

Manual Prático LoRaWAN - TTN - KoRe

Alexandro Vanderley dos Santos

30 de junho de 2020

1 Introdução

Com o aumento do mercado da Internet das Coisas (IoT), o controle e monitoramento remoto de dispositivos como sensores e atuadores está cada vez mais em evidência. Para boa parte dos casos a taxa de transmissão não precisa ser alta. Por exemplo, a intensidade da luz solar pode ser monitorada com poucas amostras durante o dia. Nestes casos, uma rede sem fio com baixa taxa de transmissão, mas com consumo reduzido e longo alcance, torna viável a coleta de dados por dispositivos alimentados a bateria, com baixo custo de implantação e manutenção.

Nesta linha, LoRa é uma tecnologia de camada física que permite comunicação a longas distâncias com pouco consumo de energia entre dois dispositivos [1]. Já o protocolo LoRaWAN, mantido pela LoRa Alliance, especifica o funcionamento de uma rede de dispositivos que utilizam LoRa na camada física, incluindo sua conexão a um servidor de rede, que então pode ser conectado à Internet [1]. A The Things Network (TTN), por sua vez, é uma iniciativa com origem na Holanda, mas já espalhada por diversos países, que busca prover um conjunto de soluções para a construção de uma rede LoRaWAN global, aberta, que permita o teste e a implantação de aplicações de IoT com baixo custo [5].

Neste documentos descrevemos os passos para programar um kit que contém um rádio LoRa, configurar alguns parâmetros básicos do protocolo LoRaWAN, conectar à TTN, ou Kore, uma rede comercial, e visualizar na Internet os dados coletados por um sensor ligado ao kit.

2 Material Utilizado

Para a execução dos experimentos será necessário o seguinte:

2.1 Hardware

- Discovery Kit da STMicroelectronics, modelo STM32L072CZY6TR MCU [2];

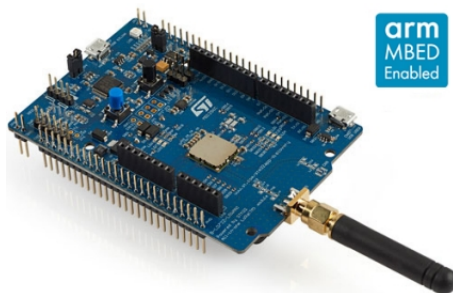


Figura 1: Kit da STMicroelectronics

- Cabo micro USB;
- Desktop ou notebook;
- Sensores ou dispositivos externos, como o módulo IKS01A2.

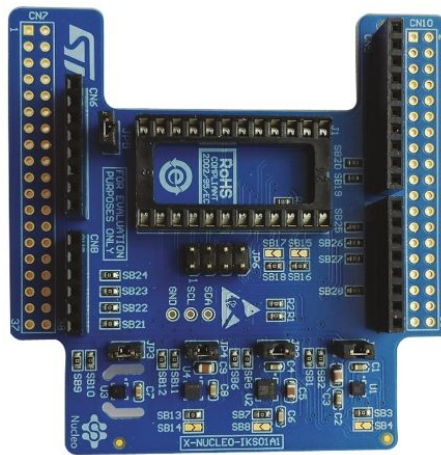


Figura 2: Módulo sensor IKS01A2.

- Gateway LoRa nas proximidades, conectado ao servidor de rede da TTN ou uma rede comercial;
- Outros sensores ou dispositivos (se necessário).

2.2 Software

- Sistema operacional Linux (Ubuntu 18.04 ou superior, ou Mint 19.3 ou superior) [3];
- Software LoraMac [4];
- Cmake;
- Cutecom.

3 Preparação do Sistema

3.1 Baixando LoRaMAC

1. Abrir um Terminal;
2. Instalar o Cmake:
`sudo apt-get install cmake`
3. Instalar o Cutecom:
`sudo apt-get install cutecom`
4. Instalar o compilador GCC-ARM:
`sudo apt-get install gcc-arm-none-eabi`

Obs.: No Linux Mint 19.3 devemos instalar os seguintes pacotes extras:

`sudo apt-get install libnewlib-dev libnewlib-arm-none-eabi`

5. Instalar o gerenciador de versão GIT:
`sudo apt-get install git`
6. Utilize o GIT para baixar o LoRaMAC:
`git clone https://github.com/AlexandroSantos/LoraMac-IKS01A2`
7. Entrar no diretório LoraMac-IKS01A2.
`cd LoraMac-IKS01A2`
8. Permissão de execução. Neste passo indicamos ao sistema operacional que os arquivos **.sh** podem ser executados.
`sudo chmod +x *.sh`
9. Permissão de uso da porta serial. Algumas distribuições necessitam de uma liberação para conectar com a porta serial, para isso usamos o seguinte comando, lembrado que devemos substituir **usuario** pelo seu nome de usuário:
`sudo usermod -a -G dialout usuario`

Obs.: Caso não saiba o seu nome de usuário, digite no terminal o seguinte comando: `users`

No momento estamos com o sistema pronto para começar a configurar nossa aplicação, bastando apenas conectar o kit da STMicroelectronics (Discovery kit) e seguir a Seção 3.2 a seguir.

Atenção: Caso o usuário não use o Ubuntu, alguns erros, por falta de bibliotecas, podem ocorrer. Tais bibliotecas devem ser instaladas, conforme cada distribuição Linux.

3.2 Preparando o Discovery Kit da STMicroelectronics

1. Conectar o cabo usb;
2. Após conectar, o Ubuntu deverá alertar sobre a inserção de um disco removível (`DIS_L072Z`).
3. Uma vez que o kit foi reconhecido, devemos abrir um navegador de internet e criar uma aplicação para definir as configurações necessárias de forma a conectar nosso hardware à rede IoT.

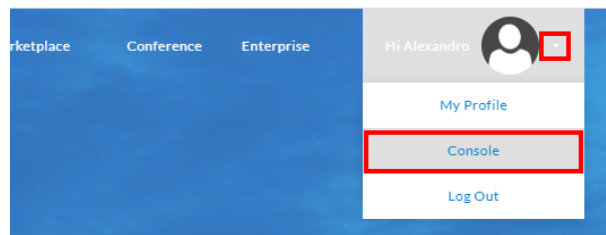
Atenção: Para que o disco removível (o kit) realmente fique disponível para gravação, é necessário que você abra pelo menos uma vez, após conectar o kit. Isto equivale a dizer que devemos montar o dispositivo para uso.

4 Criando uma Aplicação na TTN

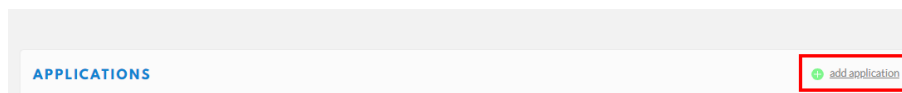
A próxima etapa é criar uma aplicação na TTN.

1. Abrir o site da **TTN**;
2. Criar uma conta (SIGN UP) e acessá-la:

3. Entrar em Console (seta ao lado da foto do usuário);



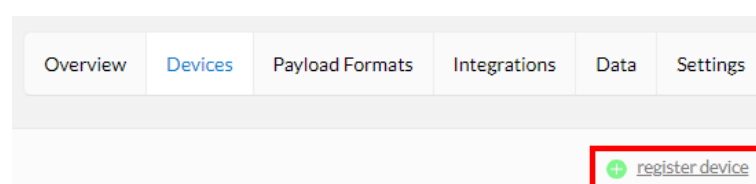
4. Acessar APPLICATIONS para abrir, ou criar, suas aplicações;
5. Criar um nova aplicação em **add application**;



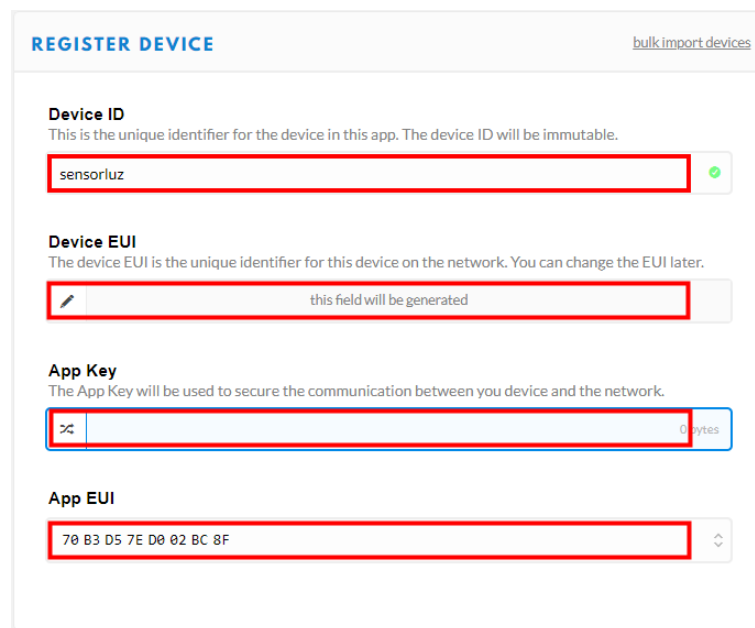
6. Preencher os campos e clicar em **Add application**;

A screenshot of the 'ADD APPLICATION' form. The form has four sections: 'Application ID' with the value 'letrasnumeros', 'Description' with the value 'Sensores para monitoramento de luz solar', 'Application EUI' with the value 'EUI issued by The Things Network', and 'Handler registration' with the value 'ttn-handler-brazil'. Each input field has a green checkmark icon to its right. At the bottom right, there is a green 'Add application' button (highlighted with a red box) and a 'Cancel' button.

7. Clicar no botão **Devices** para adicionar um dispositivo;
8. Clicar em **register device**;

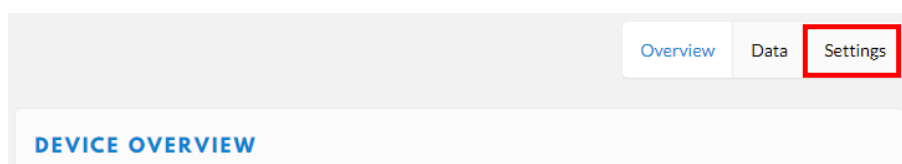


9. Preencher o campo **Device ID** como for mais conveniente;



Se você já possui um Device EUI e uma App Key, basta clicar na figura da caneta para inserir os valores. Caso não tenha, a TTN gerará uma App Key e uma Device EUI.

10. Clicar em **Settings**;



Nesta aba, devemos desabilitar o **Frame Counter**, responsável pela contagem de pacotes enviados. Esta opção implica em não disponibilização dos dados até que os contadores do **nó** e do site (aplicação) fiquem sincronizados. Por exemplo, se o contador da aplicação estiver em 100, somente após o centésimo pacote os dados começam a ser apresentados. Este método pode ser utilizado, após a estruturação do projeto, como forma de não receber dados repetidos. Mas em casos onde o dispositivo é reinicializado várias vezes para testes, pode ser muito inconveniente.



11. Definir o **Activation Method** em ABP;

Após ativar o modo ABP, clique em **Save**. Após este passo, já temos todas as informações para configurar o software LoraMac, baixado do repositório.

DEVICE OVERVIEW

Application ID letrasnumeros

Device ID sensorluz

Activation Method ABP

Device EUI <> ⇌ 00 17 AA 38 F8 E0 78 14 📄

Application EUI <> ⇌ 70 B3 D5 7E D0 02 BC 8F 📄

Device Address <> ⇌ 26 03 14 D7 📄

Network Session Key <> ⇌ 🔒 0F 78 42 D4 25 EC 0B CC 49 A5 CA 54 57 E7 97 8B 📄

App Session Key <> ⇌ 🔒 6E 19 04 1B 3D FF 5A 0B B9 AE 61 81 4A 73 C5 89 📄

Status ● never seen

Frames up 0 [reset frame counters](#)

Frames down 0

Figura 3: Informações para configurar o LoraMac

Atenção: Alteração de servidor na **aplicação** implicará em perda da configuração do dispositivo, portanto é importante manter os dados da tela acima guardados para futuras alterações. O mesmo ocorre quando alteramos algo no dispositivo, como o Device Address não pode ser editado, é sempre gerado pela TTN, pode haver a necessidade de recompilar o projeto com as novas especificações.

Os dados que aparecem em **Device Overview**, como ilustrado na Figura 3, serão usados na edição do firmware a ser compilado e gravado no dispositivo.

5 Configurando o Código Fonte LoRaMac

Uma vez obtidas as chaves e outras informações relevantes para conexão à rede IoT, precisamos inserir estas informações no código fonte, de forma a ser interpretado corretamente pelo compilador.

Dentre os arquivos, fornecidos pelo desenvolvedor original, temos o *main.c* e o *Commissioning.h*. No arquivo *Commissioning.h* encontramos as constantes de configuração de rede, tais como o intervalo de envio, a faixa de frequência utilizada pelo rádio e a subbanda, que deverá ser definida conforme a operadora a ser utilizada. Já em *main.c* estão as configurações de hardware, identificação de sensores, protocolo de comunicação entre o kit e os sensores além do *payload*, o pacote a ser enviado.

Para que nosso rádio conecte-se de forma adequada ao gateway e a TTN, precisamos editar o *Commissioning.h*, seguindo os seguinte passos:

- Passo 1: Abrir, com um editor de sua preferência (gedit, nano, vi, geany, sublime), o arquivo **Commissioning.h** localizado na pasta **src/apps/LoRaMac/classA/B-L072Z-LRWAN1**. Neste arquivo, serão editadas as linhas contendo as seguinte definições (`#define`), seguindo os dados da Figura 3 (os valores em vermelho abaixo são só exemplos, siga os dados que aparecem em Device Overview na sua conta, pois dois os mais dispositivos com a mesma configuração poderão gerar dados inconsistentes para a aplicação):

- [Linha 4] `OVER_THE_AIR_ACTIVATION 0`
Desativa ao método de ativação OTAA, passado a ser ABP.
- [Linha 8] `IEEE_OUI 0x00, 0x17, 0xAA`
Utilizar os 3 primeiro bytes de **Device EUI**
- [Linha 9] `LORAWAN_DEVICE_EUI {IEEE_OUI, 0x38, 0xF8, 0xE0, 0x78, 0x14}`
Utilizar os 5 últimos bytes de **Device EUI**
- [Linha 10] `LORAWAN_JOIN_EUI {0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x02, 0xBC, 0x8F}`
Utilizar **Application EUI**
- [Linha 11] `LORAWAN_APP_KEY` e [Linha 20] `LORAWAN_APP_S_KEY`
Ambos devem ser configurados com **App Session Key** `0x6E, 0x19, 0x04, 0x1B, 0x3D, 0xFF, 0x5A, 0x0B, 0xB9, 0xAE, 0x61, 0x81, 0x4A, 0x73, 0xC5, 0x89`
- [Linha 13] `LORAWAN_NWK_KEY`, [Linha 17] `LORAWAN_F_NWK_S_INT_KEY`, [Linha 18] `LORAWAN_S_NWK_S_INT_KEY` e [Linha 19] `LORAWAN_NWK_S_ENC_KEY`
Todos devem ser configurados com **Network Session Key** `0x0F, 0x78, 0x42, 0xD4, 0x25, 0xEC, 0x0B, 0xCC, 0x49, 0xA5, 0xCA, 0x54, 0x57, 0xE7, 0x97, 0x8B`
- [Linha 16] `LORAWAN_DEVICE_ADDRESS (uint32_t) 0x260314D7`
Deve ser configurado com **Device Address**
- [Linha 21] `APP_TX_DUTYCYCLE 60000`
Nesta, configuramos o intervalo de envios, dado em milisegundos.
- [Linha 25] `SUBBAND 2`
Nesta, configuramos o intervalo de envios, dado em milisegundos.
- Passo 2: Salve o arquivo **Commissioning.h**;

Atenção: Mesmo que a TTN seja uma rede gratuita, não significa dizer que não haja limites de uso, quer para fins de teste, ou para fins comerciais. O `APP_TX_DUTYCYCLE` está setado para 1 minuto, mas este valor deve ser utilizado apenas para verificação do dispositivo. Em operações cotidianas, devemos utilizar intervalos maiores, a cada 1 hora, por exemplo, ou, então, em casos de violação de regra, como o aumento da temperatura de uma geladeira. Lembrando que na rede LoRaWAN preza-se pela economia de tráfego de dados, informando apenas o estritamente necessário.

