**CENTRO PAULA SOUZA**

**RUBENS DE FARIA E SOUZA**

**Eletrônica**

**Antonio Zanini de Farias**

**Caio César Corrá Bello**

**Gustavo Teixeira Morais**

**Hebert Prestes Lima**

**CONTROLE DE ALIMENTAÇÃO ANIMAL**

**SOROCABA**

**2020**

**Antonio Zanini de Farias**

**Caio César Corrá**

**Gustavo Teixeira Morais**

**Hebert Prestes Lima**

**CONTROLE DE ALIMENTAÇÃO ANIMAL**

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Eletrônica da Etec Rubens de Faria e Souza, orientado pelos Professores Carlos Lourenço Magoga e Fausto de Barros, como requisito parcial para obtenção do título de técnico em Eletrônica.

**SOROCABA**

**2020**

BANCA DE AVALIAÇÃO

Examinador 1: Nome: Assinatura:

Titulação:

Examinador 2: Nome: Assinatura:

Titulação:

Examinador 3: Nome: Assinatura:

Titulação:

Data: Menção:

RESUMO

O projeto descrito nesse trabalho de conclusão de curso foi idealizado para atender à proposta de criar um dispositivo que venha a facilitar a vida das pessoas, para isso, escolhemos o desenvolvimento de um alimentador de animais automatizado.

Um dispositivo como esse pode ser de grande valia para uso de pessoas com animais domésticos, assim como, pessoas que possuem grandes criações de animais. Tendo isso em vista, utilizamos uma metodologia de pesquisa-ação para solucionar este desafio.

Como consequência deste projeto, conseguimos unir vários tipos de tecnologias distintas para construir a nossa solução planejada, nos levando a adquirir valiosos conhecimentos na área de estudo.

Palavras-Chave: Alimentação. Animais. Criação de Animais. Pessoas.

ABSTRACT

The project described in this course conclusion work was idealized to meet the proposal to create a device that will facilitate people's lives, for this, we chose the development of an automated animal feeder.

A device like this can be of great value for use by people with pets, as well as people who have large livestock. With this in mind, we use an action research methodology to solve this challenge.

As a result of this project, we managed to unite several types of different technologies to build our planned solution, leading us to acquire valuable knowledge in the study area.

Keywords: Feeder. Livestock. People. Pets.

LISTA DE ILUSTAÇÕES

[Figura 1. Exemplo de programa Node-RED, uma aplicação de alerta climático. 14](#_Toc55736772)

[Figura 2. Barramento I2C. 15](#_Toc55736773)

[Figura 3 - Circuito de Alimentação 17](#_Toc55736774)

[Figura 4 Circuito Regulador de Tensão para 3.3V. 18](#_Toc55736775)

[Figura 5. Motor DC GA12-N20-200. 19](#_Toc55736776)

[Figura 6. Módulo Relé 5V 1 Canal 20](#_Toc55736777)

[Figura 7 - Vista frontal da rosca transportadora 20](#_Toc55736778)

[Figura 8 - Vista isométrica da rosca transportadora. 21](#_Toc55736779)

[Figura 9 - Modelo do corte a laser da rosca transportadora 22](#_Toc55736780)

[Figura 10 - Sensor de Presença e Seus Terminais 23](#_Toc55736781)

[Figura 11 - Display 16x2 e a indicação de seus terminais 24](#_Toc55736782)

[Figura 12 - Descrição do Módulo I2C 25](#_Toc55736783)

[Figura 13 - Configuração Física de um Módulo Buzzer Passivo. 25](#_Toc55736784)

[Figura 14 - Placa ESP32 DEVKIT V1 26](#_Toc55736785)

[Figura 15 - Ligações físicas do ESP32 com os sensores e atuadores. 26](#_Toc55736786)

[Figura 16 - Fluxograma do Processo de Alimentação 29](#_Toc55736787)

[Figura 17 - Fluxograma da Manutenção de Tempo 32](#_Toc55736788)

[Figura 18 - Interface do Reservatório 34](#_Toc55736789)

[Figura 19 - Fluxo da interface do reservatório 35](#_Toc55736790)

[Figura 20 - Interface do Abastecimento 36](#_Toc55736791)

[Figura 21 - Fluxo da interface do abastecimento 37](#_Toc55736792)

LISTA DE ABREVIATURAS E SIGLAS

Direct Current (DC)

Inter-Integrated Circuit (I2C)

Internet of Things (IoT)

Internet Protocol (IP)

Liquid Crystal Display (LCD)

Megabit (Mbit)

Message Queue Telemetry Transport (MQTT)

Microfarad (µF)

Quilobit (Kbit)

Rotações por Minuto (RPM)

Secure Sockets Layer (SSL)

Service Set Identifier (SSID)

Transmission Control Protocol (TCP)

Transport Layer Security (TSL)

Nanofarad (nF)

SUMÁRIO

[1 INTRODUÇÃO 10](#_Toc55736793)

[2 DESENVOLVIMENTO 11](#_Toc55736794)

[2.1 Fundamentação Teórica 11](#_Toc55736795)

[2.1.1 IOT 11](#_Toc55736796)

[2.1.1.1 IoT Industrial 12](#_Toc55736797)

[2.1.2 WI-FI 12](#_Toc55736798)

[2.1.3 MQTT 12](#_Toc55736799)

[2.1.3.1 Características 13](#_Toc55736800)

[2.1.3.2 Funcionamento do MQTT 13](#_Toc55736801)

[2.1.4 Node-RED 14](#_Toc55736802)

[2.1.5 I2C 15](#_Toc55736803)

[2.2 Projeto 16](#_Toc55736804)

[2.2.1 Hardware 16](#_Toc55736805)

[2.2.1.1 Circuito de Alimentação 16](#_Toc55736806)

[2.2.1.1.1 Alimentação Chaveada 17](#_Toc55736807)

[2.2.1.1.2 Circuito Regulador de Tensão 18](#_Toc55736808)

[2.2.1.2 Abastecimento 18](#_Toc55736809)

[2.2.1.2.1 Motor de Abastecimento 19](#_Toc55736810)

[2.2.1.2.2 Relé de Acionamento do Motor 19](#_Toc55736811)

[2.2.1.2.3 Rosca helicoidal 20](#_Toc55736812)

[2.2.1.3 Sensores e Atuadores Auxiliares 22](#_Toc55736813)

[2.2.1.3.1 Sensores Abastecimento 22](#_Toc55736814)

[2.2.1.3.1.1 Configuração e Funcionamento 23](#_Toc55736815)

[2.2.1.3.2 Display informativo 23](#_Toc55736816)

[2.2.1.3.2.1 Display 24](#_Toc55736817)

[2.2.1.3.2.2 Módulo I2C 24](#_Toc55736818)

[2.2.1.3.3 Buzzer Sinalizador 25](#_Toc55736819)

[2.2.1.4 ESP32 26](#_Toc55736820)

[2.2.2 Software 27](#_Toc55736821)

[2.2.2.1 Linguagem C (ESP32) 27](#_Toc55736822)

[2.2.2.1.1 Alimentador 27](#_Toc55736823)

[2.2.2.1.2 WI-FI 30](#_Toc55736824)

[2.2.2.1.3 MQTT 30](#_Toc55736825)

[2.2.2.1.4 Armazenamento 31](#_Toc55736826)

[2.2.2.1.5 Horário 31](#_Toc55736827)

[2.2.2.1.6 Buzzer 32](#_Toc55736828)

[2.2.2.2 Node-RED 33](#_Toc55736829)

[2.2.2.2.1 Interface do Reservatório 34](#_Toc55736830)

[2.2.2.2.2 Interface do Abastecimento 36](#_Toc55736831)

[3 CONCLUSÃO 38](#_Toc55736832)

[REFERÊNCIAS 39](#_Toc55736833)

[ANEXO 1 – Programa em linguagem C (ESP32) 41](#_Toc55736834)

[Header do Projeto (AlimentadorESP32.h) 41](#_Toc55736835)

[Módulo Principal (AlimentadorESP32.ino) 42](#_Toc55736836)

[Módulo do Alimentador (Alimentador.ino) 44](#_Toc55736837)

[Módulo do Armazenamento (Armazenamento.ino) 48](#_Toc55736838)

[Header do Módulo Buzzer (Buzzer.h) 51](#_Toc55736839)

[Módulo Buzzer (Buzzer.ino) 58](#_Toc55736840)

[Módulo Display (Display.ino) 62](#_Toc55736841)

[Módulo MQTT (MQTT.ino) 63](#_Toc55736842)

[Módulo de Gerenciamento de Tempo (Time.ino) 66](#_Toc55736843)

[Módulo WI-FI (WIFI.h) 69](#_Toc55736844)

[Módulo de Funções Auxiliares (Helper.ino) 71](#_Toc55736845)

[ANEXO 2 – Programa da Interface Node-RED 72](#_Toc55736846)

[Tela do Abastecimento 72](#_Toc55736847)

[Tela do Reservatório 77](#_Toc55736848)

[ANEXO 3 – Material na Internet 80](#_Toc55736849)

1. INTRODUÇÃO

Em nossa sociedade, a criação de animais está presente em uma parcela considerável da população, seja de forma comercial ou afetiva. Essas atividades exigem muito cuidado e atenção, porém existem aquelas que são repetitivas e acabam por ocupar períodos de tempo que poderiam serem empregados em outros tipos de atividades, benéficas às pessoas e aos animais.

Pensando justamente nas necessidades das pessoas e dos animais, idealizamos este projeto com o objetivo de disponibilizar mais tempo às pessoas através da automatização de tarefas, porém, sem qualquer perda na qualidade dos cuidados com os animais.

Através de pesquisas e experimentos, compilamos diversos tecnologias e técnicas que nos auxiliaram no desenvolvimento do projeto. Como resultado, o dispositivo projetado atende a proposta de controlar a alimentação animal de forma remota e programada, tendo indicadores que informam a situação atual do abastecimento e possuindo um sistema de alimentação que é resistente a falhas de alimentação elétrica.

1. DESENVOLVIMENTO

O desenvolvimento do projeto contou duas etapas. A fundamentação teórica, onde foram levantadas informações e técnicas sobre conceitos e tecnologias necessárias à implementação do dispositivo proposto. A etapa que se segue é a de projeto, onde a subdividimos em duas, Hardware e Software, com a finalidade de dividir o tempo de desenvolvimento de forma mais eficiente.

* 1. Fundamentação Teórica

A seguir são expostos trechos sintetizados das pesquisas realizadas, contendo conhecimentos que foram essenciais para o desenvolvimento do projeto.

* + 1. IOT

IoT (*Internet of Things*), ou Internet das Coisas em português, é um conceito que envolve utilização de uma rede que conecta pequenos computadores, dispositivos, sensores e atuadores com o intuito de compartilhar e manipular dados para a criação de processos automatizados e monitoramento de atividades.

Apesar de tecnologias que compõe o IoT já existirem há décadas, a grande expansão recente do IoT se deve a alguns fatores:

* Popularização de sensores de baixo custo e baixa potência;
* Desenvolvimento da internet com protocolos desenvolvidos para comunicação de pequenos dispositivos;
* Armazenamento de dados na nuvem e processamento/análise de dados avançados através de *Machine Learning*; e
* Avanço de tecnologias baseada em processamento de linguagem natural.
  + - 1. IoT Industrial

Aplicado a indústria, o IoT assume o papel de dar suporte à instrumentação, ao controle de sensores com interação com tecnologias da nuvem e comunicação máquina a máquina. As seguintes áreas são as principais aplicações do IoT Industrial:

* Cidades inteligentes;
* Manufatura Inteligente;
* Cadeia de suprimentos digitais inteligentes;
* Manutenção preventiva e preditiva;
* Logística conectada e inteligente; e
* Redes elétricas inteligentes.
  + 1. WI-FI

Wi-Fi é uma tecnologia de comunicação de rede sem fio baseada em transmissões de radiofrequência na faixa entre 2.4Ghz e 5Ghz.

Os padrões da família IEEE 802.11 definem os protocolos que envolvem a conexão entre computadores, dispositivos móveis e outros equipamentos. Cada padrão é estabelecido e ratificado, sucedendo o padrão vigente e estabelecendo uma versão com mais funcionalidades.

* + 1. MQTT

Este protocolo foi criado pela empresa IBM, no final dos anos 90, como o objetivo vincular suas estruturas de sensores em oleodutos prolíferos à rede de satélites. Como o desenvolvimento do IoT, o protocolo MQTT se tornou o protocolo mais utilizado na comunicação entre pequenos dispositivos.

* + - 1. Características

O MQTT possui a seguintes características que o qualificam como uma excelente opção de comunicação entre sistemas embarcados:

* Protocolo leve: Este protocolo foi desenvolvido para suportar bom desempenho hardware com capacidades limitadas, larguras de banda pequenas e transmissões de alta latência;
* Flexibilidade: O modo de transmissão e a organização de tópicos confere uma alta maleabilidade sobre qual a informação será trafegada o como ela o será;
* Suporte à Segurança: Apesar de leve e flexível, o MQTT ainda é capaz de fornecer funcionalidades de segurança como autenticação, restrição a conteúdos e suporte a conexões seguras (SSL/TSL).
  + - 1. Funcionamento do MQTT

O MQTT funcionará através da rede sem fio, sendo executado em cima do protocolo TCP/IP. Em sua organização, este protocolo possui duas entidades (broker e cliente), dois papéis (assinante e publicador) e uma estrutura de organização de mensagens (tópicos).

O **broker**, é um centralizador e redistribuidor de mensagens, mantendo registro dos destinatários das mensagens e redirecionando as mensagens tão logo elas são recebidas. Apenas um dispositivo pode ser o broker de uma conexão MQTT.

