UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ - UNIOESTE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Estudo de dispositivos LoRa e proposta de alterações visando redução de consumo em modo deep sleep

Trabalho de Conclusão de Curso

Roberta Aparecida da Silva Alcantara

Cascavel–PR

2021

UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ - UNIOESTE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Roberta Aparecida da Silva Alcantara

Estudo de dispositivos LoRa e proposta de alterações visando redução de consumo em modo deep sleep

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel.

Orientador(a): Marcio Seiji Oyamada

Cascavel–PR

2021

ROBERTA APARECIDA DA SILVA ALCANTARA

ESTUDO DE DISPOSITIVOS LORA E PROPOSTA DE ALTERAÇÕES VISANDO REDUÇÃO DE CONSUMO EM MODO DEEP SLEEP

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Marcio Seiji Oyamada (Orientador)
Colegiado de Ciência da Computação, UNIOESTE

Prof. Edmar André Bellorini
Colegiado de Ciência da Computação, UNIOESTE

Javan Ataíde de Oliveira Junior
Mestre em Ciência da Computação

Cascavel, 26 de Julho de 2022

Agradecimentos

Aos meus familiares, que me deram todo o apoio durante essa jornada e me incentivaram a chegar até aqui. Sem vocês nada disso seria possível!

Aos meus amigos, que compartilharam comigo todas as experiências vividas durante esse período. O companheirismo e cumplicidade de vocês foi essencial.

Aos meus professores, em especial a meu orientador e tutor Marcio Oyamada, por todos os ensinamentos e dedicação em compartilhar o conhecimento durante esses anos de curso.

Resumo

A Internet das Coisas abrange uma diversidade de tecnologias que ganharam grande destaque na última década. Para a comunicação, as redes LPWAN tem como característica o baixo consumo de potência e o longo alcance de transmissão. Um dos protocolos de grande destaque que é utilizado nessas redes é o LoRaWAN, desenvolvido exclusivamente para a modulação de sinal LoRa. De acordo com as especificações desse protocolo, um *transceiver* LoRa utilizando uma bateria de 2000 mAh tem um tempo de vida de 105 meses. Em estudos preliminares a respeito de como as principais características de uma rede LoRa se comportam na prática, foi feita uma estimativa do tempo de vida de uma bateria com as mesmas condições da que é apresentada na especificação do protocolo, utilizando como objeto de análise um dispositivo TTGO LORA 32. Como resultado para o consumo obteve-se 10,5 mA em modo *deep sleep* no estado inativo e 180 mA durante a transmissão de pacotes, resultando em apenas 8 dias de duração da bateria nessas condições. Esse consumo tão elevado obtido na prática impacta de forma direta na manutenção desse dispositivo. Dessa forma, para tornar mais aplicável o seu uso é essencial que se tenha um baixo consumo de energia em modo *deep sleep*. A partir do problema encontrado, analisou-se um fórum de discussão e selecionou-se possíveis soluções para o problema, que abarcam tanto modificações no *software* quanto no *hardware*. Além do dispositivo TTGO já utilizado anteriormente, a análise dessas soluções foi realizada também sobre o dispositivo Heltec WiFi LoRa 32. Ao aplicar tais soluções e testá-las, constatou-se que uma das soluções proporcionou um consumo de 800 μA para o dispositivo Heltec e 1,4 mA para o TTGO, mostrando a eficiência do estudo, aumentando para 94 e 56 dias o tempo de vida desses dispositivos, respectivamente.

Palavras-chave: Redes de longo alcance e baixo consumo de energia. Consumo de energia. Modo de economia de energia.

Lista de figuras

Lista de quadros

Lista de códigos

Lista de abreviaturas e siglas

BMS	*Battery Management System*
CSS	*Chirp Spreading Spectrum*
FSK	*Frequency shift-keying*
GND	*Ground*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
IoT	*Internet of Things*
LoRa	*Long Range*
LoRaWAN	*Long Range Wide Area Network*
LPWAN	*Low Power Wide Area Network*
MCU	*Microcontroller Unit*
MISO	*Master Input Slave Output*
MOSI	*Master Out Slave In*
OLED	(*Organic Light-Emitting Diode*)
RFID	*Radio Frequency Identification*
RSSFs	Redes de Sensores Sem Fio
RTC	*Real Time Clock*
SCK	*Serial Clock*
SF	*Spreading Factor*
SPI	*Serial Peripheral Interface*
SS	*Slave Select*
ULP	*Ultra Low Power*

Lista de símbolos

CA	Custo total do período de envio de mensagens (ativo) em miliamperes
CCA	Corrente consumida por mensagem durante o estado ativo em miliamperes
CCI	Corrente consumida por mensagem durante o estado inativo em miliamperes
CH	Custo total para o estado de hibernação (inativo) em miliamperes
CT	Custo total em um período de uma hora (mAh)
CTB	Carga total da bateria
EA	Tempo em estado ativo em milissegundos
EI	Tempo em estado inativo em milissegundos
PI	Porcentagem do tempo em estado inativo
QE	Quantidade de envios em uma hora
TE	Tempo entre envio de mensagens em minutos
$TEMS$	Tempo entre envio de mensagens em milissegundos
TV	Tempo de vida da bateria em horas

Sumário

1 Introdução

Assim como apresentado por Santos et al. (2017), o termo Internet das Coisas (do inglês *Internet of Things* - IoT) surgiu no ano de 1999 como uma referência à tecnologia RFID (*Radio Frequency Identification*). Com o passar dos anos esse termo foi se tornando mais abrangente, englobando conceitos como redes de sensores sem fio (RSSFs) e até mesmo automação de residências e indústrias. Em 2012 identificou-se a IoT como uma tecnologia emergente, devido a sua capacidade de conectar dispositivos que apresentam processamento, sensoriamento e comunicação próprios (SANTOS et al., 2017).

Ter dispositivos com tais características tornou muito vantajoso o uso da IoT, fazendo com que ela fosse inserida em distintos contextos que envolvem, por exemplo, a segurança de locais, a medicina, o tráfego de veículos e até mesmo o setor militar. No entanto, apesar dos benefícios tem-se também as suas limitações que envolvem principalmente questões de processamento e de consumo de energia. Além disso, por se tratar de comunicação entre dispositivos, outro critério que deve ser analisado é o alcance de transmissão de dados entre eles (LOUREIRO et al., 2003).

Grande parte das vezes, as aplicações com o uso da Internet das Coisas são aplicadas a áreas remotas, dificultando o processo de manutenção dos equipamentos. Como o funcionamento dessas aplicações depende principalmente da quantidade disponível de energia que geralmente é proporcionada por baterias, é necessário balancear o uso dos recursos e buscar alternativas que diminuam o consumo, para assim, ter maior tempo de vida e consequentemente baixa manutenção (LOUREIRO et al., 2003).

Dentre os protocolos de IoT mais conhecidos e utilizados estão o Wi-Fi e o Bluetooth. O Wi-Fi costuma ser utilizado em contextos que se tem uma alta taxa de transmissão de dados e o consumo energético e o alcance de transmissão pelo dispositivo não são um problema, visto que possui um alto consumo de energia e pouco alcance, chegando a cerca de 50 metros apenas (VIDAL, 2017). Quanto ao Bluetooth, o consumo energético é bastante inferior, tornando-o bastante aplicável às novas soluções de IoT. Porém, o alcance continua sendo baixo, variando

entre 1 e 240 metros de acordo com a classe que se encontra (VIDAL, 2017).

Pensando em cenários em que é estritamente necessário um alto alcance de transmissão e um baixo consumo energético tem-se as redes LPWAN (do inglês, *Low Power Wide Area Network*). Protocolos de grande destaque nessas redes são o LoRaWAN (*Long Range Wide Area Network*) e o Sigfox. O Sigfox é proprietário e tem acesso liberado apenas em locais que se tenha cobertura de seu sinal, além de não permitir a criação de redes internas. No entanto, como o envio de mensagens se dá de forma duplicada, a chance de se ter ruídos ou interferências de sinal é baixa. O LoRaWAN atinge uma maior largura de banda e como consequência sofre maior interferência, porém possui um menor custo de investimento inicial. A principal vantagem do Sigfox se comparado ao LoRaWAN é o alcance, atingindo uma diferença de até 8 km em ambiente urbano e em campo aberto de no mínimo 15 km, podendo chegar até mesmo a 35 km. Em contrapartida, o LoRaWAN tem como principais vantagens o tamanho de sua mensagem e ser um protocolo aberto (BARBOSA, 2017).

De acordo com as especificações do protocolo LoRaWAN (LORA ALLIANCE, 2015) que é implementado pelo *chip* LoRa SX1276 nos dispositivos utilizados neste trabalho, tem-se um tempo de vida estimado em 105 meses com uma bateria de 2.000 mAh, avaliando apenas o consumo por parte do *chip* referente ao protocolo. No entanto, a partir de capturas empíricas do dispositivo TTGO LORA32 (LILYGO, 2020) onde foi analisado o consumo por completo pelo dispositivo, incluindo demais módulos além do LoRa, obteve-se um consumo de 10,5 mA com o uso de economia de bateria através do *deep sleep* (modo de maior economia de energia) e durante a transmissão da mensagem cerca de 180 mA (ALCANTARA; OYAMADA, 2020). O impacto dessa diferença de valores pode ser observado no tempo de vida da bateria, que na prática seria em torno de 8 dias, sendo extremamente abaixo do que é esperado se comparado ao apresentado na especificação.

Há diversas pesquisas que utilizam como base dispositivos LoRa em suas aplicações. Cruz et al. (2021) emprega o dispositivo TTGO com o uso da tecnologia LoRa em um sistema de mensagens instantâneas em áreas sem serviço de telecomunicação. Esse mesmo dispositivo é utilizado por Kodali, Kuthada e Borra (2018) no desenvolvimento de um sistema de irrigação inteligente e por Baumgärtner et al. (2018) em um monitoramento ambiental. O dispositivo Heltec WiFi LoRa 32, bastante semelhante ao modelo TTGO LoRa 32, também foi amplamente aplicado em diversos contextos. Pereira e Cruvinel (2019) utiliza-o em um sistema de coleta automática de dados agrícolas aplicado à plantações, enquanto Redzuan et al. (2021) emprega-o para o monitoramento da qualidade da água em regiões florestais por meio de um sensor de amônia. Em ambos os cenários apresentados observa-se que o uso de equipamentos portáteis para a alimentação energética como as baterias são essenciais, considerando a dificuldade de acesso e a falta de distribuição de redes elétricas a tais locais.

1.1 Objetivos

1.1.1 Geral

O objetivo geral deste trabalho é analisar possíveis soluções para o problema encontrado com o consumo em *deep sleep*, a fim de diminuir o consumo em tal modo e assim, aumentar o tempo de vida do dispositivo, tornando seu uso ainda mais vantajoso para aplicações futuras.

1.1.2 Específicos

- Catalogar possíveis soluções para o problema encontrado
- Pesquisar informações sobre como implementar tais soluções
- Implementar tais modificações que envolvem alterações no *software*
- Aplicar alterações no *hardware*
- Testar as soluções nos dispositivos TTGO e Heltec
- Aplicar os resultados dos testes no modelo analítico para análise do tempo de vida do dispositivo

1.2 Organização do texto

O Capítulo 2 aborda o levantamento bibliográfico utilizado no decorrer do desenvolvimento do projeto. São abordados de forma mais aprofundada os principais conceitos que envolvem rede LoRa (*Long Range*), bem como as principais soluções que a utilizam e os possíveis modos de economia de energia. No Capítulo 3 é descrito o percurso metodológico seguido no desenvolvimento do trabalho, incluindo descrição do estudo preliminar desenvolvido, as possíveis soluções catalogadas para o problema encontrado, os circuitos empregados para a medição de corrente e o processo para a aplicação de cada uma das soluções. No Capítulo 4 são apresentados os resultados obtidos com o uso de cada uma das soluções para cada modelo de dispositivo LoRa, avaliando o consumo de corrente em *deep sleep* e o impacto no tempo de vida da bateria. Por fim, no Capítulo 5 são apresentadas as considerações finais acerca do objeto de estudo deste trabalho mediante os resultados obtidos, bem como a sugestão de trabalhos futuros que possam complementar essa linha de pesquisa.

2

Referencial Teórico

Na seção 2.1 são apresentados os principais conceitos que envolvem uma rede LoRa, com detalhes sobre a modulação e o protocolo utilizados. A seção 2.2 descreve os principais modos de economia de energia para o *chip* LoRa utilizado, com suas respectivas faixas de consumo, enquanto a seção 2.4 apresenta essa relação para o *chip* base dos dispositivos. Existem diversos modelos de dispositivos capazes de operar como um *end-point* em uma rede LoRa e na seção 2.3 encontra-se a descrição e as caracteríticas de soluções que se destacam nesse cenário.

2.1 LoRa e LoRAWAN

A LoRa se trata de uma tecnologia de modulação de sinal presente na camada física de uma rede, podendo ser atrelada a distintos protocolos. Ela foi desenvolvida pela Semtech Corporation e promovida pela LoRa Alliance, uma associação sem fins lucrativos e aberta (TEIXEIRA; ALMEIDA, 2017).