6 Formatando o Payload

O pacote de dados a ser enviado segue um padrão, um formato pré-definido que facilitará a interpretação após o envio. Utilizamos neste exemplo, dentro do arquivo *main.c*, o seguinte formato:

```
//***** Preparando o pacote para o envio *****//
float temperature=HTS221_Temperature(CELSIUS); //captura a temperatura em Celsius do sensor
float humidity=HTS221_Humidity(); //captura a umidade relativa do sensor
float pressure=LPS22HB_Pressure(); //captura a pressão em hPa do sensor
float volts= (float)(AdcReadChannel(&AnalogIn_PA_0, 1)/100.0); //captura a tensão no pino PA_0 do kit

//***** Determina o tamanho do pacote *****//
AppDataSizeBackup = AppDataSize = 9; //define o tamanho, em bytes, do pacote

//***** Configura o pacote *****//
AppDataBuffer[0] = (int)temperature; //salva a parte inteira da temperatura
AppDataBuffer[1] = (int)((temperature-AppDataBuffer[0])*10); //salva a parte fracionária da temperatura

AppDataBuffer[2] = (int)humidity; //salva a parte inteira da umidade
AppDataBuffer[3] = (int)((humidity-AppDataBuffer[2])*10); //salva a parte fracionária da umidade

AppDataBuffer[4] = (int)(pressure/100); //salva as centenas da parte inteira da pressão
AppDataBuffer[5] = (int)(pressure-(AppDataBuffer[4]*100)); //salva as dezenas e unidades da parte inteira da pressão
AppDataBuffer[6] = (int)((pressure-(int)(pressure))*10); //salva a parte fracionária da pressão

AppDataBuffer[7] = (int)volts; //salva a parte inteira da tensão
AppDataBuffer[8] = (int)((volts-AppDataBuffer[7])*10); //salva a parte fracionária da tensão
//*****//
```

6.1 Recuperando os Dados

Para recuperar a informação original da temperatura, umidade e tensão usamos respectivamente:

```
float temperature=AppDataBuffer[0]+AppDataBuffer[1]/10 ;
float humidity=AppDataBuffer[2]+AppDataBuffer[3]/10 ;
float volts=AppDataBuffer[7]+AppDataBuffer[8]/10 ;
```

Para a pressão devemos usar:

```
float pressure=AppDataBuffer[4]*100+AppDataBuffer[5]+AppDataBuffer[6]/10 ;
```

Existem inúmeras formas de montar o pacote de dados, mas, devemos sempre utilizar a forma mais compacta, assim reduzimos o tráfego de dados.

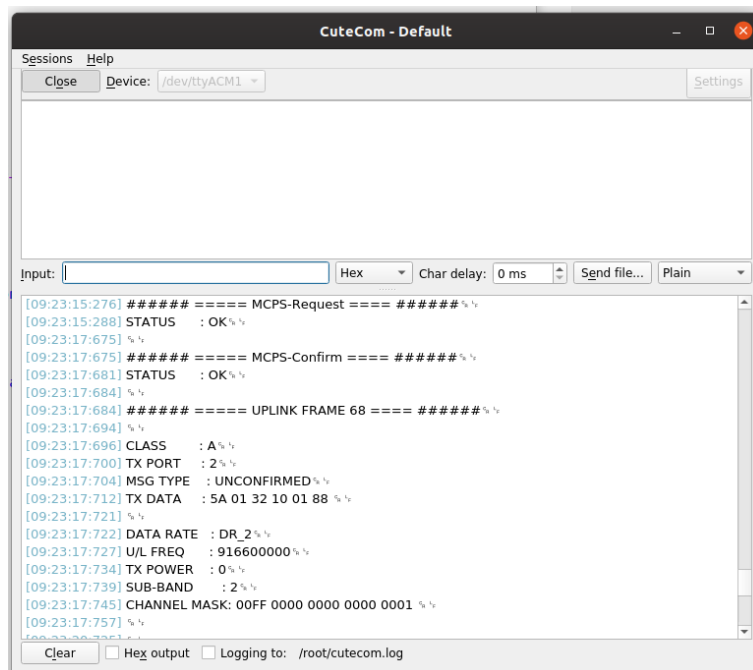
7 Compilando o Projeto e Verificando

No diretório raiz do projeto encontramos os scripts **create.sh** e **program.sh**. Para apenas compilar use **./create.sh**, este pequeno script criará o diretório **build**. No caminho **build/src/apps/LoRaMac**, você encontrará o arquivo **LoRaMac-classA.bin**. Este deverá ser copiado para o disco removível **DIS_L072Z**, caso queira rodar o programa na placa.

Para compilar e programar a placa você pode utilizar **./program.sh**, que todos os passos serão feitos automaticamente.

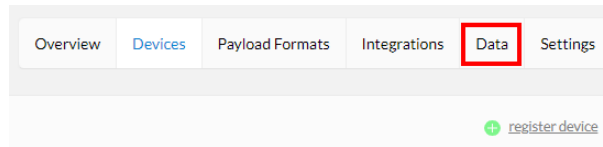
7.1 Verificação Local

Utilizando o programa Ccutecom (ou outro software terminal de serial de sua preferência), podemos observar o comportamento do rádio, quais pacotes foram transmitidos e recebidos. Ao abrir o Ccutecom você deve selecionar o **Device** (estará em alguma das portas seriais **/dev/ttyACM***) e pressionar o botão **Open**. Com o rádio conectado, as informações começarão a rolar a cada 5 segundos, tempo este pré-configurado no software LoraMac. Você pode pressionar o botão de reset no kit para ver as mensagens iniciais, que indicam os parâmetros configurados em **Commissioning.h**.



7.2 Verificação Remota

No site da TTN, com a aplicação aberta, pressione o botão **Data**.



Agora, podemos monitorar os pacotes recebidos e enviados pelo dispositivo, como na Figura 4. Também podemos verificar várias informações, como parâmetros de transmissão usados pelo rádio, potência recebida no gateway, razão sinal ruído, entre outros. Para isso devemos pressionar a seta azul no início da mensagem.

A screenshot of the 'APPLICATION DATA' table in the TTN interface. The table has columns for 'time', 'counter', 'port', 'payload', and 'potValue'. It shows a list of data packets with expandable rows indicated by blue arrows. The 'Filters' section at the top includes 'uplink', 'downlink', 'activation', 'ack', and 'error'.

time	counter	port	payload	potValue
09:21:22	0			
09:21:21	45	2	payload: 5A 01 31 0E 02 06	potValue: 1.1602243076218814
09:21:07	0			
09:21:07	42	2	payload: 5A 01 3C 1D 02 7B	potValue: 1.1602243076218814
09:20:37	0			
09:20:36	36	2	payload: 5A 01 2F 29 04 81	potValue: 1.1602243076218814
09:20:32	0			
09:20:31	35	2	payload: 5A 01 37 3C 01 13	potValue: 1.1602243076218814
09:20:21	0			
09:20:21	33	2	payload: 5A 00 FC 25 01 59	potValue: 1.1601739528496222
09:19:12	0			
09:19:12	19	2	payload: 5A 01 1A 45 04 4F	potValue: 1.1602243076218814

Figura 4: Dados transmitidos e recebidos

Repare que o payload é apresentado como uma sequência de números em hexadecimal, de tamanho 9 bytes, conforme definido no arquivo *main.c* previamente. Para visualização dos dados em um formato legível, podemos configurar esta apresentação na aba **Payload Formats**.

Código de recuperação de dados:

```
function Decoder(bytes, port) {  
  var decoded = {};  
  //Decode bytes to int  
  var temperatureInt = bytes[0] + bytes[1]/10;  
  var humidityInt = bytes[2] + bytes[3]/10;  
  var pressureInt = bytes[4]*100 + bytes[5] + bytes[6]/10;  
  var voltageInt = bytes[7] + bytes[8]/10;  
  //Decode int to float  
  decoded.temperature = temperatureInt; // Temperature in C  
  decoded.humidity = humidityInt; //Humidity in %  
  decoded.pressure = pressureInt; //Pressure in hPa  
  decoded.voltage = voltageInt; //Voltage in V  
  return decoded;  
}
```

Copie e cole na aba indicada, conforme a Figura 5.

PAYLOAD FORMATS

Payload Format

The payload format sent by your devices

Custom

decoder

converter

validator

encoder

```
1 function Decoder(bytes, port) {  
2   var decoded = {};  
3   // Decode bytes to int  
4   var temperatureInt = bytes[0] + bytes[1]/10;  
5   var humidityInt = bytes[2] + bytes[3]/10;  
6   var pressureInt = bytes[4]*100+bytes[5]+bytes[6]/10;  
7   var voltageInt = bytes[7]+bytes[8]/10;  
8   // Decode int to float  
9   decoded.temperature = temperatureInt; // Temperature in °C  
10  decoded.humidity = humidityInt;      // Humidity in %  
11  decoded.pressure = pressureInt;       // Pressure in hPa  
12  decoded.voltage = voltageInt;         // Voltage in V  
13  return decoded;  
14 }
```

Figura 5: Decodificação do pacote de dados.

Agora na aba **APPLICATION DATA**, expandindo a visualização dos dados, podemos observar a informação devidamente representada de forma legível ao usuário.

The screenshot shows the 'APPLICATION DATA' interface. At the top, there are buttons for 'pause' and 'clear'. Below this, there are filter buttons: 'uplink', 'downlink', 'activation', 'ack', and 'error'. A table displays data points with columns for 'time', 'counter', and 'port'. One data point is expanded, showing 'payload: 1C 06 49 05 0A 0F 03 03 02' and 'humidity: 73.5'. The expanded view includes sections for 'Uplink', 'Payload' (showing the hex string '1C 06 49 05 0A 0F 03 03 02'), 'Fields' (showing a JSON object with humidity, pressure, temperature, and voltage), and 'Metadata' (showing a timestamp).

time	counter	port
20:38:18	0	1

payload: 1C 06 49 05 0A 0F 03 03 02 humidity: 73.5

Uplink

Payload

1C 06 49 05 0A 0F 03 03 02

Fields

```
{  
  "humidity": 73.5,  
  "pressure": 1015.3,  
  "temperature": 28.6,  
  "voltage": 3.2  
}
```

Metadata

```
{  
  "time": "2020-03-28T23:38:18.953890458Z"  
}
```

Figura 6: Expandindo os dados.

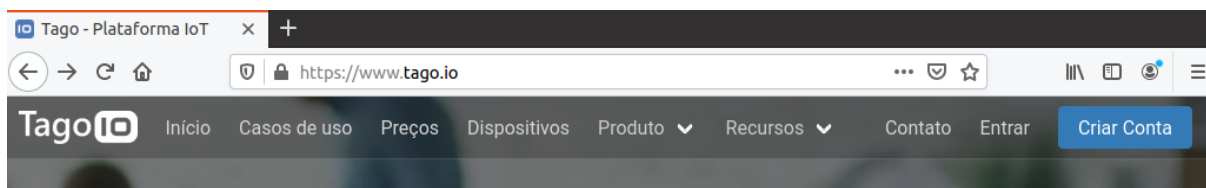
8 Interface Gráfica - Dashboard

Uma vez configurado e comprovado o seu funcionamento, a apresentação dos dados obtidos ou tomada de decisão torna-se uma etapa importante para que o usuário final.

Uma ferramenta, combinada com plataforma da TTN, pode ser utilizada para a construção, visualização e publicação dos dados de uma forma legível e interativa. A solução apresentada pela **TagoIO** traz, de forma gratuita, alguns recursos para podermos avaliar e começar a compreender o porquê da importância e a abrangência eminente dessas ferramentas e suas aplicações.

8.1 Criando uma Conta

Acessando o site da **TagoIO**, devemos pressionar o botão **Criar Conta**.



A tela abaixo será apresentada e preenchendo os campos solicitados criamos uma conta que permitirá o acesso a algumas funções e ferramentas disponibilizadas pelos idealizadores do site.

Após pressionar o botão **Registre-se**, uma mensagem será enviada para sua caixa de e-mail, seguindo as instruções, informadas na mensagem, sua conta será ativada.

8.2 Acessando sua Conta

Agora com sua conta ativa, devemos acessá-la. Insira o e-mail, utilizado no registro, e a senha definida. Depois, pressione o botão **Autentique-se**.

Uma vez autorizado, a tela principal será apresentada (Figura 7), nela podemos encontrar informações de uso e equipamentos. Vale lembrar que, em uma conta gratuita, os limites de transações entre os dispositivos e a interface são pequenos.

