GOVERNO DO ESTADO DO RIO DE JANEIRO
SECRETARIA DE ESTADO DE CIÊNCIA, TECNOLOGIA E INOVAÇÃO - SECTI
FUNDAÇÃO DE APOIO À ESCOLA TÉCNICA - FAETEC
ESCOLA TÉCNICA ESTADUAL FERREIRA VIANA - ETEFV
COORDENAÇÃO DE ELETRÔNICA

Ana Claudia Ramos Mariano
Italo Albuquerque Nunes Abrahão
Luiz Henrique dos Santos da Silva

ALIMENTADOR AUTOMÁTICO PARA GATOS DOMÉSTICOS

Rio de Janeiro
2024

Ana Claudia Ramos Mariano
Italo Albuquerque Nunes Abrahão
Luiz Henrique dos Santos da Silva

ALIMENTADOR AUTOMÁTICO PARA GATOS DOMÉSTICOS

Trabalho de Conclusão de Curso apresentado à ETE Ferreira Viana, da Fundação de apoio à Escola Técnica, como requisito parcial para a obtenção da habilitação profissional de Técnico de Nível Médio em Eletrônica sob a orientação do Professor Luis Henrique Monteiro de Castro e do Professor Marcelo de Almeida Duarte.

Rio de Janeiro
2024

Ana Claudia Ramos Mariano
Italo Albuquerque Nunes Abrahão
Luiz Henrique dos Santos da Silva

ALIMENTADOR AUTOMÁTICO PARA GATOS DOMÉSTICOS

Aprovada em : ____ / _____ / _____ Conceito: _____

_____	_____
Prof. Luis Henrique M. de Castro ETE Ferreira Viana Orientador	Prof. Marcelo de Almeida Duarte ETE Ferreira Viana Orientador

_____
Prof. XXXXXXXXXXXXXX
ETE Ferreira Viana
ID:

_____
Prof. XXXXXXXXXXXXXX
ETE Ferreira Viana
ID:

_____
Prof. XXXXXXXXXXXXXX
ETE Ferreira Viana
ID:

Rio de Janeiro
2024

RESUMO

MARIANO, Ana Claudia Ramos; ABRAHÃO, Italo Albuquerque Nunes; DA SILVA, Luiz Henrique dos Santos. *Alimentador automático para gatos domésticos.* 2024. Trabalho de Conclusão de Curso - Curso técnico em Eletrônica, Escola Técnica Estadual Ferreira Viana, Fundação de Apoio à Escola Técnica, Rio de Janeiro, 2024.

No Brasil, vem crescendo o número de adoções de gatos. Tal cenário aliado à rotina corrida provocaram a idealização do projeto Alimentador automático para gatos domésticos. Este visa a saúde e a otimização do tempo através da automatização do processo de alimentação do felino. A proposta é estruturada por um conjunto de reservatórios controlados pela plataforma de prototipagem Arduino. Ao usuário, será possível, através de uma aplicação, estabelecer a quantidade de ração e o horário das refeições. A interface também exibirá a ração consumida e armazenada. Caso desejado, o aplicativo alertará ao cliente quando a porcentagem mínima de ração for atingida. Esta será estabelecida pelo dono do animal. O reservatório primário, onde será depositado o total de alimento, terá capacidade de 2,0 kg, enquanto o pote do pet, 100g. O projeto contará com um sistema de balanças e medidores de nível que se comunicarão com o controlador.

Palavras-chave: Automatização; Arduino; Alimentação

ABSTRACT

MARIANO, Ana Claudia Ramos; ABRAHÃO, Italo Albuquerque Nunes; DA SILVA, Luiz Henrique dos Santos. *Automatic feeding for domestic cats*. 2024. Trabalho de Conclusão de Curso - Curso técnico em Eletrônica, Escola Técnica Estadual Ferreira Viana, Fundação de Apoio à Escola Técnica, Rio de Janeiro, 2024.

In Brazil, the number of cat adoptions has been growing. This scenario, combined with the busy routine, instigated the creation of the Feeder automatic for domestic cats project. This aims to improve health and optimize time by automating the feline's feeding process. The proposal is structured by a set of reservoirs controlled by the Arduino prototyping platform. The user will be able, through an application, to establish the amount of food and meal times. The interface will also display the feed consumed and stored. If desired, the application will alert the customer when the minimum feed percentage has been reached. This will be established by the animal's owner. The primary reservoir, where the total food will be deposited, will be able to store 2.0 kg, while the pet pot will have a capacity of 100g. The project will have a system of scales and level meters that will communicate with the controller.

Keywords: Automation; Arduino; Feeding

SUMÁRIO

1. INTRODUÇÃO

De acordo com o censo realizado pelo Instituto Pet Brasil, a população de gatos domésticos brasileira cresceu 6% entre os anos de 2020 e 2021, demonstrando que mais pessoas optam por eles quando vão adotar. Contudo, esses felinos necessitam de cuidados específicos, sobretudo relacionados à saúde. O estilo de vida sedentário, a alimentação e a predisposição genética são fatores que tornam a obesidade a principal enfermidade que atinge os gatos domésticos. Controlar a quantidade e a frequência que o animal se alimenta é essencial para minimizar e prevenir o sobrepeso. Apesar disso, a falta de tempo dos tutores ocasionada pela rotina atarefada torna essa tarefa secundária e negligenciada.

Portanto, visando resolver a dificuldade, este projeto foi desenvolvido. Pode ser resumido como um dispositivo que oferta ração ao gato em horários e porções definidas pelo tutor por meio de um aplicativo. O processo é controlado pelo Arduino e é composto por dois motores servos, uma célula de carga e um sensor de ultrassom. Os motores são responsáveis por permitir que a ração chegue ao recipiente, a célula de carga é utilizada para pesar a quantidade de ração definida e o sensor de ultrassom tem como objetivo verificar quanto de comida há no reservatório. Caso esteja no nível abaixo daquele determinado, o usuário será alertado através da aplicação.

O objetivo do projeto é garantir a alimentação de gatos domésticos com intervenção mínima da ação humana, utilizando a plataforma de prototipagem Arduino como centro de controle do processo.

O projeto é estruturado em três capítulos além da introdução. O segundo esclarece os componentes utilizados e o esquema elétrico produzido. O próximo tem o intuito de apresentar o desenvolvimento do programa que controlará o alimentador e o aplicativo responsável pela interface homem-máquina. O último e quarto capítulo descreve o funcionamento geral do sistema.

2. METODOLOGIA

O projeto foi inicialmente pensado a partir da coleta de informações acerca da população de gatos domésticos no Brasil e obesidade, doença frequente nesses animais, no Instituto Pet Brasil e em plataformas de empresas que prestam serviço a esses pets. Em seguida, foram realizadas pesquisas acerca dos componentes e materiais apropriados de acordo com as necessidades do protótipo.

Foi utilizado como material auxiliar o trabalho Alimentador automático para animais domésticos com supervisão e controle via smartphone (LEITE FILHO, 2023) para o arquitetar os sistemas eletrônicos do projeto. Assim, como o protótipo desenvolvido no Instituto Federal da Paraíba (IFPB), este também utilizará a plataforma de prototipagem arduino para controlar e automatizar o processo de nutrição do animal. Contudo, diferentemente do proposto por Filho, o sistema responsável por permitir a passagem e a pesagem da ração é composto por três motores servos e uma seção responsável por mensurar o alimento antes que seja disponível ao animal. Desta forma, devido à simplicidade na programação dos motores servos, é possível atenuar a complexidade do sistema. Ademais, a pesagem da ração torna-se mais precisa, pois é impassível de sofrer interferências externas.

O tutor do animal poderá monitorar os níveis de ração disponíveis e os horários de alimentação a partir da comunicação entre o Arduino e o Smartphone estabelecida por um módulo Bluetooth.

A interface que permitirá ao usuário configurar o horário e a quantidade de alimento que deve ser disponibilizado ao animal será incrementada pelo software APP Inventor que possibilita a criação de aplicativos para Android que se comuniquem com o Arduino, a partir de um pareamento Bluetooth. A programação é feita de forma simples: “arrastando” e “soltando” blocos.

