UNIVERSIDADE ESTADUAL PAULISTA

CAMPUS DE GUARATINGUETÁ

BRUNO FEU DE BRITO SANTOS

ALIMENTADOR AUTOMÁTICO PARA ANIMAIS UTILIZANDO ARDUINO

Guaratinguetá

2015

BRUNO FEU DE BRITO SANTOS

ALIMENTADOR AUTOMÁTICO PARA ANIMAIS UTILIZANDO ARDUINO

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador: Professor Dr. Daniel Julien B. da S. Sampaio

Guaratinguetá

2015

Santos, Bruno Feu de Brito
S237a Alimentador automático para animais utilizando Arduino / Bruno Feu de Brito Santos – Guaratinguetá : [s.n], 2015.
51 f : il.
Bibliografia: f. 37

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2015.
Orientador: Prof. Dr. Daniel Julien Barros da Silva Sampaio

1. Arduino (Controlador programável) 2. Controladores programáveis 3. Microcontroladores I. Título

CDU 621.381

UNIVERSIDADE ESTADUAL PAULISTA

CAMPUS DE GUARATINGUETÁ

BRUNO FEU DE BRITO SANTOS

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE "GRADUADO EM ENGENHARIA ELÉTRICA"

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Prof. Dr. Leonardo Mesquita

Coordenador

BANCA EXAMINADORA:

Prof. Dr. Daniel Julien Barros da Silva Sampaio
Orientador/UNESP-FEG

Prof. Dr. Durval Luiz Silva Ricciulli
UNESP-FEG

Prof. Eng. Fernanda Trujillo Leon
UNESP-FEG

Abril 2015

Dedico este trabalho a meus pais, Amarildo e Rosa, pelo amor, incentivo, dedicação e colaboração para o cumprimento deste tão sonhado momento em minha vida.

AGRADECIMENTOS

Agradeço primeiramente a Deus pela minha vida, por tudo que Ele me proporcionou até o momento. Por ter me dado saúde e força para superar todas as dificuldades.

Aos meus pais, Amarildo e Rosa, pois sem eles eu não estaria nem perto de onde estou e não conseguiria conquistar metade do que consegui. Obrigado pela força, pela luta, pelo suor e pelo tempo dedicado à minha educação e formação pessoal e profissional.

Aos meus irmãos Igor e Lucas, por todo o apoio, amizade, brigas, discussões e, acima de tudo, por toda a admiração e companheirismo. O orgulho de fazer parte desta família é indescritível, e agradeço vocês por isto.

Ao meu orientador, Prof. Daniel, por todo o tempo e paciência dedicados e à sua ajuda, que foi o que me permitiu concluir este trabalho.

À minha segunda família, a República WC Kzona, por todo o companheirismo durante todos estes anos. Vocês são, sem sombra de dúvidas, exemplos que terei sempre comigo. Aos meus irmãos, Ricochete, Churros, Temo, Dutra, Dexter, Marquito, Xiuaua, Currida, Muamba, Guaxinim, Cirilo, Gargamel, Jeremias, Tinho Dudu, Salsicha, Polengo, Série B, Brioco, Cascão, Magal, Watchatcha, Rosinha, Zéco, Hipólito, Pépe, Angélica, Sérgio, Bigorna, Jão e Tortuguita. Obrigado por todas as risadas, festas, brincadeiras e babaquices que construíram todos estes anos de faculdade.

À todos da República 69, lugar onde tive o prazer de morar os últimos anos da minha graduação.

À Talita, que foi fundamental do início ao fim da realização deste trabalho e esteve sempre presente nos momentos finais desta tão sonhada graduação.

E a todos que de alguma forma, direta ou indiretamente, tornaram este trabalho possível, meu muito obrigado.

SANTOS, B. F. B. **Alimentador automático para animais utilizando Arduino.** 2015. 51 f. Trabalho de graduação (Graduação em Engenharia Elétrica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2015.

RESUMO

Este trabalho apresenta um projeto de um alimentador automático para animais domésticos utilizando um Arduino Uno como central de controle. Por meio de estudos sobre este controlador foi possível criar um dispositivo com uma interface capaz de receber dados fornecidos pelo usuário e depois utilizá-los para ativar o alimentador de acordo com horários definidos. Para a montagem do equipamento utilizou-se motores de passos, sensores, teclado e display, que trabalham em conjunto para o funcionamento do aparelho. O objetivo do projeto foi alcançado e o protótipo desenvolvido indicando que o Arduino pode ser utilizado para várias aplicações que podem simplificar tarefas quotidianas.

PALAVRAS-CHAVE: Arduino. Alimentador para Animais. Motor de passo.

SANTOS, B. F. B. **Automatic Animal feeder using Arduino.** 2015. 51 f. Graduate Work (Graduate in Electricial Engineering) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2015.

ABSTRACT

This paper presents a Project of an automatic feeder for pets using an Arduino Uno as the control center. Through studies on this driver was possible to create a device with an interface capable to receiving user input and then use it to activate the feeder in the defined hours. For mounting equipment were used steps motors, sensors, keyboard and display, which work together to instrument operation. The project goal was reached and the prototype developed indicating that the Arduino can be used for various applications that can simplify daily tasks.

KEYWORDS: Arduino. Animal Feeder. Stepper Motor.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

A	Ampere
C++	Linguagem de Programação C++
CA	Corrente Alternada
CC	Corrente Contínua
DC	Corrente contínua
EEPROM	Electrically-Erasable Programmable Read-Only Memory
GND	Terra
I/O	Entrada/Saída
PWM	Modulação por largura de pulso
SRAM	Static Random Access Memory
USB	Universal Serial Bus
V	Volt

SUMÁRIO

1 INTRODUÇÃO

Hoje em dia os animais de companhia vêm apresentando um gradativo aumento de peso devido à má alimentação. A origem deste sobrepeso geralmente é causada pela falta de controle da quantidade de alimento servido ao animal. Normalmente ele está disponível durante todo o dia, quando deveria ser servido em horários e quantidade controladas. Esta falta de controle pode estar associada à fatores como falta de tempo dos donos, que podem não estar presentes quando é necessário ou esquecer dos horários, ou até mesmo alguma viagem, onde o animal não pode estar junto e deve ficar em casa sozinho.

Este trabalho é desenvolvido afim de solucionar este problema. A criação de um dispositivo que serve água e ração automaticamente. O usuário insere os horários que o animal será alimentado, enche os reservatórios de armazenamento e o dispositivo faz todo o trabalho.

Para a montagem do alimentador são utilizados diversos componentes como sensor de nível, motores de passo, display e Arduino. O sensor é responsável por fazer a leitura do ambiente e transmitir a informação capturada. Os motores de passo são utilizados para o aparelho funcionar e fornecer o alimento ou a água para o animal. O display, com botões físicos para ajustes, é a interface entre o usuário e o dispositivo, onde são configurados todos os parâmetros para o equipamento funcionar. O Arduino é responsável por receber a informação dos sensores, atuando conforme a programação, enviando sinais aos motores e ao display.

1.1 OBJETIVOS

A motivação deste trabalho é que o animal seja alimentado nos horários programados para prevenir problemas de saúde. O objetivo, portanto, é o projeto e desenvolvimento de um alimentador automático para animais domésticos utilizando a plataforma microcontrolada Arduino Uno como central de controle, proporcionando uma interface homem máquina onde o usuário poderá escolher os horários nos quais o dispositivo irá funcionar automaticamente.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho é apresentado em cinco capítulos.

No segundo capítulo são apresentadas informações sobre os componentes que são utilizados no alimentador automático para animais.

No terceiro capítulo são apresentados os materiais utilizados juntamente com seu respectivo funcionamento no sistema.

No quarto capítulo é mostrado todo o funcionamento do alimentador, como são os ajustes dos horários, as informações que o usuário deve inserir, até o objetivo final, que é a entrega do alimento e água no local destinado.

No quinto capítulo são apresentadas as considerações finais e conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados alguns componentes que foram utilizados e que merecem um maior aprofundamento teórico para a construção do dispositivo, além de fazer uma conexão entre todos e como eles podem ser utilizados em conjunto.

2.1 ARDUINO

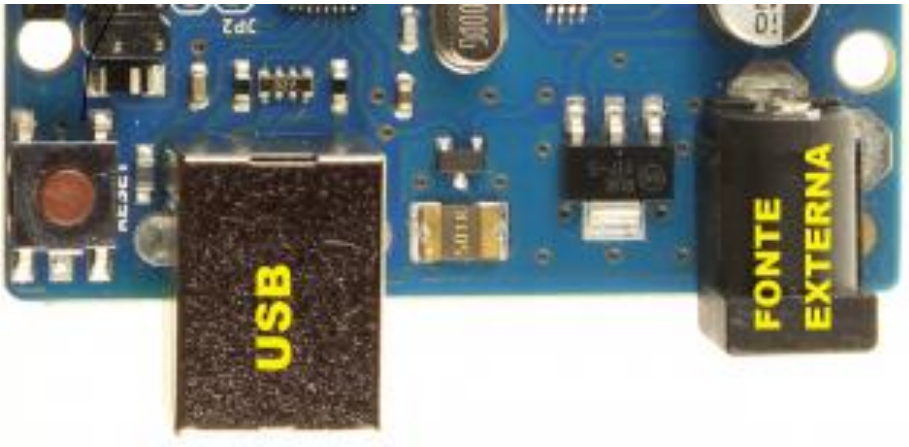
O Arduino faz parte do conceito de hardware e software livre e está aberto para uso e contribuição de toda sociedade. O conceito Arduino surgiu na Itália, em 2005, com o objetivo de criar um dispositivo para ser utilizado em projetos construídos de uma forma mais barata que outros sistemas disponíveis no mercado. O Arduino foi projetado com a finalidade de ser de fácil entendimento, programação e aplicação.

