FACULDADE DE TECNOLOGIA DE SÃO BERNARDO DO CAMPO
“ADIB MOISÉS DIB”

ADÔNIS DELVAGE FAGUNDES
DIOGO ALVES DOS SANTOS
FELIPE OLIVEIRA E SILVA
VINICIO NICOLAU DA SILVA

ALIMENTADOR AUTOMATIZADO PARA CÃES

São Bernardo do Campo - SP
Novembro/2016

ADÔNIS DELVAGE FAGUNDES
DIOGO ALVES DOS SANTOS
FELIPE OLIVEIRA E SILVA
VINICIO NICOLAU DA SILVA

ALIMENTADOR AUTOMATIZADO PARA CÃES

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moises Dib” como requisito parcial para a obtenção do título de Tecnólogo em Automação Industrial.

Orientador: Prof. Me. Pedro Adolfo Galani

Co-orientador: Prof. Dr. Delcinio Ricci

São Bernardo do Campo - SP
Novembro/2016

ADÔNIS DELVAGE FAGUNDES
DIOGO ALVES DOS SANTOS
FELIPE OLIVEIRA E SILVA
VINICIO NICOLAU DA SILVA

ALIMENTADOR AUTOMATIZADO PARA CÃES

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moises Dib” como requisito parcial para a obtenção do título de Tecnólogo em Automação Industrial.

Trabalho de Conclusão de Curso apresentado e aprovado em: 25/11/2016.

Banca Examinadora:

_____

Prof. Me. Pedro Adolfo Galani, FATEC SBC - Orientador.

_____

Prof. Me Maurício Marsura - Avaliador.

_____

Prof. Me. Rômulo Oliveira de Albuquerque - Avaliador.

Dedicamos esse trabalho aos nossos pais e esposas e a todos os professores que nos ajudaram, e aos colegas que de alguma forma também contribuíram.

Agradecemos primeiramente a Deus ao Prof. Me. Pedro Adolfo Galani e ao Prof. Dr. Delcinio Ricci pela ajuda durante a elaboração deste trabalho.

“Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que pareciam impossíveis.”

CHARLES CHAPLIN

RESUMO

Este projeto tem por objetivo desenvolver um alimentador automático, que pode ser controlado e monitorado remotamente pelo *smartphone*, fornecendo ração e água para cães de forma controlada, propiciando o bem estar do animal e a praticidade para o seu dono. Para o desenvolvimento desse protótipo é necessário inicialmente fazer um levantamento bibliográfico em busca de conceitos como: portes de cães, quantidade e dosagem de ração e a quantidade diária de água. Ao longo desse levantamento constatou-se que no mercado atual há uma carência de produtos com as características ao que foi desenvolvido. Utilizando uma tecnologia inovadora, o protótipo é projetado e construído com dispositivos que possibilitam a operação à distância, permitindo ao usuário que alimente o cão em qualquer lugar através de um aplicativo, que pode ser baixado e instalado no *smartphone*. A operação do alimentador é bem simples, após baixar o aplicativo no *smartphone*, e instalar o alimentador, basta o usuário inserir os dados do seu cão e programar a quantidade e os períodos de fornecimento de ração e água. Após a confirmação, os dados são transmitidos do aplicativo do *smartphone* para o controlador local via *Wi-Fi*. Também podem ser visualizadas em tempo real, nesse aplicativo, informações sobre os níveis dos reservatórios de ração e água, e alertas que são emitidos automaticamente quando os níveis estão abaixo do especificado pela programação.

Palavras-chave: Cães. Ração. Água. Aplicativo. *Smartphone.*

ABSTRACT

This project aims to develop an automatic feeder, which can be controlled and monitored remotely by smartphone, providing for the dog food and water in a controlled manner**,** the animal welfare and convenience to its owner. For the development of this prototype is necessary first to review some literature in search of concepts such as dog sizes, quantity and dosage of feed and the daily amount of water. Throughout this survey it was found that in the current market there is a shortage of products with the features to what was developed. Using innovative technology, the prototype is designed and built with devices that enable remote operation, allowing the user to feed the dog anywhere through an application that can be downloaded and installed on the smartphone. The feeder operation is very simple, after downloading the app on your phone, and install the feeder, the user simply enter your dog's data and program the amount and periods of supply of food and water. After confirmation, the data is transmitted from the smartphone application to the local controller by Wi-Fi. They can also be viewed in real time, in this application, information on the levels of food and water reservoirs, and alerts that are sent automatically when the levels are below the specified programming.

Key words: Dogs. Food. Water. Application. Smartphone.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

A	Amperagem
A/D	Analógico/Digital
CLP	*Programmable Logic Controller* (Controlador Lógico Programável)
DC	*Direct Current* (Corrente Contínua)
FATEC	Faculdade de Tecnologia
FOFA	Forças, Oportunidades, Fraquezas e Ameaças
GND	Ground (Terra)
g	Gramas
IMCC	Índice de Massa Corporal Canina
I/O	Inputs/Outputs (Entradas/Saídas)
Kg	Quilograma
LED	*Light Emitting Diode* (Diodo Emissor de Luz)
LCD	*Liquid Crystal Display* (Tela de Cristal Liquido)
PDCA	*Plan, Do, Check e Action* (Planejar, Executar, Testar e Agir)
RFID	Identificação Radio Frequência
SGBD	Sistema Gerenciador de Banco de Dados
USB	*Universal Serial Bus* (Porta Universal)
US$	*Dolar* (Moeda Norte Americana)
Vca	Voltagem em Corrente Alternada
VCC	Voltagem em Corrente Contínua
V	Voltagem
XML	*Extensible Markup Language* (Linguagem Extensível de Marcação)
WI-FI	*Wireless Fidelity* (Fidellidade sem Fio)

SUMÁRIO

INTRODUÇÃO

Os cães são cada vez mais presentes nos lares, seja para alegrar o ambiente, suprir a necessidade de afeto do dono, ou até mesmo para vigiar uma residência. O fato é que em muitos lares os cães já fazem parte da família como um ente querido e é o animal que mais se afeiçoa ao homem, é um amigo sincero e dedicado e está sempre à espera do seu dono.

Na indústria cinematográfica é possível verificar vários exemplos da afeição de humanos com os cães, como no filme: Sempre ao seu lado (2009), baseado em uma história real que aconteceu no Japão em 1987 e mostra a história de um cão chamado Hachiko, que sempre aguardava seu dono na frente de uma estação de trem. Mesmo após a morte do seu dono, Hachiko passou a esperá-lo todos os dias até o fim da sua vida.

A alimentação dos cães é um grande fator para a sua saúde, pois todos os seres vivos precisam ser alimentados e os cães domésticos dependem de seus donos para ter uma alimentação diária.

Com o passar dos anos o homem fica cada vez mais ausente do lar e com tempo escasso para cuidar de seu cão. E os cães ficam a maior parte dos dias sozinhos. Este trabalho de conclusão de curso pode ser extremamente útil para grande parte dos donos de cães que não viajam em decorrência de terem que alimentar seus cães, muitas vezes por não terem ninguém para alimentá-los, ou pelo fato de não serem receptivos com pessoas estranhas, e ainda há donos que deixam excessivas quantidades de ração e água podendo estragar ou trazer insetos indesejáveis, ocasionando doenças nos cães.

Considerando o problema da alimentação dos cães, o objetivo deste trabalho é construir um Alimentador Automatizado para Cães que fornece ração e água conforme programação pré-estabelecida, através do aplicativo instalado no

smartphone. Justifica-se que o alimentador proporciona ao animal melhor bem-estar em relação à alimentação.

A automação deste processo se realiza por um sistema programável e um aplicativo, onde o dono do cão pode agendar quando a ração e a água serão fornecidas. O reservatório de ração é monitorado através de sensor, informando ao usuário se está com o nível baixo, e um motor envia, em pequenas quantidades, a ração ao prato. A água é medida por sensores que detectam o nível no reservatório, de forma que, quando a água baixar, o sensor envia um sinal para o sistema, solicitando ao usuário reabastecer o recipiente. A liberação da água é por uma válvula instalada na saída do reservatório, essa que é acionada pelo controlador. Para que a água não permaneça parada, a exemplo da proliferação do mosquito da dengue, periodicamente será substituída através de um dreno na parte inferior do reservatório que se abre para eliminar a água automaticamente.

O trabalho é dividido em três capítulos:

No capítulo 1 - Fundamentação Teórica: apresenta pesquisas que dão sustentação para o desenvolvimento do tema proposto como: alimentação canina, condição física dos animais, água para os cães, alimentadores disponíveis no mercado, plataforma de programação Arduino, dispositivos móveis, dispositivos para a automação do protótipo.

No capítulo 2 - Metodologia: aborda a trajetória percorrida para o desenvolvimento do trabalho. É composta por métodos e técnicas apropriadas.

No capítulo 3 - Desenvolvimento de Projeto: aborda passo a passo o desenvolvimento de construção do protótipo.

E finalmente, as Considerações Finais: em termos gerais, aborda-se o objetivo e justificativas, bem como relações entre os fatos verificados e as teorias, conquistas alcançadas, pontos fortes e fracos e sugestões para futuros trabalhos.

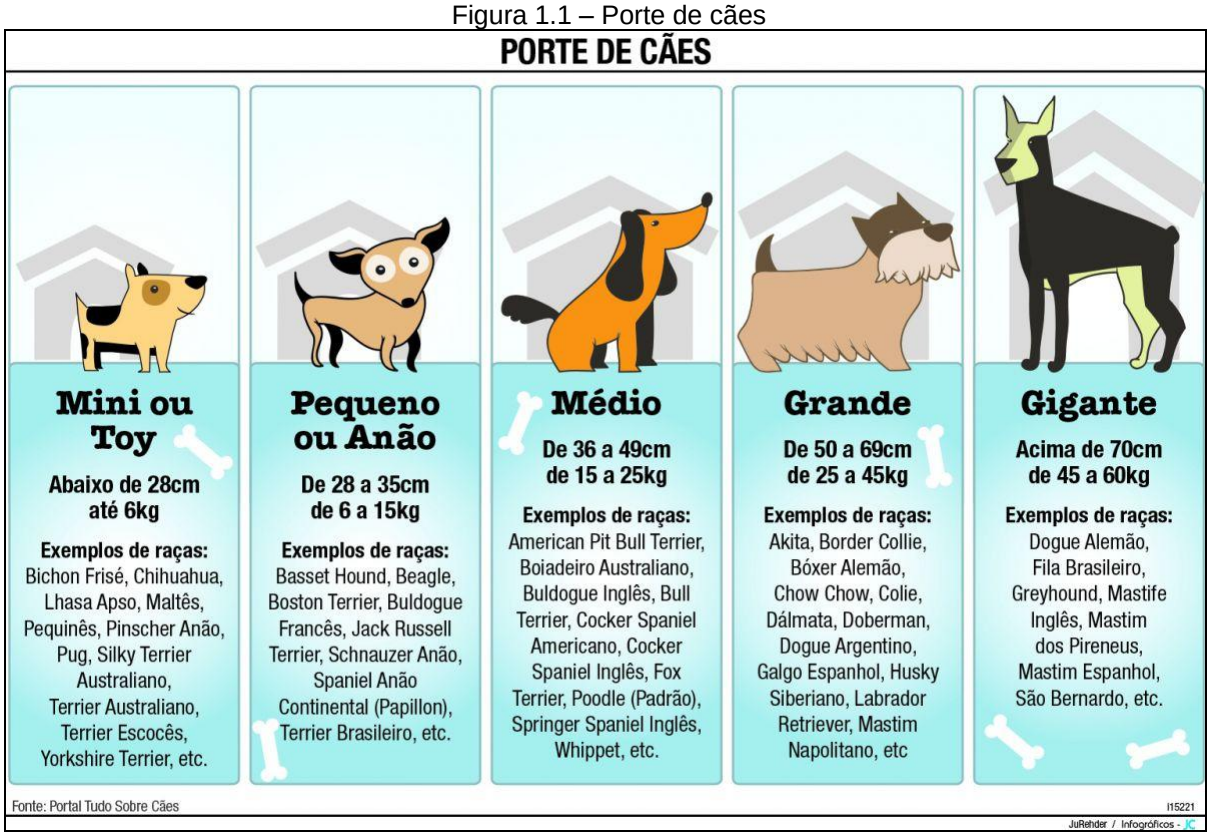
1 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo encontram-se as teorias que dão embasamento para o desenvolvimento do projeto que se intitula Alimentador Automatizado para Cães.

1.1 Alimentação para cada fase da vida canina

Tubaldini (2014) relata que é essencial ter controle sobre a quantidade de ração que os cães precisam, uma vez que o consumo inadequado da ração acarreta risco de excesso de peso, problemas gastrointestinais e contribui para um desequilíbrio nutricional. O fornecimento de ração é diferente para cada fase da vida dos cães, e é divido em três categorias: filhotes, adultos de 1 a 7 anos e adultos acima de 7 anos. Os portes de cães estão ilustrados na figura 1.1.

Figura 1.1 – Porte de cães



Fonte: www.jcnet.com.br, 2015

Filhote miniatura e pequeno: nessa fase os animais precisam de mais energia para desempenhar as funções vitais de seu organismo, estão em processo de

formação muscular e do próprio corpo. Deste modo, há mais necessidade de consumo de proteínas, gorduras e vitaminas do que em relação aos adultos, e a frequência da alimentação é maior. A tabela 1.1 ilustra a alimentação recomendada para filhotes de pequeno porte e miniatura.

Tabela 1.1 – Alimentação de filhotes de pequeno porte e miniatura

Idade em meses	Tamanho do cão	Quantidade em gramas	Frequência diária
Desmame a 3	Pequeno (5kg a 9kg)	60 - 120	3
3 a 5	Pequeno (5kg a 9kg)	135 - 180	3
5 a 8	Pequeno (5kg a 9kg)	90 - 120	2
8 a 12	Pequeno (5kg a 9kg)	120 - 135	2
Desmame a 3	Miniatura (1kg a 5kg)	60 - 120	3
3 a 5	Miniatura (1kg a 5kg)	70 - 120	3
5 a 8	Miniatura (1kg a 5kg)	70 - 135	2

Fonte: www.purinalatam.com.br, 2016

Filhote médio: a alimentação dos filhotes de médio porte segue o mesmo principio dos filhotes pequenos e miniatura, mas a quantidade deve ser maior e proporcional ao porte. A tabela 1.2 ilustra a alimentação recomendada para filhotes de médio porte.

Tabela 1.2 – Alimentação de filhotes de médio porte

Idade em meses	Tamanho do cão	Quantidade em gramas	Frequência diária
Desmame a 3	Médio (9kg a 22kg)	70 - 150	3
3 a 5	Médio (9kg a 22kg)	200 - 270	3
5 a 8	Médio (9kg a 22kg)	150 - 230	2
8 a 12	Médio (9kg a 22kg)	250 - 290	2
Desmame a 3	Médio (22kg a 34kg)	75 - 230	3
3 a 5	Médio (22kg a 34kg)	210 - 380	3
5 a 8	Médio (22kg a 34kg)	180 - 360	2
8 a 12	Médio (22kg a 34kg)	320 - 450	2

Fonte: www.purinalatam.com.br, 2016

Filhote grande e gigante: a alimentação dos filhotes de portes grandes e gigantes também segue o mesmo principio dos filhotes pequenos e miniatura, mas a quantidade deve ser maior e proporcional ao seu porte. A tabela 1.3 ilustra a alimentação recomendada para filhotes gigantes e de grande porte.

Tabela 1.3 – Alimentação de filhotes de grande porte e gigantes

Idade em meses	Tamanho do cão	Quantidade em gramas	Frequência diária
Desmame a 3	Grande (34kg a 45kg)	90 - 340	3
3 a 5	Grande (34kg a 45kg)	300 - 570	3
5 a 8	Grande (34kg a 45kg)	405 - 650	2
8 a 12	Grande (34kg a 45kg)	900 - 1080	2
Desmame a 3	Gigante (acima de 45kg)	300 - 540	3
3 a 5	Gigante (acima de 45kg)	495 - 765	3
5 a 8	Gigante (acima de 45kg)	610 - 855	2
8 a 12	Gigante (acima de 45kg)	1080 - 1260	2

Fonte: www.purinalatam.com.br, 2016

Adultos de 1 a 7 anos: nesta fase, a frequência do fornecimento de ração pode ser diminuída para duas vezes ao dia. Não há necessidade de grandes consumos de proteínas, gorduras e vitaminas em relação aos filhotes. A tabela 1.4 ilustra a alimentação recomendada para cães adultos de 1 a 7 anos.

Tabela 1.4 – Alimentação de cães adultos de gigante, grande, pequeno porte e miniatura.

