UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MÔNICA MARIA MENIN

ALIMENTADOR AUTOMÁTICO PARA GATOS CONTROLADO POR APLICATIVO ANDROID

PATO BRANCO

2022

MÔNICA MARIA MENIN

ALIMENTADOR AUTOMÁTICO PARA GATOS CONTROLADO POR APLICATIVO ANDROID

Automatic cat feeder controlled by Android app

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Robison Cris Brito

PATO BRANCO

2022

MÔNICA MARIA MENIN

ALIMENTADOR AUTOMÁTICO PARA GATOS CONTROLADO POR APLICATIVO ANDROID

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 08/dezembro/2022

Robison Cris Brito
Professor Doutor
Universidade Tecnológica Federal do Paraná

Fábio Favarim
Professor Doutor
Universidade Tecnológica Federal do Paraná

João Guilherme Brasil Pichetti
Professor Especialista
Universidade Tecnológica Federal do Paraná

PATO BRANCO
2022

Dedico este trabalho ao meu irmão Alexandre, carinhosamente chamado de Bibi, *in memoriam*, que pelo seu amor e preocupação aos animais deu a ideia deste projeto.

AGRADECIMENTOS

Agradeço a Deus pela a oportunidade de estar nessa vida, por minha família e por meus amigos que fizeram parte dessa importante fase de minha vida.

Agradeço ao meu orientador Prof. Dr. Robison Cris Brito, pela paciência, compreensão nos momentos difíceis e pela sabedoria com que me guiou nesta trajetória.

Gostaria de deixar registrado, o meu agradecimento especial aos meus pais, Alexandre e Maria Elita, pois acredito que sem o apoio, amor, compreensão e paciência deles seria muito difícil vencer mais esse desafio da minha vida. Agradeço a minha irmã, Alice que esteve comigo dia a dia nessa fase, ouvindo minhas reclamações e que nunca me deixou desistir, ao meu irmão, Alexandre, que não está mais fisicamente nesse mundo, por amor e preocupação com animais foi quem deu a ideia principal desse projeto.

A todos os meus amigos que tiveram paciência comigo quando eu só falava sobre este projeto, que me apoiaram e me deram forças pra continuar a realização desta pesquisa.

"Nós seres humanos, estamos na natureza para auxiliar o progresso dos animais, na mesma proporção que os anjos estão para nos auxiliar. Portanto quem chuta ou maltrata um animal é alguém que não aprendeu a amar."(Chico Xavier)

RESUMO

Desde os primórdios da humanidade, os humanos e os animais interagiam em grupos para ter proteção e dividir alimentos, muitas vezes escassos e difíceis de conseguir. Com o passar dos anos alguns animais, como cães e gatos, se tornaram mais próximos às pessoas, pois conseguiam comida e proteção sem muito esforço e foram domesticados, se tornando companheiros de morada e de vida. Apesar da proximidade entre eles, há muitas diferenças na alimentação e no funcionamento do seu organismo, os humanos fazem várias refeições durante o dia, tendo 3 como principal em maior quantidade de alimento, os cães fazem de 2 a 3 refeições ao dia, já os gatos fazem até 12 pequenas refeições durante 24horas (h). A carga horária de trabalho e a distância de casa fazem com que muitas pessoas não voltem a suas residências várias vezes ao dia, dificultando assim a possibilidade de controlar e oferecer a quantidade de alimento correto ao gato. Com o constante desenvolvimento das tecnologias, é possível fazer uma interligação entre os *smartphones*, os microcontroladores, e os motores de alta precisão. A fim de criar um sistema de fácil conectividade à Internet e enviar mensagens através de protocolos de comunicação, para fazer o acionamento a distância de um alimentador e proporcionar a quantidade correta de alimento a um gato, sendo disponibilizada em até 12 pequenas porções de forma automática e pré-programada ou segundo acionamento remoto feito pelo tutor, visando assim evitar o consumo excessivo de alimento pelo gato, otimizar sua saúde e o tempo de seu tutor. Ao final deste trabalho foi desenvolvido um alimentador automático para gatos controlado por aplicativo para dispositivo móvel com sistema operacional Android, que permite acionamento remoto e agendamento da alimentação, enviando e recebendo mensagens usando o protocolo *MQTT*, e o microcontrolador *ESP32* conectado a rede *Wi-Fi* recebe as mensagens, interpreta-as e controla um servomotor para dosar e distribuir o alimento ao gato, este projeto conta com uma bateria reserva em casos de falta de energia. Os testes de envio e recebimento de mensagens, assim como o acionamento foram realizados e obtendo-se resultados apropriados.

Palavras-chave: alimentador para gatos; protocolo mqtt; esp32; dispositivos moveis; iot.

ABSTRACT

Since the dawn of humanity, humans and animals interacted in groups to protect themselves and share food, which was often scarce and difficult to obtain. Over the years some animals, such as dogs and cats, became closer to people, as they managed to get food and protection without much effort and were domesticated, becoming home and life companions. Despite the proximity between them, there are many differences in food and in the functioning of their organism, humans have several meals during the day, with 3 as the main one in greater quantity of food, dogs have 2 to 3 meals a day, whereas dogs cats eat up to 12 small meals during 24h. The workload and the distance from home mean that many people do not return to their homes several times a day, thus making it difficult to control and offer the correct amount of food to the cat. With the constant development of technologies, it is possible to make an interconnection between *smartphones*, microcontrollers, and high precision motors. In order to create a system of easy connectivity to the Internet and send messages through communication protocols, to activate a feeder remotely and provide the correct amount of food to a cat, being available in up to 12 small portions automatically and pre-programmed or second remotely triggered by the tutor, thus aiming to avoid excessive consumption of food by the cat, optimizing its health and the time of its tutor. At the end of this work, an automatic feeder for cats controlled by an application for a mobile device with Android operating system was developed, which allows remote activation and scheduling of feeding, sending and receiving messages using the *MQTT* protocol, and the microcontroller *ESP32* connected to the network *Wi-Fi* receives the messages, interprets them and controls a servomotor to dose and distribute the food to the cat, this project has a backup battery in cases of power failure. Tests for sending and receiving messages, as well as activating them, were carried out and the appropriate results were obtained.

Keywords: cat feeder; mqtt protocol; esp32; mobile devices; iot.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

A	Ampers
ABINPET	Associação Brasileira da Indústria de Produtos para Animais de Estimação
APOP	Association for Pet Obesity Prevention
cm	centímetros
CoT	Nuvem das coisas, do inglês *Cloud of Things*
CPU	Unidade Central de Processamento, do inglês *Central Processing Unit*
DC	Corrente direta ou Corrente Contínua, do inglês *Direct Current*
ENVA	Escola Nacional de Medicina Veterinária d'Alfort
g	gramas
h	horas
I2C	Circuito Interintegrado, do inglês *Inter-Integrated Circuit*
IBM	*International Business Machines Corporation*
IDE	Ambiente de desenvolvimento integrado, do inglês *Integrated Development Environment*
IoT	Internet das coisas, do inglês *Internet of Things*
IoT-A	Arquitetura IoT
kg	kilogramas
mAh	miliampers por hora
min	minutos
ml	mililitros
mm	milimetros
MQTT	Transporte de telemetria de enfileiramento de mensagens, do inglês *Message Queuring Telemetry Transport*

NEM	Necessidade energética de manutenção
PDU	Unidade de dados de protocolo, do inglês *Protocol Data Unit*
QoS	Qualidade de serviço ,do inglês *(Quality of Service)*
s	segundos
SPI	Interface Periférica Serial, do inglês *Serial Peripheral Interface*
TCP	Protocolo de controle de transmissão, do inglês *Transmission Control Protocol*
TIC	Tecnologias de informação e comunicação
UART	Porta universal de recebimento/transmissão assíncrona, do inglês *Universal Asynchronous Receiver/Transmitter*
USB	Porta universal serial, do inglês *Universal Serial Bus*
V	Volts
XML	Linguagem de marcação com regras, do inglês *Extensible Markup Linguage*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo está dividido em considerações iniciais, objetivos e justificativa.

1.1 Considerações iniciais

Ao longo dos anos os humanos procuram se aproximar de outras espécies de animais, como cães e gatos, tornando-os companheiros de morada e de vida.

> A ausência de um neocórtex não parece impedir que um organismo experimente estados afetivos. Evidências convergentes indicam que animais não humanos tem os substratos neuro anatômicos, neuroquímicos e neurofisiológicos de estados de consciência juntamente como a capacidade de exibir comportamentos intencionais (LOW, 2012).

Segundo a afirmação de (LOW, 2012), na declaração de *Cambridge*, os animais têm consciência e são suscetíveis a estados de ansiedade, depressão, medo, angústia e entre outros sentimentos que possam alterar o seu comportamento.

A Associação Brasileira da Indústria de Produtos para Animais de Estimação (ABINPET), estima que sejam 54,2 milhões de cães, 23,9 milhões de gatos no Brasil.

Para Grandjean (2006), professor e pesquisador de nutrição animal da unidade de Medicina de reprodução e Esporte da Escola Nacional de Medicina Veterinária d'Alfort (ENVA) em Paris, a proximidade entre os humanos e seus animais de estimação, traz a tendência para o antropomorfismo, a falsa impressão de conhecimento sobre o funcionamento do organismo dos mesmos, impondo-lhes o estilo de vida humana e deixando de lado suas diferenças e necessidades específicas. Já se passou o tempo em que os cães eram alimentados com os restos das refeições e os gatos bebiam leite, que não supre as suas necessidades nutricionais. Um erro comum é não estabelecer um limite na alimentação do gato ou oferecer uma quantidade diária muito maior ou menor que a recomendada. Visto que alguns não possuem autocontrole, caso o limite não seja estabelecido, os animais continuarão ingerindo alimento enquanto este estiver disponível e isso ocasionará sérios problemas para sua saúde.

A relação do brasileiro com a Internet mudou significativamente nos últimos anos, com a diversidade de formas de acesso a ela, o computador já não é a forma de acesso preferencial, dando espaço aos dispositivos móveis, como *smartphones* e *tablets*. A parcela da população, no Brasil, conectada à Internet saltou de 41% em 2010 para 70% em 2018 (MOURA; CAMARGO, 2020). O *smartphone* é o meio mais usado para se conectar à Internet, segundo dados da empresa americana de consultoria de gestão global, *Bain & Company*, e em 2020, 97% dos usuários no Brasil acessam a rede através de um *smartphone* e 51% a acessam exclusivamente dessa forma.

Nos últimos 5 anos, estima-se que 24 milhões de brasileiros tenham sido introduzidos à Internet por um dispositivo móvel com sistema operacional Android. Por ser um sistema operacional aberto, gratuito, dinâmico e acessível, o Android tem se mostrado altamente atrativo para a comunidade de desenvolvedores e empresas de *software*, sendo o motivo que 83% dos desenvolvedores escolhem esse sistema (MOURA; CAMARGO, 2020).

Com o avanço das tecnologias, o uso das Internet das coisas, do inglês *Internet of Things* (IoT), traz várias possibilidades de comunicação entre objetos que usamos diariamente com a Internet, tendo funções de acionamento ou obtenção de dados de uso. Uma palavra amplamente utilizada no desenvolvimento tecnológico, é a automação, que usa dispositivos mecânicos ou eletrônicos para facilitar o trabalho humano e os cuidados com os animais. Um dos componentes principais da automação é o microcontrolador, responsável por fazer a integração do *hardware* com o *software*, realizando tarefas previamente programadas e específicas para sua utilização final. Por existirem muitas categorias de microcontroladores, os seus valores variam conforme a sua capacidade de funcionamento e os diferentes componentes de conexão, os usados para automações residenciais e de pequeno porte geralmente são mais baratos e possuem maior facilidade de programação, contendo elementos de conexão *Bluetooth* e *Wi-Fi*. Utilizando das diversas formas de comunicação entre *software* e *hardware*, a mais comum é a integração via Internet, por não limitar a distância entre as partes, essas conexões podem ser feitas através da *Wi-Fi* e implementadas por protocolos destinados à comunicação de dados.

Visando otimizar o tempo dos donos e proporcionar ao gato a alimentação correta, pode-se usufruir destas vantagens para desenvolver um dispositivo alimentador Automático acionado à distância, que consiste em um dispositivo móvel com sistema operacional Android conectado à Internet, que envia instruções pelo protocolo Transporte de telemetria de enfileiramento de mensagens, do inglês *Message Queuring Telemetry Transport* (MQTT) a um *Broker*, o intermediário que localiza e redireciona as instruções ao microcontrolador através do mesmo protocolo, para acionar um servomotor que permite movimentar um dosador para disponibilizar a ração ao gato.

1.2 Objetivos

Esta seção está dividida em objetivo geral e objetivos específicos.

1.2.1 Objetivo geral

Criar um alimentador Automático, controlado remotamente por aplicativo para dispositivo móvel com sistema operacional Android para a liberação automática de alimento para um gato doméstico.

