UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

KELWYN DA SILVA OLIVEIRA

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE REMOTO PARA ALIMENTADOR AUTOMÁTICO DE ANIMAIS DOMÉSTICOS

Manaus
2023

KELWYN DA SILVA OLIVEIRA

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE REMOTO PARA ALIMENTADOR AUTOMÁTICO DE ANIMAIS DOMÉSTICOS

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Controle e Automação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Controle e Automação.

Orientador: Prof. Msc. Cleto Cavalcante De Souza Leal

Manaus

2023

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).
Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.

O48dd Oliveira, Kelwyn da Silva
Desenvolvimento de um sistema de controle remoto para alimentador automático de animais domésticos / Kelwyn da Silva Oliveira. Manaus : [s.n], 2023.
91 f.: color.; 30 cm.

TCC - Graduação em Engenharia de Controle e Automação; - Universidade do Estado do Amazonas, Manaus, 2023.
Inclui bibliografia
Orientador: Cleto Cavalcante De Souza Leal

1. Automação. 2. Alimentador automático,. 3. nternet das Coisas. 4. Protocolo MQTT. 5. ESP32. I. Cleto Cavalcante De Souza Leal (Orient.). II. Universidade do Estado do Amazonas. III. Desenvolvimento de um sistema de controle remoto para alimentador automático de animais domésticos

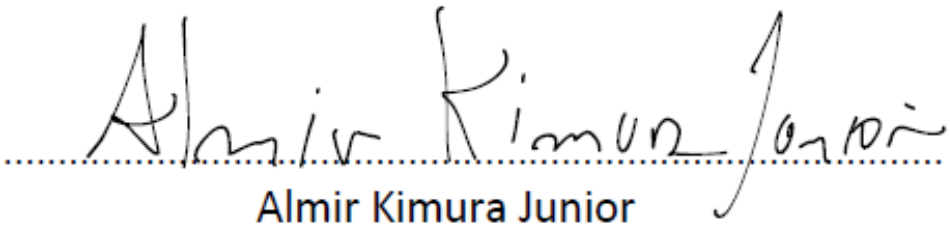
Elaborado por Jeane Macelino Galves - CRB-11/463

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE REMOTO PARA ALIMENTADOR AUTOMÁTICO DE ANIMAIS DOMÉSTICOS

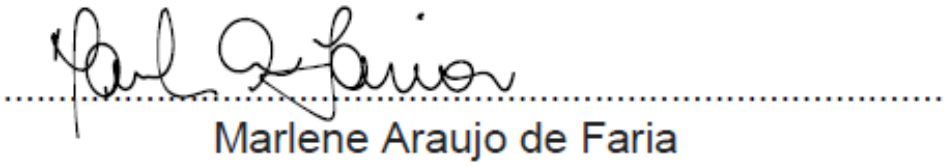
KELWYN DA SILVA OLIVEIRA

Trabalho de Conclusão de Curso (TCC) apresentado à Escola Superior de Tecnologia da Universidade do Estado do Amazonas como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia de Controle e Automação.

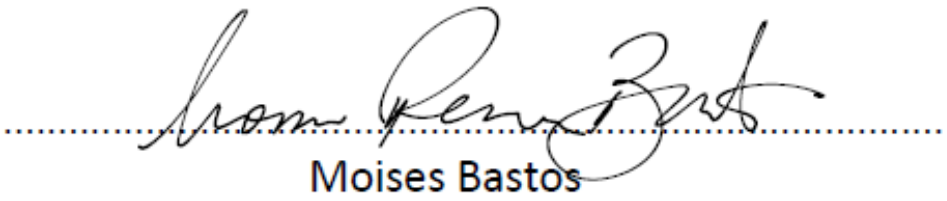
Aprovado por:



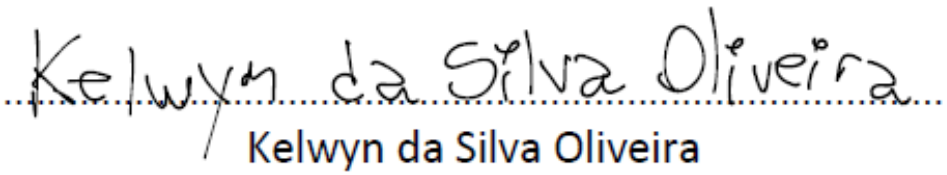
Almir Kimura Junior



Marlene Araujo de Faria



Moises Bastos



Kelwyn da Silva Oliveira

Manaus-AM, 24 de março de 2023

RESUMO

A utilização de dispositivos automáticos está cada vez mais presente nas casas das pessoas, desde lâmpadas que acionam automaticamente como portões de garagem que abrem e fecham sem a interferência humana. Levando-se em consideração que a maioria das famílias possui algum animal de estimação, tomar conta deles durante os períodos que o tutor se encontra ocupado é a parte mais difícil de se criar um *pet*. Sendo assim, este trabalho propõe o desenvolvimento de um sistema de controle remoto para um alimentador automático de animais domésticos capaz de despejar o alimento em um horário especificado pelo tutor bem como a quantidade desejada. O modelo CAD foi desenvolvido no *software Inventor* e a partir das dimensões desejadas foi feito o dimensionamento do motor que impulsionaria o alimento armazenado no silo do alimentador para o comedouro do animal. O alimentador será monitorado através da internet da coisas, para isso foi utilizado como central de controle do alimentador o microcontrolador ESP32 que permite a conexão com a internet e a comunicação foi baseada no protocolo MQTT. O dispositivo consiste em um sensor ultrassônico para medir a quantidade de ração disponível no alimentador, um módulo RTC para contagem de horas e uma célula de carga para o controle de *feedback*. Através do aplicativo no celular do tutor ele pode verificar o horário no alimentador, o nível de ração no reservatório e a quantidade de alimento no comedouro do animal, bem como programar os horários e a quantidade de ração que será despejada para o animal.

Palavras-chave: Automação, Alimentador automático, Internet das Coisas, Protocolo MQTT, ESP32.

ABSTRACT

The use of automatic machines are increasing in people's homes, from lamps that activate automatically to garage doors that open and close without human interference. Considering that most families have a pet, taking care of them during periods when the owner is busy is the main problem in mantaining a *pet*. Therefore, this work proposes the development of a remote control system for an automatic feeder of domestic animals capable of feeding the animal at a time specified by the tutor as well as the desired amount. The CAD model was designed in the *software Inventor* and from the desired dimensions, the dimensioning of the motor that would drive the food stored in the silo to the animal's feeder was made. The feeder will be monitored through Internet of Things, for this, the ESP32 microcontroller was used as the feeder's control center, which allows the connection to the internet and the communication was based on the MQTT protocol. The device consists of an ultrasonic sensor to measure the amount of food available in the feeder, an RTC module for counting hours and a load cell for *feedback* control. Through the application on the tutor's cell phone, he can check the time at the feeder, the level of feed in the silo and the amount of food in the animal's feeder, as well as schedule the times and the amount of food that will be feeded to the animal.

Keywords: Automation, Automatic pet feeder, Internet of Thing, MQTT protocol, ESP32.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BNDES	Banco Nacional de Desenvolvimento Econômico e Social
CAD	Computer Aided Design
CPU	Central Processing Unit
I2C	Inter-Integrated Circuit
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	Integrated Development Environment
ITU	International Telecommunications Union
IoT	Internet of Things
M2M	Machine-to-Machine
MQTT	Message Queuing Telemetry Transport
OASIS	Organization for the Advancement of Structured Information Standads
ONU	Organização das Nações Unidas
RTC	Real Time Clock
SCL	Serial Clock
SDA	Serial Data
TCP/IP	Transmission Control Protocol/Internet Protocol
TIC	Tecnologia da Informação e Comunicação

SUMÁRIO

1 INTRODUÇÃO

Segundo dados do Instituto Brasileiro de Geografia e Estatística (IBGE), em 2020 cerca de 44,3% das residências brasileiras possuem ao menos um cão de estimação (DESTAQUE NOTÍCIAS, 2020). Mas se antes eles eram utilizados para caça ou segurança hoje são tratados como membro da família, fazendo com que os animais de estimação sejam "antropomorfizados", onde são tratados como crianças, servindo como companhia e terapia (PASTORI; MATOS, 2015).

Contudo, devido a certas circunstâncias é provável que o tutor tenha que passar horas e horas longe de casa, e isso de certa forma pode causar preocupação, visto que o animal pode ficar sem comida por um tempo, dessa forma abrindo espaço para soluções inovadoras.

Atualmente as famílias têm investido em sistemas automatizados como lâmpadas de acionamento automático, sensores de detecção de movimento, portões eletrônicos, entre outros. Nesse contexto o mercado de *pets* também tem desenvolvido soluções para facilitar o dia a dia de tutores de animais de estimação.

Com a finalidade de otimizar o pouco tempo que as pessoas possuem atualmente e evitar que o animal fique sem se alimentar mesmo na ausência de seu tutor, com a automatização da alimentação é possível solucionar tal problema.

Além disso, com a popularização dos *smartphones*, é possível desenvolver um alimentador automático que através de um aplicativo o tutor pode programar os horários, o peso a ser despejado e acompanhar a quantidade de ração ainda disponível no reservatório de forma simples e remotamente.

1.1 PROBLEMÁTICA

É possível dar mais segurança aos tutores que queiram manter seus cães bem alimentados em sua casa enquanto ele desempenha suas atividades profissionais longe?

1.2 HIPÓTESE

Por meio de um sistema automatizado, é possível desenvolver um sistema de controle remoto de alimentador para cães domésticos em que o tutor seja capaz de programar os horários e a quantidade de alimento oferecido ao animal, além de poder verificar a quantidade de ração ainda disponível no reservatório.

1.3 OBJETIVOS

1.3.1 Objetivo geral

Desenvolver um sistema de controle remoto para um alimentador automático de cães domésticos utilizando microcontrolador, atuadores e sensores para programar e gerenciar os horários de alimentação do animal por meio de um *smartphone*.

1.3.2 Objetivos específicos

1. Realizar o levantamento bibliográfico de alimentadores automáticos já no mercado e dos temas necessários para desenvolvimento do trabalho.
2. Projetar o alimentador em um *software* de modelagem de estrutura mecânica;
3. Dimensionar o motor automático.
4. Projetar o sistema embarcado para leitura dos sensores e transmissão de dados;
5. Desenvolver o aplicativo que controlará o despejo das refeições;
6. Conectar o protótipo à rede WiFi para que os dados possam ser vistos no aplicativo pelo usuário.

1.4 DIVISÃO DO TRABALHO

O trabalho está organizado da seguinte maneira: após a presente introdução, o Capítulo 2 aborda o Referencial Teórico, o qual aborda todos os assuntos necessários para a implementação do projeto. O Capítulo 3 consiste na implementação do projeto, indicando os materiais e métodos utilizados na elaboração do mesmo. O Capítulo 4 apresenta os resultados e discussões obtidos no decorrer do projeto. Por fim, no Capítulo 5, apresentam-se as considerações finais e sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo, será abordada a teoria necessária para compreensão do conceito e procedimento do projeto. Inicialmente uma apresentação sobre a nutrição de cães e gatos e os modelos de alimentadores automáticos já no mercado. Então apresentam-se os dispositivos eletrônicos e ambientes de programação a serem utilizados para o desenvolvimento do projeto.

2.1 Nutrição de animais domésticos

Os alimentos, atualmente, buscam além de nutrir, a promoção da saúde, bem estar e longevidade. Contudo muitos tutores tratam cães e gatos como membros da mesma categoria chegando a alimentá-los da mesma forma, ao considerarmos que atualmente os alimentos estão cada vez mais sofisticados isso facilita o desenvolvimento de doenças no animal devido a sua má alimentação (OGOSHI et al., 2015).

No decorrer da evolução humana os cães e gatos domesticados passaram a ter seus alimentos controlados pelos seres humanos, mesmo assim esses animais domésticos evoluíram suas dietas de forma diferente. Os cães e os gatos são membros da ordem Carnívora, contudo os cães apresentam uma dieta mais onívora se comparado aos gatos, o que significa dizer que os gatos possuem uma dieta mais especializada. Assim, os alimentos disponíveis no mercado chegaram a se especializar nas exigências nutricionais e hábitos alimentares de cada espécie (OGOSHI et al., 2015).

Para que os animais domésticos se mantenham sudáveis é essencial entendermos a quantidade e a frequência de alimento a ser ingerida. Para os cães é recomendado fornecer uma quantidade dividida em duas porções diárias. Já os gatos, por serem de pequeno porte, fazem várias refeições ao dia voluntariamente em pequenas quantidades, o ideal é não restringir horários, pois isso pode causar uma diminuição de consumo, portanto pode-se deixar a sua porção diária disponível por 24 horas (OGOSHI et al.,

2015).

A alimentação também varia de acordo com o período de vida do animal. Com base na fabricante Max Total Alimentos (2022) a recomendação diária de sua ração da linha *Max Professional Line Performance Adultos* para cães adultos é apresentada na Tabela 1.

Tabela 1: Peso do cão e quantidade diária

Peso do Cão	**Quantidade Diária**
De 3,5 a 7 kg	De 67 a 112 g
De 7 a 13,5kg	De 112 a 184 g
De 13,5 a 22 kg	De 184 a 265 g
De 22 a 38 kg	De 265 a 399 g
De 38 a 48,5kg	De 399 a 479 g

Fonte: MAX PET FOOD (2022)

2.2 Alimentadores no mercado

Em alguns casos o tutor pode passar várias horas longe de casa não podendo alimentar seu cão em horários regulares. Visando solucionar este problema existem no mercado vários modelos de alimentadores automáticos em comercialização, alguns exemplos são:

a) Trixie Tx9: programa até seis refeições e libera a ração por porções, de 1 até 8 porções, cada porção equivale a 12 ml. Possui a função gravadora de voz. A programação é feita por um *display* digital e a alimentação é feita via conexão à rede elétrica (TRIXIE, [s.d.]).

Figura 1: Trixie Tx9 modelo TP9202

Fonte: TRIXIE ([s.d.])

b) Tander Pet TP9202: é possível programar até 3 refeições diárias e especificar a quantidade de porções para cada horário configurado sendo de 1 até 12 porções. Também tem a função de gravação de voz, sua configuração é feita no próprio alimentador através de botões e um *display* digital e pode apresentar o horário em formato de 12h ou 24h. A sua alimentação é unicamente através de pilhas (TANDER PET, [s.d.]).

Figura 2: Tander Pet modelo TP9202

Fonte: TANDER PET ([s.d.])

c) ARF PETS APAFNEW2: é possível programar até quatro refeições diárias de 1 até 10 porções por refeição. Possui a função gravadora de voz e programação dos horários no próprio alimentador. A sua alimentação é feita principalmente através da rede elétrica, mas conta com uma bateria reserva de pilhas para continuar funcionando mesmo com a queda de energia na rede elétrica (ARF PETS, 2018).

Figura 3: ARF PETS modelo APAFNEW2

Fonte: ARF PETS (2018)

2.3 Bem-estar animal e os alimentadores automáticos

Para alguns tutores, deixar seu animal de estimação sozinho em casa é um evento diário. E ainda existem os tutores que viajam e deixam seu animal sozinho por mais de um dia, dentre estes os tutores de gatos são os mais propensos a deixarem seu animal sozinho.

Porém, os gatos não correm grandes riscos de desenvolverem obesidade ou outro distúrbio alimentar pela ausência de seu tutor durante o dia. Isso porque, antes de sair de casa, o tutor pode deixar disponível para o seu gato a quantidade de ração adequada para todo o dia que o gato porcionará suas refeições naturalmente, sem qualquer necessidade de intervenção. Mas isso não quer dizer que eles não sintam falta do seu tutor. Segundo um estudo de 2019 sobre o vínculo entre os gatos domésticos e humanos mostrou que os gatos se sentem mais confiantes e calmos quando seu tutor está perto e mais estressado quando o tutor sai (VITALE; BEHNKE; UDEL, 2019).

Se os gatos que são tidos como independentes sentem falta do tutor, os cães chegam a se apresentar ainda mais dependente dos seres humanos. Ao contrário dos gatos, os cães costumam comer todo o alimento disponível para eles em apenas uma refeição. Isso inviabiliza ao tutor disponibilizar toda a quantidade de ração para o dia ao sair de casa, sendo necessário o porcionamento intervencionado em duas a três vezes ao longo do dia.

Apesar do tutor saber que seu animal depende dele às vezes esse encontrará situações em que se vê obrigado a sair de casa para realizar suas atividades deixando o animal sozinho em casa. Portanto, especialmente para os cães, o alimentador automático se torna uma ferramenta auxiliar de cuidado com o seu animal, visto que a obesidade canina pode afetar o bem-estar do animal de diversas maneiras e impedir que ele tenha energia e condicionamento físico para realizar as atividades cotidianas de um cão saudável, como correr, pular e brincar com o seu tutor.

É necessário enfatizarmos o papel do alimentador como uma ferramenta auxiliar, e não de substituição da interação homem-animal; visto que o momento de alimentar o seu *pet* pode e deve ser utilizado para estreitar os laços entre o animal e seu tutor. Além disso, o olhar atento do tutor ao seu *pet* também não deve ser negligenciado ou substituído, pois esse cuidado permite ao tutor entender melhor os hábitos do seu animal e adaptar a sua rotina pessoal às necessidades do mesmo (WELLS, 2019).

Por esse motivo, buscou-se desenvolver um aplicativo que permitisse o ajuste livre e remoto dos horários e quantidade em gramas de porções a serem dispensadas para o *pet*. Assim, o tutor conta com mais flexibilidade para se adequar às necessidades do seu animal e zelar pela saúde do mesmo, ainda que deva se ausentar de casa por um período mais longo.

2.4 Automação

2.4.1 Automatos

Por muito tempo a mecânica tentou imitar os fenômenos naturais a fim de compreender como eles funcionavam. Com antecedentes que remontam a Grécia helênica, a construção de autômatos se tornou então um dos modos mais notórios dessa imitação. Os autômatos são comumente conhecidos por serem aparelhos com aparência de seres animados – humanos e animais – criados para reproduzirem movimentos por meios

mecânicos ou eletrônicos, em seu princípio é um engenho composto de mecanismos que permitem a execução de determinados movimentos.

No século XVIII a construção de autômatos foi tratada com mais seriedade, mas eles não eram classificados nem como arte, nem como mecânica utilitária. Dentro das máquinas era inserido um motor oculto onde a força motriz comum eram os pesos e as molas, mas também existiam autômatos movidos pela força do vento ou da água.

