Alexandro Vanderley dos Santos

Integração entre Dispositivos LoRa e Servidores de Aplicação Utilizando o Protocolo LoRaWAN

Florianópolis
2021

Alexandro Vanderley dos Santos

INTEGRAÇÃO ENTRE DISPOSITIVOS LORA E SERVIDORES DE APLICAÇÃO UTILIZANDO O PROTOCOLO LORAWAN

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica.
Orientador: Richard Demo Souza

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Santos, Alexandro Vanderley dos
Integração entre dispositivos LoRa e servidores de aplicação utilizando o protocolo LoRaWAN / Alexandro Vanderley dos Santos ; orientador, Richard Demo Souza , 2021.
71 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2021.

Inclui referências.

1. Engenharia Eletrônica. 2. Engenharia Eletrônica.. 3. Comunicação sem fio.. 4. LoRaWAN. 5. Servidores de Aplicação. I. , Richard Demo Souza. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Alexandro Vanderley dos Santos

Integração entre dispositivos LoRa e servidores de aplicação utilizando o protocolo LoRaWAN

Este Trabalho foi julgado adequado para obtenção do Título de Bacharel em Engenharia Eletrônica em sua forma final pela sua Banca Examinadora.

Florianópolis, 14 de maio de 2021

Documento assinado digitalmente
Fernando Rangel de Sousa
Data: 18/05/2021 09:24:45-0300
CPF: 884.649.114-91
Verifique as assinaturas em https://v.ufsc.br

Prof. Fernando Rangel de Sousa, Dr.
Coordenador do Curso

Banca examinadora:

Documento assinado digitalmente
Richard Demo Souza
Data: 17/05/2021 21:34:20-0300
CPF: 004.267.379-89
Verifique as assinaturas em https://v.ufsc.br

Prof. Richard Demo Souza, Dr.
Orientador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Leonardo Silva Resende
Data: 18/05/2021 10:03:08-0300
CPF: 524.735.206-82
Verifique as assinaturas em https://v.ufsc.br

Prof. Leonardo Silva Resende, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Willian Henrique
Data: 18/05/2021 08:03:09-0300
CPF: 074.931.159-23
Verifique as assinaturas em https://v.ufsc.br

Willian Henrique, M.Sc.
SENAI/SC

“Este trabalho é dedicado à minha família."

AGRADECIMENTOS

Agradeço, inicialmente, aos meus pais por todo apoio durante estes anos de faculdade, sempre acreditando que a vida nos leva ao futuro que sempre almejamos. À minha namorada, Sanny Santos Serafim, por estar ao meu lado em todos os momentos, compartilhando os desafios e vitórias durante a vida escolar e pessoal. E ao professor Richard Demo Souza, pela orientação durante o trabalho, uma pessoa pela qual tenho muita admiração, respeito e considero ser um profissional diferenciado no âmbito da educação, com suas práticas, sempre procurando iluminar o caminho e direcionando todos a realizarem escolhas profissionais mais concisas e seguras.

Agradeço à sociedade brasileira, pela oportunidade de frequentar uma instituição pública e valor inestimável. Aos demais professores e colegas de estudos, um profundo desejo de sucesso e realizações.

“Persistence is the path to success."
(Charles Chaplin)

RESUMO

O uso massivo da IoT, *Internet of Things*, através de redes sem fio, traz consigo a necessidade de uma forma de monitorar e interagir com dispositivos que fazem o sensoriamento de diferentes grandezas e/ou realizam atividades de acionamento, permitindo a automação de uma série de operações e decisões em diferentes lugares e instantes de tempo. Dentre as diversas camadas envolvendo a comunicação entre dispositivos, servidores e usuários, o *Application Server*, ou servidor de aplicação, é o responsável por tornar legível, ao usuário, informações cruciais à tomada de decisão. Neste trabalho, descrevemos uma integração completa entre um dispositivo IoT, utilizando as tecnologias LoRa e LoRaWAN, com um servidor de aplicação. Todos os passos são devidamente detalhados, enquanto o desenvolvimento completo serve de apoio à uma disciplina optativa dos cursos de Engenharia Elétrica e Eletrônica da UFSC.

Palavras-chave: Comunicações sem Fio. LoRA. LoRaWAN. TTN. Kore. IoT.

ABSTRACT

The massive use of IoT, Internet of Things, through wireless networks, brings with it the need for a way to monitor and interact with devices that sense different phenomena or perform triggering activities, allowing the automation of a series of operations and decisions in different places and moments of time. Among the several layers involving communication between devices, servers and users, the application server is responsible for making crucial information readable to the user. In this work, we describea a complete integration of an IoT device, using LoRa and LoRaWAN technologies, with an application server. All steps are duly detailed, while the complete development supports an elective course in the Electrical and Electronics Engineering undergraduation program at UFSC.

Keywords: wireless communication. lora. lorawan. ttn. kore. iot.

LISTA DE FIGURAS

Figura 1 – Topologia de uma rede LoRaWAN convencional. 21
Figura 2 – Kit da STMicroelectronics 22
Figura 3 – Módulo sensor IKS01A2. 22
Figura 4 – Gateway IGT 200 Indoor da Khomp. 23
Figura 5 – Dashboard criada na TagoIO. 24
Figura 6 – Acoplamento do módulo IKS01A02 no Discovery kit. . . 27
Figura 7 – Menu de acesso ao console da TTN. 29
Figura 8 – Botão adicionar aplicação. 29
Figura 9 – Formulário de cadastro da aplicação. 30
Figura 10 – Menu de acesso aos *end nodes*. 30
Figura 11 – Formulário de registro de *end nodes*. 31
Figura 12 – Menu de acesso às configurações. 31
Figura 13 – Informações para configurar o LoRaMAC 31
Figura 14 – Tela de identificação de usuário. 32
Figura 15 – Tela de seleção de organização. 33
Figura 16 – Campo de inserção do nome da aplicação. 33
Figura 17 – Cadastro de *end nodes*. 33
Figura 18 – Selecionar o tipo de rede. 34
Figura 19 – Ativação do dispositivo. 34
Figura 20 – Tela de visualização do dados. 34
Figura 21 – Tela do Cutecom. 38
Figura 22 – Menu de acesso aos dados. 39
Figura 23 – Dados transmitidos e recebidos 39
Figura 24 – Decodificação do pacote de dados. 40
Figura 25 – Expandindo os dados. 41
Figura 26 – Tela principal da TagoIO. 43
Figura 27 – Menu de acesso a registro de *end nodes*. 44
Figura 28 – Tela de conexão TTN/TagoIO. 44
Figura 29 – Direcionando ao dispositivo. 45
Figura 30 – Acesso ao Gerador de chave de autorização. 45
Figura 31 – Menu de acesso à configuração de integração da TTN. . . 46

Figura 32 – Integrando TagoIO à TTN. 46
Figura 33 – Configurando a integração (Site TTN). 47
Figura 34 – Tela de configuração da Kore. 48
Figura 35 – Definindo a forma de encaminhamento. 48
Figura 36 – Simulando o envio de dados à TTN. 49
Figura 37 – Tela configuração do desempacotador de dados. 49
Figura 38 – Ativar a simulação. 52
Figura 39 – Configurando o pacote no emulador. 53
Figura 40 – Apresentação dos dados recebidos já decodificados. . . . 53
Figura 41 – Acesso à configuração do armazenamento de dados. . . . 54
Figura 42 – Seleção dos dados a serem salvos. 54
Figura 43 – Botão para adicionar uma *dashboard*. 55
Figura 44 – Inserção do nome para criação da *dashboard*. 55
Figura 45 – Adicionar *widgets*. 56
Figura 46 – Objetos ou *widgets*. 56
Figura 47 – Configurando um *widgets*. 57
Figura 48 – Botão de inserção de ações. 57
Figura 49 – Configurando uma ação. 58
Figura 50 – Definido tipos de mensagens. 59
Figura 51 – Publicando o *dashboard*. 60
Figura 52 – Pinagem dos conectores CN1 e CN5. 68
Figura 53 – Pinagem dos conectores CN4 e CN6. 68
Figura 54 – Código de montagem dos pacotes para o envio. 69
Figura 55 – Código de montagem da apresentação no terminal serial. 70
Figura 56 – Entrada de dados. 70
Figura 57 – Envio do comando apagar Led4, no site da TTN - Aba Devices. 70
Figura 58 – Posição dos leds na placa de desenvolvimento. 71

SUMÁRIO

1	**INTRODUÇÃO**	**19**
1.1	**Trabalho Proposto**	**20**
2	**REDE DE SENSORES LORAWAN**	**21**
2.1	**End Nodes**	**21**
2.2	**Gateway**	**22**
2.3	**Network Server**	**23**
2.4	**Application Server**	**24**
3	**PREPARANDO O AMBIENTE PARA O DESENVOLVIMENTO**	**25**
3.1	**Preparação do Sistema Operacional**	**25**
3.2	**Preparando o Discovery Kit da STMicroelectronics**	**26**
4	**CRIANDO UMA APLICAÇÃO NA TTN E NA KORE**	**29**
4.1	**TTN**	**29**
4.2	**Kore**	**32**
5	**CONFIGURAR E COMPILAR O CÓDIGO FONTE**	**35**
5.1	**Formatando o Payload**	**36**
5.2	**Recuperando os Dados**	**37**
5.3	**Compilar e verificar o projeto**	**38**
5.4	**Verificação Local**	**38**
5.5	**Verificação Remota**	**39**
6	**INTERFACE GRÁFICA - DASHBOARD**	**43**
6.1	**Criando uma Conta na TagoIO**	**43**
6.2	**Conectando a TagoIO com a TTN**	**44**
6.3	**Conectando a Kore à TagoIO**	**46**
6.4	**Simulando o Envio de Dados**	**48**
6.5	**Interpretando os Dados**	**49**
6.6	**Simulando o Recebimento de Dados**	**52**

6.7 **Armazenamento de Dados** **54**
6.8 **Dashboard - Apresentação dos Dados** **55**
6.9 **Alertas e Mensagens** **57**
6.10 **Publicando o Dashboard** **59**

7 **CONCLUSÃO** **61**

REFERÊNCIAS **63**

APÊNDICES **65**

APÊNDICE A – PERSONALIZAÇÃO DO CÓDIGO FONTE LORAMAC **67**

1 INTRODUÇÃO

Quando tratamos de comunicação entre dispositivos, ou entre dispositivo e interface do usuário, o termo Internet das Coisas, comumente chamada pela sigla IoT (do inglês *Internet of Things* [1]), aparece como uma solução imediata para este tipo de demanda. Soluções baseadas em baixo custo, financeiro e operacional, ou que promovem a redução deste, são cada vez mais requisitadas. A abrangência desse mercado é global.