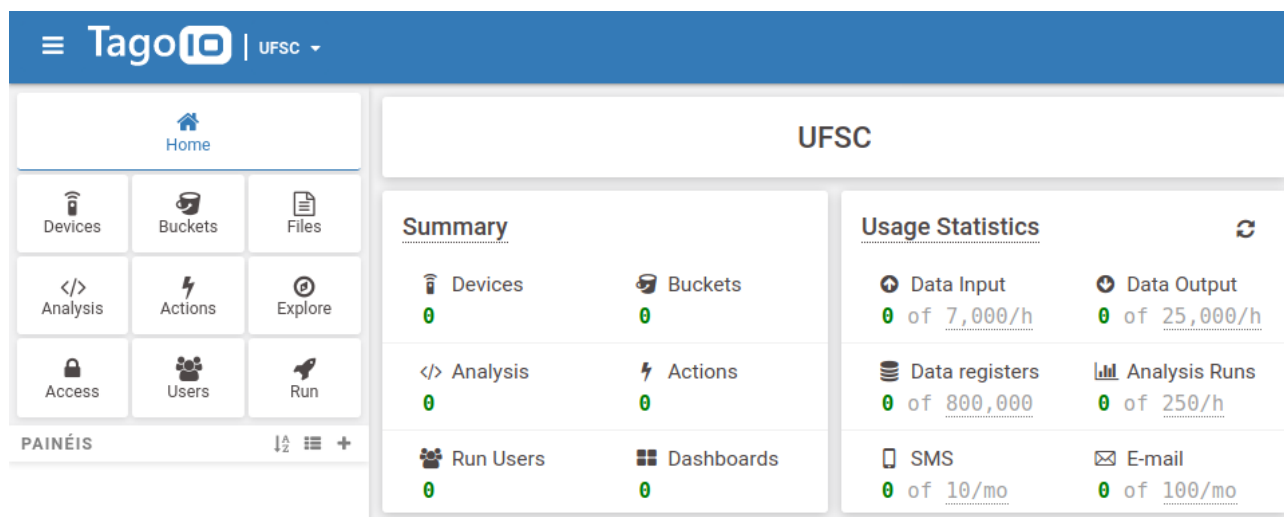
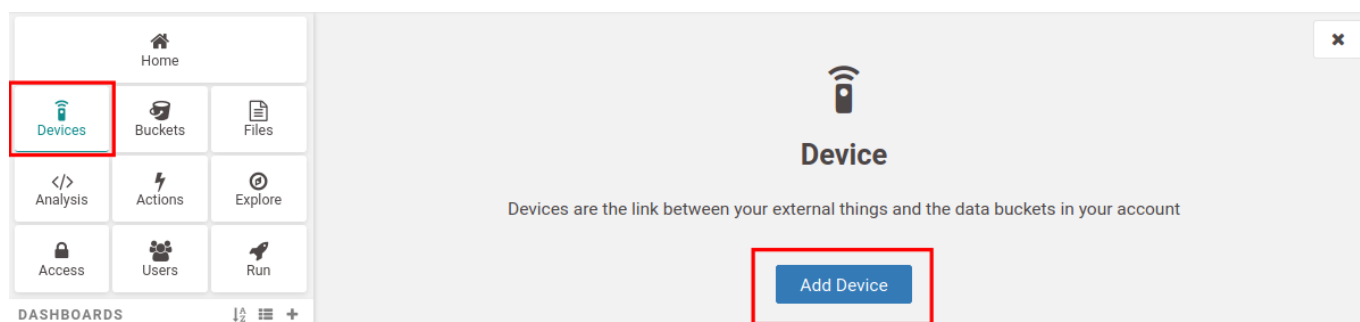


Figura 7: Tela principal.

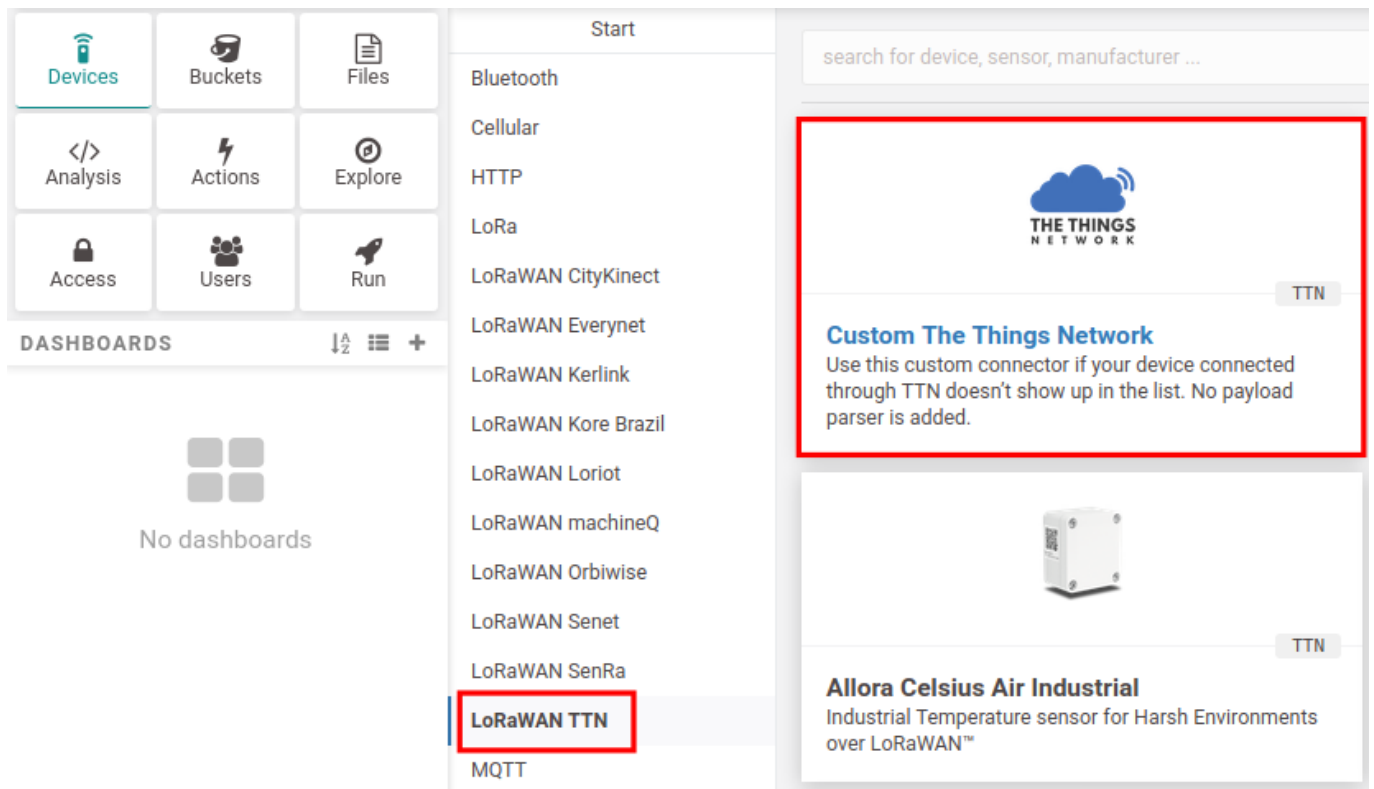
Assim como no site da TTN, devemos configurar a aplicação. Usaremos os dados recebidos pela TTN e enviados à TagoIO, mostrando-os numa interface gráfica que facilitará sua interpretação e tomada de decisão.

8.3 Conectando a TagoIO com a TTN

Como no exemplo da TTN, aqui também devemos adicionar um dispositivo. Para isso, utilizando a aba esquerda, pressionamos em **Devices** depois em **Add Device**.



Agora vinculamos o dispositivo à TTN.

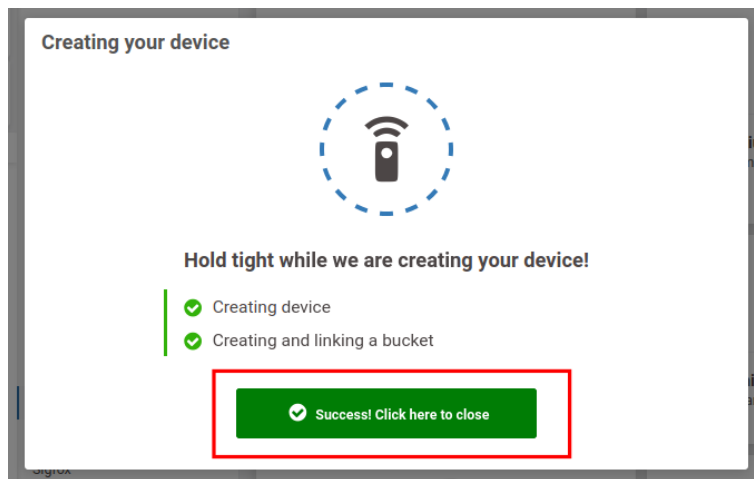


No site da TTN podemos ter cadastrados diversos dispositivos, então precisamos direcionar a um específico. Para isso, devemos preencher a solicitação, conforme a Figura 8, onde o campo **Device EU** identifica o dispositivo.

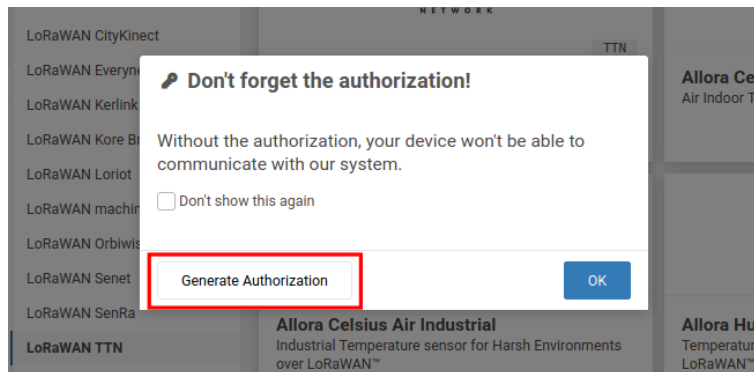
The screenshot displays the 'Complete your configuration' form for 'Custom The Things Network'. The form has a teal header with the TTN logo and title. Below the header, there are two main sections. The first section, 'Device name', has a text input field containing 'sensor'. The second section, 'Device EU', has a text input field containing a 15-digit hexadecimal string: '3132373758378805'. To the right of the 'Device EU' field, there is a 'Scan Qr Code' button. Below the input fields, there is a 'Connector [Integration]' section with a dropdown menu showing 'Custom The Things Network [TTN]'. At the bottom of the form, there is a 'Cancel' button on the left and a 'Create device' button on the right, which is highlighted with a red box. A diagram at the bottom of the form illustrates the connection between a LoRa device, the TTN network, and a TagoIO connector.

Figura 8: Direcionando ao dispositivo.

Uma vez adicionado os dados necessários, pressionamos o botão **Create device** e a tela abaixo será apresentada em caso de sucesso na montagem do dispositivo dentro da TagoIO.



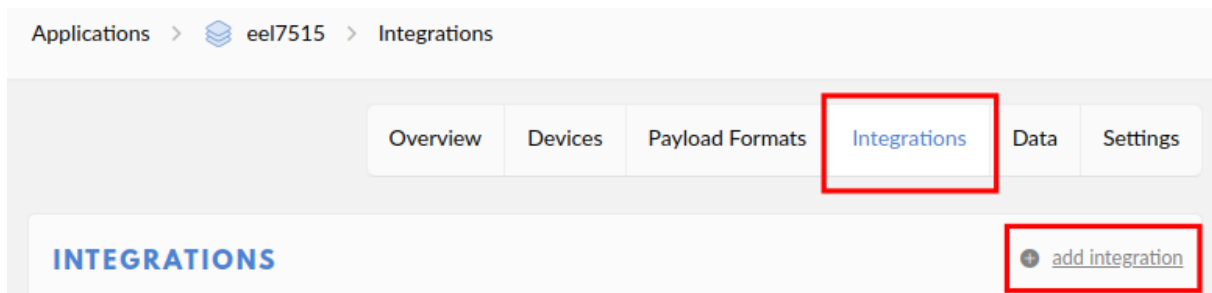
Agora que temos uma ligação entre a TagoIO e a TTN, devemos criar uma chave de segurança para que a troca de dados ocorra de forma correta sem a possibilidade de haver a sobreposição de dados das diversas aplicações criadas na TTN.



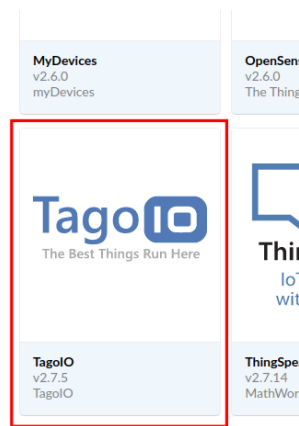
Após pressionarmos o botão **Generate Authorization**, devemos adicionar um nome à chave a ser criada, no campo **Name**, a seguir solicitar a geração no botão **Generate**.

Com a chave criada, devemos informar, no site da TTN, qual chave devemos enviar para TagoIO a fim de que as informações sejam contabilizadas no dispositivo criado. Como a chave possui uma sequência de letras e números extensa, por exemplo **at6f6eff80a28c442e82846d498e36a991**, podemos usar o botão **Copy** e copiá-lo para área de transferência, evitando assim um possível erro de digitação.

De volta ao site da **TTN**, temos que criar uma integração entre os sites, para isso dentro da aplicação devemos abrir a aba **Integrations** e adicionar uma nova interação com o botão **add integration**.




Uma lista de possíveis integradores será aberta e devemos escolher a TagoIO.



Selecionada a integração, agora inserimos um identificador único qualquer (**Process ID**), tipo de chave de acesso (neste caso **default key**), e a autorização, que está armazenada na área de transferência, bastando usar **Ctrl+V**.

A Figura 9 mostra um exemplo deste preenchimento. Configuração feita, devemos pressionar o botão **Add integration**.

ADD INTEGRATION



The Best Things Run Here

TagoIO (v2.7.5)
TagoIO

Tago offers the tools for your business to manage devices, store data, run analytics and integrate services. It combines everything with an easy-to-use application and user management system. This integration allow bi directional communication with the tago.io platform. You will have access to payload, decoded payload fields and metadata to build cool IoT platforms. Also, you can send downlink messages via the dashboard.

[documentation](#)

Process ID
The unique identifier of the new integration process

Access Key
The app access key

default key

devices

messages

Authorization
The value of the Authorization header

Cancel

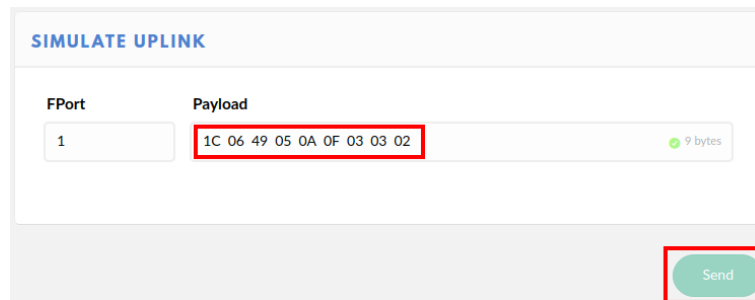
Add integration

Figura 9: Configurando a integração (Site TTN).

A partir deste ponto os sites estarão integrados e os dados serão transmitidos para TagoIO. Lembrando que agora serão contabilizados os pacotes de dados e, para evitar o consumo desnecessário, caso vários dispositivos estejam conectados, mantê-los desativados até o término da criação da interface, sendo que a TagoIO também possui um simulador para teste da mesma.

8.4 Simulando o Envio de Dados

Para verificar se a ligação foi devidamente construída, podemos utilizar na aba **Devices**, do site da TTN, o simulador de recebimento de dados.

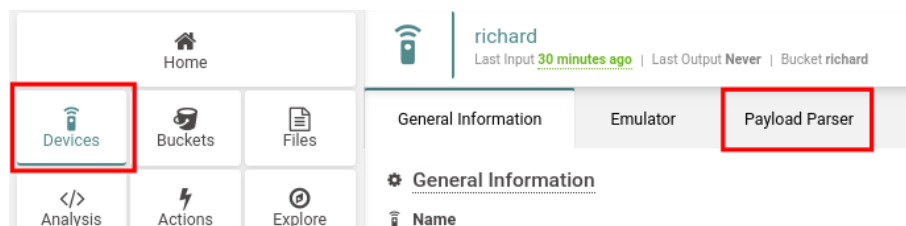


Ex. 1: O pacote 1C0649050A0F030302_{Hex} representa uma temperatura de 28.6 °C, pressão atmosférica de 1015.3 hPa, humidade de 73.5 % e tensão de 3.2 V.

