PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

KL PET CARE - ALIMENTADOR AUTOMÁTICO PARA PETS

LUCIANA ALVES DE MORAIS

GOIÂNIA
2023

LUCIANA ALVES DE MORAIS

KL PET CARE - ALIMENTADOR AUTOMÁTICO PARA PETS

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a):

Profa. Ma. Ludmilla Reis Pinheiro dos Santos

Banca Examinadora:

Prof. Me. Olegário Correa da Silva Neto

Prof. Dr. Nilson Cardoso Amaral

GOIÂNIA

2023

LUCIANA ALVES DE MORAIS

KL PET CARE - ALIMENTADOR AUTOMÁTICO PARA PETS

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação, em _____/_____/_____.

_____
Orientador(a): Ma. Ludmilla Reis Pinheiro de Santos

_____
Prof. Me. Olegário Correa da Silva Neto

_____
Prof. Dr. Nilson Cardoso Amaral

GOIÂNIA

2023

AGRADECIMENTOS

Agradeço a Deus em primeiro lugar por me acompanhar e me dar forças nos momentos difíceis. Agradeço à minha família pelo apoio e por acreditar em mim.

“Cada escolha uma renúncia, essa é a vida.”

Charlie Brown Jr

RESUMO

Durante o período de pandemia, houve um notável aumento na adoção de animais de estimação. Isso se deve ao fato de que o distanciamento social levou muitas pessoas a encontrarem conforto ao terem animais em suas residências. Após a pandemia, a maioria dos brasileiros voltaram a suas rotinas normais, como ir ao serviço, viajar e até mesmo ficar um tempo maior fora de sua residência. Os animais permanecem muito tempo sozinhos, e às vezes fazendo apenas uma refeição por dia. Pensando nisso, este trabalho tem como objetivo apresentar um protótipo de um alimentador automático de animais. O protótipo possui um reservatório de ração e um reservatório para água. Foi utilizado o aplicativo BLYNK, para o monitoramento do nível de água no reservatório. Este protótipo tem como principal foco facilitar o dia a dia das pessoas que possuem animais de estimação, permitindo uma alimentação adequada todos os dias.

Palavras-Chave: Blynk. Alimentador automático de pets.

ABSTRACT

During the pandemic period, there was a notable increase in pet adoption. This is due to the fact that social distancing led many people to find comfort in having pets in their homes. After the pandemic, the majority of Brazilians returned to their normal routines, such as going to work, traveling, and even spending more time away from their homes. The animals are left alone for extended periods, sometimes having only one meal a day. Taking this into consideration, this project aims to present a prototype of an automatic pet feeder. The prototype features a food reservoir and a water reservoir. The BLYNK application was used to monitor the water level in the reservoir. This prototype focuses on making the daily lives of pet owners easier, ensuring proper feeding every day.

Keywords: Blynk. Automatic pet feeder.

LISTA DE FIGURAS

Figura 1 – Recomendação Diária para Gatos ---- 17
Figura 2 - Alimentador GINGAPETS ---- 18
Figura 3 - Alimentador 3D ---- 19
Figura 4 - Alimentador Inteligente ---- 20
Figura 5 - Alimentador Thymos Pet ---- 21
Figura 6 - Diagrama de Refeições ---- 23
Figura 7 - Real Time Clock (RTC) ---- 24
Figura 8 - ESP8266 ---- 25
Figura 9 - Motor de Passo ---- 25
Figura 10 – Fonte 5v 1.2A ---- 26
Figura 11 - Conexão Blynk ---- 27
Figura 12 - Protótipo de Água e Ração ---- 29
Figura 13 - Protótipo Água e Ração ---- 30
Figura 14 - Ligação ESP8266 e Motor de Passo ---- 31
Figura 15 - Ligação ESP8266 e RTC ---- 32
Figura 16 - Ligação ESP8266 e Boia Sensor de Nível ---- 34
Figura 17 - Nível de água visualizada pelo IP ---- 34
Figura 18 - Blynk App - Nível baixo e Alto de Água ---- 35

LISTA DE TABELAS

Tabela 1 - Recomendação Diária..... 17

LISTA DE SIGLAS

ABINPET	Associação Brasileira da Industria de Produtos para Animais
CDC	Centro de Controle e Prevenção de Doenças
COMAC	Comissão de Animais de Companhia
GPIO	General Purpose Input/Output
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
LED	*Light Emitting Diode*
RTC	*Real Time Clock*
Wi-Fi	*Wireless Fidelity*

SUMÁRIO

1 INTRODUÇÃO 13

1.1 Objetivo Geral 13

1.2 Objetivos Específicos 14

1.3 Justificativa 14

1.4 Resultados Esperados 14

1.5 Método 14

1.6 Estrutura do Trabalho 15

2 REFERENCIAL TÉORICO 16

2.1 Vantagens de ter um Animal de Estimação 16

2.2 Recomendação Diária de Consumo dos *Pets* 16

2.3 Trabalhos Correlatos 18

2.3.1 Protótipo de Alimentador Automático para Animais Domésticos 18

2.3.2 Autofeeder: Alimentador Automático para Animais Domésticos de Pequeno Porte 19

2.3.3 Alimentador Inteligente para Cães e Gatos Baseados no Conceito de Internet das Coisas 20

2.3.4 Dosador Alimentar Automático para Animais Domésticos 21

2.3.5 Considerações sobre os Trabalhos Relacionados 22

3 PROPOSTA DO TRABALHO 23

3.1 Descrição da Proposta de Solução 23

3.2 Descrição dos Hardwares 24

3.2.1 RTC 24

3.2.2 Esp8266 24

3.2.3 Motor de Passo 25

3.2.4 Fonte de Alimentação 26

3.3 Blynk 26
4 DESENVOLVIMENTO DO HARDWARE - RAÇÃO / ÁGUA 28
4.1 Protótipo de Ração / Água 28
4.1.1 Testes 31
5 CONSIDERAÇÕES FINAIS 36
REFERÊNCIAS 38
APÊNDICE A 41

1 INTRODUÇÃO

A Associação Brasileira da Indústria de Produtos para Animais de Estimação (ABINPET), aponta que existem 144,3 milhões de animais de estimação no Brasil (ABINPET, 2016). O Brasil possui 213,7 milhões de habitantes, portanto o número de pets corresponderia a 7,6% da população brasileira de acordo com a Instituto Brasileiro de Geografia e Estatística (IBGE, 2021). Houve um aumento de 30% de animais em lares brasileiros durante a pandemia do COVID-19 segundo dados da pesquisa divulgada pela Comissão de Animais de Companhia (COMAC, 2021).

Ter um animal de estimação proporciona vários benefícios, como companhia e redução de estresse. Vale lembrar que, muitos cuidados devem ser tomados para evitar que os animais adquiram alguma doença, por exemplo verificar se a vacinação está em dia e quais alimentos eles estão consumindo.

Destes cuidados, a alimentação adequada é primordial para a saúde dos animais, recomendações nutricionais certas, ajudam a prolongar a saúde e o bem-estar do animal de estimação. Uma alimentação balanceada evita risco de desenvolver algumas doenças, por exemplo, a cegueira e o ressecamento da córnea por falta de vitaminas A, E e K (FACETTI, 2020; BRAGHIROLLI, 2020).

São recomendados alimentos industrializados, secos ou úmidos. Esses alimentos contêm vitaminas, sais minerais e proteínas, elementos que garantem a nutrição e o desenvolvimento do animal (ABINPET, 2016).

Diante deste contexto, esta pesquisa pretende responder à seguinte questão: - **Como automatizar um sistema de alimentação para pets, usando ESP8266?**

1.1 Objetivo Geral

Este trabalho tem como objetivo o desenvolvimento de um protótipo de alimentador automático de baixo custo para alimentar *Pets*.