Foram realizados softwares para simular, desenvolver e arquitetar o projeto, como SimulIde, TinkerCad, Dia e Arduino IDE.

2.1. Componentes

2.1.1. Arduino

Em 2005, na Itália, um grupo de estudantes de Design desenvolveu uma placa de circuito com o microcontrolador ATmega 128, cuja interação entre hardware e usuário é feita pela linguagem Wiring, também criada por Benzi, Reas e Fry. Assim, surgiu o Arduino, plataforma de prototipagem de fácil controle e entendimento. A Integrated Development Environment - IDE - é compatível com diversos ambientes, como Windows, Mac e Linux, possibilitando a versatilidade da programação. Para o protótipo, será usado a versão Uno.

2.1.2. Servo Motor

É um componente eletromecânico utilizado para movimentação, permitindo rotação de até 180°. Este motor é largamente utilizado na automação devido a precisão e a capacidade de oferecer respostas em tempo real. Portanto, é ideal para a interface de controle do Arduino.

No projeto, são implementados três motores servos, chamados de Servo Disco, Servo Porta e Servo Alavanca.

2.1.3. Célula de Carga

É um dispositivo utilizado para medir a massa de um corpo. O peso de um objeto realiza uma deformação em sua estrutura, modificando sua resistência e consequentemente, gerando uma resposta em tensão ou corrente. Para o Arduino, é recomendável utilizar uma célula de carga do tipo Strain Gauge acoplada a um módulo Hx711, um conversor analógico/digital. Devido aos seus atributos, este é o componente que pesa o alimento.

2.1.4. Sensor Ultrassônico

É um sensor capaz de medir distâncias através da velocidade do som. Quando acionado, envia uma onda sonora que, ao encontrar um obstáculo, é refletida para o dispositivo. Sabendo-se a velocidade da onda e o tempo necessário para o sinal retornar, é possível calcular a distância. No protótipo, é o responsável por aferir a quantidade de ração no reservatório.

2.1.5. Módulo RTC

É um dispositivo que contém um relógio em tempo real (Real Time Clock), comunicando tais dados a partir de protocolo I2C, comunicação serial que permite que dispositivos diferentes sejam conectados em um mesmo barramento. No projeto, será utilizado o modelo DS1307.

2.1.6. Módulo Bluetooth

É um módulo que permite que o microcontrolador receba e envie informações por meio de Bluetooth. Esta é uma tecnologia de comunicação por ondas de rádio, com alcance de até dez metros. É uma maneira gratuita, pois não é necessário pagar taxas ou conectá-los à internet. Assim, é possível conectar o Arduino ao celular do usuário facilmente.

2.2. Funcionamento

O intuito do projeto é alimentar o animal em períodos e porções pré-determinadas pelo tutor sem interferência humana. Para tal, um Arduino Uno será utilizado para automatizar o processo, controlando três motores servos e recebendo informações da célula de carga e do sensor ultrassom.

Em primeira instância, quando o horário estabelecido for atingido, dado enviado pelo módulo RTC, o Servo Disco será acionado, permitindo o acesso da ração do reservatório à câmara, local responsável por mensurar o alimento. Nessa etapa, o Servo Disco se encontra em regime de alternância entre manter a passagem aberta e fechada até a quantidade desejada de comida ser alcançada. Essa informação é constantemente enviada para o Arduino pela célula de carga, fixada na câmara.

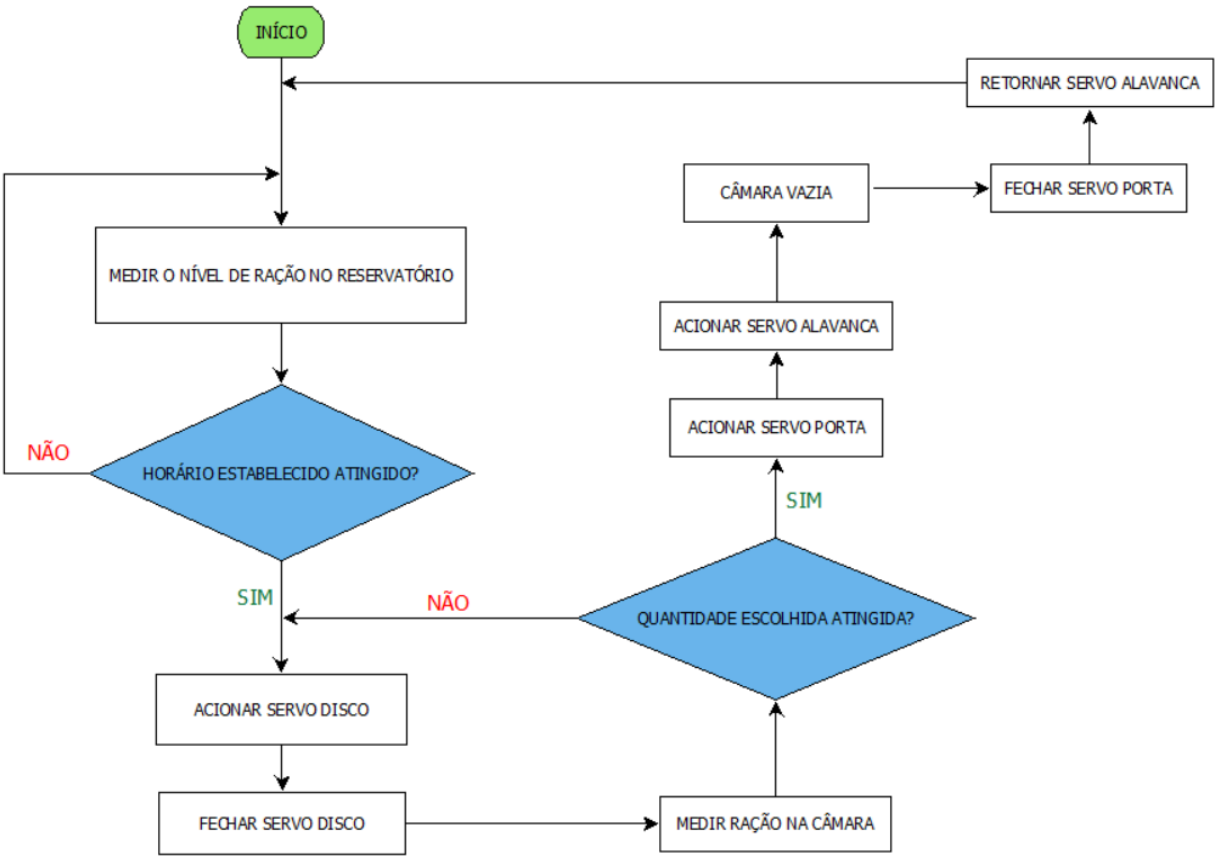
Ao completar-se a porção estabelecida pelo usuário, o Servo Disco selará a passagem. Portanto, o Servo Porta abrirá uma seção da câmara. Em seguida, o Servo Alavanca é ativado. Ele controla uma estrutura semelhante a um braço, na qual é atribuída a tarefa de

empurrar a ração da câmara para o pote de ração através da abertura proporcionada pelo acionamento do Servo Porta.

É essencial ressaltar que o sensor ultrassom está constantemente compartilhando dados acerca da disponibilidade de alimento no reservatório a fim de notificar o usuário quando a quantidade estiver próxima de se tornar insuficiente. Essa quantia e os horários serão configurados quando o tutor inicializar o aplicativo.

O funcionamento do sistema é representado por um fluxograma projetado no software Dia.

Figura 2.2.1: Fluxograma do funcionamento.



3. DESENVOLVIMENTO

3.1. Modelo e Design

Foi elaborada, em primeira instância, a estrutura do protótipo. Esta foi modelada com papelão devido à praticidade e ao baixo custo. De acordo com o percurso realizado pelo alimento, pode ser dividida em três partes principais: reservatório, câmara e pote de ração. Todas as subestruturas conversam entre si por meio do acoplamento proporcionado por um suporte e pela movimentação dos servos.