1.2.2 Objetivos específicos

- Regrar a alimentação do gato;
- Otimizar o tempo do tutor;

1.3 Justificativa

A obesidade e as doenças relacionadas a ela, vem aumentando rapidamente tanto em animais domésticos como em humanos, a incidência da obesidade em gatos adultos era de cerca de 35%, (LUND *et al.*, 2005) e 34% de cachorros adultos, (LUND *et al.*, 2006), nos anos de 2005 e 2006. Um levantamento da Association for Pet Obesity Prevention (APOP), em 2018, diz que 59,5% dos gatos e 55,8% dos cães foram classificados como com sobrepeso ou obesos. Entre as doenças relacionadas a obesidade estão a diabetes *mellitus*, doenças cardiovasculares, hipertensão, dislipidemia, pedras vesiculares e alguns tipos de câncer (SAMARTIN; CHANDRA, 2001). Embora algumas doenças, alguns medicamentos e raros defeitos genéticos possam ser a causa da obesidade, a principal razão é a incompatibilidade entre ingestão de alimentos e o gasto energético, portanto, a alimentação em horários e quantidades adequadas é uma das alternativas para evitar a obesidade animal. A integração entre as tecnologias e o cuidado com os animais visa facilitar o manejo e saúde dos animais, otimizando tempo dos tutores.

1.4 Estrutura do trabalho

A estrutura do trabalho consiste em uma relação do trata cada capítulo.

O primeiro capítulo apresenta uma introdução a alimentação dos gatos, o uso de *smartphones* e suas tecnologias, objetivo geral, objetivo específico e a justificativa.

O capítulo dois compõem a revisão de literatura sobre a alimentação de gatos, os alimentadores disponíveis no mercado e o funcionamento do protocolo MQTT.

O capítulo três apresenta a visão geral do projeto, os materiais a serem utilizados e os métodos para a implementação.

O capítulo quatro apresenta os resultados e discussão do projeto.

O capítulo cinco apresenta as conclusões e perspectivas do trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos básicos da alimentação dos gatos, assim como alguns alimentadores para animais disponíveis no mercado. Ao final, informações sobre o sistema operacional Android e o protocolo de comunicação MQTT.

2.1 Alimentação dos gatos

Os gatos têm necessidades nutricionais diferentes dos cães e dos humanos, o consumo de proteínas pelos gatos é grande se comparada com os demais animais domésticos.

> Os felinos são carnívoros e dependem de fontes proteicas para produção de energia e síntese de taurina. Este último elemento é um aminoácido essencial para a vida do gato, sendo fundamental para a visão, reprodução e funções cardíacas (MAXTOTAL, 2021).

Conforme a fase de vida do animal, as suas necessidades nutricionais podem variar, os filhotes precisam de mais energia, pois estão em crescimento, os gatos adultos e castrados precisam de cuidados especiais com a ração que evite o ganho excessivo de peso. O gato, um animal caçador individual, faz cerca de doze pequenas refeições durante o dia e a noite, não podendo ultrapassar um limite de quantidade em gramas de ração ingeridas ao dia, para que não desenvolva a obesidade.

Conforme a produtora de alimentos para animais Royal (2021), um elemento essencial a vida é a energia, e o cálculo da necessidade energética de um animal é o ponto de partida para a formulação e balanceamento do alimento. A energia das dietas para gatos e cães pode ser originária de três macronutrientes, os carboidratos, os lipídeos e as proteínas. O requerimento energético difere para cada etapa de vida do animal, calculado com base no peso metabólico de cada um. Pode-se calcular o peso metabólico dos gatos pela Equação 1 e para os cães pela Equação 2:

$$PM_{Gato} = (P_{Vivo})^{(0,67)} \tag{1}$$

$$PM_{Cao} = (P_{Vivo})^{(0,75)} \tag{2}$$

A Necessidade energética de manutenção (NEM), é o cálculo do valor diário de calorias para animais adultos estimada como:

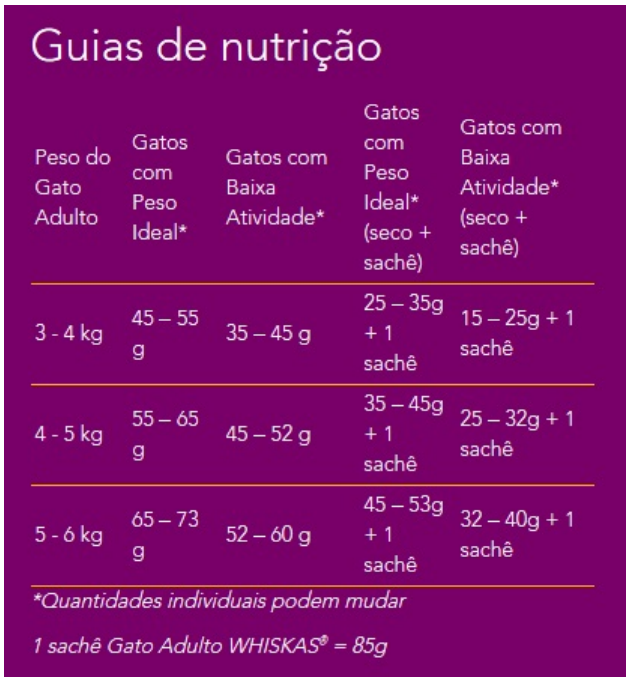
- Cães: 90 a 130 kcal / Peso Metabólico
- Gatos: 100 kcal / Peso Metabólico

Os filhotes em crescimento requerem cerca de 2 vezes a *NEM*, visto que estão em período de crescimento e desenvolvimento de ossos e musculatura (COUNCIL, 2006).

A castração ou esterilização tem um impacto significativo no peso do gato, suas necessidades de energia caem 30%, mas seu apetite pode aumentar em até 26% nos dias seguintes à operação. Portanto, gatos castrados e esterilizados correm um risco maior de engordar e tornarem-se obesos (ROYAL, 2021).

A Nutrição Animal é uma área em amplo desenvolvimento e existem várias categorias de rações, como secas ou úmidas, para as mais diversas situações e condições. Conforme a fabricante de ração SpecialCat (2021), a composição básica da ração é farinha de vísceras de frango, plasma suíno em pó, arroz quebrado, milho integral moído, farelo de glúten de milho 60*, soja micronizada, gordura de frango, óleo de peixe, farelo de arroz, farelo de arroz desengordurado, celulose, levedura seca de cana-de-açúcar, extrato de Yucca schidigera (0,05%), mananoligossacarídeos (0,048%), hidrolisado de fígado de frango e suíno, cloreto de sódio (sal comum), cloreto de potássio, L-lisina, taurina, L-carnitina, vitamina A, vitamina D3, vitamina E, vitamina K3, vitamina B1, vitamina B2, vitamina B6, vitamina B12, vitamina C, niacina, cloreto de colina, ácido pantotênico, ácido fólico, biotina, iodato de cálcio, sulfato de cobre, sulfato de ferro, levedura enriquecida com selênio, proteinato de manganês, proteinato de zinco, propionato de amônia, sorbato de potássio, parede celular de levedura, acidificante, antioxidantes (BHA e BHT), corante natural caramelo.

Dada a idade e peso do gato, a porção diária de ração seca é definida por um veterinário ou na embalagem do alimento escolhido existe a recomendação de ingestão diária Figura 1.



Guias de nutrição

Peso do Gato Adulto	Gatos com Peso Ideal*	Gatos com Baixa Atividade*	Gatos com Peso Ideal* (seco + sachê)	Gatos com Baixa Atividade* (seco + sachê)
3 - 4 kg	45 – 55 g	35 – 45 g	25 – 35g + 1 sachê	15 – 25g + 1 sachê
4 - 5 kg	55 – 65 g	45 – 52 g	35 – 45g + 1 sachê	25 – 32g + 1 sachê
5 - 6 kg	65 – 73 g	52 – 60 g	45 – 53g + 1 sachê	32 – 40g + 1 sachê

*Quantidades individuais podem mudar

1 sachê Gato Adulto WHISKAS® = 85g

Figura 1 – Recomendação diária de ração para gatos castrados
Fonte: Whiskas (2022).

Como visto na Figura 1, se o gato é adulto e castrado, com peso de 5 a 6 kilogramas (kg) pode-se oferecer a quantidade de ração seca de 65 a 73 gramas (g) ao dia, se o gato tem baixa atividade o recomendado é 52 a 60g de ração ao dia, se o tutor preferir alimentar o gato

com ração seca e ração úmida, em forma de 1 sachê de 85g, a quantidade de ração seca deve ser de 32 a 40g ao dia. Alguns animais por serem mais ativos ou por recomendação veterinária podem ter as quantidades individuais modificadas.

2.2 Alimentadores disponíveis no mercado

O mercado de alimentadores automáticos para animais domésticos tem crescido cada dia mais, assim como outros produtos relacionados ao bem-estar dos animais de estimação. Existe uma variedade muito ampla de alimentadores disponíveis no mercado, eles variam de acordo com a capacidade de armazenamento, o número de refeições, a possibilidade da programação de horários para a liberação de comida, diferentes fontes de energia do dispositivo e controle à longas distâncias.

O Alimentador Automático Pet Top[1] da fabricante Sudoeste (2012), é um alimentador automático para alimentar peixes e outros pequenos animais, como pássaros, cachorros e gatos. Consiste em um contêiner capaz de armazenar cerca de 8kg de comida e uma saída que despeja a ração diretamente no recipiente de alimentação do animal, o temporizador aceita até 8 programação no mesmo dia, programado uma vez ele irá acionar todos os dias, possui fonte de energia de 127 Volts (V) e 220 V, Figura 2.

Figura 2 – Alimentador Pet Top
Fonte: Sudoeste (2012).

O alimentador para cães e gatos [2], do inglês *Dog Timed Feeder* da marca Pawise (2022), serve para cães e gatos, com armazenamento de 0,3 kg, ele possibilita que a comida seja liberada apenas uma vez ao dia, programando o horário, esse temporizador automático funciona por até 48 h, como vantagem ele traz uma bolsa térmica que pode ser congelada e colocada

[1] Alimentador Automático Pet Top referido no texto como *Alimentador Pet Top*
[2] Alimentador para cães e gatos é referido no texto como *Alimentador Pawise*

embaixo da refeição para mantê-la em uma temperatura mais baixa, possui base antiderrapante e pode ser higienizado de maneira fácil, funciona através de pilha AA, Figura 3.

Figura 3 – Alimentador da *Pawise*
Fonte: Pawise (2022).

O alimentador da marca PlayPet (2022), oferece a possibilidade do tutor programar as refeições do animal através de um aplicativo, assim se ocorrer algum problema é possível acionar o alimentador através do *smartphone*. Esse alimentador armazena até 2kg e dispensa até 20 porções de 5g de ração a cada refeição, deixando o gato bem satisfeito, funciona com pilhas D ou fonte de energia de 127V e 220V, Figura 4.

Figura 4 – Alimentador PlayPet
Fonte: PlayPet (2022).

Com o comedouro automático programável Chalesco para cães e gatos da marca Pet (2022)[3], é possível programar até 3 refeições diárias para o seu gato, composta por até 12 porções cada, possui a possibilidade de gravar uma mensagem de voz para o animal ouvir quando liberada a comida, possui capacidade de armazenamento de 2,6 kg de ração, funciona a pilhas do tipo "D", Figura 5.

A Tabela 1 relaciona os alimentadores apresentados, de acordo com a sua capacidade de armazenamento em kg, o número de refeições ao dia, os tipos de fontes de energia, se possui acionamento remoto e adereços extras.

[3] comedouro automático programável Chalesco para cães e gatos da marca Magma está referido no texto como *Alimentador Magma*

Figura 5 – Alimentador Magma
Fonte: Pet (2022).

Alimentador	Pet Top	Pawise	PlayPet	Magma
Armazenamento (kg)	8	0,3	2	2,6
N.º refeições ao dia	8	1	20	3
Fonte de energia	127V / 220V	Pilhas AA	127V / 220V ou Pilhas D	Pilhas D
Acionamento remoto	Não	Não	Sim	Não
Extras	–	Bolsa térmica	–	Mensagem de voz

Tabela 1 – Comparação dos alimentadores

2.3 Sistema Operacional Android

Em 2003 uma equipe formada pelo *Rich Miner, Nick Sears, Chris White e Andy Rubin*, criaram uma companhia chamada *Android Inc.*, nomeado pelo *Rubin*, que era apaixonado por robôs e já possuía o domínio ”android.com”, a princípio a ideia era criar um sistema operacional para câmeras fotográficas, mas o mercado não se mostrou interessado. A empresa *Google Inc.* se interessou pelo sistema, porém iria utiliza-lo para celulares e ofereceu U$50 milhões pelo sistema e sua equipe.

Segundo a empresa Google (2021), o Android é um conjunto de *softwares* com base em *Linux* de código aberto criada para diversos dispositivos móveis, a camada de abstração de *hardware* (HAL), consiste em módulos de biblioteca, que implementam uma *Interface* para um tipo específico de componente de *hardware*. O desenvolvimento de melhorias e conexões com diversos dispositivos, por meio do uso de protocolos de conexão, tem sido feitos desde que o sistema operacional foi implantado, tornando a *Google Inc.* líder absoluta nesse mercado.

Um aplicativo móvel para a plataforma Android é comumente desenvolvido usando a linguagem *Java*, uma linguagem de programação orientada a objetos, desenvolvida na década de

90 por uma equipe de programadores chefiada por *James Gosling*, na empresa *Sun Microsystems*, hoje propriedade da *Oracle Corporation*, possui algumas semelhanças com o *Java ME* e utiliza arquivos Linguagem de marcação com regras, do inglês *Extensible Markup Linguage* (XML) para o desenvolvimento de *Interfaces* visuais, mas também pode ser escrito usando outras linguagens de programação existentes, como *Kotlin* e *C/C++*.

