Centro Federal de Educaçào Tecnológica de Minas Gerais
Campus Divinópolis
Graduação em Engenharia Mecatrônica

Lucas de Aguiar Gontijo

Desenvolvimento e Implementação de um Painel de Controle para Alimentadores Automáticos para Animais Domésticos

Divinópolis
2021

Lucas de Aguiar Gontijo

Desenvolvimento e Implementação de um Painel de Controle para Alimentadores Automáticos para Animais Domésticos

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenhario Mecatrônico.

Áreas de Integração: Eletrônica e Computação.

Orientador: Prof. Dr. Alan Mendes Marotta

Divinópolis

2021

Centro Federal de Educação Tecnológica de Minas Gerais
CEFET-MG / Divinópolis
Curso de Engenharia Mecatrônica

Monografia entitulada “Desenvolvimento e Implementação de um Painel de Controle para Alimentadores Automáticos para Animais Domésticos”, de autoria do(as) graduando(as) Lucas de Aguiar Gontijo, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Alan Mendes Marotta

Prof. Dr. Thiago Magela Rodrigues Dias

Prof. Dr. Lúcio Flávio Santos Patrício

Coordenador do Curso de Engenharia Mecatrônica

Prof. Me. Marlon Antônio Piheiro

Divinópolis

Abril de 2021

À minha família, pelo apoio incondicional nesta jornada.

Agradecimentos

Agradeço,

à minha mãe e meus irmãos por todo o apoio e confiança em mim depositados;

ao Prof. Alan por toda a paciência e empenho nesse período de orientação;

aos colegas de curso que me acompanharam por todos esses anos;

aos demais amigos que o CEFET me trouxe: professores, funcionários, alunos dos cursos técnicos;

a todos que de alguma forma contribuíram com o meu progresso.

A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo.

Albert Einstein

Resumo

Alimentar devidamente os animais de estimação pode ser uma tarefa desafiadora dentro de um dia-a-dia muito movimentado ou durante viagens por exemplo. O desafio se torna ainda maior quando se deseja regular horários e quantidades de ração que os animais comem. Com isto em mente, neste trabalho foi proposto o desenvolvimento de um painel de controle de baixo custo para dispensadores automáticos de ração. O dispositivo pode ser programado de maneira que o usuário possa escolher a quantidade e frequência das doses. O painel pode ser utilizado em diversos dispensadores de ração para diferentes animais. Como componente central do painel foi utilizada uma placa *Arduino UNO* que conta com um microcontrolador ATMEGA328P. A programação foi feita em software próprio para *Arduino* e que recebe o mesmo nome. O *software* Proteus foi utilizado para simulação e desenvolvimento do desenho da placa eletrônica. A programação do *Arduino* também foi testada no Proteus.

Palavras-chave: : Alimentador Animal, Automação Doméstica, *Arduíno*.

Abstract

To properly feed a pet can be a challenging task inside of a busy daily routine or during travels for example. The challenge gets even bigger when its necessary to regulate times and quantities of food the animal has to eat. With that in mind, in this project it was proposed to develop a low cost programmable control panel for automatic animal food dispensers. The device can be programmed to allow the user to chose the size and quantity of doses per day. The panel can be used with various food dispensers for many pet species. As central component of the panel, an *Arduino UNO* board was used that counts with an ATMEGA328P microcontroller. Its programming was done in the *Arduino* Software. The software Proteus was used for the simulations and development of the drawing of the electronic circuit. The *Arduino* programming was also tested in Proteus.

Keywords: Pet Feeder, Domestic Automation, *Arduino*.

Sumário

Lista de figuras

Lista de acrônimos e notações

EEPROM *Electrically Erasable Programmable Read-Only Memory* ou Memória de Leitura Apenas Programável Apagável Eletricamente

IDE *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado

IOT *Internet of Things* ou *Internet* das Coisas

MHz Mega *Hetz*

MIPS *Millions of Instructions Per Second* ou Milhões de Instruções por Segundo

PWM *Pulse Width Modulation* ou Modulação por Largura de Pulso

RISC *Reduced Instruction Set Computer* ou Computador com um Conjunto Reduzido de Instruções

SRAM *Static Random-access Memory* ou memória de acesso aleatório estática

USART *Universal Synchronous Asynchronous Receiver Transmitter* ou Transmissor/Receptor Universal Síncrono e Assíncrono

USB *Universal Serial Bus* ou Porta Serial Universal

Capítulo 1

Introdução

De acordo com o Instituto Pet Brasil existiam, em 2018, cerca de 139,3 milhões de animais de estimação no Brasil. Destes, 54,2 milhões eram cães, 39,8 milhões aves, 23,9 milhões de gatos, 19,1 milhões de peixes e 2,3 milhões de répteis e pequenos mamíferos. Alimentar devidamente os animais de estimação pode ser uma tarefa desafiadora dentro de um dia-a-dia muito movimentado ou durante viagens por exemplo. O desafio se torna ainda maior quando se deseja regular horários e quantidades de ração que os animais comem.

Assim, muitos modelos de equipamentos para realizar esta tarefa surgiram no mercado. Muitos destes utilizam medidas fixas para as doses ou são limitados a uma pequena quantidade de doses. Portanto o que se propõe é desenvolver um painel de controle de baixo custo e que possa comandar o alimentador desejado, podendo servir a animais de todos os portes. Propões-se que o painel trabalhe em conjunto com atuadores que utilizem o mecanismo de rosca helicoidal como atuador, uma vez que tais sistemas permitem uma maior variedade com relação à quantidade de alimento, além de poder possuir um reservatório de grande volume, de forma a poder alimentar o(s) animal(is) por vários dias.

1.1 Motivação

A principal motivação deste trabalho é contribuir, com uma solução de baixo custo, para o conforto dos donos de animais domésticos, bem como para a saúde e bem estar de tais animais.

A partir da demanda de um amigo, o professor Alan Marotta e seu pai montaram um

dosador de ração para cães. Para comandar tal dispositivo, foi criado um programa de computador que proporcionasse ao usuário a seleção de quantidade de doses e dosagem de ração. Este programa foi apelidado de Trata Fácil, e sua interface é mostrada na Figura 1.1. Porém, para utilização deste *software*, era necessário manter um computador sempre conectado ao dispositivo. Esta condição era inviável, por motivos de custo, consumo de energia e até mesmo espaço. Com isto surgiu a ideia de transformar o Trata Fácil em um dispositivo micro controlado portátil, que fosse menor, oferecesse as mesma funções, e principalmente que fosse mais barato.

Figura 1.1 – Tela do "Trata fácil".

Avaliando os modelos disponíveis no mercado, foi observado que os dispositivos encontrados possuem preço elevado, como mostrado na Figura 1.2, que mostra alguns resultados mostrados na primeira página da pesquisa. O dispositivo mostrado na Figura 1.2 A e B possuem funções avançadas, como conexão à internet, porém apresentam baixa capacidade de armazenamento, além do alto custo. Já os mostrados na Figura 1.2 C e D são equipamentos mais simples, porém também apresentam preços elevados. A opção mostrada na Figura 1.2 C apresenta um temporizador pouco intuitivo, com muitas teclas, e

sem a possibilidade, por exemplo, de uma melhora como a conexão à internet. Isto motivou ainda mais a desenvolver um dispositivo de custo reduzido como opção aos existentes no mercado.

