INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* BETIM
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Matheus Rodrigues Vieira Silva

PROTÓTIPO DE UM ALIMENTADOR AUTOMATIZADO DE ANIMAIS DE ESTIMAÇÃO MONITORADO E CONTROLADO VIA TELEGRAM.

Betim

2023

MATHEUS RODRIGUES VIEIRA SILVA

PROTÓTIPO DE UM ALIMENTADOR AUTOMATIZADO DE ANIMAIS DE ESTIMAÇÃO MONITORADO E CONTROLADO VIA TELEGRAM.

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais Campus Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Arthur Hermano Rezende Rosa

Coorientador: Helbert Ribeiro de Sá

Betim

2023

FICHA CATALOGRÁFICA

S586p Silva, Matheus Rodrigues Vieira
Protótipo de um alimentador automatizado de animais de estimação monitorado e controlado via telegram / Matheus RodriguesVieira Silva. – 2023.
70 f. : il.

Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Câmpus Betim, 2023.

Orientador: Prof. Dr. Arthur Hermano Rezende Rosa
Coorientador: Prof. Me. Helbert Ribeiro de Sá

1. Automatização. 2. Monitoramento. 3. Animais de estimação. 4. Controle. I. Matheus Rodrigues Vieira Silva. II. Título.

CDU: 681.5

Ficha catalográfica elaborada pelo Bibliotecário Denísio P. Marcos CRB-6/3142

Matheus Rodrigues Vieira Silva

Protótipo de um alimentador automatizado para animais de estimação, monitorado e controlado via Telegram

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Controle e Automação do Instituto Federal de Minas Gerais Campus Betim como requisito parcial à obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Betim, 16 de Junho de 2023.

BANCA EXAMINADORA

Prof. Arthur Hermano Rezende Rosa (Orientador)
IFMG Campus Betim

Prof. Helbert Ribeiro de Sá (Coorientador)
IFMG Campus Betim

Prof.(a) Amara Fuccio de Fraga e Silva
IFMG Campus Betim

Prof. Virgil Del Duca Almeida

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus por ter conseguido chegar até aqui e à minha família que serviu de base em todos os momentos que precisei. Agradeço também aos colegas que fiz dentro do IFMG, foram anos de convivência e aprendizado. A todos os professores que de maneira especial, puderam contribuir para minha formação e experiência no meio pessoal e estudantil.

Meu amigo Frederico Marazzi e minha amiga Cecília Ferreira que me ajudaram e apoiaram em todo processo de construção desse trabalho, a vocês toda minha gratidão.

RESUMO

O número de adoção de animais de estimação tem progredido a cada dia, isso indica que os animais fazem cada vez mais parte da vida dos seres humanos. Consequentemente, a preocupação com o bem-estar deles, seja em produtos, comércio e serviços disponibilizados aumenta.

Dentre os produtos disponibilizados no mercado, tem-se os alimentadores automatizados, o qual possui a função de alimentá-los de forma autônoma quando o dono não está presente.

O alto custo deste produto inviabiliza sua aquisição para pessoas que procuram comodidade para cuidar de seu animal, mas não possuem condições de obter. Com isso é apresentado um alimentador monitorado e controlado via Telegram, com funções de monitoração e controle, construído com o custo relativamente baixo quando comparado a produtos similares no mercado, a fim de trazer praticidade ao usuário e melhor alimentação do animal.

Como propostas de melhoria deste projeto, temos a possível instalação de câmera e sensores que auxiliariam no monitoramento e proteção.

Palavras-chave: Animais de estimação. Alimentador. Monitoramento. Automatizado. Conforto.

ABSTRACT

The number of pet adoption has progressed every day, this indicates that animals are increasingly part of human life. Consequently, the concern for their well-being, whether in products, commerce and services available, increases.

Among the products available on the market, there are automated feeders, which have the function of feeding them autonomously when the owner is not present.

The high cost of this product makes its acquisition unfeasible for people looking for convenience to take care of their animal, with this a feeder monitored and controlled via Telegram is presented, with monitoring and control functions, built with a relatively low cost in order to bring practicality to the user and better feeding of the animal.

As proposals for improving this project, we have the possible installation of a camera and sensors that would help in monitoring and protection.

Keywords: Pets. Feeder. Monitoring. Automated. Comfort.

LISTA DE FIGURAS

LISTA DE TABELAS E GRÁFICOS

LISTA DE SIGLAS E ABREVIATURAS

CPU - Central Processing Unit

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

I^2C - Inter Integrated Circuit

IoT- Internet of Things

RTC - Real Time Clock

RAM - Random Access Memory

SCL - Serial Clock

SDA - Serial Date

LISTA DE SÍMBOLOS

<	Menor que
>	Maior que
=	Igual

SUMÁRIO

1. INTRODUÇÃO

A busca do homem por melhorias para a sua vida tem ajudado significativamente na evolução da sociedade. Descobrindo o fogo e inventando ferramentas que auxiliam no trabalho, o homem sempre está em busca de maior comodidade e conforto. Como consequência, surgiu a automação (LIMA, 2003).

A automação predial e residencial foi baseada na automação industrial, bem conhecida e difundida a mais tempo. Porém, em virtude da diferente realidade entre o uso dos dois tipos de arquiteturas, têm sido criados sistemas dedicados para ambientes onde não se dispõe de espaço para grandes centrais controladoras e extensos sistemas de cabeamento (BOUWER,2017).

Produtos automatizados relacionados ao mercado dos animais de estimação, os chamados pets, têm ganhado seu lugar de destaque na economia do país, pois com o passar dos anos se percebe o interesse que os seus donos têm dado aos produtos e serviços disponibilizados no mercado. Essa importância tem se expandido graças ao processo/fenômeno de humanização dos animais domésticos o qual está diretamente ligada a evolução do consumo de produtos e serviços voltados para este novo tipo de consumidor (ELIZEIRE, 2013).

Dentre os produtos presentes no mercado nacional, tem-se os alimentadores automatizados que possuem funções referentes a alimentação do animal. Através do uso de componentes eletrônicos acoplados a estes, é possível realizar o monitoramento do animal visando seu bem-estar. Entretanto, o alto custo destes produtos inviabiliza sua aquisição e consequentemente impede sua popularização.

Com base neste cenário, este trabalho consiste na construção de um alimentador automatizado de animais de estimação de pequeno porte, que visa a alimentação saudável do animal, além da redução da alimentação manual do animal, o que faz com que se poupe tempo e se torne uma maneira interativa e viável de obter um projeto de custo relativamente baixo que envolva a automação residencial.

Desta forma, o animal será alimentado no momento e na quantidade desejada, garantindo que tenha uma alimentação balanceada e de qualidade, evitando assim, possíveis doenças como a obesidade.

1.1 Justificativa

Devido à rotina agitada, a tarefa de cuidar do animal de estimação pode se tornar um desafio e fazer com que ele não receba os cuidados necessários, como sua alimentação, na quantidade e horário correto.

Em se tratando dos cuidados, a alimentação é uma das principais questões a serem avaliadas, pois assim como nos seres humanos, uma alimentação inadequada pode gerar obesidade, diabetes entre outras doenças, pois eles não têm controle de quando e quanto comem.

Desse modo, o projeto visa melhorar o cuidado dos donos em relação aos seus animais quando não estiverem presentes para alimentá-los. É importante o fornecimento de alimento adequado para nossos animais, pois uma proporção correta de proteínas e vitaminas garantem a nutrição e desenvolvimento deles.

Além disso, uma alimentação na quantidade adequada, garante a saúde e bem-estar, além da prevenção contra possíveis doenças.

1.2 Problema

A busca pelo bem-estar do animal muitas vezes não é realizada devido a rotina agitada de seus cuidadores, como por exemplo a não realização da troca da ração diariamente, o que faz com que os animais não agradarem desta sem seu cheiro característico, pois seu olfato aguçado faz com que o cheiro seja o fator principal que determina seu interesse.

Quando adicionada em grande quantidade e exposta ao tempo por determinado período, além de perder seu aroma, a ração umedece e se torna apta a contrair bactérias, ratos e baratas, o que pode gerar uma série de doenças ao animal, além do desperdício da mesma e, consequentemente, maior custo por ter que trocá-la frequentemente.

O animal quando possui uma alimentação desequilibrada e baixa em nutrientes, a longo prazo pode resultar na obesidade e ter complicações à sua saúde, pois de maneira geral são sensíveis e vulneráveis.

1.3 Objetivo Geral

Construção de um alimentador automatizado de animais domésticos de

pequeno porte de baixo custo que visa diminuir o desperdício e a substituição do processo de alimentação manual diária do animal.

1.3.1 Objetivos específicos

- Construção de um protótipo capaz de alimentar o animal a distância.
- Monitoramento do alimentador via Telegram.
- Criação de um Chat Bot para receber e realizar os comandos feitos pelo usuário.
- Permitir o controle de agendamento do horário para alimentar o animal.

1.4 Organização do trabalho

O primeiro capítulo do trabalho consiste na descrição e funcionalidade do projeto, juntamente com a solução apresentada e os objetivos específicos do mesmo.

No capítulo 2 é relatado a fundamentação teórica, no qual são abordados projetos simulares que solucionaram o problema proposto.

No capítulo 3 será descrita as ferramentas a serem utilizadas para a construção do projeto juntamente com cada processo realizado.

No quarto capítulo será discutido os resultados obtidos que serão analisados baseado na proposta inicial.

O capítulo 5 relata as principais conclusões e possíveis contribuições futuras do trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Projetos similares

O projeto de Oliveira (2017) visa a alimentação de animais domésticos a distância que foi programada através da utilização de um Raspberry Pi 2 Modelo B. A imagem do protótipo construído não foi apresentada.

Em Farhat (2019), é construído um alimentador acionado em horários controlados, como diferencial foi utilizado uma câmera que faz possível o

monitoramento visual do nível de ração que há no recipiente.

O alimentador automático de Almeida (2016) foi desenvolvido apenas para cães de pequeno porte, o qual um aplicativo de celular controla a quantidade e horário que o animal será alimentado, além do histórico de alimentação do mesmo.

O trabalho de Sena (2021) mostra um alimentador automático desenvolvido de maneira prática e sustentável para gatos e cães em situação de abandono. Este faz a utilização de um módulo fotovoltaico para fazer a captação da luz do sol e convertê-la em energia elétrica.

Outro protótipo consiste na construção de um alimentador automático que pode ser programado para liberar alimento em períodos agendados pelo usuário e que possua autonomia para dar comida ao cão ou gato por vários dias (OCHAKOWSKI, 2007). Este foi feito em um sistema de malha fechada, pois antes de liberar o novo alimento, são despejados em outro reservatório os restos de ração deixados pelo animal.

Os projetos relatados acima, apesar de serem eficazes, são relativamente caros para se construir, o que pode se tornar inviável dependendo do orçamento do usuário.

2.2 Internet das Coisas (IoT)

A automação residencial vem sendo amplamente utilizada no setor comercial de países desenvolvidos, e pode ser definida como "O conjunto de serviços proporcionados por sistemas tecnológicos integrados como o melhor meio de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação" (OLIVEIRA, 2005).

Para que se possa entender mais sobre o que a automação residencial engloba, é necessário introduzir o conceito da “internet das coisas”, além de analisar os componentes que serão utilizados no projeto.

A internet das coisas (IoT) é um paradigma inovador que está crescendo rapidamente e ganhando muito terreno no cenário atual, o qual as redes Wi-Fi e telecomunicação possuem bastante influência.

A ideia deste conceito é o fato de tudo estar ligado de alguma maneira, dispositivos (sensores, celulares, etc..) são capazes de interagir um com o outro,

facilitando ainda mais o cotidiano do ser humano em diversos aspectos, fazendo com que tudo e todos estejam conectados (CORDEIRO, M. 2019).

À princípio, esta ideia teve início no ano de 1999, quando o britânico Kevin Ashton propôs a comunicação e a coleta de informações sem a intervenção humana feita via computadores, o que otimizaria a produção da época. Após isto, com o avanço da tecnologia, tudo se tornou possível.

2.3 Telegram

De acordo com Silva e Brito (2018), o Telegram tem como foco a segurança e velocidade, além de ser compatível com todas as plataformas e totalmente integrado entre elas, ou seja, pode ser utilizado em diversos dispositivos ao mesmo tempo, sendo sincronizado a cada mensagem enviada.

Todos os dados enviados e recebidos são criptografados e armazenados na nuvem, fazendo com que não sejam decifrados quando interceptados por um provedor de serviço de internet.

