

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE UM ALIMENTADOR AUTOMÁTICO DE
ANIMAIS COM RASPBERRY PI

JONATHAN ANDRÉ SCHWEDER

BLUMENAU
2015

2015/2-13

JONATHAN ANDRÉ SCHWEDER

**PROTÓTIPO DE UM ALIMENTADOR AUTOMÁTICO PARA
ANIMAIS COM RASPBERRY PI**

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Miguel Alexandre Wisintainer , Mestre - Orientador

**BLUMENAU
2015**

2015/2-13

PROTÓTIPO DE UM ALIMENTADOR AUTOMÁTICO PARA ANIMAIS USANDO RASPBERRY PI

Por

JONATHAN ANDRÉ SCHWEDER

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:

Prof. Miguel Alexandre Wisintainer, Mestre – Orientador, FURB

Membro:

Prof. Gabriele Jennrich Bambineti, Especialista – FURB

Membro:

Prof. Francisco Adell Péricas, Mestre – FURB

Blumenau, 11 de dezembro de 2015

Dedico este trabalho a minha família, pelo amor, apoio e compreensão.

AGRADECIMENTOS

A Deus...

À minha família...

Aos meus amigos...

Ao meu orientador...

A minha namorada...

Aos meus sogros...

A persistência é o menor caminho do êxito.

Charles Chaplin

RESUMO

Este trabalho apresenta a implementação de um protótipo de um alimentador automático para animais de estimação utilizando o mini PC Raspberry Pi modelo B+, a linguagem PHP para desenvolver a interface gráfica, utilizando o *framework* Laravel, para acesso as funcionalidades do protótipo, a linguagem Python para desenvolver o software embarcado, utilizando as bibliotecas já existentes de *GPIO* para o controle dos periféricos utilizados. Utilizando um motor de passo para girar um recipiente onde se encontra a ração e um sensor magnético para determinar o momento de parada do motor. Toda a estrutura do protótipo foi construída usando material plástico para possibilitar o uso tanto com alimento úmido como água. Através deste protótipo é possível determinar os horários para liberação do alimento e ainda é possível através de uma câmera visualizar em tempo real o local onde se encontra o alimentador.

Palavras-chave: Raspberry Pi. Alimentador automático. Interface WEB.

ABSTRACT

This work presents the implementation of a prototype of an automatic feeder for pets using the mini PC Raspberry Pi model B +, the PHP language for developing GUI, using the Laravel framework to access the features of the prototype, the Python language for developing the embedded software using the existing libraries GPIO used to control the peripherals. Using a stepper motor to rotate a container where the food is located and a magnetic sensor to determine when the engine stopped. The entire structure of the prototype was constructed using plastic material to enable the use of both wet food such as water. Through this prototype can determine the times for food release and it is possible through a camera view in real time the location of the feeder.

Keywords: Raspberry Pi. Automatic feeder. WEB interface.

LISTA DE FIGURAS

Figura 1– Alimentador Magic Feeder	16
Figura 2– Alimentador Feed & Go.....	16
Figura 3– Compartimentos para armazenamentos de alimento.....	17
Figura 4– Estrutura do protótipo anterior	17
Figura 5– Versão final do protótipo anterior	18
Figura 6– Raspberry Pi modelo B+	19
Figura 7– Alimentador Feed&Go.....	20
Figura 8– Visão superiora da estrutura.....	22
Figura 9– Visão lateral da estrutura.....	22
Figura 10– Recipiente maior	23
Figura 11– Recipiente menor	23
Figura 12– Detalhe do sensor magnético no recipiente maior	24
Figura 13– Diagrama da programação do alimentador	25
Figura 14– Diagrama de sequência	26
Figura 15– Diagrama de atividade do processo.....	27
Figura 16– Mapa de pinos disponíveis no Raspberry Pi modelo B+	33
Figura 17– Página de login.....	35
Figura 18– Página principal.....	35
Figura 19– Página de cadastro do horário de liberação do alimento.....	35
Figura 20– Visualização da câmera.....	36
Figura 21– Embalagem utilizada na construção do protótipo	37

LISTA DE QUADROS

Quadro 1– Arquivo de configuração do motion.....	28
Quadro 2– Embalagem utilizada na construção do protótipo	30
Quadro 3– Programa controlador do motor e sensor magnético	31
Quadro 4– Modelos de Raspberry Pi.....	38

SUMÁRIO

1 INTRODUÇÃO.....	12
1.1 OBJETIVOS.....	12
1.2 ESTRUTURA.....	13
2 FUNDAMENTAÇÃO TEÓRICA	14
2.1 NUTRIÇÃO DE CÃES E GATOS	14
2.2 ALIMENTADORES DISPONÍVEIS NO MERCADO.....	15
2.2.1 Magic Feeder.....	15
2.2.2 Feed & Go.....	16
2.3 PROTÓTIPO ANTERIOR.....	17
2.4 RASPBERRY PI.....	19
2.5 TRABALHOS CORRELATOS.....	20
3 DESENVOLVIMENTO	21
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	21
3.2 ESPECIFICAÇÃO	21
3.2.1 Especificação da estrutura.....	22
3.2.2 Especificação do software.....	24
3.3 IMPLEMENTAÇÃO	27
3.3.1 Técnicas e ferramentas utilizadas.....	27
3.3.2 Operacionalidade da implementação	34
3.4 RESULTADOS E DISCUSSÕES.....	36
4 CONCLUSÕES	40
4.1 EXTENSÕES	40

1 INTRODUÇÃO

É fato que o convívio com animais domésticos está cada vez mais frequente no país e no mundo, seja como companhia, auxílio a pessoas com alguma deficiência ou terapêutico como no caso de tratamento de crianças. Animais como cães e gatos estão a gerações na companhia do homem. Segundo a Associação Brasileira da Indústria de Produtos Para Animais de Estimação (ABINPET, 2012) a estimativa atual da quantidade de animais é de 37,1 milhões de cães e 21,3 milhões de gatos, um crescimento considerável considerando por exemplo os dados de Orsini (2004) onde as estimativas eram de 27 milhões de cães e 11 milhões de gatos. Esse crescimento que segundo estimativas da ABINPET (2012) giram em torno de 5% ao ano, gerando um mercado promissor para investimentos em produtos e serviços especializados.

Estes que são vendidos em lojas de produtos para animais para consumidores que estão cada vez mais procurando produtos diferenciados, exclusivos e que se adaptem ao seu dia a dia. Muitas vezes estes consumidores, que por motivos de força maior, responsabilidades com o meio profissional, familiar e o ritmo frenético de suas vidas são obrigados a ficarem longe de seus animais de estimação, gerando um sentimento muitas vezes de abandono e preocupação.

Na tentativa de sanar estas necessidades o mercado disponibiliza alguns produtos com foco em tentar diminuir a distância física entre animal e seu dono. Alguns produtos como alimentadores automáticos de animais, possibilitam que mesmo pessoas com uma rotina corrida possam alimentar seus animais diversas vezes ao dia. Alguns produtos, porém possibilitam que o dono possa até visualizar através de uma câmera de vídeo seu animal passeando pela casa, tendo assim uma maior garantia de que seu amigo está bem e que ele pode continuar sua rotina com um pouco mais de tranquilidade.

Neste contexto este trabalho tem o objetivo da construção de um protótipo de alimentador automático para animais de estimação, com base no protótipo criado por Ochaowski (2007), utilizando o mini *Personal Computer* (PC) de baixo custo Raspberry PI para a montagem do hardware necessário e a linguagem Python para a programação do software de controle através de uma interface web.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um protótipo de um alimentador automático para animais de estimação.

Os objetivos específicos do trabalho são:

- a) disponibilizar a estrutura mecânica do alimentador;
- b) disponibilizar um hardware dotado de motores e sensores;
- c) desenvolver o software responsável pelo controle do alimentador;
- d) desenvolver uma interface web para acesso as principais funcionalidades do software;
- e) desenvolver a função de transmissão da imagem da câmera.

1.2 ESTRUTURA

Este trabalho está organizado em quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusões. O capítulo dois apresenta a fundação teórica para o desenvolvimento do projeto. O capítulo três aborda o desenvolvimento presente do projeto, com especificações e diagramas detalhados. O capítulo quatro apresenta as conclusões e sugestões para possíveis extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta sessão é apresentado a fundação teórica do trabalho. Inicialmente são apresentados alguns dados sobre nutrição de cães e gatos. Logo após são apresentados alguns alimentadores disponíveis no mercado. Por fim são apresentados os trabalhos correlatos.

2.1 NUTRIÇÃO DE CÃES E GATOS

Há séculos o homem trouxe para si a companhia de animais, seja para a finalidade de trabalho ou para alimentação, fazendo com que fosse preciso se preocupar com as necessidades básicas desses animais para que os mesmos se mantivessem vivos e tivessem suas finalidades cumpridas. O campo de nutrição animal tem o objetivo de trazer aos animais os nutrientes necessários para suprir as necessidades de seus organismos, ou seja, fornecer os “combustíveis” adequados para que todas as funções de seus corpos estejam funcionando corretamente, somando-se ao fato de que cada organismo possui um metabolismo diferente, necessidades proteicas e nutricionais das mais variadas proporções, aumenta consideravelmente a importância deste área e seus estudos, pois se deve haver uma escolha adequada para cada indivíduo (ALBANO, 2007).