O reservatório possui o formato cilíndrico, com aproximadamente 7,5cm de raio e 25cm de altura. Desta forma, é possível armazenar até 4,5kg de ração, aproveitando todo o espaço do poliedro. Também foi feita uma tampa para vedar o reservatório de modo a impedir que corpos indesejados entrem em contato com o alimento. Nesta estrutura, além de um sensor ultrassom para medir o nível de ração disponível, foi projetado um funil, também com papelão, para facilitar a passagem do alimento para a câmara. O molde arredondado do reservatório contribuiu para isso.

Figura 3.1.1: Reservatório com tampa.

O que separa o reservatório e a câmara é um disco com uma seção transversal cortada. Ao rotacionar, este item irá permitir que o alimento vá para a câmara quando o horário estabelecido for alcançado.

A câmara foi uma subestrutura arquitetada com o intuito de resolver o problema da pesagem. Inicialmente, foi pensado em medir a comida no pote de ração. Entretanto, o processo sofreria interferências do animal que, impaciente, poderia ingerir a ração antes do fim da pesagem. Por isso, esta função foi atribuída à câmara, onde está fixada a célula de

carga. Suas dimensões foram projetadas para receber pequenas porções de ração, suficientes para a nutrição do gato doméstico.

Para que, durante a mensuração, os grãos não caiam para o pote de ração, foi fixada uma portinhola na câmara. Ao fim do processo, a ração será empurrada para a próxima subestrutura por um elemento semelhante a um braço que "varrerá" a câmara.

A câmara e o pote são conectados por uma rampa. Diferentemente do resto da infraestrutura, essa união foi realizada com um pedaço de garrafa Pet por causa de sua maleabilidade. O pote de ração possui 6cm de raio e 3cm de altura.

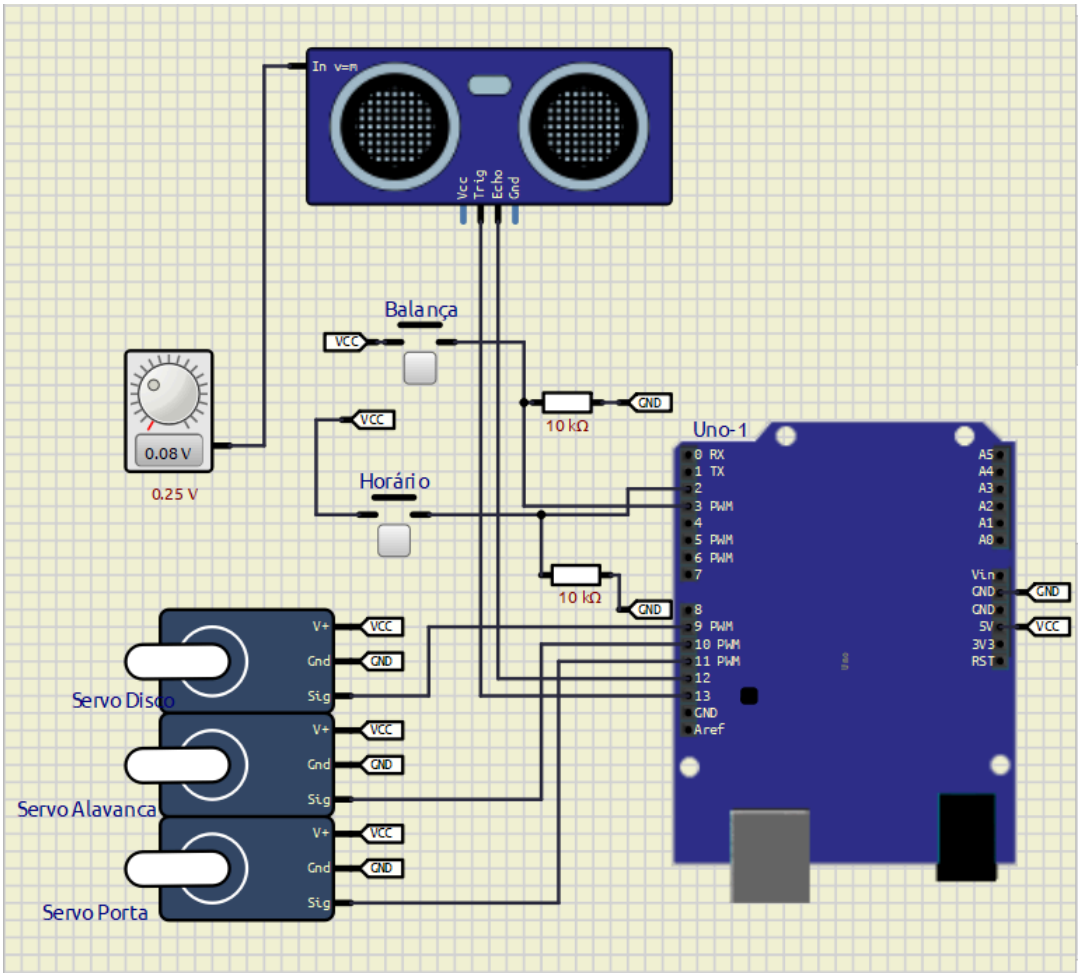
Na base do suporte, utilizou-se uma caixa de sapato com proporções ideais para o acoplamento e para armazenar o circuito projetado.

As demais medidas e modelos projetados estão no anexo C.

3.2. Programação

Em paralelo com o modelo, foi desenvolvido um programa piloto simulando a coordenação dos motores servos. Para tal, utilizou-se o aplicativo *SimulIde* e o processo foi simplificado: o módulo RTC e a célula de carga foram substituídos por botões NA (normalmente aberto).

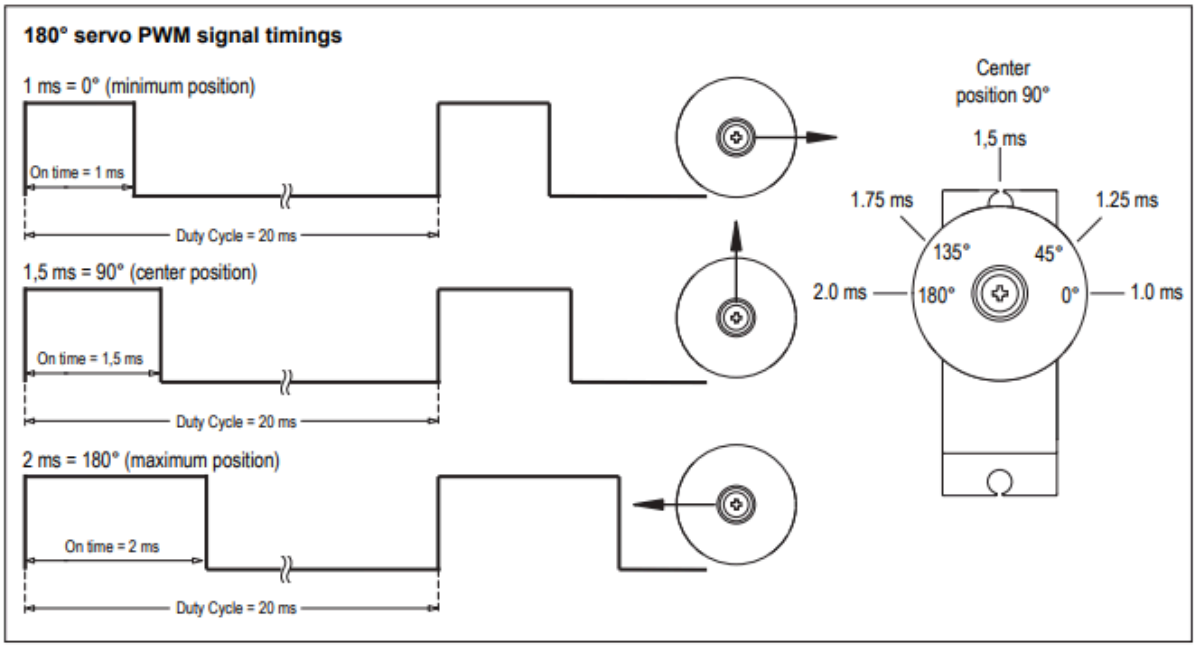
Figura 3.2.1: Circuito implementado.



