Universidade Federal de Uberlândia

Anderson Giovanini Silva

Datalogger aplicado no monitoramento dos equipamentos de armazenamento e refrigeração de leite nas pequenas propriedades rurais

Uberlândia

2023

Anderson Giovanini Silva

Datalogger aplicado no monitoramento dos equipamentos de armazenamento e refrigeração de leite nas pequenas propriedades rurais

Trabalho de conclusão de curso apresentado no programa de graduação em Engenharia Elétrica da Universidade Federal de Uberlândia, como parte dos requisitos para obtenção do título de Bacharel em Engenharia Elétrica.

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica

Orientador: Prof. Dr. Luciano Coutinho Gomes

Uberlândia

2023

Anderson Giovanini Silva

Datalogger aplicado no monitoramento dos equipamentos de armazenamento e refrigeração de leite nas pequenas propriedades rurais

Trabalho de conclusão de curso apresentado no programa de graduação em Engenharia Elétrica da Universidade Federal de Uberlândia, como parte dos requisitos para obtenção do título de Bacharel em Engenharia Elétrica.

Uberlândia, 15 de junho de 2023.

Prof. Dr. Luciano Coutinho Gomes, UFU/MG

Prof. Dr. Carlos Eduardo Tavares, UFU/MG

Prof. Dr. Isaque Nogueira Gondim, UFU/MG

Dedico este trabalho ao meus pais, que sempre estiveram ao meu lado, a vida inteira.

Agradecimentos

Aos meus pais e irmão, que sempre estiveram ao meu lado e me incentivaram nos momentos difíceis;

Ao professor Dr. Luciano Coutinho Gomes, pela compreensão, paciência e suporte na elaboração deste trabalho;

Aos professores que me acompanharam ao longo do curso e que, com empenho, se dedicam à arte de ensinar;

Aos meus colegas de curso, por todo o companheirismo ao longo desta trajetória;

À Universidade Federal de Uberlândia, pelas oportunidades e experiências que me proporcionou ao longo do curso.

Resumo

Com a modernização da indústria leiteira brasileira, associado com a crescente exigência de produção em ampla escala com qualidade, o leite se tornou um dos alimentos mais analisados nesse critério. Neste sentido, propõe-se a aplicação de um *datalogger* nos equipamentos de armazenamento e refrigeração do leite, a fim de realizar o monitoramento das principais grandezas que afetam a qualidade do leite nas pequenas propriedades rurais. Para isso, o dispositivo irá coletar valores de tensão elétrica da rede, temperatura do leite e do compressor, registrando os dados em cartão microSD juntamente com o exato momento da coleta, por meio de um módulo RTC. Sendo todos os dispositivos controlados pela placa de prototipagem Arduino UNO. A extração dos dados armazenados será feita via retirada do cartão microSD e cópia do arquivo para o sistema de visualização, que será responsável por tratar e representar graficamente os valores coletados. Este trabalho foi desenvolvido utilizando a metodologia de validação e incrementação de recursos, onde cada parte da ferramenta foi testada e validada de forma individual. Todos os componentes apresentaram resultados satisfatórios durante as etapas de validação e aplicação, atendendo todos os requisitos levantados nesse projeto.

Palavras-chave: *Datalogger*. Sensor. Módulo. Dados. Power BI.

Abstract

With the modernization of the brazilian dairy industry, associated with the growing demand for large-scale production with quality, milk has become one of the foods most analyzed in this standard. In this sense, it is proposed the application of a datalogger in milk storage and refrigeration equipment, in order to monitor the main quantities that affect the quality of milk in small rural properties. For this, the device will collect values of electrical voltage from the network, temperature of the milk and the compressor, recording the data on a microSD card together with the exact moment of collection, through an RTC module. All devices being controlled by the Arduino UNO prototyping board. The extraction of the stored data will be done by removing the microSD card and copying the file to the visualization system, which will be responsible for treating and graphically representing the collected values. This work was developed using the validation and resource enhancement methodology, where each part of the tool was tested and validated individually. All components presented satisfactory results during the validation and application stages, meeting all the requirements raised in this project.

Keywords: Datalogger. Sensor. Module. Data. Power BI.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MAPA	Ministério da Agricultura Pecuária e Abastecimento
RTC	Real Time Clock
IDE	Integrated Development Environment
SD	Secure Digital
NTC	Negative Temperature Coefficient
BI	Business Intelligence
QEE	Qualidade de Energia Elétrica
PWM	Pulse Width Modulation
CSV	Comma-separated values

Sumário

1. INTRODUÇÃO

Desde o início da década de 1990, a indústria leiteira brasileira passou por grandes transformações, buscando ser competitiva e inovadora no mercado internacional, tendo como objetivo a produção em ampla escala com qualidade, agregação de valor e industrialização dos mais diferentes tipos de produtos (CORRÊA *et al*, 2010). De acordo com Brito e Dias (1998), o leite se tornou um dos alimentos mais analisados nos critérios de higiene e qualidade pelas indústrias.

Por outro lado, quando comparamos o nível de evolução tecnológica aplicada nas pequenas propriedades rurais produtoras de leite com as atuais demandas de qualidade das indústrias leiteiras, percebemos uma divergência considerável entre os dois setores, partindo desde da disponibilidade do sistema de armazenamento e refrigeração do leite até a forma de atuação do suporte técnico, no qual Novo (2001), destaca a baixa eficiência da assistência técnica aos produtores rurais como um dos principais limitantes do desenvolvimento do setor produtivo leiteiro e segundo Leira *et al* (2018), para que os produtores de leite consigam atender o crescente nível de exigências de qualidade das indústrias e consumidores, é preciso investir em tecnologias de monitoramento e qualidade do leite.

Diversos são os fatores que podem influenciar a qualidade do leite nas propriedades rurais, como clima, higiene, nutrição animal e temperatura de armazenamento (PEREIRA *et al*, 2010). A temperatura de armazenagem é diretamente influenciada pela confiabilidade do sistema de refrigeração, que consequentemente tem seu funcionamento atrelado a boas práticas de manutenções e uma assistência técnica qualificada, rápida e eficiente (CAVALCANTE; ALMEIDA, 2005). Todavia, devido à grande distância entre produtor rural e a equipe de manutenção, combinado a falta de informações técnicas e histórico de funcionamento, dificulta-se a correção dos problemas de forma rápida e eficaz, causando, na maioria das vezes, danos físicos no sistema de refrigeração e até mesmo à perda total do leite.

Dessa forma, o monitoramento das grandezas que influenciam diretamente na qualidade do leite e funcionamento do sistema de refrigeração, nas pequenas propriedades rurais, é essencial para que o suporte técnico possa diagnosticar problemas

de maneira rápida e até mesmo preventiva (ROGÉRIO, 2020). Para isso, pensando em possíveis ferramentas de armazenamento de dados baseado em leitura de sensores, destaca-se os microcontroladores. Em sua grande maioria, esses dispositivos são contemplados por plataformas abertas que permitem os usuários desenvolverem códigos para adaptar a computação física e digital a sua necessidade, no qual através da programação combinada a diferentes tipos de sensores, motores e outros dispositivos eletrônicos, é possível criar diversos projetos interativos (MCROBERTS, 2015).

Pensando nisso, esse projeto busca desenvolver um *datalogger* para registrar os dados de funcionamento do sistema de armazenamento e refrigeração do leite. Tendo como objetivo a construção de uma ferramenta a custo acessível, que visa encurtar a distância entre produtor rural e assistência técnica através do disparo de dados via internet ou extração de dados no local, facilitando a identificação de problemas, redução de custos com manutenção e análise de tendências com base em dados para a realização de diagnósticos preventivos.

1.1 Justificativa

A aplicação do *datalogger* no monitoramento dos equipamentos de armazenamento e refrigeração de leite nas pequenas propriedades rurais justifica-se pelo fato de que, apesar do avanço na área de informação e comunicação de dados, ocorreu um processo de abandono digital na zona rural, circunstância que impossibilitou a reprodução das tecnologias empregadas pelas indústrias em seus procedimentos de manutenção e processos nas propriedades rurais. Podemos perceber o impacto dessa diferença tecnológica pela demora ou até mesmo ausência de informações na comunicação de defeitos para o suporte técnico, causando, na maioria das vezes, danos físicos ao sistema de refrigeração e até mesmo perda do leite.

Dessa forma, o presente trabalho partiu da necessidade de entender como a implementação de ferramentas de armazenamento e comunicação de dados, combinada a um baixo valor de investimento, pode contribuir para encurtar a distância entre o produtor rural e a assistência técnica, visando melhorar a confiabilidade e a qualidade do sistema de armazenamento e refrigeração do leite.

1.2 Objetivos

Esse trabalho tem por objetivo projetar e construir um datalogger para registrar e possibilitar a extração do histórico de funcionamento do sistema de armazenamento e refrigeração de leite, no próprio local ou à distância, a fim de evitar a perda da qualidade do leite.

Os objetivos específicos são listados a seguir:

- Identificar as grandezas que podem ser monitoradas e que influenciam diretamente na qualidade do leite;
- Definir e programar sensores para coletar dados;
- Definir e programar o sistema de armazenamento de dados;
- Garantir a sincronização entre dados obtidos e o exato momento da coleta;
- Unificar todos os sistemas em um único código;
- Desenvolver a estrutura para abrigar todos os componentes;
- Mostrar os resultados da implementação do *datalogger* em uma propriedade rural.

2. REVISÃO DE LITERATURA

Neste capítulo serão apresentados conceitos e definições que foram essenciais para a definição da estrutura do *datalogger*, assim como o entendimento dos dispositivos utilizados nos processos de obtenção, armazenamento e visualização dos dados.

2.1 Qualidade do leite e o sistema de refrigeração

O sistema de armazenamento e refrigeração do leite nas propriedades rurais é uma das principais medidas para de garantir a conservação e qualidade microbiológica do leite a fim de evitar sua perda (NERO *et al*, 2005).

De acordo com a Resolução IN 51/2002, do Ministério da Agricultura Pecuária e Abastecimento (MAPA), o sistema de refrigeração deve garantir o resfriamento do leite em até três horas após a finalização de cada ordenha e manter a temperatura igual ou inferior a 4°C. Caso o leite não seja resfriado rapidamente e mantido a uma temperatura menor que 4°, poderá ocorrer o acréscimo da população bacteriana e consequentemente a degradação do leite (BRITO; DIAS, 1998).

2.2 Qualidade da energia elétrica

Um dos principais problemas enfrentados pelos produtores rurais, no quesito energia elétrica, é a sua baixa qualidade. Isso ocorre, essencialmente, devido a grande distância entre as propriedades rurais e os centros de distribuições, resultando em linhas de energia extensas, no qual, consequentemente, interfere na qualidade necessária para o correto suprimento das atividades desenvolvidas pelos produtores rurais (SILVA *et al*, 2002).

Segundo Deckmann e Pomilio (2018), o monitoramento da qualidade de energia elétrica (QEE) é necessário quando acontece interferências no funcionamento regular dos equipamentos elétricos, como vibrações mecânicas ou sobreaquecimento de motores, constante atuação de proteções, variações luminosas em lâmpadas, etc.

A perda da qualidade de energia elétrica pode gerar despesas gigantescas para os consumidores finais (KAGAN, 2009). Essas despesas podem ser a redução da vida útil dos equipamentos, queda de eficiência produtiva de dispositivos e pessoas, mau

funcionamento do sistema, perda de produtos perecíveis, paralisação de serviços essências, entre outros (DECKMANN; POMILIO, 2018).

Como o intuito desse projeto é monitorar as grandezas que afetam diretamente a qualidade do leite, torna-se necessário analisar, pela a equipe de manutenção, como a qualidade de energia elétrica no campo pode interferir na conservação do leite. Para isso iremos registrar a tensão de funcionamento do sistema de refrigeração, assim como a temperatura do compressor (motor responsável pelo resfriamento do leite), visando fornecer informações relevantes (histórico de funcionamento) para o suporte técnico conseguir encontrar fatores que possam estar atrapalhando o correto funcionamento do tanque de leite.

2.3 Datalogger

O *datalogger* é uma combinação de dispositivos eletrônicos que juntos são capazes de registrar dados ao longo do tempo, adaptando-se a cada situação de aplicação, e possibilitando a comunicação com outros sistemas através de softwares para transferência de dados (KUCMANSKI, 2018).

De acordo com Piovesan (2008), o *datalogger* é composto basicamente por:

- microcontrolador para executar cálculos e processar sinais;
- memória interna para armazenar informações;
- módulo RTC (*Real Time Clock*) com bateria interna para associar a exata data e hora do registro;
- sensores para coletar valores das grandezas medidas;
- fonte de energia para funcionamento dos componentes.

Nos próximos itens desse tópico, será referenciado os componentes escolhidos para construir o *datalogger* desse projeto, assim como a função de cada um.