Idade em anos	Tamanho do cão	Quantidade em gramas	Frequência diária
1 a 7	Miniatura (1kg a 5kg)	30 - 105	2
1 a 7	Pequeno (5kg a 9kg)	105 - 135	2
1 a 7	Médio (9kg a 22kg)	160 - 315	2
1 a 7	Médio (22kg a 34kg)	315 - 430	2
1 a 7	Grande (34kg a 45kg)	430 - 520	2
1 a 7	Gigante (acima de 45kg)	520+45 para cada 5kg de peso corporal acima de 45kg	2

Fonte: www.purinalatam.com.br, 2016

Adultos acima de 7 anos: a partir desta idade, os cães podem ser considerados idosos. Com isso a quantidade deve ser diminuída e a ração deve ser comprada de acordo com a idade do cão, pois os nutrientes devem ser diferentes da fase adulta entre 1 e 7 anos. A tabela 1.5 ilustra a alimentação recomendada para cães adultos acima de 7 anos.

Tabela 1.5 – Alimentação de cães adultos de gigante, grande, pequeno porte e miniatura.

Idade em anos	Tamanho do cão	Quantidade em gramas	Frequência diária
Acima de 7	Miniatura (1kg a 5kg)	45 – 90	2
Acima de 7	Pequeno (5kg a 9kg)	90 – 135	2
Acima de 7	Médio (9kg a 22kg)	135 - 270	2
Acima de 7	Médio (22kg a 34kg)	270 - 390	2
Acima de 7	Grande (34kg a 45kg)	390 - 475	2
Acima de 7	Gigante (acima de 45kg)	475+30 para cada 5kg de peso corporal acima de 45kg	2

Fonte: www.purinalatam.com.br, 2016

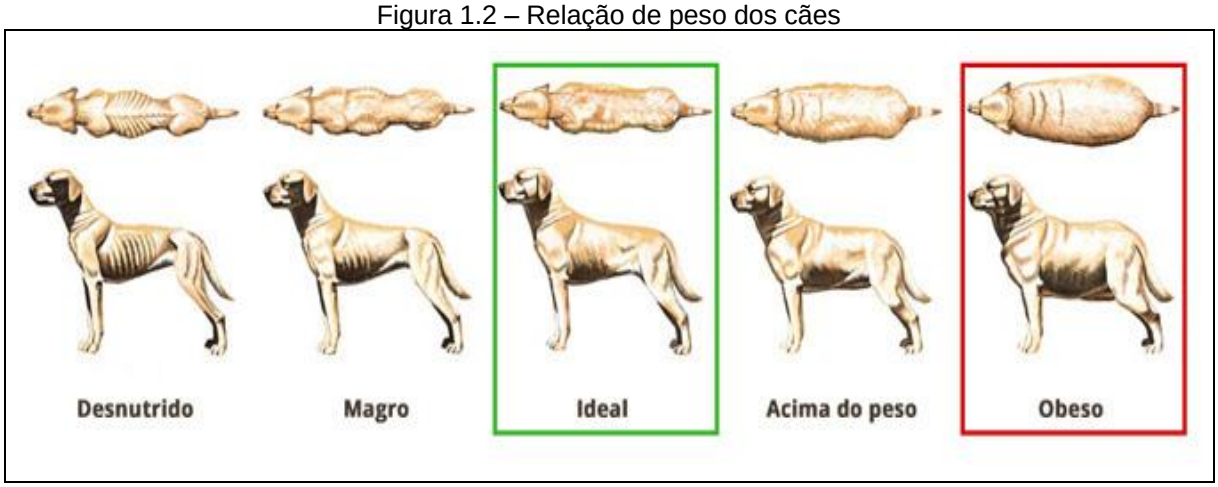
Nabais (2010) destaca que até o quinto mês de idade é fornecido três refeições diárias, a partir do sexto mês o número de refeições deve diminuir para duas até o final do período de crescimento, que é entre 1 e 7 anos. Na idade adulta, acima dos 7 anos, o animal pode fazer uma única refeição diária, mas é preferível duas.

Os intervalos das refeições para cães de até 5 meses, pode ser em um intervalo de 8 horas, programando o primeiro fornecimento pela manhã, o segundo pela tarde e o terceiro pela noite. Para cães acima de 6 meses, as rações podem ser fornecidas em um intervalo de 12 horas, programando o primeiro fornecimento pela manhã e o segundo pela tarde/noite.

1.2 Condição física dos cães

De acordo com Rodrigues (2011), um animal está equilibrado energeticamente, quando a energia ingerida é consumida pelo seu organismo. Quando o balanço energético está positivo significa que a ingestão de energia excede o consumo, de modo que a energia excedente se acumula no tecido adiposo deixando o cão com sobrepeso ou até mesmo obeso. Quando o balanço energético é negativo significa que a ingestão não está suprindo as necessidades, o organismo começa a degradar seus tecidos para cobrir as necessidades energéticas, à medida que os depósitos vão se esgotando, o animal passa a perder peso. A figura 1.2 ilustra a relação de peso dos cães.

Figura 1.2 – Relação de peso dos cães



Fonte: www.arcadenoe.pt, 2016

Segundo Rodrigues (2011), a obesidade é uma das formas mais importantes e frequentes da má nutrição. Numerosos fatores podem predispor um animal à obesidade, dentre eles a genética, a intensidade de atividade física e a quantidade de energia da dieta. As principais doenças agravadas pela obesidade canina são: aparecimento de problemas articulares e locomotores (discopatias e rompimento de ligamento cruzado), alterações endócrinas, intolerância a glicose e consequentemente risco aumentado e agravado de diabetes, dificuldades respiratórias, risco aumentado de neoplasias e pancreatite, aumento de risco cirúrgico, hérnia inguinal, diminuição da resistência física, presença de constipação e flatulência.

Avaliar o peso é uma medida usada como estimativa do estado nutricional. Para determinar o sobrepeso ou desnutrição, compara-se o peso atual com os pesos anteriores registrados. Em casos de raças puras compara-se com o peso padrão da raça. Muitos proprietários não fazem ideia de quanto pesa o seu cão, mas se soubessem poderiam evitar diversas doenças e até mesmo o óbito do animal.

Guimarães (2009 apud RODRIGUES, 2011) testou quatro equações de índice de massa corporal canina (IMCC), e o IMCC (3) foi a mais representativa para expressar a relação entre o peso e o comprimento do animal.

- IMCC (1) = peso (kg) / altura da cernelha (m)
- IMCC (2) = peso (kg) / altura da cernelha² (m)

- IMCC (3) = peso (kg) / comprimento da coluna (m)
- IMCC (4) = peso (kg) / comprimento da coluna² (m)

O IMCC tem utilidade tanto para detectar o excesso de peso, como para identificar animais que estejam abaixo do peso. O dono do cão pode verificar as variações do peso de seu cão, mas é de extrema importância que, ao detectar variações bruscas, o animal deve ser levado ao medico veterinário para que ele dê um real diagnóstico.

1.3 Água para os cães

Alessandro (2012) relata que o cão considerado saudável ingere aproximadamente 20 a 90 ml/dia de água, lembrando que a variação ocorre devido ao estilo de vida de cada cão, se é bem agitado ou preguiçoso e do teor de umidade da ração ingerida. Considerar também o tamanho do animal, pois o bom senso é importante neste caso, à água não pode faltar para os cães nos dias de calor intenso, ela tem um papel fundamental por controlar o aumento da temperatura do animal. Nos dias mais quentes os cães ingerem um volume maior de água. Deve-se ter uma atenção especial com a reposição de água.

1.4 Alimentadores disponíveis no mercado

O *site* petlove (2016) relata que o *Trixie TX4 Plus* é um alimentador programável para cães de todos os portes, seu reservatório tem capacidade de 500 ml. Possui um temporizador programável de 96 horas e sua estrutura comporta apenas a ração, conforme ilustra a figura 1.3.

Figura 1.3– Alimentador automático Trixie TX4 Plus

Fonte: www.petlove.com.br, 2016

O *site* megatnt (2016) aponta que o *Asahome* é um alimentador automático para cães e gatos de pequeno porte, seu reservatório tem capacidade de 2 kg. À liberação da ração é feita através da gravidade, conforme ilustra a figura 1.4.

Figura 1.4 – Alimentador automático Asahome

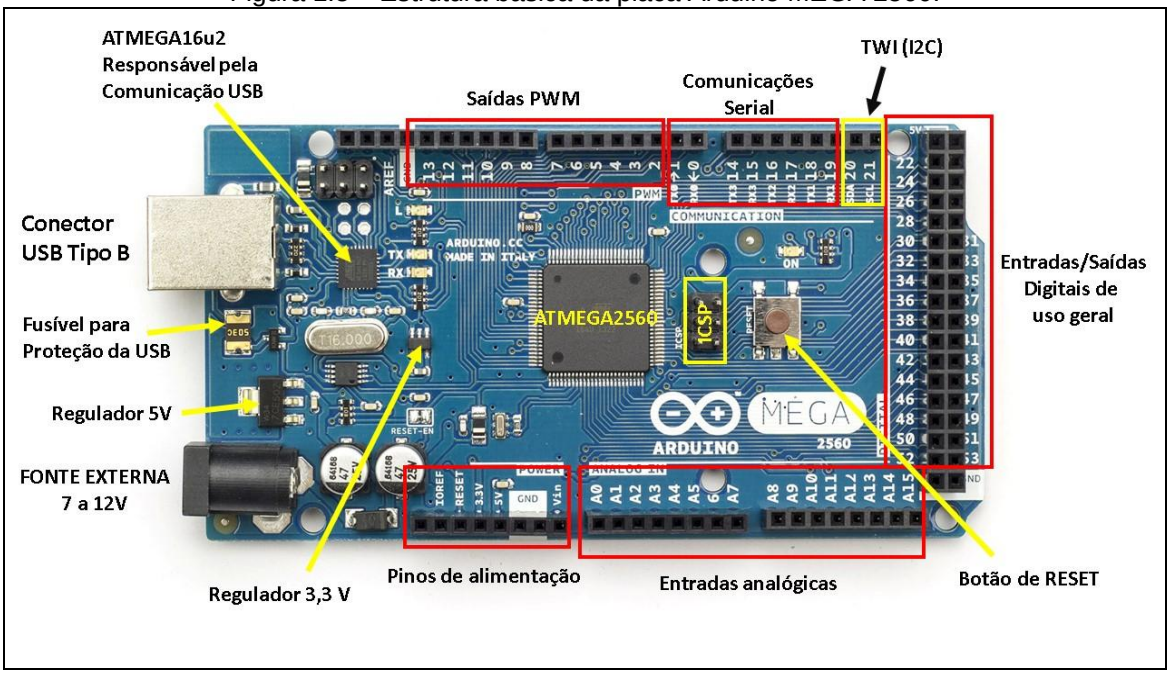
Fonte: www.megatnt.com.br, 2016

1.5 Controlador Arduino

Lemos (2015) relata que o Arduino é uma plataforma dedicada para prototipagem eletrônica baseada em código aberto. Anteriormente as plataformas existentes eram de alto valor (em US$) e possuíam um alto nível de complexidade, quando comparadas as necessidades dos projetos desenvolvidos.

O *site* Arduino (2016) destaca que o cérebro da interface é um microcontrolador da marca Atmel modelo 2560, porém pode variar de acordo com o modelo de Arduino. Esse microcontrolador é acoplado a um circuito que possui as entradas e saídas para a alimentação, comunicação com os periféricos e programação do sistema. A figura 1.5 ilustra a placa MEGA 2560.

Figura 1.5 – Estrutura básica da placa Arduino MEGA 2560.



Fonte: www.bodgarage.repofy.com, 2015

De acordo com Machado (2011), o microcontrolador da placa Arduino trabalha com a tensão de operação em 5 Vcc. O sistema possui faixas de tensões para a alimentação, sendo que a recomendada é entre 7 Vcc a 12 Vcc, o limite para a variação da tensão de entrada é entre 6 Vcc a 20 Vcc. Tensões fora destes limites não garantem a perfeita operação da plataforma.

Para adequar a tensão de entrada para os 5 Vcc requeridos pelo microcontrolador, a placa Arduino possui um circuito integrado que reduz a tensão de entrada para 5 Vcc, sendo possível alimentá-la pela porta *USB* do computador, utilizada para programar o microcontrolador. Para a segurança da comunicação do sistema Arduino com o computador, a placa possui um circuito integrado responsável por impedir que tensões externas superiores a 5 Vcc retornem pelo cabo *USB*. Outro componente importante é um botão de reinicialização do sistema.

O *site* Arduino (2016) relata que a placa de prototipagem eletrônica possui um sistema de portas de comunicação *(I/O)*, sendo elas analógicas e digitais. Portas são pontos físicos nos quais são feitas conexões para entrada e saída de dados, por exemplo, um sensor que envia um sinal para a porta de entrada *(I)* e após a leitura desse sinal é acionada uma porta de saída *(O)* para ligar um *LED*. As portas digitais podem ser utilizadas tanto para a entrada de dados como para saída, nesse caso a escolha é feita na programação. A corrente máxima de trabalho nas portas são de 20 mA.

De acordo com Gregory (2014), as portas analógicas são responsáveis por ler sinais a partir de sua grandeza em volts (V) ou Ampères (A). Por exemplo, um sensor de temperatura que a cada medida é gerada uma tensão referente à temperatura captada. O microcontrolador do sistema Arduino trabalha apenas com sinais digitais 0 ou 1, seu nível lógico baixo é de 0 Vcc e o alto de 5 Vcc, neste caso é necessário converter o sinal analógico em digital, para isso a placa possui um conversor analógico digital (A/D). A função deste conversor é quantificar o valor analógico em *bits* que representem esse valor.

Periféricos, como o *Shields,* podem ser integrados à placa, conectados pelas portas *I/O*. Esses periféricos podem ser *display LCD*, interface *WI-FI*, *Bluetooth*, teclado e relés.

O site Arduino (2016) destaca que para programação de seu sistema é disponibilizado um ambiente integrado de desenvolvimento, onde o usuário cria o programa lógico. A programação é feita na linguagem C/C++ com algumas modificações especificadas pelo fabricante. Possui um compilador que transforma o

programa escrito em linguagem nível usuário, neste caso C/C++, em linguagem de máquina, criando um arquivo que é enviado ao microcontrolador da placa Arduino. Para a comunicação entre o computador e a interface Arduino é necessário um *driver* de comunicação, disponibilizado pelo fabricante.

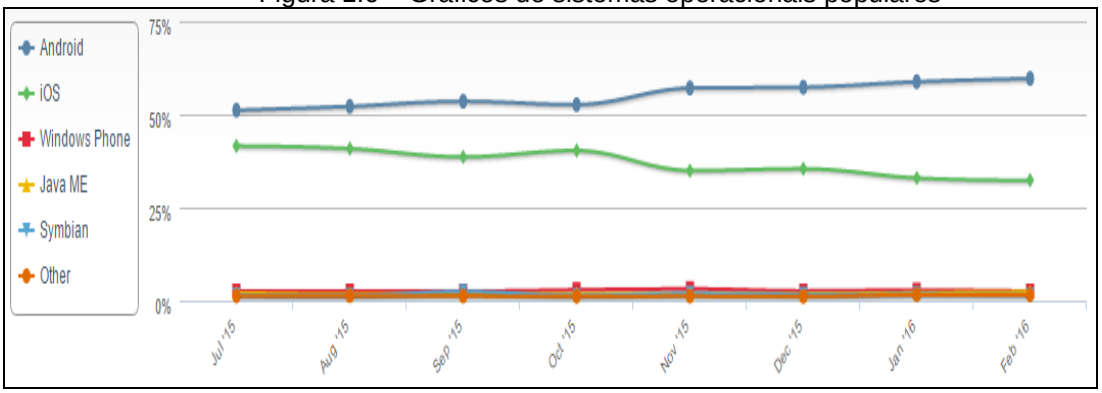
1.6 Dispositivos móveis

Augusto (2011) aponta que a cada dia as pessoas revelam um interesse maior pelo acesso a informações com mobilidade, através de dispositivos como *smartphones*, *tablets* e *notebooks*. Com o crescimento da utilização de dispositivos móveis, a internet móvel cresceu de modo rápido, promovendo assim a qualidade do acesso às informações *online.*

Brigatto (2015) informa que o número de usuários de *smartphones* no Brasil cresceu em 48% no 3º semestre de 2015 em comparação com o mesmo período no ano de 2014, mostrando a importância do desenvolvimento de aplicações para dispositivos.

Segundo o *site* Netmarketshare (2015), os *smartphones* possuem uma plataforma que é um sistema operacional, sendo diversos os modelos, de acordo com os fabricantes. Os sistemas operacionais mais comuns em celulares são: Android, que é mantido pelo *Google*, o *IOS,* que é mantido pela *Apple* e *Windows Phone,* que é mantido pela Microsoft. Os dados estatísticos sobre sistemas operacionais estão na figura 1.6.

