GABRIEL CORRADINI DA CUNHA

ALIMENTADOR AUTOMÁTICO PARA ANIMAIS DOMÉSTICOS CONTROLADO VIA APLICATIVO

CATANDUVA

2021

GABRIEL CORRADINI DA CUNHA

ALIMENTADOR AUTOMÁTICO PARA ANIMAIS DOMÉSTICOS CONTROLADO VIA APLICATIVO

Projeto de Trabalho de Conclusão de Curso apresentado ao IFSP - Câmpus Catanduva como requisito básico para a conclusão do curso de Engenharia de Controle e Automação

Orientador: PAULO CESAR MIORALLI

CATANDUVA

2021

Elaborada pela biblioteca do IFSP Câmpus Catanduva
com dados fornecidos pelo autor.

Cunha, Gabriel Corradini da

C972a

Alimentador automático para animais domésticos controlado via aplicativo / Gabriel Corradini da Cunha. – Catanduva, SP: IFSP, 2021.
51 f.

Orientador: Paulo Cesar Mioralli

Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) -- Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Catanduva, 2021.

1. Comedouro. 2. Automação. 3. Internet das coisas. I. Mioralli, Paulo Cesar. (Orient.). II. Título.

CDD (23. ed.): 629.895

GABRIEL CORRADINI DA CUNHA

ALIMENTADOR AUTOMÁTICO PARA ANIMAIS DOMÉSTICOS CONTROLADO VIA APLICATIVO

Projeto de Trabalho de Conclusão de Curso apresentado ao IFSP - Câmpus Catanduva como requisito básico para a conclusão do curso de Engenharia de Controle e Automação

Orientador: Paulo Cesar Mioralli

PAULO CESAR MIORALLI

Instituto Federal de São Paulo – câmpus Catanduva

JAIR DE MARTIN JUNIOR

Instituto Federal de São Paulo – câmpus Catanduva

ALDO MARCEL YOSHIDA RIGATTI

Instituto Federal de São Paulo – câmpus Catanduva

Catanduva - SP, ___ de _____ de 2021.

RESUMO

Com o crescimento da quantidade de animais de estimação, a importância da alimentação deles só aumenta. Os comedouros ou alimentadores automáticos se destacam como solução para controlar e balancear a alimentação dos pets. Atualmente, os animais domésticos atingiram um novo estrato cultural: estão cada vez humanizados e são considerados membros da família. Entretanto, com a rotina das famílias na sociedade contemporânea, na qual as pessoas estão cada vez mais atarefadas e, muitas vezes, necessitam passar um certo tempo ausentes de casa, os proprietários de animais domésticos vêm sendo cada vez menos cuidadosos no que se diz respeito à alimentação de seus pets. Igualmente aos seres humanos, os animais necessitam de uma alimentação balanceada, variando de acordo com seu porte físico e espécie. Diante desse problema, o presente trabalho tem como proposta a confecção de um protótipo para alimentação de pets (ração e água), junto com uma aplicação IoT, funcionando à distância, de forma que o dono do animal possa verificar tudo em tempo real, tendo o controle da quantidade e os horários de abastecimento dos alimentos, com as ações de controle efetuadas a partir de um computador ou *smartphone* com e acesso à *internet*. O protótipo confeccionado atendeu aos objetivos estipulados no projeto, constituindo-se em um dispositivo funcional de controle à distância que facilita o cotidiano do usuário quanto à alimentação de seus pets.

Palavras-chave: comedouro; automação; internet das coisas.

ABSTRACT

With the growth of the number of pets, the importance of feeding them only increases. Automatic feeders or feeders stand out as a solution to control and balance the feeding of pets. Currently, domestic animals have reached a new cultural stratum: they are increasingly humanized and are considered members of the family. However, with the routine of families in contemporary society, in which people are increasingly busy and often need to spend some time away from home, pet owners have been less and less careful about to feed your pets. Like humans, animals need a balanced diet, varying according to their physical size and species. Faced with this problem, the present work proposes the making of a prototype for feeding pets (food and water), together with an IoT application, working remotely, so that the pet owner can check everything in real time, having control of the quantity and timing of food supply, with control actions carried out from a computer or smartphone with internet access. The prototype made met the objectives stipulated in the project, constituting a functional remote control device that facilitates the user's daily life in terms of feeding their pets.

Keywords: feeder; automation; internet of things.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

Cresce a cada ano, no Brasil, a quantidade de animais de estimação. Segundo o IBGE, em 2013, esse número alcançava em torno 132,4 milhões, dos quais 52,2 milhões representavam a população de cães e 22,1 milhões, a de gatos. Atualmente, os animais domésticos atingiram um novo estatuto cultural: estão cada vez mais "antropomorfizados", são considerados membros da família e tratados, em muitos casos, como crianças, servindo como companhia e terapia para os humanos (PASTORI; MATOS, 2015). Entretanto, a rotina das famílias na sociedade contemporânea, na qual as pessoas estão cada vez mais atarefadas e, muitas vezes, necessitam passar um certo tempo ausentes de casa, tem feito com que os proprietários de animais domésticos sejam cada vez menos cuidadosos no que diz respeito à alimentação de seus pets. Devido a isso, a saúde nutricional dos animais pode ser afetada pelo excesso ou falta de nutrientes em sua dieta, resultando em graves consequências, como patologias (obesidade e desnutrição) e suas complicações (ALVES, 2020).

Igualmente aos seres humanos, os animais necessitam de uma alimentação balanceada, variando de acordo com seu porte físico e espécie. Ela é de suma importância para que o animal tenha uma vida saudável, mas infelizmente essa questão pode ser negligenciada pelos donos dos pets. Muitas pessoas, por exemplo, oferecem comida destinada a seres humanos para seus animais de estimação, podendo acarretar doenças, devido à carência ou excesso de nutrientes. Segundo Grandjean (2006), outro erro comum que pode afetar também a saúde dos animais, é não estabelecer um limite na alimentação, oferecendo uma quantidade diária muito maior ou menor que a recomendada. Pois uma boa parte dos animais não possuem autocontrole e caso um limite não seja estabelecido, eles continuarão ingerindo alimento enquanto estiver ao seu alcance.

Diante de todos esses problemas, é possível ter como solução a tecnologia e a automação, mais especificamente, a automação residencial. Com a crescente utilização da automação em benefício da otimização das atividades humanas e, consequentemente, de tudo aquilo que as cercam, houve uma redução da maioria dos problemas do dia a dia que, antes, eram considerados irremediáveis (FIRMINO; MATEUS, 2020). Uma boa alternativa para amenizar e até mesmo resolver o problema da alimentação dos pets, seria a utilização de um dispositivo eletrônico e mecânico, capaz de fornecer e controlar, de forma balanceada, a quantidade de alimento adequada para o animal de estimação, de

modo que o pet se alimente corretamente, sem excessos ou carências, que prejudiquem a sua saúde e qualidade de vida.

E esse dispositivo pode ser um alimentador controlado, ou melhor, um alimentador automático. Com o avanço contínuo da tecnologia, esse tipo de dispositivo vem sendo cada vez mais utilizado pelos donos de pets, desde o mais simples, sem controle ou automação, até o totalmente automático. O uso de um alimentador pode fornecer, para o dono, a possiblidade de sair de casa sem grandes preocupações com a alimentação de seu pet e até realizar viagens curtas, dependendo do tipo de alimentador, além de também poder fornecer uma alimentação mais balanceada e saudável para o animal.

1.1. Objetivos

1.1.1. Objetivo Geral

Este trabalho tem como objetivo geral desenvolver um protótipo de um alimentador automático para pets controlado via aplicativo.

1.1.2. Objetivos Específicos

- Construir a estrutura do alimentador, com uma vasilha de ração e outra de água, e seus respectivos reservatórios;
- Montar circuitos eletrônicos para medição (sensores), controle (microcontrolador) e acionamento (válvula e motor de passo);
- Desenvolver um programa para medição, controle e acionamento;
- Desenvolver um aplicativo com uma interface, para que o usuário possa ter todas as informações e realizar as ações necessárias do alimentador.

1.2. O Presente Trabalho

Através de pesquisas realizadas sobre trabalhos relacionados a alimentadores automáticos, constatou-se que, na maioria deles, o usuário necessita estar relativamente perto do equipamento para ter acesso às informações e realizar o controle. A comunicação via *bluetooth*, por exemplo, é muito utilizada, porém com ela existe a limitação do raio de alcance do seu rádio. E mesmo quando os equipamentos são comerciais, é um pouco

difícil encontrar algum que o usuário tenha acesso de qualquer lugar. E quando é encontrado, o alimentador tem preços muito elevados, tornando-o inacessível para muita gente, além dos equipamentos, geralmente, focarem em só um recipiente de alimentação, tendo que escolher entre água e ração.

Visando contribuir para a minimização desses problemas, como diferencial, o presente projeto tem a intenção de implementar o acesso remoto em um alimentador, através de uma aplicação IoT (*Internet of Things*) que estará na nuvem, podendo ser acessado de qualquer lugar. Sendo assim, este trabalho tem como proposta a confecção de um protótipo para alimentação de pet (ração e água), com o menor custo possível, que funcione à distância, de forma que o dono do animal possa verificar tudo em tempo real, tendo o controle da quantidade e os horários de abastecimento dos alimentos, com as ações de controle efetuadas a partir de um *smartphone* com *Android* e acesso à *internet*. É importante ressaltar que o alimentador deste projeto é recomendado para animais de pequeno porte, principalmente cães e gatos, por conta da estrutura do equipamento, que possui os recipientes em uma altura baixa, sendo prejudicial para a coluna e até a digestão dos animais de grande porte.

2. REVISÃO DE LITERATURA

2.1. Mercado Pet e Público-Alvo

O ser humano tem o domínio dos animais domesticados a milhares de anos, pois eles fornecem aos humanos muitos benefícios, como afetividade, proteção, guarda e até entretenimento (FIRMINO; MATEUS, 2020). E o mercado dos animais domésticos teve uma grande crescente de consumo de produtos e serviços nos últimos anos. A partir dos anos 1990, os donos de pets e consumidores passaram a mudar a forma de como tratavam os animais e a maneira de se relacionar com eles (MOURA, 2013).

Essa mudança de comportamento dos donos dos bichos, segundo Poli (2017), acontece com os animais estando cada vez mais com o *status* de membros da família, com o aumento no número de lares com só uma pessoa, famílias tendo filhos cada vez mais tarde, entre outros diversos fatores, que acabam gerando a necessidade de uma companhia animal. Com os pets cada vez mais humanizados, o mercado cresceu radicalmente e tende a crescer ainda mais.

Em 2018, no Brasil, o setor de animais domésticos movimentou cerca de 20 bilhões de reais, sendo quase 10% a mais do que no ano anterior, tornando o segundo maior mercado de pets do mundo, atrás apenas dos Estados Unidos (SILVEIRA, 2019). E nesse mercado, existem diversos produtos relacionados à saúde, higiene e, principalmente, à alimentação dos animais. Deixar o animal saudável e com uma boa aparência só é possível com uma correta e balanceada alimentação, fornecendo todos os nutrientes que o animal necessita.

Após pesquisas e estudos referentes ao mercado dos pets, constatou-se que público-alvo do mercado se concentra no público adulto, não podendo ter uma faixa etária fixa, com dois extremos, mas sim com apenas uma idade mínima, que seria em torno de 24 anos. Essa faixa de idade, segundo Ulrich e Novak (2012), tem um número grande de consumidores, pois para uma compra regular dos produtos, como um alimentador, por exemplo, necessita-se ter uma estabilidade financeira. Estudantes ou jovens que não possuem uma fonte segura de renda teriam certas dificuldades para obter o produto (protótipo) que será desenvolvido.