Em princípio, decidiu-se recorrer às bibliotecas para controlar os motores. No contexto da programação, bibliotecas são um conjunto de instruções desenvolvidas para simplificar e facilitar a execução de alguma tarefa, principalmente quando relacionadas a dispositivos externos ao microcontrolador. Estes códigos podem ser desenvolvidos e utilizados por quaisquer pessoas, de forma que a própria IDE do Arduino possui bibliotecas padrões pré-instaladas, como a Servo.h, utilizada primeiramente. Apesar da praticidade, esta aplicação não foi capaz de atender às necessidades do sistema: o motor não permanecia na posição desejada quando se controlava outro servo. Invés disso, o componente retornava para a posição original. Portanto, se tornou necessário controlar os servos sem o auxílio de bibliotecas.

A maioria dos servos motores funciona em um ciclo em que a duração do pulso positivo determina a angulação da sua rotação. O SG90, modelo escolhido para o projeto, possui um ciclo de 20ms e seu *datasheet* apresenta informações acerca do período do pulso e sua posição correspondente. Desta forma, é possível controlar este motor com o Arduino ao determinar em quanto tempo o sinal PWM ficará em nível alto.

Figura 3.2.2: Pulsos positivos x ângulos.



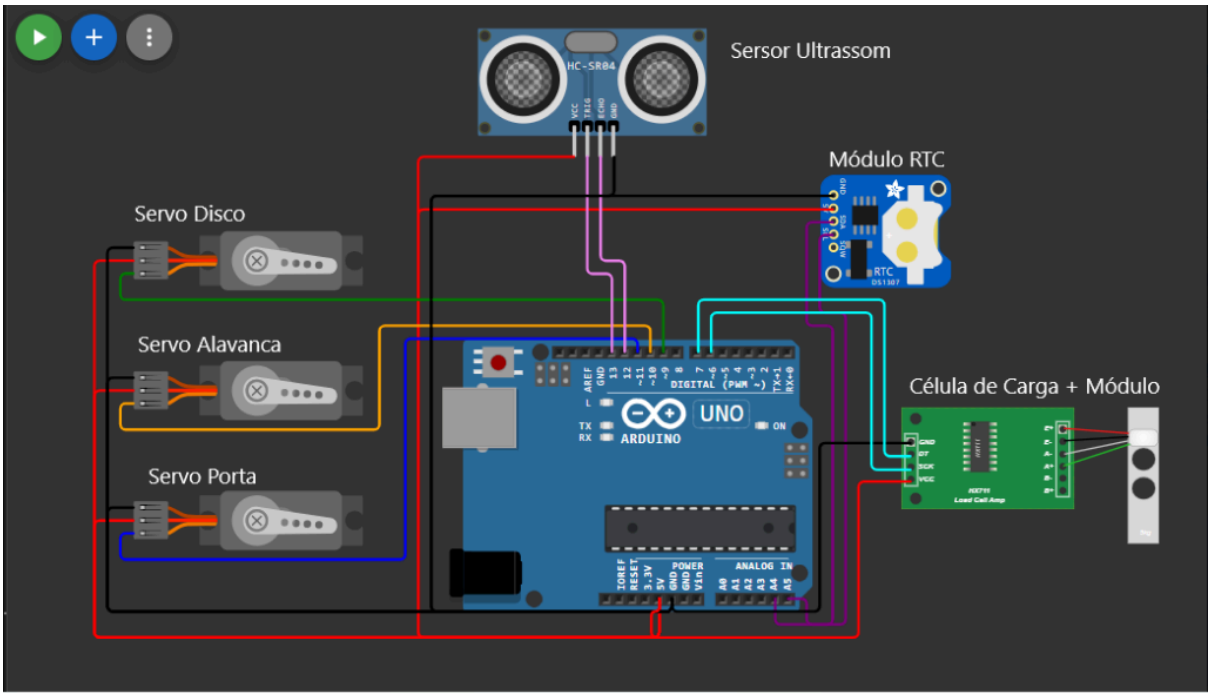
Também foi utilizada uma biblioteca para captar as informações do sensor ultrassom. Neste componente, há uma entrada em que o valor da tensão injetada gera uma leitura de distância de mesmo valor. Assim, foi utilizado esse recurso para simular o funcionamento do dispositivo.

A partir da operação esperada desta etapa, os botões foram substituídos respectivamente pelo módulo RTC e pela célula de carga com o módulo Hx711. Para tal, foi necessário mudar o simulador. A versão utilizada do *SimulIde* (1.1.0) não possui esses componentes. Assim, para evitar ocupar a memória da máquina a cada atualização de software, o *Wokwi* foi o substituto. Este pode ser acessado via Internet e permite, a partir de

um login, salvar os projetos de forma online, facilitando a visualização e a edição do código e do circuito.

Ademais, utilizou-se bibliotecas para operar o módulo RTC e a célula de carga, simplificando o controle e encurtando o código. Ainda não é possível escolher um horário por meio de um aplicativo: isto deve ser feito no próprio programa, inserindo a hora e o minuto requerido nas respectivas variáveis inteiras *horarioHora* e *horarioMinuto*. O mesmo ocorre com a porção de ração: deve ser estabelecida definindo um valor da variável *balancaUser*. É importante ressaltar que, no simulador, é possível alterar quanto a célula de carga está medindo (kg). A interface serial deve mostrar o nível do reservatório, tempo e a quantidade de alimento a cada dois segundos.

Figura 3.2.3: Novo circuito.



O programa desenvolvido após todas estas etapas e substituições está no Anexo A.

Em seguida, após extensas simulações, a fase do projeto avançou para o mundo físico.

3.3. Calibração da célula de carga

Ao testar o funcionamento dos componentes na vida real, tornou-se fundamental atender demandas do circuito que não eram necessárias nas simulações. Primeiramente, foi preciso calibrar a célula de carga para que medisse as massas corretamente. Para isso, utilizou-se um programa próprio para tal já existente para encontrar o fator de calibração e objetos com massas conhecidas. Neste caso, foram utilizadas moedas de cinco, dez , vinte e cinco e cinquenta centavos.

A célula de carga foi fixada em uma base para balanças, um superfície retilínea, a fim de evitar erros de medições ocasionados por inclinações ou movimentos oscilatórios. A partir disso, as moedas eram postas uma de cada vez sobre o componente e o fator de calibração ajustado até o valor correto de massa ser alcançado. A calibração foi realizada dessa forma,

com “chutes”, porque encontrar a relação entre o peso e o fator de calibração exigia conhecimentos matemáticos além daqueles adquiridos no Ensino Médio.

Ao final de vários testes, apesar de uma imprecisão de 2 a 3g, o fator de calibração mais próximo da realidade foi de -700.

Quando a balança foi acoplada ao modelo de papelão, foi preciso reajustar o valor de calibração devido às particularidades da estrutura. O valor encontrado e utilizado ao final do projeto foi de -540, que conferiu uma margem de erro de aproximadamente ±1g.

O programa utilizado encontra-se nas referências.

3.4. Configuração do módulo RTC

Apesar de fácil uso por causa da biblioteca, o Real Time Clock ofereceu muitos problemas, funcionando corretamente, isto é, mostrando a passagem do tempo, de modo assíncrono. Embora o motivo disto fosse simples, demorou a ser identificado: o mau contato entre a conexão do módulo e do arduino. Além disso, o componente também foi resetado, tirando e recolocando a bateria que garante ao DS1307 energia mesmo sem operar com o arduino.

