UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA

FRANCISCO LÚCIO RODRIGUES DE ARAÚJO

DESENVOLVIMENTO DE UM SISTEMA PARA MEDIÇÃO DO NÍVEL DE ÁGUA A DISTÂNCIA UTILIZANDO O PROTOCOLO LORAWAN PARA ESTUDO DA VIABILIDADE DE MONITORAMENTOS DOS NÍVEIS DOS IGARAPÉS

Manaus
2019

FRANCISCO LÚCIO RODRIGUES DE ARAÚJO

DESENVOLVIMENTO DE UM SISTEMA PARA MEDIÇÃO DO NÍVEL DE ÁGUA A DISTÂNCIA UTILIZANDO O PROTOCOLO LORAWAN PARA ESTUDO DA VIABILIDADE DE MONITORAMENTOS DOS NÍVEIS DOS IGARAPÉS

Pesquisa desenvolvida durante a disciplina de Trabalho de conclusão de curso II e apresentada à banca avaliadora do Curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas como pré-requisito para a obtenção do título de Engenheiro em Eletrônica.

Orientador: André Luiz Printes, Me.

Manaus
2019

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia – EST

Reitor:
Cleinaldo de Almeida Costa, Dr.

Vice-Reitor:
Cleto Cavalcante de Souza Leal, Me.

Diretora da Escola Superior de Tecnologia:
Ingrid Sammyne Gadelha Figueiredo, Me.

Coordenador do Curso de Engenharia:
Daniel Gusman del Rio, Dr.

Banca Avaliadora composta por: *Data da defesa: 09/12/2019*
Prof. André Luiz Printes, Me *(Orientador)*
Prof. Jozias Parente de Oliveira, Dr (Avaliador 1)
Prof. Fábio de Souza Cardoso, Dr (Avaliador 2)

CIP – Catalogação na Publicação

Araújo, Francisco

Desenvolvimento de um sistema para medição do nível de água a distância utilizando o protocolo LoRawan para estudo da viabilidade de monitoramentos dos níveis dos igarapés / Francisco Lúcio Rodrigues de Araújo;
[orientado por] Prof. André Luiz Printes – Manaus: 2019.
61 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica).
Universidade do Estado do Amazonas, 2019.

1. monitoramento de níveis de água. 2 *LPWAN*, 3. LoRaWan 4. Sensor Ultrassônico, 5. Arduino IDE 6. LabVIEW.
I Printes, André.

FRANCISCO LÚCIO RODRIGUES DE ARAÚJO

DESENVOLVIMENTO DE UM SISTEMA PARA MEDIÇÃO DO NÍVEL DE ÁGUA A DISTÂNCIA UTILIZANDO O PROTOCOLO LORAWAN PARA ESTUDO DA VIABILIDADE DE MONITORAMENTOS DOS NÍVEIS DOS IGARAPÉS

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade Estadual do Amazonas, como pré-requisito para a obtenção do título de Engenheiro em Eletrônica.

Nota obtida: _____ (_____)

Aprovada em _____/_____/_____.

Área de concentração: Microcontroladores, comunicação sem fio.

BANCA EXAMINADORA

_____
Orientador: André Luiz Printes, Me.

_____
Avaliador: Jozias Parente de Oliveira, Dr.

_____
Avaliador: Fábio de Souza Cardoso, Dr.

Manaus
2019

DEDICATÓRIA

Aos meus pais e a todos os meus familiares e amigos que direta ou indiretamente me apoiaram nesta caminhada. Meu muito obrigado.

RESUMO

O presente documento tem a finalidade de apresentar o desenvolvimento de um sistema de monitoramento de níveis de água que pode ser aplicado para supervisionar os igarapés em locais onde há risco da ocorrência de enchentes. Este sistema tem a capacidade de enviar mensagens a longas distâncias com baixo consumo de energia, de acordo com o padrão *LPWAN*. O sistema é apresentado por meio de protótipo e a transmissão de dados é executada por meio de dispositivos *LoRaWan*. Primeiramente mostra-se a fundamentação teórica das tecnologias utilizadas neste projeto: Medição de nível, tecnologia *LoRaWan*, parâmetros de configuração *LoRa*, Sensor ultrassônico, Arduino IDE e *LabVIEW*. Em seguida, em Métodos e Materiais, são apresentadas as etapas de desenvolvimento do projeto e os materiais utilizados. Posteriormente, em Implementação do projeto, há a exposição do que foi executado durante o projeto. Nesta etapa são apresentados os fluxogramas dos programas desenvolvidos na Arduino IDE para a realização dos testes. Foram realizados testes de distâncias com envio de pacotes por meio da tecnologia LoRa. Em seguida, fez-se a implementação do protótipo para simular o aumento dos níveis de água. Nesta etapa também foram feitos testes de consumo de energia dos dispositivos utilizados para medir o nível. E posteriormente é apresentado o desenvolvimento da interface gráfica para visualização desses níveis. Na sequência, em Resultados Obtidos, são mostrados e analisados os dados obtidos dos testes. Finalmente, conclui-se por meio dos dados obtidos que a tecnologia *LoRaWan* é uma alternativa viável para monitoramento a distância dos níveis dos igarapés em locais com risco de inundações. Os resultados apresentaram bom desempenho no envio de mensagens a distâncias superiores a 4,5 Km ou 2,8 milhas e consumo máximo de corrente de 186 mA no momento da transmissão.

Palavras-chaves: monitoramento de níveis de água, *LPWAN*, *LoRaWan*, sensor ultrassônico, Arduino IDE, *LabVIEW*.

ABSTRACT

The purpose of this document is to present the development of a water level monitoring system that can be applied to supervise streams where there is a risk of flooding. This system has the ability to send long distance messages with low power consumption according to the LPWAN standard. The system is prototyped and data transmission is performed via LoRaWan devices. Firstly, we show the theoretical foundation of the technologies used in this project: Level measurement, LoRaWan technology, LoRa configuration parameters, Ultrasonic sensor, Arduino IDE and LabVIEW. Then, in Methods and Materials, are presented the stages of development of the project and the materials used. Later, in Project Implementation, there is an exposition of what was performed during the project. At this stage, are presented the flowcharts of the programs developed in the Arduino IDE for the performance of the tests. Distance testing with packet sending was performed using LoRa technology. Then the prototype was implemented to simulate the increase of water levels. At this stage, power consumption tests were also performed on the devices used to measure the level. And later is presented the development of the graphical interface for visualization of these levels. Then, in the Obtained Results, the data obtained from the tests are shown and analyzed. Finally, it is concluded from the data obtained that LoRaWan technology is a viable alternative for remote monitoring of stream levels at flood risk sites. The results showed good performance in sending messages at distances of more than 4 km or 2,8 miles and maximum current consumption of 186 mA at the time of transmission.

Keywords: Water level monitoring, LPWAN, LoRaWan, ultrasonic sensor, Arduino IDE, LabVIEW.

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇAO

A região Amazônica possui baixo relevo e é banhada por uma quantidade bastante relevante de rios e afluentes, lagos e igarapés. É comum a habitação de muitas famílias em locais de riscos de alagamento. Esses riscos se intensificam ainda mais no tempo chuvoso. Sendo assim, foi proposto para esta pesquisa o seguinte tema: desenvolvimento de um sistema para medição do nível de água a distância utilizando o protocolo LoRawan para estudo da viabilidade de monitoramentos dos níveis dos igarapés. Para realização deste monitoramento foi utilizado protocolo de rede sem fio *Long Range Wide Area Networking* (*LoRaWAN*) que é uma tecnologia de comunicação sem fio baseada no conceito *Low Power Wide Area Networking* (*LPWAN*). O conceito LPWAN consiste em tecnologias capazes de enviar mensagens a longas distâncias com baixo consumo de energia (GARCIA;KLEINSCHMIDT, 2017)

O presente trabalho expõe o problema da carência de trabalhos voltados para monitoramento de locais com riscos de enchentes utilizando tecnologias que apresentam baixo consumo de energia e sejam capazes de enviar dados a longas distâncias. Nesta pesquisa, foi testada a hipótese de que seja possível o desenvolvimento de um sistema de monitoramento sem fio dos níveis dos igarapés. O sistema é desenvolvido com dispositivos baseados na filosofia LPWAN, ou seja, é capaz de se comunicar a longas distâncias e ter grande autonomia devido ao baixo consumo de energia. O sistema também é capaz de exibir alertas em um monitor para que seja possível salvaguardar a população que mora em áreas de risco.

Para teste e validação da hipótese acima, foram implementados os seguintes objetivos. Primeiro fez-se o estudo da programação dos parâmetros dos dispositivos LoRa. Em seguida, a realização dos testes de alcance e envios de mensagens para que se possa mensurar o alcance máximo e a atenuação dos sinais. Também foram realizados testes de consumo de energia para fonte de energia que garanta a autonomia.

Após a etapa de testes, foi desenvolvido um protótipo para simular as cheias e uma unidade receptora composta por um receptor LoRa e um computador onde foi desenvolvida uma interface gráfica que é capaz de plotar os níveis de água e gerar alarmes.

Esta pesquisa se justifica, pelo fato de que os problemas com inundações são recorrentes nas áreas urbanas e rurais da cidade de Manaus, devido ao grande número de igarapés que cortam a cidade. Sendo assim o monitoramento dos igarapés em áreas de riscos poderá auxiliar na prevenção de catástrofes. Complementarmente, pelo fato dos dispositivos LoRa serem uma tecnologia emergente, o estudo destes dispositivos poderá servir como base para outras aplicações também de grande importância.

Realizou-se uma revisão teórica que contempla os assuntos de medição de níveis por ultrassom, o funcionamento do sensor de ultrassom, o protocolo de comunicação *LoRaWan* e tópicos sobre alguns parâmetros importante da comunicação LoRa. Também foram abordados os softwares que foram empregados para programação do sistema microcontrolado utilizado nesta pesquisa.

Este trabalho está dividido nos seguintes capítulos: Referencial teórico, Métodos e Materiais, implementação do projeto e resultados obtidos.

No primeiro capítulo é abordada uma revisão dos assuntos referentes aos dispositivos e softwares utilizados e conceitos do protocolo de comunicação LoRaWan. No segundo capítulo são mostrados os métodos utilizados para se obter o objetivo final. No terceiro capítulo, implementação do projeto, é abordada a execução dos passos citados no capítulo de métodos. Mostra-se todos os testes executados, para se obter os dados de alcance e consumo de energia. Também é apresentada a construção de um protótipo e a criação de uma interface para simulação e visualização dos níveis de água. Os programas feitos durante a implementação do projeto, são mostrados nos Apêndices.

No quarto capítulo, Resultados obtidos, mostram-se os resultados dos experimentos realizados na implementação, e faz-se análises baseadas no referencial teórico e no conhecimento adquirido durante todo o processo de pesquisa e implementação do projeto.

E por fim é apresentada a conclusão que aborda a análise do comportamento dos dispositivos LoRa e a discussão acerca da utilização destes dispositivos para a aplicação em monitoramento de níveis dos igarapés.

1 REFERENCIAL TÉORICO

1.1 MEDIÇÃO DE NÍVEL

O nível é definido como sendo a altura do conteúdo líquido ou sólido presente em um reservatório. A medição de níveis pode ser feita de forma direta ou indireta.

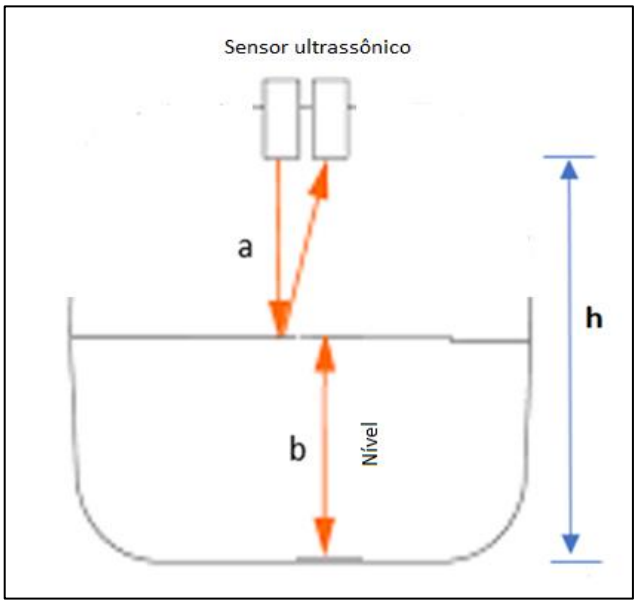
Na medição direta é tomada como referência a posição mais elevada do material a ser medido, é o caso do uso das réguas ou gabaritos, visores de nível e boias. Na medição indireta é feita indiretamente por meio de grandezas físicas, como, Pressão, medição por capacitância, empuxo e ultrassom (BROCKVELD JUNIOR, 2017).

Na medição por pressão utiliza-se dispositivos do tipo pressão diferencial. Nesse sistema é medida a pressão da coluna liquida sobre o sensor de pressão. O sensor de pressão fica imerso ao líquido e a altura da coluna liquida é a razão entre a pressão e a densidade do líquido que está sendo medido.

Na medição por capacitância, os capacitores tipo cilíndricos são normalmente empregados para esse tipo de medição e baseia-se no princípio de que um cilindro imerso no líquido funciona como uma das placas do capacitor e as paredes do recipiente que contém o líquido forma a outra placa e o fluido comporta-se como o dielétrico. Nesse tipo de medição é necessário saber a constante dielétrica do líquido que está sendo medido e o reservatório deve ter paredes condutoras. A mudança de nível altera a capacitância e esse valor é utilizado para calcular a quantidade de material.

Na medição por ultrassom (Figura 1), consiste em disparar uma onda sonora na direção do material com frequência de oscilação maior que o valor perceptível pelo ouvido, ou seja, maior que 20KHz, e contabilizar o tempo em que onda atinge o objeto e retorna para o sensor. Obtendo-se o tempo de propagação da onda e sabendo-se a velocidade do som calcula-se a distância entre o sensor e o líquido a ser medido. Porém pretende-se determinar a altura do líquido. Para isso o sensor é instalado em uma altura conhecida em relação a um ponto de referência, que em um recipiente seria a base inferior. Então faz-se a subtração entre altura do sensor e a distância medida do sensor para o líquido e obtém-se a altura do líquido. (BEGA. et al, 2011).

Figura 1 - Medição de nível por Ultrassom



Fonte: O próprio autor

Para este projeto a forma mais viável e adequada para medição de nível é utilizando ultrassom pois tem a vantagem de não entrar em contato com o líquido. E levando-se em consideração o fato de que em um igarapé o líquido não é homogêneo devido a poluição, podendo ter diferentes densidades e capacitâncias.

1.2 LORAWAN

A tecnologia *Long Range Wide Area Networking (LoRaWAN)* é um protocolo de rede sem fio do padrão *LPWAN* voltado para internet das coisas. Baseia-se na tecnologia de envio de dados com longo alcance e baixo consumo de energia e utiliza a topologia de rede do tipo estrela. Os dispositivos da rede *LoRa* são divididos em três tipos; os *end-device* que são os dispositivos utilizados diretamente na aplicação, ou seja, ficam no final da arquitetura estrela e são usados para ler sinais de sensores e enviar para outro *end-device* ou *gateways*. Os *gateways* que recebem os sinais dos *end-devices*. O *NetworkServe* faz o chaveamento do *gateway* com a rede internet.

O *LoRaWAN* tem três classes diferentes de dispositivos de ponto final para atender às diferentes necessidades refletidas na ampla gama de aplicações:

Dispositivos classe A, são dispositivos bidirecionais de menor potência. Todos os dispositivos *LoRa* suportam a classe A. O *end-device* inicia uma transmissão de *uplink* a qualquer instante em seguida abre duas janelas curtas de tempo para *downlink*, para que seja feito comunicação bidirecional ou comandos de controle de rede. Se não houver comunicação da rede o dispositivo final entra em modo de suspensão por um tempo programado. A Classe A é o modo de operação com menor consumo de energia (LORA ALIANCE, 2019).

Os dispositivos classe B, são dispositivos finais bidirecionais com latência de *downlink* determinística. Os dispositivos classe B são sincronizados com o *gateway* usando *beacons* periódicos que abrem janelas para *downlink* em horários programados. O consumo de energia na classe B ainda é baixo suficiente para aplicações com uso de bateiras.

Os dispositivos classe C, são dispositivos bidirecionais de menor latência. Trabalham com a estrutura de *uplink* de classe A seguida por duas janelas de *downlink*, porém a classe C reduz ainda mais a latência no *downlink*, mantendo o receptor do dispositivo final ativo em todos os momentos mesmo quando o dispositivo não está transmitindo. Na classe C o servidor pode iniciar uma transmissão em qualquer instante, pois o *end-device* deve estar em modo ativo para receber a mensagem de *downlink*. Com esta configuração perde-se autonomia, não sendo indicado o uso de baterias. Em aplicações com uso de baterias é possível alternar entre classe A e C de modo temporário, por exemplo para atualização de *firmware* por meio da rede (LORA ALIANCE, 2019).