Figura 1.2 – Alguns produtos disponíveis no mercado.

Durante as pesquisas por dispositivos existentes no mercado,checou-se ainda a existência de dosadores à venda que seriam compatíveis o painel proposto neste trabalho, como mostrado na Figura 1.3.

Figura 1.3 – Alguns dosadores disponíveis no mercado.

1.2 Objetivos

1.2.1 Objetivo Geral

Projetar e construir um protótipo de painel de controle programável capaz de controlar dosagem e frequência de doses de alimento animal, utilizando micro controlador. Além disso, o presente trabalho visa também, implementar o painel a um dispensador de ração para cães já existente.

1.2.2 Objetivos Específicos

- Construir o circuito eletrônico;

- Construir o painel;
- Programar o microcontrolador;
- Construir o circuito eletrônico para acionamento do motor do protótipo de dispensador de ração para cães;
- Implementar o painel ao dispensador de ração para cães.

Capítulo 2

Fundamentos

2.1 Estado da Arte

Neste tópico serão discutidas algumas abordagens recentes que tratam de alimentadores automáticos para animais domésticos e outros tópicos tratados neste trabalho. Diversos trabalhos foram desenvolvidos nos últimos anos buscando facilitar a tarefa de alimentar os animais domésticos.

O trabalho de **Farhat2019** trás uma solução de um alimentador utilizando uma placa *Raspberry* PI. Nota-se que em tal desenvolvimento não há a preocupação com baixo custo, sendo que até mesmo a escolha da plataforma *Raspberry* apresenta um custo elevado, sendo esta indicada para aplicações muito mais complexas.

Buscando uma abordagem mais recente, alguns trabalhos como os de **Varandas2019**, **Silva2020** e **Rodrigues2019** utilizam conceitos de *Internet of Things* ou *Internet* das Coisas (IOT) , de maneira a permitir que os donos gerenciem a tarefa de alimentar seus animais de qualquer lugar, através de *smartphones* e conexão via *internet*.

Indo além da tarefa de alimentar os animais, o trabalho de **Wu2018** apresentou uma proposta de um dispositivo capaz de se locomover e dotado de uma câmera, permitindo que o dono encontre e observe seu animal, de maneira remota.

O trabalho de **Vineet2019** avalia alguns outros projetos recentes de alimentadores automáticos para animais domésticos e aborda a possibilidade da utilização de processamento de imagem neste contexto.

Dutra2020, **Buogo2017**, **Navarro2017** e **Oliveira2017** são algumas outras soluções recentes para o Problema de alimentar animais domésticos automaticamente. Estes

incluem a aplicação de comunicação remota, em que o usuário consegue programar e fiscalizar o sistema através de *smartphone*.

Diferente das produções citadas, o presente trabalho busca uma solução simples e de custo reduzido, que possa servir a diferentes espécies de animais.

Belvedere2017 mostra em detalhes o funcionamento da placa *Arduino*, servindo de base para a simulação do modo *standalone*. Neste modo o *Arduino* é substituído pelo microcontrolador ATMEGA328P, junto componentes básicos, de maneira a manter o mesmo funcionamento da placa, e garantindo a confiabilidade da simulação.

Belvedere2017, **Marotta2020**, **Arduino2015** e **Arduino2020** são alguns exemplos de livros atuais e que detalham o funcionamento e programação da placa *Arduino*. Eles mostram exemplos, funções e detalhes imprescindíveis para o desenvolvimento deste trabalho.

2.2 Fundamentação Teórica

2.2.1 Sistemas Computacionais

Um Sistema Computacional é um conjunto de circuitos eletrônicos (*hardwares*) interligados capaz de processar informações de acordo com um programa (*software*). É formado por: processadores, memórias, registradores, barramentos, monitores de vídeo, impressoras, mouses, discos magnéticos, além de outros dispositivos físicos. Em geral, são quatro os tipos de componentes de um sistema computacional: o processador, as memórias, dispositivos de entrada e saída (e periféricos) e o barramento que conecta todos os outros.

A Arquitetura de Von Neumann é uma arquitetura de sistema computacional caracterizada pela pela possibilidade da máquina digital de armazenar seus programas no mesmo espaço de memória que os dados. Tal arquitetura é um modelo de projeto de computador digital, que usa uma unidade de processamento, além de uma de armazenamento para armazenar instruções e dados. A Figura 2.1 ilustra a Arquitetura de Von Neumann, conforme apresentado por **Guarienti2019**.

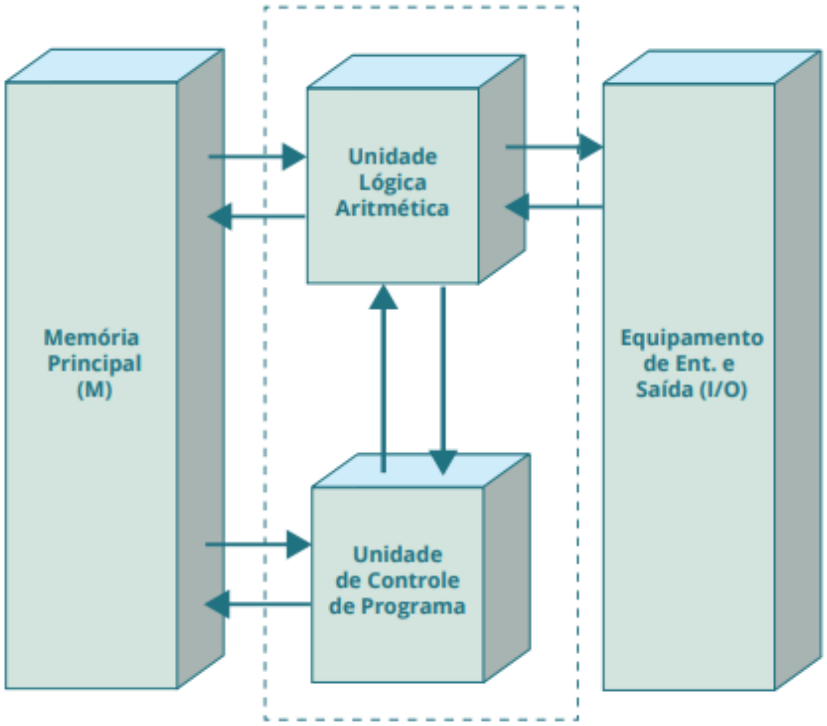

Figura 2.1 – Arquitetura de Von Neumann.

2.2.1.1 Processador

Também conhecida como CPU, A unidade central de processamento tem a função de controlar as operações realizadas por cada unidade funcional. É composto por: unidade de controle, que gerencia as atividades de todos componentes do sistema, unidade lógica e aritmética, que realiza operações lógicas e aritméticas, registradores que armazenam dados temporariamente e o barramento, que interliga os componentes do processador.

2.2.1.2 Memórias

A memória principal ou memória primária, é aonde são armazenados instruções e dados. É composta por unidades de acesso chamadas células. Geralmente cada célula possui tamanho de 1 *byte* e seu acesso se dá por meio da especificação de um número, o seu endereço. A memória principal pode ser de vários tipos, podendo ser volátil, não volátil ou programável somente uma vez.