Figura 1.6 – Gráficos de sistemas operacionais populares



Fonte: www.netmarketshare.com, 2016

Elgin (2005) relata que o sistema operacional Android é desenvolvido para dispositivos móveis. Ele possui núcleo baseado em Linux e tem sua própria interface, é uma máquina virtual que roda sobre o *kernel* do sistema Linux. O fato de o núcleo ser em Linux faz com que ele seja o responsável pelo gerenciamento da memória, processos, segurança e gerenciamento das redes do dispositivo e dos *drivers*.

De acordo com Lecheta (2009) a arquitetura do sistema Android é composta por quatro camadas, sendo elas:

Aplicações**:** é a camada a que o usuário comum tem acesso, onde ficam as aplicações como telefone, navegador e jogos;

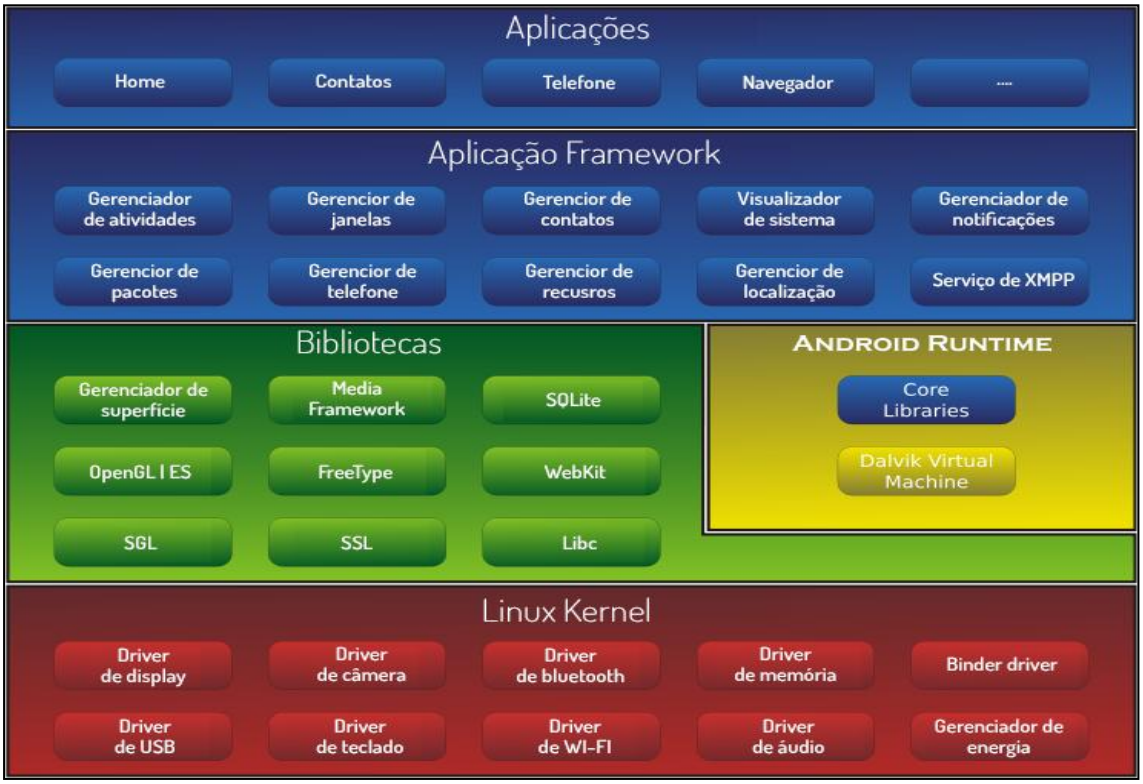
Bibliotecas**:** é a camada onde ficam armazenadas as bibliotecas do sistema. São arquivos que contém instruções específicas necessárias para o funcionamento do sistema ou de determinada função, como bibliotecas para abrir um vídeo, uso da câmera fotográfica ou abrir um banco de dados. Essas bibliotecas, normalmente, são programadas em linguagem C/C++;

Android Runtime: é uma máquina virtual que roda por meio do sistema Linux. Para que os aplicativos possam ser executados, eles utilizam uma máquina virtual dentro do Android;

Kernel Linux: é o núcleo do sistema, onde tudo é controlado. Ele é responsável por enviar os dados para o processador do dispositivo.

A figura 1.7 ilustra as camadas do sistema Android.

Figura 1.7 – Camadas do sistema Android



Fonte: www.javamanbr.com, 2016

Lecheta (2009) informa que as aplicações para sistema Android são desenvolvidas em linguagem Java. Para o desenvolvimento das aplicações existem diversos ambientes de desenvolvimento integrado.

O Google disponibiliza gratuitamente o *kit* de desenvolvimento Android *SDk,* que é um pacote composto pelo ambiente de desenvolvimento Android Studio*,* bibliotecas, sistemas Android em diversas versões e um simulador, onde o desenvolvedor pode simular o aplicativo antes do envio para um dispositivo móvel. É possível desenvolver 100% do aplicativo como *layout* da tela, programação, contato com o kernel e com *drivers*. Também é possível utilizar pré-telas fornecidas.

A parte artística é desenvolvida em XML que é uma linguagem de marcação, onde é definido o posicionamento dos itens na tela, tamanho dos objetos, cores, entre outros.

1.7 *Wi-Fi/Wireless*

Segundo o *site* Infowester (2016), a tecnologia *Wi-Fi* ou *wireless* tem a função de permitir que dispositivos eletrônicos se comuniquem sem a utilização de fios. A formação dessa rede é baseada no padrão internacional IEEE 802 e para que haja uma comunicação, é necessário que sejam geradas ondas de rádio através de um dispositivo roteador, que irá trocar sinais de comunicação com o módulo *Wi-Fi* dos dispositivos eletrônicos conectados a rede local.

Segundo o *site* Tecmundo (2016), a distância de alcance da rede *Wi-Fi* pode chegar até 100 metros, em áreas internas e 300 metros para áreas externas, porém seu funcionamento depende do roteador e da antena que está sendo utilizada.

1.8 Modelagem de dados

Segundo o *site* Regilan (2016), o desenvolvimento de uma estrutura para armazenamento eletrônico de informações, pode ser chamado de modelagem de dados. Essas informações devem ser elaboradas de forma ordenada e dividida por classes, formando assim um banco de dados. Sua função principal é garantir ao usuário uma forma eficiente de armazenamento e consulta de informações, podendo exportá-las para diversas aplicações, como: gráficos, relatórios, formulários, etc.
Regilan, também afirma que um banco de dados informatizado deve ser desenvolvido em um *software* SGBD (Sistema Gerenciador de Banco de Dados), onde é possível visualizar graficamente as informações armazenadas em forma de tabela, desenvolver operações algébricas ou lógicas e criar restrições de acesso.

1.9 Internet das coisas

Segundo o *site* Sebrae (2016), o tema internet das coisas é algo em crescimento no mundo todo, esse conceito surgiu com o objetivo de conectar diversos equipamentos ou objetos, que são utilizados no dia-a-dia, à rede da internet. Essa tecnologia irá aperfeiçoar o tempo de atividades domésticas tornando

a vida das pessoas mais fácil. Estão sendo desenvolvidos estudos com conectividade via *wireless* ou *RFID* (Identificação Rádio Frequência), e espera-se que o aumento de objetos inteligentes ligados à rede, aumente nos próximos anos.

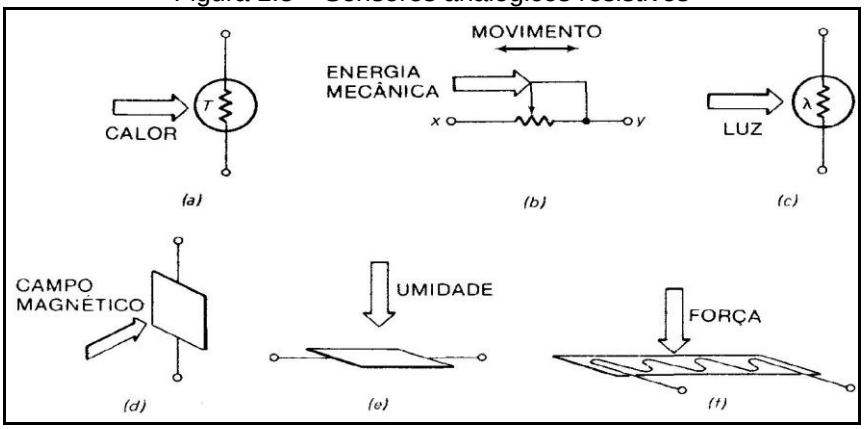
1.10 Sensores

Thomazzini (2007) relata que sensores são dispositivos responsáveis por detectar formas de energia presentes em um processo ou ambiente, podendo ser ela térmica, cinética e luminosa. Cada sensor é utilizado para a medição de uma grandeza especifica. Esses dispositivos se classificam em dois grandes grupos, os sensores analógicos e os digitais.

Sensores analógicos: são aqueles que apresentam diferentes níveis de tensão, conforme ocorrem as mudanças físicas no ambiente que está sendo mensurado. Essa variação ocorre ao longo do tempo, conforme a faixa de atuação especificada pelo fabricante, como sensores de pressão, temperatura, umidade, vazão e luminosidade.

No grupo dos analógicos têm-se os sensores resistivos que são aqueles onde os circuitos comportam-se como resistores, mas devido a certas propriedades físicas ou químicas, variam o valor de sua resistência de acordo com certas características, como luminosidade ou temperatura. A figura 1.8 ilustra os sensores analógicos resistivos.

Figura 1.8 – Sensores analógicos resistivos

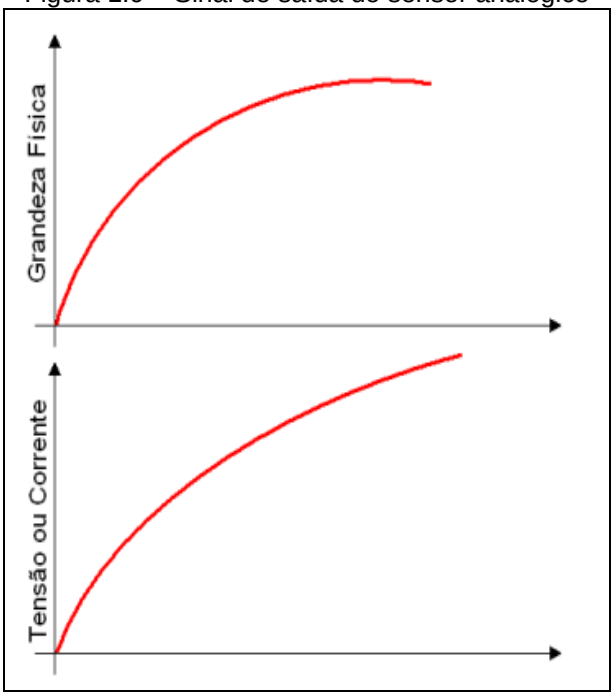


Fonte: www.newtoncbraga.com.br, 2016

Existem também os sensores capacitivos, que atuam analogicamente, pois conforme a alteração de uma determinada condição do processo ou do ambiente em que o sensor está instalado ocorre uma variação na capacitância do componente, por exemplo, um sensor régua de nível d'água instalado na parte externa de um tanque translúcido, nota-se durante seu funcionamento que conforme a água escoa do tanque, a superfície do sensor sente a variação da capacitância através da diminuição do nível da água em contato com o vidro do tanque, ocasionando a mudança da tensão ou da frequência do sinal de saída do sensor (BRAGA, 2016).

De acordo com Thomazzini (2007) a saída de um sensor analógico permite que seus valores se assemelhem à variação da grandeza física avaliada, sendo a saída medida em tensão ou corrente. A figura 1.9 ilustra o sinal de saída do sensor analógico.

Figura 1.9 – Sinal de saída do sensor analógico



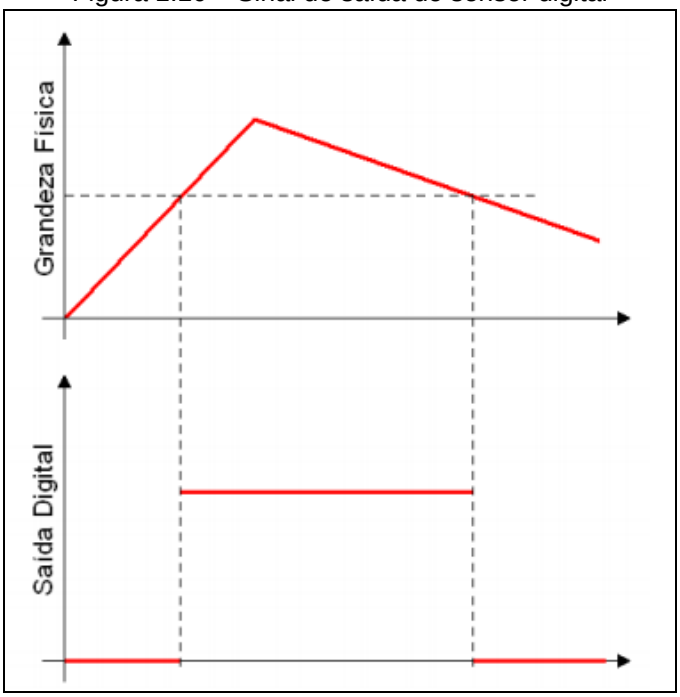
Fonte: ALBERTO, 2016, p.8

Sensores digitais: apresentam apenas dois sinais, ligado ou desligado, nível lógico 0 ou 1. Esse tipo de sensor monitora o estado e posição de um determinado material ou situação, e em caso de alteração o sinal de saída passa de 0 para 1.

Ainda, Thomazzini (2007) afirma que o sensor digital não possuiu nível valor intermediário entre 0 e 1, ao contrário do sensor analógico. São mais utilizados para detecção de materiais e em *encoders* para determinação de distância.

O fato de o sensor digital trabalhar com níveis lógicos 0 e 1 faz com que ele só consiga determinar seu estado após atingir um valor predeterminado, ou seja, ele precisa de um valor mínimo para ativação a partir da grandeza física que está sendo verificada. A figura 1.10 ilustra o sinal de saída do sensor digital.

Figura 1.10 – Sinal de saída do sensor digital



Fonte: ALBERTO, 2016, p.10

1.11 Válvulas

Palhares (2015) relata que as válvulas são um dos instrumentos mais utilizados nas indústrias com a função de permitir ou não a passagem do fluido e, ainda, dosar a quantidade de fluido.

É comum se encontrar válvulas em torneiras de cozinha, no quintal, em filtros, registro de gavetas, em tanques industriais, entre outros. O controle de abertura e

fechamento é muito importante e útil. As válvulas são classificadas da seguinte forma:

- Manual**:** o próprio operador realiza a operação da abertura e fechamento;

- Autorreguladora: o controle é realizado pelo fluido através de sua energia contida;

- Controle: a operação é feita através de uma força auxiliar para o acionamento, de acordo com os sinais dos controladores pneumáticos, elétricos ou eletropneumáticos.

Controlando a posição da válvula se controla a vazão do fluido que passa na tubulação. De acordo com seu estado aberta ou fechada, a válvula permite que um CLP ou computador a monitore. A figura 1.11 ilustra uma válvula controle.

Figura 1.11 – Válvula controle

Fonte: www.alfamatec.com.br, 2016

2 METODOLOGIA

Neste capítulo encontra-se a trajetória percorrida para o desenvolvimento do trabalho intitulado Alimentador Automatizado para Cães. Trata-se de pesquisa experimental, visando à construção de um protótipo. É desenvolvida nas dependências da FATEC SBC e nas residências dos integrantes do grupo.

Dentre os vários autores que tratam da metodologia científica, Severino (2007) relata que a metodologia desperta no pesquisador a necessidade de buscar compreensões a respeito do fato. Ela mostra métodos e técnicas, que são procedimentos de organização e preparação de etapas para construção de trabalhos científicos.

- determinação do tema-problema e justificativa;
- levantamento bibliográfico referente ao tema;
- leitura dessa bibliografia após a seleção;
- construção lógica do trabalho;
- redação do texto e considerações finais.

A redação do texto tem como base as normas da ABNT que se encontram resumida no Manual de Normalização de Projeto de Trabalho de Graduação da Fatec São Bernardo do Campo (2016). Há recursos ilustrativos que explicam e complementam o texto com figuras e tabelas.

2.1 O tema-problema e justificativa

Em sala de aula realizou-se um *brainstorm* com os integrantes do grupo para definir o tema-problema, onde se constatou a falta de opção de um alimentador inteligente para alimentar o cão enquanto o dono está ausente, e que possibilite a monitoração da situação à distância, acompanhando se algum mantimento do alimentador esta acabando. Em seguida foi definido o tema Alimentador

Automatizado para Cães, cujo objetivo é construir um alimentador que forneça ração e água conforme programação pré-estabelecida através de um *smartphone*. A proposta se justifica porque o alimentador visa proporcionar ao animal melhor bem-estar em relação à alimentação, assim como tranquilidade para seu dono.

