UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Renata de Camillo Corrêa Bernardes

Sistema IoT de Identificação de Problemas de Infraestrutura Urbana

São Carlos

2019

Renata de Camillo Corrêa Bernardes

Sistema IoT de Identificação de Problemas de Infraestrutura Urbana

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheira Eletricista.

Orientador: Prof. Dr. Leonardo André Ambrosio

São Carlos

2019

de Camillo Corrêa Bernardes, Renata
D518s
Sistema IoT de Identificação de Problemas de Infraestrutura Urbana / Renata de Camillo Corrêa Bernardes ; orientador Leonardo André Ambrosio. São Carlos, 2019. Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) – Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019. 1. Internet das Coisas. 2. Cidades Inteligentes. 3. Comunicação 3G. 4. Rede Sigfox. 5. LPWAN 6. GPS. I. Titulo.

FOLHA DE APROVAÇÃO

Nome: Renata de Camillo Corrêa Bernardes

Título: "Sistema de identificação de problemas de infraestrutura urbana"

Trabalho de Conclusão de Curso defendido e aprovado em 06/06/19,

com NOTA 10 (DEZ, ZERO), pela Comissão Julgadora:

Prof. Dr. Leonardo André Ambrosio - Orientador - SEL/EESC/USP

Profa. Titular Vilma Alves de Oliveira - SEL/EESC/USP

Prof. Titular Murilo Araujo Romero - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado à minha mãe, Claudia, ao meu pai, Luiz Henrique, e ao meu padrinho, José Carlos.

AGRADECIMENTOS

Agradeço primeiramente à minha família, à minha mãe, Claudia, e ao meu pai, Luiz Henrique, por me apoiarem e incentivarem durante o período de graduação.

Ao meu orientador, Prof. Dr. Leonardo André Ambrosio, que desde nossa primeira atividade juntos me apoiou, confiou e guiou para que eu fizesse sempre o melhor trabalho possível.

Aos meus colegas de curso, principalmente Henrique e Jordão, pelo companheirismo durante esta jornada e troca de aprendizado.

Às empresas Duodigit e Telit, por ajudarem através do empréstimo dos equipamentos utilizados neste trabalho.

"Nunca permita que a imaginação limitada dos outros o limite"
Mae Jemison

RESUMO

BERNARDES, R. C. C. **Sistema IoT de Identificação de Problemas de Infraestrutura Urbana**. 2019. 82p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Grandes centros urbanos apresentam inúmeros problemas, os quais influenciam diretamente na qualidade de vida de sua população. Levando em conta agilizar a identificação e localização de problemas, o trabalho proposto apresenta o SIPIU - Sistema de Identificação de Problemas de Infraestrutura Urbana, que busca identificar, localizar e cadastrar diversos problemas de infraestrutura urbana em prol de agilizar seus reparos. Propondo uma mudança em relação a como os problemas de infraestrutura urbana são atualmente denunciados e cadastrados, o SIPIU utiliza conceito de Cidades Inteligentes, Internet das Coisas (IoT) e tecnologias envolvidas. Pensando em mapear as cidades de modo eficiente, seriam necessários funcionários que percorrem longas distâncias durante seu turno de trabalho. Os garis, responsáveis pela limpeza das vias públicas, adequariam-se a essa categoria. O sistema é composto por um *hardware* constituído por um processador, um dispositivo GPS e um módulo de comunicação, inicialmente com tecnologia 3G e posteriormente com tecnologia Sigfox. Este, é acoplado nos carrinhos utilizados por esses profissionais, de modo que ao identificar um problema, o sistema seria acionado e os dados coletados enviados para a nuvem e chegam para o usuário final, prefeituras por exemplo, por meio de uma interface. Por fim, são apresentados os testes conduzidos sobre o sistema utilizando as tecnologias 3G e Sigfox. Os resultados são discutidos e ambas as tecnologias são comparadas em relação a custo, operação e consumo energético. E, com isso, chegar a conclusão dos benefícios de utilizar de conceitos de IoT e Cidades Inteligentes para a população de centros urbanos.

Palavras-chave: Internet das Coisas. Cidades Inteligentes. Comunicação 3G. Rede Sigfox. LPWAN. GPS.

ABSTRACT

BERNARDES, R. C. C. **Urban Infrastructure Problem Identification IoT System**. 2019. 82p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Large urban centers have numerous problems, which directly influence the quality of life of their population. The proposed work presents the SIPIU - Urban Infrastructure Problem Identification System, which seeks to identify, locate and register several problems of urban infrastructure in order to expedite its repairs. Proposing a change regarding how urban infrastructure problems are currently reported and registered, SIPIU uses the concept of Smart Cities, Internet of Things (IoT) and technologies involved. Thinking of mapping cities effectively, it would be necessary for employees to travel long distances during work. Sweepers, who work professionally in the cleaning of public roads, would fit into this category. The system is composed by a hardware that consists of a processor, a GPS module and a communication module, initially with 3G technology and then with Sigfox technology. The hardware is coupled in the cars used by these professionals, so that by identifying a problem, the system would be triggered and the data collected will be sent to the cloud and sent to the end user, prefectures for example, through an interface. Finally, the tests conducted on the system using 3G and Sigfox technologies are presented. The results are discussed and both technologies are compared in terms of cost, operation and energy consumption. And with that, coming to the conclusion of the benefits of using IoT and Smart Cities concepts for the population.

Keywords: Internet of Things. Smart Cities. 3G Communication. Sigfox Network. LP-WAN. GPS.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

SIPIU	Sistema de Identificação de Problemas de Infraestrutura Urbana
IoT	*Internet of Things*
GPS	*Global Positioning System*
LPWAN	*Low Power Wide Area Network*
LCD	*Liquid Crystal Display*
USB	*Universal Serial Bus*
UART	*Universal Asynchronous Receiver/Transmitter*
UMTS	Sistema Universal de Telecomunicações Móveis
ERBs	Estações de Rádio Base
CDMA	*Code Division Multiple Access*
GSM	*Global System for Mobile Communications*
GPRS	*General Packet Radio Service*

SUMÁRIO

1 INTRODUÇÃO

Grandes centros urbanos apresentam inúmeros problemas de infraestrutura, os quais influenciam diretamente na qualidade de vida de sua população (SANTOS, 1996). Levando em conta agilizar a identificação e localização destes problemas, o projeto proposto intitulado de SIPIU - Sistema de Identificação de Problemas de Infraestrutura Urbana - neste documento busca identificar, localizar e cadastrar de forma inteligente diversos problemas de infraestrutura urbana, com o intuito de agilizar seus reparos. Dentre os problemas mais comuns estão: vazamentos de água e esgoto, buracos e imperfeições no asfalto e calçadas, terrenos baldios irregulares e entulho em locais proibidos (NÓBREGA et al., 2013).

Pensando em mapear as cidades de modo eficiente, seria necessário o uso de tecnologias específicas e de profissionais que percorrem longas distâncias durante seu turno de trabalho. Os garis, os quais atuam profissionalmente na limpeza de vias públicas, adequam-se a esta categoria. Por exemplo, a cidade de São Paulo possui 13 mil profissionais neste ramo, os quais percorrem 7,8 quilômetros de vias públicas por dia cada (PREFEITURA DE SÃO PAULO, 2017).

O *hardware* - composto por um microcontrolador, um GPS e um módulo de comunicação 3G - é acoplado aos carrinhos utilizados por estes profissionais (Figura 1) de modo que ao identificar um problema, eles, através de chaves e botões, ativariam o sistema e, consequentemente, o GPS, para em seguida escolher uma categoria na qual o problema é classificado. Esses dados são enviados para a nuvem, transmitidos através da rede 3G e armazenados em uma plataforma IoT, na qual os órgãos competentes pelos reparos teriam acesso rapidamente às informações e, consequentemente, geraria a possibilidade de equacionar e solucionar os problemas reportados.

O projeto, devido as tecnologias e metodologias utilizadas, encaixa-se dentro dos conceitos de Internet das Coisas (IoT) e Cidades Inteligentes, que serão abordados no capítulo seguinte. E para comparar o desempenho e custo-benefício do sistema em relação à sua rede de comunicação utilizada, o módulo 3G será substituído por um que opere através da rede Sigfox, de modo a analisar o custo de envio das informações e o uso da bateria.

A escolha da tecnologia 3G foi com base em sua aplicabilidade e cobertura em centros urbanos. Já a escolha do uso da tecnologia Sigfox foi devido a ela ser uma rede específica para aplicações em Internet das Coisas e atender as especificações deste projeto.

A presente proposta de trabalho de conclusão de curso (TCC) é a evolução de um projeto participante da competição “Contest Embarcados: Conectando à Internet das

Figura 1: Lixeira utilizada por garis na cidade de São Paulo.

Fonte: Foto obtida pela autora.

Coisas com NXP 2016", na qual a estudante foi a primeira colocada (EMBARCADOS, 2016; SIPIU, 2016).

1.1 Objetivos

O objetivo deste projeto consiste na compreensão e aprofundamento em conceitos de Internet das Coisas e Cidades Inteligentes. Além disso, visa levantar problemas de infraestrutura no contexto urbano que mais influenciam a vida da população local, através da implementação de diversos equipamentos formando um sistema, incluindo *hardware*, *software*, protocolos e redes de comunicação e computação em nuvem. Outro ponto a ser considerado é a comparação do uso no sistema de tecnologia 3G e tecnologia Sigfox, em relação a custo e aplicabilidade.

1.2 Estrutura do Trabalho

O trabalho em questão está dividido em 5 capítulos, sendo o primeiro a apresentação da problemática a ser abordada e seus objetivos. O capítulo 2, uma revisão bibliográfica de conceitos envolvidos. O capítulo 3 apresentará os materiais e métodos utilizados no projeto, tanto com tecnologia 3G quanto com tecnologia Sigfox. O capítulo 4 apresenta os resultados e discussões em relação à comparação das tecnologias utilizadas e a aplicação

do sistema. E, por fim,o capítulo 5 é a conclusão do trabalho.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão abordados conceitos que são importantes para entendimento e contextualização do leitor ao projeto.

2.1 Internet das Coisas

Internet das Coisas (do inglês *Internet of Things* - IoT) pode ser definido como a interação entre diversos tipos de dispositivos, objetos, sensores, pessoas e o ambiente através de uma rede de conexão entre eles, a Internet. O conceito "coisa" exemplifica tudo que cria essa interação, para a partir dela gerar um processamento, que neste contexto podem ser os dispositivos. E, com isso, a troca de informações ocorre de forma mais rápida e eficiente (KOPETZ, 2011).

Esse conceito ocorreu a partir do desenvolvimento de diversas áreas, como por exemplo: sistemas embarcados, microeletrônica, redes e protocolos de comunicação e sensores. Com o passar do tempo, vem recebendo mais atenção tanto da academia quanto da indústria, devido ao seu potencial de uso nas mais diversas áreas, como por exemplo indústria e agricultura. (SANTOS et al., 2016)

Quando algo está conectado à internet, isso significa que o mesmo está apto a enviar ou receber informações, ou ambos. Essa capacidade de enviar e/ou receber informações torna as coisas "inteligentes". Pode-se usar como exemplo os *smartphones*: através deles os usuários são capazes de ouvir praticamente qualquer música do mundo, não porque seus *smartphones* possuem todas as músicas do mundo armazenadas nele, mas porque todas as músicas do mundo estão armazenadas em outro lugar, sendo que os *smartphones* podem enviar informações - pedindo por essa música - e receber informações - transmitindo essa música em seus *smartphones* (MIORANDI et al., 2012).

Para ser inteligente, uma coisa não precisa ter uma enorme capacidade de armazenamento ou um supercomputador em seu interior. Tudo o que uma coisa precisa fazer é se conectar a algum lugar com uma grande capacidade de armazenamento ou a um super computador. Na Internet das Coisas, todas as coisas que estão sendo conectadas à internet podem ser colocadas em três categorias: (i) as coisas que coletam informações e depois enviam; (ii) as coisas que recebem informação e depois agem nela, e (iii) as coisas que fazem ambas. E se houver combinação entre os três casos citados ocorre um aumento dos benefícios da solução aplicada (MIORANDI et al., 2012).

As aplicações em IoT se estendem por uma ampla variedade de casos de uso, agregando valor tanto nas coisas quanto nos dados coletados e transmitidos. No entanto, todos os sistemas IoT são similares, pois representam a integração de quatro componentes

distintos: sensores e dispositivos, conectividade, processamento de dados e uma interface de usuário. Os próximos tópicos serão compostos da descrição, o que cada um deles significa e como eles se juntam para formar um sistema IoT completo (KHAN et al., 2012).

2.1.1 Sensores e dispositivos

Primeiro, sensores ou dispositivos coletam dados do ambiente no qual estão inseridos, dados estes que podem ser simples como uma leitura de temperatura ou complexos como um vídeo. Eles são utilizados muitas vezes em conjunto: sensores são agrupados com dispositivos para criar sistemas mais complexos para coleta de dados, de forma que sua função seja específica e eficiente para a tarefa, e, com isso, é necessário utilizar diversos dispositivos e técnicas de eletrônica e computação para que o *hardware* do sistema IoT seja eficiente (KHAN et al., 2012).

2.1.2 Conectividade

Os dados coletados são enviados para a nuvem ou para algum tipo de plataforma de coleta, mas antes disso precisam de uma maneira de chegar até esses locais. Os sensores e dispositivos podem ser conectados a nuvens e plataformas através de uma variedade de métodos: tecnologia celular (2G, 3G, 4G, 5G), satélite, *WiFi*, *Bluetooth* e redes LPWAN. Cada uma dessas opções possui compromissos entre o consumo de energia, o alcance e a largura de banda. Para ser feita a escolha de qual opção de conectividade é melhor para determinada aplicação IoT, deve ser levada em conta diversos fatores tanto físicos quantos do ambiente, custo e aplicabilidade (KHAN et al., 2012).