1.2 Objetivos Específicos

- Projetar a estrutura mecânica do alimentador;
- Desenvolver um protótipo a fim de demonstrar o funcionamento do alimentador automático;
- Integrar um *app* de monitoramento para monitorar o nível de água no reservatório.

1.3 Justificativa

A alimentação dos animais exige um cuidado especial, pelo fato de dependerem de outras pessoas para se alimentar. Pensando nisso, a utilização de um alimentador automático, facilitaria a vida das pessoas que passam um longo período fora de casa, e até mesmo para aquelas pessoas que costumam viajar aos finais de semana.

1.4 Resultados Esperados

Espera-se que os resultados deste trabalho contribuam para:

- Cuidar da alimentação dos Pets;
- Permitir o monitoramento do nível de água, baixo e alto, no reservatório;
- Prevenir contra doenças, por falta de alimentação adequada;
- Facilitar a alimentação dos animais, quando dono estiver ausente.

1.5 Método

O método de pesquisa utilizado neste trabalho, segundo sua natureza, é um resumo do assunto, pois busca sistematizar uma área de conhecimento, indicando sua evolução, neste caso, a área de conhecimento em automação (WAZLAWICK, 2014).

Quanto aos objetivos, é uma pesquisa descritiva, pois busca obter os dados consistentes sobre determinada realidade, também serão explorados os

materiais utilizados para a automatização, que é o estudo de cada componente usado (WAZLAWICK, 2014).

Este trabalho, quanto aos procedimentos técnicos, trata-se de uma pesquisa bibliográfica e experimental, pois consiste na coleta de requisitos e materiais. O planejamento para criação do alimentador automático, se baseia em um microcontrolador Arduino, com sensores e célula de carga (MONTEIRO, 2013).

1.6 Estrutura do Trabalho

Este trabalho está estruturado em quatro capítulos.

O capítulo 1 apresentou a introdução, juntamente com os objetivos, resultados esperados, justificativa e o método de pesquisa adotado.

O capítulo 2 apresenta a recomendação alimentar para os Pets e os trabalhos correlatos.

O capítulo 3 apresenta a descrição da proposta de solução e os componentes de hardware utilizados para a implementação do protótipo.

O capítulo 4 apresenta os testes realizados com componentes de hardwares para construção do protótipo.

O capítulo 5 apresenta as considerações finais do trabalho e as dificuldades enfrentadas.

2 REFERENCIAL TÉORICO

Este capítulo apresenta as vantagens de se ter um animal de estimação e qual a recomendação de consumo para Pets, além de trabalhos correlatos ao tema.

2.1 Vantagens de ter um Animal de Estimação

Ter um animal de estimação em casa, traz uma série de benefícios para a saúde física e mental. De acordo com uma estimativa do Instituto Brasileiro de Geografia e Estatística (IBGE) eles estão presentes em quase 140 milhões de lares brasileiros (IBGE, 2021).

Após realizar uma pesquisa nos Estados Unidos, o Centro de Controle e Prevenção de Doenças (CDC), afirmou que ter um pet ajuda a reduzir a pressão sanguínea e os níveis de colesterol e triglicerídeos (MELO, 2021).

Em contato com os bichos, o ser humano ativa o sistema límbico, responsável pelas emoções mais instintivas. Com isso, libera endorfinas, gerando sensação de tranquilidade e bem-estar (SEGUROS, 2021).

2.2 Recomendação Diária de Consumo dos *Pets*

O excesso de ração pode causar desinteresse do animal, posteriormente o alimento é rejeitado. Por isso, não é recomendado deixar a tigela com muita ração, pois prejudica a boa alimentação do animal (ZANOLINI, 2016).

A Tabela 1 e a Figura 1 apresenta a quantidade de alimento recomendada para cães e gatos, respectivamente.

Tabela 1 - Recomendação Diária para cães

PORTE	PESO [kg]	PORÇÃO [g/dia]
Pequeno	5 a 8	44
Médio	15 a 21	135
Grande	31 a 40	264
Gigante	51 a 60	497

Fonte: Premier Pet, 2021

Figura 1 – Recomendação Diária para Gatos

Fonte: Kimoko, 2022

2.3 Trabalhos Correlatos

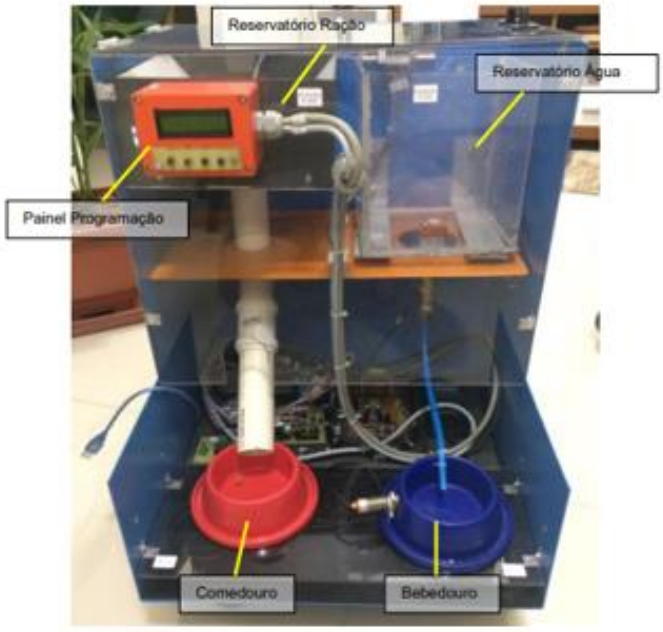
Esta seção apresenta os trabalhos relacionados a um alimentador automático para Pets.

2.3.1 Protótipo de Alimentador Automático para Animais Domésticos

Este trabalho apresentou um protótipo, denominado GINGAPETS, com o objetivo de alimentar os animais domésticos na ausência dos seus tutores. O sistema utilizou o microcontrolador Arduino, motor de passos NEMA17, sensores capacitivos e display LCD para visualização da hora e data das refeições, conforme a Figura 2 (BUOGO, 2017).

O GINGAPETS proporciona autonomia para servir 4 refeições no dia, para animais de pequeno porte e atender a maioria das rotinas diárias de alimentação, ilustrado conforme a Figura 2 (BUOGO, 2017).

Figura 2 - Alimentador GINGAPETS



Fonte: Buogo, 2017.

2.3.2 Autofeeder: Alimentador Automático para Animais Domésticos de Pequeno Porte

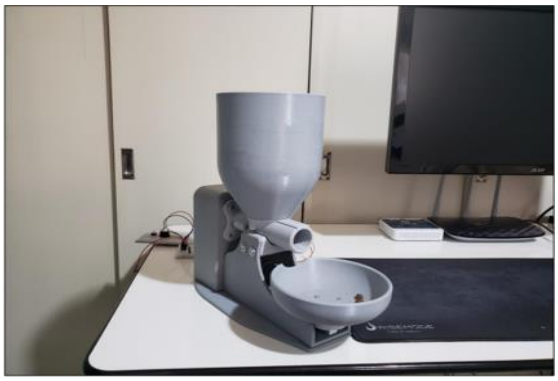
AUTOFEEDER foi desenvolvido para auxiliar os donos de animais que passam a maioria do tempo fora de sua residência, conforme a Figura 3. Com a intenção de resolver este problema foi desenvolvido um aplicativo que proporcionou ao usuário realizar a programação das refeições (LEMOS, 2020).