Algumas definições reconhecem como autômatos qualquer objeto que se mova por motor oculto, como os relógios, outros apenas reconhecem como autômato os engenhos com aparência antropomorfa. Mas seja qual for a definição, os autômatos foram construídos por muito tempo e, muito da tecnologia atual deriva dos desenvolvimentos mecânicos necessários para se desenvolver a automação (CASTRO, 2014).

Figura 4: O autômato de Maillardet - automato de escrever e desenhar

Fonte: MDIG (2022)

2.4.2 Automação

Desde a pré-história o homem busca facilitar seu trabalho, muitas vezes desenvolvendo mecanismos que o auxiliem principalmente em suas atividades manuais.

Enquanto evoluía foram criadas máquinas que substituíram completamente o esforço físico empregado em certas atividades, contudo os comandos que faziam a máquina operar ainda permaneciam a cargo do ser humano.

A automação surge primeiro na indústria em meados do século XVII, onde temos o exemplo da inserção de máquinas a vapor que ajudavam na produção de tecidos, o que tornou a industria têxtil mais eficiente e rentável. No início do século XX surgem os primeiros sistemas inteiramente automatizados. Isso quer dizer que para funcionar as máquinas não precisavam do constante controle e supervisão de um operador pois ela já era capaz de coletar dados, analisar, decidir e agir independentemente. A automação continuou a evoluir e se desenvolver ao longo do século XX, com a introdução de novas tecnologias como computadores, servomecanismos, controladores programáveis e Internet das coisas (IoT) (RIBEIRO, 2003).

Alguns dos benefícios da automação incluem o aumento da eficiência, melhoria da qualidade e redução de custos. A automação também pode ajudar a eliminar erros humanos e aumentar a segurança no local de trabalho. Mas a automação também pode apresentar algumas desvantagens como, o custo elevado de implementação e manutenção, e a necessidade de treinamento para operadores e engenheiros. Além disso, a automação pode levar ao desemprego, já que as máquinas podem realizar tarefas que antes eram realizadas por trabalhadores humanos.

Hoje, a automação está presente em praticamente todas as indústrias e tem um papel importante na melhoria da eficiência, qualidade e segurança. Mas a automação não se resume apenas ao âmbito industrial, ela também desempenha papel fundamental no avanço da ciência e da engenharia, hoje presente no espaço, na aviação e em casas. No geral, a automação é uma tecnologia poderosa que tem muitos benefícios para as empresas e os indivíduos (LIMA, 2003).

2.5 Desenho assistido por computador

Do inglês *Computer Aided Design* (CAD), é o termo utilizado para se referir a *softwares* que auxiliam/assistem na elaboração do desenho em duas dimensões (2D) e/ou três dimensões (3D) de projetos de engenharia. Em seus primórdios os desenhos arquitetônicos utilizavam o papel e o lápis como principais ferramentas para representar as intenções construtivas. Com o passar do tempo outras ferramentas surgiram para

auxiliar no processo da elaboração dos desenhos, contudo ainda havia a necessidade de espaços grandiosos, equipes numerosas e um longo tempo de dedicação (FARIAS, 2021).

Figura 5: Desenho técnico sem auxílio de computador

Fonte: NEATU (2018)

No século XIX o químico francês Alphonse Louis Poitevin desenvolveu a solução para a criação das cópias dos projeto chamada de *Blueprints*, com essa tecnologia foi possível a reprodução de cópias fiéis dos projetos construtivos. Mesmo com essa solução ainda havia um grande desperdício de tempo dentro do processo de projeto.

Com o desenvolvimento da tecnologia computacional foram criados *softwares* capazes de auxiliar no desenho de projetos. Inicialmente esses sistemas eram básicos e limitados, mas permitiam que os projetistas desenhassem e editassem desenhos em um monitor de computador em vez de em papel. Ao longo dos anos esses sistemas foram se tornando cada vez mais sofisticados permitindo a criação de projetos mais detalhados e precisos. Hoje esses sistemas permitem não só o desenvolvimento de desenhos em 2D como também em 3D possibilitando a visualização do projeto em vários diferentes ângulos, alguns desses sistemas chegando até a incluir recursos de análise, como simulações de fluxo de ar e análise estrutural (FARIAS, 2021).

2.5.1 Inventor®

O CAD 3D Inventor® é um *software* de projeto mecânico de alta precisão desenvolvido pela Autodesk®. Com ele é possível criar o projeto mecânico, documentação e simulação do produto. O Inventor® permite aos projetistas criar modelos 3D precisos, realistas e interativos de seus projetos, e fornece ferramentas para simular e analisar o comportamento mecânico dos componentes e conjuntos.

O Inventor® permite definir como parâmetros as dimensões e as propriedades dos componentes, dessa forma é possível criar vários modelos do projeto apenas alterando os parâmetros. Sendo assim possível experimentar diferentes opções de projeto rapidamente e facilmente, sem precisar recriá-lo.

Além disso, o Inventor® possui ferramentas para simular o comportamento mecânico de componentes e conjuntos, como análise de esforços, análise de vibrações e simulação de fluxo. Isso permite verificar se o projeto atende aos requisitos de desempenho antes da construção, o que pode ajudar a evitar problemas e erros durante a fase de construção.

O Inventor® também possui ferramentas para documentação de projetos, incluindo desenhos 2D e listas de peças. Sendo assim possível a criação de uma documentação precisa e detalhada dos projetos, que é essencial para a construção e manutenção de produtos mecânicos (AUTODESK, 2022).

Figura 6: Máquina de preenchimento de tambor projetada no Inventor®

Fonte: AUTODESK (2022)

2.6 Internet das coisas

A definição para Internet das Coisas (IoT, na sigla em inglês), segundo a *International Telecommunications Union* (ITU), do Setor de Normatização das Telecomunicações (ITU-T) das Organização das Nações Unidas (ONU), é:

> Do ponto de vista da padronização técnica, a IoT pode ser vista como uma infraestrutura global para a sociedade da informação, possibilitando serviços avançados através da interconexão (física e virtual) de coisas baseadas em tecnologias de informação e comunicação (TIC) interoperáveis existentes e em evolução.

De forma geral a Internet das Coisas é uma rede de objetos físicos, animais ou pessoas equipados com tecnologias de comunicação, como sensores, atuadores e dispositivos de computação, que permite coletar e compartilhar dados através da internet. Esses objetos podem ser qualquer coisa, desde dispositivos domésticos inteligentes, como termostatos e lâmpadas, até veículos, equipamentos industriais e até mesmo seres humanos com dispositivos de rastreamento (BNDES, 2017).

Nesse sentido, essa tecnologia traz uma grande vantagem para os tutores que não possam estar em casa no momento de alimentar os seus *pets*, visto que podem acionar o alimentador automático de qualquer lugar do planeta, desde que tenham acesso à *internet*.

2.6.1 Protocolo de comunicação MQTT

MQTT é o acrônimo de *Message Queuing Telemetry Transport* que significa Transporte de dados de Telemetria por Fila de Mensagem. Ele é um protocolo de comunicação do tipo *M2M* (Máquina para Máquina) criado por Andy Stanford-Clark e Arlen Nipper em 1999, com a intenção de conectar sistemas de telemetria de oleodutos via satélite. Originalmente a propriedade do protocolo pertencia à IBM, mas em 2010 teve seu uso gratuito liberado para o público e em 2014 se tornou um padrão OASIS (*Organization for the Advancement of Structured Information Standads*).

O protocolo MQTT está cada vez mais se tornando um padrão para *IoT* por conta de suas características. Segundo a Standard OASIS (2019), o MQTT é um protocolo de transporte de mensagem do tipo cliente/servidor utilizando o conceito de publicação e

subscrição. Foi desenvolvido para ser leve, simples aberto e de fácil implementação. Dessa forma ele é ideal para uso em cenários com recursos limitados como os casos de comunicação de máquina para máquina e internet das coisas onde é requerido pequenos pacotes de dados e/ou largura de banda de rede mínimo.

Este protocolo é executado sobre TCP/IP, ou sobre qualquer outro protocolo de rede que forneça conexão de forma ordenada, bidirecional e sem perdas. Suas características são:

- Uso do padrão de mensagem no tipo publicação/assinatura que fornece mensagem de um para nenhum, um para um ou de um para muitos.
- Seu foco está mo transporte da mensagem e não no conteúdo do *payload*.
- Ele possui três níveis de serviço que garantem a entrega da mensagem:
  - "No máximo uma vez- neste nível pode ocorrer a perda da mensagem.
  - "Pelo menos uma vez- é garantido que a mensagem chegue pelo menos uma vez.
  - "Exatamente uma vez- garante a entrega da mensagem exatamente uma vez.
- Possui um controle de desconexão com notificação.

Os principais conceitos do MQTT são:

- *publisher*, cliente publicador responsável por enviar os pacotes de dados.
- *subscriber*, cliente assinante que recebe os pacotes de dados.
- *broker*, agente que recebe e armazena os dados do *publisher* e envia para os *subscribers*.
- *message*, é o pacote de dados trafegado entre o cliente e o *broker*.
- *topic*, o tópico é o repositório de dados único que recebe a mensagem e é acessível ao cliente, de forma instantânea e não possui histórico.
- *unsubscribe* é a ação de solicitar o não recebimento de mensagens de um determinado tópico.
- *payload*, é o conteúdo.

- *QoS*, representa a qualidade/nível do serviço sob o qual o broker deve receber e/ou transmitir os dados.
- *Last Will*, é a mensagem que deve ser enviada pelo broker caso o cliente perca a conexão.
- *ping*, um teste periódico de conexão entre o cliente e o *broker*.
- *session*, período no qual o cliente está conectado ao *broker*.
- *clean*, indica se a sessão deve persistir, receber mensagem enquanto o cliente está desconectado, ou descartar os dados da sessão anterior quando uma nova é iniciada.
- *retain*, indica se a mensagem deve ser mantida pelo *broker* até uma próxima ser recebida.

Sendo que os cliente podem ser simultaneamente publicadores como assinantes.

2.7 Estratégias de controle

Para que o sistema produza uma resposta desejada utiliza-se estratégias de controle. O sistema de controle é a relação de componentes ou processo que podem ser representados por blocos, onde cada bloco, através das interconexões, mantém uma relação de causa e efeito gerados pelo processamento de um sinal de entrada em um sinal de saída (DORF; BISHOP, 2013).

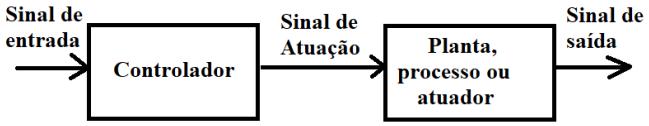
Controlar um sistema significa medir o valor da variável controlada e aplicar o sinal de controle que irá corrigir ou limitar os desvios medidos com base no valor desejado. Dessa forma existem então duas variáveis: a variável controlada que é o sinal medido sendo normalmente o sinal de saída, e a variável manipulada que é o sinal de controle (OGATA, 2010).

2.7.1 Sistema de controle de malha aberta

Nos sistemas de controle de malha aberta o sinal de saída não é medido nem realimentado para comparar com o sinal de entrada, como visto na Figura 7, ou seja, o sinal de saída não exerce qualquer ação de controle no sistema (OGATA, 2010). Este

tipo de sistema não utiliza sensores, ele faz uso de cálculos baseados no modelo físico para obter um sistema que preveja o quanto de energia um atuador necessitará para manter o estado desejado.

Figura 7: Controle de malha aberta

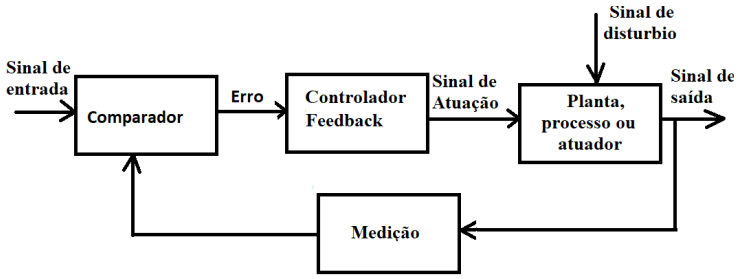


Fonte: DORF; BISHOP (2013, p.67)

2.7.2 Sistema de controle de malha fechada

Os sistemas de controle de malha fechada comparam os sinais de entrada e saída para verificar o erro entre eles, dessa forma o controlador envia o sinal de atuação para que o erro seja reduzido e o sistema chegue à saída desejada (OGATA, 2010). Nesse tipo de sistema existe uma monitoração contínua dos sensores para assim ajustar os atuadores. É o sistema de controle mais utilizado em robótica, visto que o sistema de controle de malha aberta é geralmente utilizado quando o ambiente é praticamente estático e previsível.

Figura 8: Controle de malha fechada



Fonte: DORF; BISHOP (2013, p.5)

2.8 Programação em blocos

Em termos gerais programar é criar um código ou sequência de comandos para que um computador execute alguma ou algumas tarefas. Das informações utilizadas em

um programa podem estar presentes nomes, números, sequências lógicas, operações matemáticas dentre outras. A estrutura básica de um programa consiste na parte inicial com a presença dos dados necessários ao programa, o meio com os processos e o fim que apresenta os resultados esperados.

A programação em blocos é uma forma de programação de computadores que utiliza blocos de construção lógicos, em vez de escrever códigos usando uma linguagem de programação tradicional como o C++ ou Java. Os blocos de construção são geralmente figuras e representam diferentes comandos ou ações que podem ser arrastados e soltos para criar um programa.

Quando comparado com as linhas de códigos escritas em uma linguagem de programação, a programação em blocos se mostra uma ferramenta com metodologia mais amigável para os iniciantes em programação. A programação em blocos foi criada com o objetivo de ensinar os conceitos básicos de lógica em geral, as cores e formatos específicos auxiliam no entendimento do funcionamento do código. Ao se combinar os blocos é possível obter uma estrutura lógica com início, processo e resultado.

Além da área educacional a programação em blocos também é usada em áreas, como a robótica e a automação industrial. Por exemplo, alguns robôs e drones podem ser programados usando blocos de construção para controlar seus movimentos e sensores. Além disso, algumas ferramentas de automação industrial, como programas de automação de fábrica, também usam blocos de construção para criar programas de controle (I DO CODE, 2022).

Para o desenvolvimento do aplicativo será utilizada a plataforma *MIT App Inventor*, ela é uma plataforma online de desenvolvimento de aplicativos para dispositivos móveis que utilizam tanto o sistema operacional Android como iOS, essa plataforma permite com que pessoas sem conhecimento prévio de programação criem seus próprios aplicativos.

É uma ferramenta que utiliza a codificação baseada em blocos, onde os usuários arrastam e soltam blocos de construção para criar seus aplicativos. Os blocos representam diferentes comandos e ações, como clicar em um botão, ler dados de um sensor ou enviar uma mensagem de texto. O App Inventor também fornece uma interface visual para desenhar a interface do usuário do aplicativo. Com essa metodologia é mais fácil a visualização da lógica e do aplicativo (MIT APP INVENTOR, 2022).

Essa plataforma é mantida como um projeto de código aberto e é amplamente

utilizado em escolas e universidades para ensinar programação e desenvolvimento de aplicativos. Sendo utilizada tanto por indivíduos como pequenas empresas que desejam criar seus próprios aplicativos.

Figura 9: MIT App Inventor

Fonte: ALBRITTON (2018)

2.9 Microcontrolador

O microcontrolador é um dispositivo eletrônico que consiste em um circuito integrado com um núcleo processador, memória e periféricos de entrada e saída de dados. A vantagem dos microcontroladores está em que tendo todos os periféricos em um mesmo componente é mais fácil utilizar e também mais barato pois sistemas microcontrolados não precisam de muitos componentes (KERSCHBAUMER, 2018).

2.9.1 Arduino

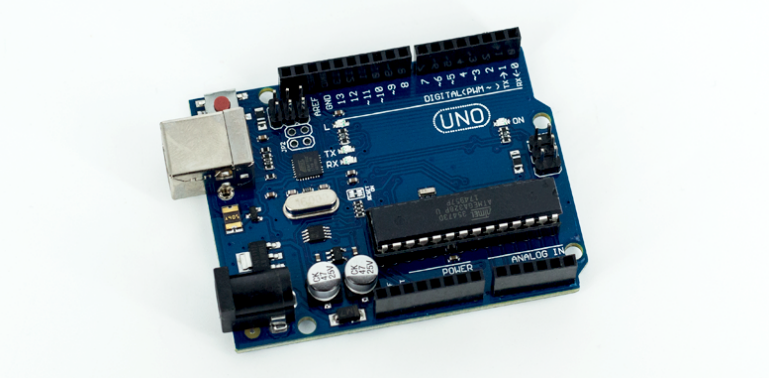
A criação do Arduino se deu em 2005 pelos pesquisadores Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis que se juntaram a fim de desenvolverem um dispositivo barato e de fácil utilização para que fosse acessível a estudantes e projetistas amadores (FILIPEFLOP, 2015).

Com o Arduino é possível interagir com o ambiente por meio de seu *hardware* e *software*, ele é como um pequeno computador que pode ser programado através de uma IDE, geralmente em linguagem C++, para realizar determinadas tarefas através da conexão de seus pinos de entrada e saída a dispositivos externos (MCROBERTS, 2011).

Seu principal componente é um microprocessador programável que processa os dados de entrada e saída dos componentes conectados a ele, além do Arduino poder ser utilizado de forma independente ele também pode estar conectado à um computador ou mesmo à Internet (MCROBERTS, 2011).

A plataforma Arduino se trata de um conceito de *hardware* e *software* abertos, isso quer dizer que o projeto e o código podem ser utilizados por qualquer pessoa, o que também permite a sua modificação e livre distribuição (MCROBERTS, 2011).