2.1.3 Processamento de dados

Quando os dados coletados pela aplicação IoT chegam à nuvem ou plataforma, o *software* local realiza algum tipo de processamento. Este processamento pode ser algo simples, como verificar se a leitura da temperatura está dentro de um intervalo aceitável, ou mais complexo, como usar a visão computacional em vídeo para identificar objetos. Esta etapa é bem importante para a IoT pois, neste momento são agregados valores aos dados, os quais são processados de acordo com a necessidade da aplicação (KHAN et al., 2012).

2.1.4 Interface de usuário

Por fim, as informações que são úteis para o usuário final devem chegar a ele de alguma forma. Isso pode ser feito por meio de um alerta: *e-mail*, texto, notificação, SMS. Por exemplo, um *e-mail* pode ser enviado quando a temperatura estiver muito alta no armazenamento a frio de uma determinada empresa. A interface tem que ser uma via de mão dupla, já que o usuário também deve poder interagir e executar ações que influenciem

o sistema e a solução IoT, como ajustar a temperatura do aquecedor de sua casa sem estar nela, através de seu *smartphone* (KHAN et al., 2012).

Além disso, deve realizar funções automaticamente por meio de regras pré-definidas. Isso é bem importante quando relacionado a aplicações IoT na área de segurança. O sistema identifica um intruso em sua casa e, ao mesmo tempo, envia um alerta acionando as autoridades locais (KHAN et al., 2012).

2.2 Cidades Inteligentes

As Cidades Inteligentes são cidades projetadas para fazer um melhor uso dos recursos públicos, aumentar a qualidade dos serviços oferecidos aos cidadãos e reduzir os custos operacionais e administrativos. Tais características podem ser alcançadas através da implementação de soluções IoT no contexto urbano, ou seja, uma infra-estrutura de comunicação que ofereça acesso unificado, simples e econômico a uma infinidade de serviços públicos, desencadeando potenciais sinergias e aumentando a transparência para os cidadãos. (SCHAFFERS et al., 2011)

Soluções IoT implantadas em contextos urbanos, de fato, podem trazer diversos benefícios na gestão e otimização de serviços públicos tradicionais, como transporte, iluminação, segurança, manutenção de áreas públicas e educação. Outro ponto importante também é que a disponibilidade dos diversos tipos de dados coletados por soluções IoT podem ser utilizados para aumentar a transparência e conscientização, além de promover ações do governo atuante em relação às pessoas que vivem no local e a real situação da qual estão vivendo. Assim, é possível estimular a participação dos cidadãos na administração e manutenção dos centros urbanos e, também, impulsionar a criação de novos serviços que podem ser aplicados utilizando tecnologias que se encaixam dentro do conceito de IoT. (CUFF; HANSEN; KANG, 2008)

Com isso, a aplicação de soluções IoT dentro do conceito de Cidades Inteligentes é particularmente atraente, tanto para as administrações governamentais - que podem se tornar as primeiras a adotar essas tecnologias fazendo com que processos convencionais sejam mais eficientes - quanto para a população que sofre diretamente com as mudanças aplicadas.

Dentro do contexto do trabalho, que atua através da identificação de problemas de infraestrutura urbana, é utilizada atualmente em diversas cidades do país a ouvidoria, método que a população utiliza para fazer esse cadastramento. O cidadão que necessita fazer uma reclamação devido a algum desses problemas deve acessar o site da prefeitura e preencher um formulário dando todas as informações necessárias de identificação. Esse cadastro também é feito através do telefone, e-mail, pessoalmente ou por carta. E, depois disso, o cidadão acompanha a denúncia através do protocolo gerado. (OUVIDORIA DA

PREFEITURA DE SÃO PAULO, 2019)

Assim, com estes conceitos adquiridos é possível perceber que o SIPIU, que vem sendo apresentado neste trabalho, encontra-se dentro dos conceitos de Internet das Coisas e Cidades Inteligentes, já que ele utiliza diversas tecnologias que o caracterizam como uma solução IoT e faz com que um processo convencional sofra uma mudança de forma que seja mais eficiente dentro de um contexto urbano.

2.3 GPS

O sistema GPS (do inglês *Global Positioning System*) faz uso de satélites para saber o local no qual o receptor do sinal do satélite está naquele momento. O sistema tem como base uma rede de 24 satélites que ficam distribuídos em seis planos próximos a órbita da Terra, os quais são responsáveis por enviar sinais para o sistema GPS. E, a partir disso, o dispositivo GPS interpreta estes sinais informando a localização naquele momento, latitude e longitude (PARKINSON et al., 1996).

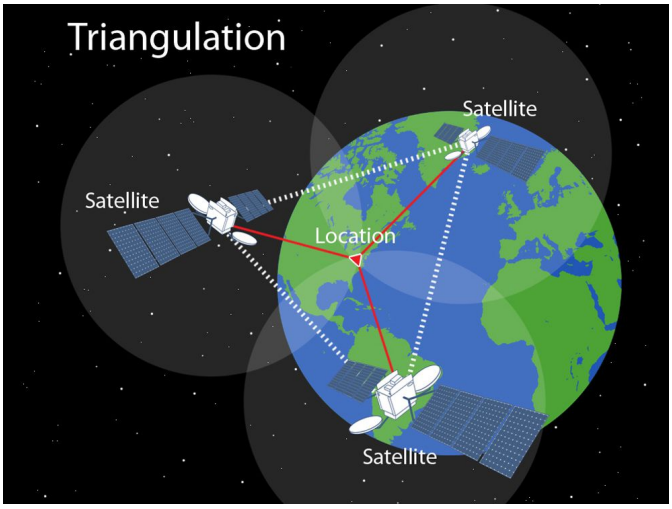
É possível dizer que o início da criação do sistema GPS foi em 1957, quando a União Soviética lançou o primeiro satélite e impulsionou estudos sobre como estes poderiam ser usados para obter localizações. Todavia, o sistema foi desenvolvido de forma concreta no projeto NAVSTAR, do Departamento de Segurança Americano, com início em 1960, e se tornou realmente funcional em 1995. Em 2000, as informações recolhidas que antes eram só de uso militar se tornaram públicas (MISRA; ENGE, 2006).

Os satélites e os receptores GPS usam como referência relógios atômicos que marcam as horas com uma incrível precisão de nano segundos. Quando o satélite emite um sinal para o sistema GPS, o horário em que ele saiu do satélite também é enviado. Os sinais enviados são sinais de rádio, que viajam a uma velocidade aproximada de 300.000 km/s no vácuo. O sistema GPS calcula qual o intervalo de tempo que o sinal demorou para chegar até ele e, consequentemente, é possível descobrir a localização. Como são enviados sinais constantemente, o sistema GPS sempre consegue determinar o local momentâneo, mantendo sua posição sempre atualizada (MISRA; ENGE, 2006).

Atualmente, os sistemas GPS fazem uso de uma técnica chamada triangulação, que determina a localização do receptor na Terra com precisão de aproximadamente 3 metros (GARMIN, 2019). Três satélites enviam o sinal para o GPS, que calcula quanto tempo cada sinal demorou a chegar nele. E, a partir desses 3 dados, é possível ter uma maior precisão da localização (MISRA; ENGE, 2006). A Figura 2 ilustra esse conceito. Nela, nota-se que o ponto de encontro dos limites de 3 satélites se refere à localização do dispositivo GPS. Além disso, devido à altitude e à velocidade de rotação da Terra, pequenos efeitos relativísticos aparecem nos relógios internos, que são então corrigidos pelas equações de Einstein da Relatividade Geral e Restrita (BAHDER, 2003). Sem essas correções, o tempo

nos satélites ficaria atrasado por cerca de 38 microssegundos diariamente, o que resultaria em erros de posicionamento de cerca de 10 km acumulados por dia.

Figura 2: Técnica de triangulação.



Fonte: (NATIONAL GEOGRAPHIC, 2018)

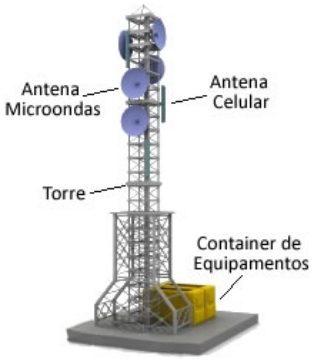
2.4 Redes de comunicação

Nos dois tópicos a seguir serão apresentados conceitos e definições em relação aos dois padrões de tecnologia de comunicação utilizados neste projeto.

2.4.1 Tecnologia 3G

Para contextualizar e ajudar no entendimento das principais caraterísticas da tecnologia 3G, primeiramente serão expostos alguns conceitos e tecnologias anteriores a ela. Inicialmente é necessário entender o funcionamento de um sistema de telefonia celular, que consiste no envio e recebimento de ondas de radiofrequência. Essas ondas são emitidas através das ERBs (Estações de Rádio Base), que possuem as antenas, conforme mostra a Figura 3. Essas antenas são responsáveis por enviar o sinal em uma pequena área chamada célula, da qual é proveniente o termo "celular"(SANTOS, 2006).

Figura 3: Exemplo de uma Estação de Rádio Base.



Fonte: (SANTOS, 2006)

As ERBs são controladas pela Central de Comutação e Controle (CCC). O processo geral se dá da seguinte forma: primeiro, ao realizar uma ligação ou envio de algum dado, o equipamento conectado se comunica com a ERB mais próxima a ele, e, após isso, a ERB encaminha a ligação ou os dados para a Central de Comutação e Controle. Dependendo do destino final desta informação, os dados são encaminhados para outra ERB ou outra CCC, de outras regiões, que então irão se comunicar com o aparelho receptor (MAIA et al., 2002).

A primeira tecnologia de comunicação relacionada à tecnologia celular foi a 1G - Primeira Geração - muito utilizada durante a década de 1980 pela rede AMPS (*Advanced Mobile Phone System*), constituída por sistemas analógicos e apenas transmissões de voz eram realizadas (SHUKLA et al., 2013).

Com o passar do tempo, a tecnologia 1G foi substutuída pela 2G - Segunda Geração. Popularizou-se durante a década de 1990, principalmente pela disseminação do uso de dispositivos celulares. O grande diferencial é que o sinal passou a ser digital ao invés de analógico, fazendo uso das tecnologias CDMA e GSM (SHUKLA et al., 2013).

A tecnologia CDMA (*Code Division Multiple Access*), por definição, é um método de acesso aos sistemas de comunicação, podendo ser utilizado tanto para telefonia celular quanto para o sistema GPS (SHUKLA et al., 2013). A tecnologia GSM (*Global System for Mobile Communications*) tem como um dos seus grandes diferenciais a utilização de criptografia ao realizar transmissão de dados, que faz com que aumente a segurança das informações trafegadas. Além disso, foi esta última tecnologia a responsável pelo início do uso de SIM *cards* nos celulares e dispositivos (SHUKLA et al., 2013).

A próxima tecnologia desenvolvida foi a 3G - Terceira Geração - a qual é a evolução da tecnologia 2G, cujo objetivo principal era a implementação de novos padrões de comunicações definidos pela UMTS (*Universal Mobile Telecommunication System*). A principal característica do 3G é o aumento significativo da capacidade de usuários de

serviços de voz e dados referente ao 2G (GARG; RAPPAPORT, 2001).

O GPRS (*General Packet Radio Service*), aplicado na tecnologia 3G, proporcionou o aumento na velocidade de transferência de dados, além de permitir ao usuário se manter sempre conectado à internet. Antes, quando era utilizado GSM, a conexão do dispositivo com a internet demorava entre 15 e 30 segundos, o que ficou mais rápido com o GPRS. Permitiu também que as operadoras realizassem a cobrança do serviço de internet de acordo com a quantidade de dados utilizados, ao invés de cobrar de acordo com o tempo que a pessoa permanecesse conectada, como era feito anteriormente (HALONEN; ROMERO; MELERO, 2004).

A tecnologia 3G é, atualmente, a rede em que a maioria da população mundial está conectada, apresentando grande capacidade de cobertura em centros urbanos e em algumas regiões rurais, e isso vem crescendo com o passar do tempo (GARG; RAPPAPORT, 2001). Apesar de seu custo de transmissão elevado e alto gasto energético, um dos grandes pontos positivos é que ela abrange uma vasta área de cobertura, levando a possibilidade de soluções IoT para diversos locais (GARCIA; KLEINSCHMIDT, 2017).

2.4.2 Tecnologia Sigfox

Como visto no tópico anterior, a rede celular 3G foi se adaptando para atender as necessidades atuais, dentre as quais aquelas voltadas a aplicações IoT. Porém, como essa rede possui limitações de custo e de gasto energético, foram realizados estudos para que houvesse um novo padrão de rede que atingisse os requisitos da IoT.