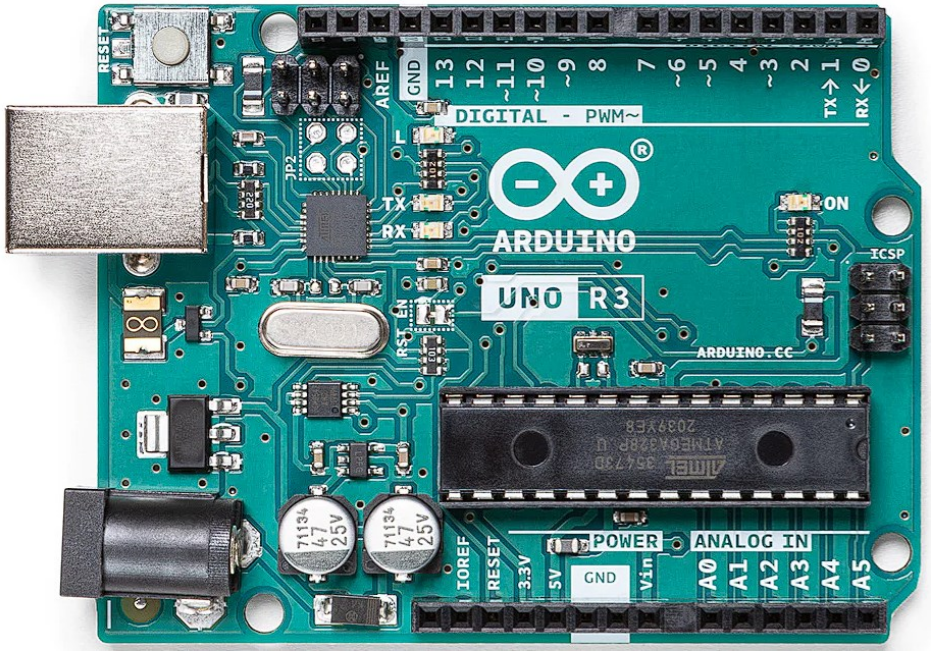
2.4 Microcontroladores e Arduino

O microcontrolador é a integração de um processador com memória e uma diversidade de circuitos embutidos em um único chip para aplicações de controle (GRIDLING; WEISS, 2007). Segundo Ibrahim (2006), esses circuitos embutidos podem

ser conversores de sinais analógico para digital ou vice-versa, receptores de dados digitais, atuadores, temporizadores e lógica de interrupção. Nesse trabalho iremos utilizar o microcontrolador para controlar e tratar os sinais analógicos recebidos dos sensores, sincronizar com a informação de data e hora do módulo RTC, e transmitir esses dados para o módulo de armazenamento para serem salvos.

Existem vários tipos de microcontroladores disponíveis no mercado, entre os principais, destaca-se o Atmel ATmega328P, que é contemplado pela plataforma Arduino. Essa plataforma tem como grande vantagem o desenvolvimento de hardware e software *open-source* associados a um baixo custo de investimento (MCROBERTS, 2015). Devido a sua característica *open-source*, diversas bibliotecas foram desenvolvidas e compartilhadas pelos criados na internet, condição que facilita a utilização e emprego dos mais diferentes tipos de componentes por novos usuários (TORRES et al, 2015). A Figura 1 mostra um microcontrolador da família Arduino.

Figura 1 – Placa de prototipagem eletrônica, Arduino UNO



Fonte: <https://store.arduino.cc/products/arduino-uno-rev3>, acesso em 16/04/2023

A forma de programação e compilação do código é realizada dentro do software oficial da plataforma, que se chama Arduino IDE. Basicamente, o ambiente IDE é composto por duas partes, editor e compilador, no qual no editor é utilizado para

desenvolver o código da aplicação, em C ou C++, e o compilador para transferir o código para o módulo Arduino (FEZARI; AL DAHOUD, 2018).

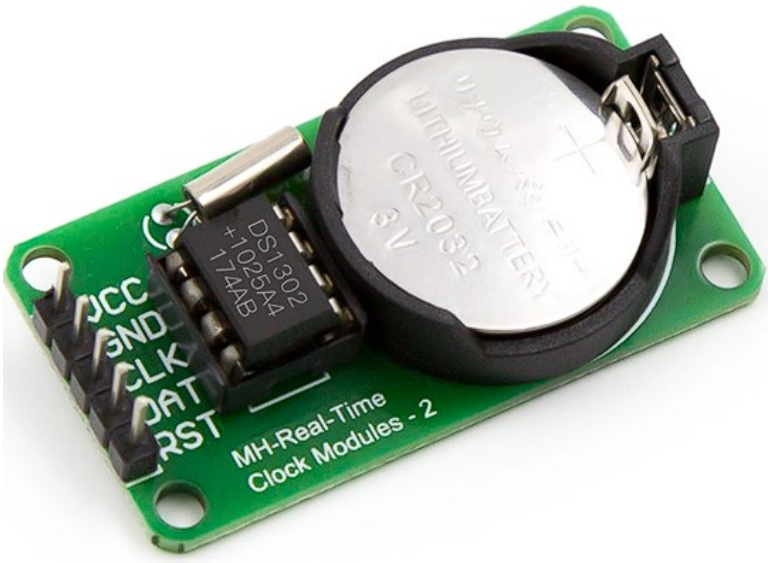
2.5 Módulo RTC DS1302

O módulo RTC DS1302 tem como principal função manter os dados de data e hora atualizados, mesmo com o sistema de alimentação do Arduino desligado ou com uma eventual falta de energia (ROCHA, 2014). Essa função só é possível devido a presença de uma bateria acoplada diretamente no módulo RTC, que na ausência de energia elétrica, utiliza a bateria como fonte de energia para continuar contabilizando as medidas de tempo, garantindo dessa forma a coerência das informações fornecidas (ESTEVES *et al*, 2010). Segundo Torres *et al* (2015), a forma de funcionamento do módulo RTC, pode ser comparada ao funcionamento de um relógio a bateria.

A comunicação do módulo com o Arduino, acontece via a utilização da biblioteca ThreeWire.h (DE ABREU VIEIRA, 2014). Essa biblioteca é responsável por identificar automaticamente, no momento da compilação do código, a atual data e hora do sistema em que o módulo está inserido, passando para o a memória do RTC as novas referências de tempo que ele irá contabilizar a partir daquele momento.

Podemos verificar na Figura 2 um módulo RTC DS1302.

Figura 2 - Módulo RTC DS1302



Fonte: < https://www.baudaeletronica.com.br/modulo-real-time-clock-rtc-ds1302.html>, acesso em 17/04/2023

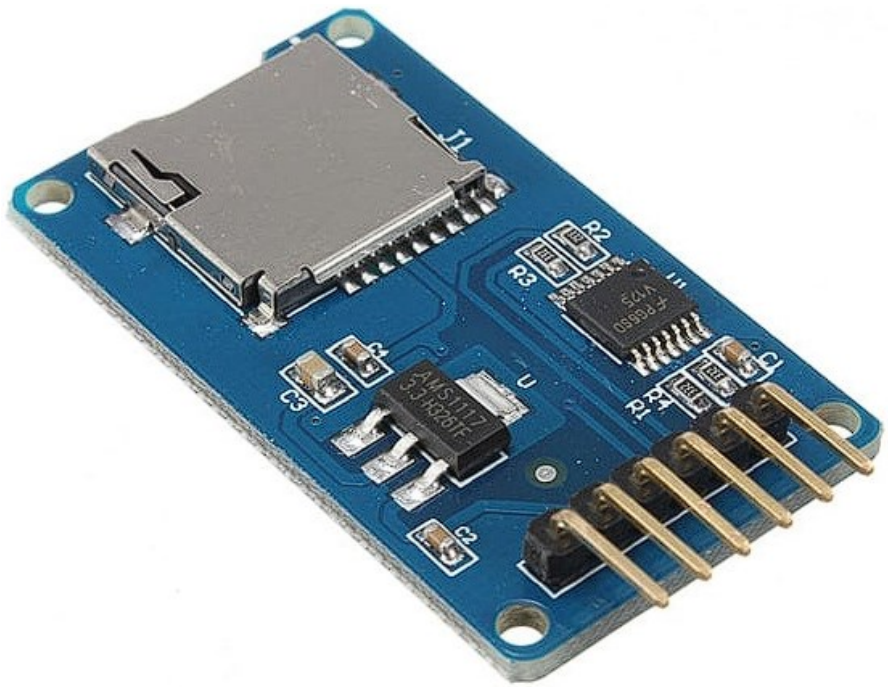
2.6 Módulo cartão microSD 4MD36

Esse módulo tem por objetivo receber e armazenar todos os dados provenientes dos sensores e do módulo RTC, em um cartão de memória microSD, através da sequência de ações definidas na lógica de programação do microcontrolador (TORRES *et al*, 2015).

A comunicação entre o módulo SD e o microcontrolador acontece através da codificação utilizando a biblioteca SD.h (DE ABREU VIEIRA, 2014). Essa biblioteca garante que os dados dos sensores, que foram tratados pelo Arduino, sejam sincronizados com o exata data e hora da coleta, que é fornecido pelo módulo RTC, para que então possam ser gravados em um arquivo no formato de texto dentro do cartão microSD e posteriormente utilizado para a geração de relatórios.

A Figura 3 mostra um módulo cartão SD 4MD36.

Figura 3 - Módulo cartão microSD 4MD36



Fonte: <https://www.makerhero.com/produto/modulo-cartao-micro-sd/>, acesso em 17/04/2023

2.7 Sensor termistor NTC 3950 100K

Com o objetivo de registrar a qualidade do leite ao longo do tempo, torna-se necessário a verificação das grandezas que prejudicam suas propriedades alimentícias,

como a temperatura. Por esse motivo, iremos utilizar o termistor NTC 3950 100K para monitorar a temperatura do leite durante todo o seu processo de armazenagem, assim como a temperatura do compressor do sistema de refrigeração.

Esses sensores NTC (*negative temperature coefficient*) tem como principal característica uma relação inversamente proporcional entre variação da temperatura e resistência, no qual o aumento da temperatura diminui sua resistência e vice-versa (WENDLING, 2010).

Outra característica importante desses termistores é o seu revestimento de vidro envolto por uma película de platina (LINARES, 2021). Devido a essa propriedade, o sensor consegue operar em uma faixa de temperatura de -40ºC a 270ºC, tornando-se ideal para o monitoramento de baixas e elevadas temperaturas.

Também vale ressaltar a compacta dimensão desses sensores, como podemos verificar na Figura 4, aspecto que facilita a utilização e aplicação no campo.

Figura 4 - Termistor NTC 100K

Fonte: <https://www.makerhero.com/produto/termistor-ntc-100k-com-cabo/>, acesso em 18/04/2023

Para a utilização dos termistores com o Arduino, é necessário a aplicação de uma equação que relaciona temperatura e resistência, já que o Arduino não irá receber de

forma direta o valor da temperatura, mas sim a queda de tensão sobre a resistência do termistor, que varia com a temperatura. Devido isso, iremos utilizar a derivação da fórmula de Steinhart-Hart, conhecida como equação do parâmetro β, que é responsável por estabelecer uma relação entre resistência e temperatura dos termistores NTC (GERONYMO, 2017).

Abaixo podemos visualizar a equação do parâmetro beta:

$$\frac{1}{T} = \frac{1}{To} + \frac{1}{\beta} * \ln\left(\frac{Rt}{Ro}\right)$$

Onde:

- T é a temperatura que buscamos medir em Kelvin;
- To é a temperatura ambiente (25ºC) em Kelvin;
- Ro é a resistência do termistor utilizado (100kΩ);
- Rt é a resistência calculada pelo o Arduino;
- ß é um parâmetro específico do componente, fornecido pelo fabricante.

2.8 Sensor tensão ZMPT101B

A tensão elétrica é uma grandeza muito importante e que afeta diretamente a qualidade do leite nas propriedades rurais, pois para o correto funcionamento do sistema de refrigeração, necessita-se de uma rede elétrica estável e de qualidade. Por esse motivo, a utilização de sensores para monitorar a tensão torna-se algo necessário, a fim de acompanhar possíveis causas que possam influenciar na qualidade do leite.

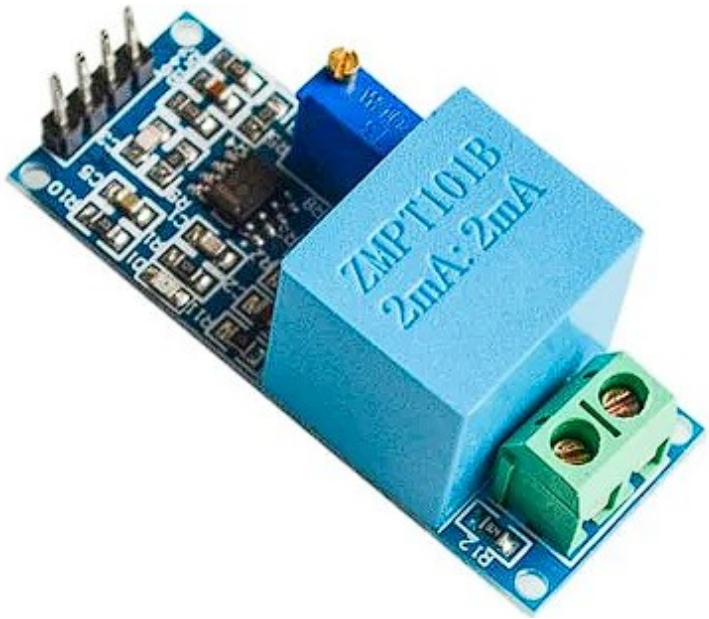
Buscando atender essa necessidade, iremos utilizar o sensor de tensão ZMPT101B para coletar os valores de tensão da rede. Segundo Abubakar (2017), esse sensor apresenta alta precisão e boa consistência para leituras de tensão, podem medir valores AC de 0 a 250V.

O módulo também contempla um transformador de tensão, que tem como função referenciar os valores da rede para os de leitura do Arduino (OLIVEIRA, 2022). A interação do sensor com o microcontrolador ocorre por meio da biblioteca open-source

EmonLib, no qual é a responsável por disponibilizar as funções necessárias para obtenção dos valores de tensão lidos da rede.