O equipamento é uma plataforma de computação física com sistemas digitais ligados a sensores e atuadores. Esses componentes permitem a construção de sistemas que percebem a realidade e responde com ações físicas desejadas. Ele é baseado em uma placa microcontrolada, com acessos de Entrada/Saída (I/O), e trabalha com bibliotecas desenvolvidas com funções para simplificar sua programação.

Como o Arduino é baseado em um microcontrolador, sendo então programável, é possível criar diversas aplicações diferentes, como a realizada neste trabalho. Os programas implementados são similares à linguagem C++. Preservando sua sintaxe na declaração de variáveis, nos operadores, ponteiros, vetores e em muitas outras características da linguagem.

O Arduino Uno pode ser alimentado por uma conexão USB ou por uma fonte de alimentação externa, como mostrado na Figura 1, selecionando automaticamente qual será usada. Para a alimentação externa pode-se utilizar um adaptador CA para CC ou uma bateria, que deve ser ligado à um conector de 2,1 mm com o positivo no centro.

Figura 1 - Alimentação Arduino Uno



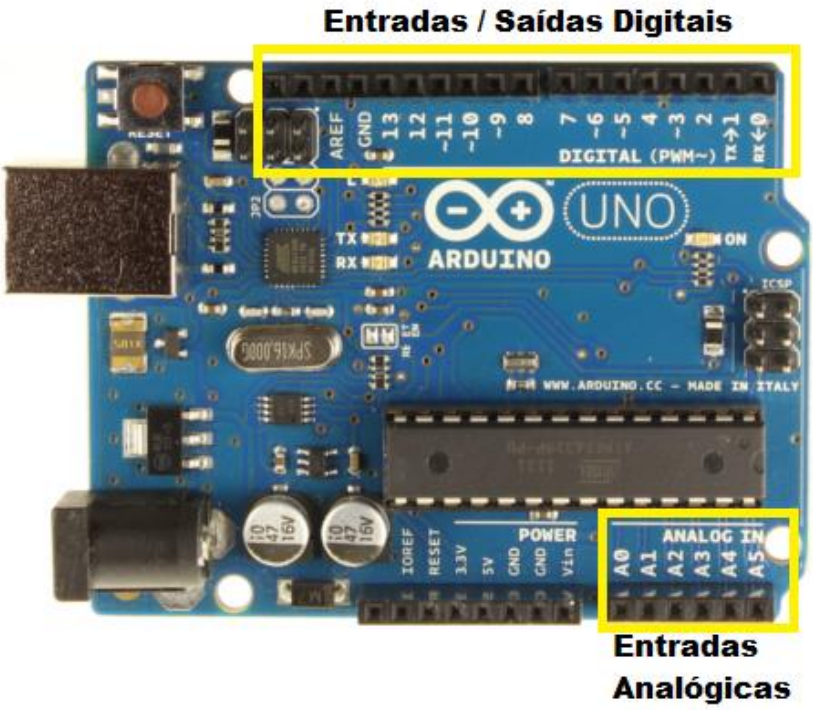
Fonte: (http://www.embarcados.com.br/arduino-uno/)

A alimentação externa pode ser feita com tensões de 6 a 20 volts. No entanto se a alimentação for inferior a 7 V, o pino 5 V pode fornecer menos que a tensão nominal e a placa pode se mostrar instável. E se a alimentação for maior do que 12 V o regulador de voltagem pode superaquecer e danificar a placa.

A placa apresenta uma memória ATmega328 que tem 32 kB (dos quais 0,5 são utilizados na inicialização). Também tem 2 kB de SRAM e 1 kB de EEPROM (que pode ser lido ou gravado com a biblioteca EEPROM).

Cada um dos 14 pinos digitais do Arduino Uno podem ser utilizados como uma entrada ou uma saída. Eles operam a 5 V sendo que cada pino pode fornecer ou receber um máximo de 40 mA. A placa possui também 6 entradas analógicas, etiquetadas de A0 a A5, cada uma tem 10 bits de resolução (i.e., 1024 valores diferentes). Por padrão elas medem de 0 a 5 V. A Figura 2 destaca as entradas do Arduino Uno.

Figura 2 - Destaque das Entradas do Arduino Uno



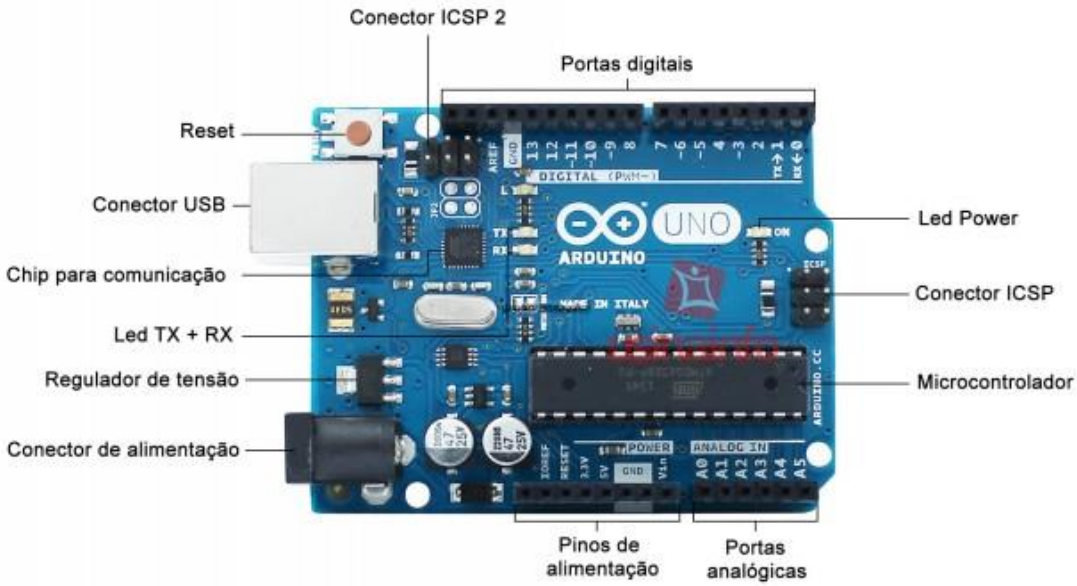
Fonte: (http://www.embarcados.com.br/arduino-uno/)

Segundo o site oficial do produto (ARDUINO, 2014), as especificações são:

- Microcontrolador: ATmega328
- Tensão de operação: 5 V
- Tensão de entrada (recomendada): 7-12 V
- Tensão de entrada (limite): 6-20 V
- Pinos de entrada/saída digitais: 14 (6 podem fornecer saídas PWM)
- Pinos de entrada analógica: 6
- Corrente DC por pino I/O: 40 mA
- Corrente DC por pino 3,3 V: 50 mA
- Memória Flash: 32 kB (ATmega328) dos quais 0,5 kB é usado para inicialização
- SRAM: 2 kB (ATmega328)
- EEPROM: 1 kB (ATmega328)
- Frequência do Clock: 16 MHz

Na Figura 3 é possível visualizar a posição de todos os componentes da placa.

Figura 3 - Localização dos Componentes do Arduino Uno



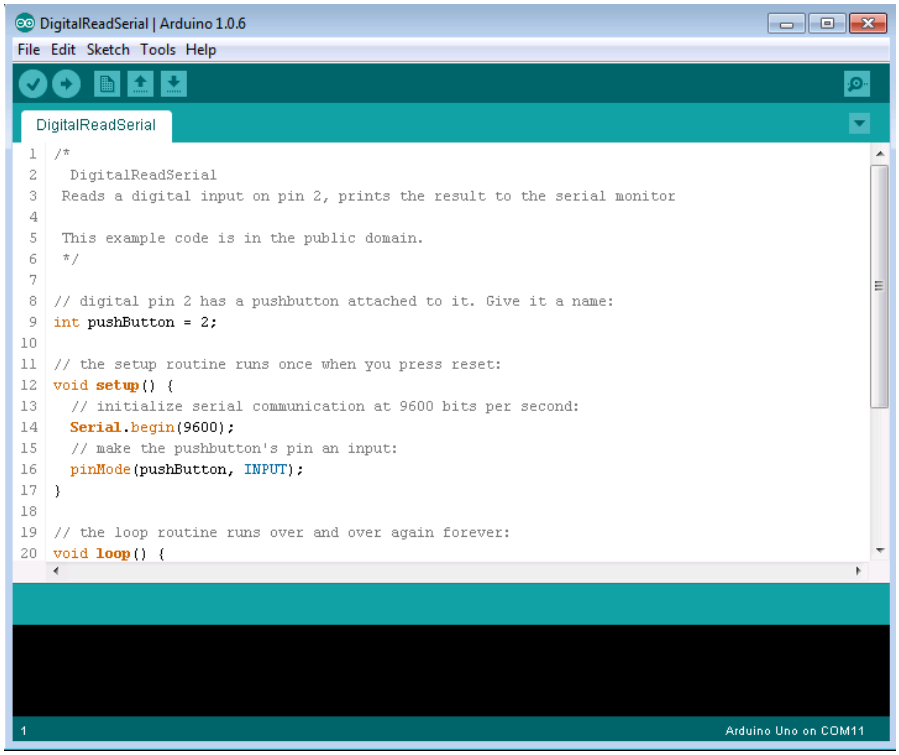
Fonte: (http://www.equibancada.com.br/produto/Arduino-UNO.html)

2.2 INTERFACE DE PROGRAMAÇÃO

A fácil programação é também uma característica do Arduino. No site do desenvolvedor é possível encontrar o programa Arduino IDE que faz a comunicação entre o computador e o microcontrolador ATmega328. A conexão é feita através de um cabo USB e, como já citado no tópico 2.1, a linguagem de programação utilizada é C++.

