

SECRETARIA DE EDUCAÇÃO E CIÊNCIA
INSTITUTO POLITÉCNICO DE BRAGANÇA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

LICENCIATURA EM ENGENHARIA DE INFORMÁTICA
6º PERÍODO

FERNANDO SOUZA FURTADO CARRILHO
JOSÉ RAFAEL SOARES BORGES

RELATÓRIO PRÁTICO 3:
NODE-RED, ESP e INFLUX-DB

BRAGANÇA

2023

FERNANDO SOUZA FURTADO CARRILHO
JOSÉ RAFAEL SOARES BORGES

RELATÓRIO PRÁTICO 3:
NODE-RED, ESP e INFLUX-DB

Este relatório objetiva a obtenção de nota na disciplina de Internet das Coisas dos graduandos no curso de Engenharia de Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança. Seu conteúdo é composto pela observação, descrição e aplicação referente ao Protocolo MQTT por intermédio do Node-RED, microcontrolador ESP e o gerenciador de dados em massa InfluxDB.

BRAGANÇA
2023

Sumário

1 Questão 02: InfluxDB	6
1.1 O que é o InfluxDB?	6
1.2 Time-Series Database	7
1.3 O que é Flux?	8
2 Questão 03	9
2.1 Questão 03 [c] - Função <i>range()</i> no FLUX	9
2.2 Questão 03 [d] - Função <i>filter()</i> no FLUX	11
2.3 Questão 03 [e] - Função <i>filter()</i> no FLUX com MEASUREMENT	12
2.4 Questão 03 [f] - Função <i>aggregateWindow()</i> no FLUX	14
3 Questão 07 - Temperatura e Humidade no InfluxDB	15
4 Questão 08 - Gráficos no InfluxDB	20
4.1 Gráfico dos valores de Temperatura	20
4.2 Gráfico dos valores de Humidade	21
4.3 Gráfico dos valores de Temperatura e Humidade na última hora	22
4.4 SingleStats: Gráfico max e min <i>AirQuality</i> Temperatura última hora	23
4.5 SingleStats: Gráfico max e min <i>AirQuality</i> Humidade última hora	24
4.6 SingleStats: Gráfico max e min <i>ExternalNode</i> Temperatura última hora . .	25
4.7 SingleStats: Gráfico max e min <i>ExternalNode</i> Humidade última hora	26
5 Questão 09 - Sistema com Node-RED, InfluxDB e ESP	27

Lista de Figuras

1	Função range() no FLUX	9
2	Função filter() no FLUX	11
3	Função filter() no FLUX com <i>measurement</i>	12
4	Função <i>aggregateWindow()</i> no FLUX	14
5	Flows do AirQuality e ExternalNode no Node-RED	15
6	Código JavaScript fluxo AirQuality Node-RED	16
7	Debug do Fluxo 1 do Node-RED	17
8	Configuração InfluxDB no Node-RED	17
9	InfluxDB do Fluxo 1 do Node-RED da Figura 5	18
10	InfluxDB do Fluxo 1 do Node-RED da Figura 5	19
11	Tópico <i>IPB/IoT/Lab/AirQuality</i> Temperatura 22.67	20
12	Tópico <i>IPB/IoT/Lab/ExternalNode</i> Temperatura 19.85	20
13	Tópico <i>IPB/IoT/Lab/AirQuality</i> Humidade 40.02%	21
14	Tópico <i>IPB/IoT/Lab/ExternalNode</i> Humidade 47.55%	21
15	Tópico <i>IPB/IoT/Lab/AirQuality</i> 20.43 °C e 50.98%	22
16	Tópico <i>IPB/IoT/Lab/ExternalNode</i> 23.44 °C e 42.78%	22
17	Tópico <i>IPB/IoT/Lab/AirQuality</i> valor máximo temperatura	23
18	Tópico <i>IPB/IoT/Lab/AirQuality</i> valor mínimo temperatura	23
19	Tópico <i>IPB/IoT/Lab/AirQuality</i> valor máximo humidade	24
20	Tópico <i>IPB/IoT/Lab/AirQuality</i> valor mínimo humidade	24
21	Tópico <i>IPB/IoT/Lab/ExternalNode</i> valor máximo temperatura	25
22	Tópico <i>IPB/IoT/Lab/ExternalNode</i> valor mínimo temperatura	25
23	Tópico <i>IPB/IoT/Lab/ExternalNode</i> valor máximo humidade	26
24	Tópico <i>IPB/IoT/Lab/ExternalNode</i> valor mínimo humidade	26
25	Temperatura no External Node pelo InfluxDB	27
26	Fluxo no Node-RED com conexão ao InfluxDB	28
27	Configuração Repeat do Inject no Node-Red	28
28	Dashboard Node-RED do nó <i>numeric</i>	29

29	Script da query no InfluxDB	29
30	Configuração da abertura da Janela no Node-RED	30
31	Saída do valor do fluxo em OPEN	31
32	ESP LED azul ativo igual a janela aberta	31
33	Configuração para o LED azul ativar no Node-RED	32
34	Saída do valor do fluxo em CLOSE	32
35	Caption	33
36	Código inicial do ESP32	33
37	Função connectWifi ESP32	34
38	Função brokerReconnect ESP32	34
39	Função callback ESP32	35
40	Função setup e void ESP32	35

1 Questão 02: InfluxDB

Neste tópico será abordado o que é o InfluxDB e suas relações. Seu intuito é compreender de maneira abrangente o que é essa ferramenta e quais são suas vantagens e utilidades, com três questionamentos: (1) O que é o InfluxDB, (2) Time-series database e (3) O que é Flux.

1.1 O que é o InfluxDB?

De acordo com a definição do site ServerDoin (ServerDoin, 2023), o InfluxDB é um banco de dados de código aberto, com intuito de trabalhar com alto volume de dados, consultas e escritas por segundo, sem causar forte impacto no sistema operacional, no qual está atuando.

Por conseguinte, esse site afirma ainda que, no InfluxDB há a distribuição de ferramentas para gestão como o Chronograf, Kapacitor e o Telegraf, a exemplos. Diante disso, o InfluxDB também é provido de sintaxe semelhante ao SQL, para consultas à base de dados.

Por fim, dessa forma, o InfluxDB, banco de dados, também pode ser usado por uma interface HTTP, sendo possível criar, excluir, escrever, modificar, entre outras operações, com uma base de dados, isto é, realizar o CRUD (Create, Read, Update e Delete - traduzido: criar, ler, alterar e deletar).

1.2 Time-Series Database

Para o sítio InfluxData (InfluxData, 2023), o **Time-Series DataBase** (TSDB), traduzido, Banco de dados temporal, é um banco de dados otimizado para dados com carimbo de data e/ou hora, ou até mesmo série temporal.

Para este site, os dados de série temporal são medições ou eventos nos quais são rastreados, monitorados, reduzidos e agregados ao percorrer do tempo. Consoante, o TSDB podem ser métricas de servidor, monitoramento de desempenho de aplicativos, dados de rede, dados de sensores, eventos, cliques, negociações em um mercado e dentre outros tipos de dados analíticos.

À vista disso, um banco de dados de séries temporais é criado com o intuito de lidar com métricas e eventos ou medições com registro de data e hora. Dessa forma, o TSDB é otimizado para medir as mudanças ao longo do tempo percorrido e apresentar as análises ao usuário.

Em sequência, são os gerenciamentos de ciclos de vida dos dados, varreduras de grande intervalo de registro, e resumo, as propriedades que tornam os dados de séries temporais muito diferentes de outras cargas de trabalho de dados. Ademais, mesmo não sendo novos, os TSDBs, teve a sua primeira geração atuante no aspecto de análise de dados do mercado financeiro.

Por fim, por quase tudo estar conectado à um sensor, usar seus dados, ou usufruir de seus serviços, tais como, carros, redes elétricas, telefone e semelhantes, agora, mais que nunca, com o fluxo incansável de métricas e eventos ou até mesmo dados de séries temporais, os TSDBs são fundamentais.

1.3 O que é Flux?

De maneira simplificada, o Flux lang é uma linguagem de script de dados funcional projetada para consultar, processar, escrever, analisar e atuar em dados do InfluxDB e muitas outras fontes, como bancos de dados SQL (Big Query, PostgreSQL, MS SQL Server), CSVs anotados, JSON e Bigtable (Influx Data, 2023).

Por fim, é válido apontar que o Flux têm, aproximadamente, 400 funções disponíveis na biblioteca padrão, as quais podem corroborar na recuperação, transformação, processamento e na geração de dados.

2 Questão 03

Nesta sequência deste relatório, nos subtópicos, da Questão 03, serão abordados algumas funções de operação fundamental na ferramenta FLUX, tais como: *range()*, *filter()* e *aggregateWindow()*. O intuito é conhecer melhor as funcionalidades tidas como fundamentais para atividades posteriores.

2.1 Questão 03 [c] - Função *range()* no FLUX

Neste exercício, é pedido que se utilize o Flux, acesse-o [aqui](#), para que, após isso, acesse a seção “Script Editor” e então que se a função *range()* para filtrar dados, com base em limites de tempo. Neste caso, para o filtro, foi considerado o intervalo de tempo em: início, *start* (2019-08-22T00:00:00Z) e parada, *stop* (2019-08-25T00:00:00Z).

Por conseguinte, feito as ações do parágrafo anterior, pedido pela atividade, encontra-se os resultados da Figura 1.



Figura 1: Função *range()* no FLUX

À vista disso, conforme é visível na Figura 1, no Flux a função *range()* com o intervalo dado pela atividade, de início (*start*) e parada (*stop*), o explorador de dados (data

explorer) do Flux apresentou as seguintes acima elucidadas.

Por fim, ao selecionar em um dado ponto no gráfico, no explorador de dados do Flux, ver-se-á os valores, na data e horário apontado, do campo de temperatura, pH, nível de água e do índice.

2.2 Questão 03 [d] - Função *filter()* no FLUX

Consoante, no atual exercício é desejado que ao utilizar a função *filter()* que se filtre os dados com base na localização das estações. Diante disso, a Figura 2, abaixo expressa, presenta apenas os dados da estação de Santa Mônica no intervalo de tempo definido anteriormente, da Figura 1.



Figura 2: Função *filter()* no FLUX

A partir da Figura 2, acima apresentada, é possível ver os valores de índice, da temperatura, pH e nível da água, captada por sensores, na localidade de Santa Mônica, no intervalo de tempo especificado na atividade anterior.

Por fim, ainda na Figura 2, repare que a função, dentro do *script* do Flux, é mantido a função *range()* juntamente com a função *filter()*, pois as duas em conjunto cumpre os requisitos solicitados no atual exercício.

2.3 Questão 03 [e] - Função *filter()* no FLUX com MEASUREMENT

Em sequência, com base na utilização da função *filter()*, do exercício anterior, para filtrar dados com base nos parâmetros medidos, este exercício perfaz uma especificação mais atenuada dos valores transmitidos pelo filtro.

E isso implica, que dos valores arrecadados na Figura 2, a especificação, isto é, a outra variável de seleção, inserida no atual contexto, é a variável *measurement*. Isto implica que, apenas o argumento passado nesta variável que terá seu valor aplicado e apresentado do explorador de dados do Flux.