1.3 PARÂMETROS DE CONFIGURAÇÃO LORA

Um dispositivo LoRa pode ser configurado com alguns parâmetros que podem alterar o comportamento da transmissão e recepção e o consumo de energia. Alguns parâmetros importantes serão listados a seguir:

Transmission Power (TP): A potência de transmissão do rádio lora pode ser ajustada desde -4 dBm e 20 dBm, em passos de 1 dB, mas por causa dos limites de implementação de hardware, a gama é frequentemente limitada a 2 dBm e 20 dBm. Além disso, por causa de limitações de hardware, os níveis de potência mais elevada do que 17 dBm só pode ser utilizado em um ciclo de trabalho de 1% (BOR; ROEDIG, 2017).

Carrier Frequency (CF): Frequência da portadora é a frequência central que pode ser programada em etapas de 61 Hz entre 137MHz e 1020 MHz. Dependendo do chip LoRa específico, esse intervalo pode ser limitado de 860MHz a 1020 MHz (BOR; ROEDIG, 2017).

Spreading Factor (SF): O princípio básico do Spreading Factor (Fator de Espalhamento) é que cada bit de informação é codificado por um pulso modulado em frequência chamado de *chirps*. Esta modulação faz com que a frequência mude ligeiramente para mais (*Up-Chirp)* ou para menos (*Down Chirp*) dentro da largura de banda fornecida. Com o aumento de uma unidade do SF a taxa de transmissão se reduz pela metade, por conseguinte, a duração da transmissão dobra, e consequentemente, há um aumento do consumo de energia. O fator de espalhamento pode ser selecionado a partir de 6 a 12 (BOR; ROEDIG, 2017). É necessário que o transmissor e receptor estejam parametrizados com o mesmo fator de espalhamento. No entanto é possível a comunicação com vários nodes utilizando uma rede de separação com SF diferentes (BOR; ROEDIG, 2017).

Bandwidth (BW): BW ou Largura de Banda, é a largura das frequências na banda de transmissão. O BW mais alto fornece uma taxa de dados mais alta (portanto menor tempo no ar), mas uma sensibilidade mais baixa (devido à integração de ruído adicional). Um BW mais baixo fornece uma sensibilidade mais alta, mas uma taxa de dados menor. BW inferior também requer cristais mais precisos (menos ppm). Os dados são enviados a uma taxa de chip igual à largura de banda; uma largura de banda de 125 kHz corresponde a uma taxa de chip de 125 kcps. Embora a largura de banda possa ser selecionada em um intervalo de 7,8 kHz a 500 kHz, uma típica rede LoRa opera em 500 kHz, 250 kHz ou 125 kHz (BOR; ROEDIG, 2017).

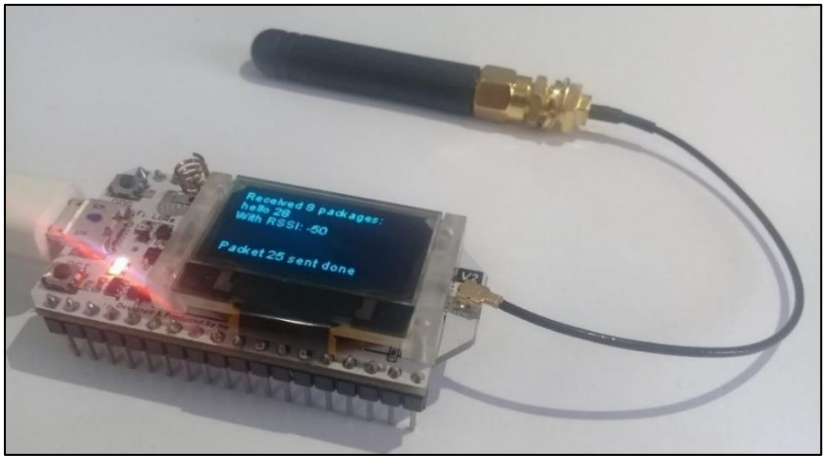
1.4 MÓDULO WIFI LORA 32(V2)

O módulo *Wifi LoRa 32(V2)* fabricado pela *heltec* (Figura 2) é um sistema embarcado que possui integrado o chip microcontrolador Esp32-D0WDQ6 que apresenta alto nível de desempenho e baixo consumo de energia. Possui modo de economia de energia e regulação dinâmica de tensão (HELTEC AUTOMATION, 2019).

Este módulo possui *Wifi*, *bluetooth*, e conexão com o chip *LoRa* SX125X, para comunicação por meio de ondas de rádio utilizando o protocolo *LoRaWAN*. O sistema possui baixo consumo de energia principalmente quando está operando em classe A, pois estará ativo somente no momento da transmissão, ou seja, durante o tempo da medição do sensor e envio

da mensagem. Após o envio o sistema entra em suspensão e fica hibernando por tempo programado até o instante de outro envio de mensagem. (HELTEC AUTOMATION, 2019).

Figura 2 - Sistema embarcado Wifi LoRa 32(V2)

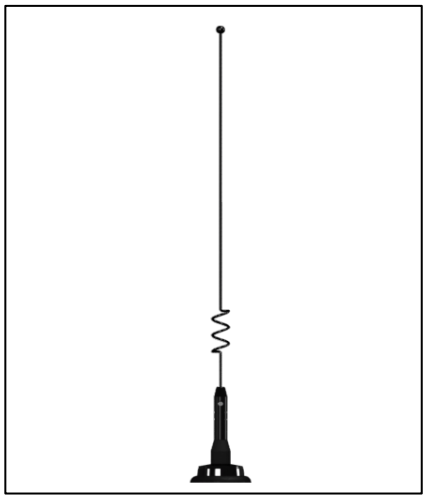


Fonte: O próprio autor

1.5 ANTENA AP3900

Para melhor qualidade de sinal é importante que uma antena com melhor ganho seja empregada na transmissão e recepção. A antena modelo AP3900 mostrada na Figura 3 foi utilizada e apresentou bom desempenho.

Figura 3 - Antena AP3900

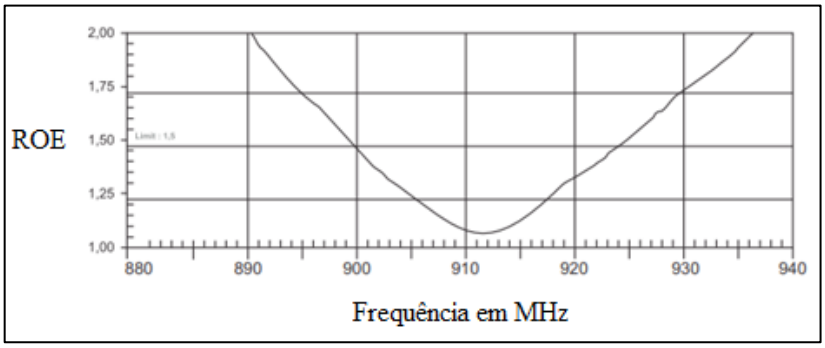


Fonte: STEELBRAS ANTENAS (sd, p.2)

No gráfico da Figura 4 é ilustrado o comportamento de perdas de retorno da antena em função da frequência. Essas perdas conhecidas como ROE (Relação de ondas estacionárias) ou

seu termo em inglês *SWR (Standing Wave Ratio)*, representa parte da energia que retorna da antena para o transmissor quando não se tem um casamento de impedância adequado. O valor de ROE = 1 é o ideal que significa que a antena e o transmissor estão casados, ou seja, com mesma impedância.

Figura 4 - Gráfico de ganho da antena AP3900



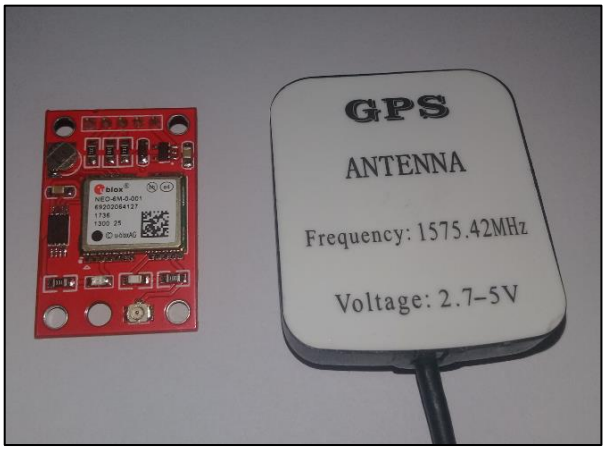
Fonte: STEELBRAS ANTENAS (sd, p.2)

A antena possui as seguintes características importantes: Frequência na faixa de 900MHz, impedância de 50Ω, Ganho de 5,15 dBi e 320mm de altura (STEELBRAS ANTENAS, sd).

1.6 MODULO GPS NEO-6M

A placa GY-NEO6MV2 mostrada na Figura 5, possui o módulo GPS u-blox NEO-6M com antena e *EEPROM* embutida. Isso é compatível com várias placas controladoras de vôo projetadas para funcionar com um módulo GPS (SYNACORP, sd).

Figura 5 - Módulo gps NEO-6MV2



Fonte: O próprio autor

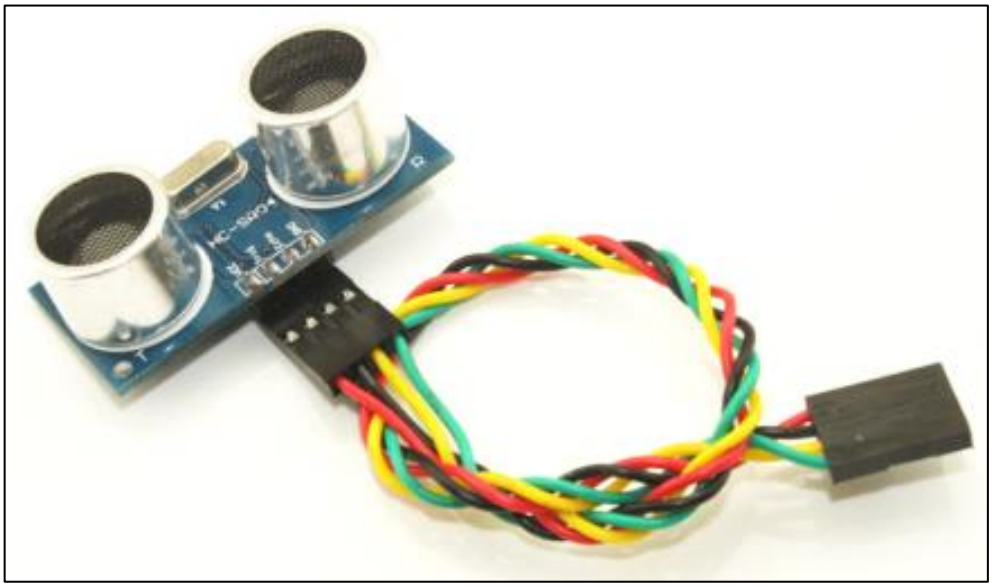
Abaixo estão listadas algumas características do módulo GPS NEO-6M:

a) Tensão de alimentação: 3 V a 5 V;
b) Nível TTL, compatível com sistema 3.3V / 5V;
c) A taxa de transmissão padrão: 9600 bps;
d) LED indicador de sinal;
e) Bateria de *backup* recarregável, pode salvar os dados quando desligar (SYNACORP).

1.7 SENSOR ULTRASSÔNICO HC - SR04

O sensor de ultrassom HC- SR04 possui quatro pinos de conexão (Figura 6). Dois de alimentação, 5V e GND, o pino *Trigger* para dar o início da medição de distância e o pino *Echo* que retorna o tempo por meio de transições para nível alto e para nível baixo. O pino *Echo* permanece em alto até que o sensor detecta o retorno do sinal sonoro. Após receber um pulso de 10 microssegundos no *Trigger*, o sensor emite uma onda de 40KHz que reflete em obstáculos e retorna para o sensor (ELECFREAKS, 2015).

Figura 6 - Sensor de ultrassom HC-SR04

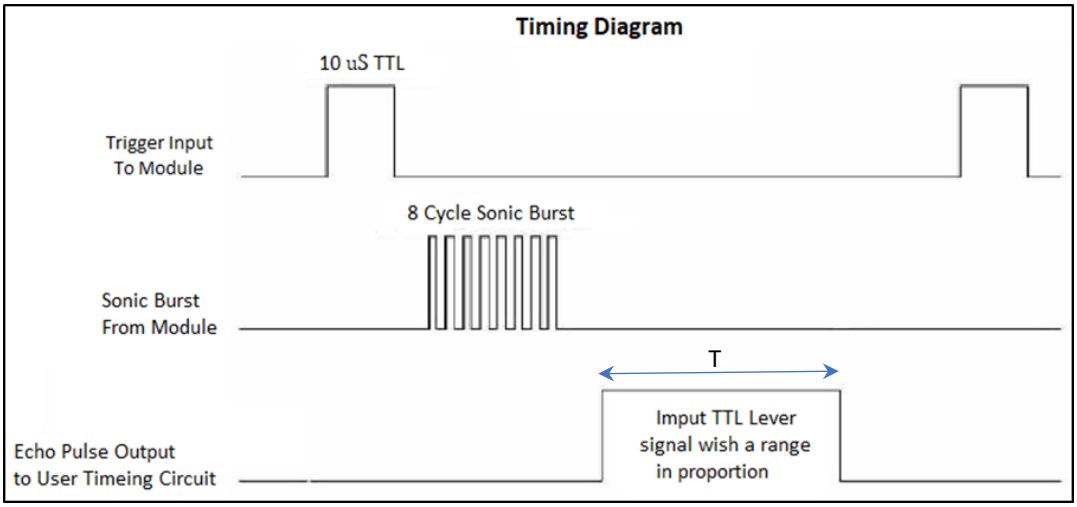


Fonte: ELECTREAKS (2015, p. 3).

Quando o sensor emite o sinal de ultrassom o pino *Echo* do sensor vai a nível alto e permanece em alto até que o sensor capture o retorno da onda sonora. Utilizando-se a informação de tempo em que o pino *echo* fica em nível alto é possível calcular a distância do obstáculo (ELECFREAKS, 2015).

O diagrama de tempos que mostra o comportamento dos pinos *Trig* e *Echo* pode ser observado na Figura 7.

Figura 7- Diagrama de tempos do sensor HC-SR04



Fonte: ELECTREAKS (2015, p. 2).

1.8 SOFTWARES

1.8.1 Arduino IDE

O ambiente de programação do Arduino utiliza a linguagem C++ que é compilador GCC e que inicialmente foi desenvolvida para o Arduino uno que utiliza o microcontrolador Atmega328p. Porém muitos outros fabricantes de sistemas embarcados criaram e disponibilizaram bibliotecas para desenvolvimento em suas plataformas. Por exemplo, o stm32F103X, o esp32, *NodeMCU*, arduino mega etc. As bibliotecas são um conjunto de códigos que se inclui no programa para se aprimorar o projeto. Assim pode-se importar uma biblioteca desenvolvida por outra pessoa ou fabricante de dispositivos. No menu Sketch em incluir bibliotecas, pode-se baixar uma biblioteca nova ou apontar para uma pasta do computador onde já encontra os códigos que serão inseridos (BLUM, 2016)

Para o módulo *wifi LoRa 32(v2)* utilizado neste projeto a fabricante *heltec* disponibiliza as bibliotecas para a IDE arduino, incluído a biblioteca do display *Oled* e as rotinas de comunicação do microcontrolador Esp32 com o chip SX125X *LoRa*.

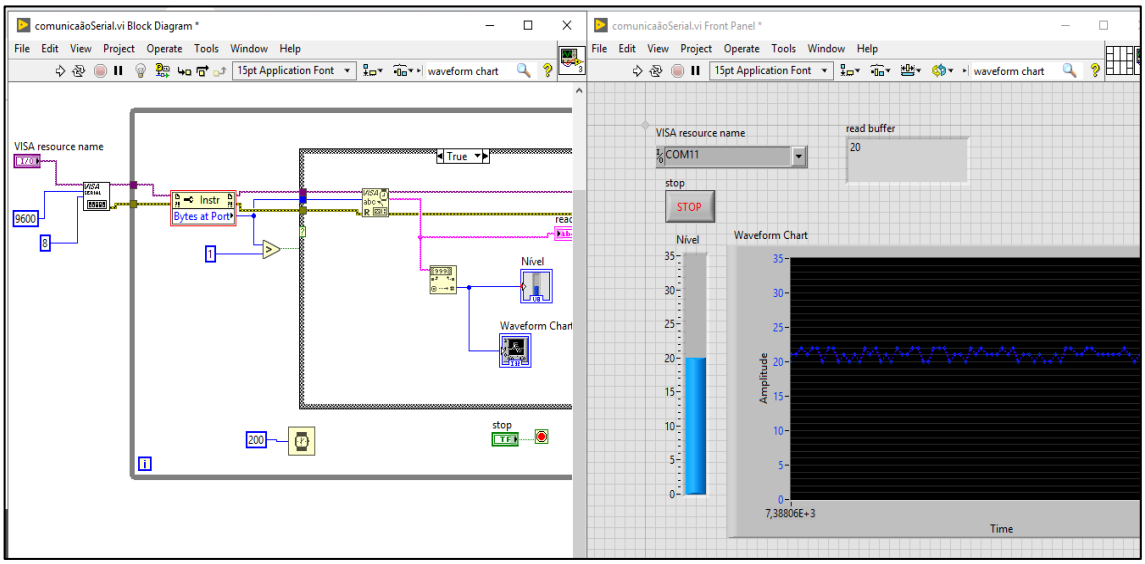
1.8.2 LabVIEW

O Software, *Laboratory Virtual Instruments Engineering Workbench (LabVIEW)*, é um ambiente de programação que utiliza linguagem gráfica (G), que é uma ferramenta que possibilita alta produtividade no desenvolvimento de sistemas de aquisição de dados, instrumentação e controle.