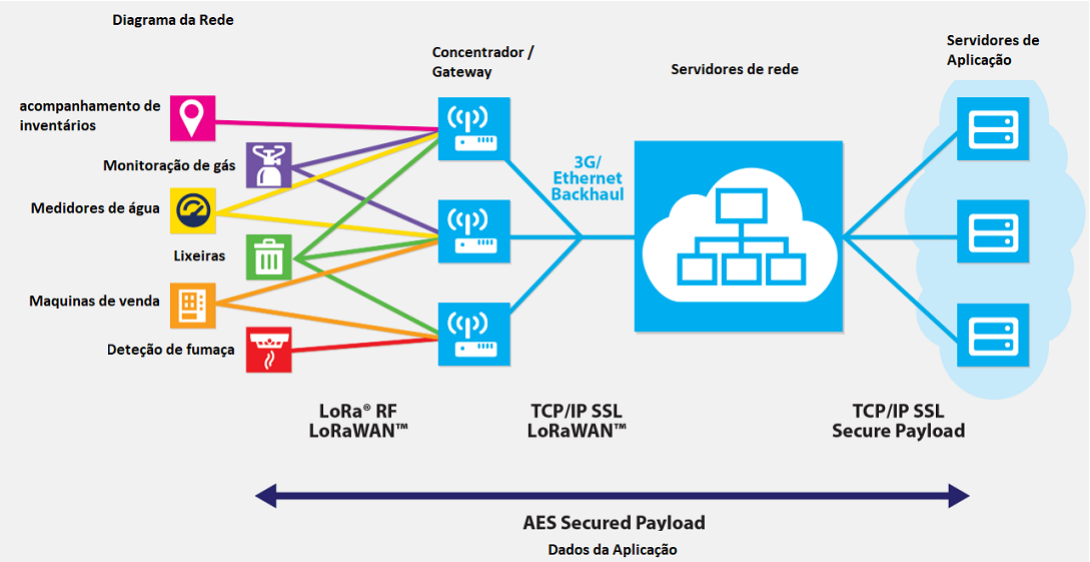
A técnica dessa modulação tem como conceito o espalhamento espectral de *chirps* (*Chirp Spreading Spectrum* – CSS), cujo objetivo é otimizar o desempenho da rede, possibilitando a troca de taxa de transmissão por robustez do sinal. Há uma taxa de transmissão variável que utiliza fatores de espalhamento ortogonais, permitindo a transmissão e recebimento ao mesmo tempo. O fator de espalhamento (*Spreading Factor* – SF) é um dos parâmetros de configuração da camada física da LoRa que influencia na duração do *chirp* no tempo (SEMTECH, 2015). Os valores de SF podem estar entre 7 e 12. O aumento do SF impacta diretamente no aumento da relação sinal-ruído, da sensibilidade e do alcance (ORTIZ, 2018).

A LoRa pode ser utilizada com variados protocolos, dos quais há um desenvolvido exclusivamente para uso com tal modulação: o LoRaWAN. A versão inicial desse protocolo foi desenvolvida pela LoRa Alliance e tinha como objetivo garantir todas as características base de uma rede LoRa (TEIXEIRA; ALMEIDA, 2017). Esses fatores estão presentes em seu manual

(SEMTECH, 2015), dos quais destaca-se largura de banda escalável, encapsulamento e consumo de energia constante, alta robustez e capacidade de alto alcance.

O protocolo LoRaWAN tem como princípio de organização a topologia estrela, em que cada dispositivo é conectado a uma unidade central (*gateway)*, assim como pode ser observado na Figura 1. A composição dos dispositivos na rede é dividida nos seguintes elementos: módulos (*end-point*), *gateways*, servidores de rede e servidores de aplicação. Os módulos abarcam todos os dispositivos que vão gerar dados que precisam ser enviados à rede, como é o caso dos sensores. Além de transmitirem mensagens (*uplink*), os nós podem também receber (*downlink*) (LORA ALLIANCE, 2015). Não existe conexão direta entre os módulos, mas sim através do *gateway*, que recebe as informações enviadas por eles e repassa aos servidores de rede. Cada *gateway* pode receber dados de milhares de dispositivos, desde que estejam dentro da sua área de alcance. Além de receber, os servidores de rede são responsáveis pelo armazenamento e gerenciamento dos dados. Caso haja o envio de pacotes de dados idênticos por *gateways* diferentes, o servidor de rede é o responsável pela eliminação da duplicação de informações. Os servidores de aplicação se utilizam do servidor de rede fornecendo o acesso do usuário às informações através de alguma aplicação útil resultado do processamento dos dados (SANT'ANA, 2017).

Figura 1 – Topologia estrela do protoloco LoRaWAN



Fonte: (LORA ALLIANCE, 2015)

Os nós são dispositivos assíncronos, de modo que a comunicação possa ser programada ou possa ocorrer em momentos que o dispositivo se encontra pronto para tal. Como consequência, o consumo de energia é reduzido e a bateria tem o aumento de seu tempo de vida, visto que não é necessário o custo de despertar o dispositivo para que haja sincronização, assim como ocorre quando se tem redes síncronas (TEIXEIRA; ALMEIDA, 2017). Diante disso, os dispositivos terminais podem ser subdivididos nas classes A, B e C, que se diferenciam desde a como a

comunicação entre nó e *gateway* ocorre, até ao consumo energético que isso acarreta (LORA ALLIANCE, 2015).

Na classe A, assim que ocorre um *uplink*, abre-se duas janelas de recepção que terão informações de *donwlink* vindas do servidor. As duas janelas não são criadas de forma simultânea, mas sim com um determinado intervalo de tempo entre elas. O dispositivo se encontra disponível apenas quando apresenta janelas de recepção. Portanto, para que receba mensagens do servidor, necessariamente deve haver o envio de mensagens pelo dispositivo anteriormente (LORA ALLIANCE, 2015). Como o dispositivo depende apenas dele para controlar quando pode receber informações, essa classe é a que apresenta o menor consumo de energia. Todos os dispositivos podem ser categorizados como classe A, mas se configurados, podem assumir as demais classes também (SANT'ANA, 2017; ORTIZ, 2018).

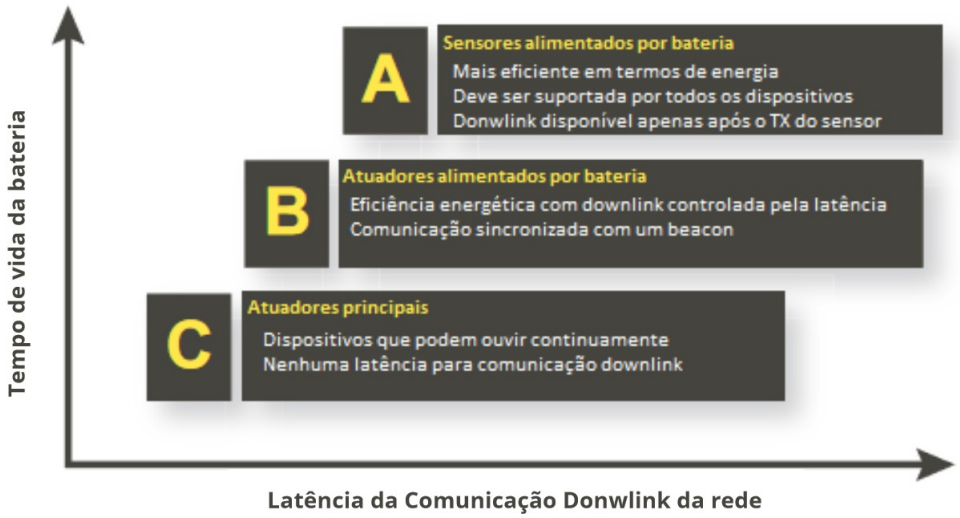
Semelhante a classe A, com a B são criadas duas janelas de recepção quando uma transmissão ocorre. Além disso, pode haver a criação de uma janela de recepção extra, chamada de *ping-slot*, criada a partir de mensagens enviadas pelo *gateway* por meio de *beacons*. A vantagem de se ter uma janela extra é garantir que o dispositivo estará disponível para recepção de mensagens em um momento determinado pelo *gateway*, além de se ter o sincronismo garantido pelo *beacon*. Como esse processo é realizado em tempos determinados, não é sempre que o dispositivo estará disponível, mas se comparado à classe A, a criação de janelas extras eleva o consumo de energia. Outro benefício proporcionado pela classe B é a possibilidade de realizar trocas de mensagens *unicast* e *multicast*, de modo que um dispositivo final envie uma única mensagem a um *gateway* e um *gateway* possa enviar mensagens a múltiplos dispositivos (SANT'ANA, 2017; ORTIZ, 2018).

Com a classe C sempre haverá uma janela de recepção disponível, desde que a primeira janela não esteja recebendo ou enviando mensagens. Assim que ocorre uma transmissão essa janela é criada e permanece aberta enquanto não houver uma nova transmissão (LORA ALLIANCE, 2015). Como a recepção de mensagens vindas do *gateway* pode ser feita de forma direta, o tempo de latência em *donwlinks* é reduzido e considerando o fato do dispositivo final estar sempre disponível, o consumo energético é alto. Não é recomendado que esse tipo de dispositivo seja alimentado via baterias, justamente por ter um consumo tão alto. Dessa forma, é indicado que eles sejam conectados diretamente a uma rede elétrica (ORTIZ, 2018). As frequências tanto de transmissão quanto de recepção são definidas no *Frequency Plan* do padrão LoRaWAN (NETWORK, 2021) e são configuradas através do servidor de rede (por exemplo, o TTS - *The Things Stack*).

Dessa forma, a relação entre as classes pode se dar comparando o tempo de latência entre as comunicações de *donwlink* e o impacto disso no consumo e consequentemente, no tempo de vida da bateria. A Figura 2 apresenta como cada uma das classes se relaciona com tais fatores. Nota-se que o tempo de vida da bateria é muito maior quando se tem dispositivos classe A, principalmente se comparada à classe C. O benefício proporcionado por esse consumo maior pela classe C é justamente a diminuição da latência nos *donwlinks*. No entanto, por conta do

grande impacto no tempo de vida, o uso da classe C torna-se inviável em cenários em que o baixo consumo é um fator essencial.

Figura 2 – Relação entre as classes de dispositivos com LoRaWAN



Fonte: Adaptado de (LORA ALLIANCE, 2015)

2.2 Modos de operação LoRa

Uma transmissão pode ser dividida em componentes caracterizados como ativos ou inativos. Os ativos referem-se ao momento em que a mensagem está sendo enviada, e a partir do momento que o sinal para de se propagar, predomina-se o estado inativo em que o transmissor LoRa permanece em modo de hibernação. Ela permanece nesse estado até que passe para o modo ativo novamente, quando uma nova mensagem for enviada. Diante disso, nota-se que os estados que a placa se encontra formam um *loop* que se repete a cada taxa de amostragem (LIANDO et al., 2019).

Assim como apresentado no manual do LoRa SX1276 (SEMTECH, 2022), *chip* que se encontra nos dispositivos TTGO e Heltec utilizados, existem variações em seus modos de operação que podem ser empregados quando o estado é inativo. A maioria desses modos possui pouca variação quanto à economia que proporciona, dos quais se destaca o modo *standby*, e como diferencial tem-se o modo *deep sleep*. O *chip* se encontra em modo de hibernação com o *standby*, porém ele se mantém em um estado de espera, o que diminui o consumo, mas ainda o mantém na faixa dos miliamperes, com um valor médio de 1,6 mA. O *deep sleep* é o modo de baixo consumo, e nesse modo o SX1276 tem um consumo médio de corrente de 0,2 μA e 1 μA de pico (SEMTECH, 2022). É importante notar que esse consumo é relacionado somente ao *transceiver* LoRa, sendo que em uma aplicação real, é necessário considerar os outros componentes da solução como microcontrolador, interface, memória entre outros.

Como o objetivo é ter o maior tempo de vida da bateria, o ideal é utilizar o modo em que se tem a maior economia de energia. Diante disso, a ideia do *deep sleep* é bastante aplicável considerando que o dispositivo irá trabalhar apenas no momento do envio e ficará ocioso até a próxima transmissão, evitando o consumo de bateria desnecessariamente.

2.3 Soluções LoRa para prototipação

Diversos dispositivos foram desenvolvidos para a prototipação de sistemas com o uso da rede LoRa. Comumente os módulos necessários para que a comunicação seja estabelecida encontram-se embutidos em um único produto, juntamente de demais tecnologias que auxiliam no sensoriamento, como o GPS (*Global Positioning System*), por exemplo. Dentre eles, serão abordadas as principais características dos dispositivos TTGO e Heltec, e como diferencial, o Shield LoRa.