O protótipo tinha com um compartimento para fixação do motor e instalação de componentes eletrônicos, um reservatório para ração com tampa, um sistema para transporte de comida e um recipiente onde era despejado o alimento. Através do módulo RTC DS3231, o Arduino realizava a comparação do horário atual com os horários das refeições programadas pelo usuário e, se essa condição fosse verdadeira, o motor era acionado, iniciando o processo de transporte de ração do reservatório até o recipiente, ilustrado conforme a Figura 3 (ALVES, 2020).

A célula de carga foi instalada na base do alimentador, embaixo do pote, e é responsável por informar ao Arduino a quantidade de ração existente naquele momento, fazendo com que o motor seja parado assim que a porção de ração presente no pote for igual a que foi programada pelo usuário no aplicativo (ALVES, 2020).

Com a utilização do alimentador foi possível proporcionar uma alimentação adequada de forma prática e eficiente aos *pets*, tornando-o útil tanto para os proprietários que possuem escassez de tempo, quanto para os que procuram facilidades para o seu dia a dia (ALVES, 2020).

Figura 3 - Alimentador 3D.



Fonte: Alves, 2020.

2.3.3 Alimentador Inteligente para Cães e Gatos Baseados no Conceito de Internet das Coisas

Este trabalho foi criado para automatização de alimentos, com o propósito de alimentar animais de rua, usando Arduino, sensores e NodeMCU, que é uma plataforma de desenvolvimento, que faz uso do microcontrolador ESP8266 (SILVA, 2021).

Por meio desse modelo foi possível observar que os materiais de baixo custo pode-se construir um protótipo funcional. Concluiu-se que um alimentador para cães e gatos projetado com NodeMCU é uma opção para colaborar com os cuidados dos animais de estimação (SILVA, 2021).

Foi possível utilizar o projeto para ajudar os animais de rua, e usar para os cuidados de animais em residências, agilizando no cuidado dos *Pets*. Foi considerado um produto de simples manuseio, que pode ser desenvolvido com materiais que podem ser achados em lojas de materiais de construção e peça de hardware de baixo custo, ilustrado conforme a Figura 4 (SILVA, 2021).

Figura 4 - Alimentador Inteligente

Fonte: Silva, 2021.

2.3.4 Dosador Alimentar Automático para Animais Domésticos

Este trabalho apresentou um protótipo com objetivo de alimentar animais de pequeno a médio porte. O protótipo utiliza o microcontrolador ESP8266 NodeMCU como central de controle e linguagem C na sua programação. Contém um aplicativo responsável por controlar o alimentador à distância, permitindo ao usuário alimentar o seu animal de estimação mesmo estando longe, chamado Blynk (MATHEUS, 2020).

Pelo aplicativo foi possível ativar o alimentador conforme os horários pré-programados pelo usuário ou ativá-lo de modo síncrono. A criação do protótipo teve como objetivo facilitar o dia a dia das pessoas que possuem um animal de estimação e, que nem sempre, dispõem de tempo hábil para tratá-los adequadamente, ilustrado conforme a Figura 5 (MATHEUS, 2020).

Figura 5 - Alimentador Thymos Pet

Fonte: Matheus, 2020.

2.3.5 Considerações sobre os Trabalhos Relacionados

O que diferencia os alimentadores automáticos da solução proposta por esse trabalho, é que o alimentador GINGAPETS por exemplo possui sua programação totalmente manual.

O alimentador Autofeeder foi programado apenas para animais de pequeno porte, e seu propósito era apenas servir ração.

O alimentador Pet Friendly possuía seus componentes visível a exposição de água, e até mesmo do próprio Pet, provocando a desconexão da fiação ou desconfiguração do sistema programado.

Já o alimentador THYMOS PET possibilitava a configuração de dias e horários através de um aplicativo chamado BLYNK, para a distribuição de ração, já o bebedouro de água possuía um acoplador para garrafa pet.

Em relação a isso, o objetivo desse trabalho foi desenvolver um protótipo corrigindo e melhorando essas falhas apontadas, pelos trabalhos anteriores.

3 PROPOSTA DO TRABALHO

Este capítulo apresenta a descrição da proposta de solução e os componentes de hardware utilizados para o desenvolvimento do trabalho.

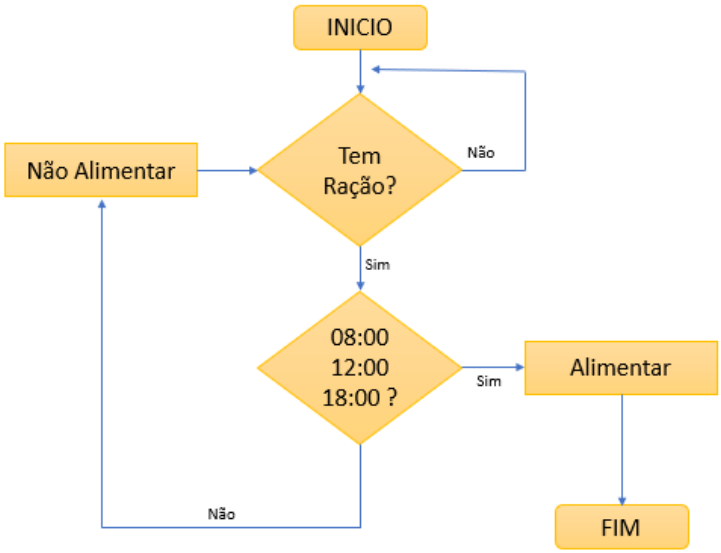
3.1 Descrição da Proposta de Solução

Construir um alimentador automático para pets, utilizando ESP8266, Relógio RTC DS3231, hélice e motor de passo. A hélice irá dentro do reservatório, sendo controlada pelo motor de passo, de acordo com a programação dos dias e horários programados na ESP8266. A função do RTC DS1307 é manter a contagem precisa de horas, minutos, segundos, dia, mês e ano, mesmo quando o sistema é desligado. Ele possui uma bateria interna que alimenta o circuito e mantém os dados de tempo em seu registrador interno.

Foi utilizado um aplicativo chamado Blynk que faz conexão com ESP8266 via Wi-Fi, para o monitoramento do nível baixo de água no reservatório.

A figura 6 ilustra o acionamento de uma refeição do pet em um dos três horários distintos: 08:00, 12:00 ou 18:00.

Figura 6 - Diagrama de Refeições



Fonte: Autoria Própria

3.2 Descrição dos Hardwares

Esta seção apresenta os componentes de hardware que serão utilizados para o desenvolvimento do projeto.

3.2.1 RTC

O *Real Time Clock* (RTC) DS3231 é um relógio de tempo real de alta precisão e baixo consumo de energia, conforme ilustrado na Figura 7. A placa possui um sensor de temperatura e um cristal oscilador para garantir sua exatidão.

O RTC é um processador especialmente dedicado ao controle de horários e datas, sendo que essas informações não são perdidas. As informações são transmitidas para a BIOS2, ao ambiente operacional e a outros programas que requisitam da informação (GUIMARÃES, 2017).

Figura 7 - Real Time Clock (RTC)

Fonte: Filipeflop, 2022

3.2.2 Esp8266

O microcontrolador ESP8266 da Espressif Systems conforme ilustrado na Figura 8, projetado para a IoT e projetos relacionados a sistemas embarcados, possui baixo custo e baixo consumo de energia. O NodeMCU possui um conector micro-USB que permite a conexão com um computador para programação e alimentação da placa.

Além disso, a placa é equipada com uma antena integrada, que facilita a conexão sem fio à rede Wi-Fi (FILIPEFLOP, 2019).

Figura 8 - ESP8266

Fonte: Filipeflop, 2022

3.2.3 Motor de Passo

Na alimentação com tensão de corrente contínua, ele polariza os eletroímãs e assim define seu sentido de rotação. Seu princípio é o de atração e repulsão de ímãs, de modo que polos iguais se repelem e polos diferentes se atraem, motor de passo ilustrado na Figura 9 (FILIPEFLOP, 2019).