Os estudos, resultados e necessidades desta área fez com que a qualidade da indústria de alimentação animal no preparo dos alimentos balanceados fosse tão exigente quanto à fabricação de alimentos para o consumo humano. Todo o processo é estudado para oferecer um produto que satisfaça plenamente o exigente mercado. Neste processo quando a matéria-prima chega à fábrica, técnicos examinam a qualidade dos cereais, carnes e peixes e começam a separar os produtos de acordo com a análise de suas características nutricionais, bacteriológicas e de digestão predominantes (ALBANO, 2007).

Tudo deve passar por um controle rigoroso que vai determinar se as matérias-primas estão de acordo com as exigências para entrar na composição dos produtos. As linhas de produção destas fábricas são totalmente automatizadas e asseguram a precisão na dosagem dos ingredientes, eliminado o risco de erro humano e o contato físico com os ingredientes, frisando pela qualidade. Além do controle sanitário oficial, as indústrias mantêm seu próprio sistema de análise em diferentes fases do processo de produção (PETBR, 2015).

Contudo não só deve-se ter uma preocupação com a procedência da ração e verificar se o tipo dela é o ideal para a raça do animal de estimação, mas também se deve haver uma dosagem correta para que o animal tenha suas necessidades atendidas sem exageros. A comida em abundância, além de provocar obesidade, também pode causar outros problemas de saúde. Alimentos como rações que ficam muito tempo expostos ao ar livre acabam

absorvendo muita umidade, fazendo com que adquiram um gosto azedo, além de ser um possível atrativo a outros animais, aumentando as chances de transmissão de doenças (ALBANO, 2007).

Além de cada animal possuir suas diferenças metabólicas, animais como cães e gatos possuem diferentes comportamentos na hora de se alimentar. Como por exemplo, segundo Albano (2007), o ideal é que cães recebam a ração de maneira fracionada e em horários fixos, pois em geral, eles tendem a ter sonolência após cada refeição.

Um estudo mais detalhado do comportamento de algumas raças de cães e gatos, integrando este conhecimento ao protótipo deste trabalho, pode criar uma inteligência capaz de prevenir doenças nesses animais, através de dosagens corretas e recomendações dadas ao dono do animal.

2.2 ALIMENTADORES DISPONÍVEIS NO MERCADO

Os alimentadores automáticos para animais de estimação não são nenhuma novidade no mercado, atualmente existem diversos modelos, porém, infelizmente a falta de recursos é algo frequente na maioria dos modelos encontrados.

A maioria dos alimentadores automáticos se limita a controlar, por exemplo, um recipiente de água e ração, esvaziando-os ao final de um ciclo de tempo dado por algum mecanismo temporizador. Porém estes mecanismos não possuem uma grande precisão de horários, não sendo possível determinar a hora exata que em se deseja fazer a liberação da ração, permitem apenas determinar qual o tempo para a liberação, também não possuem alguma forma de interação com o animal quando o dono está longe.

2.2.1 Magic Feeder

O Magic Feeder, ilustrado na figura 1, da empresa Prodac (2010), é um alimentador programável para animais aquáticos, com capacidade para armazenar porções de até cento e cinquenta gramas de alimento. Sua estrutura dispõe de um recipiente para comida e uma bomba de ar para ventilar e lançar o alimento para ser consumido.

Figura 1 - Alimentador Magic Feeder



Fonte: Prodac (2010).

O funcionamento do equipamento ocorre da seguinte forma:

Deve-se separar o depósito da comida do alimentador, retirar a tampa com a marca “Up” e encher o depósito com alimento, voltar a colocar a tampa; com a marca “Up” perto do local de saída do alimento. Virar o depósito ao contrário, controlar que a quantidade de comida não exceda o nível “Max” nem que seja inferior do nível “Min”. Deslocar a tampa interna dosadora para a quantidade de alimento que deseja liberar. Regular o temporizador girando o botão central no sentido dos ponteiros do relógio por exemplo, se for 4 da tarde, girar o botão até que o número dezesseis fique em linha com o botão “Manual Start”. Inserir os alfinetes no horário de alimentação selecionado, de 1 a 6 vezes por dia. Não se esquecer de colocar as pilhas (PETLOVE, 2015).

2.2.2 Feed & Go

O Feed & Go da empresa FeedAndGo (2015), ilustrado na Figura 2, é outro dispositivo de alimentação automática para animais de estimação que permite ao dono alimentar e além disso observar o animal.

Figura 2 – Alimentador Feed & Go



Fonte: FeedAndGo (2015).

O dispositivo possui uma câmera para que o dono possa ver o seu animal enquanto estiver fora de casa. O mesmo ainda possui um sistema de gravação de voz para que o dono possa gravar sua voz indicando ao animal a hora da refeição. Por fim para facilidade o dispositivo possui conectividade sem fio para que seja possível colocá-lo em qualquer local da casa. Para a alimentação, o dispositivo possui seis compartimentos para a colocação de

alimento e água, é possível controlar o dispositivo por qualquer computador ou dispositivo móvel através de aplicativos (FEEDANDGO, 2015).

No caso deste alimentador é possível através de qualquer aparelho celular com sistemas operacionais *Android* e *IOS* programar o horário que será liberado a comida para o animal. A comida é armazenada no equipamento em seis compartimentos, conforme ilustrado na Figura 3.

Figura 3 – Compartimentos para armazenamento de alimento

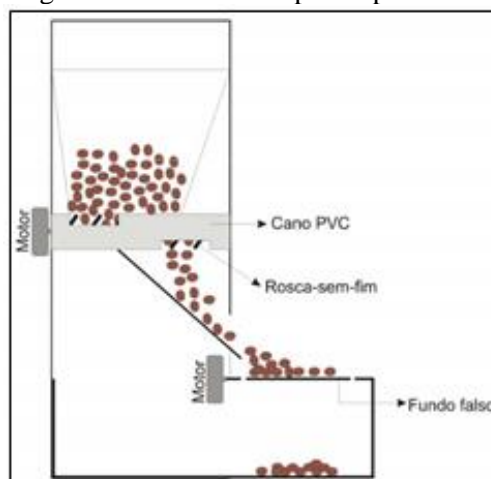


Fonte: FeedAndGo (2015).

2.3 PROTÓTIPO ANTERIOR

O trabalho apresentado por Ochaowski (2007) fez a implementação de um protótipo de um alimentador automático para animais de estimação utilizando o micro controlador PIC16F877, utilizando a linguagem de programação C para desenvolver o *software* embarcado, o circuito integrado RTC para manter atualizada a hora, um visor LCD e um teclado telefônico para a interação do usuário e dois motores de passo, um para acionar a rosca sem fim e outro para girar em 90 graus o fundo falso do recipiente de comida, a Figura 4 mostra a estrutura do alimentador.

Figura 4 – Estrutura do protótipo anterior



Fonte: Ochaowski (2007).

Através do protótipo foi possível, determinar a quantidade de refeições, agendar horários para refeição, determinar a quantidade de ração a ser liberada. A Figura X mostra a foto da versão final do

Em relação ao trabalho realizado foram dadas algumas observações com relação aos componentes utilizados. Segundo Ochaowski (2007) num primeiro momento tentou-se utilizar a estrutura de um alimentador por gravidade disponível no mercado, contudo devido à falta de espaço para acoplar a rosca sem fim utilizada para transportar o alimento até o recipiente de para consumo, isto não foi possível. A Figura 5 demonstra a versão final do alimentador feito por Ochaowski (2007).

Figura 5 – Versão final do protótipo anterior



Fonte: Ochaowski (2007).

Outros fatores como o fato da maioria dos alimentadores serem de material plástico, e este não ser muito facilmente moldável, dificultaram a reestruturação do protótipo. Por estes motivos decidiu-se criar uma nova estrutura para o protótipo, que foi desenvolvida visando atender a praticamente todas as necessidades e gerar um bom desempenho dos mecanismos acoplados (OCHAKOWSKI, 2007).

2.4 RASPBERRY PI

Sistemas embarcados consistem em sistemas microprocessador no qual o poder de processamento é voltado à realização de uma atividade específica. Comparados a um computador normal, os sistemas embarcados são construídos para realizarem tarefas mais rapidamente, ocupando menor espaço com um menor consumo de energia, a um custo reduzido. Existem dispositivos com sistemas embarcados com diversos propósitos, alguns mais dedicados e outros mais abrangentes (BARROS, 2013, pg. 7).

O projeto será desenvolvido utilizando um Raspberry PI modelo B+, ilustrado na Figura 6, um microcomputador desenvolvido na Grã-Bretanha de dimensões reduzidas, idealizado sob o paradigma *System On a Chip* (SO) (XUN et al., 2001).