O *LabVIEW* é um ambiente que tem como padrão a programação em blocos que podem ser conectados formando um sistema maior, dando ao programador uma visão modular e mais organizada do sistema. Além de ser uma excelente ferramenta para criação de interface gráficas possuindo indicadores numéricos, gráficos de ondas, gráficos com cursores etc (REGAZZI; PEREIRA; SILVA JR, 2005).

Na Figura 8 pode-se observar um exemplo de interface desenvolvida no LabVIEW.

Figura 8 - Exemplo de Interface no LabVIEW



Fonte: O próprio autor.

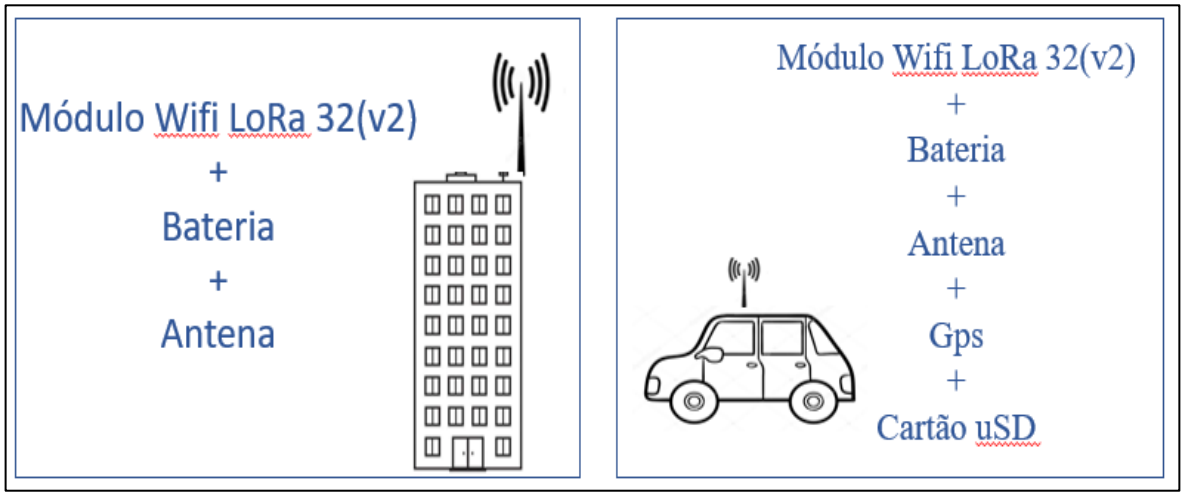
2 MÉTODOS E MATERIAIS

Neste capítulo é mostrado como se deu o andamento do projeto desde a pesquisa e instalação da biblioteca *Wifi LoRa 32(V2)* até a comunicação com o *labview* e a criação da interface de monitoramento. O sistema consiste em monitorar os níveis de um reservatório por meio de sensor ultrassônico que vai simular uma unidade remota, em um igarapé. Em seguida vai enviar esse nível por meio do rádio LoRa para uma unidade de monitoramento que apresentará os dados em uma interface gráfica em um computador. O sistema foi desenvolvido nas seguintes etapas:

Na primeira etapa foram realizadas pesquisas na área de programação, dando ênfase na programação C++, para melhor entendimento e uso das bibliotecas do dispositivo *Wifi LoRa* 32(V2) fornecidas pelo fabricante. Foi acessado o site do fabricante para download da biblioteca e instalação na IDE arduino.

Na segunda etapa foram feitos testes de envio de dados para se observar o comportamento de atenuação, perdas de pacotes e o alcance. Para realizar esses testes foi necessário a utilização de um modulo gps e um gravador de cartão micros para registrar o trajeto e a distância. No desenvolvimento desta tarefa uma unidade estática contendo um modulo *Wifi LoRa 32(v2)* ficou posicionada em local estratégico e emitindo sinal. Já o *receptor Wifi LoRa 32(v2*) com bateria e antena foi instalado em um veículo (Figura 9). Com o veículo em movimento, uma vez que uma mensagem nova chega ao receptor, o processador grava a mensagem e as coordenadas da localização no cartão micro SD.

Figura 9 - Segunda etapa, teste de distância.



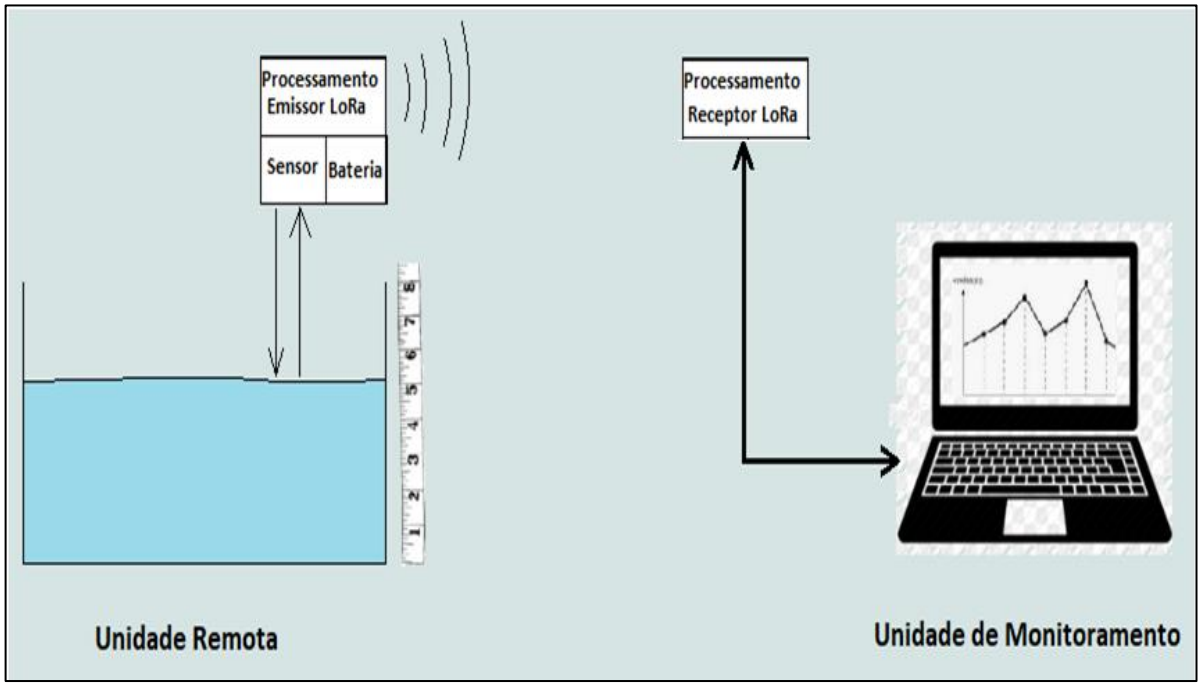
Fonte: O próprio autor

Na terceira etapa fez-se a confecção de um protótipo para simulação das cheias. Este protótipo contém um reservatório, uma placa alimentada por bateria e conectada a um sensor ultrassônico e a uma antena. Uma estrutura mecânica de alumínio foi utilizada para fixar os dispositivos. Na placa de medição de nível foi desenvolvido um sistema transistorizado para alimentar o sensor. Esta placa será mais detalhada na seção 3.2.4. Nesta placa também foram realizadas as medições de consumo uma vez que ficará na unidade remota e deve ter grande autonomia.

Na quarta etapa realizou-se a comunicação do dispositivo *Wifi LoRa 32(V2)* com um computador por meio de comunicação serial. Para exibir os dados foi desenvolvida uma interface gráfica com softwares dedicados para a supervisão de sistemas. Foi utilizado o *LabVIEW* para o desenvolvimento desta etapa.

Ao finalizar as etapas 3 e 4, obteve-se um sistema conforme o esquema ilustrado na Figura 10. Onde se pretende variar os níveis no recipiente, visualizar essa variação no computador, e presentar alarmes quando ultrapassar determinados valores.

Figura 10 - Esquema da unidade Remota e unidade de monitoramento



Fonte: O próprio autor

2.1 MATERIAIS UTILIZADOS NO TESTE DE DISTÂNCIA

No equipamento estático, fez-se uma montagem do transmissor com os seguintes itens.

1 modulo *Wifi LoRa 32 (V2)*

1 protoboard para encaixe do modulo.

1 bateria

Durante a montagem da placa de recepção de sinal e registro da localização, foi utilizado os materiais listados abaixo.

1 modulo de sistema embarcado *Wifi LoRa 32 (v2)*

1 modulo gps NEO-6M-0-001

1 modulo para cartão micro SD.

2 leds

2 resistores de 1k Ω

2 barras de bornes de 40 encaixes.

2 barras de 40 pinos.

1 Bateria.

1 placa de fenolite universal 9x7cm

Ferro de solda

Solda (estanho).

2.2 MATERIAIS DA PLACA DE MEDIÇÃO DE NÍVEL

Durante a criação do protótipo criou-se outra placa para realização das medidas de nível. Nesta placa fez-se a conexão do sistema *Wifi LoRa 32 (v2)* com o sensor ultrassônico. E também foi desenvolvido um circuito para alimentação do sensor. O sistema contém os seguintes materiais.

1 placa de fenolite universal 5x7 cm

1 módulo *Wifi LoRa 32 (V2)*

2 barras de bornes para encaixe do modulo *Wifi LoRa 32 (V2).*

1 transistor PNP BC556

1 transistor NPN BC 337

2 conectores bornes KRE, 2 vias.

1 suporte metálico para fixar o conector SMA onde será conectada a antena.

2 resistores de 10kΩ, 1 resistor de 5k1Ω e 1 resistor de 33k2Ω

3 IMPLEMENTAÇÃO DO PROJETO

Este capítulo apresenta a realização do projeto em questão. Foi desenvolvida a programação no ambiente arduino IDE, uma placa para testes de alcance, atenuação. Também mostra-se a construção do protótipo para medição de nível e o desenvolvimento do sistema de supervisão.

São apresentados neste capítulo os seguintes tópicos:

a) Instalação das bibliotecas na arduino IDE;

b) Teste de Distância;

c) Desenvolvimento do protótipo da unidade remota;

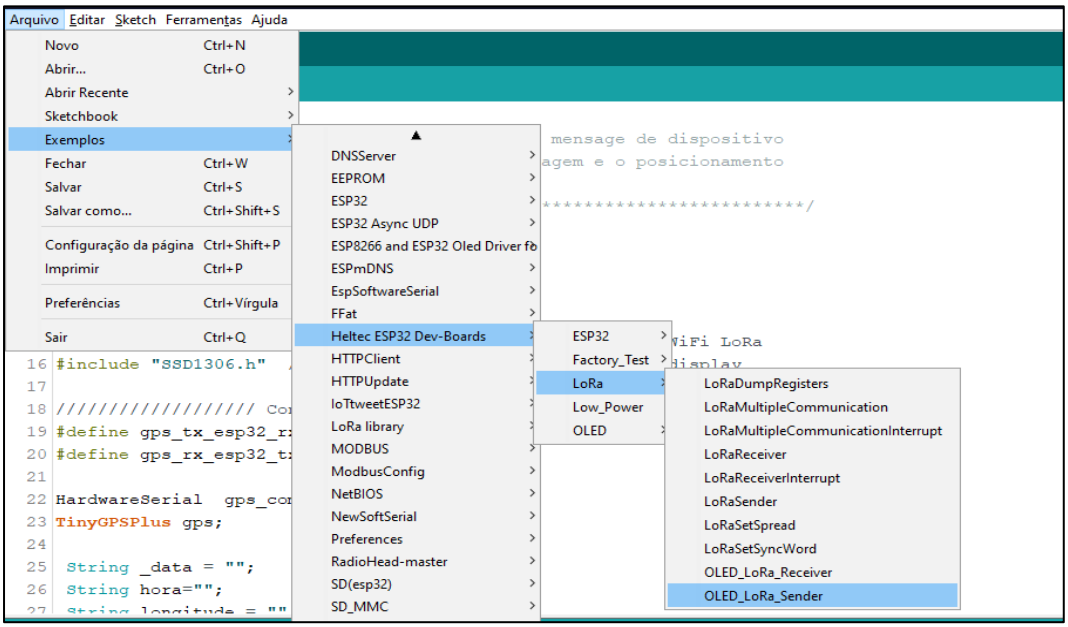
d) Desenvolvimento do sistema de supervisão.

3.1 INSTALAÇÃO DAS BIBLIOTECAS NA ARDUINO IDE

Para o desenvolvimento deste trabalho foram instaladas algumas bibliotecas especificas que serão usadas na programação do módulo Wifi LoRa 32(v2).

A primeiro foi necessário baixar os arquivos para a instalação da placa Wifi LoRa 32 da seguinte forma. Na arduino IDE, arquivo/preferencias e URLS adicionais para gerenciador de placa, foi adicionado o seguinte endereço *https://docs.heltec.cn/download/package_heltec_esp32_index.json.* Em seguida em placa/gerenciador de placas, foi digitado '*heltec'*, após a atualização foi escolhido a opção '*heltec ESP32 dev-board by heltec automation'* e clicado em instalar. Após a instalação dos drives da placa, foram instaladas as bibliotecas e os exemplos. Em Sketch/Incluir bibliotecas/Gerenciador de bibliotecas, foi digitado '*Heltec ESP32'* e clicado em instalar. Em seguida no mesmo gerenciador de bibliotecas, digitou-se *LoRa*, para encontrar a biblioteca '*LoRa by sandeep mistry',* que dá suporte ao chip Sx1276. Em seguida fez a instalação da biblioteca do display oled. No gerenciador de bibliotecas foi escolhida a opção '*ESP8266 and ESP32 Oled Drivers for SSD1306 by Daniel Eichhron, Fabrice Weinberg'*. Após a instalação é possível compilar um exemplo conforme visto na Figura 11.

Figura 11 - Biblioteca Heltec LoRa na Arduino IDE



Fonte: O próprio autor

Também foi necessária a biblioteca *ultrassonic.h.* Ela pode ser baixada em um arquivo zip no site https://portal.vidadesilicio.com.br/hc-sr04-sensor-ultrassonico/. Ao baixar a pasta com arquivos compactados, abriu-se a IDE arduino, no menu Sketch/incluir biblioteca, foi selecionado incluir biblioteca zip e depois direcionado para pasta onde se encontra o arquivo zip. Esta biblioteca já faz o cálculo da distância. E foi utilizada da seguinte forma na arduino IDE. Usa-se a diretiva #include<*ultrassonic.h*> para incluir os métodos da biblioteca. No comando *Ultrasonic ultrasonic(trig,echo)*, definiu-se os pinos *trig e echo*. No caso desta aplicação foram o pino 12 e 13. E a função *ultrasonic.Ranging(CM)*, retorna um número do tipo *long* em centímetros. Se a função for usada sem o argumento, *ultrasonic.Ranging( )*, ela retorna um valor em polegadas.

Foi baixado a biblioteca *tinygpsplus*, no link http://arduiniana.org/libraries/tinygpsplus/ e foi instalada da mesma forma que a biblioteca *ultrassonic*. A inclusão desta biblioteca no programa do teste de distância fez-se utilizando o seguinte comando, #include<TinyGPS++.h>.

Abaixo estão alguns comandos importantes da biblioteca *tinypgsplus:*

O comando *TinyGPSPlus gps*, define um objeto chamado gps do tipo TinyGPSPlus.

O comando *gps.encode(gps_com.read())*, ler a porta serial conectada no dispositivo gps NEO-6M-0-001

O comando, *latitude = String(gps.location.lat(),6),* devolve o valor coordenada Latitude do tipo *float* com seis casas decimais, em seguida é convertido em uma *String* e gravada na variável latitude.

O comando *longitude = String(gps.location.lng(),6),* devolve o valor coordenada Longitude do tipo float com seis casas decimais, e em seguida é convertido em uma *String* e gravada na variável longitude.

3.2 TESTE DE DISTÂNCIA

Para fazer testes de envio de mensagem e mensurar a distância foi preciso de um equipamento estático e um móvel. O equipamento estático, constitui um módulo *Wifi LoRa 32(v2),* conectado a um protoboard, uma bateria para alimentação do módulo e um adaptador *pigtail* para SMA fêmea. O lado *pigtail* é conectado na placa e o conector SMA foi fixado em uma parte metálica, onde recebe o conector SMA macho da antena.