2.3.1 TTGO

Um dos dispositivos objeto de análise deste trabalho consiste na placa TTGO LORA32 versão 1.0, que possui um módulo transmissor e receptor LoRa SX1276 embutido em sua estrutura. Além de possuir conectividade com a modulação LoRa, ele pode se comunicar via Wi-Fi e Bluetooth. No Quadro 1 encontra-se os principais dados técnicos de seu microcontrolador ESP32.

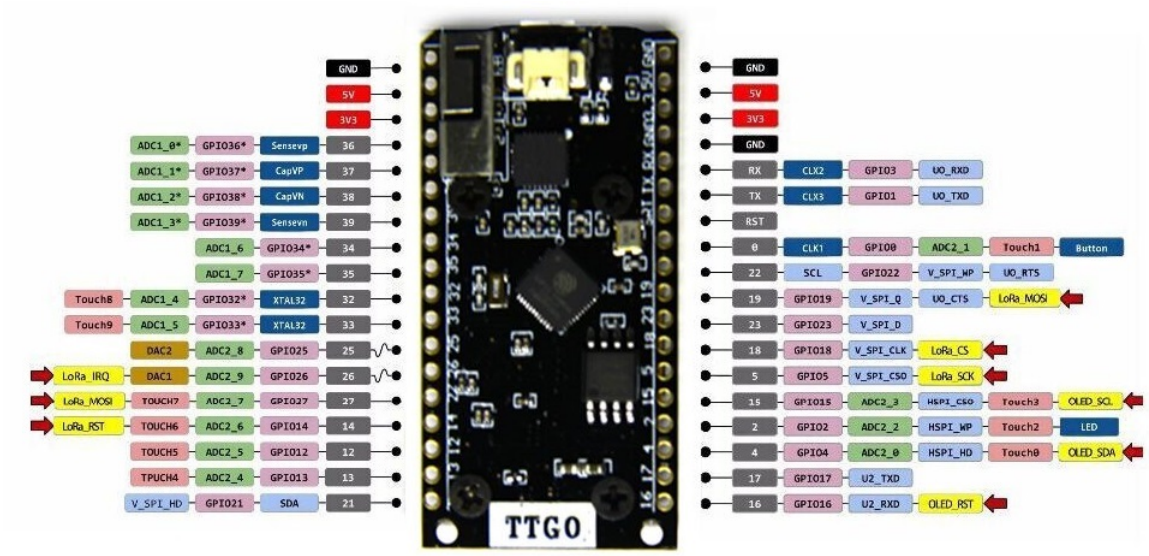
Quadro 1 – Especificações ESP32

MCU	
Chip base	ESP32-D0WDQ6
SRAM	520 Kb
Flash	32-Bit de acesso e 4 Mb
ROM	448 KB
Frequência	868/915MHz
Chip LoRa	SX1276

Fonte: Adaptado de Espressif Systems (2022)

A distribuição dos pinos presentes no *hardware* desse dispositivo é apresentada na Figura 3. Nota-se que além dos pinos externos de uso geral, há pinos específicos para o acesso ao *chip* LoRa identificados pelas setas vermelhas na imagem.

Figura 3 – Relação dos pinos da TTGO LoRa32



Fonte: (LILYGO, 2020)

Além do mapeamento dos pinos, ressalta-se a importância do esquemático dos dispositivos para que se tenha conhecimento de como os circuitos se integram em cada um dos *chips* e qual o processo utilizado para a alimentação energética dos componentes do dispositivo. No entanto, conforme apresentado pela LilyGo (XINYUAN-LILYGO, 2020) - empresa responsável pela produção desse modelo de dispositivo - não se encontra disponível um esquemático para o modelo LORA32 e versão 1.0 utilizados neste trabalho.

Tal dispositivo pode ser alimentado por meio de um cabo USB ou através de uma bateria. Com uma bateria, tal alimentação pode ser realizada por meio de um conector JST, cuja entrada se encontra disponível na parte inferior do dispositivo, ou de forma direta através dos pinos 3,3 V e GND.

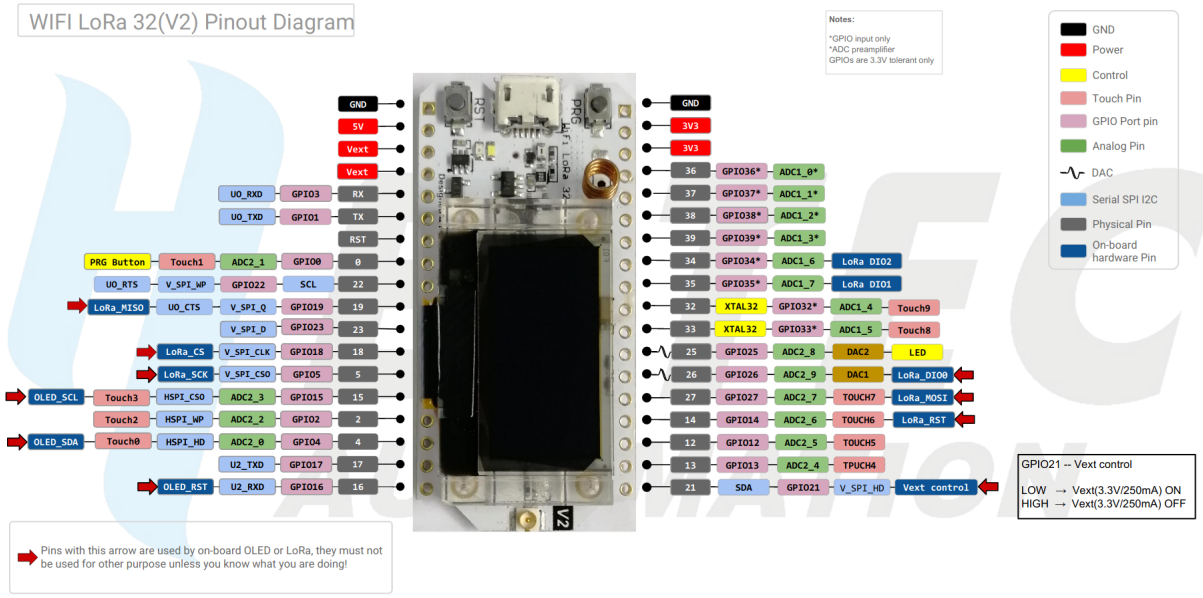
Quando se tem alimentação via USB o dispositivo apresenta um conversor USB/UART em seu circuito, cujo modelo é o CP2102 (SILICON LABS, 2017). Além da alimentação, o USB é o responsável por programar a *flash* do dispositivo. Quando se tem alimentação via bateria pelo JST, a carga é controlada pelo TP4054. Em ambos os cenários utiliza-se do regulador de tensão presente no próprio dispositivo, com o modelo ME6211 (MICRONE, 2022).

2.3.2 Heltec

Este dispositivo assemelha-se muito ao TTGO, considerando que ambos apresentam o mesmo *chip* LoRa e módulos como Wi-Fi e Bluetooth. Existem diversos modelos que variam o tamanho da memória Flash (4 ou 8 MB), presença ou não de um *display* OLED (*Organic Light-Emitting Diode*), modelo de MCU (*Microcontroller Unit*) e consumo de potência (entre 10 e 800

μA) (HELTEC AUTOMATION, 2020). Dentre esses modelos, o que mais se assemelha ao modelo TTGO utilizado é o Heltec WiFi LoRa 32, que possui o mesmo modelo de microcontrolador da TTGO. Diante disso, esse será o segundo objeto de análise deste trabalho, com a versão 2.0 desse modelo, cujo diagrama de pinos se encontra na Figura 4.

Figura 4 – Diagrama de pinos da Heltec LoRa 32



Fonte: (HIVEEYES, 2021)

A alimentação energética também pode ser realizada por uma bateria ou conectando o dispositivo a um USB. O conversor USB/UART (CP2102) e o controlador de carga da bateria (TP4054) são os mesmos apresentados pelo TTGO e como diferencial, tem-se o regulador de tensão da bateria que tem o modelo CE6260 (CHIPOWER TECHNOLOGY, 2022).

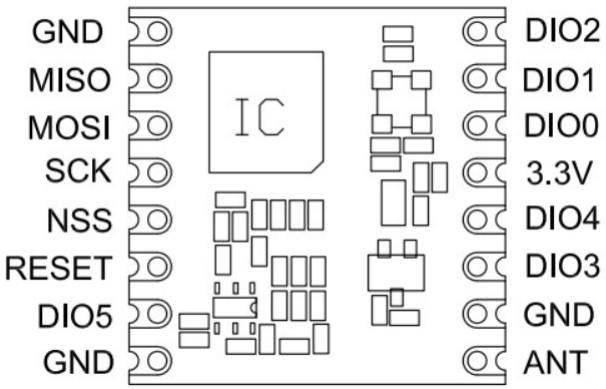
2.3.3 Shield LoRa

Diferente das demais soluções citadas, essa configuração não se encontra embutida em um único dispositivo. Os componentes que envolvem o processamento são de responsabilidade de uma plataforma externa como Arduino ou Raspberry, cujo modelo é definido de acordo com o tipo de aplicação que se deseja desenvolver. Por padrão, esse tipo de dispositivo não possui suporte à rede LoRa. Portanto, para que possa haver essa integração, é necessário o uso de um mecanismo secundário que viabilize isso.

Uma das possibilidades para comunicação LoRa é por meio do transmissor RFM95W (HOPE MICROELECTRONICS, 2006), conectado a plataforma via protocolo SPI (Interface periférica serial, do inglês *Serial Peripheral Interface*), cujo diagrama de pinos pode ser observado na Figura 5. Por conta do SPI, devem estar definidos os pinos SCK (*Serial Clock*), MISO (*Master*

Input Slave Output), MOSI (*Master Out Slave In*) e SS (*Slave Select*), ligados a pinos equivalentes na plataforma. Além disso, para a alimentação do módulo, deve-se conectar os pinos VCC (tensão de 3.3 V) e GND (*ground*) (NETO, 2018).

Figura 5 – Diagrama de pinos do RFM95W



Fonte: (HOPE MICROELECTRONICS, 2006)

Esse transmissor também suporta os principais modos de operação que se tem com o *chip* LoRa SX1276 (ESPRESSIF SYSTEMS, 2022), atingindo os mesmos valores de consumo. Logo, seria uma boa alternativa para quando já se tem posse de um Arduino por exemplo, mas não de um dispositivo que possui um módulo LoRa a ele acoplado.

2.4 Modos de operação ESP32

O módulo ESP32 presente nos dispositivos TTGO e Heltec utilizados possui variação quanto aos modos de operação, semelhante ao que se tem para o *chip* LoRa SC1276. Os possíveis modos de utilização são ativo (95-240 mA), *modem-sleep* (20-68 mA), *light-sleep* (0,8 mA), *deep-sleep* (10-100 μA), hibernação (5 μA) e desligado (1 μA) (ESPRESSIF SYSTEMS, 2022). Alguns modos podem ter alteração em seu consumo mediante a variação de alguns parâmetros e a diferença observada entre o consumo dos modos ocorre pois em cada modo alguns módulos específicos são desligados. De acordo com a especificação de cada um dos modos, optou-se pelo uso do modo *deep sleep*, pois apresenta um baixo consumo e a possibilidade de utilização da memória e periféricos RTC que permanecem ligados utilizando o coprocessador ULP (*Ultra Low Power*).

3
Metodologia

Neste capítulo será apresentado o estudo preliminar desenvolvido (seção 3.1), que tem como objetivo aferir os valores práticos de consumo com uma rede LoRa, indicando o tempo de vida dos dispositivos mediante a variação dos parâmetros da rede. A partir dos estudos preliminares obteve-se um consumo de corrente em *deep sleep* bastante elevado se comparado às especificações, sendo esse um problema para dispositivos alimentados à bateria. Na seção 3.2 é apresentado quais soluções foram catalogadas para serem testadas com o intuito de diminuir o valor de corrente consumida pelo dispositivo nesse cenário. Para a realização desses testes práticos utilizou-se de um código base que tem como modo de operação o uso do *deep sleep* (seção 3.3). Para aferição da corrente consumida quando o dispositivo envia mensagens ou permanece em modo de economia de energia utiliza-se de um circuito adaptado, cujas características se encontram na seção 3.4. Por fim, na seção 3.5 encontra-se o processo desenvolvido para a aplicação de cada uma das soluções catalogadas.

3.1 Estudo preliminar

Inicialmente, foram realizados testes práticos visando estimar a duração média de uma bateria conectada a um módulo enviando pacotes LoRa. Os dados enviados em cada pacote contavam com a geolocalização da placa, com latitude, longitude e altitude, além do número de satélites encontrados na região. Para isso, foi utilizado a placa TTGO LoRa32 e uma bateria de 2000 mA, desconsiderando o fator de decaimento da carga da bateria e levando em conta um cenário em que o consumo ocorre de forma linear.

