Comedouro eletrônico de dietas restritivas para animais com sobrepeso

Danie Auler
Universidade de Brasília
Faculdade Gama
Brasília, Brasil
danielauler7@gmail.com

Paulo Franklin Gomes
Universidade de Brasília
Faculdade Gama
Brasília, Brasil
pgomespereiradeoliveira@gmail.com

I. Justificativa

Em se tratando do século XXI, muitas pessoas que moram em apartamentos na zona urbana, possuem animais de estimação como gatos e cachorros. Muitos destes animais sofrem por ter sobrepeso, pois eles acabam se alimentando de maneira muito rápida, o que corrobora para o estresse desses animais, além do peso exacerbado[1].

A ideia de construir um comedouro eletrônico de dieta restritivas para animais com sobrepeso surgiu a partir de um problema vivenciado por um cachorro que mora perto da minha casa. Este possui um problema intestinal e, devido a isso, ele deve usufruir de uma dieta balanceada e cadenciada em horários previamente determinados.

Figura 1. Bichinho de estimação se alimentando rapidamente

Contudo, uma solução encontrada para contornar esse problema é projetar um sistema que é capaz, em horários previamente determinados, enviar uma foto da tijela do animal, via o aplicativo do telegram, que terá como função verificar o quanto de ração este comeu, bem como ter a funcionalidade de cancelar o bastecimento de ração na tijela do bichinho, caso esta esteja cheia no momento. Caso contrário, o sistema liberará aos poucos a quantidade de comida necessária para alimentar o animalzinho.

II. Objetivos

Criação de um comedouro eletrônico que na hora específica irá enviar para o usuário via telegram uma foto com a tigela de comida para que o mesmo saiba o quando da ração o animal comeu. Caso o usuário não cancele o acionamento da tigela, a mesma irá liberar aos poucos ou de uma vez (escolha do usuário) a quantidade de ração necessária nos horários especificados.

III. Requisitos

O comedouro será equipado com um raspberry pi zero conectado a internet e conectado a uma câmera apontada para a tigela do animal. Além disso, o comedouro contará com um sensor de presença ultrassônico para saber se o animal comeu a porção parcial. O rasp fará requisições http para o bot para avisar o usuário sobre as ações que aconteceram e estará com um server aberto para ouvir as requisições do bot quando o usuário quiser inserir um comando de intervenção ou ajuste nos horários de alimentação.

IV. Benefícios

O projeto tem como principal beneficiário o dono de animais com transtorno de ansiedade ou alguma doença que necessite de restrição alimentar com horários especificados. Em consequência disso, temos como beneficiário também pessoas que deixam o animal de estimação sozinho em casa e não querem deixar um pote muito grande com muita ração parada para que o animal consiga se alimentar ao longo do dia.

V. Servo motor

O servo motor é muito utilizado em projetos de automação industrial, onde de fato é necessário um controle de precisão maior em relação ao sistema que está sendo projetado. Outrora, imaginava-se que este só serviria para aplicações que envolvessem apenas projetos relacionados ao controle preciso da direção do torque, além da velocidade e da posição. Entretanto, este dispositivo vem sendo uma alternativa muito boa para substituir motores convencionais tais como, indução e atuadores pneumáticos e hidráulicos.

Todavia de fato esta é uma máquina, eletromecânica, que apresenta movimento proporcional a um comando, como dispositivo de malha fechada, ou seja: recebe um sinal de controle que verifica a posição atual para controlar o seu movimento indo para a posição desejada com velocidade monitorada externamente sob feedback de um dispositivo denominado taco ou sensor de efeito Hall ou encoder ou resolver, ou tachsin, dependendo do tipo de servomotor e aplicação.

Para o ponto de controle 4 foi aperfeiçoado o desenvolvido do código em C, que pode ser observado no apêndice A, com

o objetivo de controlar um micro servo motor da Tower Pro 9g SG90 com auxílio da Raspberry Pi 3.

Nesse código foi criado três funções, uma que faz o servo motor girar 90 graus e outra que faz girar -90 graus, além da função alimentar, que tem como objetivo liberar a ração que irá cair na tijela do animal.