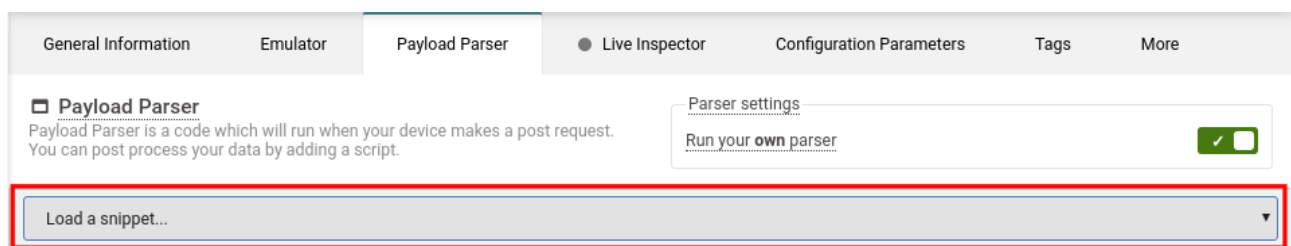
Usando o exemplo citado, pode-se testar tanto o envio para a TagoIO quanto o decodificador de payload no site da TTN.

8.5 Interpretando os Dados

Como visto na TTN, os dados estão codificados e devem ser interpretados pelo site. Tendo isto em mente, devemos criar uma forma de decodificação dos campos enviados pelos dispositivos, no nosso caso temperatura, umidade, pressão atmosférica e tensão.



Acessando a aba **Payload Parser** podemos encontrar alguns exemplos e maneiras de decodificar e criar novas variáveis, necessárias para a aplicação.



Utilizando como base o **Parse Example**, que pode ser observado pressionando a área destacada como indicado acima, suprimimos variáveis sem uso e acrescentamos as de nosso interesse. Dentro do código editado, existe uma indicação onde ele sofreu alguma alteração.

```
//*****  
//***** Início da alteração do exemplo *****  
//*****  
Código alterado  
//*****  
//***** Fim da alteração do exemplo *****  
//*****
```

Segue o código completo que deve ser substituído, no lugar do exemplo, e gravado utilizando o botão **Save** no final da página.

```

/* This is an generic payload parser example.
** The code find the payload variable and parse it if exists.
**
** IMPORTANT: In most case, you will only need to edit the parsePayload function.
**
** Testing:
** You can do manual tests to this parse by using the Device Emulator. Copy and Paste the following code:
** [{ "variable": "payload", "value": "1E033F030A00030207" }]

const ignore_vars = [];

/**
 * This is the main function to parse the payload. Everything else doesn't require your attention.
 * @param {String} payload_raw
 * @returns {Object} containing key and value to TagoIO
 */
function parsePayload(payload_raw) {
  try {
    // If your device is sending something different than hex, like base64, just specify it bellow.
    const buffer = Buffer.from(payload_raw, 'hex');

    //*****
    //***** Início da alteração do exemplo *****
    //*****
    // Lets say you have a payload of 9 bytes.
    // 0,1 - Temperature
    // 2,3 - Humidity
    // 4,5,6 - Pressure
    // 7,8 - Voltage
    // More information about buffers can be found here: https://nodejs.org/api/buffer.html
    const data = [
      { variable: 'temperature', value: buffer.readInt8(0) + buffer.readInt8(1)/ 10, unit: 'C' },
      { variable: 'humidity', value: buffer.readUInt8(2)+buffer.readInt8(3)/ 10, unit: '%' },
      { variable: 'pressure', value: buffer.readUInt8(4)*100+buffer.readUInt8(5)+buffer.readInt8(6)/ 10, unit: 'hPa' },
      { variable: 'voltage', value: buffer.readUInt8(7)+buffer.readInt8(8)/ 10, unit: 'V' },
    ];
    //*****
    //***** Fim da alteração do exemplo *****
    //*****

    return data;

  } catch (e) {
    console.log(e);
    // Return the variable parse_error for debugging.
    return [{ variable: 'parse_error', value: e.message }];
  }
}

// Remove unwanted variables.
payload = payload.filter(x => !ignore_vars.includes(x.variable));

// Payload is an environment variable. Is where what is being inserted to your device comes in.
// Payload always is an array of objects. [ { variable, value...}, {variable, value...} ...]
const payload_raw = payload.find(x => x.variable === 'payload_raw' || x.variable === 'payload' || x.variable === 'data');
if (payload_raw) {
  // Get a unique serie for the incoming data.
  const { value, serie, time } = payload_raw;

  // Parse the payload_raw to JSON format (it comes in a String format)
  if (value) {
    payload = payload.concat(parsePayload(value).map(x => ({ ...x, serie, time: x.time || time })));
  }
}

```

8.6 Simulando o Recebimento de Dados

Uma forma de testar a interpretação de dados é utilizar a ferramenta de emulação. Na aba **Live Inspector** devemos iniciar a operação.

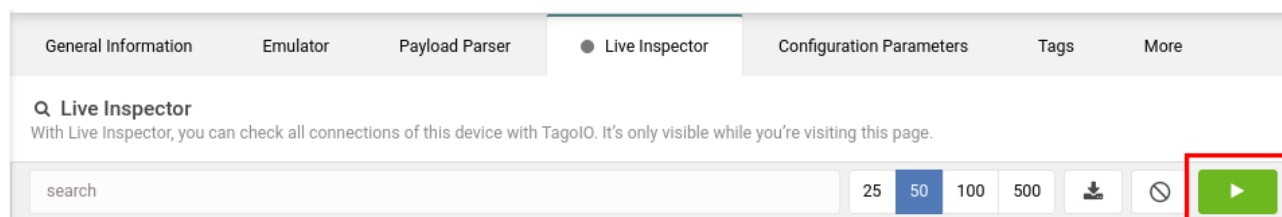


Figura 10: Ativar a simulação.

Já na aba **Emulator** podemos iniciar a variável **payload**, no formato **JSON**, que é responsável pelo transporte dos dados enviados pelos dispositivos ao site. Segue o código abaixo:

```
[
  {
    "variable": "payload",
    "value": "1E033F030A00030207"
  }
]
```

Conforme mostrado na Figura 11, pressionando a seta destacada, o servidor exibirá no campo **Server response**, a data, hora e a quantidade de dados adicionada. Aqui devemos entender que **payload** também é uma variável e será armazenada, portanto, neste exemplo 5 dados serão armazenados ao invés de 4.

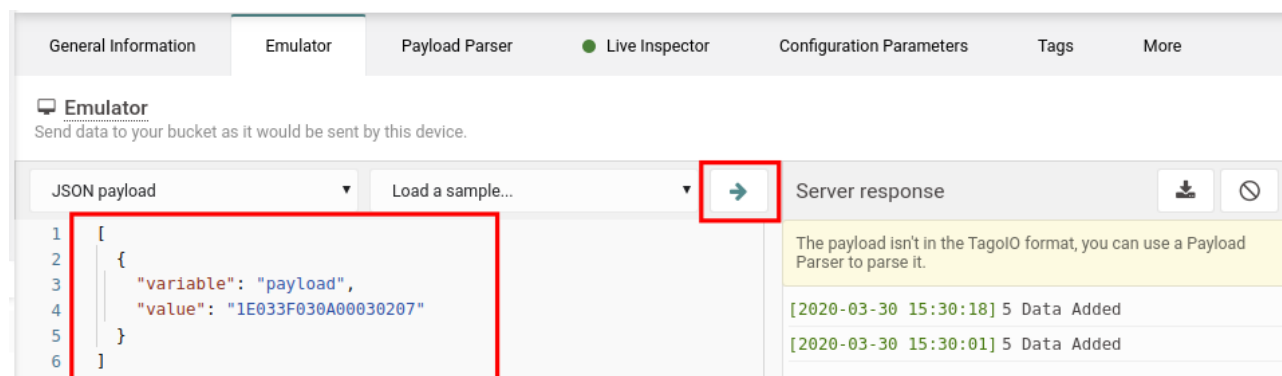
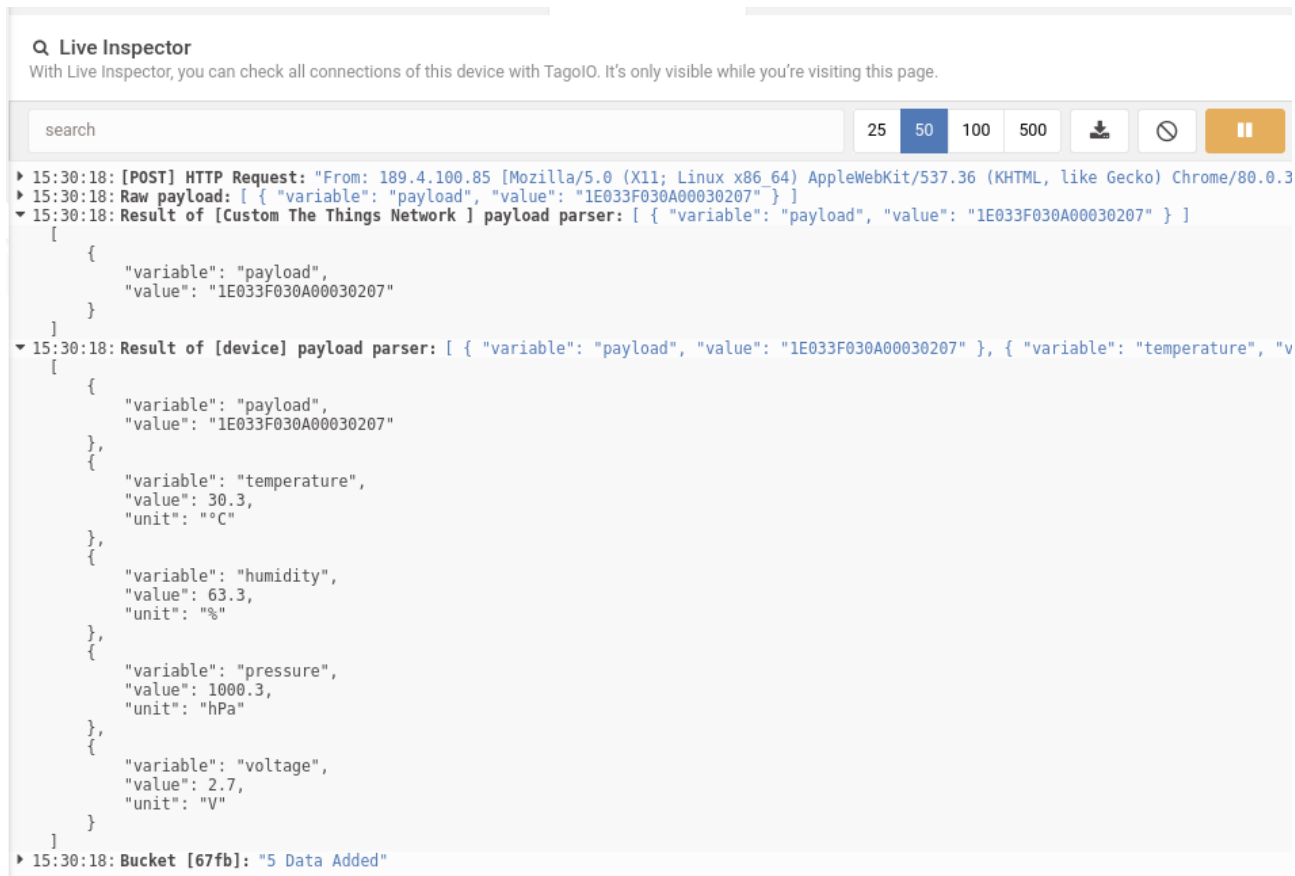


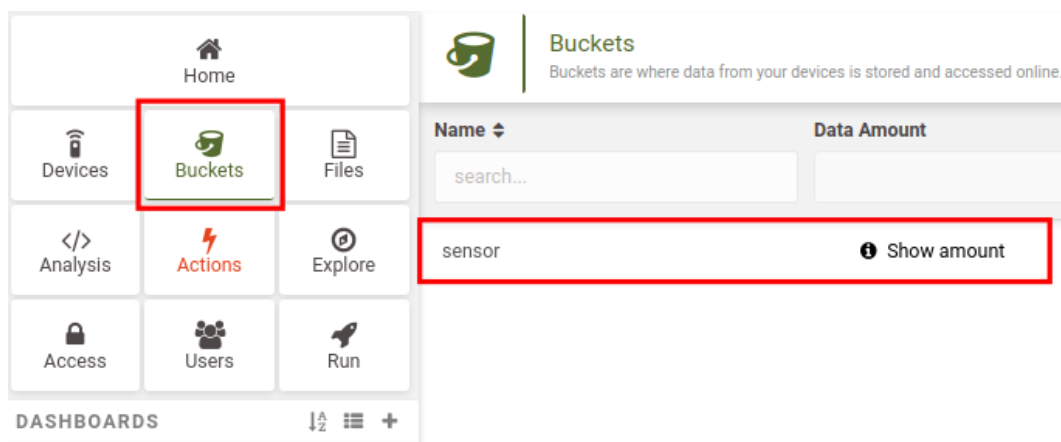
Figura 11: Configurando o pacote no emulador.

Retornando à aba **Live Inspector** é possível verificar se os dados foram devidamente identificados e as variáveis corretamente inicializadas. Neste ponto, é interessante o acionamento do dispositivo e verificar se os pacotes enviados terão o mesmo tratamento, o que significaria uma correta configuração de todos os passos anteriores e validaria a aplicação, deixando agora a cargo do desenvolvedor a montagem da tela de apresentação dos dados (ou **Dashbord**).



8.7 Armazenamento de Dados

Quando enviamos dados a TagoIO, estes ficarão armazenados por um tempo determinado. No caso de conta gratuita, por 1 (um) mês, nos chamados baldes (ou **Buckets**).



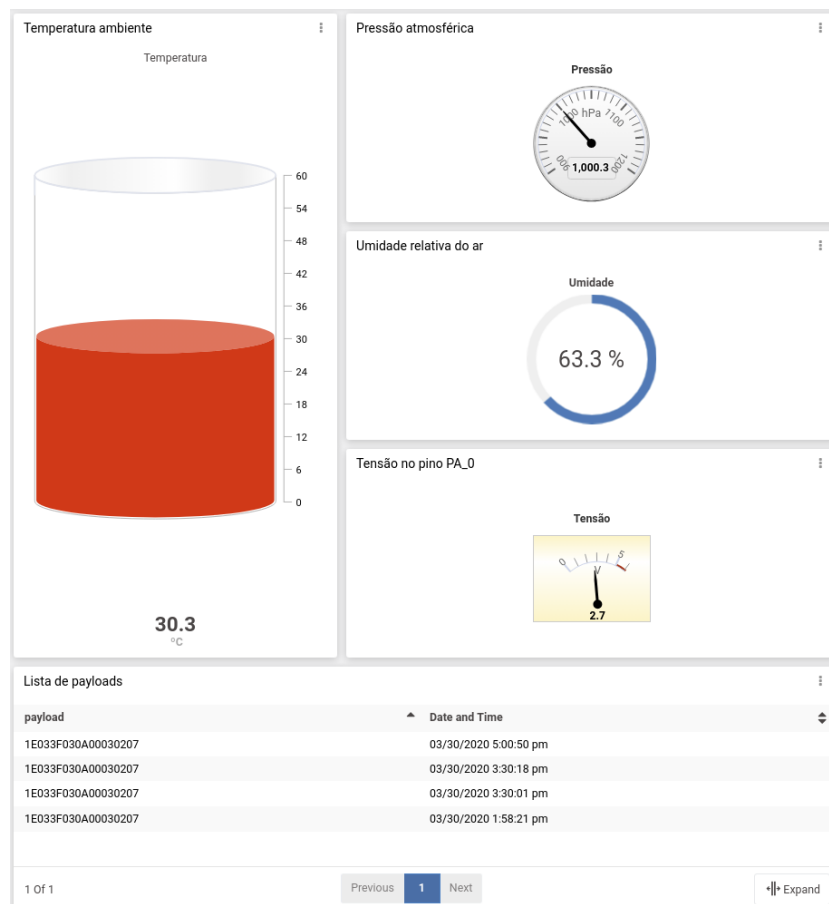
Selecionando o *Bucket*, criado automaticamente quando adicionamos o primeiro dispositivo, na aba **Variables**, estão listadas as variáveis encontradas pelo gerenciador. Se somente o simulador foi utilizado, apenas as variáveis criadas no interpretador são mostradas mas, quando

um dispositivo real é utilizado, outras variáveis aparecerão e deverão ser também tratadas, devendo ser escolhidas as que serão ou não salvas em banco de dados. Uma escolha bem aplicada implicará em menor quantidade de dados armazenados e custo de operação mais baixo.