Na Figura 5, podemos visualizar o sensor de tensão ZMP101B e notar seu formato compacto e prático.

Figura 5 - Sensor tensão ZMPT101B



Fonte: <https://www.masterwalkershop.com.br/sensor-de-tensao-ac-0-a-250vac-zmpt101b>, acesso em 18/04/2023

2.9 Impressão 3D

Visando garantir eficiência, flexibilidade e economia para o projeto, a estrutura responsável por abrigar os componentes do *datalogger* foi desenvolvida utilizando a impressão 3D, que segundo Volpato (2021), esse recurso é ideal para pequenos volumes de produção.

As impressoras 3D utilizam como princípio de funcionamento a adição sucessiva de material na forma de camadas, no qual a peça em processo de impressão, tem seu modelo computacional 3D cortado, na maioria das vezes horizontalmente, em várias fatias, que empilhadas representam o objeto a ser impresso (VOLPATO, 2021).

Durante o processo de fatiamento, o software utilizado para preparação da impressão, gera um mapa de coordenadas tridimensionais, que define, para a impressora, a sequência e os caminhos que deverão ser seguidos para a impressão de

cada camada (ALCAIDE; WILTGEN, 2018). A fixação do modelo na base da impressora e junção entre camadas, ocorre devido a fundição da matéria-prima (filamento plástico) dentro do conjunto extrusor da impressora, transformando-a em um líquido aquecido, que é expelido e utilizado na forma de um fino cordão durante o processo de impressão das camadas.

Na Figura 6, podemos verificar um exemplo de impressora 3D.

Figura 6 - Exemplo de impressora 3D

Fonte: <https://www.makerhero.com/produto/impressora-3d-creality-ender-3/>, acesso em 19/04/2023

2.10 Power Query e Power BI Desktop

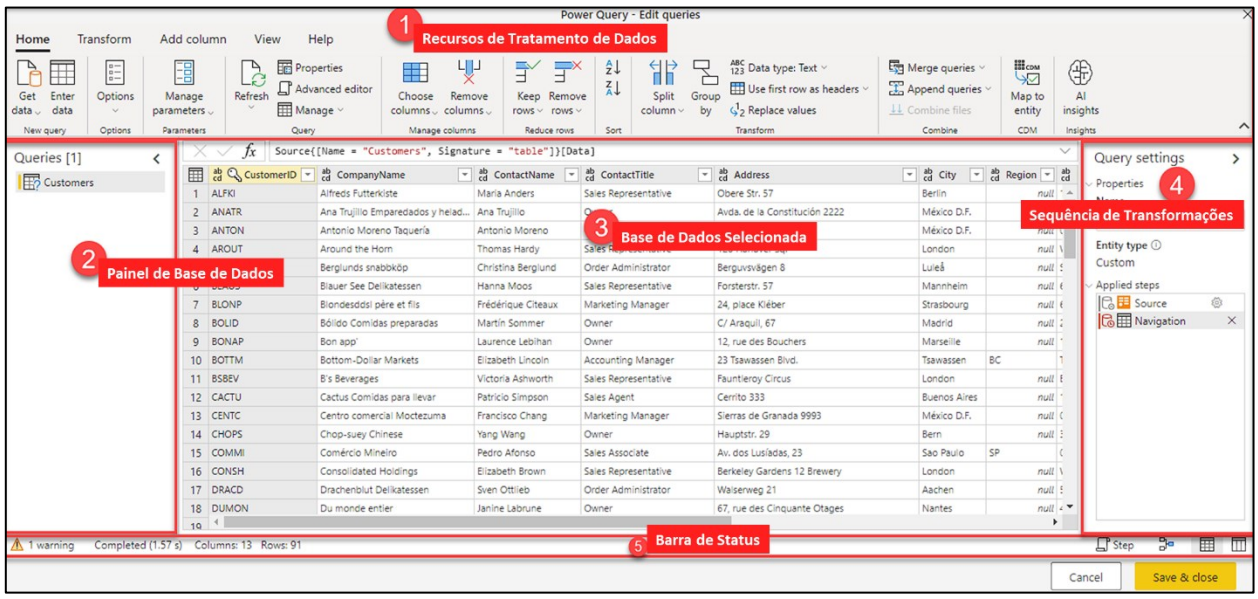
Buscando uma maneira eficiente e sólida para tratar e visualizar os dados registrados pelo *datalogger* ao longo do tempo, optamos pela utilização das ferramentas

Power Query e Power BI Desktop, no qual a primeira será responsável pelo tratamento dos dados e a segunda pela construção dos gráficos que irão interagir com o usuário.

O Power Query é uma ferramenta que está inserida dentro de vários produtos da Microsoft, como o Power BI e Excel. Ela é uma ferramenta que possibilita a conexão com diferentes fontes dados e por meio de uma simples interface, o usuário realiza o tratamento dos dados uma única vez, no qual a ferramenta grava a sequência de transformações e replica o passo a passo de tratamento, todas as vezes que ocorrer uma atualização no relatório que está utilizando essa base (RAVIV, 2018). Tornando-se ideal para o tratamento dos dados do *datalogger*, visto que os dados não terão variação de colunas, mas simplesmente inserção de novas coletas (linhas na tabela) ao longo do tempo.

Na Figura 7, podemos verificar o layout do Power Query:

Figura 7 - Layout do Power Query



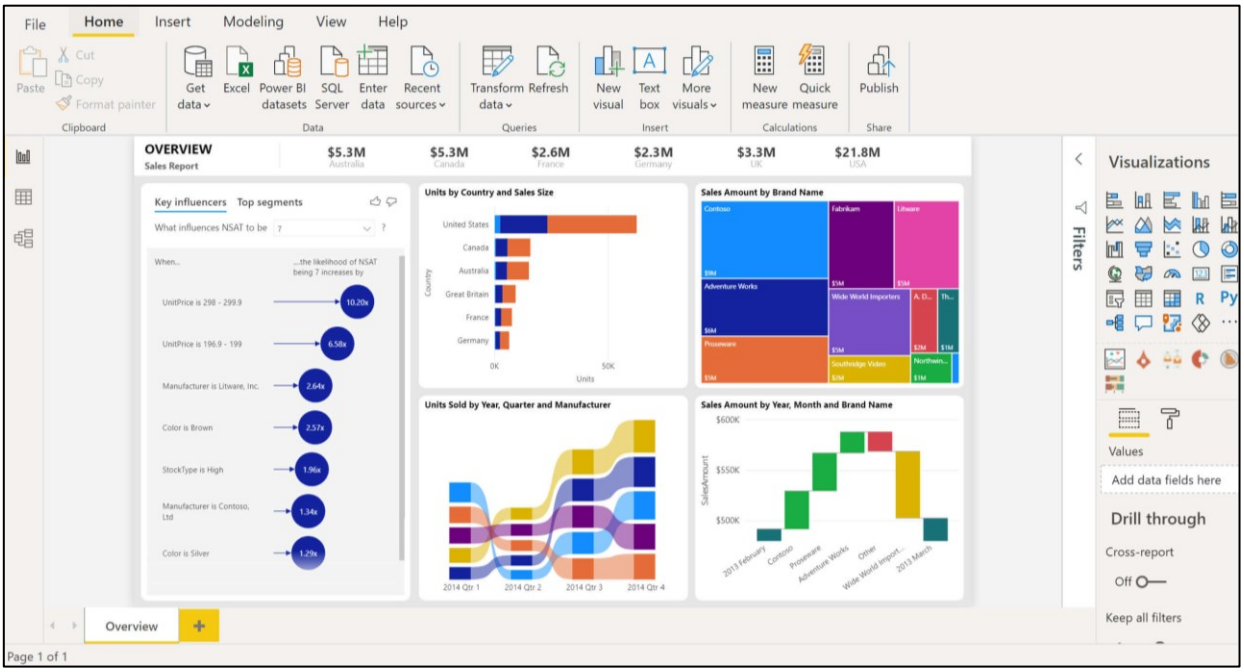
Fonte: <https://powerbi.microsoft.com/pt-br/>, acesso em 20/04/2023

Com o intuito de reduzir os custos de investimento desse projeto, optamos por utilizar o Power BI Desktop para o desenvolvimento dos gráficos baseado nos dados (tratados pelo Power Query) do *datalogger*, pois se trata de uma ferramenta totalmente gratuita e de fácil utilização. Segundo Aspin (2016), o Power BI Desktop é ideal para a criação de métricas e medidas que suportaram no processo de criação de gráficos e

entendimento dos dados, assim como a construção de relatórios objetivos e interativos com o usuário.

Na Figura 8, podemos visualizar um exemplo de relatório desenvolvido no Power BI Desktop.

Figura 8 - Exemplo de relatório criado com o Power BI Desktop



Fonte: <https://powerbi.microsoft.com/pt-br/>, acesso em 20/04/2023

3. METODOLOGIA E DESENVOLVIMENTO

Este trabalho foi desenvolvido utilizando a metodologia de validação e incrementação de recursos, no qual o processo de desenvolvimento consiste na realização de códigos e testes de forma individual, para posteriormente serem integrados em uma única ferramenta.

A estrutura adotada para a construção do *datalogger* será apresentada nos próximos itens da seguinte forma:

- Levantamento dos requisitos da aplicação;
- Estruturação da arquitetura geral do *datalogger*;
- Definição dos componentes do *datalogger*;
- Unificação dos circuitos;
- Construção da estrutura física de armazenamento dos componentes;
- Sistema de visualização de dados;
- Relação de materiais e custos;
- Resultados e discussões.

3.1 Levantamento dos requisitos do *datalogger*

Antes de iniciar o projeto de construção do *datalogger*, foi necessário entender o funcionamento do sistema de armazenamento e refrigeração do leite, assim como o ambiente e recursos disponíveis onde a ferramenta seria implementada, no qual os seguintes pontos e requisitos foram levantados:

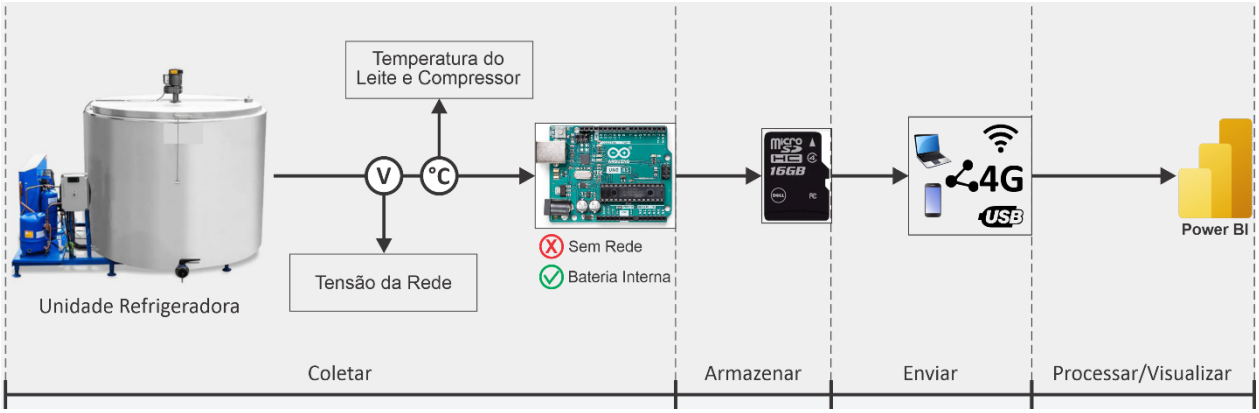
- O sistema precisa funcionar sem depender de conexão à internet, uma vez que não há sinal Wi-Fi, cabo de rede ou rede móvel estável no local de utilização da ferramenta;
- O sistema precisa armazenar informações em um cartão de memória, que suporte longos períodos de gravação de dados;
- O sistema precisa garantir dados precisos de data e hora mesmo durante eventuais quedas de energia;

- Possuir dois sensores capazes de coletar valores de temperaturas entre -3ºC a 150ºC e com pelo menos 1m de comprimento;
- Possuir um sensor com capacidade de coletar valores de tensão AC entre 0V a 250V e com pelo menos 2m de comprimento;
- Garantir a sincronia de tempo entre o momento da captura dos dados pelos sensores com a exata data e hora da coleta;
- Dados serem armazenados em um único arquivo, seguindo uma sequência fixa para o registro das informações coletas;
- Fácil acesso e remoção do cartão de memória, possibilitando a inserção dele em computadores, tablets e celulares, para fins de compartilhamento em zonas com internet ou para visualização dos dados no próprio local de forma offline.
- Construção de uma plataforma digital para visualização dos dados, com interface simples e de fácil utilização.

3.2 Arquitetura geral do projeto

Com base nos requisitos levantados no tópico anterior, foi estruturado a arquitetura geral do projeto, conforme representado pela Figura 9. Através dela foi possível delimitar e entender os objetivos a serem alcançados pela ferramenta, desde do processo de coleta de dados na fazenda até a etapa de visualização pela a assistência técnica, facilitando o processo de construção e desenvolvimento do projeto.

Figura 9 - Arquitetura geral do *datalogger*



Fonte: Autor

Vale ressaltar que o requisito de independência de internet para funcionamento da ferramenta, citado como necessidade no tópico passado, não é considerado como uma opção vantajosa ou simplificadora, mas algo necessário, devido à ausência de infraestrutura ou sinal estável de rede móvel nas pequenas propriedades rurais, condição que aumentaria, excessivamente, o custo da implementação do dispositivo e a sua estabilidade de operação. Mais adiante, esse tema será abordado novamente, trazendo mais considerações.

