PADO

# Labs


# Microcontroladores Aplicados a IoT

Dilson Liukiti Ito


# 09 – mDNS, JSON, Linked List e Websocket Server com ESP32

**Microcontroladores Aplicados a IoT**

# mDNS


- o protocolo multicast DNS (mDNS) resolve nomes de host para endereços IP em pequenas redes que não incluem um servidor de nomes local.
- útil quando os endereços IP e nomes de host do ponto de extremidade de destino não são conhecidos em tempo de design.

**DNS:** www.hausenn.com.br => (173.254.29.38)

**mDNS:** (192.168.1.100) => mdns_hostname.local

https://en.wikipedia.org/wiki/Multicast_DNS

https://docs.microsoft.com/pt-br/azure-sphere/app-development/service-discovery

# mDNS no ESP32


- hostname: o nome do host ao qual o dispositivo irá responder. Se não for definido, o nome do host será lido na interface. Exemplo: my-esp32 resolverá para my-esp32.local

- default_instance: nome amigável para seu dispositivo, como Jhon's ESP32 Thing. Se não for definido, o nome do host será usado.

```
#include "mdns.h"
```

```
ESP_ERROR_CHECK(mdns_init());
ESP_ERROR_CHECK(mdns_hostname_set(MDNS_NAME));
ESP_LOGI(TAG, "mdns name: %s", MDNS_NAME);
ESP_ERROR_CHECK(mdns_instance_name_set(INSTANCE_NAME));
```

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mdns.html

# Exemplo mDNS

- Baixe o arquivo Aula09.zip
- Utilize o arquivo exemplo mdns.c

- Altere o #define NOME
  - Para o seu nome;

- Rode o programa
- Tente acessar pelo navegador: http://seunome-esp32.local

# avahi


- Para que o mDNS funcione no Linux, é necessário instalar e executar o avahi

```
$ sudo apt-get –y install avahi-daemon
```

```
$ sudo apt-get –y install libnss-mdns
```

```
$ service avahi-daemon start
```

https://installati.one/ubuntu/20.04/avahi-daemon/ https://www.thegeekdiary.com/linux-os-service-avahi-daemon/ https://wiki.debian.org/Avahi

# JSON


- JavaScript Object Notation
- JSON é um formato leve para armazenar e transportar dados, frequentemente usado quando os dados são enviados de um servidor para uma página da web
- JSON é "autodescritivo" e fácil de entender

https://www.w3schools.com/whatis/whatis_json.asp

# Dados JSON

- Os dados JSON são escritos como pares de chave/valor;
- Um par de chave/valor consiste em uma string (aspas duplas), seguido por dois pontos e por um valor;

```
"chave": valor
```

https://www.w3schools.com/whatis/whatis_json.asp

# Valores JSON

- Os valores podem ser um dos seguintes tipos de dados:
  - uma string, deve ser escrito em aspas duplas `"nome": "Dilson"`
  - um número `"idade": 35`
  - um objeto `"cad_profissional": { "empresa": "Hausenn", "cargo": "desenvolvedor"}`
  - um array, deve ser escrito entre colchetes `"medidas_roupas": [2, 38, 39]`
  - um booleano `"professor_padolabs": true`
  - Nulo `"cadastro_plano_saude": null`

https://www.w3schools.com/whatis/whatis_json.asp

# Objetos JSON


- Os objetos JSON são escritos dentro de chaves.

```
{
    "nome": "dilson"
}
```

- Os objetos podem conter vários pares de nome/valor, que são separados por ","

```
{
    "nome": "dilson",
    "sobrenome": "ito"
}
```

https://www.w3schools.com/whatis/whatis_json.asp

# Exemplo objeto JSON


```
{
    "nome": "dilson",
    "sobrenome": "ito",
    "idade": 35,
    "cad_profissional": {
                            "empresa": "Hausenn",
                            "cargo": "desenvolvedor"
                        }
    "tamanhos": [ 2, 38, 39 ],
    "professor_padolabs": true,
    "cadastro_plano_saude": null
}
```

# Regras de sintaxe JSON


- Os dados estão em pares chave/valor
  - "nome": "dilson"
- Os dados são separados por vírgulas
  - "nome": "dilson", "sobrenome": "ito"
- O objeto JSON é cercado por chaves
  - { "nome": "dilson", "sobrenome": "ito"}
- Os colchetes cercam os arrays:
  - "tamanhos": [ 2, 38, 39]

```
{
    "nome": "dilson",
    "sobrenome": "ito",
    "tamanhos": [ 2, 38, 39 ]
}
```

https://www.w3schools.com/whatis/whatis_json.asp

PADO Labs


https://www.reddit.com/r/ProgrammerHumor/comments/lt407m/json/


https://www.reddit.com/r/ProgrammerHumor/comments/miaxc5/_/

# cJSON


- Analisador JSON ultraleve em ANSI C

https://github.com/DaveGamble/cJSON

```
#include "cJSON.h"
```

```
cJSON *object_json = cJSON_CreateObject();
```

```
cJSON_Delete(object_json);
```