Figura 12 - Instalação do equipamento no veículo

(a)Receptor no interior do veículo (b) Antena no teto no veículo

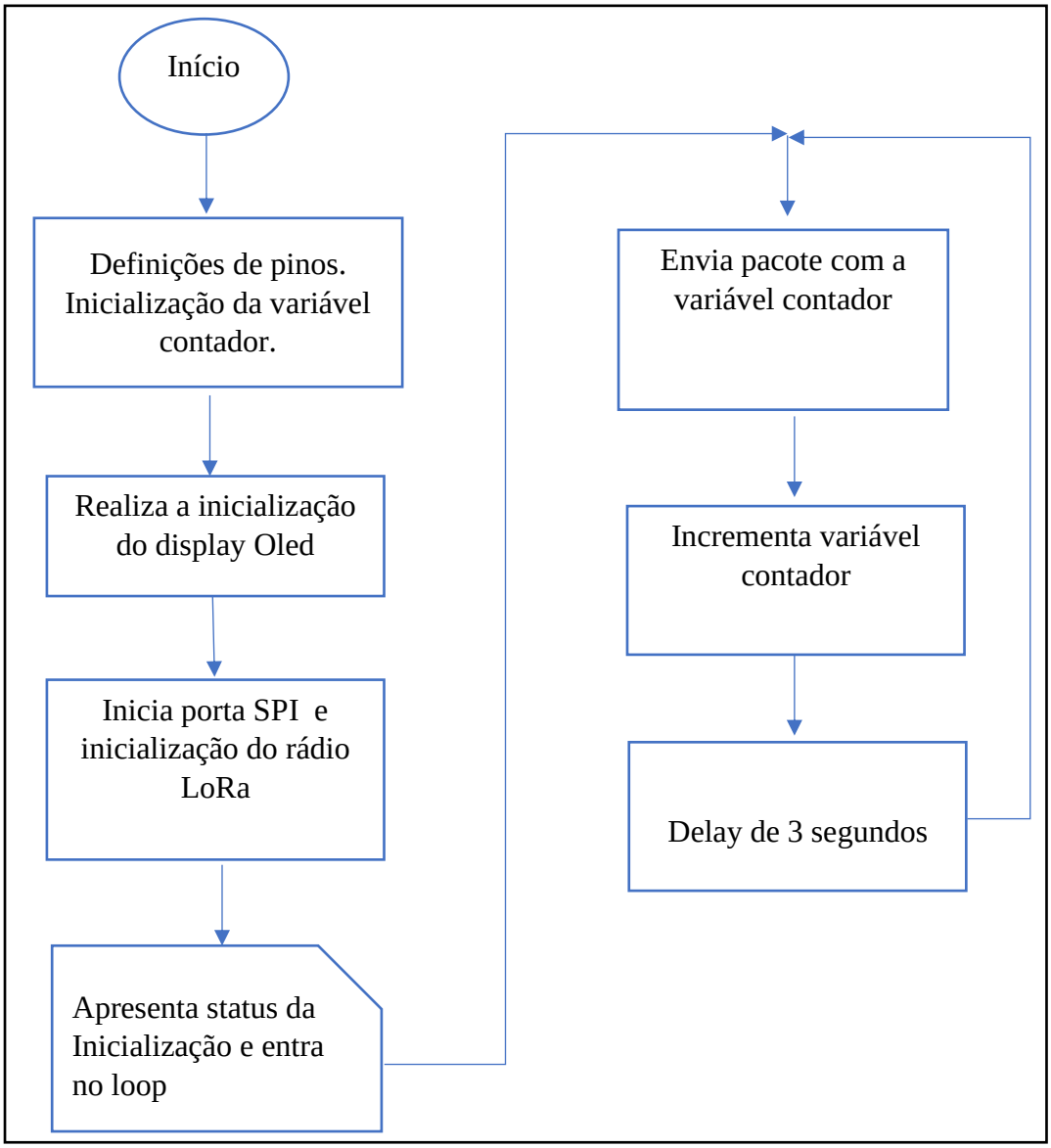
Fonte: O próprio autor

O equipamento móvel (Figura 12) contém um módulo gps, um módulo para gravação do cartão micro SD, um módulo *Wifi LoRa 32(v2).* A antena utilizada foi o modelo AP3900, que possui base magnética e foi instalada no teto do veículo. E a placa de recepção no interior do veículo.

3.2.1 Fluxograma do transmissor do teste de distância

A unidade transmissora ficou estática e realizando uma tarefa mostrada no passo a passo da Figura 13. Envia um número, incrementa um contador, esperar um tempo de três segundo e volta a enviar o dado repetindo o ciclo. O programa deste algoritmo pode ser visto no apêndice A.

Figura 13 - Algoritmo para unidade transmissora do teste de distância

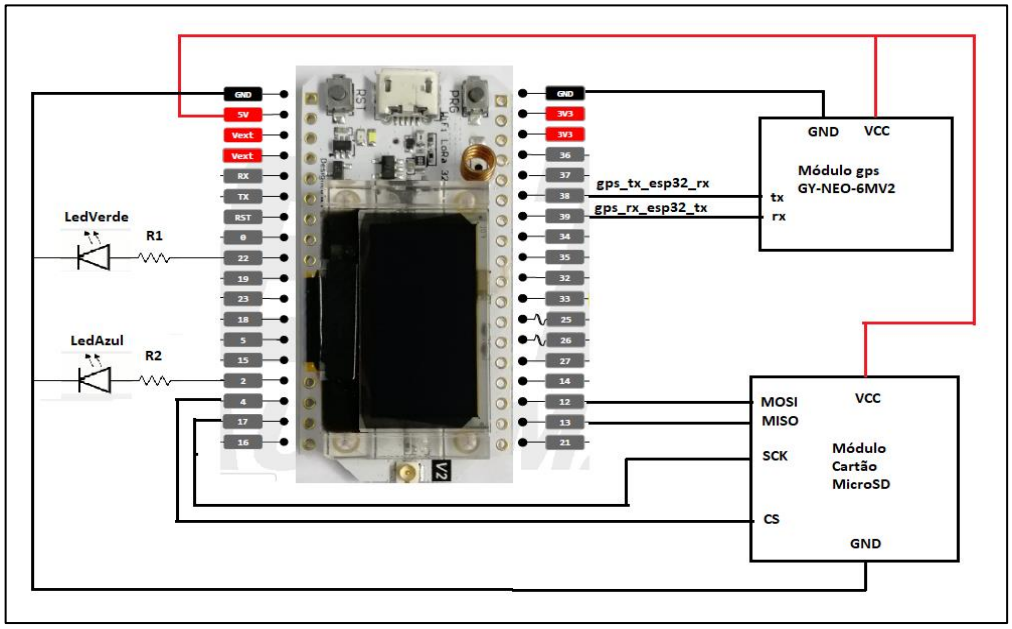


Fonte: O próprio autor

3.2.2 Desenvolvimento da placa de recepção e gravação de posicionamento.

O sistema foi montado com a seguinte configuração, mostrada na Figura 14. Para R1 e R2 foram utilizados resistores de 1KΩ. A tensão admitida para os leds de alto brilho foi V=2,4V, que alimentados pela tensão dos GPIOs, Vout = 3,3V nos dá uma corrente de 9mA. Antes da montagem foram feitos testes em protoboard com resultados satisfatórios.

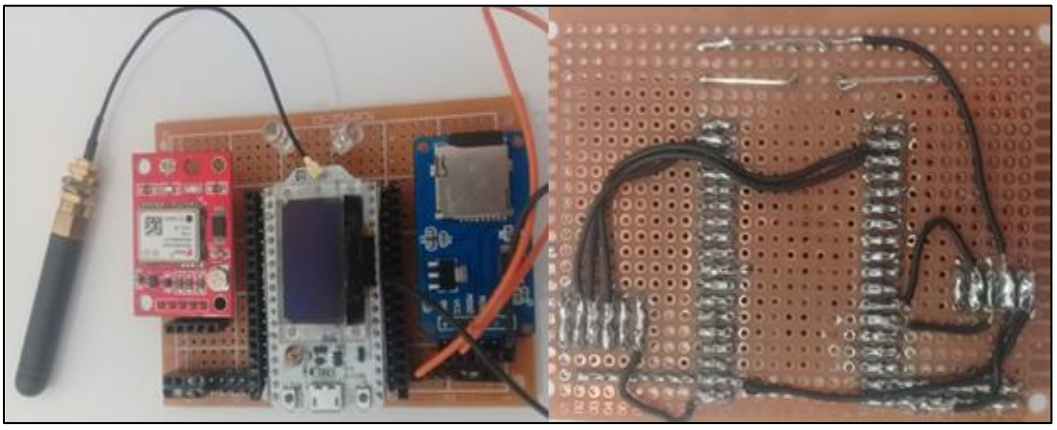
Figura 14 - Sistema para registro da distância



Fonte: O próprio autor

O algoritmo da seção 3.2.3 foi desenvolvido para ser executado no sistema da Figura 15, que contém o modulo *Wifi LoRa 32(v2)*, modulo para cartão *micro SD* e modulo gps.

Figura 15 - Sistema para recepção e gravação da localização

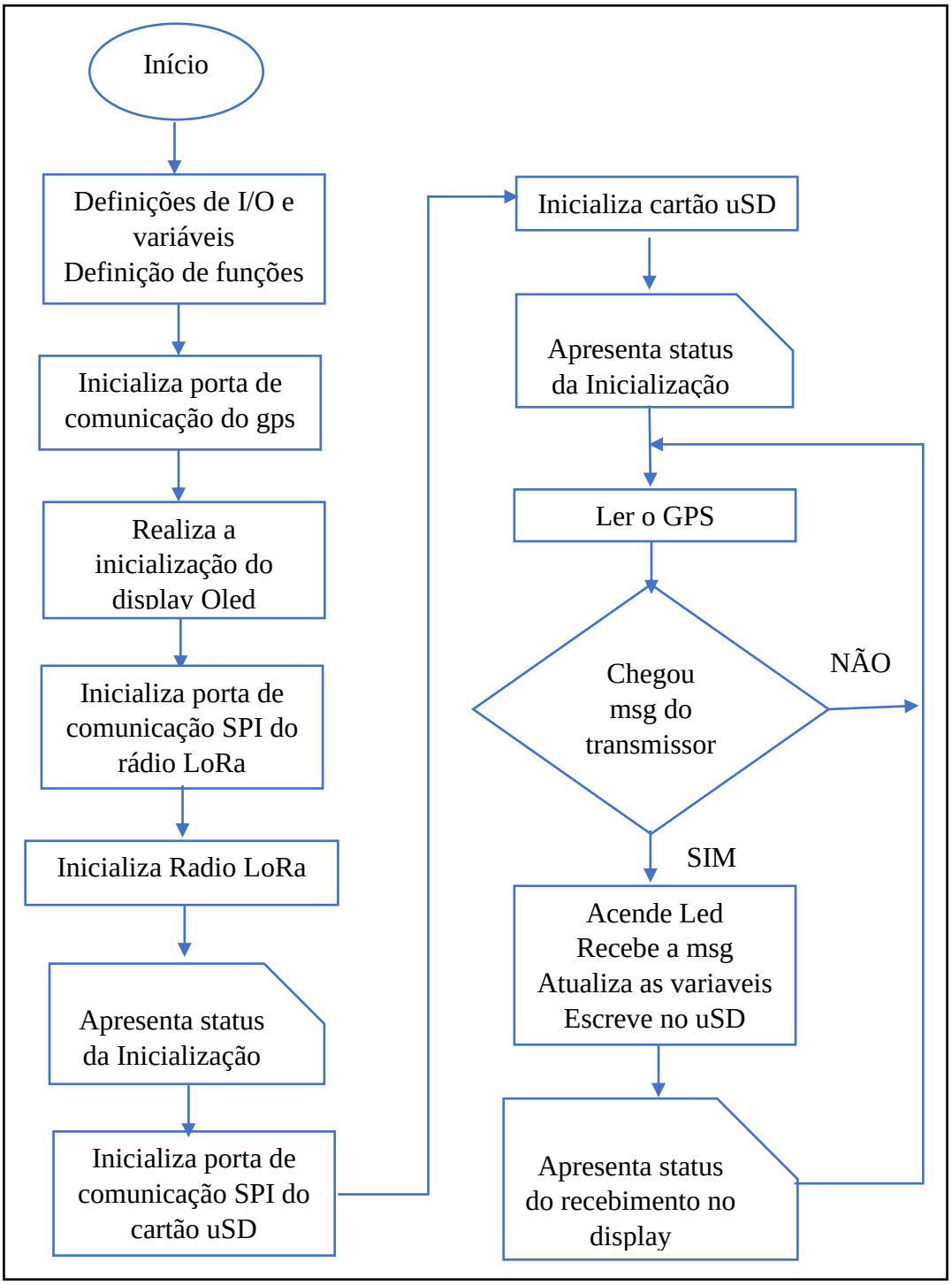


Fonte: O próprio autor.

3.2.3 Fluxograma para recepção e registro do posicionamento.

O algoritmo do sistema de recepção para o teste de distância dar-se conforme a Figura 16.

Figura 16 - Algoritmo para recepção do sinal e registro de posição

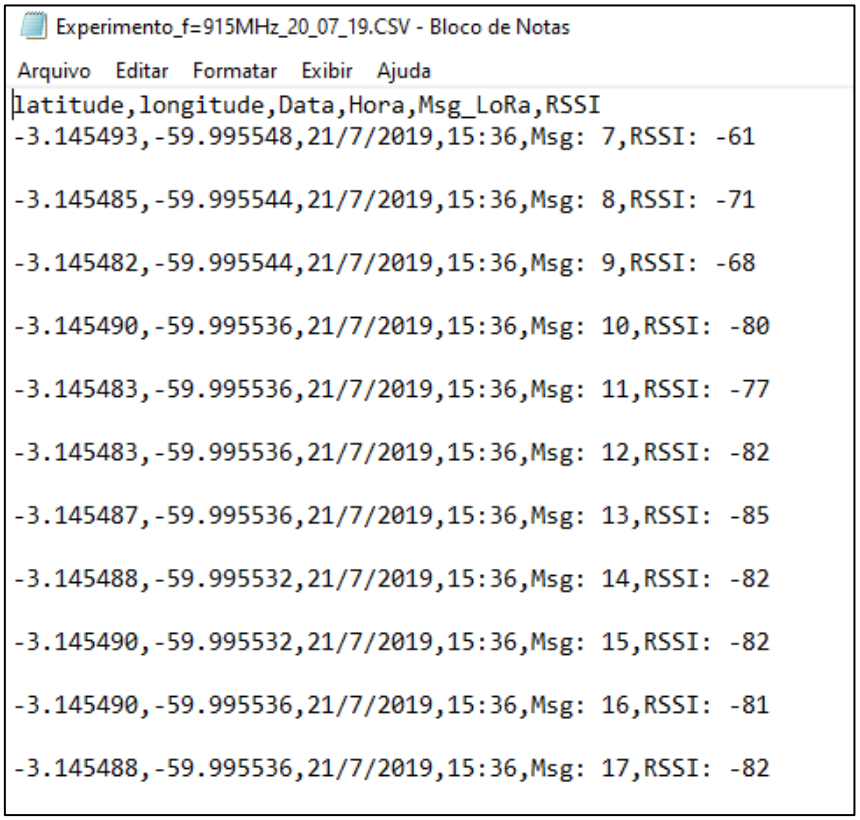


Fonte: O próprio autor

A explicação do algoritmo dar-se da seguinte forma. Quando a placa é ligada, o controlador ESP32 realiza as definições de variáveis e as configurações do módulo gps, do chip *LoRa SX1276* e do cartão micro SD. Em seguida o programa entra em um laço infinito onde fica fazendo a leitura do gps e verificando se chegou alguma mensagem do transmissor. Se nenhuma mensagem chegar ele volta a ler o gps. Quando o sistema recebe uma mensagem, ele atualiza as variáveis lidas do gps que são a latitude, longitude, data e hora, e outras duas variáveis *msg_LoRa* e *rssi*, que são a mensagem enviada do transmissor e a atenuação. Em seguida, o sistema escreve esses dados em um arquivo com extensão CSV no cartão micro SD. Assim se obtém a posição onde se recebeu a mensagem. O programa que executa estas tarefas pode ser visto no apêndice B.

O arquivo registrado no cartão micro SD tem as características mostradas na Figura 17.

Figura 17 - Arquivo com os dados recebidos



Fonte: O próprio autor

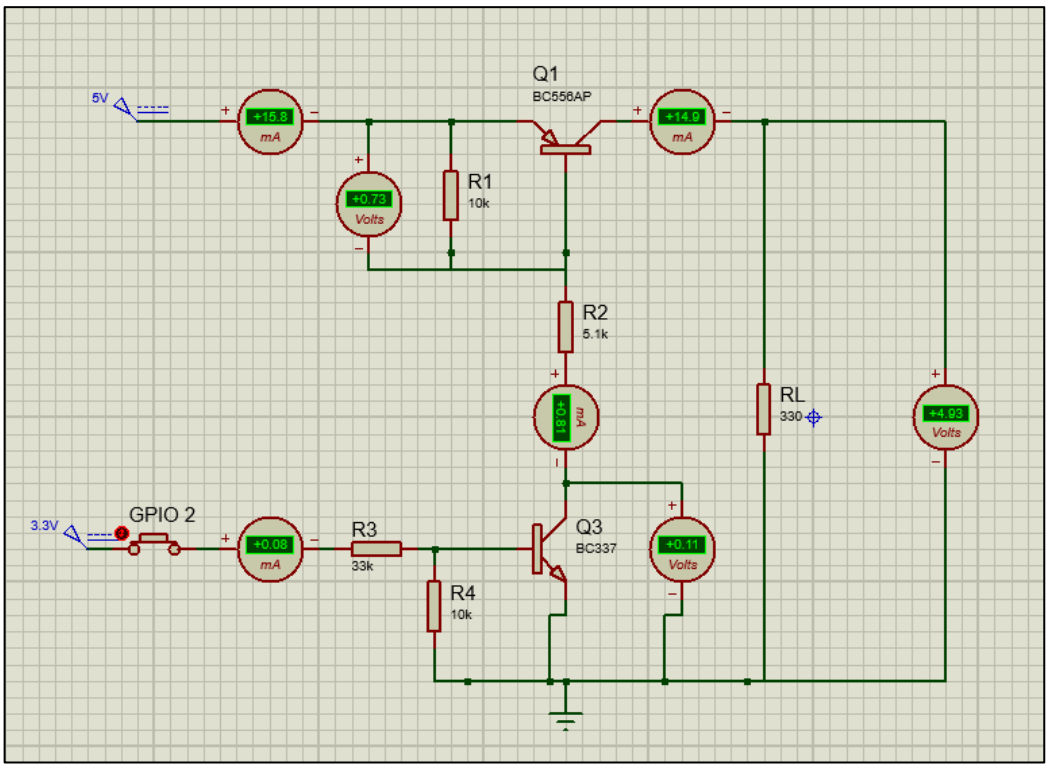
3.3 DESENVOLVIMENTO DO PROTOTIPO DA UNIDADE REMOTA

Nesta etapa mostra-se como foi construída a placa para medição do nível e a estrutura do protótipo onde foram fixados a placa, antena, sensor e a bateria.