Segundo um estudo realizado entre a *Bain & Company* e a *Google*, o Android aparece em 90% dos *smartphones*, ou seja, está instalado em nove a cada dez dispositivos. Por ser um sistema operacional aberto, gratuito, dinâmico e acessível, o Android tem se mostrado altamente atrativo para a comunidade de desenvolvedores e empresas de *software*, sendo o motivo que 83% dos desenvolvedores escolhem esse sistema, (MOURA; CAMARGO, 2020).

2.4 Internet das coisas - *Internet of Things* (IoT)

A Oracle (2021) descreve a IoT como uma rede de objetos físicos incorporados a sensores, *software* e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos e sistemas pela Internet. Esses dispositivos variam de objetos domésticos comuns à ferramentas industriais sofisticadas. Com mais de 10 bilhões de dispositivos IoT conectados em 2020, os especialistas esperam que esse número cresça para 22 bilhões em 2025.

Nos últimos anos, a IoT se tornou uma das tecnologias mais importantes do século XXI, na qual é possível conectar objetos do cotidiano à Internet por meio de dispositivos incorporados e realizar uma comunicação perfeita entre pessoas, processos e outras coisas.

A IoT apresenta diversos desafios, principalmente por lidar com uma imensa quantidade de dados gerados, suas limitações em relação a armazenamento, processamento e comunicação são supridas com o uso de computação em nuvem, portanto, essas tecnologias complementam uma à outra, enquanto a IoT produz uma imensa quantidade de dados, a computação em nuvem fornece mecanismos para mitigar suas limitações de armazenamento e processamento dos dados, essa combinação de tecnologias é conhecida como Nuvem das coisas, do inglês *Cloud of Things* (CoT), (MASCHIETTO *et al.*, 2021).

Por meio da computação de baixo custo, nuvem, *big data*, análise avançada e tecnologias móveis, coisas físicas podem compartilhar e coletar dados com o mínimo de intervenção humana. Nesse mundo hiper conectado, os sistemas digitais podem gravar, monitorar e ajustar cada interação entre itens conectados, o mundo físico encontra o mundo digital, e eles trabalham em conjunto, (ORACLE, 2021).

De acordo com *Magrani*(2018), conforme citado por (MASCHIETTO *et al.*, 2021), “A IoT está aumentando exponencialmente a quantidade e os tipos de dados que estão sendo produzidos. Para obter valor de tais dados, empresas e indivíduos devem obter os dados de seus dispositivos ou coisas, e então inserir tais dados em aplicativos, que podem ser usados para derivar informações e valor em seu benefício.”

2.4.1 Arquitetura IoT-A

A definição de Arquitetura IoT (IoT-A), segundo *Pandikumar e Vetrive* (2014), citados por (MASCHIETTO *et al.*, 2021), é a arquitetura de uma convergência de tecnologias, a partir de sensores e atuadores, que fazem uso de protocolos de internet em dispositivos encapsulados (sistemas embarcados) por Tecnologias de informação e comunicação (TIC), até sua utilização de computação ubíqua/pervasiva, que se voltam novamente aos sensores e atuadores.

Para o mesmo autor existem algumas vantagens e desvantagens atuais da Arquitetura IoT (IoT-A), um paralelo pode ser enumerado como:

- *Comunicação versus compatibilidade* — vantagens de conexão e transparência, reduzindo a ineficiência, porém, ainda há diferentes fabricantes com incompatibilidades de padrões e interconexão.
- *Automação e controle versus complexidade* — sem envolvimento humano, máquinas estão automatizando e controlando grande quantidade de informações, gerando respostas mais rápidas e oportunas, mas uma rede diversa é complexa, e qualquer falha ou erro de *software/hardware* tem graves consequências, mesmo uma simples falta de energia pode causar muita inconveniência.
- *Melhor qualidade de vida versus tecnologia assumindo o controle* — vantagens de conforto e melhor gestão em nossa vida cotidiana, no entanto, nossas vidas serão cada vez mais controladas por tecnologias, das quais passaremos a depender.

2.4.1.1 Camadas da IoT-A

A arquitetura para IoT não possui um consenso sobre qual arquitetura deve ser seguida para o desenvolvimento das soluções, existem arquiteturas de três camadas, quatro camadas, cinco e até 6 camadas. Aarquitetura de 3 camadas possui a parte tecnológica da *IoT* está dividida em: Camada de Percepção, Camada de Rede e Camada de Aplicação, conforme Figura 6.

A camada de Percepção contém os dispositivos como sensores, atuadores, câmeras, entre outros, para coletar dados do mundo real, junto a ela, o *gateway* fornece o mecanismo e os protocolos para os dispositivos exporem seus dados detectados à Internet via *Wi-Fi, Ethernet, GSM, etc.*. A camada de Rede facilita e gerencia a comunicação entre as atividades detectadas no mundo real e a camada de Aplicação, estando junto a ela funcionalidades de segurança. Por fim, a terceira camada é mapeada para aplicações que podem ser usadas pelo consumidor dos dados e para enviar comandos para os dispositivos do mundo real, via Internet por meio de aplicativos móveis, aplicativos da *web*, dentre outras possibilidades (MASCHIETTO *et al.*, 2021).

Figura 6 – Arquitetura básica de IoT em três camadas
Fonte: Autoria Própria.

2.4.2 Protocolos de comunicação

Protocolo, nesse sentido, significa um conjunto de informações, decisões, normas e regras definidas para gerenciar ações que envolvam o funcionamento de algo no mundo real e tecnológico. No cenário da Internet, existem protocolos desenvolvidos para cada camada de atuação, os quais definem regras e formatos para o bom funcionamento da comunicação. Em redes *IoT*, as pilhas de protocolos também estão associadas às camadas de rede para garantia de transferência de dados entre os dispositivos envolvidos nas mais diversas soluções (MASCHIETTO *et al*., 2021).

2.4.2.1 Protocolo MQTT

O MQTT foi criado pela *International Business Machines Corporation* (IBM), pela necessidade de um protocolo simples e leve que conseguisse comunicar várias máquinas entre si, uma comunicação que ocorreria utilizando microcontroladores para a obtenção de dados, que tivesse uma taxa de transmissão leve para a comunicação entre as máquinas e os sensores, que atualmente é um padrão aberto de comunicação entre dispositivos (OASIS.ORG, 2020).

O MQTT possui algumas vantagens perante outros protocolos de comunicação, a qualidade de serviço, facilidade de implementação, baixa alocação de banda, bibliotecas compatíveis com várias linguagens de programação, maior nível de segurança e garantias de entrega.

Com foco na IoT, o MQTT funciona com base no protocolo de rede *TCP/IP*, sua comunicação consiste em duas categorias de entidades numa rede, um servidor de aplicação chamado *broker*, que é responsável por filtrar as mensagens e saber exatamente para quem enviar e os inúmeros clientes, que podem ser qualquer dispositivo que possa interagir com o *broker* e receber e/ou enviar mensagens.

Dessa forma o "*publisher*", ou cliente, e o "*subscriber*", que podem ser cliente ou servidor, não precisam estar diretamente ligados, somente ligados a um *broker*, sendo ele quem vai fazer a notificação de envio ou recebimento de mensagens, Figura 7.

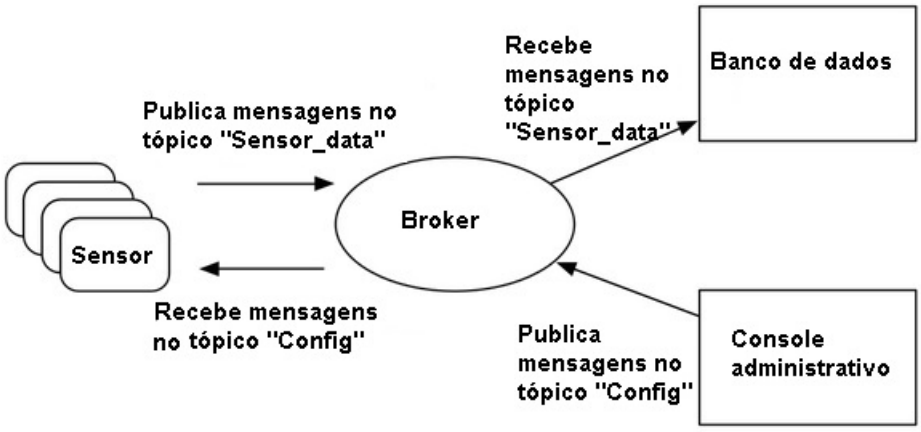


Figura 7 – O modelo de publicação e assinatura do MQTT
Fonte: Adaptada de Yuan (2017).

Na Figura 7, os sensores publicarão suas leituras no tópico *Sensor_data* e se inscreverão no tópico *Config*. Os aplicativos de processamento de dados que salvam os dados do sensor em um banco de dados se inscreverão no tópico *Sensor_data*. Um aplicativo de console administrativo pode receber comandos do administrador do sistema para ajustar as configurações dos sensores e publicar essas alterações no tópico *Config*.

As mensagens MQTT são organizadas por tópicos, onde é possível determinar quais clientes tem acesso a tais tópicos e possui diferentes garantias de entrega, denominado de Qualidade de serviço ,do inglês *(Quality of Service)* (QoS), definidos como:

- QoS 0 (*at most once*): Este nível não tem confirmações de entrega de mensagem e não mantém armazenada a mensagem para retransmissões;
- QoS 1 (*at least once*): Neste nível existe a confirmação de entrega de uma mensagem, mas pode ser que a mensagem chegue várias vezes ao destino por um atraso na chegada de confirmação de recebimento, a mensagem de quem enviou fica armazenada até a confirmação de recebimento.
- QoS 2 (*exactly once*): Garante que a mensagem seja entregue exatamente uma vez, com confirmações de recebimento nos dois sentidos, e a mensagem fica armazenada até que seja confirmado o recebimento.

Segundo Menezes *et al.* (2017), a Unidade de dados de protocolo, do inglês *Protocol Data Unit* (PDU) do protocolo MQTT é encapsulada pelo Protocolo de controle de transmissão, do inglês *Transmission Control Protocol* (TCP), ou seja, o cabeçalho e os dados do MQTT são enviados na área de dados do TCP.

De acordo com Yuan (2017), desenvolvedor da IBM, para iniciar uma comunicação com o *broker*, primeiro o cliente envia uma mensagem *CONNECT* que possui os parâmetros de conteúdo especificados no Quadro 1.

Quadro 1 – Parâmetros da mensagem *CONNECT*

Parâmetro	**Descrição**
cleanSession	Esta sinalização especifica se a conexão é persistente ou não. Ser persistente significa que ela armazena todas as assinaturas e as mensagens possivelmente perdidas (dependendo do QoS) no broker.
username	As credenciais de autenticação e autorização do broker.
password	As credenciais de autenticação e autorização do broker.
lastWillTopic	Se a conexão for perdida o broker publicará automaticamente uma mensagem de “último desejo” no tópico.
lastWillQos	O QoS da mensagem de “último desejo”.
lastWillMessage	A própria mensagem de “último desejo”.
keepAlive	Intervalo de tempo para manter a conexão ativa entre o broker e o cliente.

Como resposta da mensagem *CONNECT*, o cliente recebe uma mensagem *CONNACK*, que possui os parâmetros listados no Quadro 2.

Quadro 2 – Parâmetros da mensagem *CONNACK*

Parâmetro	**Descrição**
sessionPresent	Verifica se existe uma sessão persistente, que a conexão já tenha tópicos assinados
returnCode0	Se retornar zero indica sucesso. Outros valores identificam a causa da falha.

Com a conexão estabelecida, o cliente pode enviar uma ou mais mensagens *SUBSCRIBE* ao *broker* para indicar que ele receberá mensagens de determinados tópicos. A mensagem pode ter uma ou várias repetições dos parâmetros do Quadro 3.

Quadro 3 – Parâmetros da mensagem *SUBSCRIBE*

Parâmetro	**Descrição**
QoS	Indica com que consistência as mensagens neste precisam ser entregues aos clientes.
tópico	O tópico a ser assinado.

Após um tópico ser assinado com sucesso pelo cliente, o *broker* retorna uma mensagem *SUBACK* com um ou mais parâmetros *returnCode*, um código de retorno para cada um dos tópicos no comando *SUBSCRIBE*. Os valores de retorno são os seguintes:

- Valores 0 a 2: sucesso como nível de QoS correspondente.
- Valor 128 = falha.

Para cancelar a assinatura de um ou mais tópicos, o cliente pode enviar uma mensagem *UNSUBSCRIBE* contendo o nome do tópico a parar de receber e enviar mensagens.

Para enviar mensagens, o cliente envia uma mensagem *PUBLISH* ao *broker*. A mensagem contém um tópico e um *payload* ou carga útil de dados, em seguida, o *broker* encaminha a mensagem a todos os clientes que assinam esse tópico, Quadro 4.

Quadro 4 – Parâmetros da mensagem *PUBLISH*

Parâmetro	Descrição
topicName	O tópico no qual a mensagem é publicada.
Qos	O nível de qualidade de serviço da entrega da mensagem.
retainFlag	Sinalização para que o broker retenha a mensagem como a última mensagem conhecida deste tópico.
payload	Os dados reais na mensagem, sequência de texto ou um binário grande de dados.