A memória de dados é onde são armazenadas as variáveis do programa e outras informações, ou seja, os dados a serem processados. Elas são voláteis e operam a grande velocidade. Alguns microcontroladores disponibilizam também memórias EEPROM para

armazenar dados que não devem ser perdidos com o desligamento do sistema.

As memórias secundárias são lentas, não voláteis e de maior capacidade de armazenamento. São utilizadas para armazenar grandes quantidades de dados e programas de maneira permanente.

2.2.1.3 Dispositivos de Entrada e Saída

São responsáveis por permitir a comunicação entre o sistema computacional e o mundo externo, seja esta comunicação homem-máquina ou entre o sistema e o ambiente ou outros equipamentos. Alguns exemplos de dispositivo de entrada são teclados, mouses, câmeras *web-cam*, leitores de códigos de barra e sensores em geral. Já alguns exemplos de dispositivos de saída são caixas de som, impressoras, projetores, monitores, motores ou outros atuadores conectados a um sistema. Há ainda os dispositivos que praticam ambas funções, de entrada e saída, como por exemplo discos compactos, *pen-drives*, roteadores e telas *touch-screen*.

2.2.1.4 Barramento

O barramento é o meio físico que conecta as outras partes de um sistema computacional. Ele é um conjunto de linhas de comunicação que possibilitam a interligação entre dispositivos, incluindo a CPU, a memória, etc. Ele é composto por 3 tipos de linhas de comunicação: de dados, de endereços e de controle. A linha de dados é responsável pelo transporte de dados, e é do tipo bidirecional. Já a linha de endereços possui a função de indicar os endereços de memória de dados que o processador deverá retirar ou enviar. Esta linha é do tipo unidirecional. Por fim,a linha de controle é aquela que controla as ações dos barramentos anteriores, ou seja, ela controla solicitações e confirmações. Esta linha é do tipo bidirecional.

2.2.2 Microcontroladores

Um microcontrolador é um circuito integrado que possui em seu interior todos os componentes necessários para seu pleno funcionamento, apenas dependendo de uma fonte externa de alimentação. Em outras palavras um microcontrolador é um computador em um único chip. Este chip contem um processador, memória, periféricos de entrada e saída,

dentre outros. Esta integração é uma das principais vantagens destes dispositivos, pois faz com que sua utilização seja mais fácil e mais barata.

Standalone é um sistema que é autossuficiente, que não necessita de um semelhante para executar sua função. Um exemplo de sistema *standalone* pode ser composto por um microcontrolador, um cristal de *clock*, uma fonte de alimentação e os dispositivos de entrada e saída desejados para a aplicação. A vantagem deste tipo de sistema é seu custo reduzido e simplicidade, se comparado com sistemas mais complexos e caros como o *Arduino*. Neste trabalho o modo *standalone* é utilizado para simulação, uma vez que a placa *Arduino* não está disponível nas bibliotecas do *software*.

2.2.3 ATMEGA328

ATMEGA328 é um modelo de microcontrolador criado pelo fabricante Atmel. É um chip do tipo AVR de 8 bits, baseado em *Reduced Instruction Set Computer* ou Computador com um Conjunto Reduzido de Instruções (RISC). Possui uma memória flash de 32kB, 1kB de memória *Electrically Erasable Programmable Read-Only Memory* ou Memória de Leitura Apenas Programável Apagável Eletricamente (EEPROM), 2kB de *Static Random-access Memory* ou memória de acesso aleatório estática (SRAM), 23 linhas de entrada/saída de propósito geral e 32 registradores para diversos propósitos. Também possui serial programável *Universal Synchronous Asynchronous Receiver Transmitter* ou Transmissor/Receptor Universal Síncrono e Assíncrono (USART), conversor analógico/digital de 6 canais e 10 bits, temporizador do *watchdog* interno programável com oscilador e 5 modos de economia de energia selecionáveis por *software*. Ele opera entre 1,8 e 5,5v e alcança uma taxa de transferência que se aproxima de 1 *Millions of Instructions Per Second* ou Milhões de Instruções por Segundo (MIPS) por Mega *Hetz* (MHz), com uma frequência máxima de 20 MHz.

2.2.4 *Arduino UNO*

Dotada de um microcontrolador ATMEGA328, a *Arduino UNO* é uma placa de uso altamente difundido para projetos como prototipação e no meio educacional. Ela é mostrada na Figura 2.2.

O *Arduino UNO* possui 14 pinos que podem ser configurados como entrada ou saída digitais, sendo que 6 destes podem ser usados como *Pulse Width Modulation* ou Modulação

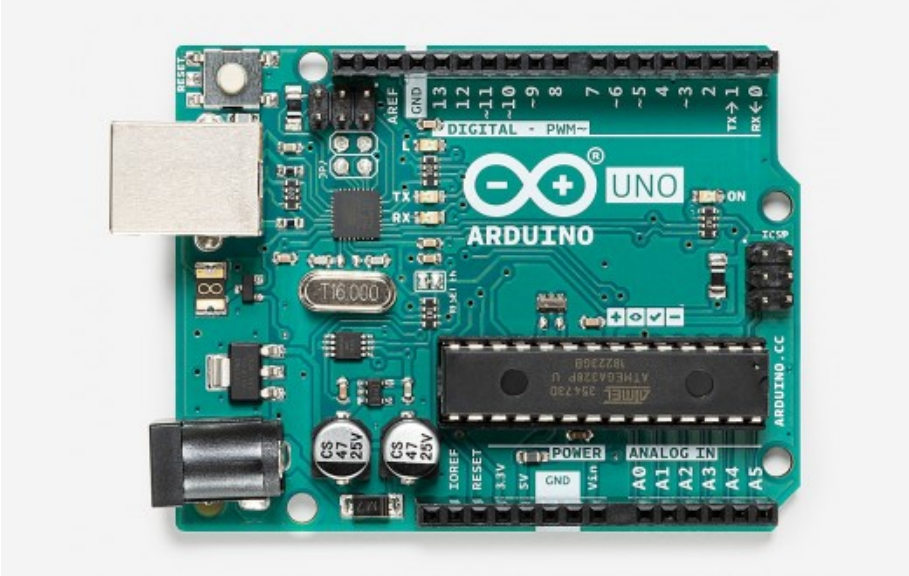

Figura 2.2 – *Arduino UNO* fonte: https://store.arduino.cc/usa/arduino-uno-rev3

por Largura de Pulso (PWM), outros 6 pinos de entrada analógica, um cristal oscilador de 16MHz, conector *Universal Serial Bus* ou Porta Serial Universal (USB) e conector para alimentação.

Para programação do *Arduino UNO*, existe a *Arduino Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado (IDE), um *software* de programação próprio, que utiliza linguagem C, com adição de alguns comandos e características próprias.

2.2.5 *Pull-up* e *pull-down*

Para a utilização de teclas com maior confiabilidade, podem ser utilizadas as técnicas de *pull-up* e *pull-down*, ou resistores de *pull-up* e *pull-down*, que têm a função de evitar que a entrada fique em estado flutuante, o que pode ocasionar uma falsa ativação. A Figura 2.3 ilustra estas ligações e a Figura 2.4 exemplifica como elas são realizadas em conexão com um *Arduino UNO*, conforme **Marotta2020**.

