# APLICAÇÃO DO CONCEITO DE INTERNET DAS COISAS NO DESENVOLVIMENTO DE UM ALIMENTADOR PARA ANIMAIS DOMÉSTICOS

Thiago Piffer Ramos[1], Valter Blauth Junior[2]

**Resumo**: A criação de animais domésticos pelas famílias é uma prática comum na sociedade. Porém, o descuido com a alimentação dos pets ou a falta de tempo para organizar uma rotina para o fornecimento da ração pode levar ao desenvolvimento de patologias, como a obesidade. A partir disso, foi desenvolvido o protótipo de um comedouro para auxiliar o proprietário na alimentação de seus pets onde, através do desenvolvimento do comedouro, implementou-se o conceito de Internet das Coisas, possibilitando o controle de dosagem de ração para o recipiente através de um aplicativo móvel. Os resultados obtidos foram satisfatórios, onde os dispositivos foram capazes de se comunicar por intermédio de um banco de dados em tempo real, porém, um dos sensores utilizados apresentou leituras inconsistentes.

**Palavras-chave** Internet das coisas. Aplicativo móvel. Saúde animal.

**ABSTRACT:** The domestication of pets by families is a common practice in society. However, neglecting the feeding of pets or lacking time to establish a routine for providing food can lead to the development of pathologies, such as obesity. Based on this, a prototype of a feeder was developed to assist pet owners in feeding their pets. Through the development of the feeder, the concept of the Internet of Things (IoT) was implemented, enabling the control of food dosage to the container through a mobile application. The obtained results were satisfactory, where the devices were able to communicate through a real-time database. However, one of the sensors used presented inconsistent readings.

**Keywords**: Internet of Things. Mobile application. Animal health.

[1] Autor da pesquisa. thiago.piffer@unesc.net
[2] Orientador. valterblauth@unesc.net

## 1 INTRODUÇÃO

A preferência pela adoção e criação de animais de estimação sempre foi consideravelmente comum entre as famílias brasileiras. De acordo com a última edição da Pesquisa Nacional de Saúde (PNS), publicada em 2020 pelo Instituto Brasileiro de Geografia e Estatística (IBGE), cerca de 46,1% dos domicílios brasileiros possuem ao menos um cachorro. A posse destes animais de estimação requer uma considerável responsabilidade por parte dos seus donos, prestando o devido cuidado com a saúde e o bem-estar do animal e evitando o surgimento de patologias.

A obesidade é descrita como uma patologia onde ocorre acúmulo de excessivas quantidades de tecido adiposo no corpo (GERMAN, 2006 apud SILVA et al., 2019). Esta condição afeta também a vida de animais domésticos como cães e gatos e, se não tratada, pode desencadear outras enfermidades tidas como secundárias (FEITOSA et al., 2015 apud SILVA et al., 2019).

A providência de uma alimentação balanceada considerando a condição corporal do animal é importante para a sua saúde, visto que a subestimação desta configura em um fator predisponente para a obesidade (ALVES et al., 2017). Os proprietários que não prestam a devida atenção a estas necessidades contribuem significativamente para o acúmulo de peso em seus animais (MARKWELL & BUTTERWICK, 1994 apud CARCIOFI, 2005).

No Brasil a obesidade é associada ao consumo de ração em conjunto com alimento caseiro e/ou petiscos e ao hábito de deixar a ração sempre disponível ao animal, levando-o a um superconsumo de calorias (CARCIOFI, 2005). De acordo com ALVES (2017), o livre acesso de felinos à comida aumenta as chances de sobrepeso em 13 vezes e as chances de obesidade em seis vezes. Haja vista a estes pontos, é necessário que haja um controle no fornecimento de ração por parte dos proprietários, visando diminuir os riscos de obesidade no animal.

Porém, com as suas ocupações do dia a dia, muitas pessoas não têm o tempo necessário para prover a alimentação de seus pets com o devido cuidado. Em outros casos, estas pessoas enfrentam complicações para se ausentar de suas casas por um longo período (para uma viagem, por exemplo), pois precisam estar presentes em casa para fornecer a ração de seus animais.

Nos tempos atuais, o uso da tecnologia está altamente difundido na sociedade, solucionando os mais diversos problemas que existem. Um dos mais

importantes conceitos tecnológicos que é amplamente usado nos dias de hoje é a Internet das Coisas (*IoT*). De acordo com ROSE et al. (2015, tradução nossa), a Internet das Coisas é um tópico emergente de significância técnica, social e econômica, que promete transformar diversos aspectos da vida das pessoas estando presente em veículos, componentes industriais e outras utilidades, sensores, bens duráveis, produtos de consumo e entre outros objetos do dia a dia.

Em poucas palavras, a Internet das Coisas é uma extensão da Internet atual a objetos cotidianos que possuem capacidades computacionais o suficiente para se conectarem à rede. Com isso, a Internet das Coisas proporciona o controle destes objetos de maneira remota através de qualquer meio, bem como o acesso a estes objetos como provedores de serviços (SANTOS et al., 2016).

