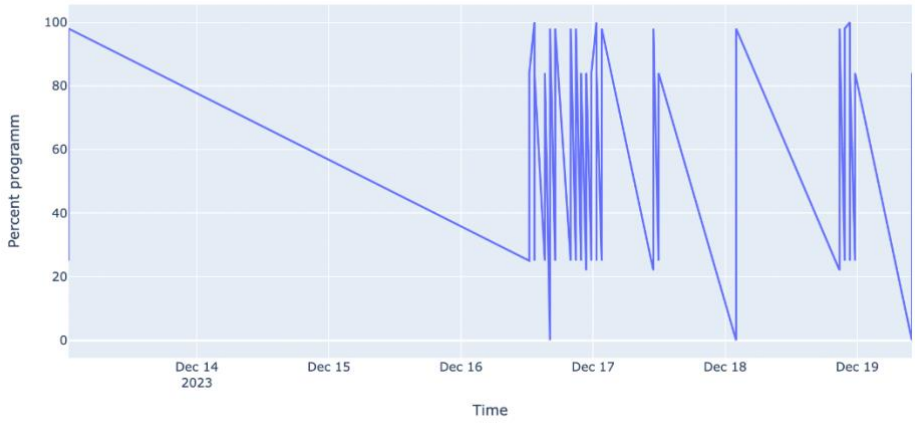




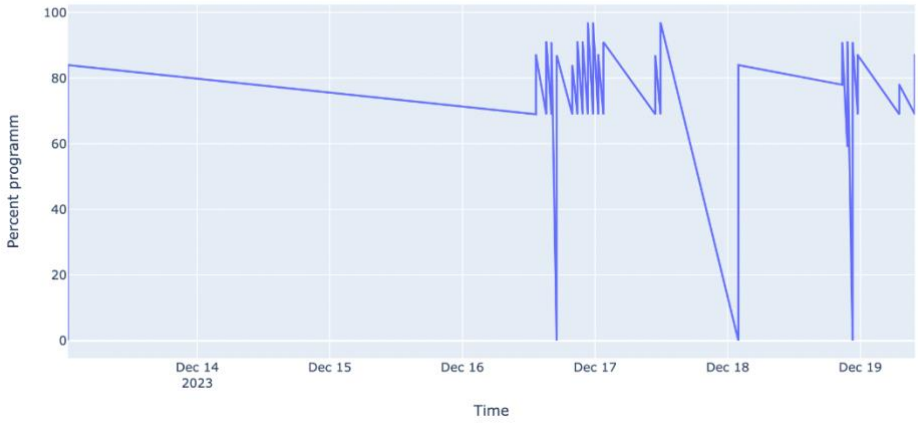


The programs worked mostly closer to December 17. Their activity was constantly changing, judging by the sharp changes. More graphs of program activity can be found in the Data Analysis cnc3 notebook.

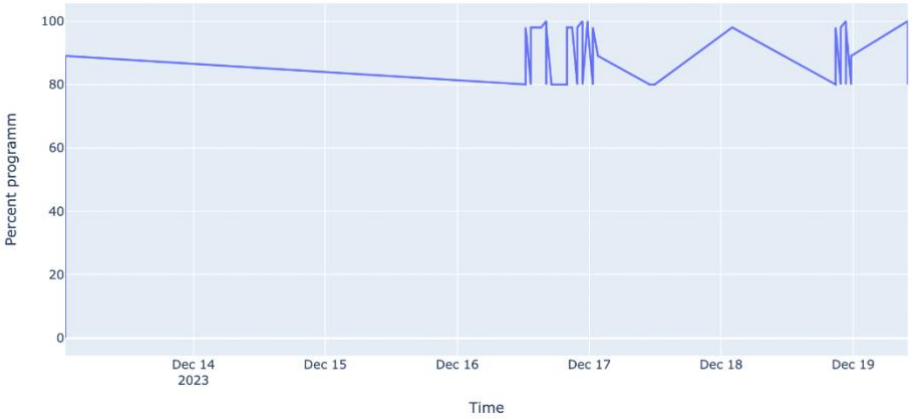
6. Percentage of program completion (Процент выполнения программы)



Status Over Time: /NC/examples/05_18.nc



Status Over Time: /NC/examples/07_06.nc



For the most part, the programs also began to perform work closer to December 17. More program schedules can be found in the Data Analysis cnc3 notebook.

7. Program (Программа)

There are 111 programs in total. Below is a list of these programs:

array(['/NC/examples/05_17.nc', nan, '/NC/examples/ToZero.nc',
'/NC/examples/05_18.nc', '/NC/examples/05_19.nc',
'/NC/examples/06_01.nc', '/NC/examples/06_02.nc',

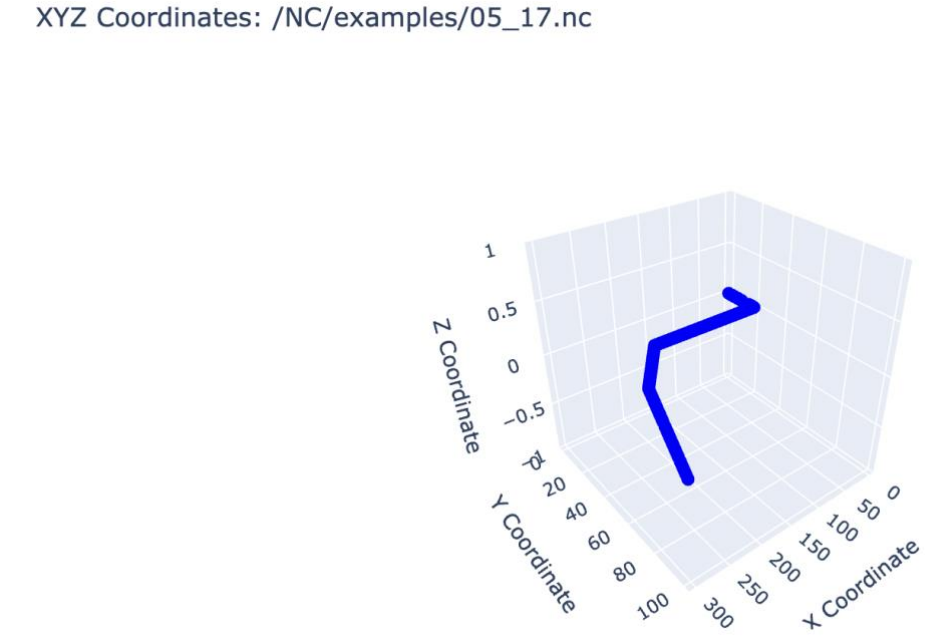
'/NC/examples/06_03.nc', '/NC/examples/06_04.nc',
'/NC/examples/06_05.nc', '/NC/examples/06_06.nc',
'/NC/examples/06_07.nc', '/NC/examples/06_08.nc',
'/NC/examples/06_09.nc', '/NC/examples/07_01.nc',
'/NC/examples/07_02.nc', '/NC/examples/07_03.nc',
'/NC/examples/07_04.nc', '/NC/examples/07_05.nc',
'/NC/examples/07_06.nc', '/NC/examples/07_07.nc',
'/NC/examples/07_08.nc', '/NC/examples/08_01.nc',
'/NC/examples/08_02.nc', '/NC/examples/08_03.nc',
'/NC/examples/08_04.nc', '/NC/examples/08_05.nc',
'/NC/examples/08_06.nc', '/NC/examples/08_07.nc',
'/NC/examples/08_08.nc', '/NC/examples/08_09.nc',
'/NC/examples/08_10.nc', '/NC/examples/08_11.nc',
'/NC/examples/08_12.nc', '/NC/examples/08_13.nc',
'/NC/examples/10_09.nc', '/NC/examples/11_06.nc',
'/NC/examples/11_07.nc', '/NC/examples/11_08.nc',
'/NC/examples/11_09.nc', '/NC/examples/11_10.nc',
'/NC/examples/11_11.nc', '/NC/examples/11_12.nc',
'/NC/examples/12_01.nc', '/NC/examples/08_14.nc',
'/NC/examples/09_01.nc', '/NC/examples/09_02.nc',
'/NC/examples/09_03.nc', '/NC/examples/09_04.nc',
'/NC/examples/10_01.nc', '/NC/examples/10_02.nc',
'/NC/examples/10_03.nc', '/NC/examples/10_04.nc',
'/NC/examples/10_05.nc', '/NC/examples/10_06.nc',
'/NC/examples/10_08.nc', '/NC/examples/10_10.nc',
'/NC/examples/10_11.nc', '/NC/examples/10_12.nc',
'/NC/examples/10_13.nc', '/NC/examples/10_14.nc',
'/NC/examples/11_01.nc', '/NC/examples/11_02.nc',
'/NC/examples/11_03.nc', '/NC/examples/11_04.nc',
'/NC/examples/11_05.nc', '/NC/examples/12_02.nc',
'/NC/examples/12_03.nc', '/NC/examples/04_01.nc',
'/NC/examples/04_02.nc', '/NC/examples/04_03.nc',
'/NC/examples/04_04.nc', '/NC/examples/04_05.nc',
'/NC/examples/04_06.nc', '/NC/examples/04_07.nc',
'/NC/examples/04_08.nc', '/NC/examples/04_09.nc',

'/NC/examples/04_10.nc', '/NC/examples/04_11.nc',
'/NC/examples/04_12.nc', '/NC/examples/04_13.nc',
'/NC/examples/04_14.nc', '/NC/examples/04_15.nc',
'/NC/examples/04_16.nc', '/NC/examples/04_17.nc',
'/NC/examples/04_18.nc', '/NC/examples/04_19.nc',
'/NC/examples/04_20.nc', '/NC/examples/04_21.nc',
'/NC/examples/04_22.nc', '/NC/examples/04_23.nc',
'/NC/examples/04_24.nc', '/NC/examples/04_25.nc',
'/NC/examples/04_26.nc', '/NC/examples/04_27.nc',
'/NC/examples/04_28.nc', '/NC/examples/05_01.nc',
'/NC/examples/05_02.nc', '/NC/examples/05_03.nc',
'/NC/examples/05_04.nc', '/NC/examples/05_05.nc',
'/NC/examples/05_06.nc', '/NC/examples/05_07.nc',
'/NC/examples/05_08.nc', '/NC/examples/05_09.nc',
'/NC/examples/05_10.nc', '/NC/examples/05_11.nc',
'/NC/examples/05_12.nc', '/NC/examples/05_13.nc',
'/NC/examples/05_14.nc', '/NC/examples/05_15.nc',
'/NC/examples/05_16.nc'], dtype=object)

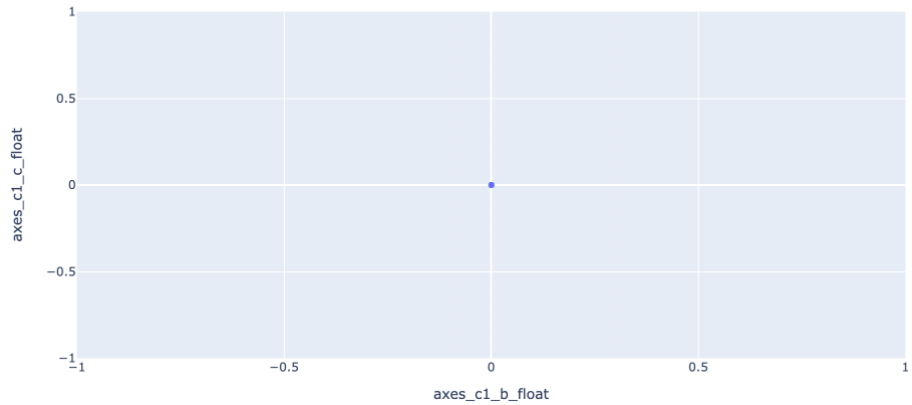
(visualization for the part is in the 3.2 point of the document)

