PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
ENGENHARIA DE COMPUTAÇÃO

SISTEMA DE ALERTA PARA IDOSOS EM RISCO – INTEGRAÇÃO DE DADOS MULTISSENSORIAIS PARA AVALIAÇÃO REMOTA UTILIZANDO O HT32SX

Wiliam Lucas Nascimento

Trabalho de Conclusão de Curso.

Orientador: Prof. Júlio César Marques de Lima

Porto Alegre
2024

SUMÁRIO

1. INTRODUÇÃO

Na atual conjuntura tecnológica, a Internet das Coisas (IoT) destaca-se como um dos pilares fundamentais da transformação digital. O setor tem atraído investimentos globais significativos, com organizações desenvolvendo soluções práticas e economicamente viáveis para atender à crescente demanda do mercado.

A Internet das Coisas representa um segmento da computação que busca estabelecer conexões entre elementos físicos diversos, abrangendo desde dispositivos domésticos em centros urbanos até sistemas de monitoramento agropecuário em áreas rurais. Esta integração visa diminuir a distância entre os ambientes físico e virtual, mediante dispositivos interconectados que se comunicam entre si e com infraestruturas de processamento de dados, sejam locais ou em nuvem.

Neste contexto, inúmeros protocolos de redes *Wireless* foram desenvolvidos com o intuito de otimizarem a troca de dados entre os dispositivos, visando essencialmente a eficiência energética.

Com o desenvolvimento destes protocolos, emergiu o conceito de LPWAN *ou Low-power Wide Area Network* (Redes de Longa Distância e Baixo Consumo), que engloba tecnologias como LoRa/LoRaWAN e Sigfox. Estas soluções foram concebidas para possibilitar comunicações de longo alcance com consumo energético reduzido.

No cenário nacional, destaca-se a iniciativa da empresa brasileira HT Micron, que em 2019 apresentou o iMCP HT32SX, um Sistema em Pacote (SiP) desenvolvido especificamente para aplicações IoT. Este componente integra três elementos principais: um transceptor Sub-1 GHz, um amplificador de potência e um microcontrolador, todos projetados com foco na otimização do consumo energético.

A partir de 2020, com o início da produção em escala industrial, a empresa intensificou suas ações para estimular o desenvolvimento de aplicações práticas baseadas nesta tecnologia. A iniciativa visa fomentar a criação de conteúdo técnico e estabelecer uma rede colaborativa de desenvolvedores, contexto no qual se insere este trabalho acadêmico.

Perante o contexto atual, no qual a expectativa de vida de pessoas idosas tem aumentado consecutivamente, temos em contrapartida o aumento da taxa de mortalidade no respectivo grupo, devido a quedas e acidentes, principalmente acidentes domésticos. Desta forma, o projeto irá explorar a tecnologia *Sigfox*, desenvolvendo uma solução de *hardware* e *firmware* que contará com o uso do *SiP HT32SX aliado a diversos sensores*, para análise e monitoramento de pessoas idosas, focando no cuidado do indivíduo. Ao final, todo o material gerado por este projeto será compartilhado em repositório público no *GitHub*.

1.1 Motivação

Este projeto tem como foco principal a ampliação do conhecimento técnico-científico em uma área estratégica e emergente do setor de Tecnologia da Informação. A iniciativa permitirá o aprofundamento prático com o HT32SX, um dispositivo de última geração que se destaca no cenário tecnológico atual.

O crescente aumento da expectativa de vida da população idosa traz consigo desafios significativos relacionados à segurança e bem-estar deste grupo. As

estatísticas apontam para um preocupante aumento nas taxas de mortalidade devido a acidentes domésticos, especialmente quedas. Diante desta realidade, este trabalho propõe o desenvolvimento de uma solução tecnológica baseada na infraestrutura Sigfox, visando o monitoramento contínuo e a análise comportamental de idosos, com o objetivo de proporcionar maior segurança e autonomia para este público. É com este sentimento, que este trabalho de conclusão de curso busca, dar alguma contribuição ou inspiração para futuras ideias que tenham objetivos similares.

1.2 Objetivos

O objetivo deste trabalho de conclusão de curso é o desenvolvimento de uma solução tecnológica baseada na infraestrutura Sigfox, integrando hardware e firmware específicos para aplicação no monitoramento de idosos. O sistema utilizará o SiP HT32SX em conjunto com uma rede de sensores especializados, visando o monitoramento contínuo e a análise comportamental de idosos, com o objetivo de proporcionar maior segurança e autonomia para este público.

2. DESCRIÇÃO DAS TECNOLOGIAS

O projeto de TCC explorou a tecnologia *Sigfox* por meio do circuito integrado *iMCP HT32SX*, projetado pela HT Micron, juntamente com os sensores MAX30102, MLX90614, MMA8452Q e o módulo GPS GY-NEO6MV2. Esta sessão irá detalhar o funcionamento dessas tecnologias e seus respectivos dispositivos.

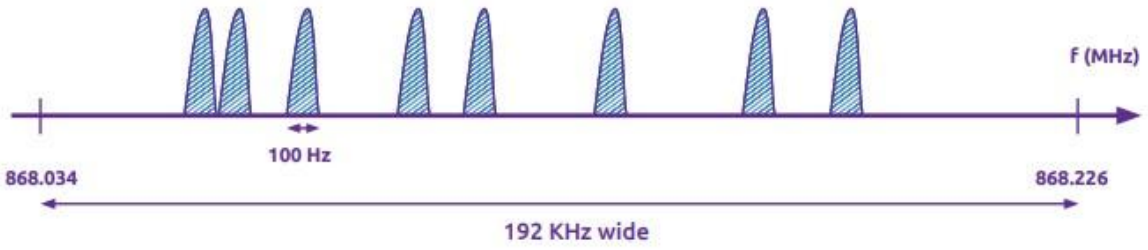
2.1 Sigfox

Sigfox é uma rede *LPWAN* global, com protocolo próprio, destinada a aplicações de Internet da Coisas. Sua infraestrutura é gerenciada por *Sigfox Operators (SO)* locais, responsáveis por fornecerem toda a estrutura necessária à conectividade, desde a instalação de *Base Stations (BS)*, a contratos que concedem o acesso à rede. No Brasil, a empresa responsável por esse gerenciamento é a WND Brasil.

2.1.1 *Ultra-Narrow Band (UNB)*

A rede *Sigfox* baseia-se na técnica de *Ultra-Narrow Band* para a transmissão de mensagens. A técnica permite o envio de *uplinks* através de sinais com largura de banda muito limitada: 100Hz nas regiões *ETSI* e *ARIB* (Europa, Japão), com taxa de transmissão de 100bps; e 600Hz na região *FCC* (Américas, Oceania), com taxa de transmissão de 600bps [1]. A Figura 2.1 ilustra a técnica de *Ultra-Narrow Band* usada nas regiões *ETSI* e *ARIB*:

Figura 2.1 – Tecnologia de *Ultra-Narrow Band* utilizada pela *Sigfox*.



Fonte: Modificado de *"Sigfox Technical Overview"* (2017, p.8).

A tecnologia *Ultra-Narrow Band* não suporta mensagens muito grandes e enviadas em um curto intervalo de tempo. Por esta razão, o protocolo *Sigfox* limita aos dispositivos o envio de 140 *uplinks* por dia, com pacotes de até 12 *bytes* (sem contar com *headers*), e o recebimento de 4 *downlinks* por dia, com pacotes de até 8 *bytes*. Esta particularidade faz com que os dispositivos apresentem um baixo consumo de energia, pois quanto maior é *payload*, maior é o sinal e mais demorada será a transmissão (uma mensagem de 12 bytes, a uma taxa de 100 bps, demora em torno de 2,08 segundos para ser transmitida [2]), assim como reduz o custo de conectividade em virtude do baixo tráfego de mensagens pela rede. A Figura 2.2 apresenta algumas outras características da tecnologia de *Ultra-Narrow Band*:

Figura 2.2 – Características da técnica de *UNB* usada pela *Sigfox*.

Fonte: (WND BRASIL, 2017, p.2).

2.1.2 Regiões *Sigfox* ou *RCZs (Radio Configuration Zones)*

A *Sigfox* divide a sua rede global em 7 zonas, cada uma com diferentes especificações de potência e frequência. O Brasil pertence à RC2, cuja frequência central de *uplink* está em 902.200MHz [3]. A Figura 2.3 apresenta a tabela de regiões *Sigfox* atualmente disponíveis:

Figura 2.3 – *Sigfox Radio Configuration (RC)*.

	RC1	RC2	RC3	RC4	RC5	RC6	RC7
Uplink center frequency (MHz)	868.130	902.200	923.200	920.800	923.300	865.200	868.800
Downlink center frequency (MHz)	869.525	905.200	922.200	922.300	922.300	866.300	869.100
Uplink data rate (bit/s)	100	600	100	600	100	100	100
Downlink data rate (bit/s)	600	600	600	600	600	600	600
Sigfox recommended EIRP (dBm)	16	24	16	24	14	16	16
Specifics	Duty cycle 1% *	Frequency hopping **	Listen Before Talk ***	Frequency hopping **	Listen Before Talk ***		Duty cycle 1% *

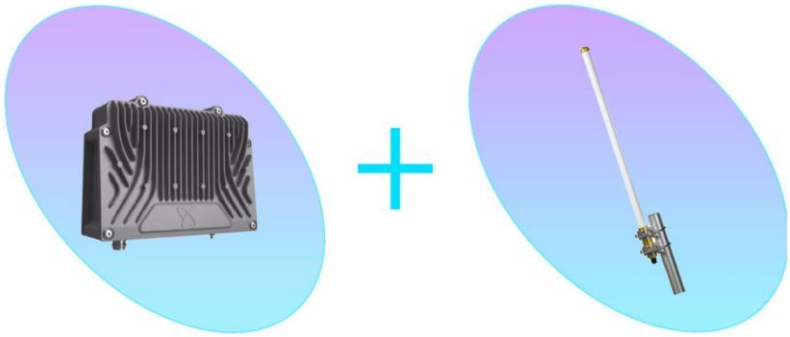
Fonte: Modificado de *Sigfox Build - "Radio Configuration"* (2024, p.2 e 3).

2.1.3 *Base Stations*

As *Base Stations* são antenas fixas responsáveis pelas principais operações da rede. São elas que recebem os sinais enviados pelos dispositivos, enviam pacotes de *downlink* e emitem sinais de *beacon* para o funcionamento da tecnologia *Monarch*. A estrutura de uma *BS* é

composta basicamente por três elementos: Uma antena; um *LNA (low-noise amplifier)*, para amplificar o sinal e filtrar ruído; e um *access point*, dispositivo capaz de decodificar os *payloads* e enviar para a nuvem [2]. A Figura 2.4 mostra o formato de uma *Base Station* da *Sigfox*:

Figura 2.4 – *Sigfox Base Station*.



Fonte: Modificado de *Sigfox Build - "What is Sigfox?"* (2024, p.4).

2.1.4 Tecnologia *Monarch*

O *Monarch* é uma *função* do protocolo Sigfox que fornece aos dispositivos a capacidade de detectarem a *RC Zone* da área em que estão operando. Essa informação permite com que o dispositivo se reconfigure e passe a transmitir as suas mensagens na frequência adequada ao país.

A detecção da região *Sigfox* acontece quando uma *Base Station* envia um *beacon over-the-air* e um dispositivo que está escaneando a área o detecta. O *beacon* é um sinal de rádio frequência que possui um formato específico para cada *RCZ*. É normalmente transmitido por *Base Stations* instaladas nas fronteiras e nas proximidades de aeroportos de países com cobertura *Sigfox*. Os dispositivos capazes de usarem esta *feature* conseguem diferenciar o formato de cada sinal, e a partir desta informação, conseguem se adaptar à frequência da região em que estão operando.

2.1.5 Acesso à Rede

O acesso à rede *Sigfox* é feito através de dispositivos específicos chamados de *Access Points*. Os *Access Points* são dispositivos que receberam o certificado de *Sigfox Verified* e estão aptos a transmitirem pacotes conforme as especificações determinadas pelo protocolo.

A Sigfox é um serviço pago. O desenvolvedor ou empresa que deseja utilizar a infraestrutura da rede deverá entrar em contador com o provedor local para negociar um contrato de conectividade.

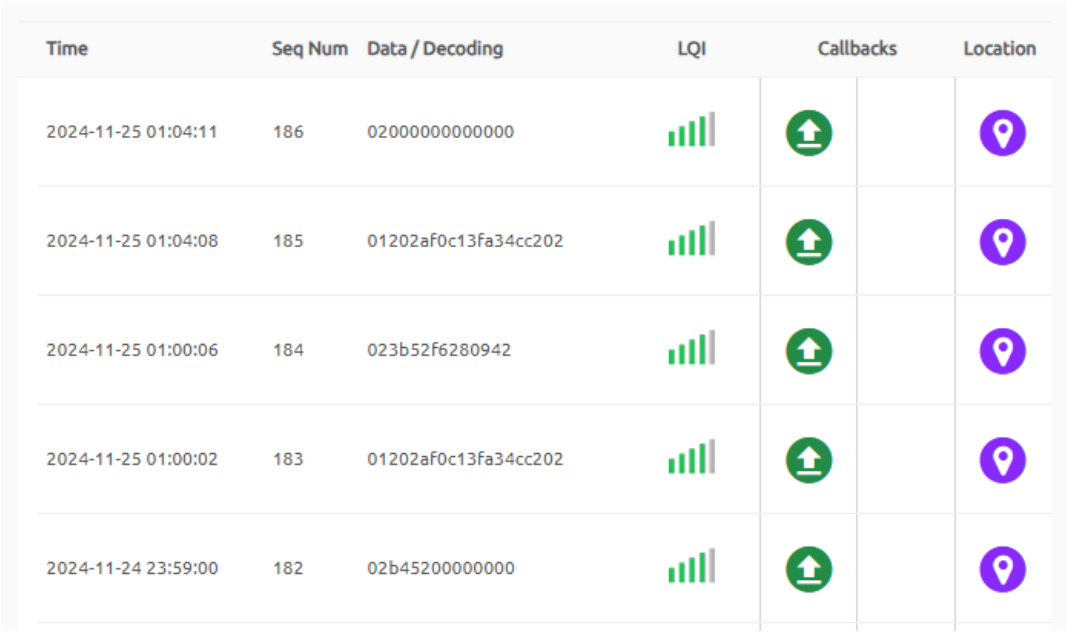
2.1.6 *Sigfox Back-end*

O *Sigfox Back-end* é um serviço na nuvem disponibilizado pela própria *Sigfox*. A ferramenta oferece a possibilidade de o usuário validar as mensagens enviadas pelos seus dispositivos, criar e configurar *callbacks*, verificar o *RSSI (Received Signal Strength Indicator)* do sinal

recebido pela *Base Station* e obter uma estimativa da localização do dispositivo no momento do envio da mensagem.