E para melhor compreender este cenário, a Figura 3, abaixo expressa essa realidade.

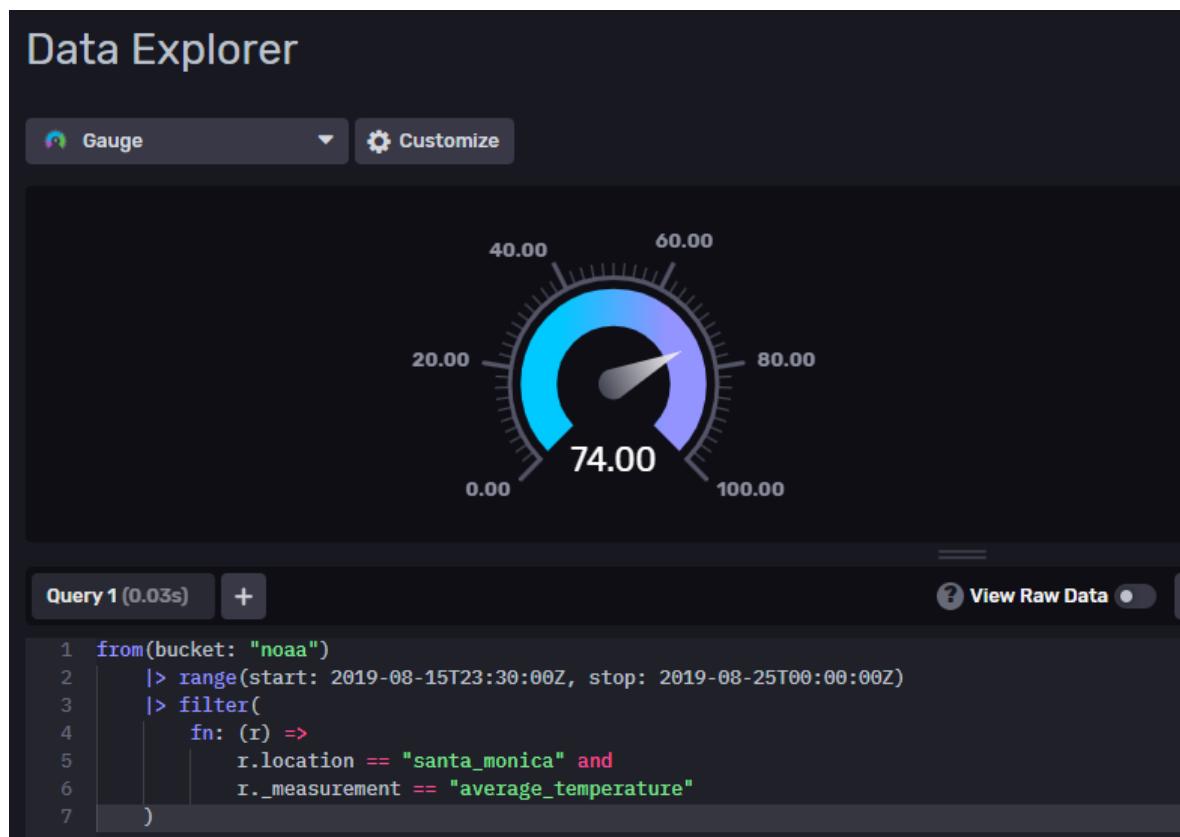


Figura 3: Função *filter()* no FLUX com *measurement*

Observe que na Figura 3, apenas a média da temperatura da água que é apresentado, enquanto na Figura 2, são apresentados os valores de: média de temperatura, qualidade, pH e o nível da água.

Destarte, o motivo disso, do gráfico ser apenas em média da temperatura, da Figura 3,

é encontrado ao passo em que se olha na imagem acima, na qual a variável *measurement*, tem como argumento o *average_temperature*, traduzido em, média da temperatura. Por esse motivo, apenas o valor da média da temperatura encontra-se apresentado na Figura 3.

Por fim, em continuidade, repare que a apresentação dos dados da Figura 3, é diferente da Figura 2 e Figura 1. Isto é dado ao ponto em que as apresentações das duas primeiras (Figura 1 e Figura 2), a apresentação dos dados é expresso na modalidade gráfico (*graph*), enquanto o da Figura 3 é obtido em medidor (*gauge*), em questão, no valor 74.

2.4 Questão 03 [f] - Função *aggregateWindow()* no FLUX

Em seguida, este exercício solicita o uso da função *aggregateWindow()* para agrregar dados. Neste caso em específico, com base no que foi realizado nos itens anteriores, é desejado que se mostre os valores máximos (*fn:max*) de temperatura a cada intervalo de uma hora (*every:1h*).

Dessa forma, com base nisso, após os ajustes dos exercícios anteriores, em adequação à função *aggregateWindow()*, é possível ver que com os valores apresentados na Figura 4, elucidada abaixo.

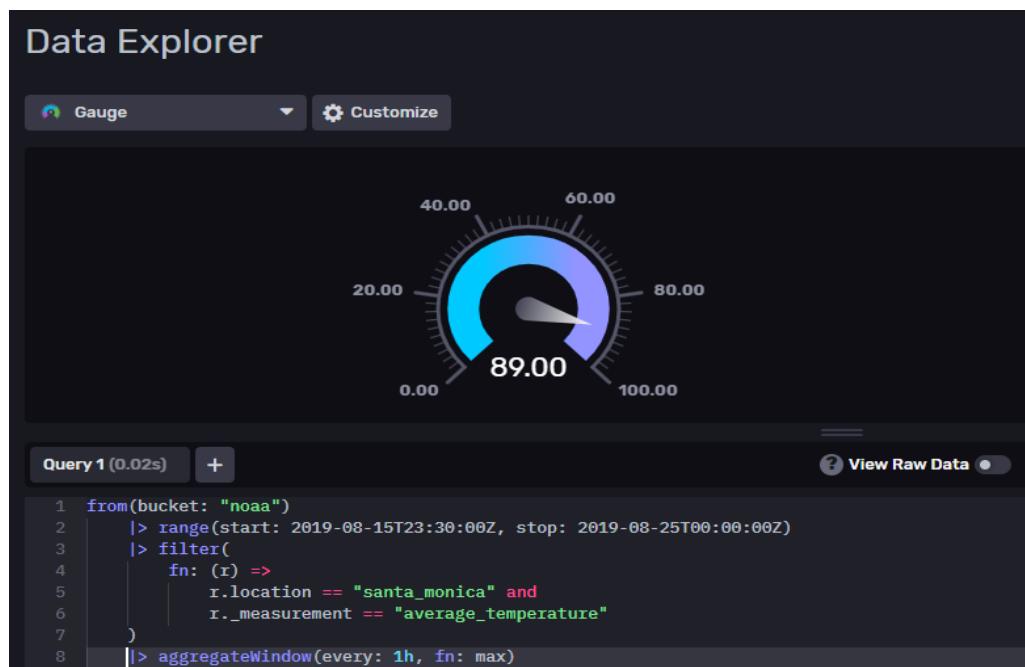


Figura 4: Função *aggregateWindow()* no FLUX

Em continuidade, é bom salientar que na Figura 4 no intervalo de uma hora o valor máximo da média da água foi em 89, na localidade de Santa Mônica em que o intervalo de tempo lido foi de início, *start* (2019-08-22T00:00:00Z) e parada, *stop* (2019-08-25T00:00:00Z).

Por fim, percebe-se que a função *aggregateWindow()* é utilizada para agregação de valores, com intuito de encontrar valores, respostas, ainda mais especificadas, o que fornece ao usuário informações mais precisas, ou próximas a isso, o que é torna válido o uso do FLUX para alto nível de dados.

3 Questão 07 - Temperatura e Humidade no InfluxDB

Para receber os dados lidos pelos sensores instalados, através do Node-Red construi-se os conjuntos de nós para o captamento dos valores de temperatura e humidade de ambos os tópicos (*IPB/IoT/Lab/AirQuality* e *IPB/IoT/Lab/ExternalNode*).

Em ambos os flows, converte-se os valores lidos para um objeto *JSON*, para mais fácil interpretação e tratamento. Para demonstar, ambos os flows estão apresentados na Figura 5, abaixo expressa.

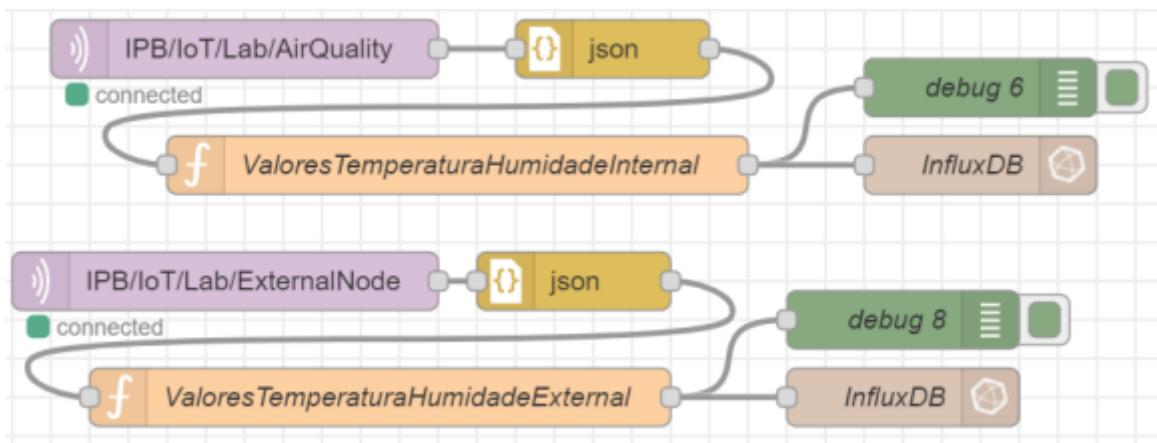


Figura 5: Flows do AirQuality e ExternalNode no Node-RED

À vista disso, repare na Figura 5, que há dois flows, fluxos, o primeiro iniciado, da esquerda para a direita, pelo bloco roxeado de nome *IPB/IoT/Lab/AirQuality*, enquanto o segundo flow, fluxo, é o segundo, de cima para baixo, de nome, da esquerda para a direitira, roxeado, *IPB/IoT/Lab/ExternalNode*.

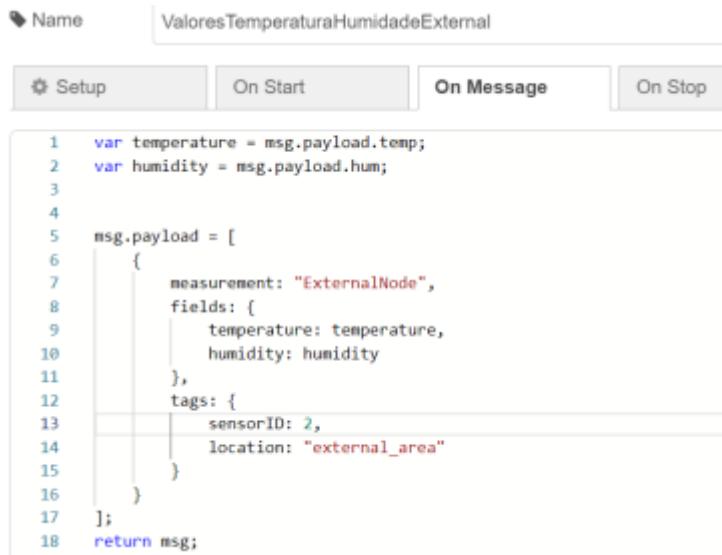
Em seguida, para cada flow executamos o nó de função para filtrar apenas os valores de temperatura e humidade dos dados recebidos de cada um dos tópicos, guardando-os em 2 variáveis de temperatura e humidade.