3.3.1 Placa para medição do nível

Para fazer a medição do nível foi desenvolvida uma placa para fazer a alimentação e a leitura do sensor de ultrassom. O circuito de alimentaçao do sensor foi simulado no software Protheus 8, e pode ser visto na Figura 18.

Figura 18 - Circuito de alimentação do sensor de ultrassom



Fonte: O próprio autor

Foram usados resistores e um transistor Q1 pnp e outro Q2 npn para fazer o circuito de alimentação do sensor. Uma vez que o sensor consome 15mA quando está ligado na fonte. Então foi desenvolvido um circuito para alimentar o sensor somente quando se precisa fazer uma leitura. O botao está simulando o GPIO2 do esp32 e RL a resistência do sensor. Quando o botâo é pressionado polariza e faz conduzir Q2 e o coletor de Q2 está conectado na base do transistor Q1 que o faz conduzir também e alimentar a carga RL. O circuito também foi testado em protoboard antes de ser soldado na placa.

Para conectar o pino *Trig* do sensor foi utilizado o GPIO12 e para a saida *Echo* o GPIO13.

Os cálculos para se obter os resistores se deu conforme abaixo.

Os resistores R1 e R4 foram previamente definidos com o valor de 10KΩ. E o sistema foi dimensionado para poder fornecer uma corrente para carga de até 60mA.

Para os transistores Q1 e Q2 tem-se o ganho β1= β2=100. A queda de tensão de saturação entre coletor e emissor do transistor Q2 VCEsat = 0,7Vdc. Desta forma segue o cálculo sendo I1 a corrente que passa por R1 e I4 a corrente que passa por R4.

$$I1 = I4 = \frac{0.7V}{R4} = \frac{0.7V}{10K\Omega} = 0{,}07\ mA$$

A corrente na base de Q1 fica:

$$I_{b1} = \frac{60\ mA}{100} = 0{,}6\ mA$$

A corrente que passa por R2 é a soma das correntes anteriores.

$$I2 = I1 + I_{b1} \Rightarrow I2 = 0{,}67\ mA$$

Como VCEsat = 0,7Vdc, o valor de R2 é calculado da seguinte forma:

$$R2 = \frac{(5V - V_{be1} - V_{CEsat})}{I2} = \frac{(5V - 0{,}7V - 0{,}7V)}{0{,}67\ mA} = 5{,}37\ K\Omega$$

A corrente na base de Q2 fica:

$$I_{b2} = \frac{I2}{100} = \frac{0{,}67\ mA}{100} = 0{,}0067\ mA$$

Como I4 = I1 = 0,07 mA, a corrente I3 que passa por R3 se dar pela seguinte soma:

$$I3 = 0{,}0067\ mA + 0{,}07\ mA \Rightarrow I3 = 0{,}0787\ mA$$

Então o valor de R3 é calculado da seguinte forma:

$$R3 = \frac{(3{,}3V - 0{,}7V)}{0{,}0767x10^{-3}} = 33{,}89\ K\Omega$$

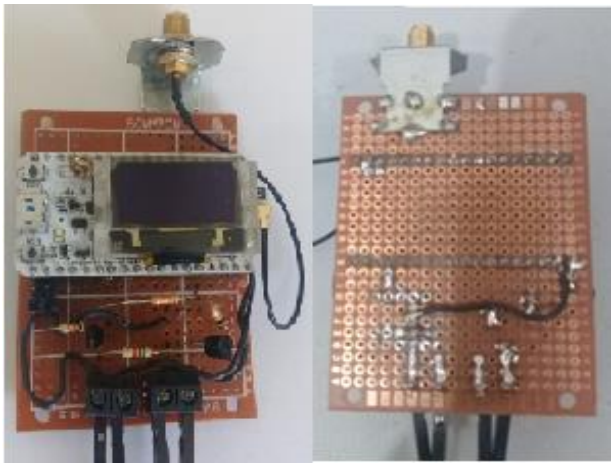
Os valores comercias mais próximos encontrados e utilizados foram:

$$R2 = 5{,}1K\Omega$$

$$R3 = 33K\Omega$$

A placa da unidade remota montada pode ser observada na Figura 19.

Figura 19 - Placa de leitura e transmissão do Nível

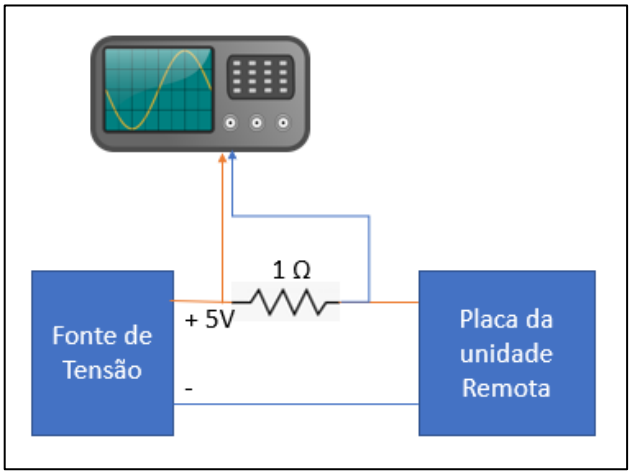


Fonte: O próprio Autor

3.3.2 Consumo de energia da placa de medição de Nível

Pra medir o consumo de energia da placa da Figura 19, foi conectado um resistor Shunt de 1Ω em série com o módulo *WiFi LoRa 32(v2)* e utilizado um osciloscópio para medir o comportamento da tensão nos terminais do resistor. Na Figura 20, observa-se a conexão da fonte de tensão com a placa utilizando-se de um resistor em série e o osciloscópio para realizar a s medidas.

Figura 20 - Teste de consumo da placa de medição de nível

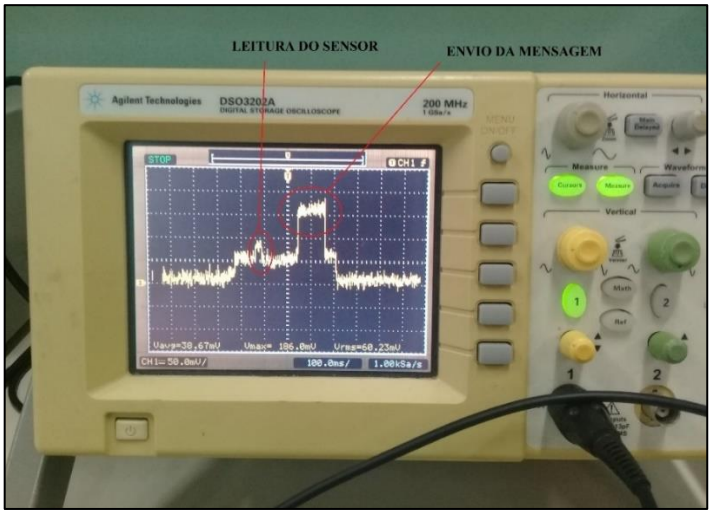


O próprio autor

Foi utilizado um osciloscópio da marca *Agilent Tecnologies DS0302A* (Figura 21), uma fonte regulada modelo MPL-3303 ajustada em 5V. Foi utilizado o multímetro *Agilent U1251A*

onde foi medido o valor de 1,08 Ω para resistor. Durante as medicões foi alterado o parametro do fator de espalhamento e observado as mudaças no comportamento na tensão no resistor.

Figura 21- Medição no osciloscópio



Fonte: O próprio autor

3.3.3 Estrutura do Protótipo

Foi desenvolvido um suporte para fixar o sensor de ultrassom e a placa para medição do nível descrita no tópico 3.2.4. O prototipo pode ser visto na **Erro! Autoreferência de indicador não válida.**.

Figura 22 - Protótipo da unidade remota

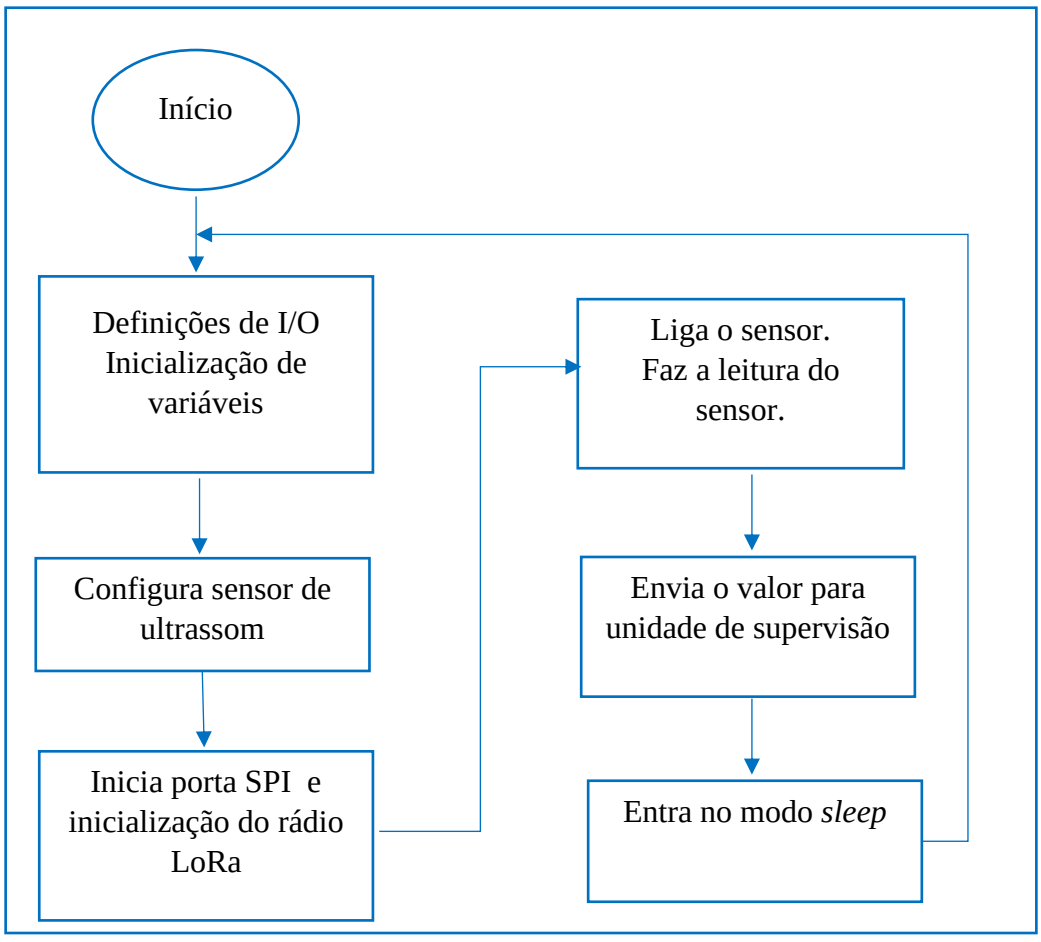
Fonte: O próprio autor

Ele foi construído com perfil de alumínio estrutural, possui um metro de altura, o sensor foi instalado a 85 cm da base e ao lado foi instalado um suporte para a antena e bateria. O recipiente utilizado possui 40 cm de largura, 33cm de comprimento e 22 cm de profundidade.

3.3.4 Fluxograma do código da unidade remota

O programa da unidade remota realiza um fluxo conforme a Figura 23. Inicialmente são realizadas as definiçoes de entradas e saidas, em seguida são configurados o pinos do sensor de ultrasson e a porta SPI de comunucação com o rádio LoRa SX1276. Em seguida o ESP32 polariza o transistor NPN do circuito de alimentação do sensor por meio do GPIO 2. Com o sensor ultrassônico ligado é feita a leitura da distância por meio dos pinos *trig* e *echo*. Em seguida o sensor é desligado. O valor lido é enviado para unidade de supervisão e o dispositivo entra em modo *sleep*. O programa é mostrado no apêndice C.

Figura 23 - Fluxograma do código da unidade remota

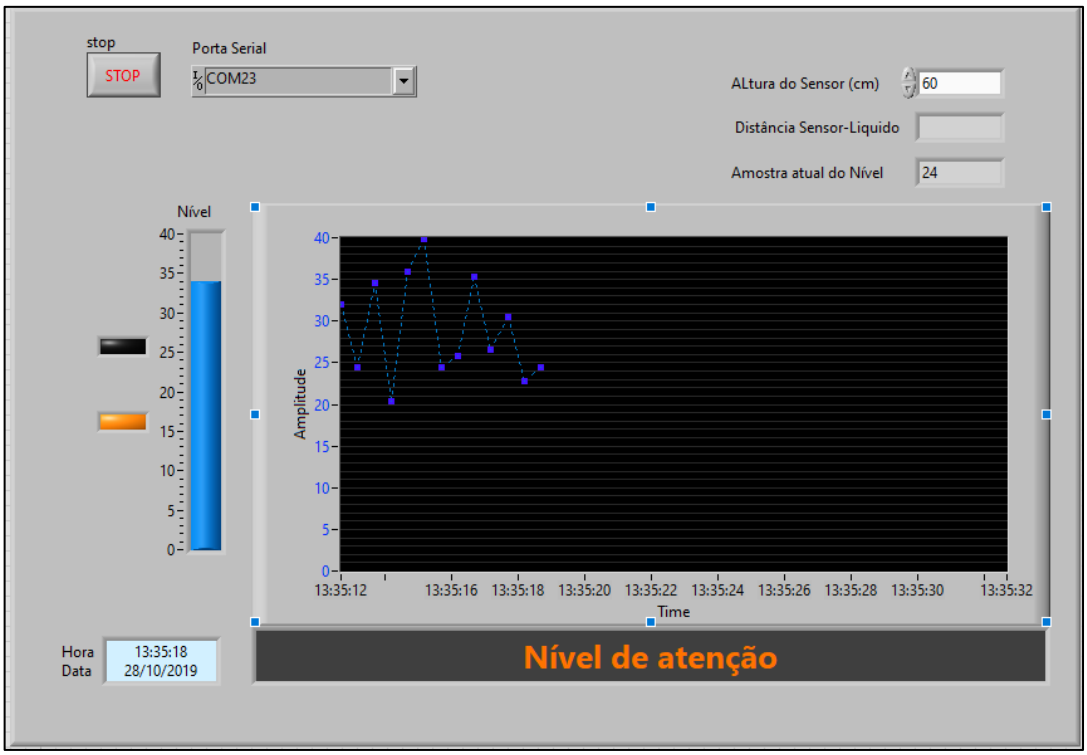


Fonte: O Próprio autor

3.4 DESENVOLVIMENTO DO SISTEMA DE SUPERVISÃO

No desenvolvimento da interface do sistema de supervisão mostrada na Figura 24, foram inseridos no painel frontal uma caixa de seleção para escolha da porta de comunicação serial e um botão de stop para parar o sistema. A esquerda tem-se um Slider que vai representar a régua e indicar o nível do protótipo. Ao lado do *slider* foram adicionados dois indicadores luminosos, um indicador laranjado para mostrar quando o nível ultrapassar um nível de atenção e um indicador vermelho para indicar emergência. No centro, há um gráfico que mostra a variação das leituras de níveis no decorrer do tempo. No canto superior direito tem-se um campo numérico para inserir a altura do sensor, um indicador para mostrar a distância do sensor para o líquido e o última amostra lida. Na parte inferior da interface é mostrada a data, hora e um campo texto para mostrar os alarmes.

Figura 24 - Interface do sistema de supervisão



Fonte: O próprio autor

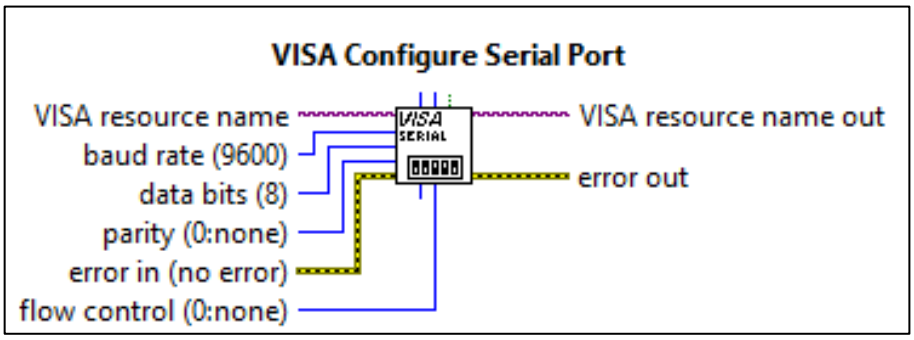
2.3 Configuração da porta serial

Para fazer a leitura da porta serial foi necessário escolher o bloco de configuração da serial no diagrama de blocos. Para acessar este bloco clica-se com o botão direito no diagrama

de blocos para abrir a paleta de funções. No *menu Instrument I/O* e em Serial escolhe-se o bloco Visa *configure Serial Port.*

Na Figura 25, pode-se observar uma representação do bloco de configuração VISA. Na entrada Visa *resource name* foi adicionado um controle para escolha da porta serial. Este controle é visto no painel frontal no canto superior esquerdo e foi adicionado o *label* 'Porta serial'. Em *baud rate* foi adicionada uma constante com valor 9600 bps. E o tamanho do dado foi configurado com outra constante de 8 bits em data bits.