3.5. Interface Homem-Máquina

Também foi notado que o recolhimento de informações do usuário era simplória, pois era necessário importar o programa para o arduino toda vez que algum dado era alterado. Além disso, havia a possibilidade de ajustar apenas um horário. Por isso, adicionou-se mais linhas de código ao programa que permitem ao usuário escolher, no máximo, até cinco períodos de refeição. Essa comunicação foi estabelecida através do monitor serial do Arduino.

Assim que o programa se inicia, é dito ao usuário para colocar “p” para aumentar, ou seja, passear pelas possibilidades de forma crescente, e “s” para selecionar. Neste caso, o usuário é capaz de escolher quais dos horários deseja ajustar, a hora, a minutagem, de dez em dez minutos, e a quantidade de ração, de cinco em cinco gramas.

3.6. Montagem

Após os ajustes descritos acima, foi realizada a união entre os componentes do sistema e o modelo de papelão. Para fixar as estruturas, foram usados fita crepe, palitos de churrasco e cola quente. Foram adicionadas a pá, responsável por empurrar a comida para o pote, a partir de recortes de isopor e palitos.

Apesar do planejamento inicial, não foi possível utilizar todo o arranjo em conjunto. Isto ocorre por causa de convergências do espaço ocupado por cada parte do protótipo, da movimentação das peças controladas pelos servos motores e da fixação dos mesmos. Portanto, o reservatório não foi acoplado, além de não haver carcaça para “esconder” as estruturas.

Por outro lado, a exposição torna possível visualizar as etapas, esclarecendo o funcionamento a fim de estudos ou continuidade do projeto. De qualquer forma, o protótipo final não foi concluído, pois necessita de melhorias no arranjo mecânico e visual, como pintura, acabamento, disfarce, entre outros.

O teste final foi realizado simulando a ração com moedas de peso conhecidas para confirmar a precisão da célula de carga e o funcionamento do sistema.

Figura 3.5.1: Protótipo final.

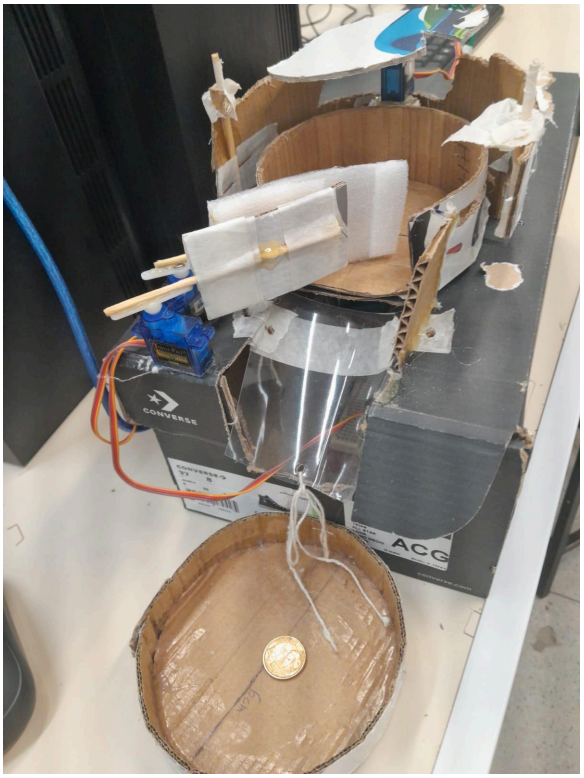


Figura 3.5.2: Teste final, realizado no dia 09/11/24.

4. CONCLUSÃO

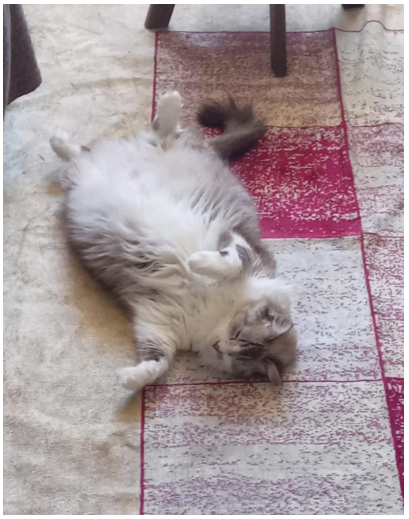
O projeto foi de suma importância para o aprendizado e desenvolvimento de práticas e conceitos apresentados em sala de aula. Mesmo com esforços, há etapas idealizadas não implementadas. Uma delas é a criação de um aplicativo capaz de recolher os dados configurados pelo usuário e enviá-los ao arduino para a inicialização do sistema. O programa desse artifício teve o início de sua elaboração interrompido por causa da necessidade de focar em partes indispensáveis para o funcionamento do sistema, como a calibração da célula de carga. Por isso, embora tenham sido apresentados como, respectivamente, um dos componentes a serem usados e um dos softwares de desenvolvimento, o módulo Bluetooth e o APP Inventor não foram implementados. A comunicação com o usuário ficou restrita ao monitor serial do Arduino.

Como possibilidade para trabalhos futuros, é válido considerar a melhora da aparência do alimentador, assim como o encaixe e a fixação das estruturas, de forma que a operação de cada uma não seja prejudicial ao de outra. Também seria interessante um sistema de alimentação do circuito por meio de baterias, garantindo o funcionamento do alimentador mesmo com quedas de energia na rede. Ademais, a principal opção de aprimoramento seria a introdução de uma interface de comunicação que possibilite ao usuário customizar os horários, receber notificações acerca do nível de ração do reservatório e do status da bateria a fim de evitar contratempos, a partir de um dispositivo móvel, como o celular.

De todo modo, a elaboração e execução do projeto conferiu o contato com um tipo de sistema amplamente implementado em várias áreas industriais e cotidianas: autônomos com a utilização de diversos sensores e atuadores em conjunto, enviando e recebendo informações ao microcontrolador, com o objetivo de agirem sem a necessidade da interferência humana.

A partir desse projeto, visa-se, também, inspirar novas iniciativas no ambiente escolar e motivar o aprimoramento desse alimentador automático, a fim de transformá-lo em um produto de fácil comunicação e aquisição para auxiliar na manutenção da saúde dos gatos domésticos.

Figura 4.1: Animal privilegiado pela existência e possível aprimoramento do alimentador automático.



5. REFERÊNCIAS

Censo Pet IPB: com alta recorde de 6% em um ano, gatos lideram crescimento de animais de estimação no Brasil. **Instituto Pet Brasil**, 2022. Disponível em: <https://institutopetbrasil.com/fique-por-dentro/amor-pelos-animais-impulsiona-os-negocios-2-2/>. Acesso em: 08 jul. de 2024.

6 fatores de risco para obesidade em gatos. **Royal Canin.** Disponível em: <https://portalvet.royalcanin.com.br/saude-e-nutricao/controle-de-peso/riscos-do-sobrepeso-e-obesidade-em-gatos/#h2-0>. Acesso em: 08 jul. de 2024.

Cada vez mais brasileiros veem pets como filhos, tendência criticada pelo papa. **BBC**, 2022. Disponível em: <https://www.bbc.com/portuguese/geral-59989766>. Acesso em: 09 jul. de 2024.

Arduino Reference. **Arduino.** Disponível em:<https://www.arduino.cc/reference/en/>. Acesso em: 04 jul. 2024

FILHO, Demarkson Leite. **Alimentador automático para animais domésticos com supervisão e controle via smartphone**. 2023. Trabalho de Conclusão de Curso - Curso Superior de Automação Industrial, Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Campus Cajazeiras. Paraíba, 2023.
Disponível em: <https://repositorio.ifpb.edu.br/handle/177683/2892>. Acesso em: 09 jul. 2024.

A história do Arduino Parte 1: Apresentando o Arduino. **Embarcados.** Disponível em: <https://embarcados.com.br/a-historia-do-arduino-parte-1-apresentando-o-arduino/>. Acesso em : 11 jul. 2024.

Comunicação I2C. **Embarcados**, 2023. Disponível em: <https://embarcados.com.br/comunicacao-i2c/>. Acesso em 11 jul.2024

O que é bluetooth?. **Superinteressante**, 2016. Disponível em: <https://super.abril.com.br/tecnologia/o-que-e-bluetooth>. Acesso em: 10 jul. 2024.