Consoante, o código cria um array, contendo um objeto JSON na propriedade "*msg.payload*", que contém informações sobre a temperatura e humidade de ambos os tópicos.

Dessa forma, esse objeto tem três propriedades: "*measurement*", que indica a medida sendo feita (no caso, para cada flow, os valores de "*AirQuality*" e "*ExternalNode*"), "*fields*", que contém informações sobre os dados sendo medidos (no caso, temperatura e humidade)

e ”*tags*”, que contém informações adicionais sobre a medição (no caso, ID do sensor e localização).

A seguir, na Figura 6, consta o código no qual retorna a mensagem modificada com o novo objeto *JSON* na propriedade ”*msg.payload*”, que pode ser processada pelo próximo nó no fluxo.



The screenshot shows a Node-RED node configuration window for an 'On Message' node. The node has the name 'ValoresTemperaturaHumidadeExternal'. It has four tabs: 'Setup', 'On Start', 'On Message' (which is selected), and 'On Stop'. The code in the 'On Message' tab is:

```

1 var temperature = msg.payload.temp;
2 var humidity = msg.payload.hum;
3
4
5 msg.payload = [
6   {
7     measurement: "ExternalNode",
8     fields: {
9       temperature: temperature,
10      humidity: humidity
11    },
12    tags: {
13      sensorID: 2,
14      location: "external_area"
15    }
16  }
17];
18 return msg;

```

Figura 6: Código JavaScript fluxo AirQuality Node-RED

Em resumo, o código da Figura 6 extrai dados de temperatura e umidade de uma de dados coletados por sensores, cria um array, contendo o objeto *JSON* com esses dados e retorna uma nova mensagem modificada para ser processada pelo próximo nó no fluxo.

E é com este intuito, que a partir do primeiro fluxo da Figura 5, ao adicionar o nó Debug, é possível ver o resultado do fluxo de acordo com a Figura 7, abaixo elucidada.

Em seguida, depois de construir o array contendo as informações, é necessário enviar para o Influx. Em continuidade, no node-red configurou-se para cada um dos flows, de acordo como é expresso na Figura 8, na próxima página, o nó influx batch node, identificando o endereço do servidor, o bucket, onde os dados serão armazenados e o token privado associado com a nossa conta, para haver ligação.

Em sequência, os dados armazenados no InfluxDB, com eles, é possível realizar diversas operações e análises. É possível filtrar e selecionar dados com base em diferentes critérios,

```

14/04/2023, 10:11:01 node: debug 8          14/04/2023, 10:11:05 node: debug 6
IPB/IoT/Lab/ExternalNode : msg.payload : array[1] IPB/IoT/Lab/AirQuality : msg.payload : array[1]
  ▼array[1]                                ▼array[1]
    ▼0: object                            ▼0: object
      measurement: "ExternalNode"        measurement: "AirQuality"
      ▼fields: object                  ▼fields: object
        temperature: 19.85              temperature: 22.67
        humidity: 47.55                humidity: 40.33
      ▼tags: object
        sensorID: 2                   sensorID: 1
        location: "external_area"     location: "internal_area"

```

Figura 7: Debug do Fluxo 1 do Node-RED

como tempo, tags e campos, e realizar operações de agregação e redução de dados, como média, soma, mínimo e máximo.

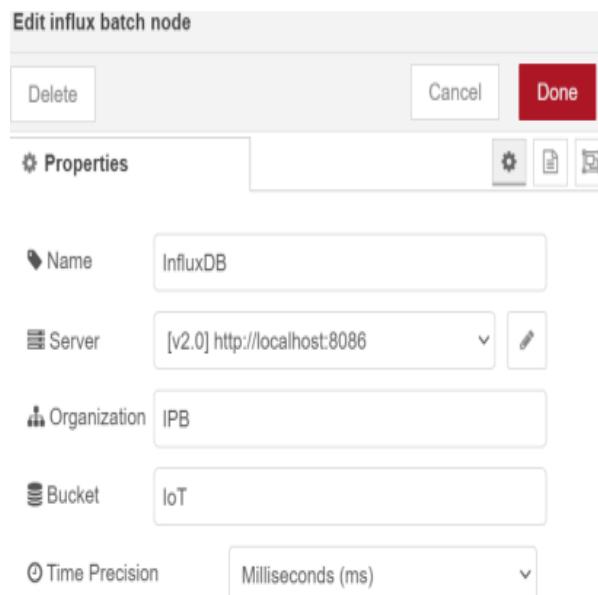


Figura 8: Configuração InfluxDB no Node-RED

Não distante, é possível também visualizar e monitorar os dados em tempo real, criar gráficos e dashboards, e realizar análises preditivas para identificar possíveis problemas e tendências futuras com antecedência. O InfluxDB é compatível com diversas ferramentas e tecnologias, permitindo a integração com outras soluções de monitoramento e análise de dados.

E assim, essas operações e análises permitem que se obtenha insights mais precisos e tome decisões mais informadas com base nos valores coletados.

Em seguida, pelo Fluxo 1 da Figura 5, é possível atuar no *InfluxDB* e encontrar a Figura 9, a qual expressa no Bucket *IoT*, *measuroment "AirQuality"*, filtro *humidade e temperatura*, filtro *internal_area* e pelo *aggregateWindow*; seu valor é de: 42.03 para a humidade e 23.36 para a temperatura.

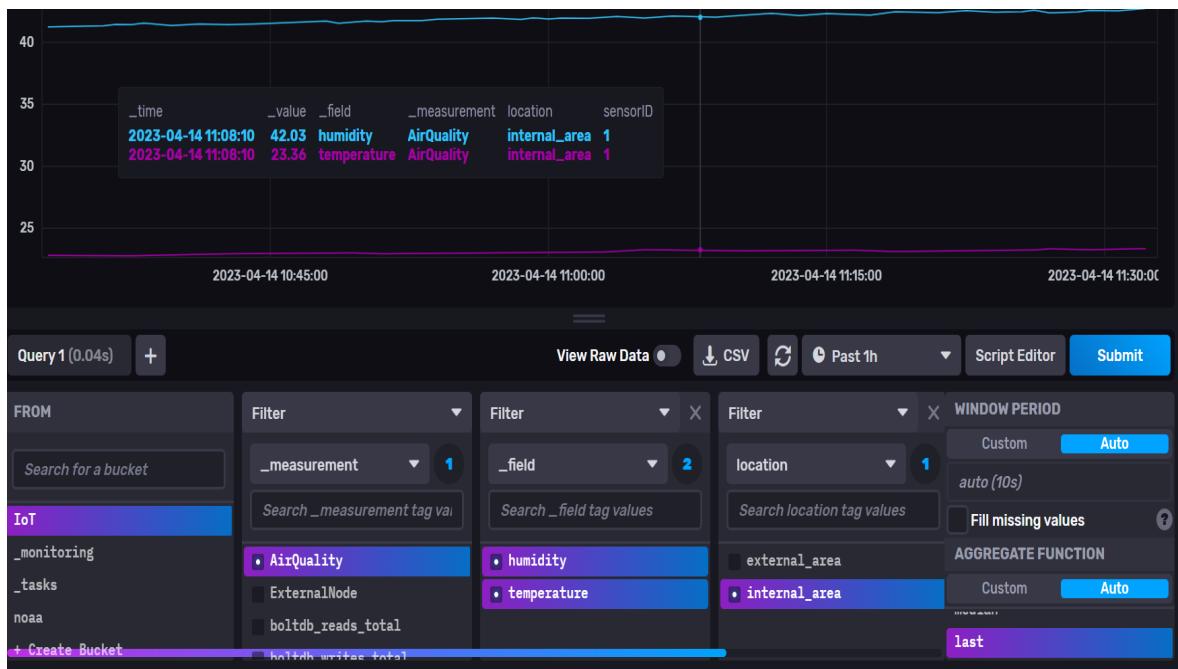


Figura 9: InfluxDB do Fluxo 1 do Node-RED da Figura 5

Dessa forma, enquanto a Figura 9, acima expressa salienta sobre o Fluxo 1 da Figura 5, a Figura 10 apresenta o Fluxo 2 da mesma Figura 5.

Em sequência, note que ambos os fluxos, da Figura 5, estão singularmente representados. Isto é, o primeiro e o segundo fluxos contém seus próprios nós e no *InfluxDB* é possível pegar seus dados separadamente.

E é com vista à isso que a Figura 9 e a Figura 10, representam os resultados de ambos os fluxos, respectivamente, Fluxo 1 e Fluxo 2, da Figura 5. Veja que, como o Node-RED captura os dados em tempo real, o *InfluxDB* captura os dados e gera o gráfico do momento real e apresenta de acorco com as configurações aplicadas no *query* do banco de dados do próprio *Influx*.

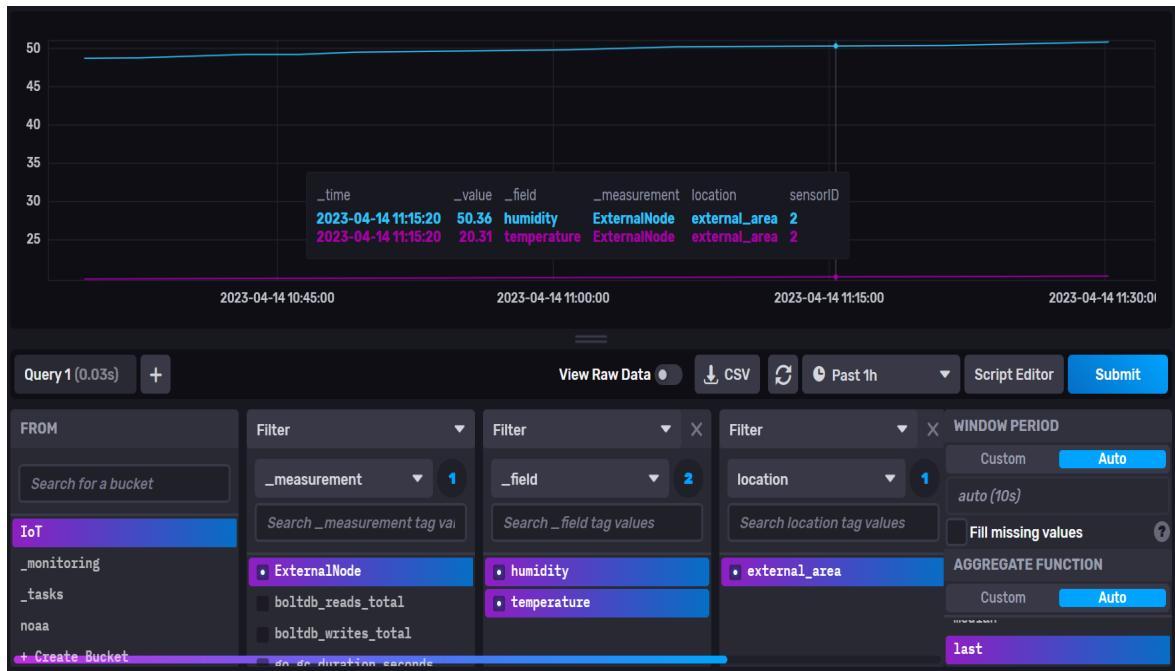


Figura 10: InfluxDB do Fluxo 1 do Node-RED da Figura 5

Por fim, conforme vê-se acima o resultado no *InfluxDB* do Fluxo 2 da Figura 5, é possível implicar no *InfluxDB* e encontrar a Figura 9, a qual expressa no Bucket *IoT*, *measuroment "ExternalNode"*, filtro *humidade e temperatura*, filtro *external_area* e pelo *aggregateWindow*; seu valor é de: 50.36 para a humidade e 20.31 para a temperatura.