A técnica se baseia em conectar um resistor entre a alimentação positiva e a entrada, no caso de *pull-up*, ou entre a entrada e a alimentação negativa ou terra, no caso *pull-down*.

2.2.6 *Debounce*

Quando uma tecla é pressionada em um sistema digital, um sinal ruidoso é gerado, podendo acarretar o resultado de vários pressionamentos rápidos ou outras decorrências.

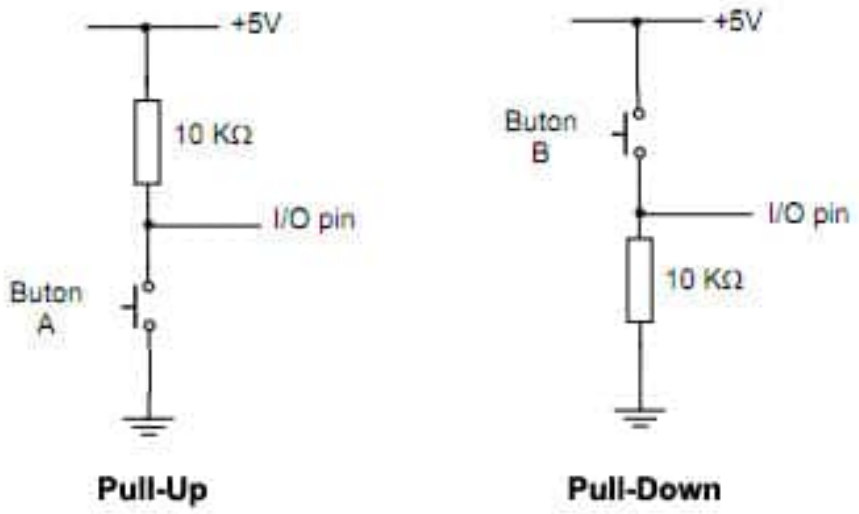

Figura 2.3 – Ligações de teclas com resistor de *pull-up* (esquerda) e *pull-down* (direita)

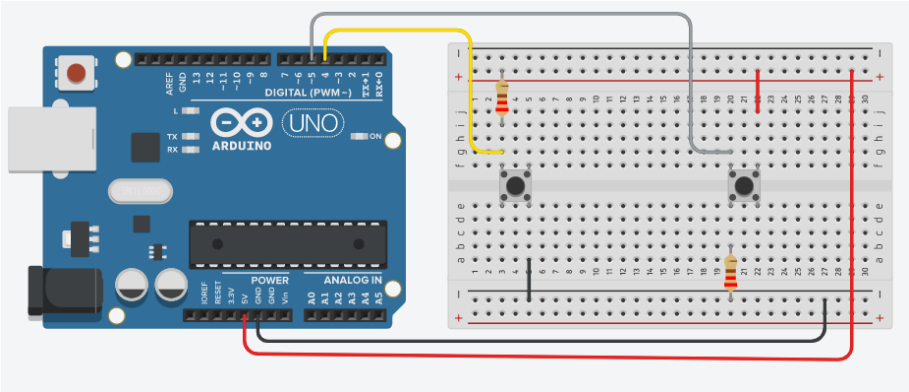

Figura 2.4 – Conexão arduino UNO com teclas, ligações *pull-up* (esquerda) e *pull-down* (direita)

Este efeito, conhecido como efeito *bounce*, é demostrado na Figura 2.5, em que o sinal ideal é mostrado à esquerda, enquanto o real é mostrado à direita, evidenciando a presença da oscilação do sinal. Para evitar falsos acionamentos, existem técnicas conhecidas como *debounce*. Exemplos de implementações de técnicas *debounce* estão disponíveis em **Marotta2020** e também nos exemplos do *software Arduino*.

Uma técnica *debounce* se baseia em desconsiderar variações que tenham duração inferior a um certo período, variando de acordo com a aplicação. Nesta técnica, quando uma variação no sinal advindo ta tecla é detectada, é iniciado um temporizador. A cada nova variação, este tempo é reiniciado, sendo que a ação aferida à tecla somente é executada quando se chega ao final do tempo determinado, sem que haja mais variações. Desta forma, garante-se que os pulsos de curta duração característicos do efeito *bounce*

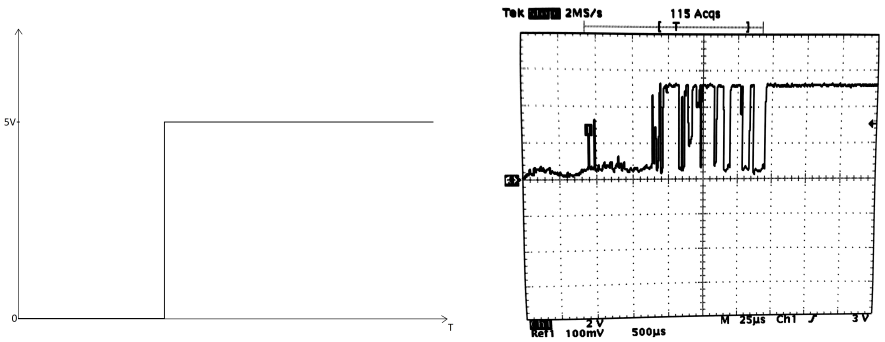

Figura 2.5 – Efeito *Bounce*

sejam ignorado e a ação somente seja executada uma vez, quando passado o tempo transiente, em que ocorrem as variações. Esta técnica pode ser encontrada em *Arduino*.cc ou diretamente na IDE *Arduino*, podendo encontrá-la justo aos exemplos oferecidos pelo *software*.

Capítulo 3

Desenvolvimento

3.1 Materiais e Métodos

O painel conta com 4 interfaces de acionamento através de botoeiras compostas por teclas de membranas. Duas destas permitem ao usuário aumentar ou diminuir a dosagem, outra serve para definir a quantidade de doses diárias, e a última permite se testar o tamanho da dose. Para se testar a dosagem (tamanho da dose) ao pressionar a tecla, uma dose de ração é liberada, de maneira que o usuário possa observar se aquela é a quantidade desejada. Para indicar a programação selecionada para quantidade de doses diárias, foram utilizados 4 leds, indicadores de 1 a 4 doses. Também foram utilizados resistores de diferentes valores de resistência para complementar o circuito e garantir seu correto e seguro funcionamento.

Como componente central do painel é utilizada uma placa *Arduino UNO* que conta com um microcontrolador ATMEGA328P. A programação deste foi feita em software próprio para *Arduino* e que recebe o mesmo nome (IDE *Arduino* versão 1.8.13 ou mais recente). A linguagem de programação utilizada é C, com características próprias da IDE *Arduino*.

O protótipo dispensador de ração é composto por um motor de máquina de costura e eixo de moedor de carne reaproveitados, além de partes estruturais de metal. O modelo escolhido utiliza um mecanismo de rosca sem fim. O motor utilizado é alimentado com 127 V de tensão alternada, não sendo necessária a utilização de uma fonte extra ou qualquer forma de adaptação.