Tal função alimentar é chamada na função principal, ou seja, a função main. Após ser chamada, a mesma tem como funcionalidade chamar a função girar motor, na qual ela fará o servo motor girar 90 graus de modo a liberar a ração durante um determinado tempo. Com o passar desse tempo, a função alimentar executar a função girar motor, por mais uma vez, porém no sentido de -90 graus com intuito de fechar a escotilha na qual cai a ração.

Pois posteriormente este será utilizado para abrir a escotilha pra cair a ração. Tal código movimenta o servomotor 5 vezes, usando o pino GPIO4, que é referenciado pela biblioteca wiringPi utilizando o pino 7.

Logo, abaixo é mostrado como foi feito a conexão entre os pinos do servo motor e da Raspberry Pi 3.

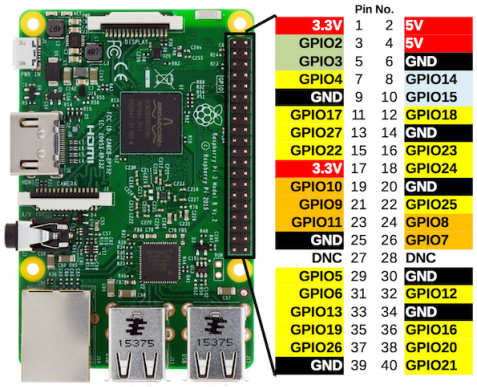

Figura 2. Pinagem da Raspberry Pi 3 com o servo motor

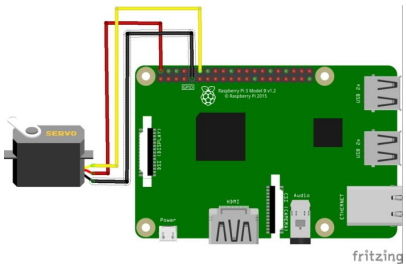

Figura 3. Conexão entre os pinos da Raspberry Pi 3 com o servo motor

VI. Bot do telegram

Com objetivo de conectar o usuário ao comedouro foi feito uma plataforma do telegram com sua API aberta e bem documentada, bem como um fluxograma demostrando como é feita essa conexão entre o usuário, o bot do telegram e o comedouro (figura 3). Para testarmos essa conexão foi realizado um teste com foco de enviar um comando via o bot do telegram, no qual este acende um led com uso da raspberry, além de verificar o recebimento desse comando. Tal código pode ser visualizado no apêndice B em anexo.

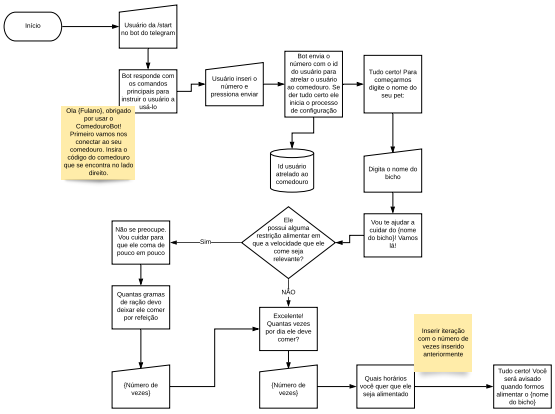

Figura 4. Conexão entre os pinos da Raspberry Pi 3 com o servo motor

VII. Servidor

Nesta etapa foi implementado um servidor em linguagem C. Porém neste código consta uma biblioteca denominada tgbot, que está implementada em C++. Esta biblioteca tem a incubencia de fazer a conexão com o telegram. Apesar dessa biblioteca ser na linguagem C++, é possível executá-la dentro do código em C. Por isso, não terá prolemas na hora da execução do projeto com um todo.

VIII. Protótipo da estrutura do comedouro etrônico

Pensando em como testar o projeto de fato com um menor investimento possível foi criado um protótipo da estrutura do comedouro eletrônico. Tal estratégia está relacionada com a a sigla MVP, que é a sigla Minimum Viable Product (Produto mínimo viável).

Esta estrutura representa uma versão mínima do produto, na qual pussui a finalidade de cumprir com os requisitos necessários para a validação do projeto.

A partir da confecção dessa estrutura foi possível testar realmente a teoria na prática. Nela foi observado que a mesma cumpria com os requisitos preestabelicos para o projeto. Uma delas foi a liberação de ração com a utilização do servo motor. Além disso, foi testado o servidor para verificar se o mesmo enviava e recebia as mensagens via o bot do telegram.