Após definir o tema o grupo fez uma reunião com o orientador, onde o mesmo solicitou uma pesquisa de mercado com o objetivo de conhecer melhor os produtos que são oferecidos atualmente no mercado nacional. Constatou-se a necessidade de construir um alimentador mais inteligente, que não só alimenta o cão, mas que forneça dados de como está o reservatório de ração e água. Todas estas informações disponíveis em um dispositivo *smartphone*.

A partir deste ponto realizou-se uma análise FOFA (Forças, Oportunidades, Fraquezas e Ameaças) para a viabilização da especificação técnica do projeto.

Forças: alimentadores automatizados são fabricados aos montes por diversas empresas, mas nenhum possui tecnologia com informações do alimentador disponíveis no *smartphone*. Como dito na introdução, os donos de cães passam mais tempo trabalhando e estudando do que em casa com seu cão, sem contar as viagens.

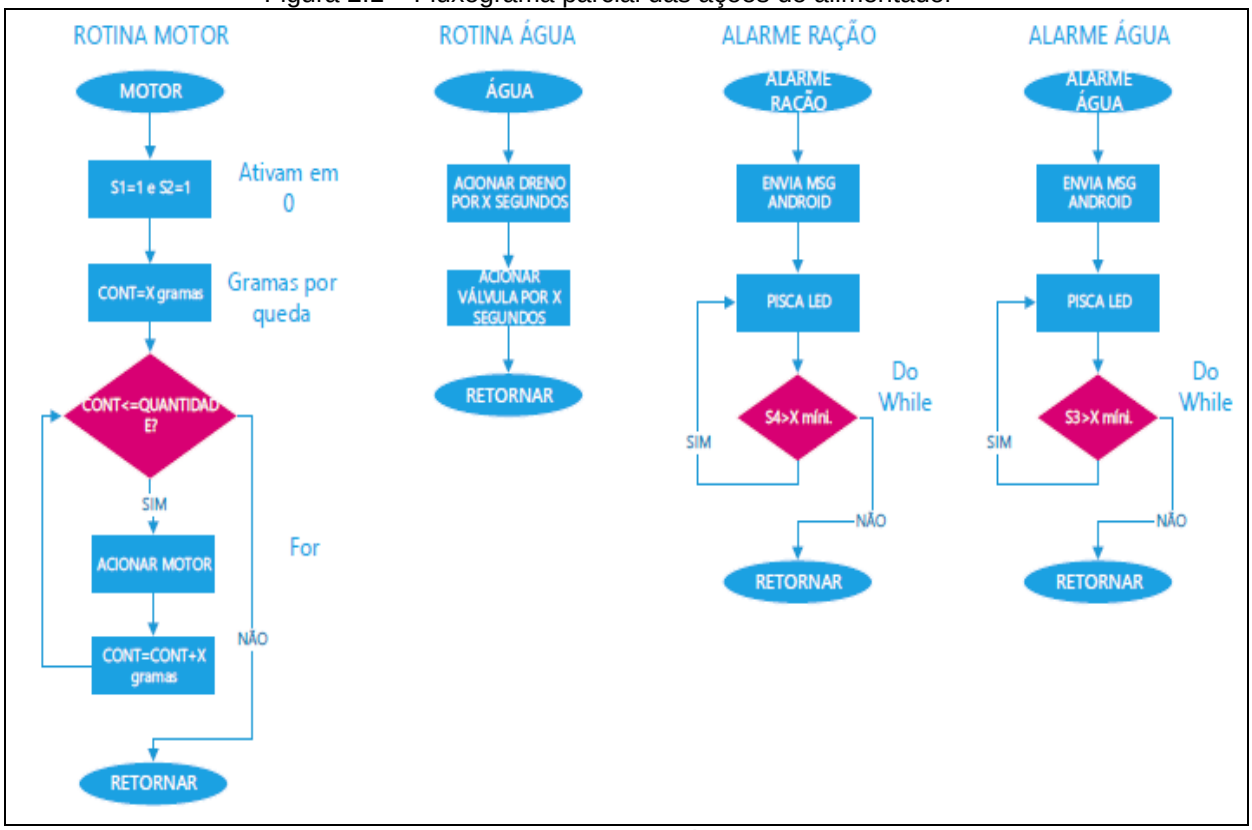
Oportunidades: o alimentador visa alcançar um público que passa grande parte do dia e noite fora do seu lar, que viagem a trabalho deixando seus cães sozinhos e na pesquisa de mercado constou-se que o mercado de *pet* está em crescimento.

Fraquezas e ameaças: o alimentador pode vir a ter um custo elevado, não pode ser utilizado para cães gigantes, e o usuário precisa ter um *smartphone* compatível com a tecnologia do aplicativo.

2.2 Objetivo geral do projeto

Com o resultado das pesquisas e análises o grupo delimitou o objetivo geral do projeto, e concluiu-se que, o alimentador será construído para fornecer ração e água para cães de pequeno porte. Com uma tecnologia inovadora o protótipo será construído de forma a permitir que o usuário alimente seu cão de qualquer lugar e horário. Todo o sistema de fornecimento será automático, visando à praticidade ao dono e o bem estar ao animal. O diferencial deste protótipo é um aplicativo que controla o fornecimento de ração e água, remotamente, e visualiza alerta de nível baixo nos reservatórios de ração e água. A figura 2.1 ilustra como se pretende alcançar a funcionalidade deste objetivo.

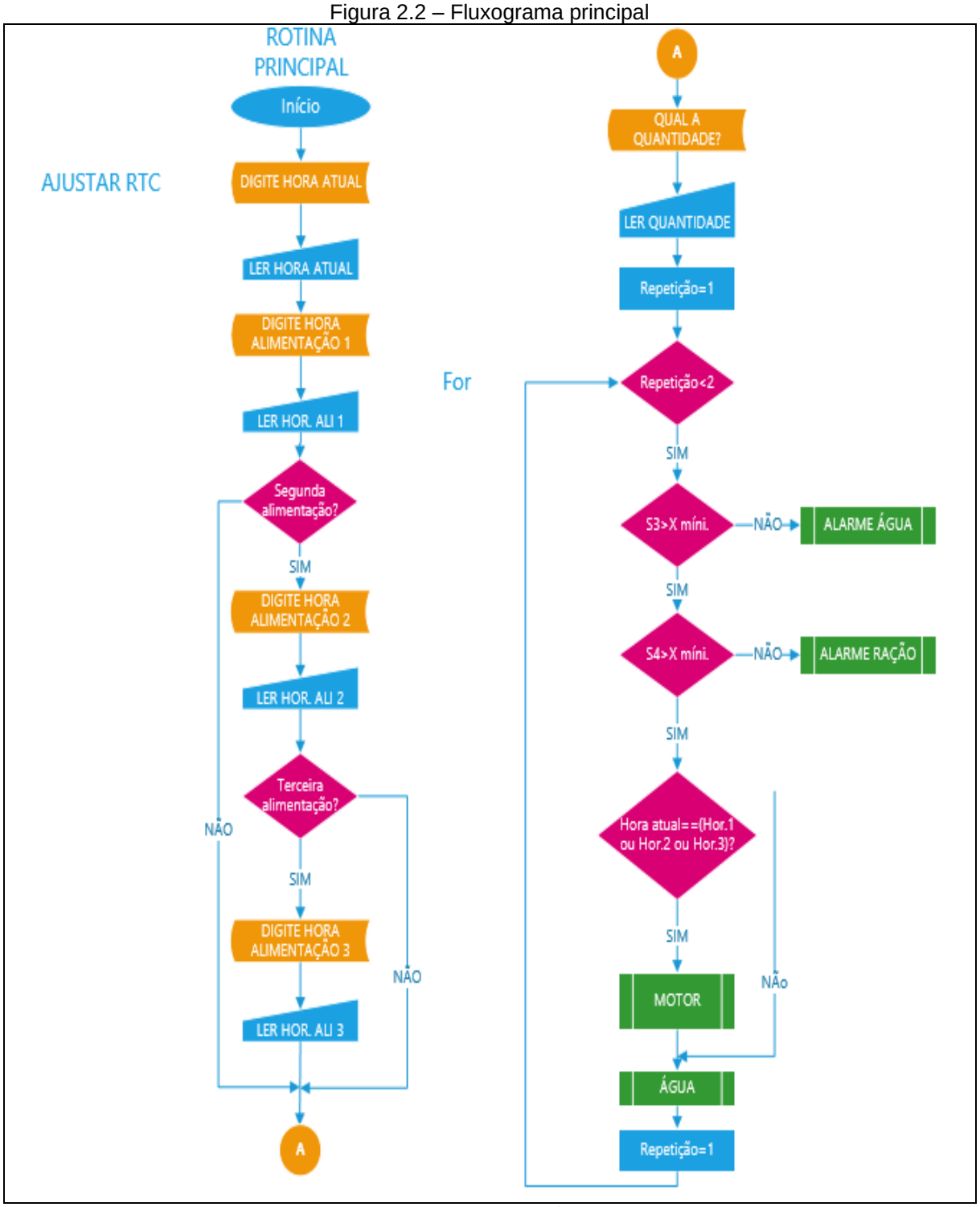
Figura 2.1 – Fluxograma parcial das ações do alimentador



Fonte: Autoria própria, 2016.

Após o entendimento das ações parciais do alimentador, é desenvolvido um fluxograma principal com todas as rotinas integradas, visando especificar as principais funções do controlador e do *software* aplicativo. A figura 2.2 ilustra o fluxograma principal do projeto.

Figura 2.2 – Fluxograma principal



Fonte: Autoria própria, 2016.

2.3 Planejamento da execução

Após os integrantes do grupo e o orientador definirem o tema-problema a ser executado e o objetivo do projeto, foi realizado um planejamento com o objetivo de dividir e desenvolver o projeto em partes. É importante destacar que se seguiu a metodologia PDCA (*Plan, Do, Check e Action* ou Planejar, Executar, Testar e Agir). Deu-se inicio às seguintes etapas:

Primeira etapa: reunião do grupo com o orientador para traçar diretrizes e cronogramas de como efetuar as pesquisas teóricas e sua seleção para construção do projeto. Orientador se colocou à disposição para atender quando solicitado e, marcou obrigatoriamente, um dia por semana para se apresentar o que foi realizado.

Segunda etapa: o levantamento bibliográfico deu-se na biblioteca da FATEC São Bernardo do Campo, pesquisas em *sites* especializados, em bibliotecas virtuais, manuais e catálogos de fabricantes.

Terceira etapa: após leitura e releitura da bibliografia consultada fez-se uma seleção das mesmas que compõem o capítulo 1 – Fundamentação Teórica.

Quarta etapa: estudo da viabilidade econômica dos materiais. Pesquisas em lojas e *sites* especializados para a aquisição dos materiais, conforme tabela 2.1.

Tabela 2.1 – Lista de materiais para construção do projeto

TAG	DESCRIÇÃO	VALOR/REAIS	QTD
RES1	RESERVATÓRIO DE RAÇÃO COM CAPACIDADE DE 1,5 LITROS	50,00	1
M1	MOTOR DC 12V	50,00	1
S1/S2	SENSOR ULTRASSÔNICO	54,00	2
U1	CONTROLADOR ARDUINO MEGA COM DISPLAY	250,00	1
RES2	RESERVATÓRIO DE ÁGUA COM CAPACIDADE DE 9 LITROS E SUPORTE	40,00	1
V1	SERVO MOTOR COM VÁLVULA ESFÉRICA	50,00	1
P1	PLATAFORMA DE FIXAÇÃO DOS RESERVATÓRIOS	150,00	1
PR1/PR2	PRATO DE RAÇÃO OU ÁGUA	14,00	2
FT1	CONVERSSOR DE TENSÃO 220/127Vca PARA 24Vcc	15,00	1
FT2	CONVERSSOR DE TENSÃO 220/127Vca PARA 12Vcc	15,00	1
FT2	CONVERSSOR DE TENSÃO 220/127Vca PARA 5Vcc	15,00	1
B1	BOMBA DRENO 12V	25,00	1
ACESSÓRIOS	FIOS, TERMINAIS, PARAFUSOS.	60,00	1
	VALOR TOTAL GASTO	788,00	

Fonte: Autoria própria, 2016

Quinta etapa: limites de engenharia e especificações técnicas para início do desenvolvimento das partes que compõem o protótipo (*hardware* e *software*).

Sexta etapa: montagens e desenvolvimentos das partes que compõem o protótipo - (*hardware*).

Sétima etapa: elaboração da programação do microcontrolador e dos aplicativos no *smartphone* e no servidor *WEB - (software)*.

Oitava etapa: testes de desempenho visando comparar o funcionamento das partes desenvolvidas com as especificações técnicas. Esses testes ocorreram de forma concomitante às etapas de desenvolvimento.

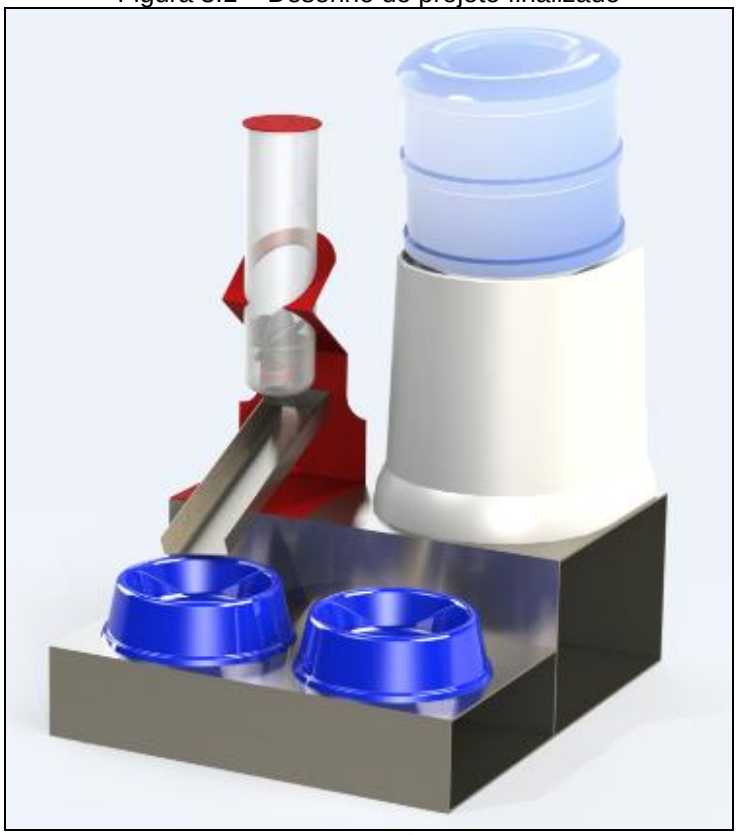
Nona etapa: Integração das partes desenvolvidas e testes finais do sistema integrado com análise dos resultados obtidos.

Décima etapa: após a conclusão do desenvolvimento do projeto faz-se as considerações finais e o resumo.

3 DESENVOLVIMENTO DE PROJETO

Neste capítulo encontra-se o passo a passo do desenvolvimento do protótipo intitulado Alimentador Automatizado para Cães. Para melhor visualização do projeto a figura 3.1 ilustra o desenho do trabalho finalizado

Figura 3.1 – Desenho do projeto finalizado



Fonte: Autoria própria, 2016

O desenvolvimento do projeto tem como alicerce as seguintes etapas principais:

- descrição dos materiais para construção do projeto;
- especificação técnica e limites de engenharia do projeto;
- desenvolvimento, montagem e testes;
- montagem e testes finais de todo sistema do alimentador;
- obstáculos e soluções.

3.1 Descrição dos materiais para construção do protótipo

Para o desenvolvimento e construção do projeto são descritos alguns materiais que são primordiais.

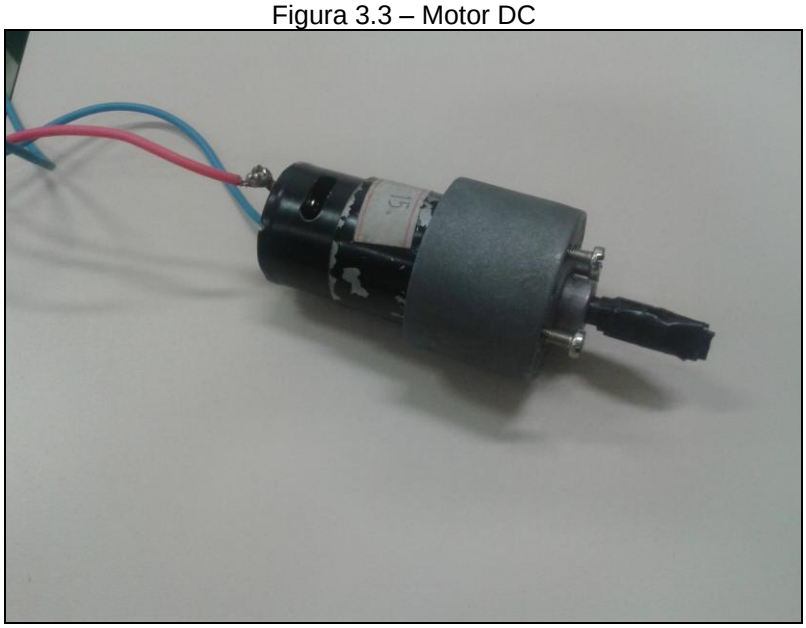
Reservatório de ração (RES1): para armazenar a ração é utilizado um *dispenser* de cereais de 1,5 litros. Este *dispenser* é adaptado para armazenar ração em grãos, com um motor DC e sensores serão realizados a automatização do mesmo. A figura 3.2 ilustra o reservatório de ração.