Figura 10: Placa Arduino UNO

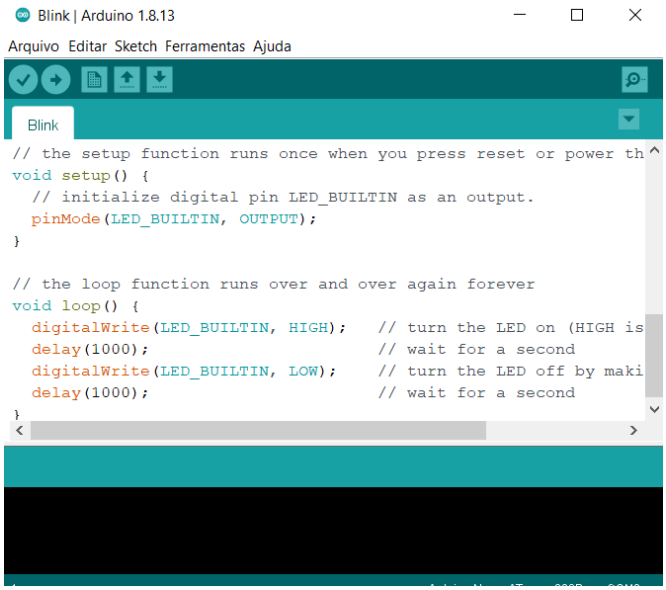


Fonte: FILIPEFLOP (2015)

2.9.1.1 IDE do Arduino

O *software* Arduino *Integrated Development Enviroment* (IDE) é uma ferramenta de código aberto utilizada para facilitar o processo de escrita e implementação do código em qualquer placa Arduino. Nesse ambiente é utilizada a linguagem C/C++ para desenvolver a lógica, sendo compatível com os sistemas operacionais Windows, Mac OS e Linux (ARDUINO, 2022).

Figura 11: Arduino IDE



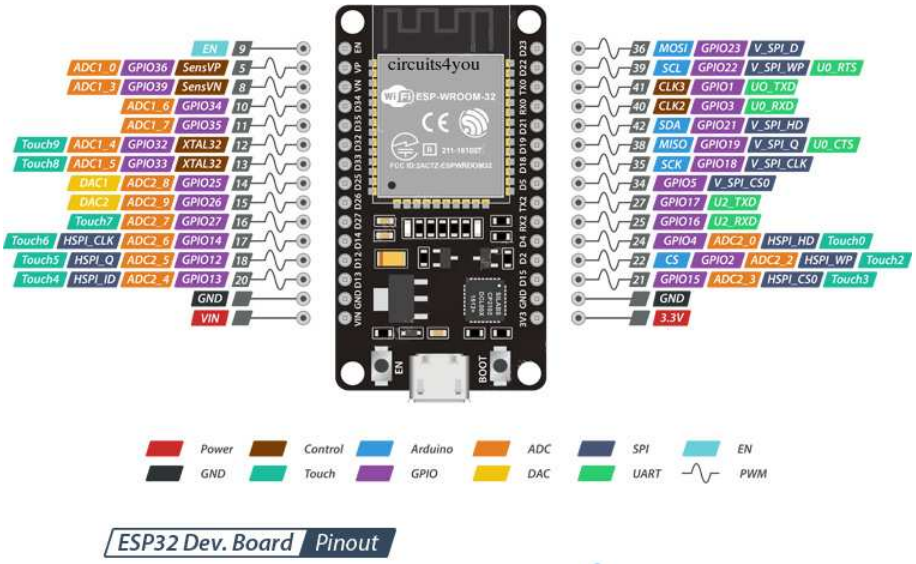
Fonte: QUINTINO (2021)

2.9.2 Módulo ESP32

O módulo ESP32 foi desenvolvido pela empresa DOIT (*Doctors of Intelligence & Technology*) similar à placa nodeMCU, que utiliza o *chip* ESP32-D0WDQ6, evolução do *chip* ESP8266, feito pela empresa Espressif. Este módulo inclui a utilização de WiFi, *Bluetooth* e microprocessador que permite o desenvolvimento de projetos simples até os mais complexos como a codificação de voz e transmissão de música.

É utilizado microprocessador Xtensa de 32 Bits, ele possui dois núcleos de CPU que podem ser controlados individualmente, podendo até desligar a CPU e utilizar o co-processador para monitoramento dos dispositivos periféricos. A antena de WiFi suporta uma taxa de 150 Mbps e potência de 20,5 dBm, permitindo uma conexão direta à internet. O *Bluetooth* permite com que dispositivos móveis se conectem ao módulo para enviar e receber informações, além de possuir baixo consumo de energia. Essa integração com WiFi e *Bluetooth* permite o desenvolvimento de diversos projetos voltados à IoT (Arduino Santa Efigenia, 2022).

Figura 12: *ESP32 DevKit v1*



Fonte: *Espressif* Systems (2022)

2.9.2.1 Características gerais

Tabela 2: Características gerais - Módulo ESP32

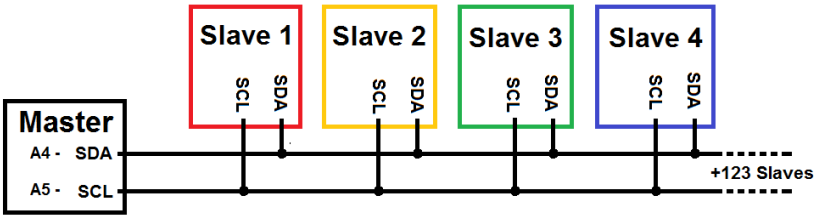
Chip Base	ESP32-D0WDQ6
Processador	Xtensa 32-Bit LX6 Dual Core
Clock	80 à 240 MHz (Ajustável)
Memoria ROM	448KB
Memória SRAM	520Kb
Memória Flash Externa	32-Bit de acesso e 4Mb
Tensão de Alimentação	4,5 à 12,0 VDC (Pino Vin)
Tensão de nível lógico	3,3VDC (não tolera 5V)
Corrente de consumo típica	80mA
Corrente de consumo máxima	500mA
Interfaces	Cartão SD, UART(3 canais), SPI (3 canais), SDIO, I2C (2 canais), I2S (2 canais), IR, PWM LED (2 canais) e PWM motor (3 canais)
Tipos GPIO	Digital IO (36), ADC 12-Bits (16 canais), DAC 8-Bits (2 canais), Sensor Capacitivo (10 canais); LNA pré-amplificador
WiFi	WiFi 802.11 b/g/n: 2.4 à 2.5 GHz
Segurança WiFi	WPA / WPA2 / WPA2-Enterprise / WPS
Criptografia WiFi	AES / RSA / ECC / SHA
Bluetooth	Bluetooth 4.2 BR / EDR e BLE (Bluetooth Low Energy)
RTC	RTC Integrado de 8Kb (Slown / Fast)
Sensor integrado	Temperatura e Hall
Temperatura de trabalho	-40° à +85° C
Dimensões	27,5 x 51,0 x 7,0 mm

Fonte: Arduino Santa Efigenia (2022)

2.9.3 Protocolo de comunicação I2C

O protocolo I2C, criado nos anos 90 pela Philips, trata-se de um protocolo de barramento serial com dois fios, bidirecional, que possibilita a conexão de dispositivos periféricos de baixa velocidade de comunicação à circuitos de aquisição e gerenciamento de dados. Os fios do protocolo são o SDA (dado serial) e o SCL (clock serial) (CASTRO, 2021).

Figura 13: Exemplo de esquema de protoclo I2C



Fonte: AUTODESK INSTRUCTABLES (2022)

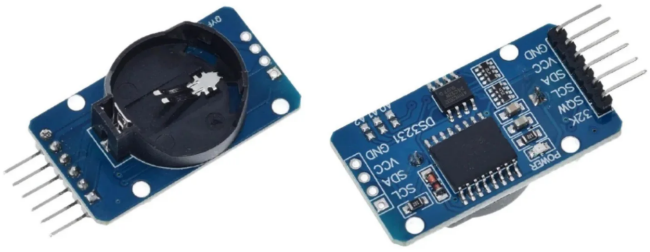
Cada dispositivo conectado ao barramento possui um endereço único e pode operar tanto como um transmissor (mestre) quanto como um receptor (escravo). O endereço de cada dispositivo é no formato hexadecimal, sendo possível escolher ou alterar dependendo do dispositivo. A comunicação acontece quando o mestre envia o endereço do escravo a ser comunicado pelo *Serial Data* (SDA), caso o dispositivo exista ele enviará um aviso, ou um pulso no pino *Serial Clock* (SCL), iniciando-se assim a transferência de dados (MADEIRA, 2017).

2.9.4 Real Time Clock

O módulo *Real Time Clock* (RTC) é um relógio de tempo real de alta precisão, ele fornece dados além dos segundo, minutos e horas, que podem ser tanto no formato 12h como 24h, como também informações de data se ajustando automaticamente para anos bissextos (THOMSEN, 2014).

O RTC DS3231 possui um circuito capaz de detectar falhas de energia, isso permite com que mesmo que o equipamento fique sem energia o módulo automaticamente aciona a bateria para que as informações não sejam perdidas. Suas informações são transmitidas através do protocolo I2C, é um circuito programável de 32kB de memória RAM e consome menos de 500nA (ELETROGATE, 2022).

Figura 14: Módulo RTC DS3231



Fonte: ELETROGATE (2022)

2.10 Sensores

O sensor é um dispositivo que tem como objetivo detectar e responder à algum estimulo ocorrido no ambiente, podendo ser estimulo químico ou físico como a umidade e a luminosidade. Assim a leitura é transformada em outra grandeza física para que possa ser feita a medição e monitoramento do ambiente. Para que o sinal do sensor possa ser lido pelo controlador geralmente utiliza-se um circuito de interface para manipular o sinal de saída do sensor, podendo ser um sinal do tipo analógico ou digital (WENDLING, 2010).

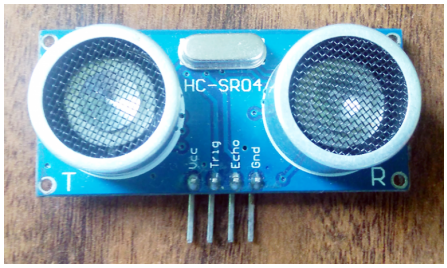
2.10.1 Sensores de distância

Existem vários tipos de sensores medidores de distância, mas para medir o nível de ração dentro do reservatório é possível que alguns tenham um desempenho melhor que outros.

2.10.1.1 Sensor ultrassônico

Envia impulsos sonoros em ultrassom e aguarda o retorno dessa onda para realizar a medição da distância, tudo o que é capaz de refletir o som pode ser detectado, com ele é possível medir enchimento, curvatura e altura sem entrar em contato com o material. O sensor ultrassônico mede a distância independente da cor e do acabamento da superfície do objeto (SILVEIRA, 2020).

Figura 15: Módulo de sensor ultrassônico HC-SR04



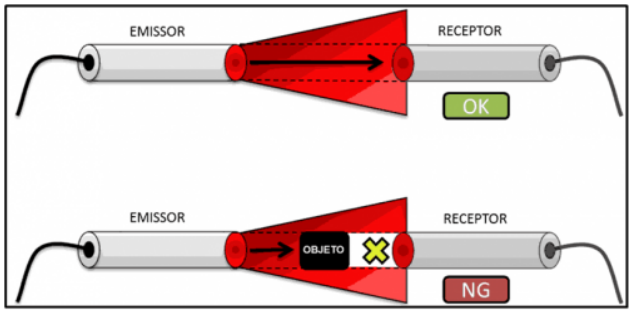
Fonte: GUIMARAES (2018)

2.10.1.2 Sensor óptico

Os sensores ópticos consistem no princípio da propagação da luz e existem basicamente três tipos (SILVEIRA, 2018):

Feixe contínuo ou barreira: possui duas unidades, uma é a emissora do feixe de luz e a outra é a receptora, quando um objeto intercepta o feixe luminoso por completo o sensor atuará;

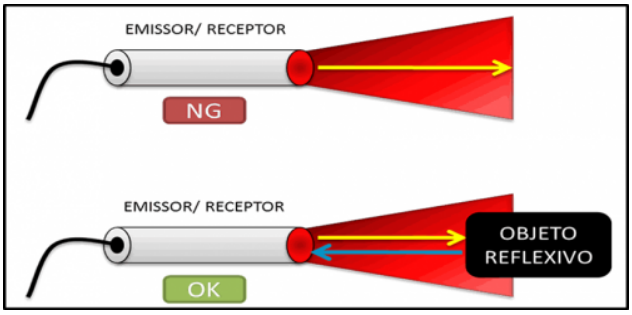
Figura 16: Sensor óptico de barreira



Fonte: SILVEIRA (2018)

Difuso: as unidades emissora de luz e receptora estão em um mesmo componente, espera-se que o feixe luminoso seja refletido por um objeto a fim de detectar a sua distância do sensor. Quanto mais escuro o objeto menor é a capacidade de detecção;

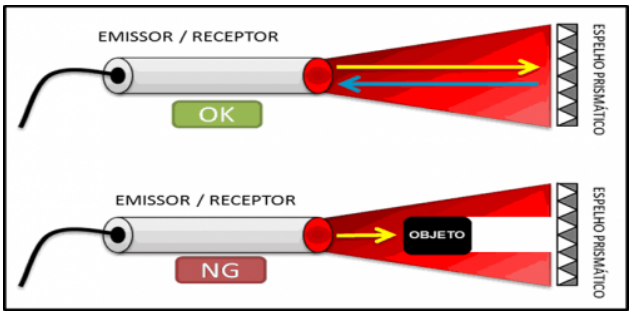
Figura 17: Sensor óptico difuso



Fonte: SILVEIRA (2018)

Retro reflexivo: diferente do tipo difuso, este tipo de sensor utiliza um espelho prismático para polarizar a luz emitida e retornar para o sensor. Com a polarização da luz é garantido que o sensor detecte apenas a luz emitida quando um objeto intercepta o feixe luminoso.

Figura 18: Sensor óptico retro reflexivo



Fonte: SILVEIRA (2018)

2.10.1.3 Sensor capacitivo

Seu princípio de funcionamento se dá pela aproximação de objetos metálicos ou não metálicos dentro do seu campo capacitivo. Quando um objeto se aproxima o valor da capacitância medida pelo sensor é alterado, dessa forma é possível identificar a distância de objetos (OLIVEIRA, 2019).

Figura 19: Sensor capacitivo

Fonte: SILVEIRA (2018)

2.10.1.4 Sensor indutivo

Utiliza o campo eletromagnético do sensor para identificar a proximidade de objetos metálicos em pequenas distâncias (PANIN, [s.d.]).

Figura 20: Modelos de sensor indutivo

Fonte: PANIN ([s.d.])

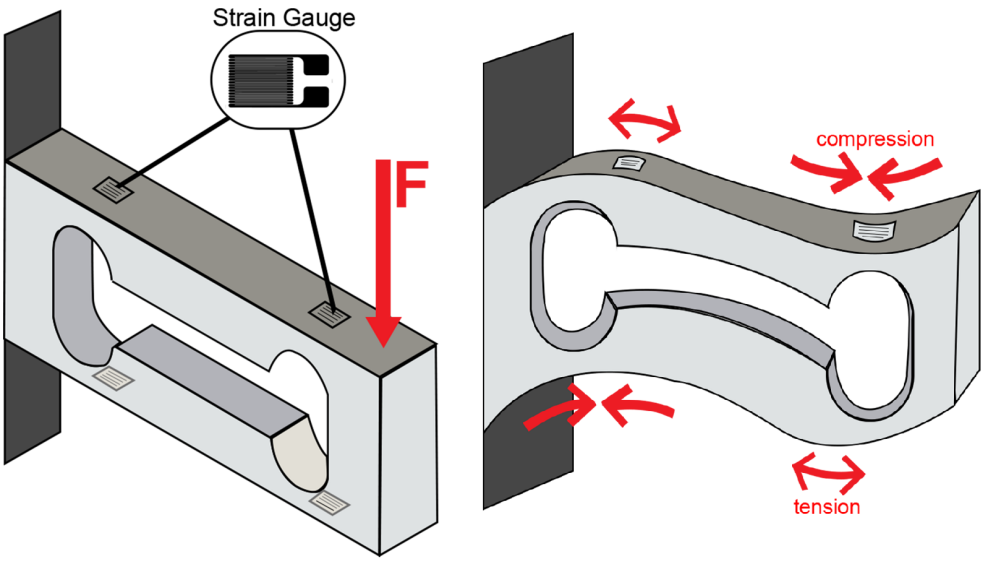
2.10.2 Célula de carga

A célula de carga é um dispositivo que transforma uma deformação de um corpo em um sinal de tensão ou corrente, que é proporcional ao valor da força aplicada por certa carga mecânica que gera a deformação (FRADEN, 2004).

O principal componente em uma balança é a célula de carga, que é geralmente feita de aço inox ou alumínio, onde os extensômetros, mais conhecidos como *strain gauges*, são fixados. Sabendo-se os parâmetros mecânicos do material da célula de

carga é possível quantificar o deslocamento sofrido ao se exercer tração ou compressão sobre o dispositivo. Os *strain gauges* são dispositivos cuja a propriedade elétrica de resistência é alterada com a aplicação de deformação elástica gerando então um sinal de saída. Quanto maior a deformação sofrida mais pesado é o objeto (LOCATELLI, 2019). A Figura 21 ilustra o princípio de funcionamento de uma célula de carga.

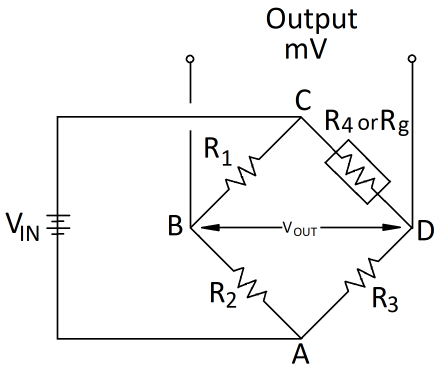
Figura 21: Princípio de funcionamento de célula de carga



Fonte: LOADSTARSENSORS (2022)

Uma célula de carga é constituída por um ou mais *strain gauges* em formação de ponte de *Wheatstone*, quando os sensores sofrem variação ôhmica a ponte sofre desbalanceamento fazendo com que varie o sinal de tensão gerado (FRADEN, 2004). O esquema elétrico da ponte de *Wheatstone* é representado na Figura 22 e o sinal de tensão de saída exemplificado é dada pela Equação 2.1:

Figura 22: Ponte de Wheatsone

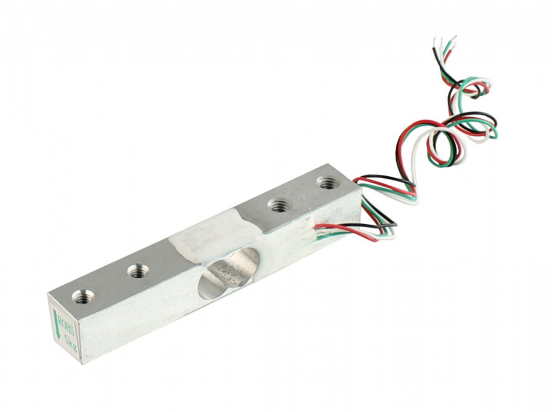


Fonte: OMEGA (2022)

$$V_{out} = V_{in}\left[\frac{R_3}{R_3 + R_g} - \frac{R_2}{R_1 + R_2}\right] \tag{2.1}$$

No projeto de uma balança é importante especificar qual o peso máximo que será lido, pois se o peso ultrapassar a capacidade de deformação da célula de carga as leituras dos pesos obtidos estarão incorretos podendo causar a deformação permanente do componente. Para aumentar a capacidade de peso é possível inserir no projeto mais de uma célula de carga (LOCATELLI, 2019).