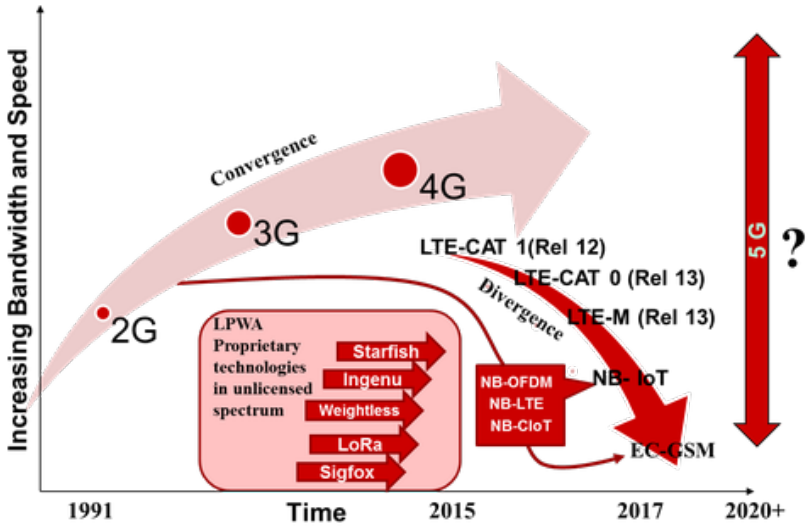
Observando a Figura 4, relata-se a evolução temporal relacionada ao surgimento de novas tecnologias para serem usadas em aplicações IoT com longo alcance. As redes celulares 2G, 3G e 4G ganham destaque em relação à velocidade e largura de banda. Mas, na área central do gráfico, nota-se a presença das redes LPWA, também chamadas de LPWAN (*Low-Power Wide-Area Network*) (LIGHTREADING, 2016).

Essas redes são um balanço de pontos extremamente importantes levados em conta na hora da confecção de um projeto IoT: se o alcance da rede é longo e atingirá os locais desejados, baixo consumo de energia, baixo custo de envio e recebimento de dados e também baixo custo de *hardware*. E estudos nas áreas calculam que com o uso dessas redes o número de dispositivos conectados pode chegar a bilhões de unidades em poucos anos (GARCIA; KLEINSCHMIDT, 2017).

Uma das redes LPWAN utilizadas atualmente é a Rede Sigfox, criada na França e que atualmente opera no Brasil através da operadora WND (WND, 2016). A tecnologia Sigfox apresenta um protocolo compacto e otimizado, oferecendo uma solução de comunicação baseada em software, na qual toda a complexidade da rede e da computação é gerenciada em nuvem e não nos dispositivos. A combinação de todas essas características

reduzem de forma significativa o consumo de energia e os custos dos dispositivos conectados (SIGFOX, 2019).

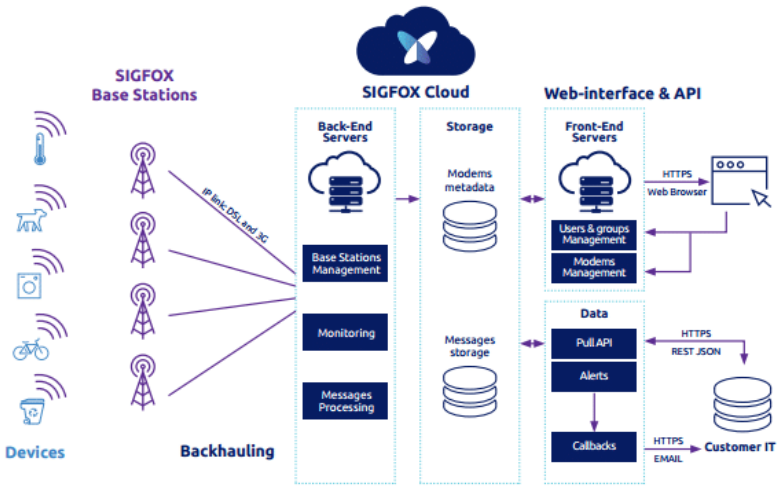
Figura 4: Relação entre aumento da largura de banda e velocidade de diversas redes usadas em IoT ao longo das últimas três décadas.



Fonte: (LIGHTREADING, 2016)

A arquitetura da rede Sigfox pode ser dividida em 4 elementos (vide Figura 5): (i) os dispositivos que contém o módulo Sigfox; (ii) as *Base Stations*, que funcionam de forma semelhante as ERBs; (iii) a Nuvem, na qual é realizado o armazenamento dos dados coletados, e (iv) a interface do usuário, que é como o usuário final tem acesso aos dados (SIGFOX, 2019).

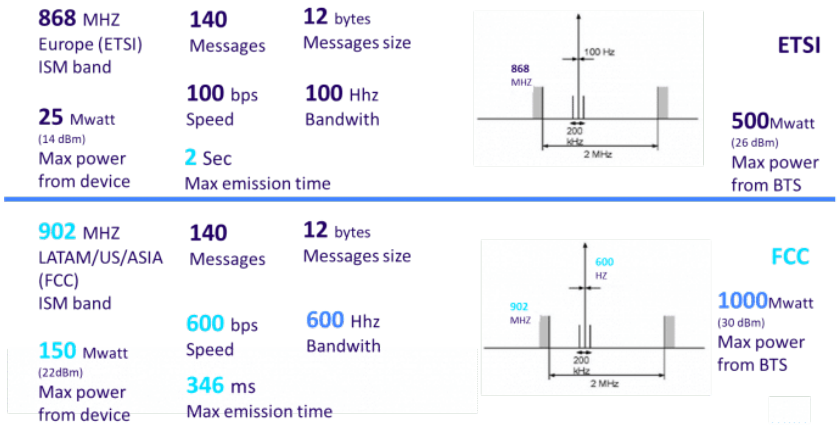
Figura 5: Arquitetura da rede Sigfox.



Fonte: (SIGFOX, 2019)

Em relação à transmissão de dados, a rede faz uso da tecnologia *Ultra Narrow Band*, a qual utiliza canais de 100 Hz de largura de banda nas regiões que seguem as normas dos órgãos reguladores ETSI e ARIB (Europa e Japão) e de 600 Hz na região que segue a norma do órgão regulador FCC (Américas e Oceania). Nas regiões ETSI os dados são transmitidos a 100 bps e nas regiões FCC a 600 bps (SIGFOX, 2019). As regiões FCC, na qual se encontra o Brasil, a frequência utilizada é a de 902 MHz, com largura de banda de 600 MHz, capacidade de 140 mensagens por dia por dispositivo e cada mensagem enviada pode ter o tamanho de até 12 *bytes* - Figura 6 (SIGFOX, 2019).

Figura 6: Comparação do uso na tecnologia *Ultra Narrow Band* nas regiões ETSI e FCC.



Fonte: (SIGFOX, 2019)

A tecnologia *Ultra Narrow Band* também se caracteriza por um uso otimizado da potência disponível, o que permite que os dispositivos Sigfox se comuniquem por longas distâncias de forma confiável, mesmo em canais com interferências e ruídos (SIGFOX, 2019).

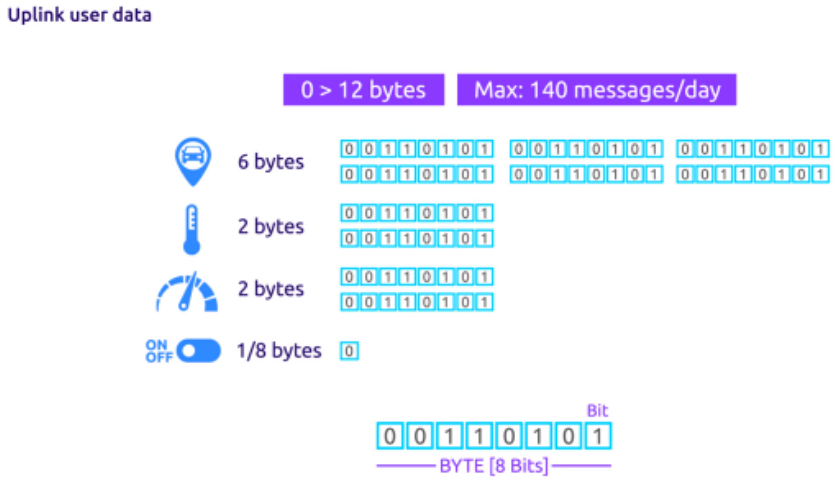
Um dos diferenciais da rede aqui estudada é o acesso aleatório (transmissão não sincronizada entre rede e dispositivo), recurso fundamental para alcançar uma alta qualidade de serviço. O acesso aleatório funciona da seguinte maneira: o dispositivo envia a mensagem em uma frequência aleatória e, em seguida, envia duas réplicas em diferentes frequências e tempo, criando diversidade de tempo e frequência. Através da diversidade criada, ocorre um grande aumento em relação à resistência do sistema a interferências. Um dos problemas evitados é quando uma mesma mensagem tenta vários caminhos, já que frequências e tempos diferentes garantem a recepção dos dados necessários (SIGFOX, 2019).

Diferente do que é feito em redes celulares, o dispositivo quando conectado à rede Sigfox não está vinculado a apenas uma estação base. Os dados enviados são recebidos pela estação mais próxima e também por mais outras três estações que se encontram na região, ocorrendo o conceito chamado de diversidade espacial. Os conceitos de diversidade

espacial e de acesso aleatório são os grandes diferenciais da rede Sigfox e influenciam de forma direta também na segurança dos dados (SIGFOX, 2019).

Como um dos diferenciais da rede Sigfox é o baixo custo energético, ele tem que ser obtido de alguma maneira, e a encontrada foi utilizar o protocolo de envio de mensagens pequenas, o que se encaixou bem em IoT, já que muitas vezes são sequências de dados curtos que são enviados. As mensagens ou pacote de dados enviados variam de 0 a 12 bytes (SIGFOX, 2019). Mesmo parecendo um pacote reduzido, observa-se na Figura 7 que é possível mandar diversos tipos de dados, como por exemplo localização através de um GPS com 6 *bytes*, abordado neste trabalho.

Figura 7: Exemplos de tamanho de possíveis mensagens a serem transmitidas.



Fonte: (SIGFOX, 2019)

Além de enviar mensagens, os dispositivos também precisam recebê-las. Na rede 3G, por ser celular, o processo de transmissão e recepção de informação é praticamente o mesmo. Entretanto, na rede Sigfox ela é um pouco diferenciada. Como esta rede é específica para aplicações IoT, não existe grande necessidade de recebimento de dados, exceção feita à mudança de algum tipo de configuração do dispositivo, ao ajuste da escala de algum sensor ou da frequência das mensagens, à solicitação de alguma informação diferente das que estão sendo enviadas e à atualização do *firmware*. Para isso, cada dispositivo tem disponível 4 mensagens de 8 *bytes* de capacidade por dia (SIGFOX, 2019).

3 MATERIAIS E MÉTODOS

3.1 Materiais

De forma a auxiliar a visualização do sistema foi criado um esquema em 3D, no programa *SketchUp*, do sistema físico a ser implementado no carrinho utilizado pelos garis, que pode ser visualizado na Figura 8. Foi decidido colocá-lo na lateral do carrinho para que não influenciasse no processo de deslocamento e/ou pudesse de alguma forma atrapalhar o trabalho do funcionário. E o sistema precisava também ficar acoplado de modo que fosse resistente o suficiente para sobreviver sem danos aos fatores externos e ambientais que pudesse sofrer durante a jornada de trabalho do funcionário.

Figura 8: Esquema 3D do sistema acoplado ao carrinho.

Fonte: Elaborado pela autora.

Na Figura 9 é possível observar o sistema de forma mais detalhada. O retângulo cinza é o *display* LCD, no qual aparece a lista de programas e informações pertinentes ao envio das informações requeridas. Os botões de coloração azul são para realizar a seleção e percorrer a lista de possíveis problemas e, por último, o botão vermelho é o botão de confirmação e de tomada de decisão, responsável principalmente pelo sinal que vai ser enviado ao sistema, quando necessário.

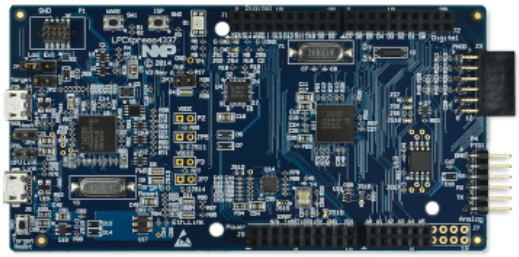
Figura 9: Esquema 3D detalhado do sistema acoplado ao carrinho.

Fonte: Elaborado pela autora.

3.1.1 Processador

O *hardware* é constituído pelo microcontrolador que tem função de processador. Para este projeto adotou-se a placa OM13070: LPCXpresso4337 *Development Board* da NXP, Figura 10.

Figura 10: Placa de desenvolvimento LPCXpresso4337 da NXP.

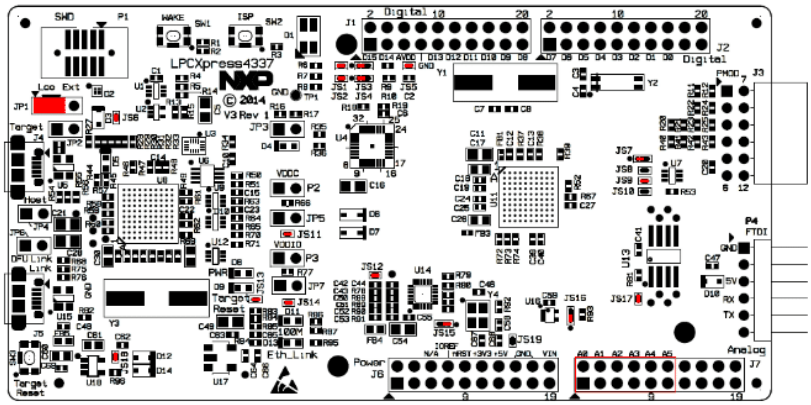


Fonte: (NXP, 2015)

A placa de desenvolvimento LPCXpresso4337 é uma placa de baixo custo, com diversos *headers* e conectores para que, de forma fácil, seja possível conectá-la a circuitos externos. O microcontrolador é o NXP LPC4337, um *dual core* com *core* ARM Cortex-M4 com ponto flutuante e um ARM Cortex M0+. O *core* Cortex-M4F pode operar até 208 MHz. A placa também contém um microcontrolador responsável pela interface de programação e *debug* da placa com o computador, com uma memória Quad SPI de 8 Mb, um Led RGB e um CI de Phy Ethernet (NXP, 2015).