Um público-alvo também muito interessante, fugindo do fator idade, são as pessoas que costumam ficar longe de casa, seja para trabalhar ou até mesmo por realizar muitas viagens. A ausência dos donos dos pets causa enormes preocupações, além de gerar muitos gastos com cuidadores de animais ou com algo muito comum hoje em dia, que é deixar os animais em hotéis para pets.

2.2. Alimentação de Cães e Gatos

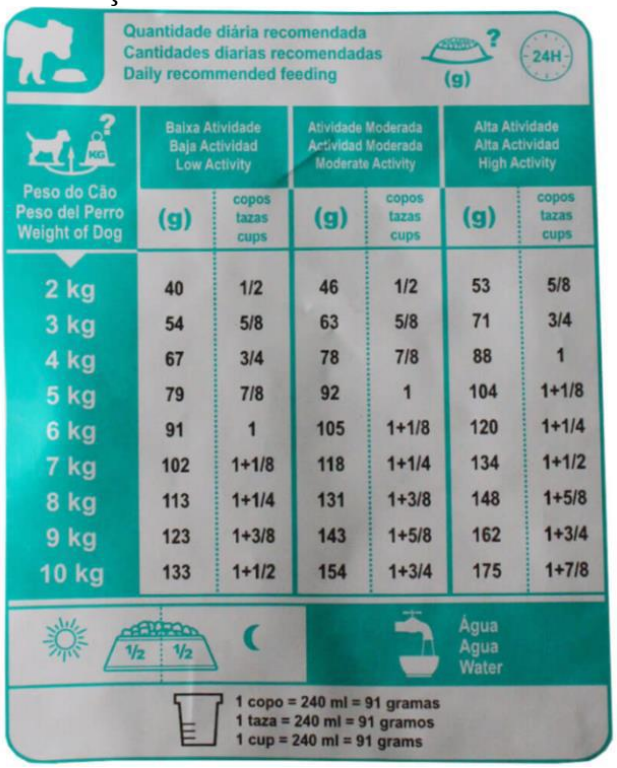
Pensando em uma boa alimentação dos animais, segundo Ruiz (2013), o dono necessita saber a quantidade de ração que deve ser dada por dia. Os cães, por exemplo, precisam comer a ração de maneira fracionada e, de preferência, em horários fixos, com uma recomendação entre duas e três refeições por dia. Ignorar esses fatores podem acarretar reações ruins, a médio e longo prazo. Para toda essa organização, é importante considerar o tipo de animal, a sua idade, sua massa e até o seu nível de atividade física, sempre com a ajuda de um profissional qualificado. Porém, com algumas definições já conhecidas, é possível entender um pouco melhor a necessidade alimentícia de cada tipo de animal (ULRICH; NOVAK, 2012).

Uma recomendação importante é sempre utilizar a ração produzida pelas empresas de nutrição de cães e gatos, pois elas produzem alimentos certificados, balanceados, de

acordo com as características dos animais, além de ser muito prático. Diferente do que muitos acham, existe a possiblidade de oferecer alimentação caseira, porém, é preciso, obrigatoriamente, que seja feito com supervisão e recomendação de um veterinário (MUNDO VETERINÁRIO, 2018).

A nutrição de um cão varia muito de acordo com marca da ração. Por conta disso, é importante sempre ler as embalagens, para ter uma noção dos nutrientes. Como já mencionado, fatores muito importantes, são o tamanho ou massa do animal e a atividade física que ele realiza diariamente, pois quanto maior o porte de um cachorro, maior será sua necessidade energética (ALVES, 2020). Na Figura 1 pode-se ver exemplos da quantidade de comida, de acordo com a massa e atividade física do animal.

Figura 1. Alimentação de acordo com massa e atividade física do cão.



Fonte: Geração Pet (2021).

Para os gatos, as recomendações não são muito diferentes. Segundo Martins (2005), mesmo que eles gostem de comer praticamente o dia inteiro, não é recomendável deixar os gatos comerem sem limite algum. Por isso, é extremamente importante também ter uma quantidade diária de alimento definida, de preferência, com horários fracionados, assim como para os cães. A Figura 2 apresenta uma tabela semelhante à dos cães, só que para gatos.

Figura 2. Alimentação de acordo com massa e atividade física do gato.

Quantidade diária recomendada						
Peso do gato (kg)	Baixa atividade		Atividade normal		Ativo	
	g	copo*	g	copo*	g	copo*
2	25	1/2	30	1/2	35	1/2
3	35	1/2	40	1/2	50	1/2
4	40	1/2	50	1/2	60	1/2
5	45	1/2	60	1/2	70	1
6	55	1/2	65	1	80	1
7	60	1/2	75	1	90	1
8	65	1	85	1	200	1
9	70	1	90	1	110	1
10	75	1	100	1	115	1 1/2

*1 copo 240 ml = 94 gramas

Fonte: PetlandBrasil (2021).

2.3. Alimentadores Comerciais

Assim como tudo que possui tecnologia envolvida, o alimentador para pets está continuamente evoluindo, se tornando cada vez mais prático para o consumidor. Porém, os primeiros alimentadores com uma certa automação ou tecnologia não possuíam ou possuem (é preciso situar também no presente, pois boa parte deles ainda existe no mercado) toda a praticidade e recursos para o dono de pet.

Os primeiros alimentadores ou comedouros “automáticos” (Figura 3), que não possuem absolutamente nada eletrônico, são aqueles que armazenam grandes quantidades de comida em reservatórios, no qual, conforme o animal se alimenta, a ração ou água desce do reservatório para a vasilha por meio da própria gravidade, proporcionando que o recipiente esteja sempre cheio. E apesar desse tipo de alimentador não ter controle algum da quantidade de alimento, ele acaba sendo um produto bem atrativo, por conta do seu baixo preço.

Figura 3. Comedouro automático PetDog.

Fonte: Royalpets (2020).

Um passo a mais que o alimentador anterior, no fator de automatização, são os alimentadores chamados de semiautomáticos. São alimentadores eletrônicos ou não, que necessitam de um comando do dono do animal no aparelho, apertando um botão do alimentador, por exemplo, para que seja liberado água ou ração. A Figura 4 apresenta um exemplo.

Figura 4. Comedouro com botão Vida Mansa.

Fonte: Petlove (2020).

Já no mercado atual, há uma grande variedade de alimentadores, com poucas ou diversas funções. Um tipo bastante requisitado são os alimentadores programáveis, que possuem horários predefinidos para as alimentações. Porém, geralmente, estes alimentadores possuem apenas um recipiente, tendo que escolher entre água e ração, além de não ter a possibilidade de saber como está o recipiente, se o animal está comendo ou não, não tendo o controle de sobras dos alimentos.

Por fim, é possível encontrar diversos alimentadores automáticos e inteligentes no mercado, tendo muitas funções que tornam tudo mais fácil para o usuário, porém, os custos são muito elevados, havendo produtos com valores superiores a R$ 1.000,00 facilmente. Na Figura 5 há um desses exemplos, um alimentador muito moderno com diversas funções, inclusive com aplicativo integrado para controlar remotamente.

Figura 5. Kit comedouro pet automático KaBuM.

Fonte: KabuM (2021).

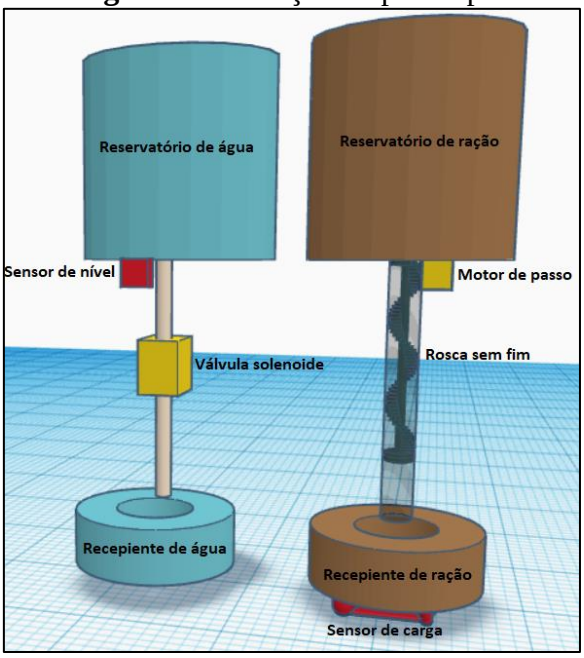
3. MATERIAIS E MÉTODOS

3.1. O Protótipo

O protótipo desenvolvido tem uma estrutura com dois recipientes, um para água e outro para ração. Sob o recipiente de água está um sensor ultrassônico apontando para a água, para realizar a medida de nível. Abaixo da vasilha de ração há uma célula de carga, realizando a coleta de dados de massa. E tanto o sensor ultrassônico quanto a célula de carga enviam os dados para serem processados pelo microcontrolador.

Para o acionamento, há uma válvula solenoide para o controle da água e um motor de passo com uma rosca sem fim para o controle da ração. A válvula e o motor de passo, que liberam a passagem da água e da ração dos reservatórios, são acionados pelo microcontrolador, que é controlado pelo usuário, por meio do aplicativo. A Figura 6 ilustra o protótipo desenvolvido neste trabalho.

Figura 6. Ilustração do protótipo.

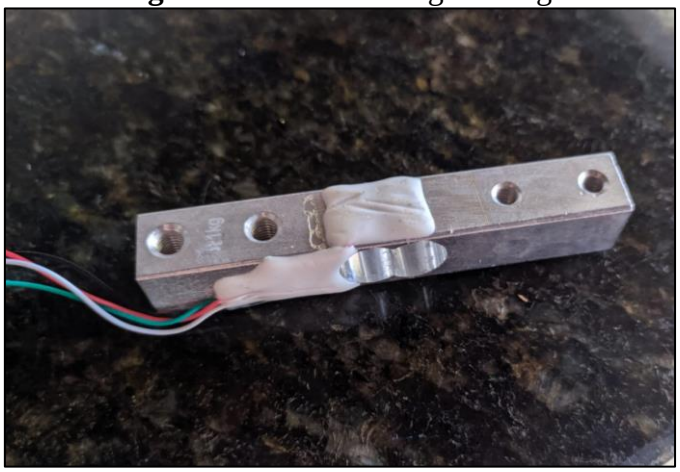


Fonte: Autor.

3.2. Sensor de Carga

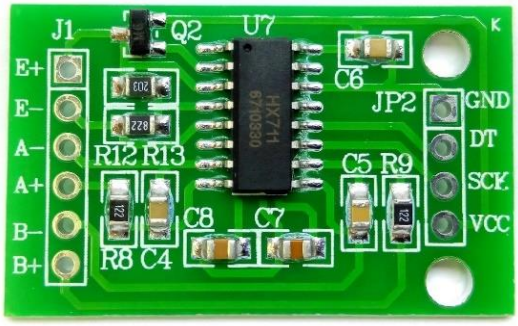
O sensor de carga ou célula de carga utilizado (Figura 7) é de 1 quilograma (kg), com tensão de operação recomendada de 3 a 12 Volts. Ele é posicionado abaixo do recipiente de ração e é a partir dele que será realizado o cálculo da massa. O sensor coleta os dados, que posteriormente passam pelo módulo HX711 (Figura 8), que é um circuito de amplificação e conversão. O sinal digital amplificado chega no microcontrolador, que converte o sinal de tensão em gramas.

Figura 7. Célula de Carga de 1kg.



Fonte: Autor.

Figura 8. Módulo HX711.



Fonte: Arduino e Eletrônica (2021).

3.3. Sensor de Nível

O sensor de nível é um módulo de sensor ultrassônico HC-SR04 posicionado sob o recipiente de água. Ele capta sua distância até a água, podendo então, a partir de medições (com uma fita métrica) da distância e da altura do recipiente, calcular o nível da água no microcontrolador. Uma proteção para o sensor é necessária para evitar ruídos. A Figura 9 apresenta o sensor de nível utilizado, que possui a tensão de operação de 5V, sendo necessário um divisor de tensão para a leitura, já que o microcontrolador trabalha com tensão de 3,3V.