Segundo Maschietto *et al.* (2021) é possível afirmar que o protocolo MQTT é fundamental para a subsistência da IoT, principalmente em relação aos dados compartilhados por dispositivos sem fio através da rede de computadores.

Neste projeto será utilizado um *broker* gratuito para fins acadêmicos e pago para fins comerciais, o *HiveMQ MQTT Broker* da HiveMQ (2021). Conforme os seus desenvolvedores, o *HiveMQ* é baseado no padrão IoT aberto MQTT, para que as empresas tenham acesso a uma ampla variedade de clientes MQTT de comunidades de *software* livre, como *Eclipse Paho*, bibliotecas MQTT customizadas e bibliotecas direto do *HiveMQ*. A sua maior vantagem é possuir um *dashboard* que permite visualizar e controlar o tráfego de dados desse *broker*, por meio de tópicos, e disponibiliza instruções para usar as bibliotecas necessárias para fazer a comunicação com o mesmo, Figura 8.

Figura 8 – HiveMQ MQTT Broker
Fonte: HiveMQ (2021).

3 MATERIAIS E MÉTODOS

Este capítulo descreve a visão geral do projeto, os requisitos funcionais, os materiais e métodos a serem utilizados para alcançar os objetivos propostos nesse trabalho. Os materiais podem ser divididos em duas partes: os para desenvolvimento do *hardware* e os para desenvolvimento do *software*.

3.1 Visão geral

O alimentador Automático para gatos controlado por aplicativo para dispositivos móveis com sistema operacional Android é constituído por um aplicativo para um *smartphone* e com conexão a Internet via *Wi-Fi*, que envia instruções a um *broker* através de um protocolo de comunicação, o protocolo MQTT, e faz a comunicação a distância com o microcontrolador *ESP32* conectado a Internet via *Wi-Fi*, que controla o acionamento de um servomotor, responsável por movimentar o dosador e distribuir a ração ao gato, Figura 9.

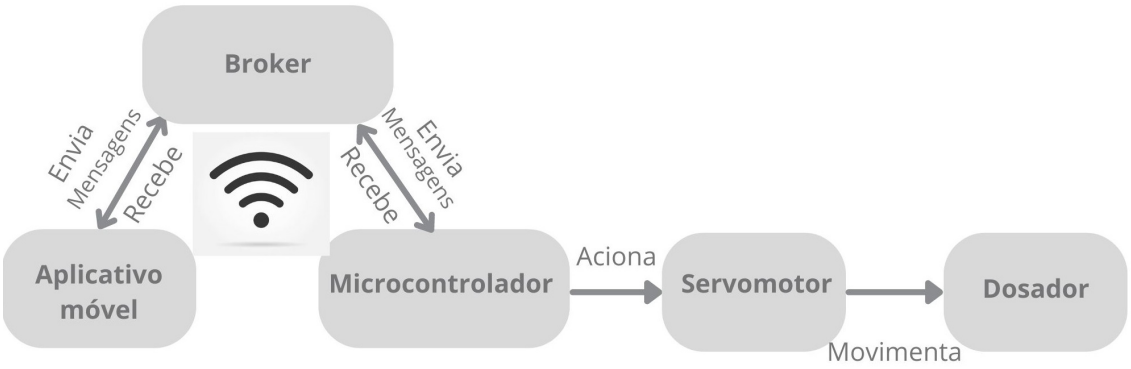


Figura 9 – Elementos do alimentador automático para gatos
Fonte: Autoria própria.

3.2 Requisitos funcionais

Os requisitos funcionais desejados para este trabalho são:

- Controlar a liberação do alimento;
- Dosar aproximadamente $5g$ por porção;
- Funcionamento sem conexão a Internet;
- Fazer a comunicação via *Wi-Fi* entre o *ESP32* e o Android;
- Apresentar a data e hora da última porção liberada na tela do aplicativo.

3.3 MATERIAIS

3.3.1 HARDWARE

Os elementos de *hardware* que serão utilizados para o desenvolvimento do projeto foram:

1. Microcontrolador *ESP32* — Módulo de alto desempenho para aplicações envolvendo *Wi-Fi*;
2. Servomotor — Motor de precisão;
3. Dosador — Separador de porções;
4. Fonte de alimentação — Conversor de corrente alternada para corrente contínua;
5. Bateria — Fonte de energia de reserva quando não alimentado pela rede de energia elétrica;
6. Estrutura acrílica — Reservatório para armazenamento da ração e suporte de sustentação do alimentador.

3.3.1.1 **Microcontrolador ESP32**

Conforme Camargo (2014), um microcontrolador é um sistema computacional quase completo, encapsulado em um único componente, normalmente contém uma Unidade Central de Processamento, do inglês *Central Processing Unit* (CPU), memória, interfaces de entrada e saídas binárias e/ou analógicas e uma ou mais portas de comunicação padrão Porta universal de recebimento/transmissão assíncrona, do inglês *Universal Asynchronous Receiver/Transmitter* (UART), Interface Periférica Serial, do inglês *Serial Peripheral Interface* (SPI) ou Circuito Interintegrado, do inglês *Inter-Integrated Circuit* (I2C).

O módulo *ESP32*, Figura 10, consiste em um microprocessador de baixa potência, *dual core Tensilica Xtensa 32-bit LX6*, com suporte embutido a rede *Wi-Fi* e *Bluetooth*, projetado com a tecnologia de alto desempenho para aplicações envolvendo *Wi-Fi*, baixíssimo consumo de energia, robustez, versatilidade, Interface *usb-serial*, regulador de tensão 3.3V e memória *flash* integrada Espressif (2019).

Este microcontrolador é utilizado para se conectar a Internet via *Wi-Fi*, assinar um tópico no *broker* para receber e enviar instruções ao aplicativo para dispositivos móveis com sistema operacional Android via protocolo MQTT e para acionar o servomotor quando for recebida uma mensagem do aplicativo.

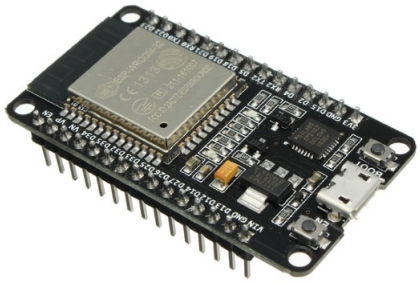


Figura 10 – Módulo Esp32
Fonte:
.
Esp322019

3.3.1.2 **Servomotor**

Os motores elétricos são máquinas elétricas que possuem como característica transformar energia elétrica em energia mecânica. Os motores de corrente contínua — CC ou também chamados motores Corrente direta ou Corrente Contínua, do inglês *Direct Current* (DC), são motores que possuem ímãs permanentes ou motores com campo e armadura, estes são acionados a partir de uma fonte de corrente contínua e possuem diversas aplicações. Os servomotores são feitos a partir de motores CC com a adição de alguns componentes como engrenagens de redução, sensor de posição do eixo do motor, para medir o quanto o motor está girando e em qual direção, são muito utilizados em brinquedos para o ajuste de direção ou ajuste de posição de braços robóticos em que a precisão dos movimentos é essencial.

Conforme dito por Mataric (2014), a operação do servomotor resume-se a deixar o eixo do motor na posição desejada, essa posição está em algum lugar ao longo de 180 graus em qualquer direção a partir do ponto de referência e é especificada por um sinal eletrônico.

De acordo com o fabricante TowerPro (2014) do servomotor do modelo *MG90S*, Figura 11, a velocidade de operação de 0,1 segundos (s) equivale a 60 graus se alimentado com tensão de 4,8V e 0,08s equivale a 60 graus com tensão de 6V, o sinal desses ângulos indica qual a orientação do movimento, ângulos positivos tem orientação para a direita e ângulos negativos orientação para a esquerda.

Por precisar de precisão, neste projeto, um servomotor, foi usado para movimentar o dosador, com a quantidade correta por porção, e assim transportar a ração até despejar no comedouro do gato. Este servomotor é constituído por engrenagem de metal com um rolamento, é pequeno e leve, ele pode girar aproximadamente 180 graus, 90 graus em cada direção, com peso 13,4g, torque de $1{,}8\frac{kg}{centmetros(cm)}$ quando alimentado por 4,8V e $2{,}2\frac{kg}{cm}$ quando alimentado com 6V, ou seja, sua tensão de operação é entre 4,8V a 6V.

Figura 11 – Servomotor
Fonte: TowerPro (2014).

3.3.1.3 **Dosador**

No mercado existem várias categorias de dosadores, o mais comumente utilizado é a rosca transportadora helicoidal ou rosca sem fim, desenvolvida para facilitar o processo de manuseio e transporte de produtos a granel, contudo neste alimentador é preciso separar as porções de aproximadamente 5 g cada, tendo este parâmetro como diferencial aos demais alimentadores existentes no mercado, este dosador é projetado de acordo com esse parâmetro.

3.3.1.4 **Fonte de alimentação**

As fontes de alimentação, são usadas para transformar corrente elétrica da rede de distribuição, que chegam com tensão alta, para uma tensão menor, indicada para cada categoria de *hardwares*.

A rede de distribuição elétrica possui tensão de 127V ou 220V, conforme a região. Os componentes eletrônicos desse projeto trabalham com tensões de 3,3V a 12V, a fonte utilizada é uma fonte DC chaveada, Figura 12, de 9V e 1 Ampers (A), necessitando assim de uma fonte ajustável,Figura 13, que converta a tensão de entrada para 3,3V ou 5V, para tornar compatível com as tensões responsáveis por alimentar o microcontrolador e o servomotor. A fonte garante a estabilidade da tensão, pois a rede elétrica pode sofrer variações, o que pode ser prejudicial se ligada diretamente aos componentes.

3.3.1.5 **Bateria**

A bateria consiste em um conjunto de acumuladores elétricos interligados em série e armazenam energia eletroquímica, baseada na conversão de energia química em energia elétrica e vice-versa. Um *Powerbank* é um dispositivo carregador portátil que possui baterias de íons de lítio que armazena e distribui energia, utilizado principalmente para *Smartphones* por seu

Figura 12 – Fonte DC chaveada
Fonte: FilipeFlop (2022).

Figura 13 – Fonte ajustável
Fonte: FilipeFlop (2022).

tamanho pequeno, pode ser levado a qualquer lugar e conexão simples via cabo Porta universal serial, do inglês *Universal Serial Bus* (USB). Este projeto contou com um *Powerbank*, Figura 14, com capacidade de 2800miliampers por hora (mAh), com entrada e saída DC, de 5V e 1A.

3.3.1.6 **Reservatório**

O reservatório é o lugar de armazenagem da ração, para poderem ser limpos com facilidade, evitar a entrada de insetos e umidade, o material indicado é *PVC* ou acrílico. Por serem materiais fáceis de serem encontrados, podem ser usados em inúmeras situações, além das comumente utilizadas em tubulações de água e esgoto, pela possibilidade de encaixar os tubos e fazer conexões sem precisar soldar ou colar, facilitando assim o seu desmonte. O reservatório escolhido é um pote acrílico transparente com tampa hermética, Figura 15 com vedação que bloqueia totalmente o ambiente interno, protegendo e conservando o seu conteúdo por

Figura 14 – Powerbank Mox
Fonte: Mox (2022).

mais tempo, com capacidade de armazenamento de 2kg de ração e dimensões aproximadas de 23cm de altura, 12cm de largura e comprimento de 12cm.

Para melhor visualização dos seus materiais, caixas de acrílico transparente, Figura 16, foram utilizados para os demais componentes da estrutura do alimentador.

Figura 15 – Reservatório
Fonte: http://www.injeplastec.com.br.

3.3.2 SOFTWARE

Os elementos de *software* para o desenvolvimento do projeto foram:

1. Arduino IDE — Ambiente de desenvolvimento de *software* para placas *Arduino* e compatíveis;
2. Android IDE — Ambiente de desenvolvimento integrado Android;

Figura 16 – Caixa Acrílica Transparente
Fonte: Autoria própria.

3.3.2.1 **Arduino IDE**

Conforme seus criadores, Banzi *et al.* (2018), o *Arduino* é uma plataforma eletrônica de código aberto baseada em *hardware e software* fáceis de usar. A linguagem de programação *Arduino* é baseada em *Wiring*, e o *Arduino Software (IDE)*, baseado em *Processing*.

A *Wiring* é uma estrutura de programação de código aberto para microcontroladores que permite escrever programas de plataforma cruzada para controlar dispositivos conectados a uma ampla gama de placas de microcontrolador para criar todos os tipos de codificação criativa, objetos interativos, espaços ou experiências físicas (BARRAGAN, 2003).

O *Processing* promoveu a alfabetização em *software*, especialmente nas artes visuais, e a instrução visual em tecnologia. Criado inicialmente para servir como um caderno de rascunhos de *software* e para ensinar os fundamentos da programação em um contexto visual, o *Processing* também se tornou uma ferramenta de desenvolvimento para profissionais, sendo um *software* de processamento gratuito e de código aberto rodando em *Mac OS, Windows e Linux* (FRY; REAS, 2003).

A Ambiente de desenvolvimento integrado, do inglês *Integrated Development Environment* (IDE) do *Arduino*, a *Arduino IDE* permite a criação de *sketches* ou esboços para as placas.

Para Souza (2013), a IDE apresenta um alto grau de abstração, possibilitando o uso de um microcontrolador sem que o usuário conheça o mesmo, nem como deve ser usado os registradores internos de trabalho. Ela possui uma linguagem própria baseada na linguagem C e C++.