4 Questão 08 - Gráficos no InfluxDB

Nesta atividade, as questões serão expressas unicamente pela imagem e por sua descrição no qual há a resposta. Diante disso, essa ação é tida pela intenção de ter uma resposta direta e objetiva, uma vez que não é necessário haver textos explicativos para as imagens, sendo que o exercício os pede visualmente.

4.1 Gráfico dos valores de Temperatura

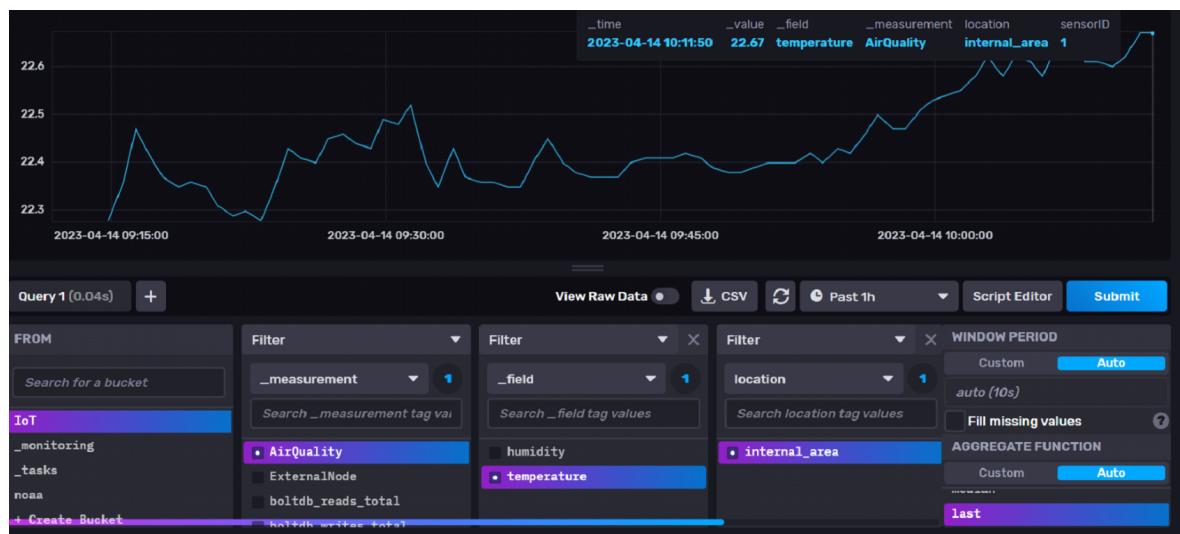


Figura 11: Tópico *IPB/IoT/Lab/AirQuality* Temperatura 22.67

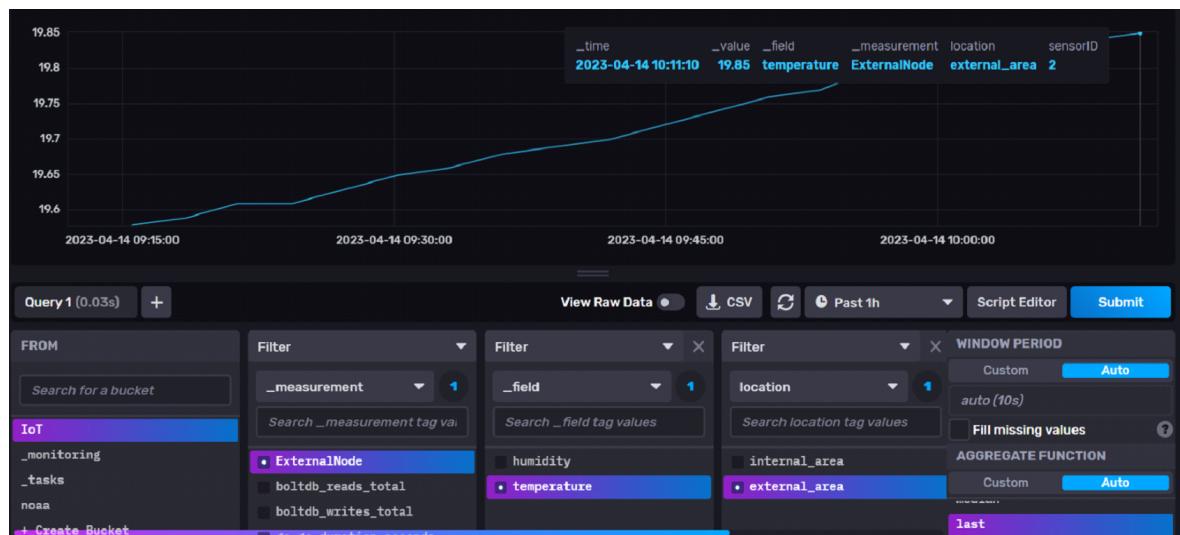


Figura 12: Tópico *IPB/IoT/Lab/ExternalNode* Temperatura 19.85

4.2 Gráfico dos valores de Humidade

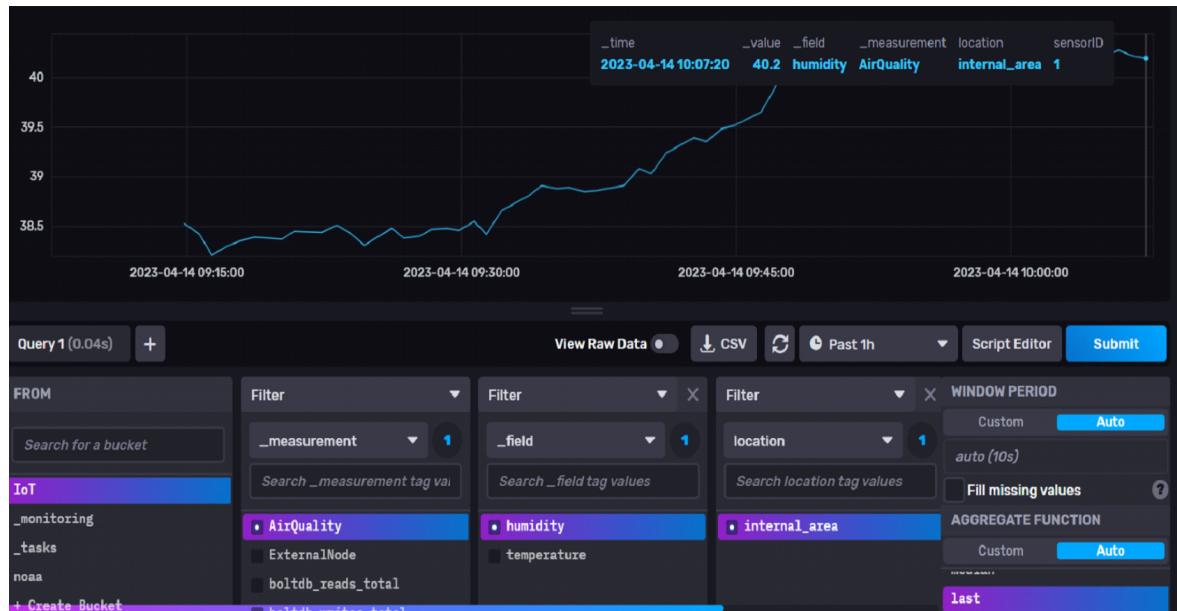


Figura 13: Tópico *IPB/IoT/Lab/AirQuality* Humidade 40.02%

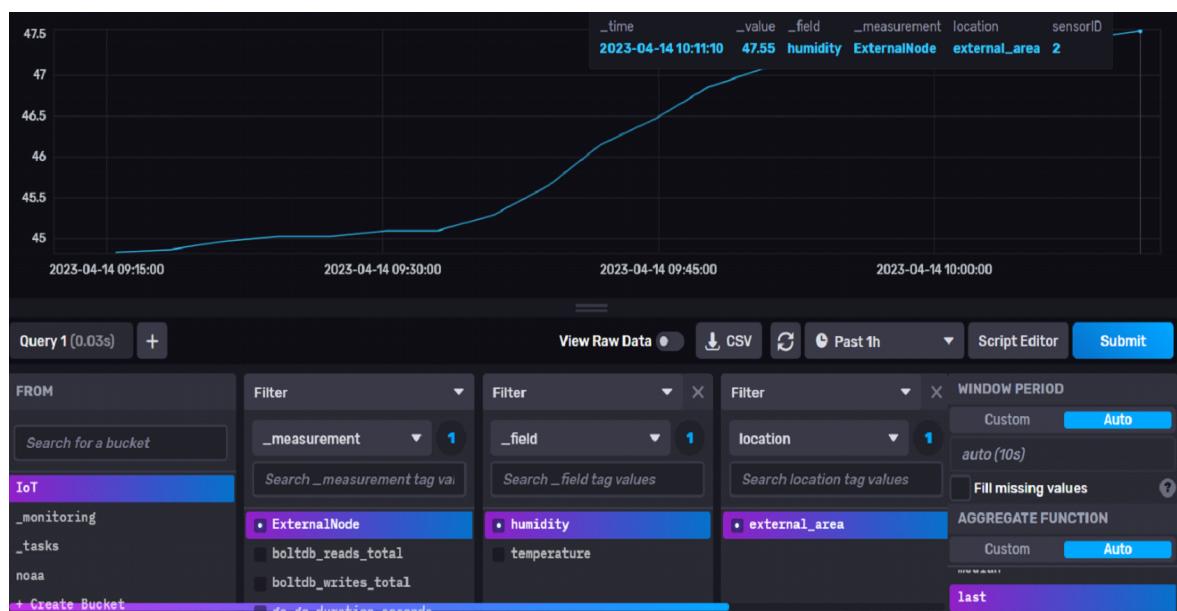


Figura 14: Tópico *IPB/IoT/Lab/ExternalNode* Humidade 47.55%

4.3 Gráfico dos valores de Temperatura e Humidade na última hora

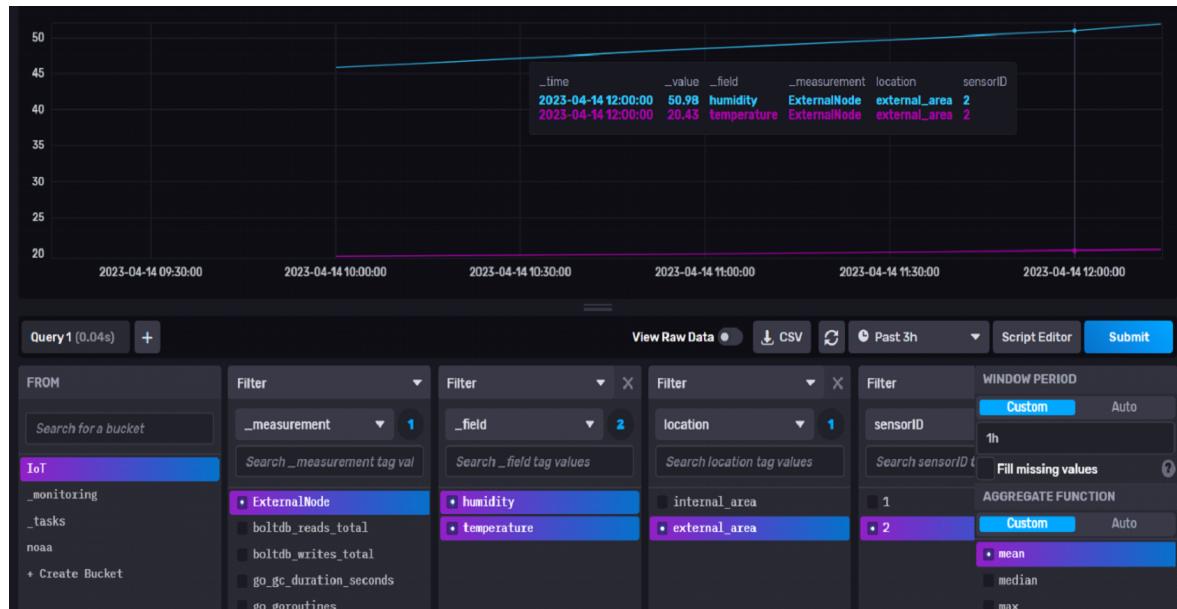


Figura 15: Tópico *IPB/IoT/Lab/AirQuality* 20.43 °C e 50.98%

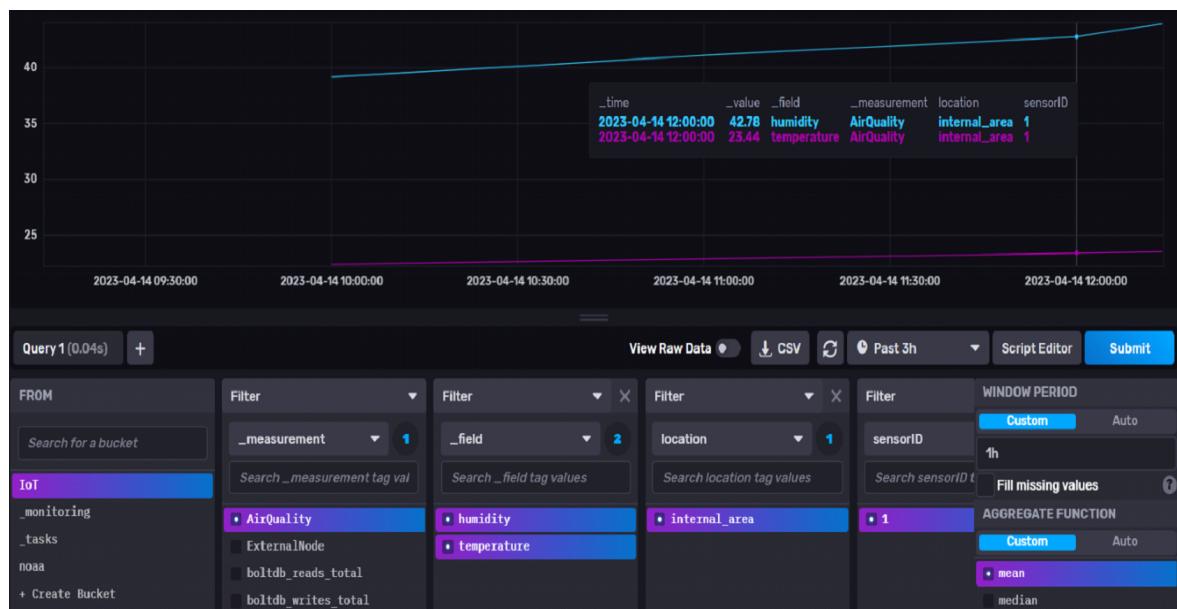


Figura 16: Tópico *IPB/IoT/Lab/ExternalNode* 23.44 °C e 42.78%

4.4 SingleStats: Gráfico max e min AirQuality Temperatura última hora

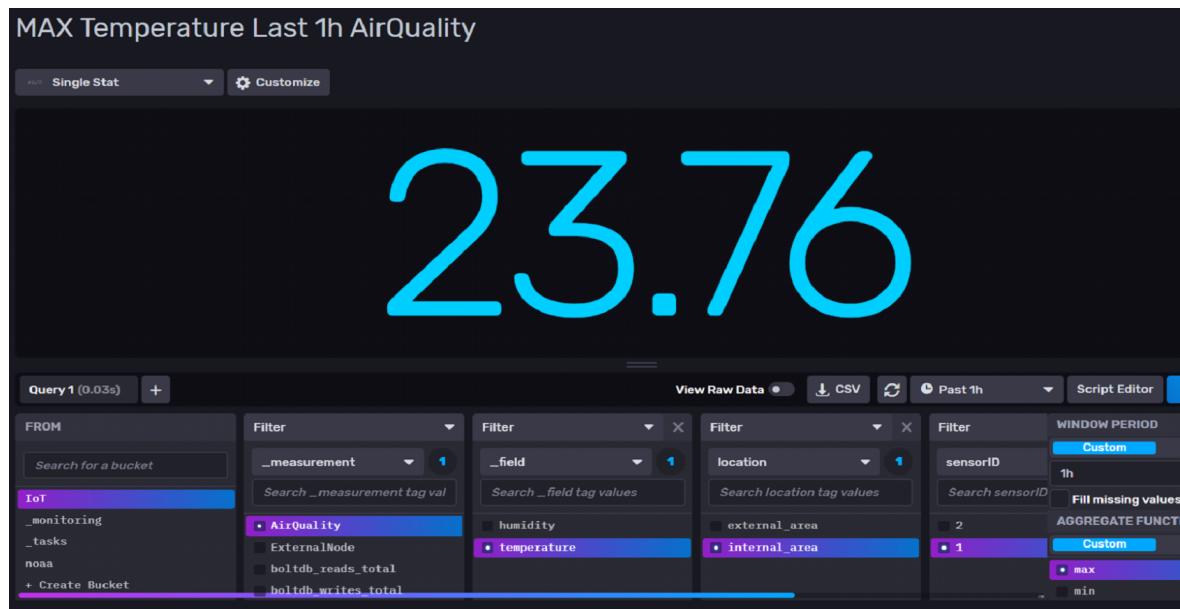


Figura 17: Tópico *IPB/IoT/Lab/AirQuality* valor máximo temperatura

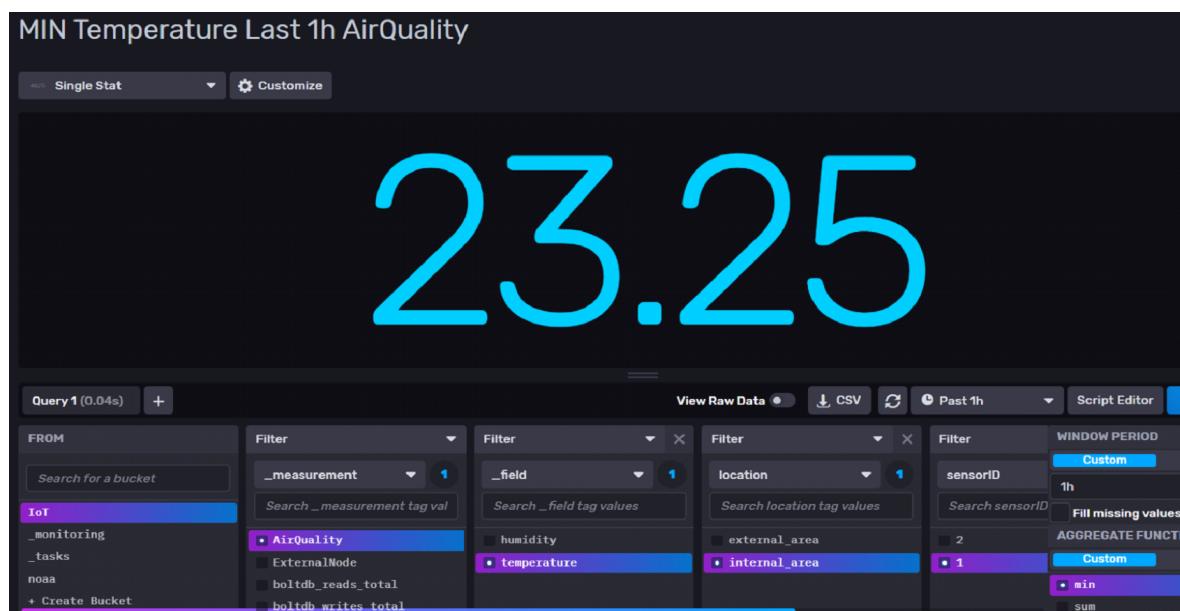


Figura 18: Tópico *IPB/IoT/Lab/AirQuality* valor mínimo temperatura