Figura 9. Módulo Sensor Ultrassônico HC-SR04.

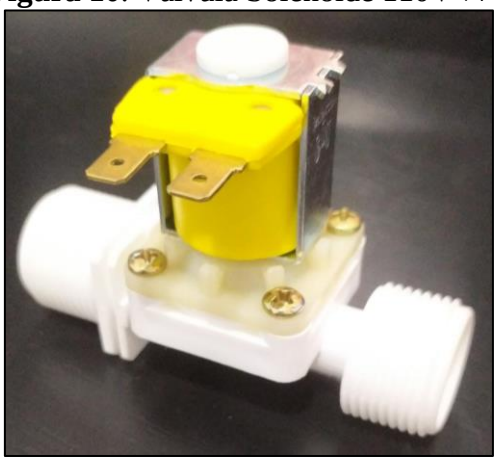
Fonte: Autor.

3.4. Válvula Solenoide

Para o controle do nível foi utilizada uma válvula solenoide de corrente alternada, normalmente fechada (NF), com uma rosca de três quartos de polegada, tensão de 110 Volts (V) e a passagem de água é de meia polegada (Figura 10). É ela que irá liberar a

passagem de água do reservatório para o recipiente, através de um circuito de acionamento, que é controlado pelo microcontrolador.

Figura 10. Válvula Solenoide 110V ¾”.

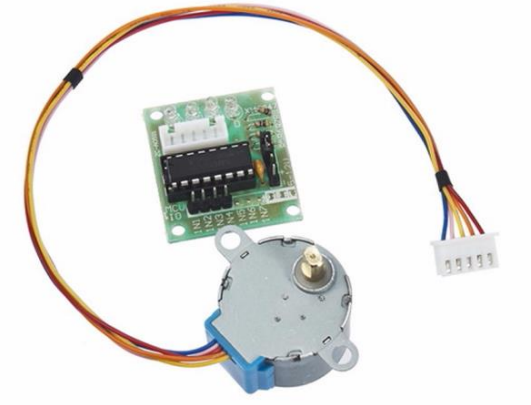


Fonte: Autor.

3.5. Motor de Passo e Rosca sem Fim

Para o controle da carga, foi utilizado um motor de passo de baixo custo, que é o modelo 28BYJ-48 (Figura 11), além de uma rosca sem fim conectada no motor para realizar o transporte. O motor é alimentado com a tensão de 5V e possui aproximadamente 2050 passos por revolução. Acompanhando o motor, também foi utilizado o *driver* ULN2003, para o controle das 4 bobinas do motor e amplificação da corrente de saída do microcontrolador.

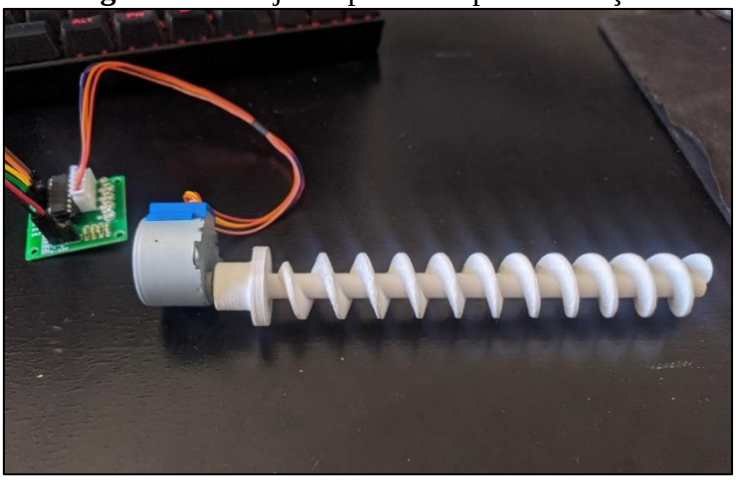
Figura 11. Motor de passo 28BYJ-48 e driver ULN2003.



Fonte: Oliveira (2017).

É o motor que libera ou não a passagem da ração do reservatório para o recipiente, através do sistema com a rosca sem fim, que é rotacionada quando o motor é acionado, transportando a ração do reservatório para o recipiente. A Figura 12 apresenta o conjunto com o motor de passo, o *driver* de amplificação de corrente e a rosca sem fim.

Figura 12. Conjunto para transporte da ração.



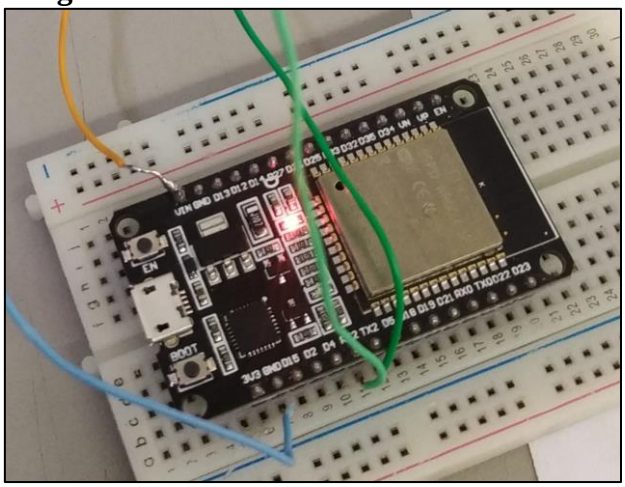
Fonte: Autor.

3.6. Microcontrolador ESP32

O *hardware* utilizado é um microcontrolador ESP32-WROOM-32, um componente de baixo custo, que segundo seu fabricante (ESPRESSIF SYSTEMS, 2021), o módulo possui 4 Megabytes (MB) de memória *Flash*, frequência do *clock* de 240 Megahertz (MHz) e diversas funções, tais como as comunicações dos rádios *Bluetooth* e *Wi-Fi*.

Ele é o componente principal do projeto, realizando toda a programação de recepção e escrita de dados, controlando todo o protótipo. O microcontrolador recebe os dados da célula de carga e do sensor ultrassônico, além dos comandos recebidos via aplicativo, mandados pelo usuário. E na parte do controle, é o ESP32 que mandará sinais de tensão para os circuitos de acionamento da válvula solenoide e do motor de passo. A Figura 13 apresenta o microcontrolador utilizado.

Figura 13. Módulo do ESP32 DEVKIT V1.



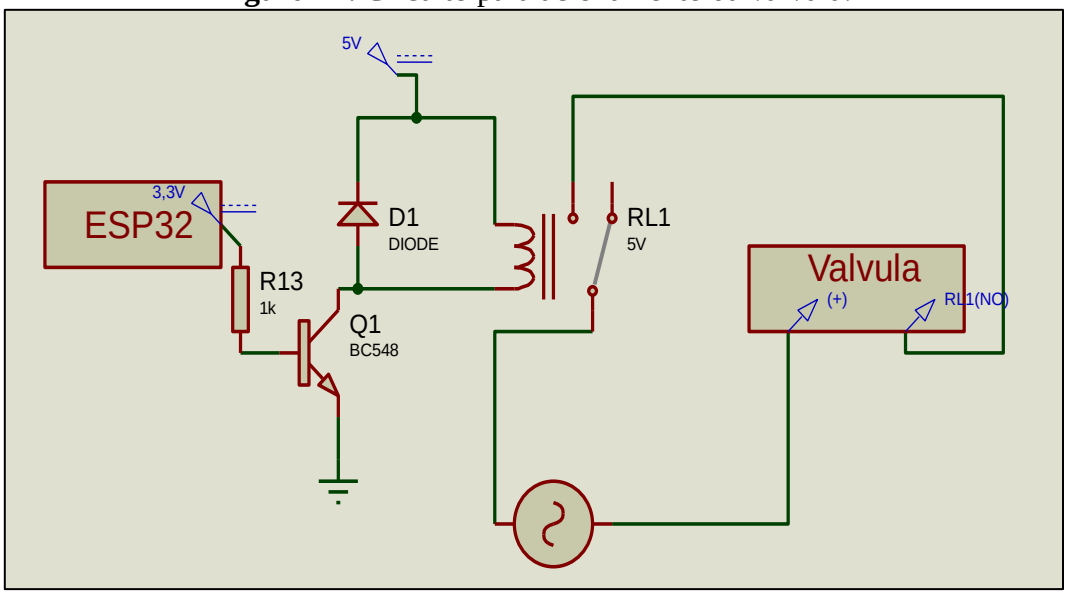
Fonte: Autor.

3.7. Circuitos Eletrônicos

Foram necessários quatro circuitos eletrônicos para o projeto, dois para as leituras e dois para os acionamentos. Na parte do controle do nível da água, primeiramente foi feito o circuito para realizar a leitura do sensor de nível, por meio da conexão direta dos pinos no microcontrolador.

Ainda no controle da água, o segundo circuito foi para o acionamento da válvula solenoide (Figura 14), realizado a partir de um relé de 5V, com auxílio de um transistor de junção bipolar, que recebe sinais do microcontrolador de 3,3V, abrindo a válvula. O circuito foi desenvolvido no *software Proteus 8*.

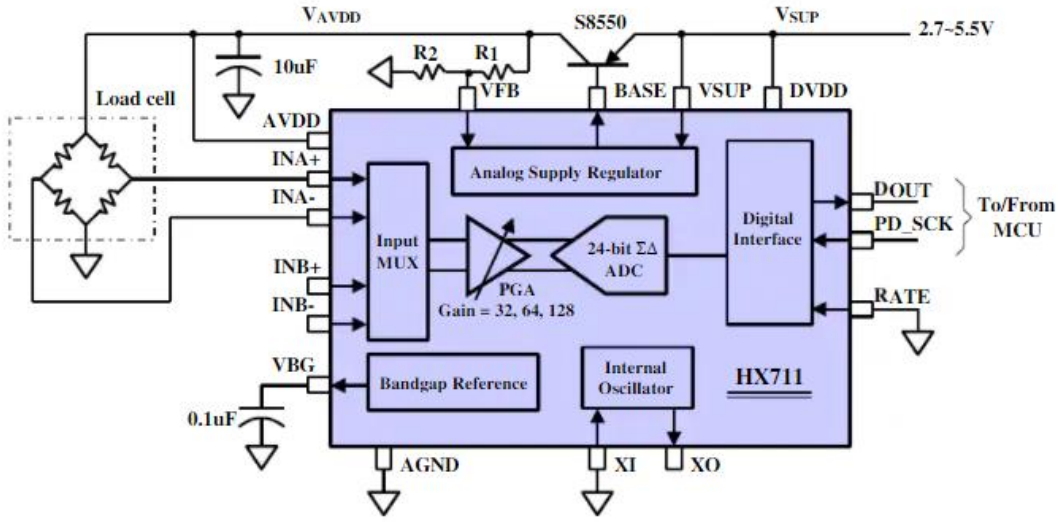
Figura 14. Circuito para acionamento da válvula.



Fonte: Autor.

Já pensando no controle da ração, o terceiro circuito necessário foi para realizar a medida da massa, utilizando a célula de carga e o módulo HX711, que já fornece um valor digital e amplificado para o microcontrolador. O esquema do circuito pode ser visto na Figura 15, que ilustra o funcionamento para o sinal chegar no microcontrolador.