Para que o sistema funcione nos horários desejados, ele conta com um CI relógio. O

modelo escolhido é o DS1307, por ser um dispositivo barato, facilmente encontrado no mercado de maneira integrada com um módulo. Este módulo possui todos os componentes que possibilitam seu funcionamento, incluindo cristal, suporte para bateria e um CI de memória EEPROM. Este módulo é mostrado na Figura 3.1, à direita na imagem há uma moeda para comparação, para se observar o tamanho reduzido do módulo. O circuito DS1307 é capaz de calcular informação de horas, minutos, segundos, dia, mês, ano e dia da semana. Ele ainda considera dia extra em anos bissextos e quantidade correta de dias de cada mês. As informações podem ser gravadas e alteradas via *software*. Sem a utilização deste dispositivo, o equipamento apresentaria a necessidade de ser ligado no momento desejado para a primeira dose do dia, além de que a cronometragem seria perdida em caso de falha de energia ou desligamento por exemplo. Com a utilização do CI, é possível alimentá-lo com uma pequena bateria para que as informações não sejam perdidas mesmo na ausência de energia. Esta bateria é capaz de alimentar o circuito por vários anos.

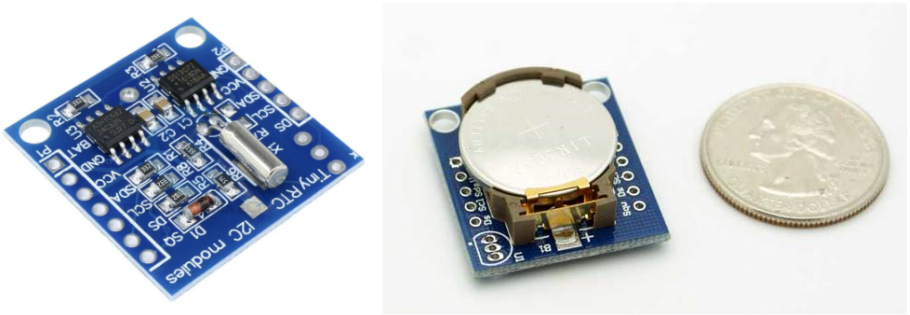

Figura 3.1 – CI relógio DS1307

Para o circuito que aciona o motor do dispensador, deverá ser utilizado um relé, um transistor e resistores.

Para as simulações de circuitos eletrônicos foi utilizado o *software* Proteus versão 8 *Professional*. Já para o desenho da estrutura do painel, foi utilizado o *software Solidworks*.

3.2 Simulação

Utilizando o *software* Proteus, o circuito eletrônico foi simulado. Utilizou-se o modelo do microcontrolador ATMEGA328P existente nas bibliotecas do *software* para representar o *Arduino* como um todo. Para que o mesmo funcionasse foi adicionado um cristal oscilador de 16 MHz, com 2 capacitores de 22 pF entre seus terminais e o terra, carac-

terizando a configuração *standalone*. Para programar o microcontrolador, foi inserido o arquivo de extensão .hex gerado pela IDE *Arduino* ao se compilar o programa escrito.

Foram adicionados 4 LEDs conectados aos pinos PD2, PD3, PD4 e PD5 do microcontrolador, correspondentes aos pinos digitais 2 a 5 do *Arduino* e ao terra. em série com cada LED foi conectado um resisto de valor 220 Ohms de resistência. Cada LED corresponde a uma quantidade de doses diárias, de 1 a 4.

Também, foram adicionados 4 teclas, conectadas à alimentação de 5 V e aos pinos PD6, PB0, PB1 e PB2 do microcontrolador, correspondentes aos pinos digitais 6, 8, 9 e 10 do *Arduino*. Além disso, foram conectados resistores de 10 KOhms entre cada um desses pinos e a referência (terra ou 0V), configurando-os como resistores de *pull-down*.

Além disso foi adicionado o CI relógio DS1307. Para seu correto funcionamento, foi preciso adicionar um cristal entre seus terminais X1 e X2. Conforme o *datasheet* do CI, o cristal foi configurado para 32768 KHz. O conector SCL foi conectado à entrada analógica 4 do *Arduino*, enquanto o conector SDA foi conectado à entrada analógica 5 do *Arduino*, isto conforme indicado na documentação da biblioteca utilizada para este circuito. Em ambos terminais foram adicionados resistores de *pull-up*, para garantir seu correto funcionamento, também conforme indicado no *datasheet*.

Para representar, de maneira visual, o motor, foi utilizado um led, acionado por um transistor NPN. Assim como os demais LEDs, foi utilizado um resistor de 220 Ohms em série para limitar a corrente passando pelo LED. Conectado à base do transistor foi adicionado um resistor no valor de 10 KOhms.

3.3 Programação

Fazendo uso da IDE *Arduino*, foi criado o código empregando a linguagem *Arduino*, conforme a documentação presente no site Arduino.cc, também informações de outras fontes, como **Arduino2015**, **Arduino2020** e **Marotta2020**. Para gerar o arquivo .hex utilizado na simulação no Proteus, foi preciso compilar o código e copiar o diretório temporário criado e mostrado no *log* de compilação. O código gerado pode ser visto no link apresentado no Apêndice A.

Para comunicação com o CI relógio, é utilizada a biblioteca DS1307new.h. Esta permite a utilização de funções simples e diretas que permitem a leitura de horas, mi-

nutos e segundos, informações utilizadas para determinar os horários de alimentação. Com esta biblioteca, para se obter informações de horário atuais, é utilizado o comando RTC.getTime(). Em seguida tais informações podem ser utilizadas através dos comandos RTC.hour, RTC.minute e RTC.second. Assim, basta comparar os valores retornados por estes comandos com o horário desejado, iniciando a rotina de liberação de ração caso positivo. Como exemplo, para uma dose diária selecionada, está programada a alimentação às 8 horas. Assim, é feita a comparação das horas (RTC.hour) com o valor oito, em conjunto com os minutos e segundos (TRC.minute e RTC.second) com o valor zero. Para funcionamento destas funções há a necessidade da adição também da biblioteca wire.h.

Para que as informações de quantidade de doses e tamanho das doses não sejam perdidas, é preciso armazená-las. Para isto, é utilizada a memória EEPROM presente na placa Arduino. De forma a facilitar a utilização deste componente, foi utilizada a biblioteca EEPROM.h. Com esta biblioteca podem ser utilizados os comandos EEPRO.read(addr) para leitura de um dado armazenado e EEPROM.write(addr,val) para se escrever na memória. Como parâmetros para estes comandos, temos o endereço desejado para leitura ou escrita (addr) de 0 a 1023 e o valor que se deseja escrever (val), de tamanho 1 byte (máximo 255).

Durante a simulação no Proteus, o sistema apresentou uma lentidão acentuada quando adicionados os comandos referentes ao CI relógio. Para reduzir esta morosidade, foi criada uma base de tempo da ordem de 100 milissegundos, válida para tais comandos, utilizando o comando millis() (contador de tempo do *Arduino*). Dentro de tal base os registradores de horário são atualizados (comando RTC.getTime()) e são feitas as comparações. Com isto foi possível reduzir o custo computacional do ciclo médio, diminuindo os atrasos durante as simulações.