8. All axes

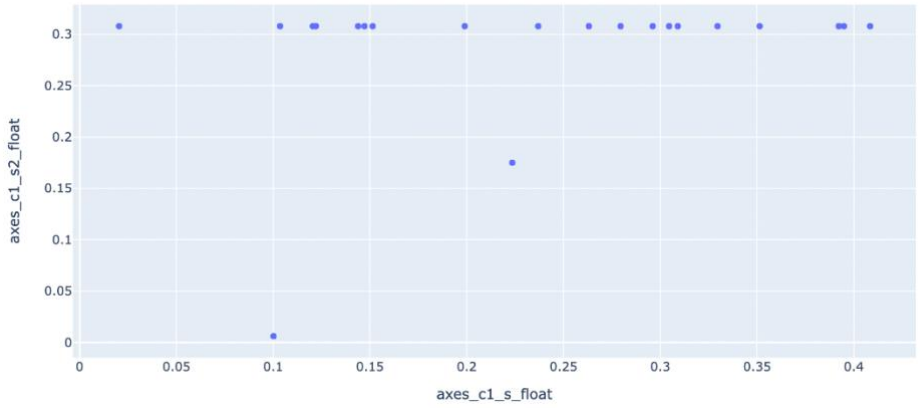
Channel 1 coordinates



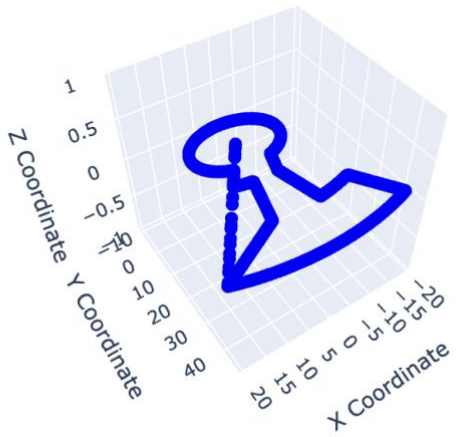
B C coordinates: /NC/examples/05_17.nc



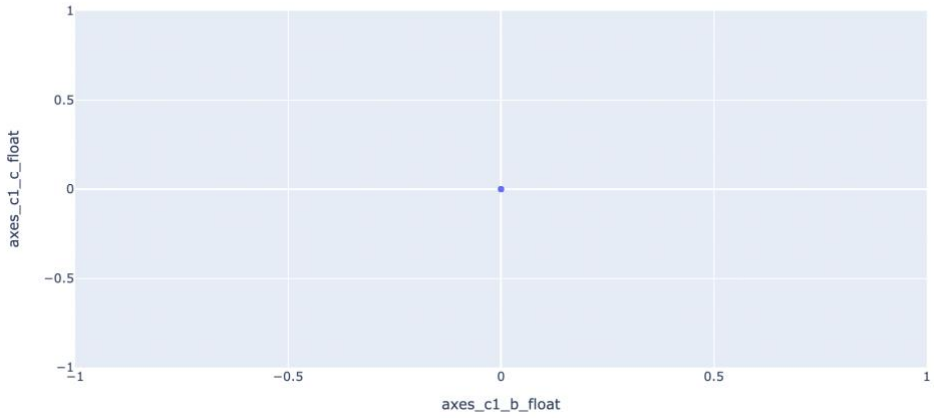
S S2 coordinates: /NC/examples/05_17.nc



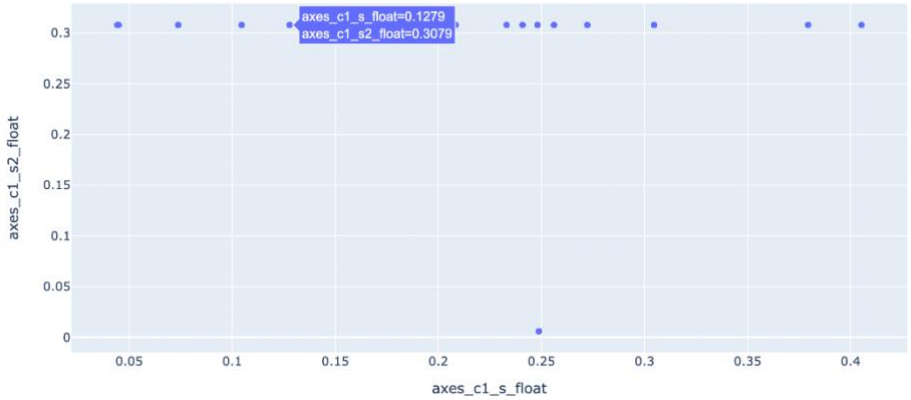
XYZ Coordinates: /NC/examples/04_28.nc



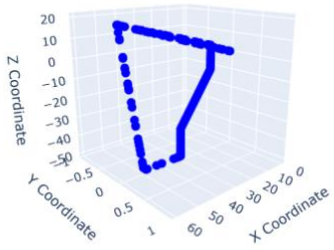
B C coordinates: /NC/examples/04_28.nc

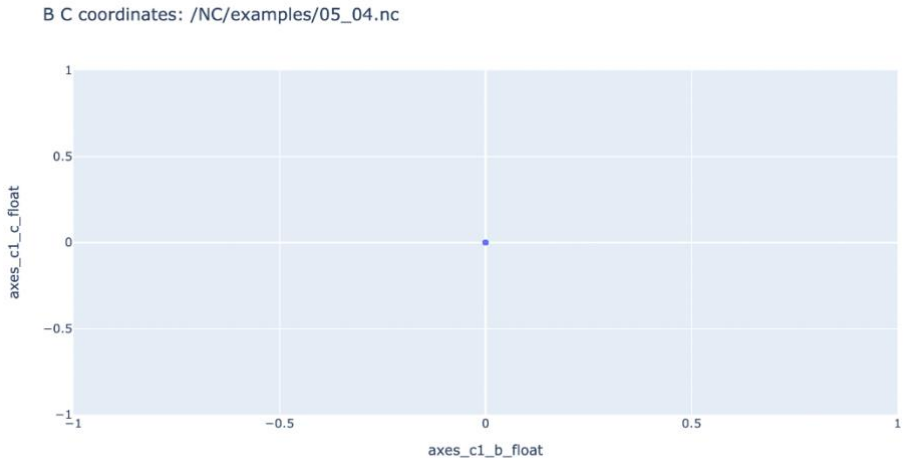


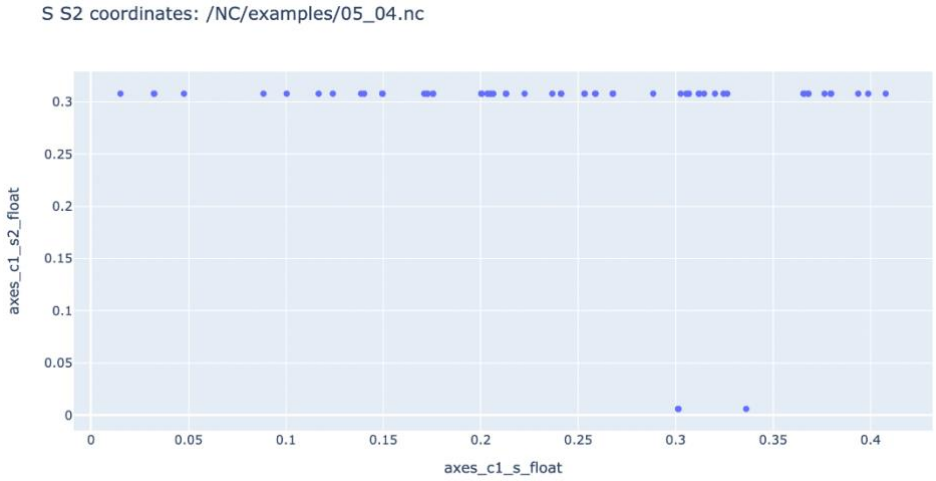
S S2 coordinates: /NC/examples/04_28.nc



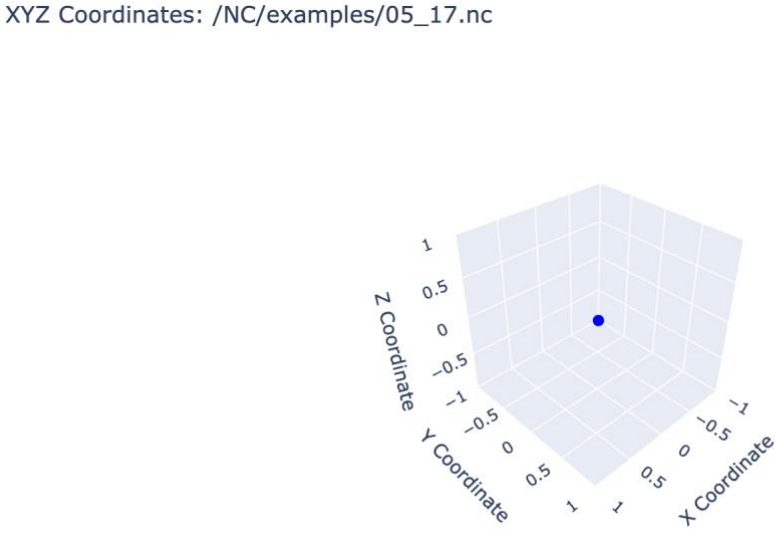
XYZ Coordinates: /NC/examples/05_04.nc



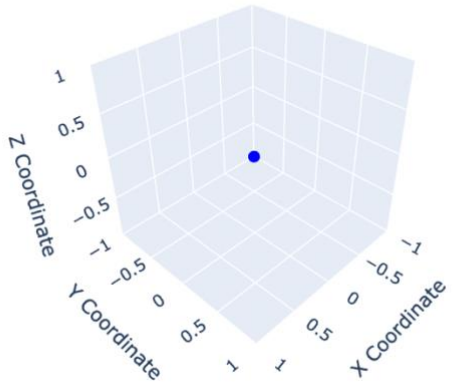




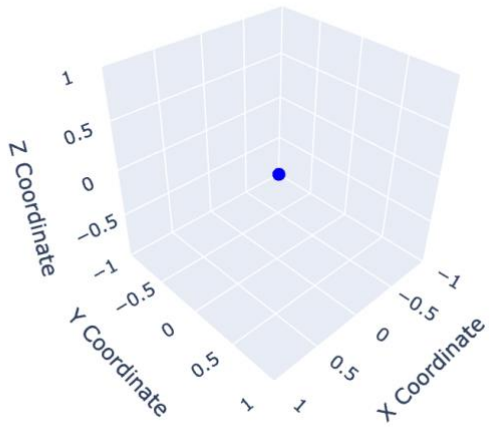
Channel 2 coordinates



XYZ Coordinates: /NC/examples/04_28.nc

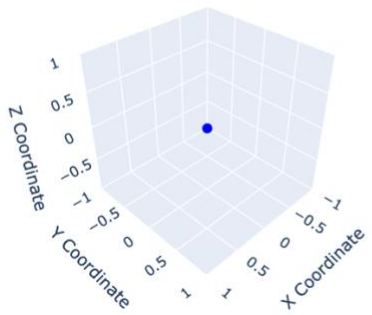


XYZ Coordinates: /NC/examples/05_04.nc

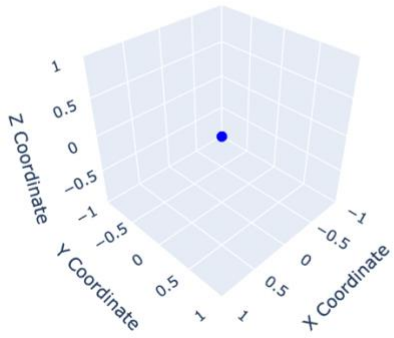


Channel 3 coordinates

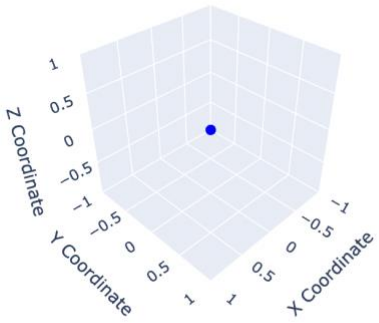
XYZ Coordinates: /NC/examples/05_17.nc



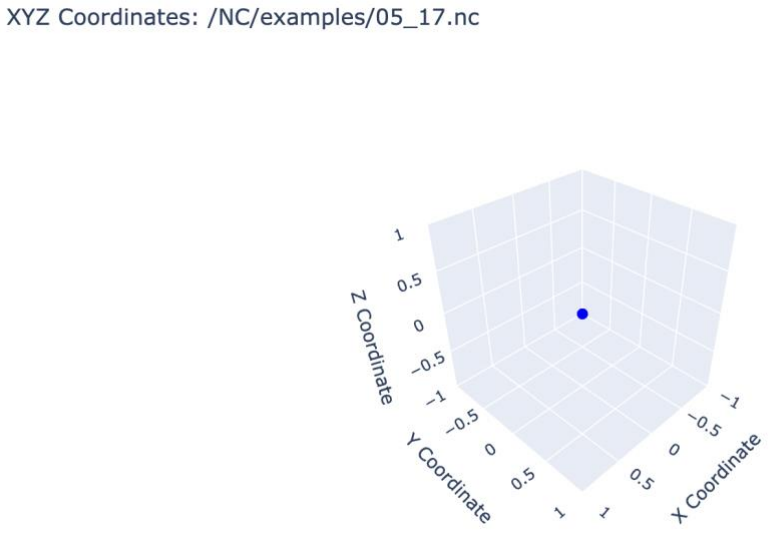
XYZ Coordinates: /NC/examples/04_28.nc

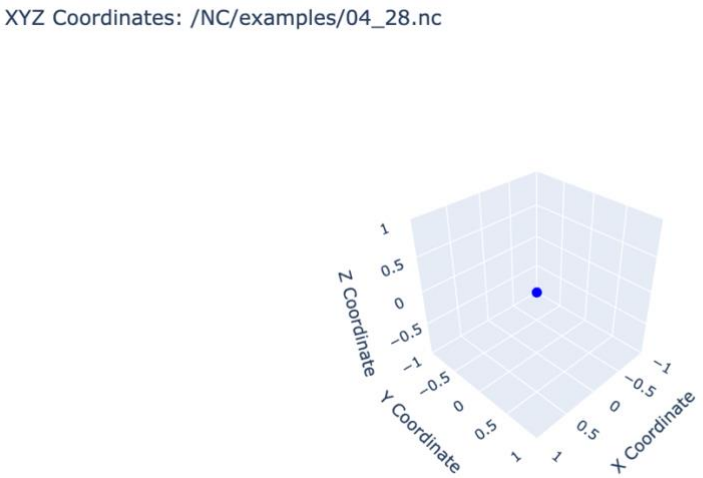


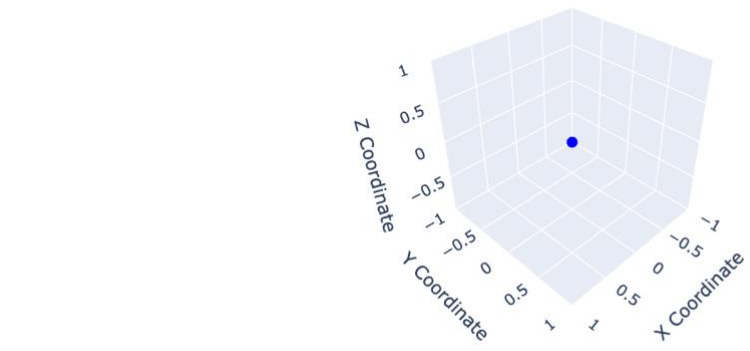
XYZ Coordinates: /NC/examples/05_04.nc



Channel 4 coordinates







9. Errors

There were no errors in the CNC3, as the program was running smoothly

1.2.2 Visualization in Python for CNC 5

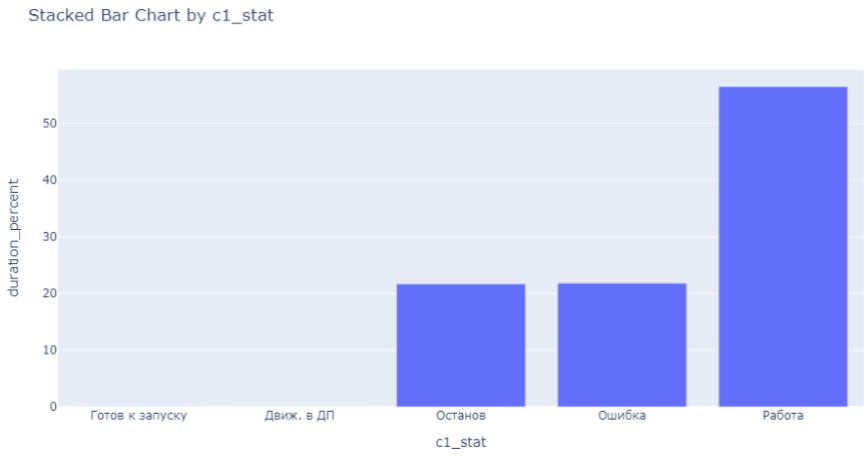
To visualize data from CNC 5 the following plots were done:

1. Core Work Time Current (Время работы ядра (текущее):

2. Channel Total Work Time (Время работы канала (всего))

As we can see in the CNC 5 (the same way as in CNC3) occurred an error due to which after the machines went back online the total and the current time of the core rebooted.