4.5 SingleStats: Gráfico max e min *AirQuality* Humidade última hora

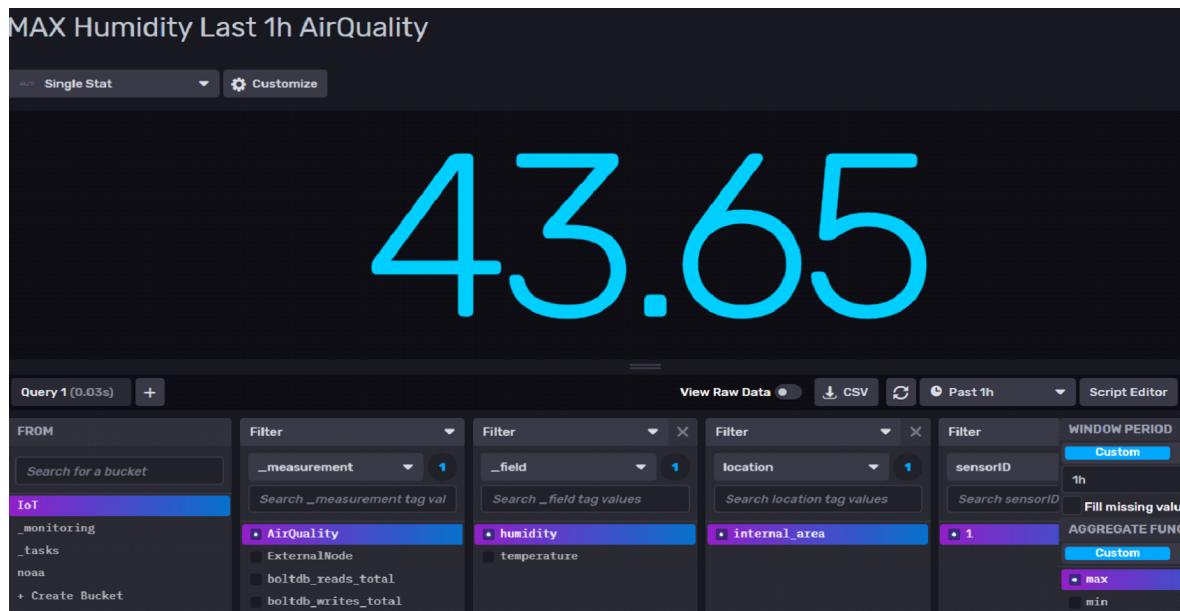


Figura 19: Tópico *IPB/IoT/Lab/AirQuality* valor máximo humidade

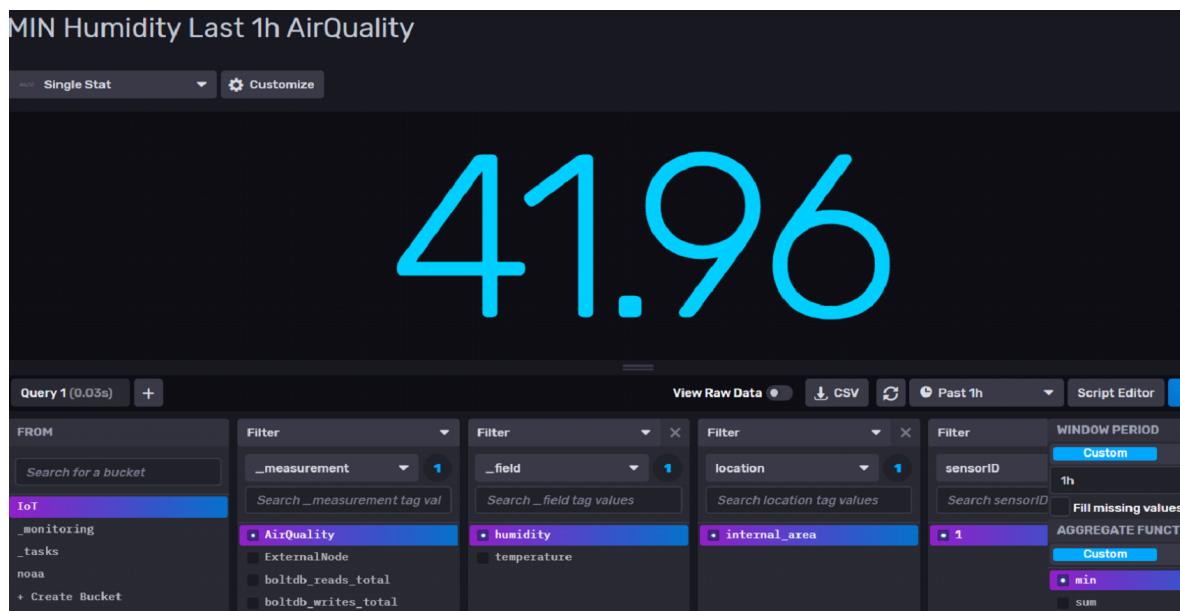


Figura 20: Tópico *IPB/IoT/Lab/AirQuality* valor mínimo humidade

4.6 SingleStats: Gráfico max e min *ExternalNode* Temperatura última hora

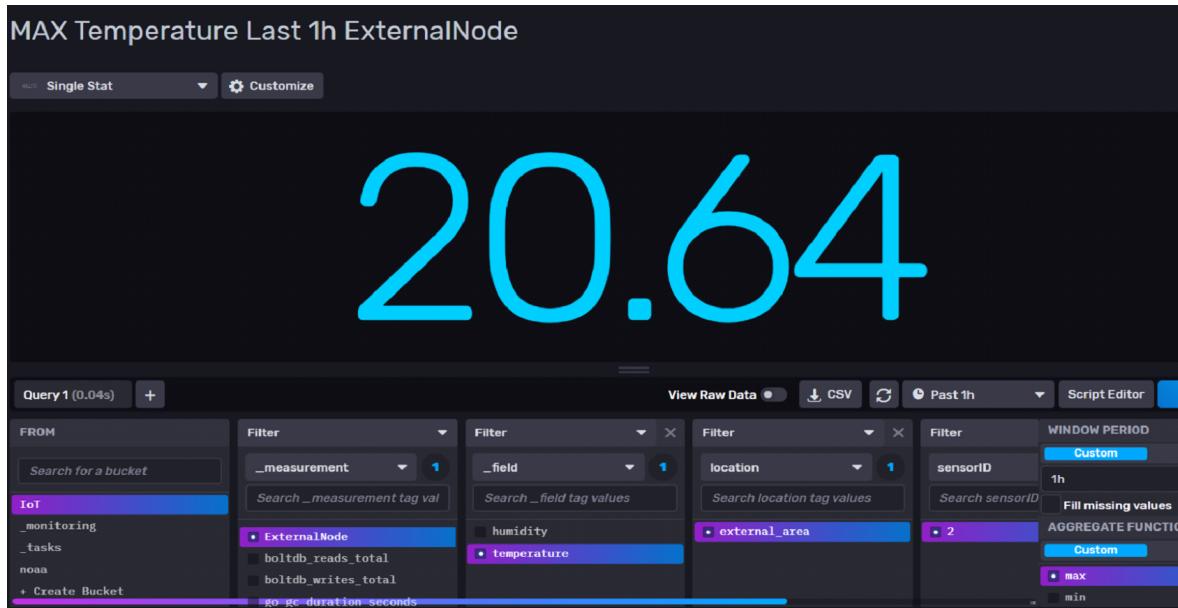


Figura 21: Tópico *IPB/IoT/Lab/ExternalNode* valor máximo temperatura

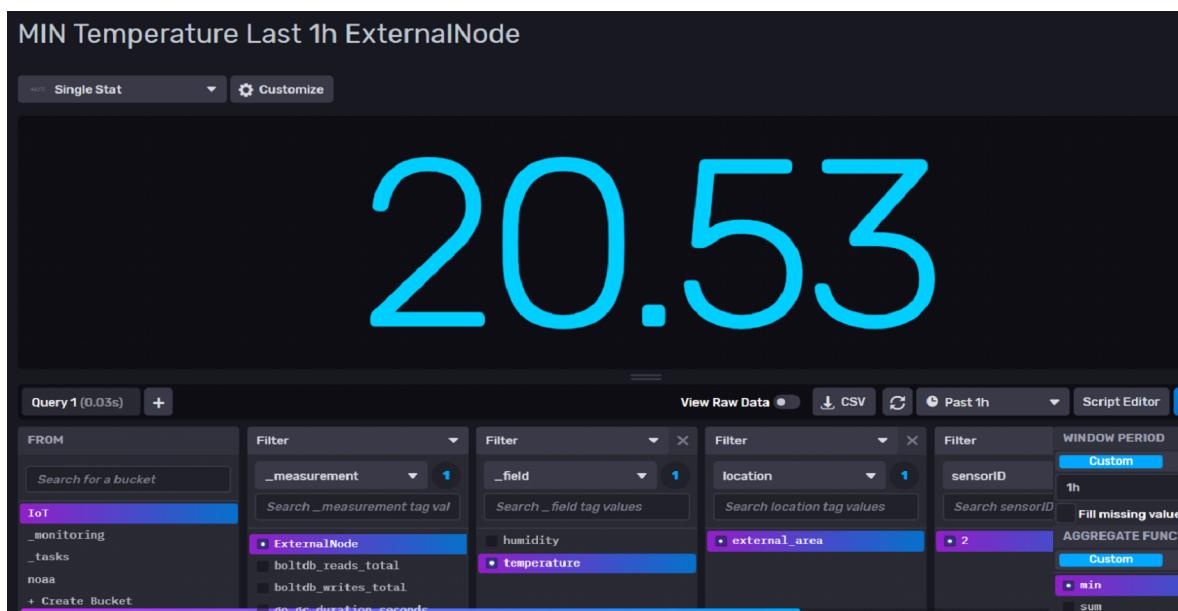


Figura 22: Tópico *IPB/IoT/Lab/ExternalNode* valor mínimo temperatura

4.7 SingleStats:Gráfico max e min *ExternalNode* Humidade última hora

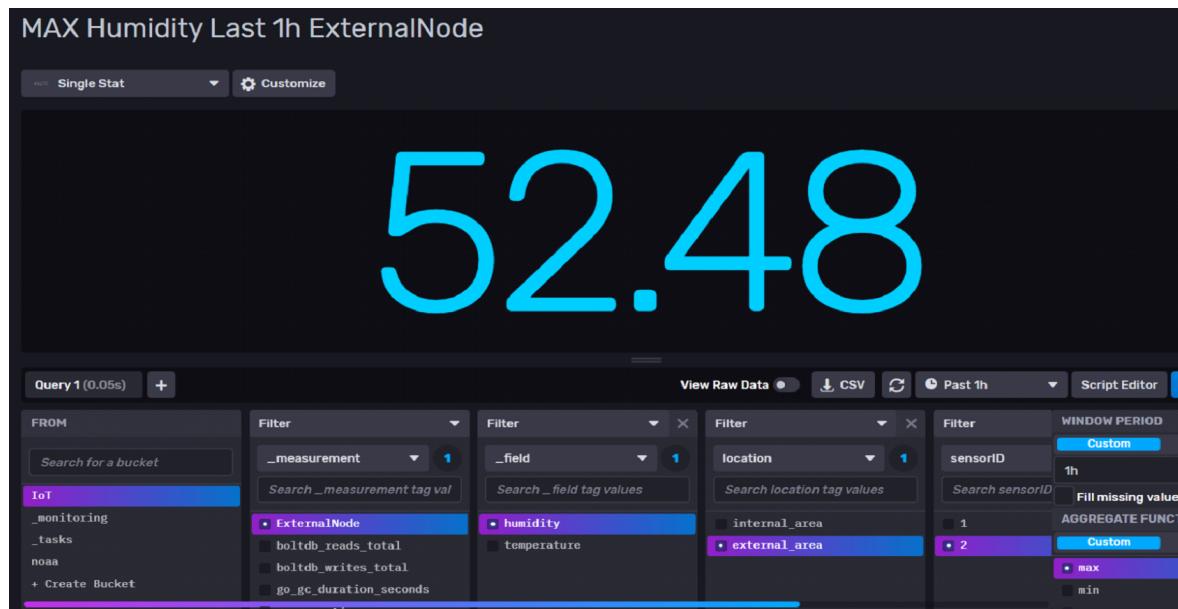


Figura 23: Tópico *IPB/IoT/Lab/ExternalNode* valor máximo humidade

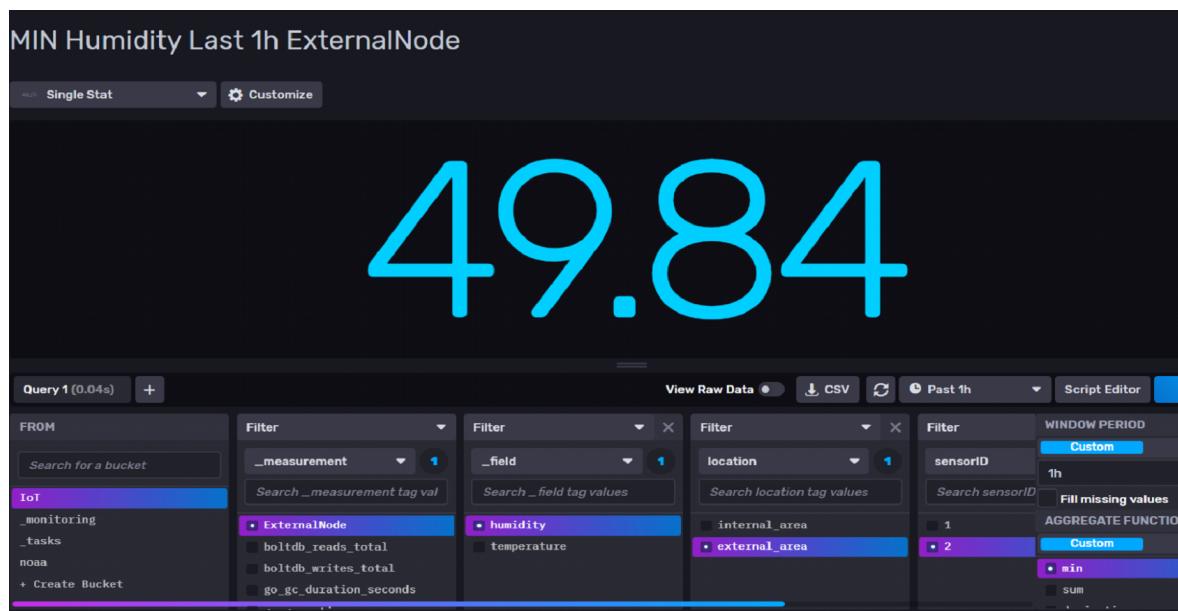


Figura 24: Tópico *IPB/IoT/Lab/ExternalNode* valor mínimo humidade

5 Questão 09 - Sistema com Node-RED, InfluxDB e ESP

Nos exercícios anteriores, foi subscrito o tópico *IPB/IoT/Lab/ExternalNode* e o qual redirecionou-se apenas as informações de temperatura e humidade num *array* que é armazenado no InfluxDB, em um Bucket.

Diante disso, deseja-se fazer o controlo automático de uma janela, aberta/fechada, com base na temperatura média da última hora desses valores guardados no InfluxDB, e um valor de temperatura definido pelo utilizador atuando como um limitador, a ver mais à frente.