Cada teste tem como objetivo analisar o consumo da bateria durante o período de uma hora, mediante diversos intervalos de tempo para envio dos pacotes (1 minuto, 5 minutos e 10 minutos). O teste inicia com a bateria completa e o dispositivo segue enviando pacotes de acordo com a taxa de envio definida. Após o período de uma hora, ao recarregar a bateria utilizada,

era indicado pelo próprio equipamento que realizou a recarga da bateria quanto havia sido consumido de sua carga durante esse espaço de tempo. Levando em consideração a quantidade de corrente consumida em uma hora, era possível estimar qual seria o tempo necessário para consumir a carga por completo, obtendo assim, qual o tempo de vida dessa bateria para cada um dos cenários de tempo de envio de pacotes. O resultado obtido foi bastante discrepante, visto que não havia um padrão seguido conforme a variação nos intervalos entre tempos de envio. Com taxas de envio menores, o dispositivo vai se encontrar em estado ativo mais vezes se comparado às outras taxas maiores. Como o estado ativo é o momento de maior consumo pelo dispositivo, espera-se que com uma taxa de envio menor, o consumo durante o período de uma hora seja maior se comparado a taxas de envio maiores. No entanto, a partir dos resultados obtidos não foi possível observar esse padrão ao comparar diferentes taxas de envio, visto que em vários casos, ao comparar uma taxa de envio menor com outras taxas de envio maiores, obteve-se um consumo menor, diferente do que era esperado. Dessa forma, optou-se por avaliar a duração da bateria utilizando um outro método que fosse mais consistente.

O comportamento de um sinal LoRa se dá mediante a definição de seus parâmetros e dentre eles, um dos que afeta diretamente no consumo de energia é o SF. Dessa forma, realizou-se experimentos variando tal parâmetro com o objetivo de analisar qual a relação entre o custo de corrente e o tempo de transmissão.

Por conta da inconsistência nos testes anteriores, a segunda tentativa consistia na utilização de um circuito medidor INA219 (TEXAS INSTRUMENTS, 2015) para medir o consumo de corrente na placa durante todo o período de transmissão nesse estudo preliminar. A leitura era realizada a cada 69 ms, utilizando a média de 128 amostras. No momento em que a placa se encontrava transmitindo pacotes, obteve-se um pico de corrente de 180 mA que permaneceu constante durante o tempo em que a mensagem estava sendo propagada. Foram feitas medições com valores de SF igual a 9 e a 12, com o objetivo de compará-los e entender qual a implicação dessa variação. Com o SF9, nota-se que a placa permanece aproximadamente 289 ms no pico de corrente, enquanto com SF12 a placa permanece 2257 ms. Tal informação é documentada no LoRa, onde o valor maior de SF possibilita um maior alcance do sinal, no entanto implica em um maior consumo de energia. Quanto ao estado inativo, o consumo de corrente tanto no SF9 e SF12 com a plataforma em *deep sleep* é de 10,5 mA e quando em *delay*, sem nenhum módulo desligado é de 65 mA. A partir da Figura 6 é possível visualizar a relação entre os custos de corrente em função do estado em que se encontra, bem como o tempo que a mensagem permanece se propagando em um cenário de SF12 (iniciando em aproximadamente 298 ms e finalizando em 2555 ms) e uso de *delay* entre os envios das mensagens.

Ao comparar os valores obtidos para o tempo de propagação do sinal com o que é apresentado pela documentação da tecnologia (SEMTECH CORPORATION, 2019), nota-se uma leve diferença para o SF9, que segundo a especificação tem um tempo de transmissão de 144 ms. Para o SF12 tem-se uma diferença um pouco maior, visto que é especificado um valor de

Figura 6 – Cenário de mensagem com SF12 e *delay*

Fonte: Autoria própria

991 ms. Apesar dessa divergência encontrada, considera-se que esse método de aferição seja preciso e por ser um valor obtido na prática, será utilizado como base nas próximas etapas.

Com base nos dados obtidos diretamente na placa, foi criado um modelo analítico para estimar o tempo de vida do dispositivo tendo como objeto de entrada o tempo entre envio de mensagens (TE) em minutos. A partir desse valor, todos os cálculos feitos posteriormente levam em consideração um espaço de tempo de uma hora e adotou-se como padrão a utilização dos dados na medida de milissegundos para a realização das operações. Dessa forma, a partir de TE é feita uma conversão de medidas para se obter o tempo entre envio de mensagens em milissegundos (TEMS), conforme a Equação 1:

$$TEMS = TE * 60000 \tag{1}$$

O primeiro parâmetro a ser definido é a quantidade de envios que acontecem nesse espaço de tempo. Portanto, considerando que uma hora é equivalente a 3600000 ms, a quantidade de envios (QE) pode ser obtida através da Equação 2:

$$QE = (3600000)/TEMS \tag{2}$$

Sabendo quantos envios são realizados em um intervalo de uma hora e com o tempo que a mensagem permanece sendo transmitida definido com base em seu valor de SF, toma-se conhecimento de qual o tempo em que a placa se manteve em estado ativo (EA) enviando mensagens nesse período por meio da Equação 3:

$$EA = QE * tempo \tag{3}$$

Considerando que a placa pode assumir dois estados, a partir do tempo total de vida da bateria e do tempo em que ela se encontra no estado ativo, é possível se obter o tempo em que a mesma se encontrará no estado inativo (EI), seja em *delay* ou *deep sleep*, utilizando a Equação 4:

$$EI = 360000 - EA \tag{4}$$

Visando facilitar os cálculos de custos de corrente, utilizou-se a porcentagem em valores decimais do tempo em que a placa se manteve no estado inativo (PI) durante o período de experimento. A relação entre o EI e o tempo total, bem como a representação da porcentagem já em formato decimal é obtida pela Equação 5:

$$PI = EI/3600000 \tag{5}$$

A partir do custo de corrente consumida por uma única mensagem durante o estado inativo (CCI) e com a porcentagem PI, define-se qual o custo total em mA durante o período de hibernação (CH), definido com a Equação 6:

$$CH = CCI * PI \tag{6}$$

A porcentagem referente ao tempo em que a placa se manteve no estado ativo é apenas o complemento da que foi obtida para o estado inativo. Dessa forma, para se obter o custo total do período de envio de mensagens (CA) em mA basta relacionar esse complemento com o custo de corrente para uma única mensagem no estado ativo (CCA) por meio da Equação 7:

$$CA = (1 - PI) * CCA \tag{7}$$

Como os valores dos custos CH e CA se encontravam na medida de miliamperes (mA), o valor do custo total (CT) durante o espaço de uma hora também é representado por essa medida, de acordo com a Equação 8, assim obtendo o custo energético em mAh:

$$CT = CH + CA \tag{8}$$

Sabendo qual o custo de corrente consumida durante uma hora, basta relacionar esse valor com a carga total da bateria (CTB) em mAh, assim como apresentado na Equação 9, e obter qual o tempo de vida em horas (TV) da bateria utilizada:

$$TV = CTB/CT \tag{9}$$

3.2 Soluções catalogadas

Analisando um fórum de discussão (LEE, 2017), em que foi abordado o mesmo problema encontrado nos estudos preliminares deste trabalho, foram catalogadas as soluções que mostraram ser mais plausíveis de implementar, levando em consideração a forma na qual será implantada e o benefício que proporciona. Essas soluções se diferenciam quanto ao tipo, podendo ser uma adaptação no *hardware* do dispositivo ou comandos lógicos via *software*. As soluções escolhidas a serem testadas, seu tipo, um número de identificação e comentários relevantes quanto ao resultado reportado pelos usuários do fórum são apresentadas no Quadro 2.

Quadro 2 – Soluções catalogadas para o problema do consumo

Id	**Solução**	**Tipo**	**Comentários**
1	Habilitar a função HOLD para forçar os periféricos a ficar em *deep sleep*	*Software*	Consumo abaixo de 1 μA
2	Adicionar resistores *pull-up* nos pinos de seleção e SPI	*Hardware*	Consumo de 7 μA
3	Iniciar conectado via USB, depois ligar à bateria	*Hardware*	Não há um valor de consumo apresentado, mas foi testado por várias pessoas que obtiveram redução na corrente
4	Jumper para uso do 5V do USB ou de um regulador de tensão	*Hardware*	Atingiu consumo de 1,18 mA

Fonte: Adaptado de Lee (2017)

3.3 Código teste

Para a realização dos testes tem-se como base um código para envio de pacotes (LILYGO, 2018) com adaptações para o uso do *deep sleep*, onde a cada X minutos um pacote é enviado, cujo valor X é escolhido de acordo com o contexto da utilização. Como a ideia é identificar o consumo quando se está enviando dados e quando a placa entra em modo de economia de energia, o melhor é manter o valor X em cerca de um minuto, pois caso fosse menor seria difícil de identificar a transição entre os estados e caso fosse maior, seria necessário analisar por um tempo muito grande até que se tenha um número considerável de amostras.

Esse código é escrito por meio da IDE Arduino (ARDUINO, 2022), em uma linguagem bastante semelhante com C++, com algumas modificações próprias da plataforma. A IDE possui integração com vários modelos de *hardware*, incluindo as placas da TTGO e Heltec. Portanto,

para que o código seja carregado nos dispositivos, utiliza-se "TTGO LoRa32-OLED V1" e "Heltec WiFi LoRa 32" como identificação para as placas na plataforma.

As adaptações realizadas no código para garantir o uso de *deep sleep* iniciam com a definição do tempo em que o dispositivo irá permanecer nesse modo. Esse valor é padrão para todas as mensagens que serão enviadas, portanto é definido dentro do bloco *setup*, por meio da função "esp_sleep_enable_timer_wakeup". Por fim, basta chamar a função que inicia o *deep sleep* ao final do bloco de execução com a função "esp_deep_sleep_start" (linha 24) e o método *sleep* da biblioteca LoRa (linha 23), assim como se encontra no Código 1.

Código 1 – Bloco *loop* com função *deep sleep*

```
1  void loop() {
2    digitalWrite(blueLED, ON);  // Turn blue LED on
3    //send packet
4    LoRa.beginPacket();
5    LoRa.print("HeLoRa! ");
6    LoRa.print(counter);
7    LoRa.endPacket();
8    digitalWrite(blueLED, OFF); // Turn blue LED off
9    // Display Info
10   Display.clearBuffer();
11   Display.setCursor(0,30); Display.print("Sent Packet:");
12   Display.setCursor(0,48); Display.print(" # " + (String)counter);
13   Display.sendBuffer();
14   counter++;
15
16   for (int i=0; i < 10; i++) {
17      Display.setCursor(0,12);
18      Display.print("deep sleep in"+ i);
19      Display.sendBuffer();
20      delay(1000);
21   }
22
23   LoRa.sleep();
24   esp_deep_sleep_start();
25 }
```

3.4 Medição de corrente nos testes

Na realização dos testes com as soluções catalogadas utilizou-se o multímetro digital ET-2060. Esse dispositivo apresenta diversas funcionalidades que envolvem circuitos elétricos, mas utiliza-se apenas da medição de corrente.

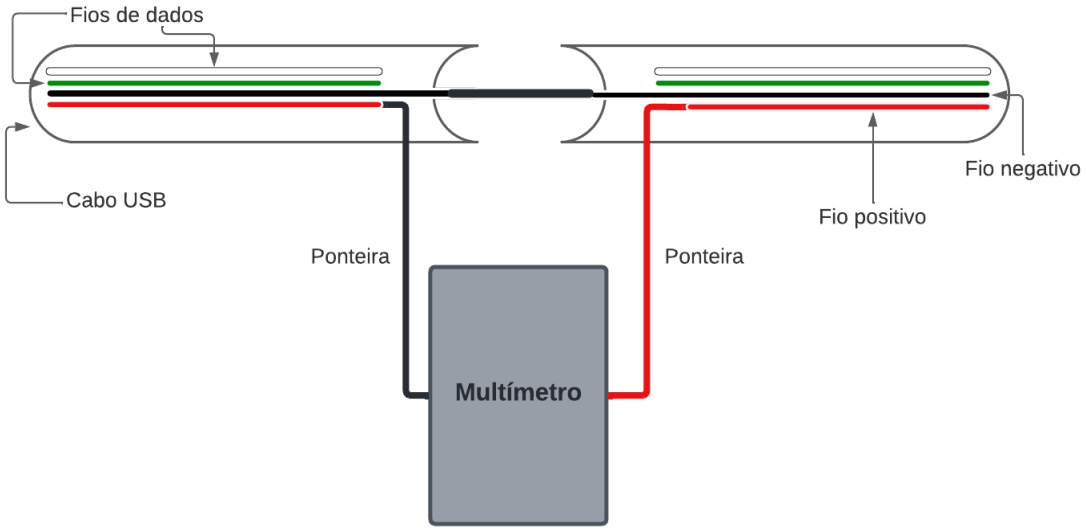
Para a realização da medição, o dispositivo contém duas ponteiras que são posicionadas em série com o circuito que se deseja medir, sendo a região que se encontra entre elas o local específico em que a medição é realizada.

Como o objetivo é medir a corrente consumida pela placa, houve variações de circuitos de medição de acordo com o estilo de alimentação, visto que há testes em que a placa recebe

alimentação via USB e outros por meio de uma bateria.