Na Figura 4 é mostrada a interface de programação do Arduino. Além da compilação do programa, o Arduino IDE permite a inclusão de bibliotecas e alguns exemplos prontos de aplicações.

Figura 4 - Interface de Programação Arduino IDE



Fonte: O autor.

2.3 MOTORES DE PASSO

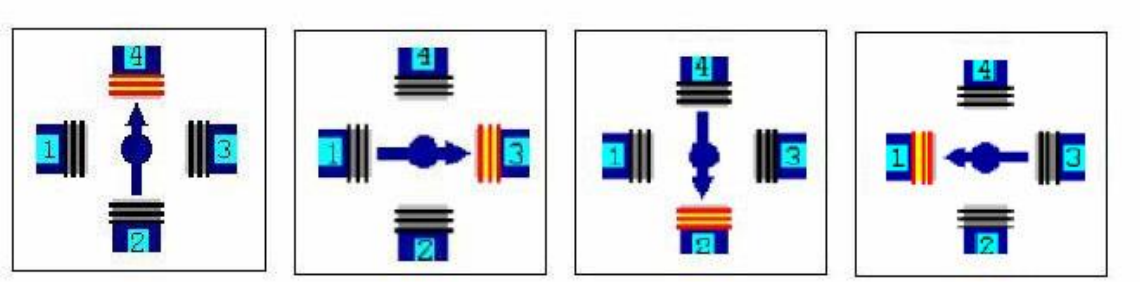
Motores de passo são dispositivos que transformam uma grandeza elétrica em movimento de rotação. Esses motores são caracterizados pela possibilidade de controlar precisamente a posição de seu eixo, através de pulsos elétricos aplicados sequencialmente em seus terminais. Por serem tão precisos são aplicados geralmente em impressoras, scanners, equipamentos cirúrgicos ou em robôs industriais.

Eles não utilizam escovas ou comutadores, como em outros motores, isso faz com que sua manutenção seja quase nula por não terem componentes que desgastam com o uso. Eles têm um número fixo de polos magnéticos que dividem uma volta completa em diversos passos. A quantidade de passos é dada pelo número de alinhamentos possíveis entre o rotor e as bobinas. Então quanto maior o número de bobinas, maior é o número de passos. Um pulso de energia é enviado sequencialmente em cada bobina fazendo com que o motor gire passo por passo. Esses motores giram com uma determinada velocidade independentemente da carga, desde que esta não exceda o torque.

A Figura 5 é um exemplo de um motor de passo de imãs permanentes de quatro bobinas que são alimentadas individualmente. A seta central é um imã e ao se energizar a bobina 4, é

criada uma atração magnética entre eles. Para dar um passo, deve-se desligar a bobina 4 e ligar a bobina 3, assim o imã é atraído e ocorre uma rotação. A mesma coisa ocorre quando se desliga a bobina 3 e liga a bobina 2, o imã continua a rotação e assim por diante.

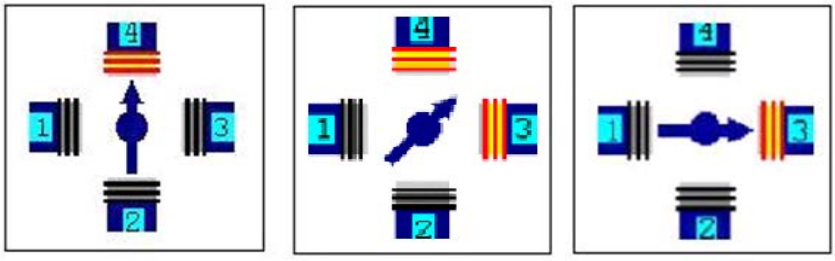
Figura 5 - Funcionamento motor de passo



Fonte: O autor.

Esta é a configuração mais simples para o funcionamento de um motor de passo, porém sua ligação também pode ser realizada de outras maneiras, como por meio passo. Neste caso, diferentemente do exemplo anterior que era ligado apenas uma bobina de cada vez, a ligação por meio passo energiza as bobinas conforme a Figura 6.

Figura 6 - Motor de passo (Meio passo)



Fonte: O autor.

Primeiramente é energizada a bobina 4, o próximo passo é dado ligando-se a bobina 4 e 3 ao mesmo tempo, assim o imã gira até o ponto físico intermediário entre as duas. Depois desse processo, desliga-se a bobina 4 e somente a bobina 3 permanece ligada, fazendo com que o imã continue a rotação. Como é possível notar, esta configuração dobra o número de passos que o motor é capaz de dar a cada volta, aumentando então sua precisão.

Para que o circuito seja alimentado sequencialmente de forma que o motor gire como o esperado, é necessário um driver que fará todo este processo. No caso deste trabalho, para

operar juntamente com o Arduino, utiliza-se um circuito integrado chamado Easy Driver, que é abordado no próximo tópico.

Existem ainda outras configurações de ativação e outros modelos de motores de passos, como o motor de relutância variável e o motor de passo híbrido, que não são abordados neste trabalho.

2.4 EASY DRIVER

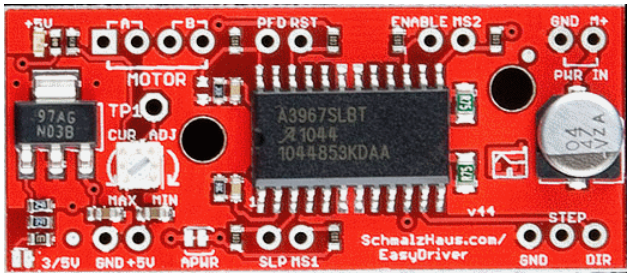
Para o funcionamento dos motores de passo é necessário um driver que faça a alimentação sequencial das bobinas. Este driver pode ir do mais simples para os casos em que a rotação do motor é feita em passo completo ou meio passo, ou mais complexo, para os casos em que se queira ter ainda mais passos entre uma bobina e outra.

Neste trabalho utilizou-se o Easy Driver, que é um circuito integrado que já contém o driver necessário para o controle do motor. O Arduino não pode ser ligado diretamente nos motores de passo, pois cada porta de saída suporta apenas 40 mA de corrente, e a bobina do motor precisa de um valor maior para o funcionamento adequado. A saída do Arduino é conectada ao driver e este é ligado diretamente no motor, funcionando como uma interface de potência entre os dois.

Uma grande vantagem ao utilizar o Easy Driver é que são necessárias apenas duas portas de saída do Arduino para o funcionamento, sendo necessário informar apenas a direção do movimento e o número de passos desejado, e o driver faz a alimentação sequencial das bobinas para a rotação do motor. E se tratando do Arduino Uno, componente utilizado neste trabalho, quanto menos portas puderem ser utilizadas melhor, pois elas não são muitas.

De acordo com o desenvolvedor (BRIAN SCHMALZ, 2014) o Easy Driver necessita de uma alimentação entre 7 V e 30 V e pode fornecer cerca de 750 mA por fase de um motor de passo. O padrão deste driver é a utilização de 8 micropassos por passo, ou seja, se o motor tem 200 passos por rotação completa, nesta configuração ele daria 1.600 passos para dar uma volta inteira. Mas não necessariamente precisa ser usado desta maneira. Ao ligar os pinos MS1 e MS2, que podem ser identificados na Figura 7, ao GND esta configuração pode mudar para 1/8, ¼ ou ½ do micropasso.

Figura 7 - Easy Driver



Fonte: (http://www.schmalzhaus.com/)

Na tabela 1 é mostrado como podem ser ligados MS1 e MS2 para a obtenção do passo desejado.

Tabela 1 – Configuração para definição do número de passos do Easy Driver

Definição do número de passos utilizando o Easy Driver		
MS1	**MS2**	**Passo**
Low	Low	1
High	Low	½
Low	High	¼
High	High	1/8

Fonte: (http://www.schmalzhaus.com/)

2.5 SENSOR DE NÍVEL

Sensores são dispositivos sensíveis a algum tipo de energia do ambiente, eles detectam uma mudança no meio e transformam esta informação em dados que possam ser processados por um microcontrolador. Existem diversos tipos de sensores, como de luz, som, temperatura, radiação e etc., mas o utilizado neste projeto é um sensor de nível, que pode ser usado para identificar a altura de um líquido em um reservatório. Este sensor funciona como uma boia, quando o nível do líquido atinge o ponto em que o sensor foi inserido, um contato ON/OFF é ativado ou desativado e assim é possível identificar a altura na qual está. Ele pode ser usado para identificar, por exemplo, quando o tanque está muito cheio ou muito vazio, o que em um processo industrial pode significar um defeito, ou algum perigo. Dependendo da aplicação podem ser usados vários deste sensor, assim é possível identificar com maior precisão a altura

que o fluido medido está. No entanto neste projeto é utilizado apenas um, que identifica quando o recipiente estiver vazio.

3 MATERIAIS E MÉTODOS

Neste capítulo são abordados os materiais usados na montagem do equipamento, o seu funcionamento e como é utilizada a interface entre usuário e dispositivo.

Estão listados a seguir os materiais utilizados na montagem do sistema e são detalhados em seguida:

Uma placa Arduino Uno R3

Um display LCD 16x2 com teclado acoplado

Dois Easy Driver

Dois motores de passos

Um sensor de nível

Uma fonte de alimentação 12 V

Um protoboard e fios para a conexão

3.1 ARDUINO UNO R3

Por ser uma placa de fácil programação e baixo custo, o Arduino Uno, mostrado na Figura 8, foi a escolha mais acertada para o desenvolvimento deste trabalho. Existem outras placas Arduino no mercado, com mais portas de entradas e saídas, mas para este projeto o número de portas disponíveis no Uno são suficientes.