Por conseguinte, em primeiro passo, no InfluxDB, com os valores de temperatura a serem armazenados, em tempo real, a partir do tópico *IPB/IoT/Lab/ExternalNode* é possível obter o valor médio desses conjuntos de valores, ao aplicar uma série de filtros.

Em seguida, pode-se definir um *windowPeriod* de 1 hora, para a função de agregação, a média, do inglês, *mean*. Diante disso, com todos estes passos, é possível obter o valor da média da última hora, que na Figura 25, na próxima página elucidada, é definida como 19.73 graus.

À vista disso, perceba que na Figura 25, para que se encontre o valor de temperatura desejado, é preciso saber escolher as configurações/filtros almejados para alçar seu fim.

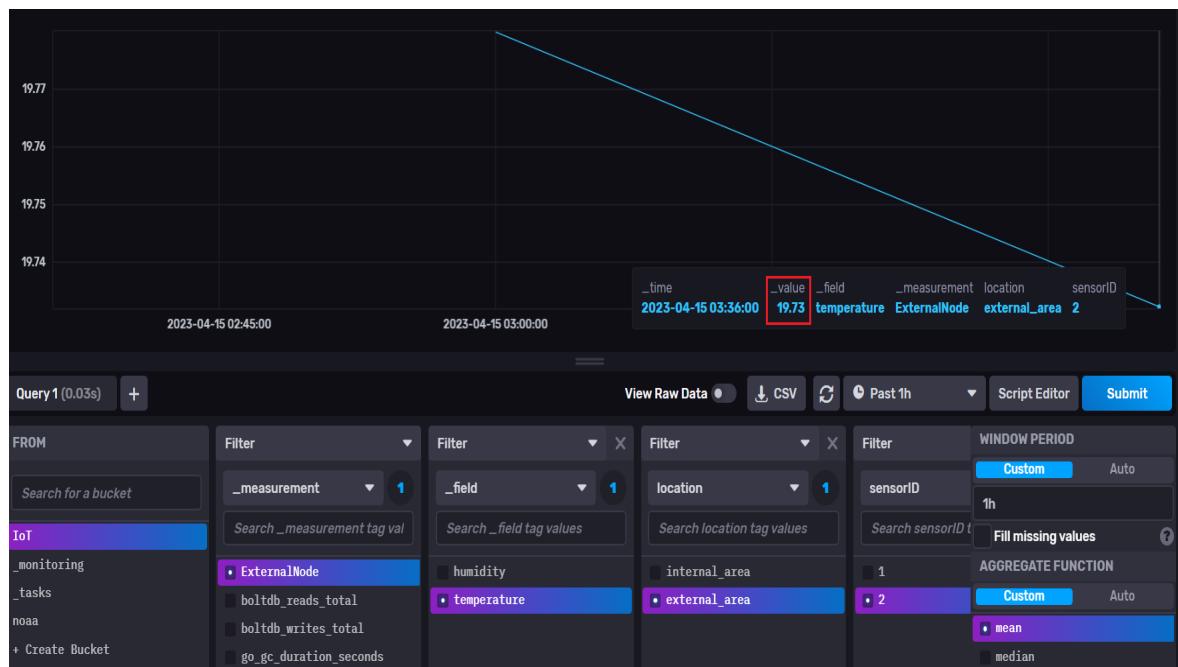


Figura 25: Temperatura no External Node pelo InfluxDB

Diante disso, para encontrar este valor, veja na figura que no Bucket IoT, continue para o *ExternalNode*, em seguida filtre por *temperature*, assim para a *external_area*, *sensorID* de número 1 e pela função *aggregateWindow* em *mean*.

Em seguida, após obter o valor da temperatura média dos valores da subscrição ao tópico *IPB/IoT/Lab/ExternalNode*, no Node-RED é possível definir o flow da nossa aplicação. Diante disso, na Figura 26, na próxima página, é possível ver a ilustração de todo o processo.

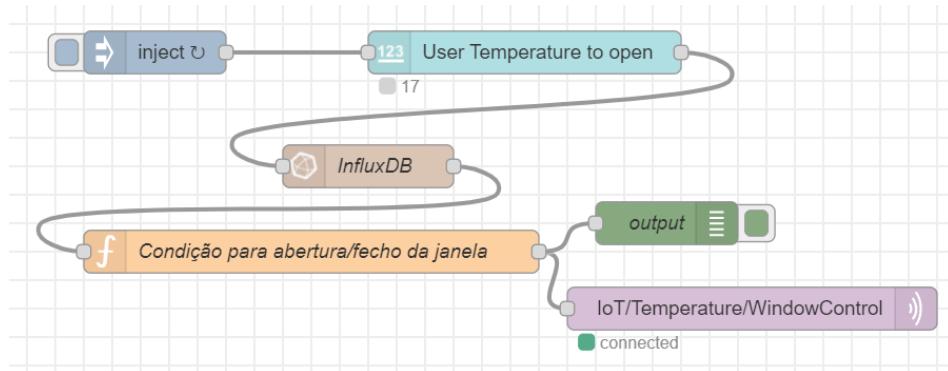


Figura 26: Fluxo no Node-RED com conexão ao InfluxDB