A LPCXpresso4337 possui 4 conectores laterais e dois conectores micro USB. O conector micro USB J5 é utilizado para programar e debugar o microcontrolador. O conector micro USB J4 é a interface para a comunicação UART do microcontrolador. Os conectores podem ser observados na Figura 11 (NXP, 2015).

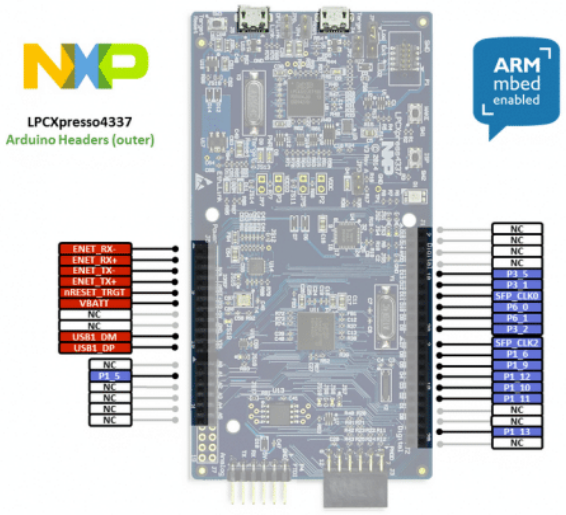
Figura 11: Esquemático da pinagem da placa de desenvolvimento LPCXpresso4337 da NXP.



Fonte: (NXP, 2015)

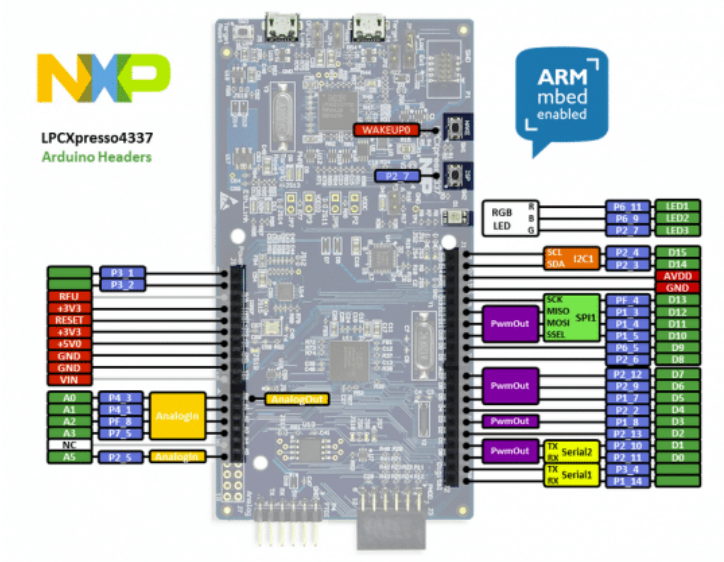
Nas Figuras 12 e 13 estão destacadas as pinagens dos conectores laterais da LPCXpresso4337. São quatro conectores duplos disponíveis nas laterais da placa, sendo a pinagem interior desses conectores compatível com o padrão Arduino. Na parte externa dos conectores estão disponíveis todos os pinos necessários para comunicação Ethernet (NXP, 2015).

Figura 12: Pinagem externa da placa de desenvolvimento LPCXpresso4337 da NXP.



Fonte: (NXP, 2015)

Figura 13: Pinagem interna da placa de desenvolvimento LPCXpresso4337 da NXP.

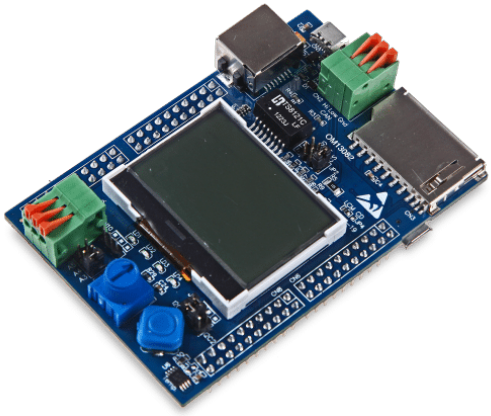


Fonte: (NXP, 2015)

Para fazer a compilação do software na placa é adotada a plataforma Mbed, que permite o desenvolvimento de projetos apenas utilizando o *browser*, sem a necessidade de instalar nenhum programa no computador. Através da plataforma Mbed é possível desenvolver o firmware, baixar e testar na placa apenas conectando a placa a USB do computador, sendo montada como um *drive*. Após isso realiza o *download* do arquivo binário compilado do *browser* e esse arquivo é adicionado a pasta correspondente a placa (MBED, 2019).

A placa de desenvolvimento LPCXpresso4337 será utilizada em conjunto com o seu *shield* de funcionalidades, a placa OM13082: LPC *General Purpose Shield* da NXP, Figura 14.

Figura 14: Placa OM13082 da NXP.



Fonte: (NXP - SUPPORT, 2015)

A OM13082 é uma placa de expansões, que possui vários periféricos e pode ser encaixada em uma outra placa da mesma família. Neste último caso, o usuário apenas precisa de um *firmware* para controlar os dispositivos conectados.

Seguem os periféricos da placa OM13082 (NXP - SUPPORT, 2015):

- LCD gráfico de 128x64;
- Sensor inercial, com integração a um acelerômetro de 3 eixos e um giroscópio de 3 eixos;
- Sensor I2C de temperatura;
- 4 Leds controlados via um expansor de IOs;
- *Joystick* de 5 posições;
- Potenciômetro;
- *Slot* para cartão SD;
- Conector *Ethernet* RJ45;
- Conector Micro USB que conecta direto aos pinos USB HOST do microcontrolador com proteção de ESD.

Ambas as placas abordadas neste tópico foram doadas pela empresa NXP devido a participação da aluna na competição citada anteriormente.

3.1.2 Módulo 3G

O módulo 3G escolhido foi o *Modem* 3G DD3G – 910 da Duodigit (Figura 15). O DD3G é um módulo de dados 3G robusto que pode ser utilizado em locais sujeitos a fatores ambientais e externos. Opera em 3G nas frequências UMTS|HSPA: 800/850, 900 e 2100 MHz, e em GSM/EDGE nas frequências 850, 900, 1800 e 1900 MHZ, atendendo todas as operadoras no Brasil. Alterna de forma automática para EDGE/GSM caso o sinal 3G não esteja disponível no momento a ser utilizado. O *Modem* possui interfaces serial RS232 e USB 2.0 de alta velocidade e também dispõe de alternância entre dois *SIM Cards*. Sua programação é realizada através da linguagem Python e de comandos AT (DUODIGIT, 2018).

O DD3G possui uma antena interna e três LEDs que informam a qualidade do sinal e se ele se encontra em atividade, o que facilita o processo de programação e configuração para a atividade proposta (DUODIGIT, 2018). A empresa Duodigit emprestou o módulo em questão para que fosse possível realizar os testes para validação da solução aqui proposta.

Figura 15: Módulo 3G DD3G – 910.

Fonte: (DUODIGIT, 2018)

3.1.3 GPS

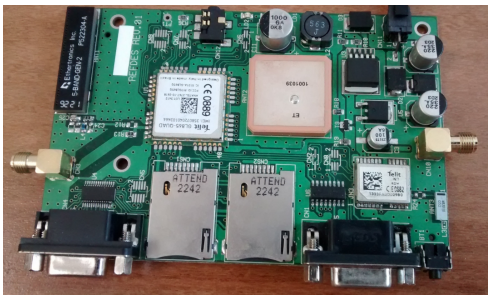
O dispositivo GPS utilizado neste trabalho foi o GPS JN3 da Telit (Figura 16), que também funciona como um sistema GPS tradicional. Seu grande diferencial é que, por ser bem compacto, pode ser incorporado a projetos de forma mais fácil e rápida (TELIT, 2017).

Figura 16: GPS JN3.

Fonte: (TELIT, 2017)

Como este trabalho tem objetivo de desenvolver um protótipo, foi utilizado o GPS JN3 junto com a sua placa de desenvolvimento EVK-JN3 (Figura 17) para obtê-lo de forma mais rápida e poder validar a proposta do projeto. A empresa Telit doou o módulo em questão para que fosse possível realizar os testes para validação da solução aqui proposta.

Figura 17: Placa de desenvolvimento EVK-JN3.



Fonte: Foto obtida pela autora.

3.1.4 Módulo Sigfox

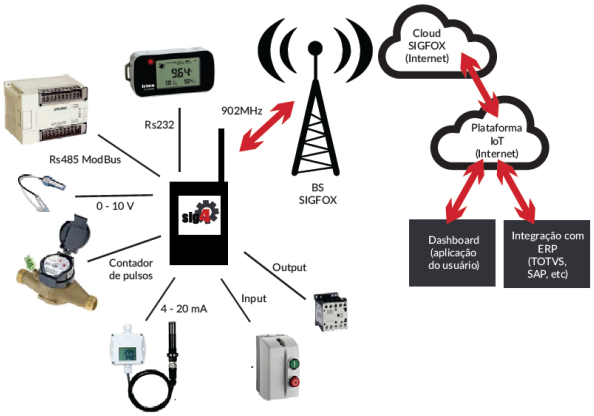
Um dos objetivos deste trabalho é comparar o funcionamento do SIPIU utilizando duas tecnologias de comunicação, a 3G e a Sigfox. Para a tecnologia Sigfox o módulo 3G foi substituído pelo *Gateway* sig4 (Figura 18).

Figura 18: *Gateway* sig4.

Fonte: (DUODIGIT, 2018)

O *Gateway* tem o objetivo de conectar sensores de diversos tipos através da rede LPWAN Sigfox. O sig4 possui uma versatilidade de conexões com RS232, RS485 (Modbus), 4 a 20 mA, 0 a 10 Volts, Inputs e Outputs (Figura 19). Os dados coletados são enviados para uma plataforma IoT, a TagoIO, na qual são tratados e armazenados (DUODIGIT, 2018).

Figura 19: *E*squema de funcionalidades do Gateway sig4.



Fonte: (DUODIGIT, 2018)

O *gateway* possui um módulo Sigfox, uma antena *on-board* e já vem configurado para o envio de dados para a rede Sigfox, sendo necessário apenas conectá-lo ao processador a ser utilizado e enviar os comandos AT. Outro ponto positivo é que, como o sig4 usa a rede Sigfox, quando ocorre o envio de um dado automaticamente através da triangulação

entre as *Base Stations*, é possível obter a localização da transmissão (DUODIGIT, 2018). A empresa Duodigit emprestou o módulo em questão para que fosse possível realizar os testes para validação da solução aqui proposta.

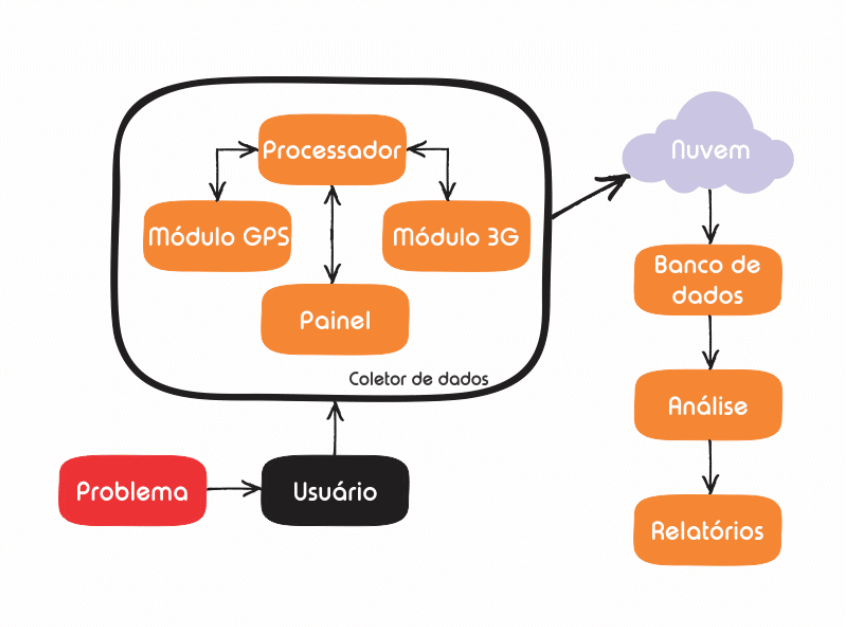
3.2 Métodos

Para síntese e visualização do desenvolvimento do projeto em suas mais diversas etapas, bem como a dinâmica de seu funcionamento, foi elaborado um diagrama de blocos inicial, como mostra a Figura 20.

O usuário, que no caso é um gari, quando identifica um problema de infraestrutura aciona o sistema através de 3 botões e um *display* LCD: dois botões de seleção que percorrem uma lista de possíveis problemas e um outro botão que faz o acionamento do envio.

Dentro da caixa acoplada ao carrinho se encontram o processador, um módulo GPS e um módulo 3G. O primeiro é responsável pela tomada de decisão e acionamentos dos dois outros equipamentos; o GPS envia a data e coordenadas do local no qual o sistema foi acionado e o módulo 3G pega esse pacote de informações e os envia para a Nuvem. É ali, ou através de uma Plataforma IoT, que os dados coletados serão armazenados formando um banco de dados, para posterior análise e geração de relatórios que servirão de base para os órgãos responsáveis pela manutenção e reparo dos problemas de infraestrutura urbana identificados através do sistema.