Figura 9 - Motor de Passo

Fonte: Filipeflop, 2022.

3.2.4 Fonte de Alimentação

A Fonte de Alimentação Chaveada 5V 1A é amplamente empregada em diversos projetos de automação residencial e para fornecer energia a pequenos circuitos. Ela é especialmente útil para alimentar componentes como LEDs, câmeras de segurança e dispositivos eletrônicos que requerem uma fonte de 5V e uma corrente de saída estável de até 1 A, conforme ilustrado na Figura 10 a seguir (FILIPEFLOP, 2022).

Figura 10 – Fonte 5v 1.2A

Fonte: Filipeflop, 2022

3.3 Blynk

O Blynk é um aplicativo disponível tanto para o sistema Android quanto para o sistema iOS, sendo projetado para ser utilizado em projetos *Internet of Things* (IoT). A IoT possibilita a monitoração, controle e interação inteligente de objetos com seu ambiente. Através de sensores e dispositivos integrados, esses objetos coletam dados em tempo real, como temperatura, umidade, localização, movimento, entre outros. Esses dados são enviados para a nuvem, onde são processados, analisados e utilizados para tomar decisões, automatizar tarefas e aprimorar a eficiência e a qualidade de vida das pessoas. A principal característica do Blynk é permitir que microcontroladores possam ser controlados remotamente, de forma que dados de sensores e módulos possam

ser obtidos e exibidos no aplicativo que fica instalado no dispositivo móvel. Permite também que cargas sejam acionadas, além de muitas outras funcionalidades que a ferramenta disponibiliza (BLYNK, 2019).

Um dos pontos principais do Blynk é a possibilidade de controlar a plataforma de qualquer lugar do mundo através do aplicativo instalado no dispositivo móvel. Para isto, basta que a plataforma esteja configurada e conectada ao servidor Blynk através da internet e que o aplicativo no dispositivo móvel também possua conexão com a internet.

Figura 11 - Conexão Blynk

Fonte: Blynk, 2020

4 DESENVOLVIMENTO DO HARDWARE - RAÇÃO / ÁGUA

Este capítulo apresenta os testes realizados com componentes de hardwares para construção do protótipo, a montagem do circuito foi realizada de forma gradativa, testando cada componente individualmente e depois integrando um por vez.

4.1 Protótipo de Ração / Água

O desenvolvimento do protótipo teve por base animais de pequeno a médio porte. Sua estrutura foi projetada com fibra de vidro, devido sua resistência.

O protótipo de ração tem capacidade de armazenar 3 porções por dia e o protótipo de água tem capacidade de armazenar 1,5 litros de água.

O protótipo de água é baseado no princípio de Stevin, também conhecido como princípio de Pascal ou lei de Stevin, que é um princípio fundamental da hidrostática, uma área da física que estuda os fluidos em repouso (MUNDO DA EDUCAÇÃO, 2022). Baseado nessa ideia, foi construído o protótipo de água como mostra a Figura 12.

Figura 12 - Protótipo de Água e Ração

Fonte: Autoria Própria

O reservatório de água foi colocado do lado invertido encaixado na tigela, assim a água começa a fluir para baixo devido à força da gravidade. Conforme a água desce, a pressão aumenta à medida que a profundidade aumenta, de acordo com o princípio de Stevin. A pressão no ponto mais baixo do reservatório será maior do que a pressão atmosférica.

Na tigela existe um funil de pequena profundidade criando uma restrição para o fluxo de água. O funil reduz a área disponível para a água fluir, o que aumenta a velocidade do fluxo. No entanto, de acordo com o princípio de Stevin, a pressão no ponto mais baixo do reservatório deve ser maior do que a pressão atmosférica. Se a pressão no ponto mais baixo do reservatório for maior do que a pressão atmosférica, a água fluirá do reservatório para a tigela através do funil até que a pressão se equilibre. Isso ocorre porque a água fluirá do ponto de alta pressão (reservatório) para o ponto de baixa pressão (tigela) para buscar o equilíbrio.

No protótipo de ração foi utilizado o motor de passo, juntamente com uma hélice que controla o fluxo da distribuição da ração, foi realizado os testes para a distribuição do alimento e monitoramento da falta de líquido no reservatório.

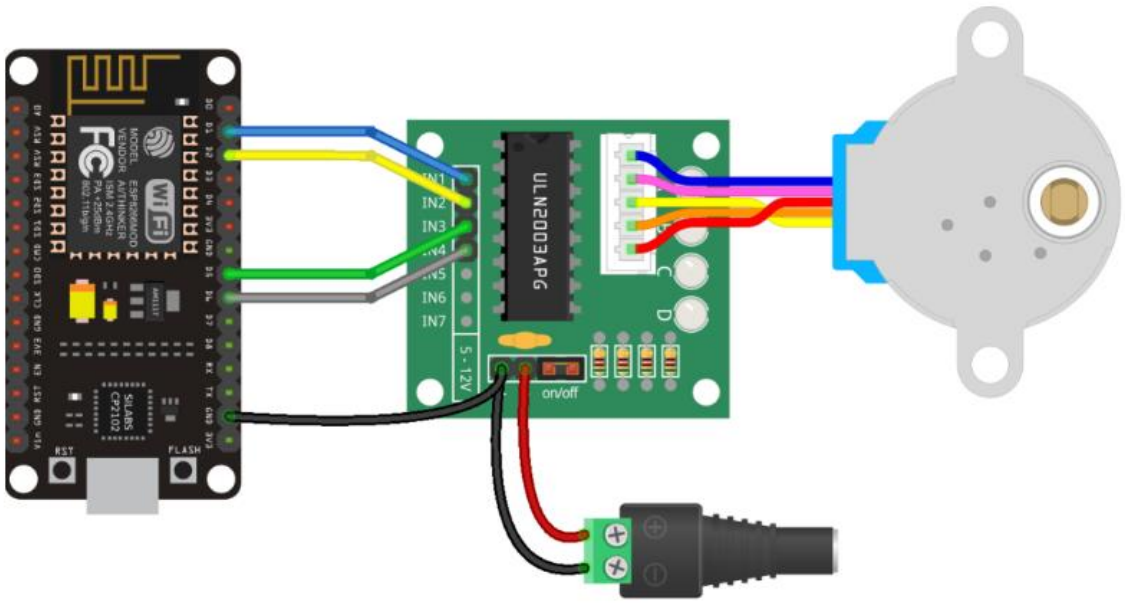
Figura 13 - Protótipo Água e Ração

Fonte: Autoria Própria

4.1.1 Testes

O primeiro teste objetivou verificar o funcionamento do motor. A Figura 14 mostra o diagrama de ligação do motor de passos com a ESP8266, através do driver do motor, ligados nos pinos GPIO (GPIO5, GPIO4, GPIO14 e GPIO12), código implementado no IDE do Arduino, (APÊNDICE A – Código 1).

Figura 14 - Ligação ESP8266 e Motor de Passo



FONTE: WIKIPÉDIA, 2022.