Figura 6 – Raspberry Pi modelo B+



Fonte: Adafruit (2015).

Os primeiros modelos do Raspberry PI foram anunciados em 2011 e lançados em 29 de fevereiro de 2012, idealizados pelo inglês Pete Lomas para serem os computadores mais baratos do mundo, com preços a partir de 25 dólares (Modelo A) e 35 dólares (Modelo B), tendo finalidades educativas, foram criados para estimular o ensino da ciência da computação e áreas relacionadas (RASPBERRY PI FOUNDATION, 2015).

O modelo B do Raspberry PI originalmente possuía um microprocessador *Broadcom BCM2835 ARM11*, comumente encontrado em aparelhos móveis, com desempenho de 700 MHz, uma *GPU* com 4 núcleos capazes de realizar a reprodução de vídeo em qualidade *Full HD*, 512 MB de memória *RAM*, 2 portas *USB 2.0*, saída de vídeo *HDMI*, saída *RCA*, saída de áudio, entrada para conexão *Ethernet 10/100 MB*, fonte de alimentação de 5 V, conector para cartão *SD* e 26 pinos disponíveis para realizar operações de entrada e saída (RASPBERRY PI FOUNDATION, 2015).

O modelo utilizado neste trabalho, o *Raspberry PI* modelo *B+* foi lançado em 2014 como substituto do antigo modelo *B*. Este novo modelo trouxe diversas melhorias em

comparação ao modelo anterior, possuindo agora 40 pinos, 4 entradas USB 2.0, redução do consumo de energia ficando entre 0,5 W e 1 W (RASPBERRY PI FOUNDATION, 2015).

Em relação ao sistema operacional, existem diversas opções disponíveis no site oficial, a maioria destes, são sistemas baseados no *GNU/Linux*, um sistema operacional aberto. Por possuírem a mesma base que os sistemas para microcomputadores tradicionais, existem o suporte para linguagens como *C* e *Python*.

2.5 TRABALHOS CORRELATOS

Barros (2013, p. 8) apresenta um projeto onde é mostrado a utilização do Raspberry PI utilizando as funções relacionadas à sua câmera e a seus pinos de entrada e saída para a parte de servidor da aplicação, e o acesso a este servidor pela parte cliente é feito através de um software desenvolvido para a plataforma móvel Android feito na linguagem de programação Java. As operações de entrada e saída feitas, assim como o acesso remoto ao aparelho serão semelhantes às que serão usadas neste trabalho.

O alimentador automático para animais desenvolvido por FeedAndGo (2015), demonstrado na Figura 7, possui suporte a câmera para que o dono possa visualizar o ambiente e seu animal, o sistema de gravação de voz para que seja tocado um áudio toda vez que for a hora do animal comer indicando a ele isto, a possibilidade de acesso via dispositivo móvel ou qualquer computador com acesso à internet, são essas e outras funcionalidades que servirão de inspiração para a criação do protótipo.

Figura 7 – Alimentador Feed&Go



Fonte: Fonte: FeedAndGo (2015).

3 DESENVOLVIMENTO

Este capítulo está dividido da seguinte forma:

- a) definição dos requisitos: detalha os requisitos funcionais e não funcionais do software e do hardware;
- b) especificação: nesta seção é apresentada a especificação da estrutura do alimentador, do hardware e do software;
- c) implementação: esta seção aborda as técnicas e ferramentas utilizadas no desenvolvimento deste protótipo, assim como apresenta trechos do código fonte desenvolvimento. Também é demonstrado a operacionalidade da implementação através do estudo de casos;
- d) resultados e discussões: nesta última seção são apresentados os resultados obtidos e conclusões feitas.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Abaixo seguem os requisitos funcionais (RF) e não funcionais (RNF) do protótipo:

Os RF e RNF do hardware são:

- a) utilizar a placa de circuito integrado Raspberry Pi modelo B+ (RNF);
- b) utilizar um sensor magnético para parar a rotação do reservatório de comida (RF);
- c) utilizar um motor de passo para acionar a rotação do reservatório de comida (RF);
- d) utilizar uma câmera para capturar vídeo do local aonde se encontra o protótipo (RF).

Os RF e RNF do software são:

- a) possibilitar a marcação do horário de liberação da comida (RF);
- b) possibilitar visualizar a imagem da câmera acoplada ao protótipo (RF);
- c) a software de interface deve ser implementada utilizando a linguagem *PHP* (RNF);
- d) a software de integração com o hardware deve ser implementado utilizando a linguagem de programação *Python* (RNF).

3.2 ESPECIFICAÇÃO

O usuário deve acessar a página *WEB* e cadastrar-se. Após o cadastro ele pode acessar com seu usuário e senha e cadastrar os horários que deseja que a comida seja liberada. Estes horários podem ser visualizados na página principal. Ao cadastrar os horários o usuário deve ter consciência do número de liberações até que o repositório de comida esteja vazio.

Ao ser acionado para liberar comida, o motor de passo será ligado e fará o recipiente onde se encontra o alimento girar, até que o sensor magnético seja acionado.

3.2.1 Especificação da estrutura

A Figura 8 apresenta uma visão superior do esquema do projeto, já a Figura 9 apresenta a visão lateral. Nesta estrutura estão presentes todos os componentes necessários para o funcionamento das funcionalidades do protótipo.

Figura 8 – Visão superior da estrutura

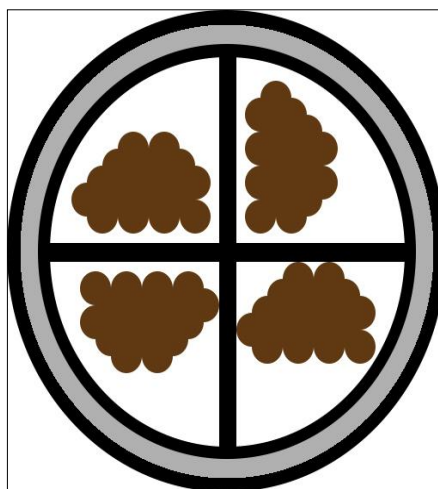
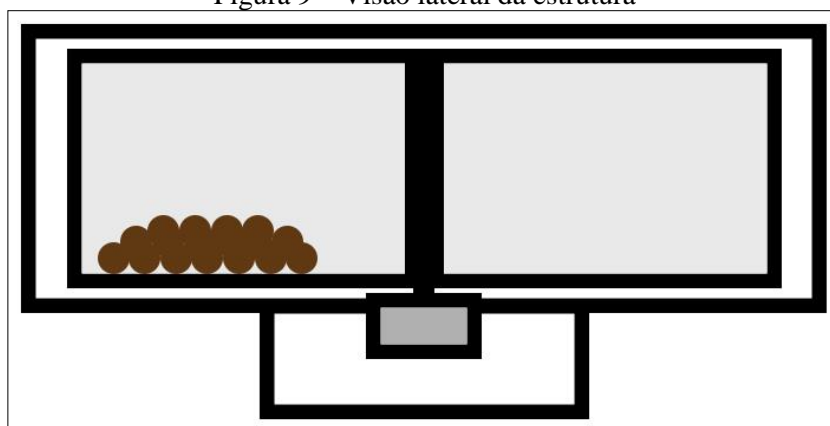


Figura 9 – Visão lateral da estrutura



O material utilizado na construção da estrutura do alimentador foi plástico. A escolha deste material se deu pela sua impermeabilidade, pois o alimento muitas vezes pode estar úmido, o que faz com que materiais como papelão absorvam esta água e acabem se desfazendo.

A estrutura está dividida em duas partes, um recipiente maior, que serve como base para o recipiente menor, e também possui uma tampa para proteger toda a estrutura, conforme pode ser visto na Figura 10. Para a construção desse foi utilizado uma embalagem para bolo, por já possuir o formato necessário, facilitando assim a confecção do mesmo.

Figura 10 – Recipiente maior



O recipiente interno, mostrado na Figura 11, também foi feito com base em uma embalagem de bolo, porém menor que o recipiente maior, para que seja possível colocar dentro deste. Este recipiente possui duas divisórias, como pode ser visto na Figura 10, que o dividem em quatro partes, cada uma destas partes pode armazenar o alimento do animal. Além disto também pode ser visto na Figura 12 o detalhe dos ímãs acoplados a estrutura, estes têm a funcionalidade de acionar o sensor magnético acoplado ao recipiente maior. A câmera não foi acoplada a estrutura.