Figura 20: Diagrama de blocos inicial do sistema.



Fonte: Elaborado pela autora.

3.2.1 Tecnologia 3G

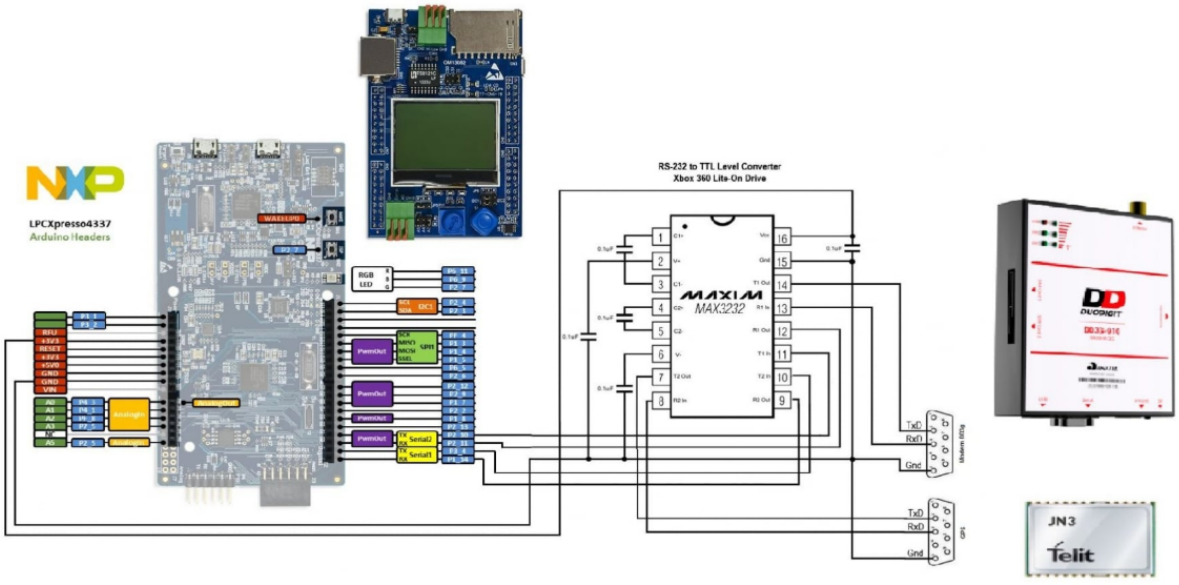
Nesta seção será apresentado o desenvolvimento do sistema na sua primeira versão, a que fazia uso da tecnologia 3G de comunicação. E também serão apresentados todos os equipamentos utilizados e suas principais características.

3.2.1.1 Montagem do *Hardwrare*

Definidos os parâmetros de funcionamento do sistema, foi feito um estudo de viabilidade da comunicação entre o microcontrolador OM13070: LPCXpresso4337 - que possui duas seriais - e os módulos 3G e GPS. Para realizar essa ligação foi utilizado um conversor, o MAX3232, o qual converte sinais do tipo RS232 em sinais de nível Serial de nível TTL para facilitar a comunicação entre o componentes deste sistema. O esquema elétrico de montagem do *hardware* pode ser visto na Figura 21.

Em seguida, foram conectadas as placas OM13082: LPC *General Purpose Shield* com a placa OM13070: LPCXpresso4337, sendo possível obter novas funções a partir desta junção, dentre as quais vale ressaltar o LCD, para visualização dos resultados e parâmetros importantes no projeto. Para a seleção destes parâmetros fisicamente, pelo usuário da interface, foi utilizado o *joystick* para manuseio das opções disponibilizadas na tela.

Figura 21: Esquema elétrico da montagem do *hardware*.

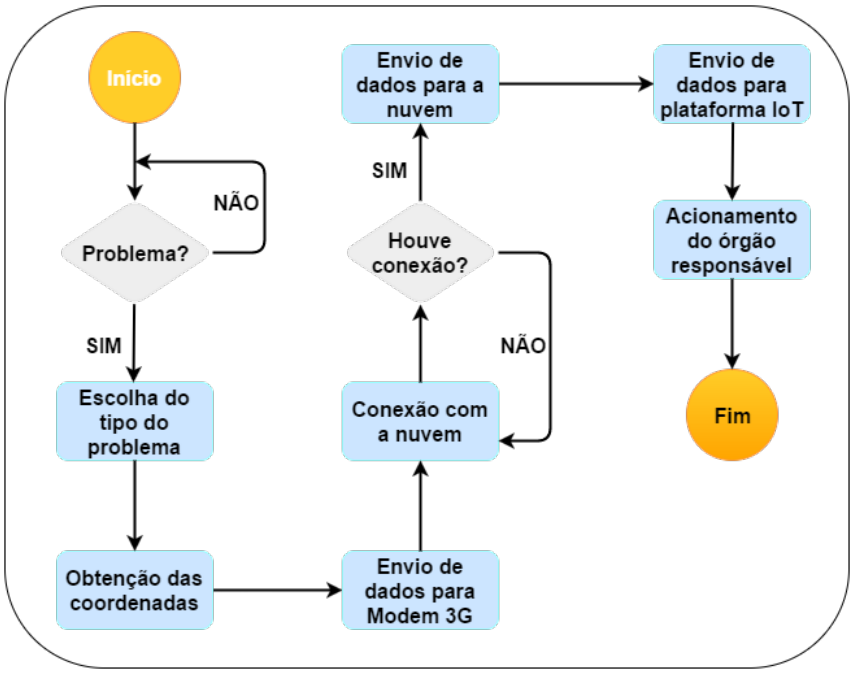


Fonte: Elaborada pela autora.

3.2.1.2 *Software*

Para o *software* no processador foi empregado o algoritmo que obedece o diagrama de blocos funcional, apresentado na Figura 22. Para a sua aplicação em linguagem C foi utilizada a plataforma Mbed.

Figura 22: Diagrama de blocos funcional do *software* implementado.



Fonte: Elaborado pela autora.

Como é possível observar na Figura 22, o início do algoritmo acontece quando um problema é encontrado; caso o que for encontrado ainda não for um problema, o algoritmo volta ao início. Se caso for um problema, o usuário necessita escolher - dentre as opções oferecidas - a categoria na qual o problema encontrado se encaixa. Com isso realizado e confirmado pelo usuário, o processador lê através da porta serial as coordenadas do local, e posteriormente o algoritmo junta o tipo do problema e as coordenadas em uma *string* e envia através de comunicação serial para o módulo 3G.

O código compilado no processador necessita fazer uma série de inicializações, declarar as bibliotecas a serem utilizadas, iniciar o *display* LCD e definir os pontos de início da escrita, iniciar o *joystick* e também inicializar os parâmetros necessários para realizar as comunicações seriais, tanto de envio quanto de recebimento. Para ajudar na visualização e escolha dos problemas listados no sistema, foi criada uma função que coloca uma espécie de seta que acompanha a lista de problemas de acordo com a seleção feita através do *joystick*. E, de acordo com as ações que forem sendo realizadas, o sistema se reinicia ou dá continuidade ao processo de coleta e envio de dados do GPS e módulo 3G, respectivamente. No Anexo A é possível analisar o código completo com comentários, e é possível observar passo a passo o que foi realizado.

O módulo 3G foi programado utilizando a linguagem Python e comandos AT, sendo necessário realizar dois códigos. O primeiro é responsável pela inicialização do módulo 3G, para testar a conexão local e ver se o módulo se encontra preparado para receber dados

para serem transmitidos: realiza uma série de inicializações e definições para testes de conexão, dentre elas a busca pela operadora do *SIM card* utilizado e teste para ver se recebe resposta após envio de um estímulo. Após a confirmação da conexão, o módulo recebe de volta um endereço de IP, e para começar a transmitir e receber dados, é necessário registrar o módulo com o seu nome de usuário. Este código também é responsável pelo envio dos dados do sistema para a nuvem ou plataforma IoT que estiver conectada. O código completo se encontra no Anexo B deste trabalho .

Já o segundo código compilado no módulo 3G é responsável pela conexão do módulo 3G com a plataforma IoT que foi utilizada. A plataforma em questão é a *DeviceWise*, da Telit. Nela é possível cadastrar seu dispositivo e obter um *token*, que funciona como uma espécie de senha, e deve ser utilizado quando se deseja conectar a plataforma e realizar o armazenamento dos dados coletados (TELIT IOT PORTAL, 2019). No código, que encontra-se no Anexo C, é possível observar que são definidos inicialmente o *token* e os parâmetros da comunicação serial. Ele se relaciona com o código relatado anteriormente (Anexo C) para checar se caso a conexão se encontra em situação de uso. Após isso, é feita a conexão com a Plataforma IoT escolhida e são enviados alguns dados para confirmar se o envio de dados pode ser realizado. E, com isso, os dados são enviados a Plataforma.

3.2.2 Tecnologia Sigfox

Para utilizar a tecnologia Sigfox o processador foi mantido, o módulo 3G foi substituído pelo *Gateway* sig4 e, devido à funcionalidade de obtenção de localização através da rede Sigfox, o GPS não será mais necessário.

O processador e o *gateway* sig4 foram conectados através de comunicação serial. Por já conter um módulo Sigfox é necessário apenas enviar alguns simples comandos AT para a sua configuração. E depois disso, o módulo Sigfox está pronto para receber os dados que necessitam ser enviados para a Nuvem. O código compilado no processador sofreu poucas mudanças, apenas em relação à configuração do módulo 3G, adição de comandos AT para envio de dados e remoção da seção que envolvia o GPS. É possível observar essas alterações realizadas no código do compilador no Anexo D, ao final deste trabalho.

Outra alteração em relação aos testes realizados utilizando tecnologia 3G foi a dispensa da Plataforma IoT *DeviceWise*, já que a tecnologia Sigfox oferece uma Nuvem na qual o usuário pode acessar os dados transmitidos. Mas, para melhor visualização para o usuário, foi utilizada a Plataforma IoT TagoIO, a qual já possui integração com o *Gateway* sig 4.

A plataforma agora utilizada dispõe de criação de *Dashboards* interativos e de fácil visualização, não requerendo outra interface para o usuário final, como um *site*. Outro grande diferencial e que a torna mais acessível é que a plataforma possui um aplicativo, e com ele o usuário consegue acompanhar o sistema (TAGOIO, 2019). Para fazer a conexão

entre a Plataforma IoT e o *gateway* sig4 é necessário cadastrá-lo como um dispositivo e inserir o seu Sigfox ID (código de identificação), obtido através da Nuvem da Sigfox após cadastro do dispositivo.

4 RESULTADOS E DISCUSSÃO

O módulo 3G se conecta ao *DeviceWise* - a plataforma IoT - como visto anteriormente. A Figura 23 é um *printscreen* do *Dashboard* criado para o sistema. Na plataforma os dados são tratados, facilitando o acompanhamentos daqueles coletados e separados em categorias de acordo com as necessidades da solução. No caso, foram criadas três variáveis: latitude, longitude e *status*, que é o tipo de problema cadastrado.

De forma a facilitar a interação do usuário final com os dados, que até então ficavam centrados na plataforma IoT *DeviceWise*, e dada a simplicidade para pegar os dados e expô-los, foi desenvolvido um *site* responsivo, utilizando HTML, CSS e Javascript, no qual é possível, de forma fácil e rápida, visualizar dados coletados.

Figura 23: *Printscreen* da Plataforma IoT DeviceWise.

Fonte: (TELIT IOT PORTAL, 2019)

A Figura 24 representa o *status* inicial do *site*, no qual se encontra o problema detectado, latitude, longitude, sua localização no mapa e a data da última coleta de dados realizada. Na Figura 25 é possível observar a parte inferior do *site*.

O usuário final também pode acessar uma tabela na qual tem a listagem do histórico das coletas realizadas, como visto na Figura 25. Desta forma, uma interface para o usuário facilita com que ele possa encaminhar as informações para o órgão responsável e que sejam tomadas as providências necessárias.

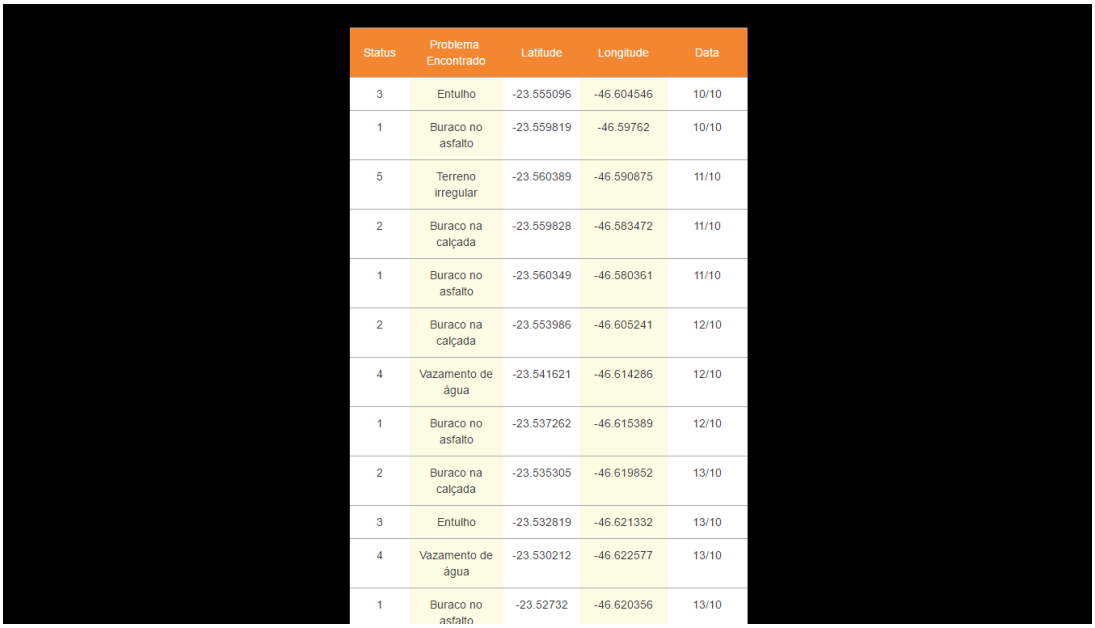
Figura 24: *Printscreen* da visão superior do *site* do sistema.