Para o circuito com USB, ambas as ponteiras foram conectadas aos fios internos positivos do cabo, de modo que fosse possível medir a corrente enviada à placa que passa por esse fio, conforme é demonstrado na Figura 7. Como os fios de dados (fio branco e fio verde) foram rompidos e não utilizados, eles foram isolados para que não entrem em contato com os demais componentes. O fio preto representa o *ground*.

Figura 7 – Esquema para o circuito de medição com USB



Fonte: Autoria própria

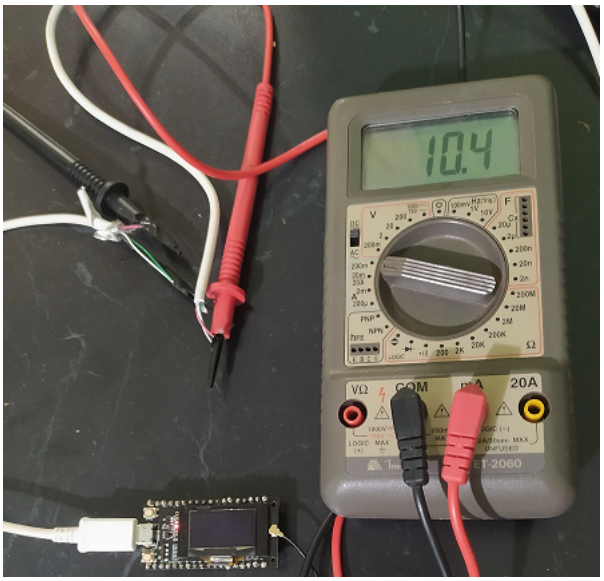
A partir desse esquema, o circuito final com os dispositivos utilizados e as adaptações realizadas está presente na Figura 8.

Para a alimentação via bateria utilizou-se de um adaptador de 1 *slot* para inserir a bateria e de um conector JST para se conectar ao dispositivo LoRa. A Figura 9 apresenta o esquema do circuito utilizado com as adaptações realizadas, indicando o posicionamento do multímetro dentre os componentes utilizados. Nota-se que o fio negativo referente ao polo negativo da bateria foi conectado diretamente ao fio negativo do JST. Para o fio positivo, referente ao objeto de medição, foi inserido o multímetro ao seu circuito para medir a corrente que sai da bateria e é enviada ao fio positivo do conector JST.

Ao implementar esse esquema nos dispositivos utilizados, a distribuição de tais componentes segue conforme se encontra representado pela Figura 10.

Conforme apresentado no código, o tempo de *deep sleep* foi fixado em um minuto, com um dispositivo como o multímetro pode-se aferir tais dados de forma satisfatória. Como a maioria dos consumos medidos estava entre 1 e 12 mA, manteve-se a escala de 200 mA para as aferições.

Figura 8 – Circuito adaptado para medição de corrente via USB



Fonte: Autoria própria

Em soluções em que se teve um consumo abaixo de 1 mA, para se ter melhor precisão foi alterada a escala para 20m/20A.

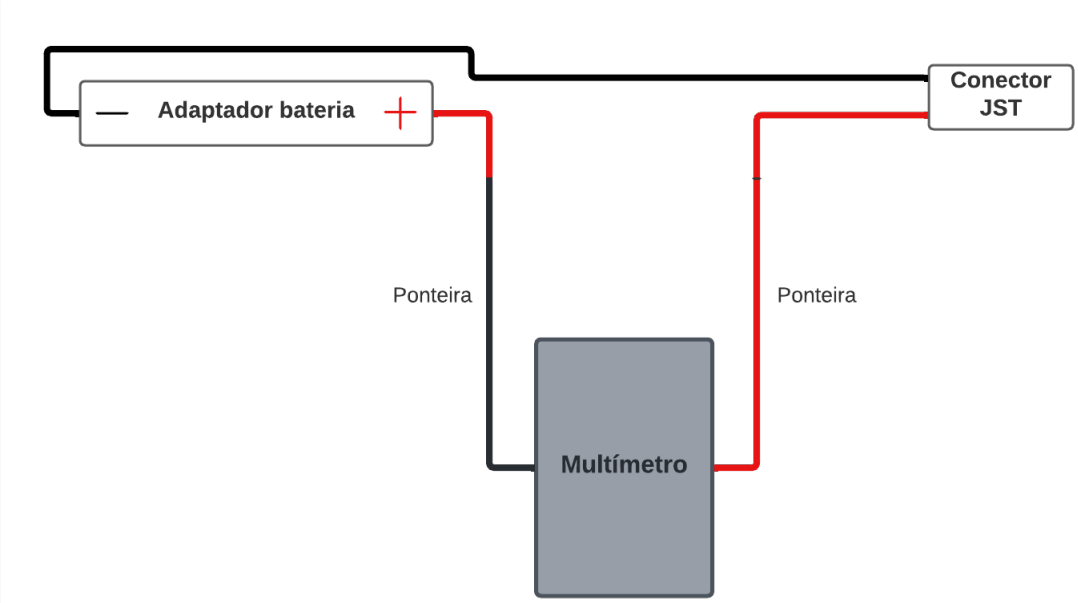
3.5 Aplicação das soluções catalogadas

As soluções catalogadas apresentadas na seção 3.2 foram aplicadas nos dispositivos TTGO e Heltec. Cada solução contém uma variação diferente a ser aplicada, cujas especificações encontram-se descritas nas subseções 3.5.1, 3.5.2, 3.5.3 e 3.5.4. Ao pesquisar a respeito dessas soluções catalogadas, buscando mais informações sobre como implementá-las, outras possibilidades semelhantes surgiram, que são apresentadas nas subseções 3.5.5 e 3.5.6. Por fim, por conta do resultado obtido com essas soluções, uma nova solução foi criada, cujas especificações estão descritas na subseção 3.5.7.

3.5.1 Solução 1 - Função *Hold*

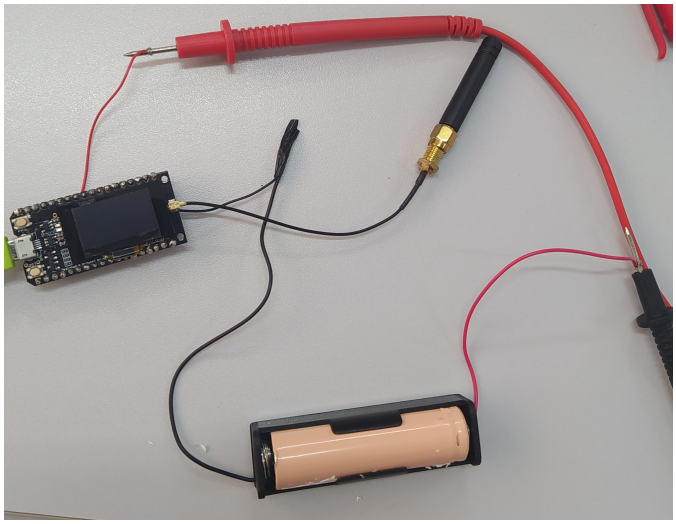
A partir do código original contendo apenas a função de *deep sleep*, essa solução acrescenta a função *hold* apresentada na documentação da GPIO (*General Purpose Input/Output*) e RTC (*Real Time Clock)* ESP32 (ESPRESSIF SYSTEMS, 2016). Para isso, utiliza-se da biblioteca "RTC-IO" (SYSTEMS, 2015), própria para a manipulação dessas funções. A ideia é aplicar essa função ao pino RST do *chip* LoRa antes que o dispositivo entre em *deep sleep* e após esse período de hibernação ela seja revertida. Conforme a especificação do *chip* LoRa SX1276, assim como se encontra na Figura 11, o pino *reset* é NReset e portanto, será ativo quando estiver

Figura 9 – Esquema para o circuito de medição com bateria



Fonte: Autoria própria

Figura 10 – Circuito adaptado para medição de corrente via bateria

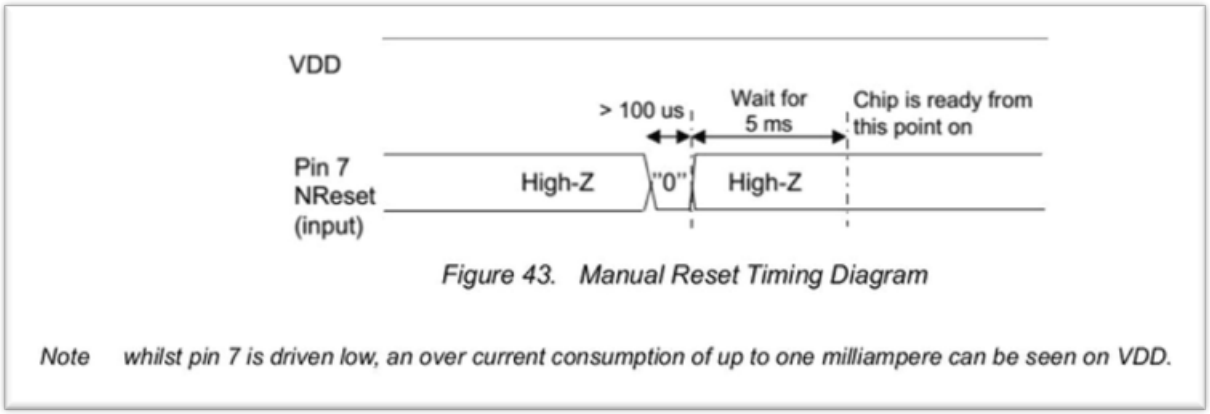


Fonte: Autoria própria

com nível lógico baixo e enquanto o NReset permanecer ativo, o consumo por parte do SX1276 é de apenas 1 mA.

O bloco de código *setup* é o primeiro a ser executado e além disso, é executado toda vez que o dispositivo encerra o bloco *loop* com a função de *deep sleep*. Dessa forma, a função que desabilita o *hold* deve estar definida nesse bloco (linha 5). Além disso, antes da função ser

Figura 11 – *Reset* no SX1276



Fonte: (SEMTECH, 2022)

chamada deve-se definir o pino RST, que corresponde ao pino 14, como um pino apenas de saída, assim como se encontra no Código 2.

Código 2 – Adaptações bloco *setup* da solução 1

```
1 void setup() {
2
3   rtc_gpio_init(GPIO_NUM_14);
4   rtc_gpio_set_direction(GPIO_NUM_14, RTC_GPIO_MODE_OUTPUT_ONLY);
5   rtc_gpio_hold_dis(GPIO_NUM_14);
```

No bloco *loop*, todas as vezes que o dispositivo entrar em *deep sleep* é necessário que a função *hold* seja ativada anteriormente através da função definida na linha 3. Além disso, o estado do pino RST do dispositivo é alterado para nível lógico baixo antes que a função *hold* seja chamada, assim como está disposto na Código 3.

Código 3 – Adaptações bloco *loop* da solução 1

```
1 //Funcao HOLD
2   rtc_gpio_set_level(GPIO_NUM_14, 0); //set low
3   rtc_gpio_hold_en(GPIO_NUM_14);
4
5   LoRa.sleep();
6   esp_deep_sleep_start();
7 }
```

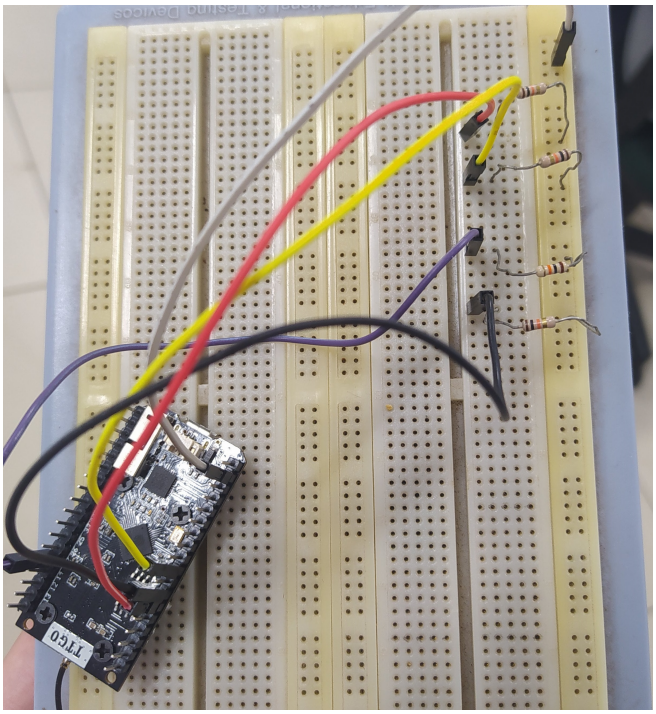
3.5.2 Solução 2 - *Pull up hardware*

De acordo com a solução apresentada no fórum, o intuito desse teste é aplicar resistores *pull-up* aos pinos de comunicação SPI (*Serial Peripheral Interface*) para manter os pinos em

nível lógico alto mesmo quando não há entradas, sendo eles: MISO (pino 19), MOSI (pino 27), SCK (pino 5) e CS (pino 18). O problema abordado é que quando se coloca o ESP32 em *deep sleep*, os pinos GPIO entram em modo de alta impedância e isso acaba sendo um ponto negativo para dispositivos com barramento SPI, pois o consumo de corrente será maior. Dessa forma, o uso de resistores *pull-up* em tais pinos diminuirá a corrente consumida pelo dispositivo durante a hibernação.