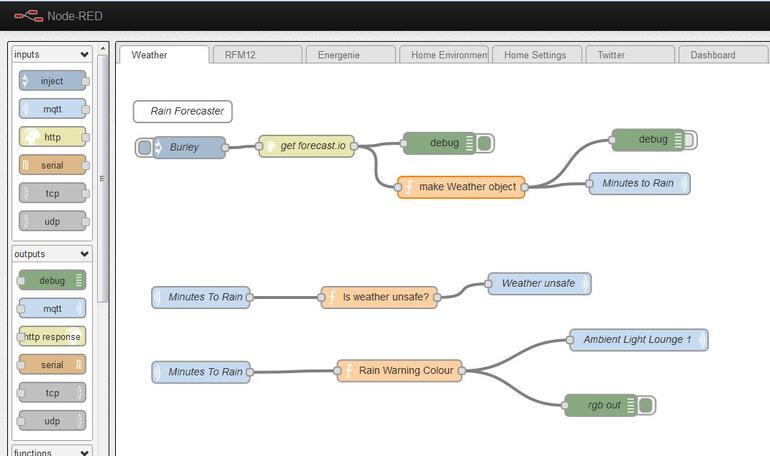
O **cliente** é qualquer dispositivo que mande ou receba mensagens através do broker. Quando está enviando, o cliente está desempenhando o papel de **publicador**, onde ele enviar uma mensagem ao broker para ser aplicada em um determinado tópico de mensagens. Ao receber mensagens, o cliente está no papel do **assinante**, onde ele define, geralmente no momento do estabelecimento de sua conexão MQTT, os tópicos de mensagens aos quais as publicações devem ser redirecionadas a ele. Múltiplos dispositivos podem ser clientes, publicadores e assinantes em uma conexão MQTT, além disso, o cliente pode desempenhar os papeis de Publisher e Subscriber simultaneamente.

Os **tópicos** são uma estrutura de armazenamento de mensagens primariamente temporário, para reter a mensagem apenas durante o período entre o broker receber a mensagem do publicador e o momento em que terminar a entrega para o último assinante de seu tópico.

* + 1. Node-RED

Node-RED é uma ferramenta de desenvolvimento de software criada pela IBM, sua operação é dada através da programação gráfica e organizada em nós e fluxos de dados. Esta tecnologia foi construída em cima de javascript e NodeJS, sendo possível criar trechos de programação textual em javascript dentro de alguns nós e gerenciar pacotes de bibliotecas através do NodeJS.

Figura 1. Exemplo de programa Node-RED, uma aplicação de alerta climático.



Fonte: TechRepublic, 2014.

Um dos grandes poderes do Node-RED é a possibilidade de adicionar novas bibliotecas de nós, e se integrar ao mais diversos tipos de dispositivos ou serviços da internet, sendo uma ferramenta muito poderosa em aplicações IoT que dependem de recursos da nuvem.

* + 1. I2C

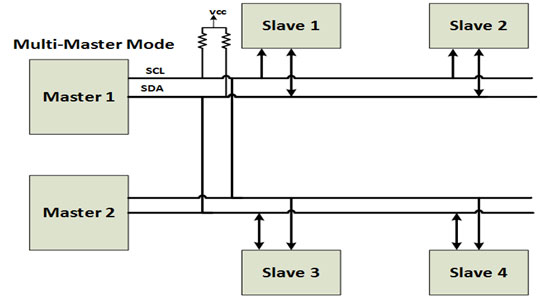
O I2C (Inter-integrated circuit) é um protocolo de comunicação serial que é utilizado para simplificar a transmissão de dados entre um dispositivo principal e seus periféricos.

A disposição de um circuito I2C é dada através de um barramento duplo que possui as vias SCL (Serial Clock Line) e SDA (Serial Data Line). Os dispositivos podem ser inseridos no circuito com o papel de mestre ou escravo.

O mestre em um circuito I2C não recebe um endereçamento no barramento e, além disso, é responsável pelo início de cada comunicação, dirigindo requisições de mensagens, assim como, definindo o clock para estas.

O escravo tem o papel de receber e mandar mensagens aos mestres. Cada escravo deve ser endereçado de forma única no barramento, dessa forma será identificado nas requisições no mestre.

Figura 2. Barramento I2C.



Fonte: Engineers Garage, 2020.

A taxa de transmissão do protocolo varia conforme o modo de operação implementado no circuito I2C em questão. Os modos de operação são os seguintes:

Tabela 1. Modos de Operação I2C

|  |  |
| --- | --- |
| **Modo de Operação** | **Taxa de Transmissão** |
| Standard Mode | 100 Kbit/s |
| Fast Mode | 400 Kbit/s |
| Fast Mode Plus | 1 Mbit/s |
| High-Speed Mode | 3,4 Mbit/s |

Fonte: Engineers Garage, 2020.

Os modos de operação mais utilizados são Standard Mode e Fast Mode.

* 1. Projeto

O projeto conta com uma divisão em hardware e software, esta divisão ocorre para separar as competências de cada trecho de forma a manter a organização ao mesmo tempo em que facilita o desenvolvimento focado de cada tema.

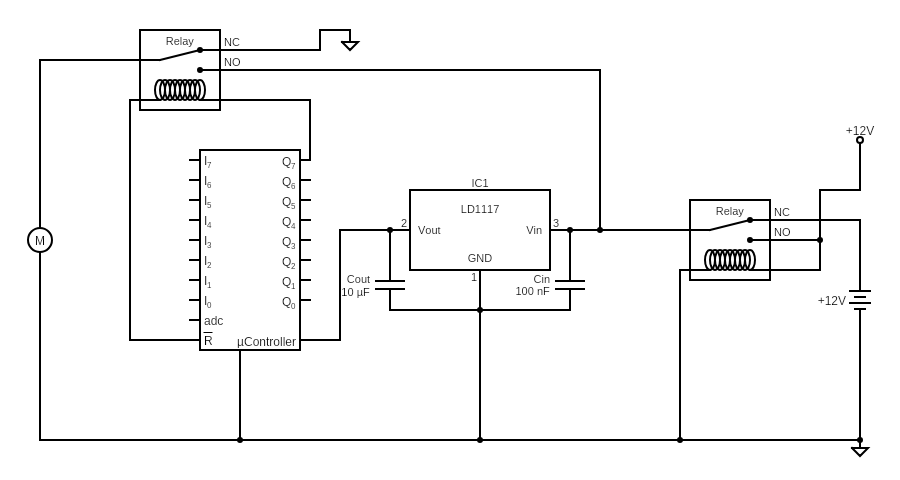
* + 1. Hardware

Esta seção do projeto se dedica a descrever e definição os componentes físicos utilizados no dispositivo, expondo suas características e informando as interligações utilizadas entre partes.

* + - 1. Circuito de Alimentação

O projeto tem como fator de grande importância o seu circuito de alimentação, pois o funcionamento do equipamento na proposta de suprir as necessidades alimentares de um animal não pode ser cerceado por um mal funcionamento momentâneo na rede elétrica. Para este fim foi desenvolvido um circuito de alimentação que mantém o dispositivo funcionando por intermédio de uma bateria quando há falta de energia elétrica na rede de distribuição.

Figura 3 - Circuito de Alimentação



Fonte: Dos próprios autores, 2020.

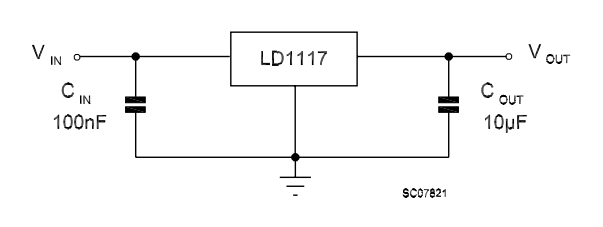
Além disso, o circuito de alimentação (Figura 3) é responsável por regular a tensão do circuito (12V), para aquela que supre as necessidades do microcontrolador (3,3V).

* + - * 1. Alimentação Chaveada
* Fonte 12V/1,5A;
* Relé 12V; e
* Bateria.

A fonte é utilizada para alimentação geral do circuito, enquanto a bateria será acionada em caso de uma queda de energia para que o equipamento não fique inutilizável e o animal fique impossibilitado de alimentar-se. Para que isso aconteça, ambos conectados à um relé normalmente fechado com a fonte,

* + - * 1. Circuito Regulador de Tensão

Figura 4 Circuito Regulador de Tensão para 3.3V.



Fonte: STMicroelectronics, 2005.

* Regulador de Tensão 3.3V LD1117;
* Capacitor cerâmico 10µF; e
* Capacitor cerâmico 100nF.

Após o relé da fonte, em paralelo com relé do alimentador, se encontra o circuito regulador de tensão (Figura 4). Composto pelo LD1117 juntamente com dois capacitores (100nF e 10µF), ele realiza a conversão da tensão de alimentação do circuito de entrada para alimentar o microcontrolador com a tensão de 3,3V.

* + - 1. Abastecimento

Para poder alimentar o PET, deve-se transportar a ração do armazenador até a tigela. Com isso em mente, utilizamos uma rosca helicoidal que transportará o alimento ao girar em seu próprio eixo. O movimento da rosca é proporcionado por um motor DC que, por sua vez, é acionado por um relé, conectado diretamente ao microcontrolador.

* + - * 1. Motor de Abastecimento

Figura 5. Motor DC GA12-N20-200.



Fonte: Handson Technology, 2019.

O motor escolhido é o Micromotor DC N20 com Caixa de Redução 12V 200 RPM, modelo GA12-N20-200. Este motor é equipado com caixa de redução, com taxa de Redução: 1:10, composta por engrenagens de metal que aumentam o torque e mantém sua rotação estável. Tem uma faixa de operação entre 6V a 12V DC e corrente de 0.07A, tem torque máximo de 2 kgf.cm e, quando carregado, apresenta uma rotação máxima de 80 RPM

* + - * 1. Relé de Acionamento do Motor

Para o acionamento do motor foi empregado um módulo relé (Figura 6) com bobina de 5V DC e 75mA, e com capacidade de até 12A em 125V AC ou 7A em 250VAC. Em seu contato normalmente aberto será conectado o positivo da bateria 12V e o terminal positivo de alimentação de motor, de modo que, quando o ESP32 enviar o sinal de ativação, através do GPIO 32, a bobina será energizada, o contato se fechará e o motor será acionado.

Figura 6. Módulo Relé 5V 1 Canal

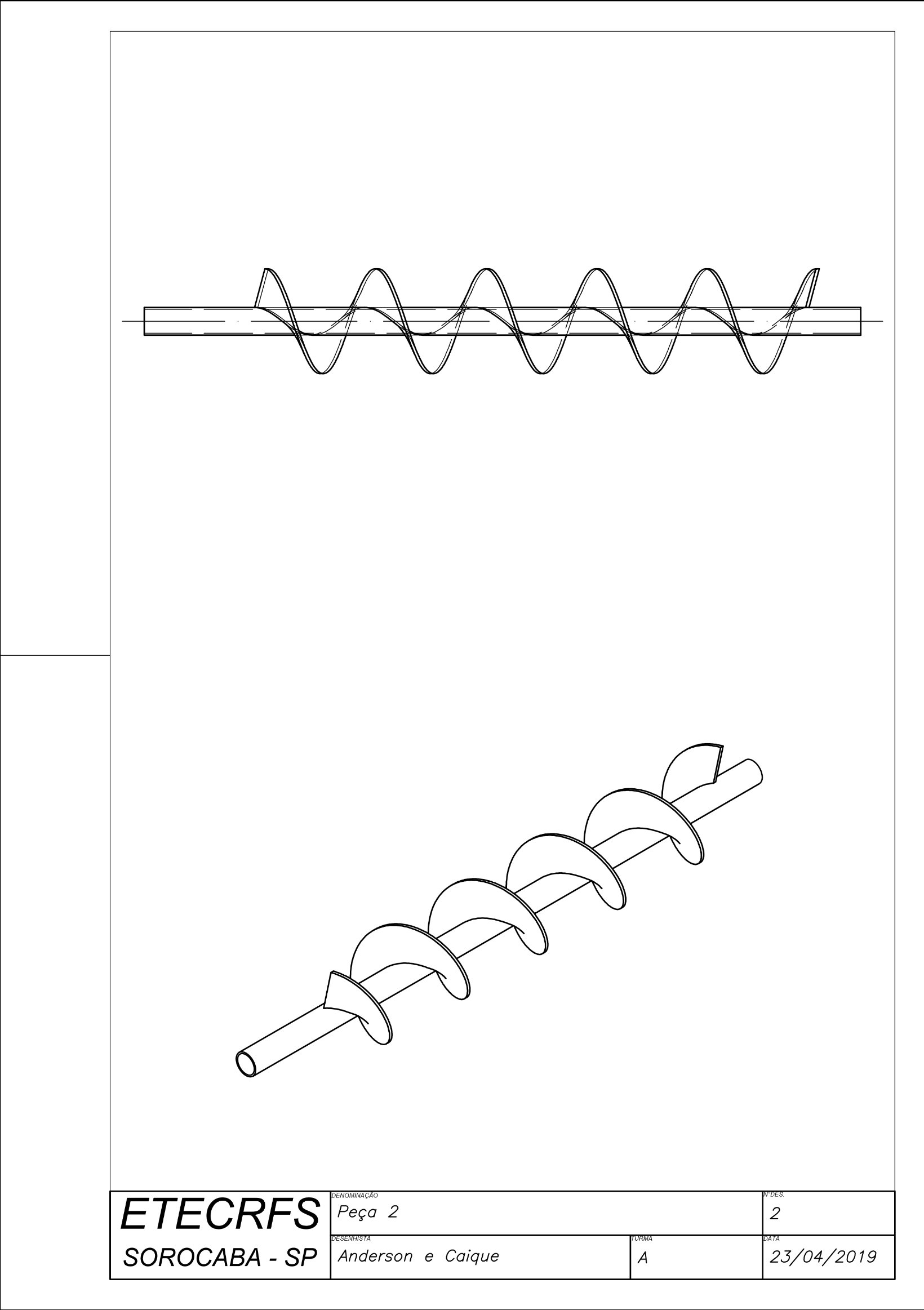
Uma imagem contendo eletrônico, circuito

Descrição gerada automaticamente

Fonte: OLIVEIRA Euler, 2018.