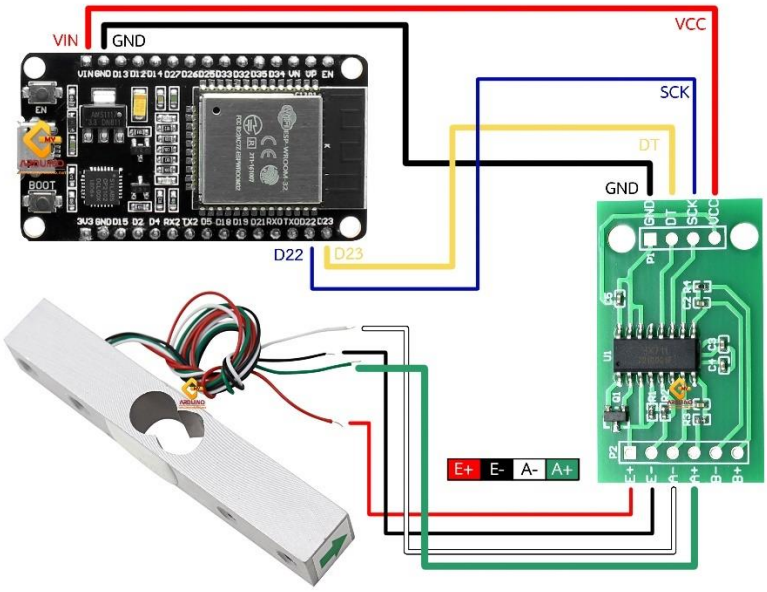
Figura 15. Circuito do módulo HX711.



Fonte: Digikey (2021).

No circuito da leitura da massa, basicamente, a célula de carga emite um sinal por meio da variação da ponte de *Wheatstone* (visto na Figura 15), que é gerado pela deformação da célula de carga. Esse sinal entra no módulo HX711, que primeiramente entra em um multiplexador, levando em conta que o módulo comporta mais de uma célula. Após o multiplexador, o sinal será convertido por um conversor Analógico-Digital, que então saíra do HX711. Por fim, o valor ainda precisa passar por um divisor de tensão de 3,3V (1 resistor de 2k Ohms e 1 resistor de 1k Ohms) antes de chegar no microcontrolador. Na Figura 16 é visto o esquemático com a célula de carga, o módulo HX711 e o ESP32.

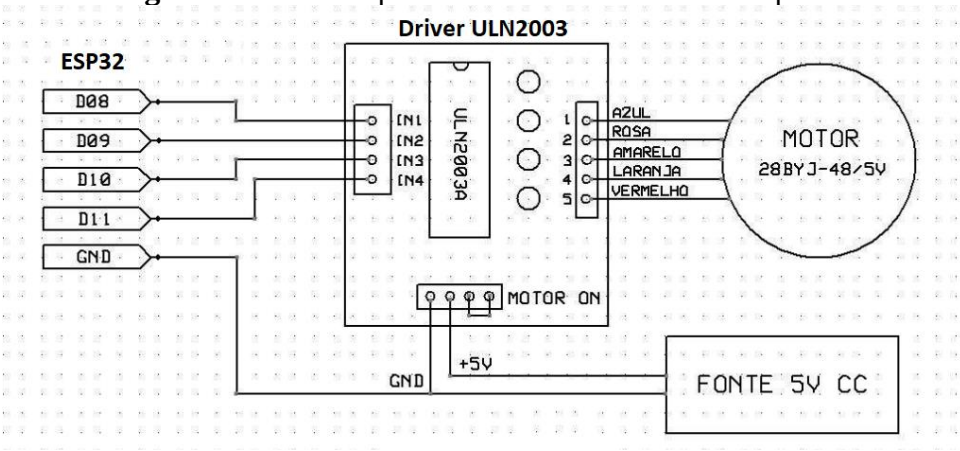
Figura 16. Esquemático para célula de carga.



Fonte: My Arduino (2020).

O último e quarto circuito é o de acionamento do motor de passo em conjunto com o *driver* ULN2003. Neste circuito, há quatro sinais de saída do microcontrolador que vão para a entrada do *driver* de amplificação de corrente, um para cada bobina do motor de passo. O *driver* basicamente amplifica os sinais e envia para o motor. O esquema é visto na Figura 17.

Figura 17. Circuito para acionamento do motor de passo.



Fonte: Eletrogate (2018).

Todos os circuitos foram alimentados, inclusive o microcontrolador, com uma fonte de 5V, com corrente máxima de 3,5A. Apenas para o circuito de acionamento da

válvula foi necessária uma alimentação direto da rede, pois a válvula é de corrente alternada.

3.8. *Firmware* e *Software* de Interface

O projeto necessita de um *firmware* e um *software* para o controle (microcontrolador) e para a interface do aplicativo, respectivamente. O *firmware* de controle tem as funções de medição, conexão com o aplicativo e acionamento. O *software* de interface possui a função de enviar os comandos do usuário, além da interface com as medições do nível da água e da massa da ração.

Na função de medição, o ESP32 recebe os dados dos circuitos da célula de carga e do sensor ultrassônico, para realizar as conversões necessárias, enviando as medições nas grandezas de mililitros (ml) para água e de gramas (g) para a ração. A função da conexão com o aplicativo irá enviar os dados já nas grandezas corretas, via *Wi-fi*, e vai receber os comandos que o usuário efetuar por meio da interface. Por fim, a função de acionamento, irá enviar sinais de tensão para os circuitos de acionamentos, tanto do motor de passo quanto da válvula solenoide, de acordo com os comandos recebidos pelo usuário.

O *software* de interface, que inicialmente foi uma aplicação *Web* e posteriormente migrado para um aplicativo, recebe os dados das medições e mostra de forma visual e interativa as medidas. Então, com os dados mostrados na interface, o usuário poderá mandar comandos de acionamento para o microcontrolador, além de configurar horários para a execução dos comandos, tudo através de botões visuais do aplicativo.

O protocolo MQTT (*Message Queuing Telemetry Transport*) foi utilizado para realizar a comunicação entre o microcontrolador e o aplicativo. Desenvolvido inicialmente pela IBM nos anos 90, com base na pilha de protocolos TCP/IP, o MQTT tornou-se o protocolo mais utilizado em comunicações IoT ou *internet* das coisas (YUAN, 2017).

O desenvolvimento do *software* de leitura e controle foi realizado no ambiente do *PlatformIO*, usando-o como extensão do *software Visual Studio Code*. Para a criação do aplicativo *Web*, primeiramente, foi utilizado o *Node-RED*, para o auxílio tanto da programação *Web* quanto da integração que foi realizada entre o microcontrolador e a rede. Posteriormente, a aplicação foi migrada para um aplicativo *Android*.

No *firmware* desenvolvido para o ESP32 utilizou-se a linguagem C++, assim como suas respectivas bibliotecas. É importante ressaltar o uso do FreeRTOS, que

basicamente é um *kernel* de sistema operacional em tempo real (AMAZON, 2021), proporcionando a execução de rotinas de programação do *firmware* instantaneamente (ou quase instantâneas), como ler e receber comandos via MQTT ao mesmo tempo que realiza as leituras dos sensores, por exemplo.

Na aplicação *Web* utilizou-se Node.js, que possui a linguagem *JavaScript* como base, já que é o padrão da ferramenta de desenvolvimento utilizada (*Node-RED*). Para hospedar a aplicação IoT, foi necessário utilizar o AWS (Amazon Web Services) da *Amazon* para criar o servidor na nuvem, já que a empresa oferece um ano de uso grátis, com alguns limites, mas suficiente para o presente projeto. Também foi utilizado a plataforma *Duck DNS* para usar o domínio gratuito que eles fornecem.

4. RESULTADOS E DISCUSSÕES

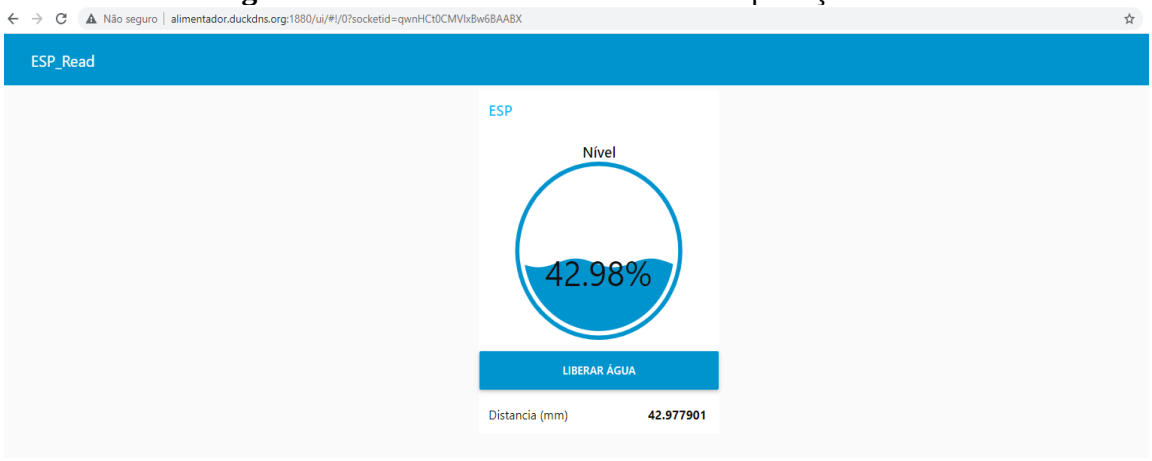
4.1. Testes Iniciais

Para os primeiros resultados, foram realizados testes tanto na programação quanto na parte eletrônica. Com a eletrônica, foram desenvolvidos alguns circuitos para teste, como um circuito para a leitura do nível da água e um circuito para testes de acionamento da válvula solenoide.

Foram desenvolvidos alguns *firmwares* para o ESP32, para a realização de testes de comunicação, por exemplo, além do desenvolvimento da programação *Web*. Um desses *firmwares* faz a leitura do sensor ultrassônico com o ESP32 e manda para a programação do *Node-RED*, via MQTT, que mostra os valores de forma visual, conforme a Figura 18. Ressalta-se que a porcentagem do nível na imagem ainda não era um valor real.

Os testes iniciais já foram promissores, mas ainda sem valores reais do nível. Foram obtidos valores de distância para testes do sensor ultrassônico e principalmente para testar a comunicação entre o microcontrolador e a aplicação IoT, sendo possível então receber e enviar dados via MQTT, podendo validar o uso do protocolo e a comunicação.

Figura 18. Interface de leitura inicial da aplicação *Web*.



Fonte: Autor.

4.2. Controle do Nível

Um recipiente de volume conhecido, e graduado com marcações intermediárias de medidas de volume, foi utilizado para a calibração do controle do nível de água. Para tal, tomando como referência 500 ml de água, testes de nível foram realizados com o sensor ultrassônico, sendo verificados o acionamento da válvula solenoide e as medições no nível da água. A Figura 19 apresenta um dos momentos de teste de controle de nível.

Figura 19. Teste de controle de nível.

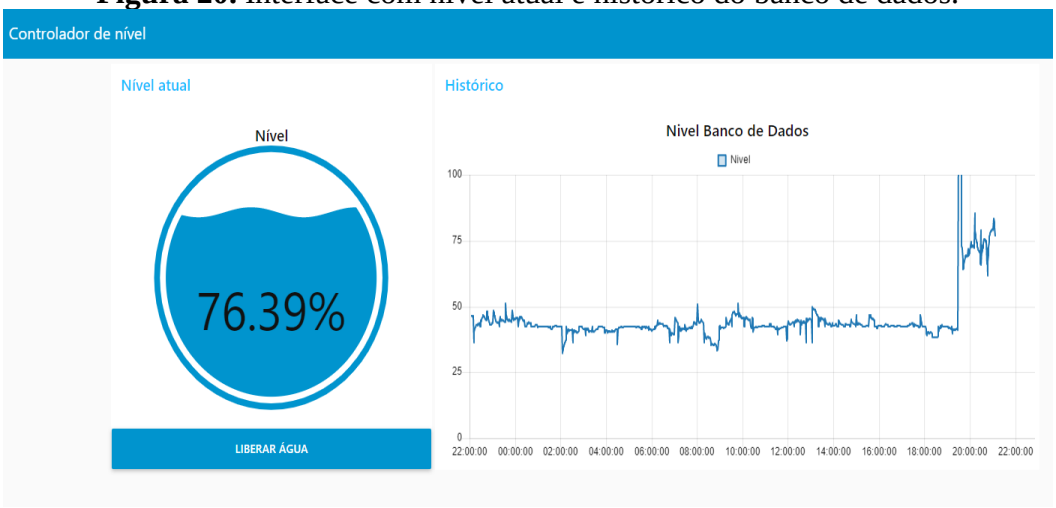
Fonte: Autor.