Na indústria, o emprego de sensores, em diversos pontos do parque fabril, é bastante comum. Este é um setor onde o monitoramento constante é de suma importância para a gestão do conceito produtividade versus qualidade. Controle de temperatura, pressão, umidade, vazão, consumo energético ou de suprimentos, dentre muitos outros, antes verificadas manualmente, agora podem ser executados por dispositivos passivos, ou com algum tipo de reatividade, reduzindo o prazo de contabilidade e custo de pessoal. Tal processo de automação torna mais preciso o monitoramento e consequentemente a tomada de decisão.

Nas cidades, aplicações para os sensores sem fios são de alta empregabilidade. Aplicações como controle de tráfego, estacionamentos, temperatura, incidência de raios UV, qualidade do ar, iluminação pública, nível dos rios e seus afluentes e até mesmo no monitoramento de estruturas prediais ou encostas de morros, são amostras do grande potencial desse sensores.

Para viabilizar tais aplicações, são necessários dispositivos, os chamados *end nodes*, que possuem, na maioria das vezes, baixo custo de fabricação e implementação além de um consumo energético reduzido. Outro ponto relevante é a necessidade de transmissão de longo alcance, se comparado às tecnologias de redes sem fio de uso tipicamente residencial ou comercial. Agregando as características de baixo custo, baixo consumo e longo alcance, as redes do tipo *Low-Power Wide Area Networks* (LPWAN) [2] tornam-se muito atrativas.

Nesta linha de LPWANs, LoRa [3] [4] é uma tecnologia de camada física que permite comunicação a longas distâncias com pouco consumo de energia. LoRa é propriedade da empresa Semtech [5], e utiliza um método de

espalhamento espectral chamado de *Chirp Spread Spectrum* (CSS) [6] [7]. Já o protocolo LoRaWAN (*Long Range Wide Area Network*) [8], de código aberto e mantido pela LoRa Alliance [9], 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. LoRaWAN é responsável pelo gerenciamento, segurança, confirmação de mensagens, ativação de dispositivos, entre outros aspectos. A The Things Network (TTN) [10], por sua vez, é uma iniciativa com origem na Holanda, mas já espalhada por diversos países, que busca prover um conjunto de soluções de software (o servidor de rede) para a construção de uma rede LoRaWAN global, aberta, que permita o teste e a implantação de aplicações de IoT com baixo custo. Também existem soluções semelhantes à da TTN, porém de aspecto comercial, como aquela fornecida pela KoRe [11]. Finalmente, existem várias opções, desde de software livre até comerciais, para a implantação do chamado servidor de aplicação, onde o usuário, por fim, pode visualizar e interagir com os dados monitorados e mesmo com os dispositivos. Um exemplo deste tipo de solução é o TagoIO [12].

1.1 TRABALHO PROPOSTO

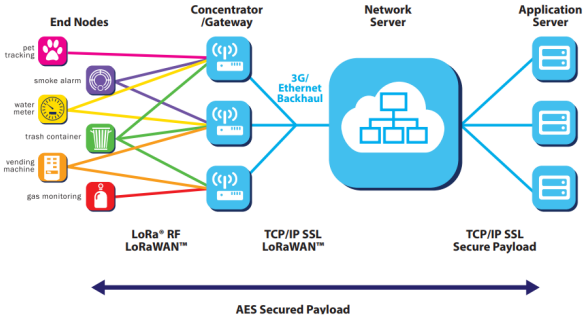
Este trabalho contém um compilado do desenvolvimento de uma aplicação para a observação dos dados capturados por sensores conectados à Internet através de rede LoRaWAN, utilizando o Discovery Kit for STM32L072 [13], desenvolvido pela STMicroelectronics [14] e plataformas como as da TTN, KoRe e TagoIO. O texto inclui os requisitos solicitados, descrição das funções e um detalhado esquema de construção desta integração.

Dentro do contexto do curso de Graduação em Engenharia Eletrônica, a disciplina EEL7515 - Tópico Avançado em Processamento de Sinais II [15], oferece atualmente uma introdução às tecnologias sem fio para IoT, com ênfase justamente em LoRa e LoRaWAN. O material apresentado neste trabalho tem como objetivo servir de apoio ao desenvolvimento de atividades práticas dentro da referida disciplina.

2 REDE DE SENSORES LORAWAN

A Figura 1 ilustra a arquitetura completa da solução proposta, contendo *end nodes* que monitoram o meio e se comunicam com o *gateway*, o qual encaminha as mensagens recebidas para um servidor de rede que, entre outras funções, envia os dados monitorados para um servidor de aplicação. Assim, a rede de sensores sem fio deve ser constituída por elementos com protocolos de comunicação compatíveis, formando um ecossistema capaz de medir, transmitir, organizar e disponibilizar os dados num formato compreensível e dinâmico, ficando atrativo ao usuário ou legível a um sistema gerenciador automatizado. A seguir, será brevemente informado como cada um destes blocos é implementado, sendo os detalhes apresentados nos próximos capítulos.

Figura 1 – Topologia de uma rede LoRaWAN convencional.



Fonte: https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf, 2021.

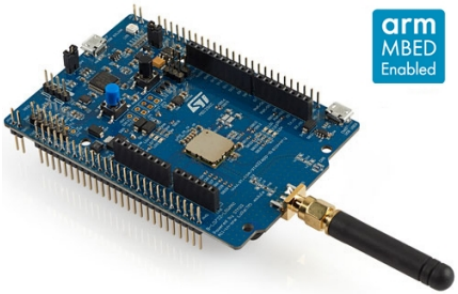
2.1 END NODES

End nodes são dispositivos que utilizam a tecnologia LoRa para se comunicar com o *gateway*, enviando as medidas realizadas por sensores ou informações pertinentes à aplicação desenvolvida.

O kit da STM utilizado neste trabalho, Figura 2, contém o rádio SX1276 [16], o qual possue todo o hardware para controle de frequência, modulação

e codificação dos dados, responsável pela comunicação LoRa. Componentes como microcontroladores e diversas interfaces fazem parte do kit [13], o qual também pode ser ligado a uma placa de extensão, como o módulo IKS01A2 [17], mostrado na Figura 3, o qual mede temperatura, pressão, umidade e aceleração, por exemplo.

Figura 2 – Kit da STMicroelectronics



Fonte: STMicroelectronics, 2021.

Figura 3 – Módulo sensor IKS01A2.

Fonte: STMicroelectronics, 2021.

2.2 GATEWAY

O *gateway* é responsável por criar uma conexão entre os dispositivos e o servidor de rede e pelo recebimento de dados dos diversos *end nodes*, além de interpretar, armazenar e repassar os dados ao servidor de rede. O

gateway está conectado ao Servidor de Rede via protocolo TCP/IP, seja por rede cabeada ou sem fio. O *gateway*, do ponto de vista dos *end nodes*, é apenas um encaminhador de mensagens para o Servidor de Rede.

Neste trabalho, nas aplicações com a rede TTN, foi utilizado o *gateway* da Khomp [18], ITG 200 Indoor, Figura 4. Este *gateway* possui, além das características de um *gateway* LoRaWAN, as seguintes funcionalidades [19]: i) duas portas RJ45 Fast Ethernet 10/100 Mbps; ii) display OLED com 4 botões; iii) antenas com ganho de 5 dBi; iv) módulo 3G de dados para até 2 SIM cards (opcional).

Figura 4 – Gateway IGT 200 Indoor da Khomp.

Fonte: https://www.khomp.com/pt/produto/itg-200/, 2021.

2.3 NETWORK SERVER

O *network server*, ou servidor de rede, valida as informações e a integridade dos pacotes recebidos, sendo também responsável pela construção dos pacotes de retorno aos *end nodes*, através do *gateway*, assim como pelo encaminhamento dos dados ao servidor de aplicação.

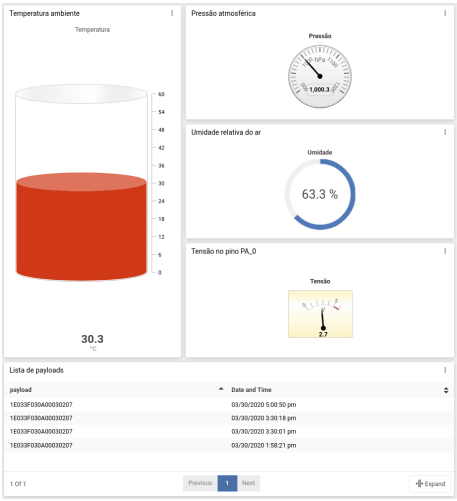
Para esta função, neste trabalho, foram escolhidos dois servidores, o da TTN [10], uma plataforma aberta, e o da Kore [11], uma plataforma comercial. Ambos os servidores escolhidos provém uma conexão com o servidor de aplicação que será usado neste trabalho.

2.4 APPLICATION SERVER

No *application server*, ou servidor de aplicação, são construídas as aplicações que vão interpretar e apresentar os dados ao usuário. Este servidor é capaz de formatar os dados que serão enviados ao dispositivo e responder, quando programado para tal, a alguma situação, com base nos dados colhidos, de forma autônoma e sem a necessidade constante de observação humana.