* + - * 1. Rosca helicoidal

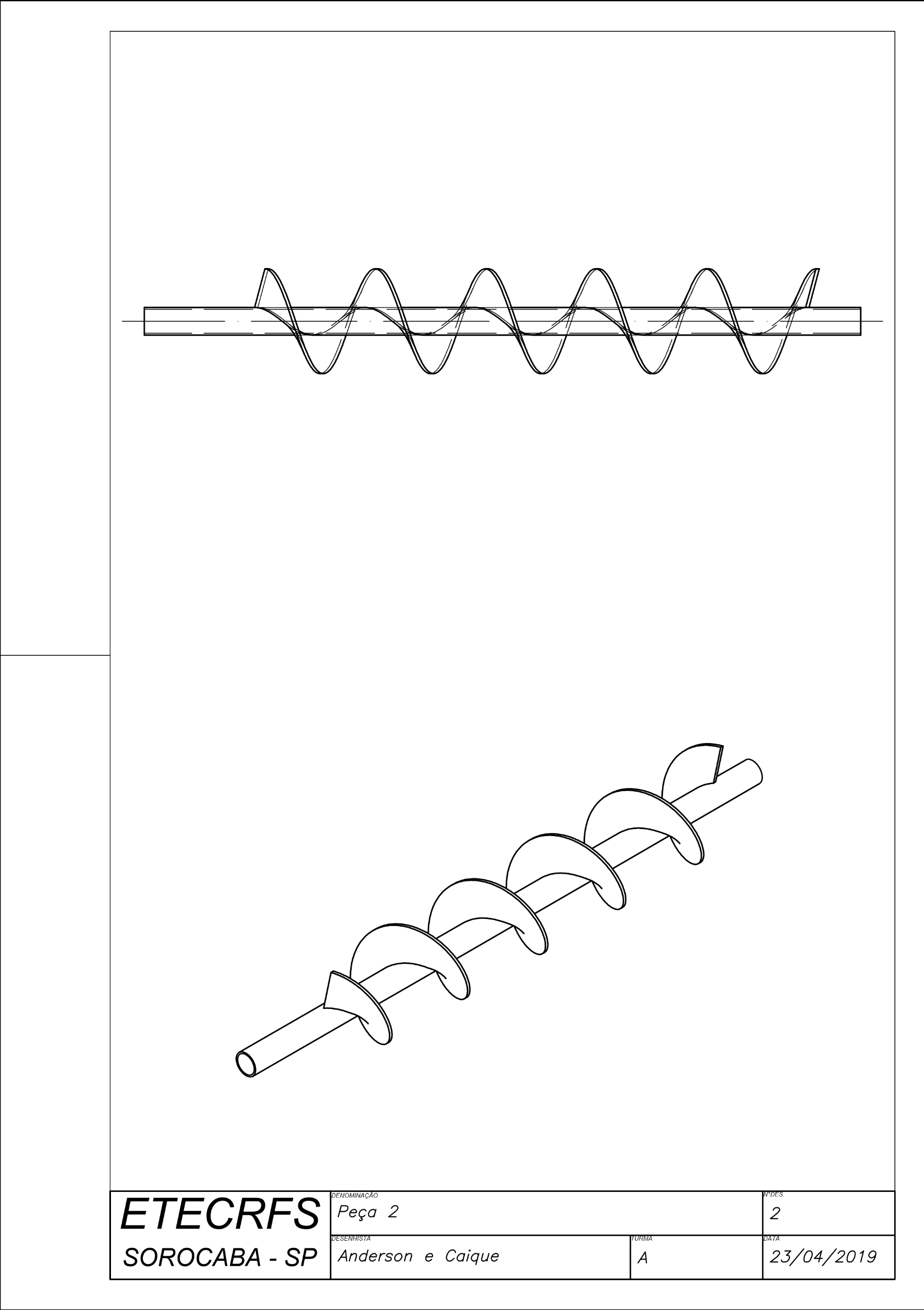
Figura 7 - Vista frontal da rosca transportadora



Fonte: Dos próprios autores, 2020.

A rosca helicoidal (Figura 7 e Figura 8) é um mecanismo mecânico que transforma movimento rotacional em movimento linear. O movimento de rotação do helicoide faz com que um ponto de material, depositado no fundo da carcaça, seja arrastado para frente. Após ser deslocado, o movimento angular fará arrastar linearmente de onde se encontrava inicialmente até o próximo ponto. Se continuar rotacionando o helicoide, o ponto seguinte do helicoide fará o papel de empurrar linearmente o material e isso se repetirá até chegar a um local que o material não esteja mais em contato com o fundo da Rosca transportadora e com o anel do helicoide, ou seja, no bocal de saída do equipamento.

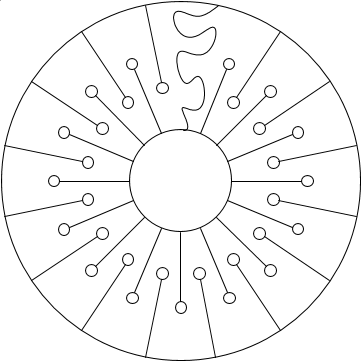
Figura 8 - Vista isométrica da rosca transportadora.



Fonte: Dos próprios autores, 2020.

A rosca necessária para o projeto se estende por 500 mm e é feita a partir de uma placa de MDF com 3 mm de espessura cortada num formato (Figura 9) afim de se tornar flexível e possibilitar que seja fixada num cano de PVC de 25 mm de diâmetro que assume a função de eixo. Para sua realização serão necessárias 5 peças de MDF que se encaixam ao longo do eixo.

Figura 9 - Modelo do corte a laser da rosca transportadora



Fonte: Dos próprios autores, 2020.

* + - 1. Sensores e Atuadores Auxiliares

Além do motor e seu relé de ativação, o projeto conta com sensores e atuadores que auxiliam no modo que são programadas as condições de ativação do motor de abastecimento.

* + - * 1. Sensores Abastecimento

Sensor Capacitivo de Proximidade LJC18A3-H-Z/BY é um dispositivo eletrônico capaz de detectar a aproximação / presença de objetos metálicos (ferro, aço, alumínio e outros) e objetos não metálicos.

Figura 10 - Sensor de Presença e Seus Terminais



Fonte: Smart Projects, 2020.

Configuração e Funcionamento

Os Sensores são responsáveis por detectar o nível de mantimentos contidos no reservatório. Os sensores estão fixados de forma alinhada verticalmente na lateral do reservatório, um dos sensores ficará no meio do reservatório e o outro próximo ao fundo. Eles possuem seu terminais de sinais (Figura 10) ligado aos GPIOs 26 (sensor inferior) e 27 (sensor superior) do ESP32 e tendo seus terminais de alimentação sendo ligados às saídas de 3.3V e GND do mesmo.

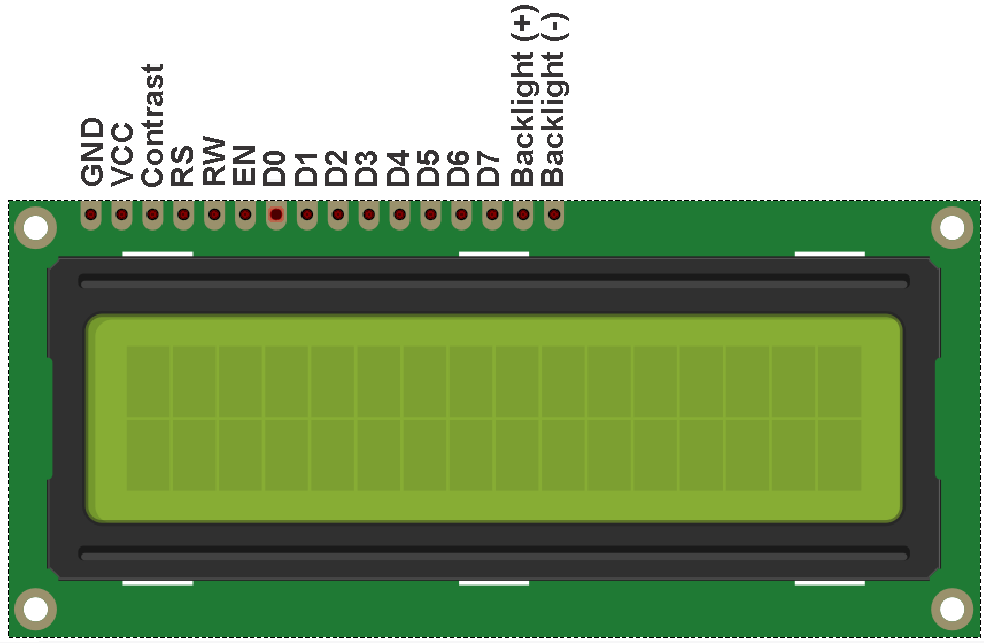
* + - * 1. Display informativo

Circuito do mostrador responsável pela informação de estados do dispositivo.

Display

Display LCD HD44780 verde (Figura 11) exibe informações através de uma matriz de caracteres personalizáveis de 16 colunas por 2 linhas. Possuindo 16 pinos de entradas, sua ligação, por vezes, necessita da utilização de vários terminais, soluções como o I2C são amplamente utilizadas como uma forma de diminuir esta necessidade.

Figura 11 - Display 16x2 e a indicação de seus terminais

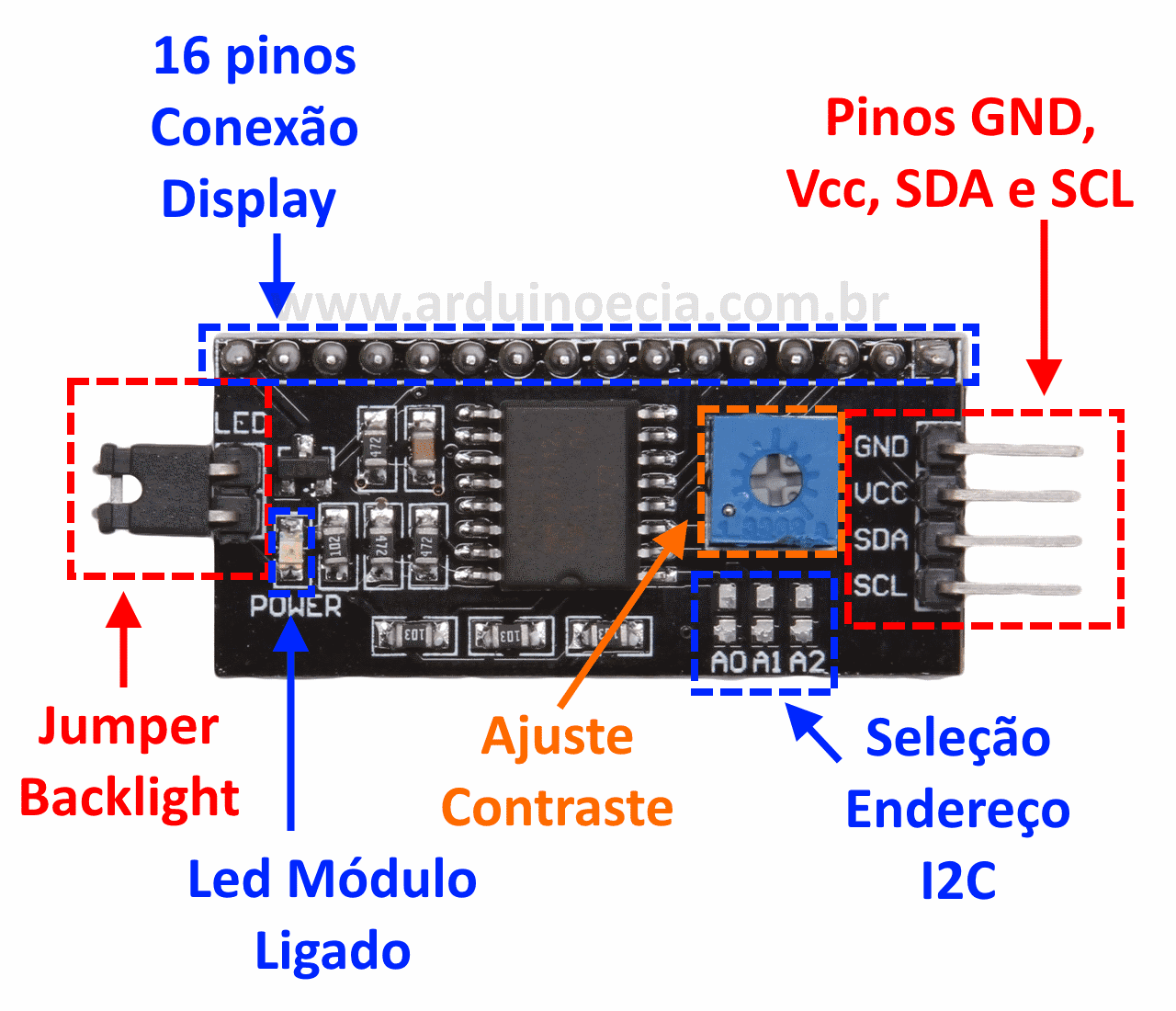


Fonte: rees52.com, 2020.

Módulo I2C

Módulo I2C PcF8574 (Figura 12), na lateral esquerda do módulo temos 4 pinos, sendo que dois são para alimentação (VCC e GND), e os outros dois são da interface I2C (SDA e SCL). O potenciômetro da placa serve para ajuste do contraste do display, e o jumper na lateral oposta permite que a luz de fundo seja controlada pelo programa ou permaneça apagada. A ligação física do módulo I2C com o ESP32 é realizada através dos GPIOs 21 e 22, SDA e SCL respectivamente.

Figura 12 - Descrição do Módulo I2C

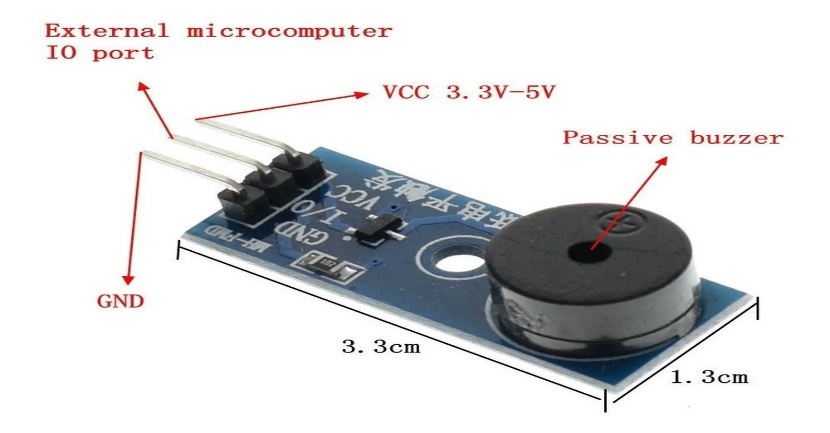


Fonte: arduinoecia.com.br, 2014.

* + - * 1. Buzzer Sinalizador

O Módulo Buzzer 5V Passivo (Figura 13) permite que você tenha mais controle sobre as tonalidades emitidas, seus terminais de alimentação estarão conectados às saídas 3.3v e GND e enviará sinal para o GPIO 25 do ESP32. Será acionado como forma de alerta sonoro para informações importantes do dispositivo.

Figura 13 - Configuração Física de um Módulo Buzzer Passivo.