Fonte: Elaborado pela autora.

Figura 25: *Printscreen* da visão inferior do *site* do sistema.

Fonte: Elaborada pela autora.

Figura 26: *Printscreen* do *site* do sistema.

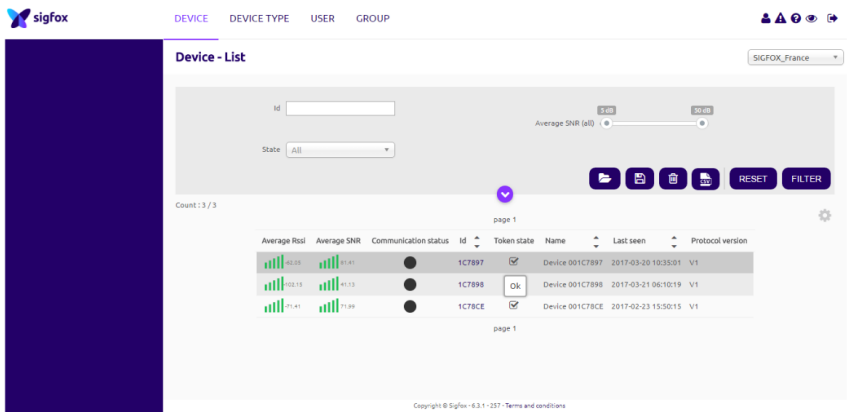


Status	Problema Encontrado	Latitude	Longitude	Data
3	Entulho	-23.555096	-46.604546	10/10
1	Buraco no asfalto	-23.559819	-46.59762	10/10
5	Terreno irregular	-23.560389	-46.590875	11/10
2	Buraco na calçada	-23.559828	-46.583472	11/10
1	Buraco no asfalto	-23.560349	-46.580361	11/10
2	Buraco na calçada	-23.553986	-46.605241	12/10
4	Vazamento de água	-23.541621	-46.614286	12/10
1	Buraco no asfalto	-23.537262	-46.615389	12/10
2	Buraco na calçada	-23.535305	-46.619852	13/10
3	Entulho	-23.532819	-46.621332	13/10
4	Vazamento de água	-23.530212	-46.622577	13/10
1	Buraco no asfalto	-23.52732	-46.620356	13/10

Fonte: Elaborada pela autora.

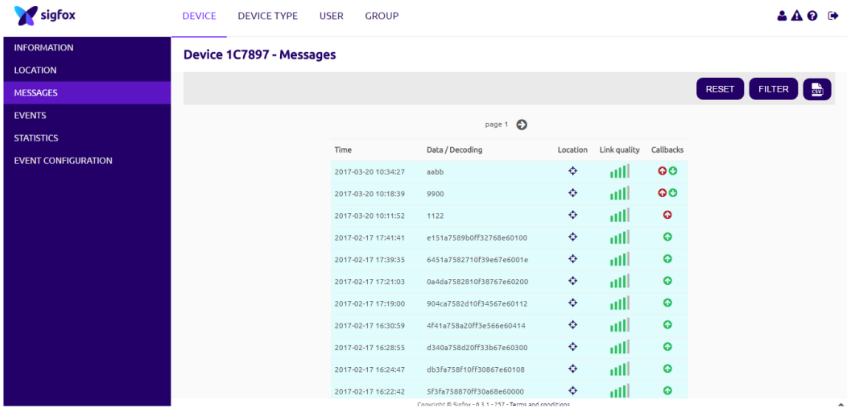
Agora serão expostos os resultados obtidos através da substituição da rede 3G pela rede Sigfox. Como a tecnologia Sigfox dá ao usuário acesso a sua nuvem, ela o permite cadastrar os dispositivos conectados e verificar se eles se encontram conectados à rede, como pode ser visto na Figura 27. Adicionalmente, em cada dispositivo é possível ver as mensagens enviadas por ele para a nuvem, conforme visto na Figura 28.

Figura 27: *Printscreen* do *back end* da tecnologia Sigfox dos dispositivos do usuário.



Fonte: (SIGFOX BACKEND, 2019)

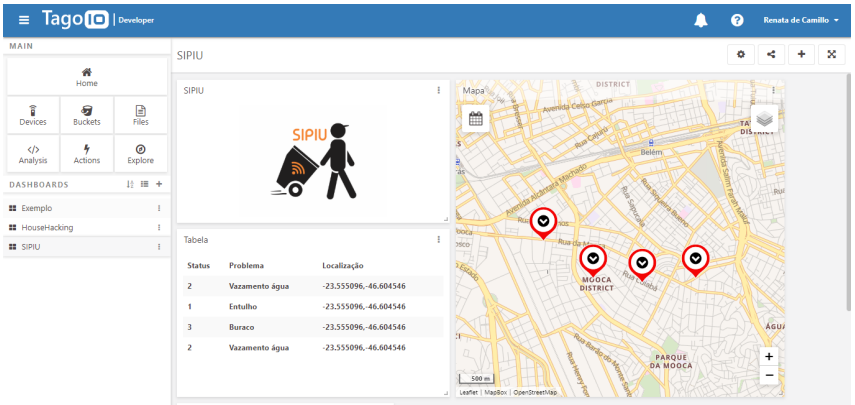
Figura 28: *Printscreen* do *back end* da tecnologia Sigfox das mensagens enviadas pelo dispositivo.



Fonte: (SIGFOX BACKEND, 2019)

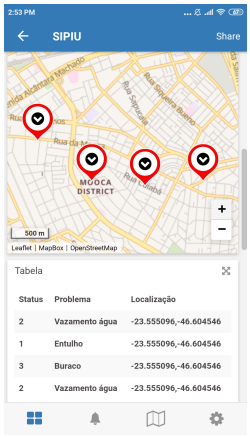
Como a visualização através do *Back end* não é muito simples, foi adotada a Plataforma IoT TagoIO. Como essa plataforma é mais limpa e fácil de ser utilizada em relação à plataforma *DeviceWise*, ela se tornou a interface para o usuário final. Através dela cria-se uma *Dashboard* com diversos *widgets* de forma a facilitar a visualização dos dados. Pode-se observar na Figura 29 que os dados coletados vão sendo salvos e visualizados em forma de tabela e também em pontos em um mapa dinâmico. Além de acesso pelo *browser*, é possível também acessar a *Dashboard* através do aplicativo (Figura 30). E, para o usuário final, como por exemplos as prefeituras, exportam-se planilhas com todos os dados.

Figura 29: *Printscreen* do *Dashboard* do projeto na plataforma TagoIO.



Fonte: (TAGOIO, 2019)

Figura 30: *Printscreen* do aplicativo com a *Dashboard* do projeto na plataforma TagoIO.



Fonte: Obtido pela autora.

Em relação aos *hardwares* utilizados, a grande diferença se dá em relação ao não uso do GPS quando utilizada a tecnologia Sigfox. Apesar de ambos provarem sua funcionalidade e atingirem seus objetivos, o *hardware* que utilizou a tecnologia Sigfox é mais simples e tem dimensões menores, sendo mais fácil de adaptá-lo ao protótipo do sistema.

Em relação à viabilidade econômica, foram calculados o custo do sistema com ambas as tecnologias, sendo primeiro apresentado o cálculo com tecnologia 3G: estudos preliminares para uma pequena produção mostram que os custos seriam em torno de R$200,00 para a placa (Figura 31), que integraria o processador, o módulo 3G e o dispositivo GPS. A caixa para proteção contra influências ambientais externas tem custo aproximado de R$40,00 e a bateria por volta de R$25,00, resultando em um total de R$265,00 para o *hardware* (SIPIU, 2016). O custo da Plataforma *Devicewise* é estimado em torno de R$10,00 por mês por dispositivo (TELIT IOT PORTAL, 2019), com um custo de telecomunicação avaliado em torno de R$15,00 por mês para cada unidade.

Considerando uma depreciação de 2 anos do *hardware*, com financiamento de 24 meses também a um juros de 2% ao mês (incluindo correção monetária) e o custo de consumo energético, cada equipamento de R$265,00 terá um custo de R$14,00 mensais (SIPIU, 2016). E o custo total mensal junto à Plataforma IoT e ao custo de telecomunicação resultante é de R$39,00 por mês para a tecnologia 3G. Se for considerado que o equipamento opera por 22 dias por mês, 8 horas por dia, resulta em um custo aproximado de R$1,77 por dia e R$0,22 por hora. Fazendo uma suposição de que o varredor acionasse o sistema e enviasse dados duas vezes por hora, é possível considerar que o custo por envio será de aproximadamente de R$0,10.

Agora, fazendo o cálculo para o uso da tecnologia Sigfox, temos que levar em consideração que o *hardware* (Figura 31), o custo de envio de dados e o gasto energético é reduzido em relação ao sistema usando tecnologia 3G. Assim, tem-se que o *hardware* fica

em torno de R$200,00 e o custo de telecomunicação por volta de R$8,00. Fazendo todos os cálculos anteriores, obtêm-se agora o custo de R$0,05 por envio de dados (SIPIU, 2016).

Figura 31: Tabela comparativa das tecnologias Sigfox e 3G.

Sigfox	ITEM	3GPP / Celular
Simples, Aberto	Sistema de Rádio	Proprietário MUITO complexo
Não Orientado a Conexão	Conexão	Orientado a Conexão
Uplink	Otimizada para	Downlink
Banda Não Licenciada	Espectro	Banda Licenciada
Sobre Internet pública, Meio não dedicado	Backhaul	TCP/IP (Rádio ou Fibra) Meio Dedicado
<3Kgs	Rádio Base	>=20Kgs
Omni directional (60cm)	Antena	Direcional, Setorizada, Painéis
Simples – 3 entidades	Arquitetura	Complexa – Multientidade
Sem Patentes	Propriedade Intelectual Dispositivo	Patentes
US$2.60	Preço Modem	US$15.00

Fonte: (WND, 2016)

Ambos os custos aproximados de transmissão de dados encontrados são baixos se for considerado que as informações geradas podem trazer redução de custos do meio urbano e aumento de produtividade. Usando como base a tecnologia 3G, que é mais cara em relação à tecnologia Sigfox, e que a cidade de São Paulo possui 13 mil garis (PREFEITURA DE SÃO PAULO, 2017), tomando como base que cada um opera com o seu carrinho, por mês o investimento total seria de R$507.000,00.

Tendo em vista que os problemas de infraestrutura são muito presentes, como por exemplo buracos no asfalto, segundo a Prefeitura de São Paulo, é estimado que surjam mil buracos diariamente, enquanto que em dias de chuva esses números dobram (PREFEITURA DE SÃO PAULO, 2017). Assim, como consequência, ocorrem inúmeros acidentes tanto com os motoristas quanto com os automóveis, e as vítimas processam os órgãos responsáveis, em sua maioria as prefeituras. Segundo a OAB, em média cada caso recebe, de indenização, R$15.000,00, valor esse que permite fechar por volta de mil buracos (OAB, 2019).

Segundo o Hospital das Clínicas de São Paulo, por dia surgem 300 buracos em calçadas (HC-USP, 2018) e, por ano, 171.000 pessoas sofrem quedas como pedestres (IPEA, 2019). O custo médio de resgate e tratamento de cada um dos pedestres está estimado em R$2.656,00, e o custo total do resgate e tratamento por ano fica em aproximadamente R$500 milhões. Para completar o cálculo da estimativa dos custos sociais das quedas de pedestres, é preciso aplicar um fator de multiplicação de 4,52 para acrescentar perda de

produção e reabilitação, resultando em R$2.260 bilhões (IPEA, 2019).

Assim, considerando que o envio de dados para cada problema utilizando tecnologia 3G custa aproximadamente R$0,10 e para a tecnologia Sigfox custa aproximadamente R$0,05, comparando-se as duas tecnologias se observa que, se caso no local tiver cobertura da rede Sigfox, ele é mais viável economicamente em relação à 3G, principalmente pelo custo de envio dos dados e uso reduzido da bateria.

E, por fim, comparando o método tradicional de cadastro de problemas de infraestrutura urbana e levando em conta os gastos associados a eles, como os citados acima, pode-se avaliar o projeto como viável, já que seria benéfico para todas as partes: Prefeituras, órgãos responsáveis e a população.

5 CONCLUSÃO

O SIPIU se diferencia do sistema tradicional, pois através do rápido cadastramento dos problemas de infraestrutura urbana e do uso de tecnologia IoT, é possível realizar uma antecipação da solução e um planejamento mais eficiente; assim, órgãos responsáveis, como prefeitura e subprefeituras, cortariam gastos desnecessários e otimizariam o tempo de resolução dos problemas.

Outro fator importante é que não seria necessário a contratação de novos funcionários, apenas a capacitação dos varredores de rua, profissionais que têm contato direto com os problemas e a população, identificando detalhes que outros não notariam. E também a qualidade dos dados coletados será superior se comparada às denúncias feitas pela população. Além disso, o *hardware* do sistema pode ser acoplado a diferentes objetos dependendo da cidade e suas necessidades.