Capítulo 4

Resultados e Discussões

Seguindo as etapas descritas no capítulo de desenvolvimento, chegou-se ao circuito mostrado na Figura 4.1.

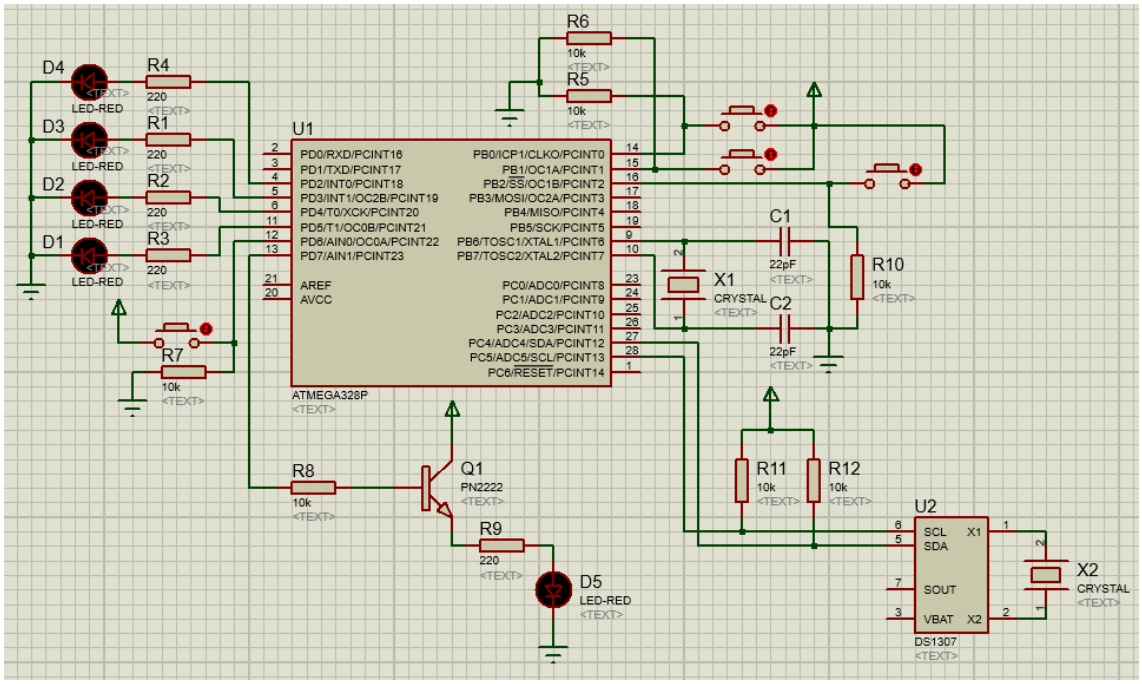

Figura 4.1 – Circuito completo da simulação realizada

Conforme o código presente no link apresentado no apêndice A, pressionando-se a tecla conectada à entrada digital 6 altera-se a quantidade de doses diárias, alterando-se assim os estados dos LEDs indicadores como mostrado na Figura 4.2. Nesta figura, é possível ver que a quantidade de duas doses diárias foi selecionada (segundo LED de cima para baixo está aceso). Além disto, no momento em que esta tecla é pressionada, o novo valor de quantidade de doses diárias é armazenado na EEPROM, para evitar a perda de tal

informação na ausência de alimentação para o sistema. O valor armazenado na EEPROM é sempre carregado de volta para a devida variável quando o sistema é iniciado.

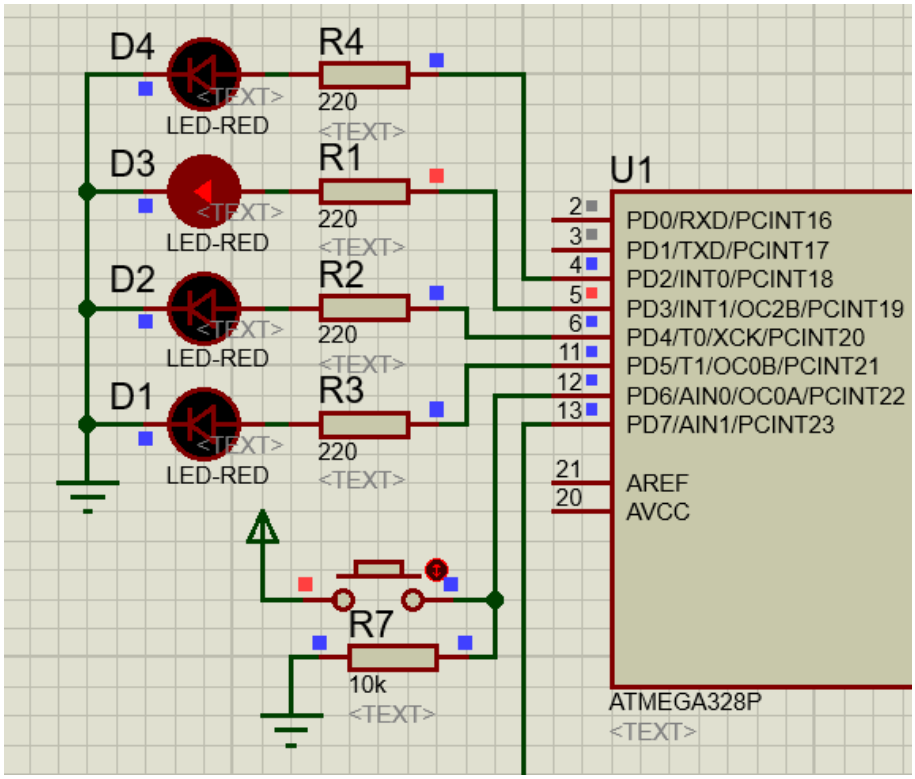

Figura 4.2 – Leds indicadores e tecla de seleção de quantidade de doses diárias

Na Figura 4.3, as duas teclas mostradas uma acima da outra, destacadas em vermelho, têm a função de aumentar ou diminuir a quantidade de ração por dose. Além disto, no momento em que estas teclas são pressionadas, o novo valor de dosagem é armazenado na EEPROM, para evitar a perda de tal informação na ausência de alimentação para o sistema. O valor armazenado na EEPROM é sempre carregado de volta para a devida variável quando o sistema é iniciado. Já a tecla mais à direita, destacada em azul, é responsável por liberar uma dose de ração instantaneamente, com o intuito de demonstrar a dosagem selecionada para o operador.