Neste trabalho, optou-se pelo uso da plataforma da TagoIO. Com ela, é possível construir, de forma intuitiva e gratuita, uma maneira de apresentar ao usuário os dados coletados pelos sensores, Figura 5, e definir ações a serem tomadas caso alguma condição seja reconhecida.

Figura 5 – Dashboard criada na TagoIO.



Fonte: O autor, 2021.

3 PREPARANDO O AMBIENTE PARA O DESENVOLVIMENTO

O sistema operacional Linux (por exemplo Ubuntu 20.04 [20]) foi escolhido para o desenvolvimento devido suas características, que reduzem os gastos com licenças, e por possuir ambientes de desenvolvimento de qualidade profissional e de código aberto. Os seguintes programas ou pacotes de *software* também foram utilizados para realizar esta integração:

- Projeto LoRaMAC [21], responsável pelo gerenciamento da placa Discovery Kit, dos sensores agregados e que contém o firmware de comunicação LoRaWAN para o dispositivo. O LoRaMAC foi desenvolvido pela Semtech e é disponibilizado em um repositório público;
- Cmake [22], gerenciador de compilação;
- GCC-ARM [23], compilador para processadores ARM [24];
- Cutecom [25], software de monitoramento e interação com a porta serial do computador.

3.1 PREPARAÇÃO DO SISTEMA OPERACIONAL

Para que o desenvolvimento ocorra sem haver problemas com falta de bibliotecas, aplicativos ou compiladores, todos necessários para a fase de programação do dispositivo que monitora os sensores, alguns passos devem ser seguidos. São eles:

1. Abrir um terminal no sistema Linux;
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
```

5. Instalar os seguintes pacotes extras:

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

6. Instalar o gerenciador de versão GIT [26]:

```
sudo apt-get install git
```

7. Utilizar o GIT para baixar o LoRaMAC:

```
git clone https://github.com/AlexandroSantos/LoraMac-IKS01A2
```

8. Entrar no diretório LoraMac-IKS01A2.

```
cd LoraMac-IKS01A2
```

9. Permissão de execução. Neste passo, é necessário indicar, ao sistema operacional, que os arquivos **.sh** podem ser executados.

```
sudo chmod +x *.sh
```

10. Permissão de uso da porta serial. Algumas distribuições necessitam de uma liberação para conectar com a porta serial. Para isso, o seguinte comando é usado, lembrado que devemos substituir **usuario** pelo seu nome de usuário:

```
sudo usermod -a -G dialout usuario
```

Caso o usuário não saiba o nome de usuário, basta digitar, no terminal, o comando: `users`

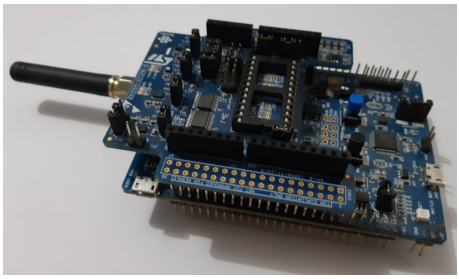
No momento, o sistema apresenta-se pronto para começar a configurar a aplicação, bastando apenas conectar o kit da STMicroelectronics (Discovery kit). Caso o usuário não tenha a distribuição Linux indicada, alguns erros, por falta de bibliotecas, podem ocorrer. Tais bibliotecas devem ser instaladas, conforme cada distribuição Linux.

3.2 PREPARANDO O DISCOVERY KIT DA STMICROELECTRONICS

Inicialmente, deve-se acoplar o módulo IKS01A02 ao Discovery Kit, sempre observando a posição dos respectivos pinos, conforme a Figura 6.

Conectar o cabo USB e, então, o sistema operacional deve alertar sobre a inserção de um disco removível (DIS_L072Z).

Figura 6 – Acoplamento do módulo IKS01A02 no Discovery kit.



Fonte: O autor, 2021.

Para que o disco removível fique disponível para gravação, é necessário abrí-lo, pelo menos uma vez, usando, por exemplo, um gerenciador de arquivos, isto equivale a dizer que devemos montar o disco no sistema.

4 CRIANDO UMA APLICAÇÃO NA TTN E NA KORE

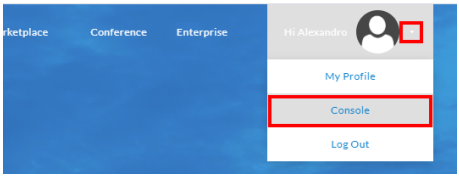
Neste capítulo discutimos como criar uma aplicação nas plataformas da TTN e da Kore.

4.1 TTN

Após a criação de uma aplicação na TTN obtém-se as informações relevantes para que seja possível configurar o *end node* de maneira que ele envie os dados para a aplicação desejada. A seguir, estão listados os passos que permitem criar uma conta na TTN e configurar um dispositivo.

1. Abrir o site da TTN;
2. Criar uma conta (SIGN UP) e acessá-la:
3. Entrar em Console, Figura 7;

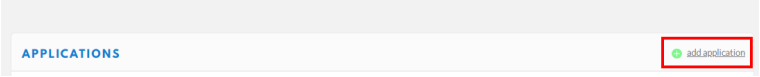
Figura 7 – Menu de acesso ao console da TTN.



Fonte: O autor, 2020.

4. Acessar APPLICATIONS para abrir, ou criar, suas aplicações;
5. Criar um nova aplicação em **add application**, Figura 8;

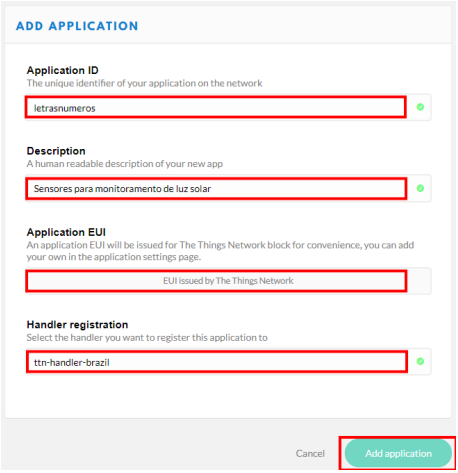
Figura 8 – Botão adicionar aplicação.



Fonte: O autor, 2020.

6. Preencher os campos e clicar em **Add application**, Figura 9;

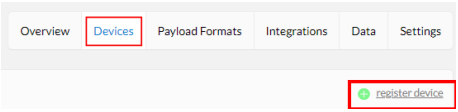
Figura 9 – Formulário de cadastro da aplicação.



Fonte: O autor, 2020.

7. Clicar no botão **Devices** para adicionar um dispositivo e, em seguida, clicar em **register device**, como indicado na Figura 10;

Figura 10 – Menu de acesso aos *end nodes*.



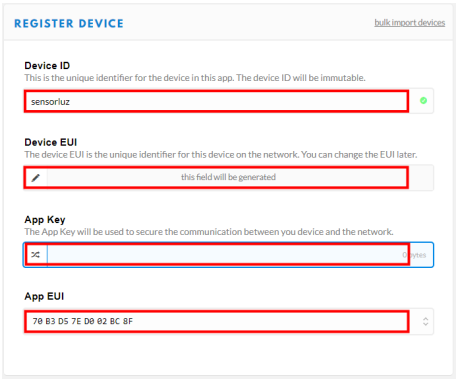
Fonte: O autor, 2020.

8. Preencher o campo **Device ID**, Figura 11, como for mais conveniente; Caso o usuário possua o 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.

9. Clicar em **Settings**, indicado na Figura 12;

10. Definir o **Activation Method** em ABP e clicar em **Save**.

Figura 11 – Formulário de registro de *end nodes*.

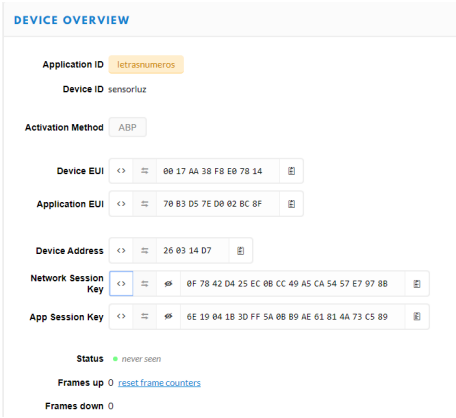


Fonte: O autor, 2020.

Figura 12 – Menu de acesso às configurações.

Fonte: O autor, 2020.

Figura 13 – Informações para configurar o LoRaMAC



Fonte: O autor, 2020.

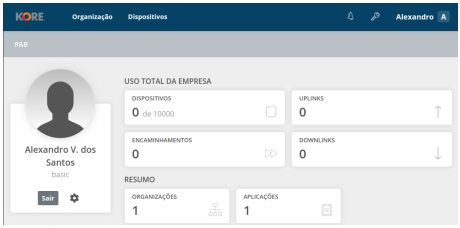
Os dados que aparecem em **Device Overview**, como ilustrado na Figura 13, são usados na edição do código fonte LoRaMAC a ser compilado e gravado no dispositivo.

4.2 KORE

Além de uma rede gratuita como a TTN, é possível utilizar, também, uma rede comercial para conectar os dispositivos. Para tanto, será utilizada a plataforma da Kore sobre a rede Everynet, seguindo os passos:

1. Abrir o site da Kore;
2. Efetuar o login e, então, tela da Figura 14 será apresentada;

Figura 14 – Tela de identificação de usuário.



Fonte: O autor, 2020.

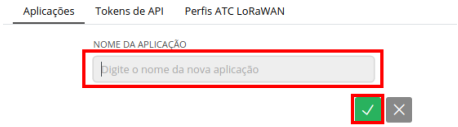
Nesta tela, estão listadas todas as aplicações e dispositivos criados, além das últimas operações realizadas.

3. Selecionar, no menu superior, **Organização**. Depois, na lista apresentada, selecionar **EEL7515**, criada para fins didáticos da disciplina, conforme Figura 15;
4. Para criar uma aplicação, pressionar o botão **+ Aplicação**.
5. Nomear sua aplicação e incluí-la no grupo ‘Principal’. Pressionar o botão verde, indicado na Figura 16.
6. Para cadastrar o dispositivo, selecionar **Dispositivos**. No menu superior da Figura 17, selecionar a aplicação desejada e, então, pressionar o botão **+ Dispositivo**.

Figura 15 – Tela de seleção de organização.

Fonte: O autor, 2020.

Figura 16 – Campo de inserção do nome da aplicação.



Fonte: O autor, 2020.

Figura 17 – Cadastro de *end nodes*.

Fonte: O autor, 2020.

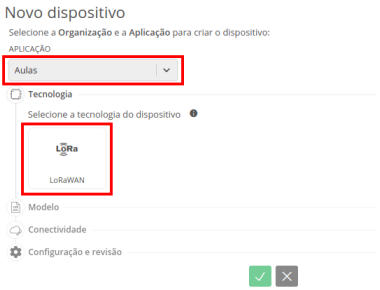
7. Selecionar a tecnologia LoRaWAN, Figura 18.

8. Preencher o formulário com as informações do dispositivo a ser usado.

9. Criado o dispositivo, selecioná-lo e, então, ativá-lo, pressionando o botão **Ativar**, Figura 19.

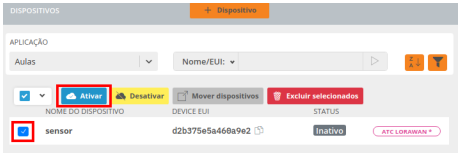
10. Quando o sensor for novamente selecionado, a tela de apresentação de dados recebidos irá aparecer, Figura 20. Nesta tela, os dados, em tempo real, e a configuração do dispositivo serão apresentados.

Figura 18 – Selecionar o tipo de rede.



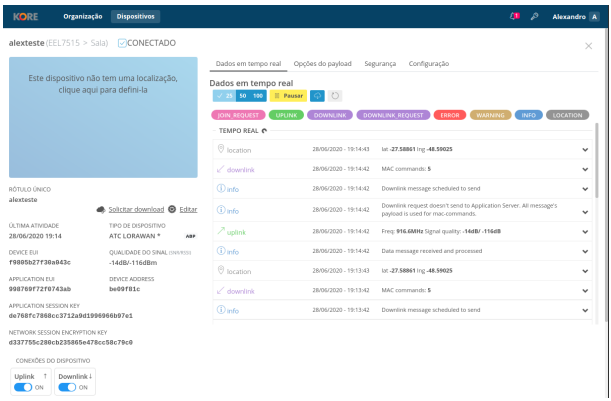
Fonte: O autor, 2020.

Figura 19 – Ativação do dispositivo.



Fonte: O autor, 2020.

Figura 20 – Tela de visualização do dados.



Fonte: O autor, 2020.

5 CONFIGURAR E COMPILAR O CÓDIGO FONTE

Uma vez obtidas as chaves e outras informações relevantes para conexão à rede IoT, é necessária a inserção dessas informações no código fonte (LoRaMAC), de forma a serem interpretadas corretamente pelo compilador.

Dentre os arquivos, fornecidos pelo desenvolvedor original, tem-se o *main.c* e o *Commissioning.h*. No arquivo *Commissioning.h* são encontradas as constantes de configuração de rede, tais como o intervalo de envio, a faixa de frequência utilizada pelo rádio e a sub-banda, que devem ser definidas 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 seja conectado ao *gateway* e à TTN ou Kore, de forma adequada, é necessário editar o *Commissioning.h*, seguindo os passos listados a seguir:

- Passo 1: Abrir, com um editor de texto (gedit, nano, vi, geany, sublime), o arquivo **Commissioning.h** localizado na seguinte pasta: **src/apps/LoRaMac/classA/B-L072Z-LRWAN1**.
  Neste arquivo, serão editadas as linhas contendo as seguinte definições (*#define*), seguindo[1] ser usados os dados conforme os dados da Figura 13 para a TTN e para a Kore os da Figura 20:

  - [Linha 4] OVER_THE_AIR_ACTIVATION 0
    Desativa ao método de ativação OTAA, passando 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**

[1] Note que estes dados são os referentes à aplicação exemplo gerada na preparação deste trabalho. Quando uma nova aplicação for gerada outras chaves serão criadas pela TTN ou Kore, e então os dados em vermelho deverão ser modificados de acordo.

- [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, configura-se o intervalo de envios, dado em milisegundos.
- [Linha 25] SUBBAND 2
  Se a rede usada for a da TTN, o valor é 2. Se for a rede da Kore, então o valor deverá ser 0.

- Passo 2: Salvar o arquivo **Commissioning.h**.

A TTN é uma rede gratuita, o que 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, os intervalos devem ser maiores, a cada 1 hora, por exemplo, excetuando-se em casos de violação de regra, como o aumento repentino da temperatura de uma câmara fria. Como a rede LoRaWAN preza pela economia no tráfego de dados, é fundamental informar apenas o estritamente necessário.

5.1 FORMATANDO O PAYLOAD

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

//*****************************************************//
```