A imagem da Figura 2.5 mostra como as mensagens são exibidas no *Back-end* da *Sigfox*.

Figura 2.5 – Mensagens recebidas no *Back-end* da *Sigfox*.



Time	Seq Num	Data / Decoding	LQI	Callbacks	Location
2024-11-25 01:04:11	186	02000000000000			
2024-11-25 01:04:08	185	01202af0c13fa34cc202			
2024-11-25 01:00:06	184	023b52f6280942			
2024-11-25 01:00:02	183	01202af0c13fa34cc202			
2024-11-24 23:59:00	182	02b45200000000			

Fonte: Própria (2024).

2.1.7 Credenciais *Sigfox*

O *Sigfox Back-end* identifica os dispositivos na rede através de credenciais que são enviadas nos cabeçalhos dos pacotes. Essas credenciais são compostas por identificadores únicos (*ID* e *PAC*), geralmente gravados na *NVM (Non-volatile memory)* dos dispositivos.

2.1.8 Rastreamento

A rede *Sigfox* oferece uma *feature* nativa de rastreamento chamada de *Sigfox Atlas*. A tecnologia fornece a localização aproximada dos dispositivos que se comunicam com a rede, por meio de cálculos probabilísticos utilizando o *RSSI* dos *frames* recebidos pelas *Base Stations*. Isto significa que quanto melhor for a cobertura da rede no local, ou seja, quanto mais *Base Stations* receberem as mensagens, melhor será a precisão da localização [4].

Algumas vantagens e limitações da tecnologia são listadas abaixo. As informações incluídas a seguir foram retiradas do *Sigfox Build* [4]:

- Vantagens da tecnologia:

    1. Não necessita de *hardware* extra: A *feature* é toda implementada na nuvem. Não utiliza *GPS.*
    2. Baixo Consumo: O consumo de energia é o mesmo de uma mensagem comum enviada a rede.

3. Não interfere no *payload*: Diferentemente do *GPS*, não há necessidade de incluir valores de latitude/longitude no *payload*.

- Limitações da tecnologia:

1. O cálculo probabilístico da localização fornece uma precisão de 1 a 10 km para mais de 80% das mensagens recebidas. Conforme dito anteriormente, essa aproximação é diretamente proporcional à cobertura *Sigfox* no ambiente em que o dispositivo está operando.
2. O serviço só pode ser utilizado enquanto o dispositivo estiver no alcance de alguma *Base station*.

2.1.9 **Mensagens: Formato e Identificação**

As mensagens Sigfox são compostas por pacotes curtos de dados, com tamanho de 12 bytes de payload máximo (ou 96 bits). Cada mensagem enviada é replicada em 3 frames idênticos, enviados em frequências pseudo-aleatórias (frequência hopping) e em tempos ligeiramente diferentes, conforme especificado na documentação oficial da Sigfox. Ou seja, quando um dispositivo Sigfox precisa enviar uma mensagem para a rede, ele transmite a mesma informação três vezes, como podemos ver no Espectro de Frequência da figura 2.6. Cada uma dessas transmissões acontece em momentos diferentes e usa frequências aleatórias distintas. É uma estratégia esperta - mesmo que duas das transmissões sejam perdidas por interferência ou outros problemas, se pelo menos uma delas chegar às antenas da rede, a mensagem será entregue com sucesso à nuvem.

Figura 2.6 – *Frame Sigfox* (salto em frequências aleatórias).

Espectro de frequência: A figura apresentada mostra o espectro de frequência utilizado para a transmissão de uma mensagem Sigfox.

No gráfico:

- O **eixo X** representa a **frequência** (em MHz).
- O **eixo Y** representa a **potência do sinal** (em dBm).

Os picos observados no gráfico correspondem às transmissões das réplicas da mensagem em diferentes frequências. Esse comportamento é parte do protocolo Sigfox para evitar interferências e aumentar a confiabilidade da transmissão.

A replicação em 3 frames distintos no envio das mensagens aumenta a confiabilidade da comunicação, pois, em ambientes ruidosos, aumenta a probabilidade de pelo menos um frame ser recebido com sucesso, em virtude de reduzir o impacto de interferências temporárias na transmissão.

Identificação da Mensagem

Cada mensagem Sigfox possui identificadores únicos associados:

1. **ID do dispositivo (Device ID):** Um identificador único que identifica o remetente (dispositivo) que enviou a mensagem.

2. **Sequência de autenticação:** Um contador ou número de sequência embutido no header/payload da mensagem, que ajuda a diferenciar mensagens distintas enviadas pelo mesmo dispositivo.

Esses elementos são utilizados para identificar e processar corretamente as mensagens na rede.

2.1.10. Risco de Duplicação de Mensagens e Tratamento

O envio de 3 frames idênticos em frequências e tempos distintos pode, eventualmente, resultar na recepção de mais de um frame pela infraestrutura Sigfox (e.g., por repetidores ou diferentes estações base da rede). Para lidar com isso, o backend da Sigfox implementa mecanismos específicos para tratamento:

1. Filtragem por ID e número sequencial:

    - O backend Sigfox considera apenas a primeira cópia válida do frame recebido.

    - Réplicas subsequentes de um frame com o mesmo ID de dispositivo e número de sequência são descartadas.

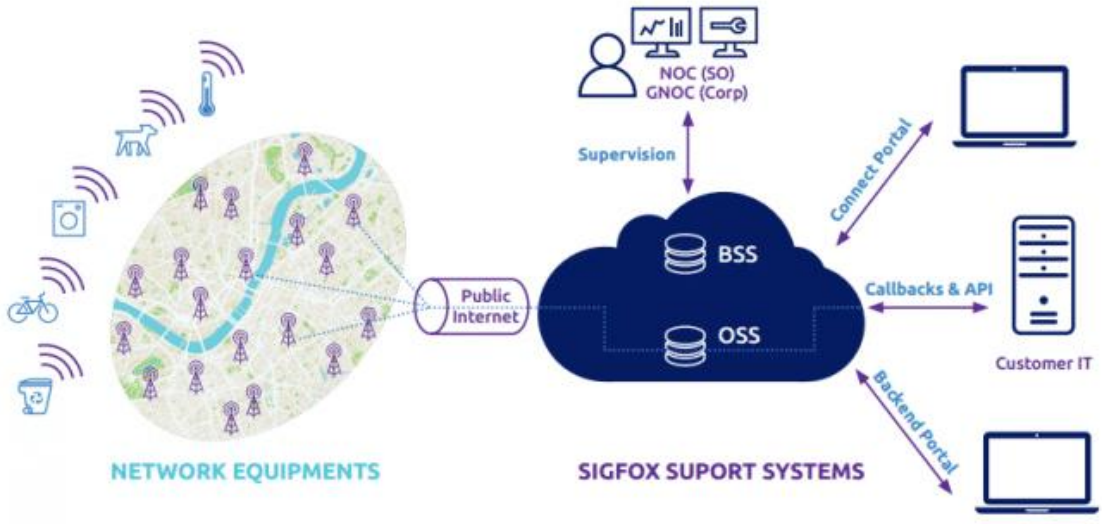
2. Detecção de duplicatas:

    - Se dois ou mais frames idênticos forem recebidos por diferentes estações base, o sistema compara o carimbo de tempo (timestamp) e o ID da mensagem para identificar duplicatas e descartar as extras.

Esses mecanismos garantem que, mesmo que múltiplos frames sejam recebidos, apenas uma instância da mensagem é entregue ao aplicativo na nuvem.

A partir do momento que uma mensagem é transmitida à rede, o caminho percorrido pelos dados, do *access point* até o *front-end* do usuário, é sempre o mesmo. A Figura 2.7 abaixo mostra um esquema do caminho percorrido pelos dados de uma mensagem enviada à rede *Sigfox*:

Figura 2.7 – Caminho percorrido pelos dados de uma mensagem enviada à rede *Sigfox*.



Fonte: Modificado de *Sigfox Build* (2024, p.3).

1. As *Base Stations* detectam os *frames* e encaminham para o *Sigfox Back-end*.
2. O *Sigfox Back-end* decodifica a mensagem, identifica o *ID* do dispositivo e encaminha, por intermédio de *callbacks* configuráveis, para a plataforma de *back-end* do usuário.
3. O *back-end* do usuário realiza um *parse* dos dados e finalmente encaminha para o *front-end*.

2.2 TagoIO

A TagoIO é uma plataforma desenvolvida para fornecer soluções de *front-end* a aplicações de Internet das Coisas. A ferramenta fornece a opção de criar suas próprias *dashboards*, utilizando diversos tipos de *widgets* disponibilizados na plataforma, construindo uma interface completa e aprimorada para visualização dos dados oriundos dos sistemas e dispositivos integrados a plataforma.

As principais vantagens da plataforma são a sua interface gráfica amigável e a ampla disponibilidade de tutoriais, que facilitam a integração com redes como a Sigfox e reduzem a curva de aprendizado. Além disso, oferece suporte a diferentes aplicações IoT, ferramentas de análise de dados e automação, dashboards personalizáveis, APIs robustas e conectores integrados. Sua escalabilidade e o suporte técnico, aliado a uma comunidade ativa, tornam a TagoIO uma solução completa e eficiente para projetos IoT de diferentes portes.

As principais desvantagens da plataforma incluem as limitações de recursos na versão gratuita, como restrições de armazenamento e número de dispositivos conectados; a dependência total de conectividade com a nuvem, o que pode impactar o acesso aos dados em

caso de interrupções; e a necessidade de conhecimentos técnicos em programação para personalização avançada, o que pode dificultar o uso por usuários menos experientes. Além disso, a plataforma possui foco em dispositivos IoT específicos, o que pode limitar sua flexibilidade para outros protocolos, e pode apresentar pequenas latências em aplicações que exigem respostas em tempo real. Esses fatores devem ser avaliados de acordo com os requisitos do projeto.

A **TagoIO** é uma plataforma robusta e amigável, mas as limitações de recursos na versão gratuita, a dependência de conectividade com a nuvem e a necessidade de conhecimentos técnicos em casos avançados podem ser desvantagens a serem consideradas, dependendo dos requisitos do projeto.

2.2.1 *Parse* dos Dados

A TagoIO oferece uma opção especifica dentro da plataforma para que o usuário implemente o código que fará o *parse* dos dados brutos. A função do *parse* desenvolvido é desserializar e interpretar os dados brutos que chegam na plataforma, criando as variáveis necessárias que poderão ser utilizadas em todas as funcionalidades da plataforma, como na criação de um *dashboard*.

A Figura 2.9 abaixo, apresenta o trecho principal do código criado na plataforma para realizar o parse dos dados brutos.

Figura 2.9 – Payload *parser* implementado para a plataforma da TagoIO.

```
20
21      dadosPayload.push({variable: 'tipo_evento', value: buffer.readInt8(0)});
22      dadosPayload.push({variable: 'latitude', value: buffer.readFloatLE(1), unit: '°'});
23      dadosPayload.push({variable: 'longitude', value: buffer.readFloatLE(5), unit: '°'});
24      dadosPayload.push({variable: 'posicao_usuario', value: buffer.readInt8(9)});
25      dadosPayload.push({variable: 'localizacao', value: "Localizacao atual", "location": { "lat": latitude, "lng": longitude}});
26
27      if(buffer.readInt8(9) === 2){
28        posicaoUsuario = "Individuo caiu.";
29        dadosPayload.push({ variable: 'posicao_usuario_status', value: posicaoUsuario });
30      }
31    }
32    else if(buffer.readInt8(0) === 0x02){   // payload_saude
33      dadosPayload.push({variable: 'tipo_evento', value: buffer.readInt8(0)});
34      dadosPayload.push({variable: 'frequencia_cardiaca', value: buffer.readUInt8(1), unit: 'BPM'});
35      dadosPayload.push({variable: 'spo2', value: buffer.readUInt8(2), unit: '%'});
36      dadosPayload.push({variable: 'temperatura_corporal', value: buffer.readFloatLE(3), unit: '°C'});
37    }
38
```

Fonte: Própria (2024)

A aplicação receberá diferentes payloads, 1 contendo os dados de Localização do usuário e outro contendo os dados de Saúde do usuário. A plataforma realizará o parse dos dados brutos e após a execução do parse, o sistema criará as oito variáveis - *“tipo_evento”, “latitude”, “longitude”, “localizacao”, “posicao_usuario”, “frequencia_cardiaca”, “spo2” e “temperatura_corporal”* em seu banco de dados, para que recebam e armazenem as informações interpretadas pelo *parser*. As variáveis declaradas acima, referem-se as variáveis definidas no código do firmware da aplicação, as quais armazenam os dados lidos e processados dos sensores utilizados.