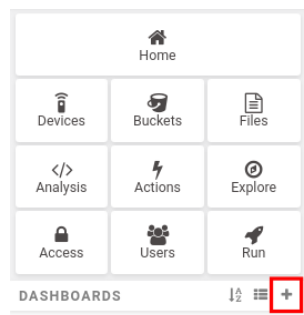
General Information	Backup	Variables	Linked Devices	Tags	More
Variable management ?					
<input type="checkbox"/> Select All		Refresh Information		Delete Selected Empty Bucket	
Variable ?	Device ?	Number of registers ?	Unit ?	Apply data retention rule ?	
search...	search...	search...	search..	search...	
<input type="checkbox"/> humidity	richard	3	%	<input checked="" type="checkbox"/>	
<input type="checkbox"/> payload	richard	3		<input checked="" type="checkbox"/>	
<input type="checkbox"/> pressure	richard	3	hPa	<input checked="" type="checkbox"/>	
<input type="checkbox"/> temperature	richard	3	°C	<input checked="" type="checkbox"/>	
<input type="checkbox"/> voltage	richard	3	V	<input checked="" type="checkbox"/>	

8.8 Dashboard - Apresentação dos Dados

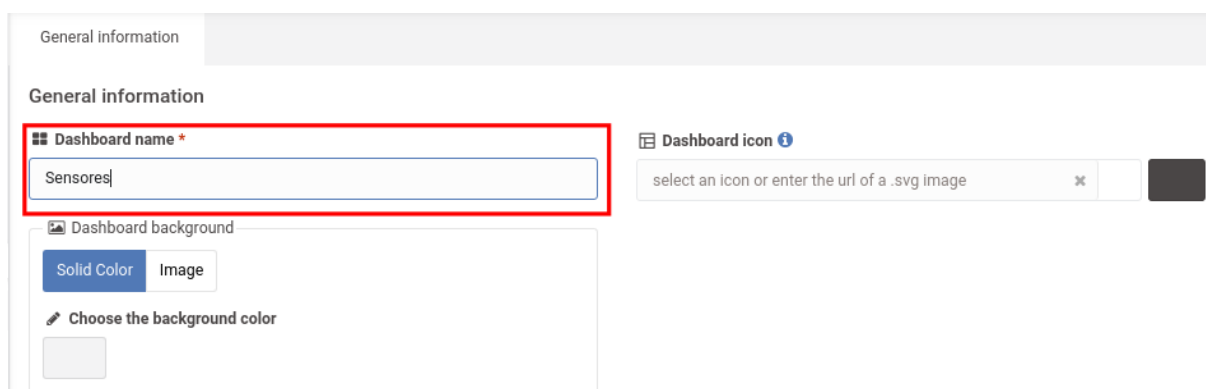
Para o usuário um dos passos mais importantes é a apresentação dos dados de forma concisa, prática e com uma visualização agradável. Então, a construção de uma interface amigável se faz necessária e aumenta o interesse no uso da ferramenta escolhida. Para o nosso estudo de caso, uma simples interface nos permite visualizar os dados recebidos dos dispositivos em uma tela onde estes são apresentados dinamicamente.



Primeiramente devemos adicionar um **dashboard**, conforme figura abaixo.



Na tela **General information**, acrescentados um nome para o dashboard e pressionamos **Save**.



Agora passaremos a construir efetivamente a apresentação, inserindo objetos de visualização de dados.

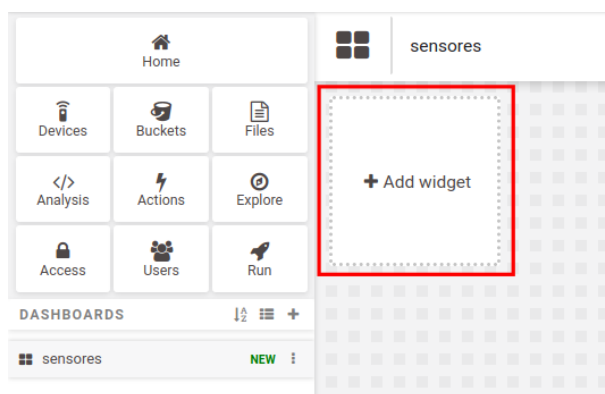


Figura 12: Configuração inicial.

Ao pressionar a área em destaque na Figura 12, uma lista de objetos é apresentada e a escolha de um deles deve-se ao tipo e forma de dados que queremos exibir. Por exemplo, para temperatura, o objeto **Cylinder** pode representar um termômetro.

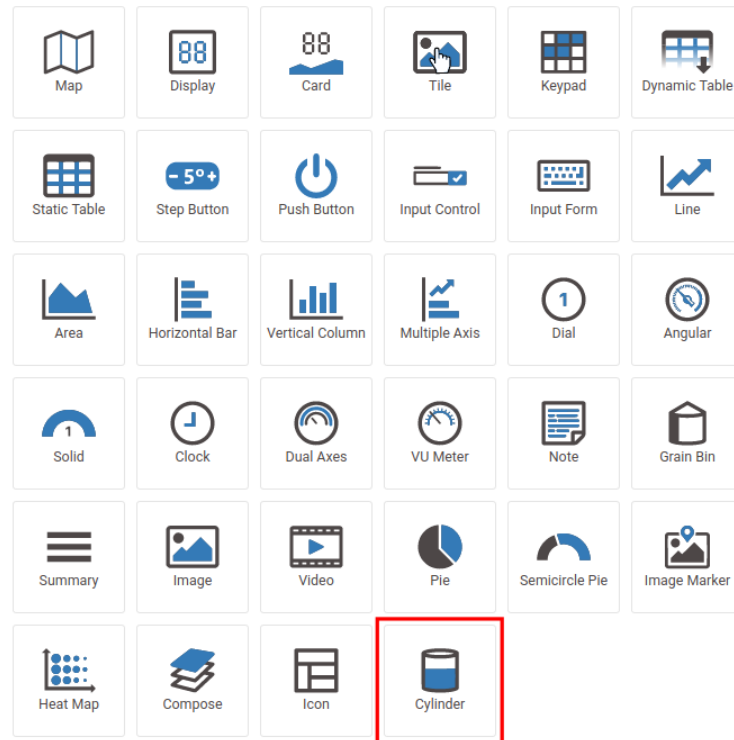


Figura 13: Objetos ou *widgets*.

Na configuração principal, entramos com um título, dispositivo, variável e a origem dos dados associada ao *widget*.

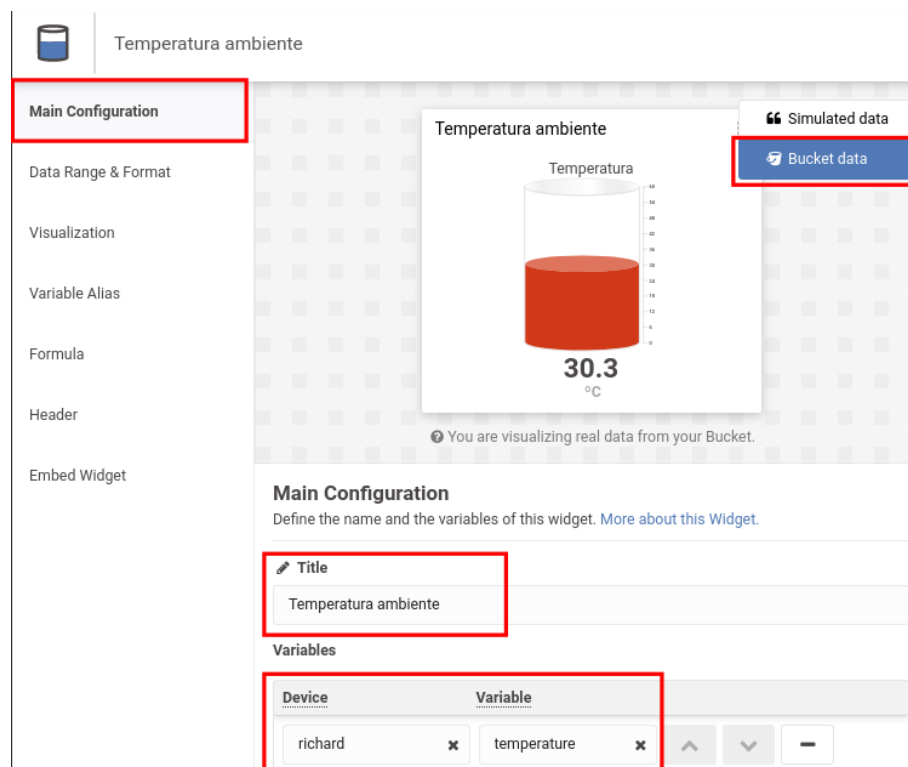


Figura 14: Criando um *widgets*.

Outras características também podem ser modificadas, como a cor, unidade física, os limites para exibição e operações, ou fórmulas, com os valores a serem apresentados.

Data Range & Format

Define the format of data and unit for variable values

Minimum ⓘ *

Maximum ⓘ *

Unit ⓘ

0

60

°C

Number format ⓘ

Variable: **temperature**, Device: **richard**

Show thousands separator ⓘ

Decimals ⓘ

1

Figura 15: Intervalo de operação do **Cylinder**.

Visualization

Define rules to cylinder

Quantity of tick marks ⓘ

5 10 20 25

Variable Conditions ⓘ

Condition ⓘ	Value ⓘ		Color ⓘ	
Less than ▼	20 *	and	enter the value	Blue -
Greater than ▼	30 *	and	enter the value	Red -
Between ▼	20 *	and	30 *	Green - +

Figura 16: Configuração das cores do **Cylinder**.

Formula

Apply a formula to modify the visualization of each variable. [More about Formula](#). To learn more about displaying units, [click here](#).

Variable: **temperature**, Device: **richard**

Enable Formula ⓘ

Formula Source ⓘ

Unit origin

Unit for variables

Formula

Output = \$VALUE\$*9/5 + 32

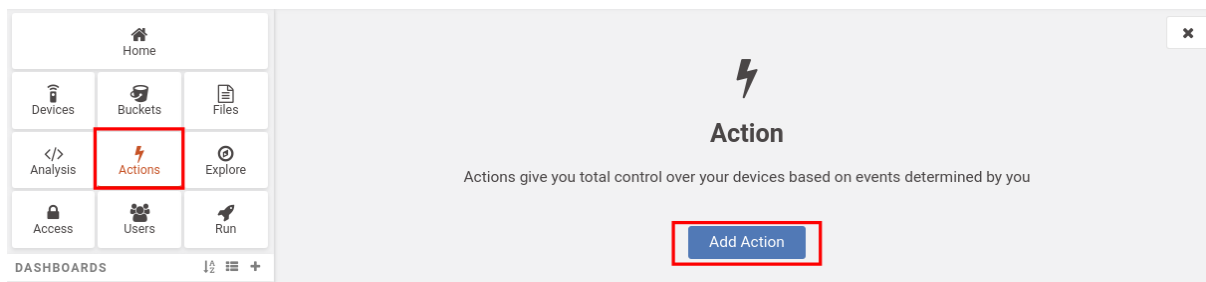
Test it

Figura 17: Fórmula para converter de Celsius para Fahrenheit.


Todos os demais *widgets* apresentados na Figura 13 podem ser configurados seguindo os mesmos passos, respeitando as particularidades de cada *widget*.


8.9 Alertas e Mensagens

Outra ferramenta disponibilizada pela TagoIO é a **Actions**, esta tem como função emitir alertas ou mensagens através de e-mail, SMS entre outras opções.





Adicionando uma nova ação, a janela abaixo será exibida, devemos preenchê-la e pressionar o botão **Create my Action**.


 **Add Action**


 **Name**


sensoraction

 **Type of trigger**


☒  **Variable**
Triggered when the selected variables meet certain conditions.


☐  **Resource**
Triggered when the selected resources change (devices, buckets, users, ...).


☐  **Schedule**
Triggered based on the selected time interval.

 **Type of action**


Send Email

 **Send to**

seuemail@provedor.com 

 **Title**

Alerta de temperatura

 **Message**

A temperatura está fora dos limites desejado.

Action is a very powerful feature that gives you total control over your devices based on events determined by you.

- **Creating Actions.** Define actions that will send Email or SMS, push a notification, post data to end-point, send data to a device using MQTT, or even trigger a script to run.
- **Real-Time.** Every action is executed in real-time. Combine Actions with Analyses and you can get to an endless number of conditions and possibilities.
- **Sending Data to Devices.** Actions can be used to send data to devices or to other external systems.

Cancel

Create my Action

26

Definido o modo de alerta, neste exemplo um e-mail, quando a temperatura exceder 40 °C a mensagem deve ser enviada. Como temos apenas um dispositivo habilitado selecionamos **Single device** e o dispositivo que configuramos.

No campo **Trigger** definimos a condição de envio, ou seja, se a temperatura for maior que 40 °C enviar a mensagem.

No campo **Trigger Unlock** definimos a condição em que uma nova mensagem pode ser enviada. Sem esta opção habilitada, a cada leitura enviada pelo dispositivo, será enviado um e-mail, o que seria redundante em caso de leituras muito próximas. Dessa forma, no exemplo, o envio somente será habilitado novamente quando a temperatura for menor que 30 °C.

No campo **Message** podemos incluir informações relacionadas à variável testada, como o valor e o momento do ocorrido.

The screenshot shows the IFTTT configuration interface for a temperature alert. The interface is divided into two main sections: **Trigger** and **Action**.

Trigger Section:

- Single device:** Selected option. A red box highlights this option and the **Select the device** dropdown menu, which currently shows "richard".
- Trigger:** A red box highlights the configuration: "If temperature is Greater than 40".
- Trigger Unlock (optional):** A red box highlights the configuration: "If temperature is Less than 30".