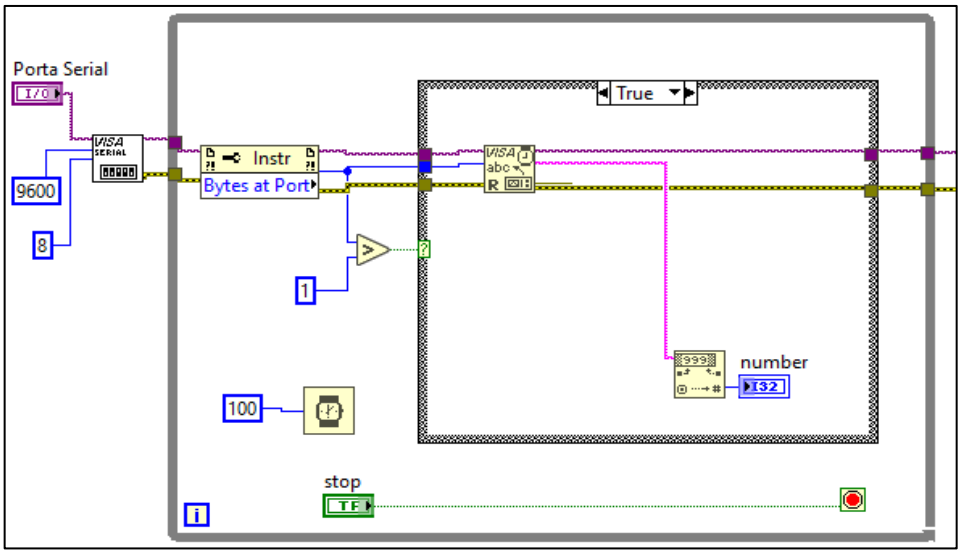
Figura 25 - Bloco de configuração da serial



Fonte: O próprio autor

Após a configuração da porta foi adicionado um laço While para ficar em um *loop* de leitura da porta (Figura 26). Dentro do laço foi adicionado um temporizador *'wait'*, para ler a porta a cada 100 ms. Dentro do laço com o botão direito abriu-se a paleta de funções e em *instrument I/O*, em seguida em Serial, selecionou-se o bloco '*Byte at Port'* . Este bloco checa se tem bytes no buffer da porta serial e devolve a quantidade de bytes disponível para leitura.

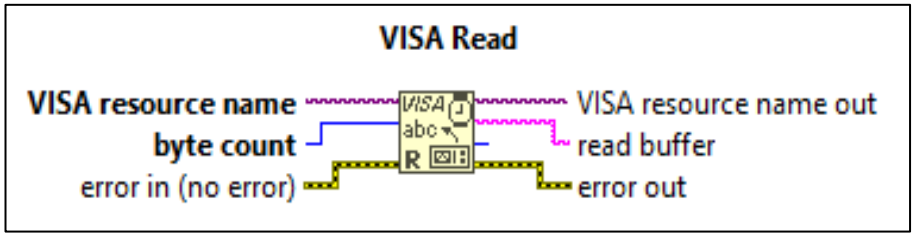
Figura 26 - Diagrama de bloco da leitura da serial



Fonte: O próprio autor

Essa quantidade de byte passa por um comparador que verifica se chegou mais de um byte. Se for verdadeiro ativa uma estrutura case, onde é feita a leitura por meio do bloco VISA *Read* (Figura 27). O bloco *VISA read* lê os bytes disponíveis na serial e devolve uma *string.* Esta *string* passa por um bloco '*Decimal String To Number'* que converte a *string* em um número de 32 bits.

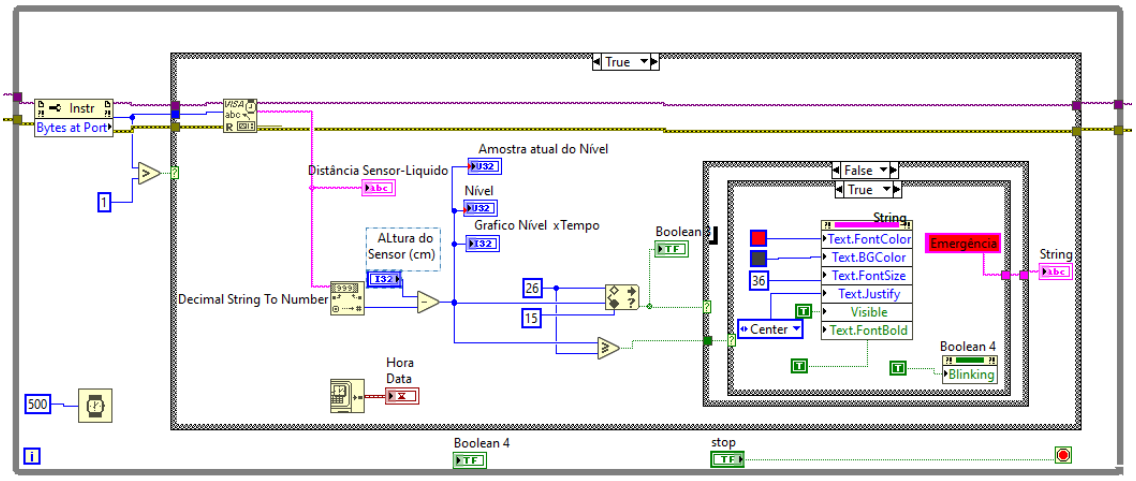
Figura 27 - Bloco VISA Read



Fonte: O próprio autor

Em seguida foi realizado o gerenciamento deste número. O valor da amostra foi direcionado para o gráfico, para o *slider* e para uma rotina de tratamento de alarme.

Figura 28 - Laço While do Programa



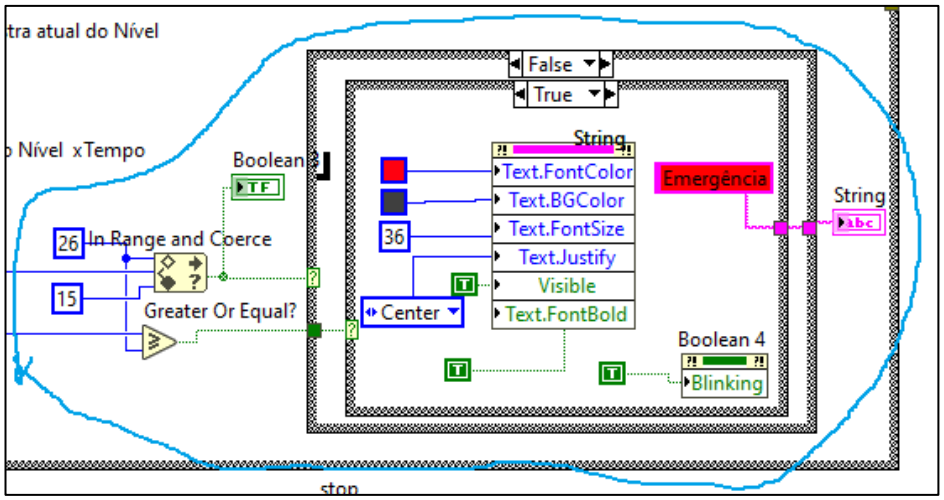
Fonte: O próprio autor

Na Figura 28, pode ser observado que o sinal lido no bloco *VISA read* que é mostrado em um campo de texto que foi chamado de distância sensor líquido. Em seguida o bloco Decimal *String To Number* converte para um número de 32 bits. Este valor subtraído da altura do sensor resulta no valor do nível que é inserido no gráfico Nível x Tempo, no *slider* e no indicador numérico que foi chamado de Amostra atual do Nível.

2.4 Configuração dos Alarmes

Foi desenvolvida uma rotina com duas estruturas ‘case’ como mostrado na Figura 29. O bloco *IN Range and Coerce* testa se o número está entre o limite inferior e o limite superior -1. Neste caso ele vai acionar o laço ‘*case*’ se estiver entre 15 e 25. O bloco *Greter Or Equal*? Verifica se o valor é maior ou igual a 26 e habilita o laço case interno.

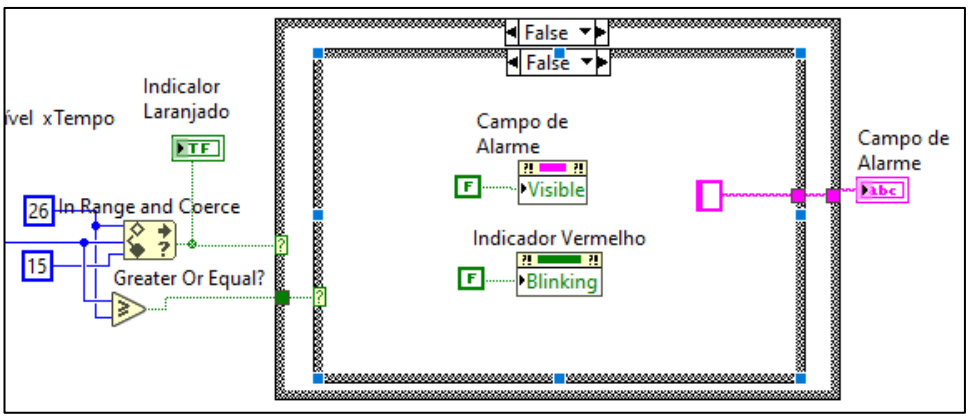
Figura 29 - Tratamento das condições de Alarme



Fonte: O próprio autor

Quando o valor for inferior a 15, tem-se a condição em que os dois casos são falsos. Na Figura 30 pode-se observar esta situação. Foi inserido um *prompt node visible* da *string* Campo de Alarme e colocada a condição falso. Ou seja, o campo de alarme fica invisível quando o número estiver abaixo de 15 cm. O diagrama de blocos completo pode ser observado no Apêndice D.

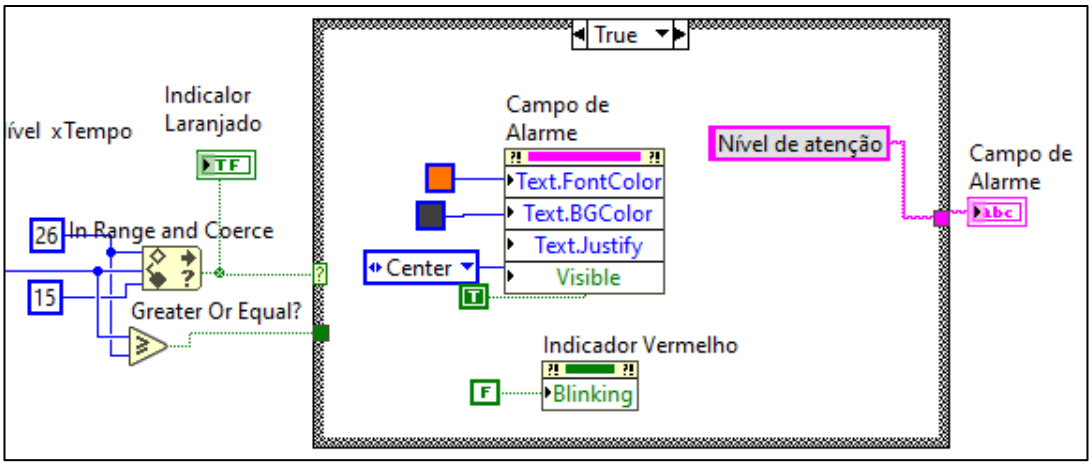
Figura 30 - Rotina de alarme para amostra menor que 15cm



Fonte: O próprio autor

Quando o valor estiver entre 15 e 25 (Figura 31) o bloco *In Range and Coerce* habilita o *Case Structure* (estrutura case) externo como verdadeiro. Nesta situação o Campo de Alarme recebe o texto “Nível de atenção”. Também foi adicionado o *prompt node* do Campo de Alarme para configurar a visibilidade como *True*, cor da fonte laranjada, o texto justificado no centro. E também foi adicionado um *prompt node* para apagar o indicador vermelho para o caso de a condição anterior ter sido de valores maiores que 25cm.

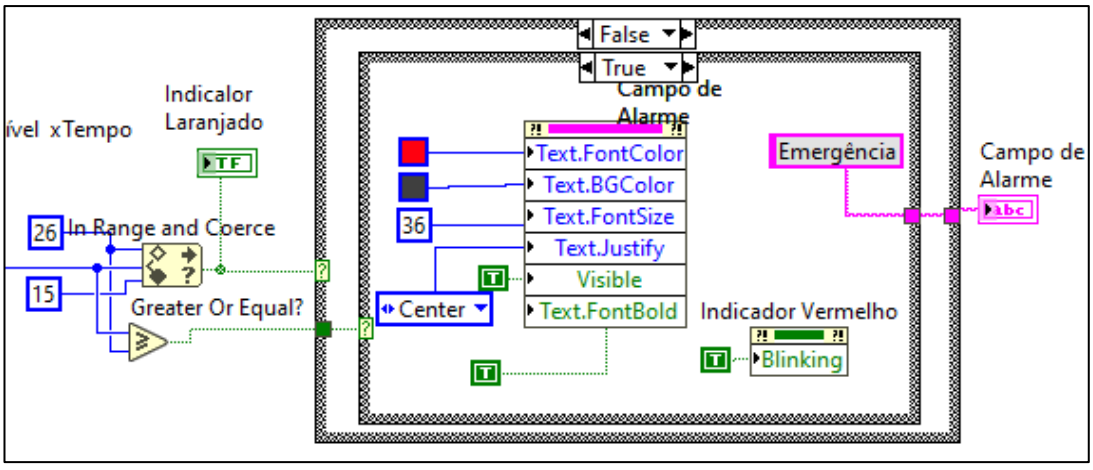
Figura 31 - Rotina de alarme para amostra entre 15cm e 25cm



Fonte: O próprio autor

Quando os valores forem maiores que 25, a estrutura interna é habilitada (Figura 32). O Campo de Alarme recebe o texto Emergência. Foi inserido um *prompt node* para configurar o texto de cor vermelha, visibilidade *true*, centralizado e negrito, fundo cinza escuro e tamanho 36.

Figura 32 - Rotina de alarme para amostra acima de 25 cm



Fonte: O próprio autor

4 RESULTADOS OBTIDOS

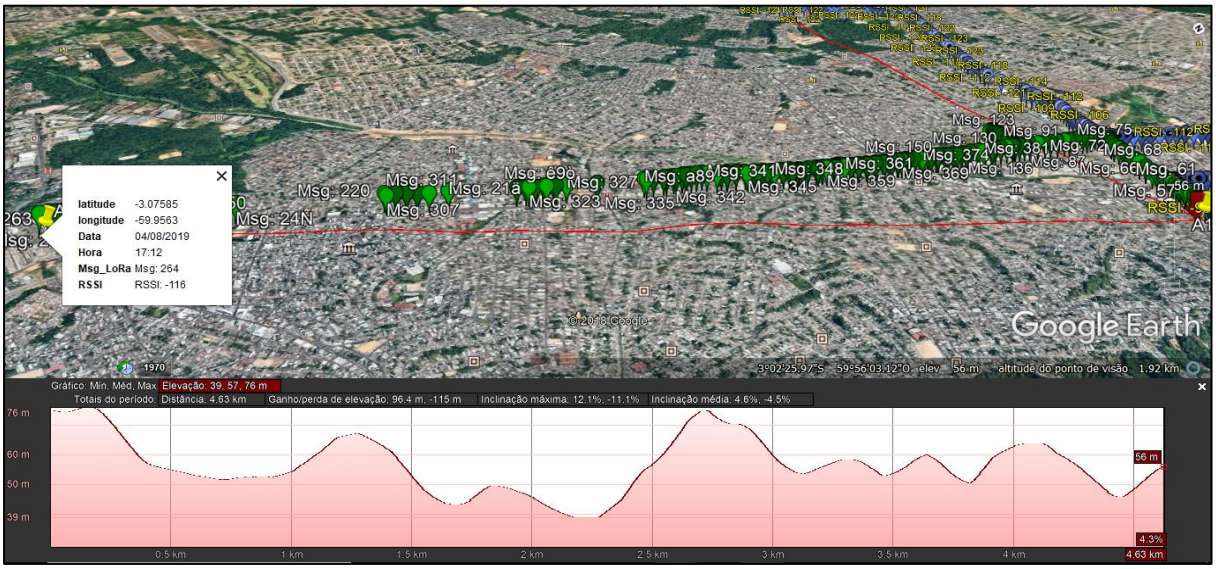
Este capítulo está separado em três seções onde são mostrados alguns resultados importantes dos testes de distância, do teste de consumo da placa de medição de nível utilizada na unidade remota e o resultado da escrita das mensagens e do gráfico na unidade de supervisão, recebendo os dados do protótipo da unidade remota.