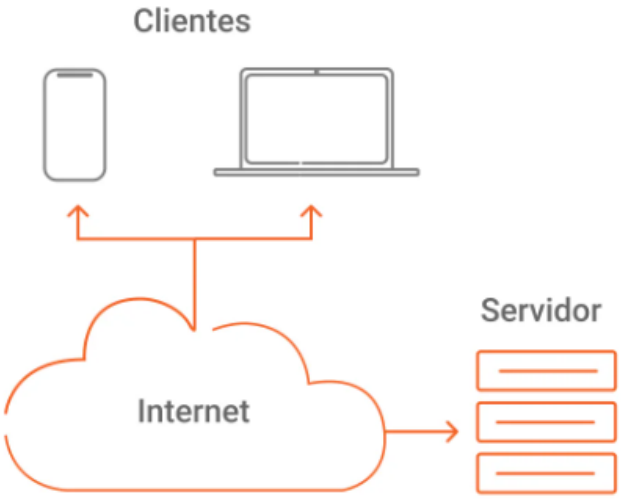
O aplicativo garantirá a troca de mensagens entre os clientes, fazendo com que toda troca de mensagens seja armazenada em seu banco de dados.

2.4 Protocolo HTTP

O HTTP (Hypertext Transfer Protocol) é um protocolo da camada de aplicação da internet, Kurose (2006) afirma que o protocolo HTTP é composto por duas partes: o programa cliente e o programa servidor. Esses programas são executados em máquinas diferentes e se comunicam através de mensagens HTTP. O HTTP define a estrutura e a forma como essas mensagens são trocadas entre o cliente e o servidor.

De forma simplificada, o cliente envia uma solicitação de um comando para o servidor por meio do protocolo HTTP, e o servidor responde ao cliente com o comando solicitado.

Figura 1: Protocolo HTTP.



Fonte: (TECHNOLOGIES, 2021).

2.5 ESP32

Criado em 2016, considerado o sucessor do ESP8266, o ESP32 é utilizado em larga escala por ter um custo relativamente baixo, além de materiais que auxiliam a sua programação devido ao seu crescimento contínuo. O microcontrolador possui duas espécies de comunicação sem fio, Wi-Fi e bluetooth; foi desenvolvido para aplicações móveis, eletrônicos portáteis e IOT, além disso é otimizado para consumir pouca energia durante sua operação (ESPRESSIF SYSTEMS, 2019).

Dentre suas características, temos:

- CPU: Xtensa® dual-core 32-bit LX6;
- Memória RAM: 520 KB;
- Memória ROM: 448 KB;
- Memória Flash: 4MB;
- Wi-Fi 2,4 GHz;
- Bluetooth BLE 4.2.

Figura 2: ESP32.

Fonte: (ELETROGATE,2021).

2.6 Servo Motor

O servo motor é um atuador que pode ser linear ou rotativo, permitindo controlar a velocidade da carga e sua posição. Possui um eixo de liberdade que realiza movimentos angulares de até 180 graus acionados por um comando externo.

O motor DC é conectado a uma caixa de engrenagens e ao eixo de saída para aumentar a velocidade e o torque do motor. O circuito interpreta os sinais enviados pelo controlador e o potenciômetro atua como feedback para o circuito controlador monitorar a posição do eixo de saída (GRACHTEN, 2022).

Figura 3: Servo motor.

Fonte: (TECNOLOGIA, s.d).

2.7 Célula de Carga

Feito geralmente de aço ou alumínio, é caracterizado como um transdutor que é capaz de medir força através da conversão das deformações mecânicas de um sensor do tipo *strain gauge*, quando sua resistência elétrica é alterada.

Este é feito de um fio de metal que é utilizado devido sua alta resistividade e capacidade de sofrer deformação sem se romper. Quando colado em um objeto o qual é submetido a uma força, o *strain gauge* se deforma ligeiramente, causando uma mudança na resistência elétrica do metal o qual é proporcional à deformação sofrida pelo objeto.

Quando se há deformação sob o efeito da carga, ele retorna à sua posição inicial, com uma resposta elástica a cada carga que podem ser medidas por ele. Com isso a deformação do *strain gauge* é interpretada pelo amplificador que permite determinar o peso.

2.8 ChatBot

Chatbots são ferramentas produtivas que facilitam e agilizam as atividades entre os indivíduos, essa facilidade ocorre pela comunicação natural em que se é desenvolvida a conversa. É possível também que chatbots consigam aprender novos conceitos com o próprio usuário através da conversação (POLATIDIS, 2014).

Segundo Kerly, Hall e Bull (2007), o chatbot é definido como um agente conversacional que através da linguagem convencional, é possível realizar negociações, fornecer informações e auxiliar usuários.

Seu funcionamento consiste no tratamento de uma dada mensagem enviada pelo usuário que é classificada de acordo com sua intenção, o módulo de reconhecimento de entidade obtém uma estrutura de dados contendo as informações na mensagem. Com base no contexto da conversação, o bot fornece as respostas possíveis pré-programadas e o seletor de resposta realiza a seleção da melhor resposta.

2.9 Real Time Clock (RTC)

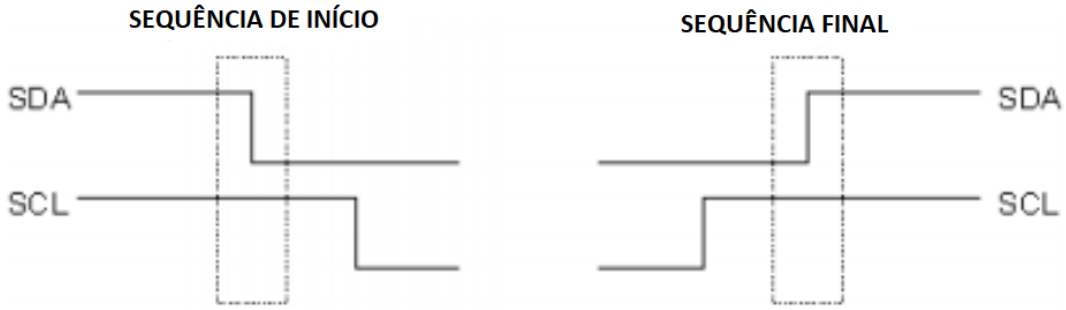
O relógio em tempo real é um chip que possui acoplado a ele uma bateria e é responsável por armazenar a hora, minutos, segundos, dia, mês e ano mesmo quando mesmo com o sistema desligado.

Possui um sistema programável de 56 bytes de RAM, consumindo menos de 500 nA, resultando em um baixo consumo e operando independente da energia principal da máquina.

Como base de comunicação, este utiliza o protocolo I²C (Inter Integrated Circuit) que permite a comunicação de diversos dispositivos no mesmo barramento sem conflito entre os sinais. Este possui o padrão mestre-escravo, o qual o mestre que geralmente possui o clock de sincronismo faz a requisição de dados através de outros processadores que são subordinados a ele.

De acordo com (Guo,F.,Zhang X., 2014), o barramento de comunicação do protocolo I²C utiliza de dois canais, o Serial Clock (SCL) gerado apenas pelo mestre do canal e servindo como sinal de sincronismo, e o Serial Date (SDA) sinal bidirecional que transporta a informação entre o mestre e os escravos. As informações são transportadas entre o start e o stop do sinal I²C, esses pontos são definidos pelo estado dos canais SCL e SDA.

Figura 4: Funcionamento RTC.



Fonte: Adaptado de (CASTRO, L. 2021).

Após o canal SDA ser comutado para nível baixo, gasta-se um período até que o SCL seja também levado para baixo. Assim, inicia-se o processo de comunicação, no qual o mestre envia o endereço do escravo a ser comunicado e, caso dispositivo exista, o mesmo irá retornar um aviso, ou um pulso no pino SCL, começando assim a transferência de dados entre o mestre e o escravo (CASTRO, L. 2021).

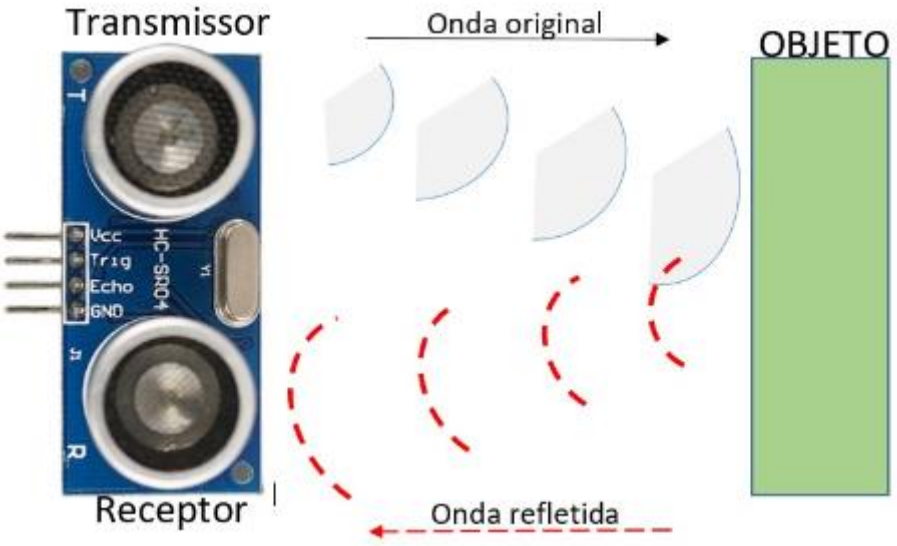
2.10 Sensor ultrassônico HC-SR04

Os sensores ultrassônicos funcionam enviando uma onda sonora em uma frequência acima do alcance da audição humana (por volta de 40kHz). Essa onda sonora viaja pelo ar e caso haja um objeto ou obstáculo em seu caminho, ela retornará esse sinal ao módulo. O *range* de leitura do sensor é de 2 cm a aproximadamente 4 metros.

Um sensor ultrassônico usa um transdutor para enviar e receber pulsos ultrassônicos que retransmitem informações sobre a proximidade de um objeto.

O transdutor do sensor atua como um microfone para receber e enviar o som ultrassônico. O sensor determina a distância até um alvo medindo os lapsos de tempo entre o envio e o recebimento do pulso ultrassônico (BARRA, 2021).

Figura 5: Funcionamento sensor ultrassônico.



Fonte: Adaptado de (STA ELETRÔNICA, s.d.).

3. METODOLOGIA

Visando o auxílio aos donos para cuidarem de seu animal da maneira mais saudável, além da possibilidade de realizar o monitoramento referente a alimentação, que posteriormente possa ser analisada em uma consulta veterinária, o desenvolvimento do projeto foi realizado em partes com o intuito de minimizar os erros durante a construção do protótipo e facilitar a organização do mesmo.

Este projeto foi construído para gatos e cachorros e pequeno porte,

podendo ser adaptável para animais de outros portes, além da utilização de outras ferramentas que forneçam maior comodidade ao usuário.

As configurações utilizadas foram de acordo com o perfil do animal, obedecendo critérios sobre a alimentação e saúde do mesmo.

3.1 Dimensionamento

A primeira etapa realizada foi o dimensionamento e material do reservatório de ração que será utilizado, como este é um projeto voltado a animais de pequeno porte, a quantidade de refeições e ração é de suma importância para evitar o excesso ou falta de ração armazenada no reservatório do alimentador.

O material do reservatório escolhido o acrílico, além de possuir uma tampa e aleta acoplada, pois através da furação, é possível passar a fiação, e a aleta servirá como dosador de ração que será depositada no pote, além de possuir boa resistência e alta proteção contra a umidade, preservando a qualidade e sabor do alimento. O reservatório possui as seguintes dimensões:

- Altura: 18 cm;
- Volume: 1,4 Litro.

Figura 6: Reservatório de ração.

Fonte: (SHOPEE, 2019).

Para avaliar quantos dias o reservatório irá abastecer o pote de ração, é necessário levar em consideração a quantidade de ração que será despejada em um ciclo de refeição e a ração que será utilizada. A ração escolhida foi a PREMIER PET para gatos castrados, que possui um floco de 5mm.

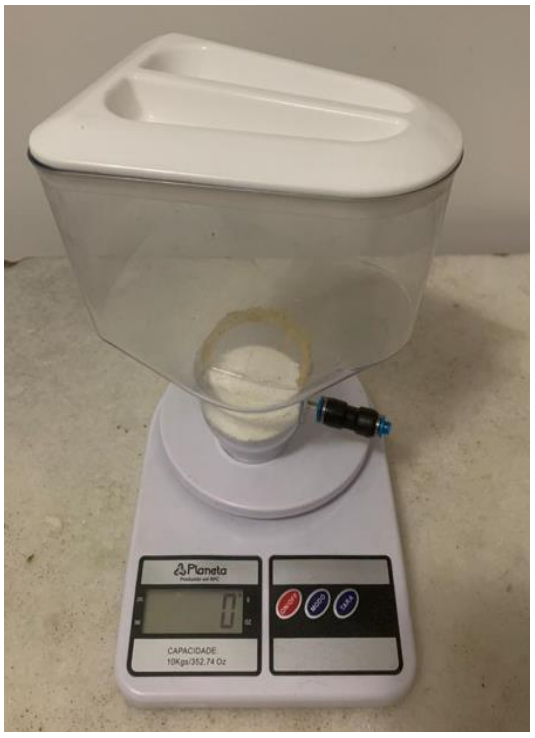
Como a quantidade de ração consumida por um animal de pequeno porte é baixa, não houve a necessidade de utilizar um reservatório com alta capacidade de armazenamento. Vale ressaltar que a litragem deste é adaptável ao porte do animal que será alimentado.