O *software IDE Arduino* para Jr e Silva (2015), é uma ferramenta de edição de linguagem de alto nível (C) que compila o código nessa linguagem, converte para a linguagem *Assembly* e posteriormente converte em código binário para embarcar no microcontrolador.

Este *software* de desenvolvimento será utilizado para programar e configurar o microcontrolador *ESP32*, que fará a comunicação com o aplicativo para dispositivos móveis com sistema operacional Android, por meio do protocolo de comunicação MQTT, pela conexão *Wi-Fi*, e comandará o acionamento do servomotor.

3.3.2.2 **Android IDE**

De acordo com seus desenvolvedores, o *Android Studio* é o ambiente oficial para o desenvolvimento de aplicativos *Android* sendo baseado no *IntelliJ IDEA* da JetBrains (2000), uma IDE para *Java* que compreende e tem assistência para codificação em outras linguagens de programação, como *SQL, JPQL, HTML, JavaScript*, entre outras.

Conforme o Google (2021), o *Android Studio* oferece mais recursos do que o editor de código e das ferramentas de desenvolvedor avançadas do *IntelliJ*, possuindo um sistema de compilação flexível baseado em *Gradle*, um emulador rápido e com muitos recursos, sendo um ambiente unificado que possibilita o desenvolvimento para todos os dispositivos Android. Projetos baseados no *Gradle* oferecem recursos consideráveis para o desenvolvedor, entre eles:

- Compatibilidade com bibliotecas binárias (AARs);
- Facilidade de configuração e personalização de compilações;
- O Gradle pode ser usado no ambiente de desenvolvimento integrado, mas também na linha de comando e de servidores de integração contínua, como o *Jenkins*, o que gera a mesma compilação em qualquer lugar, a qualquer momento.

Este *software* possui um emulador, *Android Emulator*, para simular o comportamento de dispositivos Android no computador, sendo possível testar o aplicativo em diversos dispositivos e várias versões da plataforma Android. Este emulador oferece quase todos os recursos de um dispositivo Android real, como simular o recebimento de chamadas telefônicas e mensagens de texto, especificar o local do dispositivo, simular diferentes velocidades de rede, simular rotação e outros sensores de *hardware*, acessar a *Google Play Store*. Realizar o teste do aplicativo no emulador é mais rápido e fácil do que fazer isso em um dispositivo físico (GOOGLE, 2021).

A este projeto será empregado o *Android Studio*, por aceitar as bibliotecas necessárias para utilizar o protocolo MQTT, facilitando a comunicação entre o Android e o microcontrolador e por permitir o teste de aplicativo no próprio *software*.

4 RESULTADOS

A este capítulo destina-se a uma breve descrição do sistema proposto, o desenvolvimento e os resultados, assim como os problemas encontrados no decorrer deste projeto.

4.1 Escopo do sistema

Este alimentador automático tem como proposta principal liberar pequenas porções de alimento a um animal de estimação, seu funcionamento pode ser o pré-programado, o qual funciona mesmo sem a conexão com à Internet e consiste em liberar 2 porção de aproximadamente 5g cada, no intervalo de 2h, tendo seu funcionamento de 24h, disponibilizará 60g de ração em 12 refeições, o recomendado para um gato com peso de 5 a 6kg, castrado e com baixa atividade, de acordo com a Quadro 5.

Os tipos de acionamentos podem ser configurados pelo usuário por meio do aplicativo *BiBiCat*.[1] O *Alimentar programado* é um tipo de funcionamento ao qual pode-se definir o número de porções e o intervalo em que devem ser distribuídos, ao programar pela primeira vez este modo, esta configuração de alimentação será armazenada e substituirá o pré-programado, funcionando mesmo que a conexão *Wi-Fi* seja perdida, outro funcionamento é o *Alimentar Agora*, acionado via aplicativo ao apertar um botão, que libera 1 porção de 5g imediatamente ao animal. Este projeto tem uma bateria auxiliar em caso da falta de energia, com autonomia aproximada de 7h e 40 minutos (min).

Quadro 5 – Recomendação diária de ração para gatos castrados

Peso do gato adulto	**Gatos com peso ideal***	**Gatos com baixa atividade***	**Gatos com peso ideal* (seca + sachê)**	**Gatos com baixa atividade* (seca + sachê)**
3 - 4kg	45 - 55g	35 - 45g	25 - 35g + 1 sachê	15 - 25g + 1 sachê
4 - 5kg	55 - 65g	45 - 52g	35 - 45g + 1 sachê	25 - 32g + 1 sachê
5 - 6kg	65 - 73g	52 - 60kg	45 - 53g + 1 sachê	32 - 40g + 1 sachê
**Quantidades individuais podem mudar 1 sachê gato adulto WHISKAS = 85g*				

4.2 Desenvolvimento

4.2.1 Aplicativo para dispositivos móveis no Android Studio IDE

Para o desenvolvimento do aplicativo do alimentador, um novo projeto em linguagem *Java* foi criado, utilizando a IDE escolhida, o Android usa o *Gradle* como um sistema de construção e gerenciamento de dependências, para fazer a conexão entre o dispositivo móvel e o microcontrolador, os responsáveis por esta comunicação são o protocolo MQTT e o *broker*

[1] Aplicativo *BiBiCat* = Aplicativo para dispositivos móveis com sistema operacional Android desenvolvido neste trabalho.

público, que é o intermediário que localiza e redireciona as mensagens recebidas e enviadas através do protocolo MQTT. Para fazer essa comunicação é necessário utilizar a biblioteca do *Eclipse Paho*, o *Paho Android Service*, que é uma interface para a biblioteca *Paho Java MQTT Client* para a plataforma Android e faz a conexão MQTT encapsulada em um *Android-Service*, executado no plano de fundo do aplicativo, mantendo-o ativo enquanto o mesmo altera entre diferentes atividades, a importação dessas bibliotecas é feita no *build.gradle* do projeto, Figura 17, essa camada de abstração se faz necessária para garantir o recebimento das mensagens de forma confiável através do protocolo MQTT.

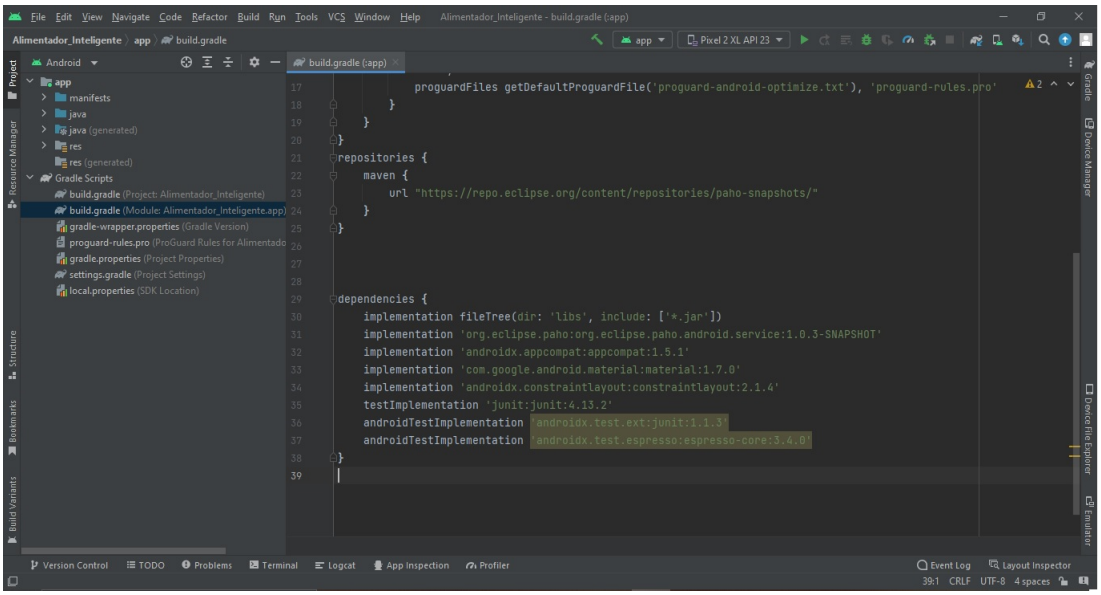


Figura 17 – Importação da biblioteca Paho Android Service
Fonte: Autoria própria (2022).

A Figura 18 apresenta as telas do aplicativo do **BiBiCat**, a tela inicial conta com as duas possibilidades de funcionamento e mostra a última alimentação disponibilizada contendo o dia da semana, data, hora e o número de porções liberadas ao longo do dia e a programação de alimentação. A tela de **Alimentar Agora** aparece ao ser selecionada na tela principal, na qual é possível liberar imediatamente 1 porção de aproximadamente 5g de ração, verificar qual a última alimentação ocorrida e o número de porções já oferecidas neste dia. Ao apertar o botão *ALIMENTAR AGORA*, o aplicativo **BiBiCat** envia uma mensagem contendo o número "1"no tópico MQTT de envio, a qual será lida pelo microcontrolador e realizará a liberação do alimento, logo após a ação ser realizada o microcontrolador envia uma mensagem no tópico MQTT de recebimento, contendo o dia da semana, a data, hora e o número de porções totais servidas no dia, contando com a alimentação liberada agora, como por exemplo, *Quinta - feira - 08/12/2022 - 11:40:45 * 10*. Na tela de **Alimentação Programada** é possível definir o intervalo das alimentações bem como o número de porções a serem distribuídas ao gato, esta programação atualizará a pré-programação de funcionamento. Ao apertar o botão *PROGRAMAR*, o aplicativo **BiBiCat** lê e converte o intervalo de horas para minutos, para enviar uma mensagem no tópico MQTT de envio com o formato, minutos-número de porções, como por exemplo alimentar com inter-

valos de 2h e liberar 2 porções de aproximadamente 5g, a mensagem enviada vai ser *120-2*, a qual será lida pelo microcontrolador, programando e acionando o seu funcionamento decorrido o intervalo definido, a mensagem contendo os dados da última alimentação são enviados pelo microcontrolador no tópico MQTT de recebimento e mostrada na tela do aplicativo.

Figura 18 – Aplicativo BiBiCat
Fonte: Autoria própria (2022).

4.2.2 Controlador

O desenvolvimento do programa para o microcontrolador foi escrito na *Arduino IDE*, Figura 20, usando o mesma linguagem para microcontroladores *Arduino*. Este *software* utiliza várias bibliotecas *Open Source*, que são bibliotecas de código aberto, desenvolvidas por inúmeras pessoas para várias utilizações e diversos microcontroladores.

Algumas pequenas mudanças são necessárias para utilizar o microcontrolador *ESP32* nesta IDE, a biblioteca da plataforma *ESP32* que configura a IDE precisa ser instalada, no menu ferramentas no gerenciador de placas a IDE mostra as diferentes placas que podem ser instaladas, neste caso a *ESP32* da Espressif (2022), Figura 19. Após instalar a placa, no menu ferramentas aparecem as configurações para usar a placa do *ESP32* como mostra a Figura 20.

As bibliotecas usadas nesse projeto, as variáveis e definições estão descritas na Listagem 1.

O objetivo deste código é realizar a conexão do microcontrolador a rede *Wi-Fi* do local em que o alimentador ficará, por meio do aplicativo móvel da Espressif (2019) chamado *Esp-Touch*, Figura 21, o qual localiza e configura a rede usada pelo *smartphone* como a adequada para o funcionamento, após recebida a confirmação que a rede está conectada, Listagem 2, conectar-se ao *broker* utilizando a porta e o endereço do mesmo e registrar-se no tópico determinado, de acordo com a Listagem 3.

O microcontrolador *ESP32* possui dois núcleos de processamento, ou seja, *dual core* e para utiliza-los é necessária a biblioteca do FreeRTOS (2022) para o *ESP32*, no qual tarefas diferentes podem ser executadas ao mesmo tempo sem que interfiram entre si. A tarefa *T1Code*,

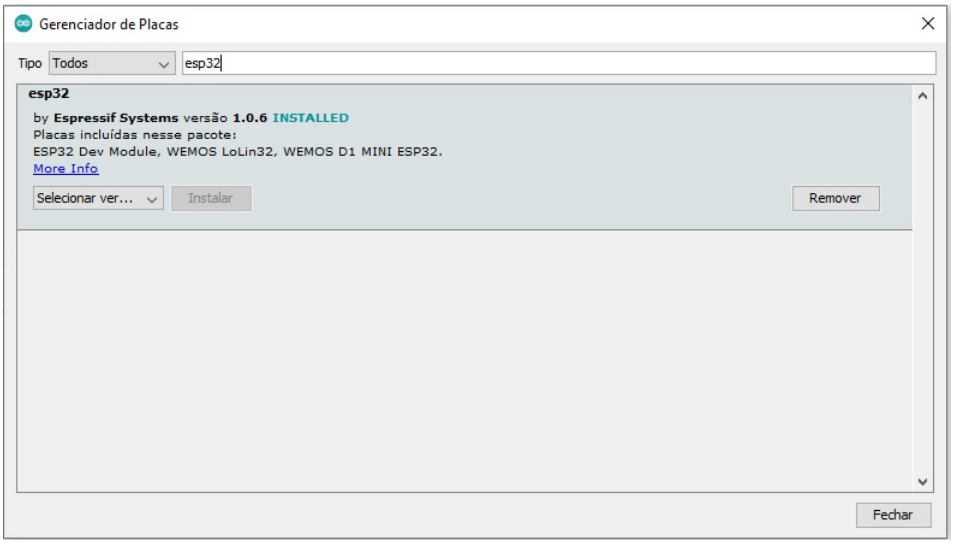


Figura 19 – Gerenciador de placas
Fonte: Autoria própria (2022).