3.3 Definição dos componentes do *datalogger*

Para a escolha dos componentes desse projeto, tomamos como base orientativa o atendimento dos requisitos levantados, comparando com a disponibilidade no mercado nacional e custo de investimento. A seguir será especificado a opção por cada componente, começando pelos sensores de tensão e temperatura, módulo microSD, módulo RTC e por fim o microcontrolador.

I. Sensor Tensão ZMPT101B

Como a tensão é umas das principais grandezas responsáveis pelo funcionamento do sistema de refrigeração, optamos por escolher um sensor que fosse capaz de realizar seu monitoramento, com o intuito de acompanhar a qualidade da tensão ao longo do tempo, verificando seu nível de oscilação, assim como eventuais quedas ou ausência de energia elétrica no campo.

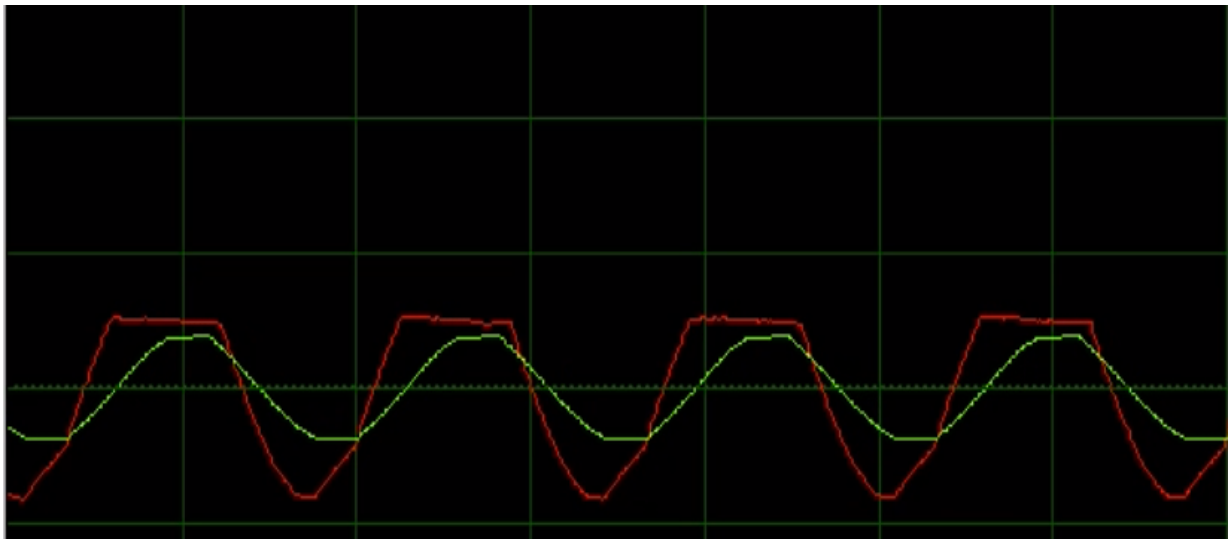
Para realizar esse monitoramento, utilizamos o sensor de tensão ZMPT101B, que é um módulo de alta precisão (±1%) com capacidade de realizar leituras de tensão entre 0 a 250 VAC e até mesmo verificar se a rede está energizada. Como a tensão da rede que iremos aplicar o *datalogger* é de 220 VAC, esse módulo atende perfeitamente nossos requisitos. Além disso, ele consegue operar em temperaturas de -40 ºC a 70 ºC e também é contemplado com um transformador que possui isolamento de 4000V, garantindo segurança para o funcionamento dos outros componentes da ferramenta.

Para garantir a qualidade dos valores amostrados pelo o Arduino, foi necessário ajustar o formato de onda da tensão de saída do sensor utilizando um osciloscópio, visto

que o sensor, na maioria das unidades produzidas, costuma vir com os valores do semiciclo positivo cortados de fábrica.

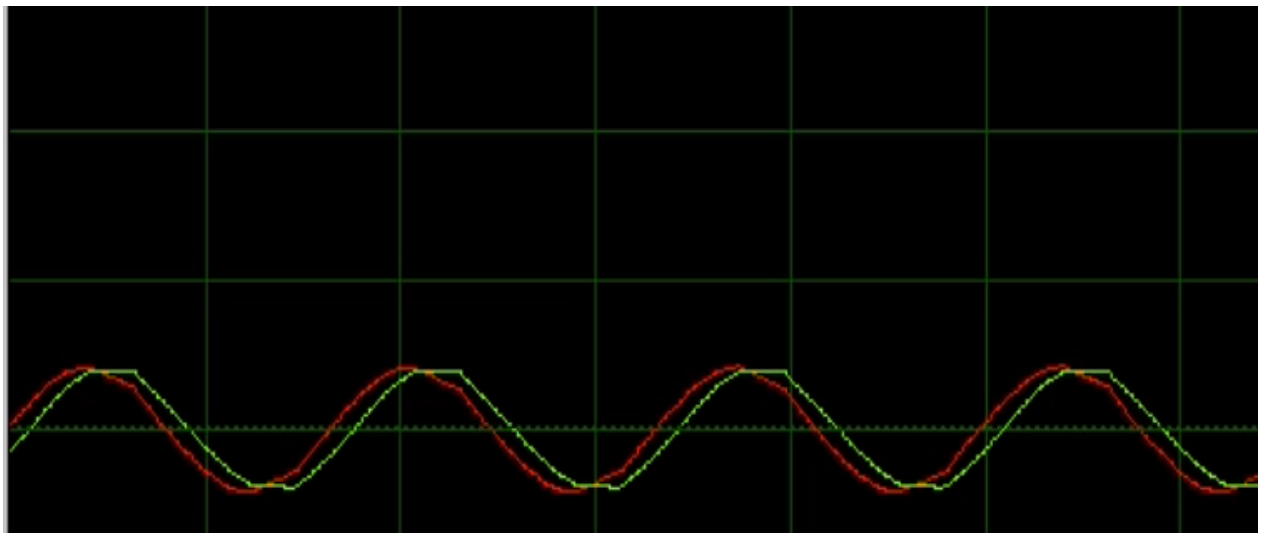
Para realizar o ajuste do formato de onda, criamos um código para o sensor medir valores de tensão comparando esses valores com a leitura de um osciloscópio em paralelo, conforme Figura 10, no qual a linha vermelha representa os valores do sensor e a linha verde do osciloscópio. E por meio do giro do pino *trimpot*, presente no módulo, calibramos o formato da onda, conforme Figura 11, ajustando o semiciclo positivo.

Figura 10 – Captura da forma de onda sem calibração, por um osciloscópio



Fonte: Autor

Figura 11 – Captura da forma de onda com calibração, por um osciloscópio



Fonte: Autor

Para a leitura dos valores de tensão pelo módulo, utilizamos as funções voltage() e calcVI() presente na biblioteca EmonLib.h. É através delas que conseguimos registrar os valores de tensão no *datalogger*.

A validação do código do módulo de tensão, foi realizado através da coleta de dados da rede elétrica e visualização em tempo real pelo serial monitor do Arduino IDE, no qual utilizando um multímetro em paralelo, comparamos os valores obtidos, com o objetivo de certificar a precisão dos valores coletados pelo sensor, garantindo dessa forma a validação do código de leitura de tensão.

II. Termistor NTC 3950 100K

Visando garantir a qualidade do leite e do sistema de refrigeração, a próxima grandeza que escolhemos monitorar é a temperatura de armazenamento do leite e funcionamento do compressor. O intuito do registro dos valores de temperatura de armazenamento do leite, é para verificar se o sistema está sendo capaz de refrigerar e manter a adequada refrigeração, já o monitoramento da temperatura do compressor é essencial devido ao fato de ser o componente mais caro do sistema e o principal responsável pelo funcionamento adequado do sistema.

O leite é inserido nos tanques de armazenamento a uma temperatura média de 36ºC e precisa ser refrigerado a pelo menos 4ºC, já a temperatura de trabalho do compressor gira entre 90ºC a 100ºC. Com base nesses requisitos, buscamos uma opção acessível, de baixo investimento e com disponibilidade no Brasil, chegando no Termistor NTC 3950 100K.

Esse medidor de temperatura, do tipo NTC, suporta leituras entre -40ºC a 270ºC, tornando-se ideal para nossa aplicação, além de um baixo valor de investimento. Seu funcionamento é baseado na variação de sua resistência com a mudança de temperatura, no caso do termistor NTC, o aumento de temperatura causa a diminuição de sua resistência. Devido a essa propriedade, por meio da associação de uma resistência fixa em série com a resistência do termistor, podemos utilizar o Arduino para medir a queda de tensão sobre esse resistor e posteriormente encontrar o valor da resistência do termistor.

Para calcular a temperatura, iremos utilizar a equação do parâmetro ß dentro do código do Arduino, que é uma derivação da fórmula de Steinhart-Hart, para chegar em valores aproximados de temperatura, uma vez que, os gráficos de variação da resistência com a temperatura geram fórmulas complexas para representação das curvas. A fórmula do parâmetro ß está representada no tópico 2.7 deste trabalho.

O nosso valor ß é 3950, conforme especificado pela a Tabela 1 do fabricante do termistor escolhido:

Tabela 1 - Coeficiente ß de Termistores

Part No.	Rated Resistance R_{25} (KΩ)	B Value (25/50°C) (K)	Rated Power(mw)	Dissi. Coef. (mW/ °C)	Thermal time Constant(S)
MF52□□□3100	0.1-20	3100	≤ 50	≥ 2.0 In Still Air	≤ 7 In Still Air
MF52□□□3270	0.2-20	3270			
MF52□□□3380	0.5-50	3380			
MF52□□□3470	0.5-50	3470			
MF52□□□3600	1-100	3600			
MF52□□□3950	5-100	3950			
MF52□□□4000	5-100	4000			
MF52□□□4050	5-200	4050			
MF52□□□4150	10-250	4150			
MF52□□□4300	20-1000	4300			
MF52□□□4500	20-1000	4500			

Fonte: <https://web.archive.org/web/20220121042654/https://www.cantherm.com/wp-content/uploads/2017/05/cantherm_mf52_1.pdf>, acesso em 25/04/2023

Não tivemos que utilizar nenhuma biblioteca especifica para o desenvolvimento dos códigos, mas apenas a função analogRead(), que é incluída na própria base do Arduino, que tem por objetivo realizar leitura dos valores de tensão de portas analógicas. Como o sensor tem uma saída analógica associada ao Arduino, apenas utilizamos essa função para monitorar os valores de tensão dessa porta e obter o valor da resistência do termistor.

Dessa forma, criamos um código para validação e calibração dos sensores, no qual se baseia na amostragem de valores de temperatura e escrita no serial monitor do Arduino IDE, para que pudéssemos comparar, em tempo real, os valores informados pelos sensores com a leitura em paralelo de um termômetro infravermelho.

III. Módulo Cartão microSD 4MD36

A principal função do *datalogger* é armazenar dados, partindo desse princípio, escolhemos o módulo cartão microSD 4MD36 para ser o componente responsável por registrar as informações dos sensores. A sua escolha foi orientada pela disponibilidade no país, baixo custo de investimento e capacidade física de armazenamento de informações.

Como iremos registrar dados em propriedades rurais, tivemos que escolher um módulo com compatibilidade com cartões de memória de grande armazenamento, uma vez que, devido a distância entre produtor rural e assistência técnica, o *datalogger* pode passar dias ou meses registrando dados, sem que o suporte técnico precise ser acionado, gerando um grande banco de dados e consequentemente um espaço na memória considerável.

De acordo com o fornecedor, esse módulo consegue trabalhar com cartões de memória SD de 2GB até 32GB. Na tabela 2, podemos verificar a relação de tempo de armazenamento, período de amostragem e tamanho do cartão. Nesse projeto foi adotado um cartão de memória com tamanho de 4GB e frequência de 12 amostragens por minuto, garantindo uma capacidade de armazenamento de 545 dias seguidos, sem a necessidade de formatar o banco de dados.

Uma das grandes vantagens desse componente referente a outros módulos, é ser contemplado por um divisor de tensão interna, que permite o funcionamento direto da placa em 5V (tensão de funcionamento do Arduino), simplificando a elaboração e montagem do circuito.