Figura 8 – Arduino Uno

Fonte: (http://arduino.cc/)

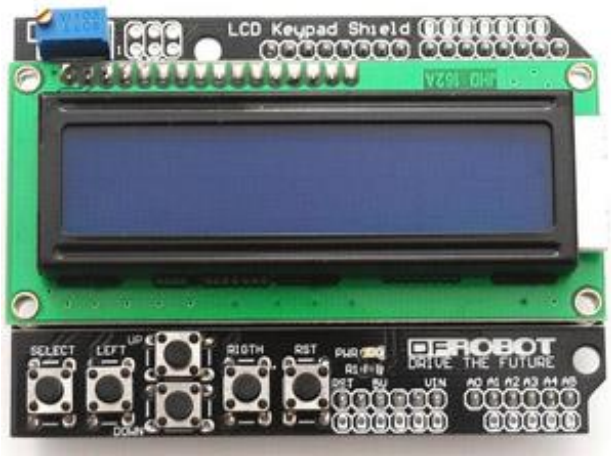
3.2 DISPLAY LCD 16X2 COM TECLADO ACOPLADO

Para a leitura das informações fornecidas pelo sistema, utilizou-se um display lcd 16x2 e devido à necessidade de teclas para inserção de dados, selecionou-se um mostrador com teclado acoplado, como o da Figura 9. Se o teclado fosse um equipamento independente seriam necessárias conexões extras para o seu funcionamento. Como este já é acoplado ao display, estas conexões já existem, porém de forma muito mais compacta.

O teclado é composto por seis botões, sendo eles: cima, baixo, esquerda, direita, select e reset. A transmissão do sinal do botão pressionado é realizada através de um pino analógico, que é ligado à entrada analógica do Arduino. Cada botão do teclado apresenta uma resistência diferente e desta forma, juntamente com a programação realizada, é possível identificar e atribuir funções a cada um dos botões.

Para a alteração de valores e posição do cursor, utilizam-se os botões direcionais. Para confirmação da informação inserida, o usuário deve pressionar o botão select e para o reinício do equipamento, o botão reset.

Figura 9 - LCD 16x2 com Teclado



Fonte: (http://nnm.me/blogs/pencraft/upravlenie-obogrevom-komnaty-pri-pomoshi-arduino-i-dallas18b20/)

3.3 EASY DRIVER

Para o funcionamento dos motores de passos é preciso uma alimentação sequencial em suas bobinas, como explicado no item 2.2. Para fazer esta alimentação é necessário um driver, podendo este ser implementado de diferentes maneiras. Neste trabalho é utilizado o easy driver

devido à facilidade de utilização deste equipamento e, como citado no item 2.3, por utilizar apenas duas portas de saída do Arduino. O dispositivo pode ser visualizado na Figura 7, apresentada no capítulo 2.

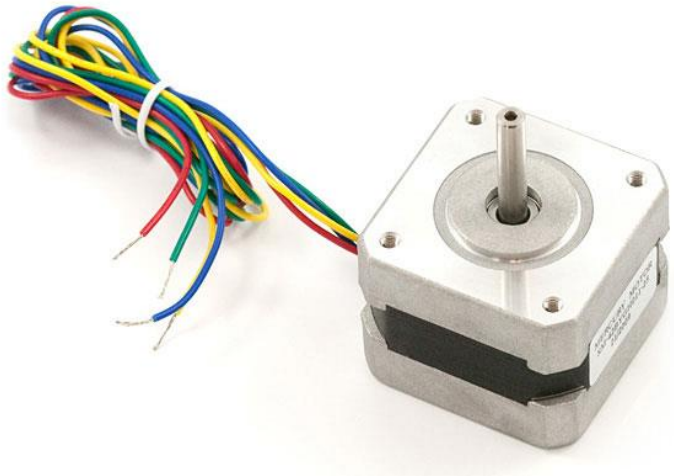
3.4 MOTOR DE PASSO

Por ser um equipamento de fácil controle, utilizou-se o motor de passo. Como o easy driver é capaz fazer a alimentação das bobinas de forma sequencial, apenas com a informação de direção e número de passos, não há dificuldade em controlar este motor. Além disto, ele tem um baixo custo e pode ser muito menor que outros tipos de motores, como motores DC.

A Figura 10 ilustra os motores utilizados neste trabalho, e suas características são:

- Voltagem operacional: 12 V
- Corrente operacional: 0,33 A
- Torque máximo: 2,3 kg/cm

Figura 10 - Motor de Passo



Fonte: (https://multilogica-shop.com/Motor_de_passo_com_cabo)

3.5 SENSOR DE NÍVEL

Como a intenção neste trabalho é identificar o nível de água em um reservatório, este é o sensor ideal para a montagem do equipamento. O sensor utilizado é do tipo boia, como mostrado na Figura 11. Quando o nível do líquido estiver baixo, a boia fecha o contato e um

sinal é enviado para o Arduino, que processa esta informação e fornece mais água ao reservatório.

Suas especificações são (COUTO, 2012):

- Tamanho do cabo: 30,5 cm
- Carga máxima: 50 W
- Tensão máxima: 100 V
- Corrente máxima: 0,5 A
- Corrente máxima de carga: 1 A
- Faixa de Temperatura: -10 ~ +85 °C

Figura 11 - Sensor de Nível

Fonte: (http://tallerarduino.com/)

3.6 DOSADOR DE COMIDA

Para o controle da quantidade de ração desenvolveu-se um equipamento que possa dosar a porção de comida fornecida, como mostrado na Figura 12. A cada giro do motor apenas uma quantidade pequena de ração passa pelo tubo, impedindo que caia além do desejado e possuindo um maior controle. O princípio deste dispositivo é o mesmo utilizado em dosadores de sementes.

Para o seu desenvolvimento é utilizado um motor de passo acoplado à um disco com uma abertura, sempre que a mesma passa pelo tubo de distribuição, uma pequena quantidade de ração é servida ao recipiente.

Figura 12 - Dosador de comida

Fonte: O autor.

3.7 FONTE DE ALIMENTAÇÃO

Para o funcionamento do easy driver é necessária uma alimentação entre 7 V e 30 V. Utiliza-se neste trabalho uma fonte universal variável, como a da Figura 13, capaz de fornecer tensões de 12 V a 24 V. Utiliza-se a tensão de 12 V, sendo que nesta configuração a fonte fornece 5,5 A de corrente máxima. Estas especificações são suficientes para o funcionamento dos motores usados sem carga. Havendo a necessidade de aumentar o torque do motor, pode-

se elevar a potência de alimentação do circuito, sendo capaz de funcionar com cargas mais pesadas.

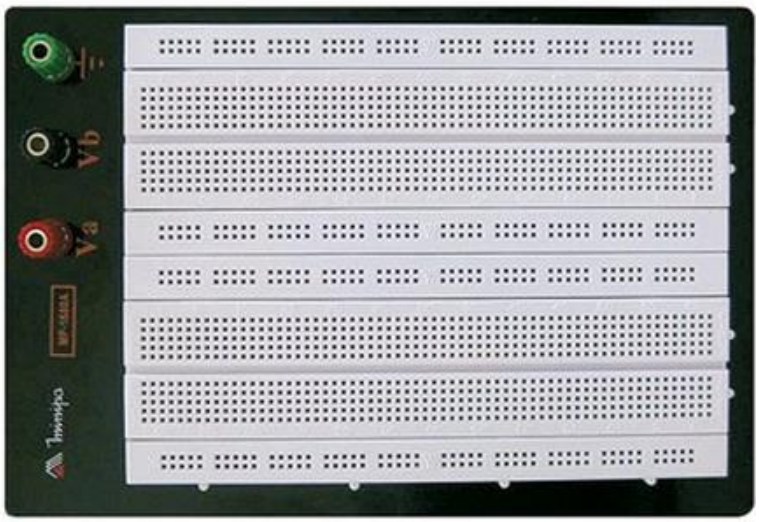
Figura 13 - Fonte de Alimentação

Fonte: (http://www.preciolandia.com/br/fonte-universal-12v-15v-16v-18v-19v-20v-7e9i9j-a.html)

3.8 MONTAGEM DO SISTEMA E CIRCUITOS

Utilizando todos os componentes descritos anteriormente, é feita a montagem do sistema em um protoboard, de modelo igual ao da Figura 14. Um protoboard é uma placa com furos e conexões condutoras para montagem de circuitos elétricos. A vantagem na utilização desta placa é a facilidade de inserção de componentes, uma vez que não necessita soldagem. O modelo utilizado apresenta 1.680 furos, podendo operar com tensão máxima de 250 V e corrente máxima de 3 A.

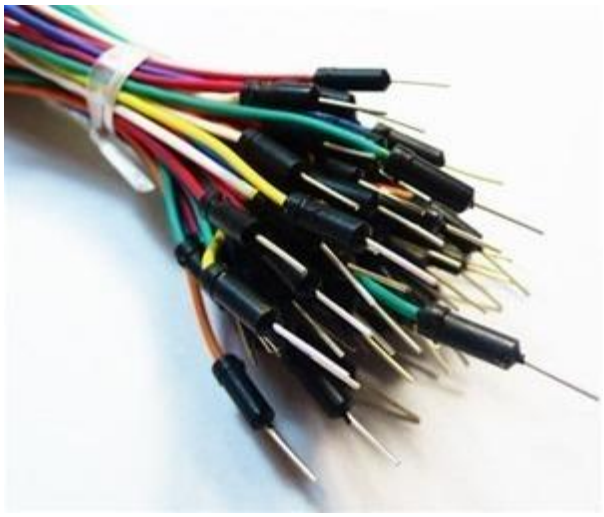
Figura 14 - Protoboard



Fonte: (http://www.tecnoferramentas.com.br/protoboard-1680-furos-com-base-minipa-mp_1680a/p?idsku=2005677&gclid=CjwKEAjwmfKpBRC8tb3Mh5rs23ASJACWy1QPX7nSzNkayY03pzkrVQ1kWxa5fdhjGYUsekPndzUgbhoC6SLw_wcB)

Para a montagem do circuito utilizou-se além do protoboard, alguns jumpers de 110, 149, 200 e 240 mm. Eles garantem facilidade e organização no processo de montagem do equipamento. Como é mostrado na Figura 15, os fios apresentam um conector em suas extremidades, eliminando a necessidade de cortar fios e diminuindo a possibilidade de interferências.