A configuração realizada é que o animal será alimentado 3 vezes ao dia, que cada ciclo deposita cerca 30g de ração no pote em que o animal irá comer e que o reservatório possui capacidade para 600 g de ração.

Para realizar a medição da quantidade de ração que o reservatório de 1,4 litro é capaz de armazenar, foi feita a pesagem da mesma em uma balança convencional, primeiramente foi pesado o reservatório vazio e feito a sua tara, após isso, com o reservatório preenchido por ração na sua capacidade máxima para obter-se a quantidade de ração que o reservatório poderá armazenar.

As figuras 7 e 8 mostram como a pesagem foi realizada.

Figura 7: Reservatório vazio.



Fonte: (Elaborado pelo autor, 2023).

Figura 8: Reservatório cheio.

Fonte: (Elaborado pelo autor, 2023).

Com a pesagem feita, é possível realizar o cálculo da quantidade de dias que o reservatório conseguirá ficar sem ser abastecido.

Considerando que:

- A = Capacidade de armazenamento do reservatório;
- C = Números de ciclos;
- N = Número de dias;
- R = Quantidade de ração por ciclo.

Com isso, temos que:

$$N = \frac{A}{C \times R} = \frac{603g}{3 \times 30g} \approx 7dias$$

Logo, conclui-se que com este protótipo, para a alimentação, o dono poderá ficar sem abastecer o alimentador por aproximadamente sete dias, e quando comparado com o método diário realizado manualmente, torna-se altamente vantojoso.

3.2 Estrutura Mecânica

A próxima etapa foi a construção da estrutura do alimentador. Como se trata de um projeto que busca o menor custo de construção, esta foi baseada em equipamentos que são relativamente baratos e de fácil acesso.

Sua estrutura foi feita em acrílico que possui 6 mm de espessura e servirá de suporte para o reservatório e a célula de carga utilizada. O acrílico foi dimensionado e chapeado para que o alimentador não ocupe muito espaço. Foram utilizadas chapas com as seguintes dimensões:

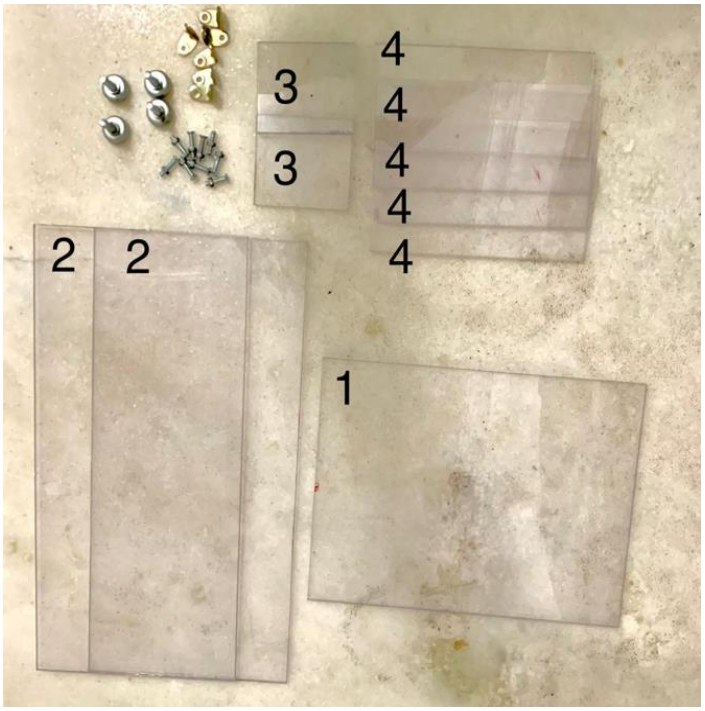
Tabela 1: Identificação das chapas acrílicas.

Número da peça	Número de unidades	Dimensões
1	1	30 x 22 cm
2	2	40 x 18 cm
3	2	8 x 8 cm

4	5	18 x 8 cm

Fonte: (Elaborado pelo autor, 2023).

Figura 9: Chapas em acrílico.



Fonte: (Elaborado pelo autor, 2023

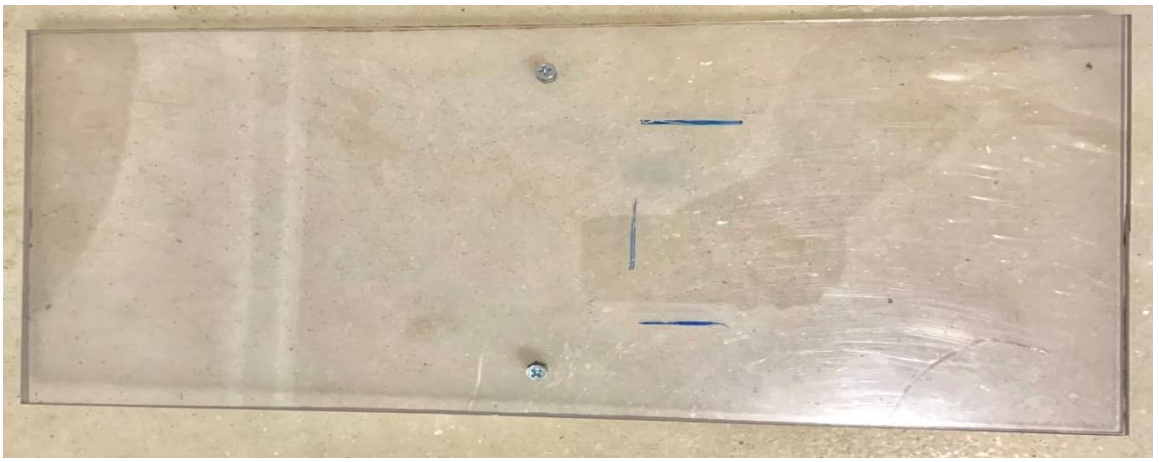
A chapa de número 1 será a base do alimentador, ela sustentará toda a estrutura do projeto e conta com quatro bases de borracha reguláveis que possibilitam a regulagem do alimentador em relação ao solo, para que caso haja contato com a água, não danifique a estrutura.

Figura 10: Base em acrílico.

Fonte: (Elaborado pelo autor, 2023).

Após a montagem da base, será montado o suporte de sustentação do reservatório de ração. As chapas de número 2 foram previamente parafusadas uma na outra para que fique com uma espessura maior e forneça maior rigidez a estrutura. Para a fixação foram utilizados dois parafusos e duas roscas de 5 mm cada.

Figura 11: Fixação das chapas acrílicas.



Fonte: (Elaborado pelo autor, 2023).

Nota-se que possui uma marcação feita a pincel na peça, esta se remete a fixação do suporte plástico que será parafusado no acrílico para que o reservatório de ração se encaixe a ele (peça acompanha o reservatório). O suporte foi fixado cerca de 15 cm em relação ao topo do acrílico, para que quando o reservatório for encaixado, fique rente ao suporte acrílico.

Figuras 12 e 13: Encaixe no suporte acrílico.

Fonte: (Elaborado pelo autor, 2023).

Fonte: (Elaborado pelo autor, 2023).

A etapa seguinte foi a construção da caixa onde ficarão os componentes eletrônicos (chapas 3 e 4), que conta com uma tampa para que fique de fácil acesso aos componentes ali presentes e os proteja de agentes externos. Para a montagem, foi utilizado adesivo instantâneo e posteriormente silicone, para melhor vedação da caixa.

Figura 14: Caixa em acrílico.

Fonte: (Elaborado pelo autor, 2023).

Com a montagem realizada, utilizando parafusos e roscas, foi feita a fixação da caixa na base, de forma que fique sólida, sendo utilizado dois parafusos e duas roscas de 5 mm.

Figura 15: Fixação da caixa sobre a base.

Fonte: (Elaborado pelo autor, 2023).

Partindo para a última etapa de montagem, com a utilização de cantoneiras, as chapas de número 2 foram parafusadas na base e na caixa dos componentes.

Figura 16: Estrutura mecânica montada.

Fonte: (Elaborado pelo autor, 2023).

A tabela 2 apresenta o custo dos materiais utilizados:

Tabela 2: Custo componentes estrutura mecânica.

Componente	**Valor**	**Fornecedor**
Reservatório acrílico 1,4 litro	R$ 28,27	Shopee
Chapas acrílicas (compra e corte)	R$ 150,00	Marcenaria
Cantoneiras, parafusos e pés emborrachados	R$ 19,00	Eletro Betim
Total	R$197,27	

Fonte: (Elaborado pelo autor, 2023).

3.3 Montagem da estrutura eletrônica

Para a implementação do sistema eletrônico, algumas condições foram utilizadas para que o alimentador funcione corretamente.

Toda forma de comunicação entre o usuário e alimentador será feita por comandos através do ChatBot criado no Telegram.

O início parte do armazenamento de dados que ocorre através da declaração das variáveis e valores configurados pelo usuário na programação. Com isso, estes dados são enviados ao ESP32 que será responsável por armazená-los.

Desta forma, toda requisição feita pelo cliente, o Telegram as receberá e irá fazer o armazenamento das mensagens. A partir disso, estas mensagens serão encaminhadas ao ESP32 que fará uma comparação com seu banco de dados, caso elas sejam compatíveis, a requisição será realizada de acordo com a configuração realizada.

A balança faz a leitura instantânea da ração que tem no pote, o peso do pote não irá ultrapassar o valor configurado na lógica de programação (30 g).

O servo motor é acionado toda vez que for o horário de alimentação ou se o usuário pretender alimentar o animal naquele exato momento. Vale ressaltar que mesmo que o pote já tenha ração, a ração é depositada até atingir o peso configurado.

O RTC irá garantir que o horário esteja sempre atualizado para que não haja erro nos horários configurados.

O sensor ultrassônico fará a leitura da quantidade de ração no reservatório toda vez que solicitado.

3.3.1 *Balança para pesagem do pote com ração*

Para pesar a ração presente no pote de comida do animal, foi utilizado uma célula de carga com um range de medição de 0 a 5 kg com 1.0 mv/v ± 0.1 mv/v de sensibilidade de saída.

Para amplificar seu sinal e converter os sinais analógicos em digitais, utilizou-se o módulo conversor e amplificador HX711. A ligação destes foi feita com a solda de fios, o esquema de ligação se encontra na tabela 3.

Tabela 3: Esquema de ligação Célula de carga e módulo HX711.

Célula de carga	**Módulo HX711**
Fio Vermelho	Pino E+
Fio Preto	Pino E
Fio Branco	Pino A+
Fio Verde	Pino A-

Fonte: (Elaborado pelo autor, 2023).

Com a conexão feita, este será conectado no ESP32 com o seguinte esquema de ligação:

Tabela 4: Esquema de ligação módulo HX711 e ESP32.

Módulo HX711	**ESP32**
Pino GND	Pino GND
Pino DT	Pino D23
Pino SCK	Pino D19
Pino VCC	+3,3V

Fonte: (Elaborado pelo autor, 2023).

Para que se possa obter o valor real do objeto sobre a balança, é necessário a calibração da célula de carga, de acordo com (STRAUB, 2019), a calibração é feita para se obter de um valor de escala, que será compilado no ESP32 (autor disponibiliza o código utilizado para calibração). Este método consiste em:

No primeiro passo é feita a conexão da célula de carga ao microcontrolador.

Posteriormente é realizado uma média abrangente de oito valores obtidos através da pesagem feita pela célula de carga vazia e sem calibrar.

A escala será obtida após a divisão da média encontrada pelo valor de um peso padrão qualquer.

No código disponibilizado (Apêndice A), utiliza-se na função "*calibracao_celula_de_carga*" o valor encontrado da escala, que foi obtida anteriormente.

Vale ressaltar que os padrões de pesagem se diferem em razão do método

de instalação e que cada balança e cada célula são diferentes uma das outras.

Para a calibração, foi utilizado um objeto padrão para realizar o teste. Na tabela 5 temos o valor da média e escala obtidos.

Tabela 5: Valores obtidos na calibração.

Parâmetros	**Objeto Padrão - 0,395kg**
Média abrangente	145.306,37
Escala	367.864.24

Fonte: (Elaborado pelo autor, 2023).