4.1 RESULTADOS DO TESTE DE DISTÂNCIA

No teste de distância realizado nos dias 03 e 04 de agosto de 2019, o transmissor foi instalando no ponto de latitude = -3.040568, e longitude = -59.933788, na avenida Itaúba no bairro Jorge Teixeira, em uma altitude de 12 metros do solo. Foram utillizados os seguintes parâmetros: Frequência = 915 MHz, Potência de transmissão = 17dBm, fator de espalhamento = 12, Largura de banda = 125 KHz.

Foi utilizado o *software google Earth Pro* versão 7.3.2 para para se obter o trajeto foi utilizando o arquivo mostrado na seção 3.2.3. Na Figura 33 pode ser observado como foram traçadas as medidas e também é possível ver o perfil de elevação do terreno.

Figura 33 - Trajeto do teste de distância



Fonte: O próprio autor

Foi realizado um trajeto que começa na avenida Itaúba e vai até a rotatoria do bairro São José, na avenida Autaz Mirim. Este trajeto pode ser observado na Figura 33, onde se observa a

mensagem 264, que foi recebita a uma distancia de 4,63 km e com RSSI = -116 dBm. Porém houve algumas perdas no decorrer do trajeto.

Também, foi realizado um trajeto pela avenida camapuã, em direção ao bairro cidade nova. Neste teste o transmissor envia uma string “Teste_8B”, que possui 8 bytes e a mensagem foi recebida na rotatória do hospital Francisca Mendes a uma distância de 3,53km com o parâmentro RSSI= -124 dBm.

4.2 RESULTADO DO TESTE DE CONSUMO DE ENERGIA DA PLACA DE MEDIÇÃO DE NÍVEL

Como observado no tópico 3.2.5, foram realizadas medidas de tensões entre os terminais do resistor shunt de 1Ω e, consequentemente obteve-se a corrente consumida pelo circuito. Foi selecionado no osciloscópio os valores de tensão rms, tensão média, máxima e o tempo de envio da mensagem. Um parâmetro que altera no tempo de envio é o fator de espalhamento. Consequentemente ao aumentar o tempo de envio aumenta-se também o consumo. Na Tabela 1, pode-se observar o aumento da tensão média ao mudarmos o fator de espalhamento de um valor entre 7 e 12.

Tabela 1 - Resultados do teste de consumo

Fator de Espalhamento	Vrms (mV)	Vmédia (mV)	Vmáx (mV)	Tempo de envio(ms)	Tempo Ligado(ms)
7	37,2	20,26	172	32	320
8	48,6	31,71	182	48	364
9	60,23	38,67	186	112	415
10	77	52	186	224	540
11	100,5	74	174	456	780
12	110	78	212	920	1256

Fonte: O próprio autor

No modo sleep o circuito apresentou consumo de 13mA.

A bateria utilizada para alimentar a unidade remota foi o modelo ES500, 20000 mAh que possui uma célula solar para carregar quando estiver exposto a luz.

Com essas características, pode-se estipular um tempo de trabalho para esta bateria com algumas condições previamente definidas. Foi tomado como exemplo o fator de espalhamento igual a 10. Neste caso foi obtido uma corrente média igual a 52mA. Se o sistema enviar uma mensagem a cada 30 minutos, tem-se:

Tempo ligado em uma hora = 2x540ms = 1,08s.

Em uma hora, este tempo corresponde a: $\frac{1.08s}{3600s} = 0,3\text{x}10^{-3}$. Que corresponde a 0,03% de uma hora.

Então o consumo médio de corrente durante uma hora fica:

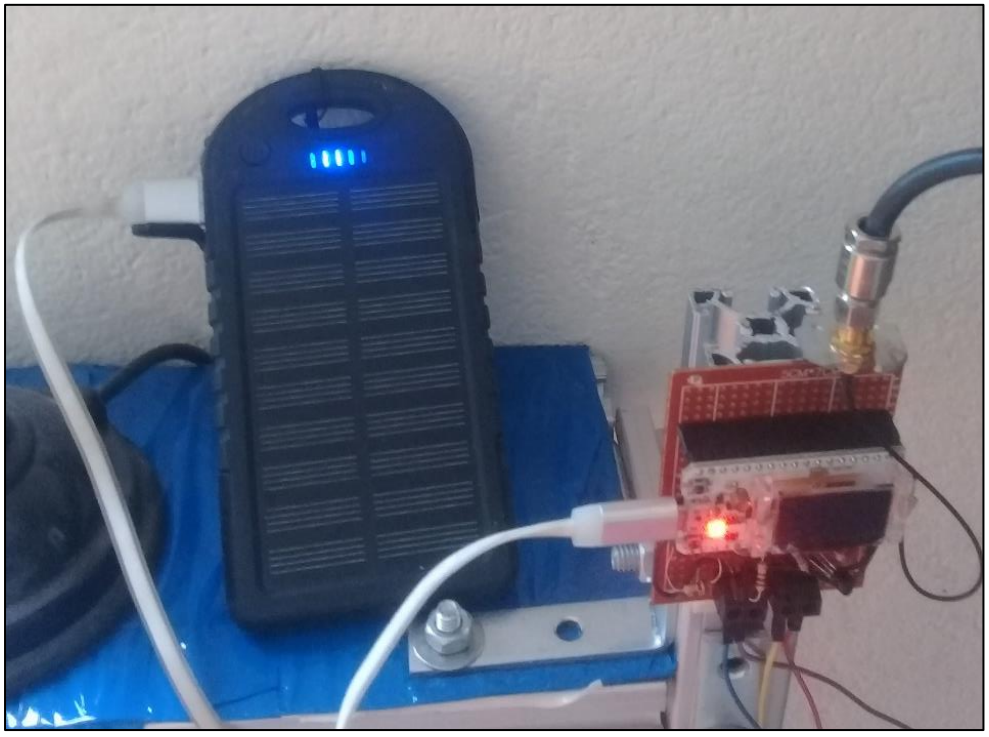
$$Imed = Isleep(1h - Tsleep) + Ion \text{ x } Tsleep$$

$$Imed = 13 \text{ x } (1 - 0,3x10^{-3}) + 52 \text{ x } 0,3 \text{ x } 10^{-3} = 13,0117\ mA$$

Então, nessas condições, o tempo para descarregar a bateria fica:

Duração da bateria = $\frac{20000\ mAh}{13,0117\ mA} = 1537,078\ h$ = 64,04 dias

Figura 34 - Unidade remota alimentada por bateria ES500, 20000mAh



Fonte: O proprio autor

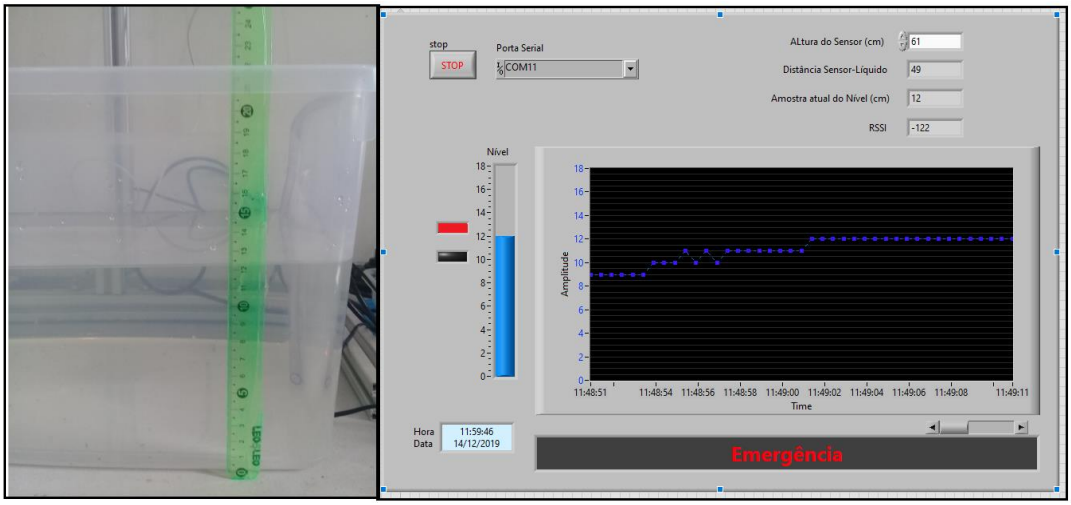
Este seria o tempo para descarregar a bateria mostrada na

Figura 34 . Porém esta bateria leva um tempo de aproximadamente 10 horas para carregar quando exposta a luz, ou seja, estará carregando durante o dia. Desta forma não foi possível mensurar o tempo de duração com a bateria recarregável por luz.

4.3 RESULTADOS DAS MEDIÇOES NO PROTÓTIPO

Os resultados das medições no protótipo pode ser observado na Figura 35. Neste teste o niveis de atenção foram ajustados entre 10 cm e 12 cm. As medições sairam como esperado e no gráfico foi possiível ver a variação do nível e a escreita das mensagens conforme programado. Observando a figura, Pode ser notado que durante a passagem de 10 cm para 11 cm, houve uma instabilidade nas medidas. Isso acontece pelo fato de que o sensor ultrassônico HC-SR04 não tem precisão milimétrica.

Figura 35 - Resultados das medições no protótipo



Fonte: O próprio autor

CONCLUSÃO

No desenvolvimento da presente pesquisa foram feitas revisões dos assuntos de medição de nível por meio de sensor ultrassônico, e da tecnologia LoRa e seus parâmetros de configuração. Foram abordados em seguida, a antena AP3900 e o módulo gps, pois estes foram utilizados para fazer o teste de distância. Também consta uma apresentação sobre os softwares Arduino IDE e LabVIEW, utilizados no desenvolviento da aplicação da medição do nével de água e no sistema de monitoramento.

No teste de distância observou-se que o fator de espalhamento é uma variável muito importante para se obter um alcance melhor. O alcance de 4,63 km foi um resultado além da espectativa pois 3 km já seria um resultado satisfatório com o módulo Wifi LoRa 32(v2). Porém no teste de consumo verificou-se que com o aumento do fator espalhamento, aumenta o consumo de energia pois o tempo de codificação e decodificação do rádio LoRa também aumentam. Na a tabela 1, observa-se o tempo de envio de 32 ms para o espalhamento igual a 7 e o tempo vai crescendo até 920 ms para o Espalhamento igual a 12.

Na unidade de monitoramento se obteve a plotagem dos dados no gráfico como o esperado. Fez-se uma ligação ponto a ponto do monitoramento com o *endpoint* da unidade remota. Pode-se sugerir como melhorias, uma unidade de monitoramento em rede com diversos *endpoints* sendo assim capaz de monitorar diversos locais ao mesmo tempo. Também pode ser sugerido o desenvolvimento de banco de dados para armazenar os dados recebidos em arquivos para que se possa fazer comparacões entre niveis em determinadas épocas do ano.

Com base nos resultados obtidos, verificar-se que, por meio da utilização da tecnologia LoRaWan é possível solucionar a problemática apresentada neste trabalho, que foi, monitorar a longas distâncias os níveis de água e com baixo consumo de energia. Desta forma, a hipótese apresentada na introdução teve sua veracidade comprovada.

REFERÊNCIAS BIBLIOGRAFICAS

BEGA, E, A. et al. **Instrumentação Industrial.** 3. ed**.** Rio de janeiro. Editora Interciência LTDA. 2011.

BLUM, J. **Explorando o Arduino.** Técnicas e ferramentas para mágicas de engenharia. Rio de Janeiro. Alta Books. 2016.

BOR, M.; ROEDIG, U**. LoRa Transmission Parameter Selection.** 2017. Disponível em <https://www.researchgate.net/publication/317585179_LoRa_Transmission_Parameter_Selection/link/59e91a800f7e9bc89b7d0033/download>Acesso em 22 Jul 2019.

BROCKVELD JUNIOR, S. L. **Aula – Medição de Nível.** Instituto Federal de Santa Catarina.2017. Disponível em <http://docente.ifsc.edu.br/sergio.brockveld/MaterialDidatico/Instrumenta%C3%A7%C3%A3o/Aula%20-%20Medi%C3%A7%C3%A3o%20de%20N%C3%ADvel.pdf > Acesso em 12 Abr 2019.

ELECFREAKS. **HC-SR04 User Guide**. 2015. Disponível em<https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf> Acesso em 06 abr 2019.

GARCIA, P. S. R.;KLEINSCHMIDT, J. H. **Tecnologias Emergentes de Conectividade na IoT: Estudo de Redes LPWAN. 2017.** Disponível em< http://www.sbrt.org.br/sbrt2017/anais/1570361881.pdf > Acesso em 25 set. 2019.

HELTEC AUTOMATION. **New version WiFi LoRa 32 (v2).** 2019. Disponível em< http://www.heltec.cn/project/wifi-lora-32/?lang=en>. Acesso em 29 mar. 2019.

LORA-ALLIANCE. **What is the LoRaWAN Specification?**. 2019.Disponível em <https://lora-alliance.org/about-lorawan> Acesso em 06 abr. 2019.

REGAZZI, R. D.; PEREIRA, P. S.; SILVA JR. M. F. **Soluções Práticas de Instrumentação e Automação**. Utilizando a Programação Gráfica LabVIEW. Rio de Janeiro: [s.n]. 2005. Seção I.4.

STEELBRAS ANTENAS. **Folha de dados da antena UHF modelo AP3900**.[s.d]. Disponível em <https://www.steelbras.com.br/wp-content/uploads/2015/02/AP3900.pdf> Acesso em 02 ago. 2019.

SYNACORP. **Arduino GY-NEO6MV2 GPS Module c/w Antenna & Flight Control EEPROM. [s.d].** Disponível em <https://www.epitran.it/ebayDrive/datasheet/NEO6MV2.pdf> Acesso em 10 out. 2019.

APÊNDICE A – CÓDIGO DO TRANSMISSOR DO TESTE DE DISTÂNCIA

```
/*************************************************************************
                        UNIVERSIDADE DO ESTADO DO AMAZONAS
 Projeto de conclusão de curso 2
 Autor: Francisco Lúcio Rodrigues de Araújo

 Este programa tem a finalidade de Transmitir uma mensagem 'contador' utilizando a plataforma Wifi LoRa 32(v2).
*************************************************************************/

#include <SPI.h>  //Responsável pela comunicação Serial do LoRa
#include <LoRa.h>  //Responsável pela comunicação com o WiFi LoRa
#include <Wire.h>  //Responsável pela comunicação i2c
#include "SSD1306.h"  //Responsável pela comunicação com o display

// Define os pinos do WiFi LoRa
#define SCK     5      // GPIO5  -- SX127x's SCK
#define MISO    19   // GPIO19 -- SX127x's MISO
#define MOSI    27   // GPIO27 -- SX127x's MOSI
#define SS      18      // GPIO23 -- SX127x's CS
#define RST     14    // GPIO14 -- SX127x's RESET
#define DI00    26    // GPIO26 -- SX127x's IRQ(Interrupt Request)
#define BAND    915E6  // Define a frequência do LoRa. 915E6 é a mesma coisa de 915MHz.
#define PABOOST true   // PABOOST é uma saída do chip SX1276.
unsigned int counter = 0;  // Variável para envio e incremento

////////////////////// Pinos do display Oled ////////////////////////
#define oledRST 16          // pino de reset do oled
#define oledSDA 4           // pino SDA para comunicação i2c com oled
```

```
#define oledSCL 15         // pino SCL para comunicação i2c com oled
SSD1306 display(0x3c, oledSDA, oledSCL);  //Define o endereço do i2c do Oled(0x3c)
                                          //e os pinos SDA(4) e SCL(15) do ESP32
#define ledPlaca 25       // GPIO 25 para o led da placa

void setup(){
  pinMode(oledRST,OUTPUT);        //Configura pino Reset do Oled como saida
  pinMode(ledPlaca,OUTPUT);       //confidura o pino do led da placa como saída

  //Reseta o display oled
  digitalWrite(oledRST, LOW);      //Coloca o pino em Low para dar um Reset no Oled
  delay(50);
  digitalWrite(oledRST, HIGH);       //Coloca em High para o Oled voltar a funcionar

  display.init();                          //Inicializa o Oled
  display.flipScreenVertically();     //Vira o Display para a vertical
  display.setFont(ArialMT_Plain_10);//Define o tipo e tamanho da fonte
  delay(1500);
  display.clear();                //Limpa a tela

  // Inicia porta SPI para comunicação com o chip SX1276
  SPI.begin(SCK,MISO,MOSI,SS);      //Inicializa a comunicação Serial com o LoRa
  LoRa.setPins(SS,RST,DI00);          //Define os pinos que serão utilizados pelo LoRa

  if (!LoRa.begin(BAND,PABOOST)){   //Verifica se o LoRa foi iniciado com sussesso
   //Seta o cursor em X=0 e Y=0 e imprimi o texto a seguir.
    display.drawString(0, 0, "Falha ao iniciar o LoRa!");
    display.display();            //Imprime o texto
    while (1);                     //Entra em um While e a execução do programa morre aqui
  }
```

```
  /*** Configuração de parametros ***/
  LoRa.setTxPower(17,PABOOST);           // Potência do trasmissor
  LoRa.setSpreadingFactor(12);           // Fator de espalhemento entre 6 e 12
  LoRa.setCodingRate4(8);                // denominador da taxa de codificação entre 5 e 8
 // LoRa.setSignalBandwidth(125E3);       // Largura de banda do sinal padrão 125E3
  display.drawString(0, 0, "Iniciado com sussesso!");
  display.display();
  delay(1000);
}
void loop(){
  display.clear();
  display.setTextAlignment(TEXT_ALIGN_LEFT);  //Alinha o texto a Esquerda
  display.setFont(ArialMT_Plain_10);                  //Define o tipo e tamanho da fonte
  display.drawString(0, 0, "Enviando pacote:");
  display.drawString(84, 0, String(counter));  //Imprime o valor da variável 'counter' em forma
                                                //de String
  display.display();

  LoRa.beginPacket();           //Inicia um pacote
  //LoRa.print("Teste_8B");  //Envia a seguinte palavra 'Teste_8B' com 8 bytes
  LoRa.print(counter);          //Envia o valor da variável 'counter'
  LoRa.endPacket();              //Fecha o pacote e envia
  counter++;                      //Incrementa +1 ao valor da variável
  digitalWrite(25, HIGH);      //Liga o led da placa
  delay(1000);
  digitalWrite(25, LOW);      //Desliga o led da placa
  delay(2000);}
```