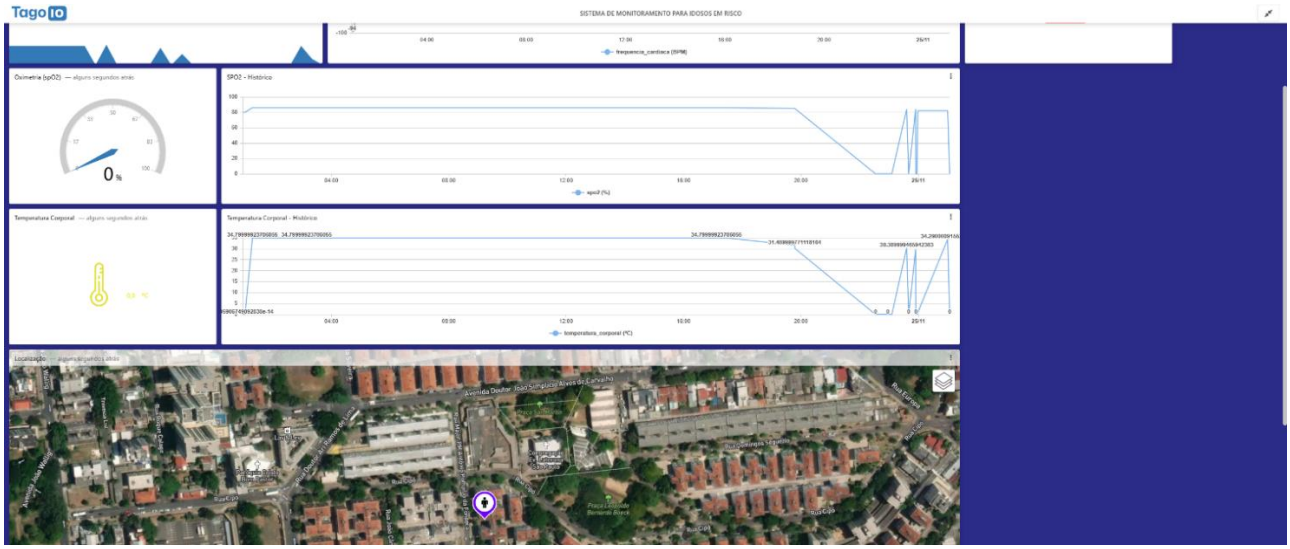
2.2.2 Dashboards

As *dashboards* são ferramentas visuais de gerenciamento de informações, frequentemente utilizadas em sistemas de rastreio e monitoramento. A TagoIO disponibiliza uma área em sua plataforma destinada à montagem e edição de *dashboards*, as quais podem receber dados de diversos dispositivos simultaneamente.

Em um sistema que utiliza o protocolo e a infraestrutura da rede *Sigfox*, as *dashboards* desenvolvidas poderão apenas receber e nunca transmitir dados para os rádios utilizados para a acesso a rede. Esta limitação é definida pelo próprio protocolo *Sigfox*, que permite o envio de dados de uma *Base Station* a um dispositivo apenas quando este mesmo dispositivo faz a requisição de um *downlink*.

A Figura 2.10 abaixo, apresenta o dashboard criado na plataforma, para apresentação dos dados lidos dos sensores e enviados para a nuvem Sigfox. Na figura serão apresentados os dados de Batimentos Cardíacos, Oximetria e Temperatura Corporal (a ESQUERDA serão apresentados os últimos valores recebidos e a DIREITA, um gráfico com o histórico dos valores recebidos e armazenados no banco de dados da plataforma), além de ser exibido um mapa com a localização do local de onde as mensagens foram enviadas.

Figura 2.10 – Dashboard criado na plataforma da TagoIO.

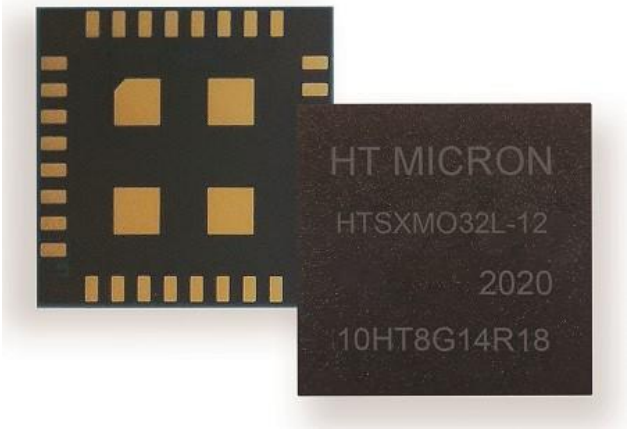


Fonte: Própria (2024)

2.3 iMCP HT32SX

O *iMCP HT32SX* (Figura 2.11) é um *System-in-Package* produzido pela HT Micron para soluções de Internet das Coisas. O sistema se trata de um *MCO (Multicomponent Integrated Circuit)*, o qual possui um microcontrolador Cortex-M0, um rádio *Sub-1GHz* e um *amplificador* no mesmo encapsulamento.

Figura 2.11 – *iMCP HT32SX* desenvolvido pela HT Micron.



Fonte: Modificado do repositório da HT Micron no *GitHub* (2024)

2.3.1 *Access Point* à rede *Sigfox*

Em 2019, o *HT32SX* recebeu o certificado de *Sigfox Verified*, que o habilitou a operar como *access point* à rede *Sigfox*. A certificação capacita o dispositivo a trabalhar em todas as 7 zonas *Sigfox* atualmente disponíveis, assim como permite utilizar a *feature* do *Monarch* em suas aplicações.

2.3.2 Outros Protocolos de Rede

Apesar da certificação recebida pela *Sigfox*, o *HT32SX* possui um rádio genérico que o permite trabalhar com outros protocolos de rede sem fio. Atualmente, a HT Micron fornece um exemplo de protocolo *Peer-to-peer (P2P)* em seu repositório público no *GitHub* [5]. A partir desse exemplo, o usuário pode desenvolver protocolos proprietários, adaptar o exemplo a protocolos conhecidos como o *Wireless M-Bus* e até mesmo criar uma rede própria de dispositivos, como uma Rede *Mesh*.

Recentemente a HT Micron lançou uma nova aplicação que combina uma *stack* de comunicação *Peer-to-peer*, com a *stack* que implementa o protocolo *Sigfox*. A aplicação realiza uma extensão da rede *Sigfox* através de uma rede própria de *HT32SX*, repetindo o *payload* de um determinado dispositivo até encontrar o *gateway*, para então transmitir os dados à rede.

2.3.3 Especificações Técnicas

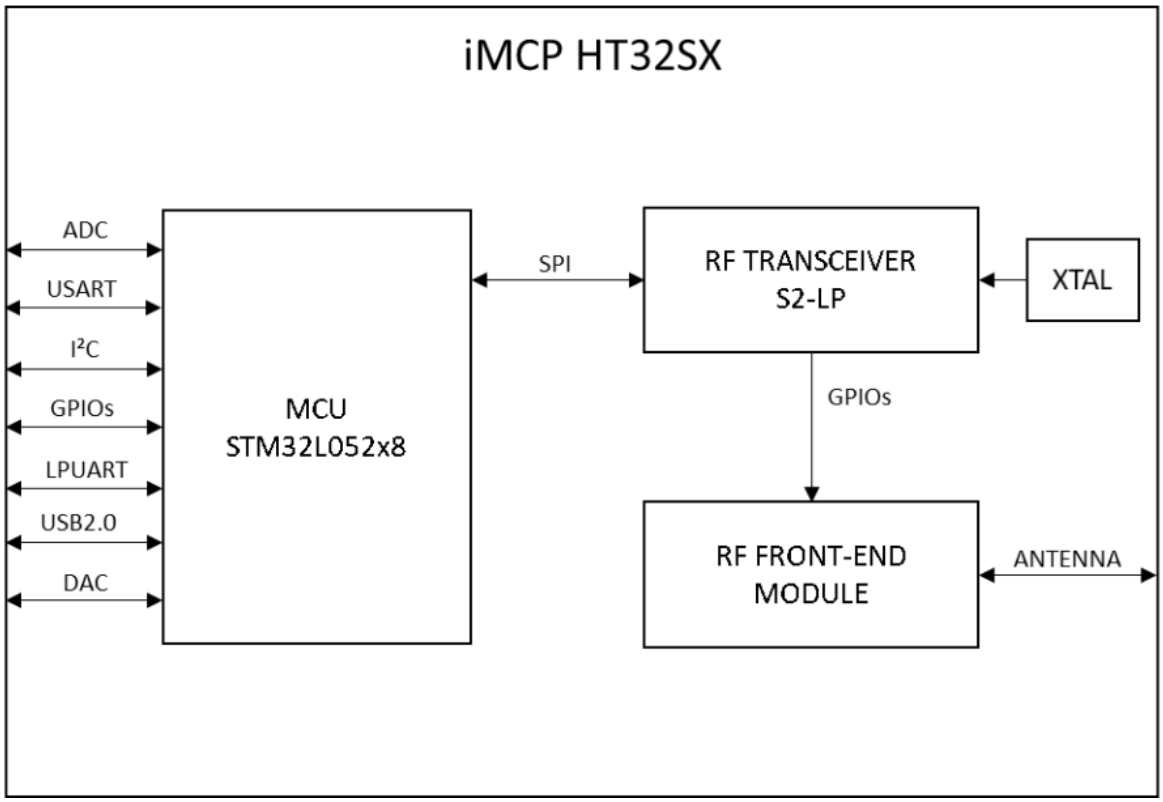
As especificações técnicas do *iMCP HT32SX,* necessárias para a compreensão deste trabalho são apresentadas a seguir. Todas as informações foram retiradas do *datasheet* oficial do dispositivo [6]:

- *Part Number:* HTSXMO32L-22;
- Microcontrolador: *ARM Cortex M0+ 32bits STM32L052T8Yx*. *64 kB* de memória *flash* e *8 kB* de memória *RAM*; • Rádio *S2-LP Sub-1GHz*;

- Bandas de Frequência:
  - 413-479 MHz.
  - 452-527 MHz.
  - 826-958 MHz.
  - 904-1055 MHz.
- *Front-End Module*: *SKY66420-11*;
- Potência de máxima saída: +26 dBm;
- *RSSI*: -128 dBm;
- Consumo em *Deep Sleep*: De 3 a 4 μA (depende do periférico escolhido para despertar).

A Figura 2.12 mostra um diagrama de blocos do *iMCP HT32SX*.

Figura 2.12 – Diagrama de blocos do *HT32SX*.



Fonte: Modificado do *datasheet* oficial do dispositivo (2024)

2.4 Sensor MAX30102

O MAX30102 é um módulo integrado de oximetria de pulso e monitor de frequência cardíaca. Ele inclui LEDs internos, fotodetectores, elementos ópticos e eletrônicos de baixo ruído com rejeição de luz ambiente. O MAX30102 fornece uma solução de sistema completa para facilitar o processo de design para dispositivos móveis e vestíveis.

As especificações técnicas do *MAX30102* necessárias para melhor compreensão deste projeto são apresentadas a seguir. Todas as informações foram retiradas do *datasheet* oficial do dispositivo:

Características Gerais:

- Módulo integrado que combina oxímetro de pulso e monitor de frequência cardíaca
- Dimensões: 5,6mm x 3,3mm x 1,55mm (módulo com 14 pinos)
- Interface de comunicação I2C
- Vidro de cobertura integrado

Especificações Elétricas:

- Tensão de alimentação (VDD): 1,7V a 2,0V
- Tensão de alimentação LED (VLED+): 3,1V a 5,0V
- Corrente em modo desligado: 0,7µA (típico)
- Consumo de energia em modo monitor cardíaco: < 1mW

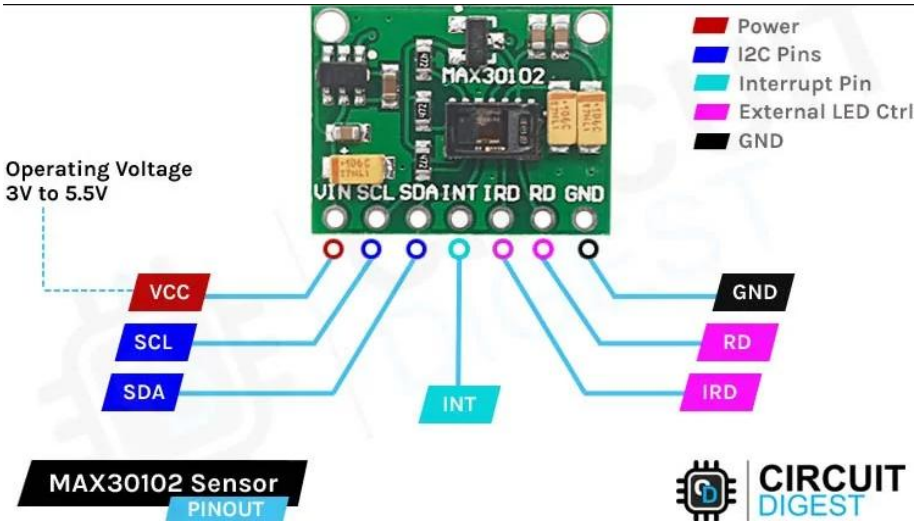
Especificações Ópticas:

- LED Vermelho: comprimento de onda 660-670nm
- LED Infravermelho: comprimento de onda 870-900nm
- ADC de 18 bits de resolução
- Alta rejeição à luz ambiente

Características Operacionais:

- Temperatura de operação: -40°C a +85°C
- Taxa de amostragem programável
- Corrente de LED ajustável
- FIFO integrado para armazenamento de dados
- Saída rápida de dados

Figura 2.12 – *Módulo Sensor de Frequência Cardíaca e Oxigênio*.



Fonte: Saravati (2024).

2.5 Sensor MLX90614

O MLX90614 é um termômetro infravermelho para medições de temperatura sem contato. Em virtude do amplificador de baixo ruído, ADC de 17 bits e a unidade DSP poderosa que possui, consegue alcançar uma alta precisão e resolução do termômetro.

As especificações técnicas do *MLX90614* necessárias para melhor compreensão deste projeto são apresentadas a seguir. Todas as informações foram retiradas do *datasheet* oficial do dispositivo:

Características Principais:

- Termômetro infravermelho para medição de temperatura sem contato
- Disponível em versões de zona única e dupla
- Interface digital compatível com SMBus e saída PWM personalizável
- Disponível em versões de 3V e 5V
- Grau automotivo

Faixas de Medição:

- Temperatura do sensor (ambiente): -40°C a 125°C
- Temperatura do objeto: -70°C a 380°C

Precisão e Resolução:

- Alta precisão de 0,5°C em faixa ampla (0°C a 50°C para temperatura ambiente e do objeto)
- Resolução de medição de 0,02°C
- ADC de 17 bits

Especificações Elétricas:

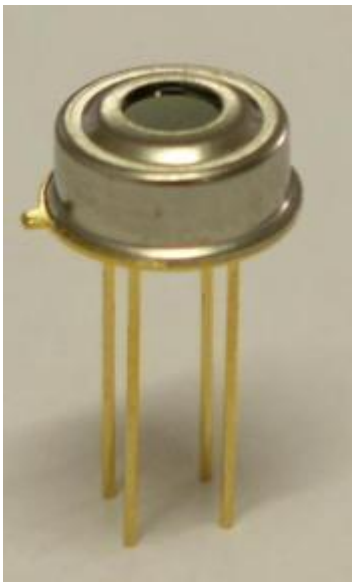
- Tensão de alimentação:
    - Versão A: 4,5V a 5,5V
    - Versão B: 2,4V a 3,6V
- Consumo de corrente típico: 1mA
- Modo de economia de energia disponível (apenas versão 3V)

Interfaces de Comunicação:
- SMBus compatível (protocolo 2-wire)
- PWM com resolução de 10 bits
- Período PWM padrão de 1,024ms
- Endereço SMBus padrão: 5Ah

Características Físicas:
- Encapsulamento TO-39
- Filtro óptico integrado para imunidade à luz solar
- Emissividade padrão calibrada para 1,0 (ajustável de 0,1-1,0)

Condições Operacionais:
- Temperatura de operação: -40°C a +125°C (versão A) / -40°C a +85°C (versão B)
- Temperatura de armazenamento: -40°C a +125°C (versão A) / -40°C a +105°C (versão B)
- Proteção ESD: 2kV

Figura 2.13 – Módulo Sensor de Temperatura Infravermelho.