O valor de escala encontrado foi 367.864.24, com o valor inserido na função "*calibracao_celula_de_carga*", o código foi compilado e uma medição com os objetos foi realizada para sua validação.

Para a validação destes resultados, é necessário o cálculo do desvio padrão para encontrar a margem de erro. Este processo foi realizado utilizando pesos distintos para melhores resultados.

Os pesos utilizados possuem valor padrão de 5 g, 10 g e 20 g, estes se encontram no laboratório do próprio campus. Os valores são baixos pois foram baseados na quantidade de ração depositada por refeição (30 g).

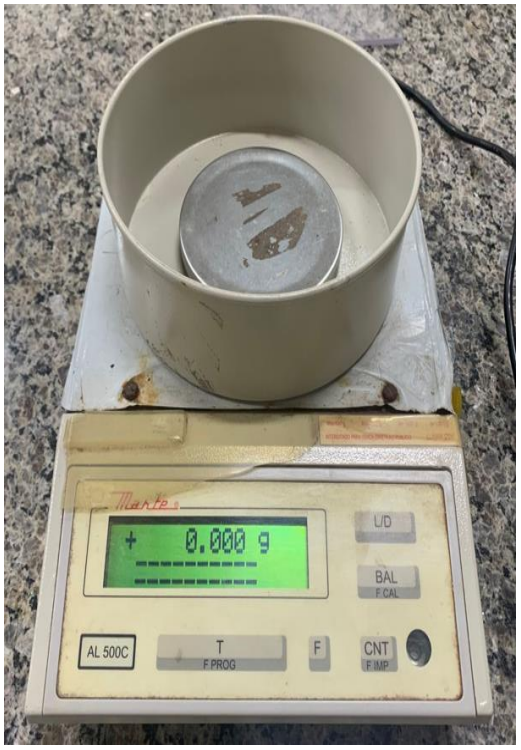
Figura 17: Peso padrão para balança.

Fonte: (Elaborado pelo autor, 2023).

Para comparação das medidas foi utilizado uma balança semi analítica MARTE AL500c, que possui precisão de "0,001 g", com o código de verificação pelo

INMETRO (N° 2.898.864), patrimônio N° 004042, disponibilizada pelo campus do IFMG BETIM.

Figuras 18 e 19: Balança semi analítica.



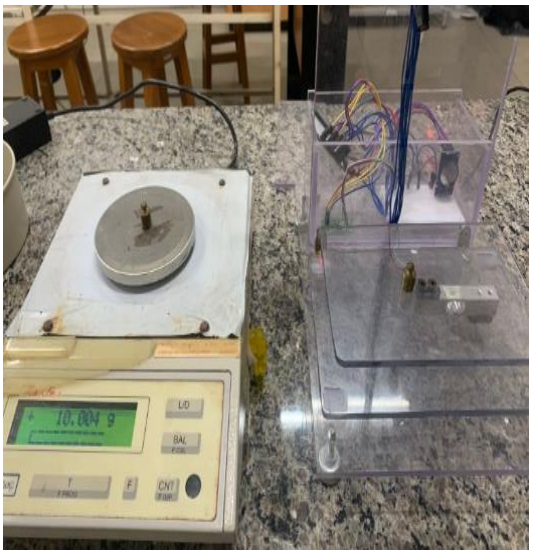
Fonte: (Elaborado pelo autor, 2023).

Fonte: (Elaborado pelo autor, 2023).

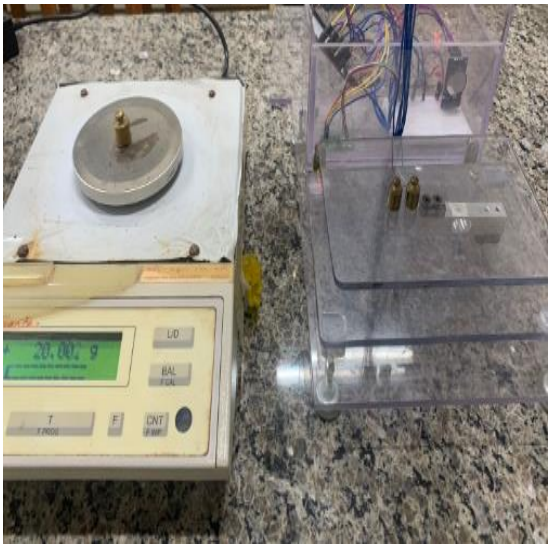
Como a balança construida possui exatidão de um 1 g, não foi possível obter valores precisos tal quanto a balança semi analítica, pois além do projeto não exigir tais valores, ambas não possuem a mesma exatidão, e caso seja depositado algumas gramas a mais por refeição, não será prejudicial para o animal, porque a quantidade de ração depositada por refeição foi previamente configurada com fundamento na falta de exatidão.

As figuram 20 e 21 indicam a pesagem realizada através da balança do protótipo:

Figuras 20 e 21: Balanças.



Fonte: (Elaborado pelo autor, 2023).



Fonte: (Elaborado pelo autor, 2023).

Foram registados 20 valores com cada peso, os resultados podem ser visualizados na figura 22.

Figura 22: Valores obtidos pela balança.

Tabela Validação							
Valores de Referência	**0,000kg**	**0,005kg**	**0,010kg**	**0,015kg**	**0,020 kg**	**0,025 kg**	**0,030kg**
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,024	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,004	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	-0,001	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,014	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	-0,001	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,004	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
	0,000	0,005	0,010	0,015	0,020	0,025	0,030
Desvio Padrão	**3,08E-04**	**3,08E-04**	**1,78E-18**	**2,24E-04**	**3,56E-18**	**2,24E-04**	**1,78E-17**
Média Desvio Padrão	**1,30E-04**						

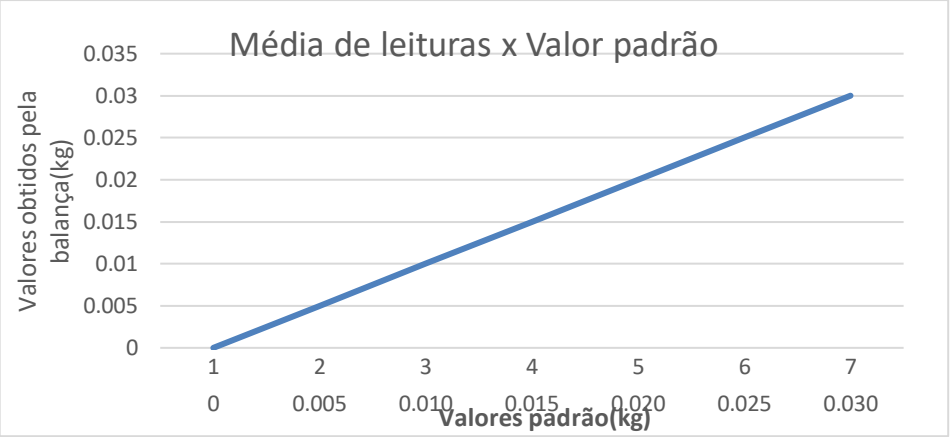
Fonte: (Elaborado pelo autor, 2023).

Os valores mostrados na tabela estão representados em quilogramas e os cálculos de desvio padrão foram realizados no Excel através da fórmula “=DESVPAD.A” (Aproximações foram realizadas pelo próprio Excel, portanto pode

haver divergências nos valores obtidos). Podemos notar que no ato da pesagem não foi encontrado divergência nas medidas, portanto é possível concluir que a balança apesar de não ser exata quanto a balança semi analítica, consegue ser precisa nas pesagens realizadas.

Um gráfico de linearidade foi utilizado para comparar os valores de peso padrão com a média dos valores obtidos na pesagem.

Gráfico 1: Gráfico de linearidade.



Fonte: (Elaborado pelo autor, 2023).

3.3.2 *Servo motor e ESP32*

O motor utilizado possui 13 kg de torque e possui um ângulo de rotação de até 180 graus. A angulação configurada (Apêndice A) foi baseada na quantidade de ração a ser depositada no pote. Como o peso por refeição é baixo, optou-se por despejar menos ração por acionamento. Vale lembrar que na programação, o servo volta para a posição inicial (0 graus) toda vez que completa um ciclo de acionamento. Sua ligação com o ESP32 está mostrada na tabela 6.

Tabela 6: Ligação Servo Motor.

Servo Motor	**ESP32**
Pino GND	Pino GND
Pino SINAL	Pino D2
Pino VCC	+3,3V

Fonte: (Elaborado pelo autor, 2023).

3.3.3 *Sensor ultrassônico e níveis de medição*

O sensor ultrassônico terá a função de monitorar o nível de ração do reservatório com o intuito de informar o usuário sobre a quantidade de ração presente no reservatório.

A configuração realizada foi baseada no tamanho do reservatório (16 cm) e o quão distante a ração se encontra do sensor.

A relação entre nível e distância é mostrado na tabela 7:

Tabela 7: Relação distância X nível sensor ultrassônico.

Distância	**Nível de Ração**
< 2 cm	ALTO
>= 2 cm, <= 12 cm	MÉDIO
> 12 cm	BAIXO

Fonte: (Elaborado pelo autor, 2023).

Com a relação feita, o sensor é conectado ao microcontrolador como mostra a tabela 8.

Tabela 8: Esquema de ligação sensor ultrassônico.

Sensor Ultrassônico	**ESP32**
Pino GND	Pino GND
Pino TRIGGER	Pino D4
Pino ECHO	Pino D5
Pino VCC	Pino VIN (5V)

Fonte: (Elaborado pelo autor, 2023).

3.3.4 *RTC*

O RTC irá garantir que mesmo que o protótipo fique sem energia externa, o horário atual não seja desconfigurado. Isso é possível através de uma bateria que é acoplada a ele.

A ligação deste é mostrada na tabela 9:

Tabela 9: Esquema de ligação RTC.

RTC	ESP32
Pino GND	Pino GND
Pino SCL	Pino D21
Pino SDA	Pino D22
Pino VCC	+ 3,3 V

Fonte: (Elaborado pelo autor, 2023).

3.4 Configuração do ChatBot

A criação do ChatBot no aplicativo Telegram é totalmente gratuito e de fácil manuseio, a própria plataforma fornece as instruções necessárias para que o usuário consiga configurar seu próprio Bot em poucos minutos.

Em (FERNANDO, 2019), temos um projeto semelhante que explica passo a passo como se criar um ChatBot pelo Telegram para se comunicar com o ESP32, todo processo de programação foi seguido e se encontra apêndice.

Seu funcionamento acontece através do envio de comandos feitos pelo usuário, que através do protocolo HTTP (disponível no item 2.4) irá fazer a solicitação e comunicação com os componentes utilizados.

Essa interação ocorre através de bibliotecas capazes de interagir com a API do Telegram Bot. A “UniversalTelegramBot.h” é uma biblioteca que facilita o desenvolvimento de bots do Telegram, permitindo enviar e receber mensagens, gerenciar comandos e consultar informações. Ela fornece uma interface simplificada, abstraindo muitos dos detalhes de baixo nível que possibilita uma simples implementação das funcionalidades do ChatBot.

Dentre suas funcionalidades, (OpenAI., 2023) destaca as seguintes:

- Envio e recebimento de mensagens: A biblioteca permite enviar mensagens para usuários ou grupos no Telegram e também receber mensagens de entrada. Isso permite que o usuário crie bots que possam responder a comandos ou interagir com os usuários;
- Gerenciamento de comandos personalizados: O usuário pode definir comandos personalizados para o seu bot e associar ações a esses comandos, facilitando a implementação de funcionalidades específicas que podem ser acionadas

pelos usuários através de comandos;

- Manipulação de tipos de mensagem: A biblioteca suporta vários tipos de mensagem, como texto, fotos, áudios, documentos, entre outros. Isso permite que o usuário processe e responda a diferentes tipos de conteúdo de forma adequada;
- Consultas e interações com o usuário: É possível receber e responder a consultas de botões interativos, possibilitando a criação de fluxos de interação mais avançados com os usuários;
- Gerenciamento de atualizações: A biblioteca cuida da lógica de receber e processar atualizações da API do Telegram.