Para outros detalhes de configuração observar o apêndice A.

5.2 RECUPERANDO OS DADOS

A recuperação das informações originais de temperatura, umidade e tensão, respectivamente, é dada por:

```
float temperature=AppDataBuffer[0]+AppDataBuffer[1]/10 ;

float humidity=AppDataBuffer[2]+AppDataBuffer[3]/10 ;

float volts=AppDataBuffer[7]+AppDataBuffer[8]/10 ;
```

Para a pressão, por sua vez:

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

Das inúmeras formas de montar o pacote de dados, deve-se utilizar a

forma mais compacta, visando a redução de tráfego de dados.

5.3 COMPILAR E VERIFICAR O PROJETO

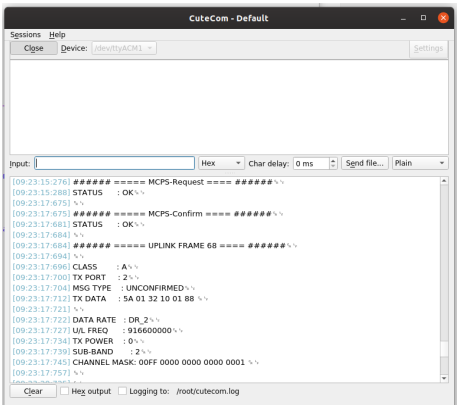
No diretório raiz do projeto, encontramos os scripts **create.sh** e **program.sh**. O script **./create.sh**, usado para compilar, cria o diretório **build**. No caminho **build/src/apps/LoRaMac**, encontra-se o arquivo **LoRaMac-classA.bin**, que deve ser copiado para o disco removível **DIS_L072Z**, para rodar o programa na placa.

Para compilar e programar a placa é utilizado **./program.sh**, fazendo com que todos os passos sejam feitos automaticamente.

5.4 VERIFICAÇÃO LOCAL

Utilizando o programa Cutecom, é possível observar o comportamento do rádio, ilustrado na Figura 21, quais pacotes foram transmitidos e recebidos. Ao abrir o Cutecom deve-se selecionar o **Device**, em uma das portas seriais /dev/ttyACM*, e pressionar o botão **Open**. Com o rádio conectado, para ver as mensagens iniciais, que indicam os parâmetros configurados em **Commissioning.h**, basta pressionar o botão de reset no kit .

Figura 21 – Tela do Cutecom.

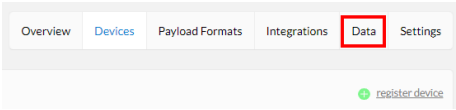


Fonte: O autor, 2020.

5.5 VERIFICAÇÃO REMOTA

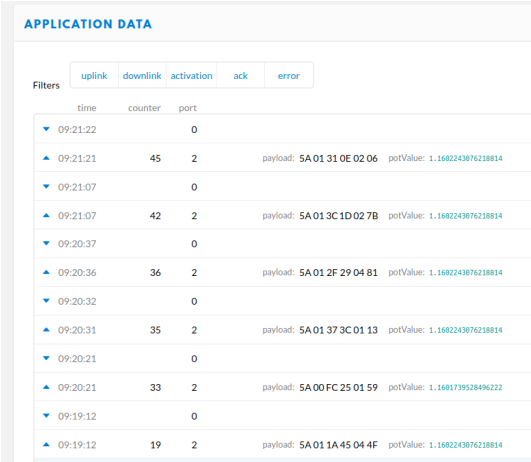
No site da TTN, com a aplicação aberta, ao pressionar o botão **Data**, Figura 22, abrirá a tela ilustrada na Figura 23.

Figura 22 – Menu de acesso aos dados.



Fonte: O autor, 2020.

Figura 23 – Dados transmitidos e recebidos



Fonte: O autor, 2020.

É possível, então, monitorar os pacotes recebidos e enviados pelo dispositivo, além de verificar várias informações, como parâmetros de transmissão usados pelo rádio, potência recebida no *gateway*, razão sinal ruído, entre outros, pressionando, para tal, a seta azul no início da mensagem.

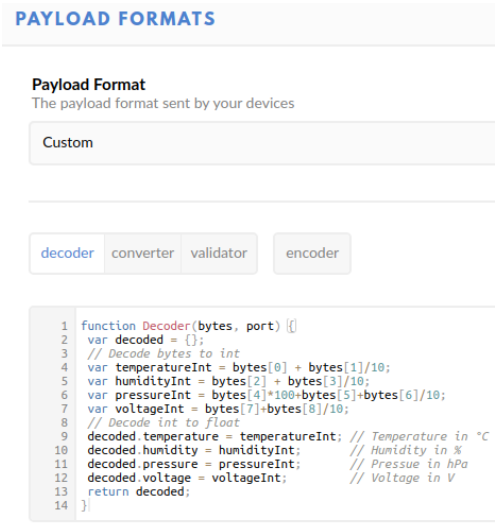
Neste painel, o *payload* é apresentado como uma sequência de números em hexadecimal, de tamanho 9 bytes, conforme definido previamente no arquivo *main.c*, que pode ser configurado, para visualização dos dados em um formato legível, na aba **Payload Formats**.