# cJSON (cont.)


```
cJSON_AddNumberToObject(object_json, "idade", 35);
cJSON_AddStringToObject(object_json, "nome","dilson");
cJSON_AddItemToObject(object_json, "cad_profissional", cad_profissional);
cJSON_AddBoolToObject(object_json, "professor_padolabs", true);

cJSON *tamanhos = cJSON_CreateArray();
cJSON *camisa = cJSON_CreateNumber(2);
cJSON_AddItemToArray(tamanhos, camisa);

cJSON_AddItemToObject(object_json, "tamanhos", tamanhos);
```

# cJSON_Print()


```
char *object_string = NULL;
object_string = cJSON_Print(object_json);
ESP_LOGI(TAG, "conteudo json: %.*s", strlen(object_string),object_string);
```

```
if (object_string)
    cJSON_free(object_string);
```

```
{
        "nome": "dilson",
        "sobrenome":    "ito",
        "idade":        35,
        "cad_profissional":     {
                "Empresa":      "Hausenn",
                "Cargo":        "desenvolvedor"
        },
        "tamanhos":     [2, 38, 39],
        "professor_padolabs":   true,
        "cadastro_plano_saude": null
}
```

# cJSON_Parse()


```
cJSON *parse_json = cJSON_Parse(object_string);
```

```
cJSON *nome_json = cJSON_GetObjectItem(parse_json,"nome");
ESP_LOGI(TAG, "valor de nome: %s", cJSON_GetStringValue(nome_json));
```

```
cJSON *idade_json = cJSON_GetObjectItem(parse_json,"idade");
ESP_LOGI(TAG, "valor de idade: %f", cJSON_GetNumberValue(idade_json));
```

# Exemplo cJSON


- Utilize o arquivo exemplo json.c

- Criação de objeto JSON
- Parse do objeto JSON criado
- Não esquecer de liberar memória dos objetos alocados.

# cJSON_Delete()


- Se você adicionar um item a um array ou objeto, não poderá liberar o item diretamente porque ele já está incluído no array ou objeto adicionado.
- O item adicionado é excluído quando o array ou objeto ao qual aquele foi adicionado é excluído.
- Se você tentar liberar um item adicionado a um objeto e depois tentar excluir o objeto, o sistema irá "crashar".

https://blog.actorsfit.com/a?ID=01700-afa5245a-dd95-4659-8d14-111698cd3a2e

# Linked List

- Uma lista vinculada é uma estrutura de dados linear que inclui uma série de nós conectados. Aqui, cada nó armazena os dados e o endereço do próximo nó. Por exemplo,


- Damos ao endereço do primeiro nó um nome especial chamado HEAD. Além disso, o último nó na lista vinculada pode ser identificado porque seu "next" aponta para NULL.

https://www.programiz.com/dsa/linked-list

# Representação Linked List


- Cada nó consiste em:
  - Um item de dados
  - Um endereço de outro nó
- Envolvemos o item de dados e a próxima referência de nó em uma estrutura como:

```
struct node
{
    int data;
    struct node *next;
};
```

https://www.programiz.com/dsa/linked-list

# Exemplo Linked List


- Vamos criar uma Linked List simples com três itens para entender como isso funciona.

https://www.programiz.com/dsa/linked-list

# Exemplo Linked List (cont.)


```c
struct node
{
    int data;
    struct node *next;
};
```

```c
/* Initialize nodes */
struct node *head;

struct node *one = NULL;
struct node *two = NULL;
struct node *three = NULL;

/* Allocate memory */
one = malloc(sizeof(struct node));
two = malloc(sizeof(struct node));
three = malloc(sizeof(struct node));
```

```c
/* Assign data values */
one->data = 1;
two->data = 2;
three->data=3;

/* Connect nodes */
one->next = two;
two->next = three;
three->next = NULL;

/* Save address of first node in head */
head = one;
```


https://www.programiz.com/dsa/linked-list

# Linked List


- O poder de uma lista encadeada vem da capacidade de quebrar a cadeia e juntá-la novamente. Por exemplo. se você quisesse colocar um elemento 4 entre 1 e 2, os passos seriam:
  - Crie um novo nó struct e aloque memória para ele.
  - Adicione seu valor de dados como 4;
  - Aponte seu próximo ponteiro para o nó struct contendo 2 como o valor de dados;
  - Altere o próximo ponteiro de "1" para o nó que acabamos de criar;
- Fazer algo semelhante em um array exigiria mudar as posições de todos os elementos subsequentes.