Fonte: Retirado do datasheet do sensor. (2024).

2.6 Sensor MMA8452Q

O MMA8452Q é um acelerômetro inteligente, de baixa potência, tridimensional, capacitivo e micromaquinado, com resolução de 12 bits. Este acelerômetro vem equipado com funções integradas e opções programáveis flexíveis pelo usuário, configuráveis para dois pinos de interrupção. As funções de interrupção integradas permitem uma economia geral de energia, aliviando o processador principal da necessidade de consultar continuamente os dados.

As especificações técnicas do MMA8452Q necessárias para melhor compreensão deste projeto são apresentadas a seguir. Todas as informações foram retiradas do *datasheet* oficial do dispositivo:

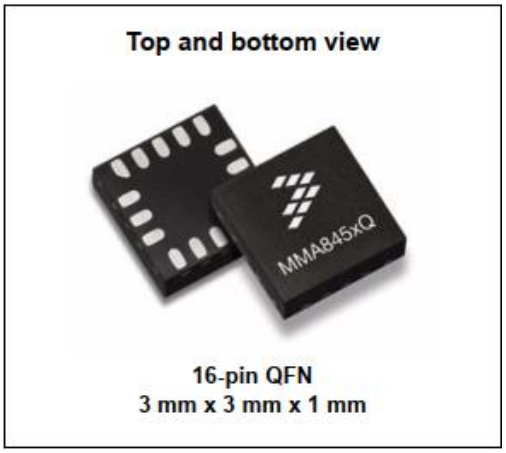
Características Principais:

- Acelerômetro digital de 3 eixos com resolução de 12/8 bits
- Faixa de medição selecionável: ±2g, ±4g ou ±8g
- Interface digital I2C
- Tensão de alimentação: 1.95V a 3.6V
- Tensão de interface: 1.6V a 3.6V
- Taxa de amostragem: 1.56 Hz até 800 Hz
- Ruído: 99 µg/√Hz
- Consumo: 6 µA a 165 µA
- Temperatura de operação: -40°C a +85°C
- Encapsulamento QFN de 16 pinos (3mm x 3mm x 1mm)

Funcionalidades:

- Detecção de orientação (retrato/paisagem)
- Detecção de queda livre e movimento
- Detecção de pulsos (single/double tap)
- Detecção de transientes
- 2 pinos de interrupção programáveis
- Filtro passa-alta em tempo real
- Auto-wake e auto-sleep para economia de energia
- Self-test integrado

Figura 2.14 – *Módulo Acelerômetro*.



Fonte: Retirado do datasheet do sensor. (2024).

2.7 Módulo GPS GY-NEO6MV2

A série de módulos NEO-6 é uma família de receptores GPS autônomos com o mecanismo de posicionamento u-blox 6 de alto desempenho. O design e a tecnologia inovadores suprimem fontes de interferência e atenuam os efeitos de multicaminho, dando aos receptores GPS NEO-6 excelente desempenho de navegação, mesmo nos ambientes mais desafiadores.

As especificações técnicas do GY-NEO6MV2 necessárias para melhor compreensão deste projeto são apresentadas a seguir. Todas as informações foram retiradas do *datasheet* oficial do dispositivo:

Características Principais:

- **Tipo de Receptor:** GPS, com suporte a SBAS (WAAS, EGNOS, MSAS).
- **Canais:** 50 canais.
- **Frequência:** GPS L1, código C/A.
- **Sensibilidade:**
  - Aquisição: -147 dBm
  - Rastreamento e Navegação: -161 dBm
  - Reaquisição: -156 dBm
- **Tempo para Primeira Correção (TTFF):**
  - Partida a Frio: 27 segundos
  - Partida a Quente: 1 segundo
  - Partida Morna: 27 segundos
- **Precisão da Posição Horizontal:** 2,5 m RMS, 2,0 m 99%.
- **Precisão da Velocidade:** 0,1 m/s.
- **Precisão da Direção:** 0,5 graus.
- **Taxa Máxima de Atualização de Navegação:** 5 Hz.
- **Protocolos:** NMEA, UBX, RTCM (entrada).
- **Interfaces:** UART, USB, SPI, DDC (I2C compatível).
- **Tensão de Alimentação:** 2.7 - 3.6 V.
- **Consumo de Energia (Indicativo):** Depende do modo de operação (Máxima Performance, Eco, Power Save). O datasheet não especifica valores numéricos para cada modo.
- **Antena:** Suporta antenas passivas e ativas.
- **Recursos Adicionais:** AssistNow Online e Offline A-GPS, Automotive Dead Reckoning (ADR).
- **Dimensões:** 16.0 x 12.2 x 2.4 mm.

Figura 2.15 – Módulo GPS.

Fonte: RoboCore (2024).

3. DESENVOLVIMENTO DO PROJETO

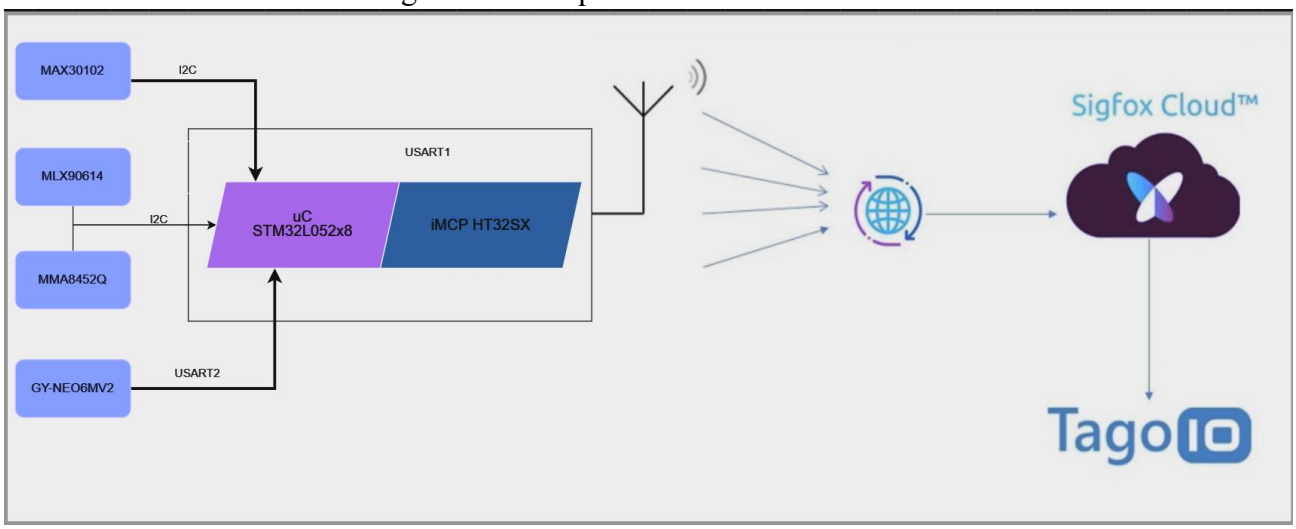
Este capítulo descreve o desenvolvimento do projeto de conclusão de curso, desde a arquitetura do sistema até a implementação do *front-end* na plataforma da TagoIO. O projeto trata-se de uma solução de *hardware* e *firmware* para aplicação no monitoramento de pessoas idosas em risco. O objetivo, conforme já mencionado no capítulo 1, visa o monitoramento contínuo e a análise comportamental de idosos, com o objetivo de proporcionar maior segurança e autonomia para este público.

3.1 Arquitetura do Sistema

A arquitetura do sistema foi organizada em um modelo Mestre-Escravo, onde o centro de todo o processamento (ou seja, o mestre) é o microcontrolador *Cortex-M0 STM32L052x8*. É ele quem faz as leituras de todos os sensores utilizados, realizada o processamento dos valores lidos, grava nos payloads Sigfox e decide quando o *HT32SX* deve enviar as mensagens à rede *Sigfox*.

Ao trabalhar com o módulo SiP iMCP HTSXMO32L-22, é disponibilizada na mesma placa o microcontrolador Cortex-M0 e o módulo HT32SX, logo, a comunicação entre eles é direta, bastando ativar o rádio Sigfox, configurar a zona a ser utilizada e então enviar as instruções/payloads via interface *USART* (*Universal Synchronous Asynchronous Receiver Transmitter*).

Figura 3.1 – Arquitetura do Sistema.



Fonte: Própria (2024).

A Figura 3.1 mostra o diagrama de blocos do sistema. O desenvolvimento foi dividido em quatro etapas:

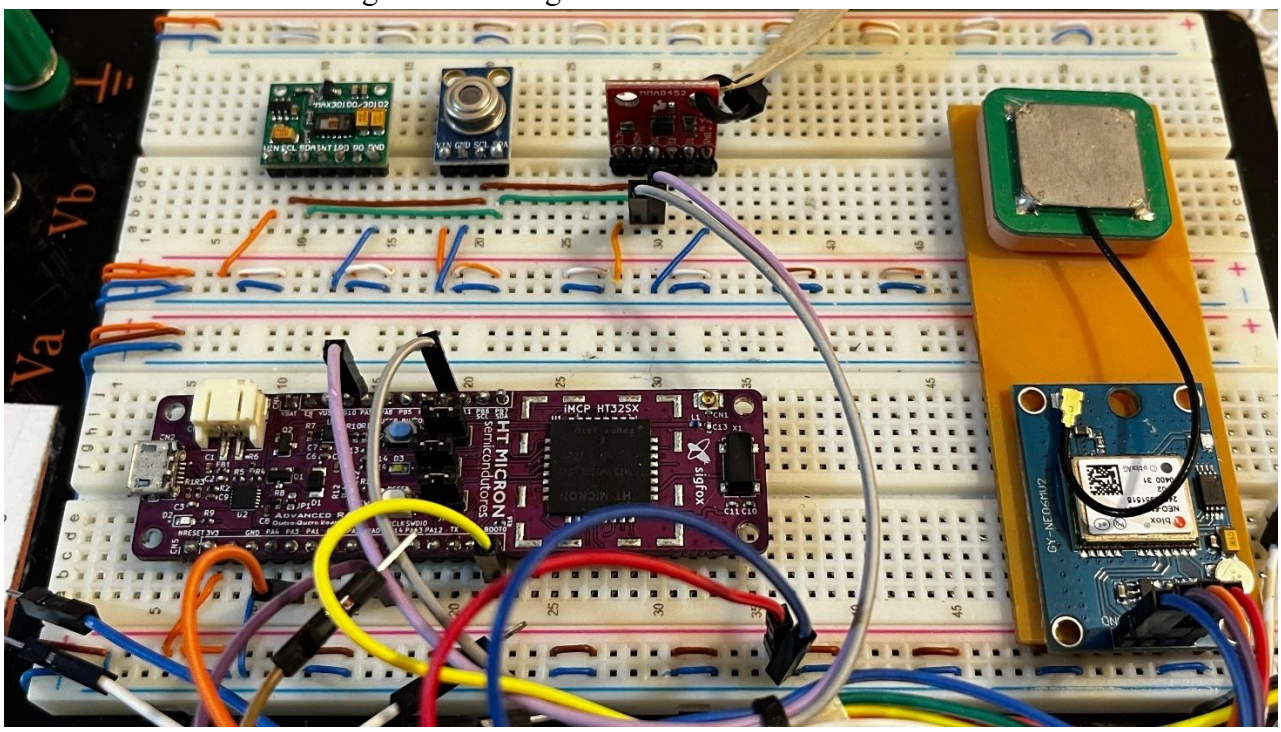
1. Projeto do *PoC*: Análise da viabilidade do projeto e de suas principais funcionalidades;
2. Projeto de *Firmware*: Adaptação do exemplo Generic-Push-Button do *HT32SX* e desenvolvimento do *firmware* do *microcontrolador para controle dos sensores utilizados*;
3. Configuração do *Sigfox Backend*: Registro do dispositivo na plataforma da *Sigfox* na nuvem e configurações de *callback*;
4. Desenvolvimento do *Front-End* na plataforma TagoIO: Registro do dispositivo na plataforma da TagoIO para integração das plataformas, configurações da conta, desenvolvimento do código responsável pelo *parse do payload* dos dados recebidos do backend Sigfox, criação da *dashboard utilizando os dados recebidos* e configuração de alertas a serem enviados a usuários, de acordo com os valores recebidos.

3.2 Projeto do PoC

Inicialmente, com o intuito de validar as principais conexões da *PCB*, foi desenvolvida uma *POC (Proof of Concept, em português* ***Prova de Conceito****, refere-se ao processo utilizado para demonstrar a viabilidade da ideia, das tecnologias, do método e da solução, desenvolvidos no trabalho)* a nível de *protoboard, com todos os sensores e módulos necessários*. O *POC* foi desenvolvido utilizando o *módulo SiP iMCP HTSXMO32L-22* de desenvolvimento do *HT32SX, a qual possui o microcontrolador Cortex-M0 STM32L052x8 embutido, juntamente com o módulo Sigfox*.

Em função do espaço de tempo disponível para desenvolvimento do trabalho, foi realizada apenas a montagem do *POC* e as funcionalidades implementadas foram validadas diretamente através do POC. A figura 3.2 abaixo, apresenta os detalhes da montagem do POC.