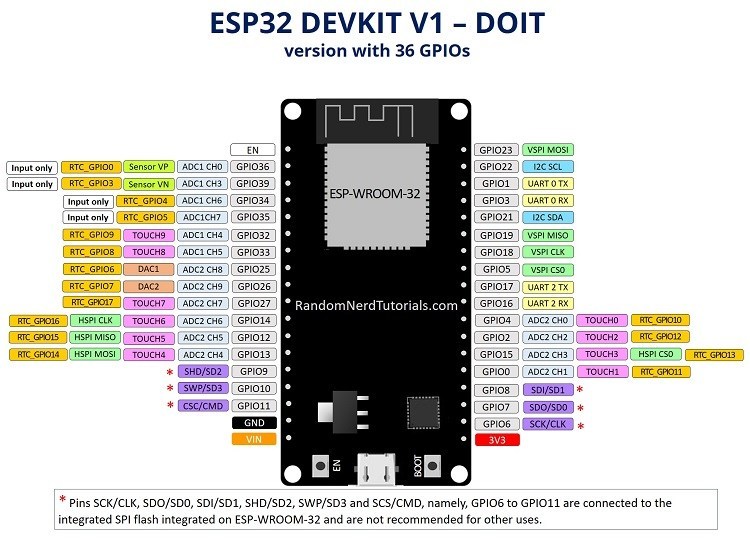


Fonte: mikroelec.com, 2020.

* + - 1. ESP32

O projeto utiliza o microcontrolador ESP32 WROOM na placa de prototipagem ESP32 DevKit V1 (Figura 14). Ele é responsável por todo o processamento de dados do projeto para os processos inclusos na alimentação e seu agendamento, assim como, ele estabelece conexão com uma rede WI-FI com a finalidade de ter a conexão com a interface remota (Node-RED) e a atualização frequente de seu relógio interno.

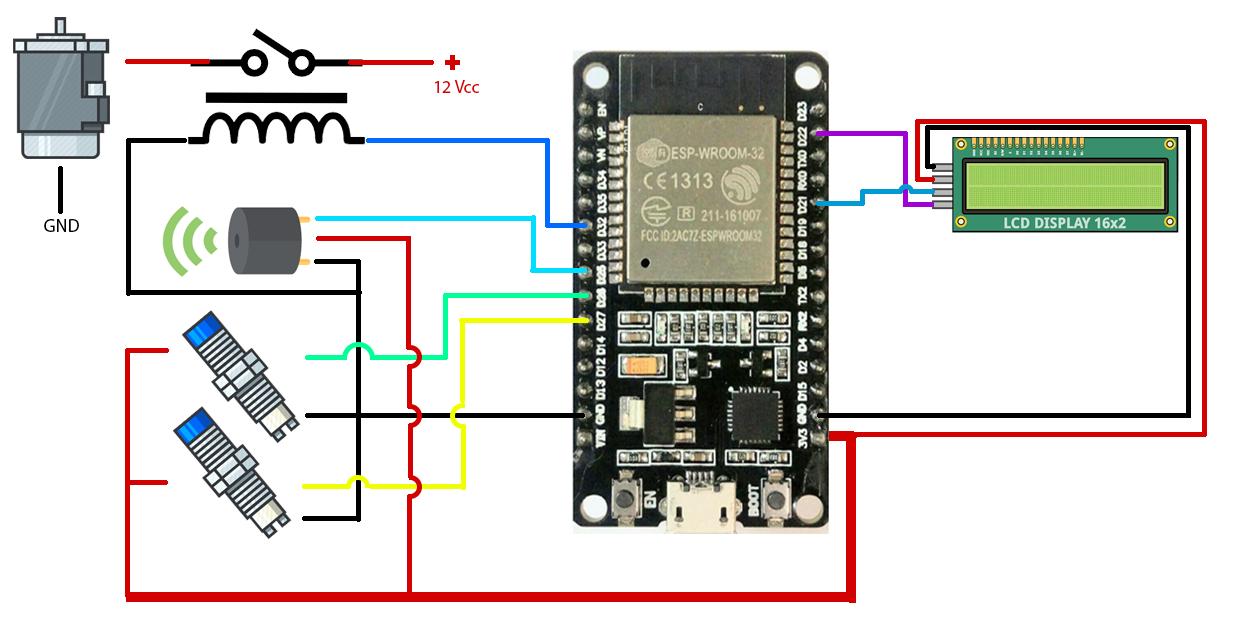
Figura 14 - Placa ESP32 DEVKIT V1



Fonte: Radom Nerd Tutorial, 2018.

O ESP32 é conectado, direta ou indiretamente, a todos os sensores e atuadores do dispositivo (Figura 15), sua programação (Item 2.2.2.1) é responsável por definir como as informações dos sensores irão interagir com o atuadores de forma ao sistema agir como o alimentador animal proposto neste projeto.

Figura 15 - Ligações físicas do ESP32 com os sensores e atuadores.



Fonte: Dos próprios autores, 2020.

* + 1. Software

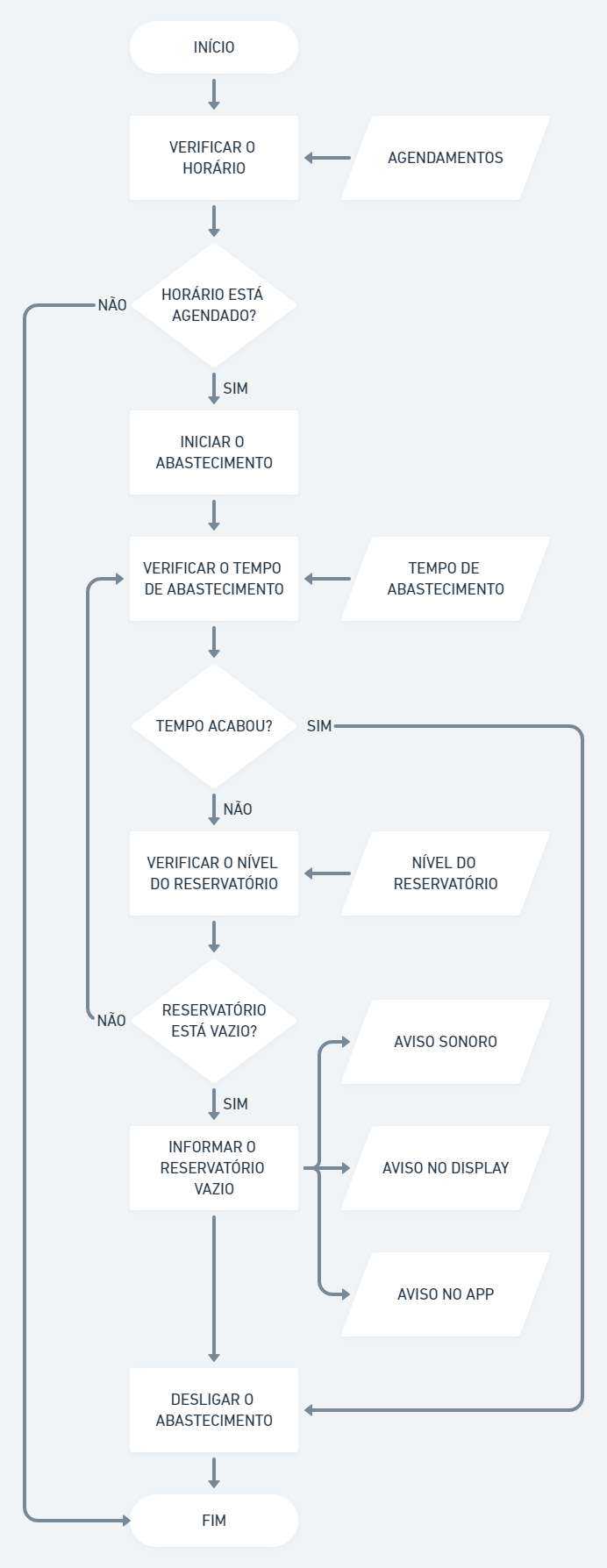
O software de nosso projeto possui várias camadas de aplicação, onde o principal processamento é concentrado no ESP32, implementado através da linguagem C, porém, a comunicação MQTT e a plataforma de Node-RED tornam-se partes essenciais por serem responsáveis pela interação e configuração do sistema.

* + - 1. Linguagem C (ESP32)
         1. Alimentador

Este é o módulo do programa responsável pela ativação do relé de alimentação, o processo principal do projeto. Para executar com sucesso esta função, o módulo de alimentação interage, direta ou indiretamente, com todos os outros módulos do projeto.

Seu processo, como descrito no fluxograma do processo de alimentação (Figura 16), realiza uma verificação nos horários agendados e, caso o horário atual esteja agendado, ele ativa o relé de abastecimento durante o período programado. O processo ainda terá o cuidado de não ativar o relé caso os sensores indiquem que o reservatório está vazio, assim interrompendo a alimentação ou nem mesmo a iniciando e, em vez disso, enviando alertas nos meios disponíveis.

Figura 16 - Fluxograma do Processo de Alimentação



Fonte: Dos próprios autores, 2020.

* + - * 1. WI-FI

Para a utilização de WI-FI no ESP32 utilizamos a biblioteca WiFi.h, ela traz funcionalidades que permitem o manuseio simplificado de uma conexão de rede sem fio.

Na implementação se faz necessária a definição do identificador da rede sem fio (SSID) e da sua respectiva senha de acesso, tendo estas informações um comando de inicialização da conexão WI-FI deve ser executado até a conexão seja estabelecida.

Uma vez tendo a conexão estabelecida, o ESP32 terá que verificar periodicamente o estado da conexão para se informar se a conexão ainda está ativa ou foi perdida e, nesse caso, reiniciar o processo de reconexão.

* + - * 1. MQTT

Como o protocolo principal de comunicação entre os clientes WI-FI, o projeto utilizará o protocolo MQTT.

Utilizamos um broker Eclipse Mosquitto com configurações iniciais instalado em rede local, como um projeto conceitual esta configuração será efetiva para os propósitos desejados. Entretanto, o broker poderia ser utilizado em um serviço hospedado em algum servidor web, adicionando a possibilidade de utilizar as funcionalidades de o projeto de forma remota através da internet.

Na utilização do MQTT pelo ESP32 precisamos da conexão WI-FI, previamente definida, assim como uma biblioteca para fornecer-nos o conjunto de comandos para sua manipulação, nesse caso, utilizaremos a biblioteca PubSubClient.h.

Para estabelecer a conexão MQTT, é necessário o endereço IP (ou web) do computador do broker e a porta configurada para o MQTT. Após realizada a conexão, deve-se configurar nesta o método de comunicação (callback), assim o ESP32 poderá receber as mensagens publicadas nos tópicos aos quais assinar.

Assim como uma conexão WI-FI, os comandos da conexão MQTT devem ser iterados até a conexão seja efetivamente estabelecida e deve ser monitorada periodicamente para tratar possíveis falhas de conexão.

Sempre após realizar a conexão (ou reconexão) o ESP32 deverá assinar os tópicos aos quais deseja receber as mensagens do broker, esses tópicos são:

* “alimentador/horariosAlimentacao”;
* “alimentador/tempoAbastecimento”;
* “alimentador/testeAbastecimento”;
* “alimentador/nivelReservatorio”;
* “alimentador/alertaSonoroAtivo”;
* “alimentador/tipoAlertaSonoro”;

Nos casos de publicação, o ESP32 apenas terá de enviar para o broker uma mensagem contendo o tópico no qual ele deseja publicar e o conteúdo publicado.

* + - * 1. Armazenamento

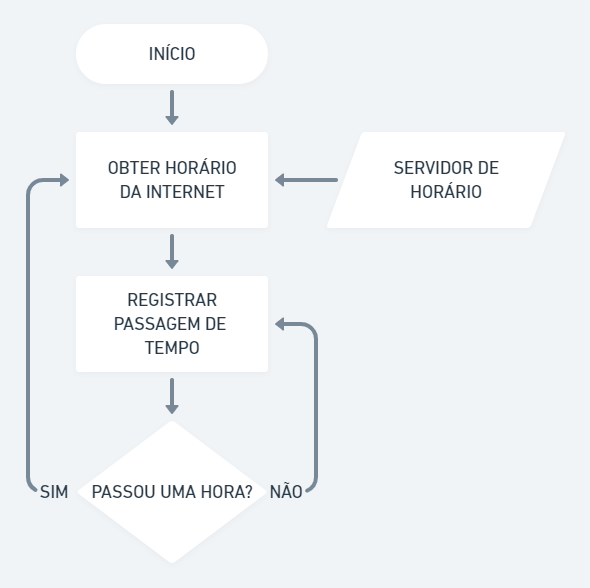
O projeto utiliza de um módulo do programa voltado ao armazenamento não-volátil de informação, este armazenamento ocorre por intermédio da biblioteca Preferences, que possui interface facilitada para a gravação e recuperação de dado dentro da memória flash do ESP32.

* + - * 1. Horário

Ao não possui um hardware RTC adicionado ao nosso projeto, se tornou necessária a adição de um módulo do programa direcionado à manutenção de um relógio interno.

A biblioteca NTPClient foi a base para o desenvolvimento, como a interface principal do projeto depende de uma conexão Wi-Fi, não se torna excessivo o requisito de conexão à internet. Porém, como precaução para que eventuais faltas de internet não atrapalhem o funcionamento do sistema, o processo que mantém o horário apenas busca a hora atual na internet a cada uma hora passada, nos outros momentos o ESP32 calcula a passagem de tempo de forma a atualizar o relógio programado a cada segundo. Processo descrito no fluxograma de manutenção de tempo (Figura 17).

Figura 17 - Fluxograma da Manutenção de Tempo



Fonte: Dos próprios autores, 2020.

Devido a precisão demandada por esta funcionalidade, ela foi separada das outras, sendo processada em modo exclusivo pelo núcleo de processamento zero do ESP32, enquanto os outros processos permanecem sendo executados em no núcleo de processamento um.

* + - * 1. Buzzer

O projeto possui um módulo do programa dedicado à execução de alerta sonoros, estes que serão executados pelo buzzer passivo.

Os alertas são programados para serem executados em momentos nos quais o microcontrolador detecta a falta de algum recurso essencial para o funcionamento pleno do sistema, sendo eles: conteúdo no reservatório e acesso à internet.

O programa envia uma variação de frequências durante intervalos de tempo predeterminados de forma a executar uma melodia que servirá como o alerta para o usuário do sistema voltar suas atenções ao elemento faltante.