# Linked List no ESP32


- Rode o arquivo exemplo list.c

- O exemplo utiliza uma biblioteca linked_list (dentro da pasta libs)

# Websocket Server no ESP32


- Rode o arquivo exemplo server.c

- Server irá gerar os dados;
- Clientes irão consumir;

- Conectar ao websocket server pelo Weasel;

# Flexdash


- Interface gráfica para websocket client

```
└─$ git clone git@github.com:tve/flexdash.git
└─$ cd flexdash
└─$ npm install
└─$ npm run dev
```

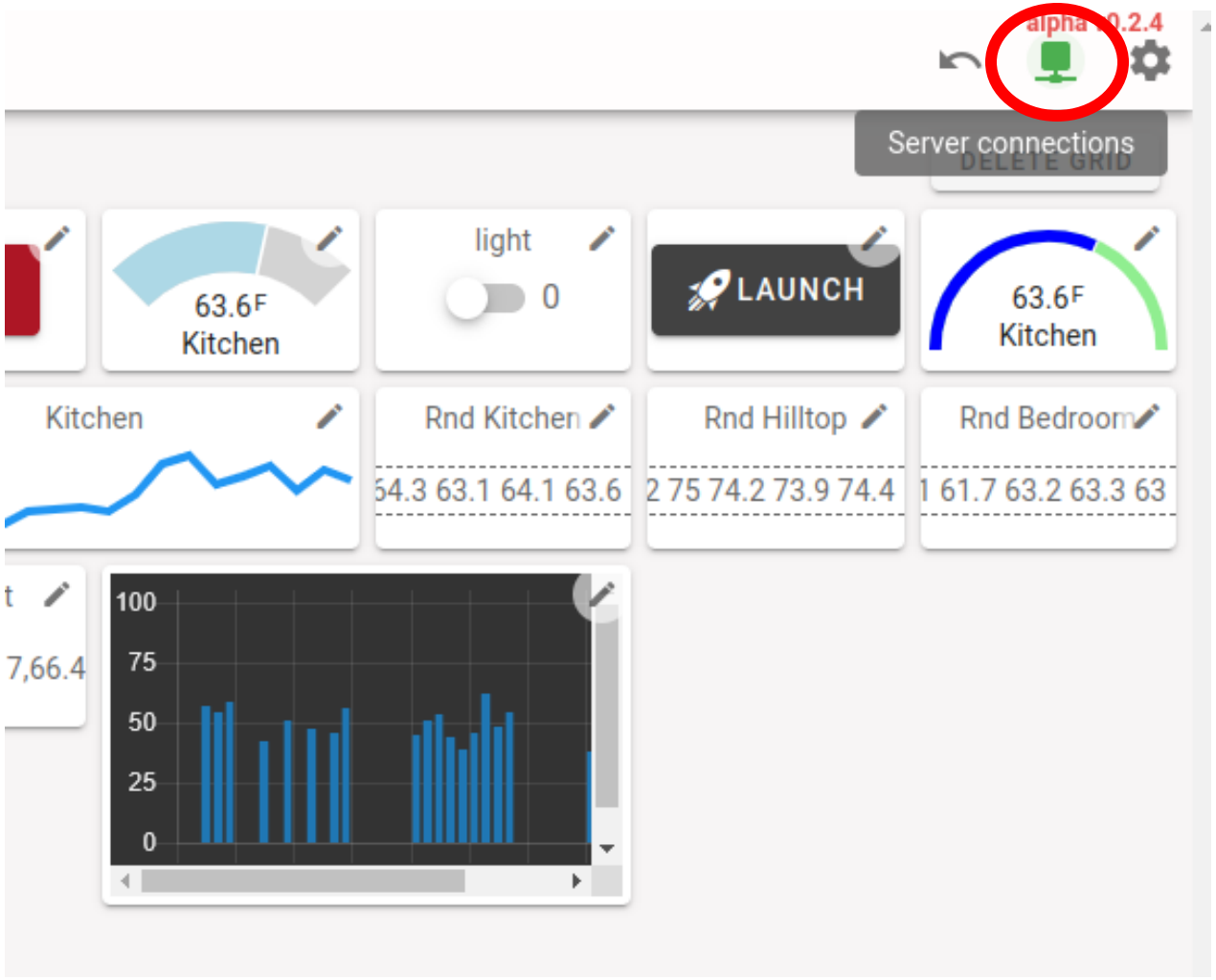
```
http://localhost:3000/
```

# Exemplo envio de dados flexdash

- Envio de números aleatórios entre 0 a 100 e exibidos em um Gauge e um SparkLine

```
{
   "topic": (string),
   "payload": (valor)
}
```