A Internet das Coisas ganhou força no cotidiano das pessoas com o uso cada vez mais frequente dos smartphones. Revolucionados pela Apple em 2007 com o lançamento do primeiro iPhone, os *smartphones* passam a ser dispositivos móveis de uso prático e com conexão à internet, abrindo caminho para o desenvolvimento de aplicativos que possibilitam o controle e acesso a dispositivos *IoT*.

O mercado de smartphones se expandiu na sociedade, sendo atualmente dominado por duas plataformas: iOS (*iPhone Operating System*), da Apple, e Android, da Google. Ambas as plataformas possuem diferenças nas tecnologias que devem ser empregadas para o desenvolvimento de aplicativos, sendo necessário que cada desenvolvedor busque conhecer essas tecnologias para então ser capaz de construir um aplicativo que funcione em todos os dispositivos. Uma solução para esta situação foi o advento do desenvolvimento híbrido, que de acordo com CARDOSO (2022), se baseia na programação usando linguagens não nativas (como JavaScript), posteriormente transformando este código em código nativo, ou usar uma *bridge* entre o código não nativo e uma máquina virtual responsável por executar os aplicativos.

Porém, o universo de desenvolvimento de aplicações vai além da aplicação em si. Em muitos casos, é necessário construir API's (*Application Programming Interfaces*) que permitem a comunicação entre a aplicação e um ou mais servidores, seja para realizar a autenticação de um usuário, manipular informações de um banco de dados, guardar imagens ou entre outras utilidades. Desenvolver estes sistemas demanda ainda mais tempo, investimento e mão de obra no processo de construção de um aplicativo. Em 2016, com o objetivo de simplificar esse processo, a Google lançou o Firebase. Trata-se de uma plataforma que une diversos serviços prontos para

uso, como banco de dados, armazenamento de imagens, autenticação, *analytics* e entre outros. A plataforma não substitui a construção de API's próprias, mas serve como um adendo a elas, facilitando o trabalho e salvando tempo (MORONEY, 2017).

Atualmente, já existem projetos que implementam algumas das tecnologias disponíveis no mercado de *IoT* que visam o desenvolvimento de alimentadores automatizados. Um deles é o trabalho desenvolvido por Rodrigues et. al., que visou implementar o protocolo MQTT juntamente com a plataforma *Adafruit.IO* para realizar o controle do alimentador através de uma interface gráfica. O protótipo contou com uma célula de carga para medir o nível da ração através do peso, e também com uma placa ESP8266 para controlar os sensores e se comunicar com a internet.

Outro trabalho que implementou o conceito de Internet das Coisas no desenvolvimento de um alimentador automático é o de Dutra el. al., onde foram usados diversos sensores e componentes eletrônicos para criar um comedouro com reservatórios de ração e água.

Tendo em vista os aspectos apresentados, o presente trabalho tem como objetivo o desenvolvimento de um alimentador automatizado, com acesso à internet e controlado por um aplicativo móvel – o qual também será desenvolvido. Os comandos emitidos a partir do aplicativo serão transmitidos para o alimentador através de um serviço de banco de dados em tempo real onde, por meio desta comunicação entre os dispositivos, implementa-se o conceito de Internet das Coisas. Através do aplicativo, será possível controlar a dosagem de ração para o pote, conferir informações acerca do nível de ração armazenada, do histórico de dosagens realizadas e agendar horários para que o comedouro libere a ração automaticamente.

## 2 MATERIAIS E MÉTODOS

O presente estudo caracteriza-se como uma pesquisa qualitativa aplicada, com caráter explicativo, baseada em revisão bibliográfica. Desenvolveu-se o protótipo de um alimentador automatizado que permite o controle de dosagem de ração para animais domésticos através da Internet das Coisas, bem como um aplicativo móvel conectado com um serviço web, o qual é intermediador entre o aplicativo e o protótipo.

O protótipo é constituído por um circuito responsável pela dosagem da ração, bem como o fornecimento de informações acerca do nível de ração armazenada e de dosagens já realizadas. O circuito é alimentado por uma fonte de

12V conectada a um módulo que diminui a tensão para 5V, sendo essa a tensão de operação da maioria dos componentes usados. Este dispositivo foi conectado a um serviço de banco de dados em tempo real, para onde foram enviadas as informações coletadas a partir dos sensores e de onde foram recebidos os dados referentes a status e agendamentos, onde, a partir destes, são realizadas as operações desejadas. Por fim, foi desenvolvida uma aplicação móvel que permitiu o controle do protótipo através de uma conexão com o mesmo serviço de banco de dados. Por este, foi possível cadastrar o usuário, realizar o login, cadastrar alimentadores através da leitura de um *QRcode* e gerenciar os alimentadores cadastrados, podendo criar agendamentos de dosagens, conferir dados de nível de ração e status, conferir um histórico de dosagens realizadas anteriormente e emitir comandos para dosar a ração.

Na figura 1 é possível compreender a arquitetura do protótipo, onde a aplicação móvel e o dispositivo se comunicam por intermédio de um servidor em nuvem.