3. Status of channels (Статус канала):

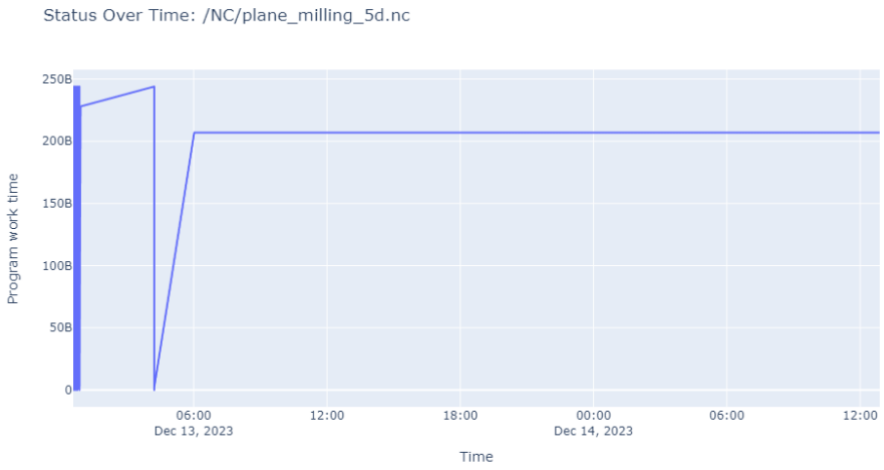


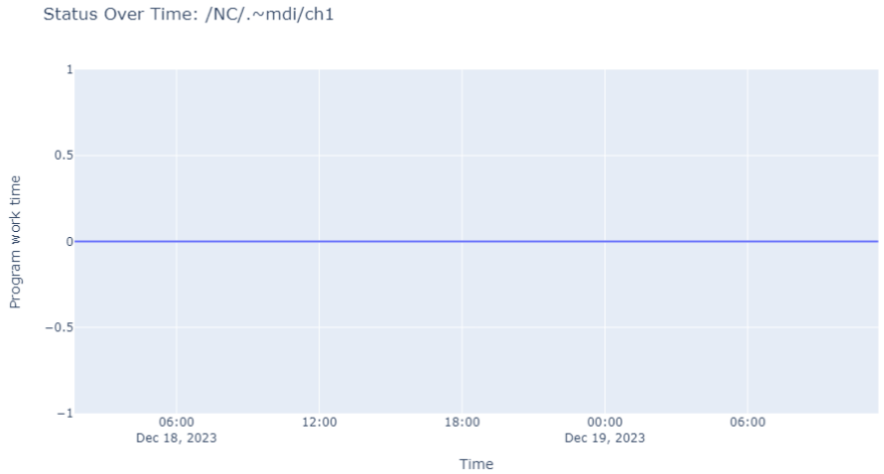
Here we can see that most of the time it was in working process, and the rest is splitted between the stop and error “modes”

4. Channel mode (Режим канала)

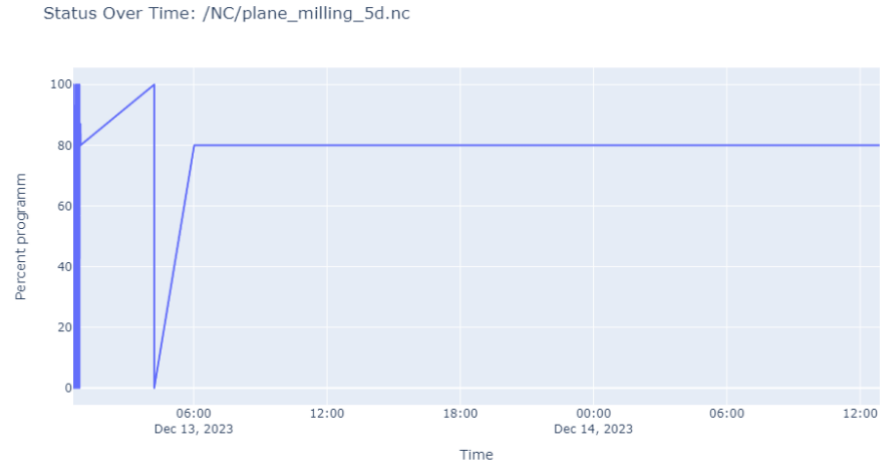
The channel was in auto mode some time and then appeared MDI.

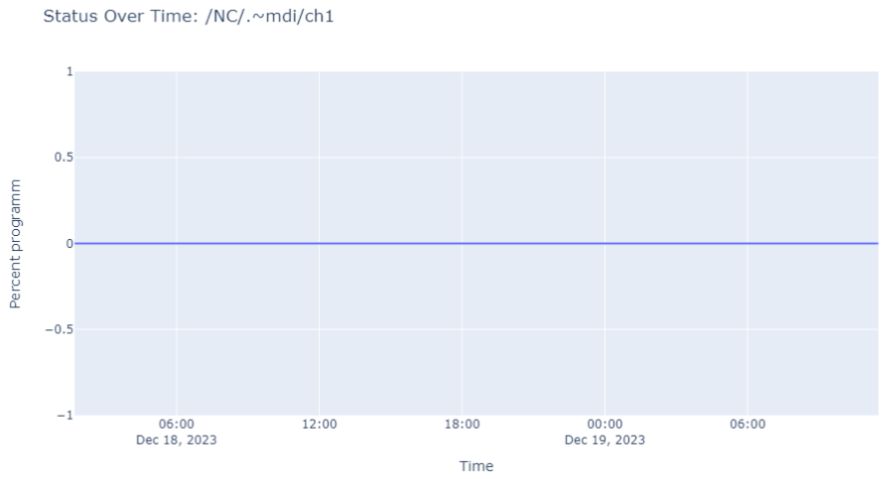
5. Program work time evaluation:

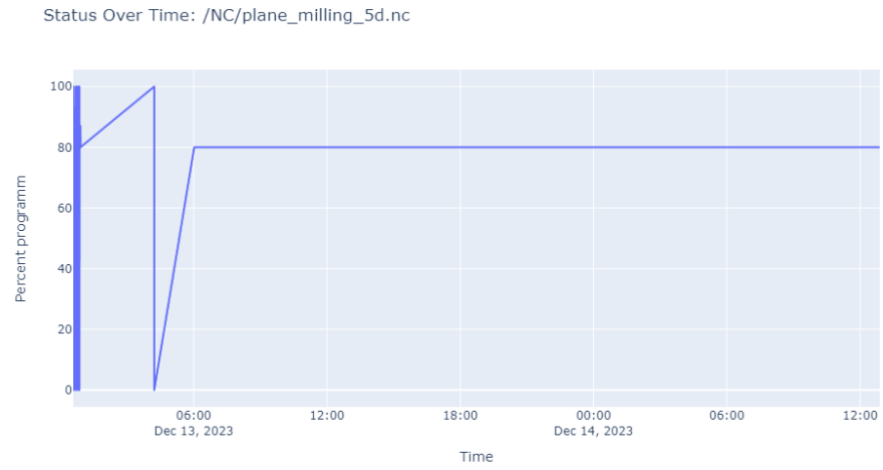


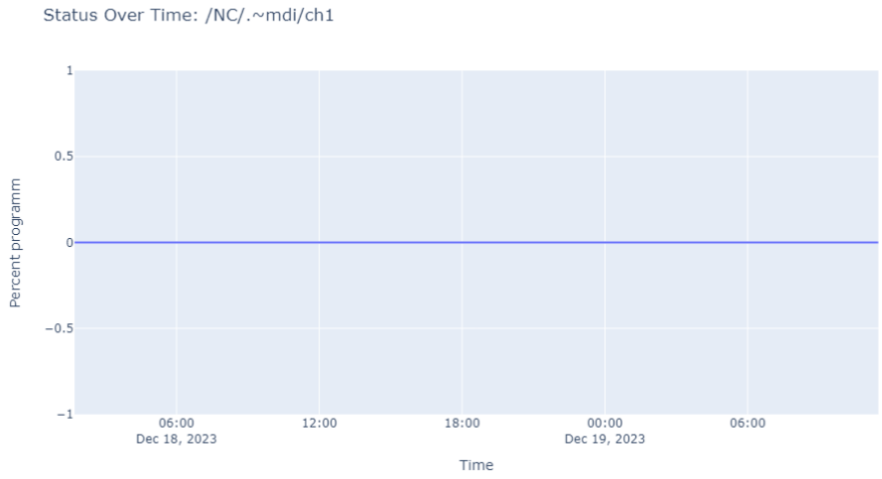


6. Percentage of program completion (Процент выполнения программы)









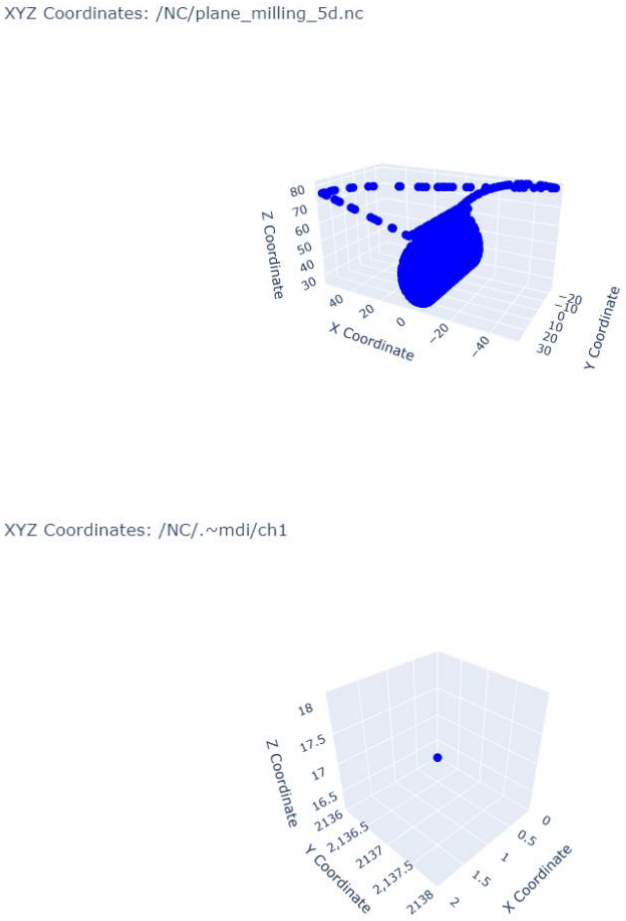
7. Program (Программа)

array(['/NC/plane_milling_5d.nc', nan, '/NC/.~mdi/ch1'], dtype=object)

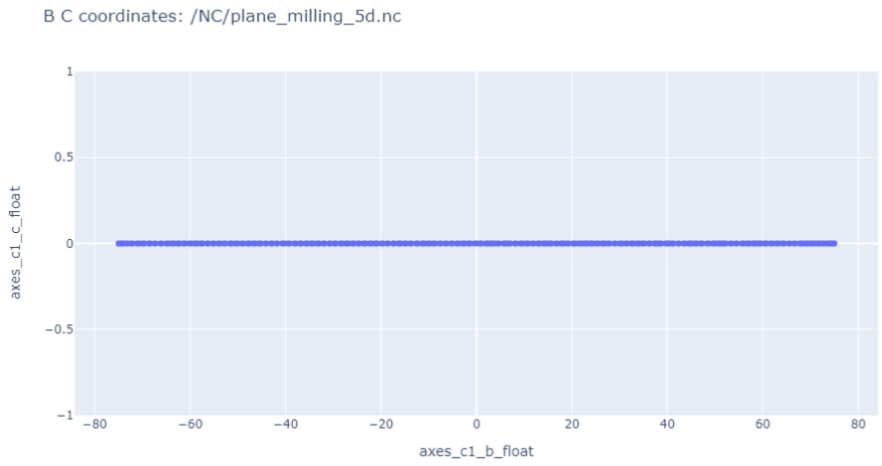
(visualization for the part is in the 3.2 point of the document)

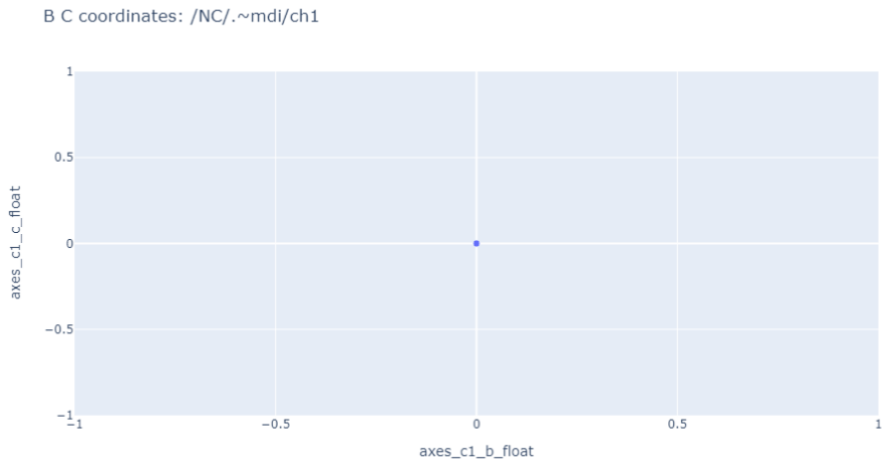
8. All axes

XYZ coordinates:



B C coordinates:





As we can see, on perspective of b-c coordinates it “moves” along the b scale.

10. Errors

A visualization of the errors will be presented in the 3.5 part of the document

1.3 Create page with MQTT data in Node-Red (Demo data and real data) (Seda, Vladimir)

To visualize data from sensors the following charts were done:

1. In node red flows were created.
    a. MQTT IN; where the server with the topic were written;
    b. 2 groups of visualization: Gauge and charts in the Mqtt tab.

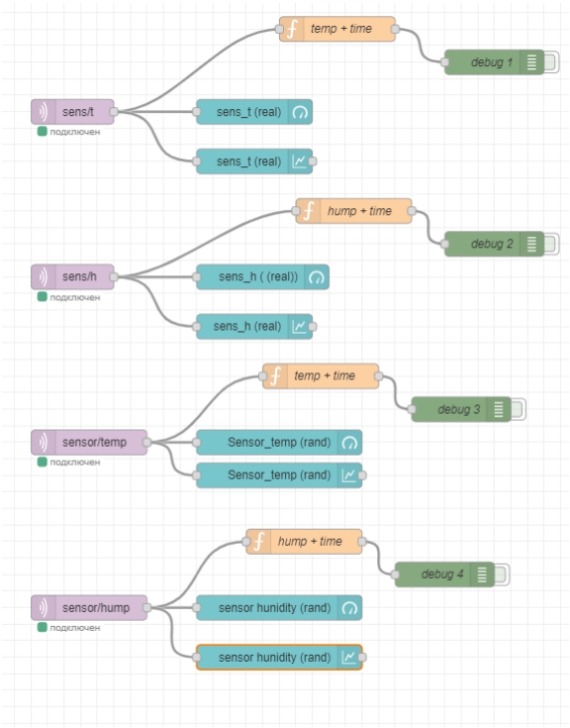


Figure 12. Node-red MQTT visualization flow

2. In the tab there are 3 groups of visualizations (Figure 12):
    a. Mqtt Gauge:
        i. sens_t (real);
        ii. sens_h ((real));
        iii. Sensor_temp (rand);
        iv. sensor hunidity (rand).
    b. Mqtt Charts real data:
        i. sens_t (real);
        ii. sens_h (real).
    c. Charts demo data:
        i. Sensor_temp (rand);
        ii. sensor hunidity (rand).