# Conexão com servidor

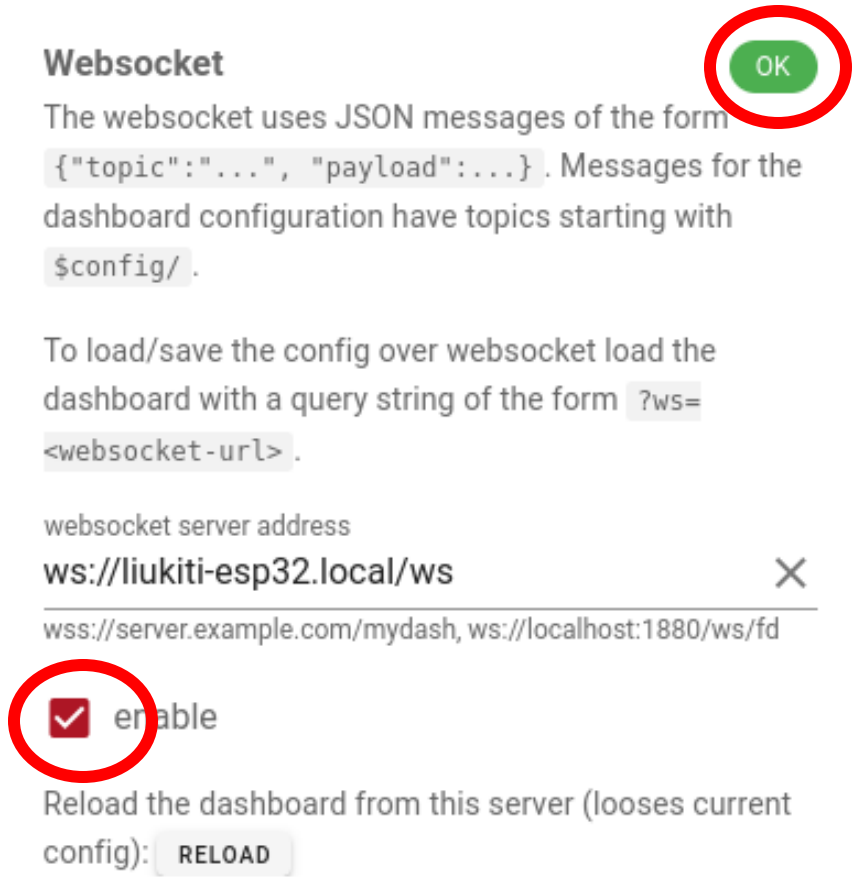



Clique em Server connections

# Websocket server address


- Digite o endereço do servidor;
- Altere adequadamente para o seu servidor

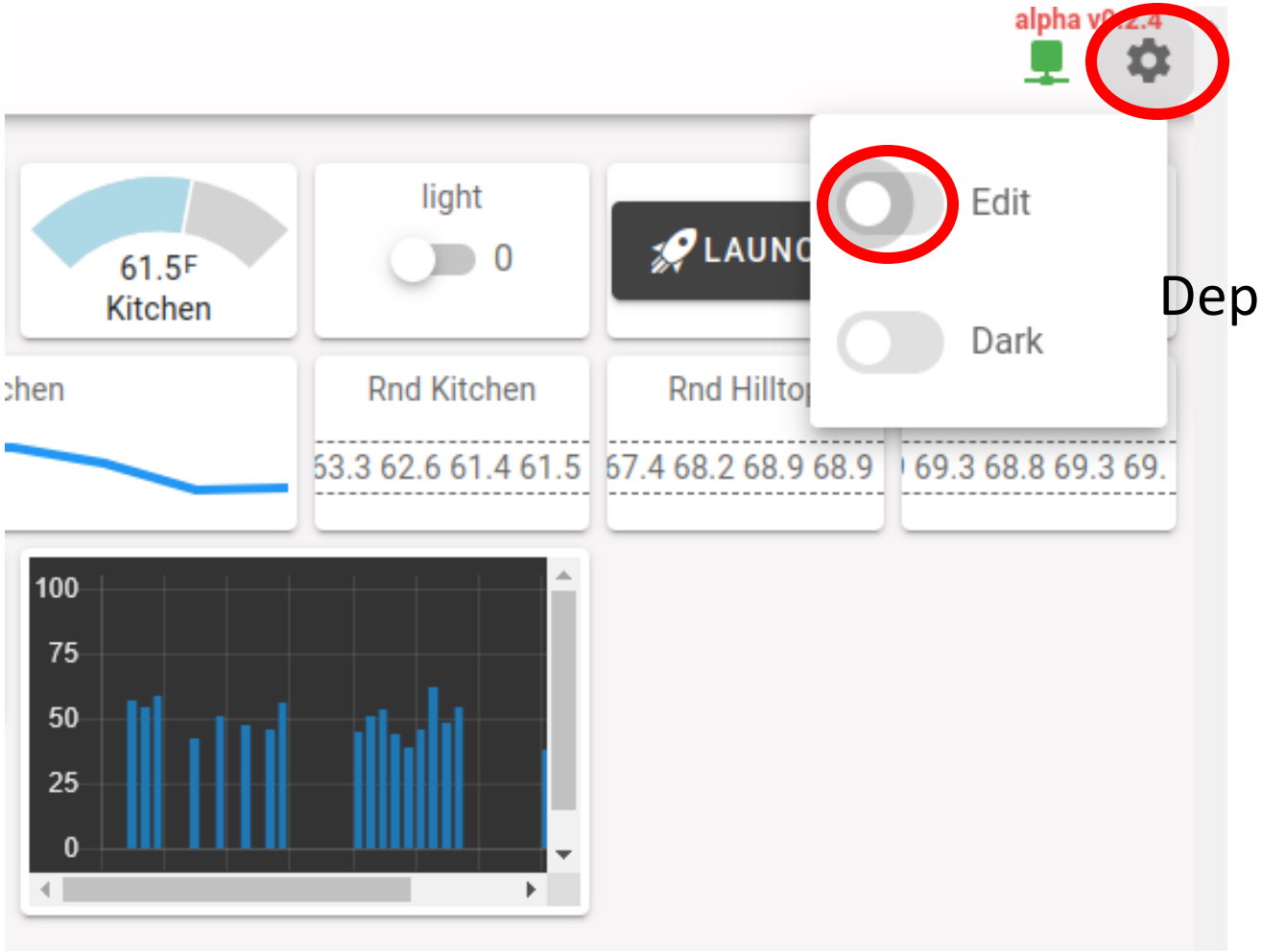
# Conectando ao servidor




O status mudará para "OK"