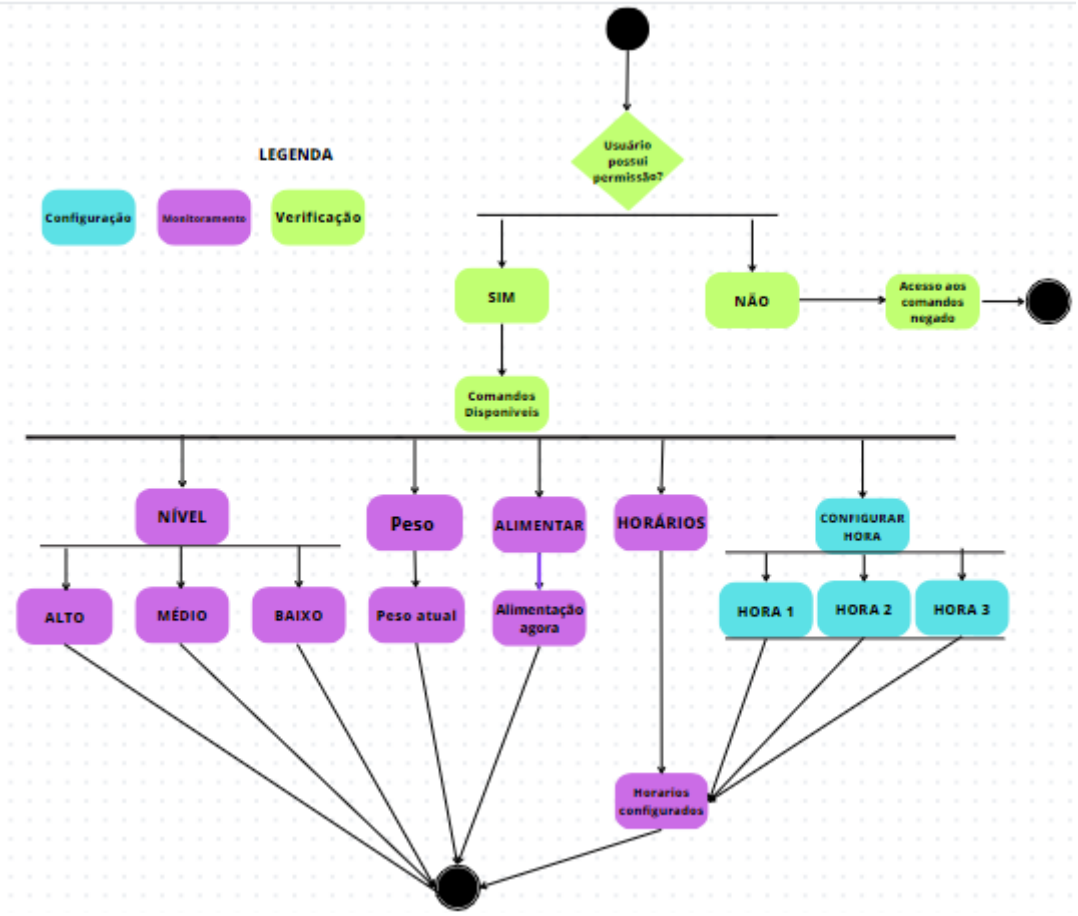
Por motivo de segurança, o chat funcionará apenas se o usuário tiver o ID previamente cadastrado, caso contrário, o acesso para comunicação com o alimentador é negado.

Os comandos disponíveis são:

- NÍVEL: Para acessar o nível de ração do reservatório;
- PESO: Para ver o peso atual do pote;
- ALIMENTAR: Para alimentar o pet;
- HORÁRIOS: Para acessar os horários de alimentação do seu pet;
- CONFIGURAR HORA: Para configurar o horário de alimentação.

O fluxograma a seguir demonstra o funcionamento do protótipo.

Figura 23: Fluxograma.



Fonte: (Elaborado pelo autor, 2023).

A tabela 10 mostra os custos dos componentes eletrônicos utilizados.

Tabela 10: Custo componentes eletrônicos.

Componente	Valor	Fornecedor
Célula de carga com HX711	R$ 30,00	Shopee
Servo motor	R$ 52,27	Mercado Livre
Real Time Clock (RTC)	R$ 25,00	Shopee
ESP32	R$ 39,24	Shopee
Jumpers e Protoboard	R$ 24,90	Shopee
Sensor Ultrassônico	R$7,99	Shopee
Total	R$ 179,40	

Fonte: (Elaborado pelo autor, 2023).

4. RESULTADOS

Este capítulo apresenta os resultados obtidos através da construção do projeto, demonstrado no capítulo anterior.

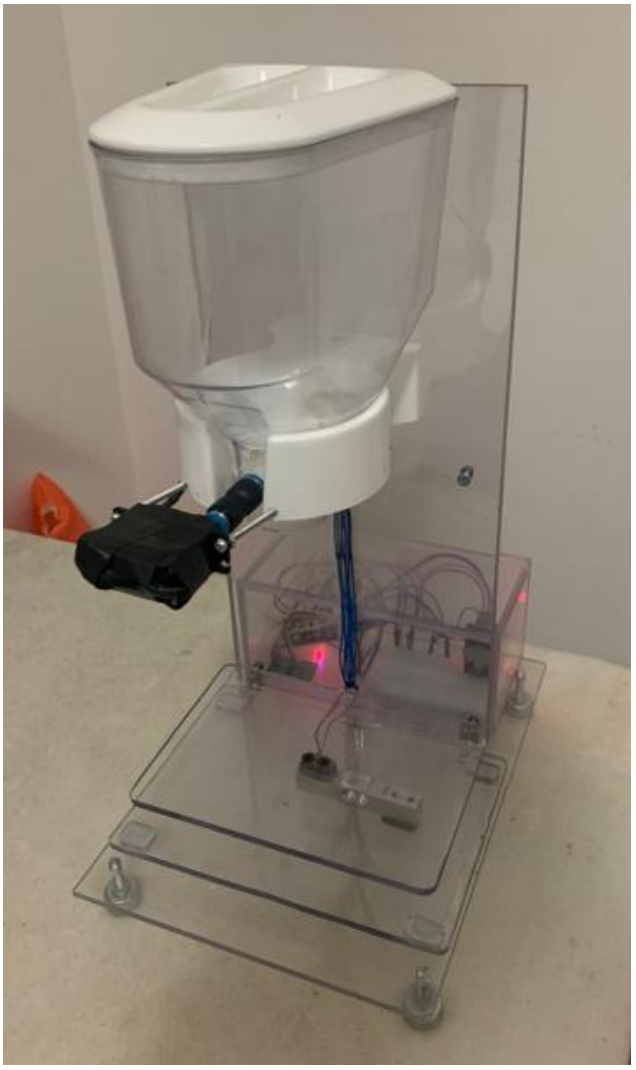
4.1 Estrutura mecânica e eletrônica

Após a montagem da estrutura e a conexão com os componentes eletrônicos, o projeto pode ter seu primeiro funcionamento. A conexão entre os componentes do projeto foi feita por jumpers e foram separadas para cada componente na intenção de ajudar na identificação caso haja algum defeito. O intuito de utilizá-los juntamente com o acrílico foi de deixar o protótipo visivelmente mais atrativo.

Ao ligá-lo em uma fonte de na tomada, o ESP32 irá se conectar automaticamente na rede Wifi configurada, porém após a realização de testes, notamos que em algumas ocasiões, ao ligá-lo, o ESP32 não conectou automaticamente a rede. Para solucionar este problema basta pressionar o botão reset (EN) do ESP32 por 3 segundos, que uma nova tentativa de conexão será realizada. Caso o problema persista, é necessário checar se o nome da rede cadastrada no código está correto.

A caixa (Figura 14) projetada irá armazenar o ESP32 e demais componentes utilizados e posteriormente tampada para evitar corrosão e umidade. Esta foi furada para a passagem do cabo de alimentação do protótipo e jumpers.

Figura 24: Alimentador montado.



Fonte: (Elaborado pelo autor, 2023).

A conexão do motor com a aleta do reservatório foi um processo que demandou certo esforço, pois como o encaixe das duas peças eram diferentes, foi adaptado primeiramente um tubo de aço com duas saídas (Figura 12) de diâmetro de 8mm para a aleta de 6mm e fixados através de parafusos para que quando o motor fosse acionado, a aleta também seria.

Porém, com os testes realizados, foi observado que quando o reservatório está cheio de ração, o motor não tinha força o suficiente para girar a aleta. Inicialmente foi utilizado um motor de 1,2 kg de torque, posteriormente com um motor de 13 kg de torque (conforme citado no item 3.0.2.2) e a adaptação da conexão do servo motor e a aleta por um engate rápido pneumático (pode ser visto nas imagens 7 e 8) com saídas de 8 mm, o problema foi solucionado parcialmente.

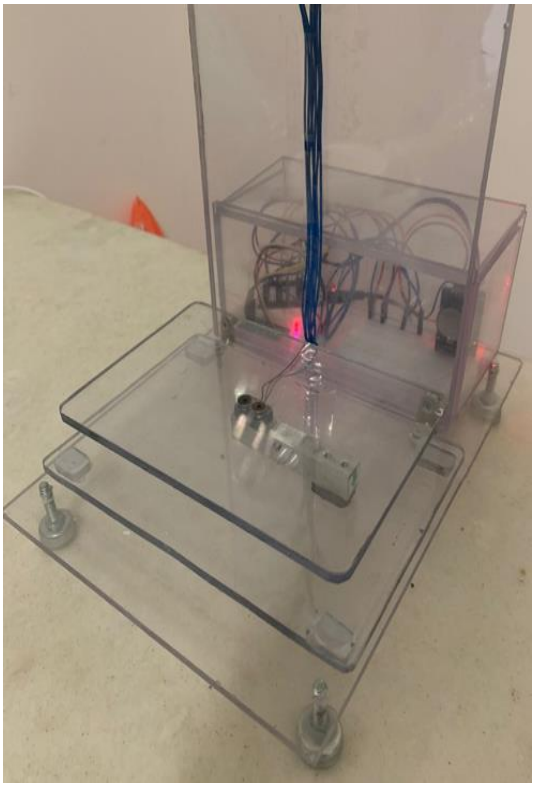
Figura 25: Conexão do motor.

Fonte: (Elaborado pelo autor, 2023).

A balança foi feita por duas chapas em acrílico e foi fixada sob a base do alimentador por fita dupla face, o pote que fica sobre ela também é colado com dupla face para não ter o risco de cair no chão, além de ser previamente tarado toda vez que o alimentador é reinicializado.

O módulo HX711 foi posicionado dentro da caixa e soldado nos cabos da célula da carga, pois a mesma não pode ser conectada por jumpers. O resultado foi satisfatório e a mesma não apresentou problemas de medição após testada.

Figuras 26 e 27: Balança.

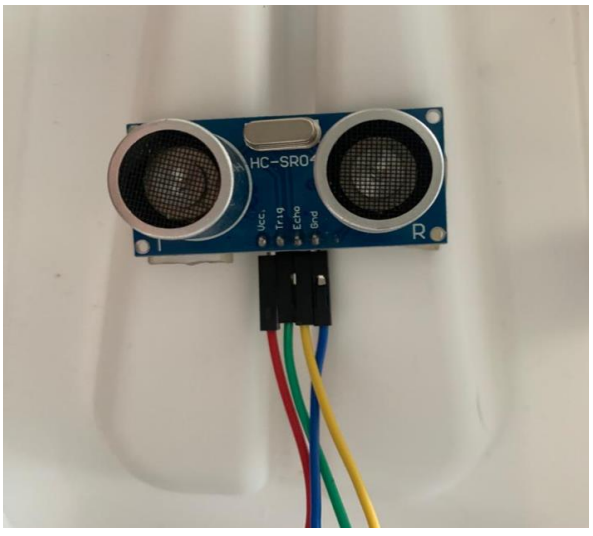


Fonte: (Elaborado pelo autor, 2023).

Fonte: (Elaborado pelo autor, 2023).

O sensor ultrassônico foi fixado também com fita dupla face na tampa do reservatório de maneira que não atrapalhasse o usuário abrir a tampa para despejar a ração e a ligação com os jumpers não se rompa, para isso os jumpers conectados a ele ficaram com folga, para que não haja este inconveniente.

Figura 28: Sensor ultrassônico.



Fonte: (Elaborado pelo autor, 2023).

O RTC é conectado junto ao protoboard e ficam dentro da caixa devidamente organizados

4.2 ChatBot

Com o desenvolvimento do ChatBot e toda programação referente a ele finalizada, foram realizados testes práticos a partir da comunicação com o usuário.

O ChatBot criado se chama “LTPet”, possui uma mensagem de saudação e como foto de perfil, uma gata.

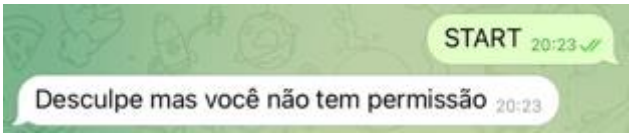
Figura 29: Perfil ChatBot.

Fonte: (Elaborado pelo autor, 2023).

Para a interação com o ChatBot, como dito anteriormente, o usuário deve ser previamente cadastrado. Para obter o cadastro, é necessário a informação do ID de cada usuário que deseja o acesso aos comandos do alimentador. Cada usuário do Telegram possui um ID, e é necessário anexá-lo ao código.