Figura 1 – Arquitetura do protótipo


Fonte: Elaborado pelo autor.

## 2.1 CONFIGURAÇÃO DO SERVIÇO DE BANCO DE DADOS

O serviço de banco de dados em tempo real foi configurado através do Firebase, uma plataforma BAAS (*Backend as a Service*) que fornece diversos serviços em nuvem para facilitar o desenvolvimento de *softwares* e sistemas *web*. Dentre os serviços disponibilizados na plataforma, foram utilizados para este projeto o Realtime Database, o Authentication e o Firestore.

O Realtime Database foi o serviço responsável pela comunicação de dados entre o protótipo e a aplicação móvel. Trata-se de um banco de dados em tempo real que permite o armazenamento e estruturação de dados no formato JSON (*JavaScript Object Notation*). A comunicação com este serviço foi feita através de uma API fornecida pelo Firebase onde, por esta, é possível realizar operações de leitura, inserção, atualização e deleção de dados. Também existem recursos para que as aplicações conectadas ao serviço “ouçam” alterações de dados, permitindo que estas realizem operações de acordo com os registros atualizados.

O Authentication é um serviço que permite gerenciar os usuários da aplicação desenvolvida. A configuração de login utilizada foi a de *email* e senha, onde o usuário pode criar uma conta usando credenciais próprias de *email* e senha para a aplicação, em conjunto com outros dados. Através deste serviço, é possível conferir todos os usuários que realizaram algum cadastro na plataforma.

O Firestore foi o serviço utilizado para armazenar os dados de usuário, necessários para a ligação deste ao alimentador. Trata-se de um outro banco de dados *noSQL* cuja estruturação dos dados se faz no formato de documentos, diferindo-se do Realtime Database.

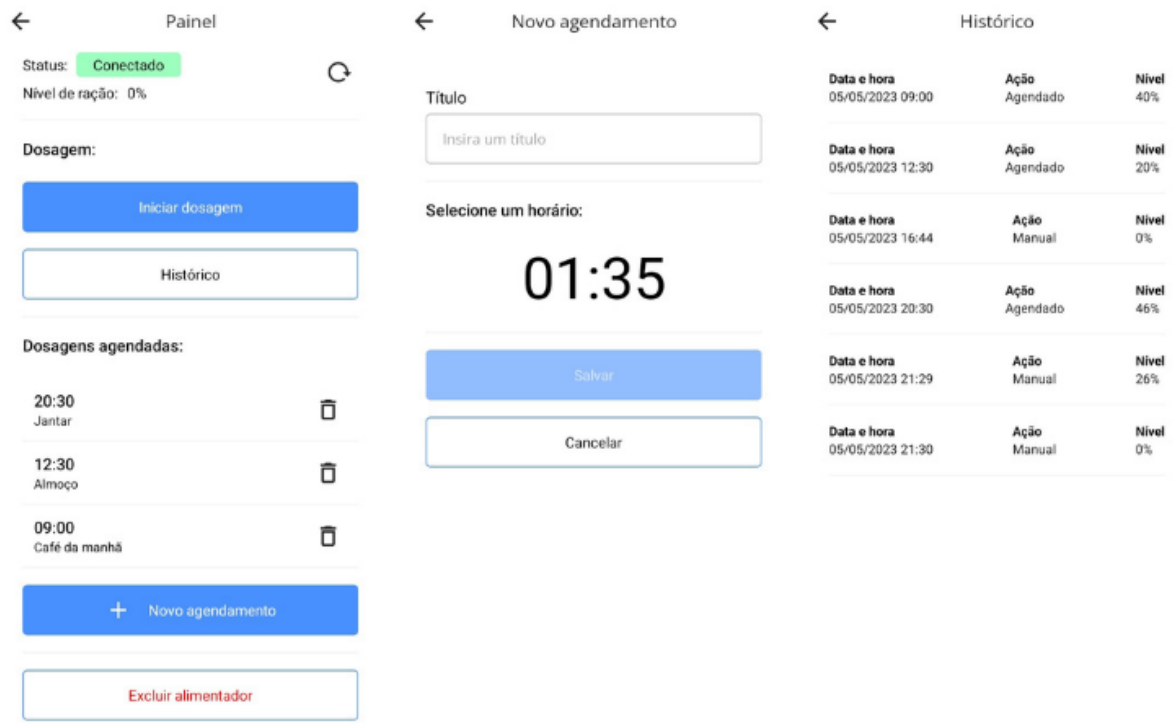
## 2.2 DESENVOLVIMENTO DO APLICATIVO MÓVEL

O aplicativo móvel foi desenvolvido usando o programa Visual Studio Code na versão 1.78.2, por se tratar de um editor de texto versátil com suporte a vários frameworks e linguagens. A linguagem utilizada foi JavaScript na versão ES6 *(Ecma Script 6)* com a biblioteca React Native, a qual permite o desenvolvimento de aplicativos móveis, e usando a plataforma Expo, a qual auxilia no processo de desenvolvimento de aplicações React Native e fornece API’s para facilitar a acesso a recursos nativos do dispositivo móvel, como a câmera, o microfone, o envio de notificações, entre outros. Também foi inclusa a biblioteca Firebase para ter acesso às API’s do serviço e a biblioteca Moment para facilitar os trabalhos com data e hora.