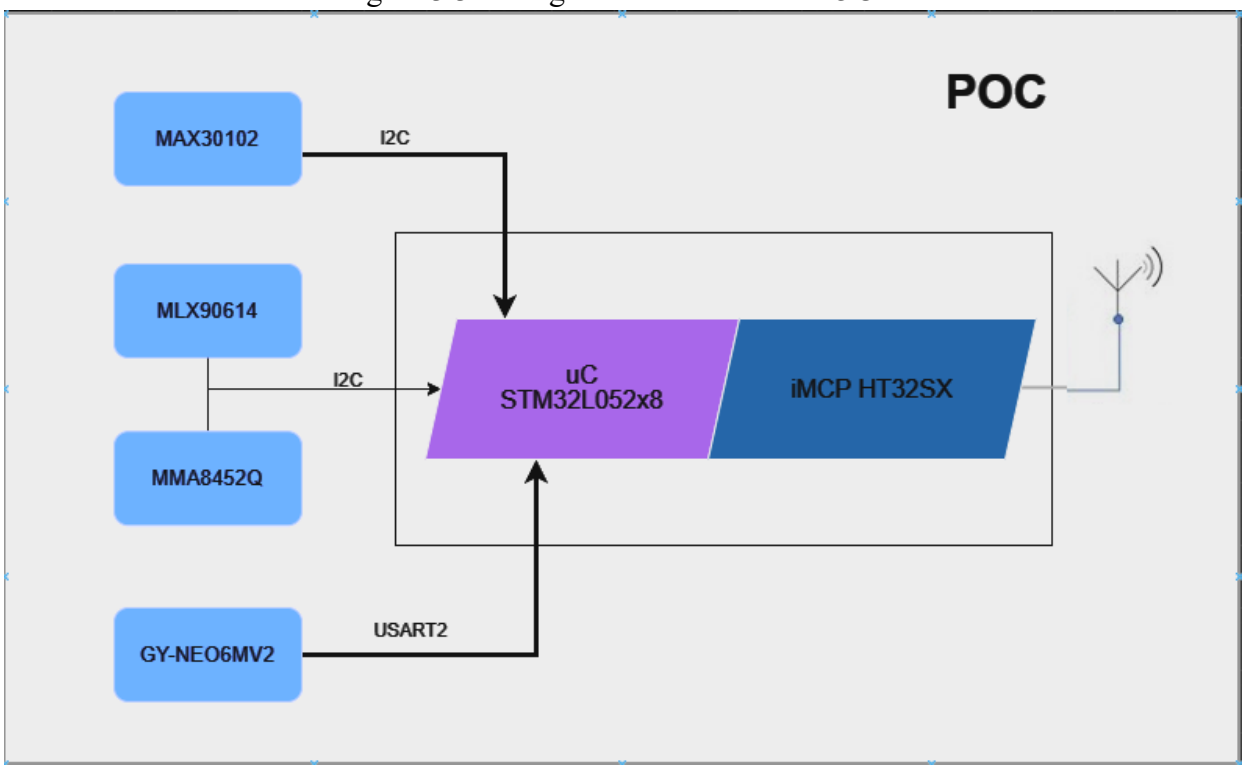
Figura 3.2 – Diagrama de Blocos do *POC*.



Fonte: Própria (2024).

O diagrama de blocos do *POC* é apresentado na Figura 3.3:

Figura 3.3 – Diagrama de Blocos do *POC*

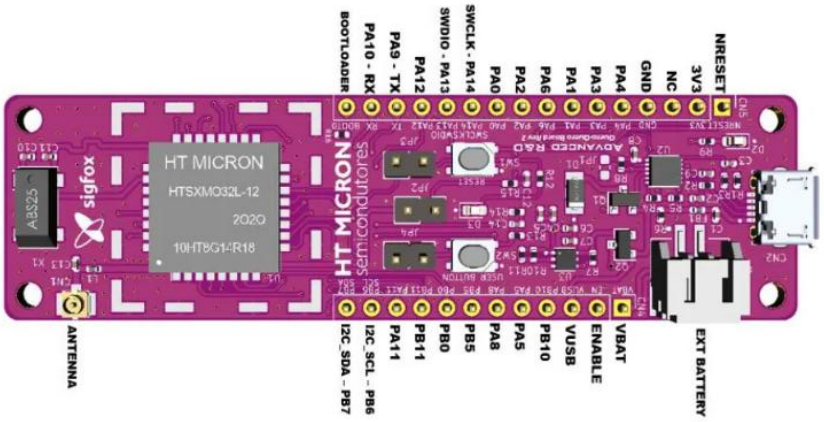


Fonte: Própria (2024).

A lista dos principais itens empregados no projeto do *POC* é exibida a seguir:

1. Placa módulo SiP iMCP HTSXMO32L-22 (Figura 3.4): Usado para integração com todos os sensores utilizados, processamento dos dados e envio dos payloads via serial.

Figura 3.4 – Placa módulo SiP iMCP HTSXMO32L-22.



Fonte: Guia de instalação da placa de avaliação do módulo SIP HT Micron IMCP HTSXMO32L-22 (2024).

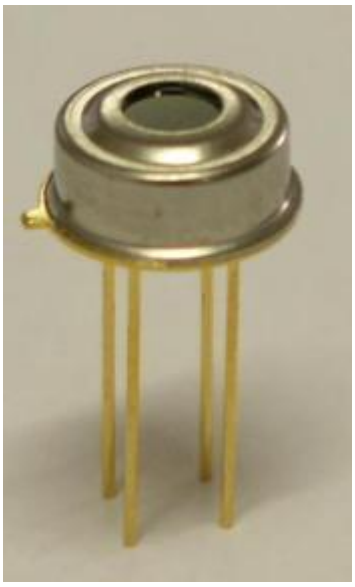
2. Sensor MAX30102 (Figura 3.5): Utilizado para leitura de oximetria (SPO2) e frequência cardíaca do usuário.

Figura 3.5 – *Módulo Sensor de Frequência Cardíaca e Oxigênio*.

Fonte: Saravati (2024).

3. Sensor MLX90614 (Figura 3.6): Sensor de temperatura infravermelho usado para obter os dados da temperatura corporal e temperatura ambiente.

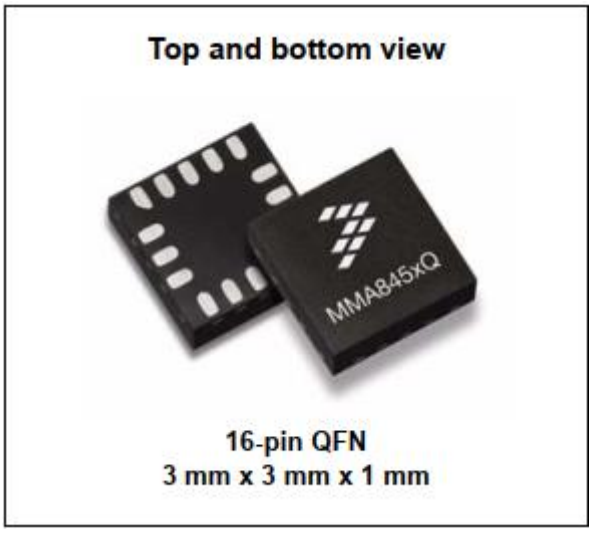
Figura 3.6 – Módulo Sensor de Temperatura Infravermelho.



Fonte: Retirado do datasheet do sensor. (2024).

4. *Sensor MMA8452Q* (Figura 3.7): Módulo acelerômetro, u*tilizado para analisar a posição do indivíduo e identificar caso ocorram quedas.*

Figura 3.7 – *Módulo Acelerômetro*.



Fonte: Retirado do datasheet do sensor. (2024).

5. Módulo GPS GY-NEO6MV2 (Figura 3.8): Módulo GPS para obter as informações de latitude e longitude, para identificar a localização precisa do usuário.

Figura 3.8 – Módulo GPS.

Fonte: RoboCore (2024).

6. Antena externa Multibanda Sub-1GHz - *Sigfox* (Figura 3.9): Converter ondas eletromagnéticas guiadas pela linha de transmissão, em ondas eletromagnéticas irradiadas, dentro das frequências Sub-1GHz da rede *Sigfox*.

Figura 3.9 – Antena Sub-1GHz para transmissão de sinais *Sigfox*.

Fonte: Própria (2024).

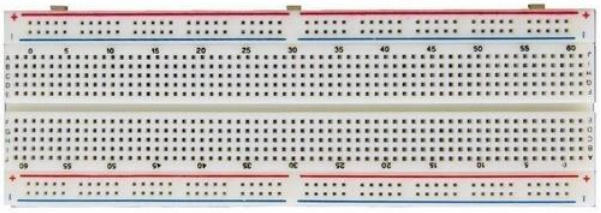
7. *Jumpers* (Figura 3.10): Utilizados para realizar as conexões necessárias.

Figura 3.10 – *Jumpers*.

Fonte: Própria (2024).

8. *Protoboard*: Montagem do *POC*.

Figura 3.11 – *Protoboard* utilizada no *POC*



Fonte: Própria (2024).

As subseções a seguir apresentam os principais pontos abordados durante o projeto do *POC*. Maiores detalhes de implementação poderão ser encontrados nas seções 3.3 e 3.4.

3.2.1 Conhecimento e familiarização do ambiente de desenvolvimento de *firmware*:

Tendo definidas as especificações e as tecnologias a serem exploradas no projeto, a primeira etapa do *POC* foi destinada à exploração e conhecimento das ferramentas de desenvolvimento. As escolhas basearam-se no que é recomendado nas documentações divulgadas pelos fabricantes, procurando sempre as soluções *open source* para que haja maior facilidade na reprodução deste projeto.

A *IDE (Integrated Development Environment) e as ferramentas utilizadas* para realizar as alterações no código de exemplo "Generic-Push-Button" do *iMCP HT32SX*, foram as mesmas informadas no *GitHub* oficial da HT Micron. A compilação e o *debug/release* do *firmware* foram realizados no *STM32CubeIDE*. E para gravação do firmware na placa e para situações específicas onde se percebeu a necessidade de apagar o conteúdo da memória *flash* do dispositivo, foi necessária a utilização do STM32CubeProgrammer. O projeto de *hardware* prevê os pinos de *serial wire* para este caso.

3.2.2 Envio de mensagens *Sigfox*

Considerando que a fase de *Proof of Concepts* foi projetada a partir do módulo SiP iMCP HTSXMO32L-22, o qual contém o microcontrolador Cortex-M0 integrado da placa do *iMCP HT32SX*, não havia a necessidade de medir a potência de saída do dispositivo, considerando que a rede já havia sido validada pelo fabricante. Desta maneira, o foco foi direcionado a simples testes de envio de mensagens à rede *Sigfox* e envio de mensagens.

3.2.3 Leitura dos sensores

O processo de validação dos sensores utilizados explorou o uso dos pinos do protocolo I2C *(Inter-Integrated Circuit)*, juntamente com os pinos TX e RX para comunicação via UART (Universal Asynchronous Receiver/Transmitter) com o módulo de GPS. O propósito desta fase do desenvolvimento do *POC era garantir a correta configuração, inicialização, leitura dos dados e todos os cálculos necessários no processamento dos dados brutos provenientes dos sensores, para posteriormente enviá-los para a plataforma Sigfox.*

3.3 Projeto de Firmware

A aplicação desenvolvida para rastreio e monitoramento de pessoas idosas em risco contou com o desenvolvimento de um *firmware para controle do* circuito integrado do sistema: o *STM32L052x8*, mestre de todo o processamento. O microcontrolador realiza a leitura e processamento dos dados, após montar payload, o microcontrolador acionado o *iMCP HT32SX* apenas para transmitir as mensagens à rede *Sigfox*. Os *softwares* usados na implementação são os mesmos mencionados na seção referente ao projeto de *POC*: *STM32CubeIDE*, para desenvolvimento e compilação *firmware*; e STM32CubeProgrammer para gravação do firmware na placa.

3.3.1 Projeto de Firmware do iMCP HT32SX

O *firmware* do *HT32SX* foi implementado a partir do exemplo "Generic-Push-Button" disponível no repositório público da HT Micron no *GitHub*.

O exemplo fornecido configura o dispositivo de modo que a USART/RF seja ativada no momento da transmissão de uma mensagem, processo esse que ocorre no momento que é pressionado o botão genérico da placa.

Utilizando o exemplo como base para configuração e envio dos dados, foram utilizadas as bibliotecas existentes no projeto de exemplo para garantir a correta configuração e ativação do rádio Sigfox. Para o envio dos dados dos sensores para a plataforma Sigfox, considerando que há o limite máximo de envio de 140 mensagens por dia, foi desenvolvido um contador no código do microcontrolador, utilizando o TIMER22 do microcontrolador, para que os payloads com os dados do usuário sejam enviados para a rede a cada 15 minutos.

3.3.2 Projeto de *Firmware* do *Microcontrolador STM32L052x8*

Sendo o *STM32L052x8* o mestre de todas as ações e de todo o processamento do sistema, a aplicação foi implementada basicamento em seu *firmware*. Esta subseção tem como objetivo detalhar as etapas de implementação do *firmware*, dividindo-as de acordo com a lista apresentada abaixo:

1. Formato de *Payload*;
2. Configuração do Sigfox Back-end;
3. Desenvolvimento do Front-end na TagoIO;

3.3.2.1 Formato de *Payload*

A aplicação central permanece coletando e processando os dados dos sensores continuamente, para montar os 2 payloads que serão enviados para o backend Sigfox e para a plataforma TagoIO. Houve a necessidade da criação de 2 payloads distintos em virtude da quantidade de informações que precisam ser enviadas, além da limitação existente na rede Sigfox, a qual permite payloads com no máximo 12 bytes por mensagem.

A Figura 3.36 apresenta o enum que define os tipos de eventos, os quais determinam qual payload será utilizado para serializar as informações enviadas nas transmissões. A descrição dos campos do pacote é feita em seguida.

Figura 3.36 – Enum tipo de evento.

```
typedef enum {
    LOCALIZACAO = 1,
    SAUDE = 2
} SIGFOX_Tipo_Evento_Enum;
```

Fonte: Própria (2024).

Os *payloads* enviados à rede *Sigfox* são compostos por quantidades distintas de *bytes, em virtude dos dados enviados por payload*. As Figuras 3.37 e 3.38 apresentam as *struct*s que serializam as informações enviadas nas transmissões. A descrição dos campos do pacote é feita em seguida.