Tabela 2 - Duração do tempo de armazenamento do cartão SD pela amostragem

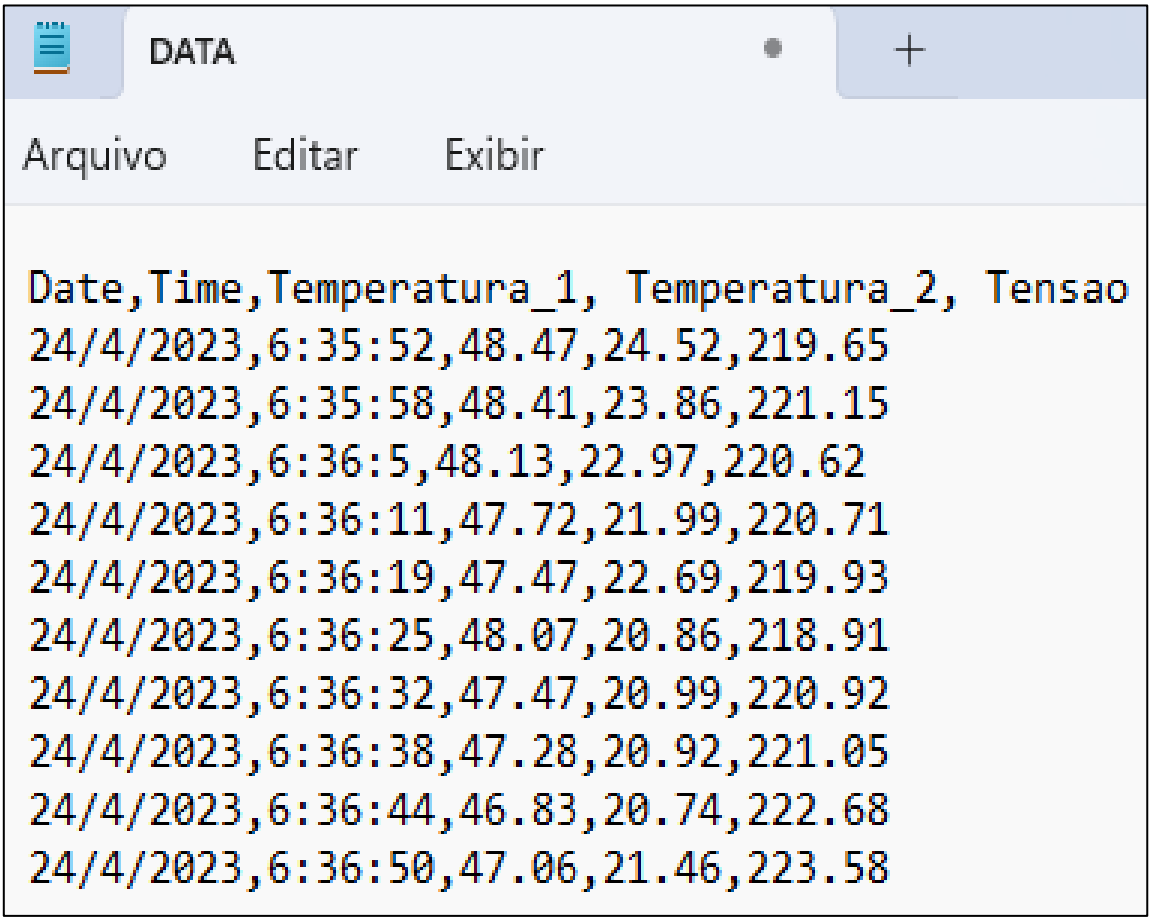
MicroSD [GB]	Amostragem por Min	Bytes por Amostragem	Quantidade de Amostragens [Un]	Tempo de Funcionamento [Dias]	Tempo de Funcionamento [Mês]
2	4	38	14.128.182	2.453	82
2	6	38	9.418.788	1.090	36
2	8	38	7.064.091	613	20

2	10	38	5.651.273	392	13
2	12	38	4.709.394	273	9
4	4	38	28.256.364	4.906	164
4	6	38	18.837.576	2.180	73
4	8	38	14.128.182	1.226	41
4	10	38	11.302.546	785	26
4	12	38	9.418.788	545	18
8	4	38	56.512.728	9.811	327
8	6	38	37.675.152	4.361	145
8	8	38	28.256.364	2.453	82
8	10	38	22.605.091	1.570	52
8	12	38	18.837.576	1.090	36
16	4	38	113.025.455	19.622	654
16	6	38	75.350.303	8.721	291
16	8	38	56.512.728	4.906	164
16	10	38	45.210.182	3.140	105
16	12	38	37.675.152	2.180	73
32	4	38	226.050.910	39.245	1308
32	6	38	150.700.607	17.442	581
32	8	38	113.025.455	9.811	327
32	10	38	90.420.364	6.279	209
32	12	38	75.350.303	4.361	145

Fonte: Autor

Para elaboração do código foram utilizadas as bibliotecas abertas SPI.h e SD.h, no qual disponibilizam funções necessárias para criação de um arquivo texto (.CSV), escrita e registro dentro do cartão de memória. Na Figura 12, podemos visualizar o arquivo “.CSV” gerado pelo código do *datalogger*, o qual está mostrando uma pequena amostra de dados coletados. Nela também podemos notar a forma de distribuição de dados, no qual a leitura de cada componente está sendo separada por uma “,” e seguindo uma sequência fixa de salvamento, gerando toda vez uma nova linha no fim de cada coleta.

Figura 12 - Exemplo de formato de arquivo CSV salvo pelo *datalogger*



Fonte: Autor

IV. Módulo RTC DS1302

Um dos principais requisitos para funcionamento do *datalogger* é a sincronização dos dados coletados com a exata data e hora do registro. Sem essa informação seria impossível estabelecer e garantir uma ordem cronológica das amostras coletas.

Para o atendimento desse requisito, escolhemos o Módulo RTC DS1302, que tem como função primária informar a exata data e hora do momento, podendo ter seu funcionamento comparado a de um relógio. Esse componente é contemplado por uma bateria interna de 3V e também por um sistema modulador, que é capaz de detectar a falta de energia e alterar a sua fonte de alimentação para a bateria, com a finalidade de preservar as informações de data e tempo.

O módulo RTC DS1302 também consegue identificar e ajustar automaticamente a transição dos meses com 28, 29 (ano bissexto), 30 e 31 dias, assim como trabalhar com horas no formato de 12 AM/PM ou 24 horas.

A tensão de trabalho desse componente pode ser feita entre 2 a 5 VDC, condição que elimina a necessidade de utilização de divisores de tensão, facilitando a montagem e complexidade construtiva do *datalogger*.

Para o desenvolvimento e validação do código, utilizamos as bibliotecas ThreeWire.h e RtcDS1302.h, as quais disponibiliza as funções necessárias para obter a data e hora do momento da compilação, ou seja, no momento de carregamento do código para o Arduino, a função RtcDateTime(), busca no sistema compilador as informações de tempo daquele momento, sendo essa informação salva em sua memória e usada como referência para os futuros registros.

V. Microcontrolador e Arduino Uno

A escolha da placa controladora foi determinada pelos 4 critérios descritos abaixo:

- Disponibilidade de portas digitais e analógicas, sendo 9 e 3, respectivamente;
- Compatibilidade com os componentes utilizados;
- Dimensões físicas compactas;
- Disponibilidade em território nacional.

Analisando esses critérios e buscando no mercado nacional, o modelo disponível de placa de prototipagem que mais se encaixou em nossa aplicação foi o Arduino UNO, que é contemplado pelo microcontrolador ATmega328. O ATmega328 possui 14 entradas/saídas digitais, sendo que 6 podem ser usadas como saída PWM, 6 entradas analógicas, tensão de operação de 5V, conversor analógico-digital de 10 bits, 68,6 mm de comprimento e 53,4 mm de largura (ARDUINO, 2023).

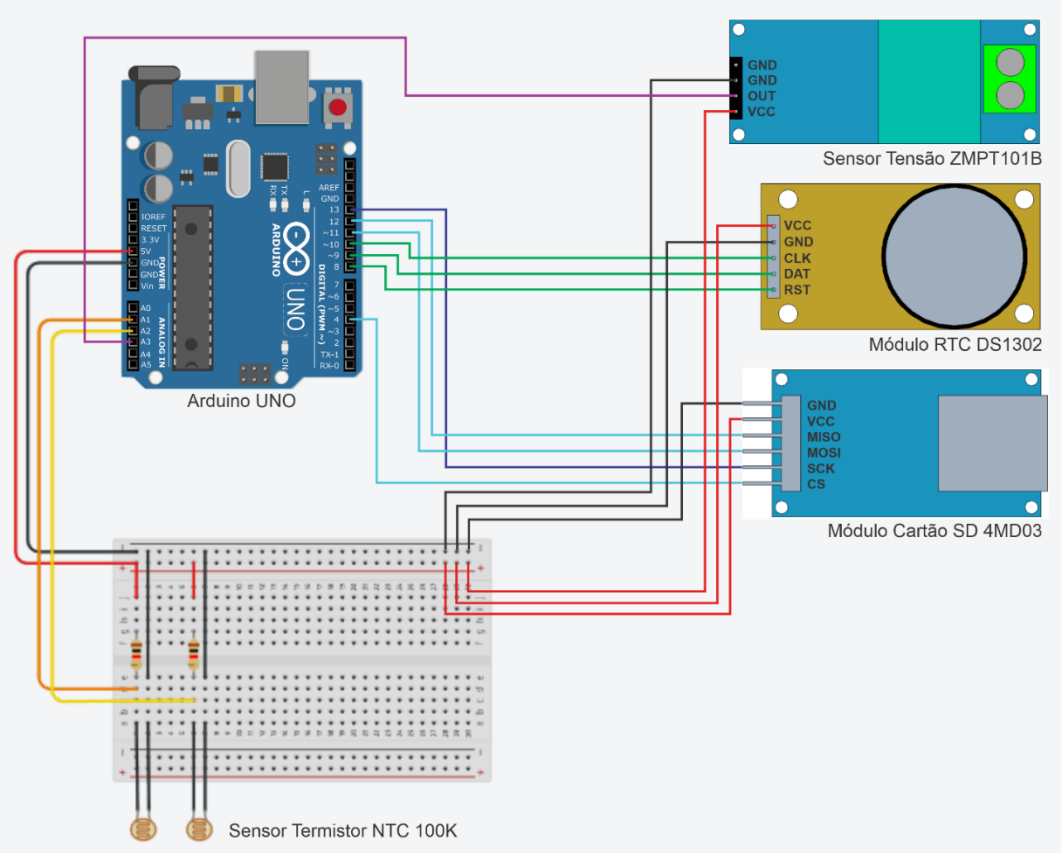
Outro ponto que pesou e justificou a escolha da placa do Arduino UNO, foi o fato de ser uma plataforma de código aberto, no qual já existe, disponível na própria comunidade do Arduino, diversas bibliotecas e vários projetos de outros usuários. Além

de tudo essa placa apresenta um baixo custo de investimento, combinado com a fácil interação entre hardware e software por meio do programa do Arduino IDE.

3.4 Unificação dos circuitos

Passado as etapas individuas de definição e validação dos códigos de cada componente, iremos mostrar nesse item a unificação de todos os circuitos, cujo o código pode ser encontrado no apêndice desse trabalho, apresentando o esquema elétrico final, representado pela Figura 13, aplicado para a montagem do *datalogger*.

Figura 13 - Circuito unificado do *datalogger*



Fonte: Autor

Um ponto importante a ser lembrado e que não foi tratado no tópico de definição de componentes, é a utilização de componentes genéricos na montagem dos circuitos, como os cabos elétricos, os resistores e o protoboard, já que se tratam de itens necessários para a construção de qualquer projeto e não especificamente o desse trabalho.

As portas utilizadas para a conexão entre o Arduino Uno e os componentes foram definidas diretamente no código desenvolvido para a construção do *datalogger*.

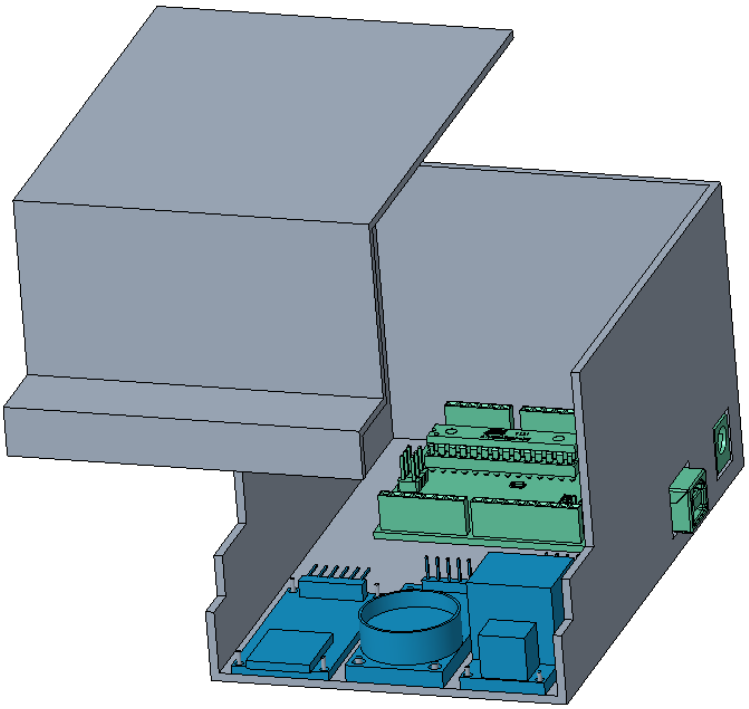
3.5 Construção da estrutura física de armazenamento dos componentes

Com o circuito unificado definido, partimos para o processo de desenvolvimento da estrutura física responsável por armazenar todos os componentes. O layout da estrutura foi dimensionado visando garantir a melhor forma de acessibilidade aos componentes e compactação possível.

Por se tratar de uma estrutura compacta, não enfrentamos dificuldades com as dimensões máximas permitidas pela a impressora 3D, nem com o seu nível de resolução de 0,1mm, cabendo como limitação somente a forma de realizar a impressão, no qual tivemos que dividir o item em duas partes, base e tampa, para que o dispositivo conseguisse imprimir os itens.

Na Figura 14, podemos visualizar o modelo tridimensional da estrutura de armazenamento, contemplando a disposição dos componentes, enquanto que na Figura 15, podemos verificar a estrutura física depois da impressão.