Para isso, utilizou-se de uma *protoboard* para conectar os componentes, que foram dispostos em um circuito assim como apresentado na Figura 12. Cada um dos pinos é alimentado pela saída de 3,3 V do próprio dispositivo LoRa, cuja corrente passa por um resistor de 10 kΩ antes de alimentar cada um dos pinos supracitados.

Figura 12 – Circuito de resistores *pull-up*



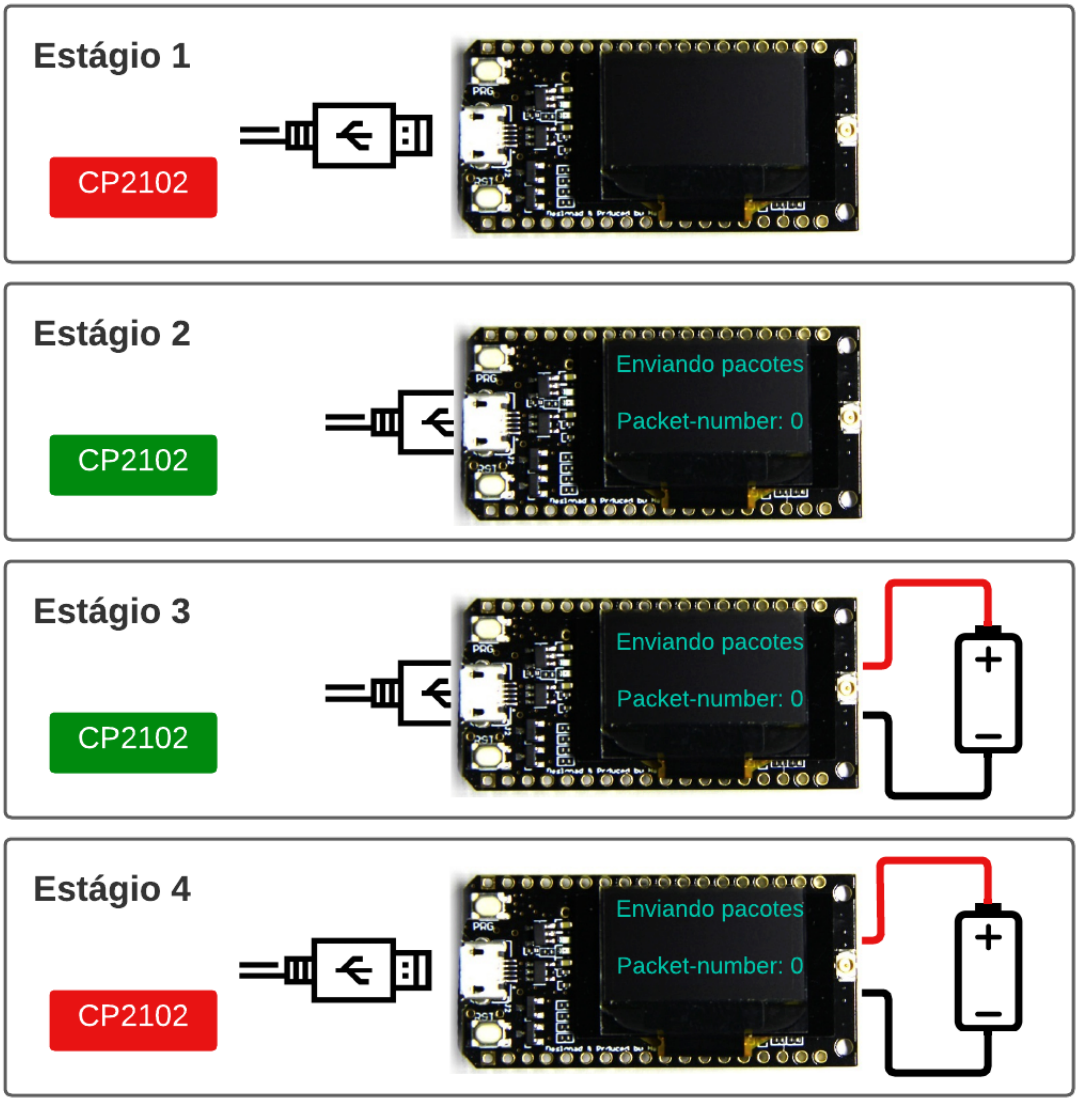
Fonte: Autoria própria

3.5.3 Solução 3 - USB/Bateria

Como se trata de um teste com variação apenas no *hardware* do circuito de teste, não houve alterações no código padrão com a função *deep sleep*. A Figura 13 apresenta o passo a passo a ser seguido para a realização dessa solução, onde ressalta-se que todos os passos são realizados manualmente. O estágio 1 consiste no estado inicial, onde o dispositivo ainda não está sendo alimentado e portanto, permanece desligado. Ao conectar o USB (Estágio 2), a corrente será enviada ao dispositivo e assim, todos seus componentes serão ligados, incluindo seu conversor USB/UART CP2102. Visualmente, a conclusão de tal etapa pode ser identificada

quando o *display* OLED é ligado. Após isso o estágio 3 pode ser executado. Ainda conectado ao USB, o dispositivo é conectado também a uma bateria por meio de sua entrada JST. Por fim, a alimentação via USB do dispositivo pode ser removida (Estágio 4), de modo que ele permaneça sendo alimentado apenas pela bateria.

Figura 13 – Esquema para a realização da solução 4



Fonte: Composição do autor a partir de LILYGO (2020), Fuji (2022), Flaticon (2022)

Essa solução apresenta o conversor CP2102 como a causa para um consumo alto pelo dispositivo, indicando que mesmo que ele não esteja sendo utilizado de forma direta, ainda continua sendo alimentado com um valor médio de 20 mA. A teoria que justifica a realização desse processo é que após desconectar o USB, o conversor é suspenso e portanto, ele terá um consumo médio de apenas 80 μA, conforme se encontra descrito em sua documentação (SILICON LABS, 2017).

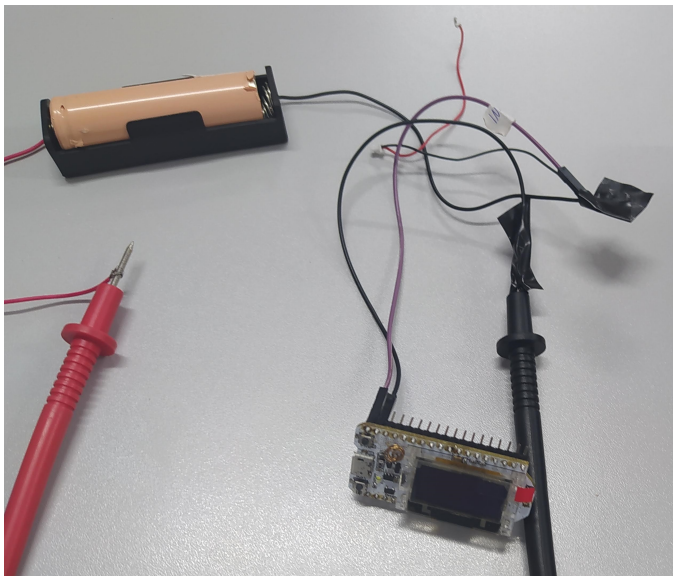
3.5.4 Solução 4 - Bateria

A solução apresentada no fórum consistia na alteração do *hardware* do dispositivo para que ele pudesse suportar a alimentação de forma direta, de modo que o conversor USB/UART não seja utilizado, mas mantendo a possibilidade de utilizar a entrada USB, principalmente para possibilitar o *upload* de código no dispositivo. No entanto, como envolve manipulações diretas no *hardware* do dispositivo, existe a possibilidade de que alguma dessas manipulações não ocorra como esperado, podendo danificar o dispositivo a ponto dele se tornar inutilizável.

Diante disso, para evitar possíveis perdas de dispositivos, optou-se por testar a alimentação via bateria ligando os fios aos pinos GND e 3,3 V de forma direta, com o mesmo intuito de não ativar o conversor USB. Ressalta-se que os testes realizados foram curtos e por isso tal configuração não teve impactos negativos sob o dispositivo. Porém, em casos de utilização real onde a alimentação se dará em um maior espaço de tempo, seria necessário o uso de um conversor LDO, considerando que a tensão da bateria de Li-Ion utilizada é entre 3,6 e 4,2 V.

Como seria necessário acrescentar o multímetro a esse circuito para realizar a medição de corrente, ao final a estrutura para teste ficou conforme se encontra na Figura 14, em que uma das ponteiras é ligada ao fio positivo que sai do adaptador da bateria e a outra ao fio que é conectado no pino 3,3 V do dispositivo.

Figura 14 – Circuito para medição na solução 4



Fonte: Autoria própria

3.5.5 Solução 5 - Função Hold All

Ao analisar as funções disponibilizadas na documentação da GPIO e RTC do ESP32 notou-se que além da função que ativa o *hold* para um único pino, assim como já apresentado

na solução 1, havia uma função responsável por ativar o *hold* para todos os pinos digitais e RTC GPIO. Dessa forma, optou-se por testar essa variação do *hold* acrescentando as funções "gpio_force_hold_all" no bloco *loop* antes da função de *deep sleep* e "gpio_force_unhold_all" no *setup*, com o mesmo propósito inicial de ativar o *hold* antes do modo de hibernação a cada iteração e que tal função fosse desativada sempre que esse período encerrasse.

3.5.6 Solução 6 - Pull up software

Essa solução se baseia na ideia de aplicar resistores *pull-up* aos pinos RTC apresentada na subseção 3.5.2, mas sem alterações no *hardware*. Ao explorar a documentação da GPIO e RTC do ESP32 notou-se que existem funções que realizam o *pull-up* e *pull-down* para um pino via comandos de *software*. Dessa forma, o objetivo dessa solução é aplicar essas funções aos pinos RTC, assim como feito pela outra solução, mas sem alterações no *hardware* do circuito.

No bloco *loop*, antes da função de *deep sleep*, os pinos são alterados para ativos seguindo o conceito de *pull-up* e é chamada a função "rtc_gpio_pullup_en(NUM_PINO)" para cada um dos pinos RTC, conforme se encontra no Código 4.

Código 4 – Adaptações bloco *loop* da solução 6

```
1  //Pull up nos pinos
2  rtc_gpio_set_level(GPIO_NUM_5, 1); //set high
3  rtc_gpio_set_level(GPIO_NUM_19, 1); //set high
4  rtc_gpio_set_level(GPIO_NUM_27, 1); //set high
5  rtc_gpio_set_level(GPIO_NUM_18, 1); //set high
6
7  //Funcao PULL-UP
8  rtc_gpio_pullup_en(GPIO_NUM_5);
9  rtc_gpio_pullup_en(GPIO_NUM_19);
10 rtc_gpio_pullup_en(GPIO_NUM_27);
11 rtc_gpio_pullup_en(GPIO_NUM_18);
```

As alterações no bloco *setup* estão presentes no Código 5, incluindo a definição dos pinos como entrada/saída e a função *pull-down* que reverte o *pull-up* dos pinos que havia sido ativado antes do dispositivo entrar em modo de hibernação.

3.5.7 Solução 7 - Função Hold + USB/Bateria

Por conta dos resultados obtidos com as soluções 1 e 3, foi criada uma nova solução que consiste na realização das duas configurações de forma simultânea. Dessa forma, houve variação no *software* com a inclusão da função *hold* para o pino RST no código assim como detalhado na subseção 3.5.1 e alterações no *hardware* no momento do teste, visto que agora, os testes iniciariam com o dispositivo conectado a um USB e logo após, seriam alimentados via bateria.

Código 5 – Adaptações bloco *setup* da solução 6

```
1  //Disabilita pull up apos deep sleep
2  rtc_gpio_init(GPIO_NUM_5);  //SCK
3  rtc_gpio_init(GPIO_NUM_19); //MISO
4  rtc_gpio_init(GPIO_NUM_27); //MOSI
5  rtc_gpio_init(GPIO_NUM_18); // CS
6
7  rtc_gpio_set_direction(GPIO_NUM_5, RTC_GPIO_MODE_INPUT_OUTPUT);
8  rtc_gpio_set_direction(GPIO_NUM_19, RTC_GPIO_MODE_INPUT_OUTPUT);
9  rtc_gpio_set_direction(GPIO_NUM_27, RTC_GPIO_MODE_INPUT_OUTPUT);
10 rtc_gpio_set_direction(GPIO_NUM_18, RTC_GPIO_MODE_INPUT_OUTPUT);
11
12 rtc_gpio_pulldown_dis(GPIO_NUM_5);
13 rtc_gpio_pulldown_dis(GPIO_NUM_19);
14 rtc_gpio_pulldown_dis(GPIO_NUM_27);
15 rtc_gpio_pulldown_dis(GPIO_NUM_18);
```

4

Resultados

Considerando o código padrão com o uso apenas da função *deep sleep* nos dispositivos TTGO e Heltec alimentados via USB, tem-se um consumo de corrente de 10,5 mA e 10,8 mA, respectivamente.