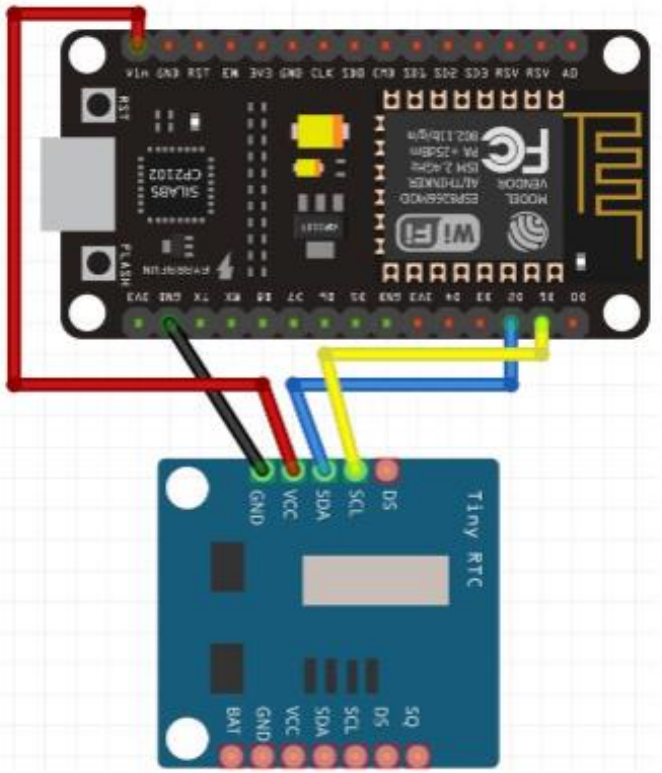
O teste foi realizado diretamente no protótipo desenvolvido, o que permitiu verificar o funcionamento real do motor em conjunto com a hélice. Essa abordagem proporcionou uma análise prática do desempenho do motor, considerando a velocidade programada e as repetições da rotação.

Obteve resultados satisfatórios, onde o motor de passo funcionou conforme o esperado. Através do código implementado, o motor foi capaz de realizar repetidas rotações junto com a hélice, evidenciando seu correto funcionamento e a capacidade de controle por parte do sistema.

O segundo teste objetivou a verificação do funcionamento do módulo RTC, ilustrado na Figura 15 mostra como foi feita a ligação do RTC, foi utilizado os pinos GPIO (GPIO5 e GPIO4) da ESP8266, código implementado no IDE do Arduino, APÊNDICE A – Código 2.

O teste foi bem-sucedido a ESP8266 foi capaz de ler as informações de data e hora do RTC.

Figura 15 - Ligação ESP8266 e RTC



FONTE: WIKIPÉDIA, 2022.

O terceiro teste objetivou a verificação do funcionamento do motor de passo e módulo RTC, código implementado no IDE do Arduino, (APÊNDICE A – Código 3).

- Foi realizado a ligação conjunta dos componentes na esp8266.
- O teste foi registrar 3 horários diferentes para que o motor fosse acionado.

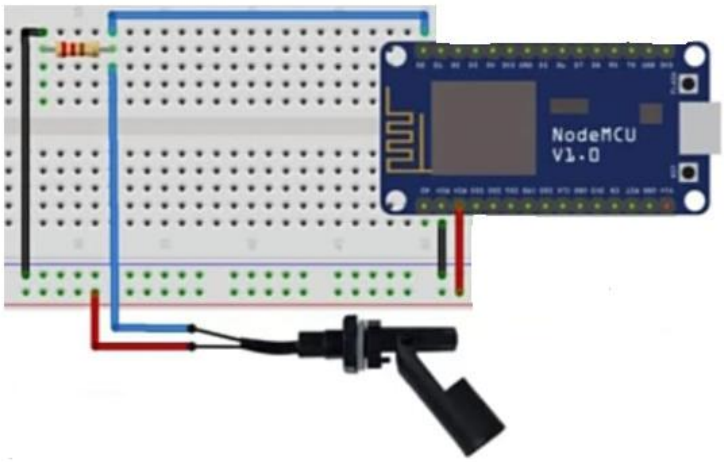
Infelizmente, o teste não obteve o êxito esperado. Embora o driver do motor de passo recebesse o sinal enviado pela ESP8266, de acordo com a sincronização do horário do RTC, o motor não apresentou força suficiente para girar adequadamente. Isso indica um problema na interação entre os componentes.

Um estudo foi realizado comparando o Motor de Passo Nema 17 com torque de 5,2 kgf.cm ao motor de passo 28BYJ-48, buscando determinar qual deles é mais apropriado para substituir o outro. O Motor de Passo Nema 17 é amplamente reconhecido e utilizado em projetos eletrônicos e de automação devido à sua confiabilidade, precisão e força de rotação adequada para uma variedade de aplicações.

O torque de 5,2 kgf.cm do Nema 17 indica sua capacidade de gerar força rotacional. Quanto maior o valor do torque, maior é a potência do motor, permitindo lidar com cargas mais pesadas e realizar tarefas que demandam maior esforço de rotação. Essa característica o torna uma opção apropriada para aplicações que requerem maior potência e torque do que o motor de passo 28BYJ-48 pode oferecer. O Motor de Passo Nema 17 oferece maior potência e força de rotação, tornando-se uma escolha mais adequada para substituir o motor de passo 28BYJ-48 em situações em que é necessária uma capacidade de torque mais robusta (TECH MAKERS, 2018).

O quarto teste objetivou a verificação do funcionamento da boia sensor de nível. A ideia é que o dono do pet consiga verificar se o nível de água está baixo antes de sair de casa, para que possa abastecer. Neste primeiro teste o acompanhamento do nível de água é no browser, o código apresentado no (APÊNDICE A – Código 4) gera um IP, com esse IP é possível verificar o nível de água, foi indicado 300 para o nível baixo e 800 para o nível alto que seria em média a quantidade de água abaixo do nível e a quantidade acima do nível conforme a Figura 17. O segundo teste é no próprio app Blynk como mostra na Figura 18, código implementado no IDE do Arduino (APÊNDICE A – Código 5).

Figura 16 - Ligação ESP8266 e Boia Sensor de Nível



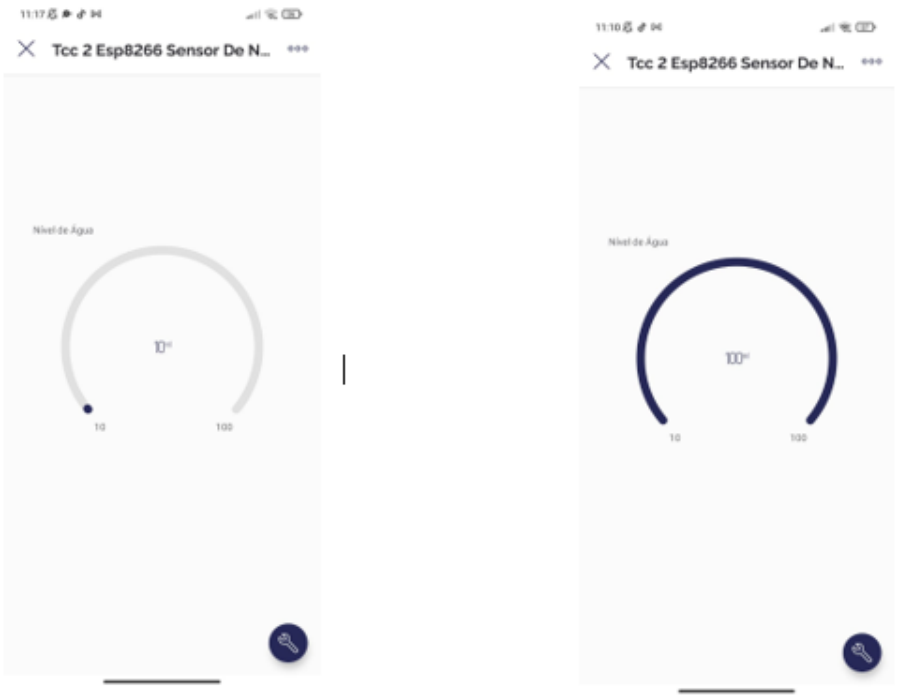
Fonte: RandomNerdTutorials, *2022.*

Figura 17 - Nível de água visualizada pelo IP

Fonte: autoria própria

O app Blynk foi integrado no projeto para realizar o monitoramento, nele o dono do pet consegue verificar se o nível de água está baixo, conforme a Figura 18, o valor 10 quer dizer que está abaixo do nível, valor 100 que está sobre o nível, o valor 10 quer dizer que está apenas abaixo do nível e não vazio totalmente, código implementado no IDE do Arduino, APÊNDICE A – Código 5.