A implementação disso no ESP32 utiliza a biblioteca EasyBuzzer, a qual disponibiliza uma série de facilitadores para a execução de notas musicais sequenciais. A principal característica utilizada desta biblioteca é a função singleBeep, esta função permite a execução deu uma frequência durante um tempo determinado e, quando o tempo acabar, esta mesma função chama uma função para executar as próximas notas.

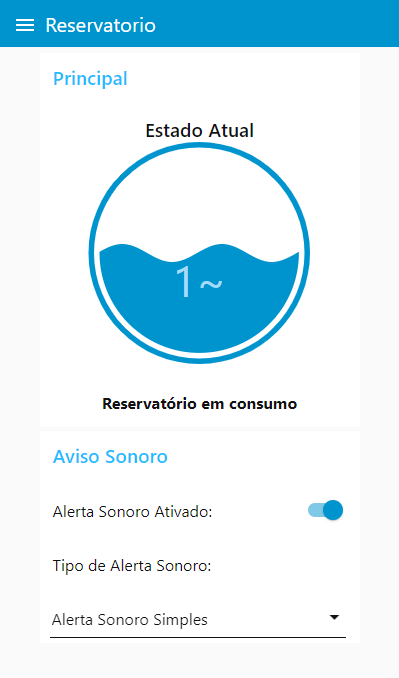
* + - 1. Node-RED

A ferramenta Node-RED está sendo aplicada como a principal responsável pelo gerenciamento de configurações do projeto. Ela implementa interfaces web que, ao interagir como broker mqtt, exibe as configurações atuais do microcontrolador, assim como, disponibiliza novas opções de configurações.

Para a criação da interface web, foi adicionado ao projeto básico do Node-RED os pacotes de ferramentas: “node-red-dashboard” e “node-red-node-ui-list”. Esses pacotes concedem nós para serem utilizados no fluxo e geram uma interface HTML com variadas opções de controles e componentes visuais.

* + - * 1. Interface do Reservatório

Figura 18 - Interface do Reservatório

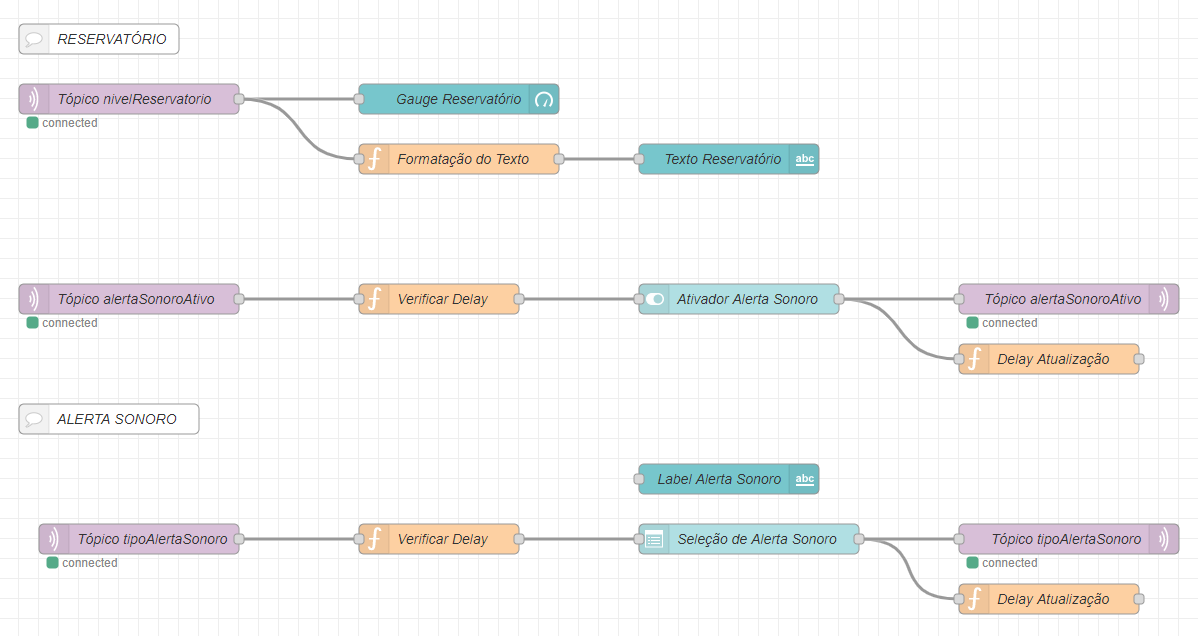


Fonte: Dos próprios autores, 2020.

Nesta tela (Figura 18) temos duas seções, a principal e a avisos sonoros, seu funcionamento é implementado na seção de código gráfico do reservatório (Figura 19).

Na seção principal utilizamos um nó que lê a quantidade de sensores de alimentação ativos e exibe uma representação gráfica do estado do reservatório, de modo a deixar informação intuitiva do ponto do usuário.

Figura 19 - Fluxo da interface do reservatório

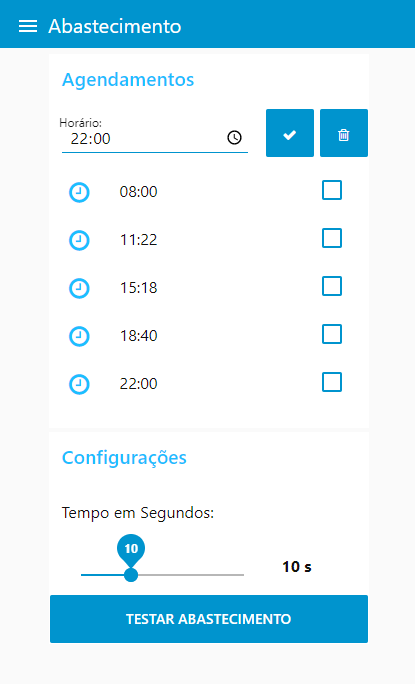


Fonte: Dos próprios autores, 2020.

Na seção aviso sonoro possuímos um nó do tipo switch para definir se o alimentador irá executar avisos sonoros quando o dispositivo não estiver detectando recursos essenciais. Logo abaixo, temos um nó de caixa de seleção, que exibirá as opções disponíveis de alerta sonoro a serem executados pelo alimentador.

* + - * 1. Interface do Abastecimento

Figura 20 - Interface do Abastecimento

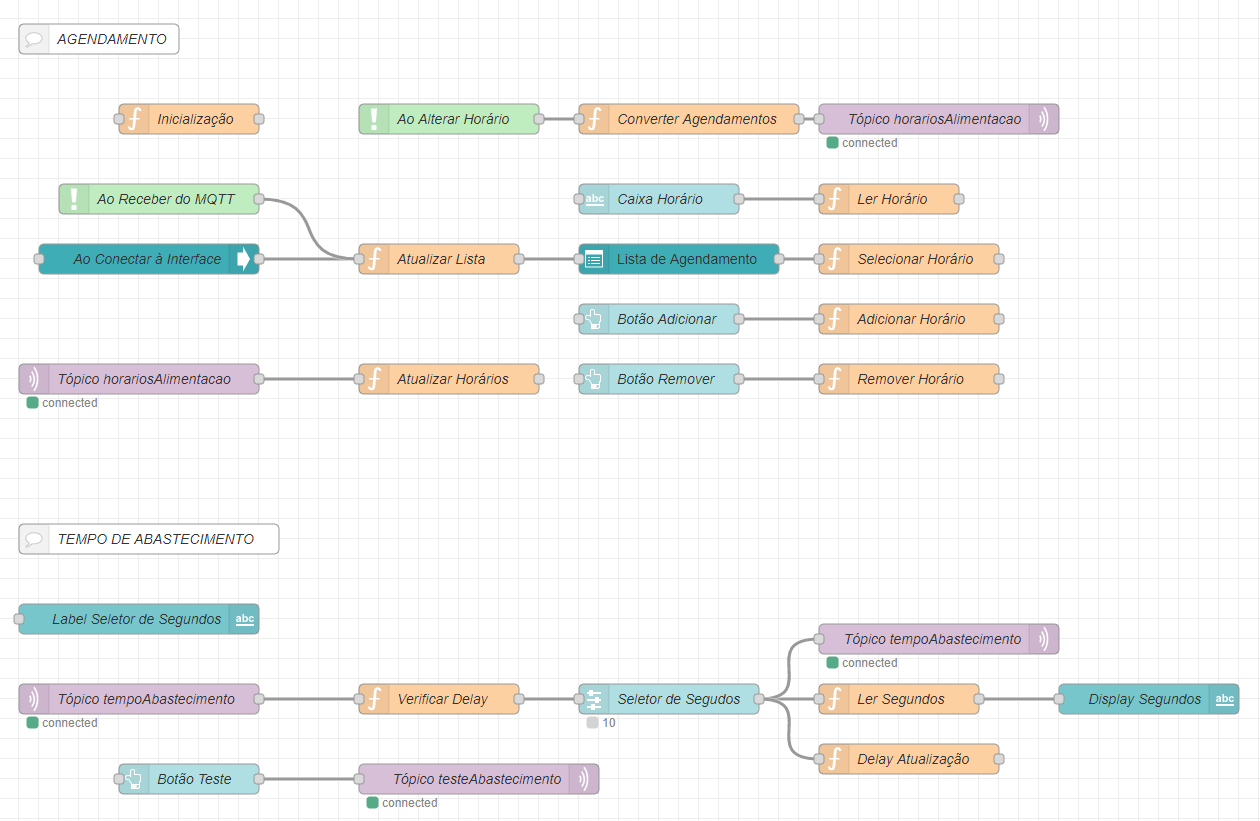


Fonte: Dos próprios autores, 2020.

Nesta tela (Figura 20) tratamos das opções do abastecimento, nela temos duas seções, de agendamentos e de configurações. O funcionamento da tela é configurado através da codificação gráfica (Figura 21) implementada no servidor Node-RED.

A Seção de agendamentos possui uma caixa para a inserção de datas em uma lista, sempre que esta lista é alterada, ela é publicada no tópico “alimentador/horariosAlimentacao” do broker MQTT e, consequentemente, enviada para o ESP32.

Figura 21 - Fluxo da interface do abastecimento



Fonte: Dos próprios autores, 2020.

Na Seção de configurações está a seleção do tempo de abastecimento, o valor selecionado é enviado ao tópico “alimentador/tempoAbastecimento” e ao ESP32, servindo como valor que define o tempo de ativação do relé do motor. Logo abaixo, o botão de teste emite uma mensagem para o tópico “alimentador/testeAbastecimento” que ativa o teste do tempo de abastecimento.

1. CONCLUSÃO

Este trabalho apresentou o alimentador animal automático com controle e interface remota via navegador web. O objetivo principal foi desenvolver um produto que auxiliasse os cuidadores de animais a otimizar seu tempo e, ainda, manter o cuidado do animal de forma satisfatória, além de colocar em prática os conhecimentos obtidos durante as aulas do curso de técnico em eletrônica e o aprendizado de novas tecnologias.

As técnicas e tecnologias aqui encontradas, foram selecionadas a fim de atender as necessidades do projeto. Essa seleção exigiu uma vasta pesquisa e o auxílio dos professores da instituição. A integração das técnicas e tecnologias aconteceu de forma positiva, levado a acreditar que tal modelo atenderá as necessidades impostas na concepção dele.

Durante o processo de criação deste projeto, foi encontrada uma limitação. Devido aos eventos ocasionados neste ano de 2020, principalmente a pandemia do vírus SARS-COV2, não foi possível o desenvolvimento de um protótipo físico para o projeto, limitando-se apenas a experimentos e testes de conceitos em ambiente virtual.

REFERÊNCIAS

BERTOLETI, Pedro. ESP32 e MQTT DASH: controle e monitoramento através de um dashboard MQTT para Android. **Filipeflop**, 2019. Disponível em: https://www.filipeflop.com/blog/esp32-e-mqtt-dashboard-android/. Acesso em: 05 out. 2020.

BHATT, Ashutosh. Understanding the I2C Protocol. **Engineers Garage**, 2020. Disponível em: https://www.engineersgarage.com/tutorials/understanding-the-i2c-protocol/. Acesso em: 08 out. 2020.

CORBARI, **Corbari**, 2018. Disponível em: http://www.corbari.com.br/rosca-transportadora. Acesso em: 12 out. 2020.

CANOVA, Henrique. Sensores Capacitivos. **Mecaweb Education SITE**, 2014. disponível em: <http://www.mecaweb.com.br/eletronica/content/e_sensor_capacitivo>. Acesso em: 12 out. 2020.

ESP32 Series Datasheet. **Espressif**, 2020. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32\_datasheet\_en.pdf. Acesso em: 05 out. 2020.

GA12-N20 Geared Mini DC Motor Datasheet. **Handson Technology**, 2019. Disponível em: https://www.handsontec.com/dataspecs/motor\_fan/GA12-N20.pdf. Acesso em: 12 out. 2020.

HEATH, Nick. **TechRepublic**, 2014. Disponível em: https://www.techrepublic.com/article/node-red/. Acesso em: 08 out. 2020.

LIMA, Izabelle. Aprenda a Reproduzir Músicas com o Módulo Buzzer. **Auto Core Robótica blog**, 2018. Disponível em: https://autocorerobotica.blog.br/aprenda-a-reproduzir-musicas-com-o-modulo-buzzer-2/. Acesso em: 05 out. 2020.

MOREIRA, Diego. Módulo I2C para LCD 16x2 e Arduino. **Auto Core Robótica blog**, 2019. Disponível em: <https://autocorerobotica.blog.br/modulo-i2c-para-lcd-16x2-e-arduino/>. Acesso em: 12 out. 2020.

O'LEARY Nick. ARDUINO Client for MQTT. **Knolleary**, 2020. Disponível em: https://pubsubclient.knolleary.net. Acesso em: 05 out. 2020.

OLIVEIRA, Euler. Como usar com Arduino – Módulo Relé 5V 1 Canal. **Blog Masterwalker Shop**, 2018. Disponível em: https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-modulo-rele-5v-1-canal/. Acesso em: 12 out. 2020.

ORACLE, **ORACLE**, 2020. Disponível em: https://www.oracle.com/br/internet-of-things/what-is-iot.html. Acesso em: 07 out. 2020.

RANDOM NERD TUTORIALS. **RANDOM NERD TUTORIALS**, 2018. Disponível em: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/. Acesso em: 07 set. 2020.

YUAN, Michael. Conhecendo o MQTT. **IBM**, 2017. Disponível em: https://developer.ibm.com/br/articles/iot-mqtt-why-good-for-iot/. Acesso em: 05 out. 2020.

WIFI library. **Arduino**, 2019. Disponível em: https://www.arduino.cc/en/Reference/WiFi. Acesso em: 05 out. 2020.