Figura 23: Células de carga



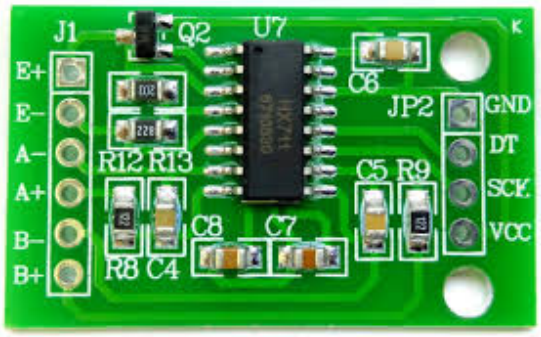
Fonte: USINAINFO (2022)

2.10.2.1 Módulo HX711

Para poder trabalhar com a célula de carga é preciso de um conversor analógico/-digital para interpretar o sinal elétrico gerado pela célula. O HX711 é um módulo amplificador e conversor de 24 bits, seu principio de trabalho é a conversão das mudanças de medidas na resistência em quantidade de tensão através de um circuito existente e amplificação do sinal de saída para que seja reconhecida pelo Arduino.

Este módulo tem uma estrutura bem simples, fácil de usar, apresenta resultados estáveis e confiáveis, tem alta sensibilidade e é capaz de medir rapidamente as variações. É utilizado quando se deseja atuar com as células de carga ligadas em ponte, amplificando o sinal de ganho por meio de seu multiplexador interno que seleciona as entradas diferenciais A+ e A- ou B+ e B-, sendo que o E+ e E- também representam, respectivamente, o VCC e o GND do circuito (LOCATELLI, 2019).

Figura 24: Módulo HX711



Fonte: LOCATELLI (2019)

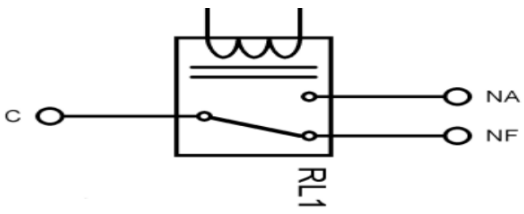
2.11 Atuadores

São chamados de atuadores dispositivos que transformam energia elétrica, pneumática ou hidráulica em energia mecânica, ou seja, os atuadores possuem a função de converter uma determinada energia em uma atuação mecânica, que é o movimento, realizando um trabalho (CYRINO, 2018).

2.11.1 Relé

O relé é um dispositivo eletromecânico, normalmente com um contato comum (C), um contato normalmente fechado (NF) e um contato normalmente aberto (NA). Para acionar o relé o circuito de comando alimenta a bobina fazendo com que os contatos tenham suas aberturas alternadas, ou seja, quando acionado o contato que antes estava normalmente aberto agora se fecha e o contato normalmente fechado se abre até que seja interrompido o fluxo de corrente elétrica na bobina do relé. O objetivo do relé é permitir com que o circuito comande uma carga maior de energia elétrica (FIRMINO; MATEUS, 2020).

Figura 25: Simbologia do relé



Fonte: FIRMINO; MATEUS (2020)

2.11.2 Motor de corrente contínua

Os motores são a principal forma de movimentar as partes mecânicas de dispositivos. Com os motores elétricos de corrente contínua é possível controlar a velocidade de rotação, acelerar, frear e reverter o sentido de rotação rapidamente. Alguns motores desse tipo permitem alta precisão da rotação do motor em questão de ângulo de giro como é o caso dos motores de passo e servomotores (MATTEDE, [s.d.]).

Para este projeto não é necessária a alta precisão de rotação, por isso não serão utilizados motores de passo nem servomotores, contudo a velocidade de rotação deve ser baixa, para isso será utilizado um motor com caixa de redução de velocidade.

Um motor de corrente contínua com caixa de redução funciona da seguinte maneira: a corrente elétrica é fornecida ao estator do motor através de terminais de corrente contínua, quando isso acontece o rotor gira devido à interação entre os campos magnéticos gerados pelo rotor e estator, a caixa de redução é acoplada ao eixo do motor e ela possui um conjunto de engrenagens que reduzem a velocidade do eixo de saída da caixa em comparação ao eixo do motor o que resulta em um torque maior na saída. Para controlar a velocidade e torque de um motor de corrente contínua basta regular a corrente elétrica fornecida ao rotor do motor.

Figura 26: Motor CC AK555/11.1PF12R83CE

Fonte: SARAVATI (2022)

2.12 Fonte de alimentação

Para que os circuitos eletrônicos funcionem é necessário energia elétrica. Considerando que a maioria dos circuitos utilizem uma baixa tensão contínua, a forma de energia fornecida pela rede elétrica não é ideal, por isso a necessidade de se converter a energia fornecida para uma que o circuito consiga trabalhar, é através de um circuito com uma configuração específica chamado de fonte de alimentação que a conversão é realizada. No caso da rede elétrica de 110V ou 220V encontrado nas tomadas a energia vem em forma alternada, nesse sentido a fonte de alimentação é responsável por transformá-la em uma corrente elétrica contínua. Em alguns casos as fontes de alimentação são utilizadas para subir ou baixar o nível de tensão, também servindo como um protetor do circuito contra os picos de energia e instabilidade na rede elétrica (EDUARDA, 2018).

3 MATERIAIS E MÉTODOS

Neste capítulo, serão abordados todos os procedimentos para realização do trabalho, assim como os materiais utilizados.

O primeiro passo para elaboração de um protótipo é o levantamento dos requisitos necessários para a escolha dos materiais utilizados e cálculos necessários para a elaboração do design mecânico do protótipo, visto que os detalhes construtivos de um produto são criados e não simplesmente escolhidos.

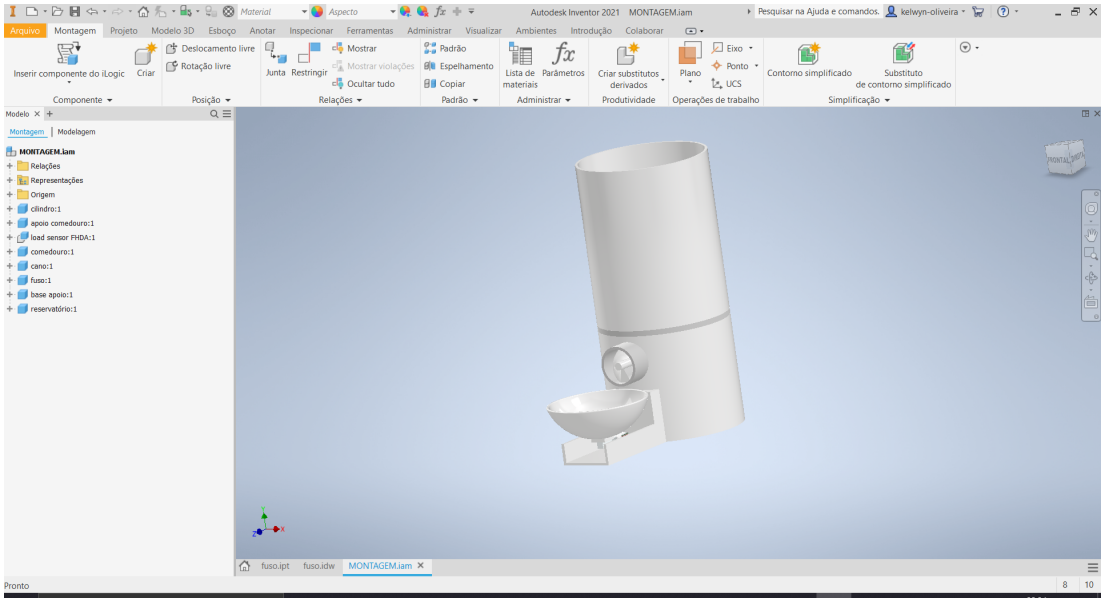
Para o desenvolvimento do sistema de controle remoto foi considerado um alimentador automático que inclui o depósito de ração, onde seu acionamento se dá por meio de um motor de corrente contínua, juntamente com o sistema de controle automático com o uso de aplicativo.

A seguir, são apresentados o projeto mecânico e dimensionamento do motor que ira transportar a ração para o comedouro do animal, o projeto eletrônico do protótipo com os sensores utilizados, a programação do *software* que irá conectar a maquina à internet, bem como processar as informações dos sensores e as informações fornecidas pelo tutor, bem como o desenvolvimento do aplicativo do alimentador.

3.1 Projeto mecânico

A projeto mecânico do protótipo foi realizado no *software Inventor* versão de 2021. A interface do *software* pode ser vista na figura 27.

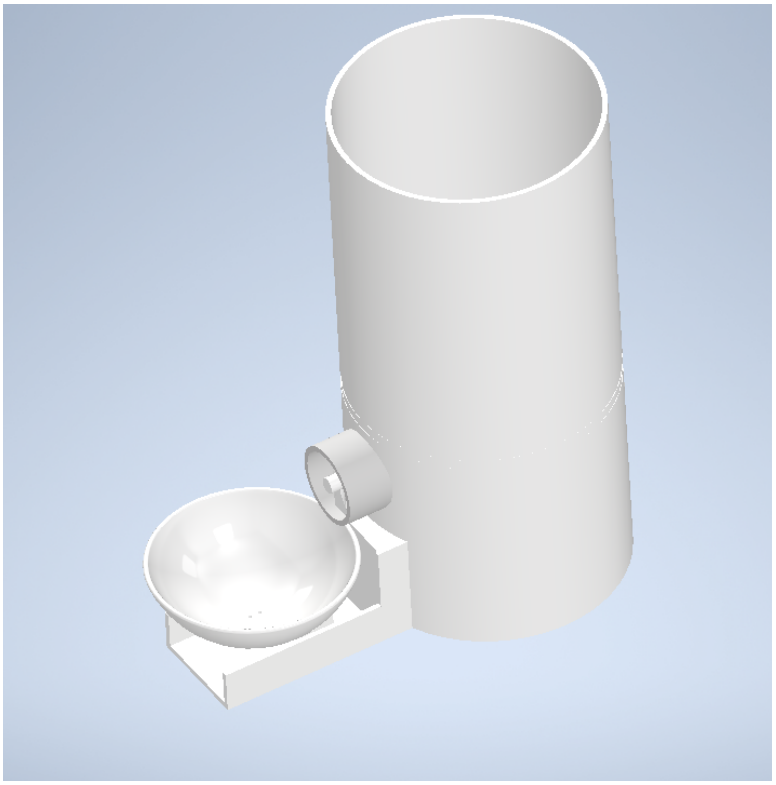
Figura 27: Ambiente de montagem do *software Inventor*



Fonte: Autor

Todas as peças foram projetadas para serem impressas utilizando impressora 3D. A figura 28 apresenta o protótipo montado no ambiente do *software Inventor*. O reservatório de alimento tem a capacidade de armazenar até 20 litros de ração.

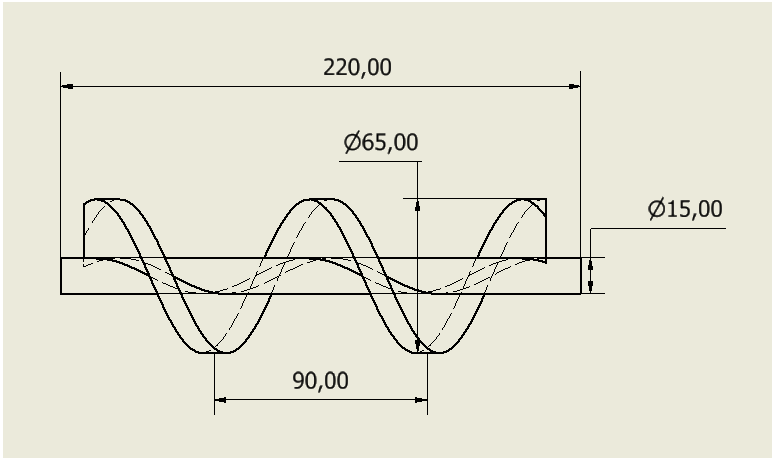
Figura 28: Modelo CAD do Alimentador



Fonte: Autor

Para realizar o transporte da ração armazenada no reservatório até o comedouro foi desenvolvido um mecanismo de dosagem, ele é composto por uma rosca helicoidal montada dentro da tubulação que fica abaixo do reservatório. O helicoide foi projetado com um diâmetro externo de 65 mm, diâmetro interno de 15 mm, com o passo de 90 mm e comprimento de 220 mm conforme mostra a figura 29.

Figura 29: Rosca de transporte ração



Fonte: Autor

A partir dos valores das peças projetadas realizou-se o dimensionamento do motor que seria necessário para transportar o alimento do reservatório até o comedouro do animal. Segundo Firmino (2020), a potência do motor é calculada conforme a equação 3.1.

$$P = 2,2 \cdot 10^{-4} \cdot Q \cdot \mu \cdot L \cdot \phi \tag{3.1}$$

Onde:

P = Potência do motor

Q = Capacidade de transporte em m^3/h

μ = Massa específica do material em kg/m^3

L = Comprimento total do transportador em m

ϕ = fator de potência

Considerando um transportador helicoidal a capacidade de transporte é calculada através da equação 3.2.

$$Q = 4,71 \cdot 10^{-5} \cdot p \cdot n \cdot (D^2 - d^2) \tag{3.2}$$

Onde:

Q = Capacidade de transporte em m^3/h

p = Passo da rosca em cm

n = Rotação em rpm

D = Diâmetro da rosca tubular em cm

d = Diâmetro interno do helicoide em cm

Incluindo 3.2 em 3.1 temos a equação 3.3:

$$P = 1,0362 \cdot 10^{-8} \cdot p \cdot n \cdot (D^2 - d^2) \cdot \mu \cdot L \cdot \phi \tag{3.3}$$

Os valores considerados no cálculo podem ser vistos na tabela 3.

Tabela 3: Valores usados no cálculo da potência do motor

Passo da rosca	$9cm$
Rotação	$65,4rpm$
Diâmetro da rosca tubular	$6,5cm$
Diâmetro interno do helicoide	$1,5cm$
Massa específica do material	$400kg/m^3$
Comprimento total do transportador	$0,22m$
fator de potência	$0,4$

Fonte: Autor

$$P = 1.0362 \cdot 10^{-8} \cdot 9 \cdot 65,4 \cdot (6,5^2 - 1,5^2) \cdot 400 \cdot 0,22 \cdot 0,4$$

$$P = 8,5810^{-3}cv$$

$$P = 6,32W$$

Para fazer a seleção do motor também foi calculado o torque necessário para que o transportador saia do estado inicial zero. A equação utilizada foi a equação 3.4.

$$T = \frac{30 \cdot P}{\pi \cdot n} \tag{3.4}$$

Onde:

T = Torque em Nm

P = Potência requeria pelo transportador em kW

p = Rotação rpm

$$T = \frac{30 \cdot 0,00632}{\pi \cdot 65,4}$$

$$T = 0,923Nm$$

$$T = 9,41kgfcm$$

Levando-se em consideração os valores obtidos nos cálculos o motor escolhido foi o motor de corrente contínua com caixa de redução AK555/11.1PF12R83CE da fabricante Akiyama. Suas características são:

Tabela 4: Caracteristicas do motor CC AK555/11.1PF12R83CE

Corrente de partida	$6A$
Potência	$5W$
Tensão (nominal)	$12V$
Tensão (faixa de operação)	$6V - 24V$
Torque	$11,1kgf.cm$
Velocidade (sem carga)	$83rpm$
Velocidade (carga máxima)	$65,4rpm$
Peso unitário	$315g$
Código	$AK555/11.1PF12R83CE$

Fonte: (NEOMOTION, 2022)

3.2 Projeto eletrônico

A montagem dos componentes se deu de forma gradativa, onde o primeiro componente a ser conectado foi o botão que vai fazer o pedido de conexão com a internet, ele foi conectado ao GND e ao pino D25 do ESP32, quando pressionado será enviado o sinal *LOW* para o ESP32 que iniciará a busca pela rede *WiFi*.

O módulo do sensor de distância ultrassônico HC-SR04, responsável por verificar a quantidade de ração ainda disponível no reservatório do comedouro. O pino VCC foi conectado à tensão de 5V e os pinos *Trigger* e *Echo* forar conectados aos pinos D15 e D2 respectivamente do ESP32.

O princípio de funcionamento do módulo HC-SR04 se baseia no tempo que leva para uma onda enviada pelo módulo voltar para ele. Quando o ESP32 envia um sinal *HIGH* para o *Trigger* o módulo envia 8 pulsos de 40 kHz e aguarda o retorno. Quando o sinal de retorno é percebido é enviado um sinal de nível *HIGH* para o pino *Echo* e ele fica nesse nível durante todo o período de envio e recepção do sinal ultrassônico (FERREIRA, 2022). O cálculo da distância se dá por:

$$D = \frac{V_s \times T_{echo}}{2} \tag{3.5}$$

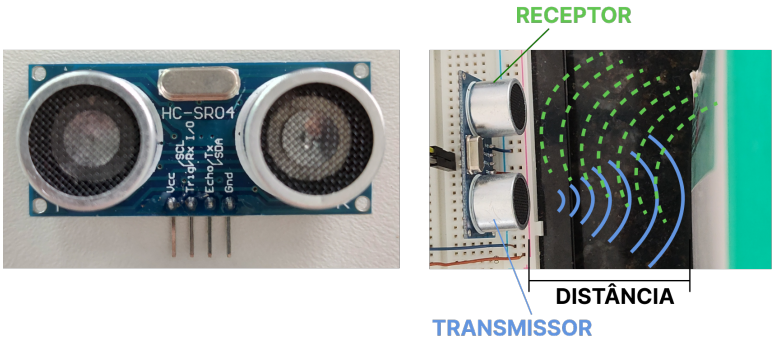
Onde:

D = Distância

V_s = Velocidade do som

T_{echo} = Tempo de *Echo* em nível alto

Figura 30: Funcionamento do módulo HC-SR04



Fonte: Autor

Para se medir o peso de alimento no comedouro foi utilizada a célula de carga de 5kg e o módulo amplificador HX711 para auxiliar na leitura do peso medido. O fio de cor vermelha da célula de carga foi conectado ao pino E+ do amplificador, e os fios preto, branco e verde foram conectados aos pinos E-, A- e A+ do amplificador respectivamente. O amplificador teve seu pino VCC conectado à tensão de 3,3V do ESP32 e os pinos SCK e DT foram conectados aos pinos D18 e D19 respectivamente ao ESP32.