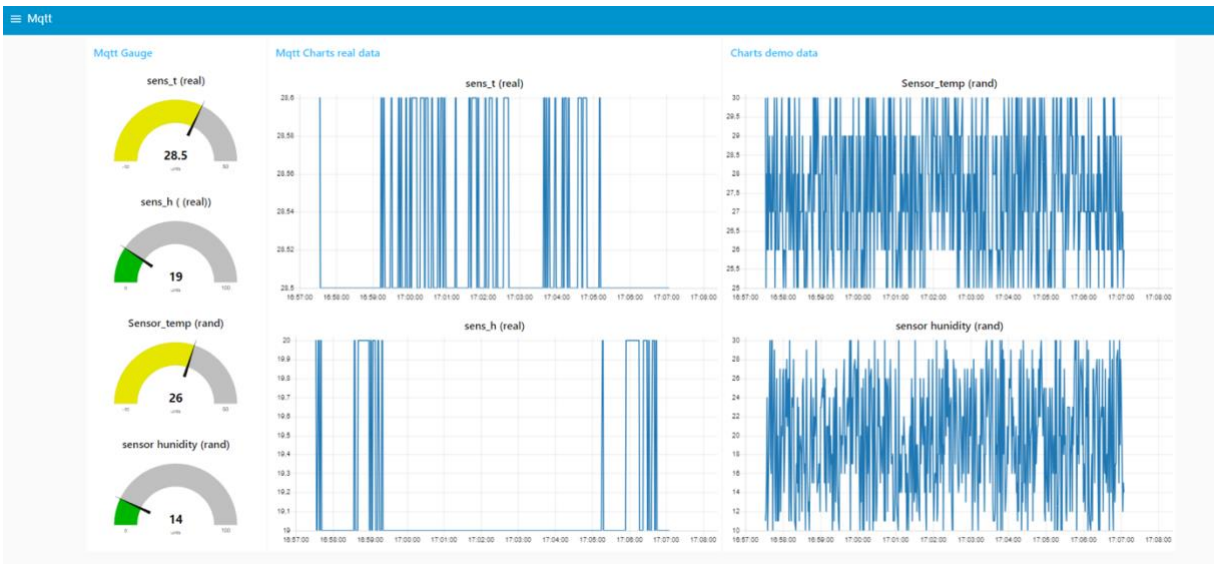


Figure 13. Node-red Mqtt tab

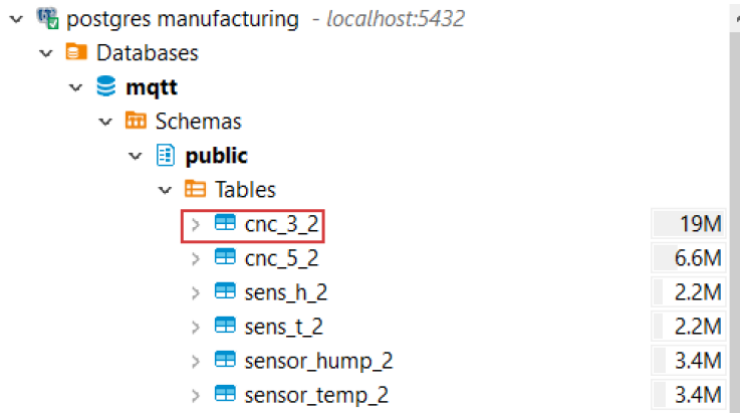
2 AGREGATION DATA (ELEONORA)

To accumulate data from CNC machines and sensors to a Postgres server, the following actions were done:

2.1 Accumulation of CNC 3 data

Firstly, to read data into db a table with all necessary fields should be created.
In Postgres DB:

1. Create “mqtt” database wich will store the data read from CNC and mqtt.
2. Create “cnc_3_2” table where data from CNC 3 machine is going to be stored



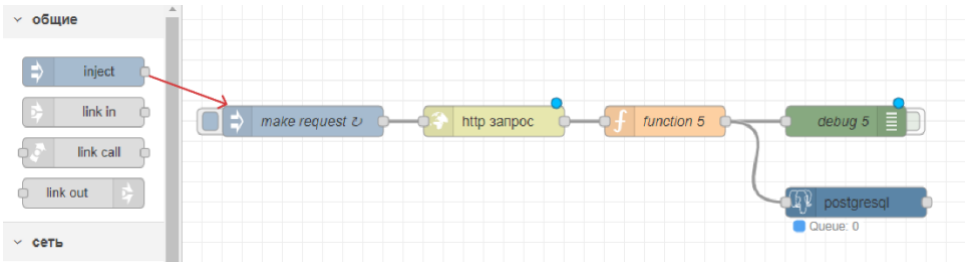
The DDL of the table:

```
CREATE TABLE public.cnc_3_2 (
motoc_hours_name varchar(25) NULL,
core_work_time_ttl_value varchar(25) NULL,
core_work_time_cur_value varchar(25) NULL,
core_restart_cnt int4 NULL,
channel_cnt int4 NULL,
c1_tot_work_time varchar(25) NULL,
c1_work_time_now varchar(25) NULL,
c1_stat varchar(25) NULL,
c1_mode varchar(25) NULL,
c1_prog_work_time varchar(25) NULL,
c1_percent_prog int4 NULL,
c1_prog varchar(25) NULL,
axes_c1_x_float numeric NULL,
axes_c1_y_float numeric NULL,
axes_c1_z_float numeric NULL,
axes_c1_b_float numeric NULL,
axes_c1_c_float numeric NULL,
axes_c1_s_float numeric NULL,
axes_c1_s2_float numeric NULL,
axes_c2_x_float numeric NULL,
axes_c2_y_float numeric NULL,
axes_c2_z_float numeric NULL,
axes_c2_b_float numeric NULL,
axes_c2_c_float numeric NULL,
axes_c2_s_float numeric NULL,
axes_c2_s2_float numeric NULL,
axes_c3_x_float numeric NULL,
axes_c3_y_float numeric NULL,
axes_c3_z_float numeric NULL,
```

```
axes_c3_b_float numeric NULL,
axes_c3_c_float numeric NULL,
axes_c3_s_float numeric NULL,
axes_c3_s2_float numeric NULL,
axes_c4_x_float numeric NULL,
axes_c4_y_float numeric NULL,
axes_c4_z_float numeric NULL,
axes_c4_b_float numeric NULL,
axes_c4_c_float numeric NULL,
axes_c4_s_float numeric NULL,
axes_c4_s2_float numeric NULL,
time_load timestamp NULL
);
```

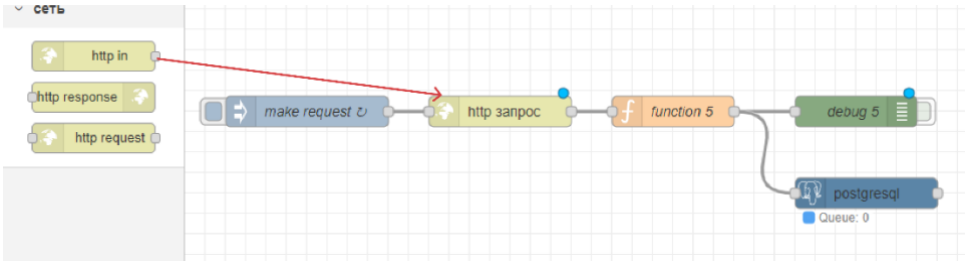
Having the table to read data into the process of data parsing could be established. In Node-RED:

1. Add “inject” element for making requests to the data source.

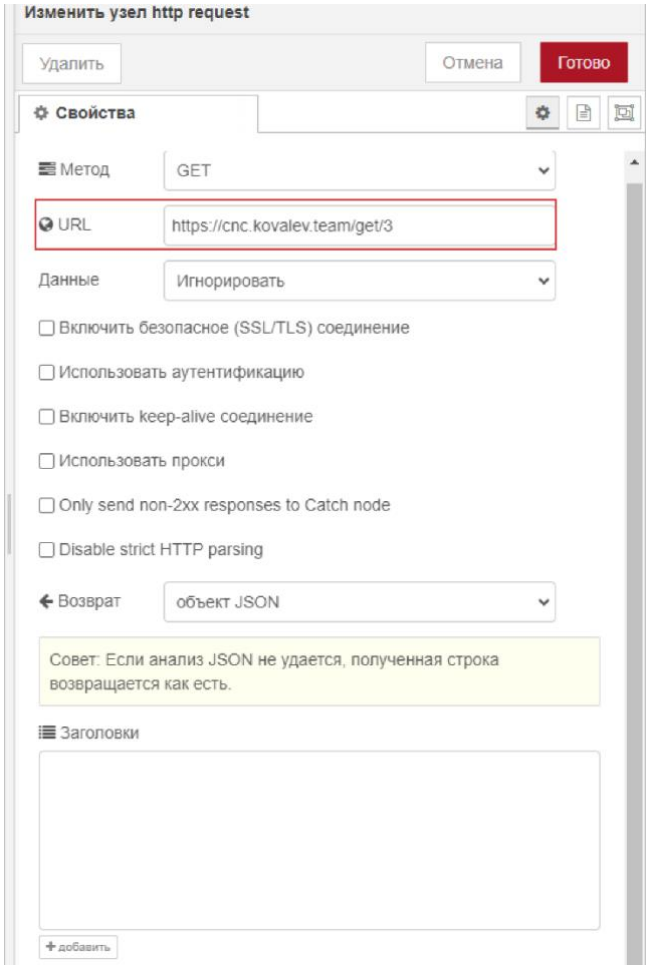


Change repeat parameter to the mode “with interval” and set interval to 1 second.

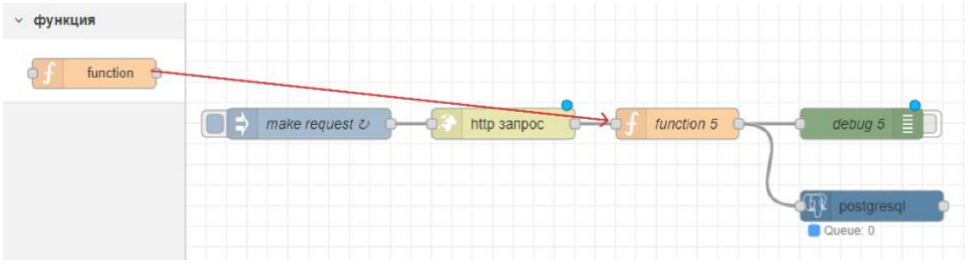
2. Add “http in” element for specifying data source.



Specify URL address which is going to be parsed.



3. Add “function” element to write a function to parse particular fields from json that comes from http request.



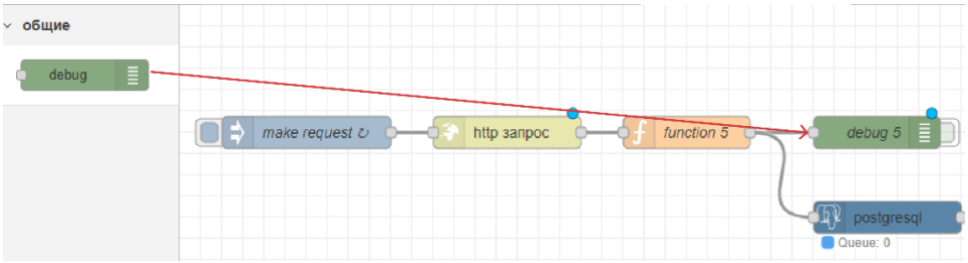
Write a function code to read all the fields in the right data type and remove service characters. Function code is as follows:

```
function formatDuration(hours, minutes, seconds) {
  // Ensure that hours, minutes, and seconds are integers
  hours = parseInt(hours, 10) || 0;
  minutes = parseInt(minutes, 10) || 0;
  seconds = parseInt(seconds, 10) || 0;

  // Format each component to have at least two digits
  const formattedHours = String(hours).padStart(3, '0');
  const formattedMinutes = String(minutes).padStart(2, '0');
  const formattedSeconds = String(seconds).padStart(2, '0');

  return `${formattedHours}:${formattedMinutes}:${formattedSeconds}`;
```