Caso o usuário não tenha a permissão, uma mensagem de aviso será enviada a ele impedindo sua interação.

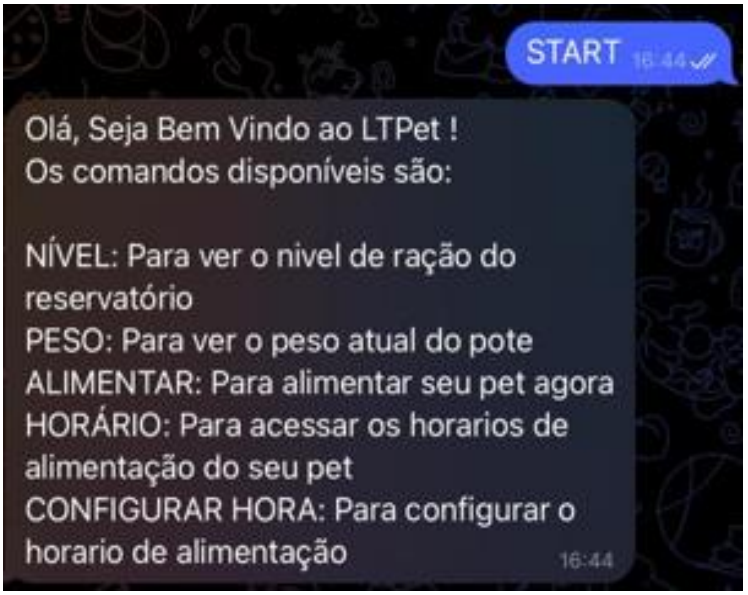
Figura 30: Usuário sem permissão de acesso.



Fonte: (Elaborado pelo autor, 2023).

Com o acesso autorizado, o usuário pode acessar as funções disponíveis (Item 3.1). O comando utilizado para acessá-las é o “START”. Vale ressaltar que todos os comandos devem ser escritos em CAIXA ALTA, caso contrário, o ChatBot não retorna nenhuma mensagem.

Figura 31: Comando START.

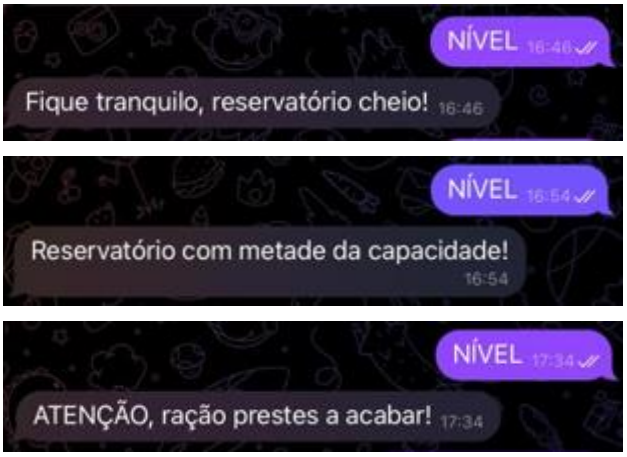


Fonte: (Elaborado pelo autor, 2023).

O ChatBot executa apenas um comando por vez e que devido a conexão este pode ter um delay até retornar às mensagens.

A primeira função disponível é a “NÍVEL”, que é utilizada para acessar a quantidade de ração existente no reservatório. Conforme a tabela 7, três níveis estão disponibilizados e a partir da quantidade de ração, uma mensagem é enviada ao usuário com a informação.

Figura 32: Comando NÍVEL.

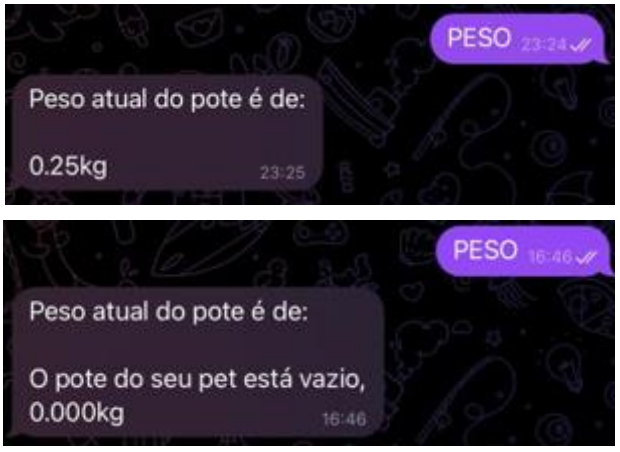


Fonte: (Elaborado pelo autor, 2023).

A função “PESO” mostra o quanto de ração (em quilogramas) o pote do animal possui. Essa função foi criada com o intuito de informar se o pet está comendo a ração entre os intervalos programados de alimentação.

A tara do pote é realizada toda vez que o ESP32 se conecta a internet, já a quantidade de ração a depositada pode ser configurada via código.

Figura 33: Comando PESO.



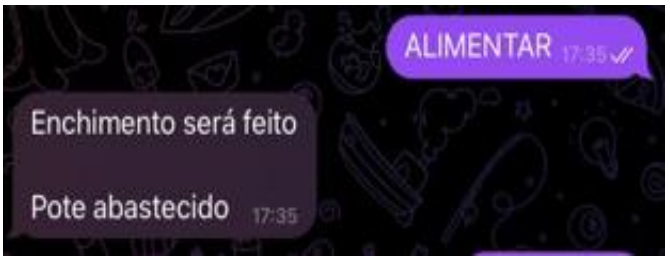
Fonte: (Elaborado pelo autor, 2023).

Caso queira alimentar o animal fora dos horários configurados, a função “ALIMENTAR” irá realizá-la. Esta função pode ser acionada a qualquer momento e fará a alimentação até atingir a quantidade de ração configurada (Item 3.0.1), desde que o pote esteja vazio ou abaixo do peso previamente configurado. Caso o pote já esteja cheio, o motor não será acionado.

Após o comando ser realizado, uma mensagem será retornada ao usuário

informando que o pote foi abastecido.

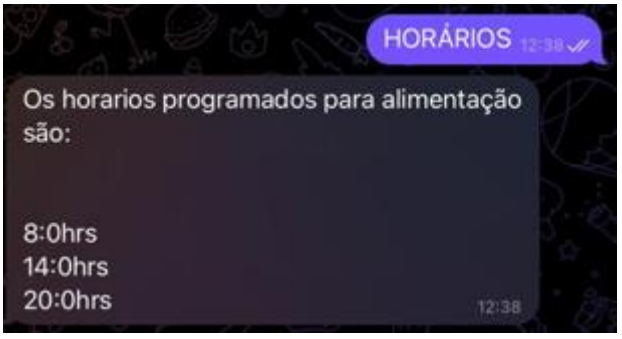
Figura 34: Comando ALIMENTAR.



Fonte: (Elaborado pelo autor, 2023).

Para o monitoramento dos horários que serão realizados a alimentação, a função “HORÁRIOS” mostra ao usuário os horários configurados. O alimentador possui três horários padrão pré configurados e no código disponibilizado é possível alterá-los.

Figura 35: Comando HORÁRIOS.



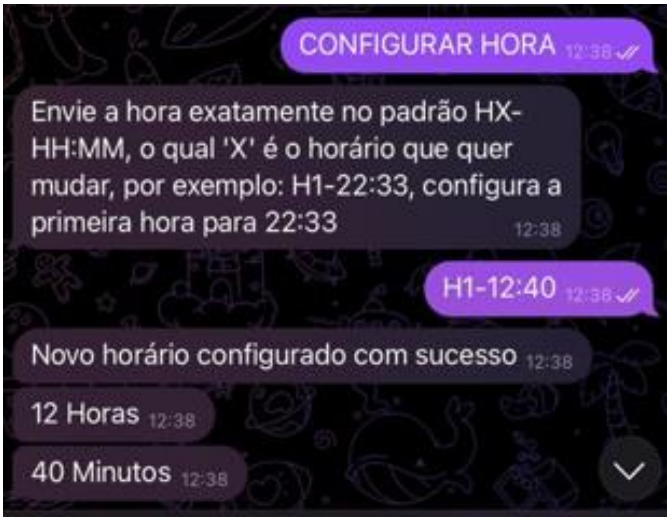
Fonte: (Elaborado pelo autor, 2023).

Caso o usuário deseje trocar os horários, através da função “CONFIGURAR HORA” é possível configurá-los. Uma mensagem de instrução aparecerá indicando o formato que horário deve ser digitado.

Os horários devem ser configurados de forma que o primeiro horário seja anterior ao próximo e assim sucessimente, por exemplo: Hora 1 - 8:00, Hora 2 - 14:00 e Hora 3 - 20:00.

Vale lembrar que é possível adicionar quantos horários o usuário desejar, porém para este projeto foram considerados apenas três.

Figura 36: Comando CONFIGURAR HORA.



Fonte: (Elaborado pelo autor, 2023).

5. CONCLUSÃO E FUTURAS MELHORIAS

Após a realização de testes e a utilização do mesmo por alguns dias, a comodidade e conforto oferecido ao usuário ao animal é satisfatória. Todas as etapas citadas ao longo do projeto foram concretizadas.

Quando comparado com alimentadores do mercado nacional, apesar de ser um projeto caseiro, este possui funções que viabilizam sua inserção no meio comercial, além do preço que seria disponibilizado, já que caso seja feito em grande escala seu preço diminuirá cada vez mais.

Na tabela 11 temos um comparativo de preço e funções dos alimentadores disponíveis no mercado.

Tabela 11: Comparativos alimentadores automatizados.

Funções	**LTPet**	**Kabum**	**BellaPets**	**MovimentoPet**	**Soluartz**
					
Alimentar a distância	✓	✓	✓	✓	✓
Câmera e microfone	✗	✓	✗	✓	✓

Agendar horário de refeição	✓	✓	✓	✓	✓
Monitorar nível de ração	✓	✗	✗	✗	✓
Configurar quantidade de ração	✓	✓	✓	✓	✓
Peso atual de ração no pote	✓	✗	✗	✗	✗
Controle de acesso de terceiros	✓	✗	✗	✗	✓
Valor	R$376,67	R$469,99	R$509,90	R$946,68	R$2.789,00

Fonte: (Elaborado pelo autor, 2023).

Na tabela comparativa nota-se que devido seu valor em relação aos demais e mesmo que não tenha algumas funcionalidades que os outros possuem, faz-se possível sua inserção no mercado e através da implantação de mais componentes referentes ao monitoramento, ele pode ser mais vantajoso para aquisição embora seu preço possa aumentar devido as implementações realizadas.

Os testes realizados foram com um tamanho de ração específico, então para possíveis problemas relacionados é necessário a utilização rações de diferentes tamanhos.

A gata utilizada no teste é calma e não se incomodou com o barulho do motor quando acionado. Cães e gatos possuem comportamento diferente a esta situação, para os mais ariscos pode haver a tentativa de danificar o alimentador.

O custo total é admissível, pois os valores encontrados foram pesquisados visando o menor gasto possível, além de serem produtos com ótimo custo x benefício.

A utilização de jumpers e protoboard não são as melhores opções, pois além de serem frágeis não fixam com muita precisão, além da possibilidade de manutenção ao longo do tempo. Uma possível solução é a montagem de uma placa de circuito impresso, pois além de ser mais tecnológica, possui maior durabilidade.

Como melhorias futuras, é possível a instalação de componentes que aumentem ainda mais a comodidade e segurança do animal, como por exemplo, uma câmera para que possa monitorar seu comportamento, um sensor de presença para

que detecte a presença do animal quando estiver comendo, para que o motor não acione mesmo sendo o horário de alimentação e a possível criação de uma página Web, para caso o usuário não queira instalar o aplicativo do Telegram, possa acessar a página web.

Pode-se implementar também um sistema de monitoramento para o histórico alimentar do animal, com dados que indicam o quanto o animal come diariamente para serem utilizados em consultas veterinárias.

Portanto, o protótipo cumpriu com os objetivos e se mostrou eficaz em relação às funções desenvolvidas, além de poder ser montado por qualquer pessoa que tenha algum conhecimento na área.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, M. N. de B. Comedouro automático para cães. 2016. Projeto de monografia – Disponível em: http://lyceumonline.usf.edu.br/salavirtual/documentos/2954.pdf. Acesso em: 4 de out. 2022.

BARRA, B., Lara Maria. Desenvolvimento de sensor ultrassônico para medição de fluxos multifásicos. 17 ago. 2021. Acesso em 4 de mai. 2023.

BOUWER, A. Automação residencial, comercial e predial: Economize com sua conta de energia! Disponível em https://clconstrutora.com/automacao-residencial-comercial-e-predial-economize-com-sua-conta-de-energia/. Acesso em 25 de mar.2023.