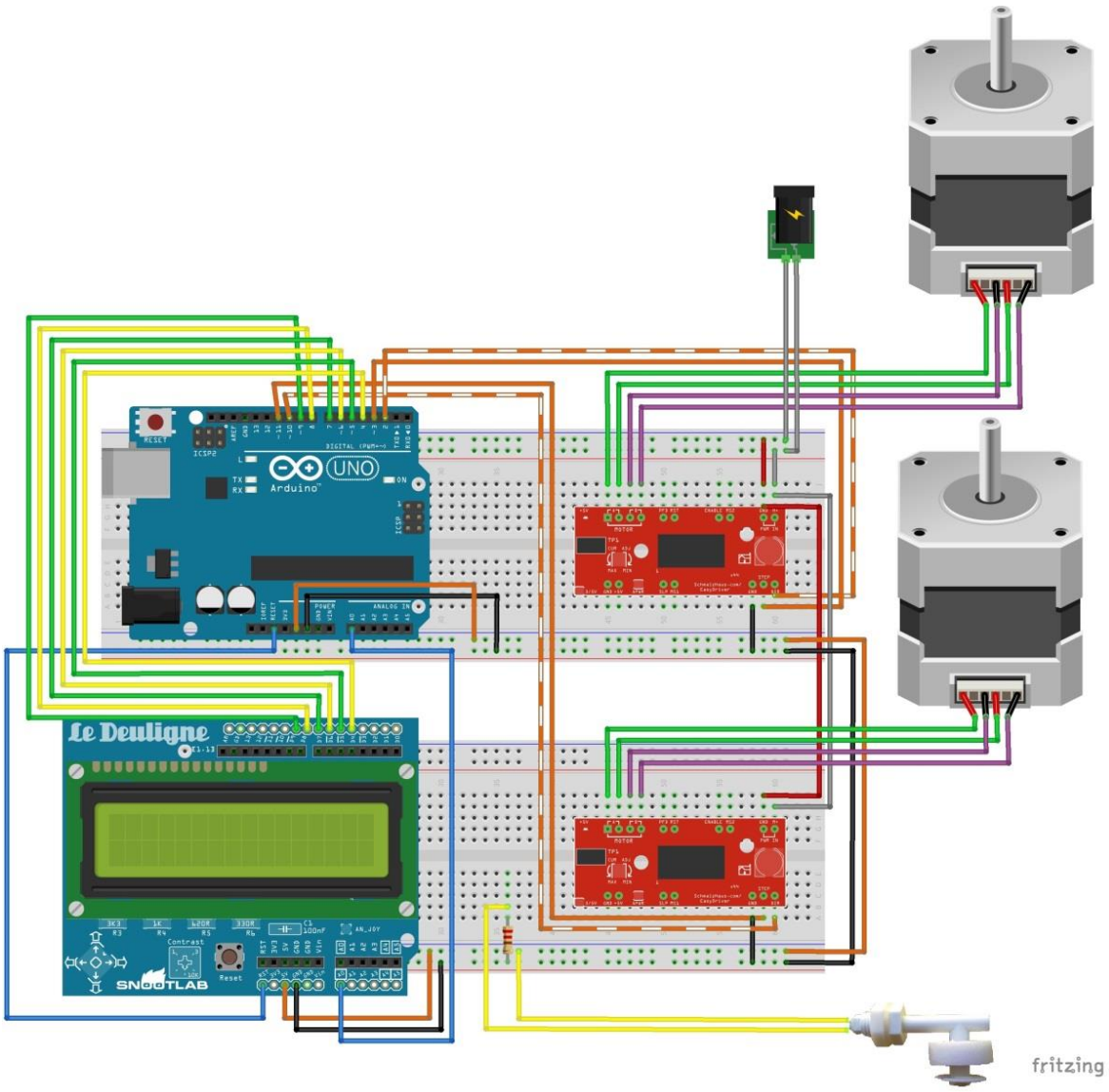
Figura 15 - Fios utilizados para a montagem do circuito



Fonte: (http://www.mercadolivre.com.br/)

O esquemático do circuito é apresentado na Figura 16. A ilustração é desenvolvida utilizando o software Fritzing.

Figura 16 - Esquemático do circuito



Fonte: O autor.

4 RESULTADOS

Neste capítulo são apresentados todos os resultados obtidos na criação do projeto. No item 4.1 resumida a programação feita para o Arduino. No item 4.2 é apresentado o protótipo e nos itens 4.3 e 4.4 é apresentado o funcionamento do equipamento, desde o momento em que é ligado, mostrando suas mensagens de inicialização, até o fornecimento do alimento e água, que é o objetivo final. Portanto ao longo deste capítulo são apresentadas várias imagens do dispositivo, para um melhor entendimento do processo.

4.1 SOFTWARE DESENVOLVIDO

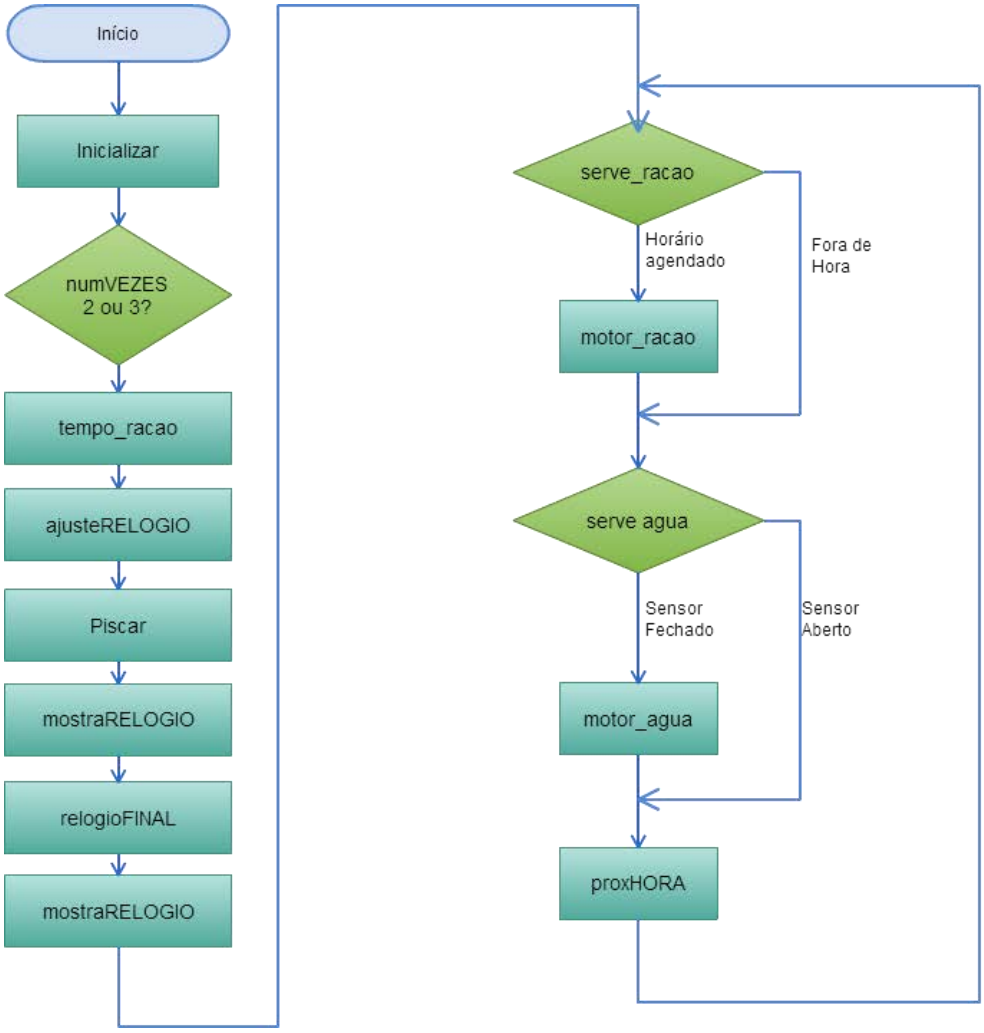
Para que o Arduino possa cumprir o que é desejado é necessário um código de programação em linguagem C. O código criado para este projeto é salvo na placa e inicia-se automaticamente quando o equipamento é ligado. Após a inicialização, o programa aguarda um sinal do teclado com as informações que são salvas. Após todos os dados inseridos, o relógio é iniciado e então não é mais necessário fornecer nenhuma informação. O programa foi feito para automaticamente comparar a hora atual com os horários inseridos na agenda, e quando estes são iguais ele liga o motor do dosador de comida.

O Arduino faz continuamente a leitura do sensor de nível para se necessário ligar o motor do fornecimento de água. Quando o nível for baixo o motor é ligado e o recipiente reabastecido.

Na Figura 17 é apresentado o fluxograma da programação. Cada bloco do diagrama representa uma função do software que é detalhado posteriormente.

O código fonte do programa desenvolvido para ser executado sobre a plataforma Arduino pode ser visto integralmente no Apêndice A.