Seguindo o padrão de programas *Arduino*, no código gerado, primeiramente são adicionadas as bibliotecas necessárias, em seguida são declaradas as variáveis globais. Então, é executado o ciclo de inicialização, chamado *setup*. Dentro deste, são definidos os pinos de entrada e saída, em seguida são carregados os dados armazenados na memória EEPROM, e por fim é chamada a rotina de acendimento dos leds. Ao fim da inicialização, o *looping* é iniciado. Neste, as linhas de código inseridas são executadas na mesma ordem que forem inseridas. Ao chegar à última linha deste ciclo, o microcontrolador volta a executá-las a

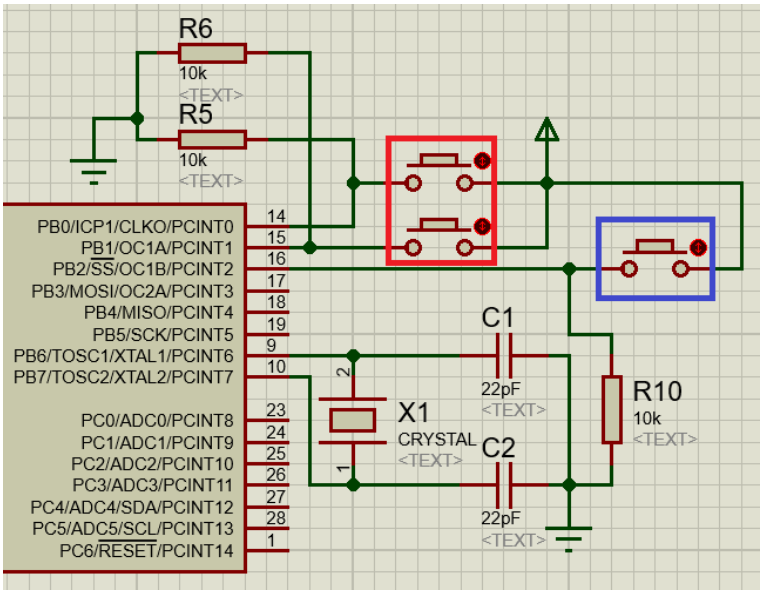

Figura 4.3 – teclas de regulagem da dose

partir da primeira, e assim segue até que seja desligado ou reiniciado. Dentro do ciclo do *looping*, primeiramente é feita a atualização dos registradores do programa com o horário atual do CI relógio, isto dentro da base de tempo de 100ms. Ainda dentro desta base de tempo, é invocada a rotina de comparação, que compara o horário atual com os pré-definidos para liberação de ração. Caso positivo a rotina de liberação é chamada. Dentro desta rotina, são dados pulsos, de duração definida, de ativação da saída referente ao acionamento do motor. Em seguida, de volta ao *looping*, é feita a leitura das 4 teclas. Então, é executada a rotina *debounce* para cada uma destas leituras. Para o caso da tecla número 1, caso o *debounce* retorne positivo, é incrementada a variável de quantidade de doses diárias, caso esta esteja em seu valor máximo, seu valor é reiniciado a 1. Após, é feita a atualização do acendimento dos leds, e por fim é atualizado o valor armazenado na EEPROM. Para as teclas 2 e 3, é respectivamente é incrementado e decrementado o valor da variável referente ao tamanho da dose, sempre limitando este aos valores máximo ou mínimo. Em seguida o valor atualizado é armazenado na EEPROM. Por fim a quarta tecla evoca a rotina de liberação de ração.

A partir do circuito simulado, mostrado na Figura 4.1, foi possível fazer o desenho da placa, utilizando o programa ARES, parte do *software* Proteus, como mostrado na Figura 4.4. Nela é possível ver o circuito e a disposição dos componentes e trilhas, em azul. À direita, é possível ver uma visualização 3D da placa, aonde se vê os leds, resistores e as teclas, representados por blocos quadrados vermelhos. Esta placa foi dimensionada de

maneira a encaixar acima da placa *Arduino*, ou seja, é uma placa do tipo popularmente chamado *shield*.

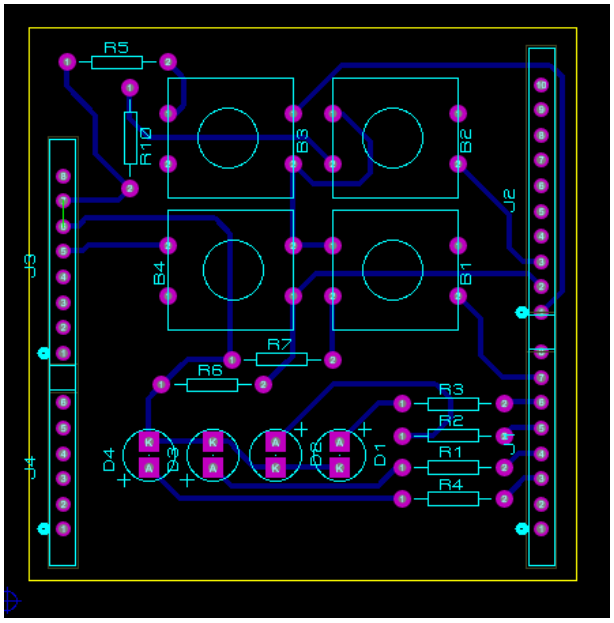

Figura 4.4 – Placa desenhada no Proteus ARES.

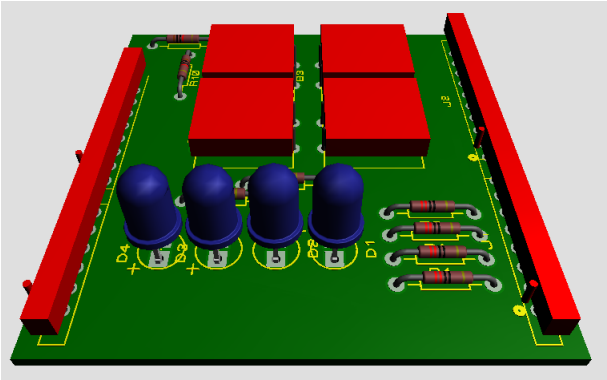

Figura 4.5 – Visualização 3D da placa desenhada no Proteus ARES.

Com as dimensões da placa desenhada, da placa *Arduino* e das botoeiras e leds, foi dimensionada e desenhada a estrutura do painel, utilizando o *software SolidWorks*. A Figura 4.6 ilustra o painel montado, sendo possível ver a estrutura, as teclas e os conectores do *Arduino*. O desenho técnico da estrutura pode ser visto no Apêndice B.

Figura 4.6 – Painel montado, desenho no *software SolidWorks*.

Capítulo 5

Considerações Finais

Foi proposto projetar e construir um painel de controle programável capaz de controlar dosagem e frequência de doses de alimento animal, utilizando microcontrolador. Constata-se que o objetivo geral foi parcialmente atendido, na medida em que foi projetado e simulado todo o sistema, mas não foi construído. O circuito eletrônico foi simulado e a placa referente a ele foi desenhada, porém esta não foi construída. O painel foi dimensionado e desenhado, de acordo com a placa eletrônica especificada e o *Arduino*, porém o painel não foi construído. A programação do microcontrolador foi realizada e foi constatada sua eficácia através da simulação. Quanto à implementação do painel ao dispensador de ração para cães nada foi feito, uma vez que esta seria relevante apenas após a construção do painel, sendo que para a simulação o sistema controlado foi representado por um led.

5.1 Proposta de Continuidade

A primeira proposta de continuidade é de construir o painel e finalizar os objetivos traçados neste trabalho. Pode-se também construir um dispensador diferente, por exemplo um pequeno dispensador para alimento de peixes, a fim de demonstrar a utilidade do painel em diferentes atuadores. Além disso é possível incrementar o painel com um módulo de conexão à internet, como o mostrado na Figura 5.1, um módulo de conexão wifi de baixo custo, modelo ESP8266. Tal conexão abre um leque de possibilidades, como a programação via *smartphone*, a possibilidade de avaliação da alimentação, da disponibilidade de ração no reservatório (basta adicionar um sensor), dentre outras.