Figura 14 - Estrutura de armazenamento no software de modelagem



Fonte: Autor

Figura 15 - Estrutura impressa do *datalogger*

Fonte: Autor

3.6 Sistema de visualização de dados

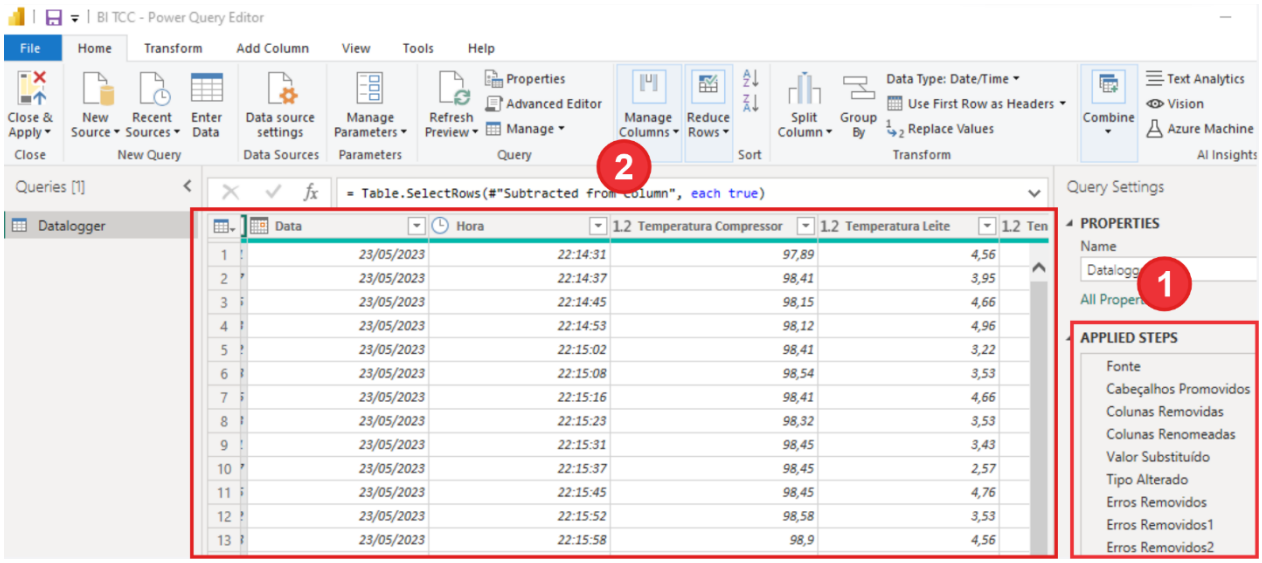
Buscando garantir a visualização dos dados coletados da forma mais prática e simplificada possível, foi desenvolvido um painel de gráficos no Power BI Desktop, vinculando como base de dados para a geração de conteúdo, o arquivo “CSV” gerado pelo *datalogger*.

Para isso, utilizamos o Power Query, ferramenta que é contemplada dentro do próprio Power BI Desktop, para realizar o vínculo e tratamento dos dados. Nessa ferramenta definimos uma sequência de etapas para adequação dos dados, já que no formato “CSV” não existe colunas, mas caracteres específicos indicando a separação de valores, conforme já observamos na Figura 12. Nela também conseguimos ajustar os tipos de dados, filtrar erros de coletas, gerar colunas condicionais, etc.

O grande diferencial dessa ferramenta é que uma vez realizado o tratamento dos dados, todas as vezes que ocorrer atualizações ou substituição do arquivo vinculado, desde que não aconteça alteração no diretório do registro, nem no formato de disposição das colunas ou inserção de novos parâmetros no arquivo (situação que não acontece com os dados armazenados pelo *datalogger*), o software irá replicar os passos definidos pelo o usuário no momento da criação do arquivo do Power BI.

Na Figura 16, podemos verificar os dados já tratados no ambiente do Power Query, no qual está destacado pelo numerador 1, a área responsável por guardar o histórico de passos aplicados para o tratamento dos dados e pelo o numerador 2, o resultado dos ajustes realizados no arquivo CSV fornecido pelo *datalogger*.

Figura 16 - Dados do *datalogger* tratados no Power Query

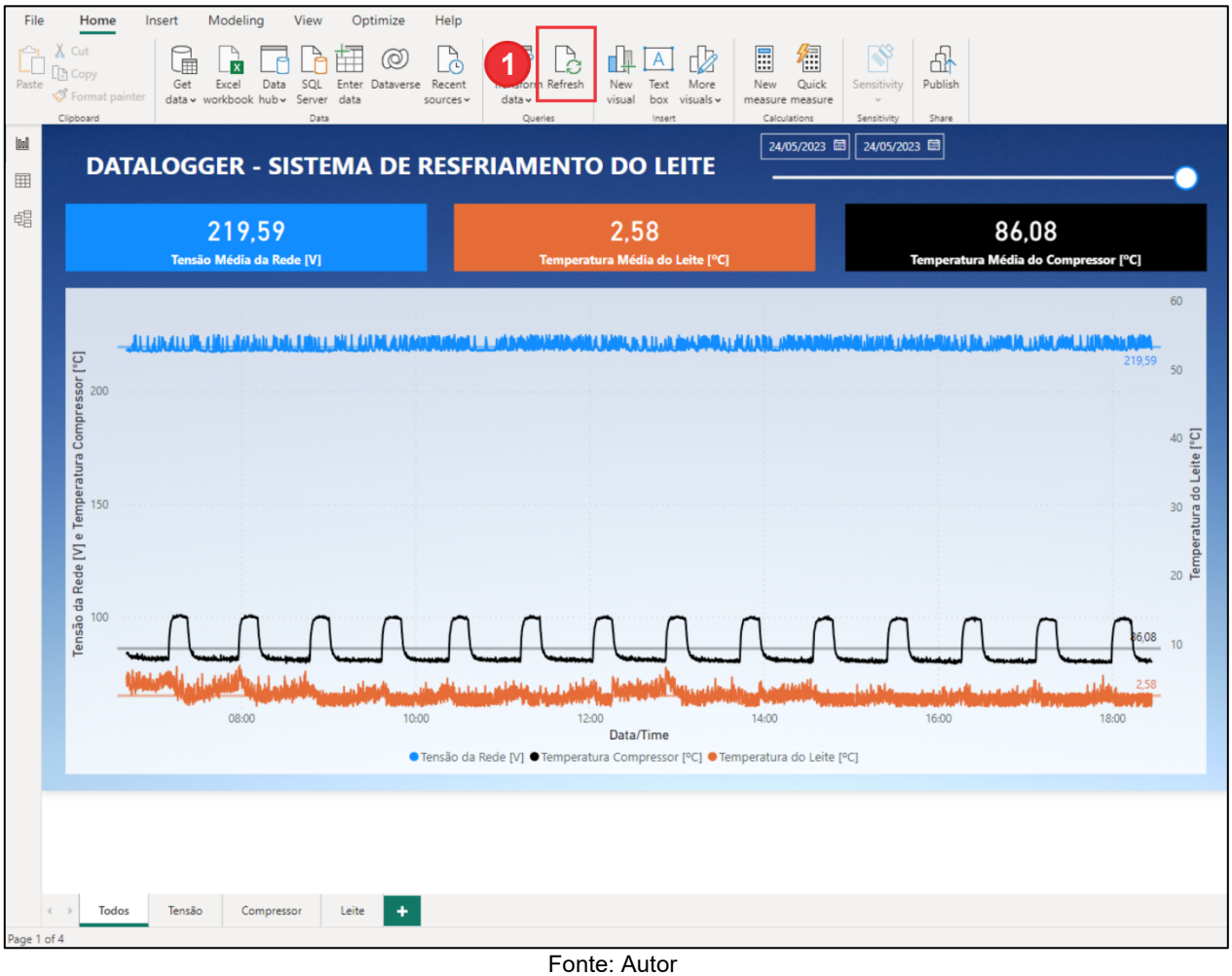


Fonte: Autor

Realizado o processo de tratamento dos dados, partimos para a etapa de criação dos gráficos, recursos que serão fundamentais para a análise e entendimento dos dados coletados. No Power BI Desktop, não enfrentamos dificuldades ou necessidade de ajustes dos dados para a construção dos gráficos, focando os esforços, basicamente, na construção do layout e definição de cores dos painéis. Na Figura 17, podemos visualizar o layout da tela de interação dos usuários com os dados armazenados pelo o *datalogger*, no qual segue destacado pelo o numerador 1, o botão que é responsável por atualizar,

identificar e refletir as alterações no banco de dados vinculado, automaticamente, sem a necessidade de uma nova tratativa de dados no Power Query.

Figura 17 - Dados do *datalogger* representado em gráficos no Power BI



Fonte: Autor

3.7 Relação de materiais e custos

Com relação a gastos e investimentos, a Tabela 3 apresenta o custo unitário e totalizado do projeto. Todos os componentes e materiais necessários para a construção do *datalogger*, são encontrados em território nacional. Os custos informados na tabela estão referenciados para a moeda brasileira.

Nesse projeto, somente o custo da impressão 3D foi tratado como um serviço realizado por terceiro, já os matérias tratados como genéricos, são os cabos elétricos, resistores, protoboard e cola quente.

Vale ressaltar que os softwares (Arduino IDE, Power Query e Power BI Desktop) utilizados para o desenvolvimento do código e criação do sistema de visualização de dados, são totalmente gratuitos, sem a necessidade de compra de licença.

Tabela 3 - Relação de custos por componentes e materiais

Descrição	Quantidade	Custo Unitário	Custo Final
Sensor de tensão ZMPT101B	1	R$ 22,90	R$ 22,90
Termistor NTC 3950 100K c/ cabo 2 m	2	R$ 9,50	R$ 19,00
Módulo cartão microSD Card 4MD36	1	R$ 8,45	R$ 8,45
Cartão microSD 4GB	1	R$ 14,99	R$ 14,99
Módulo RTC DS1302	1	R$ 9,49	R$ 9,49
Arduino Uno	1	R$ 62,91	R$ 62,91
Fonte de alimentação Arduino 9V	1	R$ 19,99	R$ 19,99
Materiais genéricos	1	R$ 20,00	R$ 20,00
Custo impressão 3D	1	R$ 30,00	R$ 30,00
		TOTAL	**R$ 207,73**

Fonte: Autor

4. RESULTADOS E DISCUSSÕES

Neste capítulo será exposto os resultados e desafios encontrados durante a aplicação do *datalogger* nas pequenas propriedades rurais, tratando desde do desempenho da ferramenta até a relação com os produtores e assistência técnica.

4.1 Qualidade dos dados

Neste item iremos apresentar os valores capturados pelo *datalogger* durante um intervalo de tempo, com a finalidade de analisar a qualidade e comportamento das curvas de cada sensor.

Na Figura 18, podemos observar as curvas de todos os sensores, no qual a temperatura do leite está representada pela cor laranja, a temperatura do compressor pela cor preta e a tensão da rede pela cor azul. Os dados são referentes o funcionamento de uma unidade refrigeradora no dia 24 de abril de 2023, das 08:15 às 17:45.

Figura 18 - Representação gráfica de dados registrados pelo *datalogger* em uma propriedade rural

Fonte: Autor

Analisando o gráfico, podemos verificar os momentos de funcionamento do compressor pelos vales e picos na sua curva de temperatura, em que os pontos baixos

mostram os momentos que o compressor entra em alívio e os pontos altos quando está operando em potência máxima, respeitando a lógica esperada, que seria funcionar somente quando a temperatura do leite estiver acima da definida no seu programador, onde trabalha até reduzir ao ponto definido e depois desliga, criando um ciclo de funcionamento baseado no monitoramento da temperatura do leite. Condição que também podemos verificar na curva do leite, que durante os momentos de alivio do compressor, os valores de temperatura tendem a subir e durante o funcionamento do compressor, os valores estão diminuindo, seguindo a lógica de operação esperada.

Quando analisamos a curva dos valores de tensão, não conseguimos tirar muitas informações, mas somente entender que foi um dia normal de operação, com tensão média de 219,66 V e sem oscilações críticas ou quedas de energia.

Para analisar a precisão e qualidade do sensor de tensão, realizamos o chaveamento do disjuntor geral do circuito de alimentação, com o objetivo de visualizar nos gráficos, os momentos de desligamento e energização, conforme a Figura 19.

Figura 19 - Representação gráfica de dados registrados pelo *datalogger* no momento de chaveamento do disjuntor geral

Fonte: Autor

Analisando a figura 19, podemos verificar a precisão do sensor de tensão, no qual conseguiu detectar todos os chaveamos efetuados e transmitir os valores para o *datalogger* realizar o armazenamento. Tornando-se ideal para a nossa aplicação, pois será capaz de detectar desarmes de disjuntor, falta de energia e oscilação de tensão, fornecendo o registro de informações essenciais para o entendimento do funcionamento do sistema de refrigeração e até mesmo da rede elétrica.

Portanto, podemos entender que os sensores se comportaram de maneira esperada e de forma sólida, garantindo segurança e precisão dos dados registrados pelo *datalogger*. Assim como, conseguimos verificar a capacidade de armazenamento de dados pelo módulo microSD, que conseguiu gravar a uma taxa de amostragem alta, 1 a cada 5 segundos. Condições que satisfazem os requisitos definidos no início do projeto, validando a aplicação dos componentes nesse trabalho.

4.2 Estrutura física

O layout da estrutura física de armazenamento dos componentes teve como objetivo garantir a organização compacta dos dispositivos, combinado a um bom nível de acessibilidade aos sensores, cartão microSD e alimentação elétrica, com a finalidade de facilitar a instalação e utilização da ferramenta no campo.

A estrutura ficou com 126 mm de comprimento, 84 mm de largura e 70 mm de altura, sendo a sua base em formato retangular. As compactas dimensões e formato da base foram essenciais para facilitar a fixação e manuseio da ferramenta no local da aplicação, permitindo seu posicionamento em vários tipos de superfícies e lugares estreitos, garantindo dessa maneira a melhor passagem para os cabeamentos dos sensores e de energia.