Inicialmente, foram desenvolvidas as telas de *login* e cadastro, onde o usuário pode entrar na aplicação informando seu *email* e senha ou, caso não possua um cadastro, criar uma conta. O usuário autenticado pode gerenciar seus alimentadores através de uma listagem e adicionar novos, através da leitura de um *QRCode*. Acessando um dos alimentadores da lista é aberto o painel de

gerenciamento do alimentador, onde existem informações do nível de ração armazenada no protótipo e o *status* de conexão. Através deste painel, o usuário pode realizar dosagens manuais, atualizar informações do nível de ração caso necessário e conferir agendamentos cadastrados. Também é possível criar agendamentos de dosagem e acessar uma tela contendo o histórico de dosagens já realizadas. Na figura 2 é possível conferir algumas das telas mais importantes do aplicativo.

Figura 2 - Telas do aplicativo




Fonte: Elaborado pelo autor

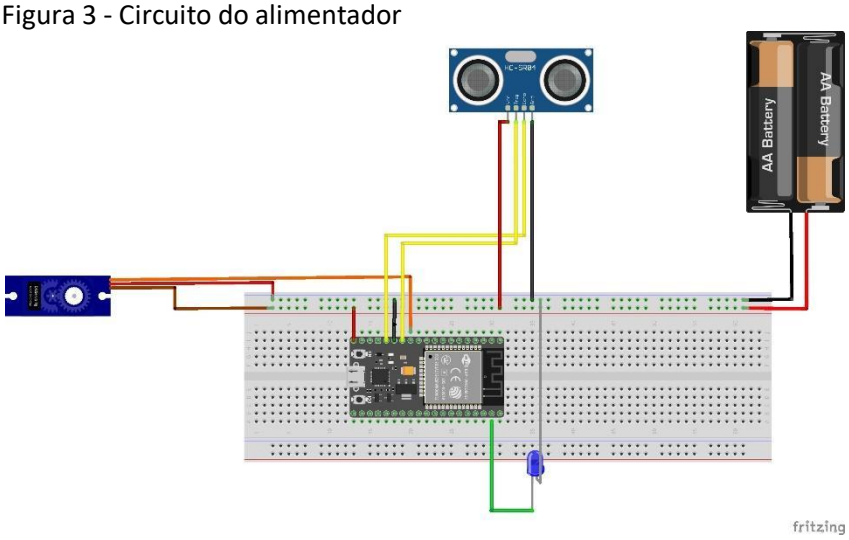
## 2.3 DESENVOLVIMENTO DO PROTÓTIPO

Para desenvolver o protótipo, foi feita a montagem de um circuito usando placas, sensores e realizada a programação de um software embarcado, responsável pelo acionamento e controle de todos os sensores utilizados.

### 2.3.1 Montagem do circuito

A plataforma utilizada para a programação e conexão dos componentes do circuito foi a ESP32, da fabricante Espressif. Este dispositivo possui 11 portas GPIO's (*General Purpose Input/Output*) que permitem o envio e recebimento de sinais de outros sensores, além de suportar conexão com *Bluetooth* e redes *Wi-fi.* A tensão de operação e de saída desta placa é de 3,3V, contendo reguladores internos que permitem a sua alimentação com tensões de 5V. A. O chip também conta com uma antena embutida, 520 KB de memória RAM, um RTC (*Realtime Clock)* próprio, que permite o controle de data e hora do dispositivo, entre outros atributos. Este dispositivo foi conectado a todos os demais componentes e sensores do circuito, conforme esquemática apresentada na figura 3:

Figura 3 - Circuito do alimentador




Fonte: Elaborado pelo autor

As baterias AA que foram usadas na esquemática são meramente ilustrativas, apenas representando a alimentação do circuito. Esta função foi exercida por uma fonte 12V conectada a um módulo conversor de 12V para 5V, o qual alimentou todo o circuito. Também foi omitida uma outra placa conversora de tensão, que foi necessária para converter os 5V de saída do pino ECHO, pertencente ao módulo HC-SR04, para os 3,3V do ESP32. Isto foi feito em virtude da tensão de operação do ESP32, evitando danos ao dispositivo.

O fluxo de ração para a vasilha foi controlado por um micro servo motor SG90, que opera com tensões de 4,8 até 7,2V. Por possuir uma caixa de redução, o motor possui torque o suficiente para rotacionar a tampa do alimentador. O motor também fornece alta precisão nos seus movimentos, permitindo rotações de até 180°.