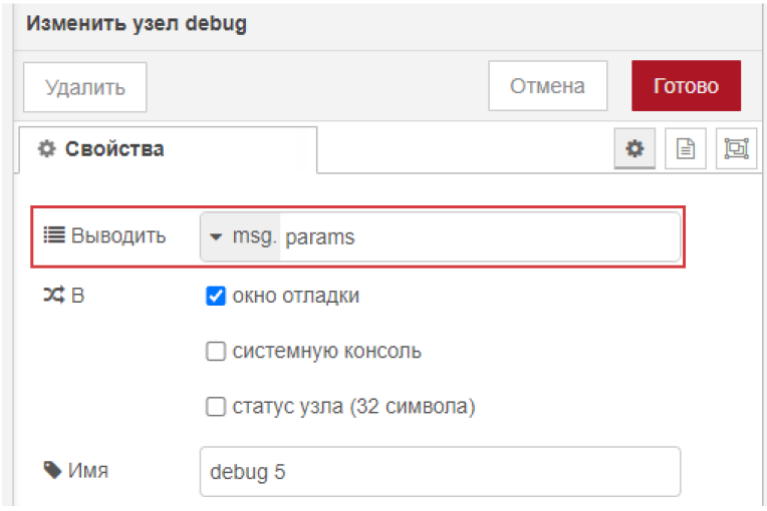
```
}

// Function to extract number and convert to float
function extractNumberAndConvertToFloat(value) {
   // Extract number from the value (remove мм)
   var extractedNumber = parseFloat(value.replace(' мм', '').replace(',', '.')) || 0;

   // Convert to float format
   return extractedNumber;
}

var jsonData = msg.payload.data;

// Extract and format data as needed for PostgreSQL
var motocHoursData = jsonData[0][1];
var motoc_hours_name = jsonData[0][0];
var core_work_time_ttl_value = motocHoursData[0][1];
var core_work_time_cur_value = motocHoursData[1][1];
var core_restart_cnt = parseInt(motocHoursData[2][1], 10) || 0;
var channel_cnt = parseInt(motocHoursData[3][1], 10) || 0;


var channel1Data = jsonData[1][1];
var c1_tot_work_time= channel1Data[0][1];
var c1_work_time_now=channel1Data[1][1];
var c1_stat= channel1Data[2][1];
var c1_mode= channel1Data[3][1];
var c1_prog_work_time= channel1Data[4][1];
// Extract the percentage value from the string
var c1_percent_prog= channel1Data[5][1];
c1_percent_prog= parseInt(c1_percent_prog.replace('%', ''), 10) || 0;
var c1_prog=channel1Data[6][1];

var axes_channel1 = jsonData[2][1];
var axes_C1_X_float = extractNumberAndConvertToFloat(axes_channel1[0][1]);
var axes_C1_Y_float = extractNumberAndConvertToFloat(axes_channel1[1][1]);
var axes_C1_Z_float = extractNumberAndConvertToFloat(axes_channel1[2][1]);
var axes_C1_B_float = extractNumberAndConvertToFloat(axes_channel1[3][1]);
var axes_C1_C_float = extractNumberAndConvertToFloat(axes_channel1[4][1]);
var axes_C1_S_float = extractNumberAndConvertToFloat(axes_channel1[5][1]);
var axes_C1_S2_float = extractNumberAndConvertToFloat(axes_channel1[6][1]);

var axes_channel2 = jsonData[3][1];
var axes_C2_X_float = extractNumberAndConvertToFloat(axes_channel2[0][1]);
var axes_C2_Y_float = extractNumberAndConvertToFloat(axes_channel2[1][1]);
var axes_C2_Z_float = extractNumberAndConvertToFloat(axes_channel2[2][1]);
var axes_C2_B_float = extractNumberAndConvertToFloat(axes_channel2[3][1]);
var axes_C2_C_float = extractNumberAndConvertToFloat(axes_channel2[4][1]);
var axes_C2_S_float = extractNumberAndConvertToFloat(axes_channel2[5][1]);
var axes_C2_S2_float = extractNumberAndConvertToFloat(axes_channel2[6][1]);

var axes_channel3 = jsonData[4][1];
var axes_C3_X_float = extractNumberAndConvertToFloat(axes_channel3[0][1]);
var axes_C3_Y_float = extractNumberAndConvertToFloat(axes_channel3[1][1]);
var axes_C3_Z_float = extractNumberAndConvertToFloat(axes_channel3[2][1]);
var axes_C3_B_float = extractNumberAndConvertToFloat(axes_channel3[3][1]);
var axes_C3_C_float = extractNumberAndConvertToFloat(axes_channel3[4][1]);
var axes_C3_S_float = extractNumberAndConvertToFloat(axes_channel3[5][1]);
var axes_C3_S2_float = extractNumberAndConvertToFloat(axes_channel3[6][1]);
```

```
var axes_channel4 = jsonData[5][1];
var axes_C4_X_float = extractNumberAndConvertToFloat(axes_channel4[0][1]);
var axes_C4_Y_float = extractNumberAndConvertToFloat(axes_channel4[1][1]);
var axes_C4_Z_float = extractNumberAndConvertToFloat(axes_channel4[2][1]);
var axes_C4_B_float = extractNumberAndConvertToFloat(axes_channel4[3][1]);
var axes_C4_C_float = extractNumberAndConvertToFloat(axes_channel4[4][1]);
var axes_C4_S_float = extractNumberAndConvertToFloat(axes_channel4[5][1]);
var axes_C4_S2_float = extractNumberAndConvertToFloat(axes_channel4[6][1]);

const currentDateTime = new Date();
const formattedTimestamp = currentDateTime.toISOString()
  .replace('T', ' ')
  .replace('Z', '');


msg.params = [motoc_hours_name, core_work_time_ttl_value
    , core_work_time_cur_value, core_restart_cnt, channel_cnt
    , c1_tot_work_time, c1_work_time_now, c1_stat, c1_mode, c1_prog_work_time
    , c1_percent_prog, c1_prog, axes_C1_X_float, axes_C1_Y_float,axes_C1_Z_float
    , axes_C1_B_float, axes_C1_C_float, axes_C1_S_float, axes_C1_S2_float
    , axes_C2_X_float, axes_C2_Y_float, axes_C2_Z_float, axes_C2_B_float
    , axes_C2_C_float, axes_C2_S_float, axes_C2_S2_float, axes_C3_X_float
    , axes_C3_Y_float, axes_C3_Z_float, axes_C3_B_float, axes_C3_C_float
    , axes_C3_S_float, axes_C3_S2_float, axes_C4_X_float, axes_C4_Y_float
    , axes_C4_Z_float, axes_C4_B_float, axes_C4_C_float, axes_C4_S_float
    , axes_C4_S2_float, formattedTimestamp];

return msg;
```

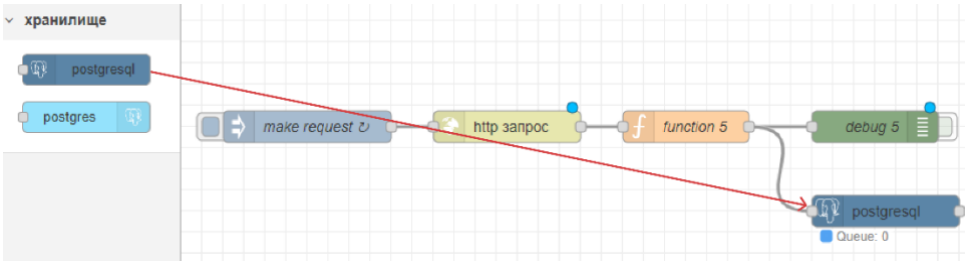
4. Add “debug” element to check if everything is being read in a proper way.



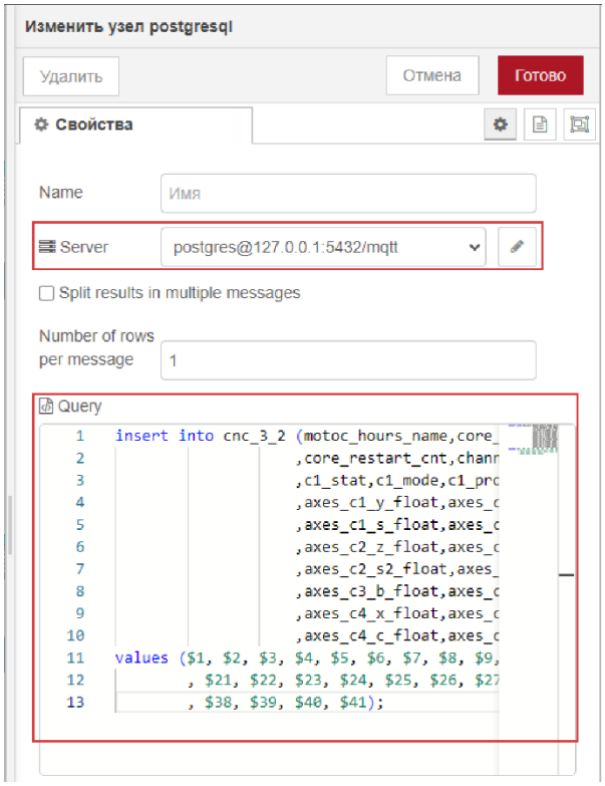
Specify what should be shown in debug window – parameters (params) of message that is being read in “function” element.



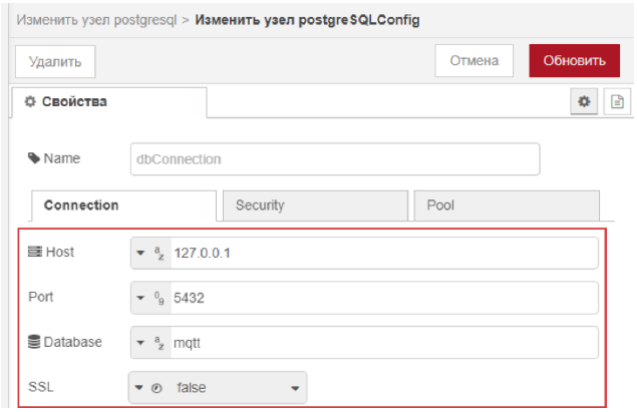
5. Add “postgresql” element to write parsed data into postgres database.



Determine the database server address where data is going to be written and the query which will insert data into database.



Set the Host, Port, Database and SSL connection parameters.



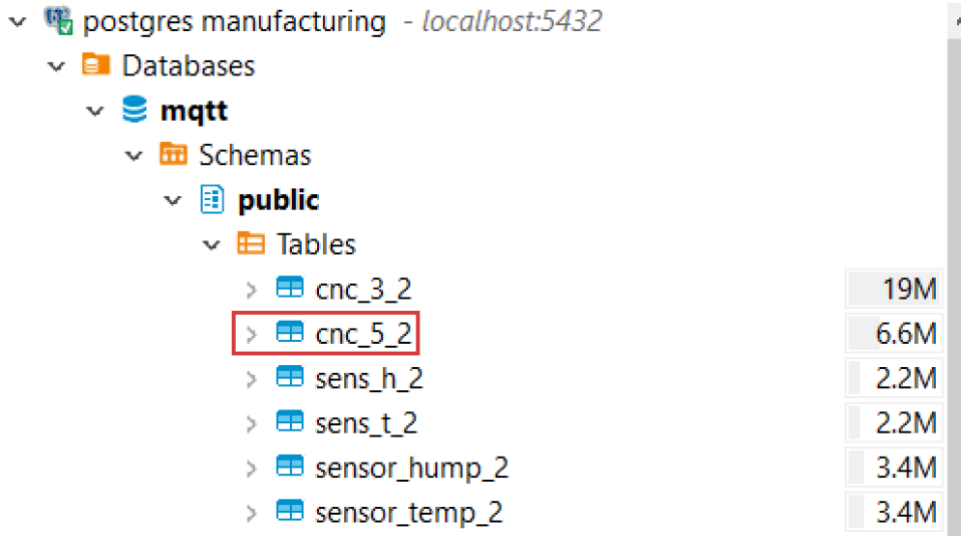
Query to insert data:

```
insert into cnc_3_2 (motoc_hours_name,core_work_time_ttl_value,core_work_time_cur_value
                    ,core_restart_cnt,channel_cnt,c1_tot_work_time,c1_work_time_now
                    ,c1_stat,c1_mode,c1_prog_work_time,c1_percent_prog,c1_prog,axes_c1_x_float
                    ,axes_c1_y_float,axes_c1_z_float,axes_c1_b_float,axes_c1_c_float
                    ,axes_c1_s_float,axes_c1_s2_float,axes_c2_x_float,axes_c2_y_float
                    ,axes_c2_z_float,axes_c2_b_float,axes_c2_c_float,axes_c2_s_float
                    ,axes_c2_s2_float,axes_c3_x_float,axes_c3_y_float,axes_c3_z_float
                    ,axes_c3_b_float,axes_c3_c_float,axes_c3_s_float,axes_c3_s2_float
                    ,axes_c4_x_float,axes_c4_y_float,axes_c4_z_float,axes_c4_b_float
                    ,axes_c4_c_float,axes_c4_s_float,axes_c4_s2_float,time_load)
values ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16, $17, $18, $19, $20
      , $21, $22, $23, $24, $25, $26, $27, $28, $29, $30, $31, $32, $33, $34, $35, $36, $37
      , $38, $39, $40, $41);
```

2.2 Accumulation of CNC 5 data

In Postgres DB:

1. “mqtt” database was already created at the previous step for CNC 3.
2. Create “cnc_5_2” table where data from CNC 3 machine is going to be stored.



The DDL of the table:

```
CREATE TABLE public.cnc_5_2 (
motoc_hours_name varchar(25) NULL,
core_work_time_ttl_value varchar(25) NULL,
```

```
core_work_time_cur_value varchar(25) NULL,
core_restart_cnt int4 NULL,
channel_cnt int4 NULL,
c1_tot_work_time varchar(25) NULL,
c1_work_time_now varchar(25) NULL,
c1_stat varchar(25) NULL,
c1_mode varchar(25) NULL,
c1_prog_work_time varchar(25) NULL,
c1_percent_prog int4 NULL,
c1_prog varchar(25) NULL,
axes_c1_x_float numeric NULL,
axes_c1_y_float numeric NULL,
axes_c1_z_float numeric NULL,
axes_c1_b_float numeric NULL,
axes_c1_c_float numeric NULL,
axes_c1_s_float numeric NULL,
time_load timestamp NULL
);
```

Steps for data parsing in Node-RED are the same as for CNC 3 except following:

The function code to read all the fields in the right data type and remove service characters. Function code is as follows:

```
function extractNumberAndConvertToFloat(value) {
    // Extract number from the value (remove мм)
    var extractedNumber = parseFloat(value.replace(' мм', '').replace(',', '.')) || 0;

    // Convert to float format
    return extractedNumber;
}

var jsonData = msg.payload.data;

var motocHoursData = jsonData[0][1];
var motoc_hours_name = jsonData[0][0];
var core_work_time_ttl_value = motocHoursData[0][1];
var core_work_time_cur_value = motocHoursData[1][1];
var core_restart_cnt = parseInt(motocHoursData[2][1], 10) || 0;
var channel_cnt = parseInt(motocHoursData[3][1], 10) || 0;

var channel1Data = jsonData[1][1];
var c1_tot_work_time= channel1Data[0][1];
var c1_work_time_now=channel1Data[1][1];
var c1_stat= channel1Data[2][1];
var c1_mode= channel1Data[3][1];
var c1_prog_work_time= channel1Data[4][1];
// Extract the percentage value from the string
var c1_percent_prog= channel1Data[5][1];
c1_percent_prog= parseInt(c1_percent_prog.replace('%', ''), 10) || 0;
var c1_prog=channel1Data[6][1];

var axes_channel1 = jsonData[2][1];
var axes_C1_X_float = extractNumberAndConvertToFloat(axes_channel1[0][1]);
var axes_C1_Y_float = extractNumberAndConvertToFloat(axes_channel1[1][1]);
var axes_C1_Z_float = extractNumberAndConvertToFloat(axes_channel1[2][1]);
var axes_C1_B_float = extractNumberAndConvertToFloat(axes_channel1[3][1]);
var axes_C1_C_float = extractNumberAndConvertToFloat(axes_channel1[4][1]);
var axes_C1_S_float = extractNumberAndConvertToFloat(axes_channel1[5][1]);
```