Clique em enable

# Editar dashboard

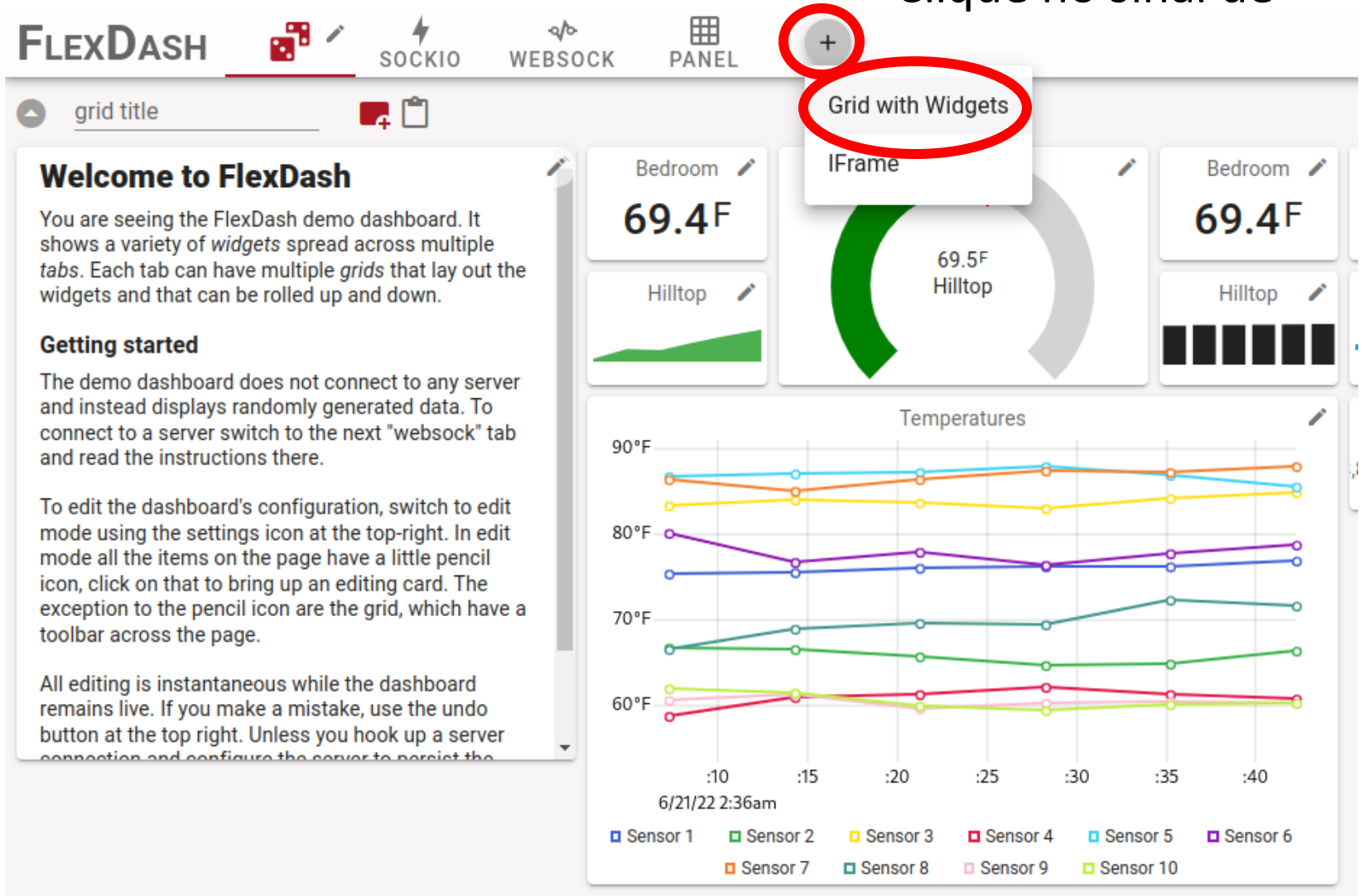



Clique na engrenagem

Depois, clique em Edit

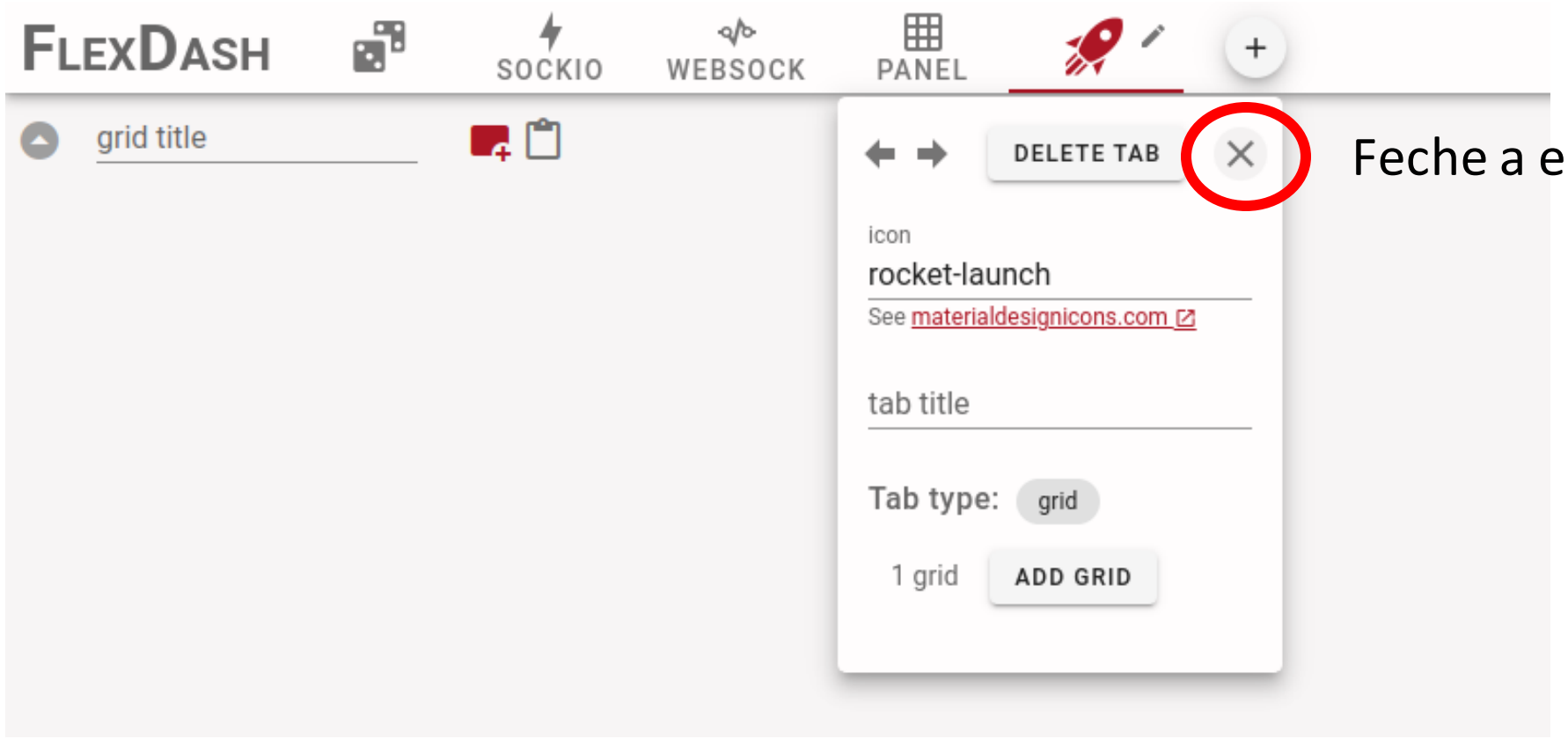
# Criar novo dashboard

Clique no sinal de +



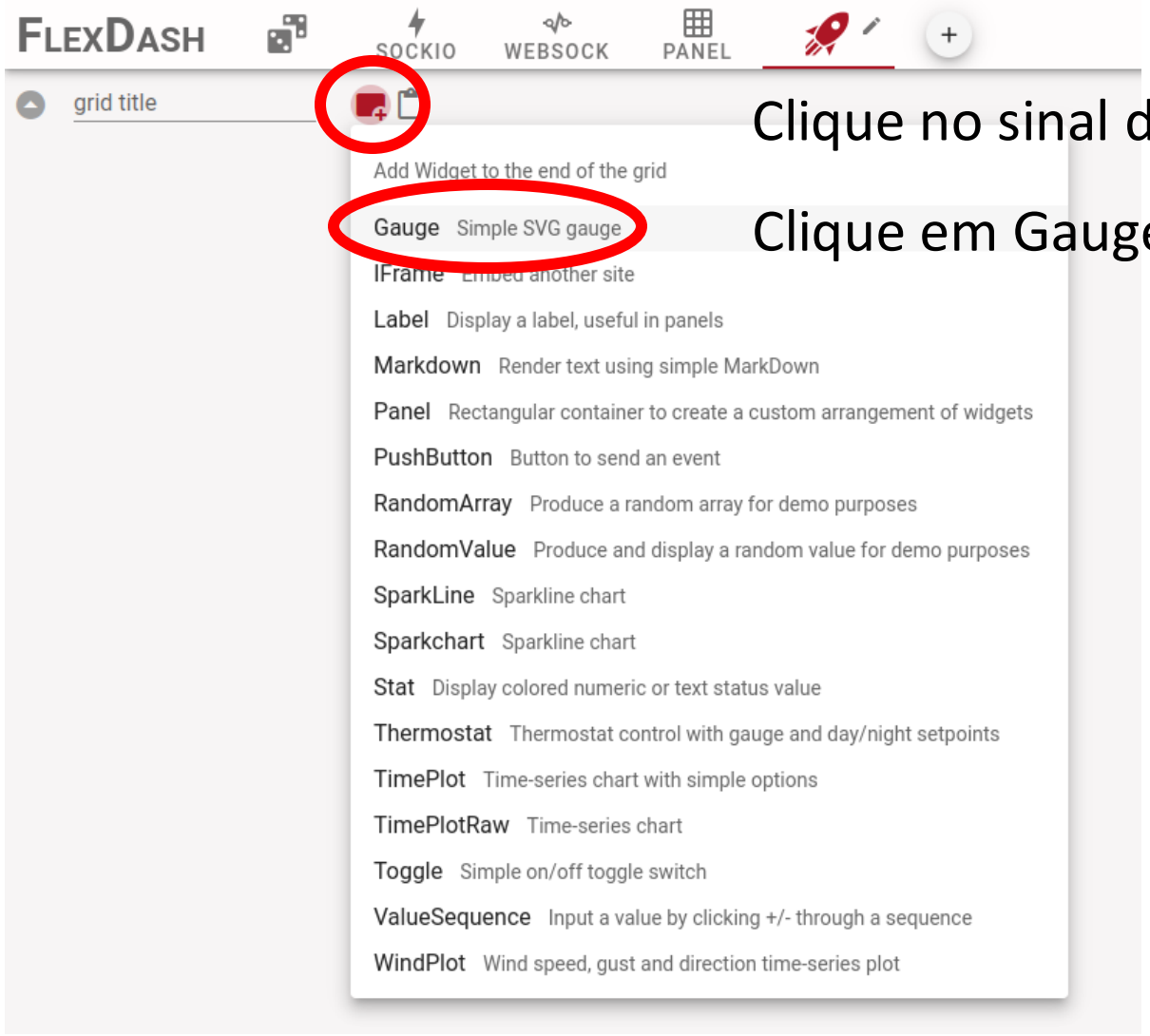

Depois, clique em Grid with Widgets

# Fechando edição de aba




Feche a edição de abas

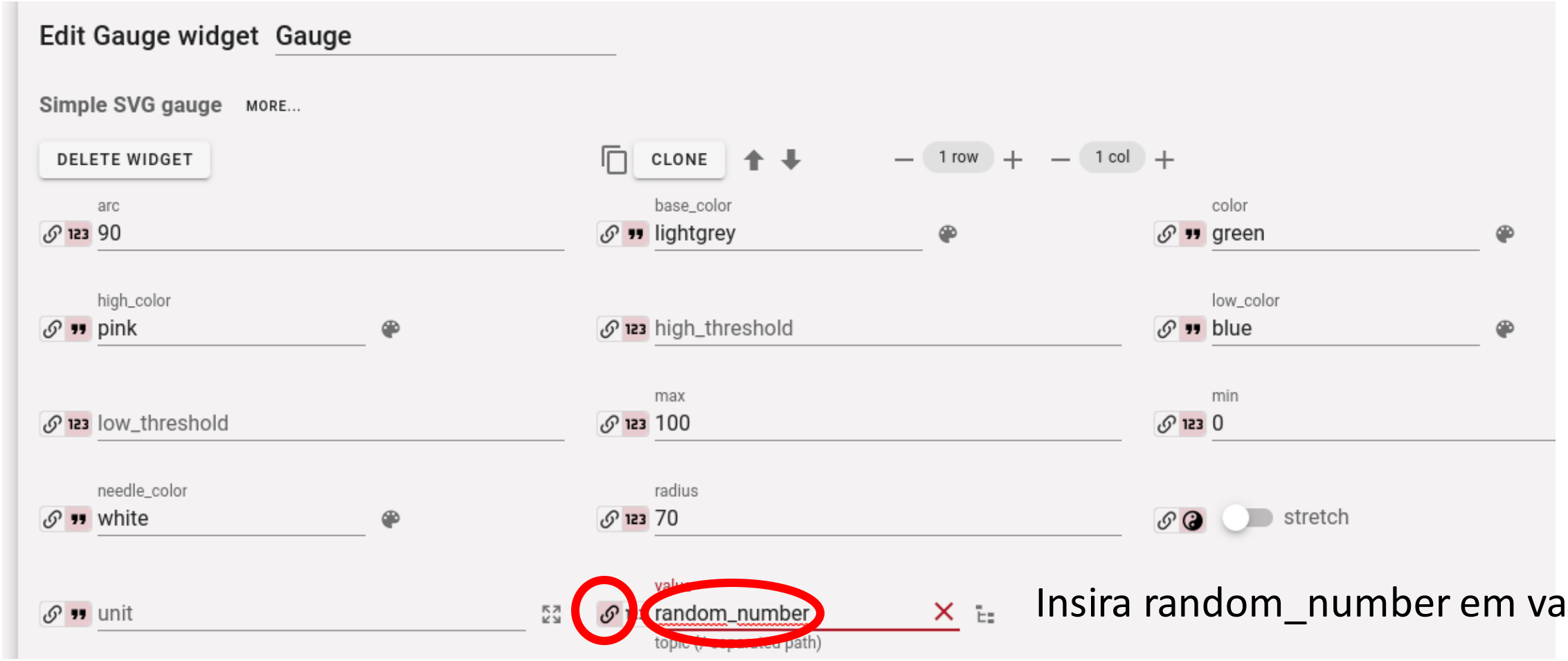