Figura 3.37 – Implementação do *Payload com dados de Localização*.

```
typedef struct {
    uint8_t tipo_evento;     //8 bits = 1 byte
    float latitude;          //32 bits = 4 bytes
    float longitude;         //32 bits = 4 bytes
    uint8_t posicao_usuario;//8 bits = 1 byte
} __attribute__((packed)) Sigfox_payload_localizacao;
```

Fonte: Própria (2024).

- tipo_evento: Carrega a informação do evento que gerou a transmissão da última mensagem, de acordo com o *"enum"* da Figura 3.36;
- latitude e longitude: Carregam as informações sobre a localização do usuário no momento do envio do dado, permitindo saber o local onde o usuário se encontra, caso ocorra alguma situação grave que necessite de atendimento.
- posicao_usuario: Carrega uma informação binária (1 ou 2) e é utilizado para identificar se o usuário está em pé ou caído. Esta informação será utilizada para gerar alertas para diferentes destinos.

Figura 3.38 – Implementação do *Payload com os dados de Saúde*.

```
typedef struct {
    uint8_t tipo_evento;           //8 bits = 1 byte
    uint8_t frequencia_cardiaca;   //8 bits = 1 byte
    uint8_t spo2;                  //8 bits = 1 byte
    float temperatura_corporal;    //32 bits = 4 bytes
} __attribute__((packed)) Sigfox_payload_saude;
```

Fonte: Própria (2024).

- tipo_evento: Carrega a informação do evento que gerou a transmissão da última mensagem, de acordo com o “enum” da Figura 3.36;
- frequencia_cardiaca: Como o próprio nome define, carrega a informação da frequência cardíaca do usuário.
- spo2: Como o próprio nome define, carrega a informações de oximetria do usuário.
- temperatura_corporal: Como o próprio nome define, carrega a informação da temperatura corporal do usuário.

3.4 Configuração do *Sigfox Back-end*

As mensagens transmitidas para a plataforma da rede Sigfox são armazenadas na nuvem, possibilitando sua recuperação através de callbacks configuráveis. Esses serviços de callback direcionam os pacotes com as mensagens recebidas, juntamente com as informações do dispositivo transmissor, para servidores e plataformas externas.

As principais configurações dos callbacks são realizadas em um campo chamado "body" na plataforma de backend da plataforma Sigfox. Este campo aceita um código em JSON, especificando os dados que devem ser enviados ao servidor de frontend do usuário. O código JSON utilizado para o callback neste projeto é ilustrado na Figura 3.39.

Figura 3.39 – *Callback body* escrito em *JSON*.

Fonte: Própria (2024).

Os principais campos que podem ser assinalados no código mostrado acima são as variáveis denominadas *“device”* e *“data”, pois e*las carregam as informações do *”ID” Sigfox* do dispositivo e o *payload com os dados recebidos pela plataforma Sigfox*, respectivamente.

3.5 Desenvolvimento do *Front-end* na TagoIO

O monitoramento dos dados enviados à rede *Sigfox* é feito por meio de uma *dashboard* desenvolvida na plataforma da TagoIO. Os dados recebidos pelo servidor da plataforma seguem o mesmo padrão do código *JSON* configurado no *callback* da *Sigfox*. O objetivo agora é descompactar os dados desejados, atribuindo-os a variáveis globais da plataforma, para então usá-los na *dashboard*.

Os dados são recuperados por meio de um parse vinculado ao dispositivo registrado na TagoIO. Esse parse gera variáveis globais e atribui a elas os valores obtidos através do callback da Sigfox.

A Figura 3.40 demonstra a implementação do *parser criado* para a aplicação desenvolvida no TCC, onde são recebidos diferentes payloads, 1 contendo os dados de Localização do usuário e outro contendo os dados de Saúde do usuário. Após a execução do parse, o sistema criará as oito variáveis *"tipo_evento", "latitude", "longitude", "localizacao", "posicao_usuario", "frequencia_cardiaca", "spo2" e "temperatura_corporal"* em seu banco de dados, para que recebam e armazenem as informações interpretadas pelo *parser*.

Figura 3.40 – *Parse* escrito para a aplicação de monitoramento e rastreio de usuários

```
// Busca a variável payload no payload global
const payload_raw = payload.find(x => x.variable === 'data' || x.variable === 'payload_raw' || x.variable === 'payload');

// Valida se payload_raw existe
if (payload_raw) {
  try {
    // Converte os dados de Hex para Buffer em JavaScript
    const buffer = Buffer.from(payload_raw.value, 'hex');
    let dadosPayload = [];
    let posicaoUsuario = "";

    console.log("payload_raw.value: ", payload_raw.value);
    console.log("buffer: ", buffer);

    if(buffer.readInt8(0) === 0x01){  // payload_localizacao
      const latitude = buffer.readFloatLE(1);
      const longitude = buffer.readFloatLE(5);

      dadosPayload.push({variable: 'tipo_evento', value: buffer.readInt8(0)});
      dadosPayload.push({variable: 'latitude', value: buffer.readFloatLE(1), unit: '°'});
      dadosPayload.push({variable: 'longitude', value: buffer.readFloatLE(5), unit: '°'});
      dadosPayload.push({variable: 'posicao_usuario', value: buffer.readInt8(9)});
      dadosPayload.push({variable: 'localizacao', value: "Localizacao atual", "location": { "lat": latitude, "lng": longitude}});

      if(buffer.readInt8(9) === 2){
        posicaoUsuario = "Individuo caiu.";
        dadosPayload.push({ variable: 'posicao_usuario_status', value: posicaoUsuario });
      }
    }
    else if(buffer.readInt8(0) === 0x02){   // payload_saude
      dadosPayload.push({variable: 'tipo_evento', value: buffer.readInt8(0)});
      dadosPayload.push({variable: 'frequencia_cardiaca', value: buffer.readUInt8(1), unit: 'BPM'});
      dadosPayload.push({variable: 'spo2', value: buffer.readUInt8(2), unit: '%'});
      dadosPayload.push({variable: 'temperatura_corporal', value: buffer.readFloatLE(3), unit: '°C'});
    }

    console.log("dadosPayload: ", dadosPayload);

    // Concatena o conteúdo enviado pelo dispositivo com o conteúdo gerado neste payload parser.
    payload = payload.concat(dadosPayload.map(x => ({ ...x, serie: payload_raw.serie, time: payload_raw.time })));
```

Fonte: Própria (2024).

O dashboard criado na plataforma TagoIO, possui visualizadores para todas as informações essenciais do sistema, incluindo widgets para dados de temperatura corporal, localização do usuário, frequência cardíaca, saturação de oxigênio no sangue (SpO2), latitude, longitude e localização estimada. Além disso, gráficos e tabelas mostram as variações dos dados durante a operação do sistema.

O processo que os dados percorrem, desde o envio à rede Sigfox até a exibição dos valores na dashboard, é detalhado a seguir.

1. *STM32L052T8Y aciona* o *iMCP HT32SX para transmissão dos dados* (Figura 3.41):

Figura 3.41 – Transmissão dos payloads pelo *HT32SX*.

Fonte: Própria (2024).

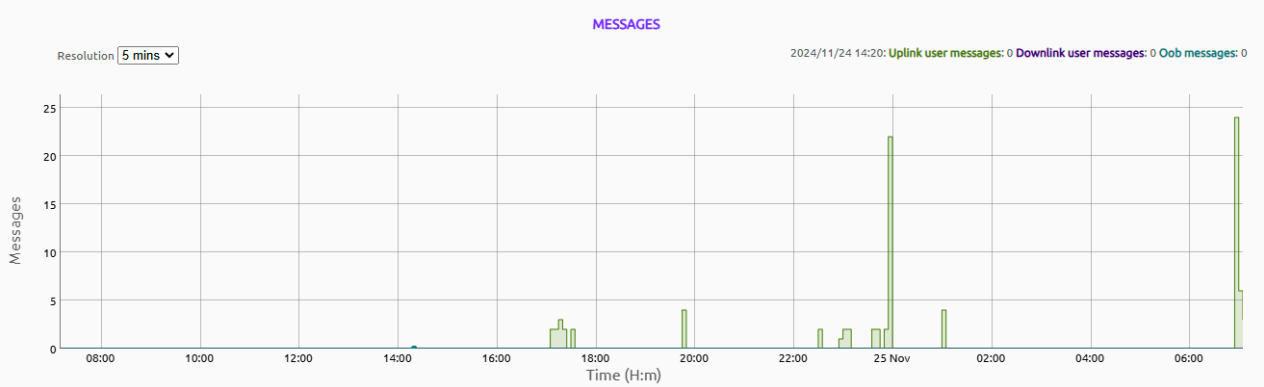
2. Recebimento da mensagem no Backend *Sigfox* (Figura 3.42):

Figura 3.42 – Mensagem recebida no *Backend*.

Fonte: Própria (2024).

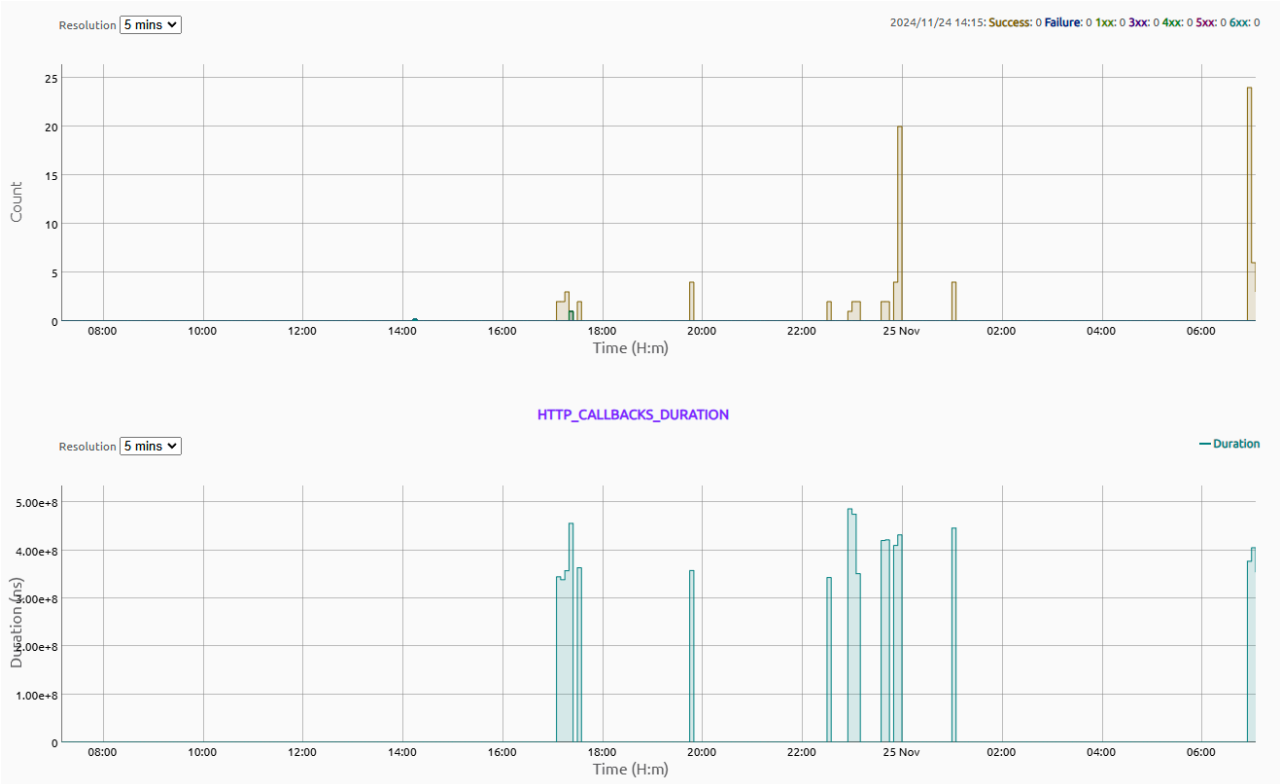
2.1. Estatísticas Backend Sigfox (Figuras 3.43 e 3.44)

Figura 3.43 – Estatísticas de mensagens recebidas no *Backend*.



Fonte: Própria Backend Sigfox (2024).

Figura 3.44 – Estatísticas de callbacks enviados pelo *Backend*.



Fonte: Própria Backend Sigfox (2024).

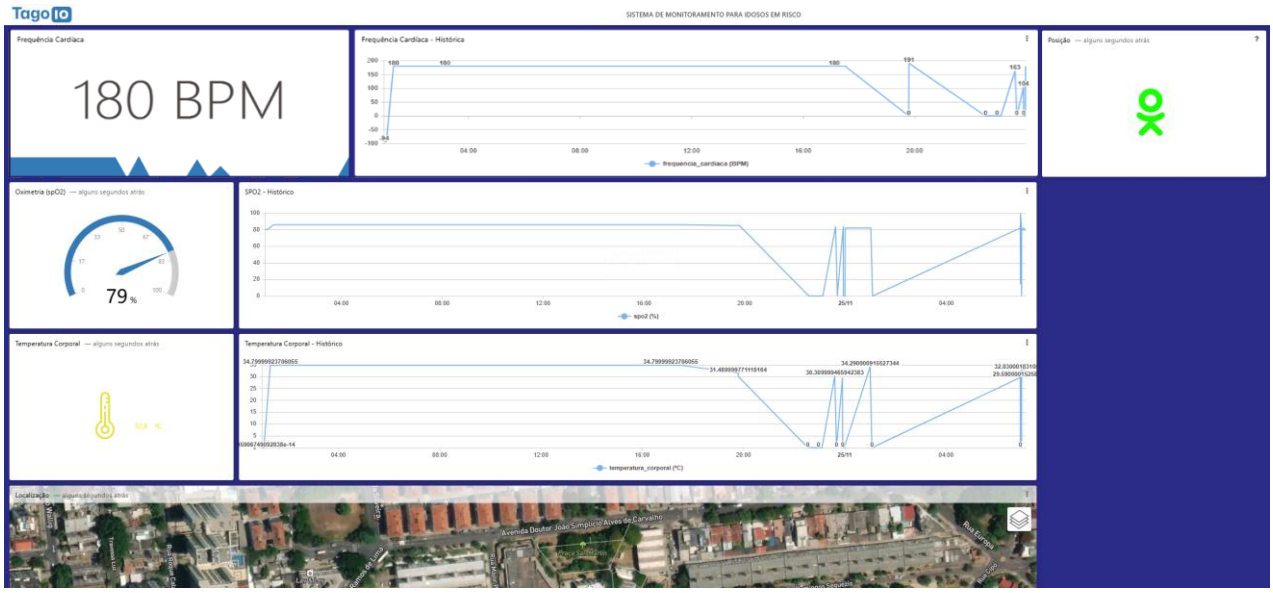
3. Dado é recebido pelo servidor da plataforma da TagoIO (Figura 3.45):

Figura 3.45 – *Callback* da *Sigfox* recebido pelo servidor da TagoIO.