Houve dificuldades na leitura do sensor ultrassônico e foi constatado que o conversor Analógico-Digital do ESP32 não apresentava linearidade em toda sua faixa de leitura, necessitando de uma calibração adicional no *firmware*. A minimização do

problema foi possível a partir da melhoria da calibração do conversor AD, por meio do uso de uma biblioteca nativa do microcontrolador.

Com o sistema para o controle do nível já funcional, inclusive com o circuito para o acionamento da válvula, foi possível realizar leituras reais do nível da água. E assim foi realizado um teste, com quase um dia de leituras do nível em porcentagem que foram armazenadas em um banco de dados na nuvem, o *Amazon* RDS (*Amazon Relational Database Service)*, um serviço que trabalha em conjunto com o AWS já mencionado. Foi obtido como resultado um gráfico com as leituras durante o tempo, chamado de histórico, e ao mesmo tempo o nível atual medido pelo sensor. Ambos são vistos na Figura 20.

Figura 20. Interface com nível atual e histórico do banco de dados.



Fonte: Autor.

As variações mais bruscas no gráfico foram quando realmente houve mudanças do nível, retirando-se um pouco da água manualmente, ou enchendo o recipiente com a abertura da válvula solenoide, por meio do botão “Liberar Água”, visto na imagem anterior. Os valores plotados nos gráficos são valores instantâneos, fato que justifica as pequenas variações não desejadas, já que ainda não existia uma média da leitura, que foi implementada posteriormente.

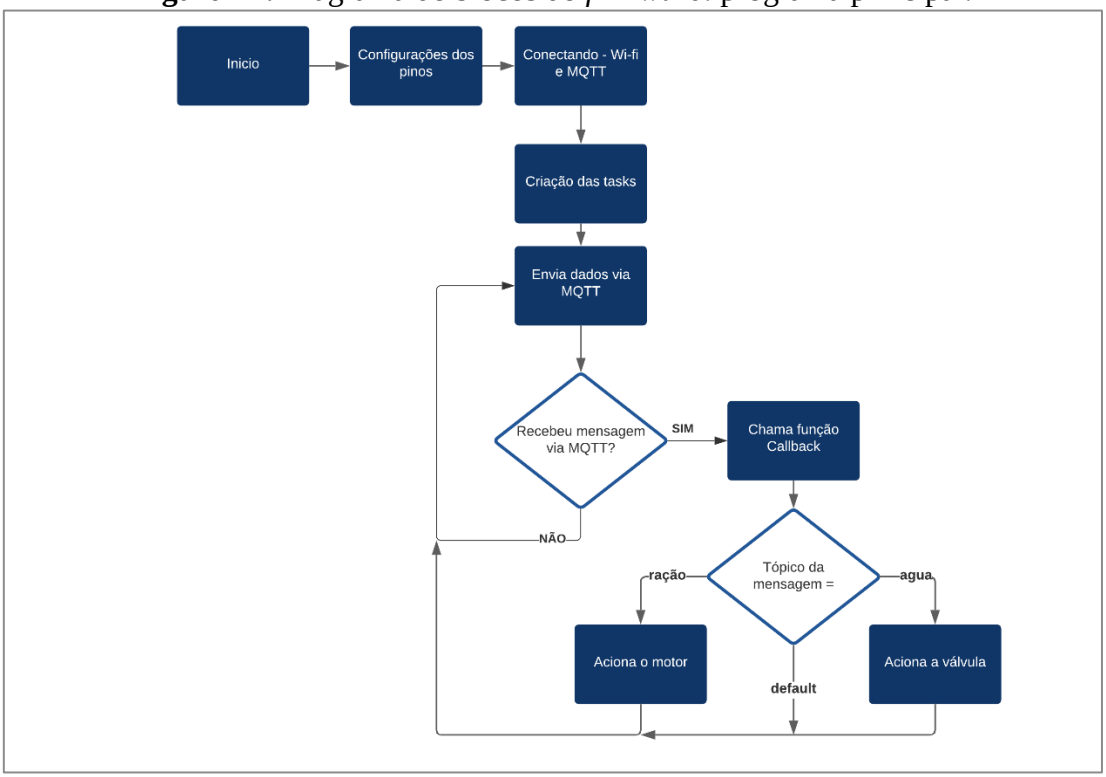
4.3. *Firmware*

O *firmware*, de forma geral, tem um funcionamento bem simples. Com o uso do FreeRTOS foi possível ter uma comunicação rápida e estável, podendo receber e enviar dados, via MQTT, do *software* de interface ao mesmo tempo em que as medições são realizadas. Isso acontece através da criação de *tasks* (ou tarefas) do FreeRTOS no início

do programa principal, que realizam as leituras dos sensores, enquanto o programa principal segue rodando normalmente.

A Figura 21 apresenta o diagrama de blocos do programa principal do *firmware*, para exemplificar de forma básica o seu funcionamento. O programa inicia com as configurações dos pinos e do *Wi-fi*, criando as *tasks* do FreeRTOS e posteriormente o programa entra em um *loop* que envia e lê os dados, via MQTT, acionando a válvula e o motor de passo ou não, dependendo dos dados enviados.

Figura 21. Diagrama de blocos do *firmware*: programa principal.



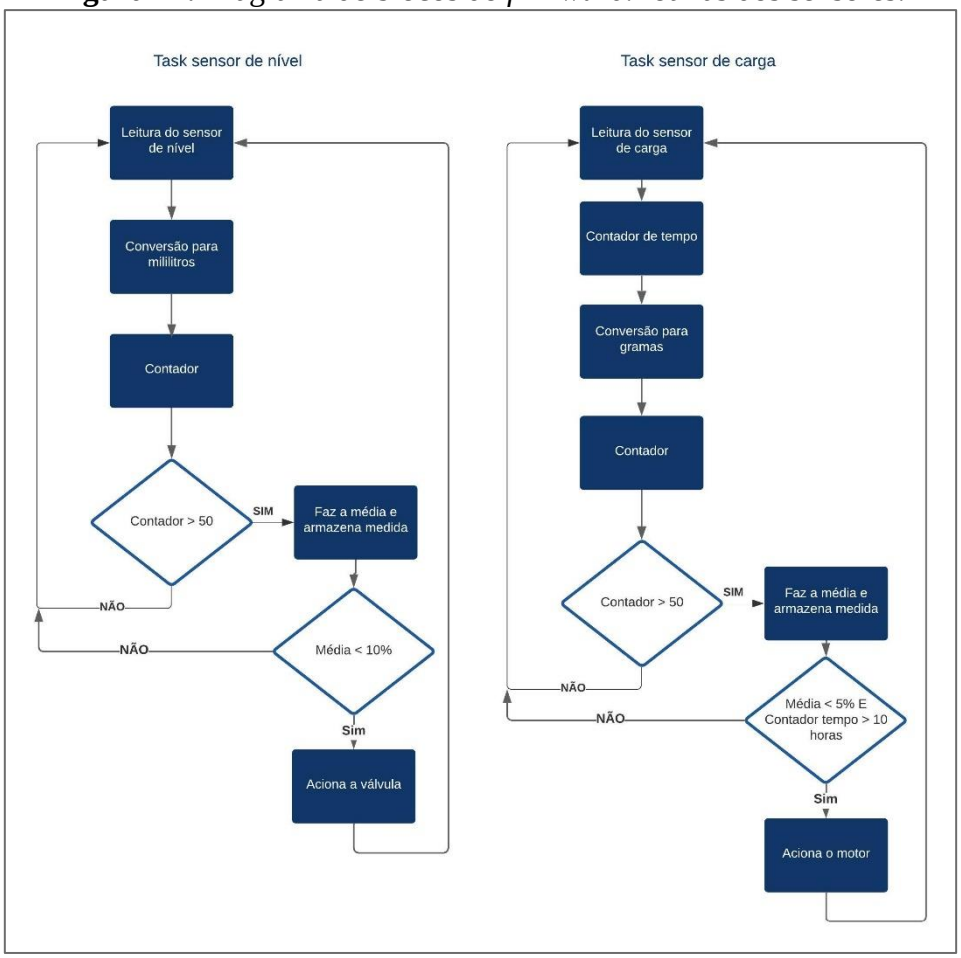
Fonte: Autor.

Além do programa principal, também há as *tasks* dos dois sensores (nível e carga) que são exemplificados pelo diagrama de blocos com as rotinas dos sensores na Figura 22. Basicamente, as duas rotinas são *loops* que realizam as leituras continuamente, calculam uma média depois de 50 valores e por fim, armazenam o valor da média em uma variável global, que posteriormente será enviada para a interface.

E no caso específico da rotina do sensor de nível, existe uma comparação, que caso o nível esteja abaixo de 10%, a válvula é acionada para encher a vasilha de água. A rotina do sensor de carga tem uma programação parecida, só que neste caso é levado em conta um tempo, que está sempre aumentando após o último acionamento do motor de

passo. Então, quando a vasilha de ração está abaixo de 5% por mais de 10 horas, o motor é acionando, enchendo a vasilha. O valor de tempo é salvo na memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory)* a cada 10 minutos, não perdendo a informação (ou minimizando a perda) quando o protótipo não estiver energizado.

Figura 22. Diagrama de blocos do *firmware*: rotinas dos sensores.



Fonte: Autor.

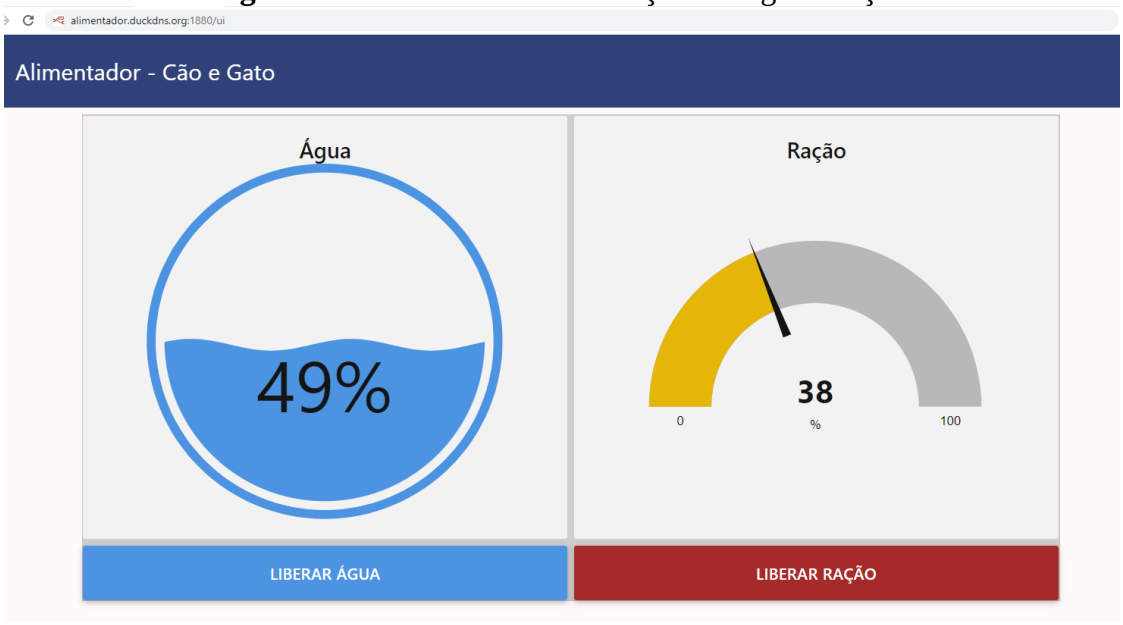
4.4. Interface

A interface desenvolvida mostra informações percentuais dos quantitativos de água e ração nos recipientes. Os dados são armazenados na nuvem. O *software* recebe as leituras do nível da água e da massa da ração, convertendo ambos em porcentagem e mostrando os valores visualmente. Na questão dos envios, basicamente, a interface só envia dados quando os botões de liberação de água e ração são pressionados. Quando isso acontece, o motor de passo ou a válvula funcionam até as vasilhas estarem

aproximadamente 100% preenchidas, sendo assim, não há derramamento caso os botões sejam pressionados indevidamente.