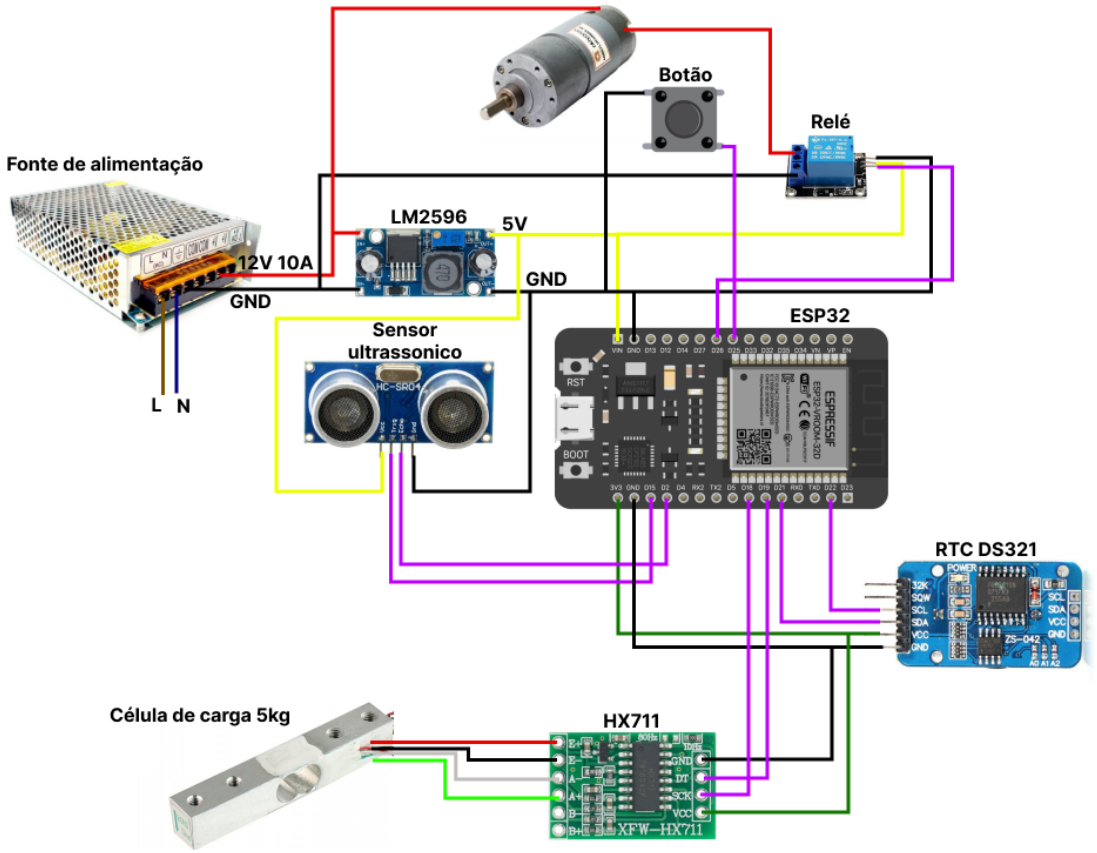
O módulo RTC DS3231, responsável por manter a hora do sistema teve seu pino VCC conectado a tensão de 3,3V do módulo ESP32, e os pinos SDA e SCL foram conectados aos pinos D21 e D22 do ESP32, que é onde ocorre a comunicação entre os dois componentes através do protocolo I2C, onde o controlador ESP32 é o mestre e o módulo RTC o escravo.

A alimentação do circuito é feita através de uma fonte DC de 12 V e 10 A, ela é ligada ao regulador de tensão LM2596, que por sua vez ajusta a tensão de saída para 5 V utilizada para alimentar o controlados ESP32 através do pino VIN.

Para ligar e desligar o motor foi utilizado um relé, sua ativação acontece quando o pino D26 do ESP32 envia sinal digital *HIGH* e desliga quando o sinal é *LOW*.

A figura 31 apresenta o esquema de ligação do circuito eletrônico.

Figura 31: Esquema de ligação entre componentes eletrônicos

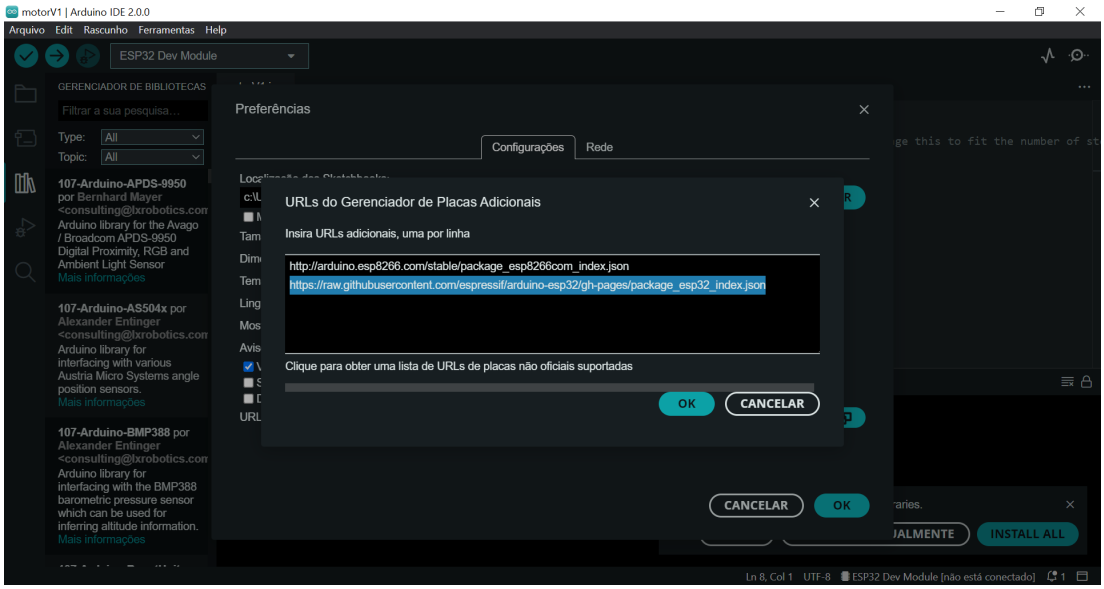


Fonte: Autor

3.3 Programação do *software*

O código foi escrito no ambiente de desenvolvimento do Arduino, através dele é feita a conversão do código escrito em uma determinada linguagem para uma outra que o controlador consiga entender. Para que o ESP32 fosse reconhecido pela IDE foi incluído a URL <https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json> no gerenciador de placas adicionais. Para fazer essa inclusão é aberta a IDE, na aba Arquivo escolhe-se Preferências, em URLs do Gerenciador de Placas Adicionais é adicionado a URL.

Figura 32: Adição de URL para placa ESP32 na IDE do Arduino



Fonte: Autor

Ao abrir a aba Ferramentas, em placa é mostrada a opção esp32. Para acessar o módulo utilizado foi escolhida opção ESP32 Dev Module.

Para que o *software* funcione foram adicionadas as bibliotecas que configuram a conexão do ESP32 à rede WiFi, bem como a biblioteca de comunicação MQTT, a biblioteca para uso do módulo RTC, módulo ultrassônico e o módulo amplificador HX711.

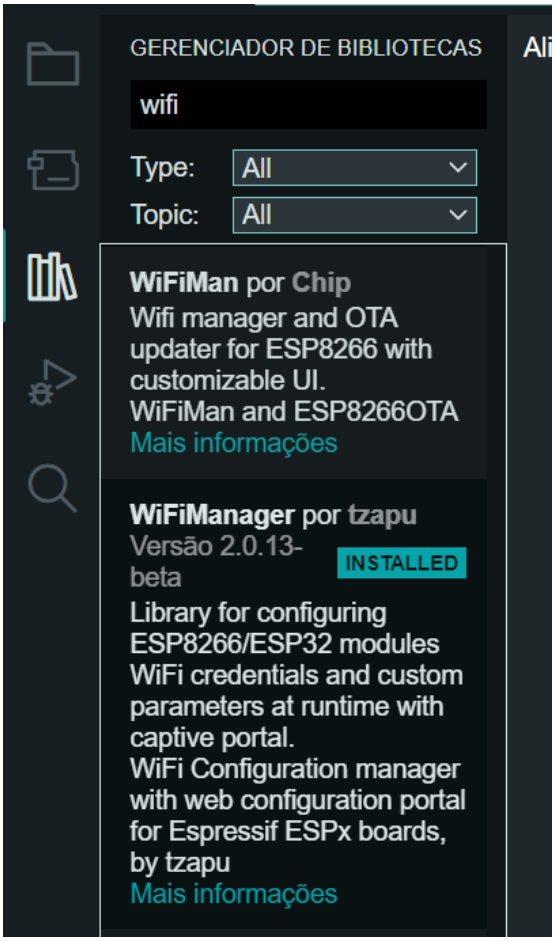
Figura 33: Bibliotecas utilizadas

```
1  #include <WiFi.h> // Configuração de conexão com WiFi
2  #include <WiFiManager.h> //App para conectar o ESP a rede WiFi
3  #include <PubSubClient.h> //biblioteca MQTT de publisher/subscribe
4  #include "RTClib.h" //Carrega a biblioteca do RTC
5  #include <Ultrasonic.h> //Carrega a biblioteca do sensor ultrassonico
6  #include "HX711.h" //Carrega a biblioteca do amplificador para celula de carga
```

Fonte: Autor

A maioria das bibliotecas foram adicionadas pesquisando no próprio gerenciador de bibliotecas da IDE, apenas a biblioteca Ultrasonic.h foi feito o *download* de um repositório do github e descompactada na pasta *libraries* da IDE.

Figura 34: Adição das bibliotecas



Fonte: Autor

Tabela 5: Bibliotecas

Biblioteca	Nome	Autor	Versão
WiFi.h	WiFi	Arduino	1.2.7
WiFiManager.h	WiFiManager	tzapu	2.0.13-beta
PubSubClient.h	PubSubClient	Nick O'Leary	2.8
RTClib.h	RTClib	Adafruit	1.2.0
Ultrasonic.h	github de filipflop	Carl Nobile	
HX711.h	HX711 Arduino Library	Bogdan Necula	0.7.5

Fonte: Autor

3.4 Conexão à Internet

A conexão à internet é feita através do aplicativo gerado pela biblioteca WiFiManager. Configurou-se para que quando o botão conectado ao pino D25 do ESP32 seja pressionado e solto dê-se início a configuração de qual rede WiFi se deseja conectar o alimentador. Com esse botão é possível que o usuário possa transportar o alimentador e se conectar a qualquer outra rede WiFi que ele desejar. A figura 35 apresenta o trecho de código que faz a configuração de conexão.

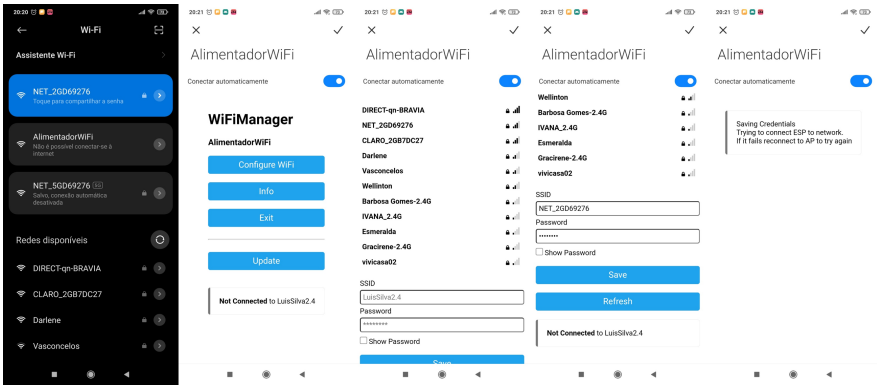
Figura 35: Trecho de código que faz a conexão com a internet

```
/* Setup de conexão WiFi */
void setup_wifi(){
  WiFiManager wm;

  wm.setConfigPortalTimeout(timeout);

  if (!wm.startConfigPortal("AlimentadorWiFi")) {
    Serial.println("A conexão falhou ou atingiu o limite de tempo");
    delay(3000);
    ESP.restart();
    delay(5000);
  }

  Serial.println("ESP32 conectado à internet");
}
```

Fonte: Autor

Após pressionado e solto o botão é necessário abrir a opção de conexão WiFi em qualquer dispositivo que permita essa conexão, escolher a rede AlimentadorWiFi, em WiFiManager escolhe-se a opção Configure WiFi. Após isso são apresentadas as opções de redes próximas, basta escolher e fazer a conexão. A figura 36 apresenta as imagens da tela de um dispositivo móvel configurando a rede WiFi a qual o protótipo irá se conectar.

Figura 36: Conectando o protótipo à rede WiFi



Fonte: Autor

3.4.1 Comunicação MQTT

A comunicação, usando o protocolo MQTT, é feita através de um cliente publicador que manda a mensagem para um tópico, essa mensagem chega à um servidor que então distribui a mensagem para os clientes inscritos no tópico.

No código foi configurado primeiro o servidor, também chamado de *broker*. Foi utilizado o *broker* gratuito da HiveMQ.

Figura 37: Configuração do *broker*

```
const char* mqtt_server = "broker.hivemq.com"; //broker mqtt
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
```

Fonte: Autor

Após se conectar à rede WiFi é feita a inscrição nos tópicos que se deseja acompanhar. O código apresentado na figura 38 é executado quando o cliente ainda não está conectado ao servidor MQTT, se a conexão acontecer é então feita a subscrição nos tópicos listados.

Figura 38: Inscrição nos tópicos que se deseja acompanhar

```
while(!client.connected() ){
  Serial.print("Tentando conexão MQTT...");
  //Cria um cliente ID
  String clientId = "ALIMENTADOR_MQTT";
  clientId += String(random(0xffff), HEX);
  //Tentativa de conexão
  if (client.connect(clientId.c_str())){
    Serial.print("Conectado");
    client.subscribe("alimentador/motor");

    client.subscribe("alimentador/alarme1");
    client.subscribe("alimentador/hora1");
    client.subscribe("alimentador/peso1");
```

Fonte: Autor

Por exemplo, o protótipo tem a função de despejar alimento sem que seja necessário a programação de um horário para isso. Caso o tutor queira despejar de forma “manual” basta que ele pressione e segure o botão despejar para saber se o usuário clicou no botão para despejar no aplicativo. Quando o botão despejar for pressionado é enviado para o tópico alimentador/motor a mensagem “ligar” e quando o botão é solto é enviada a mensagem “desligar”. Para que o ESP32 leia a mensagem é preciso que ele esteja inscrito no tópico alimentador/motor.

O alimentador passa então a acompanhar todas as informações que chegam nos tópicos aos quais ele se inscreveu. Quando é recebida uma mensagem, a função de *callback* guarda o tópico em uma variável *string* e a mensagem recebida no tópico também é salva em uma *string*.

No trecho de código da Figura 39 podemos ver que quando o tópico alimentador/-motor recebe a mensagem ligar o relé que ativa o motor para despejar o alimento é ligado, caso contrário ele será desligado.

Figura 39: Trecho de código da função de leitura das mensagens enviadas para os tópicos

```
void callback(char* topic, byte* payload, unsigned int length){
  String topico = topic;
  String mensagem;
  for (int i = 0; i < length; i++) {
    char c = (char)payload[i];
    mensagem += c;
  }

  String topMotor = "alimentador/motor";

  if(topico == topMotor){
    if(mensagem == "ligar"){
      digitalWrite(RelePin, HIGH); //Aciona o relé para ligar o motor
    }else{
      digitalWrite(RelePin, LOW);
    }
  }
```

Fonte: Autor

Para que as informações dos sensores do alimentar apareçam no aplicativo é preciso que o protótipo seja um cliente publicador de alguns tópicos. A Figura 40 apresenta o código da função que envia para o tópico alimentador/hora o horário em que o alimentador está atualmente, isso é feito para auxiliar o tutor a programar os horários em que a ração seja despejada.

Figura 40: Função que envia o horário no protótipo

```
String hora_reservatorio(){
  String hora;
  DateTime now = rtc.now(); //CHAMADA DE FUNÇÃO
  char tempo[6];
  sprintf( tempo, "%02hhu:%02hhu", now.hour(), now.minute() );
  for (int i = 0; i < 5; i++) {
    char c = (char)tempo[i];
    hora += c;
  }
  sprintf(msg, "%s", tempo);
  client.publish("alimentador/hora", msg);

  return hora;
}
```

Fonte: Autor

3.5 Automação e controle da despensa de alimento

Segundo Aurélio, dicionário da língua portuguesa (FERREIRA, 2010, p. 80), a automação é um “sistema automático pelo qual os mecanismos controlam seu próprio funcionamento, quase sem a interferência do homem”.

O protótipo foi desenvolvido para que seja substituída a ação do tutor de pessoalmente ter que despejar o alimento para o seu animal, o que torna o alimentador um mecanismo automático. A despensa de alimento ocorre quando o tutor configura um horário para despejar o alimento, configura o peso e ativa o alarme.

O código foi escrito para que seja constantemente monitorado se o alarme está ativado, além disso é verificada a hora atual no reservatório e a hora que foi programada para despejar o alimento. Caso o alarme esteja ativado e o horário atual no reservatório e o escolhido para despejar a ração sejam iguais é feito então o acionamento do motor.

Figura 41: Trecho de código que ativa a despensa de alimento

```
//Alarme 1
String switch1Arduino = alarme1;
sprintf(msg, "%s", alarme1);
client.publish("alimentador/switch1Arduino", msg);
String hora1Arduino = hora1;
sprintf(msg, "%s", hora1Arduino);
client.publish("alimentador/hora1Arduino", msg);
String peso1Arduino = peso1;
int intPeso1Arduino = atoi( peso1Arduino.c_str() );
sprintf(msg, "%s", peso1Arduino);
client.publish("alimentador/peso1Arduino", msg);

if((alarme1 == "Ligado") && (hora == hora1)){
  ligar_Motor(intPeso1Arduino);
}
```

Fonte: Autor

A função que aciona o motor recebe como parâmetro o peso configurado no alarme como o peso que o usuário deseja que tenha de ração no comedouro. Após medir o peso no comedouro é acionado o relé que liga o motor. Para controlar quando o relé que aciona o motor deve ser ligado ou desligado é utilizada a função *while*, essa função verifica constantemente se o peso na comedouro é menor que o peso programado.