Figura 11 – Recipiente menor



Figura 12 – Detalhe do sensor magnético no recipiente maior



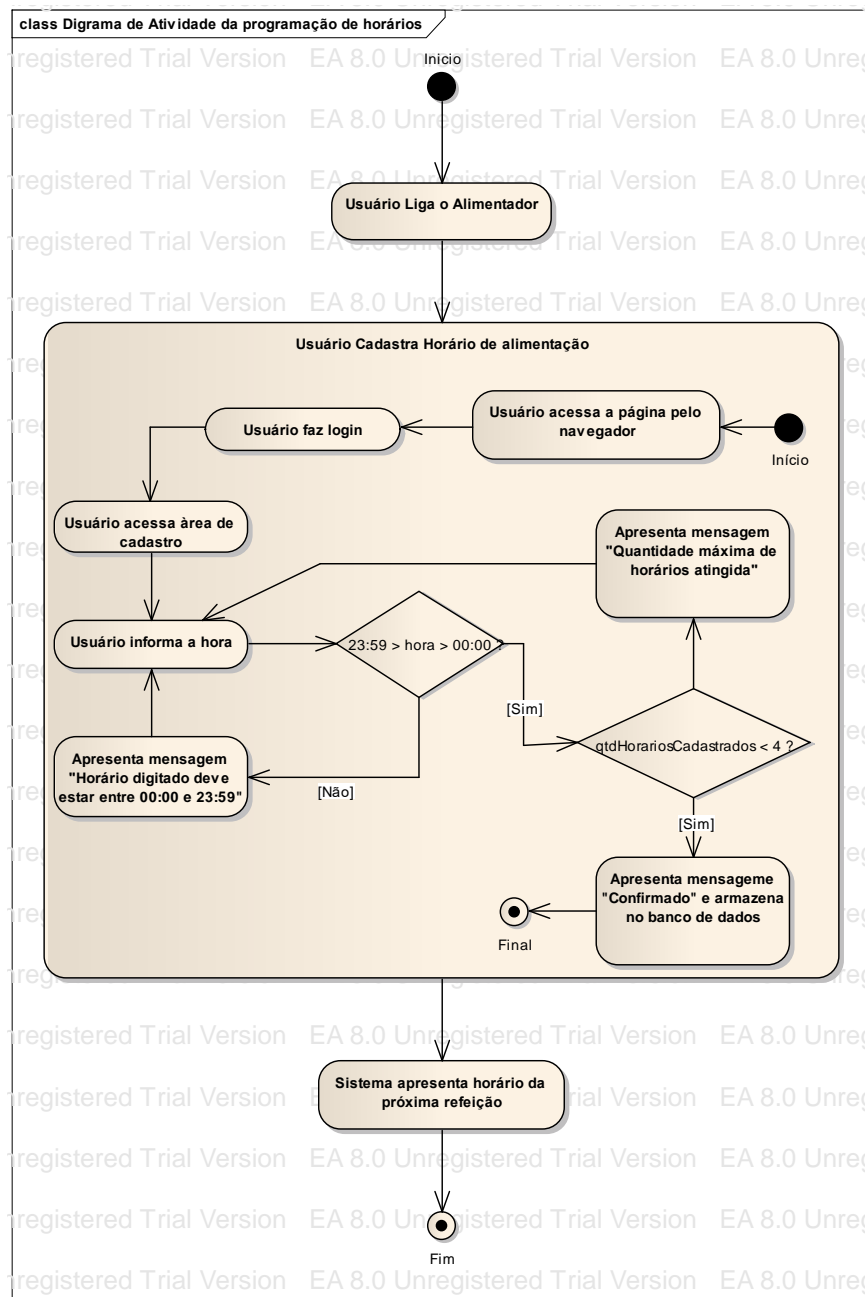
Na próxima seção será detalhado o software para controlar a estrutura do alimentador.

3.2.2 Especificação do software

As especificações do software são apresentadas na Figura 13, Figura 14 e Figura 15, através dos diagramas de atividade e sequência da UML. A ferramenta utilizada para a especificação foi o Enterprise Architect.

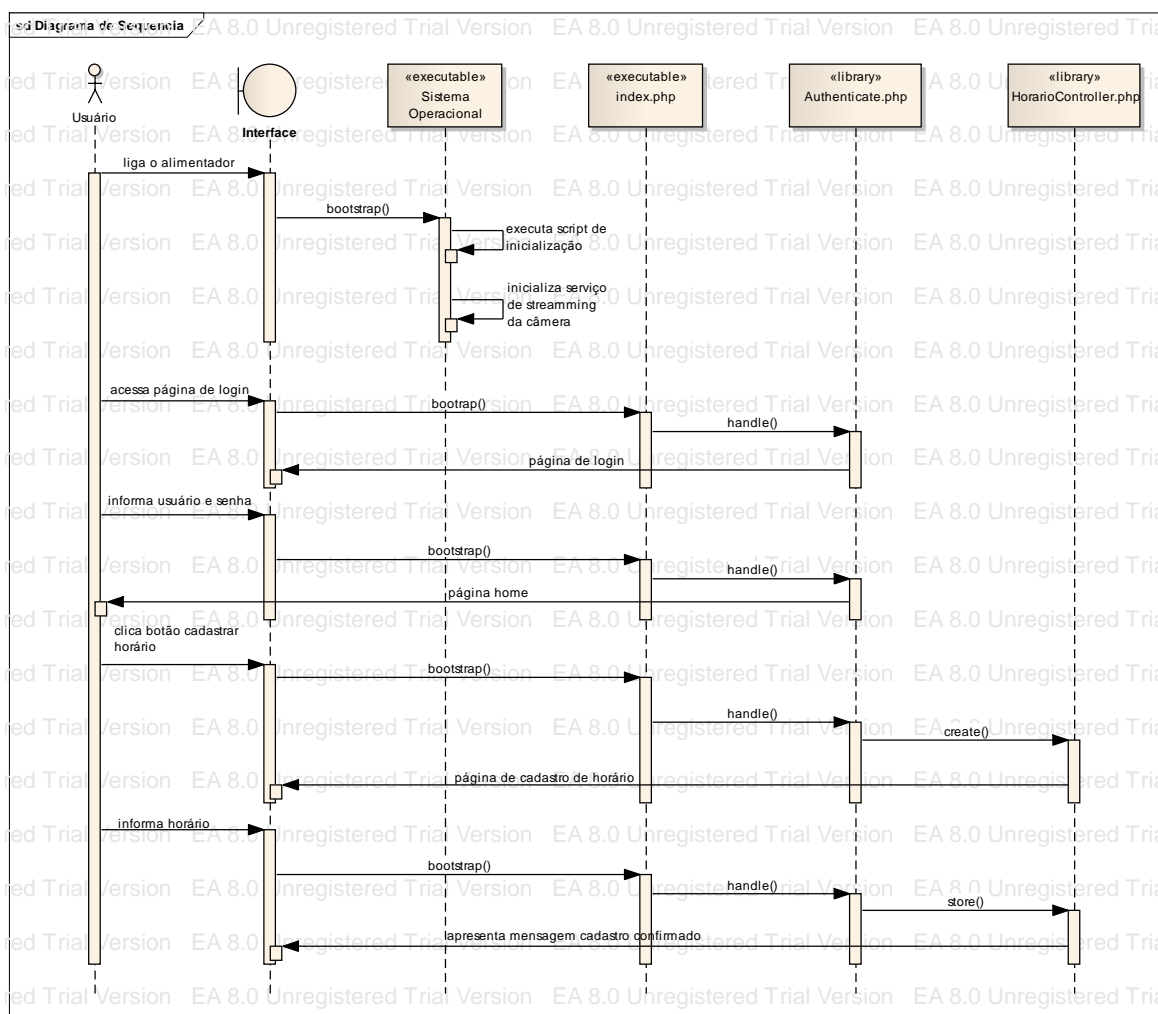
Na Figura 13 é apresentado a interação do usuário com o sistema. Este diagrama representa o momento da programação ou agendamento dos horários, sendo uma representação em nível de usuário.

Figura 13 – Diagrama da programação do alimentador



Já na Figura 14 é apresentado o diagrama de sequência que também representa a interação do usuário com no momento da programação, porém este faz uma representação em nível de software, onde são apresentadas as principais funções e as bibliotecas utilizadas.