Figura 3.2 – Reservatório de ração

Fonte: Autoria própria, 2016.

Motor DC (M1): para automatizar o processo de entrega da ração ao recipiente do cão, é utilizado um motor DC de 12 V. Este motor é adaptado no eixo do *dispenser*, quando solicitado, o motor entra em rotação liberando uma determinada quantidade de ração. A figura 3.3 ilustra o motor DC.

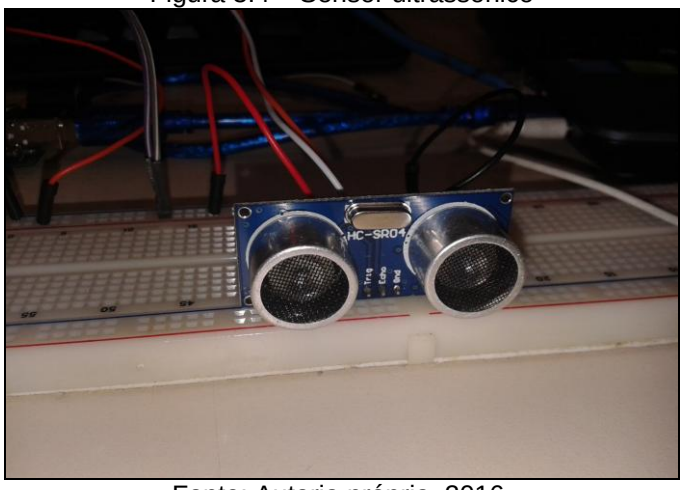
Figura 3.3 – Motor DC



Fonte: Autoria própria, 2016.

Sensor ultrassônico (S1/S2): para detectar o nível de ração e água nos reservatórios, são utilizados sensores ultrassônicos. Estes sensores vão informar para o controlador a quantidade de ração e água existente. No programa são geradas rotinas que emitem alarmes no aplicativo instalado no *smartphone* do usuário. A figura 3.4 ilustra o sensor ultrassônico.

Figura 3.4 – Sensor ultrassônico

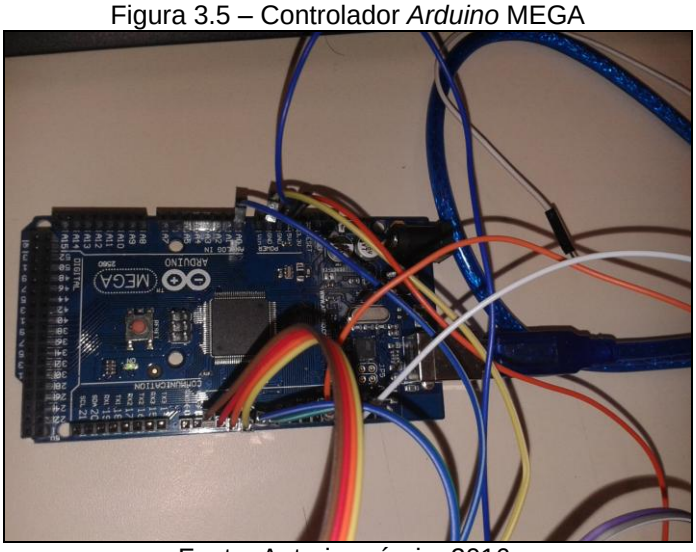


Fonte: Autoria própria, 2016.

Controlador *Arduino* MEGA (U1): para realizar a automação dos periféricos do alimentador com o aplicativo, é utilizado um controlador *Arduino* MEGA. Para integrar os sensores, motor, válvula e dreno ao aplicativo instalado no *smartphone* do usuário, foram desenvolvidas programações que são responsáveis pelo

acionamento da ração e água para o cão. A figura 3.5 ilustra o controlador *Arduino* MEGA.

Figura 3.5 – Controlador *Arduino* MEGA



Fonte: Autoria própria, 2016.

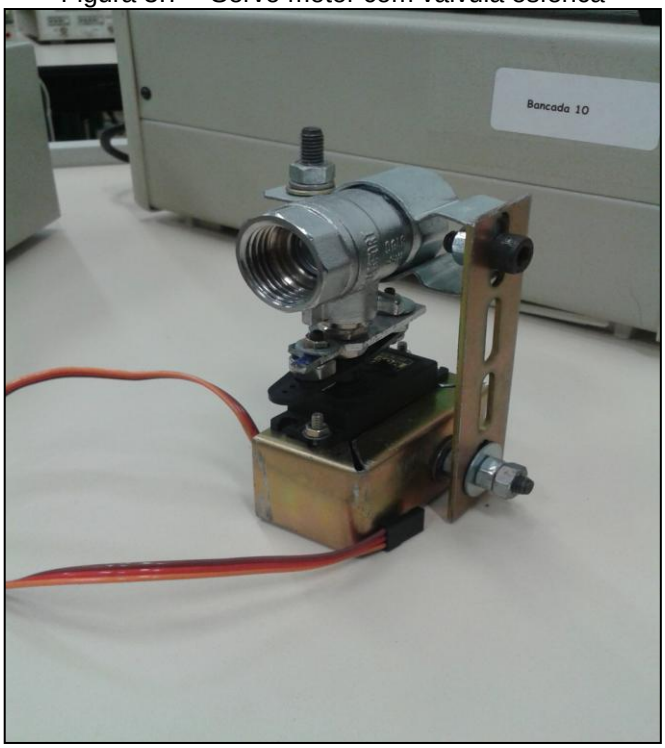
Reservatório de água (RES2): para armazenar a água, é utilizado um galão convencional com a capacidade de 9 litros, e um suporte para a sua fixação. Para automatizar este processo de entrega de água para o cão, é instalada uma válvula na saída do reservatório, para fazer a liberação da água conforme solicitado pela programação. Um sensor ultrassônico é responsável por medir a quantidade de água no reservatório. A figura 3.6 ilustra o reservatório de água.

Figura 3.6 – Reservatório de água

Fonte: Autoria própria, 2016.

Servo motor com válvula esférica (V1): para automatizar o processo de liberação da água ao recipiente do cão, é utilizado um servo motor DC (*direct current* – corrente contínua) de 5 V com uma válvula esférica acoplada no eixo, este componente foi gentilmente cedido pela empresa Embratel. A válvula é acoplada na saída do reservatório de água, quando acionada pelo controlador, a válvula abre e libera o fluxo de água. A figura 3.7 ilustra o servo motor com válvula esférica.

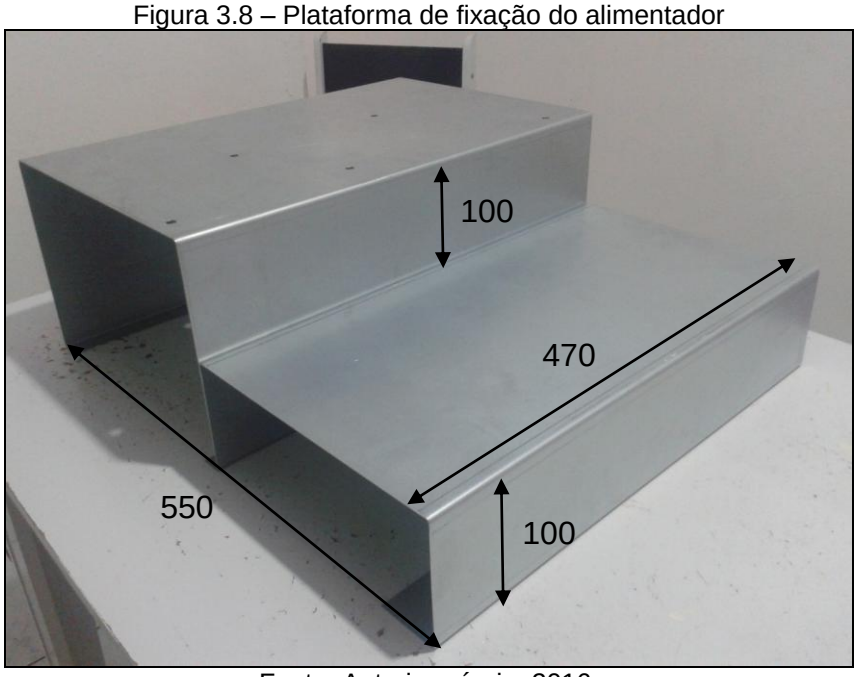
Figura 3.7 – Servo motor com válvula esférica



Fonte: Autoria própria, 2016.

Plataforma de fixação do alimentador (P1): para colocar os reservatórios e os pratos de ração e água do cão, é desenvolvida uma plataforma de chapa de aço carbono com o tratamento da superfície galvanizada. Esta plataforma foi gentilmente cedida e produzida pela empresa Novemp Indústria e Comercio Ltda. A figura 3.8 ilustra a plataforma de fixação do alimentador.

Figura 3.8 – Plataforma de fixação do alimentador



Fonte: Autoria própria, 2016.

Pratos de ração e água (PR1/PR2): para colocar a ração em grãos e água do cão, foram utilizados dois pratos. Estes pratos são responsáveis para receberem a ração e a água, no prato de água é adicionado um sistema de drenagem da água. A figura 3.9 ilustra os pratos de ração e água.

Figura 3.9 – Pratos de ração e água

Fonte: Autoria própria, 2016.

Bomba dreno (B1): para fazer a drenagem da água no prato, é utilizada uma bomba 12 V de para-brisa. Esta bomba é responsável por drenar a água que fica no prato, por muito tempo sem que o cão tome. A figura 3.10 ilustra a bomba dreno.

Figura 3.10 – Bomba dreno

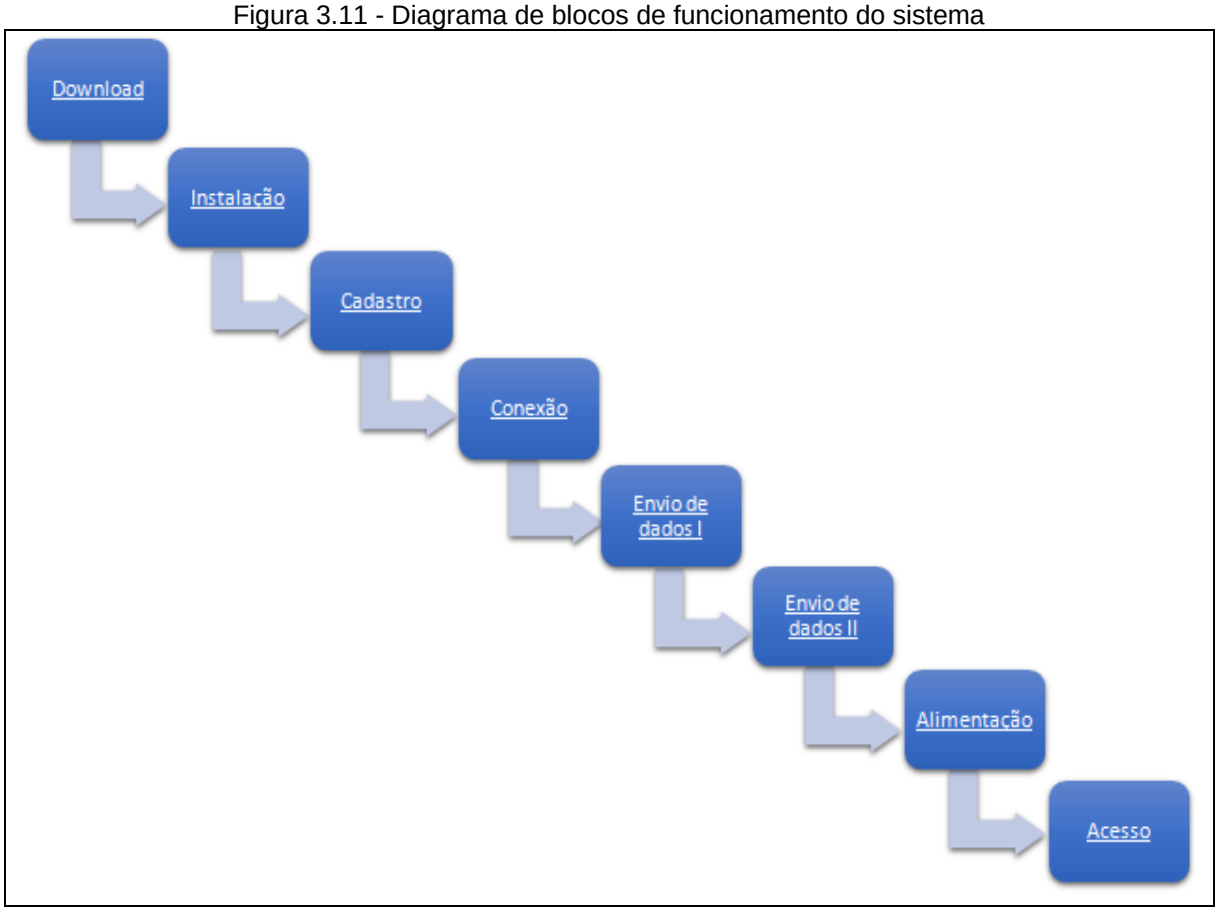
Fonte: Autoria própria, 2016.

3.2 Especificação técnica e limites de engenharia do protótipo

Após a definição dos materiais utilizados para a montagem definiu-se que, o protótipo a será construído para fornecer ração e água para cães de pequeno porte. Basicamente com o auxilio de: sensores ultrassônicos, motor DC 12 V, válvula esférica, bomba dreno, aplicativo e um controlador Arduino MEGA todo o sistema de fornecimento é automático, basta conectar o alimentador a rede 127 Vca e enviar os dados do cão, através do aplicativo, via Wi-Fi.

Para melhor visualização e esclarecimento a figura 3.11 ilustra o diagrama de blocos de funcionamento do sistema do alimentador, após a figura, estão às etapas do diagrama contendo suas explicações.

Figura 3.11 - Diagrama de blocos de funcionamento do sistema



Fonte: Autoria própria, 2016.

Download: o usuário baixa o aplicativo que está disponível na loja virtual *playstore* pelo nome de *smartdog*.

Instalação: o usuário faz uma inspeção visual no alimentador e abastece os reservatórios, em seguida, energiza o alimentador em uma tomada 127 Vca.

Cadastro: o usuário acessa o aplicativo e efetua o cadastro dos dados do cão e informa a porção em quilogramas (kg) e horários de fornecimento.

Conexão: o *smartphone* e o alimentador conectam-se a internet para realizar a transmissão de dados.

Envio de dados I: os dados cadastrados pelo usuário são enviados para um servidor *on-line.*

Envio de dados II: o servidor envia os dados recebidos do aplicativo para o controlador.

Alimentação: o alimentador fornece ração e água conforme a programação recebida do aplicativo.

Acesso: pelo aplicativo o usuário consegue verificar os níveis de ração e água, e quando estes níveis alcançarem o limite mínimo um alarme é enviado para o *smartphone.*

Limites de engenharia do projeto: o protótipo é construído para cães de pequeno porte. O reservatório de ração pode comportar 500 g de ração em grãos, suficientes para 5 refeições de 100 g. Através de dados inseridos no aplicativo é determinado a quantidade e os períodos que a ração é fornecida. O reservatório de água comporta 9 litros, o prato suporta 500 ml, a cada 6 horas o dreno será acionado para fazer a troca da água, então a cada 24 horas é utilizada 2 litros de água, com esta relação conclui-se que o reservatório de água terá autonomia de 4 dias e meio sem que seja preciso reabastecê-lo. O alimentador possui as dimensões 470x673x550mm (LxAxP) e pesa 15 kg com os reservatórios vazios, mas com os reservatórios a 100% da capacidade, o alimentador pesa 24 kg, para o funcionamento é necessário conectar a uma tomada 127 Vca.

3.3 Desenvolvimento, montagens e testes.

Para o desenvolvimento do protótipo é necessário realizar as montagens parciais de cada operação do alimentador. Durante o desenvolvimento das partes, foram realizados os testes para verificar a compatibilidade com a especificação técnica e os limites de engenharia estabelecidos. A seguir são expostas, passo a passo, as fases de cada montagem parcial e seus respectivos testes.