Em sequência, no fluxo da Figura 26, o nó **Inject** serve diretamente para iniciar fluxo definido num determinado intervalo, neste caso de 1 em 1 minuto. E para melhor apresentar isso, veja a maneira de configuração conforme é expresso abaixo na Figura ??.

Consoante, é válido apontar que este intervalo de tempo dado, é o campo que define o quanto tempo de intervalo que este **Inject** será chamado e executado.

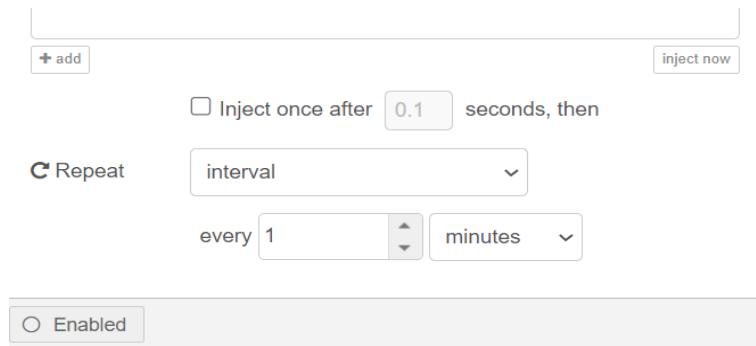


Figura 27: Configuração Repeat do Inject no Node-Red

Por conseguinte, no nó seguinte, é usado o *numeric* no node do package *dashboard*

para ir coletar a temperatura para a condição de aberta/fechada definida pelo utilizador de uma forma dinâmica.

Diante disso, o utilizador tem acesso ao *dashboard*, e como na imagem abaixo, consegue definir a sua temperatura para alterar o estado da janela. À vista disso, se a temperatura for maior que 17, pode abrir, caso contrário, permanece fechada. Veja na Figura 28, abaixo expressa, como é a visualização do Dashboard do Node-RED do nó *numeric*.

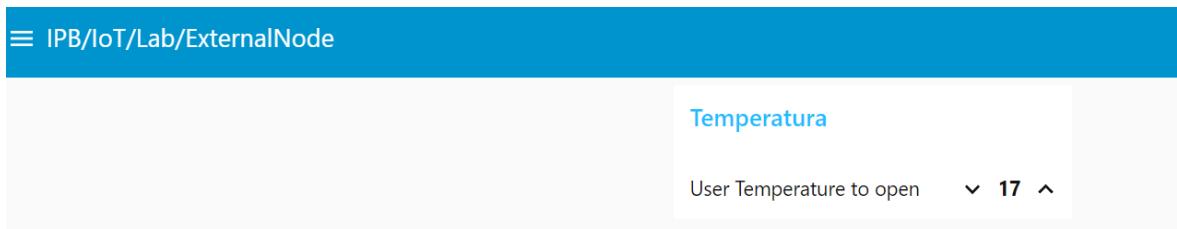


Figura 28: Dashboard Node-RED do nó *numeric*

Em sequência, o nó seguinte, da Figura 26, o terceiro nó da esquerda para a direita, é o nó no qual há a conexão do Node-RED com o InfluxDB. Nele, neste nó, sua função é de que o *InfluxDB In* node para fazer o pedido da temperatura média última hora ao banco de dados influxDB, definida pela *query*.