Fonte: Própria (2024).

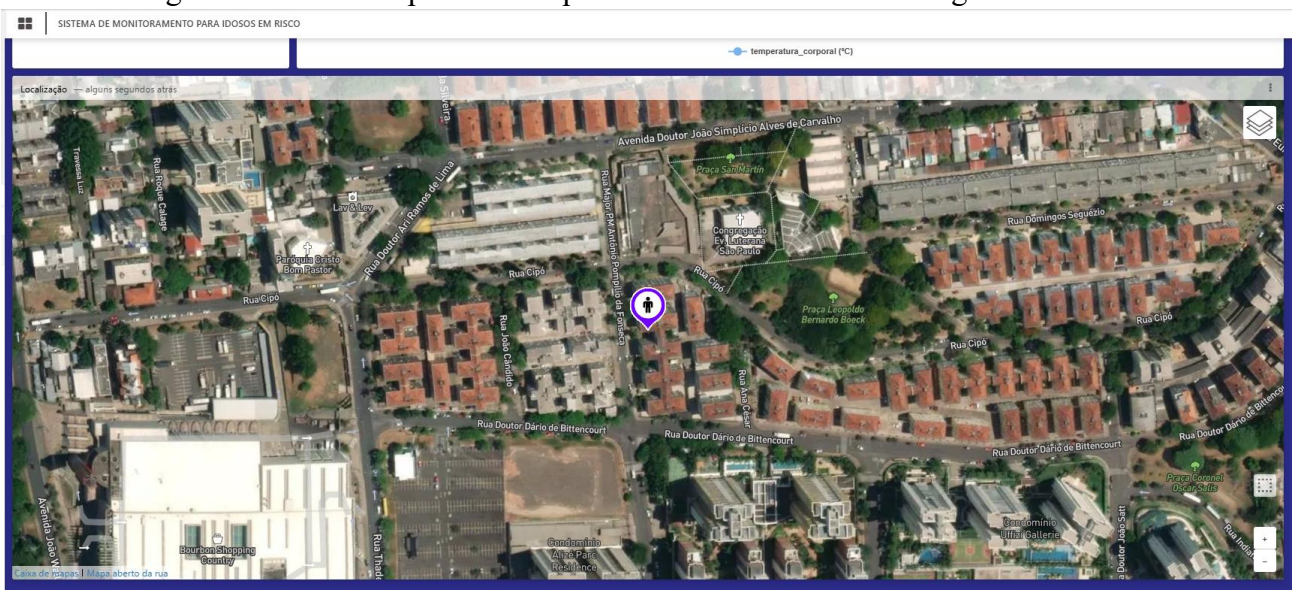
4. O *parse* faz a decodificação das informações e cria as variáveis globais. Os *widgets* configurados para plotar as informações das variáveis atualizam os seus valores e os exibem na *dashboard*, conforme mostrados nas Figuras 3.46 e 3.47:

Figura 3.46 – Dados plotados na parte superior *dashboard* na TagoIO.



Fonte: Própria (2024).

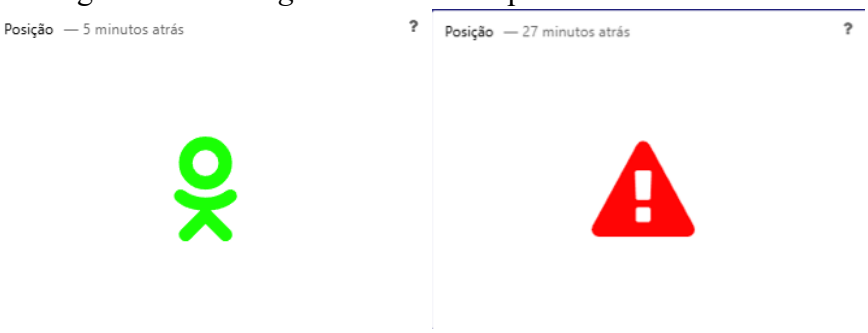
Figura 3.47 – Dados plotados na parte inferior *dashboard* na TagoIO.



Fonte: Própria (2024).

A plataforma da TagoIO oferece a opção de o usuário adicionar condições aos *widgets*, para que as informações ou as imagens plotadas reajam de acordo com os *inputs* que recebem das variáveis. O *widget* de posição do usuário, por exemplo, exibe ícones e cores distintas, caso o usuário esteja em pé ou tenha caído.

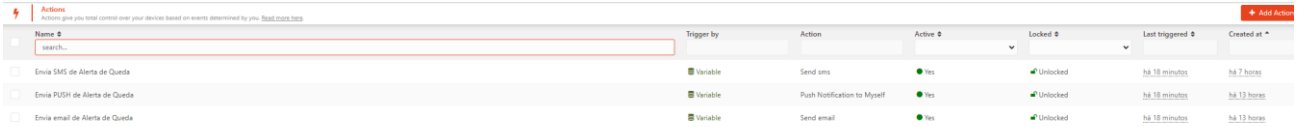
Figura 3.48 – *Widget* de eventos disponíveis na *dashboard*.



Fonte: Própria (2024).

Além da opção de adicionar condições ao dashboard, é possível criar ações especificas, as quais são executadas, conforme os parâmetros definidos. Como exemplo, foram criadas ações de alerta, as quais enviam mensagens por e-mail, por SMS e por push para os usuários definidos.

Figura 3.49 – *Tela cadastro ações/alertas.*



Fonte: Própria (2024).

Figura 3.50 – *Alerta criado.*

Fonte: Própria (2024).

4. Validação

Este capítulo detalha os processos de validação usados durante o trabalho de conclusão de curso. Todos os testes foram feitos utilizando a *POC desenvolvida*. A validação do projeto foi dividida nas seguintes etapas:

1. Testes de Funcionalidades Básicas;
2. Teste de Confiabilidade de *Firmware*;

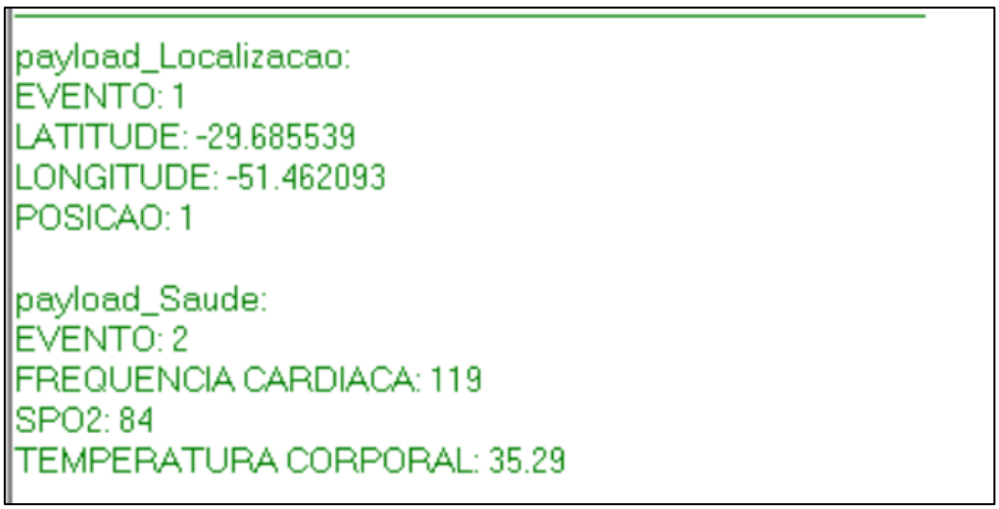
4.1 Testes de Funcionalidades Básicas

Os primeiros testes validaram de forma simples e objetiva as funcionalidades do sistema que necessitam de algum *input* de usuário. Os resultados e os detalhes são descritos a seguir.

4.1.1 Leitura e processamento dos dados dos sensores.

Validada a leitura dos dados brutos dos sensores, processamento dos respectivos e geração dos payloads para envio para rede Sigfox. A Figura 4.1 apresenta os payloas gerados com os respectivos dados, prontos para serem enviados.

Figura 4.1 – Payloads com dados lidos e processados dos sensores.

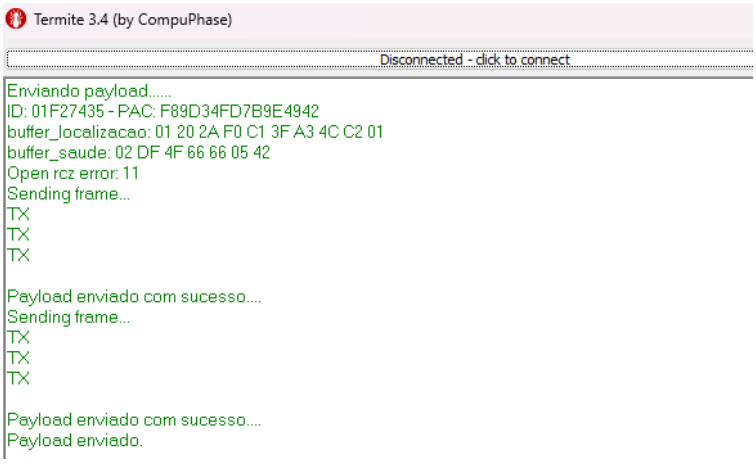


Fonte: Própria (2024).

4.1.2 Comunicação com a Rede *Sigfox*

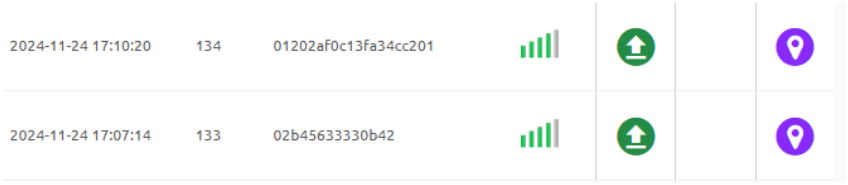
Esta etapa de testes validou o algoritmo na comunicação serial do *HT32SX* e a transmissão das mensagens à rede *Sigfox*. A Figura 4.2 mostra o envio dos comandos ao *Back-end* da *Sigfox*.

Figura 4.2 – Validação do algoritmo de envio de dados ao *HT32SX*.



Fonte: Própria (2024).

Figura 4.3 – Mensagens recebidas no backend Sigfox.

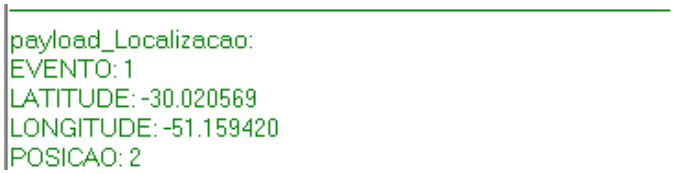


Fonte: Própria (2024).

4.1.3 Teste de Queda do Usuário

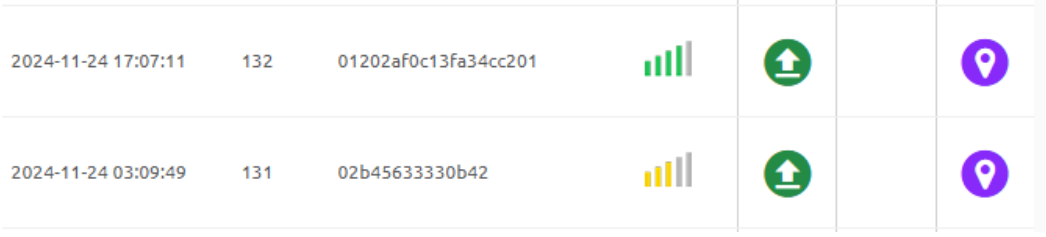
Validação do comportamento do sistema ao ser identificada a queda do usuário. As Figuras 4.3, 4.4, 4.5, 4.6 e 4.7 demonstram o teste feito, desde os *logs* impressos no terminal, até a plotagem dos valores na plataforma da *TagoIO* e os alertas gerados:

Figura 4.4 – Teste de queda do usuário (posição = 2).



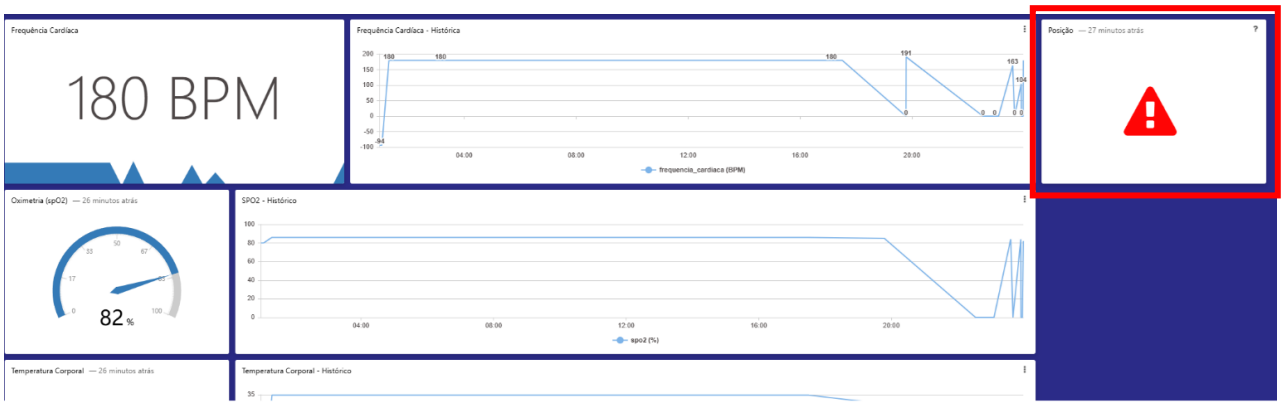
Fonte: Própria (2024).

Figura 4.5 – Teste de queda do usuário no *Back-end* da *Sigfox*.