Figura 17 - Diagrama de Blocos da Programação

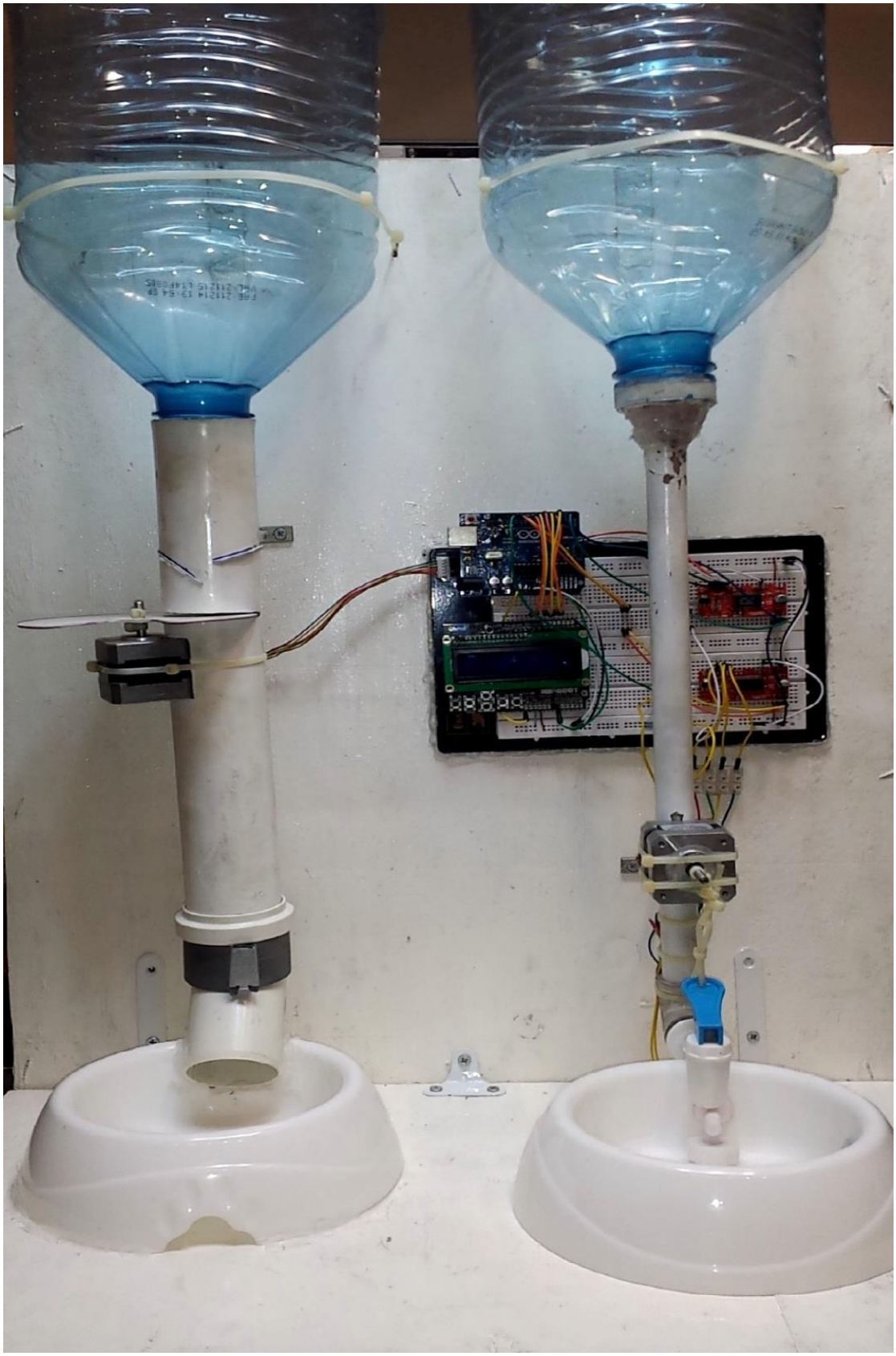


Fonte: elaborado pelo autor através do software disponível em (https://cacoo.com/lang/pt_br/sample?ref=header)

4.2 EQUIPAMENTO

O protótipo do equipamento pode ser visto na Figura 18, sua composição é feita por dois motores de passo, pelo modulo Arduino, e alguns materiais para armazenamento e transporte de ração e água.

Figura 18 - Protótipo do equipamento

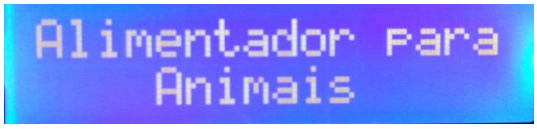


Fonte: O autor.

4.3 CONFIGURAÇÃO DOS HORÁRIOS DE RAÇÃO

Ao ligar o Arduino, o programa é carregado e começa a funcionar automaticamente. A primeira informação fornecida pelo equipamento é uma mensagem de inicialização, identificando a função do produto, como mostrado na Figura 19.

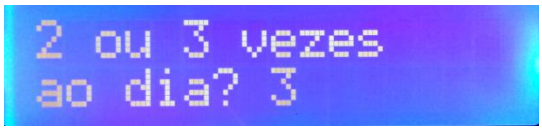
Figura 19 - Mensagem de inicialização do equipamento



Fonte: O autor.

Posteriormente inicia-se a interação entre o usuário e o equipamento, onde é necessária a inserção de informações para a sequência de seu funcionamento. Como mostrado na Figura 20, o usuário deve escolher se ele deseja que seja distribuído ração duas ou três vezes ao dia.

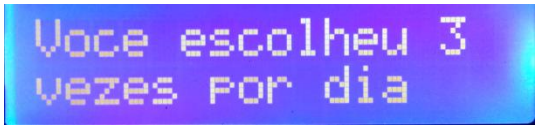
Figura 20 - Escolha de quantidade de vezes ao dia



Fonte: O autor.

Nesta etapa, como em todas as seguintes, para alterar os valores de escolha, o usuário deve utilizar o teclado, sendo que os botões para cima e para baixo aumentam e diminuem o valor a ser modificado, e os botões de direita e esquerda variam a posição do cursor. A informação selecionada estará piscando, garantindo que não haja confusão na definição dos valores. Após a inserção dos dados deve-se apertar o botão select para passar para a próxima informação a ser gravada. Ainda no momento da escolha entre duas ou três vezes ao dia, ao pressionar o botão select uma mensagem confirmando a escolha é enviada ao usuário, como mostrado na Figura 21.

Figura 21 - Confirmação da escolha



Fonte: O autor.

Em seguida o usuário deve informar os horários nos quais quer que o dispositivo forneça comida. A Figura 22 mostra a mensagem apresentada no display, aguardando que os valores desejados sejam inseridos.

Figura 22 - Ajuste dos horários da agenda

Fonte: O autor.

O usuário escolhe a hora, minuto e segundo que deseja que o dispositivo opere. Após a escolha e confirmação do primeiro horário, o display mostra a mensagem “Ajuste 2ª Hora:” e deve ser escolhido o segundo horário desejado. Se a opção inicial foi de duas vezes ao dia, o programa avança diretamente para a próxima etapa. Se a escolha foi de três vezes ao dia, aparece uma terceira opção de horário, “Ajuste 3ª Hora”, e após a definição, o programa continua para a próxima informação.

Neste ponto é ajustado o relógio com o horário real, como mostra a Figura 23. E após a confirmação, uma mensagem é mostrada no display do dispositivo dizendo “Horário Definido” e a hora escolhida.

Figura 23 - Ajuste da hora atual

Fonte: O autor.

A partir deste ponto, não é mais necessária nenhuma informação, tendo apenas que verificar se o reservatório de ração está cheio para o correto funcionamento do dispositivo. O display permanece sempre ligado mostrando o relógio, como na Figura 24.

Figura 24 - Display mostrando o relógio

Fonte: O autor.

É possível conferir os horários da agenda mesmo com o dispositivo em funcionamento, como na Figura 25. Ao apertar-se os botões da direita ou esquerda do teclado, o display fornece os horários que foram definidos anteriormente. Continuando a aperta-los é possível ver os outros horários escolhidos. Após alguns segundos o campo da agenda apaga, mostrando somente o relógio no visor.

Figura 25 - Display mostrando o relógio e a agenda

Fonte: O autor.

4.4 FUNCIONAMENTO DO FORNECIMENTO DE ÁGUA

O fornecimento de água é feito automaticamente através de um sensor. O sistema identifica quando o nível do recipiente está vazio e assim, sem precisar ajustar nenhum horário ou outra informação a água é fornecida.

Ao atingir o nível baixo o contato é fechado e um sinal é enviado ao Arduino. O motor de passo é ligado fazendo com que o mecanismo da água seja aberto durante o tempo necessário para encher completamente o recipiente. Após este tempo, o motor de passo é ligado novamente para fechar o mecanismo, assim encerrando o processo.

5 CONCLUSÃO

O protótipo montado teve sucesso em testes realizados, sendo que tanto o hardware quanto o software se mostraram funcionando corretamente. O sistema é capaz de ativar ambos os motores quando necessário, ou seja, quando falta água ou quando o horário corresponde ao programado.

De acordo com informações coletadas em hotéis especializados em animais na cidade de São Paulo, a diária para um cachorro de médio porte é em média R$75,00. Para a montagem do equipamento gastou-se aproximadamente R$300,00. Nota-se o benefício financeiro do dispositivo, considerando que seu valor equivale a apenas quatro dias de hospedagem, porém podendo ser utilizado frequentemente.

A partir dos resultados, conclui-se que foi possível realizar o objetivo deste projeto, automatizando o processo de alimentação de um animal doméstico. Os equipamentos utilizados responderam de forma satisfatória. A plataforma Arduino Uno mostrou-se ser uma ferramenta de fácil utilização e com uma boa relação custo-benefício para o controle de periféricos.

No geral pode-se concluir que os benefícios gerados pelo equipamento apontam um cenário de possibilidades que pode ser expandido a outros segmentos. Com a utilização do módulo Arduino fica fácil fazer o controle de outros sistemas, como por exemplo, sistema de alarme, controle de temperatura, automação residencial, monitoramento de crianças ou idosos, entre outros, exemplificando assim a capacidade do equipamento de ser utilizado para diversos objetivos, mas ainda mantendo a simplicidade na programação.