BITTENCOURT, S. O que é Arduino:Tudo o que você precisa saber. HostGator. 31 jan. 2017. Disponível em: <https://www.hostgator.com.br/blog/o-que-e-arduino/>. Acesso em: 12 de set. 2022.

(BOLZANI, 2004) BOLZANI, Caio A. M. Residências Inteligentes: um curso de Domótica. 1.ed. São Paulo: Editora Livraria da Física, 2004. 332 p. Acesso em 13 de set. 2022.

CARVALHO, A. S. A técnica logística no toyotismo: uma aproximação geográfica do just-in-time. GEOUSP Espaço e Tempo (Online), *[S. l.]*, v. 21, n. 1, p. 32-47, 2017. DOI: 10.11606/issn.2179-0892.geousp.2017.96023. Disponível em: https://www.revistas.usp.br/geousp/article/view/96023. Acesso em: 12 de set. 2022.

CASTRO, L. Entendendo o protocolo I2C, 2021.Disponível em:https://blog.arduinoomega.com/entendo-o-protocolo-i2c/. Acesso em 21 de abr. 2023.

CORDEIRO. M. Automação residencial via Web e App utilizando módulos Wi-Fi ESP8266 em conjunto com sensores. Curitiba, v.1, n. 1, p. 8-50, 2019. Disponível em:http://www.eletrica.ufpr.br/tcc/2019/2s/Marcos%20Vin%C3%ADcius%20de%20Sousa%20Cordeiro/Trabalho%20de%20conclus%C3%A3o%20de%20curso%20(TCC%20B)%20-%20MARCOS%20V.%20DE%20S.%20CORDEIRO.pdf. Acesso em: 13 de set. 2022.

ELIZEIRE, Mariane BrÄscher. Expansão do mercado pet e a importância do marketing na medicina veterinária. 2013. 51 f. Tese (Doutorado) - Curso de Medicina Veterinária, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013. Acesso em 04 de out. 2022.

ELETROGATE. Conhecendo a fundo o RTC, 2021.https://blog.eletrogate.com/serial-clock-serial-date-e-i%C2%B2c-conhecendo-a-fundo-o-rtc/. Acesso em 21 de fev. 2023.

ESPRESSIF SYSTEMS (Xangai). ESP32 Series Datasheet. [S. l.: s. n.], 2019. Acesso em 25 de mar. 2023.

FARHAT, N. de A.; KLINGENFUS, V. de S. Alimentador Para Pequenos Animais Controlado Por Aplicativo Móvel. 2019. Trabalho de conclusão de curso - Disponível em: http://repositorio.utfpr.edu.br/jspui/bitstream/1/8444/1/CT_DAELN_2019_1_08.pdf. Acesso em 04 de out. 2022.

FERNANDO, K. Automação com Telegram é ESP32, 2019. Disponível em: https://www.fernandok.com/2018/11/automacao-com-telegram-e-esp32.html. Acesso em 23 de abr. 2023.

GRATCHEN, E. Servo motor - o que é e como funciona. Disponível em https://www.linkedin.com/pulse/servo-motor-o-que-%C3%A9-e-como-funciona-eduardo-grachten/?trk=pulse-article_more-articles_related-content-card&originalSubdomain=pt. Acesso em 26 de mar. 2023.

Guo, F., Zhang, X. Real-time clock jump compensation for precise point positioning. *GPS Solut* **18**, 41–50 (2014). https://doi.org/10.1007/s10291-012-0307-3. Acesso em 21 de fev. 2023.

LIMA, F. S. de. A Automação e sua evolução. 2003. 3f. Trabalho – Universidade Federal do Rio Grande do Norte, Natal, 2003. Acesso em 25 de mar. 2023.

KERLY, A.; HALL, P.; BULL, S. Bringing chatbots into education:Towards natural language negotiation of open learner models.Knowledge-Based Systems, Elsevier, v. 20, n. 2, p. 177–185, 2007.<http://www.sciencedirect.com/science/article/pii/S0950705106001912aep section-id16>.Acesso em 26 de mar. 2023.

KUROSE, James. F. & ROSS, Keith W. Redes de Computadores e a internet: Uma abordagem top-down, 3ª Edição, Editora Pearson, São Paulo– SP, 2006**.** Acesso em 14 de mai. 2023.

OCHAKOWSKI, N.;Protótipo de um alimentador automático para animais de estimação. 2007. Trabalho de conclusão de curso - Disponível em: http://campeche.inf.furb.br/tccs/2007-I/2007-1nadiaochakowskivf.pdf.Acesso em 23 de out. 2022.

OLIVEIRA, A. M. Automação residencial. Monografia apresentada ao Departamento de Ciências da Administração e Tecnologia, do Centro Universitário de Araraquara, 2005. Acesso em 13 de set. 2023.

OLIVEIRA, R. C. B. de. Alimentador de Animais Domésticos Automático com Comunicação a Distância. 2017. Trabalho de monografia - Disponível em: http://monografias.poli.ufrj.br/monografias/monopoli10022726.pdf. Acesso em 04 de out. 2022.

POLATIDIS, N. Chatbot for admissions. arXiv preprintarXiv:1408.6762, 2014. Acesso em 26 de mar. 2023.

Sensor ultrassônico HC-SR04 - STA Eletrônica. Disponível em: <https://www.sta-eletronica.com.br/artigos/arduinos/sensor-ultrassonico-hc-sr04#:~:text=Os%20sensores%20ultrass%C3%B4nicos%20funcionam%20enviando>. Acesso em: 04 de mai. 2023.

SENA, G. de D.; alimentador automático para cachorros e gatos em situação de rua na UNIFERSA. 2021. Trabalho de conclusão de curso - Disponível em: https://repositorio.ufersa.edu.br/bitstream/prefix/6893/1/D%c3%b4ricoGS_ART.pdf. Acesso em 04 de out. 2022.

SHOPEE.Disponível em: <https://shopee.com.br/product/418870990/13881538234?d_id=8e450&utm_content=qVtLeVEpMzgfe13MudjMH14ev3Z>. Acesso em: 07 de mai. 2023.

SILVA, R. F. S. B. Bruno.; BRITO, P. Henrique. Telegram como Objeto de Aprendizagem para Apoiar o Ensino de Libras para Ouvintes. Disponível em: <http://www.tise.cl/Volumen14/TISE2018/283.pdf>. Acesso em 25 de mar.2023.

STRAUB, Matheus. Balança arduino com célula de carga e hx711 – tutorial calibrando everificando peso. Usinainfo. Santo Ângelo, RS, 17/09/2019. Disponível em: <https://www.usinainfo.com.br/blog/balanca-arduino-com-celula-de-peso-e-hx711-tutorial-calibrando-e-verificando-peso/>. Acesso em 23 de abr. 2023.

TECNOLOGIA, T. Servo TowerPro MG995 Metálico | Smart Kits - Componentes Eletrônicos. Disponível em: <https://www.smartkits.com.br/servo-towerpro-mg995-metalico?parceiro=9390&gclid=Cj0KCQjw0tKiBhC6ARIsAAOXutnhLZEN2lOBSICJhMjTO1EqPuykv4X1ulPZaCIcxjHMC97VKjOMP_caAtiEEALw_wcB>. Acesso em: 06/05/2023.

TECHNOLOGIES, A. O que é HTTP e como funciona? | Azion. Disponível em: <https://www.azion.com/pt-br/blog/o-que-e-http-e-como-ele-funciona/>. Acesso em: 31/05/2023.