Datasheet do SG90. Disponível em: <https://www.kjell.com/globalassets/mediaassets/701916_87897_datasheet_en.pdf?ref=4287817A7A> Acesso em: 26 jul. 2024.

Biblioteca do sensor ultrassom (Ultrasonic.h). **Arduino.** Disponível em: <https://www.arduino.cc/reference/en/libraries/ultrasonic/> Acesso em: 05 ago. 2024.

Biblioteca do módulo RTC (RTClib.h). **Arduino**. Disponível em: <https://www.arduino.cc/reference/en/libraries/rtclib/>. Acesso em: 05 ago. 2024

Biblioteca da célula de carga com módulo Hx711 (HX711_light.h). **Arduino**. Disponível em: <https://www.arduino.cc/reference/en/libraries/hx711_light/>. Acesso em: 05 ago. 2024.

Código de calibração da célula de carga. **Robocore**, 2019. Disponível em: https://www.robocore.net/tutoriais/celula-de-carga-hx711-com-arduino?srsltid=AfmBOoqtuxn3PtL7xcn0EbDN5_Bnld-53UA979MaD03Yddcb6PFjTdsj. Acesso em 06 oct. 2024.

6. ANEXO A - CÓDIGO ARDUINO

A.1 Código piloto para o sensor ultrassônico e o controle dos motores:

```
#include <Ultrasonic.h>
#include <RTClib.h>
#include <HX711_light.h>

int horarioHora;  //valores que o usuário ajustará
int horarioMinuto;  //valores que o usuário ajustará
int horarioHoraRTC;
int horarioMinutoRTC;
unsigned long tempoAtual;
unsigned long tempoAnt;
int distancia;
long balancaUser;  //valores que o usuário ajustará
long balancaHX;
int porcentagem;
Ultrasonic ultrasonic(13, 12);
RTC_DS1307 rtc;
constexpr uint8_t kClockPin= 6;
constexpr uint8_t kDataPin= 7;
HX711_light HX711(kDataPin, kClockPin);

void setup(){
        HX711.begin();
        HX711.setChannelAndGain(HX711_light::CH_A_128);
        Serial.begin(115200);
if (! rtc.begin()) {
  Serial.println("Deu ruim aqui, meu embaixador");
  Serial.flush();
  abort();
 }
}

void loop(){
        tempoAtual= millis();
if(tempoAtual-tempoAnt >= 2000){
        Ultrassom();
        tempoAnt= tempoAtual;
}
```

```
        horario();
}
void Ultrassom(){
        distancia= ultrasonic.read();
        Serial.print("Sua distancia vale ");
        Serial.print(distancia);
        Serial.println("cm");
        porcentagem= map(distancia, 1, 25, 0, 100);
        porcentagem= 100-porcentagem;
        Serial.print("Seu reservatorio esta ");
        Serial.print(porcentagem);
        Serial.println("% cheio");
}
void horario(){
        tempoAtual= millis();
if(tempoAtual-tempoAnt >= 2000){
        DateTime now = rtc.now();
        Serial.print("Horario- ");
        Serial.print(now.hour(), DEC);
 Serial.print(':');
 Serial.print(now.minute(), DEC);
        Serial.println();
        horarioHoraRTC= now.hour();
        horarioMinutoRTC= now.minute();
if(horarioHora==horarioHoraRTC && horarioMinuto==horarioMinutoRTC){
                balanca();
        }
        tempoAnt= tempoAtual;
}
}

void balanca(){
        balancaHX= HX711.readData();
        Serial.print(balancaHX);
        Serial.println("g");
if(balancaHX >= balancaUser){
        servoP();}
else{
        servoD();}
}

void servoD(){
for(int j=0;j<100;j++){
        digitalWrite(9, HIGH);
```

```
        delayMicroseconds(600);
        digitalWrite(9, LOW);
for(int i=0;i<32;i++)delayMicroseconds(600);}
for(int j=0;j<100;j++){
        digitalWrite(9, HIGH);
        delayMicroseconds(1500);
        digitalWrite(9, LOW);
for(int i=0;i<12;i++)delayMicroseconds(1500);}
        balanca();
}

void servoP(){
for(char i=0;i<100;i++){
        digitalWrite(11, HIGH);
        delayMicroseconds(600);
        digitalWrite(11, LOW);
for(int i=0;i<32;i++)delayMicroseconds(600);}
        servoA();
for(char i=0;i<100;i++){
        digitalWrite(11, HIGH);
        delayMicroseconds(1500);
        digitalWrite(11, LOW);
for(int i=0;i<12;i++)delayMicroseconds(1500);}
}

void servoA(){
for(char i=0;i<100;i++){
        digitalWrite(10, HIGH);
        delayMicroseconds(600);
        digitalWrite(10, LOW);
for(int i=0;i<32;i++)delayMicroseconds(600);}
for(char i=0;i<100;i++){
        digitalWrite(10, HIGH);
        delayMicroseconds(1500);
        digitalWrite(10, LOW);
for(int i=0;i<12;i++)delayMicroseconds(1500);}
}
```