Quando o peso no comedouro atinge o valor desejado o relé é desativado.

Figura 42: Função que liga o motor para despejar alimento

```
// Função para ligar o motor
void ligar_Motor(int pesoAlarme){
  int pesoDesejado = pesoAlarme;
  int pesoComedouro = peso_Comedouro();

  digitalWrite(RelePin, HIGH);
  while(pesoComedouro < pesoDesejado){
    digitalWrite(RelePin, HIGH);
    pesoComedouro = peso_Comedouro();
  }
  digitalWrite(RelePin, LOW);
}
```

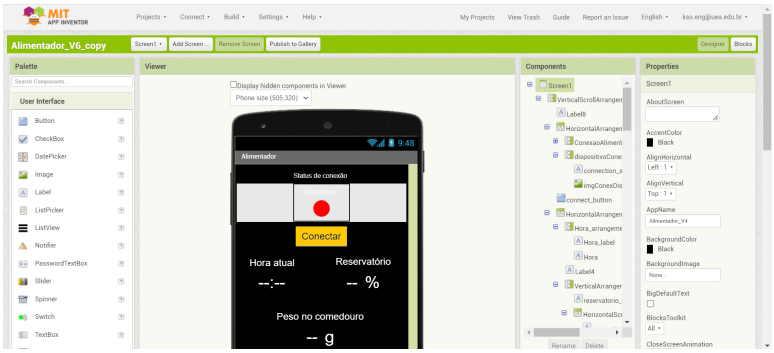
Fonte: Autor

3.6 Desenvolvimento do aplicativo

O aplicativo foi desenvolvido na plataforma do MIT APP Inventor. A sua escolha se deu pelo fato de o aplicativo estar disponível tanto para sistemas *Android* quanto para *iOS*.

O *layout* do aplicativo foi desenvolvido apenas arrastando os itens listados à esquerda para a tela do dispositivo representado no navegador. Do lado direito os nomes dos itens e sua hierarquia.

Figura 43: Ambiente de desenvolvimento de *layout* do aplicativo



Fonte: Autor

O MIT App Inventor permite a visualização do *layout* do aplicativo em um celular sem precisar fazer o *download*, função utilizada em todo o processo de desenvolvimento do aplicativo. Na Figura 44 é mostrado a tela inicial do aplicativo.

Figura 44: Tela inicial do aplicativo

Fonte: Autor

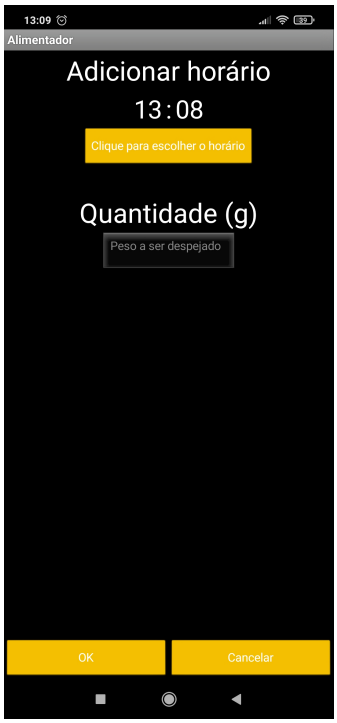
Na tela inicial o usuário fará a conexão entre o aplicativo e o alimentador que deve estar conectado à internet. Após a conexão ele poderá ver se o alimentador também

está conectado à internet. Em Hora atual é apresentado o horário no alimentador, em Reservatório o usuário acompanha a quantidade, em porcentagem, do nível de ração no reservatório do alimentador. O item Peso no comedouro é exibido o peso de ração presente no comedouro do alimentador.

Outra função do aplicativo é poder despejar a ração de forma manual, ou seja, sem precisar programar um horário, bastando apenas pressionar o botão despejar. Além disso o usuário pode programar até cinco refeições diárias, podendo programar o horário e peso de cada refeição. A refeição programada pode ser habilitada ou desabilitada através dos botões Ligar e Desligar, acima de cada horário programado é apresentado um texto que indica se o alarme está Ativado ou Desativado.

Ao clicar na engrenagem o usuário pode configurar o horário e o peso a ser despejado para o animal. A Figura 45 apresenta a tela de configuração.

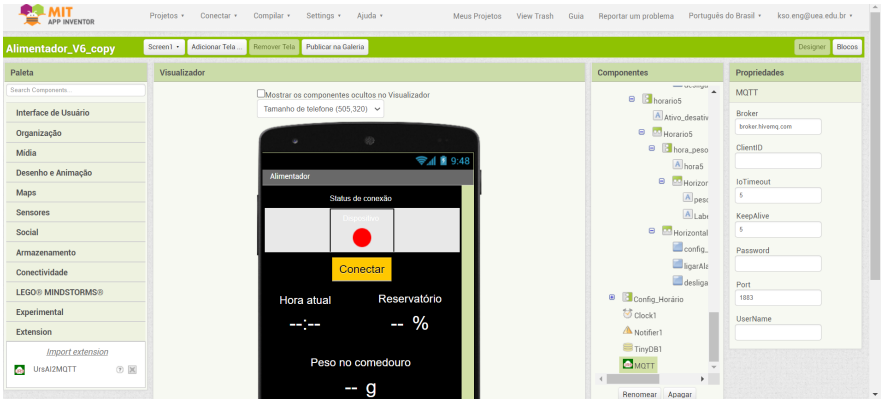
Figura 45: Tela de configuração



Fonte: Autor

Para programar o aplicativo é utilizado a programação em blocos. Como forma de comunicação entre o aplicativo e o alimentador é utilizado o protocolo MQTT, para isso foi feita a importação da extensão AI2 MQTT Extension para o MIT App Inventor. E o broker utilizado foi o broker.hivemq.com.

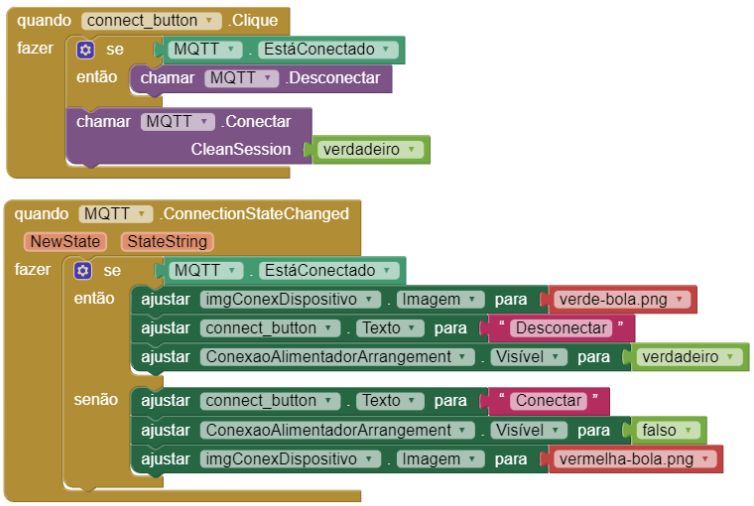
Figura 46: Extensão para uso do protocolo MQTT



Fonte: Autor

Para que o aplicativo se comunique com o dispositivo alimentador é preciso fazer a conexão do aplicativo ao *broker*. Para isso deve ser utilizado o botão Conectar. Quando pressionado é verificado se já está conectado, caso esteja desconectado é feita a conexão, além disso, todas as vezes que o estado de conexão é alterado é feita a apresentação do estado de conexão através de figuras, bola verde para o estado em que o aplicativo está conectado e bola vermelha para o estado de desconexão. Os blocos são postos segundo a Figura 47.

Figura 47: Blocos para conexão do aplicativo ao *broker*

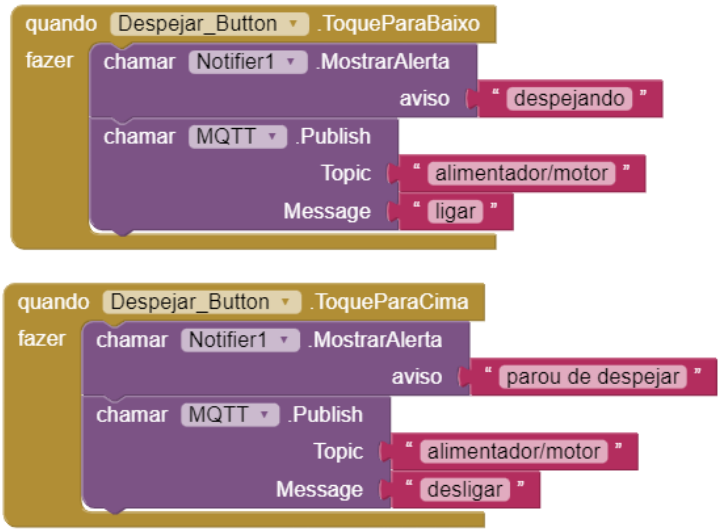


Fonte: Autor

A fim de que também possa ser feito o despejo de ração de forma manual o botão Despejar foi inserido. Assim quando o usuário pressionar o botão e segurá-lo o motor

do alimentador irá ligar até que o botão Despejar seja solto. A Figura 48 apresenta os blocos utilizados para despejar a ração de forma manual.

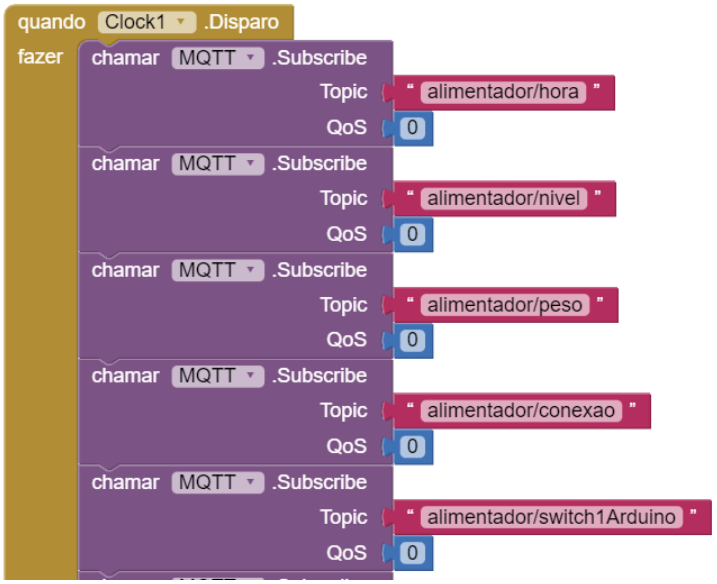
Figura 48: Blocos para ativar e desativar o despejo de ração de forma manual



Fonte: Autor

Para receber as mensagens enviadas pelos sensores do Arduino é preciso que o aplicativo seja um subscrito dos tópicos, o que justifica os blocos da Figura 49

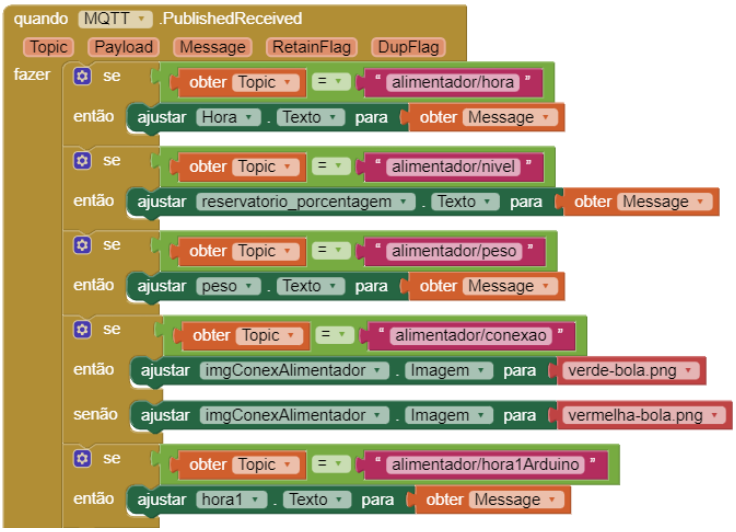
Figura 49: Blocos para subscrição nos tópicos MQTT



Fonte: Autor

Ao receber uma atualização no tópico, o aplicativo utiliza a mensagem para executar alguma ação, como apresentar as horas e nível do reservatório. Na Figura 50 é apresentado o bloco de como as mensagens recebidas são tratadas.

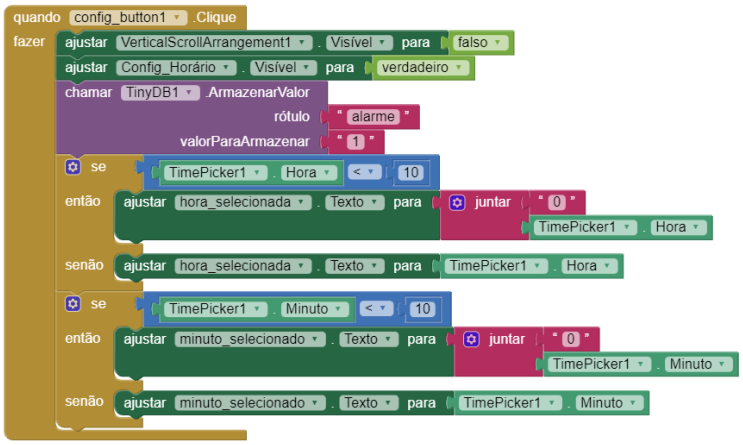
Figura 50: Blocos de leitura das mensagens



Fonte: Autor

Ao pressionar o botão em formato de engrenagem é apresentada a tela de configuração. A informação de qual alarme entre os cinco foi selecionado é salvo em um banco de dados no celular, fazendo a distinção de qual alarme está sendo configurado. Dessa forma é possível utilizar uma mesma tela de configuração para os alarmes, ou seja, não é necessário criar uma tela de configuração para cada um dos alarmes.

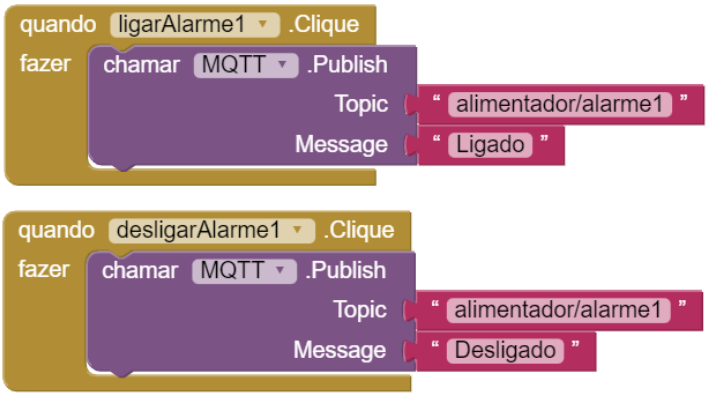
Figura 51: Blocos para configuração do alarme



Fonte: Autor

Os botões para ligar e desligar o alarme que despeja a ração foi configurado para enviar uma mensagem ao tópico alimentador/alarme com as mensagens Ligado quando se deseja ativar ou Desligado quando se deseja desativar o alarme.

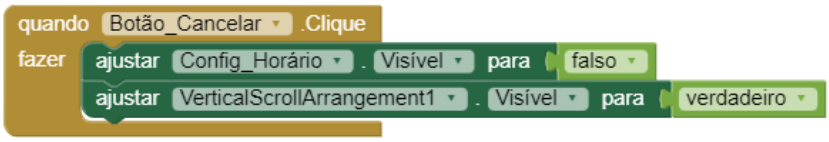
Figura 52: Blocos de botão para ligar o alarme



Fonte: Autor

Na tela de configuração, ao se pressionar o botão cancelar é fechada a tela de configuração e aberta a tela inicial sem que qualquer alteração feita seja salva.

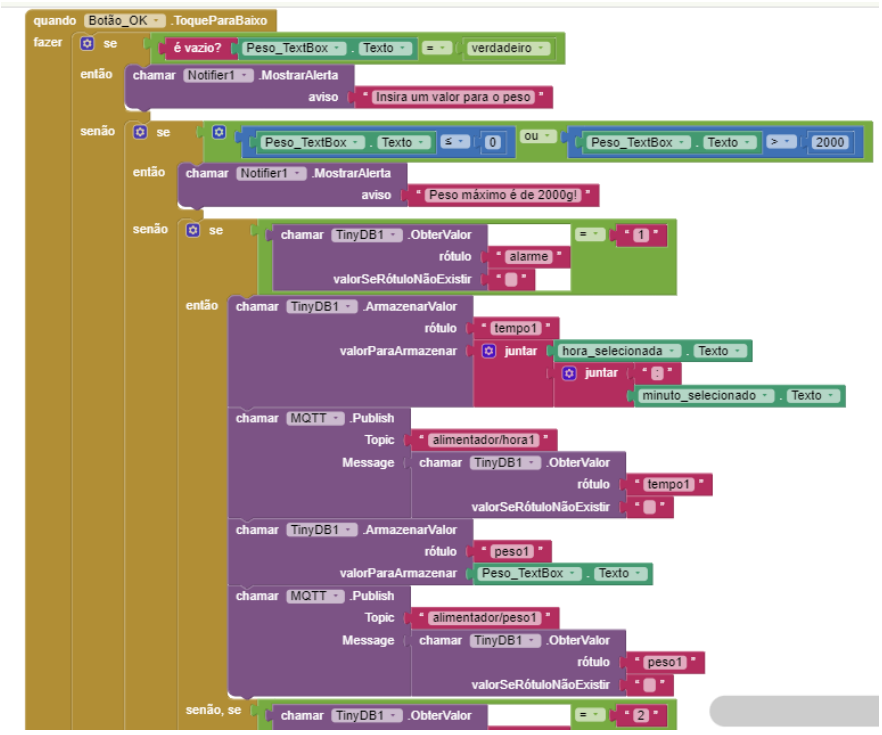
Figura 53: Blocos do botão cancelar configuração



Fonte: Autor

Ao pressionar o botão OK na tela de configuração é verificado se foi informado o peso que deve ser despejada a ração, sendo que o peso deve ser um valor maior que 0 até 2000 gramas. Então é verificado o banco de dados para verificar qual alarme está sendo configurado e então são enviadas mensagens para os tópicos que representam o horário e o peso configurado no alarme.