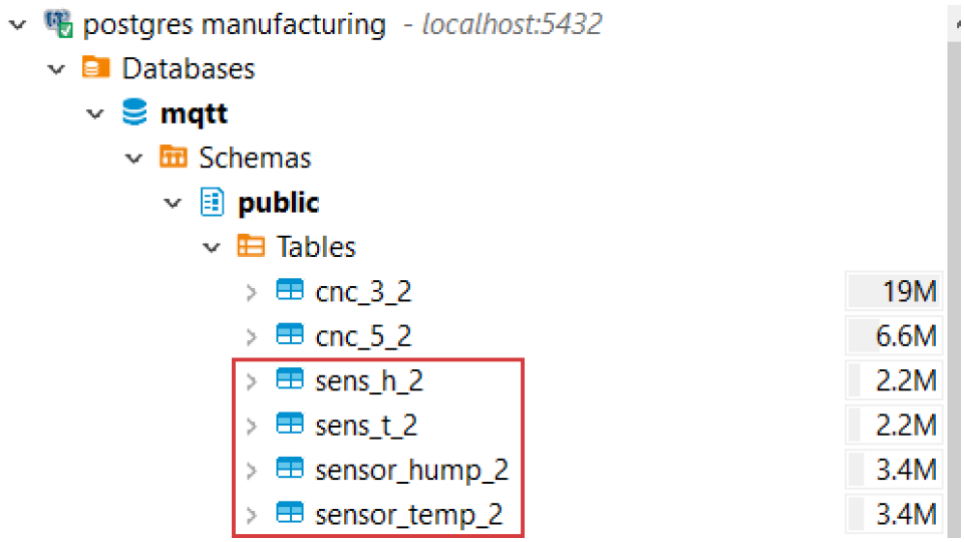
```
const currentDateTime = new Date();
const formattedTimestamp = currentDateTime.toISOString()
  .replace('T', ' ')
  .replace('Z', '');


msg.params = [motoc_hours_name, core_work_time_ttl_value
    , core_work_time_cur_value, core_restart_cnt, channel_cnt
    , c1_tot_work_time, c1_work_time_now, c1_stat, c1_mode, c1_prog_work_time
    , c1_percent_prog, c1_prog, axes_C1_X_float, axes_C1_Y_float,axes_C1_Z_float
    , axes_C1_B_float, axes_C1_C_float, axes_C1_S_float, formattedTimestamp];

return msg;
```

Query which will insert data into database:

```
insert into cnc_5_2 (motoc_hours_name,core_work_time_ttl_value,core_work_time_cur_value
                ,core_restart_cnt,channel_cnt,c1_tot_work_time,c1_work_time_now
                ,c1_stat,c1_mode,c1_prog_work_time,c1_percent_prog,c1_prog,axes_c1_x_float
                ,axes_c1_y_float,axes_c1_z_float,axes_c1_b_float,axes_c1_c_float
                ,axes_c1_s_float,time_load)
values ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16, $17, $18, $19);
```

2.3 Accumulation of sensor data

1. “mqtt” database was already created at the previous step for CNC 3.
2. Create “sens_h_2”, “sens_t_2”, “sensor_hump_2”, “sensor_temp_2” tables where data from 4 mqtt topics is going to be stored



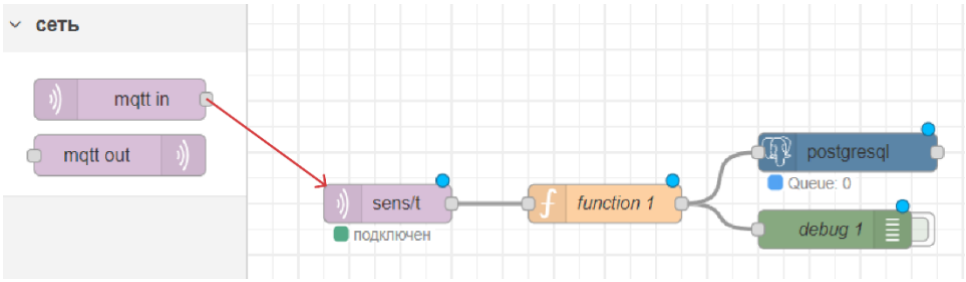
The DDL of the tables:

```
CREATE TABLE public.sens_h_2 (
hump numeric NULL,
time_load timestamp NULL
);

CREATE TABLE public.sens_t_2 (
temperature numeric NULL,
```

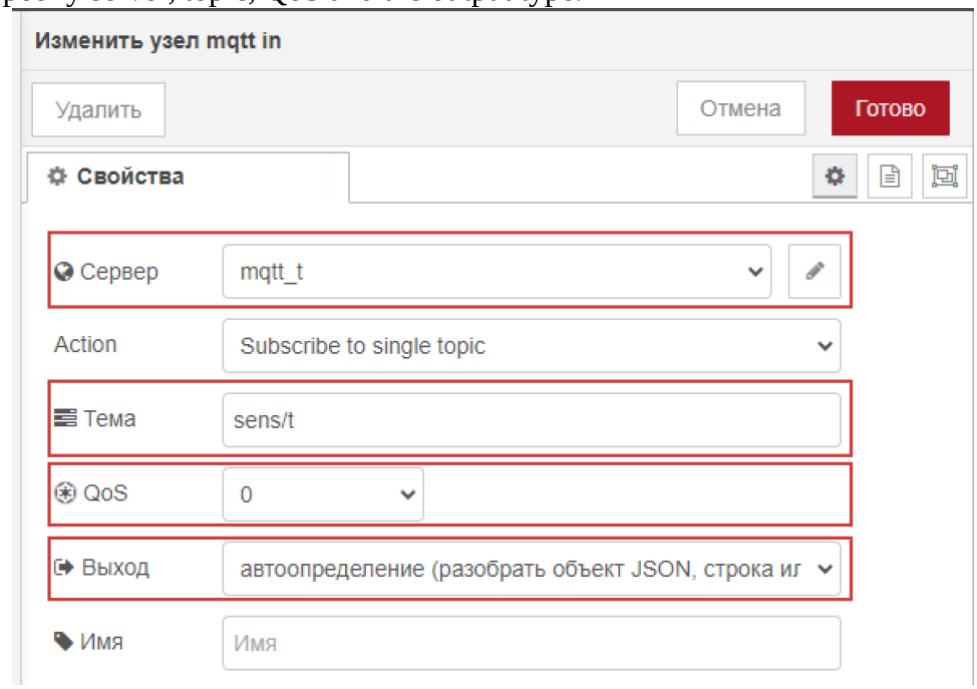
```
CREATE TABLE public.sensor_hump_2 (
hump numeric NULL,
time_load timestamp NULL
);

CREATE TABLE public.sensor_temp_2 (
temperature numeric NULL,
time_load timestamp NULL
);
```

In Node-RED:

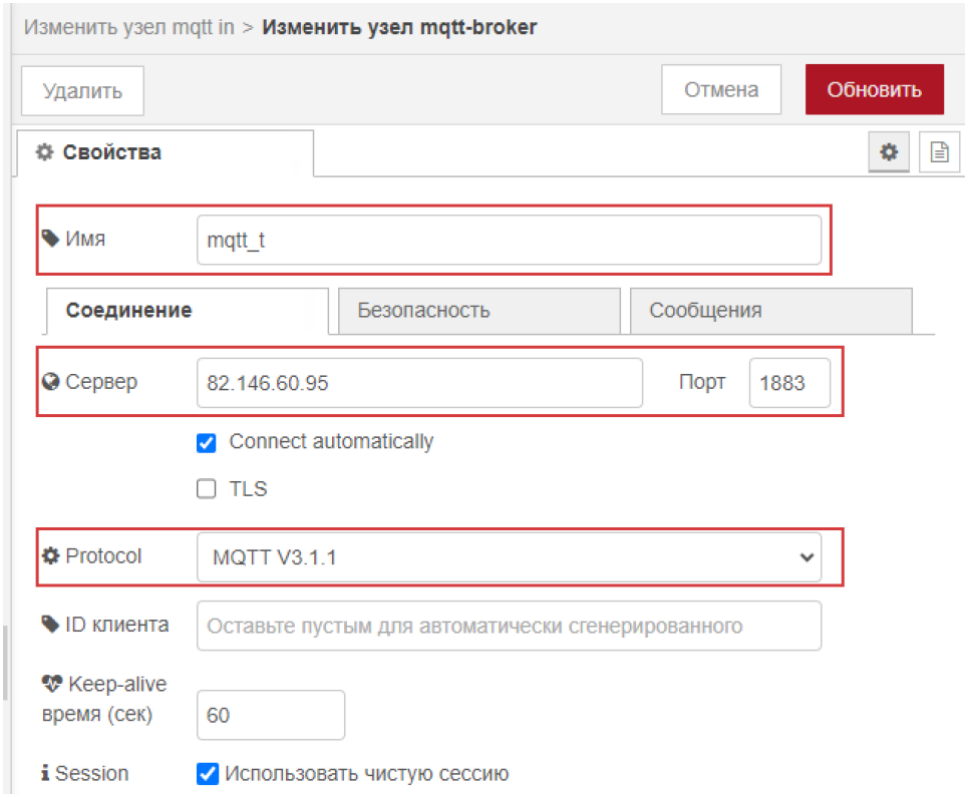
1. Add “mqtt in” element to subscribe to a topic.



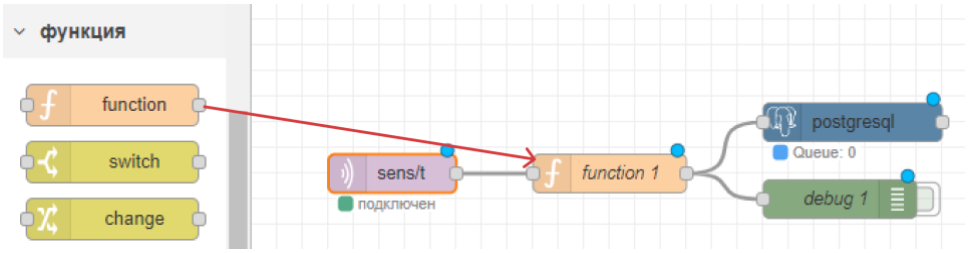
Specify server, topic, QoS and the output type.



For server set address, port, protocol.



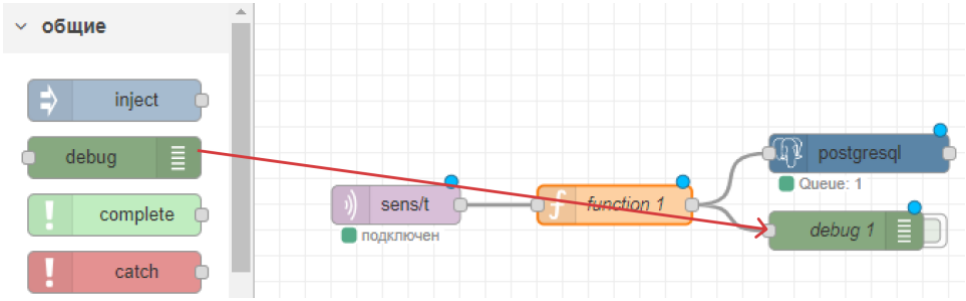
2. Add “function” element to write a function to get the value from mqtt and the timestamp when it was received.



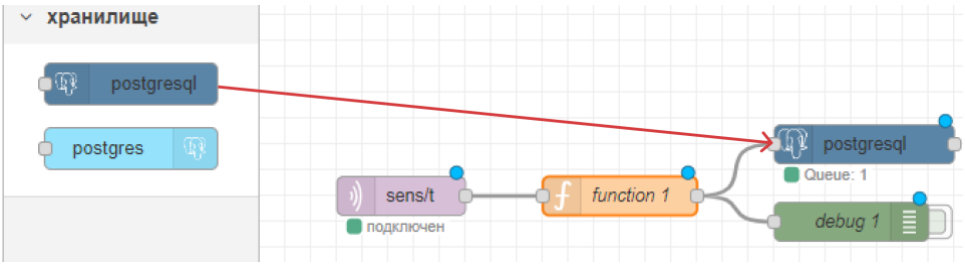
Function code is as follows:

```
const currentDateTime = new Date();
const formattedTimestamp = currentDateTime.toISOString()
  .replace('T', ' ')
  .replace('Z', '');

msg.params = [msg.payload, formattedTimestamp];
return msg;
```

3. Add “debug” element to check if everything is being read in a proper way.



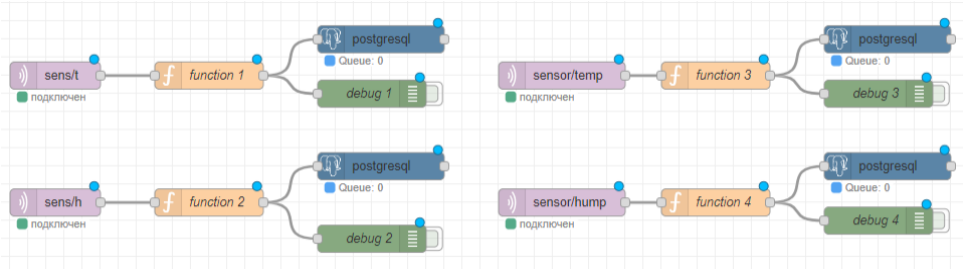
4. Add “postgresql” element to write parsed data into postgres database.



Query to insert data:

```
insert into sens_t_2 (temperature, time_load)
values ($1, $2);
```

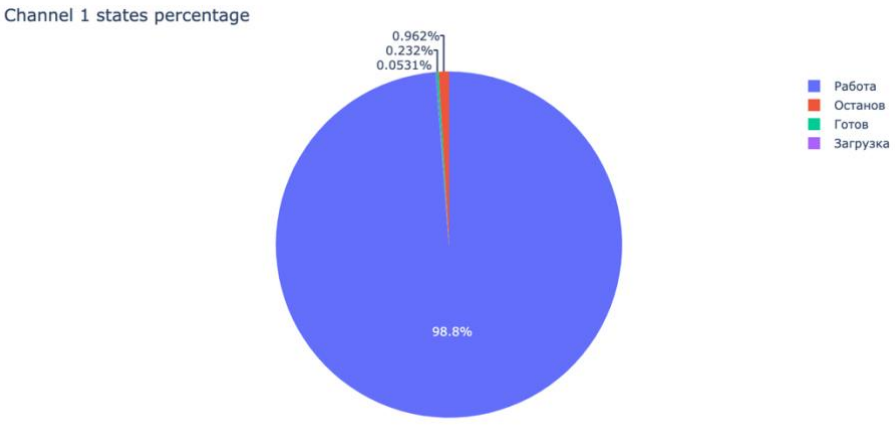
For the rest 3 topics do the same steps differing the table names where the data is going to be sent. As a result, get 4 chains of elements to read data from mqtt topics.