LD1117 SERIES Datasheet. **STMicroelectronics**, 2005. Disponível em: https://www.sparkfun.com/datasheets/Components/LD1117V33.pdf. Acesso em: 17 out. 2020.

ANEXO 1 – Programa em linguagem C (ESP32)

Header do Projeto (AlimentadorESP32.h)

#define CONTEXTO "ALIMENTADOR\_ANIMAL"

#define PINO\_MOTOR 32

#define PINO\_RESERVATORIO\_SUPERIOR 27

#define PINO\_RESERVATORIO\_INFERIOR 26

#define PINO\_BUZZER 25

#define WIFI 0

#define TIME 1

#define ALIMENTADOR 2

#define MQTT 3

#define AGENDAMENTOS "alimentador/horariosAlimentacao"

#define ALARME\_SELECIONADO "alimentador/tipoAlertaSonoro"

#define TEMPO\_ABASTECIMENTO "alimentador/tempoAbastecimento"

#define TESTE\_ABASTECIMENTO "alimentador/testeAbastecimento"

#define NIVEL\_RESERVATORIO "alimentador/nivelReservatorio"

typedef enum NivelAbastecimento {

VAZIO,

ABASTECIDO,

CHEIO

} nivel\_abastecimento\_t;

Módulo Principal (AlimentadorESP32.ino)

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <EasyBuzzer.h>

#include <WiFi.h>

#include <PubSubClient.h>

#include <NTPClient.h>

#include <Preferences.h>

#include "AlimentadorESP32.h"

#include "Buzzer.h"

TaskHandle\_t tarefaNucleoSecundario;

void setup() {

iniciarPinosAlimentador();

iniciarDisplay();

iniciarBuzzer();

iniciarWifi();

inciarMQTT();

iniciarNTPClient();

xTaskCreatePinnedToCore(

execucaoNucleoSecundario, /\* Function to implement the task \*/

"Tarefas", /\* Name of the task \*/

10000, /\* Stack size in words \*/

NULL, /\* Task input parameter \*/

0, /\* Priority of the task \*/

&tarefaNucleoSecundario, /\* Task handle. \*/

0); /\* Core where the task should run \*/

}

void loop() {

// VERIFICA A CONEXÃO WIFI E MQTT

verificarWifi();

verificarMQTT();

// VERIFICICAR OS AGENDAMENTOS DE ALIMENTAÇÃO

atualizarAlimentador();

}

void execucaoNucleoSecundario(void \* parameter) {

// LOOP DO NÚCLEO SEGUNDÁRIO

for (;;) {

// ATUALIZAÇÃO DO RELÓGIO

atualizarHorario();

}

}

Módulo do Alimentador (Alimentador.ino)

#define COM\_RACAO 1

#define SEM\_RACAO 0

short tempoAbastecimento = 0;

String agendamentos;

nivel\_abastecimento\_t nivelAtual = VAZIO;

void iniciarPinosAlimentador() {

pinMode(PINO\_MOTOR, OUTPUT);

pinMode(PINO\_RESERVATORIO\_SUPERIOR, INPUT);

pinMode(PINO\_RESERVATORIO\_INFERIOR, INPUT);

desligarMotor();

}

void atualizarAlimentador() {

if (verificarAgendamento() == true) {

while(temporizar(tempoAbastecimento \* 1000, ALIMENTADOR)){

alimentacao();

if (nivelAtual == VAZIO) { break; }

}

desligarMotor();

delay(60000);

}

}

bool verificarAgendamento() {

return false;

}

void atribuirTempoAbastecimento(short tempoNovo) {

tempoAbastecimento = tempoNovo;

}

void atribuirAgendamentos(String agendamentosNovos) {

agendamentos = agendamentosNovos;

}

void testarAbastecimento() {

ligarMotor();

delay(tempoAbastecimento \* 1000);

desligarMotor();

}

void alimentacao() {

nivel\_abastecimento\_t nivelCalculado = calcularReservatorio();

if (nivelCalculado != nivelAtual) {

publicarTopico(NIVEL\_RESERVATORIO, String(nivelCalculado).c\_str());

nivelAtual = nivelCalculado;

}

if (nivelCalculado > VAZIO) {

ligarMotor();

} else {

desligarMotor();

notificarFaltaRacao();

}

}

void notificarFaltaRacao() {

limparDisplay();

escreverDisplay("RESERVATÓRIO", 0, 2);

escreverDisplay("VAZIO!", 1, 5);

tocarAlerta();

}

nivel\_abastecimento\_t calcularReservatorio() {

bool sensorSuperior = digitalRead(PINO\_RESERVATORIO\_SUPERIOR);

bool sensorInferior = digitalRead(PINO\_RESERVATORIO\_INFERIOR);

nivel\_abastecimento\_t valorReservatorio = VAZIO;

if (sensorSuperior == COM\_RACAO && sensorInferior == COM\_RACAO) {

valorReservatorio = CHEIO;

} else if (sensorSuperior == SEM\_RACAO && sensorInferior == COM\_RACAO) {

valorReservatorio = ABASTECIDO;

} else if (sensorSuperior == SEM\_RACAO && sensorInferior == SEM\_RACAO) {

valorReservatorio = VAZIO;

}

return valorReservatorio;

}

void desligarMotor() {

digitalWrite(PINO\_MOTOR, LOW);

}

void ligarMotor() {

digitalWrite(PINO\_MOTOR, HIGH);

}

Módulo do Armazenamento (Armazenamento.ino)

#define TEXTO\_VAZIO ""

#define NUMERO\_INVALIDO -1

Preferences armazenamento;

void iniciarArmazenamento() {

if (verificarArmazenamentoTexto(AGENDAMENTOS)) {

String agendamentosGravados = obterTexto(AGENDAMENTOS);

atribuirAgendamentos(agendamentosGravados);

publicarTopico(AGENDAMENTOS, agendamentosGravados.c\_str());

}

if (verificarArmazenamentoNumero(ALARME\_SELECIONADO)) {

short alarmeSelecionadoGravado = obterNumero(ALARME\_SELECIONADO);

selecionarAlerta(alarmeSelecionadoGravado);

publicarTopico(ALARME\_SELECIONADO, String(alarmeSelecionadoGravado).c\_str());

}

if (verificarArmazenamentoNumero(TEMPO\_ABASTECIMENTO)) {

short tempoAbastecimentoGravado = obterNumero(TEMPO\_ABASTECIMENTO);

atribuirTempoAbastecimento(tempoAbastecimentoGravado);

publicarTopico(TEMPO\_ABASTECIMENTO, String(tempoAbastecimentoGravado).c\_str());

}

}

bool verificarArmazenamentoTexto(char\* chave) {

bool resultado;

armazenamento.begin(CONTEXTO, true);

resultado = !armazenamento.getString(chave, TEXTO\_VAZIO).equals(TEXTO\_VAZIO);

armazenamento.end();

return resultado;

}

bool verificarArmazenamentoNumero(char\* chave) {

bool resultado;

armazenamento.begin(CONTEXTO, true);

resultado = armazenamento.getShort(chave, NUMERO\_INVALIDO) != NUMERO\_INVALIDO;

armazenamento.end();

return resultado;

}

void armazenarTexto(char\* chave, String texto) {

armazenamento.begin(CONTEXTO);

armazenamento.putString(chave, texto);

armazenamento.end();

}

void armazenarNumero(char\* chave, int16\_t numero) {

armazenamento.begin(CONTEXTO);

armazenamento.putShort(chave, numero);

armazenamento.end();

}

String obterTexto(char\* chave) {

String resultado;

armazenamento.begin(CONTEXTO, true);

resultado = armazenamento.getString(chave);

armazenamento.end();

return resultado;

}

short obterNumero(char\* chave) {

short resultado;

armazenamento.begin(CONTEXTO, true);

resultado = armazenamento.getShort(chave);

armazenamento.end();

return resultado;

}

Header do Módulo Buzzer (Buzzer.h)

#define NOTE\_B0 31

#define NOTE\_C1 33

#define NOTE\_CS1 35

#define NOTE\_D1 37

#define NOTE\_DS1 39

#define NOTE\_E1 41

#define NOTE\_F1 44

#define NOTE\_FS1 46

#define NOTE\_G1 49

#define NOTE\_GS1 52

#define NOTE\_A1 55

#define NOTE\_AS1 58

#define NOTE\_B1 62

#define NOTE\_C2 65

#define NOTE\_CS2 69

#define NOTE\_D2 73

#define NOTE\_DS2 78

#define NOTE\_E2 82

#define NOTE\_F2 87

#define NOTE\_FS2 93

#define NOTE\_G2 98

#define NOTE\_GS2 104

#define NOTE\_A2 110

#define NOTE\_AS2 117

#define NOTE\_B2 123

#define NOTE\_C3 131

#define NOTE\_CS3 139

#define NOTE\_D3 147

#define NOTE\_DS3 156

#define NOTE\_E3 165

#define NOTE\_F3 175

#define NOTE\_FS3 185

#define NOTE\_G3 196

#define NOTE\_GS3 208

#define NOTE\_A3 220

#define NOTE\_AS3 233

#define NOTE\_B3 247

#define NOTE\_C4 262

#define NOTE\_CS4 277

#define NOTE\_D4 294

#define NOTE\_DS4 311

#define NOTE\_E4 330

#define NOTE\_F4 349

#define NOTE\_FS4 370

#define NOTE\_G4 392

#define NOTE\_GS4 415

#define NOTE\_A4 440

#define NOTE\_AS4 466

#define NOTE\_B4 494

#define NOTE\_C5 523

#define NOTE\_CS5 554

#define NOTE\_D5 587

#define NOTE\_DS5 622

#define NOTE\_E5 659

#define NOTE\_F5 698

#define NOTE\_FS5 740

#define NOTE\_G5 784

#define NOTE\_GS5 831

#define NOTE\_A5 880

#define NOTE\_AS5 932

#define NOTE\_B5 988

#define NOTE\_C6 1047

#define NOTE\_CS6 1109

#define NOTE\_D6 1175

#define NOTE\_DS6 1245

#define NOTE\_E6 1319

#define NOTE\_F6 1397

#define NOTE\_FS6 1480

#define NOTE\_G6 1568

#define NOTE\_GS6 1661

#define NOTE\_A6 1760

#define NOTE\_AS6 1865

#define NOTE\_B6 1976

#define NOTE\_C7 2093

#define NOTE\_CS7 2217

#define NOTE\_D7 2349

#define NOTE\_DS7 2489

#define NOTE\_E7 2637

#define NOTE\_F7 2794

#define NOTE\_FS7 2960

#define NOTE\_G7 3136

#define NOTE\_GS7 3322

#define NOTE\_A7 3520

#define NOTE\_AS7 3729

#define NOTE\_B7 3951

#define NOTE\_C8 4186

#define NOTE\_CS8 4435

#define NOTE\_D8 4699

#define NOTE\_DS8 4978

#define REST 0

int alertaSonoroSimples[] = {

NOTE\_E5, 8,

NOTE\_D5, 8,

NOTE\_FS4, 4,

NOTE\_GS4, 4,

NOTE\_CS5, 8,

NOTE\_B4, 8,

NOTE\_D4, 4,

NOTE\_E4, 4,

NOTE\_B4, 8,

NOTE\_A4, 8,

NOTE\_CS4, 4,

NOTE\_E4, 4,

NOTE\_A4, 2

};

int alertaMusicalCurto[] = {

NOTE\_D4, -8,

NOTE\_G4, 16,

NOTE\_C5, -4,

NOTE\_B4, 8,

NOTE\_G4, -16,

NOTE\_E4, -16,

NOTE\_A4, -16,

NOTE\_D5, 2

};

int alertaMusicalLongo[] = {

NOTE\_D5, -4,

NOTE\_A5, 8,

NOTE\_FS5, 8,

NOTE\_D5, 8,

NOTE\_E5, -4,

NOTE\_FS5, 8,

NOTE\_G5, 4,

NOTE\_FS5, -4,

NOTE\_E5, 8,

NOTE\_FS5, 4,

NOTE\_D5, -2,

NOTE\_D5, -4,

NOTE\_A5, 8,

NOTE\_FS5, 8,

NOTE\_D5, 8,

NOTE\_E5, -4,

NOTE\_FS5, 8,

NOTE\_G5, 4,

NOTE\_FS5, -1,

NOTE\_D5, -4,

NOTE\_A5, 8,

NOTE\_FS5, 8,

NOTE\_D5, 8,

NOTE\_E5, -4,

NOTE\_FS5, 8,

NOTE\_G5, 4,

NOTE\_FS5, -4,

NOTE\_E5, 8,

NOTE\_FS5, 4,

NOTE\_D5, -2,

NOTE\_D5, -4,

NOTE\_A5, 8,

NOTE\_FS5, 8,

NOTE\_D5, 8,

NOTE\_E5, -4,

NOTE\_FS5, 8,

NOTE\_G5, 4,

NOTE\_FS5, -1

};

#define TAMANHO\_ALERTA\_SIMPLES 26

#define TAMANHO\_MUSICA\_CURTA 16

#define TAMANHO\_MUSICA\_LONGA 74

Módulo Buzzer (Buzzer.ino)

short alertaSelecionado = 1;

int notaAtual = 0;

bool emExecucao = false;

int\* melodiaSelecionada;

int tamanhoMelodia;

void iniciarBuzzer() {

EasyBuzzer.setPin(PINO\_BUZZER);

}

void selecionarAlerta(short numeroAlerta) {

alertaSelecionado = numeroAlerta;

}

void selecionarMelodia() {

switch (alertaSelecionado) {

case 0:

tamanhoMelodia = TAMANHO\_ALERTA\_SIMPLES;

prepararMelodia(alertaSonoroSimples);

break;

case 1:

tamanhoMelodia = TAMANHO\_MUSICA\_CURTA;

prepararMelodia(alertaMusicalCurto);

break;

case 2:

tamanhoMelodia = TAMANHO\_MUSICA\_LONGA;

prepararMelodia(alertaMusicalLongo);

break;

default:

tamanhoMelodia = TAMANHO\_ALERTA\_SIMPLES;

prepararMelodia(alertaSonoroSimples);

}

}

void prepararMelodia(int melodia[]) {

if (melodiaSelecionada != NULL) {

free(melodiaSelecionada);

}

melodiaSelecionada = (int \*) malloc(tamanhoMelodia \* sizeof(int));

for (int i=0; i<tamanhoMelodia; i++) {

melodiaSelecionada[i] = melodia[i];