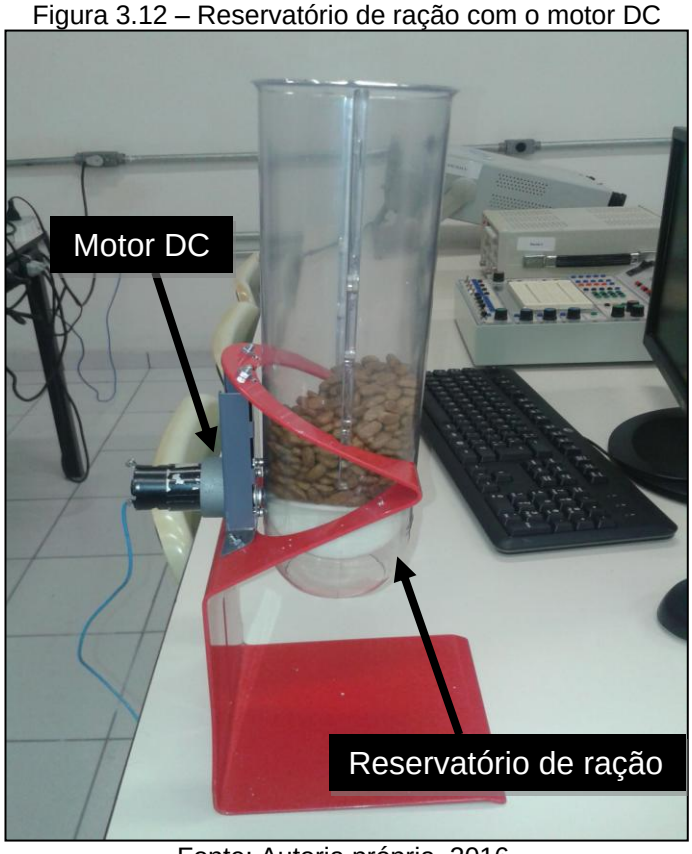
3.3.1 Reservatório de ração com motor DC acoplado

Para que seja fornecida a ração nos horários programados, é necessária a instalação de um motor DC 12 V no reservatório, este motor é responsável por fornecer a ração, quando acionado pela programação. Este motor é escolhido pelo fato de ser fácil o seu controle pelo *Arduino*, e também pelo seu baixo custo. Neste processo são seguidos os seguintes procedimentos:

- Com uma chave de fenda é retirado o eixo manual do *dispenser;*
- Após a retirada do reservatório, o eixo manual é furado com uma furadeira, para receber o motor DC 12 V;
- Após acoplar o motor no eixo, o mesmo é montado novamente no reservatório de ração;
- Após montar o eixo motorizado no reservatório, utiliza-se uma fonte 12 V para alimentar o motor, coloca-se ração em grãos e aciona o motor.

A figura 3.12 ilustra o reservatório de ração, depois da colocação do motor.

Figura 3.12 – Reservatório de ração com o motor DC



Fonte: Autoria própria, 2016.

Resultados obtidos: a montagem do motor DC no reservatório de ração é um passo importante, é possível interligar o reservatório de ração com o Arduino e testar a rotina de entrega da ração. Com os testes funcionais, alimentando o motor DC, é possível verificar que a ração não “engancha” no eixo, isso é uma preocupação do grupo. No próximo passo é instalado no reservatório de ração o sensor ultrassônico para medir a quantidade.

3.3.2 Reservatório de água com a válvula acoplada

Para que seja fornecida água para o cão nos horários programados, é necessária a instalação de uma válvula no reservatório, que libera o fluxo de água, quando acionada pela programação. Esta válvula é escolhida pelo fato de ser utilizada diretamente no *Arduino*, e também por suportar a coluna de água que é aplicada na sua entrada. Neste processo são seguidos os seguintes procedimentos:

- primeiramente, a roscas da válvula é vedada com “veda rosca”;
- é montada na válvula uma redução para levar a mangueira para o prato de água;
- com durepox, é acoplado na válvula um adaptador que faz a ligação entre a válvula e o galão que armazena a água;
- abastece-se o galão com água e energiza-se a válvula para verificar o funcionamento.

A figura 3.13 ilustra o reservatório de água depois da montagem da válvula.

Figura 3.13 - reservatório de água com a válvula

Fonte: Autoria própria, 2016.

Resultados obtidos: a montagem da válvula no reservatório de água é um passo importante. É possível interligar o reservatório com o *Arduino* e testar a rotina de fornecimento de água. Com os testes funcionais verifica-se que, a válvula atende as necessidades do alimentador, pois ao colocar água no reservatório a mesma suportou a coluna de água sem vazar, este é um possível erro que o reservatório estava sujeito. O resultado foi satisfatório, quando acionada, a válvula liberou a água perfeitamente e quando desaciona e corta o fornecimento.

3.3.3 Montagem da bomba dreno no prato de água

Para que a água fornecida pelo reservatório não fique parada por muitas horas no prato, é instalada uma bomba dreno, que faz a drenagem da água, quando acionada pela programação. Esta ação se faz necessária para que não fique água parada no prato, causando doenças ao cão e até mesmo um depósito de mosquito

da dengue. Esta bomba dreno é escolhida pelo seu baixo custo, e também por sua fácil utilização. Neste processo foram seguidos os seguintes procedimentos:

- primeiramente, adapta-se a bomba em uma garrafa *pet* para verificar o seu funcionamento;
- depois de verificar seu funcionamento individual, monta-se a bomba no prato de água e faz à furação da mangueira de sucção que é acoplada no prato;
- após montar a bomba dreno no prato de água, alimenta-se a bomba com 12 V e coloca-se água no prato para verificar o funcionamento no alimentador.

A figura 3.14 ilustra o teste com a bomba dreno em funcionamento.

Figura 3.14 – Bomba dreno em funcionamento

Fonte: Autoria própria, 2016.

Resultados obtidos: com a montagem do sistema de drenagem da água no prato é possível verificar que, o tempo de drenagem da água é em torno de 50 segundos, a bomba não retira a água constantemente, mas em pulsos. A bomba esta bem vedada, pois não possui vazamento de água. Os resultados são satisfatórios, pois atendem as solicitações do protótipo, retirando a água parada do prato.

3.3.4 Montagem dos reservatórios na plataforma de fixação

Para melhor acomodar os reservatórios e também os pratos de ração e água, é desenvolvida uma plataforma de fixação, visando acomodar perfeitamente os reservatórios e também na altura onde ficam os pratos de ração e água, pois o pescoço do cão não pode ficar muito inclinado nem para cima ou para baixo. A seguir são expostas, passo a passo, as fases de desenvolvimento da plataforma:

- primeiramente são coletadas as medidas dos reservatórios e dos pratos de ração e água;
- com o auxilio do *software Autocad*, é desenvolvida a plataforma;
- com a ajuda da empresa Novemp é produzida a peça;
- posicionados os reservatórios e os pratos na plataforma, são marcadas as furações e efetuadas;
- depois de furada, a plataforma recebe a montagem dos reservatórios e dos pratos.

A figura 3.15 ilustra a plataforma montada com os reservatórios e os pratos de ração e água.

Figura 3.15 – Plataforma montada com os reservatórios

Fonte: Autoria própria, 2016.

Resultados obtidos: com os reservatórios montados na plataforma é possível verificar as facilidades e dificuldades que se tem para fazer a instalação elétrica e mecânica dos dispositivos que faz a automatização do alimentador. O próximo passo é montar os sensores e o dreno para interligar os periféricos no *Arduino.*

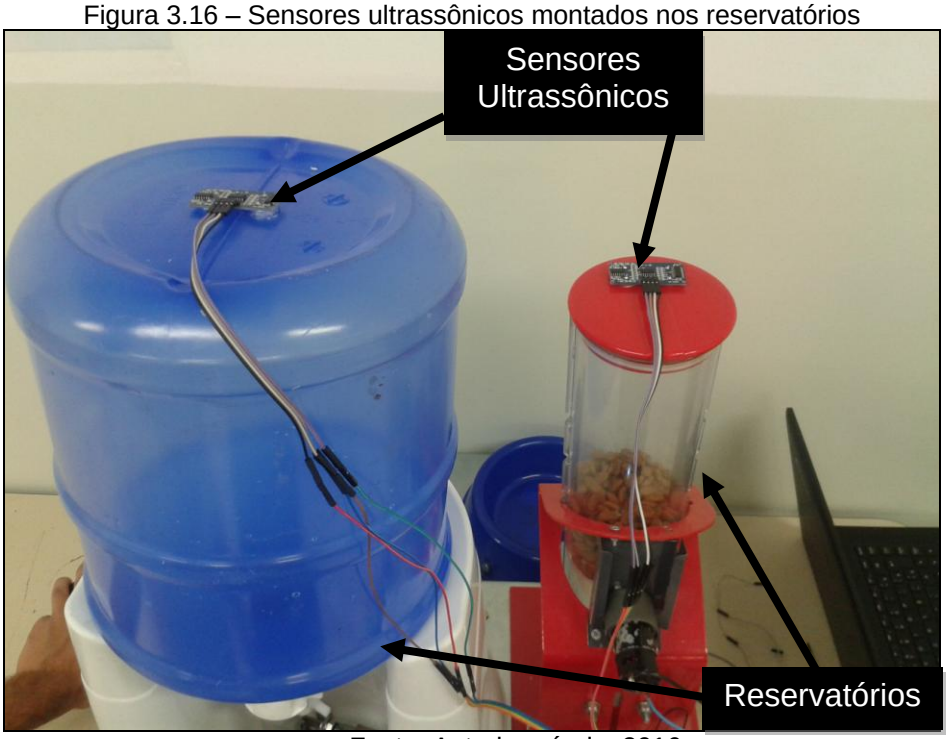
3.3.5 Montagem e testes dos sensores ultrassônicos

Para ter um controle de nível sobre os reservatórios de ração e água é necessário instalar sensores ultrassônicos. Estes sensores são responsáveis por medir a quantidade de ração e água e enviar os dados para o controlador, que por sua vez, envia alerta para o aplicativo, via *Wi-Fi*, quando os níveis estiverem baixos. Para a instalação são seguidos os seguintes passos:

- são retiradas as medidas dos sensores ultrassônicos;
- com auxilio de uma furadeira realizam-se as furações nas tampas dos reservatórios;
- com os sensores fixos nos seus respectivos reservatórios, são interligados no controlador;
- Água e ração são colocadas nos reservatórios;
- Os sensores e o controlador são energizados para verificar se as medições estão corretas.

A figura 3.16 ilustra a montagem e os testes realizados nos sensores.

Figura 3.16 – Sensores ultrassônicos montados nos reservatórios



Fonte: Autoria própria, 2016.

Resultados obtidos: com a colocação de água o sensor detectou uma determinada quantidade no reservatório, e quando a válvula liberou a água o sensor detectou automaticamente a nova medida. Similar é o teste no reservatório de ração, com a colocação de ração o sensor detecta uma determinada medida, e quando o motor libera a ração para o prato o sensor detectou a nova medida. A seguir a figura 3.17 ilustra o *display* com as medidas.

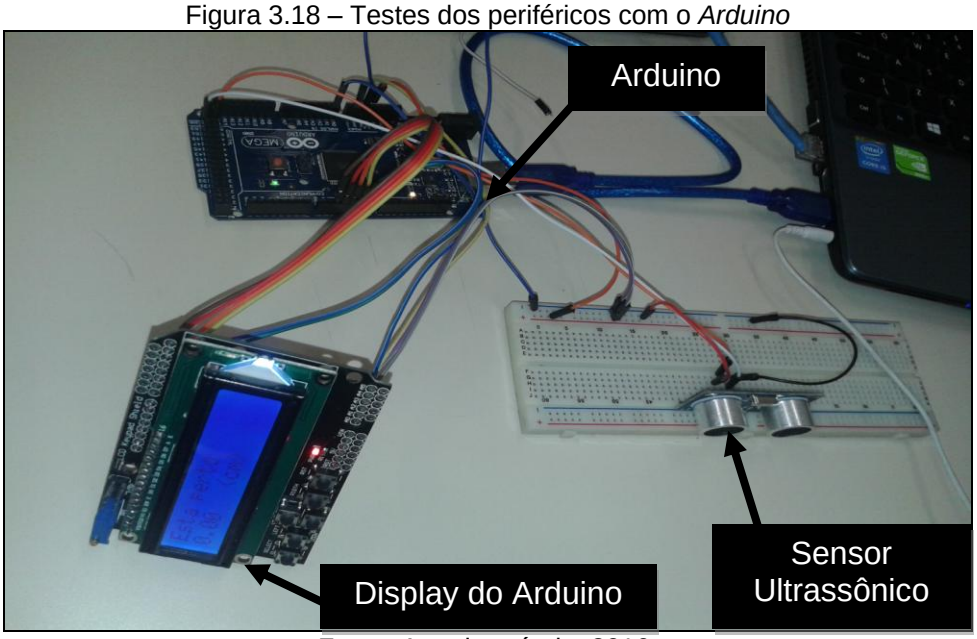
Figura 3.17 – Display com medidas dos reservatórios

Fonte: Autoria própria, 2016.

3.3.6 Controlador Arduino MEGA

Nesta fase é realizada a montagem do *display* e o sensor ultrassônico com o controlador *Arduino* MEGA. O controlador foi conectado no computador através de um cabo *USB* para alimentar com 5 V e realizar a comunicação, que é necessária para fazer o *upload* do programa e comunicação serial da aplicação. Através do *protoboard* pinos são conectados para alimentação 5 V e GND da placa *Arduino*, que alimenta o sensor ultrassônico e o *display.* Mutuamente testa-se o *display* e o sensor ultrassônico. Depois de programar, verificar e carregar o programa, o *display* é configurado para visualizar valores da distância do objeto do sensor ultrassônico, em centímetros. A figura 3.18 ilustra os testes dos periféricos com a placa *Arduino* MEGA.

Figura 3.18 – Testes dos periféricos com o *Arduino*



Fonte: Autoria própria, 2016.

Resultados obtidos: nos testes, analisa-se a forma mais objetiva de programar os periféricos e a interface de usuário através do programa, pesquisando em *sites* de estudo e fóruns, a forma mais fácil de alcançar os resultados esperados. Os resultados são satisfatórios, pois com a programação realizada é possível colocar em funcionamento o sensor ultrassônico que detectou a distancia em centímetros, e estes dados são visualizados corretamente no *display*.

3.3.7 Criação e testes do aplicativo

Para que o alimentador automatizado possua um diferencial frente às tecnologias existentes, é desenvolvido um aplicativo para *smartphone* que faz a interação com o alimentador. Neste aplicativo o usuário efetua o cadastramento do cão e em seguida programa as seguintes ações do alimentador: horários de fornecimento e a quantidade de ração e água. Também pode ser visto no aplicativo os níveis dos reservatórios, e quando estes níveis estiverem baixos, automaticamente o controlador envia alerta para o *smartphone* solicitando ao usuário que abasteça os reservatórios. Este aplicativo é desenvolvido para plataforma *Android* utilizando o programa *Android Studio* que é fornecido gratuitamente pela *Google.* A seguir estão as figuras que ilustram as telas da aplicação.

Após realizar o *download* do aplicativo o usuário tecla no ícone da aplicação na tela do *smartphone* para iniciar o cadastramento do cão. A figura 3.19 ilustra a tela de abertura do aplicativo

Figura 3.19 – Tela de abertura do aplicativo

Fonte: Autoria própria, 2016.

Depois de iniciado, o aplicativo apresenta o menu iniciar. Nesta tela o usuário escolhe se quer: cadastrar o cão, verificar se já tem algum cadastro salvo, níveis dos reservatórios e acionamento manual. A figura 3.20 ilustra a tela do menu inicial

Figura 3.20 – Tela de menu inicial

Fonte: Autoria própria, 2016

Após escolher, no menu iniciar cadastrar, o usuário informa o nome, o peso, o porte e a idade do cão. A figura 3.21 ilustra a tela de cadastro do cão.

Figura 3.21 – Tela de cadastro do cão

Fonte: Autoria própria, 2016

Em seguida informa a porção, em quilograma (kg), e depois informa os períodos e clicando em cadastrar o aplicativo envia os dados para o controlador. A figura 3.22 ilustra a tela de cadastro das refeições.

Figura 3.22 – Tela de cadastro das refeições

Fonte: Autoria própria, 2016

Depois de enviar os dados para o controlador o usuário pode verificar e editar os dados do cão já cadastrado. A figura 3.23 ilustra a tela de verificação e edição dos dados cadastrados.

Figura 3.23 – Tela de verificação e edição

Fonte: Autoria própria, 2016

No aplicativo é possível verificar os níveis dos reservatórios em tempo real. A figura 3.24 ilustra a tela de visualização dos níveis dos reservatórios.

Figura 3.24 – Tela de níveis dos reservatórios

Fonte: Autoria própria, 2016

Também é possível, no aplicativo, controlar a liberação da ração e da água e o acionamento da bomba dreno. A figura 3.25 ilustra a tela de controle das funções do alimentador.