Em relação às tecnologias de comunicação aplicadas, ambas funcionaram de acordo com o planejado e obtiveram resultados satisfatórios (Capítulo 4), mas a tecnologia Sigfox se destacou em relação à tecnologia 3G, principalmente nas questões de simplificação de *hardware*, custo reduzido de transmissão e baixo gasto energético.

Assim, o sistema aplicado é mais eficiente e rápido do que o método atual, pelo fato de possuir um sistema de IoT, a localização exata e o tipo de problema encontrado ser enviado diretamente para a Plataforma IoT que irá para o banco de dados do órgão responsável, agilizando todo o processo e o conserto ou reparo. Portanto, acaba influenciando de forma direta na melhora da mobilidade urbana e em questões socioambientais, já que aumentaria de forma considerável a qualidade de vida e satisfação da população.

Para os próximos passos do projeto seria interessante a transformação do protótipo em um equipamento final e validação real de seu funcionamento e aplicabilidade.

REFERÊNCIAS

BAHDER, T. B. Relativity of gps measurement. **Physical Review D**, APS, v. 68, n. 6, p. 063005, 2003.

CUFF, D.; HANSEN, M.; KANG, J. Urban sensing: out of the woods. **Communications of the ACM**, v. 51, n. 3, p. 24, 2008.

DUODIGIT. **Duodigit - Soluções em Telemetria**. 2018. Disponível em: <https://duodigit.com.br/produtos>. Acesso em: 29 abr. 2019.

EMBARCADOS. **Embarcados Contest - Conectando à Internet das Coisas com NXP**. 2016. Disponível em: <https://contest.embarcados.com.br/conectando-a-internet-das-coisas-com-nxp-2016/>. Acesso em: 23 nov. 2018.

GARCIA, P. S. R.; KLEINSCHMIDT, J. H. **Tecnologias Emergentes de Conectividade na IoT: Estudo de Redes LPWAN**. [S.l.]: sn, 2017. 1009–1013 p.

GARG, V. K.; RAPPAPORT, T. S. **Wireless network evolution: 2G to 3G**. [S.l.]: Prentice Hall PTR, 2001.

GARMIN. **Precisão do GPS em leituras de posição, distância e velocidade nos dispositivos Outdoor portáteis**. 2019. Disponível em: <https://support.garmin.com/pt-BR/?faq=P3DdzRfgik3fky125aHsFA>. Acesso em: 06 jun. 2019.

HALONEN, T.; ROMERO, J.; MELERO, J. **GSM, GPRS and EDGE performance: evolution towards 3G/UMTS**. [S.l.]: John Wiley & Sons, 2004.

HC-USP. **Hospital das Clínicas de São Paulo**. 2018. Disponível em: <https://sites.google.com/hc.fm.usp.br/hcfmusp/>. Acesso em: 04 mai. 2019.

IPEA. **Instituto de Pesquisa Econômica Aplicada**. 2019. Disponível em: <http://www.ipea.gov.br/portal/>. Acesso em: 04 mai. 2019.

KHAN, R. et al. Future internet: the internet of things architecture, possible applications and key challenges. In: IEEE. **2012 10th international conference on frontiers of information technology**. [S.l.], 2012. p. 257–260.

KOPETZ, H. Internet of things: Real time systems. **Real Time Systems Series**, p. 307–323, 2011.

LIGHTREADING. **How IoT Forked the Mobile Roadmap**. 2016. Disponível em: <https://www.lightreading.com/iot/iot-strategies/how-iot-forked-the-mobile-roadmap/a/d-id/720262>. Acesso em: 05 mai. 2019.

MAIA, C. C. et al. Apoio multicritério à decisão da escolha da localização de uma estação rádio base. **XXII Encontro Nacional de Engenharia de Produção**, 2002.

MBED. **Mbed**. 2019. Disponível em: <https://www.mbed.com/en/>. Acesso em: 02 mai. 2019.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. **Ad hoc networks**, Elsevier, v. 10, n. 7, p. 1497–1516, 2012.

MISRA, P.; ENGE, P. Global positioning system: signals, measurements and performance second edition. **Massachusetts: Ganga-Jamuna Press**, 2006.

NATIONAL GEOGRAPHIC. **Triangulation - National Geographic**. 2018. Disponível em: <https://www.nationalgeographic.org/photo/triangulation-sized/>. Acesso em: 02 mai. 2019.

NÓBREGA, F. A. R. et al. Infraestrutura urbana: infraestrutura e o crescimento populacional no brasil. **Caderno de Graduação-Ciências Exatas e Tecnológicas-UNIT**, v. 1, n. 2, p. 19–25, 2013.

NXP. **NXP - OM13070: LPCXpresso4337 Development Board**. 2015. Disponível em: <https://www.nxp.com/support/developer-resources/evaluation-and-development-boards/lpcxpresso-boards/lpcxpresso4337-development-board:OM13070>. Acesso em: 23 nov. 2018.

NXP - SUPPORT. **NXP - OM13082: LPC General Purpose Shield for LPCXpresso boards**. 2015. Disponível em: <https://www.nxp.com/support/developer-resources/evaluation-and-development-boards/lpcxpresso-boards/lpc-general-purpose-shield-for-lpcxpresso-boards:OM13082>. Acesso em: 23 nov. 2018.

OAB. **OAB**. 2019. Disponível em: <http://www.oabsp.org.br/>. Acesso em: 04 mai. 2019.

OUVIDORIA DA PREFEITURA DE SÃO PAULO. **Prefeitura de São Paulo - Fale com a ouvidoria**. 2019. Disponível em: <https://www.prefeitura.sp.gov.br/cidade/secretarias/ouvidoria/fale_com_a_ouvidoria/index.php?p=227268>. Acesso em: 04 abr. 2019.

PARKINSON, B. W. et al. **Global positioning system: Theory and applications, Volume II**. [S.l.]: American Institute of Aeronautics and Astronautics, 1996.

PREFEITURA DE SÃO PAULO. **Prefeitura de São Paulo - Varrição de Ruas e Limpeza Pública**. 2017. Disponível em: <https://www.prefeitura.sp.gov.br/cidade/secretarias/inovacao/varricao_de_ruas/index.php?p=4638/>. Acesso em: 23 nov. 2018.

SANTOS, B. P. et al. Internet das coisas: da teoria à prática. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos**, 2016.

SANTOS, M. Sao paulo: a growth process full of contradictions. Tokyo Japan United Nations University Press 1996., 1996.

SANTOS, P. A. B. V. dos. **USO DA ENERGIA SOLAR PARA ALIMENTAÇÃO DE ESTAÇÕES RÁDIO BASE**. 2006. Tese (Doutorado) — Universidade Federal de Lavras, 2006.

SCHAFFERS, H. et al. Smart cities and the future internet: Towards cooperation frameworks for open innovation. In: SPRINGER. **The future internet assembly**. [S.l.], 2011. p. 431–446.

SHUKLA, S. et al. Comparative study of 1g, 2g, 3g and 4g. **J. Eng. Comput. Appl. Sci**, v. 2, n. 4, p. 55–63, 2013.

SIGFOX. **Sigfox Technology Overview**. 2019. Disponível em: <https://www.sigfox.com/en/sigfox-iot-technology-overview>. Acesso em: 05 mai. 2019.

SIGFOX BACKEND. **Backend Sigfox**. 2019. Disponível em: <https://backend.sigfox.com>. Acesso em: 05 mai. 2019.

SIPIU. **Embarcados Contest - Projeto Sistema de Identificação de Problemas de Infraestrutura Urbana**. 2016. Disponível em: <https://contest.embarcados.com.br/projetos/sipiu/>. Acesso em: 23 nov. 2018.

TAGOIO. **TagoIO - Início**. 2019. Disponível em: <https://tago.io/>. Acesso em: 04 mai. 2019.

TELIT. **JN3 Archives - Telit**. 2017. Disponível em: <https://www.telit.com/product-series/jn3/>. Acesso em: 23 nov. 2018.

TELIT IOT PORTAL. **Telit IoT Portal**. 2019. Disponível em: <https://portal.telit.com/app/login>. Acesso em: 23 nov. 2018.

WND. **WND Brasil**. 2016. Disponível em: <https://www.wndgroup.io/brasil/>. Acesso em: 05 mai. 2019.

Anexos

ANEXO A – CÓDIGO EM C COMPILADO NA LPCXPRESSO4337 NO MBED USANDO TECNOLOGIA 3G

```
#include "ST7567.h"
#include "PCAL9555.h"

ST7567 lcd(D11, D13, D12, D9, D10);  // mosi, sclk, reset, A0, nCS
Serial pc(USBTX, USBRX);          //configura a comunicacao serial
Serial micro(TX, RX); //confira a entrada de dado do GPS
PCAL9555    gpio_exp(SDA, SCL); // configura o joystick
GpioBusIn   joystick(gpio_exp, X0_0, X0_1, X0_2, X0_3, X0_4);

int selectionArrow (char,int); //funcao que posiciona a seta
void lcdStart(); //funcao que inicializa o lcd

enum key_num {                          //configuracoes do joystick
    Key_Up     = (1 << X0_4),
    Key_Down   = (1 << X0_0),
    Key_Center = (1 << X0_1),
};

int main()
{
    int mainState = 18;                 // posicao inicial da seta
    char direction;
    //variavel se define se a seta sobe ou desce
    char gps[80], location[80];


    lcd.set_contrast(0x35);             //configura contraste do
       Display
    lcd.cls();                          //Limpa Display
    lcdStart(); //inicializa o display

    micro.format(8, Serial::None, 1);  //configuracoes da comunicacao
        serial com o GPS
    micro.baud(4800);
    pc.format(8, Serial::None, 1);
    pc.baud(9600);

```

```
37    while(true) {                                    // loop principal

39        int keys = joystick.read(); //leitura do joystick
          int state; //armazena a posicao
41
          if ((keys & Key_Up) == 0){ //se a seta de selecao subir
43            direction = 'U';
              state = selectionArrow(direction,mainState); //chama a
                 funcao para movimentar a flecha
45            mainState = state; //retorna a posicao atual
          }
47        if ((keys & Key_Down) == 0){ //se a seta de selecao descer
              direction = 'D';
49            state = selectionArrow(direction,mainState); //chama a
                 funcao para movimentar a flecha
              mainState = state; //retorna a posicao atual
51        }

53        if (micro.readable()) {   //recebe a string da localizacao
             do problema
            location= micro.scanf(gps);
55           break;
          }
57
          if ((keys & Key_Center) == 0){  //quando o usuario da enter
             na opcao desejada
59
              switch (mainState){    //dependendo da op   o envia para
                   o modem 3G uma string diferente
61
                 case 18 :
63                   pc.printf("1#%c", location);
                     break;
65
                  case 34 :
67                    pc.printf("2#%c", location);
                      break;
69
                  case 50 :
71                    pc.printf("3#%c", location);
                      break;
```

```
73              }
74
75
76              lcd.cls();                              //limpa o display
77              lcd.locate(0, 34);                          //Posiciona na
                    terceira linha
78              lcd.printf("TRANSMITINDO DADOS...");
79              wait(20); //delay de 20 segundos para dar o tempo de
                    transmissao
80
81              lcd.cls(); //limpa o display
82              lcd.locate(16, 34);                          //Posiciona na
                    terceira linha
83              lcd.printf("DADOS ENVIADOS");
84              wait(2);
85
86
87              lcdStart(); //funcao que inicializa o lcd
88
89          }
90          wait(0.2);
91      }
92  }
93
    int selectionArrow(char key, int actualState){ //funcao que posiciona
        a flecha de acordo com a opcao do usuario
95      if (key == 'U') {  //para subir
96          if (actualState > 18){
97              lcd.locate(0, actualState);
98              lcd.printf("  ");
99              actualState -= 16;
100             lcd.locate(0, actualState);
101             lcd.printf(">>");
102             return  actualState;
103         }
104     }
105     if (key == 'D') { //ou para descer
106         if (actualState >= 18 && actualState < 50){
107             lcd.locate(0, actualState);
108             lcd.printf("  ");
109             actualState += 16;
110             lcd.locate(0, actualState);
```

```
111             lcd.printf(">>");
                return  actualState;
113         }
    }
115     return actualState;
}
117
void lcdStart () {
119     lcd.locate(0, 1);                       //posiciona na primeira
        linha
    lcd.printf("SELECIONE O PROBLEMA");       //Escreve mensagem na
        primeira linha
121     lcd.locate(16, 18);                     //Posiciona a segunda
        linha
    lcd.printf("BURACO NO ASFALTO");          //Escreve mensagem na
        segunda linha
123     lcd.locate(16, 34);                     //Posiciona a terceira
        linha
    lcd.printf("VAZAMENTO DE AGUA");          //Escreve mensagem na
        terceira linha
125     lcd.locate(16, 50);                     //Posiciona a quarta
        linha
    lcd.printf("ENTULHO");                    //Escreve mensagem na
        quarta linha
127
    lcd.locate(0, 18);                      //Posiciona a segunda
        linha
129     lcd.printf(">>");                       //Escreve a seta na
        segunda linha
}
```