}

}

void pararAlerta() {

EasyBuzzer.stopBeep();

}

void tocarAlerta() {

selecionarMelodia();

if (notaAtual == tamanhoMelodia) {

notaAtual = 0;

pararAlerta();

return;

}

EasyBuzzer.singleBeep(

melodiaSelecionada[notaAtual],

obterTempo(melodiaSelecionada[notaAtual+1]),

tocarAlerta

);

notaAtual += 2;

}

int obterTempo(int divisor) {

int duracao = 0;

int compasso = (60000 \* 4) / 80;

if (divisor > 0) {

duracao = (compasso) / divisor;

} else if (divisor < 0) {

duracao = (compasso) / abs(divisor);

duracao \*= 1.5;

}

return duracao;

}

void atualizarMusica() {

EasyBuzzer.update();

}

Módulo Display (Display.ino)

// pinoSDA GPIO 21

// pinoSCL GPIO 22

LiquidCrystal\_I2C displayLCD(0x27, 16, 2);

void iniciarDisplay() {

displayLCD.begin();

displayLCD.setBacklight(HIGH);

}

void escreverDisplay(String texto, byte linha, byte coluna) {

displayLCD.setCursor(coluna,linha);

displayLCD.print(texto);

}

void limparDisplay()

{

displayLCD.clear();

}

Módulo MQTT (MQTT.ino)

WiFiClient wifiClient;

PubSubClient mqttClient(wifiClient);

const char\* BROKER\_ADDRESS = "192.168.2.110";

int BROKER\_PORT = 1883;

const char\* ID\_MQTT = "esp32\_mqtt";

char\* topicos[] = {

AGENDAMENTOS,

TEMPO\_ABASTECIMENTO,

ALARME\_SELECIONADO

};

void inciarMQTT() {

mqttClient.setServer(BROKER\_ADDRESS, BROKER\_PORT);

mqttClient.setCallback(receberResposta);

}

void conectarMQTT() {

while (mqttClient.connected() == false && temporizar(2000, MQTT) == true) {

if (mqttClient.connect(ID\_MQTT) == true ) {

assinarTopicos();

return;

}

else {

limparDisplay();

escreverDisplay("FALHA NO MQTT", 0, 0);

escreverDisplay("NOVA TENTATIVA", 1, 0);

delay(400);

}

}

limparDisplay();

escreverDisplay("MQTT OFFLINE", 0, 0);

}

void assinarTopicos() {

int totalTopicos = 3; //sizeof(topicos) / sizeof(topicos[0]);

for (int atual=0; atual < totalTopicos; atual++) {

mqttClient.subscribe(topicos[atual], 1);

}

}

void verificarMQTT() {

if (verificarConexao() && mqttClient.connected() == false) {

conectarMQTT();

}

mqttClient.loop();

}

void receberResposta(char\* topico, byte\* conteudo, unsigned int tamanho) {

String topicoAtual = String(topico);

String mensagem;

for (int i = 0; i < tamanho; i++) {

char caractere = (char)conteudo[i];

mensagem = mensagem + caractere;

}

if (topicoAtual.equals(AGENDAMENTOS)) {

armazenarTexto(AGENDAMENTOS, mensagem);

atribuirAgendamentos(mensagem);

} else if (topicoAtual.equals(TEMPO\_ABASTECIMENTO)) {

armazenarNumero(TEMPO\_ABASTECIMENTO, mensagem.toInt());

atribuirTempoAbastecimento(mensagem.toInt());

} else if (topicoAtual.equals(ALARME\_SELECIONADO)) {

armazenarNumero(ALARME\_SELECIONADO, mensagem.toInt());

selecionarAlerta(mensagem.toInt());

} else if (topicoAtual.equals(TESTE\_ABASTECIMENTO)) {

testarAbastecimento();

}

}

void publicarTopico(char\* topico, const char\* conteudo) {

mqttClient.publish(topico, conteudo);

}

Módulo de Gerenciamento de Tempo (Time.ino)

#define HORA\_EM\_MS 3600000

volatile byte horaAtual = 0;

volatile byte minutoAtual = 0;

volatile byte segundoAtual = 0;

int horasPassadas = 0;

WiFiUDP udpClient;

NTPClient ntpClient(udpClient, "a.st1.ntp.br", -3 \* 3600, 60000);

void iniciarNTPClient() {

ntpClient.begin();

obterHorarioInternet();

}

String obterTempoAtual() {

String textoHoras = preencherZeros(horaAtual, 2);

String textoMinutos = preencherZeros(minutoAtual, 2);

return textoHoras + ":" + textoMinutos;

}

void obterHorarioInternet(){

if (verificarConexao() == true) {

ntpClient.forceUpdate();

horaAtual = ntpClient.getHours();

minutoAtual = ntpClient.getMinutes();

segundoAtual = ntpClient.getSeconds();

}

}

void atualizarHorario(){

if (temporizar(1000, TIME) == false) {

adicionarSegundo();

}

if (millis() % (HORA\_EM\_MS \* (horasPassadas + 1)) > 0) {

horasPassadas++;

obterHorarioInternet();

}

}

void adicionarSegundo() {

segundoAtual++;

if (segundoAtual > 59) {

segundoAtual = 0;

minutoAtual++;

}

if (minutoAtual > 59) {

minutoAtual = 0;

horaAtual++;

}

if (horaAtual > 23) {

horaAtual = 0;

}

}

Módulo WI-FI (WIFI.h)

const char\* ssid = "MY\_WIFI";

const char\* password = "ITSNOT1234";

void iniciarWifi() {

WiFi.begin(ssid, password);

conectarWifi();

}

void conectarWifi() {

while (WiFi.status() != WL\_CONNECTED && temporizar(2000, WIFI) == true)

{

delay(200);

limparDisplay();

escreverDisplay("CONECTANDO", 0, 0);

escreverDisplay("A WI-FI...", 1, 0);

}

if (WiFi.status() != WL\_CONNECTED) {

notificarFaltaWifi();

return;

}

limparDisplay();

escreverDisplay("CONECTADO", 0, 0);

}

bool verificarWifi() {

if (WiFi.status() != WL\_CONNECTED) {

conectarWifi();

if ((WiFi.status() != WL\_CONNECTED)) {

return false;

}

}

return true;

}

bool verificarConexao() {

return WiFi.status() == WL\_CONNECTED;

}

void notificarFaltaWifi() {

limparDisplay();

escreverDisplay("NAO FOI POSSIVEL", 0, 0);

escreverDisplay("CONECTAR A WI-FI", 1, 0);

}

Módulo de Funções Auxiliares (Helper.ino)

volatile unsigned long tempoAtivacao[] = { 0, 0, 0, 0, 0 };

bool temporizar(int milissegundos, int remetente) {

unsigned long tempoAtual = millis();

if (tempoAtivacao[remetente] == 0) {

tempoAtivacao[remetente] = tempoAtual;

}

if (tempoAtual - tempoAtivacao[remetente] > milissegundos)

{

tempoAtivacao[remetente] = 0;

return false;

}

return true;

}

String preencherZeros(int numero, int tamanho) {

String textoNumero = String(numero);

int tamanhoNumero = textoNumero.length();

if (tamanhoNumero < tamanho) {

for (int n = 0; n < (tamanho - tamanhoNumero); n++) {

textoNumero = "0" + textoNumero;

}

}

return textoNumero;

}

ANEXO 2 – Programa da Interface Node-RED

Tela do Abastecimento

[{"id":"233f56ab.9173da","type":"tab","label":"ABASTECIMENTO","disabled":false,"info":""},{"id":"b47f64c3.d647e8","type":"ui\_slider","z":"233f56ab.9173da","name":"Seletor de Segudos","label":"","tooltip":"","group":"7409af17.191cb","order":2,"width":4,"height":1,"passthru":true,"outs":"end","topic":"","min":"1","max":"30","step":1,"x":670,"y":700,"wires":[["43bb2ad7.647cb4","59bc1d32.b98104","9c96f92b.f3b988"]]},{"id":"b196d64d.fbc878","type":"ui\_text","z":"233f56ab.9173da","group":"7409af17.191cb","order":1,"width":6,"height":1,"name":"Label Seletor de Segundos","label":"Tempo em Segundos:","format":"","layout":"row-spread","x":140,"y":620,"wires":[]},{"id":"9979f339.870ce","type":"ui\_text","z":"233f56ab.9173da","group":"7409af17.191cb","order":3,"width":2,"height":1,"name":"Display Segundos","label":"","format":"{{msg.payload}}","layout":"row-left","x":1150,"y":700,"wires":[]},{"id":"43bb2ad7.647cb4","type":"function","z":"233f56ab.9173da","name":"Ler Segundos","func":"msg.payload = msg.payload + \" s\";\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":900,"y":700,"wires":[["9979f339.870ce"]]},{"id":"b11c6ed8.0c982","type":"comment","z":"233f56ab.9173da","name":"TEMPO DE ABASTECIMENTO","info":"","x":150,"y":540,"wires":[]},{"id":"36d00409.9aa26c","type":"ui\_button","z":"233f56ab.9173da","name":"Botão Teste","group":"7409af17.191cb","order":4,"width":0,"height":0,"passthru":false,"label":"Testar Abastecimento","tooltip":"","color":"","bgcolor":"","icon":"","payload":"1","payloadType":"str","topic":"","x":190,"y":780,"wires":[["ab47f50e.eb2448"]]},{"id":"a36a3527.68aba8","type":"function","z":"233f56ab.9173da","name":"Atualizar Lista","func":"let agendamentos = global.get(\"agenda\");\n\nmsg.payload = agendamentos;\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":440,"y":260,"wires":[["e8c23639.2b2808"]]},{"id":"e8c23639.2b2808","type":"ui\_list","z":"233f56ab.9173da","group":"80a744ec.4e2588","name":"Lista de Agendamento","order":4,"width":0,"height":0,"lineType":"one","actionType":"check","allowHTML":false,"outputs":1,"topic":"","x":680,"y":260,"wires":[["f81da82b.2e4028"]]},{"id":"539e3d02.4ff9f4","type":"function","z":"233f56ab.9173da","name":"Inicialização","func":"","outputs":1,"noerr":0,"initialize":"// Code added here will be run once\n// whenever the node is deployed.\nlet inicializado = global.get(\"agenda\");\n\nif (inicializado !== undefined) {\n return;\n}\n\nconst formatarHorario = (milliseconds) =>\n{\n const minutes = Math.floor(((milliseconds / (1000\*60)) % 60));\n const hours = Math.floor(((milliseconds / (1000\*60\*60)) % 24));\n return `${hours.toString().padStart(2, '0')}:${minutes.toString().padStart(2, '0')}`;\n}\n\nconst compararHorario = (horario1, horario2) => {\n const h1 = parseInt(horario1.title.split(\":\")[0]);\n const m1 = parseInt(horario1.title.split(\":\")[1]);\n const h2 = parseInt(horario2.title.split(\":\")[0]);\n const m2 = parseInt(horario2.title.split(\":\")[1]);\n if (h1 !== h2) {\n return h1 - h2;\n }\n return m1 - m2;\n}\n\nconst toggleTipoAlerta = {\n state: 0\n};\n\nconst toggleAlertaSonoro = {\n state: 0\n};\n\nconst atualizacaoAbastecimento = {\n state: 0\n}\n\nconst agenda = [];\nglobal.set(\"agenda\", agenda);\nglobal.set(\"textoAgendamentos\", \"\");\nglobal.set(\"toggleTipoAlerta\", toggleTipoAlerta);\nglobal.set(\"toggleAlertaSonoro\", toggleAlertaSonoro);\nglobal.set(\"atualizacaoAbastecimento\", atualizacaoAbastecimento);\nglobal.set(\"formatarHorario\", formatarHorario);\nglobal.set(\"compararHorario\", compararHorario);","finalize":"","x":190,"y":120,"wires":[[]]},{"id":"de2ceeee.09718","type":"ui\_text\_input","z":"233f56ab.9173da","name":"Caixa Horário","label":"Horário:","tooltip":"","group":"80a744ec.4e2588","order":1,"width":4,"height":1,"passthru":false,"mode":"time","delay":"0","topic":"","x":660,"y":200,"wires":[["b3d4432d.c01b4"]]},{"id":"b3d4432d.c01b4","type":"function","z":"233f56ab.9173da","name":"Ler Horário","func":"const formatarHorario = global.get(\"formatarHorario\");\nglobal.set(\"horarioSelecionado\", formatarHorario(msg.payload));\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":890,"y":200,"wires":[[]]},{"id":"acc42f82.cca4f","type":"ui\_button","z":"233f56ab.9173da","name":"Botão Adicionar","group":"80a744ec.4e2588","order":2,"width":1,"height":1,"passthru":false,"label":"","tooltip":"","color":"","bgcolor":"","icon":"fa-check","payload":"","payloadType":"str","topic":"plus","x":660,"y":320,"wires":[["543eb563.63df1c"]]},{"id":"49a6e6ae.236458","type":"ui\_button","z":"233f56ab.9173da","name":"Botão Remover","group":"80a744ec.4e2588","order":3,"width":1,"height":1,"passthru":false,"label":"","tooltip":"","color":"","bgcolor":"","icon":"fa-trash-o","payload":"","payloadType":"str","topic":"minus","x":660,"y":380,"wires":[["b9b04548.968068"]]},{"id":"543eb563.63df1c","type":"function","z":"233f56ab.9173da","name":"Adicionar Horário","func":"const horarioSelecionado = global.get(\"horarioSelecionado\");\nlet textoAgendamentos = global.get(\"textoAgendamentos\");\n\nif (textoAgendamentos === undefined) {\n textoAgendamentos = \"\";\n}\nif (horarioSelecionado) {\n if (!textoAgendamentos.includes(horarioSelecionado)) {\n if (textoAgendamentos.length === 0) {\n textoAgendamentos = \"\";\n } else {\n textoAgendamentos = textoAgendamentos.concat(\";\");\n } \n textoAgendamentos = textoAgendamentos.concat(horarioSelecionado);\n }\n}\nglobal.set(\"textoAgendamentos\", textoAgendamentos);\nmsg.payload = textoAgendamentos;\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":910,"y":320,"wires":[[]]},{"id":"b9b04548.968068","type":"function","z":"233f56ab.9173da","name":"Remover Horário","func":"const agenda = global.get(\"agenda\");\nlet textoAgendamentos = global.get(\"textoAgendamentos\");\n\nif (agenda.find(i => i.isChecked)) {\n for(let i = agenda.length-1; i >= 0; i--) {\n if(agenda[i].isChecked) {\n const largura = textoAgendamentos.length;\n textoAgendamentos = textoAgendamentos.replace(agenda[i].title, '');\n textoAgendamentos = textoAgendamentos.replace(';;', ';');\n if (textoAgendamentos.charAt(0) == \";\") {\n textoAgendamentos = textoAgendamentos.substring(1);\n }\n if (textoAgendamentos.slice(-1) == \";\") {\n textoAgendamentos = textoAgendamentos.slice(0, -1);\n }\n }\n }\n global.set(\"textoAgendamentos\", textoAgendamentos);\n}\nmsg.payload = textoAgendamentos;\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":910,"y":380,"wires":[[]]},{"id":"f81da82b.2e4028","type":"function","z":"233f56ab.9173da","name":"Selecionar Horário","func":"let agenda = global.get(\"agenda\");\nlet itemHorario = msg.payload;\nlet horario = agenda.find(i => i.description === itemHorario.description);\nif (horario) {\n horario.isChecked = itemHorario.isChecked;\n}\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":910,"y":260,"wires":[[]]},{"id":"2f0eacde.17b3f4","type":"comment","z":"233f56ab.9173da","name":"AGENDAMENTO","info":"","x":100,"y":40,"wires":[]},{"id":"725059f8.6c1808","type":"mqtt in","z":"233f56ab.9173da","name":"Tópico horariosAlimentacao","topic":"alimentador/horariosAlimentacao","qos":"2","datatype":"auto","broker":"235733c0.070d7c","x":140,"y":380,"wires":[["4514b507.53cd8c"]]},{"id":"4514b507.53cd8c","type":"function","z":"233f56ab.9173da","name":"Atualizar Horários","func":"const compararHorario = global.get(\"compararHorario\");\nconst agendamentos = global.get(\"agenda\");\nlet textoAgendamentos = msg.payload;\n\nif (!textoAgendamentos || textoAgendamentos.length === 0) { \n textoAgendamentos = \"\";\n}\n\nmsg.payload = agendamentos;\n\nagendamentos.length = 0;\n\nlet horarios = textoAgendamentos.split(';');\n\nfor (let i = 0; i < horarios.length; i++) \n{\n agendamentos.push({\n icon\_name: \"fa-clock-o\",\n title: horarios[i],\n isChecked: false\n });\n}\nagendamentos.sort(compararHorario);\n\nglobal.set(\"textoAgendamentos\", textoAgendamentos);\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":450,"y":380,"wires":[[]]},{"id":"7579260b.b09978","type":"function","z":"233f56ab.9173da","name":"Converter Agendamentos","func":"msg.payload = global.get(\"textoAgendamentos\");\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":690,"y":120,"wires":[["ef5ae107.8aeba"]]},{"id":"f5f038be.d2cb08","type":"complete","z":"233f56ab.9173da","name":"Ao Alterar Horário","scope":["543eb563.63df1c","b9b04548.968068"],"uncaught":false,"x":450,"y":120,"wires":[["7579260b.b09978"]]},{"id":"ef5ae107.8aeba","type":"mqtt out","z":"233f56ab.9173da","name":"Tópico horariosAlimentacao","topic":"alimentador/horariosAlimentacao","qos":"2","retain":"","broker":"235733c0.070d7c","x":940,"y":120,"wires":[]},{"id":"30459ab4.daff66","type":"ui\_ui\_control","z":"233f56ab.9173da","name":"Ao Conectar à Interface","events":"connect","x":150,"y":260,"wires":[["a36a3527.68aba8"]]},{"id":"1b5fb7ea.f21908","type":"complete","z":"233f56ab.9173da","name":"Ao Receber do MQTT","scope":["4514b507.53cd8c"],"uncaught":false,"x":160,"y":200,"wires":[["a36a3527.68aba8"]]},{"id":"59bc1d32.b98104","type":"mqtt out","z":"233f56ab.9173da","name":"Tópico tempoAbastecimento","topic":"alimentador/tempoAbastecimento","qos":"2","retain":"","broker":"235733c0.070d7c","x":940,"y":640,"wires":[]},{"id":"26a3cc8f.782d04","type":"mqtt in","z":"233f56ab.9173da","name":"Tópico tempoAbastecimento","topic":"alimentador/tempoAbastecimento","qos":"2","datatype":"auto","broker":"235733c0.070d7c","x":140,"y":700,"wires":[["b3c2ad0.9ccfe5"]]},{"id":"9c96f92b.f3b988","type":"function","z":"233f56ab.9173da","name":"Delay Atualização","func":"const atualizacaoAbastecimento = global.get(\"atualizacaoAbastecimento\");\natualizacaoAbastecimento.state = 1;\nsetTimeout(()=> { \n atualizacaoAbastecimento.state = 0;\n}, 100);\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":910,"y":760,"wires":[[]]},{"id":"b3c2ad0.9ccfe5","type":"function","z":"233f56ab.9173da","name":"Verificar Delay","func":"const atualizacaoAbastecimento = global.get(\"atualizacaoAbastecimento\");\nif (atualizacaoAbastecimento.state === 0) {\n msg.state = msg.payload;\n return msg;\n}\n","outputs":1,"noerr":0,"initialize":"","finalize":"","x":440,"y":700,"wires":[["b47f64c3.d647e8"]]},{"id":"ab47f50e.eb2448","type":"mqtt out","z":"233f56ab.9173da","name":"Tópico testeAbastecimento","topic":"alimentador/testeAbastecimento","qos":"2","retain":"","broker":"235733c0.070d7c","x":480,"y":780,"wires":[]},{"id":"7409af17.191cb","type":"ui\_group","z":"","name":"Configurações","tab":"8b5c3e2d.1707a","order":2,"disp":true,"width":"6","collapse":false},{"id":"80a744ec.4e2588","type":"ui\_group","z":"","name":"Agendamentos","tab":"8b5c3e2d.1707a","order":1,"disp":true,"width":"6","collapse":false},{"id":"235733c0.070d7c","type":"mqtt-broker","z":"","name":"Alimentador-Broker","broker":"localhost","port":"1883","clientid":"Interface-NodeRED","usetls":false,"compatmode":false,"keepalive":"60","cleansession":false,"birthTopic":"interface/estado","birthQos":"1","birthRetain":"true","birthPayload":"ativo","closeTopic":"interface/estado","closeQos":"1","closeRetain":"true","closePayload":"desligado","willTopic":"interface/estado","willQos":"1","willRetain":"false","willPayload":"desconectado"},{"id":"8b5c3e2d.1707a","type":"ui\_tab","z":"","name":"Abastecimento","icon":"fa-filter","order":3,"disabled":false,"hidden":false}]