A.2 Código final com a interface homem-máquina:

```
#include <Ultrasonic.h>
#include <RTClib.h>
#include <HX711.h>
```

```
#include <Wire.h>

int a= 0;
int AC= 0;
int IA= -10;
int mem= 0;
int adm= -1;
int horarioUser[6][2];
int contagem=0;
int horarioHoraRTC;
int horarioMinutoRTC;
bool travaHora;
bool travaMinuto;
unsigned long tempoAtual;
unsigned long tempoAnt;
int distancia;
long balancaUser;  //valores que o usuário ajustará
int porcentagem;
Ultrasonic ultrasonic(13, 12);
RTC_DS1307 rtc;
constexpr uint8_t kClockPin= A5;
constexpr uint8_t kDataPin= A4;
HX711 escala;
float fatorC= -540;
const int pinoDT=A0;
const int pinoSCK=A1;

void setup(){
escala.begin(pinoDT, pinoSCK);
Serial.begin(115200);
DateTime time;
time.timestamp(DateTime::TIMESTAMP_FULL);
if (! rtc.begin()) {
  Serial.println("Deu ruim aqui, meu embaixador");
  Serial.flush();
  abort();
 }
//if(rtc.lostPower()) {
      //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
       //rtc.adjust(DateTime(2024, 11, 04, 11, 25, 00)); //(ANO), (MÊS), (DIA), (HORA),
(MINUTOS), (SEGUNDOS)
  }

void loop(){
```

```
if(adm == -1){
Serial.println("p para aumentar e s para confirmar");
delay(1000);
Serial.println("c para entrar no modo config");
delay(1000);
Serial.println("Selecione seu numero de horario");
delay(1000);
adm= 0;
}
if(adm == 0){
ADMContagem();
}
tempoAtual= millis();
if(tempoAtual-tempoAnt >= 3000){
Ultrassom();
horario();
configurations();
tempoAnt= tempoAtual;
}
}

void ADMContagem(){
if(adm == 0){
contagemNHorario();
}
if(adm == 1){
contagemHorarioHora();
}
if(adm == 2){
contagemHorarioMin();
}
if(adm == 3){
contagemPeso();
}
if(adm == 4){
configurations();
}
}


void contagemNHorario(){
 a= Serial.read();
 delay(200);
 if(a == -1 || a == 10){ //nada foi pressionado
```

```
  contagemNHorario();
  delay(200);
 }
 if(a == 112){ //pressionou p para aumentar
  mem= mem+1;
  Serial.print("horario ");
  Serial.println(mem);
  a= 10;
  delay(200);
  if(mem >= 5){
   mem=0;
  }
  contagemNHorario();
 }
  if(a == 115){ //pressionou s para confirmar
   horarioUser[mem][0];
   Serial.println("Ajustado!");
   Serial.println("Agora, selecione a hora");
   a= 10;
   adm++;
   ADMContagem();
  }
 if(a != -1 && a != 112 && a!= 115 && a != 10){
  Serial.println("Comando invalido");
 }
}


void contagemHorarioHora(){
a= Serial.read();
 delay(200);
 if(a == -1 || a == 10){ //nada foi pressionado
  contagemHorarioHora();
  delay(200);
 }
 if(a == 112){ //pressionou p para aumentar
  AC= AC+1;
    if(AC == 25){
   AC=1;
  }
  Serial.print(AC);
  Serial.println(":XX");
  a= 10;
  delay(200);
```

```
  contagemHorarioHora();
 }
  if(a == 115){ //pressionou s para confirmar
    horarioUser[mem][0]= AC;
    Serial.println("Ajustado!");
    Serial.println("Agora, selecione os minutos");
    a= 10;
    adm++;
    ADMContagem();
  }
 if(a != -1 && a != 112 && a!= 115 && a != 10){
  Serial.println("Comando invalido");
 }
}

void contagemHorarioMin(){
a= Serial.read();
 delay(200);
 if(a == -1 || a == 10){ //nada foi pressionado
  contagemHorarioMin();
  delay(200);
 }
 if(a == 112){ //pressionou p para aumentar
  IA= IA+10;
     if(IA == 60){
    IA= 0;
  }
  Serial.print(AC);
  Serial.print(":");
  Serial.println(IA);
  a= 10;
  delay(200);
  contagemHorarioMin();
 }
  if(a == 115){ //pressionou s para confirmar
    horarioUser[mem][1]= IA;
    Serial.println("Ajustado!");
    Serial.print("Seu horario foi ajustado para ");
    Serial.print(AC);
    Serial.print(":");
    Serial.println(IA);
    Serial.println("Agora, selecione o peso");
    a= 10;
    adm++;
```

```
    ADMContagem();
   }
 if(a != -1 && a != 112 && a!= 115 && a != 10){
   Serial.println("Comando invalido");
 }
}

void contagemPeso(){
a= Serial.read();
 delay(200);
 if(a == -1 || a == 10){ //nada foi pressionado
   contagemPeso();
   delay(200);
 }
 if(a == 112){ //pressionou p para aumentar
   balancaUser= balancaUser+5;
   Serial.print(balancaUser);
   Serial.println("g");
   a= 10;
   delay(200);
   if(balancaUser >= 60){
    balancaUser= 0;
   }
   contagemPeso();
 }
   if(a == 115){ //pressionou s para confirmar
     Serial.println("Ajustado!");
     Serial.print("Seu peso vale ");
     Serial.print(balancaUser);
     Serial.println("g");
     Serial.println("Perfeito!");
     Serial.println("Seu Pote esta pronto para uso!");
     a= 10;
     adm++;
     ADMContagem();
   }
 if(a != -1 && a != 112 && a!= 115 && a != 10){
   Serial.println("Comando invalido");
 }
}

void configurations(){
 a= Serial.read();
 delay(200);
```

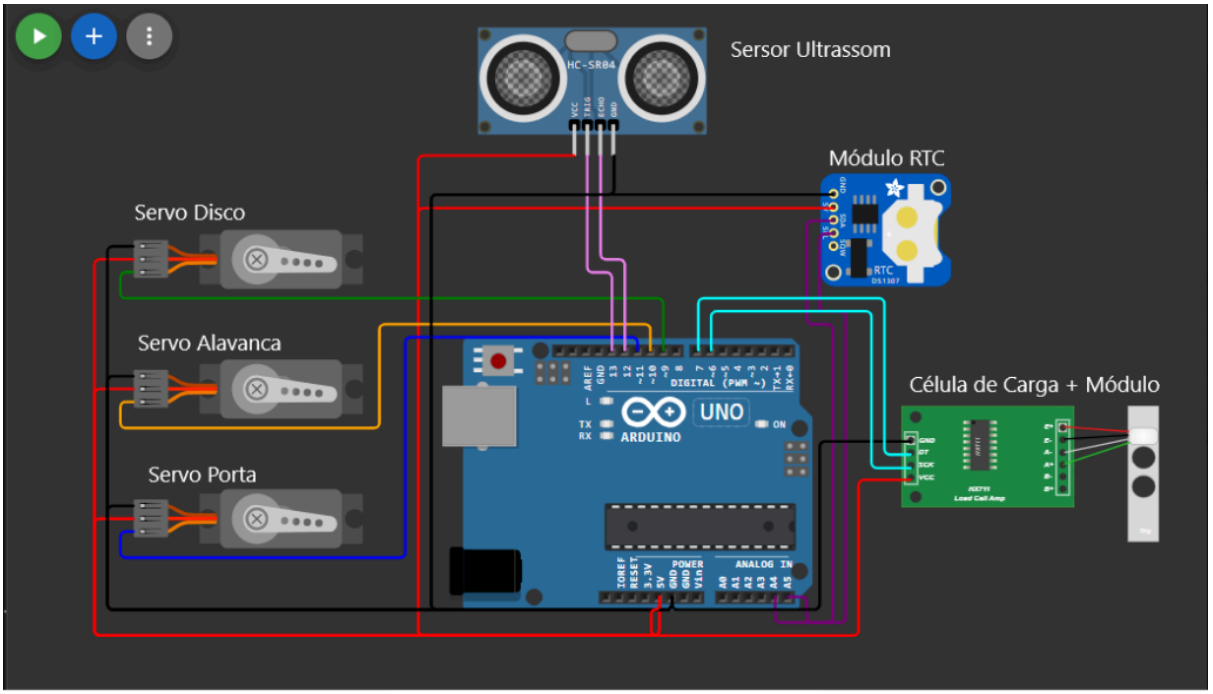
```
 if(a == 99){
   adm=0;
   Serial.println("Selecione seu numero de horario");
 }

}

void Ultrassom(){
 distancia= ultrasonic.read();
 Serial.print("Sua distancia vale ");
 Serial.print(distancia);
 Serial.println("cm");
 porcentagem= map(distancia, 1, 25, 0, 100);
 porcentagem= 100-porcentagem;
 Serial.print("Seu reservatorio esta ");
 Serial.print(porcentagem);
 Serial.println("% cheio");
 if(porcentagem <= 20){
   Serial.print("Alerta! Voce esta com ");
   Serial.print(porcentagem);
   Serial.println("% de racao restante.");
 }
 }

void horario(){
   DateTime now = rtc.now();
   Serial.print("Horario- ");
   Serial.print(now.hour(), DEC);
   Serial.print(':');
   Serial.print(now.minute(), DEC);
   Serial.println();
   horarioHoraRTC= now.hour(), DEC;
   horarioMinutoRTC= now.minute(), DEC;
   for(int i=0; i<5; i++){

       if(horarioUser[i][0] == horarioHoraRTC && horarioUser[i][0] != 0){
         if(horarioUser[i][1] == horarioMinutoRTC){
           delay(30000);
           escala.tare();
           balanca();
         }
   }
   }
 }
```

```
void balanca(){
 escala.set_scale(fatorC);
 Serial.print(escala.get_units(4), 1);
 Serial.println("g");
 if(escala.get_units(4)>= balancaUser){
  servoP();}
 else{
  servoD();}
 }

void servoD(){
 for(int j=0;j<100;j++){
  digitalWrite(9, HIGH);
 delayMicroseconds(600);
 digitalWrite(9, LOW);
 for(int i=0;i<32;i++)delayMicroseconds(600);}
 for(int j=0;j<100;j++){
  digitalWrite(9, HIGH);
 delayMicroseconds(1500);
 digitalWrite(9, LOW);
 for(int i=0;i<12;i++)delayMicroseconds(1500);}
 delay(2500);
 balanca();
 }

void servoP(){
 for(char i=0;i<100;i++){
  digitalWrite(11, HIGH);
 delayMicroseconds(600);
 digitalWrite(11, LOW);
 for(int i=0;i<32;i++)delayMicroseconds(600);}
  servoA();
 for(char i=0;i<100;i++){
  digitalWrite(11, HIGH);
 delayMicroseconds(1500);
 digitalWrite(11, LOW);
 for(int i=0;i<12;i++)delayMicroseconds(1500);}
 }

void servoA(){
 for(char i=0;i<100;i++){
  digitalWrite(10, HIGH);
 delayMicroseconds(600);
```

```
digitalWrite(10, LOW);
for(int i=0;i<32;i++)delayMicroseconds(600);}
for(char i=0;i<100;i++){
  digitalWrite(10, HIGH);
delayMicroseconds(1500);
digitalWrite(10, LOW);
for(int i=0;i<12;i++)delayMicroseconds(1500);}
}
```

7. ANEXO B - DIAGRAMA ELETRÔNICO

Figura 1B: Diagrama desenvolvido no Wokwi.