Diante disso, note que a *query* utilizada no InfluxDB é a linguagem tem como foco os comandos necessários para atuar no filtro e tratamento de dados necessários para obter uma finalidade. Tendo isso em vista, é com base nesta lógica, a seguir consta a Figura 29, a qual elucida o texto da *query* para tratamento de dados desejados.

```

from(bucket: "IoT")
|> range(start: -1h)
|> filter(fn: (r) => r["_measurement"] == "ExternalNode")
|> filter(fn: (r) => r["_field"] == "temperature")
|> filter(fn: (r) => r["location"] == "external_area")
|> filter(fn: (r) => r["sensorID"] == "2")
|> aggregateWindow(every: 1h, fn: mean, createEmpty: false)
|> yield(name: "mean")

```

Figura 29: Script da query no InfluxDB

Em seguida, na Figura 30, na próxima página, refere-se ao nó de função, a qual é

definida a condição de abertura/fechada para ser publicada num determinado tópico. Nela reconstrui-se o objeto de *payload*, que é definido com a temperatura obtida do *InfluxDB* arredondada a 2 casas decimais, com o atributo *temperatureInflux*.

Diante disso, para a temperatura do utilizador guarda-se como atributo *userTemp*, e usamos o *msg.topic* para receber o valor da temperatura seleciona no dashboard. Em continuidade, na estrutura condicional *if*, define-se a condição de abertura/fecho da janela.

Esse valor é guardado em um novo atributo *windowControl*. Se a temperatura definida pelo utilizador for maior que a temperatura média da passada hora, o atributo *windowControl* recebe o valor de “CLOSE”, que indica a janela fechada.

Caso contrário, se a temperatura seleciona, for menor que a temperatura média, a janela passa para o estado aberto, *windowControl* é definido como “OPEN”. Em adenddo, note que o valor ”OPEN” ou ”CLOSE” é definido pelo valor inserido pelo usuário no Node-RED Dashboard e a outra parte pelos valores recebidos pela base de dados do InfluxDB.



Figura 30: Configuração da abertura da Janela no Node-RED

Consequentemente, neste último processo, recorre-se ao MQTT out node, para se fazer a publicação do objeto construído no passo anterior no tópico IoT/Temperatura/WindowControl.

Dessa forma, a Figura 31, abaixo, demonstra a forma do objeto, em debug, que é publicado no tópico. Como atributos, contém o valor medido do influx referente à temperatura média(temperatureInflux), o valor especificado pelo utilizador(userTemp) e o estado da janela (windowControl).

```
15/04/2023, 03:36:10 node: output
17 : msg.payload : Object
  ▼object
    temperatureInflux: 19.73
    userTemp: 17
    windowControl: "OPEN"
```

Figura 31: Saída do valor do fluxo em OPEN

Em seguida, com intuito de continuidade, recorre-se ao microcontrolador ESP para fazer uma pequena simulação com o seu LED interno sobre o estado da janela. Para isso, usou-se a plataforma online WokWi, de simulação de hardware que permite testar o ESP32 em ambiente virtual.

Para tanto, a título de 2 exemplos, respetivamente, nas Figuras 32 e 34, nas próximas páginas apresentadas, tem-se no primeiro o objeto recebido é o apresentado acima, a temperatura do usuário é definida com o valor 17 e a temperatura média igual a 19.73.

E são nestas condições, como o valor da temperatura média é maior que a definida pelo utilizador, a o estado da janela passa para OPEN (aberta). O LED interno do simulador ESP32 acende.

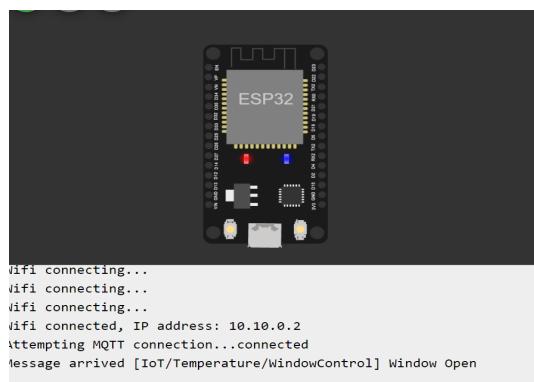


Figura 32: ESP LED azul ativo igual a janela aberta

Em seguida, para melhor exemplificar, fez-se uma pequena demonstração para o estado fechado da janela. Para tanto, define-se o valor da temperatura do utilizador igual a 21 (observer que o valor "21" é o valor inserido pelo usuário no Dashboard do Node-RED); a Figura 33, abaixo exposta, expressa essa realidade.

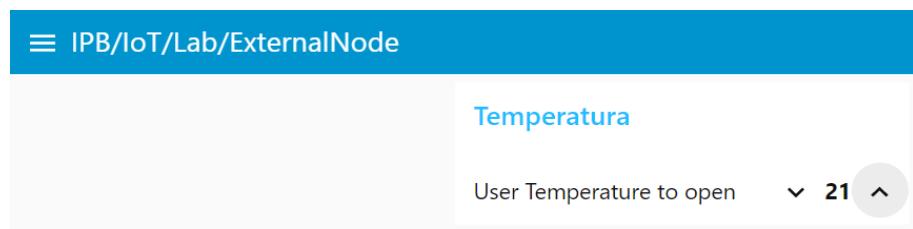


Figura 33: Configuração para o LED azul ativar no Node-RED

Dessa forma, o valor da temperatura média ainda continua com o mesmo valor, portanto não sofreu nenhuma alteração.

Em seguida, repare que o valor da temperatura inserida pelo utilizador passou a ser maior que o valor da temperatura média lido pelo influxDB, isto resulta na mudança de estado da janela, passando para o estado CLOSE(fechado). Para tanto, a Figura 34, a seguir elucidada, expressa este cenário.

```
15/04/2023, 03:37:02 node: output
21 : msg.payload : Object
  ↴object
    temperatureInflux: 19.73
    userTemp: 21
    windowControl: "CLOSE"
```

Figura 34: Saída do valor do fluxo em CLOSE

E é com base nisso que no simulador, o novo objeto recebido contém o estado da janela com o valor CLOSE(fechado), portanto o LED interno se apaga, e para isso, na próxima página, a Figura 35 apresenta este fato.

Diante disso, é bom levar em consideração que é levado em consideração pela função de verificação no Node-RED que a janela é fechada, apenas e portanto, se somente se, a comparação apontar que o valor tido pelo InfluxDB for menor ou igual ao valor inserido pelo usuário.

E para tanto, o LED azul aponta quanto ligado a representação da janela aberta, enquanto o LED azul apagado indica a janela fechada. E diante disso, com base no Node-RED, acima citado, é possível representar pelo próprio ESP32 a atuação da janela.



Figura 35: Caption

Por conseguinte, para que a placa ESP32, microcontrolador atuar, é indispensável que seja implantado um código para essa finalidade. E com este intuito, a Figura 36, abaixo apresentada, expressa partes deste código.

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

//wifi credentials
const char* ssid = "Wokwi-GUEST";
const char* password = "";

long lastMsg = 0;
char msg[50];
int value = 0;
// MQTT client
WiFiClient wifiClient;
PubSubClient client(wifiClient);

//broker variables
const char* mqtt_server = "broker.mqtt-dashboard.com";
int mqttPort = 1883;
```

Figura 36: Código inicial do ESP32

Sendo assim, essa parte do código, da Figura 36, define as credenciais da rede Wi-Fi à qual o se vai conectar, as variáveis de mensagem usadas pelo dispositivo e as variáveis relacionadas ao cliente MQTT, como o endereço do servidor MQTT e a porta que o dispositivo vai usar para se conectar ao servidor.

Em sequência, consta a Figura 37, a qual contém a função *connectWiFi()* que é responsável por conectar o dispositivo a uma rede Wi-Fi.

```

void connectWiFi(){
    Serial.print("Connecting to ");
    WiFi.begin(ssid, password,6);
    Serial.println(ssid);

    while (WiFi.status() != WL_CONNECTED) {
        delay(100);
        Serial.println("Wifi connecting...");
    }
    Serial.print("Wifi connected, IP address: ");
    Serial.println(WiFi.localIP());
}

```

Figura 37: Função connectWifi ESP32

Dito isso, a figura seguinte, Figura 38, expressa a função função *brokerReconnect()* é responsável por tentar conectar o dispositivo ao broker MQTT. Se a conexão falhar, a função tenta se reconectar após um intervalo de tempo.

```

void brokerReconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("a39041")) {
            Serial.println("connected");
            client.subscribe("IoT/Temperature/WindowControl");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

```

Figura 38: Função brokerReconnect ESP32

Em seguida, abaixo, na Figura 39, está a função *callback()* é responsável por processar as mensagens recebidas do tópico *"IoT/Temperature/WindowControl"*, e definir o estado do LED interno.

Não distante, lembre-se que são as funções os principais blocos de códigos nos quais agrupam suas variáveis e funcionalidades para que seja possível atuar para um determi-

nado fim. Consoante, as funções podem conter argumentos para tratamentos de dados, a qual pode ou não ter um retorno, desde que seu fim seja obtido.

```

void callback(String topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    String strPayload="";
    for (int i=0;i<length;i++) {
        //byte to char
        strPayload = strPayload + (char)payload[i];
    }

    StaticJsonBuffer<200> jsonBuffer;
    JsonObject& data = jsonBuffer.parseObject(strPayload);

    if(topic == "IoT/Temperature/WindowControl" ){
        if(data["windowControl"] == "CLOSE"){
            digitalWrite(LED_BUILTIN, LOW); //Turn off the LED
            Serial.println("Window Close");
        }else{
            digitalWrite(LED_BUILTIN, HIGH); //Turn on the LED
            Serial.println("Window Open");
        }
        Serial.println();
    }
}

```

Figura 39: Função callback ESP32

Em seguida, as duas últimas funções a ser tratadas constam-se na Figura 40, na próxima página elucidade, as quais são *setup()* e *void()*.

```

void setup() {
    Serial.begin(9600);
    pinMode(LED_BUILTIN, OUTPUT);
    connectWiFi();
    client.setServer(mqtt_server, mqttPort);
    // set the callback function
    client.setCallback(callback);

}

void loop() {
    if (!client.connected()) {
        brokerReconnect();
    }
    client.loop();
}

```

Figura 40: Função setup e void ESP32

São essas funções as principais para inicialização e da criação das variáveis fundamentais para a atuação do código fonte do projeto. Lembre-se que é nesta estapa em que se

deve ter muita atenção para adicionar as variáveis necessárias.

Em primeiro plano, a função *setup()* é executada com foco tendo como vista de ser de início, o responsável por inicializar o dispositivo e configurar os requisitos indispensável para atuar o cliente MQTT.

Por fim, a função *loop()* é executada continuamente após a função *setup()* e é responsável por manter a conexão com o broker MQTT e chamar a função de retorno de chamada quando uma mensagem é recebida.

Referências

[Influx Data] INFLUX DATA: *InfluxDB and Flux: What's Flux?* Disponível em: <https://www.influxdata.com/products/flux/>. Acesso em: 10 Abr. 2023.

[InfluxData] INFLUXDATA: *Time series database (TSDB) explained.* Disponível em:<https://www.influxdata.com/time-series-database/>. Acesso em: 10 Abr. 2023.

[ServerDoin] SERVERDOIN: *InfluxDB: Opção de banco de dados para um alto volume de consultas e escritas.* Disponível em: <https://serverdo.in/influxdb/>. Acesso em: 10 Abr. 2023