3 DATA ANALYSIS

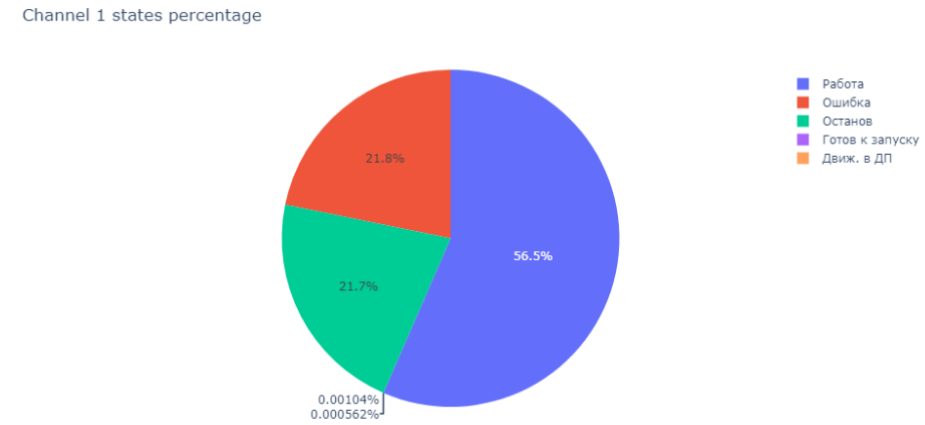
3.1 Calculate what state (Режим канала) the CNC was in the most time? (create diagram chart ->python, Excel, etc.)

CNC 3



The channel worked non-stop almost all the time.

CNC 5



As the pie chart does not show the full data, it would be more appropriate to look through it in a table format:

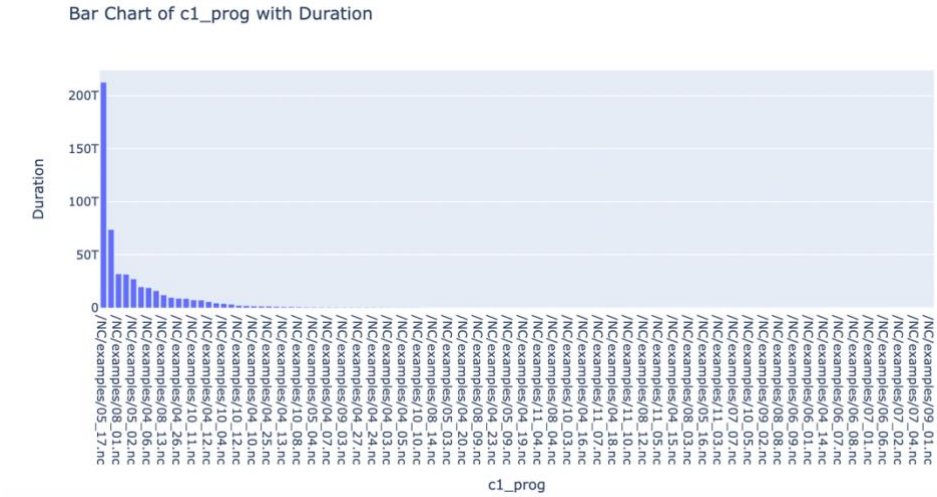
	c1_stat	cnt	duration
0	Готов к запуску	6	0 days 00:00:05.779000
1	Движ. в ДП	3	0 days 00:00:03.135000
2	Останов	12	1 days 09:35:58.190000
3	Ошибка	24375	1 days 09:45:46.579000
4	Работа	2078	3 days 15:32:28.933000

3.2 Which program (Программа) has been running the most time?

1. CNC3

	c1_prog	cnt	duration
44	/NC/examples/05_17.nc	3964	2 days 11:04:11.025000
107	/NC/examples/12_01.nc	502	0 days 20:30:52.628000
64	/NC/examples/08_01.nc	2733	0 days 08:55:21.287000
68	/NC/examples/08_05.nc	369	0 days 08:48:01.038000
29	/NC/examples/05_02.nc	803	0 days 07:34:46.518000
...	...	...	...
57	/NC/examples/07_02.nc	36	0 days 00:00:36.819000
103	/NC/examples/11_09.nc	36	0 days 00:00:36.495000
59	/NC/examples/07_04.nc	33	0 days 00:00:34.405000
82	/NC/examples/10_01.nc	26	0 days 00:00:22.758000
78	/NC/examples/09_01.nc	14	0 days 00:00:13.561000

111 rows × 3 columns



As it can be seen from the graph above the program "/NC/examples/05_17.nc " worked more than others (almost 2,5 days)

2. CNC5

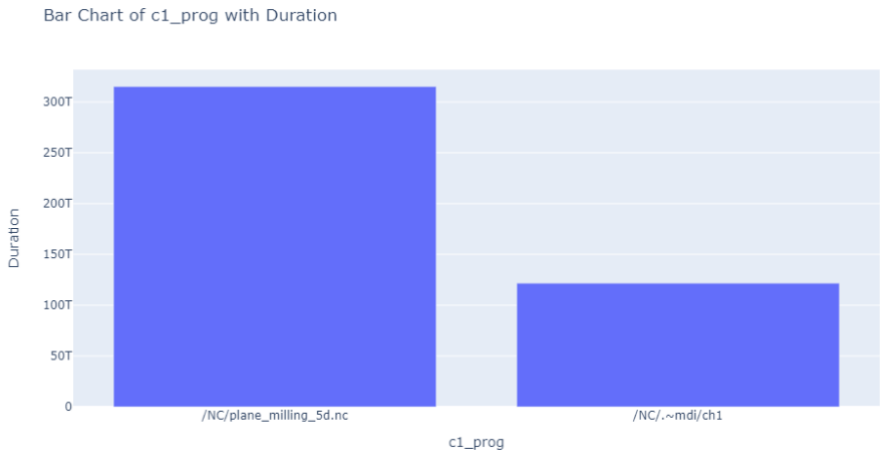
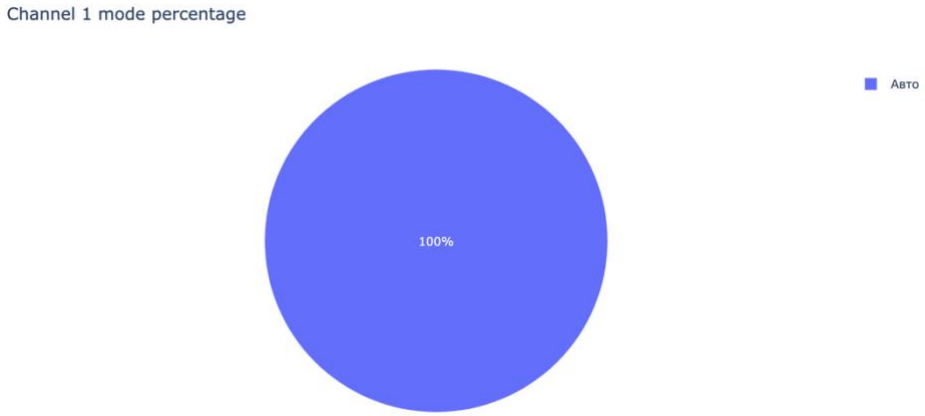


Table version:

	c1_prog	cnt	duration
1	/NC/plane_milling_5d.nc	2086	3 days 15:32:36.870000
0	/NC/.~mdi/ch1	24375	1 days 09:45:46.579000

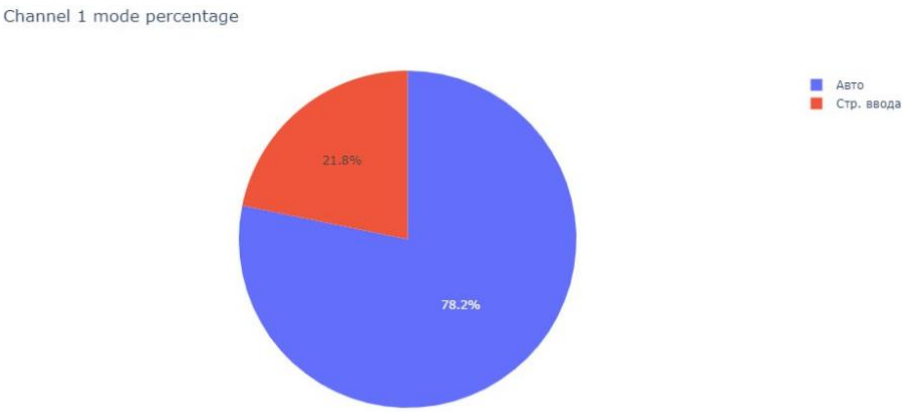
3.3 Calculate what mode (Режим канала) the CNC was in the most time? (create diagram chart)

1. CNC3 mode



The CNC3 worked in auto mode

2. CNC5:



3.4 Calculate how long the CNC worked (online) and how long it was offline?

1. The duration of work time of CNC3 (online and offline)

```
Total online duration per date:
timestamp_msc
2023-12-13   0 days 00:31:04
2023-12-14   0 days 00:02:10
2023-12-15   0 days 00:02:16
2023-12-16   0 days 08:07:22
2023-12-17   0 days 03:24:25
2023-12-18   0 days 05:07:05
2023-12-19   0 days 01:31:16
Name: duration, dtype: timedelta64[ns]

Total offline duration per date:
timestamp_msc
2023-12-13   0 days 22:53:23
2023-12-14   0 days 23:57:45
2023-12-15   0 days 23:57:41
2023-12-16   0 days 15:41:46
2023-12-17   0 days 20:31:50
2023-12-18   0 days 18:49:59
2023-12-19   0 days 09:56:53
Name: duration, dtype: timedelta64[ns]
```

Most of the time CNC3 was offline, but since December 16, the machine has been working longer compared to the previous days

2. CNC5

```
Total online duration: 0 days 07:09:52
Total offline duration: 6 days 03:38:58

If to make it more detailed:

Total online duration per date:
timestamp_msc
2023-12-13   0 days 00:31:02
2023-12-14   0 days 00:02:10
2023-12-16   0 days 00:00:03
2023-12-18   0 days 05:05:09
2023-12-19   0 days 01:31:28

Total offline duration per date:
timestamp_msc
2023-12-13   0 days 22:53:24
2023-12-14   0 days 23:57:45
2023-12-15   1 days 00:00:00
2023-12-16   0 days 23:59:57
2023-12-17   0 days 23:59:59
2023-12-18   0 days 18:50:59
2023-12-19   0 days 09:56:54
```

3.5 Find for all errors: time when errors occurred, real temperature and humidity, which program was running (from MQTT)

There were no errors during the work of CNC3

CNC 5

1. Errors Info:

	time_load	c1_prog	hump	temperature
0	2023-12-17 22:43:41.000	/NC/.~mdi/ch1	79	28
1	2023-12-17 22:43:42.000	/NC/.~mdi/ch1	68	28
2	2023-12-17 22:43:43.000	/NC/.~mdi/ch1	80	26
3	2023-12-17 22:43:44.000	/NC/.~mdi/ch1	71	25
4	2023-12-17 22:43:45.000	/NC/.~mdi/ch1	63	27
...	...	...	...	...
24454	2023-12-19 08:30:02.000	/NC/.~mdi/ch1	73	25
24455	2023-12-19 08:30:03.000	/NC/.~mdi/ch1	76	25
24456	2023-12-19 08:30:04.000	/NC/.~mdi/ch1	66	27
24457	2023-12-19 08:30:05.000	/NC/.~mdi/ch1	64	25
24458	2023-12-19 08:30:06.000	/NC/.~mdi/ch1	77	27

24459 rows × 4 columns

```
Programm: /NC/.~mdi/ch1
```

	hump	temperature
count	24459.000000	24459.000000
mean	69.987857	26.511223
std	6.059735	1.116811
min	60.000000	25.000000
25%	65.000000	26.000000
50%	70.000000	27.000000
75%	75.000000	28.000000
max	80.000000	28.000000

3.6 Find the date and time when the temperature value is equal to the humidity value (demo and real MQTT data)

There was no such value in the real sensor

The date and time when the temperature value is equal to the humidity value in the demo sensors:

	time_load	hump	temperature	diff
0	2023-12-19 09:05:16.000	26	26	0
1	2023-12-19 09:05:23.000	26	26	0
2	2023-12-19 09:05:36.000	29	29	0

4 NOTIFICATION (SEDA)

Before setting the notification flow in the Node red an email address from which the notification was going to be sent was created: manufacturingnremergency@gmail.com. In the settings for the mail, app passwords were generated/ so that Node Red would be able to log in to send notifications.

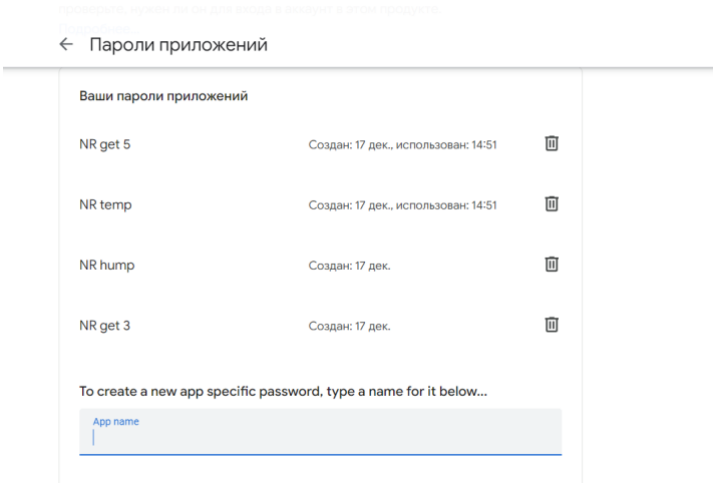


Figure 14. App passwords

4.1 Notifications for sensors

To make notifications for the sensors we did the following steps:

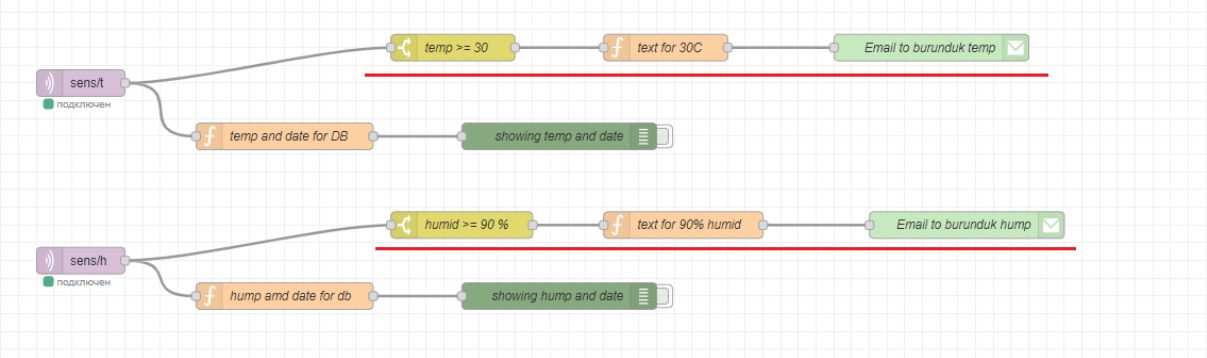


Figure 15. Sensor Notifications flow

1. To the real temperature (humidity) sensor, a switch was connected. Here we wrote the condition of the error >=30 C (>= 90%).