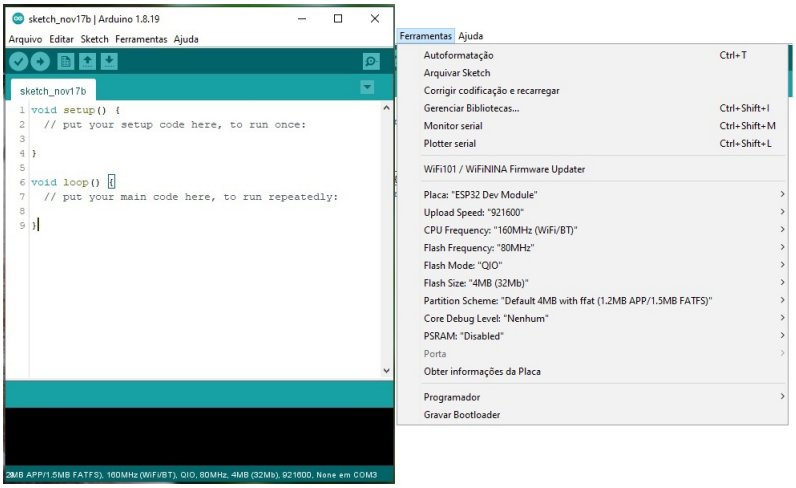


Figura 20 – Página inicial da Arduino IDE
Fonte: Autoria própria (2022).

executada em um dos núcleos permite que o alimentador funcione pré-programado mesmo sem estar conectado a rede *Wi-Fi*, em outro núcleo, a tarefa *T3Code* é responsável por chamar a configuração e conexão com a rede, e a outra tarefa *T2Code* é responsável pela função *Alimentar Agora*.

A função *mqtt_callback* é responsável por monitorar o recebimento de mensagem no tópico subscrito, ou tópico de recebimento, e de acordo com o tipo de mensagem recebida chamar a tarefa responsável pela sua execução, Listagem 5, para isso é necessário usar a biblioteca do MQTT para o *ESP32*.

Um dos tipos de mensagens que pode ser recebidas aciona o *Alimentar Agora*, que é a função de liberar uma porção de 5g imediatamente e logo após o acionamento chama a função *Dia_Hora*, que envia uma mensagem em outro tópico, o tópico de envio, com o dia da semana, a data, hora e o número de porções já disponibilizadas no dia. O outro tipo de mensagem que

Listagem 1 – Bibliotecas

```
#include <TridentTD_EasyFreeRTOS32.h> //Biblioteca do FreeRTOS para o Esp32
#include <Servo.h> //Biblioteca do servomotor para o esp32
#include <PubSubClient.h>//Biblioteca do MQTT para o esp32
#include <NTPClient.h>//Biblioteca do Relógio On-line
#include <WiFi.h> //Biblioteca da Wi-Fi para o esp32

/*.........Configurações do MQTT.......... */
// porta e endereço do broker publico
const char* mqttServer = "broker.hivemq.com";
const int mqttPort = 1883;

//tópico MQTT de recebimento de mensagens
#define Topico_Recebe "MQTTRecebe"
//tópico MQTT de envio de mensagens
#define Topico_Envia "MQTTEnvia"

void mqtt_callback(char* topic, byte* payload, unsigned int length);
WiFiClient espClient;
PubSubClient client(espClient);

/*............SERVO ............... */
static const int servoPin = 4;
Servo servo1;

/*...........TAREFAS.............. */
EasyFreeRTOS32   runner1, runner2;
void  T1Code(void*), T2Code(void*), T3Code(void*);

/*...........VARIAVEIS.............. */
int Total_porcoes;
String msg;
String tempo;
String porcao;
int intervalos;
int TEMPO_TASK = 7200000; // 2 horas em milisegundos pré definidos
int Numero_porcoes = 2;// 2 porções pré definidas

/* --------- Configurações de relógio on-line ---------- */
WiFiUDP udp;
NTPClient ntp(udp, "a.st1.ntp.br", -3 * 3600, 60000);
//Cria um objeto "NTP" com as configurações.utilizada no Brasil
```

Fonte: Autoria própria (2022).

pode ser recebida, aciona o *Alimentar Programado*, ele recebe a mensagem com o intervalo de alimentação em minutos e quantas porções devem ser liberadas passado esse intervalo, também chama a função *Dia_Hora*, que envia a mensagem no tópico de envio, contendo os mesmos dados da última alimentação. Caso o usuário decida programar a alimentação, a pré-programação é modificada de acordo com o que foi definido pelo usuário. As mensagem envia-

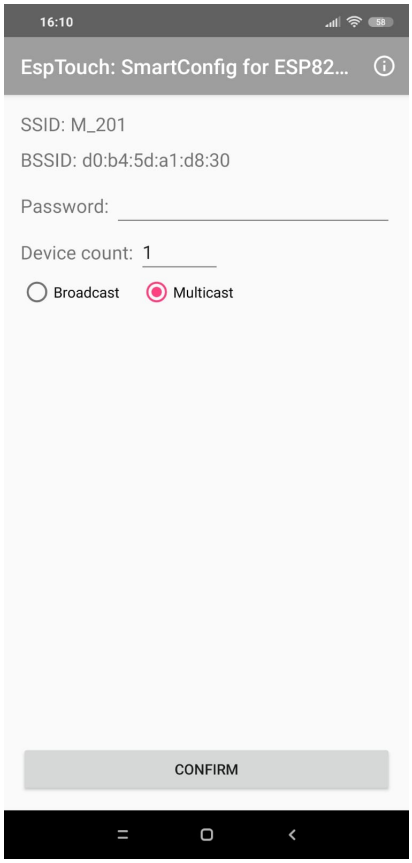


Figura 21 – Aplicativo EspTouch
Fonte: Espressif (2019).

das ou publicadas no tópico de envio serão lidas e mostradas em tela pelo aplicativo **BiBiCat**, desenvolvido para este alimentador, Listagem 4.

A função *Dia_Hora* é responsável por configurar o relógio online, a data e determinar o dia da semana, bem como enviar no tópico de envio essas informações, Listagem 6.

A função *setup()* é chamada quando um projeto se inicia, usado para inicializar variáveis, configurar o modo dos pinos de entrada e saída, inicializar bibliotecas, ela será executada apenas uma vez, após a placa ser alimentada ou reiniciada. A função *loop()* faz a execução consecutiva do que foi escrito na função *setup()* enquanto a placa estiver ligada. Neste projeto essas funções são utilizadas para iniciar as tarefas, atualizar o relógio *online* e inicializar a configuração do cliente no *broker*, Listagem 7.

4.2.3 Projeto do Dosador

O dosador foi projetado exclusivamente para este alimentador, em que a porção deve ser de aproximadamente 5g, dada essa especificação, as marcas de ração seca, SpecialCat (2021), Royal (2021), MaxTotal (2021) e Whiskas (2022) foram consultadas para saber qual o peso médio equivalente em mililitros (ml), com isso é possível calcular a área necessária para obter a porção desejada. Os fabricantes de ração estimam a quantidade de alimento compa-

Listagem 2 – Função de conexão *Wi- Fi*

```
void setup_conect() {
  // se a Wi-Fi não está conectada
 if (WiFi.status() != WL_CONNECTED){
    //Inicia a conexão da Wi-Fi do esp, começa o SmartConfig
    WiFi.mode(WIFI_AP_STA);
    WiFi.beginSmartConfig();
    delay(1000);

    //Espera o pacote SmartConfig do app
    Serial.println("Esperando pelo app EspTouch.");
    while (!WiFi.smartConfigDone()) {
      delay(5000);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("Configuração de rede recebida.");
    //Espera pela conexão Wi-Fi do esp ser estabelecida
    Serial.println("Aguardando resposta da rede Wi-Fi");
    //enquanto a Wi-Fi do esp não foi conectada
    while (WiFi.status() != WL_CONNECTED) {
      delay(5000);
      Serial.print(".");
    }
  }
  Serial.println("Wi-Fi Conectada!");
  if (WiFi.status() == WL_CONNECTED) {
    Serial.print("Conectado a rede :");
    Serial.println(WiFi.SSID());
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
 }
}
```

Fonte: Autoria própria (2022).

rando a porção em ml e g, a maioria das rações apresentadas possuem grãos com aproximadamente de 8 milimetros (mm) de largura para gatos adultos e 6 mm para filhotes. De acordo com as marcas pesquisadas, 200ml equivalem a aproximadamente 80g, com o dosador em formato de um pedaço de pizza, ou uma parte de um círculo, desenhado assim de acordo com o formato do reservatório de armazenamento, temos que aproximadamente 5g equivalem a aproximadamente 12,5ml ou cm^3, com uma altura estimada de 15mm ou 1,5cm, temos a área de aproximadamente 8,334 cm^2 e um ângulo de aproximadamente 54,143 graus. Por ser uma peça desenhada de acordo com o tamanho desejado, o *software* de modelagem 3D, Rhinoceros (2022), é utilizado para desenvolver o modelo, Figura 22 e posteriormente ser impresso em filamento *PLA*, um material derivado do milho ou outros amidos renováveis que são biodegradáveis e não tóxicos, Figura 23.

Listagem 3 – Função de conexão ao *Broker*

```
  // inicia o mqtt
  client.setServer(mqttServer, mqttPort);
  client.setCallback(mqtt_callback);
  while (!client.connected()) {
    Serial.println("Tentando se conectar ao broker MQTT: ");

    if (client.connect("")){
      Serial.println("Conexao ao broker MQTT feita com sucesso!");
    // se inscreve para poder publicar e receber as mensagens desse tópico
      client.subscribe(Topico_Recebe);
    }
    else{
      Serial.println("Falha ao se conectar ao broker MQTT.");
      Serial.println("Nova tentativa em 2s...");
      delay(2000);
    }
 }
}
```

Fonte: Autoria própria (2022).

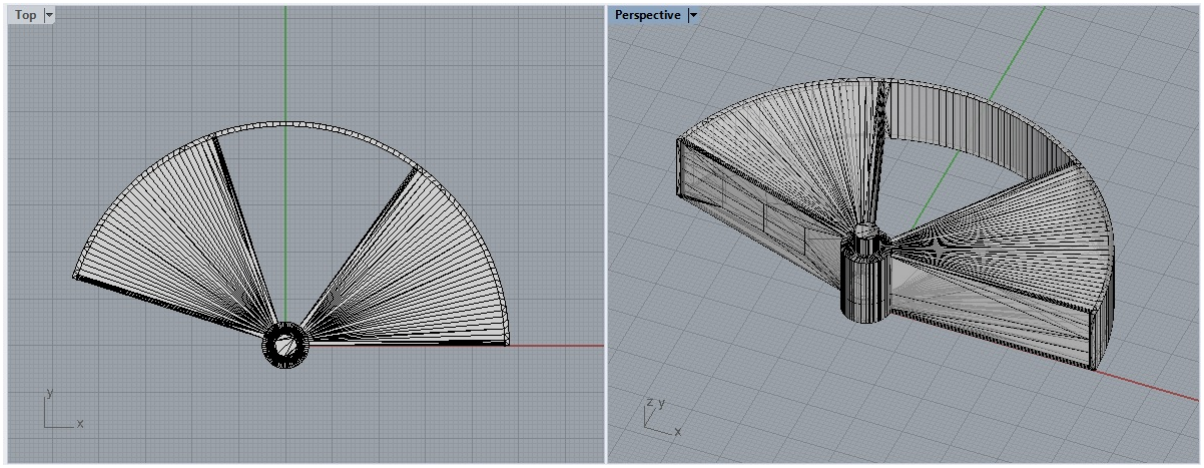


Figura 22 – Modelo 3D do dosador
Fonte: Autoria própria (2022).

O responsável pelo movimento do dosador é o servomotor, o microcontrolador recebe a mensagem de acordo com a função e aciona o servomotor, fazendo o dosador movimentar-se de acordo com o ângulo e direção definidos no projeto, o movimento se repete de acordo com o número de porções a serem distribuídas, este dosador possui duas partes fechadas e uma vazada para medir a quantidade de ração, as partes fechadas servem para separar o reservatório de ração no momento de distribuir a comida e a outra parte serve para fechar a saída e impedir a entrada de sujeira ou insetos no reservatório. Ao servomotor foi adicionado um suporte especialmente desenvolvido para este projeto, com encaixes para o motor e abertura para a passagem da ração, assim como uma superfície lisa para segurar a porção do dosador

Listagem 4 – Funções das tarefas

```
void T1Code(void *) {//Tarefa pré programada
  LOOP(){
    Serial.print("t1: ");
    Serial.print("Alimentar programado em: ");
    Serial.println(TEMPO_TASK);
    int i = 1;
    while (i <= Numero_porcoes) {
      Serial.println("Servindo...");
      Serial.println(i);
      Serial.println("Carregando por 5s");
      servo1.write(0);
      delay(5000); // Tempo de espera entre ir e voltar
      Serial.println("Descarregando por 5s");
      servo1.write(70);
      delay(5000);
      i = i + 1;
    }
    Total_porcoes += Numero_porcoes;
    Dia_Hora();
      DELAY(TEMPO_TASK);
  }
}
void T2Code(void * ) { //Alimentar agora
  Serial.println("t2: ");
  Serial.print("Alimentar Agora");
  Serial.println("Carregando por 5s");
  servo1.write(70);
  delay(5000);
  Serial.println("Descarregando por 5s");
  servo1.write(0);
  delay(5000); // Tempo de espera entre ir e voltar
  servo1.write(70);// volta ao inicio
   Total_porcoes = Total_porcoes + 1;
  Dia_Hora();
 vTaskDelete(NULL);
}
void T3Code(void * ) {//configurar Wi-Fi
  Serial.println("t3: Configurar Wi-Fi ");
  delay(500);
  servo1.attach(servoPin);
  setup_conect();
  vTaskDelete(NULL);
}
```

Fonte: Autoria própria (2022).

até a abertura que dá acesso ao tubo de distribuição, este suporte foi impresso em 3D devido suas especificações, Figura 24.