Referências

[1] dicas para auxiliar o seu cachorro comer-mais devagar. , 2018. Disponível em: <https://www.portaldodog.com.br/cachorros/adultos-cachorros/alimentacao-adulto/dicas-para-auxiliar-o-seu-cachorro-comer-mais-devagar>. Acesso em: 29 set. 2019.

[2] SERVO Motor SG90 Datasheet. [S. l.], 2014. Disponível em: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf. Acesso em: 6 out. 2019.

[3] OS BOTS do Telegram estão mais inteligentes. [S. l.], 2016. Disponível em: https://tecnoblog.net/194163/telegram-bots-2/. Acesso em: 6 out. 2019.

Figura 5. Protótipo da estrutura do comedouro etrônico

[4] CACHORRO comendo muito rápido? Como fazer ele comer mais devagar. [S. l.], 13 fev. 2017. Disponível em: https://tudosobrecachorros.com.br/cachorro-comendo-muito-rapido-como-fazer-ele-comer-mais-devagar/. Acesso em: 6 out. 2019.

[5] OS 5 Melhores Comedouros Para Cães Do Mercado: Qual é o melhor comedouro para cães?. [S. l.], 2018. Disponível em: https://amoraospets.com/melhor-comedouro-para-caes/. Acesso em: 6 out. 2019.

Apêndice A: Código em C para controlar o servo motor com intuito de abrir a escotilha para liberar ração ou não:

```
#include <wiringPi.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

#define SAIDA 7

void girar_motor(int pin, int degree, int N)
{
int t1 = (50*degree+4)/9+1500;
int t2 = 20000-t1;
int i;
for(i=0; i<N; i++)
{
digitalWrite(pin, HIGH);
usleep(t1);
digitalWrite(pin, LOW);
usleep(t2);
}
}

void alimentar(){
girar_motor(SAIDA, 90, 25);
usleep(500);
girar_motor(SAIDA, -90, 25);
usleep(500);

}

int main(void)
{
wiringPiSetup();
pinMode(SAIDA, OUTPUT);
alimentar();
return 0;
}
```

Apêndice B: Código em payton para controlar a Raspberry Pi 3 com o bot do telegram:

```
import time, datetime
import RPi.GPIO as GPIO
import telepot
from telepot.loop import MessageLoop

led = 26
now = datetime.datetime.now()
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
 #LED
GPIO.setup(led, GPIO.OUT)
GPIO.output(led, 0) #Off initially

def action(msg):
    chat_id = msg['chat']['id']
    command = msg['text']

    print 'Received: %s' % command
    if 'on' in command:
        message = "Turned on "
        if 'led' in command:
          message = message + "led"
          GPIO.output(led, 1)
          telegram_bot.sendMessage
          (chat_id, message)

    if 'off' in command:
        message = "Turned off "
        if 'led' in command:
            message = message + "led "
            GPIO.output(led, 0)
            telegram_bot.sendMessage
            (chat_id, message)

telegram_bot = telepot.Bot('Meu token')
print (telegram_bot.getMe())

MessageLoop(telegram_bot, action)
.run_as_thread()
```

```
print 'Up and Running....'

while 1:
    time.sleep(10)
```

Apêndice C: Código do servidor feito em linguagem C:

```
#include <stdio.h>
#include <tgbot/tgbot.h>

int main() {
    TgBot::Bot bot("931015860:
        AAHG6qZTMlopgG29lXC6-_rAPSmNrKiYXm4");
    bot.getEvents().onCommand
    ("start", [&bot](TgBot::Message::Ptr message) {
        bot.getApi().sendMessage
        (message->chat->id, "Hi!");
    });
    bot.getEvents().onAnyMessage([&bot]
        (TgBot::Message::Ptr message) {
        printf("User wrote %s\n",
            message->text.c_str());
        if (StringTools::startsWith
            (message->text, "/start")) {
            return;
        }
        bot.getApi().sendMessage
            (message->chat->id, "Your message is:
                " + message->text);
    });
    try {
        printf("Bot username: %s\n", bot.getApi().getMe()->username.c_str());
        TgBot::TgLongPoll longPoll(bot);
        while (true) {
            printf("Long poll started\n");
            longPoll.start();
        }
    } catch (TgBot::TgException& e) {
        printf("error: %s\n", e.what());
    }
    return 0;
}
```