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;

}
```

Copiar e colar na aba indicada, conforme a Figura 24. Na aba **APPLI-**

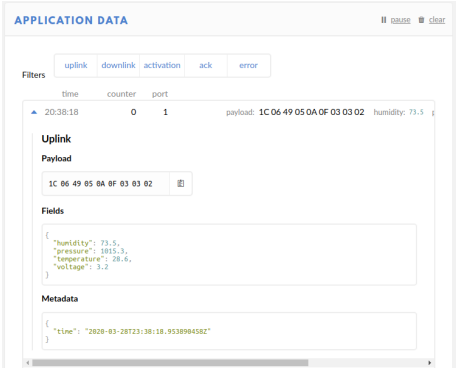
Figura 24 – Decodificação do pacote de dados.



Fonte: O autor, 2020.

CATION DATA, Figura 25, expandindo a visualização dos dados, pode-se observar a informação devidamente representada, de forma legível ao usuário.

Figura 25 – Expandindo os dados.



Fonte: O autor, 2020.

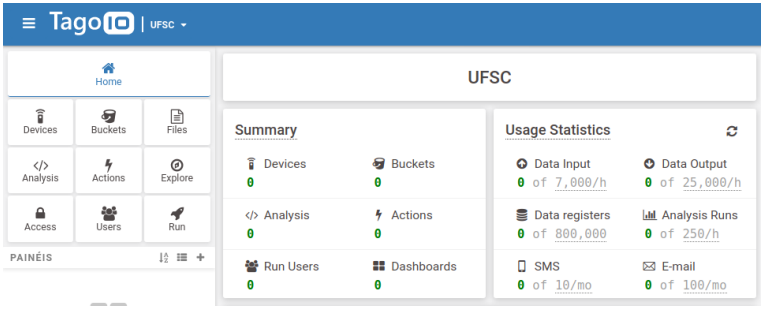
6 INTERFACE GRÁFICA - DASHBOARD

A TagoIO possui um conjunto de ferramentas que, combinadas com a plataforma da TTN ou da Kore, podem ser utilizadas 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.

6.1 CRIANDO UMA CONTA NA TAGOIO

Para acesso às ferramentas de criação da *dashboard*, é necessária a criação de uma conta na TagoIO, que permitirá o acesso a algumas funções e ferramentas. Após a ativação da conta e a autenticação, é possível encontrar informações de uso e equipamentos, na tela principal, Figura 26.

Figura 26 – Tela principal da TagoIO.



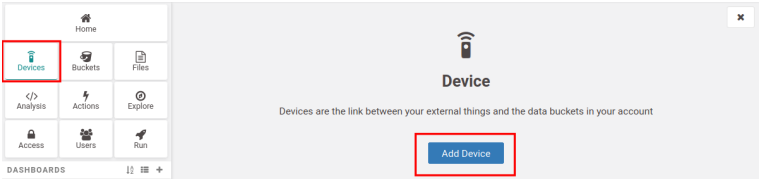
Fonte: O autor, 2020.

Assim, como no site da TTN, se faz necessária a configuração da aplicação. Para tal, serão utilizados os dados recebidos pela TTN e enviados à TagoIO, mostrando-os numa interface gráfica que facilitará sua interpretação e a tomada de decisão.

6.2 CONECTANDO A TAGOIO COM A TTN

Como no exemplo da TTN, adicionar um dispositivo se faz necessário. Para isso, utiliza-se a aba esquerda, pressionando em **Devices** e, em seguida, em **Add Device**, Figura 27.

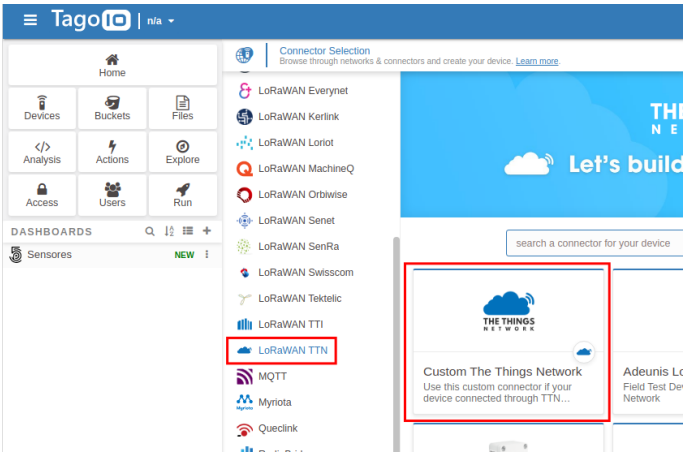
Figura 27 – Menu de acesso a registro de *end nodes*.



Fonte: O autor, 2020.

Deve-se vincular o dispositivo à TTN, Figura 28.

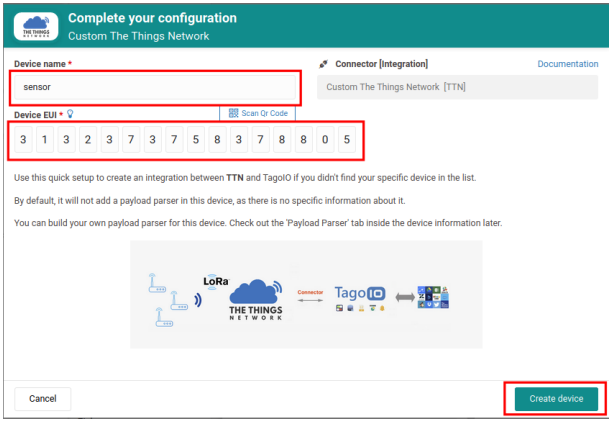
Figura 28 – Tela de conexão TTN/TagoIO.



Fonte: O autor, 2020.

No site da TTN, podem estar cadastrados diversos dispositivos, então é necessário direcionar a um específico, preenchendo a solicitação, conforme a Figura 29, onde o campo **Device EUI** identifica o dispositivo.

Figura 29 – Direcionando ao dispositivo.

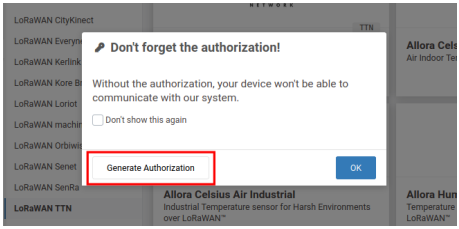


Fonte: O autor, 2020.

Uma vez adicionados os dados necessários, o botão **Create device** deve ser pressionado.

Com a interação entre a TagoIO e a TTN realizada, uma chave de segurança deve ser criada. Para isto, deve-se acionar o botão **Generate Authorization**, Figura 30.

Figura 30 – Acesso ao Gerador de chave de autorização.



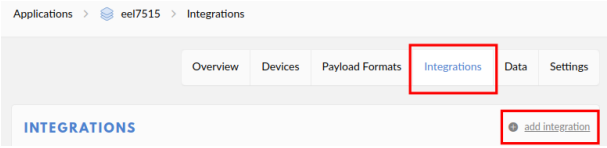
Fonte: O autor, 2020.

Com a chave criada, deve-se informar, no site da TTN, qual chave enviar para TagoIO, a fim de que as informações sejam contabilizadas no dispositivo criado.

De volta ao site da **TTN**, abrindo, dentro da aplicação, a aba **Integrations**, uma nova interação com o botão **add integration** é adicionada, Figura 31,

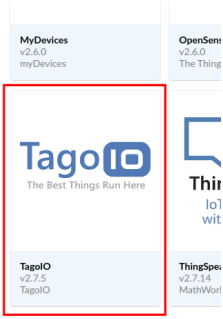
que abre uma lista de possíveis integradores, da qual deve-se escolher a TagoIO, Figura 32.

Figura 31 – Menu de acesso à configuração de integração da TTN.



Fonte: O autor, 2020.

Figura 32 – Integrando TagoIO à TTN.



Fonte: O autor, 2020.

Selecionada a integração, um identificador único (**Process ID**) é inserido, como uma chave de acesso (neste caso **default key**), além da chave de autorização. A Figura 33 mostra um exemplo deste preenchimento.

O botão **Add integration** deve ser pressionado, finalizando a configuração, integrando os sites e permitindo a transferência de dados para TagoIO.

6.3 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 LoRaWAN Everynet (Custom Everynet) ao invés da TTN .

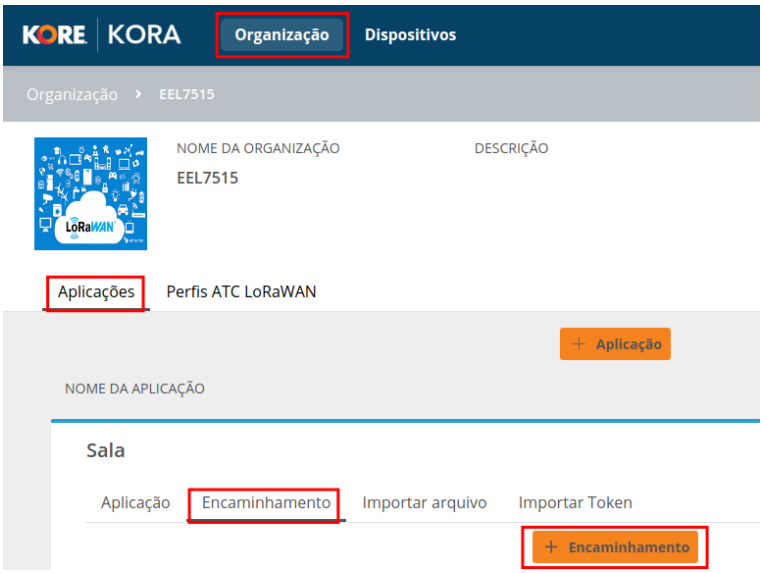
Figura 33 – Configurando a integração (Site TTN).

Fonte: O autor, 2020.

No site da Kore, é necessária a configuração do encaminhamento, de forma similar ao utilizado na TTN, respeitando as convenções de cada plataforma. Para tal, deve-se pressionar o botão **Organização**, selecionar a aplicação e, depois, pressionar **Encaminhamento**, Figura 34.

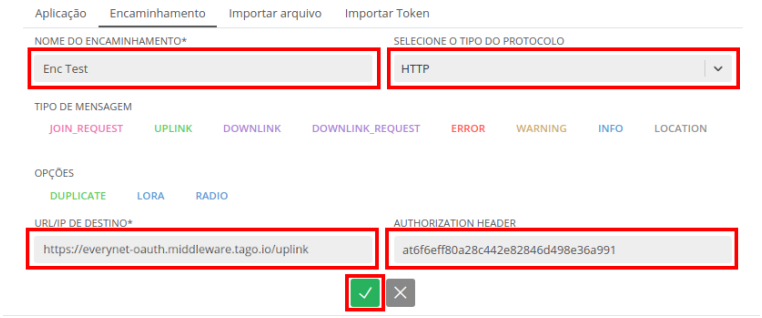
Caso não haja encaminhamento definido, a tela de adição é apresentada e, então, basta pressionar o botão **+ Encaminhamento**. Em seguida, o formulário deve ser preenchido, conforme indicado na Figura 35.

Figura 34 – Tela de configuração da Kore.



Fonte: O autor, 2020.

Figura 35 – Definindo a forma de encaminhamento.

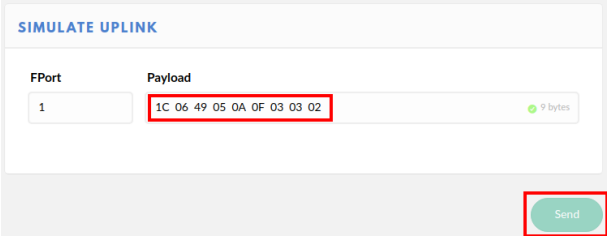


Fonte: O autor, 2020.

6.4 SIMULANDO O ENVIO DE DADOS

Para verificar se a ligação foi devidamente construída, utiliza-se o simulador de recebimento de dados, do site da TTN, na aba **Devices**, Figura 36.

Figura 36 – Simulando o envio de dados à TTN.



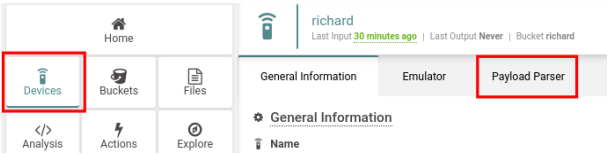
Fonte: O autor, 2020.

Neste exemplo, o pacote $1C0649050A0F030302_{Hex}$ representa uma temperatura de 28.6 ^{o}C, pressão atmosférica de 1015.3 hPa, umidade de 73.5 % e tensão de 3.2 V. Podendo-se testar tanto o envio, para a TagoIO, quanto o decodificador de *payload*, no site da TTN.

6.5 INTERPRETANDO OS DADOS

Como mencionado na TTN, os dados estão codificados e devem ser interpretados pelo site. Tendo isto em mente, uma forma de decodificação dos campos enviados pelos dispositivos, deve ser criada. Para este caso, consideramos temperatura, umidade, pressão atmosférica e tensão.

Figura 37 – Tela configuração do desempacotador de dados.



Fonte: O autor, 2020.

Acessando a aba **Payload Parser**, Figura 37, são encontrados alguns exemplos e maneiras de decodificar e criar novas variáveis, necessárias à aplicação.

Utilizando como base o **Parse Example for LoRaWAN**, que pode ser observado pressionando a área destacada como indicado acima, são suprimidas variáveis sem uso para que possam ser acrescentadas as de interesse para o objetivo deste trabalho. Estas alterações são indicadas dentro do código editado.