Listagem 5 – Função de *Callback*

```
void mqtt_callback(char* topic, byte* payload, unsigned int length)
{
  Serial.println("Mensagem chegou no tópico: [");
  Serial.print(topic);
  Serial.print("] ");
  msg = "";
  for (int i = 0; i < length; i++) {
    char c = (char)payload[i];
    msg += c;
  }
  Serial.println(msg);
  /*..............AGORA............ */
  if (msg.equals("1")) {
    Serial.println("Chamando t2...");
    delay(500);
    runner2.start(T2Code);
  }
  /*..............PROGRAMADO............ */
  else {
    tempo = msg.substring(0, 3);
    Serial.println(tempo);
    porcao = msg.substring(4, 6);
    Serial.println(porcao);
    Numero_porcoes = porcao.toInt();
    intervalos = (tempo.toInt() * 60000);
    if (intervalos != TEMPO_TASK) {
       TEMPO_TASK = intervalos;
    }
    Serial.println("Armazenando em  t1...");
    delay(500);
   runner1.start(T1Code, NULL, 10000, 0);
  }
}
```

Fonte: Autoria própria (2022).

4.3 Testes

Nesta fase de testes, foram feitos testes para saber a duração da bateria e o recebimento e envio de mensagens via *broker* pelo *ESP32* e pelo aplicativo **BiBiCat**. O teste de bateria foi realizado diretamente conectado com o microcontrolador e com acionamento do servomotor por 2 vezes a cada 5 minutos, tendo duração de aproximadamente 7h e 40min de funcionamento.

O teste de recebimento e envio de mensagens via *broker* foi executado usando o *MQTT Websocket Client* da HiveMQ (2021), onde é possível se conectar como um cliente, se inscrever nos tópicos desejados, enviar e receber mensagens do aplicativo **BiBiCat** e do microcontrolador, pelo protocolo MQTT, para garantir o funcionamento de acordo com o projeto, Figura 25.

Listagem 6 – Função Dia_Hora

```
void Dia_Hora() {
  ntp.forceUpdate();     /* Atualização do dia */
  int diaSemana = ntp.getDay();
  String diaS;
  if (diaSemana == 0){
    diaS = "Domingo";}
  else if (diaSemana == 1){
    diaS = "Segunda - Feira";}
  else if (diaSemana == 2){
    diaS = "Terça - Feira";}
  else if (diaSemana == 3){
    diaS = "Quarta - Feira";}
  else if (diaSemana == 4){
    diaS = "Quinta - Feira";}
  else if (diaSemana == 5){
    diaS = "Sexta - Feira";}
  else{
    diaS = "Sábado";}
  String Dia_str = ntp.getFormattedDate();
  String Hora_str = ntp.getFormattedTime();
  String DiaHoraS = ("" + diaS + " - " + Dia_str + " - "
  + Hora_str + " * " + Total_porcoes);
  Serial.println(DiaHoraS);
  char DHS[60];// a data, hora e dia da semana
  DiaHoraS.toCharArray(DHS, 60);
  client.publish(Topico_Envia, DHS);//publica no tópico de envio
}
```

Fonte: Autoria própria (2022).

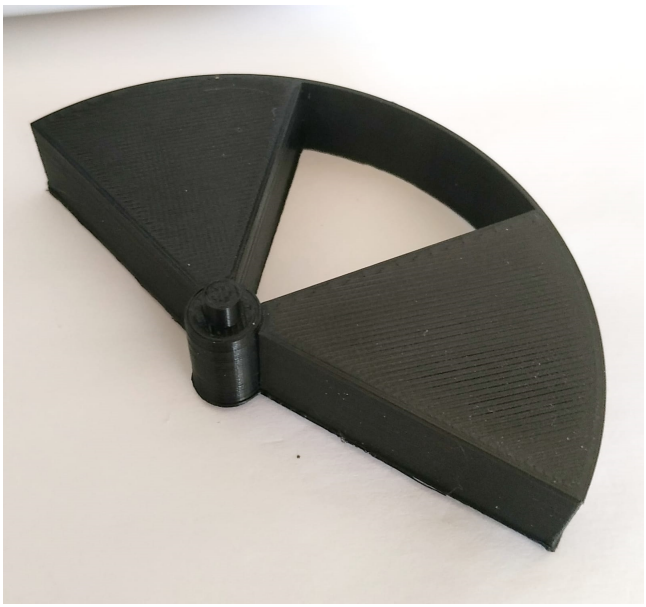


Figura 23 – Dosador impresso

Fonte: Autoria própria (2022).

Listagem 7

```
1  void setup() {
2    Serial.begin(115200);
3    delay(5000);
4    WiFi.begin();
5    Serial.println("Alimentador Automático...");
6    delay(5000);
7    Serial.println("Criando as tarefas...");
8    delay(500);
9    runner2.start(T3Code);
10   delay(60000);
11   runner1.start(T1Code,NULL, 10000, 0);
12 }
13 void loop() {
14   ntp.begin();
15   ntp.forceUpdate();
16   client.loop();
17 }
```

Fonte: Autoria própria (2022).

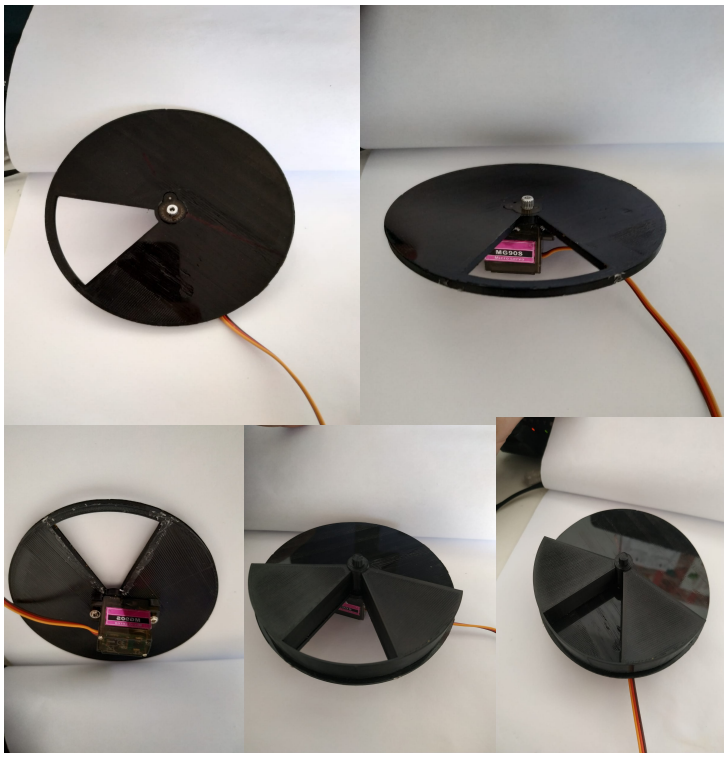


Figura 24 – Suporte do motor

Fonte: Autoria própria (2022).

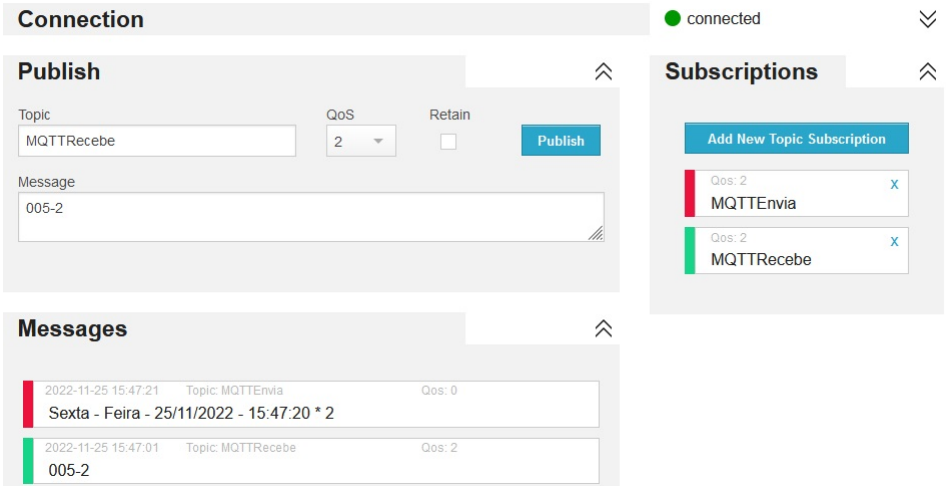


Figura 25 – MQTT Websocket Client
Fonte: Adaptado de HiveMQ (2021).

Os testes de dosagem e quantidade de ração distribuída foram feitos disponibilizando 20 porções de duas rações de tamanhos e marcas diferentes, Figura 26 ambas para gatos castrados, e pesadas separadamente obtendo-se pesos de 111g e 112g. O peso médio por porção foi de aproximadamente 5,575g e o desvio padrão foi de 0,5g por porção, o que é considerado aceitável para esse projeto.

Figura 26 – Tamanho dos grãos de ração
Fonte: Autoria própria (2022).

4.4 Problemas encontrados

No decorrer deste projeto pequenos problemas foram encontrados, podemos citar o fato de precisar utilizar a biblioteca do FreeRTOS (2022) para o *ESP32*, pois as bibliotecas destinadas ao módulo Arduino, não possuem a opção de especificar qual núcleo será usado para cada tarefa, ele executa as tarefas de modo sequencial, impedindo que duas ou mais tarefas

funcionem ao mesmo tempo de forma paralela, este problema foi resolvido ao usar a biblioteca e criar tarefas em núcleos separados.

Por ser um projeto específico para dosar pequenas porções de alimento, o dosador precisou ser projetado depois de escolher o formato do reservatório onde fica armazenada a ração, o que exigiu desenhar o dosador em um modelo 3D com dimensões em mm, onde a miníma variação de tamanho pode ser a causa do mal encaixe no eixo do motor, fazendo com que ele não movimente o dosador.

4.5 Resultados finais

Ao analisar as características de alguns dos alimentadores existentes no mercado, conforme a Tabela 2, podemos comparar com as características do alimentador projetado neste trabalho. Podemos citar como a principal diferença entre este projeto com os demais existentes, a opção de acionamento a distância. Os alimentadores Magma, Pet Top, Pawise, o acionamento é local e temporizado, ou seja, não possuem acionamento remoto. Outro diferencial oferecido por este projeto é a dupla fonte de energia, na qual a bateria, um *Power bank* com capacidade de duração de 7h e 40min, pode substituir a fonte DC em caso de falta de energia, fazendo com que o microcontrolador *ESP32* não interrompa o seu funcionamento e consequentemente não deixe de distribuir a ração ao gato. Caso o microcontrolador não esteja conectado a Internet, a alimentação funcionará de acordo com uma pré-programação e ao se conectar o usuário é capaz de modifica-lá, mantendo-a armazenada e em funcionamento mesmo que o *ESP32* se desconecte da rede.

Este alimentador automático, Figura 27, foi projetado, desenvolvido e construído, de acordo com o esquemático de ligação mostrado na Figura 28, utilizando dispositivos e tecnologias *Open source*, estrutura acrílica transparente, conectividade e acionamento a distância, assim como interação direta com o usuário, permitindo regrar a alimentação de um gato de estimação.

Tabela 2 – Comparação dos alimentadores existentes no mercado com o BiBiCat

Alimentador	**Pet Top**	**Pawise**	**Playpet**	**Magma**	**BiBiCat**
Armazenamento (kg)	8	0,3	2	2,6	2
N.º refeições ao dia	8	1	20	3	12
Fonte de energia	127V / 220V	Pilhas AA	127V / 220V ou Pilhas D	Pilhas D	127V/220V e Bateria Recarregável
Acionamento remoto	Não	Não	Sim	Não	Sim
Extras	–	Bolsa térmica	–	Mensagem de voz	Funcionamento pré-programado

Figura 27 – Alimentador automático BiBiCat
Fonte: Autoria própria (2022).