Figura 14 – Diagrama de sequência



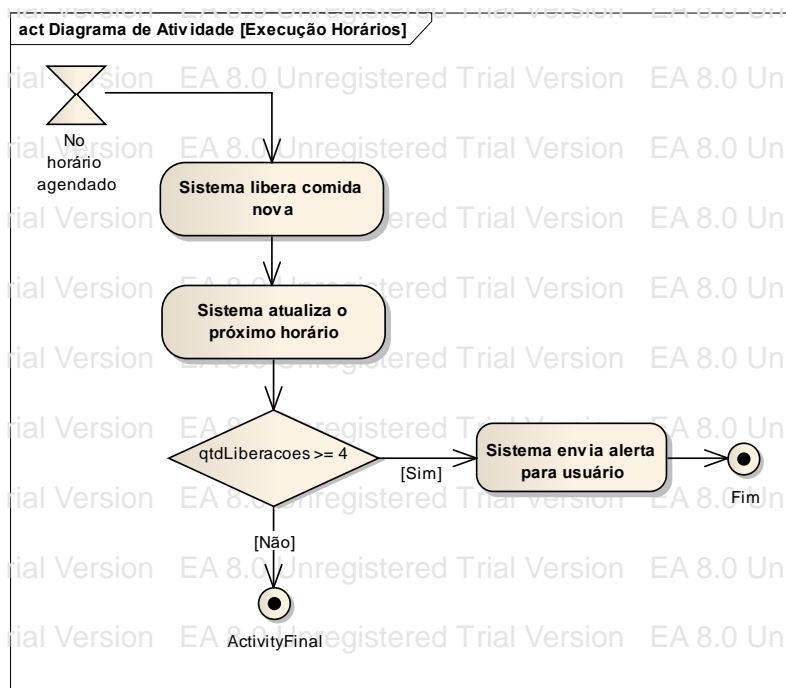
O Sistema Operacional tem a função de representar o sistema operacional presente no *Raspberry Pi*, nesse caso o *Raspbian*. Quando o usuário liga o aparelho imediatamente é executado o programa de inicialização do sistema, onde entre tantas funções executadas também é inicializado o serviço de streaming de vídeo da câmera.

Já o arquivo `index.php` tem a função de ser o ponto de entrada do usuário, sempre que é acessado alguma URL do projeto, este arquivo é que a recebe inicialmente e inicializa toda a estrutura para responder ao usuário. O arquivo `Authenticate.php` realiza as operações de segurança, caso o usuário não esteja logado e tente acessar uma rota protegida, ele redireciona o usuário para a página de login. Por fim o arquivo `HorarioController.php` é quem faz as operações de cadastros dos horários.

Na Figura 15 são apresentadas as ações tomadas pelo sistema após a programação dos horários. Neste diagrama foi utilizado uma ação temporal para demonstrar que o processo inicia apenas no horário agendado. Neste momento o sistema faz a execução do programa controlador do motor e sensores. Uma vez executado o programa é novamente adicionado um

novo agendamento para o dia seguinte no mesmo horário. Por fim, caso sejam feitas quatro liberações o sistema envia um alerta ao usuário, uma vez que a estrutura do protótipo possui apenas quatro baias e após essa quantidade de liberações o mesmo se encontra vazio.

Figura 15 – Diagrama de atividade do processo



3.3 IMPLEMENTAÇÃO

A seguir são apresentadas as técnicas e ferramentas utilizadas no desenvolvimento deste protótipo, apresentando trechos do código fonte desenvolvido. Também é demonstrada a operacionalidade da implementação, através de um estudo de caso.

3.3.1 Técnicas e ferramentas utilizadas

A implementação do software foi dividida em duas partes, uma interface acessível pelo usuário, feita na linguagem *PHP* e usando o *framework Laravel* feito por Otwell (2015). Para a parte de controle dos motores e sensores do alimentador foi utilizado a linguagem *Python*, por já possuir diversas bibliotecas disponíveis e compatíveis com o hardware utilizado. Para o *streaming* câmera de vídeo, foi utilizado o pacote *motion* disponível para sistemas operacionais *GNU/Linux*. Mais detalhes de cada item estão presentes nas seções a seguir.

3.3.1.1 Motion

O software utilizado para realizar o *streaming* do vídeo da câmera foi o pacote *motion*. O *motion* é um software criado para sistemas operacionais *GNU/Linux* que possibilita a

captura de imagens de câmeras de vídeo. Este pacote possibilita tanto tirar uma única foto, como fazer vídeos de uma sequência delas. No caso deste trabalho, o pacote foi utilizado para criar um serviço, conhecido no mundo GNU/Linux como *daemon*, que dado um intervalo de tempo, captura a imagem da câmera e disponibiliza um *streaming* pelo *IP* e um porta do *Raspberry Pi*.

A instalação se teve da seguinte maneira, primeiramente deve se atualizar o sistema operacional, para que assim todas as bibliotecas e pacotes estejam atualizadas, isso pode ser feito através do comando `sudo apt-get update` e `sudo apt-get upgrade`. Por segundo deve se executar o comando `sudo apt-get install motion`, isto fará a instalação do pacote *motion*. Logo após é preciso alterar algumas configurações do serviço, o arquivo de configuração se encontra em `/etc/motion/motion.conf`, o Quadro 1 demonstra o conteúdo que foi alterado deste arquivo.

Quadro 1 – Arquivo de configuração do motion

```
...
daemon on //indica que deve ser executado como serviço
...
webcam_localhost off //indica que não deve ser acessível apenas localmente
e sim pela rede
...
webcam_maxrate 15 //indica a frequência máxima de imagens por segundo
tiradas
...
framerate 15 //indica a frequência de imagens por segundo
...
width 640 //indica a altura das imagens capturadas
height 480 //indica a largura das imagens capturadas
...
```

Uma vez feito as alterações no arquivo, é necessário configurar o serviço, para isto é necessário apenas abrir o arquivo `/etc/default/motion`, alterando a linha `start_motion_daemon=no` para `start_motion_daemon=yes`. Por fim é preciso apenas executar o comando `sudo service motion start`, para inicializar o serviço e o *streaming* estará disponível na porta 8081, conforme consta no arquivo de configuração, do sistema operacional. O serviço executará assim que o *Raspberry Pi* for ligado, a câmera deve estar conectada em uma das portas *USB* disponíveis. A seguir são demonstrados mais detalhes sobre a interface *WEB* criada para o usuário ter acesso as funcionalidades do alimentador, e inclusive a câmera.

3.3.1.2 Interface do sistema

A interface construída para que o usuário tenha acesso as funcionalidades do protótipo, foi desenvolvida utilizando a linguagem PHP e o *framework Laravel* criado por Otwell (2015). Suas principais funcionalidades utilizadas foram:

- a) mapeamento de rotas – permite a associação de uma rota, ou URL, a uma rotina do programa;
- b) autenticação de usuário – permite que a aplicação seja multiusuário;
- c) envio de e-mail para notificação – permite o envio de mensagens de email;
- d) fila de processamento – permite o agendamento de execução das rotinas do programa.

A função de de fila de processamento funciona da seguinte maneira: inicialmente existem duas filas, uma com os trabalhos a serem executados, e outra para os trabalhos que falharam. Esta última é utilizada para armazenar os trabalhos que, durante sua execução, tiveram algum tipo erro ou exceção. Esta estrutura traz uma maior garantia para casos em que ocorra algum problema, contudo isto demanda uma maior complexidade, já que os trabalhos podem acabar sendo executados parcialmente e assim é necessário verificar se houve algum tipo de problema com os recursos alocados.

O Quadro 2 apresenta o código do programa que realiza a chamada do programa feito na linguagem *Python* para acionar o motor e liberar o alimento.

Quadro 2 – Evento de chamada do programa Python

```
<?php namespace App\Commands;

use App\Commands\Command;

use Illuminate\Queue\SerializesModels;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Bus\SelfHandling;
use Illuminate\Contracts\Queue\ShouldBeQueued;

class Feeder extends Command implements SelfHandling, ShouldBeQueued {

    use InteractsWithQueue, SerializesModels;

    /**
     * Chama o programa Python para controle do alimentador.
     *
     * @return void
     */
    public function handle()
    {
        exec('./feeder.py');
    }
}
```

Este arquivo mostra uma classe que estende da classe `Command` do *framework Laravel*, nela existe basicamente uma única função chamado `handle`, este método é chamado pela fila de execução de trabalhos. Na função `handle`, o comando `exec` realiza a execução de um comando do sistema operacional, aqui está se executando o arquivo `feeder.py`, neste arquivo está contido o código fonte que faz o controle do motor e do sensor magnético. Mais detalhes sobre este estão presentes na seção a seguir.

3.3.1.3 Controle do motor e sensor magnético

O código fonte do programa controlador do motor e do sensor magnético é demonstrado na Figura 16.