```
//************************************************************************
//***************** 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.

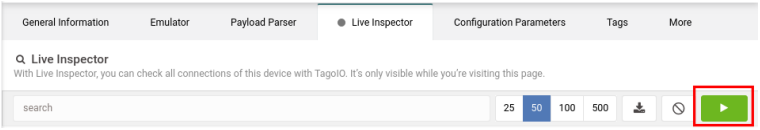
```
/* This is an generic payload parser example.
** The code find the payload variable and parse it if exists.
**
** IMPORTANT: In most case, you will only need to edit the parsePayload
** function.
**
** Testing:
** You can do manual tests to this parse by using the Device Emulator.
** Copy and Paste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

6.6 SIMULANDO O RECEBIMENTO DE DADOS

Uma forma de testar a interpretação de dados é utilizando a ferramenta de emulação, encontrada na aba **Live Inspector**, Figura 38.

Figura 38 – Ativar a simulação.



Fonte: O autor, 2020.

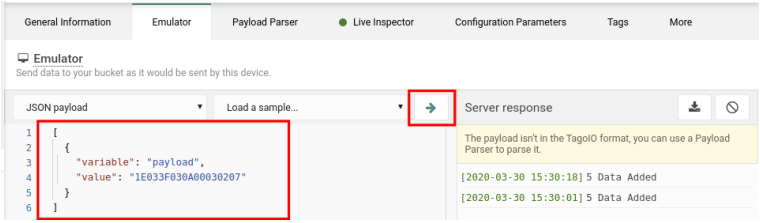
Na aba **Emulator** dá-se início à variável **payload**, no formato **JSON**, que é responsável pelo transporte dos dados enviados pelos dispositivos ao site. Segue o código, abaixo:

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

Conforme mostrado na Figura 39, pressionando a seta destacada, o servidor exibe, no campo **Server response**, a data, hora e a quantidade de dados adicionada. Neste contexto, **payload** também é uma variável e será armazenada. Neste exemplo, 5 dados serão armazenados, ao invés de 4.

Retornando à aba **Live Inspector**, é possível verificar se os dados foram devidamente identificados e as variáveis corretamente inicializadas, ilustrado da Figura 40. 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,

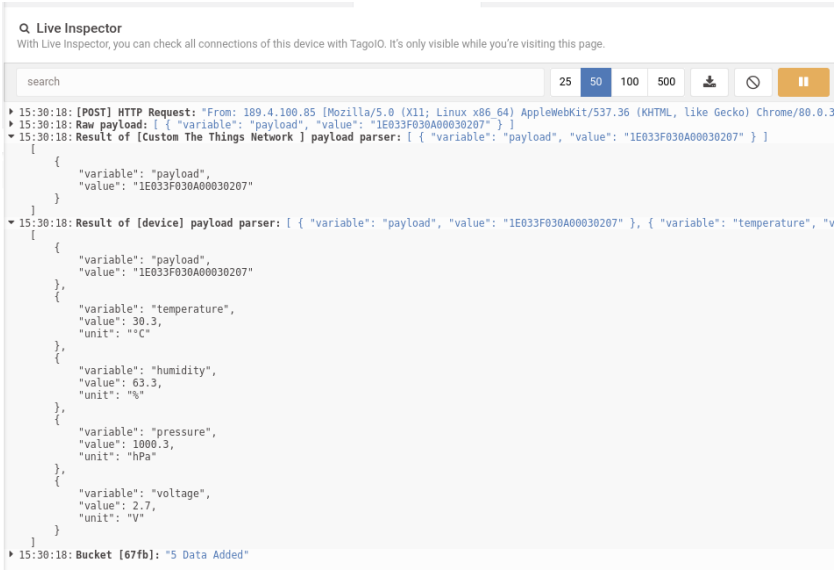
Figura 39 – Configurando o pacote no emulador.



Fonte: O autor, 2020.

deixando a cargo do desenvolvedor a montagem da tela de apresentação dos dados (ou **Dashbord**).

Figura 40 – Apresentação dos dados recebidos já decodificados.

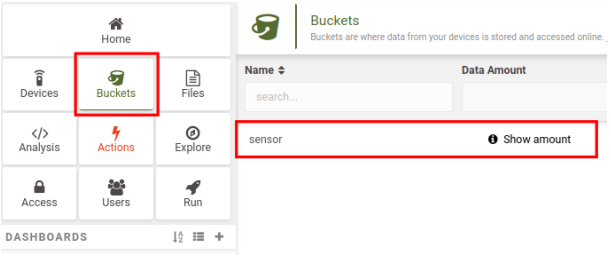


Fonte: O autor, 2020.

6.7 ARMAZENAMENTO DE DADOS

Ao enviar dados a TagoIO, estes ficarão armazenados por um tempo determinado nos chamados baldes (ou **Buckets**).

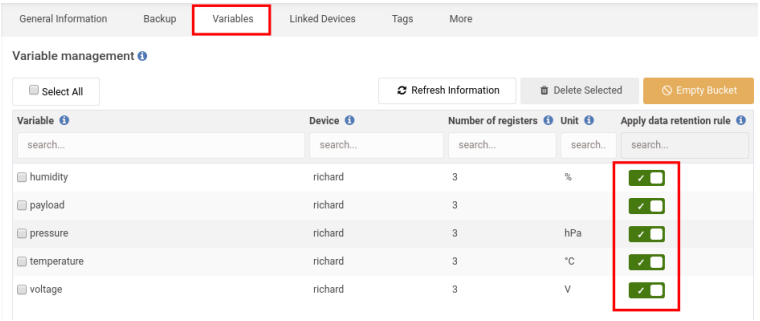
Figura 41 – Acesso à configuração do armazenamento de dados.



Fonte: O autor, 2020.

Selecionando o *Bucket*, Figura 41, criado automaticamente quando o primeiro dispositivo é adicionado, na aba **Variables**, Figura 42, surgem listadas as variáveis encontradas pelo gerenciador. Se somente o simulador for utilizado, apenas as variáveis criadas no interpretador são mostradas mas, quando um dispositivo real é utilizado, outras variáveis aparecem e devem 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.

Figura 42 – Seleção dos dados a serem salvos.



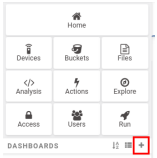
Fonte: O autor, 2020.

6.8 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 este trabalho, uma interface simples permite visualizar os dados recebidos dos dispositivos em uma tela onde estes são apresentados dinamicamente.

Primeiramente, uma *dashboard* deve ser adicionada, conforme indicado na Figura 43.

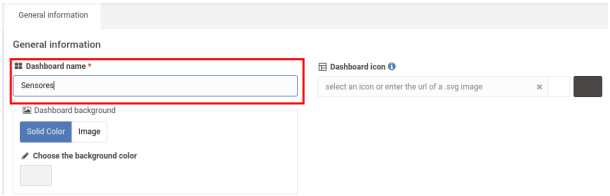
Figura 43 – Botão para adicionar uma *dashboard*.



Fonte: O autor, 2020.

Na tela **General information**, Figura 44, acrescenta-se um nome para o dashboard e, então, **Save** é pressionado.

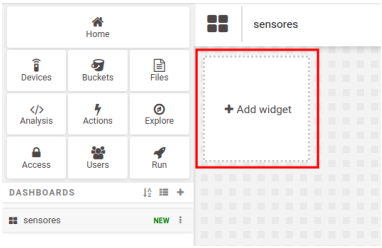
Figura 44 – Inserção do nome para criação da *dashboard*.



Fonte: O autor, 2020.

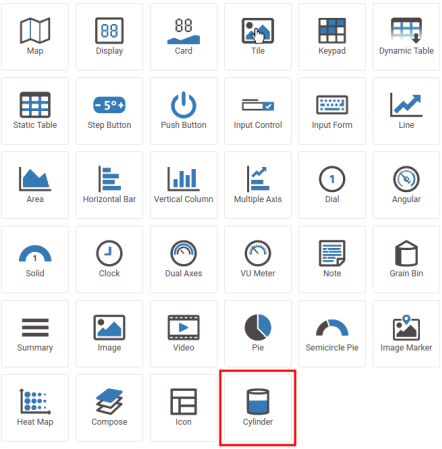
A apresentação passa a ser efetivamente construída, através da inserção de objetos de visualização de dados. Quando a área em destaque na Figura 45 é pressionada, uma lista de objetos, Figura 46 , é apresentada e a escolha de um deles deve-se ao tipo e forma dos dados a serem exibidos. Por exemplo, para temperatura, o objeto **Cylinder** pode representar um termômetro.

Figura 45 – Adicionar *widgets*.



Fonte: O autor, 2020.

Figura 46 – Objetos ou *widgets*.

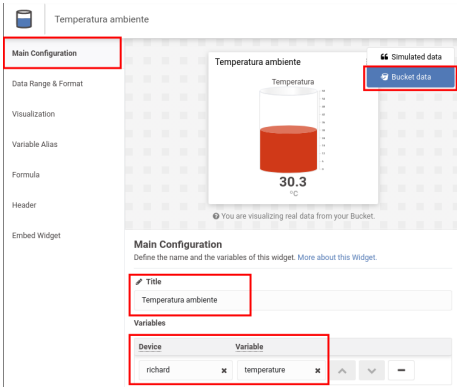


Fonte: O autor, 2020.

Na configuração principal, a inserção de título, dispositivo, variável e origem dos dados associados ao *widget* faz-se necessária, Figura 47. Outras características também podem ser modificadas, como cor, unidade física, limites para exibição e operações, ou fórmulas, com os valores a serem apresentados.

Todos os demais *widgets*, apresentados na Figura 46, podem ser configurados seguindo os mesmos passos, respeitando as particularidades de cada *widget*.

Figura 47 – Configurando um *widgets*.

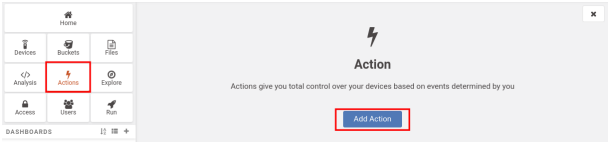


Fonte: O autor, 2020.

6.9 ALERTAS E MENSAGENS

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

Figura 48 – Botão de inserção de ações.

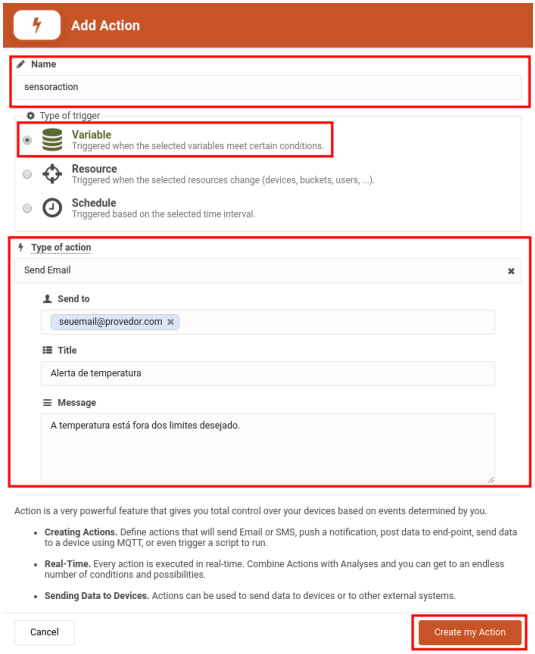


Fonte: O autor, 2020.

Adicionando uma nova ação, Figura 48, a janela, ilustrada na Figura 49, será exibida para preenchimento, em seguida, o botão **Create my Action** deve ser pressionado. Seleciona-se, então, **Single device** para inserir o dispositivo que enviará os dados.

No campo **Trigger**, Figura 50, definimos a condição de envio do alerta. Para este trabalho, o modo de alerta definido é um e-mail, enviado quando a temperatura excede 40 ^{o}C.

Figura 49 – Configurando uma ação.



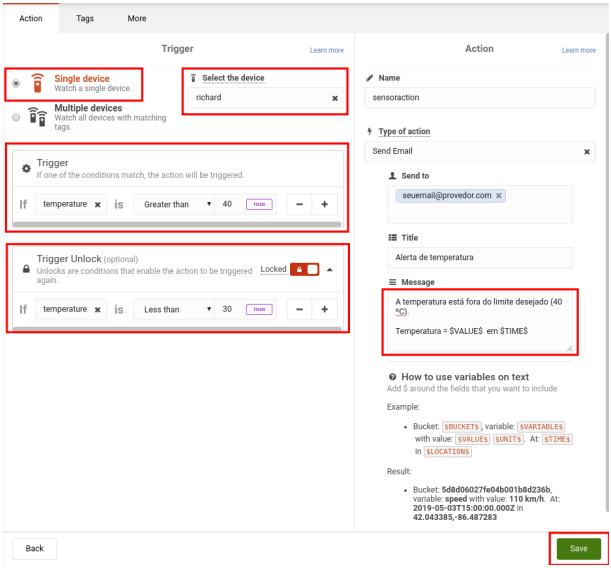
Fonte: O autor, 2020.

No campo **Trigger Unlock**, define-se a condição em que uma nova mensagem pode ser enviada. Sem esta opção habilitada, a cada leitura do dispositivo, um novo e-mail seria enviado, tornando a função redundante, em caso de leituras muito próximas. Para esta aplicação, o envio será novamente habilitado somente se a temperatura for menor que 30 ^{o}C.

No campo **Message**, são incluídas as informações relacionadas à variável testada, como valor e momento do ocorrido.

Com a aplicação devidamente configurada e operante, basta a publicação para que as informações possam ser observadas pelo usuário final.

Figura 50 – Definido tipos de mensagens.



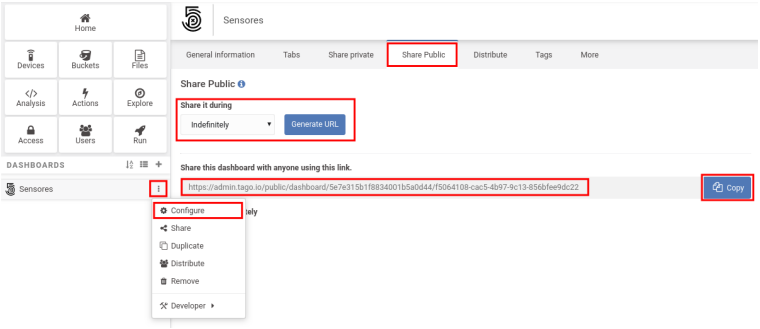
Fonte: O autor, 2020.

6.10 PUBLICANDO O DASHBOARD

Ao finalizar e testar o *dashboard*, é necessário fornecer ao usuário uma forma de acesso à aplicação criada, o que pode ser feito através de um *link* gerado pela TagoIO. Uma vez acessada a aba **Share Public**, Figura 51, o endereço já está disponível, cabendo ao desenvolvedor avaliar a necessidade de sua disponibilidade, optando por tempo indeterminado ou limitado.

O *link* fornecido dá ao usuário apenas a possibilidade de visualização, não sendo possível edição ou alteração dos dados, podendo ser compartilhado com diversos usuários, disseminando, assim, seu conteúdo.