Lembrando que a comunicação com o *hardware*, enviando e recebendo os dados, ocorrem a partir do uso do MQTT. A Figura 23, já mostra a interface com as medições bem visuais, tanto do nível da água quanto da carga da ração, sempre em porcentagem, além dos botões para acionamentos da válvula e do motor, já mencionados anteriormente.

Figura 23. Interface com as medições de água e ração.



Fonte: Autor.

Para facilitar o uso da aplicação para o usuário no celular, a interface foi migrada para um aplicativo *Android*. Para isso, foi utilizado a plataforma *Hermit*, que realiza este processo de forma simples e fácil. Sendo assim, não houve desenvolvimento *Android* de fato. A interface do aplicativo utilizado pode ser vista na Figura 24.

Figura 24. Interface do aplicativo.

Fonte: Autor.

4.5. Controle da Carga

Para o controle da massa ou carga de ração, foi necessário fazer a calibração da célula de carga, a qual foi realizada a partir do uso de algo com massa conhecida. Um recipiente vazio foi colocado inicialmente sobre a célula de carga para tarar a balança. Posteriormente, foi adicionado ao recipiente uma massa conhecida de água. Foram usadas massas de 500 gramas (Figura 25) e 800 gramas. Com ambas foram obtidos valores de calibração muito próximos.

Figura 25. Calibração do sensor de carga.

Fonte: Autor.

Com o sistema de leitura da carga pronto, foi então desenvolvido o sistema para o transporte da ração. Diferente do controle de nível que só possui uma válvula solenoide que abre e fecha, para o controle da ração foi necessário a utilização do motor de passo em conjunto com uma rosca sem fim, por meio do circuito de acionamento já visto. O transporte da ração ocorreu de modo satisfatório, mesmo que foram necessárias algumas improvisações para o seu funcionamento, como um corte feito no cano que possui a rosca sem fim em seu interior, para que a ração caia do reservatório no encanamento de transporte de ração, conforme mostra a Figura 26.

Figura 26. Cano com a rosca sem fim e o motor de passo.

Fonte: Autor.

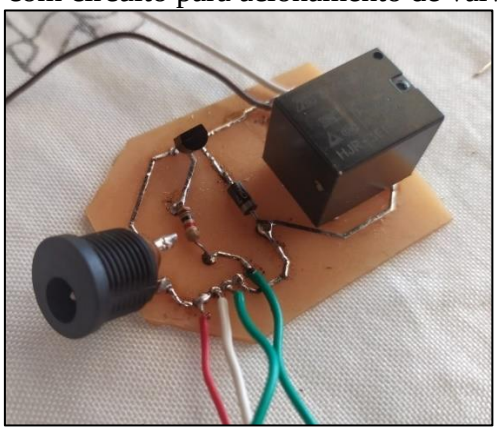
4.6. A Estrutura

Com as leituras do nível da água e da carga da ração funcionando, a interface mostrando os valores de forma visual, os acionamentos de acordo com a programação ou controle do usuário, foi montada então toda a estrutura do protótipo, com todos os componentes físicos mencionados, incluindo os reservatórios de água e ração. Buscou-se na montagem minimizar o custo total de confecção da estrutura, usando canos de PVC e materiais recicláveis.

Também foi necessário a confecção de uma placa de circuito impresso (placa PCB, do inglês *Printed Circuit Board*), desde o desenvolvimento do *layout* até a soldagem dos componentes, que comtempla o circuito de acionamento da válvula (Figura 14) e um conector para a fonte de 5V, utilizada para alimentação dos sensores, microcontrolador e

do relé utilizado no próprio circuito de acionamento da válvula. O *layout* da placa foi desenvolvido no *software Proteus 8* e a placa pode ser vista na Figura 27.

Figura 27. Placa PCB com circuito para acionamento de válvula e alimentação 5V.



Fonte: Autor.

Então, a estrutura comtempla os canos (e uma pequena mangueira para o controle do nível) que ligam os reservatórios com seus respectivos recipientes (água e ração), dois banquinhos de plásticos que foram adaptados e usados como suportes para os recipientes e de apoio para os canos, e dois galões de água recicláveis de cinco litros adaptados e utilizados como reservatórios, tanto para água quanto para a ração. Também foram utilizados algumas luvas e roscas para encaixe e ligação dos canos, além da placa PCB da Figura 27. A Figura 28 apresenta a estrutura do protótipo do alimentador automático confeccionado.

Figura 28. Estrutura do protótipo.

Fonte: Autor.

5. CONCLUSÕES

Com o término de todas as atividades deste trabalho, conclui-se que os resultados obtidos, de modo geral, foram satisfatórios. Foi construído um protótipo que, apesar de ser simples, é funcional e de baixo custo, atendendo as demandas inicialmente estipuladas, além de estar pronto para novas melhorias e aprimoramentos.

Mais especificamente, foram obtidos como resultados, um *firmware* de controle bem estruturado e totalmente funcional, uma interface muito visual e de fácil utilização, juntos com uma comunicação estável entre o microcontrolador e a interface. O desenvolvimento dos circuitos de leituras dos sensores com medidas muito aceitáveis, levando em conta o baixo custo do projeto, junto com os circuitos de acionamento. Além de toda a estrutura física, que demandou alguns improvisos, mas que de modo geral, atendeu as demandas do projeto.

Com relação a parte de controle, o ESP32 provou ser um microcontrolador de baixo custo muito eficiente, fornecendo um excelente processamento junto com as suas comunicações de rádio, como o *Wi-Fi*, utilizado neste projeto. Foi possível realizar as leituras dos sensores e acionar a válvula solenoide e o motor de passo, havendo um controle muito eficiente. Ressalta-se que, apesar do transporte da ração ter ocorrido de modo satisfatório, seria interessante utilizar um motor com um pouco mais de torque e velocidade em trabalhos futuros, podendo melhorar e agilizar o processo.

A interface também teve um bom resultado. Os valores das leituras da ração e da água foram apresentados de forma bem visual, além dos botões de acionamentos de fácil utilização, tudo desenvolvido com auxílio do *Node-RED*. As utilizações do servidor do AWS para hospedar a aplicação na nuvem e do RDS para testes com o banco de dados, também foram muito satisfatórias, sendo simples e práticos para usar. Também é importante lembrar, que não houve desenvolvimento *Android* de fato, utilizando então uma plataforma que realiza a migração da aplicação *Web* para um aplicativo *Android*. Foi visto que esse desenvolvimento precisaria de um esforço e tempo muito grandes e que talvez poderia tirar um pouco do foco da área de engenharia, porém há a possibilidade de haver esse desenvolvimento em trabalhos futuros.

O desenvolvimento da estrutura física do projeto foi a etapa em que mais foram identificados problemas de modo geral, mas também foi satisfatória, levando em conta a falta de estrutura para a confecção, já que o trabalho foi todo realizado em casa. Utilizando

objetos baratos e na grande maioria recicláveis, foram realizados alguns improvisos, para que fosse possível tanto a realização dos testes iniciais quanto para a finalização deste trabalho, já com todo o controle de água e ração funcional.

Por fim, é entendido que de modo geral este trabalho atendeu as demandas inicialmente estipuladas ou projetadas. Para isso, utilizou-se muito do conhecimento adquirido durante todo o curso de Engenharia de Controle e Automação, através das suas respectivas disciplinas e projetos de extensão ou pesquisa. O desenvolvimento deste trabalho também proporcionou muito aprendizado e conhecimento, com necessidade de pesquisas e aprofundamento de algumas áreas especificas.

Para trabalhos futuros, será interessante implementar novas funções no projeto e aprimorar as que já possui. Um exemplo de uma nova função, seria o usuário poder inserir e programar horários para o funcionamento do alimentador. Já na questão de aprimorar, seria elevar um pouco o investimento do protótipo, utilizando sensores e circuitos mais profissionais e montando uma melhor estrutura como um todo. Desta forma, seria possível melhorar o funcionamento de modo geral, tendo mais eficiência e minimizando os problemas.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, Gabriel Lemos. **Autofeeder**: alimentador automático para animais domésticos de pequeno porte. Lages: Centro Universitário UNIFACVEST, 2020.

AMAZON. ***Amazon Web Services:*** FreeRTOS. 2021. Disponível em: https://aws.amazon.com/pt/freertos. Acesso: 17 out. 2021.

ARDUINO E ELETRÔNICA. **Módulo Hx711 Conversor 24bits.** 2021. Disponível em: https://www.arduinoeeletronica.com.br/produto/modulo-hx711-conversor-24bits-p-celula-carga-peso-sensor-arduino. Acesso: 16 set. 2021.

DIGIKEY. **HX711 Datasheet.** SparkFun Electronics, 2021. Disponível em: https://www.digikey.com/htmldatasheets/production/1836471/0/0/1/hx711.html. Acesso: 18 set. 2021.

ELETROGATE. **Guia completo do Motor de Passo 28BYJ-48 + Driver ULN2003**. Blog Eletrogate, jul. 2018. Disponível em: https://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003. Acesso: 13 set. 2021.

ESPRESSIF SYSTEMS. **ESP32-WROOM-32 Datasheet.** China, v. 3.2, 2021. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Acesso: 15 ago. 2021.

FIRMINO, Marcos de Souza; MATEUS, W. R. F. **Thymos Pet** – dosador alimentar automático para animais domésticos. Tubarão: Universidade do Sul de Santa Catarina, 2020.

GEREÇÃO PET. **Ração Royal Canin Mini Adulto.** 2021. Disponível em: https://www.petlandbrasil.com.br/racao-royal-canin-mini-adulto. Acesso: 15 set. 2021.

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

IBGE**. População de animais de estimação no Brasil – 2013.** Disponível em: https://www.gov.br/agricultura/pt-br/assuntos/camaras-setoriais-tematicas/documentos/camaras-tematicas/insumos-agropecuarios/anos-anteriores/ibge-populacao-de-animais-de-estimacao-no-brasil-2013-abinpet-79.pdf. Acesso: 16 jan. 2021.

KABUM. **Kit Comedouro Pet Automático KaBuM.** 2021. Disponível em: https://www.kabum.com.br/produto/151664/kit-comedouro-pet-automatico-kabum-smart-camera-1080p-6-litros-bebedouro-pet-automatico-kabum-smart-2-5-litros. Acesso: 16 set. 2021.

OLIVEIRA, Euler. **Como usar com Arduino:** Motor de Passo 28BYJ-48 com Driver ULN2003. Master Walker Shop, 2017. Disponível em: https://blogmasterwalkershop.com.br/arduino/arduino-utilizando-motor-de-passo-28byj-48-e-driver-uln2003. Acesso: 17 set. 2021

MOURA, Wandgleisom Garcia de. **A construção social do mercado pet food no Brasil**: estudo de caso da família Sens. Florianópolis: Universidade Federal de Santa Catarina, 2013. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/114855/TCC_Wand.final.banca_11.07.2013_formatada%20A5_pronto.pdf. Acesso: 10 mai. 2021.