O sensor de distância ultrassônico HC-SR04 foi utilizado para captação de dados referentes ao nível de ração armazenada. Estas informações são coletadas através da emissão e receptação de ondas ultrassônicas, que permitem calcular a distância entre o sensor e o alvo. O componente teve seus pinos ECHO e TRIGGER conectados aos *IO's* D13 e D12 do ESP32, respectivamente. O pino ECHO é um *output* do sensor que fornece a informação coletada. Já o pino TRIGGER é um *input* usado para acionar o componente. Este módulo opera com uma tensão mínima de 5V e, como mencionado anteriormente, foi necessário um conversor de nível lógico para tornar a saída de 5V do sensor em uma tensão de 3,3V, compatível com a tensão de operação do ESP32. Também foi usado um LED difuso azul, em conjunto com um resistor de 10KΩ para indicar o estado da conexão do dispositivo com a rede w*i-fi*.

### 2.3.2 Programação do *software* embarcado

A programação do *sketch* para o ESP32 foi realizada na IDE Arduino com a versão 1.8.19, sendo esta compatível com o uso do ESP32. A linguagem utilizada foi a C++. Também foi necessário a inclusão de algumas bibliotecas, sendo elas: Firebase ESP32 Client na versão 4.3.9, Wifi na versão 1.2.7, Time e SNTP. Também foram importados os *add-ons* TokenHelper e RTDBHelper.

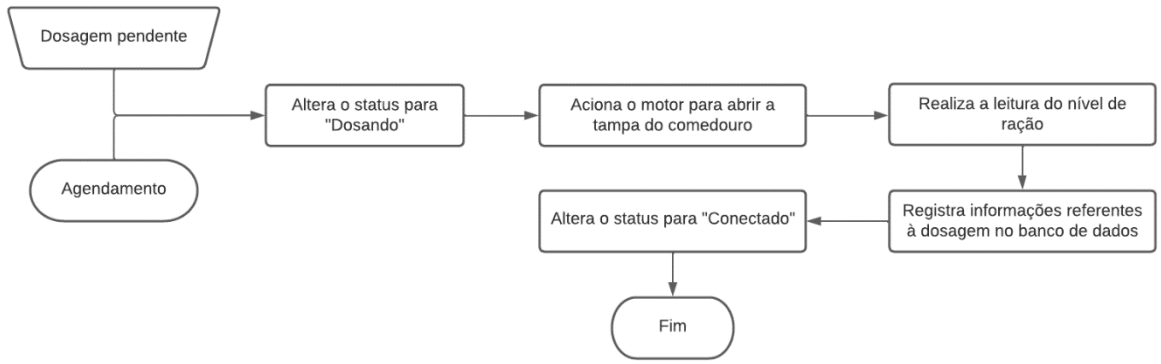
Inicialmente, foram feitas as configurações de todas as dependências do módulo. A biblioteca Wifi foi utilizada para configurar a rede *wi-fi* que será conectada ao ESP32, sendo informados no código os dados de acesso à rede. Ao iniciar este processo o ESP32 envia pulsos ao pino D23 para piscar o LED azul, indicando que ainda não há conexão com a rede. Quando o processo é finalizado, o led azul permanece aceso.

Com a conexão *wi-fi* estabelecida, são configurados dois *listeners* usando a biblioteca Firebase ESP32 Client, em conjunto com os *add-ons* TokenHelper e RTDBHelper. Estes *listeners* estabelecem uma conexão com o serviço Realtime Database, detectando em tempo real qualquer alteração nos registros dos agendamentos e do status. Além de proporcionar uma conexão com os serviços do Firebase, a biblioteca Firebase ESP32 Client também disponibiliza estruturas em JSON que foram utilizadas para a manipulação dos dados. Deste modo, foi possível programar a emissão de comandos a partir do ESP32 de acordo com as informações recebidas.

Por fim, foi utilizada a biblioteca SNTP juntamente com a biblioteca Time para configurar e manipular as informações de data e hora. A biblioteca SNTP permitiu a captação dos dados de data e hora de acordo com o horário de Brasília (GMT -3) a partir de um servidor NTP. Este procedimento foi necessário pois o RTC interno do ESP32 não possui estas informações nativamente, sendo responsável apenas pela atualização destas, mantendo o horário devidamente atualizado. A biblioteca Time permitiu o uso destes dados através da estrutura *timeinfo*.

O processo de dosagem foi configurado para ser ativo sempre que a informação de *status* for alterada para “Dosagem pendente”. No início do processo, o status é alterado para “Dosando”. Após este procedimento, o micro servo é acionado para abrir a tampa do dosador, no qual, decorridos 300 milissegundos, emite-se um outro comando para que o motor feche a tampa e interrompa o fluxo de ração para o recipiente. Ao realizar a dosagem, é emitido um pulso para acionar o módulo HC-SR04, fornecendo informações acerca do nível de ração armazenada. Para finalizar, cria-se um registro no formato JSON com informações de data, hora e nível de ração – o qual é salvo no banco de dados em tempo real – e o status é atualizado para “Conectado”. Este processo pode ser visualizado no fluxograma representado na figura 4.