Figura 51 – Publicando o *dashboard*.



Fonte: O autor, 2020.

7 CONCLUSÃO

Neste trabalho um *end node* foi configurado para a integração com servidor de rede LoRaWAN e um servidor de aplicação. Foram utilizados o projeto LoRaMAC, as redes TTN e Kore, ambas integradas a uma aplicação TagoIO. Este trabalho trouxe ferramentas para a construção de um *dashboard* e a integração com redes IoT, auxiliando a didática de uma disciplina do curso de Graduação em Engenharia Eletrônica, e propiciando maior eficiência na disseminação e fixação do conteúdo. Além disso, mostrou-se útil ao desenvolvimento de novas aplicações e plataformas fora da disciplina, já tendo sido utilizada em outros Trabalhos de Conclusão de Curso.

Como uma nota de cautela, é importante saber que as ferramentas utilizadas neste trabalho estão em constante evolução e portanto é possível que alguns detalhes venham a sofrer alguma modificação. Porém, nestes casos, o conteúdo geral poderá ser facilmente adaptado seguindo as informações dos provedores das plataformas utilizadas neste trabalho.

REFERÊNCIAS

1 DUNKO, G. et al. A reference guide to the internet of things. *Published By Bridgera LLC, North Carolina*, 2017. 19

2 MEKKI, K. et al. A comparative study of lpwan technologies for large-scale iot deployment. *ICT express*, Elsevier, v. 5, n. 1, p. 1–7, 2019. 19

3 AUGUSTIN, A. et al. A study of lora: Long range & low power networks for the internet of things. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 16, n. 9, p. 1466, 2016. 19

4 CORPORATION, S. *What is LoRa?* 2021. Acessado em : 2021-02-20. Disponível em: <https://www.semtech.com/lora/what-is-lora>. 19

5 CORPORATION, S. *Company Overview | Semtech*. 2021. Acessado em : 2021-02-20. Disponível em: <https://www.semtech.com/company>. 19

6 REYNDERS, B.; POLLIN, S. Chirp spread spectrum as a modulation technique for long range communication. In: IEEE. *2016 Symposium on Communications and Vehicular Technologies (SCVT)*. [S.l.], 2016. p. 1–5. 20

7 SPRINGER, A. et al. Spread spectrum communications using chirp signals. In: . [S.l.: s.n.], 2000. p. 166 – 170. ISBN 0-7803-6323-X. 20

8 LORAWAN. 2021. Acessado em : 2021-03-15. Disponível em: <https://lora-alliance.org/about-lorawan/>. 20

9 ALLIANCE, L. *LoRa Alliance*. 2020. Acessado em : 2021-03-15. Disponível em: <https://lora-alliance.org/>. 20

10 THE Things Network. 2021. Acessado em : 2021-03-15. Disponível em: <https://www.thethingsnetwork.org/>. 20, 23

11 KORE. 2021. Acessado em : 2021-02-20. Disponível em: <https://br.korewireless.com/empresa/sobre>. 20, 23

12 TAGOIO. *TagoIO: Cloud IoT Platform | Internet of Things*. 2021. Acessado em : 2021-03-20. Disponível em: <https://tago.io>. 20

13 STMICROELECTRONICS. *STM32L0 Discovery kit LoRa, Sigfox, low-power wireless*. 2020. Acessado em : 2020-12-01. Disponível em: <https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html>. 20, 22

14 STMICROELECTRONICS. *STMicroelectronics*. 2020. Acessado em : 2020-12-01. Disponível em: <https://www.st.com/content/st_com/en/about/st_company_information/who-we-are.html>. 20

15 UFSC/EEL. *Plano de Ensino 2020.1*. 2020. Acessado em : 2020-08-20. Disponível em: <https://geltro.paginas.ufsc.br/files/2016/07/EEL7515-T%C3%B3pico-Avan%C3%A7ado-em-Processamento-de-Sinais-II.pdf>. 20

16 WIRELESS RF LoRa Transceivers SX1276 Semtech. 2021. Acessado em : 2021-02-20. Disponível em: <https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276>. 21

17 X-NUCLEO-IKS02A1. 2021. Acessado em : 2021-05-08. Disponível em: <https://www.st.com/en/ecosystems/x-nucleo-iks02a1.html>. 22

18 KHOMP. 2021. Acessado em : 2021-04-20. Disponível em: <https://www.khomp.com/pt/institucional/>. 23

19 ITG 200 Indoor. 2021. Acessado em : 2021-04-20. Disponível em: <https://www.khomp.com/pt/produto/itg-200/>. 23

20 UBUNTU. 2021. Acessado em : 2021-02-20. Disponível em: <https://ubuntu.com/>. 25

21 LORAMAC. 2021. Acessado em : 2020-10-19. Disponível em: <https://github.com/Lora-net/LoRaMac-node>. 25

22 CMAKE. 2021. Acessado em : 2020-08-20. Disponível em: <https://cmake.org/>. 25

23 GCC ARM. 2021. Acessado em : 2020-10-19. Disponível em: <https://gcc.gnu.org/onlinedocs/gcc/ARM-Options.html>. 25

24 ARM Processors. 2021. Acessado em : 2020-10-19. Disponível em: <https://www.arm.com/products/silicon-ip-cpu>. 25

25 CUTECOM. 2021. Acessado em : 2020-08-20. Disponível em: <http://cutecom.sourceforge.net/>. 25

26 GITHUB. 2021. Acessado em : 2020-10-19. Disponível em: <https://github.com/>. 26

Apêndices

APÊNDICE A – PERSONALIZAÇÃO DO CÓDIGO FONTE LORAMAC

No projeto LoRaMAC 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, são utilizadas portas digitais, analógicas e de comunicação para implementar novas funcionalidades, respeitando a estrutura básica do código fonte. Para tanto, é necessário editar o arquivo **main.c**, encontrado na pasta **src/apps/LoRaMac/classA/B-L072Z-LRWAN1**.

No arquivo **Commissionig.h**, são encontradas algumas configurações para o rádio e para a conexão com o *gateway*, além do intervalo de transmissão periódica de dados,

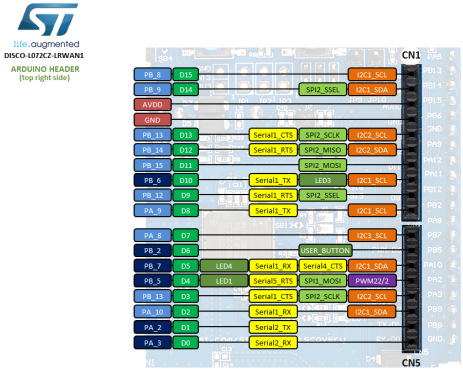
[linha 21] #define APP_TX_DUTYCYCLE 60000

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

Objetos são instanciados, como, por exemplo, o Led4, conectado à porta PB_7 da placa STM32L072CZY6TR, indicado na Figura 52.

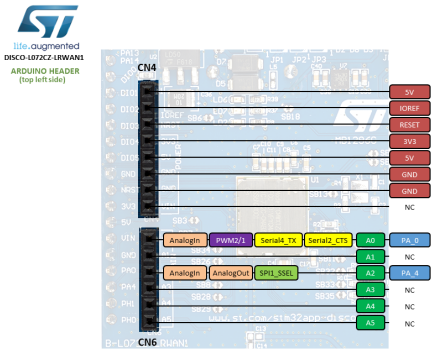
Ainda na Figura 52, observam-se os pinos PB_8 e PB_9, utilizados para comunicação i2c entre a placa de desenvolvimento e o kit de sensores. O pino PA_0, ilustrado na Figura 53, está configurado como entrada analógica. Deve-se cuidar, aqui, dos limites de tensão de operação do microcontrolador são de 0 V até 3.3 V, pois podem ocasionar dano permanente se 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.

Figura 52 – Pinagem dos conectores CN1 e CN5.



Fonte: https://os.mbed.com/platforms/ST-Discovery-LRWAN1/ (2020).

Figura 53 – Pinagem dos conectores CN4 e CN6.



Fonte: https://os.mbed.com/platforms/ST-Discovery-LRWAN1/ (2020).

A função PrepareTxFrame() é responsável por montar o pacote de dados para o envio. Nela pode-se implementar, por exemplo, a leitura de um sensor, sendo necessárias as bibliotecas responsáveis pelo controle do sensor ou dispositivo anexado, escritas em C. Dentro do arquivo **tools.c** há uma biblioteca para os sensores HTS221 (temperatura e umidade) e LPS22HB (pressão atmosférica) do Kit IKS01A2. Na Figura 54, o exemplo utilizado

para o envio do pacote de dados.

Figura 54 – Código de montagem dos pacotes para o envio.