Todas as 7 soluções desenvolvidas foram testadas nos dois modelos de dispositivos (TTGO e Heltec) seguindo o mesmo padrão de teste. Para cada uma das soluções foram coletadas 5 amostras a fim de evitar valores *outliers*. Como houve pouca variação entre essas 5 amostras, assim como pode ser observado no Apêndice A, considerou-se que eram suficientes para determinar um valor médio. A cada iteração foram coletados o valor do consumo de corrente no momento em que o dispositivo estava enviando mensagens e quando estava em modo de hibernação com o *deep sleep*. Como resultado para os testes realizados, o Quadro 3 apresenta a média do consumo em *deep sleep* a partir das 5 amostras para cada modelo de dispositivo com a respectiva solução.

A partir dos resultados obtidos para a solução 1 nota-se que a utilização da função *hold* apresentou um resultado satisfatório, mesmo que não tenha atingido o que foi descrito no fórum. Por meio de sua variação com o uso da função *hold-all* analisada pela solução 5, nota-se que a mesma melhoria não pôde ser obtida, confirmando que o uso dessa função deve ser exclusivo ao pino *reset* para que as mudanças sejam positivas.

As soluções 2 e 6 compreendem as tentativas de uso de *pull-up* nos pinos RTC por *hardware* e *software*, respectivamente. A partir dos resultados obtidos conclui-se que essa estratégia não proporciona melhoria alguma, visto que o consumo se manteve praticamente idêntico a solução original. As pequenas variações observadas foram de no máximo 0,2 mA, que podem ser ocosionados por possíveis ruídos durante a medição devido a precisão do multímetro.

Iniciando os testes com a alimentação via bateria sem a transição do USB, obtém-se o valor original de consumo, mesmo que o conversor USB/UART não esteja sendo utilizado de forma direta. Uma possibilidade para justificar esse consumo não esperado é que mesmo que a

Quadro 3 – Consumo em modo *deep sleep* pelos dispositivos (em mA)

Solução	**TTGO**	**Heltec**
Inicial - apenas *deep sleep*	10,5	10,8
1 - Função *Hold*	8,8	9,2
2 - *Pull up hardware*	10,3	10,9
3 - USB/Bateria	3,1	2,4
4 - Bateria	10,4	0,9
5 - Função *Hold All*	10,4	10,7
6 - *Pull up software*	10,3	10,8
7 - Função *Hold*+USB/Bateria	1,4	0,8

Fonte: Autoria própria

fonte energética não seja o USB, o dispositivo continua em busca de uma conexão USB e assim o circuito do conversor continua sendo alimentado. Dessa forma, apenas utilizar a bateria não é suficiente para suspender o conversor. A partir do resultado obtido para a solução 3 nota-se que a transição do USB para bateria apresentou um resultado bastante satisfatório e portanto, acredita-se que o conversor é suspenso nessas condições. Outra possibilidade levantada para a suspensão desse conversor seria através de comandos lógicos por meio do próprio microcontrolador, ficando tal análise para ser realizada em futuros trabalhos.

O uso da Heltec com a solução 4 proporcionou um resultado bastante relevante, sendo até mesmo o segundo melhor resultado atingido para esse modelo, com apenas 900 μA. No entanto, esse ótimo desempenho não pôde ser observado para o TTGO, visto que a alteração proposta pela solução não afetou em seu consumo e se manteve conforme a solução inicial. Como não se tem acesso ao esquemático do dispositivo TTGO, não há como comprovar a razão para que essa discrepância aconteça. Dessa forma, acredita-se que análises com foco no circuito desse dispositivo possam ser realizadas em trabalhos futuros a fim de entender na prática como seus circuitos internos operam, por meio de dispositivos de medição como o multímetro ou osciloscópio.

Assim como esperado, a solução 7, por ser uma junção das soluções 1 e 3 que de forma isolada já proporcionaram melhorias, apresentou um resultado bastante positivo. Para ambos os modelos de dispositivo essa foi a solução que apresentou o melhor benefício, diminuindo drasticamente o valor de consumo se comparado ao original apenas com a função de *deep sleep*.

Para o modelo Heltec, nota-se que a solução 4 também proporcionou ótimos resultados, de modo que tal solução também possa ser aplicada em futuras aplicações, considerando que a diferença de 0,1 mA possa se dar por conta de possíveis ruídos no momento do experimento.

Aplicando os valores de consumo em *deep sleep* obtidos com os testes das soluções no modelo analítico desenvolvido pode-se estimar o tempo de vida de uma bateria para cada cenário. Essa estimativa leva em consideração um SF9 e uma bateria de 2000 mAh, desconsiderando seu fator de decaimento natural em relação ao tempo. A relação do tempo de vida mediante a aplicação das soluções no dispositivo TTGO está apresentada no Quadro 4, enquanto o resultado obtido para o tempo de vida do dispositivo Heltec se encontra no Quadro 5.

Quadro 4 – Simulação do tempo de vida da bateria com dispositivo TTGO (em dias)

	Soluções						
Taxa de envio	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1 em 1 minuto	8,66	7,5	21,09	7,43	7,43	7,5	36,87
5 em 5 minutos	9,3	7,96	25,48	7,88	7,88	7,96	53,01
10 em 10 minutos	9,38	8,03	26,16	7,95	7,95	8,03	56,08

Fonte: Autoria própria

Para a TTGO, observa-se que as seis primeiras soluções possuem pouca variação no tempo de vida ao alterar a taxa de envio. Dessa forma, há pouco benefício quanto ao tempo de vida desse dispositivo se comparado ao benefício de se ter uma comunicação mais constante. No entanto, a partir do resultado obtido para a solução 7, nota-se que a variação dessa taxa afeta mais drasticamente, principalmente levando em consideração a menor taxa de envio analisada, que se comparada às demais chega a quase 20 dias de diferença.

Quadro 5 – Simulação do tempo de vida da bateria com dispositivo Heltec (em dias)

	Soluções						
Taxa de envio	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1 em 1 minuto	8,32	7,11	25,6	47,28	7,24	7,43	50,11
5 em 5 minutos	8,9	7,53	32,41	77,7	7,67	7,88	85,68
10 em 10 minutos	9	7,59	33,53	84,5	7,73	7,95	94,08

Fonte: Autoria própria

Com o modelo Heltec nota-se que as soluções 4 e 5 possuem maior disparidade quanto ao tempo de vida desse dispositivo mediante a variação da taxa de envio, sendo portanto, mais indicado o uso de taxas de envio maiores para um melhor resultado por parte do dispositivo ao utilizar tais soluções. Comparada às outras soluções, a solução 3 apresentou um resultado levemente superior, porém, a variação em sua taxa de envio teve pouco impacto. Quanto ao tempo de vida obtido para as demais soluções, como foram as soluções que tiveram os maiores consumos de corrente, verifica-se uma baixa variação entre as taxas de envio analisadas e um baixo tempo de vida ao utilizá-las.

Diante desses resultados, foi realizada uma comparação entre o melhor resultado atingido dentre as soluções analisadas (solução 7) e o valor referente ao código inicial padrão, que está retratada no Quadro 6. A partir do tempo de vida entre cada um dos cenários, nota-se que para ambos os dispositivos foi possível obter um tempo de vida muito superior ao que se tem com a solução original, principalmente para o modelo Heltec. Essa diferença se torna ainda maior quando se aumenta o tempo entre os envios de pacotes, visto que com uma taxa de 10 minutos tem-se um uma diferença ainda maior entre o resultado obtido com a solução 7 e a solução inicial, se comparado às outras taxas analisadas.

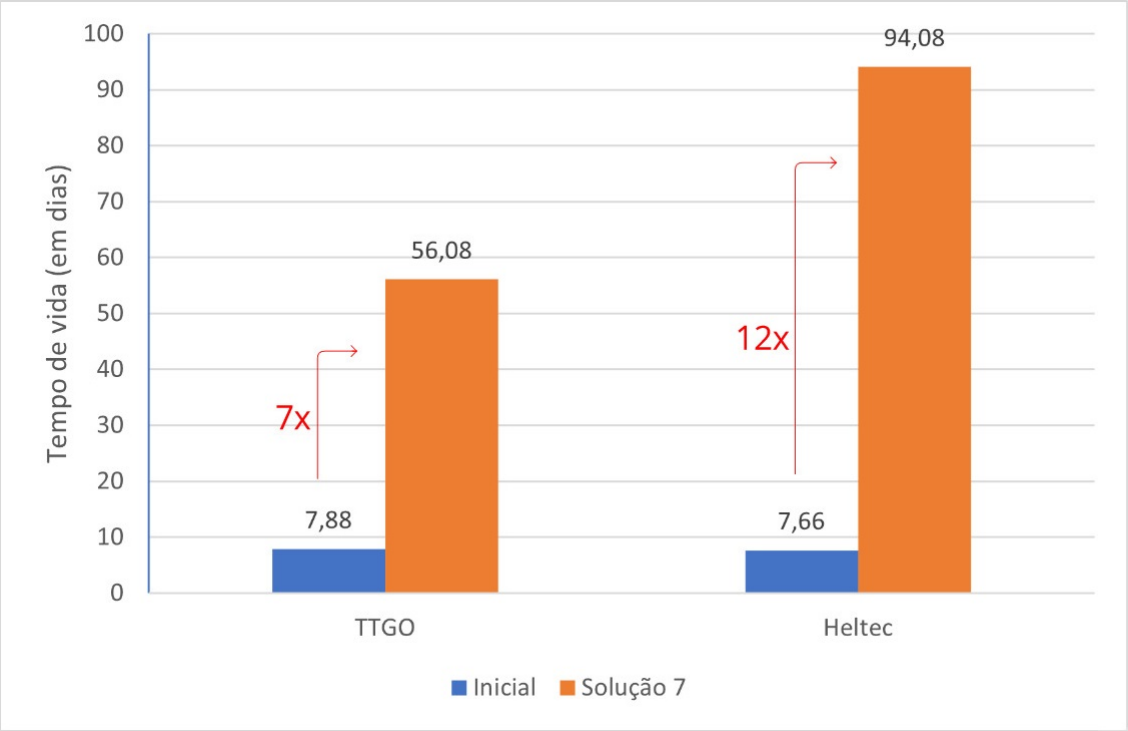
Quadro 6 – Comparação do tempo de vida da bateria (em dias): solução 7 x código inicial

	TTGO		**Heltec**	
Taxa de envio	**Inicial**	**Solução 7**	**Inicial**	**Solução 7**
1 em 1 minuto	7,37	36,87	7,18	50,11
5 em 5 minutos	7,82	53,01	7,6	85,68
10 em 10 minutos	7,88	56,08	7,66	94,08

Fonte: Autoria própria

Na Figura 15 encontra-se um gráfico comparativo entre o tempo de vida dos dispositivos apenas com a implementação inicial e aplicando a solução 7, de acordo com os resultados obtidos para uma taxa de envio de 5 minutos. Como pode ser observado, ao aplicar a solução 7 no dispositivo TTGO tem-se um ganho de 7 vezes para o tempo de vida da bateria e com o dispositivo Heltec, tem-se uma duração 12 vezes maior. Tais resultados demonstram o benefício de utilizar tal solução em uma aplicação.

Figura 15 – Comparação tempo de vida inicial x solução 7



Fonte: Autoria própria

5

Considerações Finais

O uso de dispositivos de IoT em contextos que envolvem a alimentação energética por meio de uma bateria dependem diretamente de um baixo consumo para que a manutenção seja feita com baixa periodicidade. Dessa forma, um consumo acima de 10 mA faz com que o tempo de vida do dispositivo seja extremamente baixo e por conta disso deve ser evitado. Consequentemente, torna-se indispensável a busca por novas alternativas que possam atenuar esse consumo.

A partir das soluções catalogadas e dos testes realizados com o intuito de avaliar seu comportamento na prática, nota-se que a solução 7, que corresponde a junção das soluções que acrescentam a função *hold* ao código e a transição de forma manual quanto a alimentação do dispositivo de USB para bateria, foi a que apresentou melhor resultado para ambos os dispositivos analisados.

Aplicando essa solução a dispositivos LoRa é necessário se atentar tanto às alterações no *software* quanto no *hardware*. Referente a alteração no *software*, ela deverá ser realizada uma única vez no momento em que será feito o *upload* do código desenvolvido no dispositivo a ser utilizado. Esse cenário é o ideal, pois evita manutenções no código e o dispositivo pode ser utilizado sem preocupações. Quanto a alteração no *hardware*, enquanto o dispositivo permanecer ligado não serão necessárias alterações. Porém, na queda de energia ou em situações que o dispositivo será resetado, como ao trocar sua bateria dentro da periodicidade necessária, as condições apresentadas por essa solução devem ser executadas novamente. Apesar de aumentar as operações que devem ser realizadas a cada troca de bateria, considera-se que o benefício proporcionado referente à redução no consumo seja vantajoso e por isso justifique sua utilização.