APÊNDICE A

```
/*Programa alimentador automatizado monitorado via Telegram
 * TCC de Matheus Rodrigues Vieira Silva
 * 07 de Maio de 2023
 */
//Inclusão das bibliotecas.
```

```
#include <Adafruit_BusIO_Register.h>
#include <Adafruit_I2CDevice.h>
#include <Adafruit_I2CRegister.h>
#include <Adafruit_SPIDevice.h>
#include <RTClib.h>
#include <Wire.h>
#include <ESP32Servo.h>
#include <HX711.h>
#include <analogWrite.h>
#include <ESP32Tone.h>
#include <ESP32PWM.h>
#include <Ultrasonic.h>
#include <UniversalTelegramBot.h>
#include <WiFiClientSecure.h>
//Declaraçao do servo.
Servo servo;
//Declaracao do objeto escala na classe HX711 da biblioteca.
HX711 escala;
// Inicio da posição do servo motor.
int pos = 0;
//Código do bot.
#define BOT_TOKEN "*****"
//Dedinição do ID do usuário que utilizará o alimentador.
#define CHAT_ID "*****"
//Quantidade de usuários que podem interagir com o bot.
#define SENDER_ID_COUNT 2
//Ids dos usuários que podem interagir com o bot.
//É possível verificar seu id pelo monitor serial ao enviar uma mensagem para o
bot.
String validSenderIds[SENDER_ID_COUNT] = {"739113236"};
//Configuração dos pinos para o modulo HX711.
#define DT  23
#define SCK  19
// Configuração pinos do sensor ultrassônico.
```

```
#define TRIGGER  4
#define ECHO     5
//pinagem do servo motor.
const int servoPin = 2;
//Controle do ultrassônico.
Ultrasonic ultrasonic(TRIGGER, ECHO);
//pre-definição da variavel de calibracao.
float calibracao_celula_de_carga = -367864.24;
//parâmetros de horário que serão atualizados.
int horaAtual, minutoAtual;
//parâmetros primeira alimentação.
int horaAlimentacao1, minutoAlimentacao1, demosComida1;
//parâmetros segunda alimentação.
int horaAlimentacao2, minutoAlimentacao2, demosComida2;
//parâmetros segunda alimentação.
int horaAlimentacao3, minutoAlimentacao3, demosComida3;
//Declaração das variáveis que serão utilizadas.
float peso;
long microsec = 0 ;
float distanciaCM = 0 ;
String nivelPote;
String pesoAtual;
String opcaoDisponivel;
String alimentoJa;
float pesoPote;
//Declaração do RTC.
RTC_DS3231 rtc;
// Configuração da rede Wi-Fi
const char* WIFI_SSID = "*****";
const char* WIFI_PASSWORD = "******";
// Objeto do cliente Wi-Fi e do bot do Telegram
WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);
//Conexão com Wi-fi
```

```
void conectarWifi()
{
  Serial.print("Conectando à rede Wi-Fi ");
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Endereço IP: ");
  Serial.println(WiFi.localIP());
  client.setInsecure();
}
//Horários de alimentação configurados
 String hAlimentacao()
{
   String hora1 = String (horaAlimentacao1 );
   String minuto1 = String (minutoAlimentacao1);
   String hora2 = String (horaAlimentacao2);
   String minuto2 = String (minutoAlimentacao2);
   String hora3 = String (horaAlimentacao3);
   String minuto3= String (minutoAlimentacao3);
   String horarios = (hora1 + ":" + minuto1 + "hrs\n" + hora2 + ":" + minuto2 +
"hrs\n" + hora3 + ":" + minuto3+ "hrs" );
   return horarios;
}
String alimentarAgora()
{
   //realiza uma media entre leituras com a celula sem carga.
  float media_leitura = escala.read_average();
  //pre-definição da variavel de calibração.
  float calibracao_celula_de_carga = -367864.24;
```

```
  //ajusta a escala para o fator de calibracao
  escala.set_scale(calibracao_celula_de_carga);
     servo.attach(servoPin); // Inicializando o servo motor
     servo.write (pos);//
     pesoPote = escala.get_units();
    while (pesoPote < 0.030){
      servo.write(80);
      delay(500);
      servo.write(0);
      delay(2500);
      pesoPote = escala.get_units();
  }
    servo.write(0);
    alimentoJa="Pote abastecido";
    return alimentoJa;
}
//Funções disponíveis.
String pegarOpcao()
{
  opcaoDisponivel  =  "Os comandos disponíveis são:\n\n";
  opcaoDisponivel +=  "NÍVEL: Para ver o nivel de ração do reservatório \n";
  opcaoDisponivel +=  "PESO: Para ver o peso atual do pote \n";
  opcaoDisponivel +=  "ALIMENTAR: Para alimentar seu pet agora \n";
  opcaoDisponivel +=  "HORÁRIOS: Para acessar os horarios de alimentação do
seu pet \n";
  opcaoDisponivel +=   "CONFIGURAR HORA: Para configurar o horario de
alimentação \n";
  return opcaoDisponivel;
}
//Nível de ração do reservatório.
String pegarNivel()
{
  distanciaCM = ultrasonic.convert(microsec, Ultrasonic::CM);
  if (distanciaCM >12 )
```

```
  {
    nivelPote = "ATENÇÃO, ração prestes a acabar!";
  }else if(distanciaCM <= 12 and distanciaCM >= 2){
    nivelPote = "Reservatório com metade da capacidade! ";
  }else{
    nivelPote = "Fique tranquilo, reservatório cheio!";
  }
  return nivelPote;
}
//Peso atual do pote.
String pegarPeso()
{
  float media_leitura = escala.read_average();
  float calibracao_celula_de_carga = -367864.24;
  escala.set_scale(calibracao_celula_de_carga);
  if (escala.get_units() > 0.010 )
  {
    pesoAtual ="O peso atual do pote é de:" ;
    pesoAtual =(escala.get_units());
    delay(100);
  }else  {
    pesoAtual= "O pote do seu pet está vazio, \n0.000";
  }
   return pesoAtual;
}
//Comunicação com o Bot.
void verificarMensagem()
{
   // Verifica se há novas mensagens.
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  // Processa cada mensagem recebida.
  for (int i = 0; i < numNewMessages; i++)
  {
    String chatId = String(bot.messages[i].chat_id); //ID do chat.
```

```
String senderId = String(bot.messages[i].from_id); //ID do contato.
Serial.println("senderId: " + senderId); //Mostra no monitor serial o id de quem mandou a mensagem.
boolean validSender = validateSender(senderId); //Verifica se é o id de um remetente da lista de remetentes válidos.
if(!validSender) //Se não for um remetente válido.
{
 bot.sendMessage(chatId, "Desculpe mas você não tem permissão", "HTML"); //Envia mensagem que não possui permissão e retorna sem fazer mais nada.
 continue; //Continua para a próxima iteração do for (vai para próxima mensgem, não executa o código abaixo).
}
String text = bot.messages[i].text;
 if(text == "NÍVEL" ){
  String chat_id = String(bot.messages[i].chat_id);
  String nivel = pegarNivel();
  bot.sendMessage(chat_id, " " + nivel);
 }
 if(text == "PESO" ) {
  String chat_id = String(bot.messages[i].chat_id);
  String peso = pegarPeso();
  bot.sendMessage( chat_id, "Peso atual do pote é de: \n\n" + peso + "kg");
 }
 if(text == "START" ){
  String chat_id = String(bot.messages[i].chat_id);
  String start = pegarOpcao();
  bot.sendMessage( chat_id, "Olá, Seja Bem Vindo ao LTPet ! \n" + start);
 }
 if(text == "ALIMENTAR" ){
  String chat_id = String(bot.messages[i].chat_id);
  String alimentar = alimentarAgora();
  bot.sendMessage( chat_id, "Enchimento será feito \n\n" + alimentar);
 }
```

```
    if(text == "HORÁRIOS"){
      String chat_id = String(bot.messages[i].chat_id);
      String horario1,horario2,horario3 = hAlimentacao();
      bot.sendMessage( chat_id, "Os horarios programados para alimentação são:\n" + horario1 + "\n" +horario2+ "\n"  + horario3 );
    }
    if(text == "CONFIGURAR HORA" ){
     String chat_id = String(bot.messages[i].chat_id);
     bot.sendMessage( chat_id, "Envie a hora exatamente no padrão HX-HH:MM, o qual 'X' é o horário que quer mudar, por exemplo: H1-22:33, configura a primeira hora para 22:33");
    }
     int colonIndex = text.indexOf(':')
    if (colonIndex >= 0)
    {
      Serial.println("A string contém ':'");
   int len = text.length();
  String horaMudar = text.substring(static_cast<uint16_t>(0), static_cast<uint16_t>(2));
  String newHora = text.substring(static_cast<uint16_t>(3), static_cast<uint16_t>(5));
  String newMinuto = text.substring(static_cast<uint16_t>(6), static_cast<uint16_t>(8));
      // Imprime os valores
      Serial.print("len: ");
      Serial.println(len);
      Serial.print("horaMudar: ");
      Serial.println(horaMudar);
      Serial.print("newHora: ");
      Serial.println(newHora);
      Serial.print("newMinuto: ");
      Serial.println(newMinuto);
      if(horaMudar == "H1"){
        Serial.print("Hora atual configurada: ");
```

```
  Serial.print(horaAlimentacao1);
  Serial.print(":");
  Serial.println(minutoAlimentacao1);
  //Atribui o valor recebido na variável já existente
  int newHoraConfigurada = newHora.toInt();
  int newMinutoConfigurado = newMinuto.toInt();
  horaAlimentacao1 = newHoraConfigurada;
  minutoAlimentacao1 = newMinutoConfigurado;
  String chat_id = String(bot.messages[i].chat_id);
  bot.sendMessage(chat_id,"Novo horário configurado com sucesso");
  bot.sendMessage(chat_id,newHora + " Horas");
  bot.sendMessage(chat_id,newMinuto + " Minutos");
  Serial.print("Hora configurada via telegram: ");
  Serial.print(horaAlimentacao1);
  Serial.print(":");
  Serial.println(minutoAlimentacao1);
 }
  if(horaMudar == "H2"){
  Serial.print("Hora atual configurada: ");
  Serial.print(horaAlimentacao1);
  Serial.print(":");
  Serial.println(minutoAlimentacao1);
  //Atribui o valor recebido na varivel já existente.
  int newHoraConfigurada = newHora.toInt();
  int newMinutoConfigurado = newMinuto.toInt();

  horaAlimentacao2 = newHoraConfigurada;
  minutoAlimentacao2 = newMinutoConfigurado;
  String chat_id = String(bot.messages[i].chat_id);
  bot.sendMessage(chat_id,"Novo horário configurado com sucesso");
  bot.sendMessage(chat_id,newHora + "Horas");
  bot.sendMessage(chat_id,newMinuto + " Minutos");
  Serial.print("Hora configurada via telegram: ");
  Serial.print(horaAlimentacao2);
```

```
          Serial.print(":");
          Serial.println(minutoAlimentacao2);
         }
         if(horaMudar == "H3"){
          Serial.print("Hora atual configurada: ");
          Serial.print(horaAlimentacao3);
          Serial.print(":");
          Serial.println(minutoAlimentacao3);
           //Atribui o valor recebido na varivel já existente.
          int newHoraConfigurada = newHora.toInt();
          int newMinutoConfigurado = newMinuto.toInt();
          horaAlimentacao3 = newHoraConfigurada;
          minutoAlimentacao3 = newMinutoConfigurado;
          String chat_id = String(bot.messages[i].chat_id);
          bot.sendMessage(chat_id,"Novo horário configurado com sucesso");
          bot.sendMessage(chat_id,newHora + "Horas");
          bot.sendMessage(chat_id,newMinuto + " Minutos");
          Serial.print("Hora configurada via telegram: ");
          Serial.print(horaAlimentacao3);
          Serial.print(":");
          Serial.println(minutoAlimentacao3);
         }
      } else {
        Serial.println("A string não contém ':'");
      }
  }
}
boolean validateSender(String senderId)
{
  //Para cada ID de usuário que pode interagir com este bot
  for(int i=0; i<SENDER_ID_COUNT; i++)
  {
  //Se o ID do remetente faz parte do array retornamos que é válido
  if(senderId == validSenderIds[i])
```

```
   {
    return true;
   }
  }
  //Se chegou aqui significa que verificou todos os ids e não encontrou no array.
  return false;
}
void setup()
{
  escala.begin (DT,SCK); //Inicialização e definição dos pinos DT e SCK dentro do objeto ESCALA.
  //Inicia o monitor serial.
  Serial.begin(9600);
  Wire.begin();
  //Inicia o módulo relógio.
  rtc.begin();
  //Wifi
  conectarWifi();
  float media_leitura = escala.read_average();
  delay(3000);
  Serial.println("Media de leituras com Celula sem carga: ");
  Serial.print(media_leitura);
  Serial.println();
  escala.tare(); //Zera a escala.
  delay(1000);

  servo.attach(servoPin); // Inicializando o servo motor.
  servo.write (pos);//
  //Determina o horário da primeira alimentação.
  horaAlimentacao1 = 8;
  minutoAlimentacao1 = 00;
  //Determina o horário da segunda alimentação.
  horaAlimentacao2 = 14;
  minutoAlimentacao2 = 00;
```

```
  //Determina o horário da terceira alimentação.
  horaAlimentacao3 = 20;
  minutoAlimentacao3 = 00;
  //Determina o status de alimentação. 0 equivale a "não" e 1 a "sim".
  demosComida1 = 0;
  demosComida2 = 0;
  demosComida3 = 0;
  rtc.adjust(DateTime(__DATE__, __TIME__));
}
void loop()
{
  DateTime now = rtc.now();
  int horaAtual = now.hour();
  int minutoAtual = now.minute();
  int segundoAtual = now.second();
  //Lendo o valor do sensor.
  microsec = ultrasonic.timing();
  //Convertendo a distância em CM.
  distanciaCM = ultrasonic.convert(microsec, Ultrasonic::CM);
  escala.set_scale(calibracao_celula_de_carga); //Ajusta a escala para o fator de
calibração.
  delay(1000);
  Serial.println();
  Serial.print("Peso: ");
  Serial.print(escala.get_units(), 3); //Retorna a leitura da variável escala com a
unidade quilogramas.
  Serial.print(" kg");
  Serial.println();
  delay (100);
  verificarMensagem();
  delay(1000);
  //verifica se é o horário da primeira alimentação
  if ((horaAtual == horaAlimentacao1 and minutoAtual == minutoAlimentacao1
and demosComida1 == 0) and (escala.get_units() < 0.030 ))
```

```
  {
    while (peso < 0.030){
    servo.write(60);
    delay(500);
    servo.write(0);
    delay(2500);
    peso = escala.get_units();
    Serial.println((peso), 3);
    delay(1000);
   }
    servo.write(0);
    demosComida1 = 1; //Altera status da comida.
  }
  if ((horaAtual == horaAlimentacao2 and minutoAtual == minutoAlimentacao2
and demosComida2 == 0) and (escala.get_units() < 0.030 ))
  {
    while (peso < 0.030){
    servo.write(60);
    delay(500);
    servo.write(0);
    delay(2500);
    peso = escala.get_units();
    Serial.println((peso), 3);
    delay(1000);
     }
    servo.write(0);
    demosComida2 = 1; //Altera status da comida.
  }
  //Verifica se é o horário da primeira alimentação.
  if ((horaAtual == horaAlimentacao3 and minutoAtual == minutoAlimentacao3
and demosComida3 == 0) and (escala.get_units() < 0.030 ))
  {
    while (peso < 0.030){
    servo.write(60);
```

```
    delay(500);
    servo.write(0);
    delay(2500);
    peso = escala.get_units();
    Serial.println((peso), 3);
    delay(1000);
    }
    servo.write(0);
    demosComida3 = 1; //altera status da comida1
  }
  if (demosComida1 == 0 or demosComida1 == 1 and demosComida2 == 1 and
demosComida3 == 1 )
  {
    DateTime now = rtc.now();
    int hour = now.hour();
    int minute = now.minute();
    int second = now.second();
    Serial.print("Horário atual: ");
    Serial.print(hour);
    Serial.print(":");
    Serial.print(minute);
    Serial.print(":");
    Serial.print(second);
    Serial.println();
    Serial.print("Próxima alimentação: ");
    Serial.print(horaAlimentacao1);
    Serial.print("h:");
    Serial.print(minutoAlimentacao1);
    Serial.println("min");
    Serial.println("  ");
  }
  if (demosComida2 == 0 or demosComida1 == 1 and demosComida2 == 1 and
demosComida3 == 1)
  {
```

```
    DateTime now = rtc.now();
    int hour = now.hour();
    int minute = now.minute();
    int second = now.second();
    Serial.print("Horário atual: ");
    Serial.print(hour);
    Serial.print(":");
    Serial.print(minute);
    Serial.print(":");
    Serial.print(second);
    Serial.println();
    Serial.print("Próxima alimentação: ");
    Serial.print(horaAlimentacao2);
    Serial.print("h:");
    Serial.print(minutoAlimentacao2);
    Serial.println("min");
    Serial.println("  ");
  }
  if (demosComida3 == 0 or demosComida1 == 1 and demosComida2 == 1 and
demosComida3 == 1 )
  {
    DateTime now = rtc.now();
    int hour = now.hour();
    int minute = now.minute();
    int second = now.second();
    Serial.print("Horário atual: ");
    Serial.print(hour);
    Serial.print(":");
    Serial.print(minute);
    Serial.print(":");
    Serial.print(second);
    Serial.println();
    Serial.print("Próxima alimentação: ");
    Serial.print(horaAlimentacao3);
```

```
    Serial.print("h:");
    Serial.print(minutoAlimentacao3);
    Serial.println("min");
    Serial.println("  ")
  }
  //Meia noite reseta o status de comida do dia.
  if (horaAtual == 0 && minutoAtual == 0)
  {
    demosComida1 = 0;
    demosComida2 = 0;
    demosComida3 = 0;
  }
  //Atualiza monitor serial.
  delay (1000);
}
```