Figura 18 - Blynk App - Nível baixo e Alto de Água



Fonte: Autoria própria

O quinto teste teve como objetivo verificar e avaliar o funcionamento e a eficiência de todos os módulos (motor de passo, módulo RTC, comunicação Blynk e sensor de nível) quando operando em conjunto. Foi implementado um código no IDE do Arduino (Apêndice A - Código 6) para esse fim.

Infelizmente, o teste não obteve o êxito esperado. Apenas o módulo RTC e a comunicação do Blynk com o sensor de nível funcionaram corretamente. No entanto, assim como no teste anterior, o motor de passo não girou conforme programado, apesar de o driver do motor receber o sinal adequadamente.

Os resultados inconsistentes do teste demonstram a persistência do problema relacionado ao motor de passo. Mesmo com os esforços anteriores para alimentar o motor de forma separada e garantir a correta sincronização do horário, o motor ainda não demonstrou força suficiente para executar o movimento desejado.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou um protótipo desenvolvido para alimentar automaticamente animais de pequeno e médio porte, seguindo os horários programados pelo usuário. Um dos principais aspectos deste projeto foi a integração do aplicativo Blynk, que desempenhou um papel fundamental no monitoramento do nível de água do reservatório.

No que diz respeito ao reservatório de água, foram implementados sensores de nível de boia, permitindo a verificação do nível tanto baixo quanto alto. Esses sensores desempenharam um papel crucial na detecção da necessidade de abastecimento do reservatório. Através do aplicativo Blynk, os usuários têm a capacidade de monitorar o nível de água em tempo real. Isso oferece a conveniência de verificar remotamente se o reservatório precisa ser abastecido, evitando preocupações e garantindo que a alimentação do animal seja mantida adequadamente.

Para a construção do protótipo, utilizou-se fibra de vidro como material principal, o que garantiu a resistência e durabilidade necessárias para o funcionamento adequado do dispositivo.

Já o protótipo de ração possui uma hélice que é controlada por um motor de passo com a funcionalidade de distribuir a ração, o motor usado neste trabalho não foi adequado para o reservatório cheio, o motor não conseguiu girar.

A primeira dificuldade foi na alimentação de todo o circuito, foi preciso separar a alimentação do motor e da ESP8266.

A segunda dificuldade foi na integração de todos os componentes, não foi possível identificar o problema, o porquê do RTC e o sensor de nível conectados juntos a ESP8266, interferir na rotação do motor. O driver do motor de passo recebia o sinal, mas o motor não tinha força suficiente para girar, mesmo ele recebendo a alimentação externa individual, pelo monitor serial era possível mostrar que o driver do motor recebia o sinal, e o motor não girava.

Futuramente, a fim de melhorar o desempenho do protótipo, podem ser consideradas diversas melhorias adicionais. Uma delas seria a utilização de um motor de passo mais potente, capaz de lidar com cargas mais pesadas e fornecer um torque maior. Isso aumentaria a eficiência e a capacidade do protótipo de transportar a ração de forma precisa.

Outra possibilidade de melhoria seria a incorporação de um sistema de câmeras ao protótipo, permitindo que os usuários monitorem seus animais de estimação enquanto a alimentação está ocorrendo. Essa funcionalidade ofereceria maior tranquilidade e controle, permitindo que os proprietários observem o comportamento dos animais e verifiquem se a alimentação está sendo realizada corretamente. Para garantir a operação contínua do protótipo, mesmo em caso de falta de energia elétrica, pode-se considerar a inclusão de uma bateria que alimente o circuito durante essas situações. Dessa forma, a alimentação do animal de estimação não seria interrompida em casos de queda de energia.

Adicionalmente, para evitar que o animal de estimação brinque com o protótipo e cause danos ou mau funcionamento, seria recomendado projetar o dispositivo de modo a fixá-lo ao chão ou à parede. Isso garantiria que o protótipo permaneça estável e seguro, reduzindo a probabilidade de interferências indesejadas.

É importante ressaltar que essas melhorias propostas visam otimizar o desempenho, a segurança e a usabilidade do protótipo de alimentação.

REFERÊNCIAS

ALVES, Gabriel Lemos. **Alimentador Automático para animais Domésticos de Pequeno Porte.** Orientador: Igor Muzeka. TCC (Graduação) - Curso de Ciência da Computação para análise e aprovação. Faculdade de Ciência da Computação, Centro Universitário Unifacvest Lages. 2020. Disponível em:< https://www.unifacvest.edu.br/assets/uploads/files/arquivos/db467-alves,-gabriel-lemos.-autofeeder-alimentador-automatico-para-animais-domesticos-de-pequeno-porte.-tcc-defendido-em-julho-de-2020.pdf >. Acesso em 23 jun. 2023

BUOGO**,** Douglas Rossi. **Alimentador Automático Para animais Domésticos**. Orientador: Prof. Dr. Michael Klug. TCC (Graduação) – Curso Tecnologia em Mecâtronica Industrial. Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, campos Joinville. 2017. Disponível em: < http://joinville.ifsc.edu.br/~bibliotecajoi/arquivos/tcc/mecind/180273.pdf > Acesso em 23 jun. 2023.

BLYNK. **Como funciona o Blynk**, 2019. Disponível em:< https://docs.blynk.cc >. Acesso em: 23 nov. 2022

BRAGHIROLLI, Eduardo. **Alimentação de cachorro: como prevenir deficiência nutricional**. Disponível em:< https://www.petz.com.br/blog/nutricao/alimentacao-cachorro/>. Acesso em: 14 mai. 2022.

FILIPEFLOP. **MAKER HERO**. Disponível em: < https://www.makerhero.com/?gclid=CjwKCAjws7WkBhBFEiwAIi1681QLSZhQTXlUO6Dl5Me4e3hj2MFKIkqjEUgB_9Km7UBF9GqTN_g_BxoCh0IQAvD_BwE >. Acesso em: 06 jun. 2023.

GUIMARÃES, Fábio. **Componentes eletrônicos e suas utilidades.** Mundo projetado, 2017. Disponível em: https://mundoprojetado.com.br/. Acesso em: 22, nov. 2022.

MATEUS, Willian da Rocha F. **Thymos Pet – Dosador Alimentar Automático para Animais Domésticos.** Orientador: Jorge Alberto Lewis Esswein Junior.2020. 97f. TCC(Graduação) – Curso de Engenharia Elétrica Da Universidade do Sul de Santa Catarina. 2020. Disponível em: < https://repositorio.animaeducacao.com.br/bitstream/ANIMA/4205/4/TCC%20THYMOS%20PET%20%e2%80%93%20DOSADOR%20ALIMENTAR%20AUTOM%c3%81TICO%20PARA%20ANIMAIS%20DOM%c3%89STICOS.pdf>. Acesso em 06 nov. 2022.

MELO, Raissa. **Conheça 5 benefícios de ter um animal de estimação.** ARTEX, 2021. Disponível em: < https://artex.com.br/emcasa/conheca-5-beneficios-de-ter-um-animal-de-estimacao/ >. Acesso em: 27, nov. 2022.

MUNDO EDUCAÇÃO. **Teorema de STEVIN**, 2023. Disponível em: https://mundoeducacao.uol.com.br/fisica/teorema-stevin.htm#:~:text=O%20teorema%20de%20Stevin%20%C3%A9%20um%20m%C3%A9todo%20de%20calcular%20a,pelo%20peso%20do%20pr%C3%B3prio%20l%C3%ADquido. Acesso em: 06 de jun. 2023.