Figura 54: Blocos do botão de OK



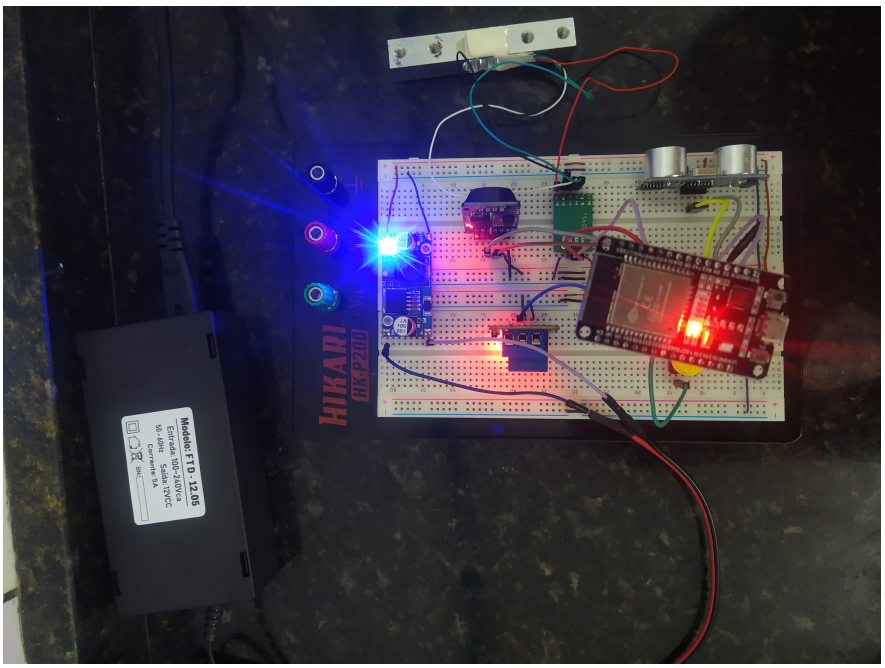
Fonte: Autor

4 RESULTADOS E DISCUSSÃO

Com este projeto obtivemos um aplicativo capaz de controlar remotamente um circuito representando um alimentador automático para cães que proporciona uma autonomia de cinco refeições programáveis em um período de 24 horas, atendendo assim a maioria das rotinas diárias de alimentações recomendadas.

Durante três dias seguidos foram realizados os testes que incluíram o funcionamento do circuito eletrônico, o desempenho do microcontrolador ESP32, a conexão WiFi e a interação do tutor via aplicativo. A Figura 55 apresenta a montagem do circuito eletrônico para o teste de funcionamento da interação com o aplicativo.

Figura 55: Circuito eletrônico montado

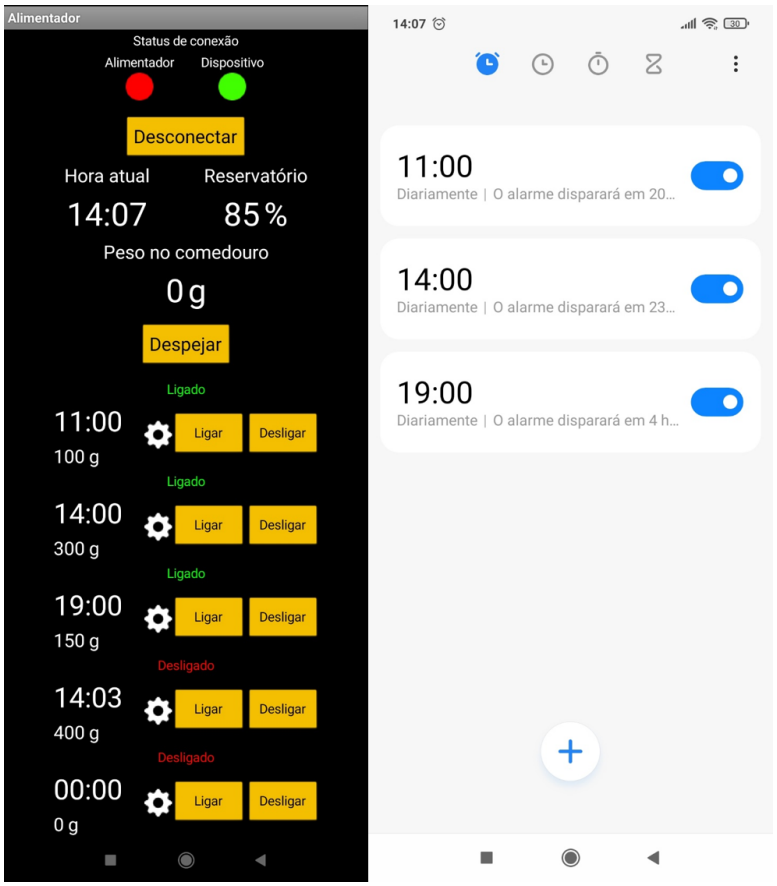


Fonte: Autor

Para os três dias foram configurados três horários para realizar a despejo de alimento:

11:00, 14:00 e 19:00. Para simular o peso do alimento despejado foi realizado pressão sobre a célula de carga com as próprias mãos. Para fazer o acompanhamento do horário do despejo foram configurados alarmes para os horários especificados anteriormente, Figura 56.

Figura 56: Configuração dos horários de teste



Fonte: Autor

Após conectar o circuito à rede WiFi e configurar todos os horários realizou-se o teste de conexão que consistia em desligar o roteador de internet por um período de tempo e depois ligar novamente. Verificou-se que o alimentador é capaz de se reconectar sem perder as informações já antes programadas.

Contudo, se o dispositivo receber a mensagem para ligar o motor e o alimentador se desconectar da rede WiFi, o motor permanecerá ligado até que o alimentador se reconecte e receba a mensagem para desligar o motor. Isso quer dizer que o motor permanecerá despejando alimento até que a reconexão seja reestabelecida e a mensagem que desativa o motor seja recebida pelo alimentador.

No segundo dia de teste a rede elétrica sofreu oscilação devido a uma forte chuva e o dispositivo desligou por um breve período de tempo. Após a rede elétrica se estabilizar o dispositivo ligou novamente, contudo todas as informações configuradas haviam sido perdidas tendo novamente que configurar o canal de conexão WiFi bem como os horários de despejo de alimento. Demonstrando assim que uma falha no fornecimento de energia elétrica, mesmo que por um breve momento, comprometeria a funcionalidade do alimentador o que pode preocupar o tutor.

Observou-se que o aplicativo deixou a programação das refeições muito mais fácil para o usuário, consequentemente reduzindo o custo do projeto visto que não se faz necessário o uso de botões e *display's* no protótipo para configurar os horários.

O sistema é capaz de realizar todas as funções propostas pelo projeto como armazenar e transmitir para o aplicativo o horário no alimentador, o nível de ração no reservatório, o peso de alimento no comedouro e ativação do relé que aciona o motor quando chega o horário corresponde ao programado ou quando o tutor pressiona o botão para despejar ração.

5 CONCLUSÃO

Este trabalho teve como objetivo o desenvolvimento de sistema de controle remoto para um alimentador automático de animais domésticos. Para isso foram pesquisados alimentadores já no mercado que poderiam auxiliar no desenvolvimento do projeto.

Foi apresentada uma revisão dos conceitos mais pertinentes para a compreensão do trabalho, em particular acerca dos dispositivos eletrônicos utilizados e o protocolo utilizado para realizar a comunicação com a internet.

Uma vez que o projeto mecânico foi desenvolvimento, foi possível dimensionar o motor que seria utilizado para despejar o alimento.

Durante três dias seguidos de experimentos foi avaliado o funcionamento e desempenho do protótipo concluindo-se que o sistema desenvolvido, juntamente com o aplicativo, atenderam a necessidade do projeto, apresentando-se de forma eficaz na aquisição de dados durantes os experimentos, assim como transmissão em tempo real para o aplicativo do usuário.

Por fim, para dar continuidade à pesquisa e melhorar o projeto apresentado, faz-se a seguir algumas sugestões e considerações para trabalhos futuros.

1. Desenvolver a construção mecânica do protótipo para acompanhar o seu desempenho em campo.
2. Desenvolver o controle de velocidade do motor para que a quantidade de ração seja despejada com maior precisão.
3. Desenvolver o projeto de uma placa de circuito impresso para facilitar a conexão de todos os dispositivos.
4. Utilização de sensor de proximidade para identificar os horários em que o animal

se aproxima do comedouro para se alimentar. Em conjunto do sensor de peso identificar a quantidade de alimento que o animal ingeriu e em qual período do dia.

5. Utilização de um cartão de memória para que as informações não sejam perdidas em uma queda de energia.

6. Implementação de conexão *Bluetooth* para que o acompanhamento e configuração do alimentador possa ser utilizado mesmo em locais sem rede WiFi.

7. Implementação de uma bateria para que o alimentador funcione mesmo quando houver queda de energia.

REFERÊNCIAS

ALBRITTON, R. **What is App Inventor?** 2018. Disponível em: <https://learn.adafruit.com/mit-app-inventor-and-particle-io>. Acesso em: 03 fev. 2022.

ARDUINO. **Software**. 2022. Disponível em: <https://www.arduino.cc/en/software>. Acesso em: 03 fev. 2022.

Arduino Santa Efigenia. **Módulo WiFi ESP32**. 2022. Disponível em: <https://arduinosantaefigenia.com.br/produto/esp32/>. Acesso em: 14 ago. 2022.

ARF PETS. **Automatic Pet Feeder**: Guia do usuário. Londres: C&A Marketing UK LTD, 2018. Disponível em: <https://cdn.shopify.com/s/files/1/2342/0799/files/updated_feeder_user_guide_2018.pdf?11441098706985957028>. Acesso em: 12 nov. 2021.

AUTODESK. **Inventor: software avançado de projeto mecânico para suas ideias mais ambiciosas**. 2022. Disponível em: <https://www.autodesk.com.br/products/inventor/overview?term=1-YEAR&tab=subscription>. Acesso em: 19 mar. 2022.

AUTODESK INSTRUCTABLES. **Arduino Tutorial - SD Card - LCD I2C**. 2022. Disponível em: <https://www.instructables.com/Arduino-Tutorial-SD-Card-LCD-I2C/>. Acesso em: 14 ago. 2022.

BNDES. **Relatório do Plano de Ação - Iniciativas e Projetos Mobilizadores**: In: Internet das coisas: um plano de ação para o brasil. BNDES, 2017. Disponível em: <<https://www.bndes.gov.br/wps/wcm/connect/site/269bc780-8cdb-4b9b-a297-53955103d4c5/relatorio-final-plano-de-acao-produto-8-alterado.pdf?MOD=AJPERES&CVID=m0jDUok>>. Acesso em: 23 jul. 2022.

CASTRO, A. B. AutÔmatos: A mecÂnica como imitaÇÃo da vida. **Anais do VII Seminário Nacional de Pesquisa em Arte e Cultura Visual**, p. 91–101, 2014. ISSN 2316-6479. Disponível em: <https://files.cercomp.ufg.br/weby/up/778/o/2014-eixo1_8_automatos-_a_mecanica_como_imitacao_da_vida.pdf>. Acesso em: 04 jan. 2023.

CASTRO, L. **Entendendo o protocolo I2C**. 2021. Disponível em: <http://blog.arduinoomega.com/entendo-o-protocolo-i2c/>. Acesso em: 14 ago. 2022.

CYRINO, L. **Atuadores de máquinas e equipamentos**. 2018. Disponível em: <https://www.manutencaoemfoco.com.br/atuadores-de-maquinas-e-equipamentos/>. Acesso em: 03 fev. 2022.

DESTAQUE NOTÍCIAS. **Brasileiros têm mais cachorros do que gatos dentro de casa**. 2020. Disponível em: <https://www.destaquenoticias.com.br/brasileiros-tem-mais-cachorros-do-que-gatos-dentro-de-casa/>. Acesso em: 12 nov. 2021.

DORF, R. C.; BISHOP, R. H. **Sistemas de Controle Modernos**. 12. ed. Rio de Janeiro: LTC, 2013.

EDUARDA, M. **Fonte de alimentação: o que é e para que serve?** 2018. Disponível em: <https://blog.iluminim.com.br/fonte-de-alimentacao-o-que-e-e-para-que-serve/>. Acesso em: 16 abr. 2022.

ELETROGATE. **Real Time Clock RTC DS3231**. 2022. Disponível em: <https://www.eletrogate.com/real-time-clock-rtc-ds3231>. Acesso em: 10 jul. 2022.

FARIAS, J. C. **O que é CAD (Desenho Assistido por Computador)?** 2021. Disponível em: <https://spbim.com.br/o-que-e-cad-desenho-assistido-por-computador/>. Acesso em: 19 mar. 2022.

FERREIRA, A. B. de H. **Míni Aurélio: o dicionário da língua portuguesa**. 8. ed. Curitiba: Editora Positivo, 2010.

FERREIRA, A. L. **Como usar o sensor ultrassônico HC-SR04 com Arduino sem o auxílio de biblioteca**. 2022. Disponível em: <http://www.squids.com.br/arduino/index.php/projetos-arduino/projetos-squids/intermediario/350-i15-como-usar-o-sensor-ultrassonico-hc-sr04-para-medir-distancia-sem-auxilio-de-biblioteca-arduino>. Acesso em: 07 set. 2022.

FILIPEFLOP. **O que é Arduíno?** 2015. Disponível em: <https://www.filipeflop.com/blog/o-que-e-arduino/>. Acesso em: 10 dez. 2021.

FIRMINO, M. D. S.; MATEUS, W. D. R. F. **THYMOS PET – Dosador alimentar automático para animais domésticos**. 97 p. Monografia (Bacharelado em Engenharia Eletrica) — Universidade do Sul de Santa Catarina, Tubarão, 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: 16 nov. 2021.

FRADEN, J. **Handbook of Modern Sensors: Physics, Designs, and Applications**. [S.l.]: Springer, 2004.

GUIMARAES, F. **Sensor de distância ultrassônico com o Arduino**. 2018. Disponível em: <http://mundoprojetado.com.br/sensor-de-distancia-ultrassonico-com-o-arduino/>. Acesso em: 12 fev. 2022.

I DO CODE. **Programação em blocos: aprendendo de maneira divertida**. 2022. Disponível em: <https://idocode.com.br/blog/programacao/programacao-em-blocos/#:~:text=Diante%20disso%2C%20a%20programa%C3%A7%C3%A3o%20em,l%C3%B3gica%20de%20uma%20forma%20geral>. Acesso em: 03 ago. 2022.

KERSCHBAUMER, R. **Microcontroladores**. Luzerna: [s.n.], 2018.

LIMA, F. S. de. **A AUTOMAÇÃO E SUA EVOLUÇÃO**. Natal - RN: [s.n.], 2003. Disponível em: <https://dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_16.pdf>. Acesso em: 14 jan. 2023.

LOADSTARSENSORS. **What is a Load Cell?** 2022. Disponível em: <https://www.loadstarsensors.com/what-is-a-load-cell.html>. Acesso em: 28 jun. 2022.

LOCATELLI, C. **Balança com Célula de Carga e HX711**. 2019. Disponível em: <https://www.curtocircuito.com.br/blog/Categoria%20Arduino/balanca-com-celula-de-carga-e-hx711>. Acesso em: 11 dez. 2021.

MADEIRA, D. **Protocolo I2C - Comunicação entre Arduinos**. 2017. Disponível em: <https://portal.vidadesilicio.com.br/i2c-comunicacao-entre-arduinos/>. Acesso em: 14 ago. 2022.

MATTEDE, H. **Motor de corrente contínua, características e aplicações!** [s.d.]. Disponível em: <https://www.mundodaeletrica.com.br/motor-de-corrente-continua-caracteristicas-e-aplicacoes/>. Acesso em: 11 nov. 2021.

MAX PET FOOD. **Quantidade ideal de ração para cães**. 2022. Disponível em: <https://www.maxtotalalimentos.com.br/dica-cao/alimentacao/quantidade-ideal-racao-para-caes/>. Acesso em: 03 nov. 2021.

MCROBERTS, M. **Arduíno básico**. São Paulo: Novatec Editora Ltda, 2011.

MDIG. **Autômato de Maillardet, o menino mecânico que revelou seu criador com letra cursiva perfeita**. 2022. Disponível em: <https://www.mdig.com.br/index.php?itemid=55773>. Acesso em: 20 fev. 2023.

MIT APP INVENTOR. **About Us**. 2022. Disponível em: <https://appinventor.mit.edu/about-us>. Acesso em: 03 fev. 2022.

NEATU, M. **20 Dicas de desenho técnico para arquitetura**. 2018. Disponível em: <https://www.archdaily.com.br/br/889405/20-dicas-de-desenho-tecnico-para-arquitetura>. Acesso em: 19 mar. 2022.

NEOMOTION. **Micromotor DC com caixa de redução**. 2022. Disponível em: <neomotion.com.br/micromotor-dc/micromotor-dc-c-caixa-de-reducao>. Acesso em: 06 ago. 2022.

OASIS Standard. **MQTT Version 5.0**. [s.n.], 2019. Disponível em: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf>. Acesso em: 05 set 2022.

OGATA, K. **Engenharia de controle moderno**. 5. ed. São Paulo: Pearson, 2010.

OGOSHI, R. C. S. et al. Conceitos básicos sobre nutrição e alimentação de cães e gatos. In: **Congresso Estudantil de Medicina Veterinária da UECE**. Fortaleza: Ciência Animal, 2015. v. 25(1), p. 64–75. Disponível em: <http://webcache.googleusercontent.com/search?q=cache:-AFzxcXGHVEJ:www.uece.br/cienciaanimal/dmdocuments/palestra06_p64_75.pdf+&cd=1&hl=pt-BR&ct=clnk&gl=br&client=firefox-b-d>. Acesso em: 7 fev. 2022.

OLIVEIRA, E. **Como usar com Arduino – Sensor Capacitivo NPN de Proximidade LJC18A3-H-Z/BX**. 2019. Disponível em: <https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-capacitivo-npn-de-proximidade-ljc18a3-h-z-bx/>. Acesso em: 12 fev. 2022.

OMEGA. **Introduction to strain gages**. 2022. Disponível em: <https://www.omega.com/prodinfo/straingages.html>. Acesso em: 28 jun. 2022.