Action Section:

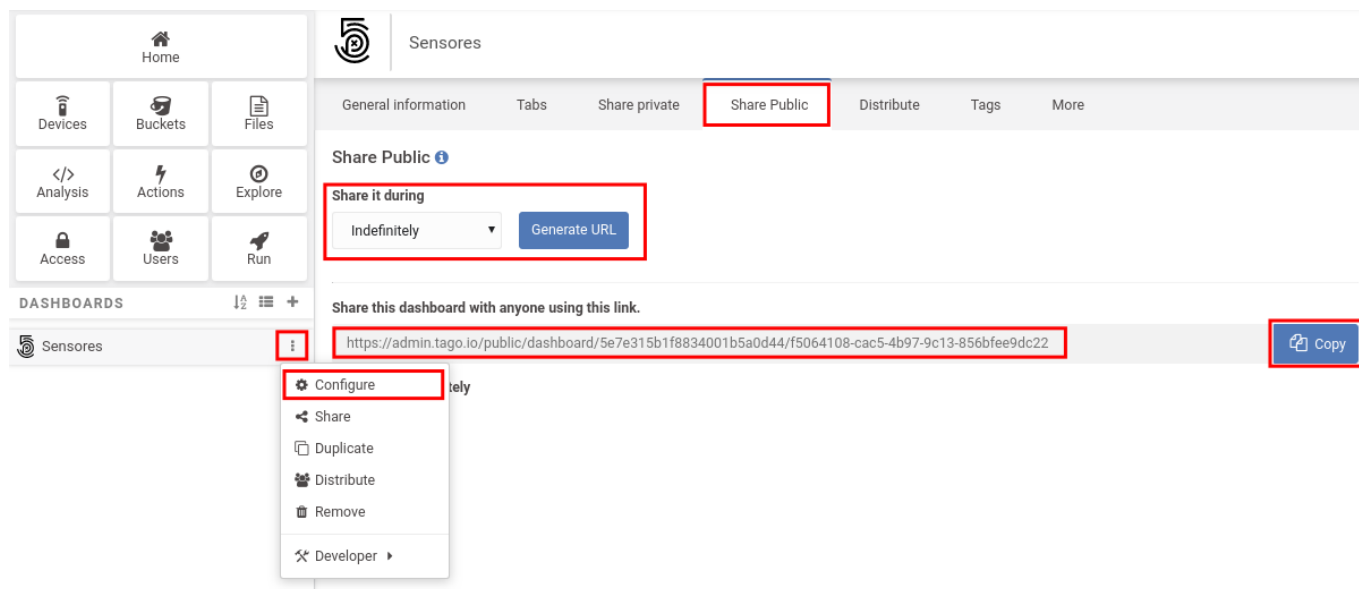
- Name:** "sensoraction".
- Type of action:** "Send Email".
- Send to:** "seuemail@provedor.com".
- Title:** "Alerta de temperatura".
- Message:** A red box highlights the message content: "A temperatura está fora do limite desejado (40 °C).
Temperatura = \$VALUE\$ em \$TIME\$".
- How to use variables on text:** A section explaining variable syntax with examples.

At the bottom of the interface, there are **Back** and **Save** buttons. The **Save** button is highlighted with a red box.

Agora nossa aplicação está devidamente configurada e operante, bastando então a publicação de forma que as informações possam ser observadas pelo usuário final.

8.10 Publicando o Dashboard

Ao finalizar e testar o *dashboard*, precisamos fornecer ao usuário uma forma de acesso aos dados. Uma maneira é utilizar um *link* gerado pela TagoIO. Este, pode ser por tempo indeterminado ou limitado, conforme a necessidade de exibição. Uma vez acessada a aba **Share Public** o endereço já está disponível, porém configurado como por tempo indeterminado, cabendo ao desenvolvedor decidir pela necessidade de alterar.

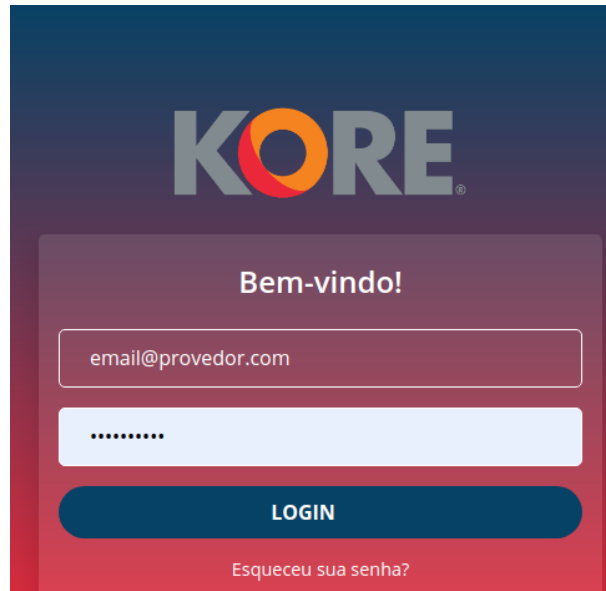


O *link* fornecido dá ao usuário apenas a possibilidade de visualização, não sendo com este possível edição ou alteração dos dados. O *link* pode ser compartilhado com diversos usuários, disseminando a informação pelo maior número de colaboradores e interessados nos dados, simultaneamente.

9 Criando uma Aplicação na Kore

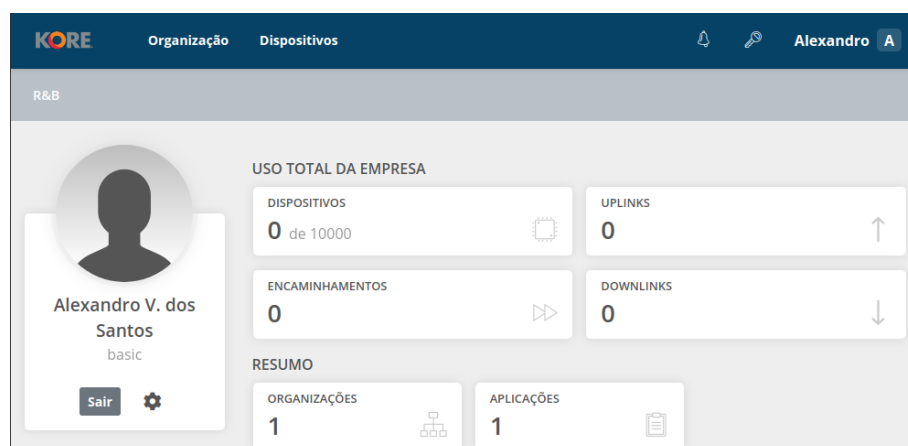
Além de uma rede gratuita como a TTN, podemos utilizar também uma rede comercial para conectar os dispositivos. Para tanto utilizaremos a plataforma da Kore sobre a rede Everynet.

1. Abrir o site da [Kore](#);
2. Efetuar o login;



Atenção: Neste passo, você deverá ter um contrato com a empresa prestadora e seu cadastro efetuado pela mesma. No momento, para uso didático, a conta será fornecida pelo professor. Outro importante detalhe é a sub-banda: na Kore devemos usar a sub-banda 0. Esta configuração é feita no arquivo *Commissioning.h* em [SUBBAND](#).

3. Uma vez feito o acesso, teremos a seguinte tela:



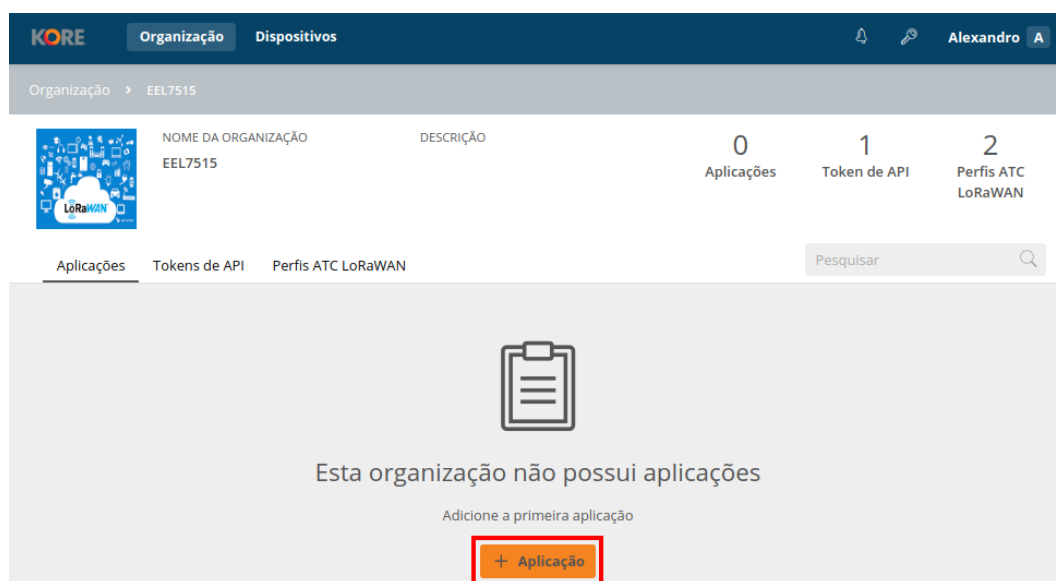
Nesta, estão listadas todas as aplicações e dispositivos criados, além das últimas operações realizadas.

4. Selecione no menu superior **Organização**, depois na lista apresentada, selecione **EEL7515**



NOME DA ORGANIZAÇÃO	DESCRIÇÃO	TOKENS DE APLICAÇÕES API	PERFIS ATC LORAWAN
My Organization		1	0
EEL7515		0	1

5. Criando uma aplicação. Pressione o botão **+ Aplicação**.



Organização > EEL7515

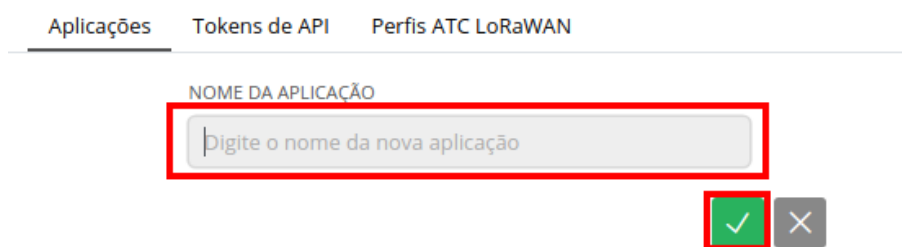
0 Aplicações 1 Token de API 2 Perfis ATC LoRaWAN

Esta organização não possui aplicações

Adicione a primeira aplicação

+ Aplicação

6. Dê uma nome a sua aplicação (por ex o nome da sua equipe) e inclua ela no grupo 'Principal'. Pressione o botão verde indicado abaixo.



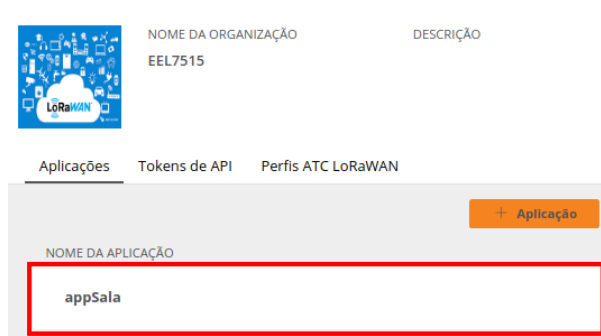
Aplicações Tokens de API Perfis ATC LoRaWAN

NOME DA APLICAÇÃO

digite o nome da nova aplicação

☒ ☐

Sua aplicação será mostrada na lista.



NOME DA ORGANIZAÇÃO DESCRIÇÃO

EEL7515

Aplicações Tokens de API Perfis ATC LoRaWAN

+ Aplicação

NOME DA APLICAÇÃO

appSala

7. Agora devemos cadastrar o dispositivo. Para isso selecione **Dispositivos**, no menu superior, selecione a aplicação desejada e então pressione o botão **+ Dispositivo**.



8. Selecione a tecnologia LoRaWAN.



9. Preencha o formulário com as informações do seu dispositivo. Caso necessário, o site poderá informar estas chaves para que configuremos nosso dispositivo posteriormente, como feito na TTN.

Tecnologia - LoRaWAN

Modelo - Dispositivo genérico

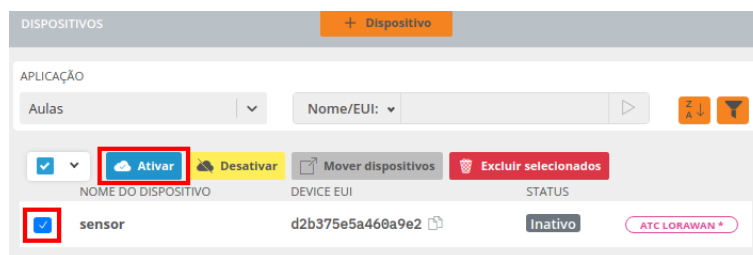
Conectividade - Conexão própria

Configuração e revisão

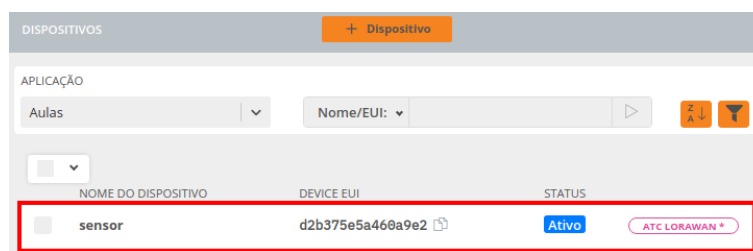
Só mais uma etapa! Este formulário exige algumas informações necessárias para configurar o seu dispositivo, preencha-o para finalizar o cadastro.

NOME DO DISPOSITIVO *	RÓTULO ÚNICO *	Resumo do dispositivo TECNOLOGIA LoRaWAN MODELO Dispositivo genérico CONECTIVIDADE Conexão própria
sensor	sensor	
PERFIL		
Não usar perfil	+ Perfil	
ATIVACÃO DE SEGURANÇA	DEVICE EUI *	
OTAA	d2 b3 75 e5 a4 60 a9 e2	
✓ NS	APPLICATION EUI *	
APP	f3 92 e7 d1 5c 59 c1 9a	
CLASSE DE DISPOSITIVO	DEVICE ADDRESS *	
✓ A	b3 59 7c b7	
C	NETWORK SESSION ENCRYPTION KEY *	
TAMANHO DO CONTADOR	66 c7 94 1d fc 00 24 99 29 53 f1 a2 6	
2	APPLICATION SESSION KEY *	
4	3f f7 6e c8 90 93 0a 72 46 59 f4 e5 f6	Gerar valor aleatório
BANDA *		
AU915-928A	MODOS ADR *	
ON		

10. Após a criação do dispositivo devemos então ativá-lo, para utilização imediata, logo selecionemos o sensor criado e pressionamos **Ativar**.