SILVA, VH do N. **Alimentador Inteligente Para Cães e Gatos Baseados no Conceito de internet das Coisas.** Orientador. Prof. Marcelo Anderson Batista do Santos. 2023. 31 f. TCC (Graduação) - Curso de Tecnologia em Sistemas para Internet. Faculdade de Engenharia, Instituto Federal de Educação, Ciência e tecnologia do Sertão Pernambucano. 2021. Disponível em: < https://releia.ifsertao-pe.edu.br/jspui/handle/123456789/682 >. Acesso em 23 jun. 2023.

TECH MAKERS. **Produtos para Automação**. 2018. Disponível: < https://techmakers.fbitsstatic.net//media/Datasheet%20Motor%20de%20Passo%20Nema%2017%201.1Kgf.cm%20P-N-AK17-1.1F6LN1.8.pdf?origem=MediaCenter/Datasheet%20Motor%20de%20Passo%20Nema%2017%201.1Kgf.cm%20P-N-AK17-1.1F6LN1.8.pdf>. Acesso em 15 jun.2023

WAZLAWICK, Raul Sidnei. **Metodologia de Pesquisa para Ciência da Computação.** 2. ed. São Paulo: Elsevier, 2014.

APÊNDICE A

CÓDIGO 1 – TESTE DO FUNCIONAMENTO DO MOTOR

```
// Luciana Alves
#include <AccelStepper.h> // Biblioteca para o controle do motor
const int stepsPerRevolution = 2048;  // Constante stepsRevolution atibuida o número de passo por revolução
// PINOS GPIO - Pinos da placa ESP8266
#define IN1 14 //GPIO14 D5 - Constante
#define IN2 12 //GPIO12 D6 - Constante
#define IN3 13 //GPIO13 D7 - Constante
#define IN4 15 //GPIO15 D8 - Constante
// Instância da classe - Os pinos são passados como argumentos que especifica os pinos GPIO mencionados anteriormentes
AccelStepper stepper(AccelStepper::HALF4WIRE, IN1, IN3, IN2, IN4);
void setup() {
// Inicia a comunicação com a taxa de transmissão
Serial.begin(115200);
stepper.setMaxSpeed(800); // Parâmetros - Define a velocidade máxima
stepper.setAcceleration(200); // Aceleração passo por segundo
stepper.moveTo(stepsPerRevolution); // Revolução completa do motor
}
//Execução continua após o setup
void loop() {
// Ele faz uma comparação - Se o motor chegou a a distancia 0, quer dizer que chegou ao destino.
if (stepper.distanceToGo() == 0){
stepper.moveTo(-stepper.currentPosition());// Caso tenha chegado, ela faz com que o motor move a posição oposta.
Serial.println("Mudando de direção");
}
// Gerar pulsos necessarios para movimentação do motor
stepper.run();
}
```

CÓDIGO 2 – TESTE DO FUNCIONAMENTO DO RTC

```
// Luciana Alves
#include <Wire.h> //Incluindo as bibliotecas necessarias
#include <RTClib.h> //Incluindo as bibliotecas necessarias
RTC_DS1307 rtc; //Instância da classe
void setup() {// Comunicação serial é iniciada com o valor em ().
  Serial.begin(115200);
  Wire.begin();
  if (!rtc.begin()) { // Verifica se o RTC está conectado na placa
    Serial.println("RTC não encontrado");
    while (1); } //caso não encontre ele entra em um loop.
 rtc.adjust(DateTime(2023, 06, 07, 14, 42, 00)); // Aqui é onde ajusta o data e hora manualmente - Pode ser comentado, após rodar uma vez
}
void loop() { // Loop é executado várias vezes após a função setup
  DateTime now = rtc.now(); //  Aqui tem a data e a hora atual do rtc utilizando metodo "noW" que retorna um objeto DateTime.
  // Exibe a data e hora atual no Monitor Serial
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(' ');
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.print(now.second(), DEC);
  Serial.println();
// Um atraso de 1 segundo, até a próxima iteração.
  delay(1000);
}
```

CÓDIGO 3 - TESTE DO FUNCIONAMENTO DO RTC E MOTOR DE PASSO

```
//Bibliotecas para iniciar os componentes
#include <Wire.h>
#include <RTClib.h>
#include <AccelStepper.h>
RTC_DS1307 rtc; // instancia pra interagir com o rtc
AccelStepper stepper(AccelStepper::HALF4WIRE, 14, 13, 12, 15);// // Instância da classe - Os pinos são passados como argumentos que especifica os pinos
const int stepsPerRevolution = 2048;// constante que representa o número de passos por rotação
// Função setup - Inciada uma única vez
void setup() {
  Serial.begin(115200); // comunicação serial
  Wire.begin(); // comunicação com a biblioteca wire.h
  if (!rtc.begin()) { // inicia comunicação com rtc
    Serial.println("RTC não encontrado"); // verica se o rtc está conectado corretamente
    //while (1); // caso n esteja fica presa no loop  }
  rtc.adjust(DateTime(2023, 6, 7, 15, 16, 00)); // configurar data e hora manual
  stepper.setMaxSpeed(800); //velocidade
  stepper.setAcceleration(100); // aceleração
  stepper.moveTo(stepsPerRevolution);  // move o motor de passo conforme os passos por rotação declarado }
void loop() {
  DateTime now = rtc.now(); // inicializa com a data e hora correspondente pelo r
    stepper.run(); // avança o motor
    delay (1000);
 // Verifica se a hora e minuto atual correspondem ao horário desejado
  if (now.hour() == 15 && now.minute() == 17) {// Verifica se a hora e minuto atual correspondem ao horário desejado
    //stepper.run(); // avança o motor
    delay(2000); // Girar o motor por 2 segundos
    stepper.stop(); // Parar o motor
    //Serial.println("Movimento concluído");
```

CÓDIGO – 4 TESTE SENSOR DE NÍVEL POR IP

```
//Teste sensor de nível
// Luciana Alves
#include <ESP8266WiFi.h> //Biblioteca necessaria para comunicação da ESP8266.
#include <WiFiClient.h> //Biblioteca necessaria para comunicação servidor WEB.
#include <ESP8266WebServer.h> //Biblioteca necessaria para comunicação WIFI .

const char* ssid = "FireFibra-Alves_Morais";  // nome da minha rede Wi-Fi
const char* password = "2708221524";  // senha do Wi-Fi

ESP8266WebServer server(80);// Instancia do objeto - Que é responsavel para lidar com as requisições HTTP recebidas pelo servidor Web

const int sensorPin = A0;  // Pino analógico onde o sensor de nível está conectado

const int nivelBaixo = 300;  // Valor analógico para o nível baixo (ajuste conforme necessário)
const int nivelAlto = 800;  // Valor analógico para o nível alto (ajuste conforme necessário)
//  Setup executada um vez, ela responsavel por configurar a ESP8266 e estabelecer conexão com wifi e iniciar o servidor.
void setup() {
  Serial.begin(115200);
  delay(10);

  pinMode(sensorPin, INPUT);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando ao WiFi...");
  }

  Serial.println("Conectado ao WiFi!");
  Serial.print("Endereço IP: ");
  Serial.println(WiFi.localIP());

  server.on("/", handleRoot);

  server.begin();
  Serial.println("Servidor web iniciado!");
}
// Essa função executa repetidamente. Responsável por lidar com as requisições HTTP recebidas pelo servidor WEB
void loop() {
  server.handleClient();
}
//Essa função é chamada toda vez que uma requisição HTTP é feita. Ela contém a lógica para processar a requisição e gerar a resposta.
void handleRoot() {
  int nivelAgua = analogRead(sensorPin);
// Lógica para obter o nível de água do sensor e determinar o status do reservátorio.
  String status;
  String corReservatorio;

  if (nivelAgua < nivelBaixo) {
    status = "Nivel Baixo";
    corReservatorio = "red";
  } else if (nivelAgua > nivelAlto) {
    status = "Nivel Alto";
    corReservatorio = "blue";
  } else {
    status = "Tem Água";
    corReservatorio = "blue";
  }
  // Gera uma pagina HTML dinâmica.
  server.send(200, "text/html", paginaWeb); }
```