PANIN, L. K. **SENSORES INDUTIVOS – CARACTERÍSTICAS E APLICAÇÕES**. [s.d.]. Disponível em: <http://www.tecniar.com.br/noticias/sensores-indutivos-caracteristicas-e-aplicacoes/>. Acesso em: 12 fev. 2022.

PASTORI, E. O.; MATOS, L. G. Da paixão à "ajuda animalitária": o paradoxo do "amor incondicional" no cuidado e no abandono de animais de estimação. **Caderno Eletrônico de Ciências Sociais**, Vitória, v. 3, p. 112–132, 09 2015.

QUINTINO, E. de C. **O que é IDE Arduino?** 2021. Disponível em: <https://www.filipeflop.com/blog/o-que-e-ide-arduino/>. Acesso em: 03 fev. 2022.

RIBEIRO, M. A. Fundamentos da automação. Salvador, 2003.

SARAVATI. **Micro Motor DC com Caixa de Redução 12V 83 RPM AK555 / 11.1PF12R83CE**. 2022. Disponível em: <https://www.saravati.com.br/motor-dc-com-caixa-de-reducao-12v-83-rpm>. Acesso em: 11 nov. 2022.

SILVEIRA, C. B. **Sensores Ópticos: Como Funcionam?** 2018. Disponível em: <https://www.citisystems.com.br/sensores-opticos/>. Acesso em: 12 fev. 2022.

____. **Sensor Ultrassônico: 10 Aplicações Para a Indústria**. 2020. Disponível em: <https://www.citisystems.com.br/sensor-ultrassonico/>. Acesso em: 12 fev. 2022.

SYSTEMS, E. **ESP32 Documentation**. 2022. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 14 ago. 2022.

TANDER PET. **Comedouro Eletrônico Automático**: Manual de instruções. [s.n.], [s.d.]. Disponível em: <https://www.tanderequipamentos.com.br/upload/download/2676.pdf>. Acesso em: 12 nov. 2021.

THOMSEN, A. **Relógio com o módulo RTC DS1307**. 2014. Disponível em: <https://www.filipeflop.com/blog/relogio-rtc-ds1307-arduino/>. Acesso em: 10 dez. 2022.

TRIXIE. **Bedienungsanleitung TX9**: Manual de instruções. [s.n.], [s.d.]. Disponível em: <https://www.trixie.de/module/mysydeshop/webforms/download_file.php?file=247523>. Acesso em: 12 nov. 2021.

USINAINFO. **Sensor de Peso Célula de Carga 0 a 5kg para Arduino**. 2022. Disponível em: <https://www.usinainfo.com.br/sensor-de-peso-arduino/sensor-de-peso-celula-de-carga-0-a-5kg-para-arduino-4413.html?search_query=sensor+de+peso&results=575>.

VITALE, K. R.; BEHNKE, A. C.; UDEL, M. A. Attachment bonds between domestic cats and humans. **Current Biology**, v. 29, p. R864 – R865, 2019. Disponível em: <https://www.cell.com/action/showPdf?pii=S0960-9822%2819%2931086-3>. Acesso em: 26 mar. 2023.

WELLS, K. **No, You Shouldn't Just Get an Automatic Pet Feeder and Skip Town**. 2019. Disponível em: <https://www.nytimes.com/wirecutter/blog/pros-cons-automatic-feeder/>. Acesso em: 26 mar. 2023.

WENDLING, M. **Sensores**. Guaratinguetá: [s.n.], 2010.

APÊNDICE A

Listing A.1: Alimentador.ino

```
#include <WiFi.h> // Configura  o de conex o com WiFi
#include <WiFiManager.h> //App para conectar o ESP a rede WiFi
#include <PubSubClient.h> //biblioteca MQTT de publisher/subscribe
#include "RTClib.h" //Carrega a biblioteca do RTC
#include <Ultrasonic.h> //Carrega a biblioteca do sensor ultrassonico
#include "HX711.h" //Carrega a biblioteca do amplificador para celula de carga


#define pino_de_conexao 26 // Pino que acionar  a conex o wifi
int timeout = 120; // Segundos para finalizar tentativa de conex o
char* ssid;
char* password;

const char* mqtt_server = "broker.hivemq.com"; //broker mqtt
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (30)
char msg[MSG_BUFFER_SIZE];

RTC_DS3231 rtc; //OBJETO DO TIPO RTC_DS3231
```

```
22
23 // Amplificador HX711
24 HX711 escala; //OBJETO variavel escala
25 #define DT 19
26 #define SCK 18
27
28 #define RelePin 25 // pino ao qual o M dulo Rel  est
      conectado
29
30 //Define os pinos para o trigger e echo do sensor
      ultrassonico
31 #define pino_trigger 15
32 #define pino_echo 2
33 //Inicializa o sensor nos pinos definidos acima
34 Ultrasonic ultrasonic(pino_trigger, pino_echo);
35
36 String alarme1, hora1, peso1;
37 String alarme2, hora2, peso2;
38 String alarme3, hora3, peso3;
39 String alarme4, hora4, peso4;
40 String alarme5, hora5, peso5;
41
42 /* Setup de conex o WiFi */
43 void setup_wifi(){
44   WiFiManager wm;
45
46   wm.setConfigPortalTimeout(timeout);
47
48   if (!wm.startConfigPortal("AlimentadorWiFi")) {
49     Serial.println("A conex o falhou ou atingiu o limite
         de tempo");
50     delay(3000);
51     ESP.restart();
52     delay(5000);
53   }
```

```
54
55    Serial.println("ESP32␣conectado␣  ␣internet");
56
57    char Ssid[WiFi.SSID().length() + 1];
58    char Password[WiFi.psk().length() + 1];
59
60    ssid = strcpy(Ssid, WiFi.SSID().c_str());
61    password = strcpy(Password, WiFi.psk().c_str());
62  }
63
64  void callback(char* topic, byte* payload, unsigned int
      length){
65    String topico = topic;
66    String mensagem;
67    for (int i = 0; i < length; i++) {
68      char c = (char)payload[i];
69      mensagem += c;
70    }
71
72    String topMotor = "alimentador/motor";
73
74    String topAlarme1 = "alimentador/alarme1";
75    String topHora1 = "alimentador/hora1";
76    String topPeso1 = "alimentador/peso1";
77
78    String topAlarme2 = "alimentador/alarme2";
79    String topHora2 = "alimentador/hora2";
80    String topPeso2 = "alimentador/peso2";
81
82    String topAlarme3 = "alimentador/alarme3";
83    String topHora3 = "alimentador/hora3";
84    String topPeso3 = "alimentador/peso3";
85
86    String topAlarme4 = "alimentador/alarme4";
87    String topHora4 = "alimentador/hora4";
```

```
88  String topPeso4 = "alimentador/peso4";
89
90  String topAlarme5 = "alimentador/alarme5";
91  String topHora5 = "alimentador/hora5";
92  String topPeso5 = "alimentador/peso5";
93
94  if(topico == topMotor){
95    if(mensagem == "ligar"){
96      digitalWrite(RelePin, HIGH); //Aciona o rel  para
          ligar o motor
97      Serial.print("motor ligado");
98    }else{
99      digitalWrite(RelePin, LOW);
100     Serial.print("motor desligado");
101   }
102 }
103
104 if(topico == topAlarme1){alarme1 = mensagem;}
105 if(topico == topHora1){hora1 = mensagem;}
106 if(topico == topPeso1){peso1 = mensagem;}
107
108 if(topico == topAlarme2){alarme2 = mensagem;}
109 if(topico == topHora2){hora2 = mensagem;}
110 if(topico == topPeso2){peso2 = mensagem;}
111
112 if(topico == topAlarme3){alarme3 = mensagem;}
113 if(topico == topHora3){hora3 = mensagem;}
114 if(topico == topPeso3){peso3 = mensagem;}
115
116 if(topico == topAlarme4){alarme4 = mensagem;}
117 if(topico == topHora4){hora4 = mensagem;}
118 if(topico == topPeso4){peso4 = mensagem;}
119
120 if(topico == topAlarme5){alarme5 = mensagem;}
121 if(topico == topHora5){hora5 = mensagem;}
```

```
122   if(topico == topPeso5){peso5 = mensagem;}
123
124 }
125
126 void reconnect(){
127   if (WiFi.status() != WL_CONNECTED) {
128     WiFi.begin(ssid, password);
129
130     while (WiFi.status() != WL_CONNECTED) {
131         delay(1000);
132         Serial.print(".");
133     }
134
135     randomSeed(micros());
136   }
137   while(!client.connected() ){
138     Serial.print("Tentando conex o MQTT...");
139     //Cria um cliente ID
140     String clientId = "ALIMENTADOR_MQTT";
141     clientId += String(random(0xffff), HEX);
142     //Tentativa de conex o
143     if (client.connect(clientId.c_str())){
144       Serial.print("Conectado");
145       client.subscribe("alimentador/motor");
146
147       client.subscribe("alimentador/alarme1");
148       client.subscribe("alimentador/hora1");
149       client.subscribe("alimentador/peso1");
150
151       client.subscribe("alimentador/alarme2");
152       client.subscribe("alimentador/hora2");
153       client.subscribe("alimentador/peso2");
154
155       client.subscribe("alimentador/alarme3");
156       client.subscribe("alimentador/hora3");
```

```
157         client.subscribe("alimentador/peso3");
158
159         client.subscribe("alimentador/alarme4");
160         client.subscribe("alimentador/hora4");
161         client.subscribe("alimentador/peso4");
162
163         client.subscribe("alimentador/alarme5");
164         client.subscribe("alimentador/hora5");
165         client.subscribe("alimentador/peso5");
166       }else{
167         Serial.print("failed, rc=");
168         Serial.print(client.state());
169         Serial.println(" try again in 5 secconds");
170         delay(5000);
171       }
172     }
173 }
174
175 String hora_reservatorio(){
176   String hora;
177   DateTime now = rtc.now(); //CHAMADA DE FUN  O
178   char tempo[6];
179   sprintf( tempo, "%02hhu:%02hhu", now.hour(), now.minute
        () );
180   for (int i = 0; i < 5; i++) {
181     char c = (char)tempo[i];
182     hora += c;
183   }
184   sprintf(msg, "%s", tempo);
185   client.publish("alimentador/hora", msg);
186
187   return hora;
188 }
189
190 // Fun  o para calcular o n vel no reservat rio
```

```
191 int nivel_reserva(){
192   int profundidadeReservatorio = 50;
193   int nivel;
194   float distanciaMedidaSensor;
195   long microsec = ultrasonic.timing();
196   distanciaMedidaSensor = ultrasonic.convert(microsec,
        Ultrasonic::CM);
197   nivel = ((profundidadeReservatorio -
        distanciaMedidaSensor)/profundidadeReservatorio)
        *100;
198   sprintf(msg, "%d", nivel);
199   client.publish("alimentador/nivel", msg);
200
201   return nivel;
202 }
203
204 // Fun   o para calcular o peso no comedouro
205 int peso_Comedouro(){
206   int pesoComedouro;
207   pesoComedouro = escala.get_units()*1000;
208   sprintf(msg, "%d", pesoComedouro);
209   client.publish("alimentador/peso", msg);
210   return pesoComedouro;
211 }
212
213 // Fun   o para ligar o motor
214 void ligar_Motor(int pesoAlarme){
215   int pesoDesejado = pesoAlarme;
216   int pesoComedouro = peso_Comedouro();
217
218   digitalWrite(RelePin, HIGH);
219   while(pesoComedouro < pesoDesejado){
220     digitalWrite(RelePin, HIGH);
221     pesoComedouro = peso_Comedouro();
222   }
```

```
223   digitalWrite(RelePin, LOW);
224 }
225
226 void setup() {
227   Serial.begin(115200);
228   WiFi.mode(WIFI_STA); // explicitly set mode, esp
        defaults to STA+AP
229   Serial.println("\n Iniciando");
230   client.setServer(mqtt_server, 1883);
231   client.setCallback(callback);
232
233   pinMode(pino_de_conexao, INPUT_PULLUP); // Bot o para
        conex o
234   pinMode(RelePin, OUTPUT);
235
236   if (! rtc.begin()) {
237     Serial.println("N  o foi poss vel encontrar RTC");
238     while (1);
239   }
240   if(rtc.lostPower()){ //SE RTC FOI LIGADO PELA PRIMEIRA
        VEZ / FICOU SEM ENERGIA / ESGOTOU A BATERIA, FAZ
241     Serial.println("DS3231 OK!"); //IMPRIME O TEXTO NO
          MONITOR SERIAL
242     //REMOVA O COMENT RIO DE UMA DAS LINHAS ABAIXO PARA
          INSERIR AS INFORMA  ES ATUALIZADAS EM SEU RTC
243     //rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); //
          CAPTURA A DATA E HORA EM QUE O SKETCH    COMPILADO
244     //rtc.adjust(DateTime(2022, 9, 22, 20, 30, 00)); //(
          ANO), (M  S), (DIA), (HORA), (MINUTOS), (SEGUNDOS)
245   }
246
247   escala.begin (DT, SCK);
248   escala.set_scale(820910); // Substituir o valor
        encontrado para escala
249   escala.tare(); // O peso e  chamado de Tare.
```

```
250
251     alarme1 = "Desligado";
252     hora1 = "00:00";
253     peso1 = "0";
254     alarme2 = "Desligado";
255     hora2 = "00:00";
256     peso2 = "0";
257     alarme3 = "Desligado";
258     hora3 = "00:00";
259     peso3 = "0";
260     alarme4 = "Desligado";
261     hora4 = "00:00";
262     peso4 = "0";
263     alarme5 = "Desligado";
264     hora5 = "00:00";
265     peso5 = "0";
266 }
267
268 void loop() {
269   // Configura  o de conex o de internet
270   if ( digitalRead(pino_de_conexao) == LOW) {
271     setup_wifi();
272   }
273
274   // Envia mensagem para avisar que est  conectado
275   if(WiFi.status() == WL_CONNECTED) {
276     client.publish("alimentador/conexao", "Alimentador␣
        conectado");
277     if(!client.connected()){reconnect();}
278   }
279
280
281   String hora = hora_reservatorio(); //Hor rio
       reserva rio
282   int nivel = nivel_reserva(); //Nivel do reservat rio
```

```
283   int peso = peso_Comedouro(); //Peso de  ra   o no
          comedouro
284
285
286   //Alarme 1
287   String switch1Arduino = alarme1;
288   sprintf(msg, "%s", alarme1);
289   client.publish("alimentador/switch1Arduino", msg);
290   String hora1Arduino = hora1;
291   sprintf(msg, "%s", hora1Arduino);
292   client.publish("alimentador/hora1Arduino", msg);
293   String peso1Arduino = peso1;
294   int intPeso1Arduino = atoi( peso1Arduino.c_str() );
295   sprintf(msg, "%s", peso1Arduino);
296   client.publish("alimentador/peso1Arduino", msg);
297
298   //Alarme 2
299   String switch2Arduino = alarme2;
300   sprintf(msg, "%s", alarme2);
301   client.publish("alimentador/switch2Arduino", msg);
302   String hora2Arduino = hora2;
303   sprintf(msg, "%s", hora2Arduino);
304   client.publish("alimentador/hora2Arduino", msg);
305   String peso2Arduino = peso2;
306   int intPeso2Arduino = atoi( peso2Arduino.c_str() );
307   sprintf(msg, "%s", peso2Arduino);
308   client.publish("alimentador/peso2Arduino", msg);
309
310
311   //Alarme 3
312   String switch3Arduino = alarme3;
313   sprintf(msg, "%s", alarme3);
314   client.publish("alimentador/switch3Arduino", msg);
315   String hora3Arduino = hora3;
316   sprintf(msg, "%s", hora3Arduino);
```

```
317   client.publish("alimentador/hora3Arduino", msg);
318   String peso3Arduino = peso3;
319   int intPeso3Arduino = atoi( peso3Arduino.c_str() );
320   sprintf(msg, "%s", peso3Arduino);
321   client.publish("alimentador/peso3Arduino", msg);
322
323   //Alarme 4
324   String switch4Arduino = alarme4;
325   sprintf(msg, "%s", alarme4);
326   client.publish("alimentador/switch4Arduino", msg);
327   String hora4Arduino = hora4;
328   sprintf(msg, "%s", hora4Arduino);
329   client.publish("alimentador/hora4Arduino", msg);
330   String peso4Arduino = peso4;
331   int intPeso4Arduino = atoi( peso4Arduino.c_str() );
332   sprintf(msg, "%s", peso4Arduino);
333   client.publish("alimentador/peso4Arduino", msg);
334
335   //Alarme 5
336   String switch5Arduino = alarme5;
337   sprintf(msg, "%s", alarme5);
338   client.publish("alimentador/switch5Arduino", msg);
339   String hora5Arduino = hora5;
340   sprintf(msg, "%s", hora5Arduino);
341   client.publish("alimentador/hora5Arduino", msg);
342   String peso5Arduino = peso5;
343   int intPeso5Arduino = atoi( peso5Arduino.c_str() );
344   sprintf(msg, "%s", peso5Arduino);
345   client.publish("alimentador/peso5Arduino", msg);
346
347   if((alarme1 == "Ligado") && (hora == hora1)){
348     ligar_Motor(intPeso1Arduino);
349   }
350   if((alarme2 == "Ligado") && (hora == hora2)){
351     ligar_Motor(intPeso2Arduino);
```

```
352   }
353   if((alarme3 == "Ligado") && (hora == hora3)){
354     ligar_Motor(intPeso3Arduino);
355   }
356   if((alarme4 == "Ligado") && (hora == hora4)){
357     ligar_Motor(intPeso4Arduino);
358   }
359   if((alarme5 == "Ligado") && (hora == hora5)){
360     ligar_Motor(intPeso5Arduino);
361   }
362
363   delay(1000);
364   client.loop();
365 }
```

APÊNDICE B

A tabela contendo os pinos utilizados e sua função no ESP32 pode ser vista em 6.

Tabela 6: Pinagem utilizada no ESP32

PINO	GPIO	Função
3,3V		
GND		
D15	15	Trigger módulo ultrassônico
D2	2	Echo módulo ultrassônico
D4	4	
RX2	16	
TX2	17	
D5	5	
D18	18	SCK do HX711
D19	19	DT do HX711
D21	21	SDA do RTC
RX0	3	
TX0	1	
D22	22	SCL do RTC
D23	23	
VIN		
GND		
D13	13	
D12	12	
D14	14	
D27	27	
D26	26	leitura do botão de conexão
D25	25	Acionar o relé do motor
D33	33	
D32	32	
D35	35	
D34	34	
VN	39	
VP	36	
EN		

Fonte: Autor