APÊNDICE B - CÓDIGO DO RECEPTOR DO SINAL LORA E REGISTRO DO POSICIONAMENTO.

```
/*************************************************************************
        UNIVERSIDADE DO ESTADO DO AMAZONAS
 Projeto de conclusão de curso 2
 Autor: Francisco Lúcio Rodrigues de Araújo
 Este programa tem a finalidade de receber uma mensage de dispositivo LoRa e receber o sinal de gps e gravar a mensagem e o posicionamento em um cartão microSD.
 *************************************************************************/

#include <HardwareSerial.h>
#include<TinyGPS++.h>
#include <SPI.h>
#include <Wire.h>
#include <SD.h>
#include <LoRa.h>        //Resposável pela comunicação com o WiFi LoRa
#include "SSD1306.h"    //Resposável pela comunicação com o display

////////////////////  Configuração do GPS  ////////////////////
#define gps_tx_esp32_rx 38
#define gps_rx_esp32_tx 39
HardwareSerial  gps_com(1); // Define a porta serial 1 para comunicar com gps.
TinyGPSPlus gps;                  // Instancia objeto gps para receber os dados do gps

 String _data = "";
 String hora="";
 String longitude = "";
 String latitude = "";
 int _hora,_minuto;
////////////////////////////////////     DEFINICOES E VARIAVEIS LORA    ////////////////////////////////////
```

```
#define SCK     5       // GPIO5  -- SX127x's SCK
#define MISO    19    // GPIO19 -- SX127x's MISO
#define MOSI    27    // GPIO27 -- SX127x's MOSI
#define SS        18      // GPIO18 -- SX127x's CS
#define RST     14      // GPIO14 -- SX127x's RESET
#define DI00    26      // GPIO26 -- SX127x's IRQ(Interrupt Request)

int ledVerde = 22;
int ledAzul = 2;
int rstOled = 16;

#define BAND    915E6    //Define a frequência do LoRa 915MHz
#define PABOOST true     // PABOOST é uma saída do chip SX1276.

SSD1306 display(0x3c, 4, 15);  //Define o endereço do i2c do Oled(0x3c) e os pinos SDA(4) e SCL(15) do ESP32
String rssi = "RSSI --";        //Declara a variável que receberá a potência do sinal
String packSize = "--";        //Declara a variável que receberá o tamanho dos pacotes enviados
String packet;                    //Declara a variável do pacote
String Msg_LoRa = "Msg: ";

////////////////////  VARIÁVEIS DO MÓDULO MICROSD   ////////////////////
#define SD_CS 23
#define SD_SCK 17
#define SD_MOSI 12
#define SD_MISO 13

SPIClass sd_spi(HSPI);  // Declara porta hardware SPI p/ comunicar com cartão uSD

#define microSD_CS 23
#define nomeArquivo "/gpslog"          // Nome do arquivo de Registro
```

```
#define numeroMaximoDeArquivos 100      // Número de arquivos de registros que podem ser feitos
#define sufixoArquivo "csv"             // Sufixo do arqui// Extensão
#define numeroColunas 6
char nomeDoArquivo[13];                 // Char string para armazenar o nome do arquivo

String cabecalho[numeroColunas] = {"Latitude","Longitude","Data","Hora","Msg_LoRa","RSSI"};


////////////////////////////////////////////////////////////////
void escreveMicroSD(String dados[numeroColunas])
{
  File logFile = SD.open(nomeDoArquivo, FILE_APPEND);

 if (!logFile)  // Se o arquivo de registro for aberto, imprime os nomes das colunas no arquivo
 {
      Serial.println("Nao escreveu no microSD. O arquivo não foi aberto");
      display.drawString(0 , 37 , "Não Escreveu no Cartão " );
      display.display();
 }
 else{
     Serial.println("Escreveu no microSD");
   display.drawString(0 , 37 , "Escreveu no Cartão " );
   display.display();
   int i = 0;
   for (; i < numeroColunas; i++)
   {
     logFile.print(dados[i]);
     if (i < numeroColunas - 1)    // If i até a penultima coluna
       logFile.print(',');         // imprime a virgula
```

```
      else                      // senao, Se imprimiu a ultima coluna
        logFile.println();      // imprime outra linha
    }
    logFile.close();            // Fecha o arquivo
  //  logFile =0;
      }
}

// Função para criar novo arquivo
void updateFileName()
{
  int i = 0;
  for (; i < numeroMaximoDeArquivos; i++)
  {
    memset(nomeDoArquivo, 0, strlen(nomeDoArquivo)); // Apaga a string nomeDoArquivo
    // Seta nomeDoArquivo com i+1 "gpslogXX.csv":
    sprintf(nomeDoArquivo, "%s%d.%s", nomeArquivo, i, sufixoArquivo);
    Serial.println(nomeDoArquivo);
    if (!SD.exists(nomeDoArquivo)) // Verifica se o arquivo não existe
    {
      Serial.println("ponto1");
      break;                    // Break out of this loop. We found our index
    }
    else                        // se o arquivo existe
    {
      Serial.print(nomeDoArquivo); // imprime nome arquivos emcontrados
      Serial.println(" exists");
      Serial.println("ponto2");
    }
  }
  Serial.print("File name: ");
```

```
  Serial.println(nomeDoArquivo);   // Imprime nome do novo arquivo
}

////////////// AJUSTA DATA E HORA PARA O FUSO HORÁRIO LOCAL //////////////
void trata_Data_hora(void)
{
    _hora = gps.time.hour();     // Retorna a hora dos sinal do gps
    _minuto = gps.time.minute(); // Retorna o minuto
     if(_hora>=4)                // se hora maior que 4:00
     {
      hora = _hora - 4;          // Ajusta fuso horários
      _data = (gps.date.day());       // Pega data do gps atual
     }else                       // Senão
     {
       hora = (_hora + 20);         // Ajusta fuso horário
       _data= (gps.date.day() - 1);  // retorna data anterior
     }
     hora+=':';hora+=_minuto;     // Concatena Hora:Minuto
     _data+='/';
     _data+=gps.date.month();     // Concatena mes na data
     _data+='/';
     _data+=gps.date.year();      // Concatena ano na data
}
////////////////////////////////////////////////////////////////////////////////

//////////////////////////  FUNÇÃO PARA LEITURA DO PACOTE LORA //////////////////////////
void cbk(int packetSize) {
  Msg_LoRa = "Msg: ";  //
  packSize = String(packetSize, DEC);  //Converte o valor da variável em quantidade de bytes recebidos
```

```
  for (int i = 0; i < packetSize; i++) {  //Executa o código abaixo até que todos os bytes recebidos foram tratados
    Msg_LoRa += (char) LoRa.read();  //Trata byte por byte ate formar o dado enviado
  }
  rssi = "RSSI " + String(LoRa.packetRssi(), DEC) ;  //Converte o valor da potência do sinal e junta a palavra "RSSI"
  loraData(packetSize);  //Chama a função que imprime os dados no Oled
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void loraData(int packetSize) {
  display.clear();
  display.setTextAlignment(TEXT_ALIGN_LEFT);  //Alinha o texto a Esquerda
  display.setFont(ArialMT_Plain_10);  //Define o tipo e tamanho da fonte
  display.drawString(0, 0, rssi);  //Imprime a variável com a potência do sinal
  display.drawString(0 , 15 , "Received " + packSize + " bytes");  //Imprime o tamanho do pacote enviado
  display.drawStringMaxWidth(0 , 26 , 128, Msg_LoRa);  //Imprime os dados enviado do outro LoRa
  display.display();
}
////////////////////////////////////////////////// SETUP ////////////////////////////////////////////////////////
void setup()
  {
     Serial.begin(9600);  //  Inicia porta de comunicação serial  com gps
     gps_com.begin(9600,SERIAL_8N1,gps_tx_esp32_rx,gps_rx_esp32_tx);
     pinMode(ledVerde,OUTPUT);
     pinMode(rstOled, OUTPUT);          //Define o pino Reset do Oled
     digitalWrite(rstOled, LOW);        //Coloca o pino em Low para dar um Reset no Oled
     delay(50);
     digitalWrite(rstOled, HIGH);       //Coloca em High para o Oled voltar a funcionar
     display.init();                    //Inicializa o Oled
     display.flipScreenVertically();    //Vira o Display para a vertical
```

```
  display.setFont(ArialMT_Plain_10); //Define o tipo e tamanho da fonte
 // delay(1500);
 display.clear();  //Limpa a tela

//// Inicializa rádio LoRa
SPI.begin(SCK,MISO,MOSI,SS);       //Inicializa a comunicação Serial com o LoRa
LoRa.setPins(SS,RST,DI00);          //Define os pinos que serão utilizados pelo LoRa

if (!LoRa.begin(BAND, PABOOST))  //Verifica se o LoRa foi iniciado com sussesso
{
display.drawString(0, 0, "Falha ao iniciar o LoRa!");  // Seta o X e Y de onde irá imprimir o
                                                       // Texto a seguir
display.display();          // Imprime o texto
while (1);                  // Entra em um While e a execução do programa morre aqui
}
display.drawString(0, 0, "Iniciado com sussesso!");
display.drawString(0, 12, "Esperando por dados...");
display.display();
delay(1000);
LoRa.setTxPower(17,PABOOST);      // Potência do trasmissor
LoRa.setSpreadingFactor(12);          // Fator de espalhemento entre 6 e 12
LoRa.setCodingRate4(8);               // denominador da taxa de codificação entre 5 e 8
//LoRa.setSignalBandwidth(125E3);    // Largura de banda do sinal padrão 125E3
LoRa.receive();                       // Habilita o LoRa para receber dados
 //delay(500);

//// Inicia a comunicão SPI com o micro SD
 sd_spi.begin(SD_SCK, SD_MISO,SD_MOSI, SD_CS);
 if (!SD.begin(SD_CS, sd_spi))
    {
      Serial.println("Error ao inicializar o uSD card.");
```

```
        display.drawString(0, 24, "uSD, Erro !!!");
        display.display();
      }else{
        Serial.println("uSD inicilizado ...");
        display.drawString(0, 24, "uSD inicializado..");
        display.display();
        updateFileName();     // Quando iniciar, cria um covo arquivo com nome incrementado
        escreveMicroSD(cabecalho);  // Escreve o cabeçalho no novo arquivo
        Serial.println("Setup ok");      // para debugar no terminal serial.
      } }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////// LAÇO PRINCIPAL ////////////////////////////////////////
void loop()
{
 ////////////////////////////////////////  LER DADOS DO GPS ////////////////////////////////////
      while (gps_com.available())
        {
          gps.encode(gps_com.read());  // Ler dados do gps pela porta serial gps_com
        }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    // gps.location.isUpdated();
     latitude = String(gps.location.lat(),6);
     longitude = String(gps.location.lng(),6);
     trata_Data_hora();
   int packetSize = LoRa.parsePacket();  // verifica se recebeu chegou algum dado
   if(packetSize) // Se recebeu dados na serial, grava no cartão com as coordenadas
   {
     digitalWrite(ledVerde,HIGH);
     Serial.println("Dados recebidos do esp32");
     cbk(packetSize);
     String dados[] = {latitude,longitude,_data,hora,Msg_LoRa,rssi};   // vertor de variaveis
```

```
  escreveMicroSD(dados);

   // Para visualização no terminal Serial. Para debugar o código.
   Serial.println(Msg_LoRa);
   Serial.println("RSSI: " + rssi);
   Serial.println(  latitude);
   Serial.println(longitude);
   Serial.println("Data: " + _data);
   Serial.println("Hora: "+ hora);
   Serial.println();
   delay(100);
   digitalWrite(ledVerde,LOW);
   ////////////////////////////////////////////
   //// RESETA VARIAVEIS //////////////////////
   packetSize=0; rssi = "";
} }
```

APÊNDICE C - CÓDIGO DA UNIDADE REMOTA

```
/*************************************************************************
        UNIVERSIDADE DO ESTADO DO AMAZONAS
 Projeto de conclusão de curso 2
 Autor: Francisco Lúcio Rodrigues de Araújo

 Este programa tem a finalidade fazer a leitura do sensor de ultrassom e
 enviar para unidade de supervisão por meio do rádio LoRa e entrar em modo
 sleep por tempo determinado para economisar energia. Em seguida repetir o
 ciclo.
*************************************************************************/
#include "esp_sleep.h"
#include <SPI.h>  //Resposável pela comunicação Serial do LoRa
#include <LoRa.h>  //Resposável pela comunicação com o WiFi LoRa
#include <Wire.h>  //Resposável pela comunicação i2c
#include "Ultrasonic.h"

////////////  DEFINICÃO DOS PINOS DO ULTRASSOM  ////////////
#define   trig  12
#define   echo  13
#define   sensorVcc 2
Ultrasonic ultrasonic(trig,echo);

// Define os pinos do WiFi LoRa
#define SCK     5      // GPIO5  -- SX127x's SCK
#define MISO    19     // GPIO19 -- SX127x's MISO
#define MOSI    27     // GPIO27 -- SX127x's MOSI
#define SS      18     // GPIO23 -- SX127x's CS
#define RST     14     // GPIO14 -- SX127x's RESET
#define DI00    26     // GPIO26 -- SX127x's IRQ(Interrupt Request)
```

```
#define BAND    915E6  //Define a frequência do LoRa. 433E6 é a mesma coisa de 433000000MHz. Você também pode usar 868E6 e 915E6.

#define PABOOST true   //Sem conhecimento dessa variavel mas ela deve aparecer para funcionar


byte enderecoLocal = 0xA1;  //Endereço da unidade Remota

byte Destinatario = 0xA0;     // Endereço da unidade Receptora

unsigned int counter = 0;      //Declara a variável que irá receber incrementos e
                               //envia os dados para o outro LoRa



//String rssi = "RSSI --";      //Declara a variável que receberá a potência do sinal

//String packSize = "--";       //Declara a variável que receberá o tamanho dos pacotes enviados

//String packet;              //Declara a variável do pacote


#define uS_TO_S_FACTOR 1000000  // Fator de conversão de milisegundos para segundos

#define TIME_TO_SLEEP  5        // Tempo que o Esp32 fica em modo sleep

//RTC_DATA_ATTR int bootCount = 0;


void setup(){


      pinMode(sensorVcc, OUTPUT);

      SPI.begin(SCK,MISO,MOSI,SS);  //Inicializa a comunicação Serial com o LoRa

      LoRa.setPins(SS,RST,DI00);         //Define os pinos que serão utilizados pelo LoRa


      if (!LoRa.begin(BAND,PABOOST))

      {  //Verifica se o LoRa foi iniciado com sussesso

            while (1);   //Entra em um While e a execução do programa

                         //Fica em loop infinito

      }
```

```
  /*** Configuração de parametros ***/
  LoRa.setTxPower(17,PABOOST);           // Potência do trasmissor
  LoRa.setSpreadingFactor(12);           // Fator de espalhemento entre 6 e 12
  LoRa.setCodingRate4(8);                // denominador da taxa de codificação entre 5 e 8
  LoRa.setSignalBandwidth(125E3);        // Largura de banda do sinal padrão 125E3

  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
 // esp_sleep_pd_config(ESP_PD_DOMAIN_MAX, ESP_PD_OPTION_OFF);
 // esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_PERIPH, ESP_PD_OPTION_OFF);
 // esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_SLOW_MEM, ESP_PD_OPTION_OFF);
 // esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_FAST_MEM, ESP_PD_OPTION_OFF);
 }

void loop(){
  digitalWrite(sensorVcc,HIGH);
  delay(80);
  String distancia = String(ultrasonic.Ranging(CM));
  Serial.println();
  Serial.print(distancia);
  Serial.println(" cm");
  digitalWrite(sensorVcc,LOW);

  // ENVIA O PACOTE
  LoRa.beginPacket();        //Inicia um pacote
  LoRa.write(enderecoLocal);
  LoRa.print(distancia);
  LoRa.endPacket();          //Fecha o pacote e envia
  esp_deep_sleep_start();  //Entra em modo sleep
}
```

APÊNDICE D – CODIGO DE BLOCOS DA UNIDADE DE SUPERVISÃO