Quadro 3 – Programa controlador do motor e sensor magnético

```

01 #!/usr/bin/env python
02 # Importação das bibliotecas utilizadas
03 import sys
04 import time
05 import RPi.GPIO as GPIO
06 # Desabilita os alertas do GPIO
07 GPIO.setwarnings(False)
08 # Utiliza a referência pelo número do pino
09 GPIO.setmode(GPIO.BOARD)
10 # Define os pinos a serem utilizados pelo motor
11 StepPins = [11,15,16,18]
12 # Inicialização das variáveis
13 StepCounter = 0
14 # Inicializa o pino do sensor magnético
15 GPIO.setup(40, GPIO.IN, pull_up_down=GPIO.PUD_UP)
16 # Variável que indica o tempo de espera entre as leituras
17 WaitTime = 1/float(1000)
18 def clearPins():
19     # Inicializa todos os pinos utilizados pelo motor como saída e coloca
o valor como falso
20     for pin in StepPins:
21         GPIO.setup(pin,GPIO.OUT)
22         GPIO.output(pin, False)
23 def runMotor():
24     global StepCounter
25     # Define a sequência do sinal a ser enviado aos pinos do motor
26     Seq = [[1,0,0,0],
27            [1,1,0,0],
28            [0,1,0,0],
29            [0,1,1,0],
30            [0,0,1,0],
31            [0,0,1,1],
32            [0,0,0,1],
33            [1,0,0,1]]
34     StepCount = len(Seq)-1
35     StepDir = 1 # 1 para sentido horário e -1 para sentido anti horário
36
37     for pin in range(0, 4):
38         xpin = StepPins[pin]
39         if Seq[StepCounter][pin]!=0:
40             GPIO.output(xpin, True)
41         else:
42             GPIO.output(xpin, False)
43
44     StepCounter += StepDir
45     # Caso chegue ao final é reiniciado a sequência
46     if (StepCounter>=StepCount):
47         StepCounter = 0
48     if (StepCounter<0):
49         StepCounter = StepCount
50
51 # Faz a leitura do pino do sensor magnético
52 def readMagnet():
53     return GPIO.input(40)
54 # Início da execução do programa
55 clearPins()
56 # Rodar o motor enquanto o sensor magnético não for acionado
57 while readMagnet():
58     runMotor()
59     time.sleep(WaitTime)

```

```
60 # Uma vez acionado é necessário retirar o sensor magnético da area do
imã
61 # por isso é novamente executado enquanto o sensor magnético é acionado
62 while not readMagnet():
63     runMotor()
64     time.sleep(WaitTime)
65 clearPins()
66 #Fim da execução do programa
```

A primeira linha do arquivo indica que este deve ser executado pelo interpretador *Python*. Entre as linhas 3 e 5, são feitas as importações das bibliotecas, respectivamente, *sys*, para acesso a algumas funções nativas do sistema operacional. A biblioteca *time*, para acesso a função *time.sleep*, fazendo o programa esperar por um tempo determinado. Por último a biblioteca *Rpi.GPIO* que permite o acesso aos pinos físicos do *Raspberry Pi*.

Entre as linhas 6 e 16 é feito as configurações necessárias para a execução do programa. A linha 7 desativa os alertas enviados pela biblioteca de *GPIO*, isto evita que o programa pare por causa destes alertas. A linha 9 configura o modo de operação do *GPIO* para trabalhar com os números dos pinos, em vez utilizar uma tabela de nomes para cada pino. A Figura 16 apresenta os pinos disponíveis para uso no modelo B+ do *Raspberry Pi*.

Figura 16 – Mapa de pinos disponíveis no Raspberry Pi modelo B+

Raspberry Pi B+ J8 Header				
Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)		DC Power 5v	04
05	GPIO03 (SCL1 , I2C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 1.1
16/07/2014

<http://www.element14.com>

Fonte: Element14 (2015).

A linha 11 inicializa a variável `StepPins` com um *array* com os números dos pinos em que o motor de passo está conectado. Já a linha 13 inicializa a variável `StepCounter`, usada como auxiliar para indicar qual a posição do *array* de estados dos pinos do motor o programa se encontra. A linha 15 inicializa o pino de número 40 como entrada, este pino será usado pelo sensor magnético. A linha 17 inicializa a variável `WaitTime`, esta indica o tempo de espera do programa para mudar o estado dos pinos do motor.

A linha 18 define a função `clearPins`, esta coloca o estado de todos os pinos para falso e como saída, isto evitará que exista algum estado fora do esperado ao inicializar o programa.

Entre as linhas 23 e 50 define-se a função `runMotor`, esta função realiza a rotação do motor de passo. Dentro desta função, a linha 24 indica que a função terá acesso a variável global `StepCounter`, desta maneira todos os ciclos de execução da função terão acesso à mesma variável e conseqüentemente ao seu estado anterior. A linha 26 define a variável `seq`, esta possuiu uma matriz com a sequência e os valores a serem usados para rodar o motor de

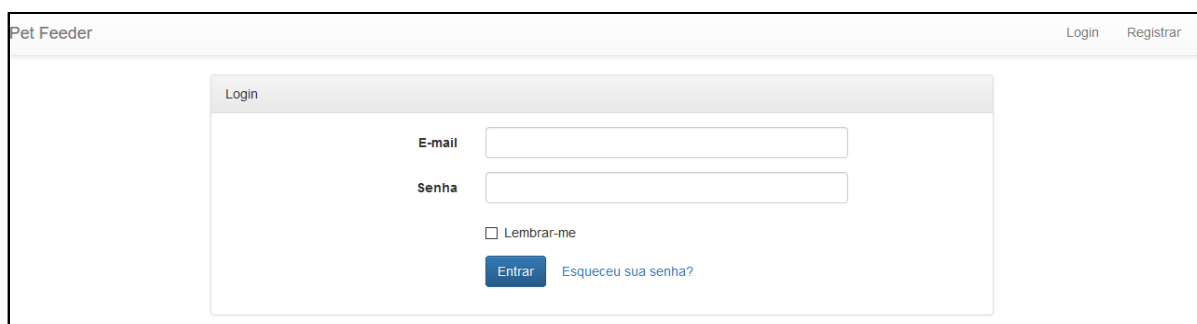
passo. A linha 34 atribui o valor da variável `StepCount` para ser utilizada no laço `for` mais a frente. A linha 35 define a variável `StepDir` que indica a direção em que o motor será acionado, sendo o valor 1 para sentido horário, e -1 para sentido anti horário. As linhas 37 a 42 é feito um laço `for` para atribuir os valores aos pinos, sendo estes valores equivalentes aos definidos na matriz `Seq`. A linha 44 incrementa o valor da variável `StepCounter` para a próxima posição da matriz. Na linha 46 é verificado se foi atingido o final da matriz, caso positivo é reiniciado para a posição inicial. Esta função não possuiu retorno.

A linha 52 define a função `readMagnet` que é usada para fazer a leitura do pino 40, que foi anteriormente utilizado para conectar o sensor magnético. Os valores retornados por esta função são 1 para quando o sensor magnético não detecta nenhum campo magnético, e 0 para quando o sensor detectar a presença de um campo magnético.

A partir da linha 55 é iniciado a execução do programa. Primeiramente é feito a limpeza dos pinos e alterado seus estados com o método `clearPins`. Logo abaixo na linha 57 é iniciado um laço `while`, que possui como parâmetro a função `readMagnet`, assim a execução da função `runMotor` é feita enquanto não for detectado nenhum campo magnético no raio de ação do sensor. Por fim, dentro do laço é chamada a função `time.sleep` para que o programa aguarde uma fração de tempo, dando tempo para o motor de passo trabalhar. Uma vez terminado a rotação, o recipiente menor do alimentador se encontra na posição correta para que o animal possa se alimentar, contudo é necessário mover o motor até o sensor sair da área de atuação do campo magnético dos ímãs. Para isso na linha 62 é feito um novo laço `while`, que executa enquanto o sensor magnético estiver acionado, colocando o recipiente numa posição livre para a próxima execução. Por fim é chamado a função `clearPins` para limpar o estado dos pinos.

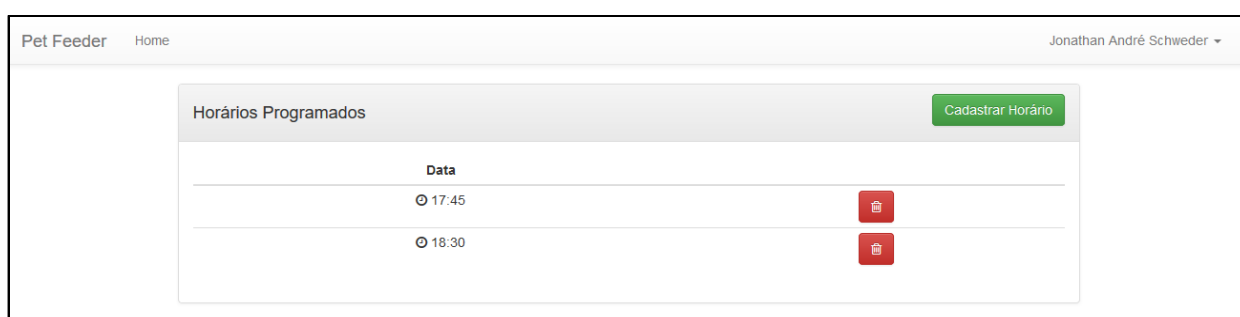
3.3.2 Operacionalidade da implementação

Esta seção descreve o funcionamento do protótipo através de um estudo de caso. Para utilizar o protótipo é necessário ligar a uma fonte de energia. Ao ligar o alimentador será necessário acessar por um navegador *WEB* o IP do mesmo. A primeira página mostrada é a página de *login*, onde é necessário apenas informar o e-mail e senha (Figura 17).