CÓDIGO 5 – SENSOR DE NÍVEL E BLYNK

```
//Dados coletados do Blynk
#define BLYNK_TEMPLATE_ID "TMPL2qC5jmFGk" // Identificação da template criada
#define BLYNK_TEMPLATE_NAME "Tcc 2 Esp8266 Sensor de nivel" // Representação do nome da
#define BLYNK_AUTH_TOKEN "uKyON0jyeJoWWvMCokhoyq4f86ncKEBU" // Esse é o token de autenticação, serve para conexão da esp e do servidor Blynk
#define BLYNK_PRINT Serial // saida
#include <ESP8266WiFi.h> // Biblioteca da esp8266
#include <BlynkSimpleEsp8266.h> // biblioteca do blynk
BlynkTimer timer;// Essa classe ajuda no agendamento das tarefas em intervalos
//Conexão wifi e servidor
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "FireFibra-Alves_Morais";
char pass[] = "2708221524";
int flag = 0; //Variavel de sinalização - Utilizada na comparação
#define WATER_SENSOR A0 // Pino analógico usado para ler o valor do sensor de nível de água
void notifyOnWaterLevel()// função que verifica o nível de agua {
  int waterLevel = analogRead(WATER_SENSOR); //variavel waterLevel incia com o valor do water_sensor, o analoREad lê o valor presente no pino.
  // Aqui inicia a verificação - as condições
  if (waterLevel < 500 && flag == 0) {
    Serial.println("Baixo nível de água detectado");
    Blynk.notify("Alerta: nível baixo de água detectado"); // preciso configurar no blynk para essa linha serval
    flag = 1; // flag recebe 1 - alerta o baixo nivel de agua
    Blynk.virtualWrite(V1, 0); // Define o valor do medidor como zero
  } else if (waterLevel >= 500) { //verifica se o waterLevel é maior ou igual a 500. Se essa condição for verdadeira, o bloco de código dentro deste else if será executado
    flag = 0; //flag recebe 0 -  nivel de agua alto
    Blynk.virtualWrite(V1, waterLevel); // Envia o valor do nível de água para o medidor }}
void setup() {
  Serial.begin(9600); // comunicação de dados
  Blynk.begin(auth, ssid, pass); // permite que o dispositivo envie e receba dados do app blynk
  pinMode(WATER_SENSOR, INPUT); //Sinal analogico - lê o valor do sesnsor
  // Configuração do widget do medidor (gauge) no aplicativo Blynk
  Blynk.setProperty(V1, "label", "Luciana Alves");
  Blynk.setProperty(V1, "min", "10");
  Blynk.setProperty(V1, "max", "100");
  timer.setInterval(1000L, notifyOnWaterLevel); // função para verificar o nivel de agua e enviar notificação - preciso configurar no app blynk }
void loop() {
  Blynk.run();
  timer.run();
}
```

CÓDIGO 6 – TODOS OS COMPONENTES

```
//Bibliotecas necessárias para comunicação da esp8266, motor de passo e o RTC

#include <Wire.h>

#include <RTClib.h>

#include <AccelStepper.h>

#include <BlynkSimpleEsp8266.h>

RTC_DS1307 rtc; //Instância da classe

const int stepsPerRevolution = 2048;  // Define o número de passos por revolução do motor

// Configurando os pinos da Esp8266

#define IN1 14 // GPIO14 D5

#define IN2 12 // GPIO12 D6

#define IN3 13 // GPIO13 D7

#define IN4 15 // GPIO15 D8

// Instância da classe - Os pinos são passados como argumentos que especifica os pinos GPIO mencionados anteriormentes

AccelStepper stepper(AccelStepper::HALF4WIRE, IN1, IN3, IN2, IN4);

// Código de autenticação - Coletei do Blynk após a configuração do pin

#define BLYNK_TEMPLATE_ID "TMPL2qC5jmFGk"

#define BLYNK_TEMPLATE_NAME "Tcc 2 Esp8266 Sensor de nivel"

#define BLYNK_AUTH_TOKEN "uKyON0jyeJoWWvMCokhoyq4f86ncKEBU"

// Conexão com meu Wifi - Esp8266

#define BLYNK_PRINT Serial

BlynkTimer timer;

char auth[] = BLYNK_AUTH_TOKEN;

char ssid[] = "FireFibra-Alves_Morais";

char pass[] = "2708221524";

int flag = 0;

#define WATER_SENSOR A0 // Pino analógico usado para ler o valor do sensor de nível de água

// Função para identificar o nível de água e enviar sinal para o blynk
```

```
void notifyOnWaterLevel()
{
int waterLevel = analogRead(WATER_SENSOR); // armazena o nível de água representado pelo sensor
  // Nesse if é verificação do nível de água, fazendo uma comparação com flag
  if (waterLevel < 500 && flag == 0) {
    Serial.println("Baixo nível de água detectado");
    Blynk.notify("Alerta: nível baixo de água detectado");
    // O vitualWrite - é usado para definir o valor do medidor no widget correspondente (V1) como zero.
    flag = 1;
    Blynk.virtualWrite(V1, 0); // Define o valor do medidor como zero
  } else if (waterLevel >= 500) {
    flag = 0;
    Blynk.virtualWrite(V1, waterLevel); // Envia o valor do nível de água para o medidor


          }
         }
         // Comunicação serial é iniciada com o valor em ().
         void setup() {
          Serial.begin(115200);
          Wire.begin();
          if (!rtc.begin()) { // Verifica se o RTC foi encontrado
            Serial.println("RTC não encontrado");
            while (1);//caso não encontre ele entra em um loop .   }
         // Aqui é onde ajusta o data e hora manualmente
          rtc.adjust(DateTime(2023, 06, 02, 21, 00, 00));
          stepper.setMaxSpeed(500);// Parâmetros - Define a velocidade máxima
          stepper.setAcceleration(100); // Aceleração passo por segundo
         // Estabelecer a conexão com servidor blynk
          Blynk.begin(auth, ssid, pass);
          pinMode(WATER_SENSOR, INPUT);
```

```
  // Configuração do widget do medidor (gauge) no aplicativo Blynk
  Blynk.setProperty(V1, "label", "Nível de Água");
  Blynk.setProperty(V1, "min", "10");
  Blynk.setProperty(V1, "max", "100");
  timer.setInterval(1000L, notifyOnWaterLevel);
}
void loop()
{
  DateTime now = rtc.now();
  int currentHour = now.hour();
  int currentMinute = now.minute();
  int currentSecond = now.second();
  // Horário 1: Acionar o motor às
  if (currentHour == 21 && currentMinute == 01 && currentSecond == 0) {
    stepper.moveTo(stepsPerRevolution); // Revolução completa do motor
  }
  // Horário 2: Acionar o motor às
  if (currentHour == 21 && currentMinute == 02 && currentSecond == 0) {
    stepper.moveTo(stepsPerRevolution);// Revolução completa do motor
  // Horário 3: Acionar o motor às
  if (currentHour == 21 && currentMinute == 03 && currentSecond == 0) {
    stepper.moveTo(stepsPerRevolution);// Revolução completa do motor
}
  stepper.run();
  Blynk.run();
  timer.run();
  delay(1000);
```