Figura 3.25 – Tela de controle das funções

Fonte: Autoria própria, 2016

Resultados obtidos: como dito no objetivo, o aplicativo é o diferencial do projeto com ele o usuário faz toda a operação do alimentador. Com a realização do aplicativo verificou-se que, não pode ser transmitido os dados diretamente para o

controlador Arduino, e com a ajuda da empresa Koben criou-se um servidor *on-line,* neste servidor estão os bancos de dados que são transmitidos para o controlador. Os resultados foram satisfatórios e os dados transmitidos do aplicativo para o servidor e consequentemente para o controlador.

3.4 Montagem e testes finais de todo sistema do alimentador

Depois da realização das montagens e dos testes parciais de cada ação do alimentador, todo o sistema é integrado para realizar os testes finais. A seguir estão os procedimentos realizados:

- todos os componentes como: motor, válvula, bomba dreno, sensores ultrassônicos e relés são interligados com o controlador através de *bornes* e fios;
- os reservatórios de ração e água são abastecidos com os mantimentos;
- os pratos de ração e água são posicionados e fixados para receber os mantimentos;
- a mangueira da bomba dreno da água é ligada na parte inferior do prato de água;
- o sistema elétrico do alimentador é energizado na tomada de 127Vca;
- no aplicativo é cadastrado um cão para simulação, e colocado os horários e a quantidade de ração e água;
- os dados são enviados para o controlador via conexão *Wi-Fi*;
- no horário programado a quantidade solicitada é despejada nos pratos.

A figura 3.26 ilustra o alimentador finalizado.

Figura 3.26 – Alimentador finalizado

Fonte: Autoria própria, 2016

Resultados obtidos: após realizar toda a montagem do alimentador e simular o seu funcionamento, constatou-se que, ao cadastrar o cão no aplicativo e enviar os dados para o controlador o mesmo recebeu os dados e os guardou, no horário pré-estabelecido o controlador acionou o motor para a liberação da ração, determinou-se no aplicativo que o cão consome 100 g de ração, no teste, verificou-se que há um erro de 17 g, o que não é prejudicial ao cão.

Com relação à água, verificou-se que o prato comporta 600 ml, e através da programação realizou-se a liberação da água, constatou-se que a um erro de 50 ml, para corrigir este erro a programação vai liberar 500 ml por segurança e a cada 6 horas o dreno é acionado para fazer a troca da água.

Com relação aos sensores ultrassônicos, verificou-se que ao abastecer os reservatórios os sensores detectam o nível máximo e quando se simulou a liberação de ração e água consegui-se obter os níveis médio e mínimo, quando chega ao nível mínimo o controlador envia um alerta para o aplicativo, informando ao usuário que abasteça os reservatórios.

Com relação ao aplicativo e o controlador, verificou-se que o cadastramento do cão e das informações como: horários de fornecimento e quantidade são transmitidos com sucesso, do aplicativo para um servidor na nuvem, e deste servidor para o controlador. É importante salientar que, uma vez enviado, os dados são gravados no controlador, ou seja, caso o usuário fique sem acesso a redes *Wi-Fi* ou de dados móveis o cão não ficará sem a sua alimentação.

3.5 Obstáculos e soluções

Controlador: inicialmente o sistema de controle do alimentador era com PIC, mas além da limitação na comunicação via *Wi-Fi* é necessário confeccionar uma placa eletrônica, tomando um tempo desnecessário. A solução encontrada é utilizar um controlador Arduino MEGA que possui facilidade para fazer esta comunicação.

Válvula: inicialmente a liberação da água era por uma válvula solenoide, mas após a realização de pesquisas constatou-se que, a solenoide não suportaria a coluna de água do reservatório, ocasionando vazamentos. A solução encontrada é utilizar uma válvula esférica com um servo motor acoplado, a cada sinal que o controlador envia a este servo motor, o meso gira 90° liberando a água.

Medição do nível de água no reservatório: inicialmente os níveis do reservatório de água seriam detectados por sensores do tipo boia, mas este sensor não possui precisão. A solução encontrada é utilizar um sensor ultrassônico que detecta qualquer variação no reservatório.

Banco de dados: inicialmente os dados do cão cadastrados no aplicativo seriam enviados diretamente para o controlador, mas através de pesquisas constatou-se que, é necessário primeiramente armazenar estes dados em um servidor *on-line,* e deste servidor enviar para o controlador. A solução encontrada é fazer um servidor on-line, e com a ajuda da empresa Kobem este servidor está em funcionamento para fazer a ponte entre o aplicativo e o controlador.

Drenagem no prato de água: inicialmente o alimentador não possuía um sistema de drenagem da água parada no prato, e isso se tornou um problema, porque nos dias atuais o crescimento da doença transmitida pelo mosquito da dengue só vem aumentando. A solução encontrada é instalar uma bomba dreno que é acionada de 6 em 6 horas, este sistema evita a proliferação do mosquito da dengue e também as doenças ao cão.

CONSIDERAÇÕES FINAIS

O objetivo do trabalho intitulado Alimentador Automatizado para Cães é desenvolver um alimentador automático, que pode ser controlado e monitorado remotamente pelo *smartphone*. Justifica-se que o alimentador proporciona ao animal melhor bem estar em relação á alimentação, fornecendo ração e água para cães de forma controlada, propiciando praticidade para o seu dono. Com a finalização do protótipo verificou-se que o objetivo principal está alcançado.

Utilizando uma tecnologia inovadora, o protótipo é projetado e construído basicamente com os seguintes dispositivos: sensores ultrassônicos, motor DC 12 V, válvula esférica, bomba dreno, aplicativo e um controlador Arduino MEGA que possibilitam a operação à distância. O aplicativo pode ser baixado e instalado no *smartphone,* e permite ao usuário alimentar o cão em qualquer lugar.

As teorias pesquisadas são de suma importância para sustentar a concretização do desenvolvimento do projeto, principalmente no que tange a eletrônica digital e analógica, redes de comunicação sem fio, sistemas de controles, hidrologia e condicionamento físico.

As diretrizes dadas pela metodologia cientifica, contribuíram para a organização, direcionamento e suporte para a concretização do projeto. Os métodos e técnicas realizados contribuíram para adquirir novos conhecimentos. Com o decorrer do desenvolvimento o conhecimento cientifico, tornou mais claro o sistema de ideias, ampliando a visão sobre a ocorrência dos fatos.

Durante o desenvolvimento sugiram alguns contratempos, tais como: a dificuldade de armazenamento dos dados cadastrados do cão, no aplicativo, pois estes dados não podem ser enviados diretamente para o controlador Arduino. Mas com o auxilio das teorias pesquisadas, experiência profissional dos integrantes do grupo e do orientador, os problemas foram solucionados.

Como vantagem pode-se destacar que, uma vez instalado o alimentador e cadastrado o cão no aplicativo, não há a necessidade de intervenção humana para o fornecimento dos mantimentos, até que os reservatórios estejam abastecidos; alerta de diminuição dos mantimentos nos reservatórios; fracionamento da porção de ração, visando o bem estar do animal; troca de água a cada 6 horas para não ficar água parada no prato.

Destacam como desvantagens, que o alimentador só pode ser utilizado com rações em grãos; precisa estar conectado a rede elétrica; não possui um sistema de *no-break* para a alimentação auxiliar do protótipo e é necessário ter um *smartphone*.

Do projeto concretizado pode-se dizer que ele proporcionou aos integrantes do grupo uma visão diferente da automação, pois a indústria esta sempre com o foco, mas através deste projeto, percebe-se que a automação possui diversos seguimentos de atuação.

Como melhoria, fica a sugestão para futuros trabalhos à implementação de um sistema de pesagem no alimentador. Este sistema pode guardar os dados da pesagem a cada dia e emitir um gráfico no aplicativo, indicando se o cão está se alimentando corretamente.

REFERÊNCIAS

ALESSANDRO, A. **Quanto seu cão e gato ingerem de água por dia.** Vila Velha-ES, 2012.
Disponível em:
< http://doutordebichoveterinaria.blogspot.com.br/2012/08/quanto-seu-cao-e-gato-ingerem-de-agua-e.html >. Acesso em: 10 mai. 2016.

ARDUINO. **Arduino UNO & Genuino UNO**. Estados Unidos, 2016.
Disponível em: < https://www.arduino.cc/en/Main/ArduinoBoardUno> Acesso em: 13 mar. 2016.

ASSOCIAÇÃO BRAILEIRA DE NORMAS TÉCNICAS. **NBR 14724 – Formatação de trabalho cientifico.**
Disponível em: <http://www.abnt.org.br>. Acesso em: 02 mar. 2016.

ASSOCIAÇÃO BRAILEIRA DE NORMAS TÉCNICAS. **NBR 10719 – Relatório técnico e/ou cientifica - Apresentação.**
Disponível em: <http://www.abnt.org.br>. Acesso em: 07 Set. 2016.

AUGUSTO, C.; LUIZ, A. **Tecnologia Móvel:** Uma Tendência, Uma Realidade. Minas Gerais**:** Universidade Estácio de Sá, 2011. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1105/1105.3715. pdf>. Acessado em: 11 mai. 2016.

BRAGA, N. **Como funcionam os sensores capacitivos**. Brasil, 2016. Disponível em:
< http://www.newtoncbraga.com.br/index.php/como-funciona/5849-como-funcionam-os-sensores-capacitivos-art761>.
Acesso em: 12 mai. 2016.

BRIGATTO, G. **Número de usuários de smartphones no Brasil cresce 48% no 3° trimestre**. São Paulo, 2015.
Disponível em:
< http://www.valor.com.br/empresas/4327844/numero-de-usuarios-de-smartphones-no-brasil-cresce-48-no-3-trimestre >. Acesso em: 24 mar. 2016.

COSTA, R. **Aquisição de dados e interface de comunicação em instrumentos de medição de massa**. São Paulo: Inmetro, 2005.

ELGIN, B. **Google Buys Android for Its Mobile Arsenal.** Estados Unidos, 2005. Disponível em: < http://www.webcitation.org/5wk7sIvVb> Acesso em 31 mar. 2016.

GREGORY. **Arduino – Entradas Analógicas**. São Paulo, 2014.
Disponível em: <http://allelectronics.com.br/arduino-entradas-analogicas>. Acesso em: 13 mar. 2016.

INFOWESTER. **Wifi**. Brasil, 2008. Disponível em:
<http://www.infowester.com/wifi.php >.
Acesso em: 30 Out. 2016.

LEMOS, M. **Massimo Banzi:** Makers que você precisa conhecer. Minas Gerais, 2015. Disponível em: <http://blog.fazedores.com/massimo-banzi-makers-que-voce-precisa-conhecer>. Acesso em: 13 mar. 2016.

LECHETA, R. **Google Android**: Aprenda a criar aplicações para dispositivos móveis com Android SDK. 2. ed. São Paulo: Novatec, 2011.

MACHADO, R. **Energizando o Arduino**. São Paulo, 2011. Disponível em:
<http://www.eaduino.com.br/2011/08/energizando-o
arduino/#sthash.zmbvzZ2p.QUpZS3f0.dpbs >. Acesso em: 13 mar. 2016.

MEGATNT. **Pesquisa de Mercado**. São Paulo, 2016.
Disponível em:
<http://www.megatnt.com.br/alimentador-automatico-de-agua-e-rac-o-para-c-o-e-gato.html?gclid=CKWY_-Luw8sCFYIGkQodg6QBVA>. Acesso em: 15 mar. 2016.

NABAIS, S. **BÊ-A-BA DO CÃO**. Portugal: Bayer HealthCare, 2010. Disponível em: <HTTP://www.livredeparasitas.com/static/documents/Livro_do_Cao.pdf>. Acesso em: 14 mar. 2016.

NETMARKETSHARE. **Mobile/Tablet Top Operating System Share Trend**. Estados Unidos, 2016. Disponível em:
<http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustomb=1&qpct=4&qpsp=198&qpnp=8&qptimeframe=M>.
Acesso em: 24 mar. 2016.

OPEN HANDSET ALLIANCE (OHA). **Members.** Estados Unidos, 2016. Disponível em: <http://www.openhandsetalliance.com/oha_members.html>. Acesso em: 31 mar. 2016

PALHARES, R. et al. **Controle de posição de uma válvula**. Minas Gerais, 2015. Disponível em: <http://www.cpdee.ufmg.br/~palhares/valvula.pdf>. Acesso em: 29 mar. 2016.

PETLOVE. **Pesquisa de Mercado**. São Paulo, 2016. Disponível em: <https://www.petlove.com.br/alimentador-automatico-trixie-para-racao-do-pet-tx4-plus-com-visor-lcd---500ml-3104104 >. Acesso em: 15 mar. 2016.

PURINALATAM. **Alimentação Canina**. São Paulo, 2016. Disponível em: < https://www.purinalatam.com/br/homepage.aspx>. Acesso em: 15 mar. 2016.

REGILAN. **Banco de dados**. Brasil, 2013. Disponível em: <http://www.regilan.com.br/wp-content/uploads/2013/10/Banco-de-Dados.pdf >. Acesso em: 30 Out. 2016.

RODRIGUES, L. **Métodos de avaliação da condição corporal em cães**. P. 3 e 5. Trabalho de Pós-Graduação em Ciência Animal – Escola de Veterinária e Zootecnia da UFG. Goiás, 2011. Acesso em: 25 mar. 2016.

SEBRAE. **Inovação e tecnologia:** Internet das coisas. São Paulo, 2016. Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/artigos/inovacao-e-tecnologia-internet-das-coisas,05d99e665b182410VgnVCM100000b272010aRCRD >. Acesso em: 30 out. 2016.

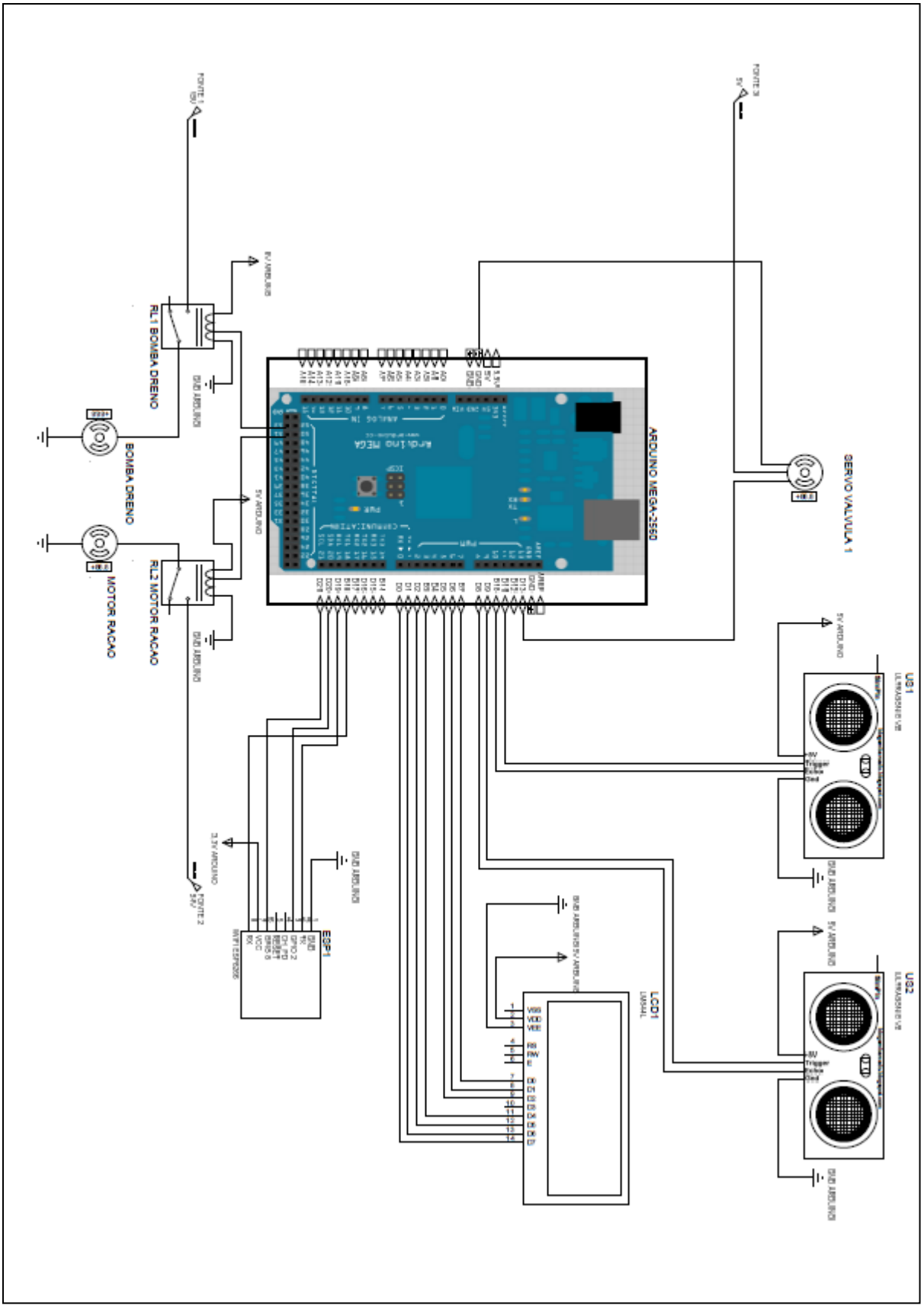
SEVERINO, A. J. **Metodologia do trabalho científico**. 23.ed. São Paulo. Ed. Cortez, 2007

TECMUNDO. **O que é Wifi?**. Brasil, 2012. Disponível em: <http://www.tecmundo.com.br/wi-fi/197-o-que-e-wi-fi-.htm >. Acesso em: 30 Out. 2016.