Figura 17 – Página de *login*

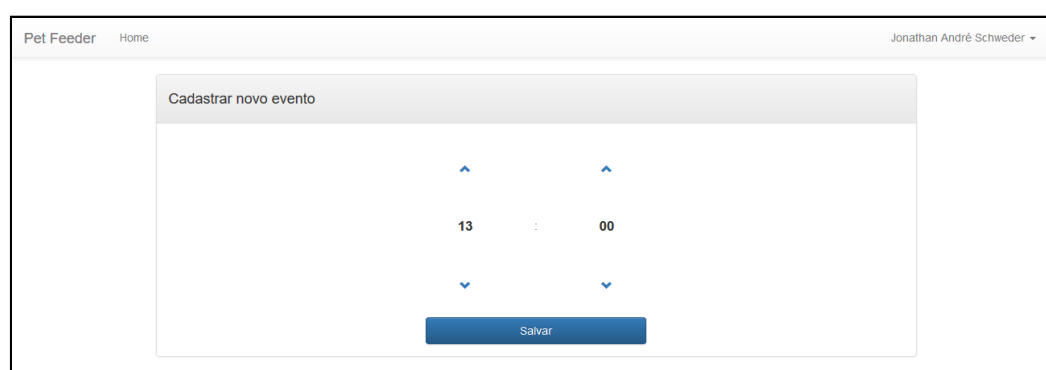
Ao digitar o e-mail e senha e clicar no botão *Entrar*, o usuário é redirecionado para a página principal (Figura 18). Nesta página é possível realizar o cadastro dos horários em que se quer liberar o alimento.

Figura 18 – Página principal



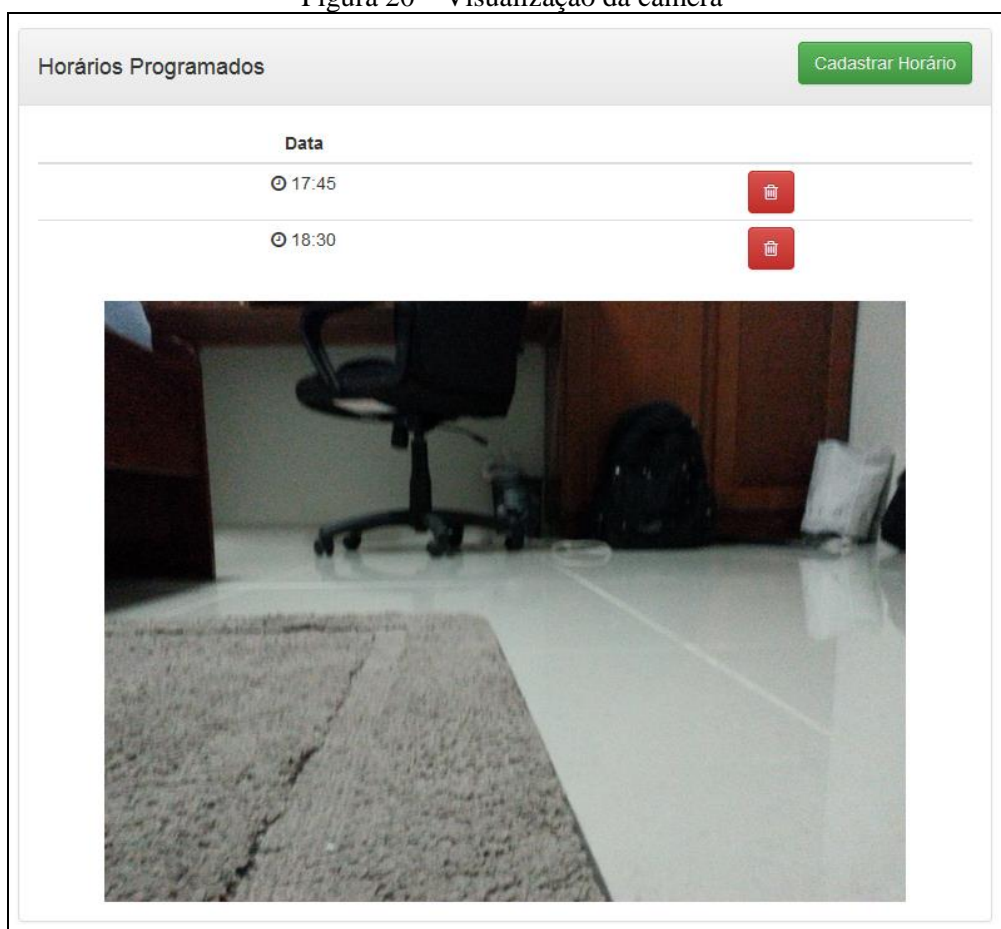
Ao clicar no botão *Cadastrar Horário* o usuário é redirecionado para a página de cadastro de horários (Figura 19). Nesta página o usuário informa a hora em que quer liberar o alimento e o sistema colocará na fila de execução o evento para o horário marcado. Os horários são diários, portanto sempre no horário cadastrado será executado o programa de controle do motor e sensor.

Figura 19 – Página de cadastro do horário de liberação do alimento



Ao cadastrar um novo horário o sistema cria novas instâncias de eventos na fila de execução, ao final da execução do evento é novamente adicionado um novo evento para ser executado no dia seguinte. Por fim, na página inicial também é possível visualizar a imagem de *streaming* de vídeo da câmera ligada ao *Raspberry Pi* (Figura 20).

Figura 20 – Visualização da câmera



3.4 RESULTADOS E DISCUSSÕES

Durante a construção da estrutura do protótipo pensou-se em diversas estruturas baseadas em outros alimentadores disponíveis no mercado. Dentre as estruturas pensadas tentou-se utilizar uma que utilizava um repositório para a comida, com um cano acoplado para despejar o alimento. Seria colocada uma rosca para controlar o despejo do alimento e a quantidade. Contudo houve dificuldade em conseguir fazer o encaixe da rosca e também ocorria em alguns testes do alimento não ser devidamente transportado. Em diversos momentos o alimento não era puxado para o cano e conseqüentemente não era despejado na tigela para o animal.

Pela experiência adquirida com este primeiro protótipo a estrutura do alimentador foi totalmente refeita. A nova estrutura segue algo semelhante ao alimentador FeedAndGo (2015). Esta nova estrutura foi confeccionada em material plástico, isto permite tornar o alimentador impermeável, podendo tanto armazenar ração úmida como água. Apesar disto é fato que o plástico não é um material fácil de ser moldado. Portanto precisou realizar diversas pesquisas de possíveis materiais e produtos que poderiam ter formas já aproveitáveis para a

construção do protótipo. No fim chegou-se as embalagens de bolo, conforme mostrado na Figura 21, que possuem diversos tamanhos, portanto a ideia da nova estrutura se tornou viável, uma vez que seria possível colocar uma embalagem menor, para armazenar o alimento, dentro de uma embalagem maior para proteger a estrutura.

Figura 21 – Embalagem utilizada na construção do protótipo



Fonte: Higipack (2015).

O *Raspberry Pi* se demonstrou eficiente no que diz respeito ao controle do motor e sensor magnético. Fazendo-se uma comparação com o trabalho feito por Ochakowski (2007), que utilizou um micro controlador PIC16F877, percebe-se que nos resultados obtidos comentou-se da lentidão no processo de leitura e escrita dos dados na memória EEPROM, já na utilização do *Raspberry Pi* que utiliza um cartão SD para armazenamento, em nenhum momento observou-se alguma lentidão na execução dos programas. Uma vez não identificado nenhuma lentidão no processo de escrita e leitura não foi necessário criar nenhuma alternativa como o apresentado por Ochakowski (2007), que utilizou um *array* em memória para fazer o armazenamento e leitura temporários.

Em se tratando de entradas efetuadas pelo usuário, a utilização de uma interface *WEB* facilitou este quesito. Sendo possível realizar todas as operações do alimentador a partir de qualquer navegador *WEB*, aumentando sua aplicabilidade, já que nos dias atuais todos os usuários que acessam qualquer serviço da internet, seja o uso de redes sociais, envio de e-mail, armazenamento de arquivos em nuvem ou qualquer outra atividade relacionada, possibilita que o usuário reaproveite este conhecimento para ter acesso às funções criadas.

Os periféricos utilizados se mostraram eficientes em suas funções, contudo precisaram de adaptações para atender ao projeto. O motor de passo modelo “28BYJ-48”, não apresentou falhas em nenhum momento. Contudo observou-se que o motor somente era capaz de girar a estrutura com determinado peso. Como tentativa de solucionar este problema foi colocada uma estrutura circular abaixo do recipiente menor, esta estrutura possui rodas que facilitam a rotação do recipiente, distribuindo melhor o peso, solucionando o problema

anteriormente descrito. Além disso com a utilização do sensor magnético para detectar o ponto de parada, observou-se que a utilização de qualquer outro tipo de motor se torna possível, não sendo obrigatório o motor ser de passo.

O sensor magnético utilizado, modelo “CONT NA SM1001” atendeu as necessidades do projeto. Em nenhum momento observou-se o não reconhecimento ou qualquer defeito do sensor. Contudo, os ímãs de neodímio no formato de “moeda”, que apesar de gerarem um campo magnético num raio suficiente para acionar o sensor magnético, tiveram que ser trocados, por ímãs do mesmo tipo, porém no formato de “bastão”. Esta mudança deveu-se pela variação causada na distância entre o sensor magnético e o recipiente menor interno, pois os ímãs estão presentes neste último.