MUNDO VETERINARIO. **Como não errar na alimentação de cães e gatos.** Syntec, jun. 2018. Disponível em: https://syntec.com.br/news/como-nao-errar-na-alimentacao-de-caes-e-gatos. Acesso: 15 abr. 2021.

PASTORI, Érica Onzi; MATOS, Liziane Gonçalves. Da paixão à “ajuda animalitária”: o paradoxo do “amor incondicional” no cuidado e no abandono de animais de estimação. **Caderno Eletrônico de Ciências Sociais**, Vitória, v. 3, n. 1, p. 112-132, set. 2015.

PETLAND. **Ração Royal Canin Premium Cat Gatos Adultos Castrados.** 2021. Disponível em: https://www.petlandbrasil.com.br/racao-royal-canin-premium-cat-gatos-adultos-castrados. Acesso: 15 set. 2021.

PETLOVE. **Comedouro Vida Mansa Alumínio para Raças Médias e Pequenas.** 2020. Disponível em: https://www.petlove.com.br/comedouro-vida-mansa-aluminio-para-racas-pequenas-branco/p?sku=1914536. Acesso: 15 set. 2021.

POLI, Mariana. **Mercado pet cresce graças a mudanças no comportamento dos donos de animais de estimação.** Você S/A, ed. 223, dez. 2016. Disponível em: https://vocesa.abril.com.br/geral/mercado-pet-cresce-gracas-a-mudancas-no-comportamento-dos-donos-de-animais-de-estimacao. Acesso: 24 mai. 2021.

ROYALPETS. **Comedouro Automático PetDog.** 2020. Disponível em: https://www.royalpets.com.br/comedouro-automatico-duplo-bebedouro-para-caes-e-gatos-3l-1-5kg.html. Acesso: 14 set. 2021.

RUIZ, Daniela Casapietra. **A importância da nutrição do cão e do gato na senilidade**. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2013. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/95068/000917345.pdf. Acesso: 17 mai. 2021.

SILVEIRA, Flávia. **Brasil fecha 2018 como segundo maior mercado pet do mundo.** Gazeta do Povo, fev. 2019. Disponível em: https://www.gazetadopovo.com.br/economia/brasil-fecha-2018-como-segundo-maior-mercado-pet-do-mundo-2vhq0n3uempvkgdcm8arh382j. Acesso: 12 mai. 2021.

ULRICH, Ana Paula e Assis; NOVAK, Jéssica Luise. **Alimentador voltado para animais domésticos.** Curitiba: Universidade Tecnológica Federal do Paraná, 2012.

YUAN, Michael. **Conhecendo o MQTT.** IBM, out. 2017. Disponível em: https://developer.ibm.com/br/articles/iot-mqtt-why-good-for-iot.
Acesso: 10 out. 2021.

APÊNDICES

APÊNDICE A – Programação do ESP32

```
//Programa: Alimentador TCC
//Autor: Gabriel Corradini da Cunha

#include <Ultrasonic.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include <HX711.h>
#include <Stepper.h>

// INSTANCIANDO O MOTOR DE PASSO E CONFIGURANDO OS PINOS
Stepper myStepper(stepsPerRevolution, 27, 14, 13, 12);

#define PORTA_TRIGGER GPIO_NUM_12
#define PORTA_ECHO GPIO_NUM_13
#define VALVULA 4
#define LED 2
#define pinDT 23
#define pinSCK 22
#define topicRacao 7
#define topicAgua 5
#define pesoMin 0.001
#define pesoMax 1.0
#define escala -352622 //valor obtido a partir de testes, especifico para
cada aplicação ou componente

HX711 scale;
Ultrasonic ultrasonic(PORTA_TRIGGER, PORTA_ECHO);

float distancia;
float sensor_nivel;
float sensor_carga;
byte valvula;
float medida = 0;
int timeout_callback;
int timer_inicio;
bool timer_inicio_controle = true;

const char *ssid = "SSID_Roteador";
const char *password = "Senha_Roteador";
const char *mqttServer = "34.224.XX.XXX";
const int mqttPort = 1883;
```

```
const char *mqttUser = "user_mqtt";
const char *mqttPassword = "senha_mqtt";

WiFiClient espClient;
PubSubClient client(espClient);
int analog_value = 0;
int contador = 1;
char mensagem[10];

void reconectabroker()
{
  //Conexao ao broker MQTT
  client.setServer(mqttServer, mqttPort);
  while (!client.connected())
  {
    Serial.println("Conectando ao broker MQTT...");
    if (client.connect("ESP32Client", mqttUser, mqttPassword))
    {
      Serial.println("Conectado ao broker!");
    }
    else
    {
      Serial.print("Falha na conexao ao broker - Estado: ");
      Serial.print(client.state());
      delay(100);
    }
  }
  //Serial.println("Conectado ao broker!");
}

void callback(char *topic, byte *payload, unsigned int length)
{
  Serial.println("Recebendo mensagem:");
  Serial.println((char)payload[0]);

  if ((char)payload[0] == topicAgua)
  {
    timeout_callback = millis();
    while (sensor_nivel < 90 && (millis() - timeout_callback) <= 20000)
//timeout de 20s
    {
      digitalWrite(VALVULA, HIGH);
      delay(200);
    }
  }
  else if ((char)payload[0] == topicRacao)
  {
    while (sensor_carga < 90 && (millis() - timeout_callback) <= 20000)
//timeout de 20s
```

```
    {
      myStepper.step(stepsPerRevolution);
      delay(200);
    }
  }
}

void sensor_nivel_task(void *p)
{
  float media_sensor_nivel;
  float leitura_sensor;
  int timeout = 0;
  while (1)
  {
    for (int i = 1; i <= 50; i++)
    {
      ultrasonic.measure();
      leitura_sensor = ultrasonic.get_cm();
      media_sensor_nivel += (1700000000000 / pow(leitura_sensor, 8));
      delay(10);
    }
    sensor_nivel = (media_sensor_nivel / i);
    media_sensor_nivel = 0;

    if (sensor_nivel < 10)
    {
      timeout = millis();
      while (sensor_nivel < 90 && (millis() - timeout) <= 20000)
//timeout de 20s
      {
        digitalWrite(VALVULA, HIGH);
        delay(200);
      }
    }

    //Serial.println(sensor);
  }
}

void sensor_carga_task(void *p)
{
  float media_sensor_carga;
  float leitura_sensor;
  while (1)
  {
    for (int i = 1; i <= 50; i++)
    {
      media_sensor_carga += scale.read();
      delay(10);
```

```
    }
    sensor_carga = (media_sensor_carga / i);
    media_sensor_carga = 0;
    //Serial.println(sensor);
    if (sensor_carga < 5)
    {
      if (timer_inicio_controle == true)
      {
        timer_inicio = millis();
        timer_inicio_controle = false;
      }
      timer = millis() - timer_inicio;
      if (timer > 36000000) //10h
      {
        while (sensor_carga < 90 && (millis() - timeout) <= 20000)
//timeout de 20s
        {
          myStepper.step(stepsPerRevolution);
          delay(200);
        }
        timer_inicio_controle = true;
      }
    }
  }
}

void setup()
{
  Serial.begin(115200);

  pinMode(VALVULA, OUTPUT);

  xTaskCreate(sensor_task, "sensor", 5000, NULL, 5, NULL);
  xTaskCreate(valvula_task, "valvula", 5000, NULL, 10, NULL);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.println("Iniciando conexao com a rede WiFi..");
  }
  Serial.println("Conectado na rede WiFi!");

  myStepper.setSpeed(5);

  scale.begin(pinDT, pinSCK); // CONFIGURANDO OS PINOS DA BALANÇA
  scale.set_scale(escala);    // ENVIANDO O VALOR DA ESCALA CALIBRADO

  client.setServer(mqttServer, mqttPort);
```

```
  client.setCallback(callback);
}

void loop()
{
  //Faz a conexao com o broker MQTT
  reconectabroker();
  client.loop();

  client.publish("ESP32_agua", sensor_nivel);
  client.subscribe("espRead_agua");

  client.publish("ESP32_racao", sensor_carga);
  client.subscribe("espRead_racao");

  delay(1);
}
```

ANEXOS

ANEXO A – HX711 Datasheet

海芯科技 AVIA Semiconductor

HX711

24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales

DESCRIPTION

Based on Avia Semiconductor's patented technology, HX711 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.

The input multiplexer selects either Channel A or B differential input to the low-noise programmable gain amplifier (PGA). Channel A can be programmed with a gain of 128 or 64, corresponding to a full-scale differential input voltage of ±20mV or ±40mV respectively, when a 5V supply is connected to AVDD analog power supply pin. Channel B has a fixed gain of 32. On-chip power supply regulator eliminates the need for an external supply regulator to provide analog power for the ADC and the sensor. Clock input is flexible. It can be from an external clock source, a crystal, or the on-chip oscillator that does not require any external component. On-chip power-on-reset circuitry simplifies digital interface initialization.

There is no programming needed for the internal registers. All controls to the HX711 are through the pins.

FEATURES

- **Two selectable differential input channels**
- **On-chip active low noise PGA with selectable gain of 32, 64 and 128**
- **On-chip power supply regulator for load-cell and ADC analog power supply**
- **On-chip oscillator requiring no external component with optional external crystal**
- **On-chip power-on-reset**
- **Simple digital control and serial interface: pin-driven controls, no programming needed**
- **Selectable 10SPS or 80SPS output data rate**
- **Simultaneous 50 and 60Hz supply rejection**
- **Current consumption including on-chip analog power supply regulator: normal operation < 1.5mA, power down < 1uA**
- **Operation supply voltage range: 2.6 ~ 5.5V**
- **Operation temperature range: -40 ~ +85℃**
- **16 pin SOP-16 package**

APPLICATIONS

- **Weigh Scales**
- **Industrial Process Control**

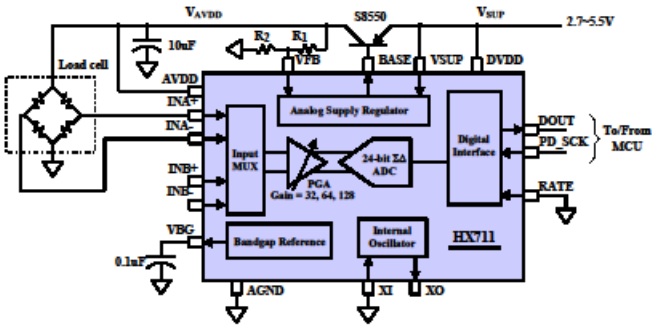
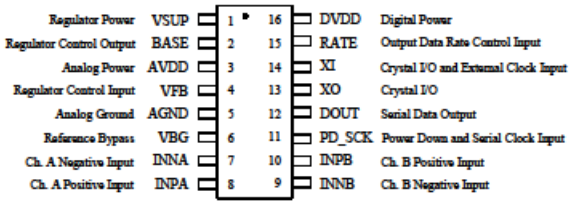