8. ANEXO C - PROTÓTIPOS

Figura 1C: Medidas do pote de ração. Raio = 6 cm. Altura = 3cm.

Figura 2C: Pote de ração.

Figura 3C: Dimensões do reservatório. Raio ~ 7,5 cm. Altura = 25cm.

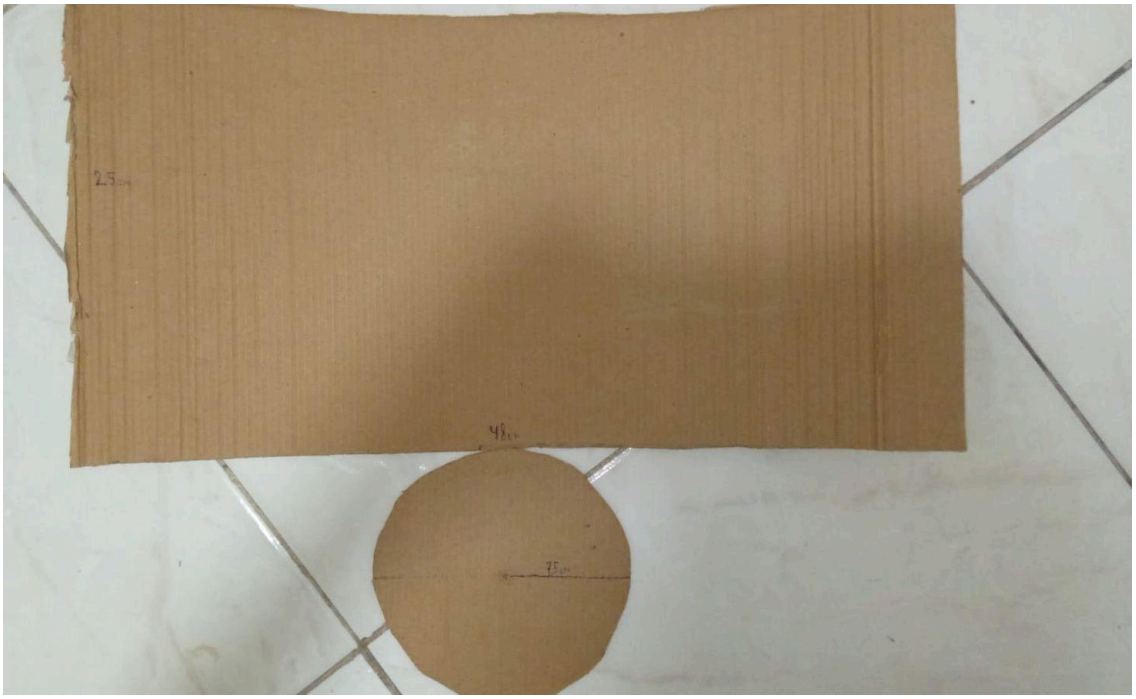


Figura 4C: Dimensões da tampa. Raio ~ 7,8cm. Altura = 2cm.

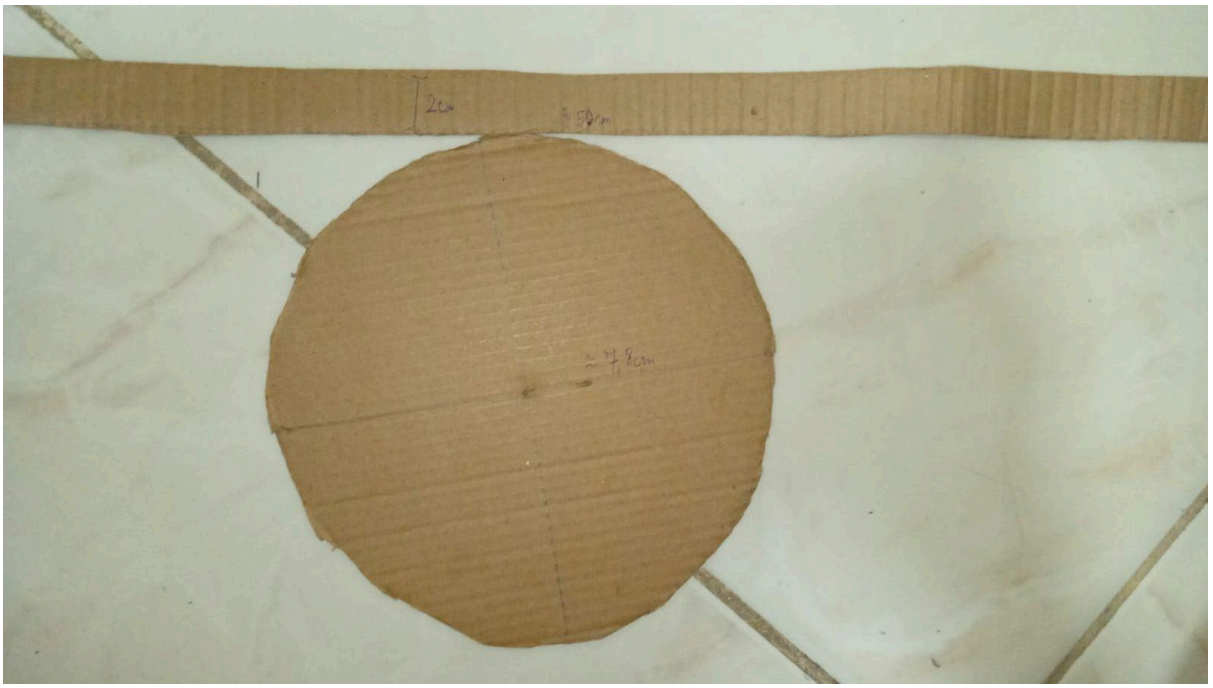


Figura 6C: Tampa do reservatório.

Figura 5C: Câmara responsável por pesar a quantidade de ração. Dimensões: Raio = 5 cm.

Figura 6C: Câmara e pote de ração acoplados no suporte. Rampa: conexão entre a câmara e o pote de ração.

Figura 6C (a): Câmara aberta, permitindo que a ração percorra a rampa até ao pote de ração.

Figura 6C (b): Câmara fechada

Figura 8C: Alimentador: conjunto formado pelo reservatório, câmara e pote de ração acoplados no suporte.

Figura 8C (a): Câmara fechada.

Figura 8C (b): Câmara aberta.

9. ANEXO D - TABELA DE CUSTOS

Segue abaixo a planilha com os custos dos componentes utilizados no projeto:

Tabela de Custos				
Componentes	Quantidade	Especificação	Site	Preço
Arduino UNO	1	UNO Rev3	Arduino	R$ 149,23
Célula de Carga + Módulo de carga	1	1kg	Mercado Livre	R$ 35,90
Conversor AC/DC	1	5V - 1A	Casa da Robótica	R$ 17,99
Módulo Bluetooth	1	HC-05	Eletrogate	R$ 42,90
Motor Servo	3	1,6kg.cm(6V)	Eletrogate	R$ 16,06
Sensor Utrassônico	1	2cm - 400cm	Robocore	R$ 8,50
Módulo RTC	1	DS3231	Eletrogate	R$ 36,90
Total	-	-	-	R$ 307,48