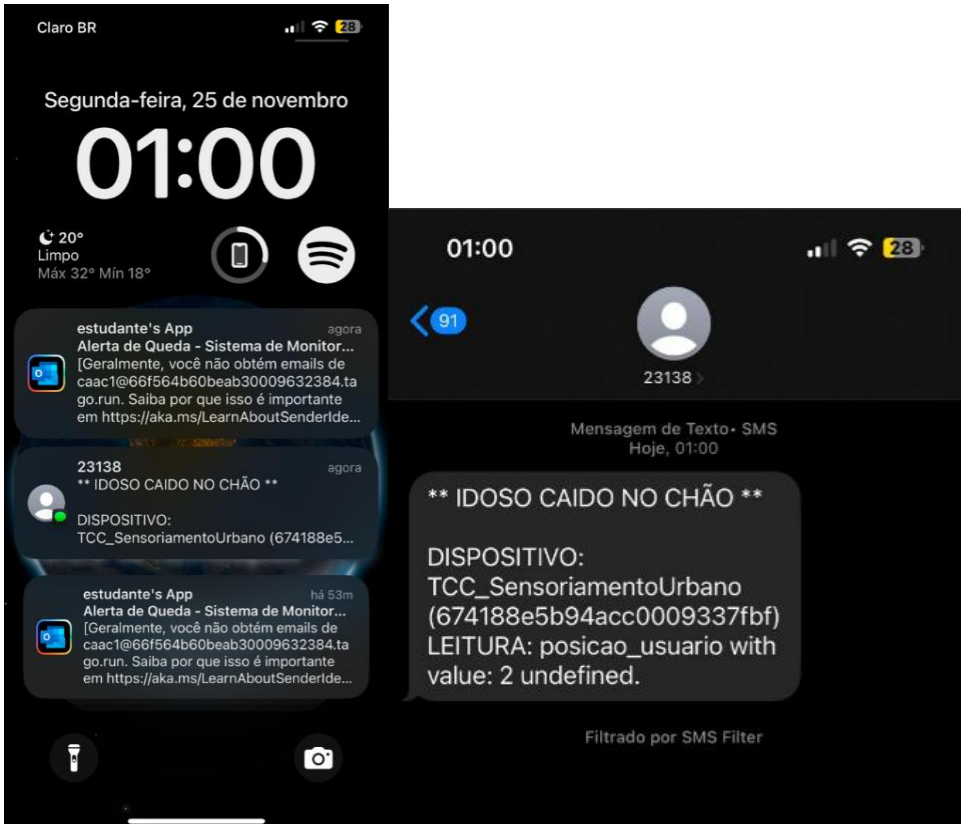
Fonte: Própria (2024).

Figura 4.6 – Teste de queda do usuário na plataforma da *TagoIO*.



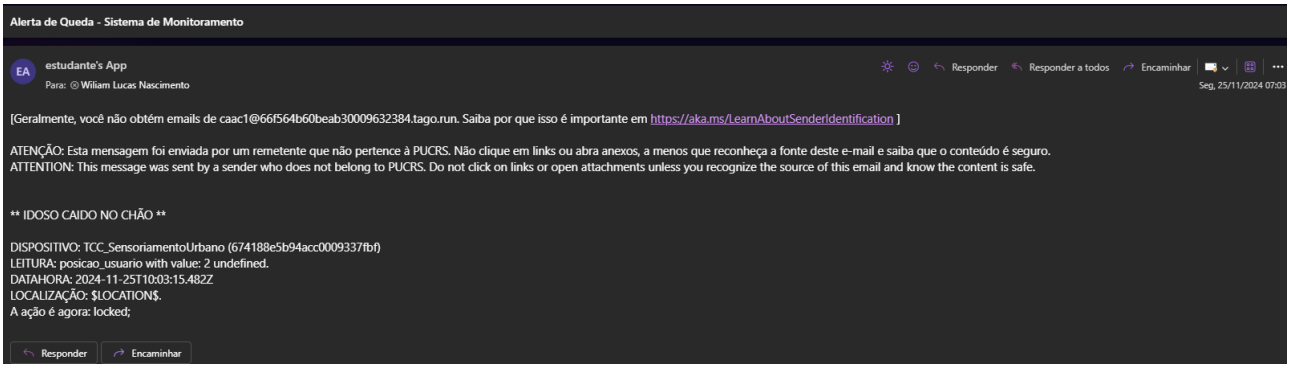
Fonte: Própria (2021).

Figura 4.7 – Teste de queda do usuário – Alertas recebidos - SMS.



Fonte: Própria (2024).

Figura 4.8 – Teste de queda do usuário – Alertas recebidos - EMAIL.



Fonte: Própria (2024).

5. Problemas e Dificuldades

Durante o processo de desenvolvimento do trabalho, foram enfrentados alguns problemas e dificuldades, gerados por forças externas.

5.1 Licenças Sigfox

No início do desenvolvimento, foram encontradas dificuldades elevadas para aquisição do contrato e licenças necessárias para utilização da rede Sigfox. Em virtude de estar utilizando o SiP da empresa HT Micron, foi entrado em contato para aquisição das licenças para uso, porém, foi informado que não realizam a venda das respectivas licenças. Ao tentar realizar a compra de um kit de desenvolvimento com as licenças para uso, foi obtido o mesmo retorno, que a empresa não realiza a venda necessária, a qual eu estava solicitando.
A partir deste retorno, foi entrado em contato com o suporte internacional da empresa responsável pela rede Sigfox, os quais informaram que não realizavam a venda direta das licenças e que a situação deveria ser verificada diretamente com o operador responsável pela rede, no respectivo país/região que será utilizada a rede.
Por fim, identificado o operador responsável pela rede no Brasil... WND Brasil e após algumas tentativas, foi possível contato com um gerente de vendas da empresa, o qual gerou um contrato temporário e disponibilização de 3 licenças/tokens para uso da rede. Todo esse processo onerou o tempo de desenvolvimento do TCC, gerando um atraso por volta de 8 semanas.

5.2. Sensores

Outro problema enfrentado, foi a falta de alguns sensores (MAX30102 e MLX90614) no laboratório da universidade, os quais realizei a compra pela internet e tive que aguardar em torno de 2 a 4 semanas para recebê-los.

5.3. Equipamentos

Enfrentados problemas com os meus equipamentos próprios, o que onerou o tempo de desenvolvimento do trabalho de conclusão.

6. Conclusão

Os resultados obtidos no trabalho de conclusão de curso atingiram o objetivo esperado. A exploração das tecnologias *Sigfox* em conjunto com as tecnologias dos sensores, apresentou algumas possibilidades para possíveis soluções de *IoT*.

A tecnologia *Sigfox* mostrou ser uma das melhores opções para sistemas de monitoramento, mesmo com todas as dificuldades e impedimentos iniciais para a configuração da rede, utilizando algumas bibliotecas disponibilizadas no GitHub público da HT Micron, ajudou significativamente na configuração do rádio Sigfox. A interface amigável da plataforma na nuvem e a facilidade na configuração de *callbacks* para o tratamento de dados, confirmou que a infraestrutura fornecida pela rede e seus respectivos provedores locais, atende de fato com as necessidades de diversos tipos de aplicações.

O *iMCP HT32SX*, usado como *access point* à rede *Sigfox*, atendeu todos os requisitos do projeto. Diferentemente de outros módulos concorrentes, o dispositivo permite com que o desenvolvedor programe o microcontrolador interno de acordo com as necessidades da aplicação, podendo inclusive alterar a potência de saída do rádio, incluir novos periféricos e otimizar os *firmwares* fornecidos como exemplo. Uma das principais vantagens do circuito integrado, do ponto de vista do desenvolvedor, é ter diversos exemplos de códigos abertos no *GitHub* do fabricante. Há um exemplo de rede *P2P*, que funciona como uma extensão da rede *Sifox*, que eventualmente pode ser usado para incrementar este projeto.

Os sensores e módulos utilizados também apresentaram excelentes resultados. O módulo GPS utilizado apresentou ótima precisão na triangulação e definição da localização do usuário.

O desenvolvimento e os testes realizados utilizando o *POC*, montado com o módulo SiP iMCP HTSXMO32L-22, seguiram o seu fluxo normal e foram suficientes para a conclusão do projeto.

A validação do sistema comprovou a funcionalidade da aplicação, conforme ideia inicial do trabalho. O projeto desenvolvido é capaz

de rastrear e monitorar indivíduos em situação de risco ou com a saúde mais debilitada, visando o monitoramento contínuo e a análise comportamental, alcançamos o objetivo de proporcionar maior segurança e autonomia para este público. Os arquivos do projeto estão armazenados em um repositório público no *GitHub* e poderão ser acessados através deste link: https://github.com/wlnascimento/TCC_SensoriamentoUrbano_Sigfox_v1

Algumas melhorias, como o desenvolvimento de uma lib própria e específica para configuração do módulo Sigfox, pode garantir uma melhoria e simplificação no processo de configuração e gerar uma redução expressiva no tamanho do firmware gerado. Sendo assim, conclui-se que o trabalho de conclusão de curso atingiu os objetivos propostos.

7. Agradecimentos

A realização deste Trabalho de Conclusão de Curso representa não apenas o encerramento de uma jornada acadêmica, mas também o reflexo do apoio e da dedicação de diversas pessoas que, direta ou indiretamente, contribuíram para que este momento fosse possível. A todas elas, deixo aqui minha mais profunda gratidão.

Primeiramente, agradeço a Deus, pela força, inspiração e perseverança em todos os momentos, especialmente nos mais desafiadores. Sem a fé e a crença em dias melhores, este projeto não seria possível.

À minha família, pelo amor incondicional, pelo apoio emocional e pela paciência ao longo de todos esses anos. Em especial, agradeço aos meus pais, João e Fátima Nascimento e aos meus irmãos, que sempre acreditaram em mim, me incentivaram a seguir em frente e me proporcionaram as condições necessárias para que eu pudesse alcançar este objetivo.

Gostaria de dedicar um agradecimento especial à minha querida avó Amélia, que hoje descansa em paz, mas que, durante sua vida, me ensinou importantes lições de amor, resiliência e coragem. Sua memória sempre será uma fonte de inspiração para mim.

Agradeço também à minha madrinha Adriana, que sempre me incentivou e esteve presente em todos os momentos importantes da minha vida. Sua confiança, carinho e apoio constante foram fundamentais para que eu chegasse até aqui, e sou imensamente grato(a) por tê-la ao meu lado.

Aos meus amigos, que compartilharam desta jornada acadêmica, seja oferecendo palavras de incentivo, seja servindo de apoio nas horas de dificuldade. As trocas de experiências e os momentos de descontração foram fundamentais para que eu me mantivesse motivado(a) durante este processo.

Agradeço imensamente do fundo do meu coração, ao meu professor e orientador Júlio César Marques de Lima, pela paciência, pela orientação e pelo conhecimento transmitido. Sua dedicação e suas contribuições foram essenciais para a realização deste trabalho e para o meu crescimento acadêmico e profissional. Muito obrigado, professor, pelas suas palavras de motivação e conforto durante os períodos de ansiedade vividos nessa reta final. Mais uma vez, muito obrigado!

Aos professores e colegas da PUCRS, que, ao longo da graduação, proporcionaram aprendizado, debates enriquecedores e experiências que levarei para toda a vida. Cada aula, cada desafio e cada troca de ideias contribuíram para a construção deste momento.

Por fim, agradeço a todos aqueles que, de alguma forma, participaram deste percurso e me ajudaram a superar os obstáculos que surgiram. Este trabalho é fruto de um esforço coletivo, e sou imensamente grato por ter contado com tanto apoio e incentivo.

Muito obrigado!

8. Referências Bibliográficas

[1] WND Brasil. Uma visão técnica da rede sigfox, aug 2017. Disponível em: https://www.embarcados.com.br/uma-visao-tecnica-da-rede-sigfox/.

[2] Sigfox. Sigfox technical overview, may 2017. Disponível em: https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf.

[3] Sigfox. Radio configurations, mar 2021. Disponível em: https://build.sigfox.com/sigfox-radio-configurations-rc.

[4] R&D HT Micron. Ht32sx - firmware examples, mar 2021. Disponível em: https://github.com/htmicron/ht32sx/tree/master_2.

[5] R&D HT Micron. *iMCP HT32SX V2.2 – SiP Sigfox Sigfox® Monarch RF Transceiver Systemin-Package, datasheet*. HT Micron Semicondutores SA, 5 edition, mar 2021.

[6] HT Micron. Guia de instalação da placa de avaliação do módulo SIP HT Micron IMCP HTSXMO32L-22.,. Disponível em: https://manuals.plus/pt/ht-micron/imcp-htsxmo32l-22-sip-module-evaluation-board-manual .

[7] HT Micron. Guia de instalação da placa de avaliação do módulo SIP HT Micron IMCP HTSXMO32L-22.,. Disponível em: https://manuals.plus/pt/ht-micron/imcp-htsxmo32l-22-sip-module-evaluation-board-manual .

[8] Maxim Integrated Products. Datasheet MAX30102.,. Disponível em: https://cdn.awsli.com.br/945/945993/arquivos/MAX30102.pdf

[9] Melexis. Datasheet MLX90614.,. Disponível em: https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90614

[10] NXP Semiconductors. MMA8452Q, 3-axis, 12-bit/8-bit digital accelerometer.,. Disponível em: https://www.makerhero.com/img/files/download/MMA8452Q-Datasheet.pdf

[11] u-blox AG. NEO-6 u-blox 6 GPS Modules Data Sheet.,. Disponível em: https://cdn.awsli.com.br/945/945993/arquivos/GY-NEO6MV2-GPS-Module-Datasheet.pdf

[12] Graziele Rodrigues. Projeto Prático: Utilizando o Módulo GPS GY-NEO6MV2 com a Franzininho C0.,. Disponível em: https://embarcados.com.br/projeto-pratico-utilizando-o-modulo-gps-com-a-franzininho-c0/

[13] Sigfox. 0G Technology Documentation.,. Disponível em: https://support.sigfox.com/docs

[14] National Falls Prevention Resource Center. Falls and Fractures in Older Adults: Causes and Prevention.,. Disponível em: https://www.nia.nih.gov/health/falls-and-falls-prevention/falls-and-fractures-older-adults-causes-and-prevention

[15] WHO. (2021). Falls.,. Disponível em: https://www.who.int/news-room/fact-sheets/detail/falls

[16] CDC.gov. Older Adult Fall Prevention.;. Disponível em: https://www.cdc.gov/falls/data-research/facts-stats/index.html