Figura 4 - Processo de dosagem




Fonte: Elaborado pelo autor

A leitura do nível de ração é realizada com base na distância entre o módulo HC-SR04 e a ração armazenada. Como consta no *datasheet* do componente, para obter esta informação, é necessário acionar o pino TRIGGER do sensor e usar a função *pulseIn* do Arduino no pino ECHO. Esta função retorna o tempo em microssegundos (uS) que aquele pino esteve em nível lógico alto, sendo este relativo

à distância lida pelo componente. Desta maneira, é possível obter a distância (D) em centímetros, conforme equação 1.

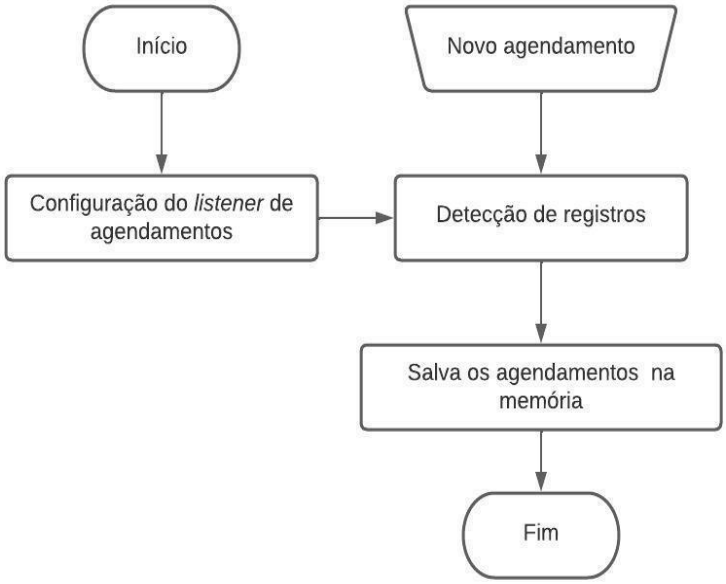
$$D = uS / 58 \quad (1)$$

Obtendo-se a distância (D) em centímetros, o nível (N) de ração armazenada pode ser calculado de acordo com a equação 2, sendo “A” a altura em centímetros do armazenador de ração:

$$N = (A – D) * 100 / A \quad (2)$$

Os agendamentos de dosagens foram salvos em uma lista estruturada, na memória do dispositivo. Cada item da lista contém as informações de hora e minuto em que a dosagem deve ocorrer. Com estas informações é possível iterar sobre esta lista no método *loop*, verificando se algum dos agendamentos corresponde ao horário naquele momento. Quando o horário de algum dos agendamentos é atingido, é iniciado o processo de dosagem de ração. O processo de detecção de alterações nos agendamentos pode ser visualizado no fluxograma representado na figura 5.

Figura 5 - Processo de detecção de agendamentos




Fonte: Elaborado pelo autor

### 2.3.3 Estrutura do protótipo

A estrutura utilizada para a implementação do protótipo foi montada com materiais caseiros. O armazenador do alimentador foi feito com uma garrafa de plástico de aproximadamente 600ml. A garrafa foi posicionada de ponta-cabeça e presa com abraçadeiras a uma estrutura de alumínio adaptada para este uso. A ponta traseira da garrafa foi cortada para permitir que a ração seja inserida nela. Para tampar esta parte foi usado um pedaço de papelão com um recorte que permitiu a inserção do sensor HC-SR04, o qual foi conectado ao circuito através de *jumpers*. Na outra ponta, o motor DC foi posicionado com uma peça de alumínio presa ao seu eixo, sendo esta a tampa do armazenador. O motor foi apoiado em uma caixa de madeira com dimensões 10 x 10 x 5 (em centímetros). Dentro desta caixa, foi inserida a ponte H, conectada ao motor e ao restante do circuito. O circuito foi montado em duas *protoboards* de 830 pontos ligadas entre si, onde foram conectados os demais componentes do protótipo. Este circuito foi inserido em uma caixa de papelão com dimensões 16 x 24 x 9 (em centímetros), e pode ser visualizado na figura 6.

Figura 6 - Protótipo do alimentador


Fonte: Elaborado pelo autor

## 3 RESULTADOS E DISCUSSÃO

O protótipo desenvolvido neste trabalho teve o seu funcionamento avaliado a partir de testes práticos, no período de 24 horas. Com a ração devidamente armazenada, foram cadastrados três agendamentos de dosagem através do aplicativo móvel e realizadas outras três dosagens manuais. Para cada dosagem foi gerado um registro de forma automática, com dados de data e hora de ocorrência da dosagem, permitindo sua conferência através do aplicativo. Após a terceira dosagem o armazenador foi reabastecido, permitindo a continuidade dos testes. Na tabela 1 constam os registros de cada dosagem realizada.

Tabela 1 – Testes realizados

| Data | Hora | Tipo de ação | Nível |
|---|---|---|---|
| 05/06/2023 | 09:00 | Agendado | 40% |
| 05/06/2023 | 12:30 | Agendado | 20% |
| 05/06/2023 | 16:44 | Manual | 0% |
| 05/06/2023 | 20:30 | Agendado | 46% |
| 05/06/2023 | 21:29 | Manual | 26% |
| 05/06/2023 | 21:30 | Manual | 0% |

Fonte: Elaborado pelo autor.