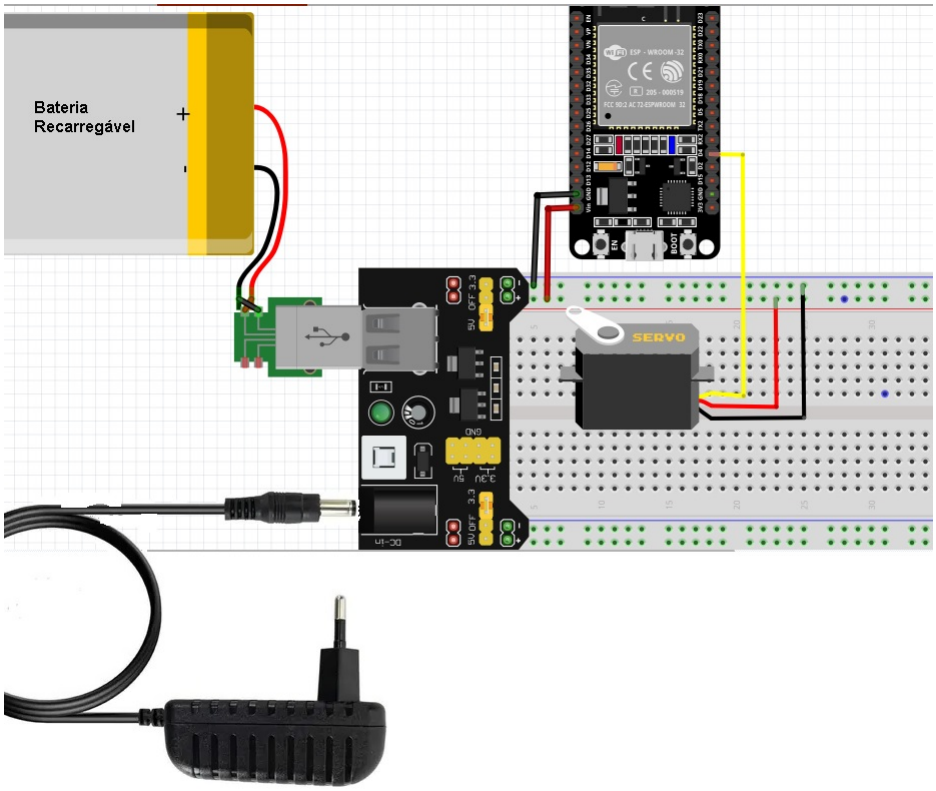


Figura 28 – Esquema de ligação elétrica

Fonte: Autoria própria (2022).

5 CONCLUSÃO

Com o intuito da construção de um Alimentador Automático para gatos controlado por aplicativo Android, acionado remotamente, usando tecnologias *Open source*, tecnologias de IoT, o protocolo MQTT, um microcontrolador *ESP32* e um servomotor para gerenciar e disponibilizar a quantidade correta de alimento a um gato, cuidar de sua saúde e otimizar o tempo de seu tutor.

O presente trabalho aborda o desenvolvimento de um aplicativo para dispositivos móveis com sistema operacional Android conectado a um *broker* público, enviando e recebendo mensagens via protocolo MQTT e um microcontrolador *ESP32* com conectividade *Wi-Fi*, e o desenvolvimento de um controle responsável por ler as mensagens recebidas através do mesmo protocolo de comunicação e conexão ao *broker*, fazer o acionamento do servomotor, afim de regrar a alimentação do gato com pequenas quantidades distribuídas ao longo do dia, com intervalos e número de porções múltiplas de aproximadamente 5 g devido ao tamanho do dosador projetado, podem ser definidas pelo usuário no aplicativo móvel **BiBiCat**, as funções de acionamento foram desenvolvidas conforme a determinação de um tipo e o formato de mensagens recebidas, tendo o seu funcionamento testado e comprovando o funcionamento dentro do que foi proposto inicialmente. Imagens reais do protótipo criado podem ser vistas no Apêndice A.

Por ser um projeto específico para dosar pequenas porções de alimento, de aproximadamente 5g cada, o dosador precisou ser projetado depois de escolher o formato do reservatório onde fica armazenada a ração, o que exigiu desenhar o dosador em um modelo 3D com dimensões em mm, em que a miníma variação de tamanho pode ser a causa do mal encaixe no eixo do motor, fazendo com que ele não movimente o dosador, mas mesmo assim contabilize como uma distribuição ocorrida.

Como trabalhos futuros, pode-se modificar o tipo do dosador, usando os já existentes no mercado, ou projetar um novo formato que se encaixe melhor no eixo do servomotor e usar um formato diferente de reservatório. Visando complementar o que foi implementado neste trabalho, podem ser adicionadas novas funcionalidades, como o controle de vários alimentadores, incluir um histórico mensal de alimentação, para saber quantos quilos de ração o animal consome por mês, uma câmera de monitoramento para ver se realmente naquele intervalo definido o gato vai se alimentar, criar uma maneira de recolher a quantidade de ração que sobrou no comedouro, no caso do gato não comer todo o montante disponibilizado e que não acumule para a próxima refeição e também a possibilidade de enviar mensagens de voz ao liberar o alimento.

REFERÊNCIAS

BANZI, M. *et al.* **What is Arduino?** 2018. Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 16 nov. 2022.

BARRAGAN, H. **What will you do with the W?** 2003. Disponível em: http://wiring.org.co/. Acesso em: 16 nov. 2022.

CAMARGO, V. L. A. D. **Elementos de Automação**. Editora Saraiva, 2014. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536518411/. Acesso em: 16 mar. 2022.

COUNCIL, N. R. **Nutrient requirements of dogs and cats**. [*S.l.*]: National Academy of Science, 2006.

ESPRESSIF. **DataSheet Esp32**. 2019. Disponível em: https://www.alldatasheet.com/datasheet-pdf/pdf/1148023/ESPRESSIF/ESP32.html. Acesso em: 30 out. 2022.

ESPRESSIF. **Espressif**. 2022. Disponível em: https://www.espressif.com/. Acesso em: 25 nov. 2022.

FILIPEFLOP, C. E. **FilipeFlop, Componentes Eletrônicos**. 2022. Disponível em: https://www.filipeflop.com/. Acesso em: 20 nov. 2022.

FREERTOS. **Real-time operating system for microcontrollers**. 2022. Disponível em: https://www.freertos.org/. Acesso em: 16 set. 2022.

FRY, B.; REAS, C. **OverView**. 2003. Disponível em: https://processing.org/overview. Acesso em: 16 nov. 2022.

GOOGLE. **Arquitetura da Plataforma Android**. 2021. Disponível em: https://developer.android.com/guide/platform?hl=pt-br. Acesso em: 17 out. 2022.

GRANDJEAN, D. **Tudo o que você deve saber sobre nutrientes para saúde de cães e gatos**. Paris: Editora Aniwa, 2006.

HIVEMQ. **HiveMQ MQTT Broker**. 2021. Disponível em: https://www.hivemq.com/hivemq/mqtt-broker/. Acesso em: 20 nov. 2022.

JETBRAINS. **IntelliJ IDEA**. 2000. Disponível em: https://www.jetbrains.com/pt-br/idea/. Acesso em: 16 nov. 2022.

JR, S. S.; SILVA, R. A. **Automação e Instrumentação Industrial com Arduino - Teoria e Projetos**. [*S.l.*]: Editora Saraiva, 2015.

LOW, P. **Consciousness in Non-Human Animals**. 2012. Disponível em: https://fcmconference.org/. Acesso em: 17 out. 2022.

LUND, E. *et al.* Prevalence and risk factors for obesity in adult cats from private us veterinary practices. **The International Journal of Applied Research in Veterinary Medicine**, v. 3, n. 2, p. 88–96, 2005.

LUND, E. *et al.* Prevalence and risk factors for obesity in adult dogs from private us veterinary practices. **The International Journal of Applied Research in Veterinary Medicine**, v. 4, n. 2, p. 117–186, 2006.

MASCHIETTO, L. G. *et al.* **Arquitetura e Infraestrutura de IoT**. [*S.l.*]: SAGAH, 2021.

MATARIC, M. J. **Introdução à robótica**. Editora Blucher, 2014. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521208549/. Acesso em: 16 mar. 2022.

MAXTOTAL, C. **Alimentação dos gatos: necessidades nutricionais**. 2021. Disponível em: https://www.maxtotalalimentos.com.br/dica-gato/alimentacao-felina/alimentacao-dos-gatos-necessidades-nutricionais/. Acesso em: 27 out. 2022.

MENEZES, A. R. *et al.* Internet das coisas e os principais protocolos. **Revista Expressão Científica**, v. 2, n. 1, p. 48–49, 2017.

MOURA, L.; CAMARGO, G. **Impacto econômico e social do Android no Brasil**. 2020. Disponível em: https://www.bain.com/pt-br/insights/economic-and-social-impact-of-android-in-brazil/. Acesso em: 15 out. 2022.

MOX, D. **Powerbank Mox**. 2022. Disponível em: https://www.moxdotcell.com.br/. Acesso em: 20 nov. 2022.

OASIS.ORG. **OASIS Message Queuing Telemetry Transport (MQTT) TC**. 2020. Disponível em: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt. Acesso em: 19 nov. 2022.

ORACLE. **O Que é Internet of Things (IoT)**. 2021. Disponível em: https://www.oracle.com/br/internet-of-things/what-is-iot/. Acesso em: 06 out. 2022.

PAWISE. **Dog Timed Feeder**. 2022. Disponível em: https://www.pawisepet.com/pros/9/80.html. Acesso em: 17 out. 2022.

PET, A. M. **Comedouro automático programável Chalesco para cães e gatos**. 2022. Disponível em: https://www.animallemundopet.com.br/comedouro-automatico-programavel-chalesco-para-c-es-e-gatos.html. Acesso em: 17 out. 2022.

PLAYPET. **PlayPet**. 2022. Disponível em: https://www.playpet.io/. Acesso em: 17 out. 2022.

RHINOCEROS. **Rhinoceros 3D**. 2022. Disponível em: https://www.rhino3d.com/. Acesso em: 16 set. 2022.

ROYAL, C. **Mantendo o gato com um peso saudável**. 2021. Disponível em: https://www.royalcanin.com/br/cats/health-and-wellbeing/keeping-your-cat-at-a-healthy-weight. Acesso em: 14 out. 2022.

SAMARTIN, S.; CHANDRA, R. Obesity, overnutrition and the immune system. **Nutrition Research**, v. 21, n. 1-2, p. 243–262, 2001.

SOUZA, F. **Introdução ao Arduino – Primeiros passos na plataforma**. 2013. Disponível em: https://www.embarcados.com.br/arduino-primeiros-passos/. Acesso em: 20 nov. 2022.

SPECIALCAT. **SPECIAL CAT CASTRADOS**. 2021. Disponível em: https://www.specialcat.com.br/produtos-caes/premium/special-cat-castrados. Acesso em: 27 out. 2022.

SUDOESTE, A. **Alimentador Automático Pet Cães Gatos Aves Comedouro Top**. 2012. Disponível em: https://www.alimentadoressudoeste.com.br/MLB-1470322943-alimentador-automatico-pet-ces-gatos-aves-comedouro-top-_JM?searchVariation=52483694516#searchVariation=52483694516&position=3&search_layout=grid&type=item&tracking_id=a779d240-7885-4b9d-b840-9b8b51817daa. Acesso em: 17 out. 2022.

TOWERPRO. **Tower Pro MG90S**. 2014. Disponível em: https://www.towerpro.com.tw/product/mg90s-3/. Acesso em: 18 mar. 2022.

WHISKAS. **Ração seca para gatos adultos castrados WHISKAS**. 2022. Disponível em: https://www.whiskas.com.br/nossos-produtos/seca-castrados. Acesso em: 16 set. 2022.

YUAN, M. **Getting to know MQTT**. 2017. Disponível em: https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/. Acesso em: 19 nov. 2022.

APÊNDICE A – Apêndice A

Figura 29 – Alimentador automático BiBiCat vista lateral
Fonte: Autoria própria (2022).

Figura 30 – Alimentador automático BiBiCat vista de cima
Fonte: Autoria própria (2022).

Figura 31 – Alimentador automático BiBiCat vista dos componentes
Fonte: Autoria própria (2022).

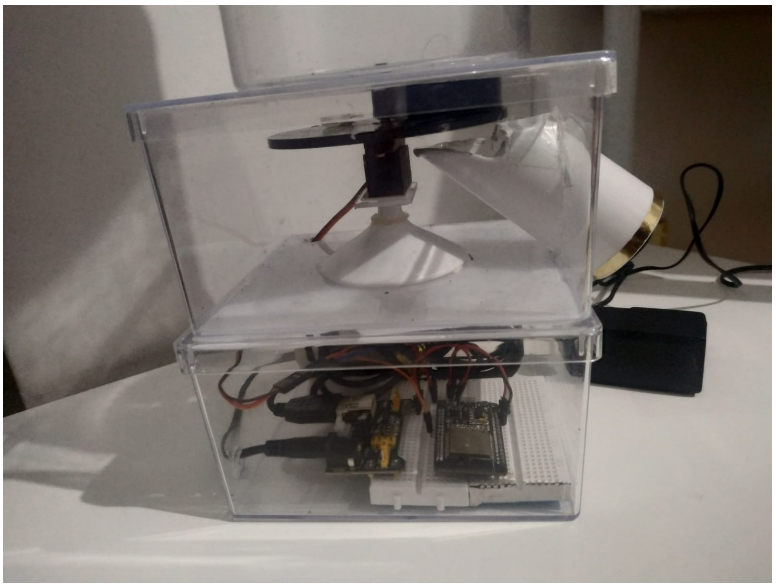


Figura 32 – Alimentador automático BiBiCat vista lateral dos componentes
Fonte: Autoria própria (2022).