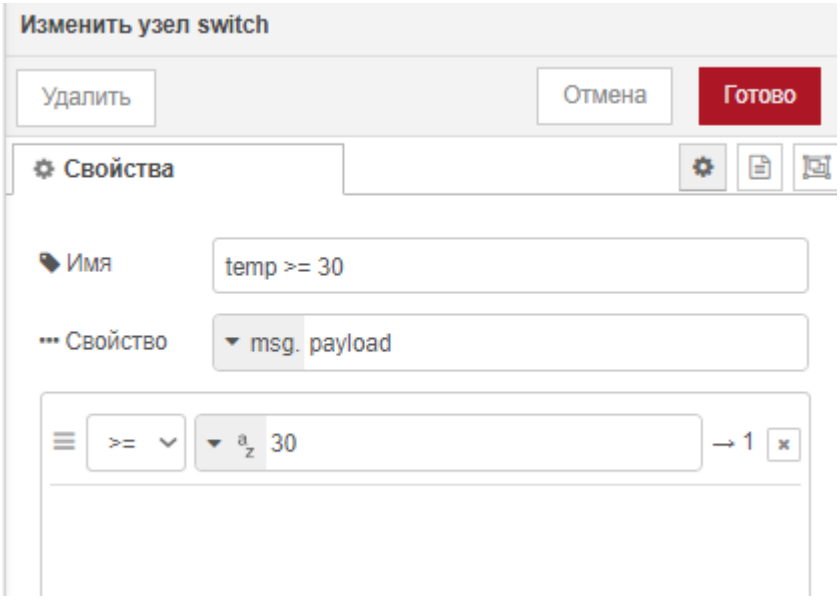


Figure 16. Switch for the temperature

2. After filtering the condition, a function was added to write the text and the topic of the email.
    a. Text for temperature: `"ATTENTION. The temperature is "+ temp +"C, this is more than (or equal to) 30C " + Date().toString(),`
    b. Text for humidity: `"ATTENTION. The humidity is " + hump + "%, this is more than (or equal to) 90% " + Date().toString(),`
    c. Topic: Emergency

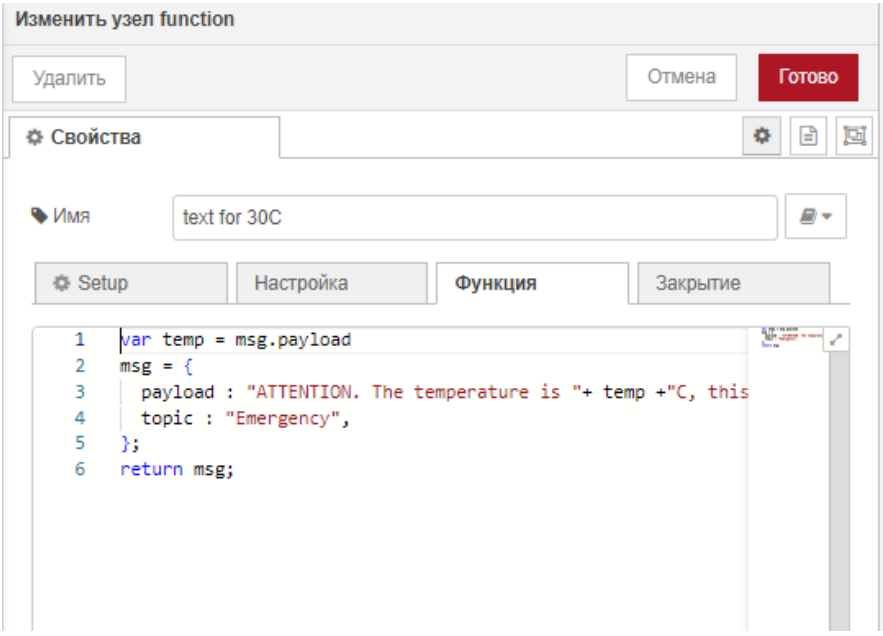


Figure 17. Email text and topic function for temperature

3. Lastly an e-mail node was created. In here the info on the receiving and sending email were written.

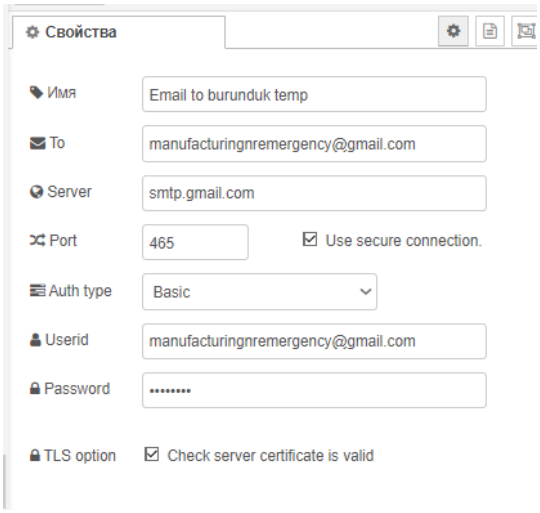


Figure 18. Email information

4. When the temperature (humidity) reaches its respective conditions, an email is sent to the manufacturingnremergency@gmail.com from itself.

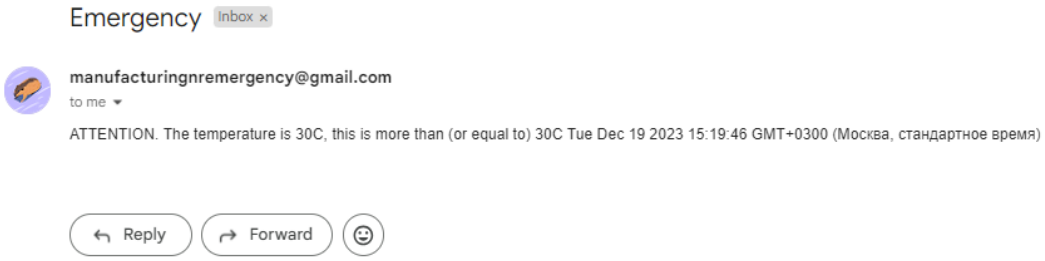


Figure 19. Email for temperature

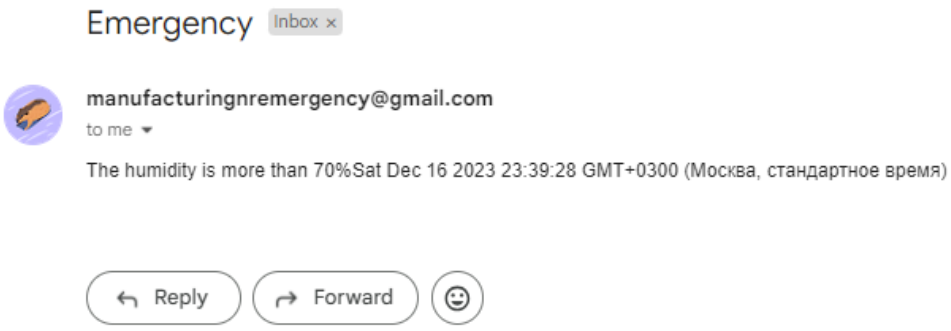


Figure 20. Email for humidity (at that time the condition was set to 70%)

4.1 Notifications for Error(ошибка) status

To make notifications for the errors in the CNC we did the following steps:

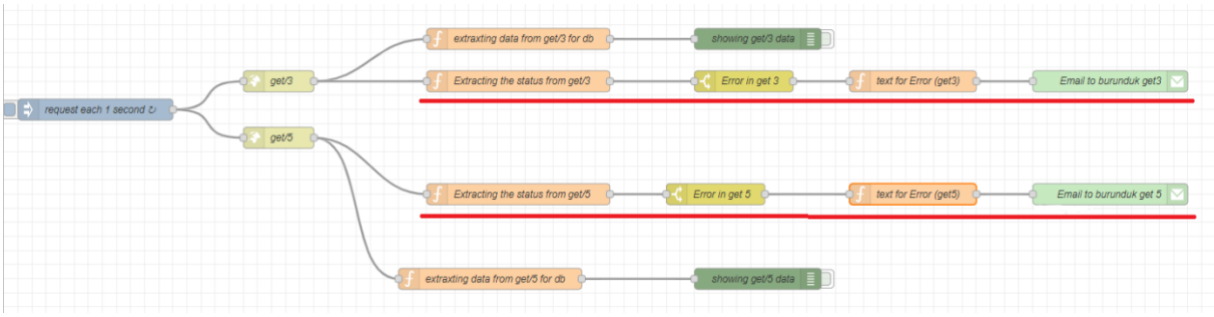


Figure 21. CNC Error notification flow

1. To the htttp request node a function node was attached. The function extracts the status data from the JSON file and saves it onto the msg.params.

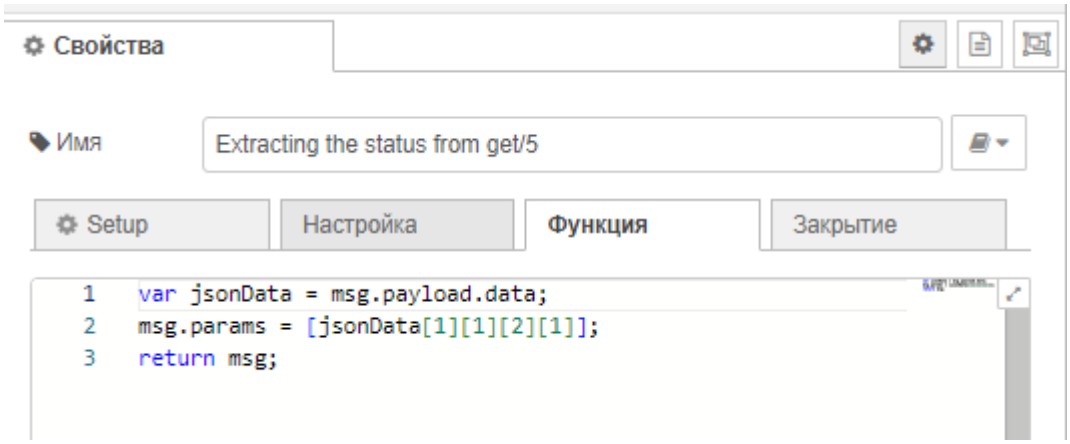


Figure 22 extracting of the status data

2. To the extracting status function a switch was added. The switch checks for the «Ошибка» status.

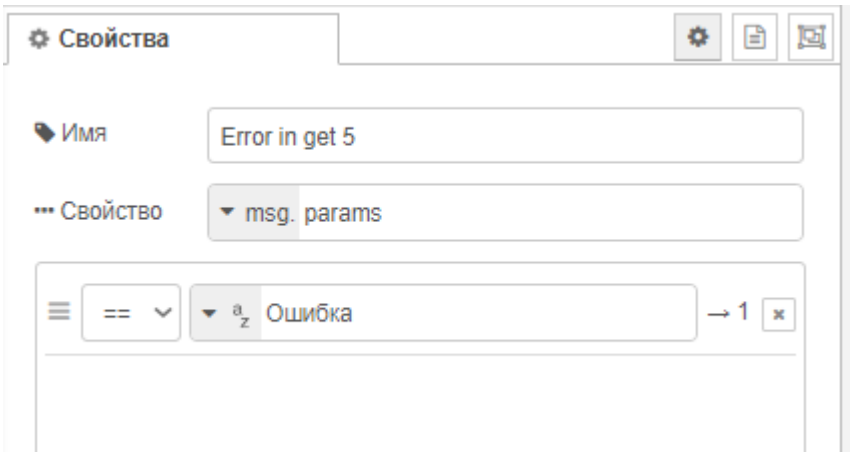


Figure 23 e Switch for the Error

3. After filtering the data an email node was connected. As mentioned above this node specifies the senders and receivers of the email.
    a. Text for CNC 5: "ATTENTION. there is an error in the 1-st channel (get5) " + Date().toString(),
    b. Text for CNC 3: `"ATTENTION. there is an error in the 1-st channel (get3) " + Date().toString(),`

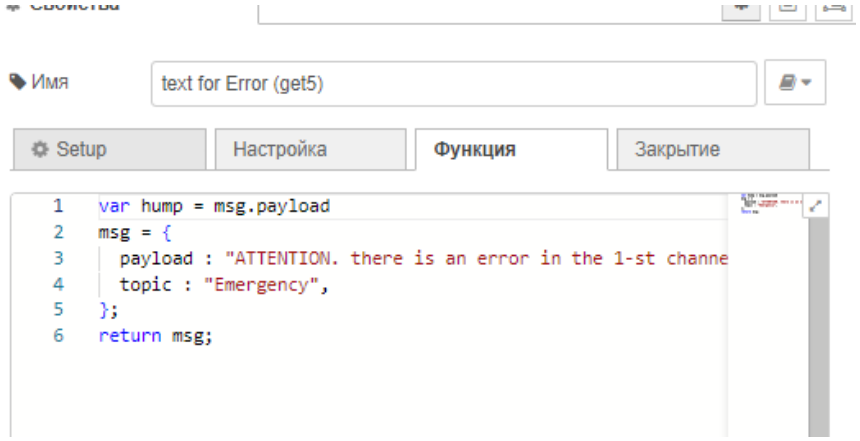


Figure 24. Email for the error status

4. When the status of the channel is error «Ошибка», an email is sent to the manufacturingnremergency@gmail.com from itself.

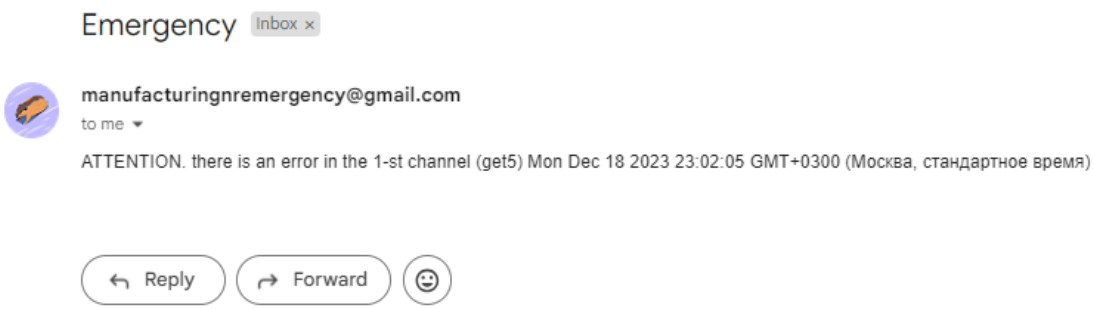


Figure 25. Email for the error status