Na Figura 20, podemos visualizar a estrutura final e noção de tamanho do *datalogger* comparado a unidade de refrigeração do leite, assim como o seu posicionamento temporário sobre o condensador do sistema de refrigeração, durante as etapas de testes da ferramenta.

Durante as fases de validação, o único problema que dificultou a instalação da ferramenta em algumas propriedades, foi o comprimento do cabo do sensor de

temperatura, no qual estava contemplado inicialmente no projeto, 1 m de comprimento, sendo necessário a substituição por um cabo de 2 m. Essa necessidade aconteceu pelo fato de que cada sensor de temperatura vai para um ponto especifico com sentidos totalmente contrários, sendo um colocado dentro do tanque de leite e o outro fixado na carcaça do compressor, condição que dificultou o posicionamento da ferramenta. Os demais cabos foram dimensionados de maneira correta, sem necessidade de realizar ajustes ou alterações.

Figura 20 - *Datalogger* coletando dados de uma unidade refrigeradora

Fonte: Autor

4.3 Extração dos dados coletados

Nesse projeto, o procedimento adotado para a extração de dados do *datalogger* nas pequenas propriedades rurais, se baseia na retirada do cartão de memória e transferência do arquivo para o dispositivo que irá processar os dados no Power BI Desktop. A transferência pode acontecer de duas maneiras, a primeira seria realizada pela a assistência técnica na própria propriedade e a segunda seria realizada pelo próprio produtor rural, que utilizando um dispositivo com acesso a internet, dispararia o arquivo para o suporte técnico.

É claro que para os atuais níveis de tecnologia disponível, seria muito mais prático e eficiente um sistema conectado a internet, porém nas pequenas propriedades rurais visitadas e utilizadas para a validação da ferramenta, não existia infraestrutura de cabos de internet e nem sinal estável de rede no local da unidade refrigeradora, que na maioria das vezes estão situadas a metros de distância das casas dos produtores, lugar que, em alguns sítios, possui conexão via satélite à internet. É devido a essa característica especifica das pequenas propriedades agrícolas, que adotamos o desenvolvimento do *datalogger* independente de internet, atrelando a extração de dados via retira do cartão de memória e transferência dos dados.

Conforme os requisitos definidos no tópico 3.1, podemos dizer que o sistema de extração de dados atendeu os pontos levantados, entretanto em algumas propriedades rurais, tivemos um pouco de dificuldade por parte dos agricultores no processo de retirada do cartão de memória do *datalogger* e disparo do arquivo para o suporte técnico, sendo necessário a realização de um treinamento de envio de dados para a capacitação do produtor rural.

Outro ponto a ser levantando, é a velocidade de transferência do arquivo do *datalogger* para a unidade responsável pelo o processamento dos dados, que em todos os testes ocorreu de maneira muito rápida. Isso se justifica pelo fato de que para a formação de grandes bancos de dados, conforme informado na Tabela 2, seria necessário um tempo significativo, tornando difícil os casos de propriedades que gerariam grandes bancos sem que a assistência técnica fosse acionada pelo menos uma vez para uma eventual manutenção corretiva ou preventiva.

4.4 Sistema de visualização dos dados

O sistema desenvolvido para a visualização dos dados coletados pelo *datalogger* teve como objetivo garantir a transformação dos dados registrados em recursos visuais, que facilitariam o entendimento das informações.

Para alcançar esse objetivo, foi utilizado o Power Query, que é um recurso interno do Power BI Desktop, para receber e processar os dados registrados. Com esse recurso, desenvolvemos uma sequência de etapas para realizar o tratamento dos dados, que irá se repetir todas as vezes que o usuário da ferramenta solicitar a atualização do banco de dados vinculado.

Em todos os testes realizados, o Power Query conseguiu tratar e carregar os dados de maneira rápida e efetiva, eliminando por completo a necessidade de retrabalho por parte do usuário. A velocidade dos processos de atualizações foi excelente, até mesmo para os grandes bancos de dados. Referente a qualidade e fluidez dos gráficos dentro do Power BI Desktop, todas as medidas e painéis tiveram respostas rápidas e coerentes. Em relação a utilização da ferramenta pela a equipe de manutenção, não tivemos problemas, pois a interface ficou bem intuitiva, didática e de fácil utilização, como podemos notar na Figura 17.

Portanto, podemos dizer que o Power BI Desktop cumpriu todos os requisitos levantados, tornando-se uma aplicação ideal para o projeto.

5. CONCLUSÃO

Esse trabalho teve por objetivo desenvolver e implementar um sistema simples e de fácil utilização capaz de realizar o monitoramento dos equipamentos de armazenamento e refrigeração do leite a um baixo custo de investimento. O sistema desenvolvido pode ser dividido em duas partes, sendo a primeira responsável pelo o armazenamento dos dados e a segunda pela a interpretação das informações registradas.

Analisando a primeira parte do projeto, verificamos que o *datalogger* foi capaz de atender todos os objetivos e requisitos levantados, em que todos os sensores utilizados foram capazes de coletar e transmitir dados coerentes com a expectativa de funcionamento do sistema de refrigeração, assim como os componentes responsáveis pelo o armazenamento das leituras, que conseguiram trabalhar de maneira eficiente, com uma taxa de amostragem de um registro a cada cinco segundos, e também possibilitar o armazenamento de grandes quantidades de dados por longos períodos de funcionamento, sem a necessidade de intervenção do usuário.

Referente a segunda parte do projeto, constatamos que o sistema de tratamento e visualização dos dados, se mostrou efetivo em sua função, pois em todas as etapas, apresentou rápidos resultados de carregamento e tratamento das informações, independentemente do tamanho do arquivo, assim como garantiu a construção de uma simples interface de interação com o usuário. Interface que foi responsável por transmitir com qualidade e simplicidade o histórico de funcionamento registrado pelo o *datalogger*, cumprindo a sua função de auxiliar a assistência técnica nos processos de diagnósticos do sistema de refrigeração.

No que diz respeito ao custo de implementação do sistema de monitoramento, como o software escolhido para o processamento dos dados é totalmente gratuito, o valor do projeto se resumiu a compra de componentes eletrônicos para a construção do *datalogger*. O investimento final ficou a um valor acessível, entretanto devido a mínima escala de produção adotada nesse trabalho, gerou-se acréscimos nos valores de cada

componentes, no qual a aquisição de quantidades maiores, podem reduzir o custo final de investimento por unidade.

A partir das considerações levantadas ao longo desse trabalho, conseguimos concluir que a aplicação dos *datalogger* no monitoramento dos sistemas de armazenamento e refrigeração de leite nas pequenas propriedades rurais, atendeu todos os requisitos definidos no escopo do projeto, onde todas as etapas construtivas da ferramenta apresentaram resultados satisfatórios, desde do processo de leitura das grandezas até a visualização das informações.

5.1 Projetos Futuros

Para trabalhos futuros algumas melhorias podem ser aplicadas nesse projeto:

- Incorporação de um display para visualização dos valores das grandezas em tempo real, juntamente com a inclusão de uma buzina e dispositivos luminosos para sinalizar valores fora dos limites permitidos. Essas melhorias visam facilitar o entendimento do produtor rural, servindo como base orientativa para acionar o suporte técnico. Além disso, funcionaria também como referência para a própria equipe de manutenção durante as visitas técnicas.
- Inserção de um módulo Bluetooth na estrutura do *datalogger*, para se comunicar com outros aparelhos eletrônicos e transferir os dados via Bluetooth. Condição que facilitaria a extração de dados, sem a necessidade de retirar o cartão de memória.
- Inserção de mais sensores na estrutura do *datalogger*, como o de pH, que possibilitaria mais análises referente a qualidade do leite, e o de corrente elétrica, que viabilizaria estudos de consumo de energia, influência no aquecimento de equipamentos e qualidade da energia.
- Desenvolvimento do código e inserção de módulos que possibilite a comunicação do *datalogger* com a internet, para os casos de propriedades rurais que apresentem sinais estáveis de rede. Assim como a adequação do Power BI Desktop para se vincular ao banco de dados que se formaria na nuvem. Condição que possibilitaria a visualização dos dados em tempo real, evitando o processo manual de transmissão de dados. Todavia, esse estudo pode requisitar altos

investimentos, comprovação do nível de estabilidade da rede naquela região e troca da licença gratuita do Power BI por uma paga, que seria capaz de realizar atualizações automaticamente na nuvem.

REFERÊNCIAS

ABUBAKAR, I. et al. Calibration of ZMPT101B voltage sensor module using polynomial regression for accurate load monitoring. Journal of Engineering and Applied Sciences, v. 12, n. 4, p. 1077-1079, 2017.

ALCAIDE, Eduard; WILTGEN, Filipe. Estudo das tecnologias em prototipagem rápida: passado, presente e futuro. 2018.

ARDUINO UNO. Arduino, 2023. Disponível em: <https://store-usa.arduino.cc/products/arduino-uno-rev3?selectedStore=us>. Acesso em: 30 abr. 2023.

ASPIN, Adam. Pro Power BI Desktop. Apress, 2016.

BRASIL. Ministério da Agricultura, Pecuária e Abastecimento. Departamento de Inspeção de Produtos de Origem Animal. Instrução Normativa nº 51, de 18 de setembro de 2002. Coleta de leite cru refrigerado e seu transporte a granel. Diário Oficial da República Federativa do Brasil, n. 172, p. 8-13, Seção I, 2002.

BRITO, J. R. F.; DIAS, J. C. A qualidade do leite. 1.ed. São Paulo: Tortuga, 1998. 88p.

CAVALCANTE, Cristiano Alexandre Virgínio; ALMEIDA, Adiel Teixeira de. Modelo multicritério de apoio a decisão para o planejamento de manutenção preventiva utilizando PROMETHEE II em situações de incerteza. Pesquisa Operacional, v. 25, p. 279-296, 2005.

CORRÊA, C. C.; VELOSO, A. F.; BARCZSZ, S. S. Dificuldades enfrentadas pelos produtores de leite: um estudo de caso realizado em um município de Mato Grosso do Sul. In: Congresso Brasileiro De Economia, Administração e Sociologia Rural. 2010.

DE ABREU VIEIRA, Pedro Henrique et al. Sistema de rastreamento solar de um eixo para módulos fotovoltaicos utilizando a plataforma Arduino. In: Congresso Brasileiro de Energia Solar-CBENS. 2014.

DECKMANN, S. M.; POMILIO, J. A. Avaliação da Qualidade de Energia Elétrica. UNICAMP/FEEC/DSCE, 2018.

ESTEVES, L; FERRAZ, V. A.; GUEDES, G. P.; FARIAS, P.C.M.A. Sistema wireless para aquisição de dados. Universidade Estadual de Feira de Santana, 2010.

FEZARI, Mohamed; AL DAHOUD, Ali. Integrated development environment “IDE” for Arduino. WSN applications, p. 1-12, 2018.

GERONYMO, Gean Marcos. Desenvolvimento de um conversor térmico com saída em frequência para padronização de transferência AC-DC. 2017.

GRIDLING, Gunther; WEISS, Bettina. Introduction to microcontrollers. Vienna University of Technology Institute of Computer Engineering Embedded Computing Systems Group, 2007.

IBRAHIM, Dogan. Microcontroller based applied digital control. John Wiley, 2006.

INTERFACE DO USUÁRIO DO POWER QUERY. Microsoft, 2023. Disponível em: <https://learn.microsoft.com/pt-br/power-query/power-query-ui>. Acesso em: 21 abr. 2023.

KAGAN, Nelson; ROBBA, Ernesto João; SCHMIDT, Hernán Prieto. Estimação de indicadores de qualidade da energia elétrica. Editora Blucher, 2009.

KUCMANSKI, Augusto. Monitoramento e armazenamento da variação de temperatura e umidade em uma residência eficiente. Universidade Federal de Santa Maria, 2018.

LEIRA, M. H., BOTELHO, H. A., BARRETO, B. B., BOTELHO, J. H. V., & PESSOA, G. O. Fatores que alteram a produção e a qualidade do leite: Revisão. Pubvet, v. 12, p. 172, 2018.

LINARES, Bruno Manoel. Desenvolvimento de um sistema para monitoramento de parâmetros operacionais de motores de indução trifásicos. Universidade Tecnológica Federal do Paraná, 2021.

MCROBERTS, Michael. Arduino Básico: Crie projetos simples e práticos com Arduino. 2. ed. São Paulo: Novatec Editora, 2015. 506p.

NERO, L. A.; MATTOS, M. R.; BELOTI, V.; BARROS, M. A. F.; PINTO, J. P. A. N.; ANDRADE, N. J.; SILVA, W. P.; FRANCO, B. D. G. M. Leite cru de quatro regiões leiteiras brasileiras: perspectivas de atendimento dos requisitos microbiológicos estabelecidos pela Instrução Normativa 51. Ciência e Tecnologia de Alimentos, v. 25, n. 1, p. 191-195, 2005.

NOVO, A. L. M. Avaliação de programas privados de assistência técnica no setor leiteiro: um estudo de caso do departamento de assistência técnica ao produtor Parmalat.

Dissertação de Mestrado. Departamento de Engenharia de Produção da Universidade Federal de São Carlos, São Carlos/SP. 2001.

OLIVEIRA, Nícolas Magno Gurgel de. Dispositivo de monitoramento e comunicação utilizando IoT em subestações abrigadas. 2022. Trabalho de Conclusão de Curso. Universidade Federal do Rio Grande do Norte.