Fig. 1 Typical weigh scale application block diagram

TEL: (582) 252-9530 (P. R. China)
EMAIL: market@aviaic.com

AVIA SEMICONDUCTOR

Pin Description



SOP-16L Package

Pin #	Name	Function	Description
1	VSUP	Power	Regulator supply: 2.7 ~ 5.5V
2	BASE	Analog Output	Regulator control output（NC when not used）
3	AVDD	Power	Analog supply: 2.6 ~ 5.5V
4	VFB	Analog Input	Regulator control input（connect to AGND when not used）
5	AGND	Ground	Analog Ground
6	VBG	Analog Output	Reference bypass output
7	INA-	Analog Input	Channel A negative input
8	INA+	Analog Input	Channel A positive input
9	INB-	Analog Input	Channel B negative input
10	INB+	Analog Input	Channel B positive input
11	PD_SCK	Digital Input	Power down control (high active) and serial clock input
12	DOUT	Digital Output	Serial data output
13	XO	Digital I/O	Crystal I/O (NC when not used)
14	XI	Digital Input	Crystal I/O or external clock input, 0: use on-chip oscillator
15	RATE	Digital Input	Output data rate control, 0: 10Hz; 1: 80Hz
16	DVDD	Power	Digital supply: 2.6 ~ 5.5V

Table 1 Pin Description

KEY ELECTRICAL CHARACTERISTICS

Parameter	Notes	MIN	TYP	MAX	UNIT
Full scale differential input range	V(inp)-V(inn)		±0.5(AVDD/GAIN)		V
Common mode input		AGND+1.2		AVDD-1.3	V
Output data rate	Internal Oscillator, RATE = 0		10		Hz
	Internal Oscillator, RATE = DVDD		80		
	Crystal or external clock, RATE = 0		f_{clk}/1,105,920		
	Crystal or external clock, RATE = DVDD		f_{clk}/138,240		
Output data coding	2's complement	800000		7FFFFF	HEX
Output settling time(1)	RATE = 0		400		ms
	RATE = DVDD		50		
Input offset drift	Gain = 128		0.2		mV
	Gain = 64		0.4		
Input noise	Gain = 128， RATE = 0		50		nV(rms)
	Gain = 128， RATE = DVDD		90		
Temperature drift	Input offset（Gain = 128）		±6		nV/℃
	Gain（Gain = 128）		±5		ppm/℃
Input common mode rejection	Gain = 128， RATE = 0		100		dB
Power supply rejection	Gain = 128， RATE = 0		100		dB
Reference bypass（V_{BG}）			1.25		V
Crystal or external clock frequency		1	11.0592	20	MHz
Power supply voltage	DVDD	2.6		5.5	V
	AVDD，VSUP	2.6		5.5	
Analog supply current (including regulator)	Normal		1400		μA
	Power down		0.3		
Digital supply current	Normal		100		μA
	Power down		0.2		

（1）Settling time refers to the time from power up, reset, input channel change and gain change to valid stable output data.

Table 2 Key Electrical Characteristics

Analog Inputs

Channel A differential input is designed to interface directly with a bridge sensor's differential output. It can be programmed with a gain of 128 or 64. The large gains are needed to accommodate the small output signal from the sensor. When 5V supply is used at the AVDD pin, these gains correspond to a full-scale differential input voltage of ±20mV or ±40mV respectively.

Channel B differential input has a fixed gain of 32. The full-scale input voltage range is ±80mV, when 5V supply is used at the AVDD pin.

Power Supply Options

Digital power supply (DVDD) should be the same power supply as the MCU power supply.

When using internal analog supply regulator, the dropout voltage of the regulator depends on the external transistor used. The output voltage is equal to $V_{AVDD}=V_{BG}*(R1+R2)/R1$ (Fig. 1). This voltage should be designed with a minimum of 100mV below VSUP voltage.

If the on-chip analog supply regulator is not used, the VSUP pin should be connected to either AVDD or DVDD, depending on which voltage is higher. Pin VFB should be connected to Ground and pin BASE becomes NC. The external 0.1uF bypass capacitor shown on Fig. 1 at the VBG output pin is then not needed.

Clock Source Options

By connecting pin XI to Ground, the on-chip oscillator is activated. The nominal output data rate when using the internal oscillator is 10 (RATE=0) or 80SPS (RATE=1).

If accurate output data rate is needed, crystal or external reference clock can be used. A crystal can be directly connected across XI and XO pins. An external clock can be connected to XI pin, through a 20pF ac coupled capacitor. This external clock is not required to be a square wave. It can come directly from the crystal output pin of the MCU chip, with amplitude as low as 150 mV.

When using a crystal or an external clock, the internal oscillator is automatically powered down.

Output Data Rate and Format

When using the on-chip oscillator, output data rate is typically 10 (RATE=0) or 80SPS (RATE=1).

When using external clock or crystal, output data rate is directly proportional to the clock or crystal frequency. Using 11.0592MHz clock or crystal results in an accurate 10 (RTE=0) or 80SPS (RATE=1) output data rate.

The output 24 bits of data is in 2's complement format. When input differential signal goes out of the 24 bit range, the output data will be saturated at 800000h (MIN) or 7FFFFFh (MAX), until the input signal comes back to the input range.

Serial Interface

Pin PD_SCK and DOUT are used for data retrieval, input selection, gain selection and power down controls.

When output data is not ready for retrieval, digital output pin DOUT is high. Serial clock input PD_SCK should be low. When DOUT goes to low, it indicates data is ready for retrieval. By applying 25~27 positive clock pulses at the PD_SCK pin, data is shifted out from the DOUT output pin. Each PD_SCK pulse shifts out one bit, starting with the MSB bit first, until all 24 bits are shifted out. The 25th pulse at PD_SCK input will pull DOUT pin back to high (Fig.2).

Input and gain selection is controlled by the number of the input PD_SCK pulses (Table 3). PD_SCK clock pulses should not be less than 25 or more than 27 within one conversion period, to avoid causing serial communication error.

PD_SCK Pulses	Input channel	Gain
25	A	128
26	B	32
27	A	64

Table 3 Input Channel and Gain Selection

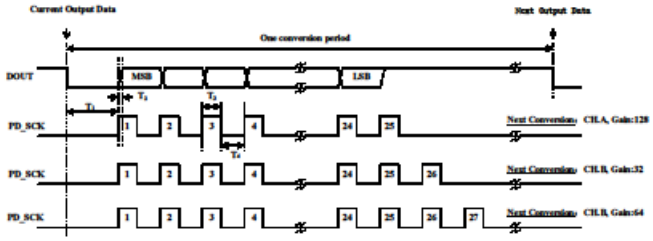


Fig.2 Data output, input and gain selection timing and control

Symbol	Note	MIN	TYP	MAX	Unit
T_1	DOUT falling edge to PD_SCK rising edge	0.1			μs
T_2	PD_SCK rising edge to DOUT data ready			0.1	μs
T_3	PD_SCK high time	0.2	1	50	μs
T_4	PD_SCK low time	0.2	1		μs

Reset and Power-Down

When chip is powered up, on-chip power on rest circuitry will reset the chip.

Pin PD_SCK input is used to power down the HX711. When PD_SCK Input is low, chip is in normal working mode.

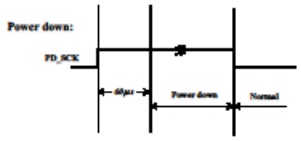


Fig.3 Power down control

When PD_SCK pin changes from low to high and stays at high for longer than 60μs, HX711 enters power down mode (Fig.3). When internal regulator is used for HX711 and the external transducer, both HX711 and the transducer will be powered down. When PD_SCK returns to low, chip will reset and enter normal operation mode.

After a reset or power-down event, input selection is default to Channel A with a gain of 128.

Application Example

Fig.1 is a typical weigh scale application using HX711. It uses on-chip oscillator (XI=0), 10Hz output data rate (RATE=0). A Single power supply (2.7~5.5V) comes directly from MCU power supply. Channel B can be used for battery level detection. The related circuitry is not shown on Fig. 1.

Reference PCB Board (Single Layer)

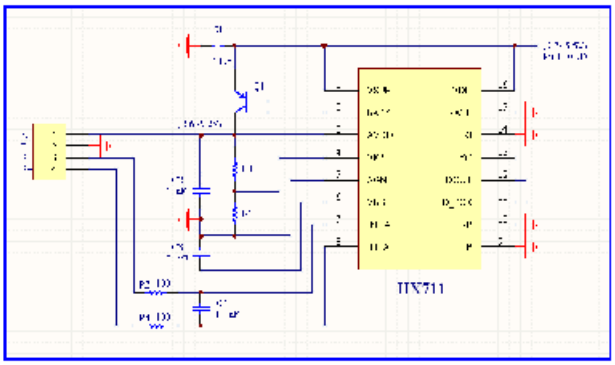


Fig.4 Reference PCB board schematic

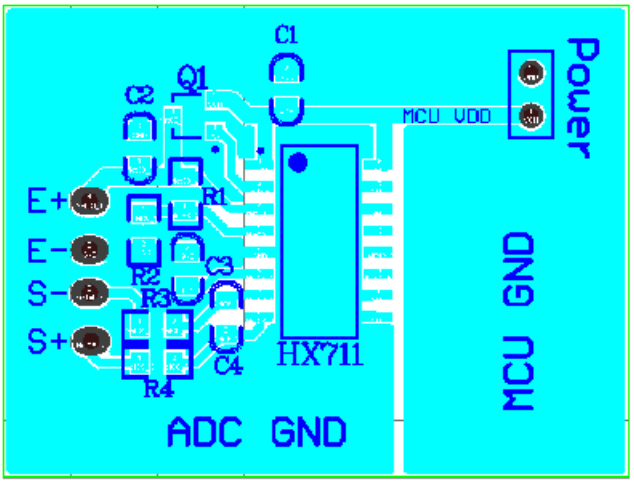


Fig.5 Reference PCB board layout

Reference Driver (Assembly)

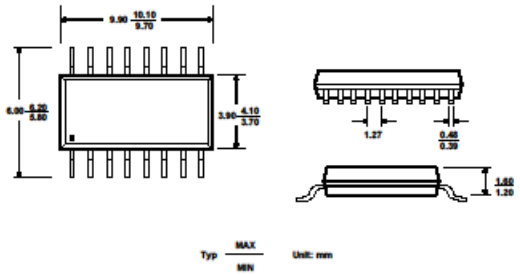
```
/*------------------------------------------------------------
Call from ASM:      LCALL   ReaAD
Call from C:    extern unsigned long ReadAD(void);
                  .
                  .
                unsigned long data;
                data=ReadAD();
                 .
                 .
------------------------------------------------------------*/
PUBLIC          ReadAD
HX711ROM        segment  code
rseg            HX711ROM

sbit            ADDO = P1.5;
sbit            ADSK = P0.0;
/*-------------------------------------------
 OUT:   R4, R5, R6, R7   R7=>LSB
-------------------------------------------*/
ReadAD:
    CLR     ADSK                //AD Enable (PD_SCK set low)
    SETB    ADDO                //Enable 51CPU I/O
    JB      ADDO,$              //AD conversion completed?
    MOV     R4,#24
ShiftOut:
    SETB    ADSK                //PD_SCK set high (positive pulse)
    NOP
    CLR     ADSK                //PD_SCK set low
    MOV     C,ADDO              //read on bit
    XCH     A,R7                //move data
    RLC     A
    XCH     A,R7
    XCH     A,R6
    RLC     A
    XCH     A,R6
    XCH     A,R5
    RLC     A
    XCH     A,R5
    DJNZ    R4,ShiftOut          //moved 24BIT?
    SETB    ADSK
    NOP
    CLR     ADSK
    RET
    END
```

Reference Driver（C）

```
//-----------------------------------------------------------------
sbit   ADDO = P1^5;
sbit   ADSK = P0^0;
unsigned long ReadCount(void){
  unsigned long Count;
  unsigned char i;
  ADDO=1;
  ADSK=0;
  Count=0;
  while(ADDO);
  for (i=0;i<24;i++){
    ADSK=1;
    Count=Count<<1;
    ADSK=0;
    if(ADDO) Count++;
  }
  ADSK=1;
  Count=Count^0x800000;
  ADSK=0;
  return(Count);
}
```

Package Dimensions



SOP-16L Package