ANEXO B – CÓDIGO EM PYTHON PARA INICIALIZAÇÃO DO MÓDULO 3G

```
import time
import SER
import MDM
import GPIO
import DW_send
import binascii


print 'Import OKrn'
SER.send('Rodando')



def Config_Oper(): #Busca a operadora e configura APN
 print 'CONFIG_APN'
 print 'Get Operadorarn'

 res=''
 res = MDM.send('AT#LSCRIPTr', 2)
 res = MDM_receive(2)
 res = res.upper()
 print res
 tesc = res.find ('CONFIG_APN.PY')
 if (tesc==-1):
 print 'apn defalt'
 res=''
 res = MDM.send('AT+COPS?r', 2)
 res = MDM_receive(2)
 res = res.upper()
 print res
 ip='zap.vivo.com.br'
 usr='vivo'
 psw='vivo'
 tes = res.find ('VIVO')
 if (tes == -1):
 ip='claro.com.br'
 usr='claro'
 psw='claro'
```

```
 tes = res.find ('CLARO')
 if (tes == -1):
 ip='tim.br'
 usr='tim'
 psw='tim'
 tes = res.find ('TIM')
 if (tes == -1):
 ip='oi.com.br'
 usr='oi'
 psw='oi'
 tes = res.find ('OI')
 if (tes == -1):
 op1 = 0
 print 'Operadora desconhecida'
 if (tesc!=-1):
 print 'arquivo apn'
 import CONFIG_APN
 ip,usr,psw = CONFIG_APN.config()
 if(ip==''):
 res = MDM.send('AT#REBOOTr', 2)
 return (ip,usr,psw)


def MDM_receive(timeout): # Recebe resposta dos comandos AT
 res = ''
 start = time.time()
 while (time.time() - start < timeout):
 res = res + MDM.read()
 return res



def Conect(ip, usr, psw): # Faz a conex o e ganha um endere o de IP
 tent=0
 con=1

 print 'configure PDP context with APN %s, username %s, password %s'
    % (ip, usr, psw)
 res=''
 res = MDM.send('AT+CGDCONT=1,"IP","' + ip + '"r', 2)
 res = MDM_receive(2)
 res = res.find ('OK')
```

```
80  if (res == -1):
    print 'error setting PDP contextr'
82
    if (usr != ''):
84  print 'set GPRS usernamer'
    res=''
86  res = MDM.send('AT#USERID="'+ usr + '"r', 2)
    res = MDM_receive(2)
88  res = res.find ('OK')
    if (res == -1):
90  print 'error setting GPRS usernamer'

92  if (psw != ''):
    print 'set GPRS passwordr'
94  res=''
    res = MDM.send('AT#PASSW="'+ psw + '"r', 2)
96  res = MDM_receive(2)
    res = res.find ('OK')
98  if (res == -1):
    print 'error setting GPRS passwordr'
100
    print 'activate GPRS contextr'
102
    while(tent<=20)&(con==1):
104 print 'iniciando tentativa ' + str(tent) + ' ...'
    res=''
106 res = MDM.send('AT#GPRS?r',2)
    res = MDM_receive(2)
108 res = res.upper()
    print res
110 tes = res.find ('#GPRS: 0')
    tes1 = res.find ('#GPRS: 1')
112 tes2 = res.find ('#GPRS: 2')
    print tes
114 print tes1
    print tes2
116
    if(tes==2):
118 res=''
    res = MDM.send('AT#GPRS=1r',2)
120 res = MDM_receive(4)
    print res
```

```
122  res = res.find ('OK')
     if(res==2):
124  print 'conectado'
     if (res == -1):
126  print res
     con=1
128  print 'Impossivel conectarr'
     if(tes1==2):
130  con=0

132  if(tes2==2):
     res=''
134  res = MDM.send('AT#GPRS=0r',2)
     res = MDM_receive(4)
136  res = res.find ('OK')
     con=1
138  if(tent==20)&(con==1):
     res=''
140  res = MDM.send('AT#LSCRIPTr', 2)
     res = MDM_receive(2)
142  res = res.upper()
     print res
144  tesc = res.find ('CONFIG_APN.PY')
     if(tesc!=-1):
146  res=''
     res = MDM.send('AT#DSCRIPT="CONFIG_APN.py"r', 2)
148  res = MDM_receive(2)
     res = res.upper()
150  print res
     tesc = res.find ('OK')
152  if(tesc==2):
     res = MDM.send('AT#REBOOTr', 2)
154

     tent=tent+1
156  print 'all settings doner'
     return con
158

print 'SIPIU_INICIO'
160

op1 = 0
162 sim1 = 0
    tent1 = 0
```

```
ok = 0
repete = 0

while((sim1==0)&(tent1<=20)):
 print 'iniciando tentativa ' + str(tent1) + ' ...'

 res = MDM.send('AT+CREG=0,0r', 2)
 res = MDM_receive(2)
 res = res.upper()
 print res

 print 'Get Operadorarn'

 res = MDM.send('AT+CPIN?r', 2)
 res = MDM_receive(2)
 res = res.upper()
 print res



 tes = res.find ('ERROR')
 if (tes == -1):

 print 'Aguardando registrorn'

 res = MDM.send('AT+CREG?r', 2)
 res = MDM_receive(2)
 res = res.upper()
 print res

 tes = res.find ('+CREG: 0,1')
 print tes
 if (tes == 2):
 print 'Registrado'
 sim1=1

 ip,usr,psw = Config_Oper()
 print 'APN'
 print ip
 print 'User'
 print usr
 print 'Password'
```

```
206  print psw


208

  tent1=tent1+1
210  #print op1

212 if(sim1==1):

214  ok=Conect(ip, usr, psw)
  print 'Chama DW Send'
216  sim1=DW_send.DW_connect()


218

 if(sim1==0):
220  print 'Nao foi possivel conectar'
 print 'Doner'
222 print ok


224

 while(repete<=10):
226  MDM.send('AT#DWCONN?r', 0);
  res = MDM_receive(1);
228  res = res.upper()
  tes = res.find ('#DWCONN: 1,2')
230  repete=repete+1
  if (tes==2):
232

  print 'Modem ja conectado no DW'
234  conect = 1
  print 'Enviando Dados...'
236  DW_send.Status_reg1()

238  if (repete==10):
  sim1=DW_send.DW_connect()
240  repete=0
  if (tes!=2):
242

  print 'Nao foi possivel enviar Dados'
244  res = MDM.send('AT#REBOOTr', 2)
```

ANEXO C – CÓDIGO EM PYTHON DA CONEXÃO DO MÓDULO 3G COM A PLATAFORMA IOT

```
import time
import SER
import MDM
import binascii


##################################
APP_TOKEN = # inserir aqui o app token
SER.set_speed('9600','8N1')
##################################

def MDM_receive(timeout): # Recebe respostas de comandos AT
 res = ''
 start = time.time()
 while (time.time() - start < timeout):
 res = res + MDM.read()
 return res

def SER_receive(timeout): # Recebe respostas da Serial
 res = ''
 fora = '0'
 start = time.time()
 while (fora=='0'):
 res = res + SER.read()
 if(res[:-1].endswith('x0d')):
 fora='1'
 SER.send(res)
 return res

def DW_connect(): #Conecta no DeviceWise
 sim1=0
 tent=0
 MDM.send('AT#DWCFG=open-api.devicewise.com,0,' + APP_TOKEN + 'r', 0)
 res = MDM_receive(5)
 res = res.upper()
 tes = res.find ('OK')
 if(tes!=(-1)):
```

```
38  print 'APP_TOKEN_OK1'
    MDM.send('AT#DWCONN?r', 0)
40  res = MDM_receive(5)
    res = res.upper()
42  print res
    tes = res.find ('#DWCONN: 1,2')
44  if (tes==2):
    print 'Modem ja conectado no DW'
46  sim1=1
    if (tes!=2):
48  MDM.send('AT#DWCONN=0r', 0)
    res = MDM_receive(5);
50  res = res.upper()
    tes = res.find ('#DWCONN: 0,0')
52  if (tes==2):
    print 'Tentando reconectar'
54  DW_connect()

56  while (sim1==0 & tent<=10):
    print 'APP_TOKEN_OK2'
58  MDM.send('AT#DWCONN=1r', 0)
    res = MDM_receive(5)
60  print res
    res = res.upper()
62  tes = res.find ('OK')
    if(tes!=(-1)):
64
    sim1=0
66  tent=tent+1
    if(tes!=(-1)):
68  print 'DW_connect_OK'
    sim1=1
70  return sim1


72
    def gravalog(msg): #Envio de LOG para o DW.
74  MDM.read()
    MDM.send('AT#DWSEND=0,log.publish,msg,' + str(msg) + 'r', 0)
76  time.sleep(2)
    #Fim do bloco gravalog.
78
```

```
80 def Status_reg1(): # Aguarda receber evento e coordenadas

82  SER.send('*')
    a = SER_receive(10)
84  print a
    cord=a.split('#')
86  if (a!=''):
    print 'dado da resposta'
88  print a
    MDM.send('AT#DWSEND=0,property.publish,key,Status,value,' + str(cord
        [0]) + 'r', 0)
90  res = MDM_receive(20)
    print res
92  print 'dado enviado'
    MDM.send('AT#DWSEND=0,property.publish,key,Latitude,value,' + str(
        cord[1]) + 'r', 0)
94  res = MDM_receive(20)
    print res
96  print 'dado enviado'
    MDM.send('AT#DWSEND=0,property.publish,key,Longitude,value,' + str(
        cord[2]) + 'r', 0)
98  res = MDM_receive(20)
    print res
100 print 'dado enviado'
    if (a==''):
102 print 'Sem resposta do medidor'
```

ANEXO D – CÓDIGO EM C COMPILADO NA LPCXPRESSO4337 NO MBED USANDO TECNOLOGIA SIGFOX

```
#include "ST7567.h"
#include "PCAL9555.h"

ST7567 lcd(D11, D13, D12, D9, D10);  // mosi, sclk, reset, A0, nCS
Serial pc(USBTX, USBRX);          //configura a comunicacao serial
// Serial micro(TX, RX);             //confira a entrada de dado do GPS
PCAL9555    gpio_exp(SDA, SCL); // configura o joystick
GpioBusIn   joystick(gpio_exp, X0_0, X0_1, X0_2, X0_3, X0_4);

int selectionArrow (char,int); //funcao que posiciona a seta
void lcdStart();    //funcao que inicializa o lcd

enum key_num {                          //configuracoes do joystick
    Key_Up     = (1 << X0_4),
    Key_Down   = (1 << X0_0),
    Key_Center = (1 << X0_1),
};

int main()
{
    int mainState = 18;                 // posicao inicial da seta
    char direction;                     //variavel se define se a
       seta sobe ou desce

    lcd.set_contrast(0x35);             //configura contraste do
       Display
    lcd.cls();                          //Limpa Display
    lcdStart();                         //inicializa o display

    pc.format(8, Serial::None, 1);  // comunica  o serial com o
       modulo sigfox
    pc.baud(9600);

    while(true) {                               // loop principal

        int keys = joystick.read(); //leitura do joystick
        int state; //armazena a posicao
```

```
36          if ((keys & Key_Up) == 0){ //se a seta de selecao subir
                direction = 'U';
38              state = selectionArrow(direction,mainState); //chama a
                   funcao para movimentar a flecha
                mainState = state; //retorna a posicao atual
40          }
            if ((keys & Key_Down) == 0){ //se a seta de selecao descer
42              direction = 'D';
                state = selectionArrow(direction,mainState); //chama a
                   funcao para movimentar a flecha
44              mainState = state; //retorna a posicao atual
            }
46

48          if ((keys & Key_Center) == 0){  //quando o usuario da enter
               na opcao desejada

50              pc.printf("AT$RC\n");     // configuracao para envio

52              switch (mainState){     //dependendo da op   o envia para
                    o modulo sigfox um status diferente

54                  case 18 :
                        pc.printf("AT$SF=1#%c\n");
56                      break;

58                  case 34 :
                        pc.printf("AT$SF=2#%c\n");
60                      break;

62                  case 50 :
                        pc.printf("AT$SF=3#%c\n");
64                      break;
                }
66

68              lcd.cls();                          //limpa o display
                lcd.locate(0, 34);                      //Posiciona na
                   terceira linha
70              lcd.printf("TRANSMITINDO DADOS...");
```

```
            wait(20);                    //delay de 20 segundos para
               dar o tempo de transmissao
72
            lcd.cls(); //limpa o display
74          lcd.locate(16, 34);                    //Posiciona na
               terceira linha
            lcd.printf("DADOS ENVIADOS");
76          wait(2);


78
            lcdStart(); //funcao que inicializa o lcd
80
        }
82      wait(0.2);
    }
84  }

86  int selectionArrow(char key, int actualState){ //funcao que posiciona
        a flecha de acordo com a opcao do usuario
    if (key == ''U'') {                          //para subir
88      if (actualState > 18){
            lcd.locate(0, actualState);
90          lcd.printf("  ");
            actualState -= 16;
92          lcd.locate(0, actualState);
            lcd.printf(">>");
94          return  actualState;
        }
96  }
    if (key == ''D'') {                          //ou para descer
98      if (actualState >= 18 && actualState < 50){
            lcd.locate(0, actualState);
100         lcd.printf("  ");
            actualState += 16;
102         lcd.locate(0, actualState);
            lcd.printf(">>");
104         return  actualState;
        }
106 }
    return actualState;
108 }
```

```
110 void lcdStart () {
        lcd.locate(0, 1);                          //posiciona na primeira
            linha
112     lcd.printf("SELECIONE O PROBLEMA");        //Escreve mensagem na
            primeira linha
        lcd.locate(16, 18);                        //Posiciona a segunda
            linha
114     lcd.printf("BURACO NO ASFALTO");           //Escreve mensagem na
            segunda linha
        lcd.locate(16, 34);                        //Posiciona a terceira
            linha
116     lcd.printf("VAZAMENTO DE AGUA");           //Escreve mensagem na
            terceira linha
        lcd.locate(16, 50);                        //Posiciona a quarta
            linha
118     lcd.printf("ENTULHO");                     //Escreve mensagem na
            quarta linha

120     lcd.locate(0, 18);                         //Posiciona a segunda
            linha
        lcd.printf(">>");                          //Escreve a seta na
            segunda linha
122 }
```