Portanto, considerando um cenário real de utilização de dispositivos LoRa TTGO ou Heltec, sugere-se fortemente a aplicação das alterações apresentadas pela solução 7 para que se tenha um baixo consumo de corrente durante o modo de hibernação e consequentemente, um maior tempo de vida para a bateria que alimenta esse dispositivo. Além disso, caso o uso

do dispositivo seja em cenários que não possuem muita criticidade quanto ao monitoramento frequente, recomenda-se também o uso de uma taxa de envio um pouco maior para proporcionar um resultado ainda mais satisfatório. Assim, é possível usufruir dos benefícios apresentados pela tecnologia com uma baixa taxa de manutenção no dispositivo.

Um dos problemas encontrados para que não seja atingido um baixo consumo de corrente pelo dispositivo, mesmo com o uso da função *deep sleep* que teoricamente seria suficiente para tal resultado, é quanto ao custo para alimentar o conversor USB/UART, mesmo que ele não esteja sendo utilizado de forma direta. Diante disso, como um futuro trabalho tem-se a análise prática dos circuitos de alimentação do dispositivo, considerando a habilitação ou não da alimentação do conversor USB/UART via uma chave, que demandaria cortar a trilha de alimentação na placa. Além disso, poderia ser analisada a possibilidade de desativar tal conversor através do envio de um sinal de suspensão pelo microcontrolador. Assim, poderão ser exploradas outras possibilidades que evitem a necessidade de intervenção manual (ligar na USB e depois retirar) para reduzir o consumo.

Além disso, como não se tem disponível um esquemático próprio para a versão 1.0 do modelo TTGO utilizado, há espaço para análises futuras quanto à corrente que está percorrendo cada um dos circuitos que alimentam componentes internos. Filtrando exatamente esse consumo, será possível encontrar respostas mais concretas para o motivo de tais resultados terem sido obtidos para esse dispositivo.

Referências

ALCANTARA, R. A. da S.; OYAMADA, M. S. Estudo de alcance e consumo de energia em uma rede LoRa na plataforma ESP32. In: *6º Encontro Anual de Iniciação Científica e Inovação da Unioeste*. Cascavel: [s.n.], 2020. Citado na página 13.

ARDUINO. *Arduino IDE*. 2022. Disponível em: <https://www.arduino.cc/en/software>. Acesso em: 13 mar 2022. Citado na página 27.

BARBOSA, R. M. R. *Sistema de monitorização de consumo de água utilizando tecnologia Sigfox*. Dissertação (Dissertação de Mestrado) — Faculdade de Engenharia da Universidade do Porto, Porto - Portugal, Julho 2017. Citado na página 13.

BAUMGäRTNER, L. et al. Environmental Monitoring Using Low-Cost Hardware and Infrastructureless Wireless Communication. In: *2018 IEEE Global Humanitarian Technology Conference (GHTC)*. [S.l.: s.n.], 2018. p. 1–8. Citado na página 13.

CHIPOWER TECHNOLOGY. *CE6260 Series*. [S.l.], 2022. Disponível em: <https://datasheetspdf.com/pdf-file/1304348/Chipower/CE6260/1>. Citado na página 21.

CRUZ, J. E. C. D. L. et al. LoRaWAn Instant Messaging System For Areas Without Telecommunication Services and Disaster Environment. In: *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. [S.l.: s.n.], 2021. p. 1–4. Citado na página 13.

ESPRESSIF SYSTEMS. *GPIO & RTC GPIO*. [S.l.], 2016. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html>. Citado na página 30.

ESPRESSIF SYSTEMS. *ESP32 Series*: datasheet. Shanghai, 2022. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Citado 2 vezes nas páginas 19 e 22.

FLATICON. *USB Cable*. 2022. Disponível em: <https://www.flaticon.com/free-icon/usb-cable_1330687>. Acesso em: 14 jul 2022. Citado na página 34.

FUJI, M. *Pilhas e baterias*. 2022. Disponível em: <https://www.montefuji.com.br/categoria-produto/pilhas-e-baterias/>. Acesso em: 14 jul 2022. Citado na página 34.

HELTEC AUTOMATION. *Heltec LoRa node products difference table*. [S.l.], 2020. Disponível em: <https://docs.heltec.cn/#/en/products/lora/lora_node/heltec_lora_node_list>. Citado na página 21.

HIVEEYES. *Heltec WiFi LoRa 32*. 2021. Disponível em: <https://community.hiveeyes.org/t/heltec-wifi-lora-32/3125>. Acesso em: 13 mar 2022. Citado na página 21.

HOPE MICROELECTRONICS. *RFM95/96/97/98(W)*: low power long range transceiver module. China, 2006. Disponível em: <https://cdn.sparkfun.com/assets/learn_tutorials/8/0/4/RFM95_96_97_98W.pdf>. Citado 2 vezes nas páginas 21 e 22.

KODALI, R. K.; KUTHADA, M. S.; BORRA, Y. K. Y. LoRa Based Smart Irrigation System. In: *2018 4th International Conference on Computing Communication and Automation (ICCCA)*. [S.l.: s.n.], 2018. p. 1–5. Citado na página 13.

LEE, A. *ESP32 WiFi Lora Board*: deep-sleep power consumption. 2017. Disponível em: <https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series/issues/6>. Acesso em: 04 mar 2022. Citado na página 27.

LIANDO, J. C. et al. Known and Unknown Facts of LoRa: Experiences from a Large-scale Measurement Study. *ACM Transactions on Sensor Networks (TOSN)*, ACM New York, NY, USA, v. 15, n. 2, p. 1–35, 2019. Citado na página 18.

LILYGO. *TTGO-LORA32*: OLED_LoRa_Sender. 2018. Disponível em: <https://github.com/LilyGO/TTGO-LORA32/blob/master/OLED_LoRa_Sender/OLED_LoRa_Sender.ino>. Acesso em: 17 ago 2022. Citado na página 27.

LILYGO. *TTGO LORA32*. China, 2020. Disponível em: <http://www.lilygo.cn/prod_view.aspx?TypeId=50060&Id=1326&FId=t3:50060:3>. Citado 3 vezes nas páginas 13, 20 e 34.

LORA ALLIANCE. *LoRaWAN What is it?*: a technical overview of LoRa and LoRaWAN. San Ramon, 2015. Disponível em: <https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf>. Citado 4 vezes nas páginas 13, 16, 17 e 18.

LOUREIRO, A. et al. Redes de sensores sem fio. *Simpósio Brasileiro de Redes de Computadores*, v. 21, 01 2003. Citado na página 12.

MICRONE. *ME6211*. [S.l.], 2022. Disponível em: <https://datasheet.lcsc.com/szlcsc/Nanjing-Micro-One-Elec-ME6211C33M5G-N_C82942.pdf>. Citado na página 20.

NETO, E. *Comunicação LoRa com arduino ponto a ponto (P2P)*. 2018. Disponível em: <https://www.filipeflop.com/blog/comunicacao-lora-com-arduino-p2p/>. Acesso em: 12 mar 2022. Citado na página 22.

NETWORK, T. T. *Frequency Plans*. 2021. Disponível em: <https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/>. Acesso em: 13 jul 2022. Citado na página 17.

ORTIZ, F. M. *Análise de desempenho de uma rede sem fio de baixa potência e longo alcance para internet das coisas*. Dissertação (Dissertação de Mestrado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro - RJ, Março 2018. Citado 2 vezes nas páginas 15 e 17.

PEREIRA, M.; CRUVINEL, P. Desenvolvimento de um sistema de coleta automática de dados agrícolas baseado em rede LoRa e no microprocessador ESP32. In: *Anais da X Escola Regional de Informática de Mato Grosso*. Porto Alegre, RS, Brasil: SBC, 2019. p. 43–48. ISSN 2447-5386. Disponível em: <https://sol.sbc.org.br/index.php/eri-mt/article/view/8592>. Citado na página 13.

REDZUAN, A. A. A. M. et al. Wireless Ammonia Sensor System for Distributed Wireless Monitoring Platform using Heltec Wifi LoRa 32 (V2). *Progress in Engineering Application and Technology*, v. 2, n. 1, p. 502–513, jun. 2021. Disponível em: <https://publisher.uthm.edu.my/periodicals/index.php/peat/article/view/910>. Citado na página 13.

SANT'ANA, J. M. de S. *Redes LoRaWAN*: implantação e desenvolvimento de aplicações. Dissertação (Trabalho de Conclusão de Curso) — Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, São José - SC, Março 2017. Citado 2 vezes nas páginas 16 e 17.

SANTOS, B. P. et al. Internet das coisas. In: ____. 7. ed. Belo Horizonte: UFMG, 2017. cap. Internet das coisas: da teoria à prática, p. 1–50. Citado na página 12.

SEMTECH. *AN1200.22*: LoRa Modulation Basics. [S.l.], 2015. Disponível em: <https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf>. Citado 2 vezes nas páginas 15 e 16.

SEMTECH. *Datasheet SX1276/77/78/79*. Camarillo, 2022. Disponível em: <https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE>. Citado 2 vezes nas páginas 18 e 32.

SEMTECH CORPORATION. *Understanding the LoRa Adaptive Data Rate*. [S.l.], 2019. Disponível em: <https://lora-developers.semtech.com/uploads/documents/files/Understanding_LoRa_Adaptive_Data_Rate_Downloadable.pdf>. Citado na página 24.

SILICON LABS. *CP2102/9*. [S.l.], 2017. Disponível em: <https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf>. Citado 2 vezes nas páginas 20 e 34.

SYSTEMS, E. *ESP-IDF*. 2015. Disponível em: <https://github.com/espressif/esp-idf/blob/5e6cffbb14/components/driver/include/driver/gpio.h>. Acesso em: 01 jul 2022. Citado na página 30.

TEIXEIRA, G. B.; ALMEIDA, J. V. P. de. *Rede LoRa e protocolo LoRaWAN aplicados na agricultura de precisão no Brasil*. Dissertação (Trabalho de Conclusão de Curso) — Universidade Tecnológica Federal do Paraná, Ponta Grossa - PR, Agosto 2017. Citado 2 vezes nas páginas 15 e 16.

TEXAS INSTRUMENTS. *Datasheet INA219*. [S.l.], 2015. Disponível em: <https://datasheetspdf.com/pdf/1417488/etcTI/INA219/1>. Citado na página 24.

VIDAL, V. *10 protocolos de IoT que você deveria conhecer*. 2017. Disponível em: <https://www.profissionaisti.com.br/10-protocolos-de-iot-que-voce-deveria-conhecer/>. Acesso em: 02 mar 2022. Citado 2 vezes nas páginas 12 e 13.

XINYUAN-LILYGO. *LilyGo LoRa Series*. 2020. Disponível em: <https://github.com/Xinyuan-LilyGO/LilyGo-LoRa-Series>. Acesso em: 05 jul 2022. Citado na página 20.

Apêndices

APÊNDICE A – Amostras dos testes das soluções

Solução 1 - Função Hold (Quadro 7):

Quadro 7 – Consumo em modo *deep sleep* (mA) com a solução 1

TTGO	Heltec
8,8	9,2
8,8	9,2
8,8	9,2
8,8	9,2
8,8	9,2

Fonte: Autoria própria

Solução 2 - Pull up hardware (Quadro 8):

Quadro 8 – Consumo em modo *deep sleep* (mA) com a solução 2

TTGO	Heltec
10,3	10,9
10,3	10,8
10,4	10.9
10,3	10,9
10,3	10,8

Fonte: Autoria própria

Solução 3 - USB/Bateria (Quadro 9):

Quadro 9 – Consumo em modo *deep sleep* (mA) com a solução 3

TTGO	**Heltec**
3,1	2,4
3,1	2,4
3,1	2,4
3,1	2,4
3,1	2,4

Fonte: Autoria própria

Solução 4 - Bateria (Quadro 10):

Quadro 10 – Consumo em modo *deep sleep* (mA) com a solução 4

TTGO	**Heltec**
10,4	0,9
10,5	0,9
10,4	0,9
10,4	1
10,5	0,9

Fonte: Autoria própria

Solução 5 - Função Hold All (Quadro 11):

Quadro 11 – Consumo em modo *deep sleep* (mA) com a solução 5

TTGO	**Heltec**
10,4	10,9
10,4	10,8
10,4	10,7
10,4	10,7
10,4	10,7

Fonte: Autoria própria

Solução 6 - Pull up Software (Quadro 12):

Quadro 12 – Consumo em modo *deep sleep* (mA) com a solução 6

TTGO	**Heltec**
10,4	10,7
10,3	10,9
10,3	10,8
10,3	10,8
10,3	10,8

Fonte: Autoria própria

Solução 7 - Função Hold + USB/Bateria (Quadro 13):

Quadro 13 – Consumo em modo *deep sleep* (mA) com a solução 7

TTGO	**Heltec**
1,4	0,8
1,4	0,8
1,4	0,8
1,4	0,8
1,4	0,8

Fonte: Autoria própria