Tela do Reservatório

[{"id":"8f5dbf10.95e12","type":"tab","label":"RESERVATÓRIO","disabled":false,"info":""},{"id":"154ead5f.1e5393","type":"ui\_gauge","z":"8f5dbf10.95e12","name":"Gauge Reservatório","group":"3e3cd49c.d121fc","order":1,"width":0,"height":0,"gtype":"wave","title":"Estado Atual","label":"~","format":"{{value}}","min":"0","max":"2","colors":["#b30000","#e6e600","#4acb3a"],"seg1":"","seg2":"","x":460,"y":100,"wires":[]},{"id":"847f9df5.0730b","type":"ui\_text","z":"8f5dbf10.95e12","group":"3e3cd49c.d121fc","order":2,"width":"6","height":"1","name":"Texto Reservatório","label":"","format":"{{msg.payload}}","layout":"row-center","x":730,"y":160,"wires":[]},{"id":"f8e89436.318fc8","type":"function","z":"8f5dbf10.95e12","name":"Formatação do Texto","func":"if (msg.payload == '0')\n{\n msg.payload = 'Reservatório vazio';\n} \nelse if (msg.payload == '1')\n{\n msg.payload = 'Reservatório em consumo';\n} \nelse if (msg.payload == '2')\n{\n msg.payload = 'Reservatório abastecido';\n}\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":460,"y":160,"wires":[["847f9df5.0730b"]]},{"id":"6d08eec9.ed9ac","type":"comment","z":"8f5dbf10.95e12","name":"RESERVATÓRIO","info":"","x":100,"y":40,"wires":[]},{"id":"3f61f1f0.6a493e","type":"ui\_switch","z":"8f5dbf10.95e12","name":"Ativador Alerta Sonoro","label":"Alerta Sonoro Ativado:","tooltip":"","group":"fb14224f.ca57","order":1,"width":0,"height":0,"passthru":true,"decouple":"false","topic":"","style":"","onvalue":"1","onvalueType":"str","onicon":"","oncolor":"","offvalue":"0","offvalueType":"str","officon":"","offcolor":"","x":740,"y":300,"wires":[["4099dcd7.7395c4","faf6faf7.a53428"]]},{"id":"8d0be730.91bc28","type":"ui\_dropdown","z":"8f5dbf10.95e12","name":"Seleção de Alerta Sonoro","label":"","tooltip":"","place":"Selecione","group":"fb14224f.ca57","order":2,"width":0,"height":0,"passthru":true,"multiple":false,"options":[{"label":"Alerta Sonoro Simples","value":"0","type":"str"},{"label":"Alerta Musical Curto","value":"1","type":"str"},{"label":"Alerta Musical Longo","value":"2","type":"str"}],"payload":"","topic":"","x":750,"y":540,"wires":[["edf0fb6a.f38768","d2f45f15.49d7e"]]},{"id":"84c94f05.ef35b","type":"ui\_text","z":"8f5dbf10.95e12","group":"fb14224f.ca57","order":1,"width":"6","height":"1","name":"Label Alerta Sonoro","label":"Tipo de Alerta Sonoro:","format":"","layout":"row-spread","x":730,"y":480,"wires":[]},{"id":"3f52b7d1.866a38","type":"mqtt in","z":"8f5dbf10.95e12","name":"Tópico nivelReservatorio","topic":"alimentador/nivelReservatorio","qos":"1","datatype":"auto","broker":"235733c0.070d7c","x":130,"y":100,"wires":[["154ead5f.1e5393","f8e89436.318fc8"]]},{"id":"c91e5563.30bee8","type":"mqtt in","z":"8f5dbf10.95e12","name":"Tópico alertaSonoroAtivo","topic":"alimentador/alertaSonoroAtivo","qos":"2","datatype":"auto","broker":"235733c0.070d7c","x":130,"y":300,"wires":[["d185500c.0f87b"]]},{"id":"faf6faf7.a53428","type":"mqtt out","z":"8f5dbf10.95e12","name":"Tópico alertaSonoroAtivo","topic":"alimentador/alertaSonoroAtivo","qos":"1","retain":"true","broker":"235733c0.070d7c","x":1070,"y":300,"wires":[]},{"id":"4099dcd7.7395c4","type":"function","z":"8f5dbf10.95e12","name":"Delay Atualização","func":"const toggleAlertaSonoro = global.get(\"toggleAlertaSonoro\");\ntoggleAlertaSonoro.state = 1;\nsetTimeout(()=> { \n toggleAlertaSonoro.state = 0;\n}, 100);\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":1050,"y":360,"wires":[[]]},{"id":"d185500c.0f87b","type":"function","z":"8f5dbf10.95e12","name":"Verificar Delay","func":"const toggleAlertaSonoro = global.get(\"toggleAlertaSonoro\");\nif (toggleAlertaSonoro.state === 0) {\n msg.state = msg.payload;\n return msg;\n}\n","outputs":1,"noerr":0,"initialize":"","finalize":"","x":440,"y":300,"wires":[["3f61f1f0.6a493e"]]},{"id":"edadefb.0da241","type":"mqtt in","z":"8f5dbf10.95e12","name":"Tópico tipoAlertaSonoro","topic":"alimentador/tipoAlertaSonoro","qos":"1","datatype":"utf8","broker":"235733c0.070d7c","x":140,"y":540,"wires":[["1e2ff173.fdb96f"]]},{"id":"1e2ff173.fdb96f","type":"function","z":"8f5dbf10.95e12","name":"Verificar Delay","func":"const toggleTipoAlerta = global.get(\"toggleTipoAlerta\");\nif (toggleTipoAlerta.state === 0) {\n msg.state = msg.payload;\n return msg;\n}\n","outputs":1,"noerr":0,"initialize":"","finalize":"","x":440,"y":540,"wires":[["8d0be730.91bc28"]]},{"id":"edf0fb6a.f38768","type":"function","z":"8f5dbf10.95e12","name":"Delay Atualização","func":"const toggleTipoAlerta = global.get(\"toggleTipoAlerta\");\ntoggleTipoAlerta.state = 1;\nsetTimeout(()=> { \n toggleTipoAlerta.state = 0;\n}, 100);\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":1050,"y":600,"wires":[[]]},{"id":"d2f45f15.49d7e","type":"mqtt out","z":"8f5dbf10.95e12","name":"Tópico tipoAlertaSonoro","topic":"alimentador/tipoAlertaSonoro","qos":"1","retain":"true","broker":"235733c0.070d7c","x":1070,"y":540,"wires":[]},{"id":"5b0a272d.e6b698","type":"comment","z":"8f5dbf10.95e12","name":"ALERTA SONORO","info":"","x":110,"y":420,"wires":[]},{"id":"3e3cd49c.d121fc","type":"ui\_group","z":"","name":"Principal","tab":"a977fbfb.051808","order":1,"disp":true,"width":"6","collapse":false},{"id":"fb14224f.ca57","type":"ui\_group","z":"","name":"Aviso Sonoro","tab":"a977fbfb.051808","order":2,"disp":true,"width":"6","collapse":false},{"id":"235733c0.070d7c","type":"mqtt-broker","z":"","name":"Alimentador-Broker","broker":"localhost","port":"1883","clientid":"Interface-NodeRED","usetls":false,"compatmode":false,"keepalive":"60","cleansession":false,"birthTopic":"interface/estado","birthQos":"1","birthRetain":"true","birthPayload":"ativo","closeTopic":"interface/estado","closeQos":"1","closeRetain":"true","closePayload":"desligado","willTopic":"interface/estado","willQos":"1","willRetain":"false","willPayload":"desconectado"},{"id":"a977fbfb.051808","type":"ui\_tab","z":"","name":"Reservatorio","icon":"fa-database","order":1,"disabled":false,"hidden":false}]

ANEXO 3 – Material na Internet

O material desenvolvido citado no trabalho está disponível no repositório de código online hospedado no endereço:

* https://github.com/AntonioZanini/TCC-ETEC-ELO20-02