Em se tratando da implementação do software, o uso do *framework Laravel* e sua estrutura base para desenvolvimento de aplicações auxiliaram e abstraíram diversos processos que demandariam muito mais tempo caso fossem implementados sem o uso deste. A documentação do *framework* foi suficiente para criar a implementação do protótipo. Para a parte de controle do hardware as bibliotecas existentes na linguagem *Python* foram suficientes para fazer o uso das funções dos periféricos utilizados.

Algo que também pode ser citado aqui é a questão do custo. A Tabela 1 demonstra dados retirados de um fornecedor brasileiro do *Raspberry Pi*.

Quadro 4 –Modelos de Raspberry Pi

Modelo	Raspberry Pi 2 B	Raspberry Pi 1 B+	Raspberry Pi A+
Preço	286,00	250,00	105,71
Processador	900MHz	700MHz	700MHz
Memória	1GB	512MB	256MB
Portas USB	4	4	1
Placa Gráfica	VideoCore IV 3D graphics core	Não possui	Não possui

Fonte: Bra (2015).

Como pode-se verificar no Quadro 4, existem diversos modelos da placa *Raspberry Pi*. O modelo utilizado para este trabalho, o modelo B+, é um modelo intermediário, tanto seu custo como suas configurações são medianas. O modelo 2 do *Raspberry Pi* trouxe diversas melhorias em relação ao seu modelo anterior. Como podemos ver houve melhorias no processador utilizado, sendo 200MHz mais rápido que seu modelo anterior. Além disso esta versão também teve melhorias na quantidade de memória disponível e foi adicionado agora uma placa gráfica para o processamento dedicado de imagens. Tudo isso faz com que a versão do 2 seja mais aconselhada a aplicações e projetos que necessitem de um poder maior de processamento e que tirem vantagem do tamanho reduzido que todos os *Raspberry Pi* possuem. O último modelo analisado, o A+, possui uma configuração mais modesta, possui

menos memória que o modelo B+, além de possuir uma quantidade menor de portas USB disponíveis, fazendo que com que não se possa conectar tantos dispositivos como nas outras versões. Contudo, com seu preço sendo muito menor que as outras versões, o modelo A+ se torna ideal para aplicações menores ou que não necessitem de tantos dispositivos com conexões USB.

Concluindo, percebeu-se que de modo geral a abstração possibilitada pelo *Raspberry Pi* de diversas funções do hardware fez com que não fosse necessários maiores conhecimentos deste, podendo-se citar aqui o controle de memória e armazenamento no cartão SD. O sistema operacional *Raspbian* foi criado exclusivamente para o hardware do *Raspberry Pi*, além de ser um sistema GNU/Linux, trazendo todo o ambiente, bibliotecas, programas e serviços criados para estes sistemas operacionais, o suporte à linguagem *PHP* possibilitou o uso do *framework Laravel* na construção da interface *WEB*, sem nenhuma adaptação necessária para que a aplicação funcione no *Raspberry Pi*.

4 CONCLUSÕES

Este trabalho apresentou o protótipo de um alimentador para auxiliar na nutrição de animais domésticos. Como visto no presente trabalho, assim como as pessoas os animais possuem horários definidos para refeições. Os objetivos do trabalho foram totalmente atingidos. As ferramentas utilizadas mostraram-se além da expectativa e foram adequadas para o êxito deste trabalho. Dando-se apenas um destaque para a dificuldade com a precisão do eixo de rotação do recipiente menor, que devido a sua montagem, em certos momentos apresentou variação na distância entre a tampa externa, onde se localizava o sensor magnético, e os imãs.

A principal vantagem deste projeto em relação aos alimentadores existentes no mercado é sua interface *WEB*, o que facilita o uso pelo usuário que nos dias atuais está acostumado a utilizar ferramentas online no seu dia a dia como redes sociais, armazenamento de arquivos na nuvem e e-mails. Além disto, o uso do *Raspberry Pi* abstrai diversas funções do hardware, como controle de memória e armazenamento, ainda sendo o sistema operacional do tipo GNU/Linux, que possui uma grande gama de projetos de código aberto, possibilita a integração com outros dispositivos. Apesar de tudo, o alimentador está limitado na quantidade de liberações disponíveis, no caso do protótipo final foram quatro compartimentos para poderem ser liberados. Além disto, não é possível determinar a quantidade de alimento a ser liberado, algo que o trabalho de Ochakowski (2007) demonstrou uma estrutura que permite isto.

4.1 EXTENSÕES

Como sugestão para futuros trabalhos

- a) adicionar novos periféricos para que o usuário possa brincar com o animal;
- b) adicionar suporte a microfones para que o usuário possa transmitir sua voz ao animal;
- c) possibilitar todo o processo de programação dos agendamentos de horários através de dispositivos móveis;
- d) utilizar a linguagem *Java* na implementação do fonte.

REFERÊNCIAS

ABINPET. **População de pets cresce 5% ao ano e Brasil é quarto no ranking mundial.** 2012. Disponível em: <<http://abinpet.org.br/imprensa/noticias/populacao-de-pets-cresce-5-ao-ano-e-brasil-e-quarto-no-ranking-mundial/>>. Acesso em: 29 mar. 2015.

ADAFRUIT. **Raspberry Pi modelo B+.** Disponível em: <<https://www.adafruit.com/datasheets/pi-specs.pdf>>. Acesso em: 03 maio 2015.

ALBANO, L. L. M. **Saúde animal: aspectos importantes da nutrição canina.** São Carlos, [2007]. Disponível em: <http://www.saudeanimal.com.br/artigo_luigi_nutricao010.htm>. Acesso em: 07 abr. 2015.

BARROS, André Ricardo Gouveia. **Implementação de um servidor utilizando Linux embarcado com acesso e gerenciamento através de smartphone.** 2013. 110 f. TCC (Graduação) - Curso de Engenharia de Computação, Engenharia Elétrica e de Computação, Universidade de São Paulo Escola de Engenharia de São Carlos, São Carlos, 2013. Disponível em: <<http://www.tcc.sc.usp.br/tce/disponiveis/97/970010/tce-04022014-160755/>>. Acesso em: 4 abr. 2015.

BRA, Raspberry Pi. **Modelos de Raspberry.** 2015. Disponível em: <<http://raspberrypibra.com/>>. Acesso em: 27 nov. 2015.

ELEMENT14. **Raspberry Pi B+ J8 Header.** 2015. Disponível em: <<http://www.element14.com/community/docs/DOC-68203/1/raspberry-pi-b-gpio-40-pin-block-pinout>>. Acesso em: 25 nov. 2015.

FEEDANDGO. **FeedAndGo.** 2015. Disponível em: <<http://www.feedandgo.com/>>. Acesso em: 5 abr. 2015.

HIGIPACK. **Embalagens para bolos.** 2015. Disponível em: <<http://www.higipack.com.br/embalagem-bolo-torta-millenium-grande-baixa-3kg/>>. Acesso em: 25 nov. 2015.

OCHAKOWSKI, Nádia. **Protótipo de um alimentador automático para animais de estimação.** 2007. 50 f, il. Trabalho de Conclusão de Curso - (Graduação em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2007. Disponível em: <http://www.bc.furb.br/docs/MO/2008/329255_1_1.pdf>. Acesso em: 03 abr. 2015.

ORSINI, S. **Mercado aposta em animais de estimação.** [S.l.], 2004. Disponível em: <<http://financas.cidadeinternet.com.br/article.asp?878~196264>>. Acesso em: 02 abr. 2015.

OTWELL, Taylor. **Laravel: The PHP Framework For Web Artisans**. Disponível em: <<http://laravel.com/>>. Acesso em: 18 nov.

PRODAC. **Magic Feeder**. 2010. Disponível em: <http://www.prodacinternational.it/as/index.php?page=shop.product_details&flypage=flypage.tpl&product_id=162&category_id=19&option=com_virtuemart&Itemid=23&lang=en>. Acesso em: 1 mar. 2015.

PETBR, **A força dos nutrientes**. Disponível em: <http://www.petbrasil.com.br>, Acesso em: 28 mar. 2015.

PETLOVE. **Alimentador Automático Prodac**. Disponível em: <<http://www.petlove.com.br/alimentador-automatigo-prodac---150ml-3103515/p?gclid=CJeS9-Gm1sQCFQITwwodTmsA-Q#box-description>>. Acesso em: 4 mar. 2015.

RASPBERRY PI FOUNDATION. **Homepage**. Disponível em: <<http://www.raspberrypi.org/>>. Acesso em: 7 abr. 2015.

XUN, Y. et al. **A platform for system-on-a-chip design prototyping**. In: ASIC, 2001. Proceedings. 4th International Conference on. [S.l.: s.n.], 2001. p. 781–784.