A prototipagem foi feita através da plataforma ESP32, que teve um excelente funcionamento durante o desenvolvimento e testes, com a conexão à rede *wi-fi* operando sem ocorrência de falhas e sem apresentar inconsistências nas saídas lógicas. A conexão com o serviço *web Firebase* a partir do protótipo através de suas bibliotecas também não encontrou problemas, sendo possível detectar qualquer mudança na base de dados em tempo real e realizar ações de acordo com os dados recebidos. As requisições ao serviço NTP para aquisição de dados atualizados de data e hora também funcionaram plenamente, sem apresentar erros, permitindo o acesso ao horário no fluxo de agendamento.

O motor utilizado para implementar a tampa do armazenador do comedouro foi um microservo motor, modelo SG90. Foi necessário fixar uma peça de alumínio ao eixo do motor para servir como tampa do armazenador, suportando o peso da ração armazenada. Este componente mostrou-se ideal para esta aplicação e funcionou corretamente em todo o desenvolvimento. A capacidade de controlar a posição exata do motor foi essencial para garantir que a tampa manipule o fluxo de ração

adequadamente. Este controle também pode ser realizado por um motor de passo, conforme implementado no protótipo desenvolvido por Dutra et. al., pois esse componente oferece movimentos ainda mais precisos.

Para controlar o nível de ração presente no armazenador, foi usado o sensor ultrassônico HC-SR04. A utilização deste componente possibilitou a leitura da distância até a ração armazenada. No entanto, seu funcionamento não foi totalmente eficiente, apresentando ocasionalmente valores inconsistentes. A estrutura do armazenador pode ser apontada como uma possível causa dessas inconsistências, uma vez que, por ser constituída de uma garrafa de plástico, interferiu nas ondas ultrassônicas emitidas pelo sensor. Esta imprecisão pode ser visualizada na tabela 2, onde, ao esvaziar o armazenador, foi apontado um nível de 13%.

Tabela 2 – Dados inconsistentes apresentados pelo sensor ultrassônico

| Data | Hora | Tipo de ação | Nível |
|---|---|---|---|
| 19/05/2023 | 23:45 | Manual | 26% |
| 19/05/2023 | 23:45 | Manual | 20% |
| 19/05/2023 | 23:45 | Manual | 13% |
| 19/05/2023 | 23:45 | Manual | 6% |
| 19/05/2023 | 23:46 | Manual | 13% |

Fonte: Elaborado pelo autor.

O aplicativo móvel desenvolvido demonstrou ter um pleno funcionamento quanto às suas funções. A conexão do aplicativo com o Firebase ocorreu sem apresentar falhas, sendo possível criar contas, autenticá-las e realizar operações nas bases de dados Firestore e Realtime Database. Também foi possível identificar o alimentador através de um *QRCode*, listar o alimentador cadastrado, acessar seu painel de controle, verificar seu nível e *status*, atualizar a informação de nível de ração, verificar seu histórico de dosagens, realizar dosagens pontuais, listar, excluir e criar agendamentos de dosagens. A aplicação foi testada em um *smartphone* Samsung Galaxy A21S com sistema Android na versão 12, e foi desenvolvida na linguagem Javascript juntamente com a biblioteca React Native e a plataforma Expo. Na figura 7, é possível conferir a tela do painel do alimentador.

Figura 7 – Painel do alimentador


Fonte: Elaborado pelo autor

Para o intermédio entre o protótipo e o aplicativo móvel, utilizou-se o serviço Firebase, da *Google,* que funcionou perfeitamente em todas as operações realizadas a partir do aplicativo móvel e do protótipo. O uso do Firebase em conjunto com o desenvolvimento de um aplicativo móvel abre uma quantidade maior de funcionalidades que podem ser implementadas no alimentador, diferente da plataforma *Adafruit.IO,* utilizada no projeto de Rodrigues et. al., onde existem limitações quanto à interface gráfica e aos limites do plano gratuito.

## 4 CONCLUSÃO

No projeto implementado a partir desta pesquisa, constatou-se que o protótipo desenvolvido teve um bom desempenho de suas funções, cumprindo os objetivos propostos. Através do intermédio do serviço *web*, foi possível implementar o conceito de Internet das Coisas através da troca de informações em tempo real entre

o dispositivo móvel e o alimentador. O uso da plataforma ESP32 foi de extrema importância para o funcionamento do protótipo, sendo possível controlar os sensores com facilidade e rapidez. A aplicação móvel também demonstrou ser eficiente e intuitiva, com uma aparência *clean* que facilitou o seu uso.

Com este protótipo, é possível controlar a alimentação do pet com facilidade. A aplicação do conceito de Internet das Coisas permite que o dispositivo receba comandos emitidos a partir de qualquer lugar. Desta forma, o dono pode se ausentar de sua casa para os seus afazeres e controlar o dispositivo, seja com agendamentos de dosagem ou com dosagens pontuais.