PEREIRA, E. S., PIMENTEL, P. G., QUEIROZ, A. C. & MIZUBUTI, I. Y. Novilhas leiteiras. Graphiti Gráfica e Editora Ltda, Fortaleza, Ceará. 2010.

PIOVESAN, Fabrício Coradini. Telemetria aplicada na mecanização agrícola utilizando o datalogger CR 1000. Universidade Federal de Santa Maria, 2008.

RAVIV, Gil. Collect, Combine, and Transform Data Using Power Query in Excel and Power BI. Microsoft Press, 2018.

ROCHA, Felipe Barbalho et al. Plataforma de comunicação sem fio aplicada a sistemas de irrigação. Holos, v. 5, p. 260-273, 2014.

ROGÉRIO, Lúcio. Desenvolvimento e construção de um datalogger aplicado ao monitoramento de temperatura e umidade relativa do ar de uma granja cunícula. Revista de Engenharia e Tecnologia, v. 12, n. 4, 2020.

SILVA, Adriano J.; MUNHOZ, Fernando C.; CORREIA, Paulo B. Qualidade na utilização de energia elétrica no setor rural: problemas, legislação e alternativas. Procedings of the 4th Encontro de Energia no Meio Rural, 2002.

TORRES, João Delfino et al. Aquisição de dados meteorológicos através da plataforma Arduino: construção de baixo custo e análise de dados. Scientia Plena, v. 11, n. 2, 2015.

VOLPATO, Neri. Manufatura aditiva: tecnologias e aplicações da impressão 3D. Editora Blucher, 2021.

WENDLING, Marcelo. Sensores. Universidade Estadual Paulista. São Paulo, v. 2010, p. 20, 2010.

APÊNDICES

APÊNDICE A – CÓDIGO DATALOGGER

```
#include "EmonLib.h"   //biblioteca do sensor de tensão ZMPT101B
#include <SPI.h>       //biblioteca1 do módulo microSD 4MD36
#include <SD.h>        //biblioteca2 do módulo microSD 4MD36
#include <ThreeWire.h> //biblioteca1 do módulo RTC DS1302
#include <RtcDS1302.h> //biblioteca2 do módulo RTC DS1302

#define pinterm A2       //porta analógica do sensor de
temperatura do leite
#define termres 100000   //resistência do termistor NTC utilizado
#define tempnominal 25   //temperatura ambiente do termistor,
obtido na tabela 1
#define numsamples 5     //quantidade de amostragens que será
feita para definir a temperatura
#define beta 3950        //constante, obtido na tabela 1
#define res 10000        //valor da resistência usado em serie
com termistor
#define pinterm_2 A1     //porta analógica do sensor de
temperatura do compressor
#define termres_2 100000 //Valor do termistor, no caso estamos
utilizando um termistor de 100k
#define tempnominal_2 25 //temperatura ambiente do termistor,
obtido na tabela 1
#define numsamples_2 5   //quantidade de amostragens que será
feita para definir a temperatura

#define VOLT_CAL 218.5 //valor de ajuste de tensão, definido com
o auxilio de um multímetro
EnergyMonitor emon1;   //inicia um ambiente para leitura de
tensão

#define DS1302_RST  8  //porta digital do pino RST do modulo RTC
#define DS1302_DAT  9  //porta digital do pino RST do modulo RTC
#define DS1302_CLK  10 //porta digital do pino RST do modulo RTC

ThreeWire myWire(DS1302_DAT, DS1302_CLK, DS1302_RST); //função
para indicar as portas utilizadas do modulo RTC
RtcDS1302<ThreeWire> Rtc(myWire); //configuração do modulo RTC

int amostra[5];  //constante para lógica de loop na amostragem da
temperatura1
int i;
```

```
int amostra2[5]; //constante para lógica de loop na amostragem da
temperatura2

const int chipSelect = 4; //portas utilizadas no modulo microSD,
interface SPI - 4 (CS), 11 (MOSI), 12 (MISO) e 13 (SCK)

File myFile; //identifica o cartão de memória

void setup() {

  Serial.begin(9600);
  emon1.voltage(3, VOLT_CAL, 1.7); //configuração do sensor de
tensão, porta analógica(A3)/valor de calibração/relação de fase

    //configuração do modulo RTC
    Rtc.Begin(); //inicia o modulo RTC
    Serial.print("Compilado em: "); //escreve um texto no serial
monitor
    RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
//funcao responsavel por receber a data/hora
    Serial.println(); //escreve a data em uma linha no serial
monitor
    Serial.println(); //escreve a hora em outra linha no serial
monitor

    if(Rtc.GetIsWriteProtected()){ //verifica se o módulo RTC
está protegido contra gravação
        Serial.println("RTC está protegido contra gravação.
Habilitando a gravação agora..."); //escreve um texto no serial
monitor
        Rtc.SetIsWriteProtected(false); //habilita a gravação no
RTC
        Serial.println(); //salta uma linha no serial monitor
    }

    if(!Rtc.GetIsRunning()){ //verifica se o modulo RTC está
funcionando,
        Serial.println("RTC não está funcionando de forma
contínua. Iniciando agora..."); //escreve um texto no serial
monitor
        Rtc.SetIsRunning(true); //inicializa o RTC
        Serial.println(); //salta uma linha no serial monitor
    }

    RtcDateTime now = Rtc.GetDateTime(); //recebe os dados de
data/hora do sistema
```

```
    if (now < compiled) { //verifica se a informação registrada é
antes que a compilada
        Serial.println("As informações atuais do RTC estão
desatualizadas. Atualizando informações..."); //escreve um texto
no serial monitor
        Rtc.SetDateTime(compiled); //atualiza as informações de
data/hora
        Serial.println(); //salta uma linha no serial monitor
    }
    else if (now > compiled){ //verifica se a informação
registrada é depois que a compilada
        Serial.println("As informações atuais do RTC são mais
recentes que as de compilação. Isso é o esperado."); //escreve um
texto no serial monitor
        Serial.println(); //salta uma linha no serial monitor
    }
    else if (now == compiled) { //verifica se a informação
registrada é igual a compilada
        Serial.println("As informações atuais do RTC são iguais
as de compilação! Não é o esperado, mas está tudo OK.");
//escreve um texto no serial monitor
        Serial.println(); //salta uma linha no serial monitor
    }

    //configuracao do modulo microSD
  Serial.print("Inicializando o Módulo microSD..."); //escreve um
texto no serial monitor

  if(!SD.begin(chipSelect)) { //verifica se os pinos do módulo SD
estão ligados corretamente
    Serial.println("Inicialização do módulo micro SD
falhou!");//escreve um texto no serial monitor
    return;
  }
  Serial.println("Inicialização do módulo micro SD
realizada!");//escreve um texto no serial monitor

  myFile=SD.open("Datalogger.txt", FILE_WRITE); //cria um arquivo
dentro do cartão de mémoria com o nome "Datalogger.txt" e depois
habilita para escrever

  if (myFile) {   //verifica se o arquivo foi aberto e habilitado
para escrita com sucesso
    Serial.println("Arquivo aberto com sucesso!"); //escreve um
texto no serial monitor
```

```
    myFile.println("Data, Hora, Temperatura Leite, Temperatura
Compressor, Tensao Rede"); //escreve um texto no serial monitor,
sequencia dos dados armazenados
  }
  myFile.close(); //caso o arquivo não seja aberto com sucesso,
reinicia o processo para tentar abrir novamente
}

void loggingTime() {//inicia loop para coleta do tempo em
sincronia com os sensores
    RtcDateTime now = Rtc.GetDateTime();//coleta data/hora do
momento

    myFile = SD.open("Datalogger.txt", FILE_WRITE);//abre o
arquivo criado para escrever os dados de data/hora
  if (myFile) {
    myFile.print(now.Day(), DEC); //escreve o dia no arquivo
criado no cartão microSD
    myFile.print('/'); // escreve "/" no arquivo criado no cartão
microSD
    myFile.print(now.Month(), DEC); //escreve o mês no arquivo
criado no cartão microSD
    myFile.print('/'); // escreve "/" no arquivo criado no cartão
microSD
    myFile.print(now.Year(), DEC); //escreve o ano no arquivo
criado no cartão microSD
    myFile.print(','); // escreve "," para separar da data no
arquivo criado no cartão microSD
    myFile.print(now.Hour(), DEC); //escreve a hora no arquivo
criado no cartão microSD
    myFile.print(':'); //escreve ":" no arquivo criado no cartão
microSD
    myFile.print(now.Minute(), DEC); //escreve os minutos no
arquivo criado no cartão microSD
    myFile.print(':'); //escreve ":" no arquivo criado no cartão
microSD
    myFile.print(now.Second(), DEC); //escreve os segundos no
arquivo criado no cartão microSD
    myFile.print(","); // escreve "," para separar a hora da
proxima leitura no arquivo criado no cartão microSD
  }
  Serial.print(now.Day(), DEC); //escreve o dia no monitor serial
  Serial.print('/'); //escreve "/" no monitor serial
  Serial.print(now.Month(), DEC); //escreve o mês no monitor
serial
  Serial.print('/'); //escreve "/" no monitor serial
```

```
  Serial.print(now.Year(), DEC); //escreve o ano no monitor
serial
  Serial.print(' '); //escreve um espaço em branco no monitor
serial
  Serial.print(now.Hour(), DEC); //escreve a hora no monitor
serial
  Serial.print(':'); //escreve ":" no monitor serial
  Serial.print(now.Minute(), DEC); //escreve os minutos no
monitor serial
  Serial.print(':'); //escreve ":" no monitor serial
  Serial.println(now.Second(), DEC); //escreve os segundos no
monitor serial
  myFile.close(); //para de escrever no arquivo e avança o código
  delay(1000);  //espera 1 seg para avançar para o próximo loop
}

void loggingTemperature() {//inicia loop para coleta de
temperatura do leite
float media;

for (i=0; i< numsamples; i++) { //coleta 5 valores de leitura de
tensao da porta do arduino
amostra[i] = analogRead(pinterm);
delay(10);
}

media = 0;
for (i=0; i< numsamples; i++) {
media += amostra[i];
}
media /= numsamples; //encontra o valor da resistência para
tensão lida
media = 1023 / media - 1;
media = res / media;
float temperatura; //faz o cálculo da temperatura usando a
formula beta
temperatura = media / termres; // (R/Ro)
temperatura = log(temperatura); // ln(R/Ro)
temperatura /= beta; // 1/B * ln(R/Ro)
temperatura += 1.0 / (tempnominal + 273.15); // + (1/To)
temperatura = 1.0 / temperatura; // inverte o valor
temperatura -= 273.15; // converte para Celsius

Serial.print("Temperatura: "); //mostra o valor no serial monitor
Serial.print(temperatura);
Serial.println(" °C");
```

```
  myFile = SD.open("Datalogger.txt", FILE_WRITE); //abre o
arquivo criado para escrever os dados de temperatura do leite
  if (myFile) {
    myFile.print(temperatura); //escreve o valor da temperatura
calculado
    myFile.print(","); //escreve uma virgula para separar da
leitura do proximo sensor
  }
  myFile.close(); //para de escrever no arquivo e avança o código
}

void loggingTemperature_2() {
float media_2;

for (i=0; i< numsamples_2; i++) {
amostra2[i] = analogRead(pinterm_2);
delay(10);
}

media_2 = 0;
for (i=0; i< numsamples_2; i++) {
media_2 += amostra2[i];
}
media_2 /= numsamples_2; //encontra o valor da resistência para
tensão lida
media_2 = 1023 / media_2 - 1;
media_2 = res / media_2;
float temperatura_2;  //faz o cálculo da temperatura usando a
formula beta
temperatura_2 = media_2 / termres_2; // (R/Ro)
temperatura_2 = log(temperatura_2); // ln(R/Ro)
temperatura_2 /= beta; // 1/B * ln(R/Ro)
temperatura_2 += 1.0 / (tempnominal_2 + 273.15); // + (1/To)
temperatura_2 = 1.0 / temperatura_2; // inverte o valor
temperatura_2 -= 273.15; // converte para Celsius

Serial.print("Temperatura 2: ");  //mostra o valor no serial
monitor
Serial.print(temperatura_2);
Serial.println(" °C");

  myFile = SD.open("Datalogger.txt", FILE_WRITE); //abre o
arquivo criado para escrever os dados de temperatura do
compressor
  if (myFile) {
    myFile.print(temperatura_2); //escreve o valor da temperatura
calculado
```

```
    myFile.print(","); //escreve uma virgula para separar da
leitura do proximo sensor
  }
  myFile.close(); //para de escrever no arquivo e avança o código
}

void loggingTensao() {
  emon1.calcVI(17,2000); //(17 semiciclo, tempo máximo para
realizar a medicao)
  float supplyVoltage   = emon1.Vrms; //recebe o valor da tensão
encontrada

  Serial.print("Tensão: "); //mostra o valor no serial monitor
  Serial.print(supplyVoltage, 0); //mostra o valor no serial
monitor
  Serial.println("V"); //mostra o valor no serial monitor

  myFile = SD.open("Datalogger.txt", FILE_WRITE);//abre o arquivo
criado para escrever os dados de tensão
  if (myFile) {
    myFile.println(supplyVoltage); //escreve o valor da tensão
lido, por ser o ultimo a ser coletado, não escreve virgula no
final
  }
  myFile.close(); //para de escrever no arquivo e avança o código
}

void loop() { //reinicia o loop de coletas de dados, seguindo a
ordem, data/temperatura leite/temperatura compressor/tensão
  loggingTime();
  loggingTemperature();
  loggingTemperature_2();
  loggingTensao();
  delay(5000);
}
```