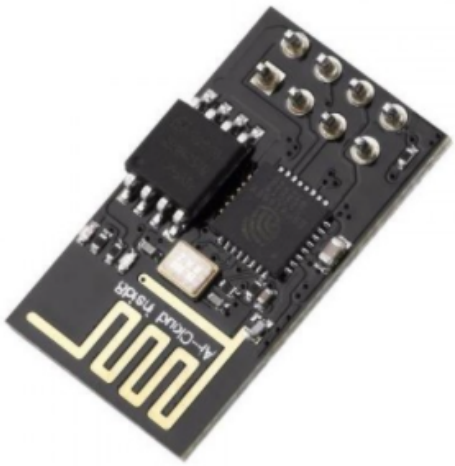

Figura 5.1 – CI wifi esp8266.

Apêndice A

Código Arduino

O código pode ser encontrado no link:
https://github.com/LAGontijo/TCC

Apêndice B

Desenhos Técnicos

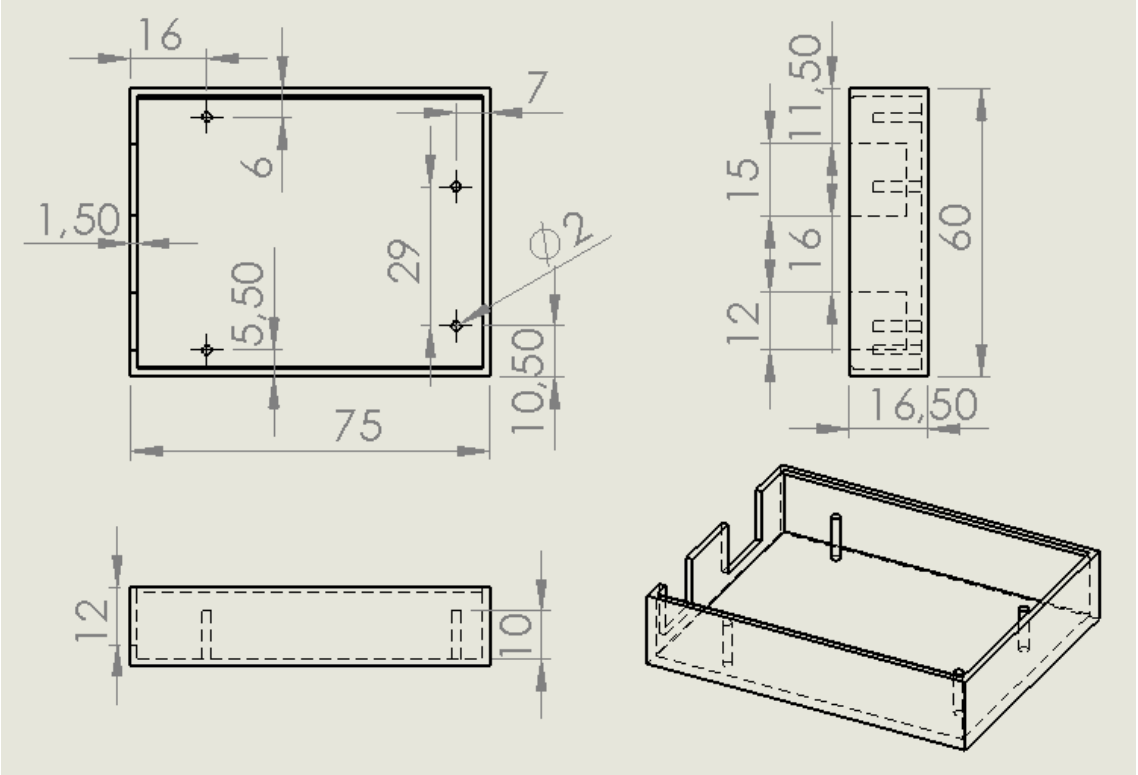

Figura B.1 – Base do painel

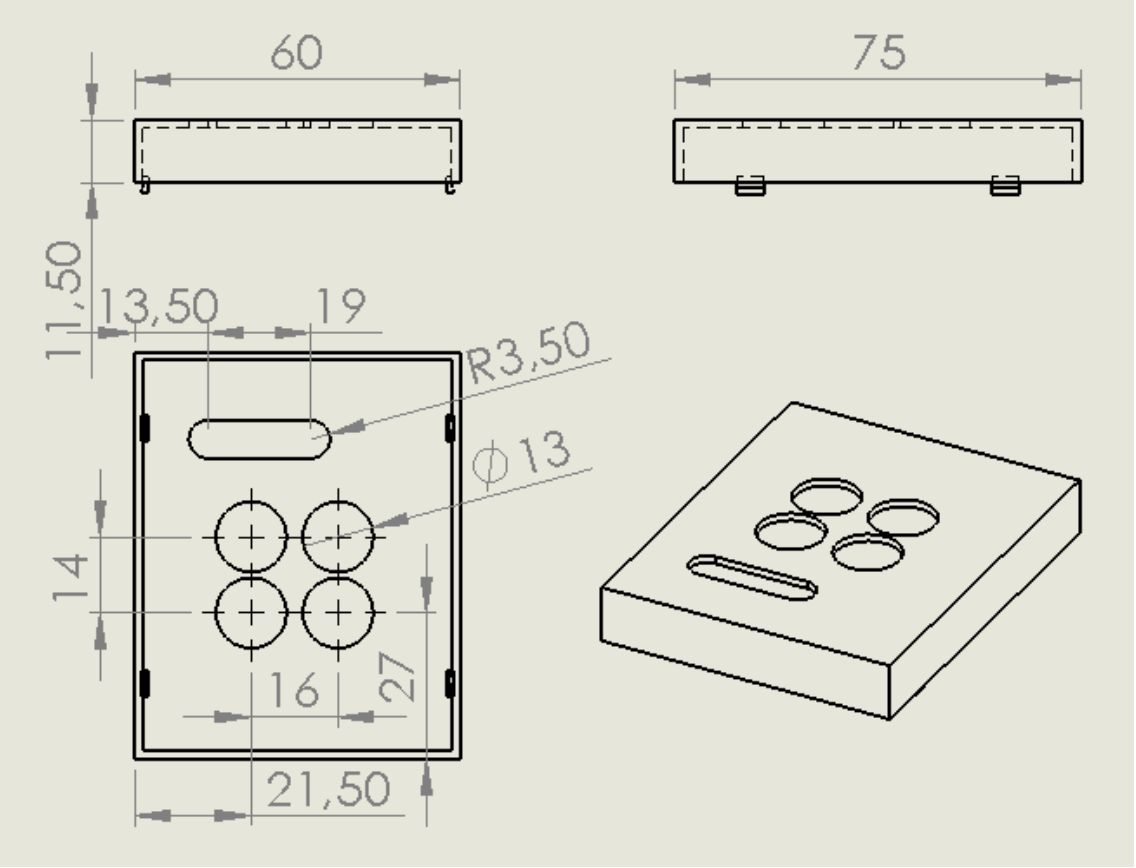

Figura B.2 – Tampa do painel