Sugestões de trabalho futuro são a inclusão de uma interface homem máquina via aplicativo de celular, na qual o usuário consiga programar ou reprogramar o dispositivo de qualquer lugar com acesso à internet e a incorporação de uma câmera para visualização dos animais.

REFERÊNCIAS

- ARDUINO. **Arduino uno.** Disponível em: <http://www.arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 01 set. 2014.

- SCHMALZ, B. **Easy driver stepper motor driver:** an open source hardware stepper motor drive project. Disponível em: <http://www.schmalzhaus.com/EasyDriver/index.html>. Acesso em: 01 set. 2014.

- CACOO. **Fluxograma.** Disponível em: <https://cacoo.com/lang/pt_br/sample?ref=header>. Acesso em: 25 abr. 2015.

- COUTO, J. **Sensor de nivel de liquido y arduino o pinguino pic.** 2012. Disponível em: <http://tallerarduino.com/2012/10/26/sensor-de-nivel-de-liquido-y-arduino-o-pinguino-pic/>. Acesso em: 24 dez. 2014.

- MULTILÓGICA SHOP. **Arduino uno R3.** Disponível em: <https://multilogica-shop.com/Arduino-Uno>. Acesso em: 25 abr. 2015.

- SOUZA, F. **Arduino uno.** Disponível em: <http://www.embarcados.com.br/arduino-uno/>. Acesso em: 25 abr. 2015.

BIBLIOGRAFIA CONSULTADA

- ARDUINO E CIA. **Shield LCD 16x2 com keypad.** Disponível em: <http://www.arduinoecia.com.br/2013/08/shield-lcd-16x2-com-keypad.html>. Acesso em: 30 dez. 2014.

- BRITES, F. G.; SANTOS, Vinicius P. A. **Motor de passo.** 2008. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/stepmotor/stepmotor2k81119.pdf>. Acesso em: 01 set. 2014.

- EVANS, B. **Beginning arduino programming.** New York: Apress, 2012.

- FRITZING. **Fritzing.** Disponível em: <http://fritzing.org/learning/>. Acesso em: 01 set. 2014.

- MCROBERTS, M. **Arduino básico.** Brasil: Novatec, 2011.

- RENNA, R. B. di et al**. Introdução ao kit de desenvolvimento arduino.** 2013. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/arduino/Tut_Arduino.pdf>. Acesso em: 25 jan. 2015.

- SANTOS, N. P. **Arduino:** introdução e recursos avançados. 2009. Disponível em: <http://www.novaims.unl.pt/docentes/vlobo/escola_naval/MFC/Tutorial Arduino.pdf>. Acesso em: 25 ago. 2014.

- TOCCI, R. J.; WIDMER, N. S. **Sistemas digitais:** princípios e aplicações. 8. ed. Brasil: Pearson/prentice Hall, 2003.

APÊNDICE A – Código Fonte

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7); //Iniciação do LCD
int vezes=2, seg=0, min=0,hor=0, x=0, y=0, z=0, proxH=0, cont;
int r=0, seg_agua5=0;
int seg_agua=0;

int direcao=0;
int passos_motor = 1;
int Cursor=0;
int Linha=0;
int A=0;


int pino_passo_racao = 3,          // Pino Passos motor ração
pino_direcao_racao = 2;            // Pino Direção motor ração

int pino_passo_agua = 11,          // Pino Passos motor água
pino_direcao_agua = 10;            // Pino Direção motor água

int sensor = 12;                   // Pino Sensor de nível



int set[] = { hor, min, seg };     // Parâmetros relógio
int hora_racao[3];                 // Vetor Horas para ração
int minuto_racao[3];               // Vetor Minutos para ração
int seg_racao[3];                  // Vetor Segundos para Ração
int thisTime = 0;                  // Posição do Cursor


void setup()
{
  Serial.begin(9600);
  pinMode(pino_passo_racao, OUTPUT);
  pinMode(pino_direcao_racao, OUTPUT);
  pinMode(pino_passo_agua, OUTPUT);
  pinMode(pino_direcao_agua, OUTPUT);
  pinMode(sensor, INPUT);

  inicializar();          // Mensagem de inicialização do programa
  numVEZES();             // Ajuste entre 2 ou 3 vezes ao dia
  tempo_racao();          // Ajuste dos horários do funcionamento do motor da ração
```

```
}

void loop()
{
  if (x==0) {
// x==0 indica que o botão select ainda não foi pressionado para confirmação da hora
atual, quando o usuário apertar select, x=1, nao realizando mais este passo.

    lcd.setCursor(0,0);
    lcd.print("Ajuste a hora");
    lcd.setCursor(0,1);
    lcd.print("atual:");
    Cursor=7;
    Linha=1;
    ajusteRELOGIO();
// Função que reconhece qual botão foi pressionado para o ajuste do relogio
    piscar();
// Pisca os digitos onde o cursor está selecionado

  }
  else{

    while (z==0) {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Horario Definido");
// Mensagem de confirmação da escolha da hora
      Cursor=4;
      Linha=1;
      mostraRELOGIO();
// Mostra o relógio no LCD
      delay(3000);
      z++;
      lcd.clear();
    }
    lcd.setCursor(0,0);

    lcd.print("Hora: ");
    Cursor=6;
    Linha=0;
    relogioFINAL();                 // Função para o funcionamento do relógio
    mostraRELOGIO();                // Mostra o relógio digital no LCD
    serve_racao();
// Compara a hora atual com a hora agendada, se for igual liga motor ração
    serve_agua();
```

```
// Faz a leitura do sensor de nível, se for baixo liga motor da água
    proxHORA();                      // Mostra os horários da agenda
  }
}

//--------------Inicializar---------------------//
void inicializar() {
  lcd.begin(16, 2);
                                     //
  lcd.setCursor(0,0);                //
  lcd.print("Alimentador para");     // Mensagem de Inicialização do Equipamento
  lcd.setCursor(4,1);                //
  lcd.print("Animais");              //
  delay(3000);
  lcd.clear();
}
//--------------Inicializar---------------------//

//--------------tempo_racao---------------------//
void tempo_racao() {
// Função responsável por receber os horários para serem armazenados na agenda

  lcd.setCursor(0,0);
  lcd.print("Defina o Horario");
  lcd.setCursor(0,1);
  lcd.print("da Racao:");
  delay(3000);
  lcd.clear();

  Cursor=4;
  Linha=1;

  while (x==0) {
    lcd.setCursor(0,0);
    lcd.print("Ajuste 1a Hora:");
    Cursor=4;
    Linha=1;
    limite();
// Limite do relogio (60 seg, 60 min, 24 horas)
    ajusteRELOGIO();
// Função que reconhece qual botão foi pressionado para o ajuste do relógio
    piscar();
// Pisca os dígitos onde o cursor está selecionado
    hora_racao[0]= set[0];           // Define a 1ª hora da agenda
    minuto_racao[0]= set[1];         // Define o 1° Minuto da agenda
```

```
    seg_racao[0]= set[2];             // Define o 1° Segundo da agenda
  }
  x=0;
  thisTime=0;
  delay(1000);
  while (x==0) {
    lcd.setCursor(0,0);
    lcd.print("Ajuste 2a Hora:");
    Cursor=4;
    Linha=1;
    limite();                         // Limite do relógio
    ajusteRELOGIO();
// Função que reconhece qual botão foi pressionado para o ajuste do relógio
    piscar();
// Pisca os dígitos onde o cursor está selecionado
    hora_racao[1]= set[0];            // Define a 2ª hora da agenda
    minuto_racao[1]= set[1];          // Define o 2° Minuto da agenda
    seg_racao[1]= set[2];             // Define o 2° Segundo da agenda
  }
  x=0;
  thisTime=0;
  delay(1000);

  if (vezes==3){
    while (x==0) {
      lcd.setCursor(0,0);
      lcd.print("Ajuste 3a Hora:");
      Cursor=4;
      Linha=1;
      limite();                       // Limite do relógio
      ajusteRELOGIO();
// Função que reconhece qual botão foi pressionado para o ajuste do relógio
      piscar();
// Pisca os dígitos onde o cursor está selecionado
      hora_racao[2]= set[0];          // Define a 3ª hora da agenda
      minuto_racao[2]= set[1];        // Define o 3° Minuto da agenda
      seg_racao[2]= set[2];           // Define o 3° Segundo da agenda
    }
    x=0;
    thisTime=0;
    delay(1000);
  }
  zera_relogio();
  lcd.clear();
}
```

```
//--------------tempo_racao---------------------//

//--------------serve_racao-------------------//
void serve_racao() {

// Função que compara o horário real com os horários da agenda. Caso iguais, liga
motor de ração

  if (set[0] == hora_racao[0] && set[1] == minuto_racao[0] && seg_racao[0] == set[2])
{
    for (int p=0 ; p<6; p++){
      motor_racao();}
// Liga motor ração

  }
  if (set[0] == hora_racao[1] && set[1] == minuto_racao[1] && seg_racao[1] == set[2]
){
    for (int p=0 ; p<6; p++){
      motor_racao();}
// Liga motor ração
  }
  if (vezes==3){
    if (set[0] == hora_racao[2] && set[1] == minuto_racao[2] && seg_racao[2] ==
set[2]){
    for (int p=0 ; p<6; p++){
        motor_racao();}
// Liga motor ração
    }}}
//----------------serve racao----------------------//

//---------------serve agua-------------------------//
void serve_agua() {
// Função que faz a leitura do sensor de nível caso o nível esteja baixo, liga motor
de água.
    if(A==0){

    if (digitalRead (sensor)==LOW){            // Leitura digital do sensor
    A=1;
    seg_agua = set[2];
    if (seg_agua >= 55) {
    seg_agua5 = (5 - (60 - seg_agua));
   }
   if (seg_agua < 50){
    seg_agua5 = (seg_agua + 5);
   }
```