O uso do serviço Firebase facilitou a integração do protótipo com o aplicativo móvel, sendo esta uma ferramenta potente para a implementação do conceito de Internet das Coisas com o fornecimento de informações em tempo real. O uso do aplicativo móvel foi essencial para permitir o controle do dispositivo. Com a sua integração ao Firebase e sendo possível ser usado a partir de um *smartphone*, a aplicação foi intuitiva e potente no exercício de suas funções.

No desenvolvimento deste projeto, foram identificados alguns problemas referentes ao sensor de distância. Ao realizar a leitura de nível de ração, obteve-se alguns valores inconsistentes com o nível real de ração. A consistência desta informação é importante para verificar se a ração foi liberada corretamente. Por isso, foi criada uma tratativa via programação, disponibilizando um botão para atualizar o nível através do aplicativo.

Para os próximos trabalhos, sugere-se a criação de uma estrutura mais robusta para o alimentador, impressa em 3D, a fim de facilitar seu desenvolvimento, manuseio e evitar interferências com outros sensores. Também é sugerida a utilização de outros componentes para a verificação do nível de ração armazenada, como células de carga para medição de peso e sensores de distância a laser para a coleta precisa do nível da ração.

## REFERÊNCIAS


ALVES, R. S.; BARBOSA, R. C. C.; GHEREN, M. W.; DA SILVA, L.; DE SOUZA, H. J. M. Frequency and risk factors of obesity in a population of domestic cats from Rio de Janeiro. **Brazilian Journal of Veterinary Medicine**, *[S. l.]*, v. 39, n. 1, p. 33–45, 2018. DOI: 10.29374/2527-2179.bjvm0081. Disponível em: https://bjvm.org.br/BJVM/article/view/81. Acesso em: 16 maio 2023.

CARCIOFI, Aulus. **Obesidade e suas conseqüências metabólicas e inflamatórias em cães e gatos**. Jaboticabal: UNESP, [S.D.]. Disponível em https://www.fcav.unesp.br/Home/departamentos/clinicacv/AULUSCAVALIERICARCIOFI/obesidade-texto.pdf

CARDOSO, Rodrigo. **Desenvolvimento híbrido de aplicativos:** frameworks e bibliotecas. Locaweb. [S.I.] 2022.Disponível em: <https://blog.locaweb.com.br/temas/codigo-aberto/desenvolvimento-hibrido/>. Acesso em: 18 nov. 2022.

DUTRA, André L. P. et al. Feeder4Pets: Alimentador Automático para Animais Utilizando a Plataforma Arduino. Tecnologia e Sustentabilidade Ceuma, v. 34, n. 2. Disponível em: http://www.ceuma.br/portalderevistas/index.php/RCCP/article/view/452. Acesso em: 30 set. 2022.

IBGE - INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Pesquisa Nacional de Saúde:** Informações sobre domicílios, acesso e utilização dos serviços de saúde. Rio de Janeiro: IBGE, 2020. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv101748.pdf>. Acesso em: 29 set. 2022.

MORONEY, Laurence. **The Definitive Guide to Firebase**: build android apps on google's mobile platform. Seattle: Apress Berkeley, Ca, 2017.

PEREIRA DE SOUZA SILVA, L.; HOOG NORA JÚNIOR, R. C.; CARLOS PEREIRA, C. M. .; PEREIRA BERNARDINO, V. M. . Manejo nutricional para cães e gatos obesos. **Pubvet**, *[S. l.]*, v. 13, n. 05, 2019. DOI: 10.31533/pubvet.v13n5a339.1-12. Disponível em: https://ojs.pubvet.com.br/index.php/revista/article/view/831. Acesso em: 16 maio 2023.

RODRIGUES, J. Marcelo De Lima; SOUZA, I.; JÚNIOR, Q. PROJETO IOT APLICADO À CONSTRUÇÃO DE UM ALIMENTADOR AUTOMÁTICO PARA ANIMAIS DOMÉSTICOS. Disponível em: https://repositorio.ufersa.edu.br/handle/prefix/4654. Acesso em: 29 set. 2022.

ROSE, Karen; ELDRIDGE, Scott; CHAPIN, Lyman. **THE INTERNET OF THINGS: AN OVERVIEW**: understanding the issues and challenges of a more connected world. [S. l]: Internet Society, 2015. Disponível em: <https://www.internetsociety.org/wp-content/uploads/2017/08/ISOC-IoT-Overview-20151221-en.pdf>. Acesso em: 25 out. 2022.

SANTOS, Bruno P. *et al*. Internet das Coisas: da teoria à prática. In: SIQUEIRA, Frank Augusto *et al*. **Livro de Minicursos SBRC 2016**. Salvador: Sociedade Brasileira de Computação (Sbc), 2016. p. 1-50. Disponível em: <https://bps90.github.io/papers/Internet-das-coisas-da-teroria-a-pratica>. Acesso em: 17 nov. 2022.