11. Selecionamos novamente o sensor e a tela de apresentação de dados recebidos irá aparecer.



12. Nesta tela os dados, em tempo real, irão ser apresentados.

KORE Organização Dispositivos alexteste (EEL7515 > Sala) ☒ CONECTADO

Este dispositivo não tem uma localização, clique aqui para defini-la

RÓTULO ÚNICO alexteste

ÚLTIMA ATIVIDADE 28/06/2020 19:14

TIPO DE DISPOSITIVO ATC LORAWAN *

DEVICE EUI f9805b27f30a043c

QUALIDADE DO SINAL (SNR/RSSI) -14dB/-116dBm

APPLICATION EUI 998769f72f0743ab

DEVICE ADDRESS be09f81c

APPLICATION SESSION KEY de768fc7868cc3712a9d1996966b97e1

NETWORK SESSION ENCRYPTION KEY d337755c280cb235865e478cc58c79c0

CONEXÕES DO DISPOSITIVO

Uplink ☒ ON

Downlink ☒ ON

Dados em tempo real

25 50 100 Pausar

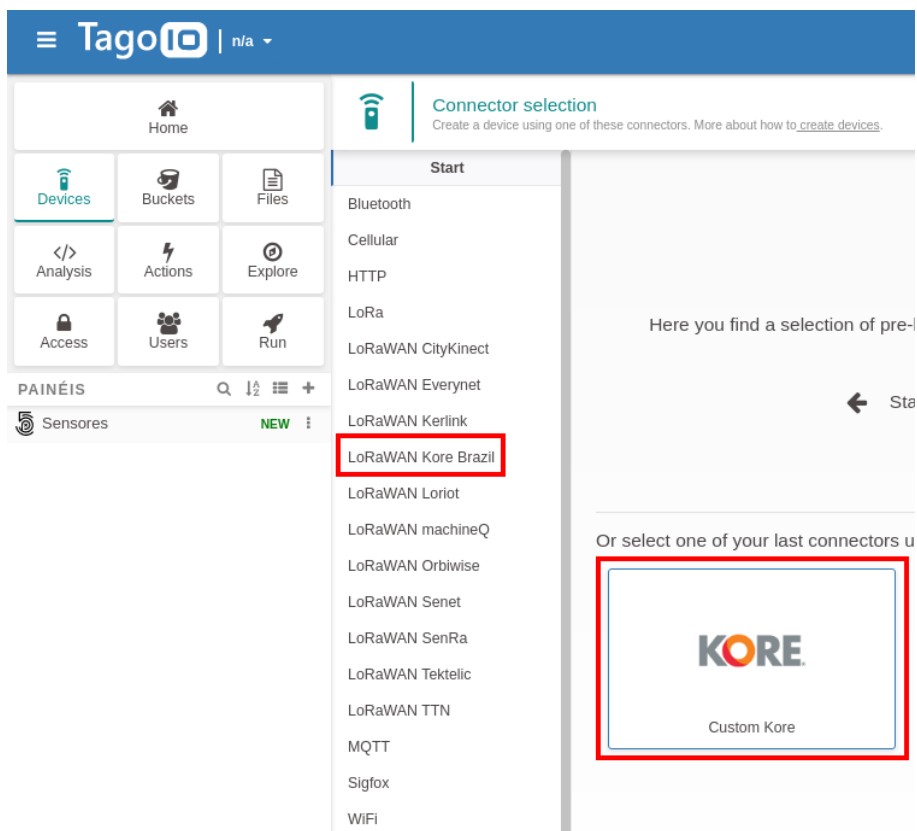
JOIN_REQUEST UPLINK DOWNLINK DOWNLINK_REQUEST ERROR WARNING INFO LOCATION

TEMPO REAL

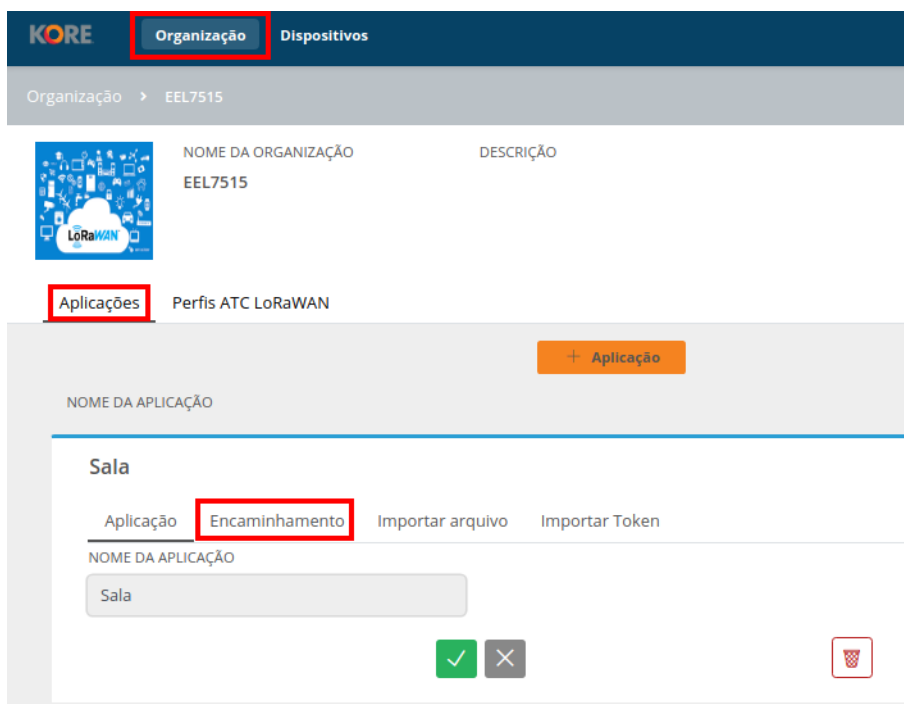
location	28/06/2020 - 19:14:43	lat -27.58861 lng -48.59025
downlink	28/06/2020 - 19:14:42	MAC commands: 5
Info	28/06/2020 - 19:14:42	Downlink message scheduled to send
Info	28/06/2020 - 19:14:42	Downlink request doesn't send to Application Server. All message's payload is used for mac-commands.
uplink	28/06/2020 - 19:14:42	Freq: 916.6MHz Signal quality: -14dB/-116dB
Info	28/06/2020 - 19:14:42	Data message received and processed
location	28/06/2020 - 19:13:43	lat -27.58861 lng -48.59025
downlink	28/06/2020 - 19:13:42	MAC commands: 5
Info	28/06/2020 - 19:13:42	Downlink message scheduled to send

9.1 Conectando a Kore à TagoIO

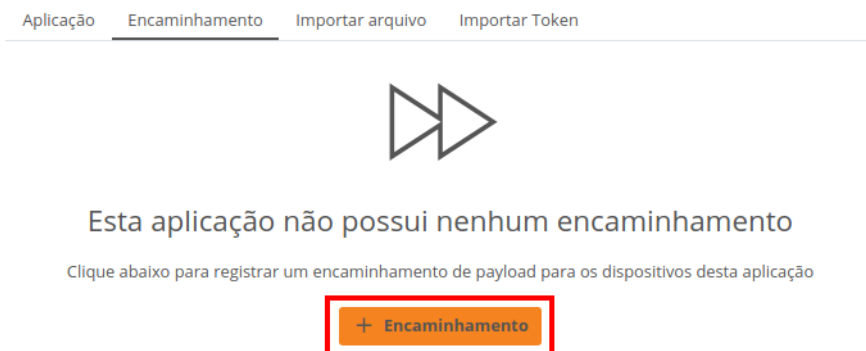
Todo o processo utilizado para a TTN, no momento da criação do dispositivo na TagoIO, é idêntico. Porém, basta selecionarmos a conexão da Kore (Custom Kore) ao invés da TTN.



No site da Kore devemos configurar o encaminhamento, de forma similar ao utilizado na TTN, respeitando as convenções de cada plataforma. Assim, pressionamos o botão **Organização**, selecionamos a aplicação e depois pressionamos em **Encaminhamento**



Caso não tenha nenhum encaminhamento definido, a tela de adição será apresentada e então devemos pressionar o botão + **Encaminhamento**.



Preenchemos o formulário conforme indicado.

A screenshot of the 'Encaminhamento' form in the application. The form has several sections. The first section has two fields: 'NOME DO ENCAMINHAMENTO*' with the value 'Enc Test' and 'SELECIONE O TIPO DO PROTOCOLO' with a dropdown menu showing 'HTTP'. The second section is 'TIPO DE MENSAGEM' with radio buttons for 'JOIN_REQUEST', 'UPLINK', 'DOWNLINK', 'DOWNLINK_REQUEST', 'ERROR', 'WARNING', 'INFO', and 'LOCATION'. The third section is 'OPÇÕES' with radio buttons for 'DUPLICATE', 'LORA', and 'RADIO'. The fourth section has two fields: 'URL/IP DE DESTINO*' with the value 'https://korebrazil.middleware.tago.io/uplink' and 'AUTHORIZATION HEADER' with the value 'at6f6eff80a28c442e82846d498e36a991'. At the bottom, there are two buttons: a green checkmark button and a grey 'X' button. The fields for 'Enc Test', 'HTTP', the URL, and the authorization header are all highlighted with red rectangular boxes.

Tipo de mensagem: Uplink

URL/IP de destino: <https://korebrazil.middleware.tago.io/uplink>

Lembrando que a **AUTHORIZATION HEADER** deve ser obtida no site da TagoIO.

A screenshot of the 'Service Authorization' form. The form has a header with a key icon and the text 'Service Authorization'. Below the header, there is a table with four columns: 'Name', 'Additional Parameter (Optional)', 'Created at', and 'Authorization'. The 'Name' column has a text input field with the value 'authorization'. The 'Additional Parameter (Optional)' column has a text input field with the value 'a few seconds ago'. The 'Authorization' column has a text input field with the value 'Copy'. The 'Name' and 'Additional Parameter (Optional)' fields are highlighted with red rectangular boxes.

Com isso o dashboard pode ser construído, conforme discutido anteriormente, e a transferência dos dados da Kore para TagoIO está garantida.

10 Personalização

No programa LoraMac fornecido é possível ligar ou desligar um dos leds da placa e, ao mesmo tempo, fazer uma leitura da tensão no pino PA_0. Informações como temperatura, pressão e umidade também são transmitidas periodicamente.

Com o intuito de personalizar a aplicação, podemos utilizar as portas digitais, analógicas e de comunicação para implementar novas funcionalidades, respeitando a estrutura básica do código fonte. Para tanto, precisamos editar o arquivo **main.c** encontrado na pasta **src/apps/LoRaMac/classA/B-L072Z-LRWAN1**.

No arquivo **Commissionig.h** encontramos algumas configurações para o rádio e para conexão com o gateway, além disso, por exemplo, o intervalo de transmissão periódica de dados:

[linha 21] `#define APP_TX_DUTYCYCLE 60000`

que informa ao módulo o ciclo de transmissão em milissegundos. No arquivo *main.c*, encontramos o direcionamento aos arquivos de cabeçalho, necessários para a utilização das funções previamente criadas ou em desenvolvimento.

```
26  #include <stdio.h>
27  #include <stdlib.h>
28  #include <string.h>
29  #include "tools.h"
30  #include "utilities.h"
31  #include "board.h"
32  #include "gpio.h"
33  #include "adc.h"
34  #include "LoRaMac.h"
35  #include "Commissioning.h"
36  #include "NvmCtxMgmt.h"
37  #include "LoRaMacTest.h"
38  #include "delay.h"
```

Figura 18: Arquivos de cabeçalho.

Objetos são instanciados, como por exemplo o Led4, conectado à porta PB_7 da placa STM32L072CZY6TR, indicado na Figura 19.

```
234  /*!
235  * LED GPIO pins objects
236  */
237  extern Gpio_t Led1; // Tx
238  extern Gpio_t Led3; // Rx
239  extern Gpio_t Led4; // App
240
241  /*!
242  * Entrada analógica
243  */
244  Adc_t AnalogIn_PA_0;
```

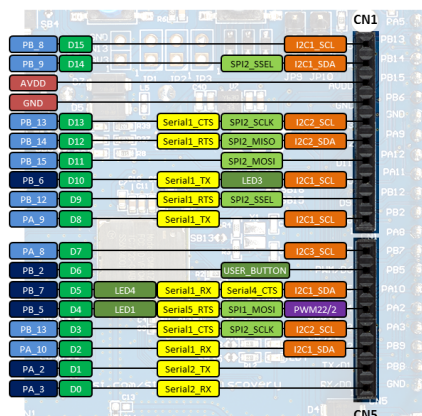


Figura 19: Pinagem dos conectores CN1 e CN5.

Ainda na Figura 19, podemos observar os pinos PB_8 e PB_9, utilizados para comunicação i2c entre a placa de desenvolvimento e o kit de sensores. O pino PA_0 (Figura 20) está configurado como entrada analógica, neste um cuidado deve ser tomado, **os limites de tensão de operação do microcontrolador são de 0 V até 3.3 V**, ocasionando dano permanente se estes não forem respeitados. Caso outros níveis de tensão necessitem ser aplicados, um dispositivo de adequação, *interface*, deverá ser desenvolvido e através de software convertido para o real valor medido.

Obs.: O manual deve ser consultado, todos pinos possuem restrições de uso, tanto em tensão quando em corrente, bem como uso interno da placa. Um exemplo de uso interno são os pinos de comunicação serial.

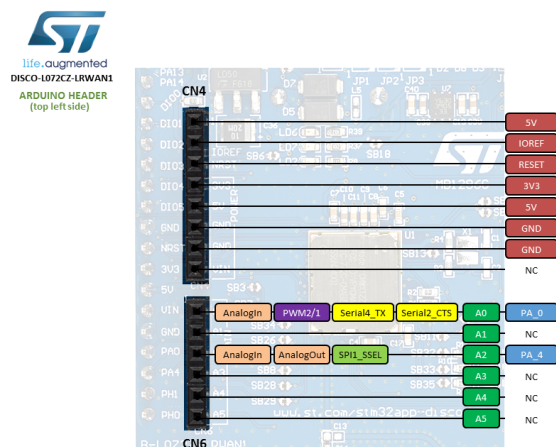


Figura 20: Pinagem dos conectores CN4 e CN6.

A função `PrepareTxFrame()` é responsável por montar o pacote de dados para o envio. Nela podemos implementar, por exemplo, a leitura de um sensor, lembrando que serão necessárias as bibliotecas responsáveis pelo controle do sensor ou dispositivo anexado, escritas em C. Dentro do arquivo `tools.c` temos uma biblioteca para os sensores HTS221 (temperatura e umidade) e LPS22HB (pressão atmosférica) do Kit IKS01A2. Na Figura 21 temos o exemplo utilizado para enviar o pacote de dados.

```

367 //***** Preparando o pacote para o envio *****/
368 float temperature=HTS221_Temperature(CELSIUS);
369 float humidity=HTS221_Humidity();
370 float pressure=LPS22HB_Pressure();
371 float volts= (float)(AdcReadChannel(&AnalogIn_PA_0, 1)/100.0);
372
373 //***** Determina o tamanho do pacote *****/
374 AppDataSizeBackup = AppDataSize = 9;
375
376 //***** Configura o pacote *****/
377 AppDataBuffer[0] = (int)temperature;
378 AppDataBuffer[1] = (int)((temperature-AppDataBuffer[0])*10);
379
380 AppDataBuffer[2] = (int)humidity;
381 AppDataBuffer[3] = (int)((humidity-AppDataBuffer[2])*10);
382
383 AppDataBuffer[4] = (int)(pressure/100);
384 AppDataBuffer[5] = (int)(pressure-(AppDataBuffer[4]*100));
385 AppDataBuffer[6] = (int)((pressure-(int)(pressure))*10);
386
387 AppDataBuffer[7] = (int)volts;
388 AppDataBuffer[8] = (int)((volts-AppDataBuffer[7])*10);
389 //*****

```

Figura 21: Código de montagem dos pacotes para o envio.