```
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     //***********************************************************//
```

Fonte: O autor, 2020.

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

- Lê-se os valores dos sensores para as variáveis correspondentes (Linhas 368-371);
- Indica-se o tamanho do pacote em bytes (Linha 374);
- Dividi-se em parte inteira e facionária (apena uma casa decimal) e armazena-se, sequencialmente, os valores no *buffer* (AppDataBuffer - Linhas 377-388).

Na Figura 55, observa-se 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.

A função McpsIndication(), além de outras atividades, é responsável pelo recebimento dos dados e seu tratamento. Nesta parte, é possível acionar dispositivos remotamente ou requisitar uma leitura adicional.

Na Figura 56, tem-se um trecho do código no qual observa-se que, com o recebimento de apenas um **byte**, com o valor de $0x01_{16} = 1_{10}$, o Led4 é acionado, confirmando o recebimento da informação.

Para (acender) apagar o Led4 é possível, também, enviar pelo sistema da TTN o valor ($0x01_{16}$) $0x00_{16}$ para o kit, como ilustrado na Figura 57.

Figura 55 – Código de montagem da apresentação no terminal serial.

```
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     //*****************************************************************************
```

Fonte: O autor, 2020.

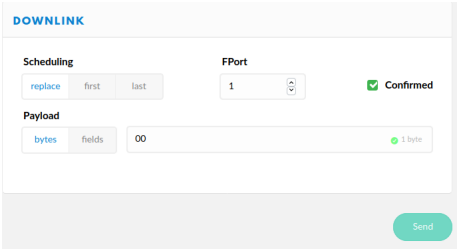
O Led4, vermelho, ilustrado na Figura 58, está posicionado, na placa de desenvolvimento, abaixo do módulo de sensores.

Figura 56 – Entrada de dados.

```
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;
```

Fonte: O autor, 2020.

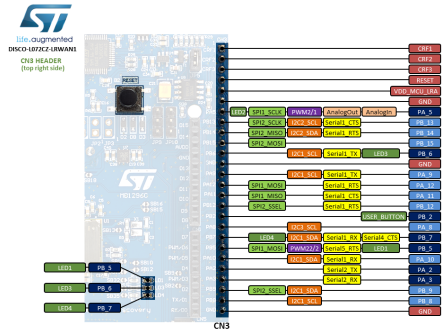
Figura 57 – Envio do comando apagar Led4, no site da TTN - Aba Devices.



Fonte: O autor, 2020.

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.

Figura 58 – Posição dos leds na placa de desenvolvimento.



Fonte: https://os.mbed.com/platforms/ST-Discovery-LRWAN1/ (2020).