```
    // Compensação caso os segundos estejam acima de 55 segundos, para não ocorrer
erro.

    for (int p=0 ; p<2; p++){
      direcao=0;
      motor_agua();                              // Liga motor água
    }
   r=1;
  }}
  if (r==1){
    if (set[2] == seg_agua5){
// Após 5 segundos faz motor fechar o dispositivo
    for (int p=0 ; p<2; p++){
     direcao=1;
     motor_agua();
     A=0;
     }}}
}
//---------------serve agua-------------------------//

//---------------Contador relogio Final-----------//
void relogioFINAL() {
//Função do contador dos segundos do relógio

  static unsigned long ult_tempo = 0;
  int tempo = millis();
  if(tempo - ult_tempo >= 1000) {
    ult_tempo = tempo;
    set[2]++;
  }
  limite();
}
//---------------Contador Relogio Final-----------//

//---------------------Limite-------------------//
void limite() {
// Função para definição do limite do relógio digital (60seg, 60min, 24hor)
  if(set[2]>=60) {               // Limite superior de 60 segundos
    set[2] = 0;
    set[1]++;
  }
  if(set[1]>=60) {               // Limite superior de 60 minutos
    set[1] = 0;
    set[0]++;
  }
```

```
  if(set[0]>=24) {               // Limite superior de 24 horas
    set[0]=0;
    set[1]=0;
  }
  if(set[2]<0) {                 // Limite inferior de 0 segundo
    set[2]=59;
    set[1]--;
  }
  if(set[1]<0) {                 // Limite inferior de 0 minuto
    set[1]=59;
    set[0]--;
  }
  if(set[0]<0) {                 // Limite inferior de 0 hora
    set[0]=23;
   }
}
//---------------------Limite------------------//

//------------ zera_relogio----------------//
void zera_relogio(){
// Zera o relógio para inserir próximo horário na agenda
  set[0]=0;
  set[1]=0;
  set[2]=0;
}
//------------ zera_relogio----------------//

//----------------Função Ajuste RELOGIO--------------------//
void ajusteRELOGIO() {

  int botao;
  botao = analogRead (0);          // Leitura do pino de entrada analógico

  if (botao < 50) {                // botão DIREITA
    thisTime++;
    delay (400);
  }
  else if (botao < 200) {          // botão CIMA
    set[thisTime]++;
    if (thisTime==2) {
      delay(130);                  // Delay curso para segundos
    }
    else delay (250);
// Delay longo para minutos e segundos para aumentar mais lentamente
  }
```

```
  else if (botao < 400) {          // botão BAIXO
    set[thisTime]--;
    if (thisTime==2) {
      delay(130);
    }
    else delay (250);
  }
  else if (botao < 600) {          // botão ESQUERDA
    thisTime--;
    delay (400);
  }
  else if (botao <800) {           // botão SELECT
    x++;
  }
  if (thisTime >2 ) {              // Limite do cursor
    thisTime = 0;
  }
  if(thisTime <0 ) {               // Limite do cursor
    thisTime = 2;
  }
  limite();
}
//----------------Função Ajuste RELOGIO--------------------//

//-------------//Função Mostra RELOGIO//-------------//
void mostraRELOGIO() {
//Função que mostra o relógio digital no formato HH:MM:SS

  if (set[0] < 10) {

    lcd.setCursor(Cursor,Linha);
    lcd.print("0");
    lcd.print (set[0]);              // HORA
  }
  else {
    lcd.setCursor(Cursor,Linha);
    lcd.print (set[0]);
  }
  lcd.print (":");
//--------------------------------------
  if (set[1] < 10) {

    lcd.print("0");
    lcd.print (set[1]);              // MINUTO
  }
```

```
  else {
    lcd.print (set[1]);
  }
  lcd.print (":");
//--------------------------------------
  if (set[2] < 10) {
    lcd.print("0");
    lcd.print (set[2]);              // SEGUNDO
  }
  else {
    lcd.print (set[2]);
  }}
//-------------//Função Mostra RELOGIO//-------------//

//----------------//Piscar//------------------//
void piscar() {
// Função para piscar os dígitos selecionados pelo cursor

  static unsigned long ult_temp1 = 0;
  int temp1 = millis();
  static unsigned long ult_temp2 = 0;
  int temp2 = millis();

  if(temp1 - ult_temp1 >= 200) {
    ult_temp1 = temp1;

    if (thisTime==0){
      lcd.setCursor(Cursor, Linha);
      lcd.print("  ");
    }
    else if (thisTime==1){
      lcd.setCursor(Cursor+3, Linha);
      lcd.print("  ");
    }
    else if (thisTime==2){
      lcd.setCursor(Cursor+6, Linha);
      lcd.print("  ");
    }
    if(temp2 - ult_temp2 >= 400) {
      ult_temp2 = temp2;
      mostraRELOGIO();
    }}}
//----------------//Piscar//------------------//

//----------------//MOTOR_RACAO//------------------//
```

```
void motor_racao() {                      // Função para ligar o motor da ração

  int p=0;
  for (int p=0 ; p < 400; p++){

    digitalWrite(pino_direcao_racao, 0);
  digitalWrite(pino_passo_racao, 1);
    delay(1);
  digitalWrite(pino_passo_racao, 0);
    delay(1);}
}
//----------------//MOTOR_RACAO//------------------//

//----------------//MOTOR_AGUA//------------------//
void motor_agua() {                        // Função para ligar o motor de água

  for (int q=0 ; q < 100; q++){

  digitalWrite(pino_direcao_agua, direcao);
  digitalWrite(pino_passo_agua, 1);
    delay(1);
  digitalWrite(pino_passo_agua, 0);
    delay(1);}
    r=0;
}
//----------------//MOTOR_AGUA//------------------//

//---------------//setVEZES//---------------//
void setVEZES() {             // Função para definir entre 2 ou 3 vezes ao dia

  int botao;
  botao = analogRead (0);     // Leitura do pino de entrada analógico

  if (botao < 50) {           // botão DIREITA
    vezes++;
    delay(400);
  }
  else if (botao < 200) {     // botão CIMA
    vezes++;
    delay (400);
  }
  else if (botao < 400) {     // botão BAIXO
    vezes--;
    delay(400);
  }
```

```
  else if (botao < 600) {      // botão ESQUERDA
    vezes--;
    delay(400);
  }
  else if (botao <800) {       // botão SELECT
    x++;
  }
  if (vezes > 3 ) {            // limite do número de vezes
    vezes = 2;
  }
  if(vezes < 2 ) {             // limite do número de vezes
    vezes = 3;
  }}
//---------------//setVEZES//---------------//

//--------------//numVEZES//---------------//
void numVEZES() {               // Função para piscar e confirmar número de vezes ao
dia

  lcd.setCursor(0,0);
  lcd.print("2 ou 3 vezes");
  lcd.setCursor(0,1);
  lcd.print("ao dia?");          // Mensagem perguntando se 2 ou 3 vezes ao dia

  while (x==0) {
    static unsigned long ult_temp1 = 0;
    int temp1 = millis();
    static unsigned long ult_temp2 = 0;
    int temp2 = millis();
    setVEZES();

    if(temp1 - ult_temp1 >= 200) {
      ult_temp1 = temp1;

      lcd.setCursor(8 , 1);
      lcd.print(" ");           // pisca cursor

      if(temp2 - ult_temp2 >= 400) {
        ult_temp2 = temp2;
        lcd.setCursor(8,1);
        lcd.print(vezes);
      }
    }
  }
  lcd.clear();
```

```
  lcd.setCursor(0,0);
  lcd.print("Voce escolheu ");
  lcd.print(vezes);
  lcd.setCursor(0,1);
  lcd.print("vezes por dia");    // Mensagem de confirmação da escolha
  delay(3000);
  x=0;
  lcd.clear();
}
//----------------numVEZES-----------------//

//---------------proxHORA-----------------//
void proxHORA(){            // Função para mostrar horários agendados

  int botao;
  botao = analogRead (0);
  if (botao < 50) {        // botão DIREITA
    proxH++;
    delay(250);
  }
  else if (botao < 200) {  // botão CIMA
  }
  else if (botao < 400) {  // botão BAIXO
  }
  else if (botao < 600) {  // botão ESQUERDA
    proxH--;
    if (proxH==0)
      proxH--;
    delay(250);
  }
  else if (botao <800) {  // botão SELECT
  }

  if (proxH != 0) {

    if (proxH > vezes ) {
      proxH = 1;
    }
    if(proxH < 1 ) {
      proxH = vezes;
    }
    if (cont != set[2]) {
      cont = set[2];
      y++;
    }
```

```
if (y>0 && y<=15){     // mostrar por 15 segundos

  lcd.setCursor(0,1);
  lcd.print("Agenda: ");
  if(hora_racao[(proxH-1)]>=10){
    lcd.print(hora_racao[(proxH-1)]);
  }
  else{
    lcd.print("0");
    lcd.print(hora_racao[(proxH-1)]);
  }
  lcd.print(":");
  if(minuto_racao[(proxH-1)]>=10){
    lcd.print(minuto_racao[(proxH-1)]);
  }
  else{
    lcd.print("0");
    lcd.print(minuto_racao[(proxH-1)]);
  }
  lcd.print(":");
  if(seg_racao[(proxH-1)]>=10){
    lcd.print(seg_racao[(proxH-1)]);
  }
  else {
    lcd.print("0");
    lcd.print(seg_racao[(proxH-1)]);
  }}
if (y>15){
  y=0;
  proxH=0;
  lcd.clear();
}}}
//---------------proxHORA-----------------//
```