A montagem deste pacote de dados é feita da seguinte maneira:

- Lemos os valores dos sensores para as variáveis correspondentes (Linhas 368-371);
- Indicamos o tamanho do pacote em bytes (Linha 374);
- Dividimos em parte inteira e fracionária (apena uma casa decimal) e armazenamos sequencialmente os valores no *buffer* (AppDataBuffer - Linhas 377-388).

Na Figura 22 vemos uma maneira de apresentar os dados no terminal serial, embora este deva ser utilizado apenas como método de depuração, em um produto final estes códigos ocupam espaço em memória e devem ser evitados.

```

578 //*****
579 //Leitura da temperatura em graus CELSIUS ou FAHRENHEIT
580 float rasc=HTS221_Temperature(CELSIUS);
581 printf("\nTemperatura = ");
582 printDouble(rasc,1);
583 //*****
584 //Leitura da umidade relativa (%)
585 rasc=HTS221_Humidity();
586 printf("    Umidade = ");
587 printDouble(rasc,1);
588 printf("%%%");
589 //*****
590 //Leitura de pressão atmosférica em hPa
591 rasc=LPS22HB_Pressure();
592 printf("    Pressao = ");
593 printDouble(rasc,1);
594 printf(" hPa");
595 //*****
596 //Leitura da tensão em Volts
597 uint16_t volts=AdcReadChannel(&AnalogIn_PA_0, 1);
598 printf("    Tensao = ");
599 printDouble((float)(volts/100),2);
600 printf(" V\r\n");
601 //*****

```

Figura 22: Código de montagem da apresentação no terminal serial.

A função `McpsIndication()`, além de outras atividades, é também responsável pelo recebimento dos dados e tratamento deles, nesta parte podemos acionar dispositivos remotamente ou requisitar uma leitura adicional, por exemplo. Na Figura 23 verificamos se recebemos apenas um **byte** e se o mesmo vale $0x01_{16} = 1_{10}$, caso afirmativo o Led4 deverá acender após a confirmação do recebimento da informação. Para (acender) apagar o Led4 podemos enviar

pelo sistema da TTN o valor $(0x01_{16})$ $0x00_{16}$ para o kit, como ilustrado na Figura 24. O Led4 (vermelho - Figura 25) está posicionado, na placa de desenvolvimento, abaixo do módulo de sensores, se este último estiver conectado.

```

712 case 2:
713     if( mcpsIndication->BufferSize == 1 )
714     {
715         AppLedStateOn = mcpsIndication->Buffer[0] & 0x01;
716         GpioWrite( &Led4, ( ( AppLedStateOn & 0x01 ) != 0 ) ? 1 : 0 );
717     }
718     break;

```

Figura 23: Entrada de dados.

Figura 24: Envio do comando apagar Led4, no site da TTN - Aba Devices.

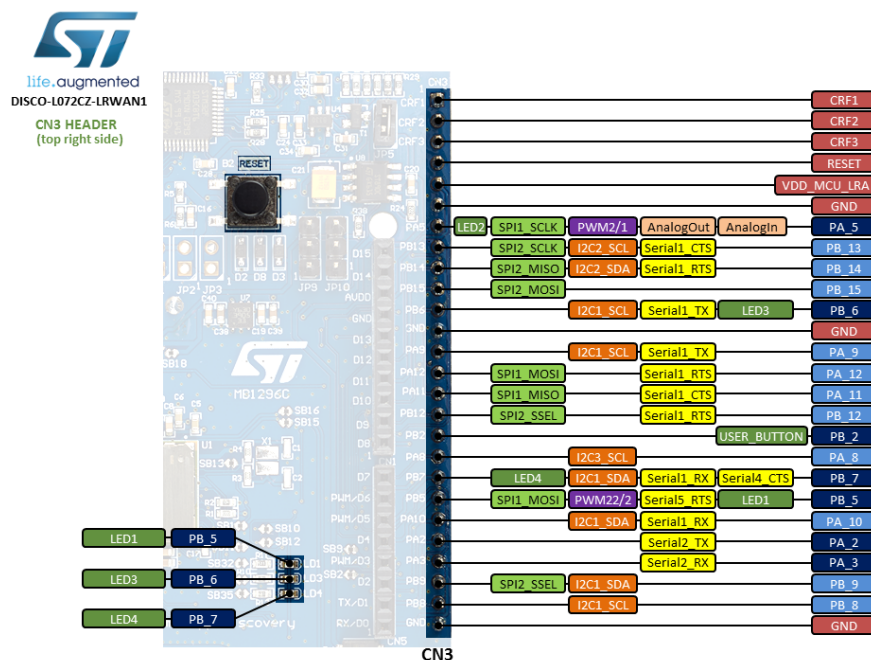
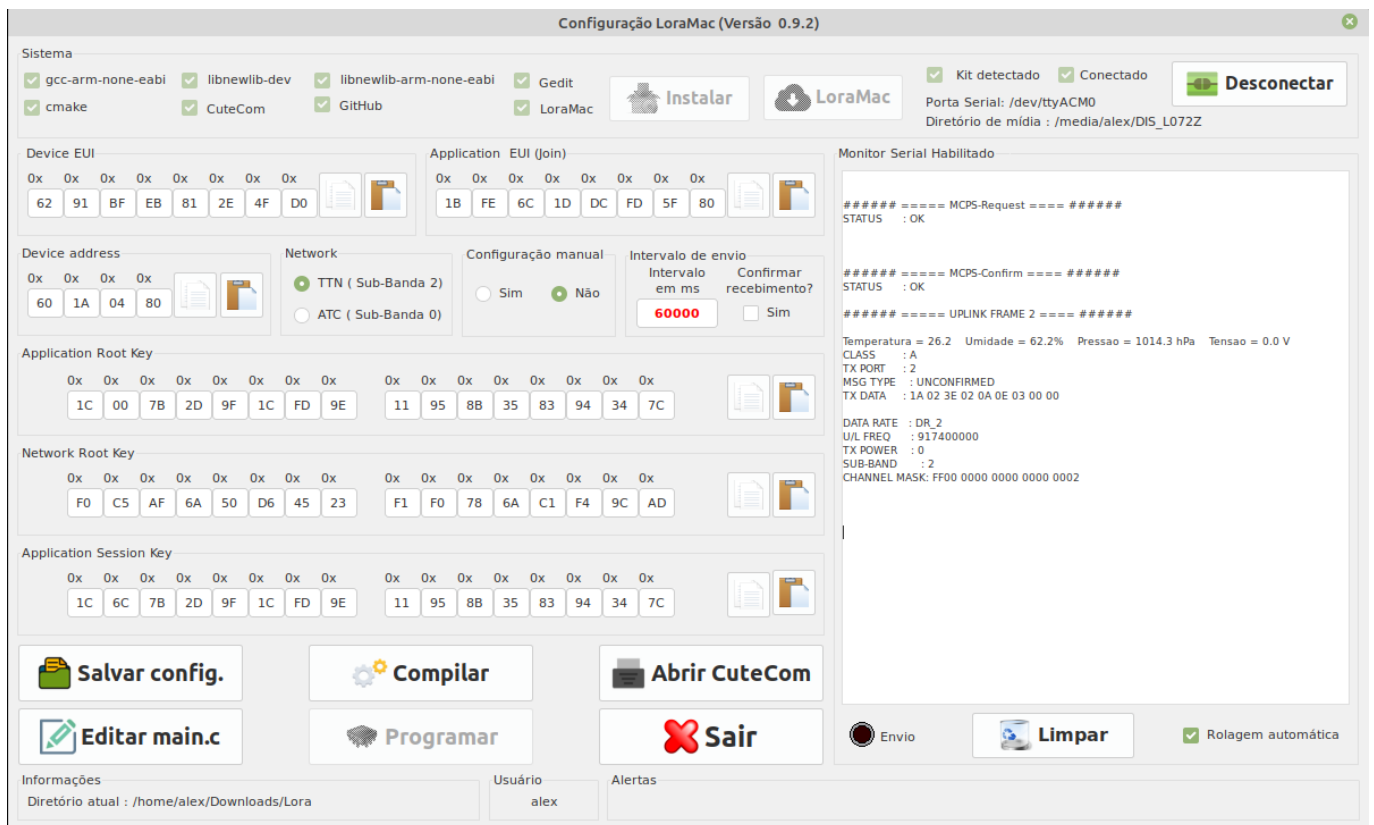


Figura 25: Posição dos leds na placa de desenvolvimento.

Na pasta **src/system** encontram-se todos os métodos necessários para utilizar o microcontrolador do kit. Drivers para utilização da Gpio, i2c, SPI e ADC são alguns exemplos, estes, por sua vez, mais utilizados na adaptação de sensores e atuadores externos.

11 Aplicativo de Configuração LoraMac



Este aplicativo foi desenvolvido com o intuito de verificar e automatizar a instalação de bibliotecas, configurar os parâmetros da rede, compilar e programar o kit, além de possuir uma maneira de monitorar localmente as informações transmitidas pelo dispositivo.

O programa pode ser baixado deste link do [Dropbox](#). Depois, no terminal, dentro do diretório onde foi baixado o programa, ceder permissões de execução:

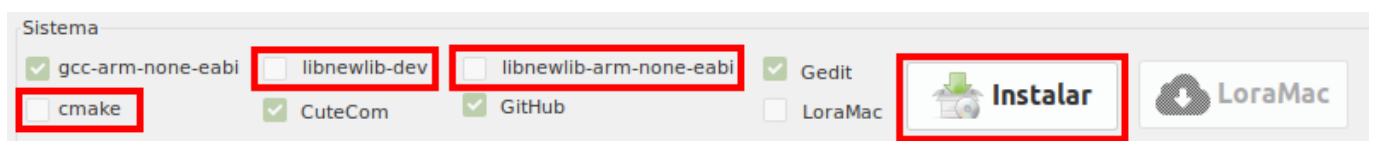
```
sudo chmod +x LoraMac
```

Em seguida o comando abaixo irá rodar o programa:

```
./LoraMac
```

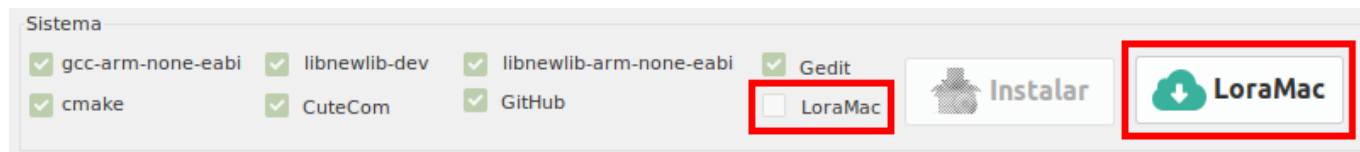
O mesmo pode ser feito no gerenciador de arquivos da distribuição em uso, adicionando as permissões de execução e leitura/escrita para o usuário.

Quando executado, o programa irá verificar se todas as bibliotecas necessárias à compilação estão devidamente instaladas. Caso haja falta de alguma biblioteca, uma mensagem de alerta será emitida, na barra **Alertas** no rodapé do programa, e o botão de **Instalar**, na aba **Sistema**, será habilitado.

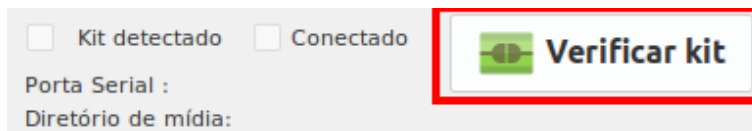


Pressione o botão **Instalar** e será solicitado sua senha de administrador, efetuando as devidas instalações.

Agora, note que o software LoraMac não está presente junto ao programa de configuração. Desta vez, pressionamos o botão **LoraMac**, e a dependência será resolvida.



Conectando o kit ao computador, e aguardando o reconhecimento pela distribuição Linux, podemos pressionar o botão **Verificar kit**. Estando tudo em conformidade, serão preenchidos os dados com a porta serial e o diretório da mídia.



Para conectar ao kit e observarmos as informações oriundas no mesmo, pressionamos o botão **Conectar**. Desta vez uma verificação, para garantir que o usuário tenha acesso às portas seriais, será executada. Caso o usuário não tenha acesso, será pedida a senha de administrador e criada a autorização para o usuário, também indicado no rodapé do programa na aba **Usuário**.










Atenção: Para que esta autorização torne-se visível pelo sistema, o Linux deverá ser reiniciado. Caso este reinício não seja realizado, não será garantido o acesso às portas seriais.

Realizando todos os procedimentos corretamente, os dados começam a fluir pelo Monitor Serial quando o dispositivo transmitir.





Observe que ao deixarmos o ponteiro do mouse sobre algum botão, uma pequena descrição da função, que ele executará, será apresentada.

Agora, com as informações das chaves da rede para o dispositivo, podemos iniciar a inserção manual dos dados ou simplesmente utilizar as teclas de copiar e colar do aplicativo. Por exemplo, na TTN, pressionando o ícone indicado, será copiado o Device EUI para a área de transferência.


Device EUI	<>	↕	31 32 37 37 58 37 88 05	
Application EUI	<>	↕	AB CD FF 00 11 22 55 AA	
Device Address	<>	↕	26 06 23 03	
Network Session Key	<>	↕		
App Session Key	<>	↕		

Por sua vez, basta, no aplicativo, pressionar o botão colar.


Device EUI

0x	0x	0x	0x	0x	0x	0x	0x		
31	32	37	37	58	37	88	05		


O mesmo poderá ser feito com os demais campos. Lembre-se que qualquer alteração feita deverá ser salva pressionando o botão **Salvar config.**. A fim de que surja efeito, devemos compilar o código e programar a placa, para tanto, nesta sequência, devemos pressionar os botões **Compilar** e **Programar**.




Salvar config.




Compilar




Abrir CuteCom



Editar main.c



Programar



Sair

Outras funcionalidades foram adicionadas, como o botão **Editar main.c**. Este abre um editor de texto com os principais arquivos de configuração. Lembre-se que, utilizando de forma inadequada, poderá corromper os arquivos tornando-se impossível a compilação. Ao final da compilação, serão apresentados no monitor serial os passos executados, devendo chegar á 100%, caso tudo ocorra de forma satisfatória. No final do relatório encontraremos:

```
[100%] Built target LoRaMac-classA.hex
Scanning dependencies of target LoRaMac-classA.bin
[100%] Built target LoRaMac-classA.bin
```

Referências

- [1] SORNIN, Nicolas et al. Lorawan specification. LoRa alliance, 2015.
- [2] <<https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html>>. Acesso em: 04 mar. 2020.
- [3] <<https://ubuntu.com>>. Acesso em: 04 mar. 2020.
- [4] <<https://github.com/Lora-net/LoRaMac-node>>. Acesso em: 04 mar. 2020.
- [5] <<https://www.thethingsnetwork.org/>>. Acesso em: 04 mar. 2020.
- [6] <<https://www.tagoio.com/>>. Acesso em: 25 mar. 2020.
- [7] <<https://kore.proiot.com.br/>>. Acesso em: 25 mar. 2020.