# Criando novo Gauge




Clique no sinal de +

Clique em Gauge

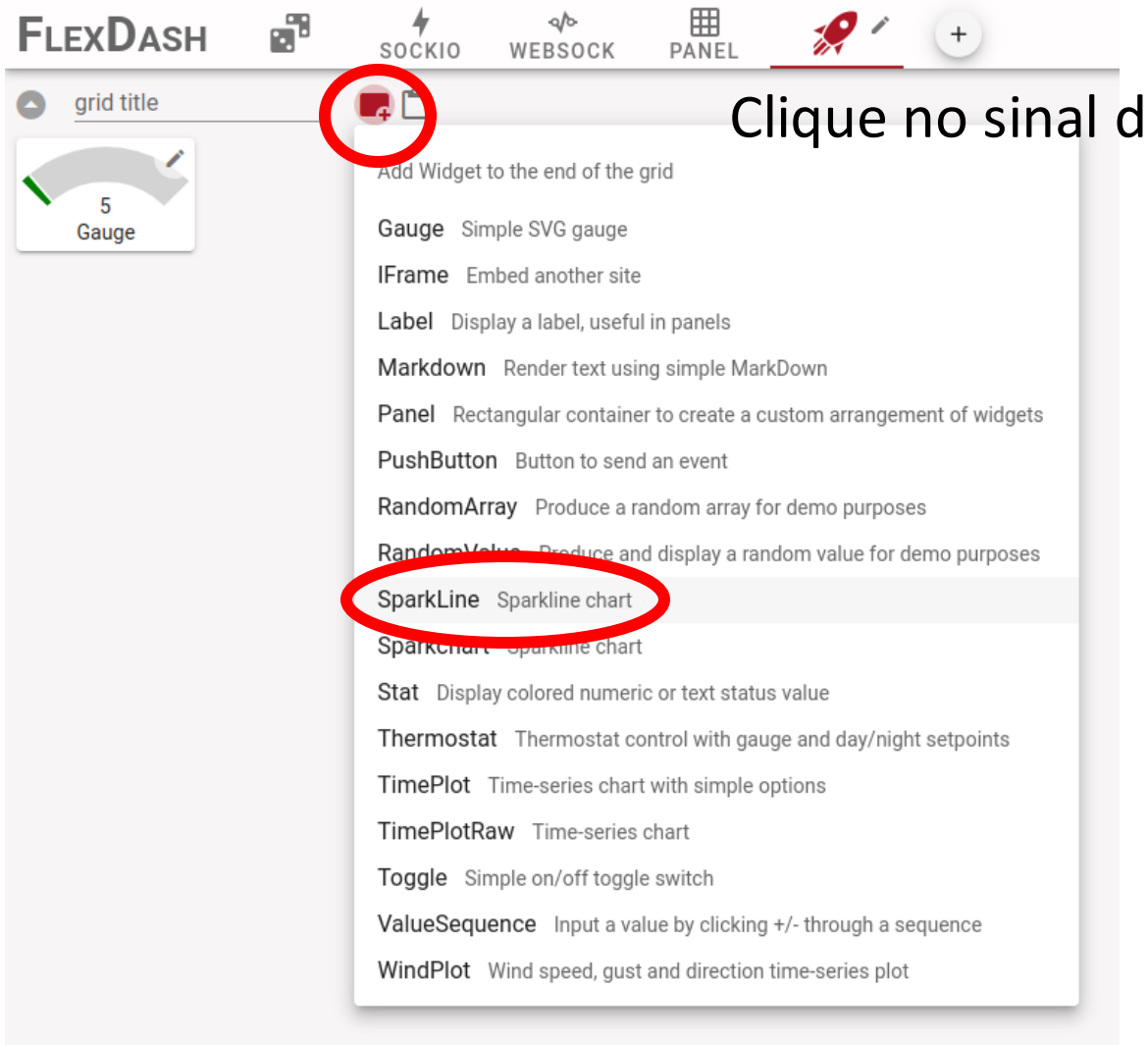
# Config. Gauge




Insira random_number em value

Certifique-se de que dynamic link esteja selecionado

# Criando novo SparkLine




Clique no sinal de +

Clique em SparkLine

# Config. SparkLine


Insira random_number em value

Certifique-se de que dynamic link esteja selecionado

# Exercício

- Faça um programa websocket servidor que envie os dados de RSSI para o cliente flexdash. Esses dados deverão ser plotados em um gauge e um timeplot.
  - Gere um valor de "topic" para cada plot;
    - Ex.: "topic": "rssi_gauge" e "topic": "rssi_timeplot"
  - O valor do payload para um timeplot deverá ser um array: onde o primeiro valor é o timestamp epoch time, seguido do valor a ser plotado.
    - Ex.: "payload": [1655785955, -44]

# Ver mais:


- A Flexible Dashboard Web UI for IoT and Node-Red with Vue: https://morioh.com/p/d604c53a3e49
- Introducing Node-RED 1.0: https://developer.ibm.com/blogs/introducing-node-red-version-1-0/
- Low-Code Server: https://nodered.org/

PADO

# Labs