TUBALDINI, R. **Quantidade de ração para cães e gatos**: Quanto dar de ração?. São Paulo, 2014. Disponível em: < http://www.cachorrogato.com.br/cachorros/quantidade-racao-caes-gatos/>. Acesso em: 25 mar. 2016.

THOMAZINI, D. **Sensores industriais, fundamentos e aplicação**. Ed. Érica. São Paulo, 2007.

APÊNDICE A – DAGRAMA ELÉTRICO DO ALIMENTADOR



APÊNDICE B – PROGRAMAÇÃO DO CONTROLADOR ARDUINO

```
 #include <LiquidCrystal.h> //biblioteca display
 LiquidCrystal lcd(6,7,2,3,4,5); //(<pino RS>, <pino enable>, <pino D4>, <pino D5>, <pino
D6>, <pino D7>)

 #include <Servo.h> //biblioteca da valvula
 Servo servo();     //declarando responsavel pelo servo

 #include<Ultrasonic.h>  //biblioteca sensor ultrssonico
 Ultrasonic ultrasonic1(10,11); //Trigger-10 e Echo-11 //ultrassonico da racao
 Ultrasonic ultrasonic2(12,13); //trigger-12 e Echo-13 // ultrassonico da agua

 #include <DS1307.h>   //biblioteca modulo rtc
 DS1307 rtc();   //pinos de dados do rtc Analogico 1 e 2

       //declara variaveis utilizadas, tais como int, float, long, etc...

//configurar os pinos corretamente
   const int dreno=22;     //Pino 22 para acionamento da bomba de dreno da agua
   const int valvula=23;   //Pino 23 para acionamento da valvula de escape de agua
   const int Motor=24;     //Pino 24 para acionamento do motor da racao
   const int sensor_cao=13;

   long microsec1 = 0;
   float ultra_racao = 0;    //Leitura do sensor ultrassonico do reservatorio de racao
   long microsec2 = 0;
   float ultra_agua = 0;     //Leitura do sensor ultrassonico do reservatorio de agua

void setup() {       // programa principal de leitura antes do Loop
 //modo de funcionamento dos pinos, se output ou input
 //inicia as aplicações declaradas. Ex.: lcd.begin(a, b); inicia o lcd
```

```
  Serial.begin(9600); //inicia o serial monitor para verificar o funcionamento
  lcd.begin(16, 2);   //inicializacao do lcd com 16 colunas e 2 linhas

     pinMode (valvula, OUTPUT);    //pino 9 como saida para acionar a valvula agua
     pinMode (dreno, OUTPUT);      //pino 8 como saida para acionar bomba dreno

//   rtc.halt(false);

    lcd.clear();
    lcd.setCursor(0,0);
    Serial.println ("Seja bem vindo");
    lcd.print("Seja bem vindo");
    delay (1500);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("Smart Dog");
    Serial.println("Smart Dog");
    delay (5000);
    lcd.clear();
}
void loop()  {      // rotina de alimentacao do animal, Loop de leitura
  int h2=0;
  int m2=0;
  int m3=0;
  int h3=0;
  int h4=0;
  int m4=0;
  int rtc[6];
   //Lendo o sensor ultrasonico
  microsec1 = ultrasonic1.timing();
  //Convertendo a distância em CM
  ultra_racao = ultrasonic1.convert(microsec1, Ultrasonic::CM);
```

```
  //Lendo o sensor ultrasonico
  microsec2 = ultrasonic2.timing();
  //Convertendo a distância em CM
  ultra_agua = ultrasonic2.convert(microsec2, Ultrasonic::CM);

  if (rtc[2] == h2 && rtc[1] == m2){      //rotina de alimentacao do 1o horario
   if (analogRead (ultra_racao) >= 20){    //nivel de racao baixo
    Serial.println ("O nivel de racao está baixo");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Racao: minimo");
    }
   if (analogRead (ultra_racao) <= 20 and analogRead (ultra_racao) >= 10){     //nivel de racao
muito baixo, emergencia!!!
   //  Serial.println ("Emergencia!!! ");
    Serial.print ("O nivel de racao esta medio");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Racao: medio");
    }
   if (analogRead (ultra_racao) <= 5){     //nivel de racao muito baixo, emergencia!!!
    Serial.print ("O nivel de racao esta maximo");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Racao: maximo");
    }
   racao();
   if (analogRead (ultra_agua) >= 20){     //nivel de agua baixo
    Serial.println ("O nivel de agua esta baixo");
    lcd.clear();
    lcd.setCursor (0,0);
```

```
    lcd.print("Agua: minimo");
    }
   if (analogRead (ultra_agua) <= 20 and analogRead (ultra_agua) >= 10)
   {
    // Serial.println ("Emergencia!!! ");
    Serial.print ("O nivel de agua medio");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Agua: medio");
    }
   if (analogRead (ultra_racao) <= 5){
    Serial.print ("O nivel de racao esta maximo");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Agua: maximo");
    }
   agua();
  }
  if (h3 != 0){                    //rotina de alimentacao do 2o horario
    if (rtc[2] == h3 && rtc[1] == m3){
   if (analogRead (ultra_racao) >= 20){    //nivel de racao baixo
    Serial.println ("O nivel de racao está baixo");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Racao: minimo");
    }
   if (analogRead (ultra_racao) <= 20 and analogRead (ultra_racao) >= 10){     //nivel de racao
muito baixo, emergencia!!!
   //  Serial.println ("Emergencia!!! ");
    Serial.print ("O nivel de racao esta medio");
    lcd.clear();
    lcd.setCursor (0,0);
```

```
  lcd.print("Racao: medio");
  }
 if (analogRead (ultra_racao) <= 5){     //nivel de racao muito baixo, emergencia!!!
  Serial.print ("O nivel de racao esta maximo");
  lcd.clear();
  lcd.setCursor (0,0);
  lcd.print("Racao: maximo");
  }
 racao();
 if (analogRead (ultra_agua) >= 20){     //nivel de agua baixo
  Serial.println ("O nivel de agua esta baixo");
  lcd.clear();
  lcd.setCursor (0,0);
  lcd.print("Agua: minimo");
  }
 if (analogRead (ultra_agua) <= 20 and analogRead (ultra_agua) >= 10)
 {
  // Serial.println ("Emergencia!!! ");
  Serial.print ("O nivel de agua medio");
  lcd.clear();
  lcd.setCursor (0,0);
  lcd.print("Agua: medio");
  }
 if (analogRead (ultra_racao) <= 5){
  Serial.print ("O nivel de racao esta maximo");
  lcd.clear();
  lcd.setCursor (0,0);
  lcd.print("Agua: maximo");
  }
 agua();
}
if (h4 != 0){                    //rotina de alimentacao do 2o horario
```

```
    if (rtc[2] == h4 && rtc[1] == m4){
   if (analogRead (ultra_racao) >= 20){    //nivel de racao baixo
    Serial.println ("O nivel de racao está baixo");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Racao: minimo");
    }
   if (analogRead (ultra_racao) <= 20 and analogRead (ultra_racao) >= 10){     //nivel de racao
muito baixo, emergencia!!!
   //  Serial.println ("Emergencia!!! ");
    Serial.print ("O nivel de racao esta medio");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Racao: medio");
    }
   if (analogRead (ultra_racao) <= 5){     //nivel de racao muito baixo, emergencia!!!
    Serial.print ("O nivel de racao esta maximo");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Racao: maximo");
    }
   racao();
   if (analogRead (ultra_agua) >= 20){     //nivel de agua baixo
    Serial.println ("O nivel de agua esta baixo");
    lcd.clear();
    lcd.setCursor (0,0);
    lcd.print("Agua: minimo");
    }
   if (analogRead (ultra_agua) <= 20 and analogRead (ultra_agua) >= 10)
   {
    // Serial.println ("Emergencia!!! ");
    Serial.print ("O nivel de agua medio");
```

```
      lcd.clear();
      lcd.setCursor (0,0);
      lcd.print("Agua: medio");
      }
    if (analogRead (ultra_racao) <= 5){
      Serial.print ("O nivel de racao esta maximo");
      lcd.clear();
      lcd.setCursor (0,0);
      lcd.print("Agua: maximo");
      }
    agua();
   }
   }
}
}
void agua()  {      // subrotina de verificacao e libera agua

  lcd.clear ();
  lcd.setCursor (0,0);
  lcd.print ("Liberando agua");
  digitalWrite(dreno, HIGH);    //aciona a saida 8
  delay(5000);                  //5 segundos para o dreno
  digitalWrite(dreno, LOW);     //desliga o dreno
  digitalWrite(valvula, HIGH);  //aciona a valvula
  delay (5000);                 //5 segundos para a valvula aberta
  digitalWrite(valvula, LOW);   //desliga valvula agua
  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print ("OK");
  delay(1000);
  lcd.clear();
}
```

```
void racao() {      // subrotina de verificacao e libera racao
   lcd.clear();
   lcd.setCursor (0,0);
   lcd.print ("Liberando racao");
   analogRead (sensor_cao);     //sensor de presença do cao
   pinMode(Motor, OUTPUT);      //pino 8 como saida para acionar bomba dreno
   digitalWrite(Motor, HIGH);   //aciona a saida 8
   delay(2000);                 //5 segundos para o dreno
   digitalWrite(Motor, LOW);    //desliga o dreno
   lcd.setCursor (0,1);
   lcd.print ("OK");
   delay(1000);
   lcd.clear();
   }
   void wifi(){
      }
```

APÊNDICE C – PROGRAMAÇÃO DO APLICATIVO

```
package tcc.smartdog;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.*;
import android.app.*;
import android.app.Activity;
import android.widget.Spinner;
import org.apache.http.client.HttpClient;

import org.apache.http.message.BasicNameValuePair;



public class main1 extends AppCompatActivity {



    Spinner porte;
    Registro pri, reg, ult, aux;
    EditText ednome, edpeso, edidade, edporte, edporcao, edumalimentacao,
eddoisalimentacao, edtresalimentacao;
    int numreg, pos;



    @Override //ROTINA INICIAL
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

```
        Menu();

       porte = (Spinner) findViewById(R.id.spinPORTE);

        ArrayAdapter adapter = ArrayAdapter.createFromResource(this,
R.array.lista_porte, android.R.layout.simple_spinner_item); // define o layout do
spinner
        porte.setAdapter(adapter);



    }

    void Menu() {
        setContentView(R.layout.menu_inicial);

        Button bt_tela_cadastro = (Button) findViewById(R.id.bt_tela_cadastro);

        bt_tela_cadastro.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Cadastro();
            }
        });

        Button bt_tela_cadastrado = (Button) findViewById(R.id.bt_tela_cadastrado);

        bt_tela_cadastrado.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Cadastrados();
            }
        });
```

```
        Button bt_tela_niveis = (Button) findViewById(R.id.bt_tela_niveis);

        bt_tela_niveis.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Niveis();
            }
        });

        Button bt_tela_manual = (Button) findViewById(R.id.bt_tela_manual);

        bt_tela_manual.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Opcoes();
            }
        });


    } //CArrega a tela inicial de menu

    void Cadastro() {

        setContentView(R.layout.cadastro1);

        porte = (Spinner) findViewById(R.id.spinPORTE);

        ArrayAdapter adapter = ArrayAdapter.createFromResource(this,
R.array.lista_porte, android.R.layout.simple_spinner_item); // define o layout do
spinner
        porte.setAdapter(adapter);

        Button btcadastrar = (Button) findViewById(R.id.btcadastrar);
```

```
btcadastrar.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {

        try {
            reg = new Registro();

            ednome = (EditText) findViewById(R.id.ednome);
            edpeso = (EditText) findViewById(R.id.edpeso);
            edidade = (EditText) findViewById(R.id.edidade);
           // edporte = (Spinner) findViewById(R.id.spinPORTE);
            edporcao = (EditText) findViewById(R.id.edporcao);
            edumalimentacao = (EditText) findViewById(R.id.umalimentacao);
            eddoisalimentacao = (EditText) findViewById(R.id.doisalimentacao);
            edtresalimentacao = (EditText) findViewById(R.id.tresalimentacao);


            reg.nome = ednome.getText().toString();
            reg.peso = edpeso.getText().toString();
            reg.idade = edidade.getText().toString();
           // reg.porte = edporte.getText().toString();
            reg.porcao = edporcao.getText().toString();
            reg.umalimentacao = edumalimentacao.getText().toString();
            reg.doisalimentacao = eddoisalimentacao.getText().toString();
            reg.tresalimentacao = edtresalimentacao.getText().toString();



            if (pri == null)
               pri = reg;

            reg.Ant = ult;
```

```
                if (ult == null)
                    ult = reg;
                else {
                    ult.Prox = reg;
                    ult = reg;
                }

                numreg++;
                showMessage("Cadastrado com Exito", main1.this);
                Cadastrados();

            }
            catch (Exception e) {
              showMessage("Erro Ao efetivar o cadastro", main1.this);
              // Cadastrados();
             }
        }
    });
    Button bt_voltar = (Button) findViewById(R.id.bt_voltar);

    bt_voltar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            Menu();
        }
    });
}   //Carrega a tela de cadastro

void Cadastrados() {
    if (numreg == 0) {
        showMessage("Não consta Cadastro", main1.this);

        // CarregaTelaInicial();
        return;
```

```
        }

        setContentView(R.layout.cadastrado);
        pos = 1;
        aux = pri;
        TextView txtnome = (TextView) findViewById(R.cadastrado.txtnome);
        TextView txtidade = (TextView) findViewById(R.cadastrado.txtidade);
        TextView txtpeso = (TextView) findViewById(R.cadastrado.txtpeso);
        TextView txtporcao = (TextView) findViewById(R.cadastrado.txtporcao);
        //TextView txtporte = (TextView) findViewById(R.cadastrado.txtporte);
        TextView txtumalimentacao = (TextView)
findViewById(R.cadastrado.txtumalimentacao);
        TextView txtdoisalimentacao = (TextView)
findViewById(R.cadastrado.txtdoisalimentacao);
        TextView txttresalimentacao = (TextView)
findViewById(R.cadastrado.txttresalimentacao);

        Button btmenu = (Button) findViewById(R.cadastrado.btmenu);
        Button btavancar = (Button) findViewById(R.cadastrado.btavancar);
        Button btvoltar = (Button) findViewById(R.cadastrado.btvoltar);

        txtnome.setText(aux.nome);
        txtpeso.setText(aux.peso);
        txtidade.setText(aux.idade);
        txtporte.setText(aux.porte);
        txtporcao.setText(aux.porcao);
        txtumalimentacao.setText(aux.umalimentacao);
        txtdoisalimentacao.setText(aux.doisalimentacao);
        txttresalimentacao.setText(aux.tresalimentacao);


        Button bteditar = (Button) findViewById(R.id.bteditar);
        bteditar.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
        public void onClick(View arg0) {

            pri = null;

            Cadastro();
        }
    });



    Button btvoltar = (Button) findViewById(R.id.btvoltar);

    btvoltar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            Menu();
        }
    });
}   //Carrega o cadastro atual

void Niveis() {
    setContentView(R.layout.niveis);

    Button bt_voltar = (Button) findViewById(R.id.bt_voltar);
    bt_voltar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            Menu();
        }
    });
}   //Carrega tela com niveis de mantimentos

void Opcoes() {
    setContentView(R.layout.tela_opcoes);
```

```
    Button bt_voltar = (Button) findViewById(R.id.bt_voltar);
    bt_voltar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            Menu();
        }
    });
}   //Carrega tela acionamentos manuais

public void showMessage(String Caption, Activity activity) {
    AlertDialog.Builder dialogo = new AlertDialog.Builder(activity);
    dialogo.setTitle("Atencao");
    dialogo.setMessage(Caption);
    dialogo.setNeutralButton("OK", null);
    dialogo.show();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.abertura, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Handle action bar item clicks here. The action bar will
    automatically handle clicks on the Home/Up button, so long
    as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
```

```
        }

        return super.onOptionsItemSelected(item);
    }
}
```